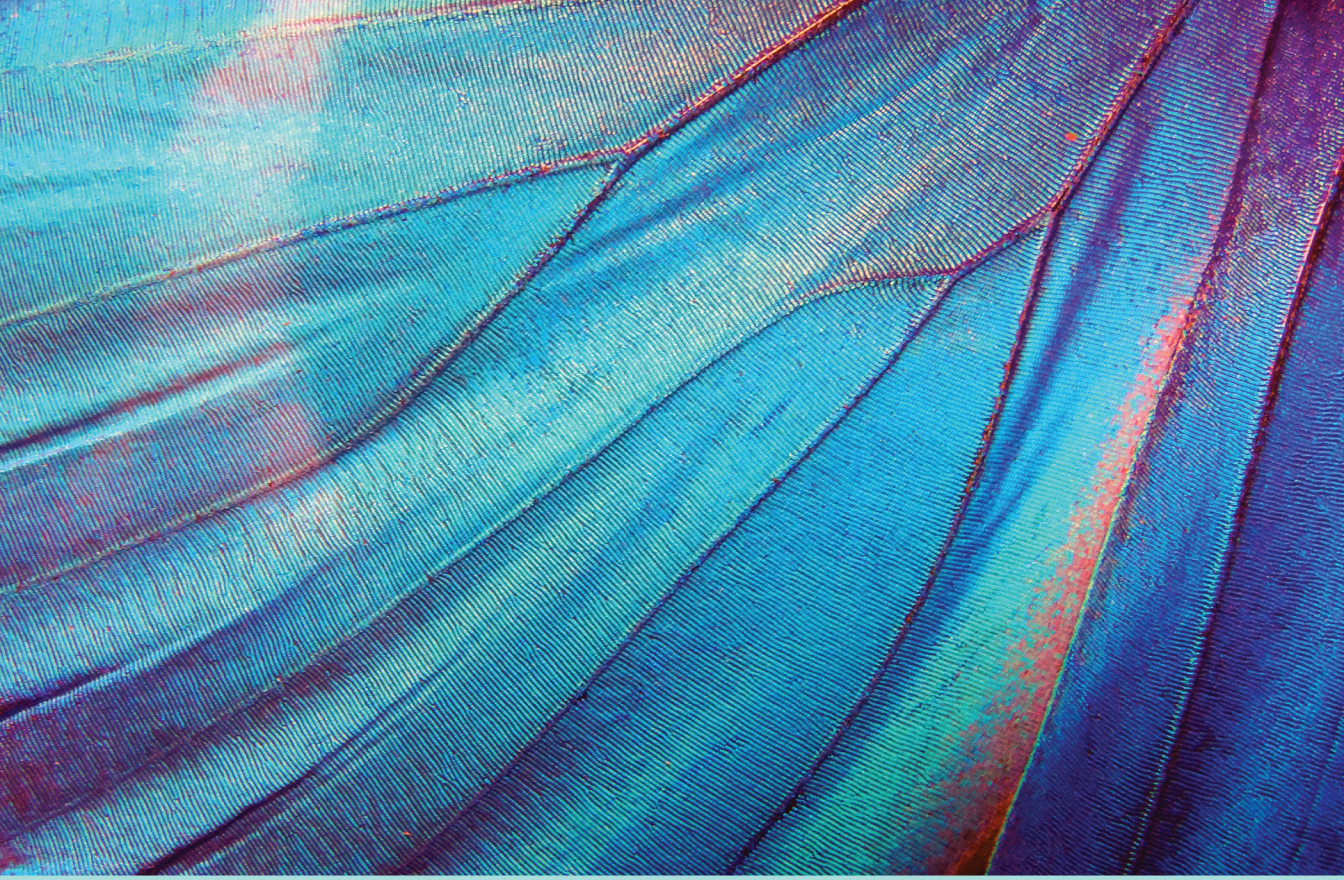


Theory, Practice and Techniques in

WEB DEVELOPMENT

Éric Fournier

2ND EDITION

THEORY PRACTICE TECHNIQUES

BIBLIOTEX
Digital Library

Theory, Practice and Techniques in Web Development

2ND EDITION

Éric Fournier

BIBLIOTEX
Digital Library

www.bibliotex.com

THEORY, PRACTICE AND TECHNIQUES IN WEB DEVELOPMENT

2ND EDITION

Éric Fournier



www.bibliotex.com

email: info@bibliotex.com

e-book Edition 2024

ISBN: 978-1-98469-093-7 (e-book)

This book contains information obtained from highly regarded resources. Reprinted material sources are indicated. Copyright for individual articles remains with the authors as indicated and published under Creative Commons License. A Wide variety of references are listed. Reasonable efforts have been made to publish reliable data and views articulated in the chapters are those of the individual contributors, and not necessarily those of the editors or publishers. Editors or publishers are not responsible for the accuracy of the information in the published chapters or consequences of their use. The publisher assumes no responsibility for any damage or grievance to the persons or property arising out of the use of any materials, instructions, methods or thoughts in the book. The editors and the publisher have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission has not been obtained. If any copyright holder has not been acknowledged, please write to us so we may rectify.

Notice: Registered trademark of products or corporate names are used only for explanation and identification without intent of infringement.

© 2024 3G E-learning LLC

In Collaboration with 3G E-Learning LLC. Originally Published in printed book format by 3G E-Learning LLC with ISBN 978-1-98468-876-7

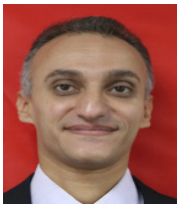
EDITORIAL BOARD



Aleksandar Mratinković is born on May 5, 1988. in Arandjelovac, Serbia. He has graduated on Economic high school (2007), The College of Tourism in Belgrade (2013), and also has a master degree of Psychology (Faculty of Philosophy, University of Novi Sad). He has been engaged in different fields of psychology (Developmental Psychology, Clinical Psychology, Educational Psychology and Industrial Psychology) and has published several scientific works.



Dan Piestun (PhD) is currently a startup entrepreneur in Israel working on the interface of Agriculture and Biomedical Sciences and was formerly president-CEO of the National Institute of Agricultural Research (INIA) in Uruguay. Dan is a widely published scientist who has received many honours during his career including being a two-time recipient of the Amit Golda Meir Prize from the Hebrew University of Jerusalem, his areas of expertise includes stem cell molecular biology, plant and animal genetics and bioinformatics. Dan's passion for applied science and technological solutions did not stop him from pursuing a deep connection to the farmer, his family and nature. Among some of his interest and practices counts enjoying working as a beekeeper and onboard fishing.



Hazem Shawky Fouda has a PhD. In Agriculture Sciences, obtained his PhD. From the Faculty of Agriculture, Alexandria University in 2008, He is working in Cotton Arbitration & Testing General Organization (CATGO).



Felecia Killings is the Founder and CEO of LiyahAmore Publishing, a publishing company committed to providing technical and educational services and products to Christian Authors. She operates as the Senior Editor and Writer, the Senior Writing Coach, the Content Marketing Specialist, Editor-in-Chief to the company's quarterly magazine, the Executive and Host of an international virtual network, and the Executive Director of the company's online school for Authors. She is a former high-school English instructor and professional development professor. She possesses a Master of Arts degree in Education and a Bachelor's degree in English and African American studies.



Dr. Sandra El Hajj, Ph.D. in Health Sciences from Nova Southeastern University, Florida, USA is a health professional specialized in Preventive and Global Health. With her 12 years of education obtained from one of the most prominent universities in Beirut, in addition to two leading universities in the State of Florida (USA), Dr. Sandra made sure to incorporate interdisciplinary and multicultural approaches in her work. Her long years of studies helped her create her own miniature world of knowledge linking together the healthcare field with Medical Research, Statistics, Food Technology, Environmental & Occupational Health, Preventive Health and most noteworthy her precious last degree of Global Health. Till today, she is the first and only doctor specialized in Global Health in the Middle East area.



Fozia Parveen has a Dphil in Sustainable Water Engineering from the University of Oxford. Prior to this she has received MS in Environmental Sciences from National University of Science and Technology (NUST), Islamabad Pakistan and BS in Environmental Sciences from Fatima Jinnah Women University (FJWU), Rawalpindi.



Igor Krunic 2003-2007 in the School of Economics. After graduating in 2007, he went on to study at The College of Tourism, at the University of Belgrade where he got his bachelor degree in 2010. He was active as a third-year student representative in the student parliament. Then he went on the Faculty of science, at the University of Novi Sad where he successfully defended his master's thesis in 2013. The crown of his study was the work titled Opportunities for development of cultural tourism in Cacak". Later on, he became part of a multinational company where he got promoted to a deputy director of logistic. Nowadays he is a consultant and writer of academic subjects in the field of tourism.



Dr. Jovan Pehcevski obtained his PhD in Computer Science from RMIT University in Melbourne, Australia in 2007. His research interests include big data, business intelligence and predictive analytics, data and information science, information retrieval, XML, web services and service-oriented architectures, and relational and NoSQL database systems. He has published over 30 journal and conference papers and he also serves as a journal and conference reviewer. He is currently working as a Dean and Associate Professor at European University in Skopje, Macedonia.



Dr. Tanjina Nur finished her PhD in Civil and Environmental Engineering in 2014 from University of Technology Sydney (UTS). Now she is working as Post-Doctoral Researcher in the Centre for Technology in Water and Wastewater (CTWW) and published about eight International journal papers with 80 citations. Her research interest is wastewater treatment technology using adsorption process.



Stephen obtained his PhD from the University of North Carolina at Charlotte in 2013 where his graduate research focused on cancer immunology and the tumor microenvironment. He received postdoctoral training in regenerative and translational medicine, specifically gastrointestinal tissue engineering, at the Wake Forest Institute of Regenerative Medicine. Currently, Stephen is an instructor for anatomy and physiology and biology at Forsyth Technical Community College.



Michelle holds a Masters of Business Administration from the University of Phoenix, with a concentration in Human Resources Management. She is a professional author and has had numerous articles published in the Henry County Times and has written and revised several employee handbooks for various YMCA organizations throughout the United States.

TABLE OF CONTENTS

Preface ..... *xi*

Chapter 1 Creating Web Pages with HTML **1**

1.1 Fundamental of HTML 1
Basics 2
HTML Tags 4
1.2 Getting Started 10
Overview 10
Creating an HTML Document 11
Naming Files and Folders 12
Create a Simple Web Page with HTML 14
HTML Attribute 18
1.3 HTML Structure 24
Perception 24
DOCTYPE 25
HTML Element 26
Summary 35
Knowledge Check 36
Review Questions 37
References 38

Chapter 2 CSS **39**

2.1 Basics of CSS 39
Definition 40
The "Cascade" Part of CSS 41
Benefits of CSS 43
CSS Box Model 44
2.2 CSS Syntax 46
Definition 46
CSS Selectors 47
Grouping Selectors 49
2.3 Types of CSS Styles 51
Definition 51
Inline Styles 52
Embedded Styles 53

External Style Sheets 55
Summary 59
Knowledge Check 59
Review Questions 61
References 61

Chapter 3 JavaScript 63

3.1 Basics of JavaScript 63
Definition 64
Client-side JavaScript 66
JavaScript Development Tools 67
3.2 DOM Manipulation and Events 69
Definition 69
DOM and JavaScript 70
The Event Object 77
Event Phases 77
3.3 Conditionals and Variable 86
Fundamental 86
Conditionals 88
Variables 94
3.4 Pair Programming 97
Concept 97
Origins 98
Pairing Mechanics 99
3.5 Debugging 104
Definition 104
The Debugging Process 105
Common Debugging Tools 106
Summary 110
Knowledge Check 110
Review Questions 112
References 112

Chapter 4 PHP 115

4.1 Concept of PHP 115
Definition 116
Common uses of PHP 117
Characteristics of PHP 117
"Hello World" Script in PHP 119
Pros and Cons of PHP 120
Different Types of PHP Tools 121
PHP Environment Setup 122
PHP Parser Installation 123
4.2 Functions of PHP 124
Definition 124
Data Types in PHP 130
Variables in PHP 137
4.3 PHP File Handling 143

Definition 143
Error & Exception Handling 147
Expressions and Operators 152
Summary 168
Knowledge Check 169
Review Questions 170
References 171

Chapter 5 Overview of AJAX 173

5.1 Basic Concept of AJAX 173
Concept 174
Characteristics of an Ajax Application 176
Using AJAX in Web Development 177
5.2 Rich Internet Applications 180
Definition 180
Features of Ajax 181
Architecture of Microsoft Ajax Applications 182
How does AJAX work? 186
5.3 Technologies used in AJAX 188
Use Technologies in AJAX 188
Steps of AJAX Operation 189
XMLHTTP Request object methods: 191
XMLHTTP Request object Properties 191
Ajax Security 192
Ajax Issues 193
What is Ajax and where is it Used in Technology? 194
Simple Ajax Application 197
XML Requests to the Server 197
XML Responses from the Server 198
Summary 203
Knowledge Check 203
Review Questions 205
References 206

Chapter 6 Essential of Web API 207

6.1 Functionality of Application Programing Interface 207
Concept 208
Detailed explanation 209
Web APIs 214
Implementations 216
6.2 API Uses 219
Basics 219
Operating systems 220
Remote APIs 221
Use in Web APIs 223
Aspect of API Documentation 224
6.3 API Testing 227
Strategy 227

Challenges with API Testing 228
Start API testing....230
Summary....234
Knowledge Check 234
Review Questions 236
References....236

Chapter 7 ColdFusion MX 237

7.1 Concept of ColdFusion MX....237
Definition 238
Flash Remoting Services 239
Web Services....239
Native XML Handling 240
Simplified Security....241
Database Integration....241
7.2 Elements of CFML....244
Definition 244
Comments 244
ColdFusion Components....250
Data Types....252
Character Case 256
Reserved Words....257
7.3 Extending ColdFusion Pages with CFScript....260
Basics 260
Comparing tags and CFScript....261
The CFScript Language 262
Using CFScript Statements 266
Summary....277
Knowledge Check 277
Review Questions 279
References....279

Chapter 8 Web Graphics....281

8.1 Basics of Web Graphics 281
Definition 282
Pixels....282
Resolution 282
Graphics and File Size 284
Uses of Graphics 284
8.2 Graphic File Formats....287
Perception 287
JPEG (Joint Photographic Experts Group)....288
GIF (Graphic Information Format) 289
PNG (Progressive Network Graphics)....291
Native File Formats 292
Other File Formats 293
8.3 Graphics as Content 294
Definition 294

Illustrations 295
Diagrams and Quantitative Data 295
Integrated Visual Presentations 296
Graphic Communication on the Web 296
Visual Communication Principles in Graphics 297
Summary 302
Knowledge Check 303
Review Questions 304
References 305

Chapter 9 Web Design and Development 307

9.1 Development in Web Design 307
Concept 308
Importance of Web Design 309
Choosing a Web Design Tool 309
Role of Web Designers 310
Related Elements of Web Design 312
Use of Web Design 314
9.2 Web Design Elements 316
Overview 316
Visual Elements 316
Functional Elements 318
Lookup for Good Web Design 320
Web design: what doesn't work? 321
How to get Web Design Done 322
9.3 Types of Website Design: Adaptive Vs. Responsive 323
Observation 323
Adaptive Websites 324
Adaptive Website Builders 324
Responsive Websites 325
Responsive Website Builders 326
9.4 Principles of Design Applied to Websites 327
Principles 327
Website Layouts 329
Functional Components of Web Design 330
Website Maintenance 332
Summary 334
Knowledge Check 335
Review Questions 335
References 336

Index 337

PREFACE

Web development is a rapidly growing field that encompasses various skills and disciplines. It involves creating websites or web applications, and it plays a crucial role in shaping the online landscape. Web developers utilize programming languages such as HTML, CSS, and JavaScript to design and build user-friendly and visually appealing websites. The process of web development begins with understanding client requirements and gathering information about the target audience. This information is used to create wireframes or mockups, which provide a visual representation of the website's structure. Once the initial planning phase is complete, developers move on to coding the website using markup languages like HTML. HTML provides structure to the content by defining paragraphs, headers, lists, images, links, etc. Cascading Style Sheets (CSS) are applied alongside HTML to enhance the appearance of web pages. CSS controls elements' style attributes like color, font size, layout positioning - allowing designers to create responsive designs that look great on all devices. In addition to HTML and CSS, JavaScript is what brings interactivity to websites through algorithms executed by web browsers flawlessly. Using JavaScript libraries or frameworks like ReactJS or AngularJS streamlines development processes by providing pre-designed building blocks for complex functionalities. Web developers should also have an understanding of databases like MySQL or MongoDB for storing data within web applications dynamically. They need knowledge of server-side scripting using PHP, Python or Node.js as well as familiarity with content management systems (CMS) such as WordPress.

The book "Theory, Practice and Techniques in Web development" 2nd Design offers a comprehensive guide for individuals entering the field of web development. Focusing primarily on front-end aspects such as HTML authoring, graphic production, and media development, this resource provides insight into real-world projects through the design, planning, and construction process. While it does not delve into programming, scripting or server functions in depth, it lays the necessary foundation by introducing relevant terminology and technologies. Suitable for all skill levels from beginners seeking detailed explanations to professionals requiring quick reference materials, this book serves as an indispensable tool for acquiring skills needed to create dynamic and interactive web applications using various frameworks—making it an essential companion for students delving into the world of web development.

CHAPTER 1

CREATING WEB PAGES WITH HTML

1.1 FUNDAMENTAL OF HTML

KEY NOTES

Basics

HyperText Markup Language (HTML) is the set of markup symbols or codes inserted into a file intended for display on the Internet. The markup tells web browsers how to display a web page's words and images.

HTML uses a system of elements and tags to describe the content and its formatting. Elements are represented by tags enclosed in angle brackets, like <tag>, and they usually come in pairs: an opening tag and a closing tag. The content to be affected by a tag is placed between these tags.

HTML Tags

HTML tag is an HTML code that defines every structure on an HTML page, including the placement of text and images and hypertext links. HTML tags begin with the less-than (<) character and end with greater-than (>). These symbols are also called "angle brackets."

Learning Objectives

- Paragraph Tag
- Line Break Tag
- Preserve Formatting
- Nonbreaking Spaces

Basics

Hypertext Markup Language (HTML) serves as the fundamental building block of web pages, providing a structured framework for organizing and displaying content on the internet. HTML employs a system of elements and tags to define the structure and format of a webpage. These elements are enclosed within angle brackets (< >) and typically consist of paired opening and closing tags, with the content placed between them.

At the root of every HTML document is the <html> element, which encapsulates the entire page. The <head> section contains metadata about the document, including the <title> tag that specifies the title displayed in the browser tab. The <body> section comprises the visible content of the webpage, containing elements such as headings (<h1>, <h2>, etc.) used to emphasize titles, and paragraphs (<p>) for standard text.

Links are integral to the web experience, and HTML's <a> element is employed to create hyperlinks. The "href" attribute within the <a> tag defines the URL the link points to, enabling users to navigate to other web pages or resources. Images can be displayed using the <img> element, with the "src" attribute specifying the image file's source. Additionally, the <ul> and <ol> elements create unordered and ordered lists, respectively, while the <li> element defines list items within these lists.

HTML tables (<table>) facilitate the organization of data into rows and columns, with <tr> indicating table rows and <td> or <th> denoting table cells, where <th> represents header cells. Forms are pivotal for user interaction, enabling data input and submission. HTML's <form> element initiates a form, while input fields such as text boxes, checkboxes, and radio buttons are created using <input> tags, each with its designated type attribute.

While HTML defines the structure of a webpage, its appearance and interactivity can be enhanced using Cascading Style Sheets (CSS) and JavaScript. CSS provides styling instructions to control the presentation of HTML elements, from fonts and colors to layout and spacing. JavaScript, on the other hand, enables dynamic behavior, allowing developers to create interactive features like pop-up dialogs, sliders, and dynamic content updates.

HTML forms the cornerstone of web development by structuring content in a standardized manner. Its simple yet powerful syntax allows developers to create a wide range of web experiences, from basic text and images to complex forms and interactive interfaces. By mastering the basics of HTML, aspiring web developers can establish a strong foundation for building functional and visually appealing websites.

Here's a simple example of HTML code:

<!DOCTYPE html>

<html>

<head>

```
<title>My First Webpage</title>
</head>
<body>
<h1>Hello, World!</h1>
<p>This is a paragraph of text.</p>
<a href="https://www.example.com">Visit Example.com</a>
</body>
</html>
```

In this example:

- **<!DOCTYPE html>** declares the document type and version of HTML being used (HTML5 in this case).
- **<html>** is the root element that encapsulates the entire HTML document.
- **<head>** contains metadata about the document, such as the title displayed in the browser's title bar.
- **<title>** sets the title of the webpage that appears in the browser tab.
- **<body>** contains the visible content of the webpage.
- **<h1>** is a heading element used for main headings, with decreasing levels of importance represented by **<h2>**, **<h3>**, and so on.
- **<p>** is a paragraph element for regular text.
- **<a>** creates a hyperlink, and the **href** attribute specifies the URL the link points to.

Document is a written, drawn, presented, or memorialized representation of thought.

HTML is a language for structuring content and doesn't handle more advanced interactions or styling on its own. Cascading Style Sheets (CSS) and JavaScript are commonly used alongside HTML to control the visual appearance and add interactivity to web pages.

The web has gone through many changes over the past few decades, but HTML has always been the fundamental language used to develop webpages. Interestingly, while websites have become more advanced and interactive, HTML has actually gotten simpler. If you compare the source of an HTML5 page with a similar page written in HTML 4.01 or XHTML 1.0, the HTML5 page would probably contain less code. This is because modern HTML relies on cascading style sheets or JavaScript to format nearly all the elements within a page.

- HTML is the standard markup language for creating Web pages.
- HTML stands for Hyper Text Markup Language
- HTML describes the structure of Web pages using markup
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as “heading”, “paragraph”, “table”, and so on
- Browsers do not display the HTML tags, but use them to render the content of the page

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Example Explained

The <!DOCTYPE html> declaration defines this document to be HTML5

The <html> element is the root element of an HTML page

The <head> element contains meta information about the document

The <title> element specifies a title for the document

The <body> element contains the visible page content

The <h1> element defines a large heading

The <p> element defines a paragraph

HTML Tags

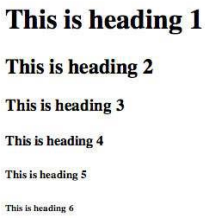
Any document starts with a heading. You can use different sizes for your headings. HTML also has six levels of headings, which use the elements <h1>, <h2>, <h3>,

<h4>, <h5>, and <h6>. While displaying any heading, browser adds one line before and one line after that heading.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Heading Example</title>
</head>
<body>
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
</body>
</html>
```

This will produce following result:



Paragraph Tag

The tag offers a way to structure your text into different paragraphs. Each paragraph of text should go in between an opening and a closing tag as shown below in the example:

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Paragraph Example</title>
</head>
<body>
```

```
<p>Here is a first paragraph of text.</p>
<p>Here is a second paragraph of text.</p>
<p>Here is a third paragraph of text.</p>
</body>
</html>
```

Rendering involves interpreting the HTML, CSS, and potentially JavaScript code of a webpage and presenting it as a visual representation that users can interact with.

This will produce following result:

Here is a first paragraph of text.

Here is a second paragraph of text.

Here is a third paragraph of text.

Line Break Tag

Whenever you use the <br /> element, anything following it starts from the next line. This tag is an example of an empty element, where you do not need opening and closing tags, as there is nothing to go in between them.

The <br /> tag has a space between the characters br and the forward slash. If you omit this space, older browsers will have trouble rendering the line break, while if you miss the forward slash character and just use <br> it is not valid in XHTML

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Line Break  Example</title>
</head>
<body>
<p>Hello<br />
You delivered your assignment ontime.<br />
Thanks<br />
Mahnaz</p>
</body>
</html>
```

This will produce following result:

Hello

You delivered your assignment ontime.

Thanks

Mahnaz

Centering Content

You can use tag to put any content in the center of the page or any table cell.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Centring Content Example</title>
</head>
<body>
<p>This text is not in the center.</p>
<center>
<p>This text is in the center.</p>
</center>
</body>
</html>
```

This will produce following result:

This text is not in the center.

This text is in the center.

Horizontal Lines

Horizontal lines are used to visually break up sections of a document. The <hr> tag creates a line from the current position in the document to the right margin and breaks the line accordingly.

For example you may want to give a line between two paragraphs as in the given example below:

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Horizontal Line Example</title>
</head>
<body>
<p>This is paragraph one and should be on top</p>
```

```
<hr />
<p>This is paragraph two and should be at bottom</p>
</body>
</html>
```

This will produce following result:

This is paragraph one and should be on top

This is paragraph two and should be at bottom

HTML documents consist of a set of elements and tags that define the various components and layout of the page.

Again <hr /> tag is an example of the empty element, where you do not need opening and closing tags, as there is nothing to go in between them.

The <hr /> element has a space between the characters hr and the forward slash. If you omit this space, older browsers will have trouble rendering the Horizontal line, while if you miss the forward slash character and just use <hr> it is not valid in XHTML

Preserve Formatting

Sometimes you want your text to follow the exact format of how it is written in the HTML document. In those cases, you can use the preformatted tag <pre>.

Any text between the opening <pre> tag and the closing </pre> tag will preserve the formatting of the source document.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Preserve Formatting Example</title>
</head>
<body>
<pre>
function testFunction( strText ){
  alert (strText)
}
</pre>
</body>
</html>
```

This will produce following result:

```
function testFunction( strText ){
  alert (strText)
}
```

Try using same code without keeping it inside <pre>...</pre> tags

Nonbreaking Spaces

Suppose you want to use the phrase "12 Angry Men." Here you would not want a browser to split the "12, Angry" and "Men" across two lines:

An example of this technique appears in the movie "12 Angry Men."

In cases where you do not want the client browser to break text, you should use a nonbreaking space entity   instead of a normal space. For example, when coding the "12 Angry Men" in a paragraph, you should use something similar to the following code:

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Nonbreaking Spaces Example</title>
</head>
<body>
<p>An example of this technique appears in the movie "12 Angry Men."</p>
</body>
</html>
```

1.2 GETTING STARTED

Key Notes

Overview

HTML was invented in 1990 by a scientist called Tim Berners-Lee. The purpose was to make it easier for scientists at different universities to gain access to each other's research documents. The project became a bigger success than Tim Berners-Lee had ever imagined. By inventing HTML he laid the foundation for the web as we know it today.

Naming Files and Folders

Filenames are not kept secret from the public; they appear in browsers' address bars. And an emerging approach in file-naming is to create more intuitive (and longer) filenames.

Create a Simple Web Page with HTML

Although you can make a webpage without knowing HTML, you will have inevitable problems no matter what webpage editor you're using and you will need to know HTML to fix it.

Learning Objectives

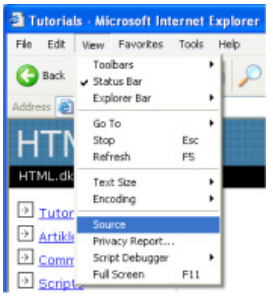
- Creating an HTML Document
- HTML Attribute

Overview

HTML is a language, which makes it possible to present information (e.g. scientific research) on the Internet. What you see when you view a page on the Internet is your browser's interpretation of HTML. To see the HTML code of a page on the Internet, simply click "View" in the top menu of your browser and choose "Source".

For the untrained eye, HTML code looks complicated but this tutorial will help you make sense of it all.

If you want to make websites, there is no way around HTML. Even if you're using a program to create websites, such as Dreamweaver, a basic knowledge of HTML can make life a lot simpler and your website a lot better. The good news is that HTML is easy to learn and use.



HTML is an abbreviation of "HyperText Mark-up Language" - which is already more than you need to know at this stage. However, for the sake of good order, let us explain in greater detail.

Hyper is the opposite of linear. In the good old days - when a mouse was something the cat chased - computer programs ran linearly: when the program had executed one action it went to the next line and after that, the next line and so on. But HTML is different - you can go wherever you want and whenever you want. For example, it is not necessary to visit MSN.com before you visit HTML.net.

Computer programs are sets of instructions written in programming languages that direct computers to perform specific tasks or functions.

Text is self-explanatory.

Mark-up is what you do with the text. You are marking up the text the same way you do in a text editing program with headings, bullets and bold text and so on.

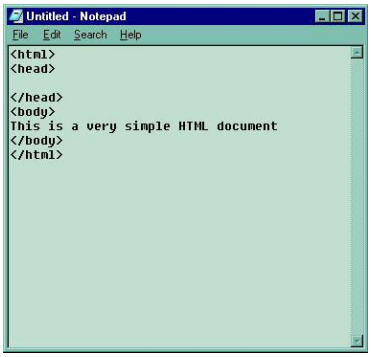
Language is what HTML is. It uses many English words.

In this tutorial you will learn so-called XHTML (Extensible HyperText Mark-up Language) which, in short, is a new and well-structured way of writing HTML.

Now you know what HTML (and XHTML) stands for let's get started with what it is all about: making websites.

Creating an HTML Document

To create a new HTML document, open a new document in Notepad (Notepad should be located under programs/ accessories). You will type all your HTML code into this document. To start, type the following HTML code into Notepad.



When you are done typing save your document. When you save an HTML document in Notepad, you must specify that it should be saved as an HTML document. To do this, choose Save

As from the file menu. Type in the file name followed by .html. You must also change the Save as Type to All Files or Notepad will automatically save your document as a text document.



Notepad is a simple text editor for Microsoft Windows and a basic text-editing program which enables computer users to create documents.

Next open an Internet browser (ex. Netscape or Internet Explorer). Choose Open or Open Page from the file menu. Locate and select your HTML document. If you cannot find your HTML document, go back to the previous step and make sure that you saved it properly. Your HTML document should now be displayed in the browser window.



You have now created your first webpage. If you want to make changes to your HTML code, you must save your HTML document in **Notepad** and then click on Reload or Refresh in the browser window in order to view your changes.

Naming Files and Folders

If the page you are creating is going to be the home page for your site, you should name it index.html.

You might think of a home page as the page visitors arrive at first when they come to your site. And you are correct. But how does a browser know that when your visitors enter “pat.com” (assuming for the moment that you are Pat and pat.com is your site), which of the many HTML pages at your site to open first?

Browsers detect your home page by looking for a file called index.html, which plays a special role. By default, when a browser is directed to a URL (like davidkarlins.com), the browser looks for a file called index.html. In other words, index.html serves as the home page within any folder. And in your site's root folder (your URL), the index.html file serves as the site home page.

These tips are intended to save you time as you create and edit your Web pages.

1. HTML file names should end in the suffix ".html" or ".htm". Whichever suffix you decide to use, please talk to your team members and use the same suffix on all your project html pages. Otherwise it is hard to remember which is which and your project will likely have more broken links in it. Example:

   myhomepage.html

   In addition, use a consistent and standard suffix on other file formats indicating the format (.jpg, .gif, .wav, . . .).

2. The file name should be no more than 32 characters, including the ".html" or ".htm" file suffix. Because your filename will become part of your Web page's URL, it is better to use a slightly longer file name that will have an easier URL to remember than a jumble of unpronounceable letters. For example: Which of these would be easier to remember and give to a friend as a URL: hurricanes.html or hrrcns.html ? More examples:

   kennedy.html

   shakespeare.htm

3. The first character of the file name should be a letter.

4. **File names and folder names** should contain **only letters, digits, and underscores—no spaces, punctuation, or funny characters.** We want to really emphasize this because people who do not follow these guidelines end up spending much more time fixing broken links, which can be a source of frustration. This is a "conservative" guideline with the goal that you should not have to spend extra time redoing your links if you move your files from one type of computer to another. This has been an issue for some past LInC participants when moving or copying their LInC project Web pages from the LInC class Web site to their school Web site.

5. Whenver possible, make relative links for links from your project Web pages to other Web pages that are in that same project. This will make it possible to move your LInC project Web site to your school Web server without having to fix dozens of links. A relative link to a file in the same folder as the web page you are editing looks like: «shakespeare.html». In contrast, an «absolute» link looks like: «http://ed.fnal.gov/lincon/projects/essays/shakespeare.shtml». You can see that if your project was copied onto a different web server, you would

have to fix all those absolute links. You will need to use absolute links when linking to Web pages for external resources that are not inside your project folder.

6. We **strongly** suggest using all lower case letters because people make fewer mistakes. Web authors (such as yourself) make fewer mistakes when creating links to your pages. Web readers make fewer mistakes when typing in the addresses to access your pages.

7. Use relatively short names that are indicative of the content of the page. Nothing «cute» because you probably won›t remember it six months from now when you need to edit it.

8. Also name your files with images, sounds, etc., with names that will help you remember what is in them. Participants who name their images something like picture1.jpg through picture20.jpg are frequently frustrated later because they have to open five or ten files later to find the image they want to edit. If you instead name your images something more descriptive, they will be easier to find later. Examples:

   lightbulb.gif

   book.jpg

   bluebullet.gif

   redline.jpg

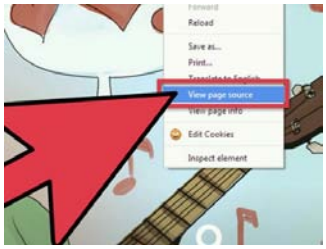
   stopsign.gif

9. Be sure to create a graphics folder (directory) inside the folder (directory) you are using for all your Web pages. It should be titled «graphics»--all lower case letters. Past participants who did not do this have told us they wish they did because it was harder to keep their files organized with so many pictures in with the Web pages.

10. Name your anchors in your pages something descriptive. An anchor named «#assessment» is easier to link to and tell others about than an anchor named «#anchor82904872».

Create a Simple Web Page with HTML

There are many good websites where you can learn HTML, the basics of creating a website by using HTML. With this basic introduction, you'll soon be able to make a webpage from scratch!



Using HTML

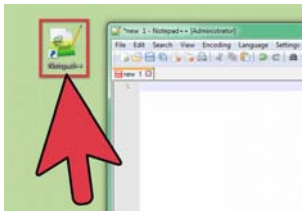
1. Understand what HTML is. HTML (Hyper Text Markup Language) is the coding language that makes web-pages, it is the skeleton of every web-page that exists. To see what it looks like, go into Internet Explorer, navigate to any website and right-click on the web-page selecting "View Source". You will see a page of code, and that is HTML. That code is what your browser sees and interprets into a human friendly web-page.

- In Google Chrome press F12.
- In Mozilla Firefox, press Ctrl + U to view the page›s source code.
- In Safari, select View→View Source (or Option + Command + U).
- In Internet Explorer, the VIEW menu option is «source».



2. Make your initial web page as simple as possible**.** If you don't, you'll likely get overwhelmed by the syntax and the script languages.

- It›s important to remember that you›ll be writing your information between an opening HTML tag and a closing HTML tag. An opening tag looks like this: <___> . A closing tag looks like this: </___> . Eventually, the ___ is replaced with a code.



Go to Start, then "Programs" and then "Accessories." and click "Notepad." It is a lot easier if you use Notepad++ once you choose the HTML language, everything you write will be automatically connected with different colors - that way it will be a lot easier to correct possible mistakes.



Tell the browser what language you are using. On the first line, type <!DOCTYPE>, this should always be the first tag in your code. On the second line add <html>. It is the second tag you write that tells the computer you're starting a web-page. It will also be closed last, so at the end of the document, close it off by typing this: </html>. This ends the web page.



Add the heading of the page as shown.



Give your page a title. A title is important because it gives your users an idea on what the page is all about. Also, when users bookmark your site, that title is all they will see in their bookmark list. The title for HTML code is <title>. Close it off at the end of your title by writing </title> The title is going to show on the tab, don't expect it to be the title of the actual website.



Work on the body of the page**.** Type <body> to open the body tag. Then close the body tag by typing </body>. The bulk of the information for your web-page goes between <body> and </body>.

- To give your web-page a background color, you can add a style to the body. To add a background color, add this tag <body style=»background-color:silver»>. You can try a different color or even a hex code. The words in the quotation marks are known as «attributes.» They must be surrounded by quotation marks!



Write some text between the body tags.

- To make the text go to the next line (like pressing “Enter” on your keyboard), write
- Want to add a marquee, otherwise known as a word that moves across a screen? Simply type <marquee>TEXT GOES HERE</marquee>.



Add some pictures. If you want to put a picture from the Internet onto your web page, the HTML code for pictures is <img src=”URL”> . The closing tag is: . However, the closing tag is optional.



Check to make sure all of your tags are closed. Your webpage should look something like this:

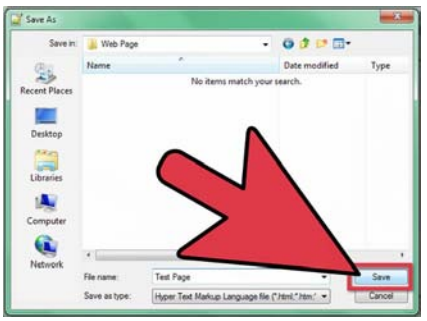
<title>My Web-page</title>

<body bgcolor=”yellow”>

I love Rose because

<marquee>It’s the best flower in the world!</marquee>

<img src=”http://www.rose.com/skins/common/images/roseimg_logo_5.gif”></body>



Save your work. Go to “save as”, put a filename with an .html extension (such as “testfile.html”) and choose “all files” or “text” under file type. It won’t work if both are not done. Go find the page wherever you saved it, double click it, and your default web browser should open up your very own web-page.

HTML Attribute

We have seen few HTML tags and their usage like heading tags <h1>, <h2>, paragraph tag <p> and other tags. We used them so far in their simplest form, but most of the HTML tags can also have attributes, which are extra bits of information.

An attribute is used to define the characteristics of an HTML element and is placed inside the element’s opening tag. All attributes are made up of two parts: a name and a value:

- The name is the property you want to set. For example, the paragraph <p> element in the example carries an attribute whose name is align, which you can

use to indicate the alignment of paragraph on the page.

- The value is what you want the value of the property to be set and always put within quotations. The below example shows three possible values of align attribute: left, center and right.

Attribute names and attribute values are case-insensitive. However, the World Wide Web Consortium (W3C) recommends lowercase attributes/attribute values in their HTML 4 recommendation.

Web page is a document that is suitable for the World Wide Web and web browsers. A web browser displays a web page on a monitor or mobile device.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Align Attribute  Example</title>
</head>
<body>
<p align=”left”>This is left aligned</p>
<p align=”center”>This is center aligned</p>
<p align=”right”>This is right aligned</p>
</body>
</html>
```

This will display following result:

This is left aligned

This is center aligned

This is right aligned

Core Attributes

The four core attributes that can be used on the majority of HTML elements (although not all) are:

- id
- title
- class
- style

The id Attribute

The **id** attribute of an HTML tag can be used to uniquely identify any element within an HTML page. There are two primary

reasons that you might want to use an id attribute on an element:

- If an element carries an id attribute as a unique identifier it is possible to identify just that element and its content.
- If you have two elements of the same name within a **Web page** (or style sheet), you can use the id attribute to distinguish between elements that have the same name.

We will discuss style sheet in separate tutorial. For now, let's use the id attribute to distinguish between two paragraph elements as shown below.

Example

```
<p id="”html”>This para explains what is HTML</p>
<p id="”css”>This para explains what is Cascading Style Sheet</p>
```

The title Attribute

The **title** attribute gives a suggested title for the element. They syntax for the **title** attribute is similar as explained for **id** attribute:

The behavior of this attribute will depend upon the element that carries it, although it is often displayed as a tooltip when cursor comes over the element or while the element is loading.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>The title Attribute Example</title>
</head>
<body>
<h3 title=»Hello HTML!»>Titled Heading Tag Example</h3>
</body>
</html>
```

This will produce following result:

Titled Heading Tag Example

Now try to bring your cursor over «Titled Heading Tag Example» and you will see that whatever title you used in your code is coming out as a tooltip of the cursor.

The class Attribute

The **class** attribute is used to associate an element with a style sheet, and specifies the class of element. You will learn more about the use of the class attribute when you will learn Cascading Style Sheet (CSS). So for now you can avoid it.

The value of the attribute may also be a space-separated list of class names. For example:

class=»className1 className2 className3»

The style Attribute

The style attribute allows you to specify Cascading Style Sheet (CSS) rules within the element.

```
<!DOCTYPE html>
<html>
<head>
<title>The style Attribute</title>
</head>
<body>
<p style=”font-family:arial; color:#FF0000;”>Some text...</p>
</body>
</html>
```

This will produce following result:

Some text...

At this point of time, we are not learning CSS, so just let’s proceed without bothering much about CSS. Here you need to understand what are HTML attributes and how they can be used while formatting content.

Internationalization Attributes

There are three internationalization attributes, which are available for most (although not all) XHTML elements.

- dir
- lang
- xml:lang

The dir Attribute

The **dir** attribute allows you to indicate to the browser the direction in which the text should flow. The dir attribute can take one of two values, as you can see in the table that follows:

Value	Meaning
ltr	Left to right (the default value)
rtl	Right to left (for languages such as Hebrew or Arabic that are read right to left)

Example

```
<!DOCTYPE html>
<html dir=”rtl”>
<head>
<title>Display Directions</title>
</head>
<body>
This is how IE 5 renders right-to-left directed text.
</body>
</html>
```

This will produce following result:

This is how IE 5 renders right-to-left directed text.

When *dir* attribute is used within the <html> tag, it determines how text will be presented within the entire document. When used within another tag, it controls the text's direction for just the content of that tag.

The lang Attribute

The **lang** attribute allows you to indicate the main language used in a document, but this attribute was kept in HTML only for backwards compatibility with earlier versions of HTML. This attribute has been replaced by the **xml:lang** attribute in new XHTML documents.

The values of the *lang* attribute are ISO-639 standard two-character language codes. Check **HTML Language Codes: ISO 639** for a complete list of language codes.

Example

```
<!DOCTYPE html>
<html lang=»en»>
<head>
<title>English Language Page</title>
</head>
<body>
This page is using English Language
</body>
</html>
```

The xml:lang Attribute

The *xml:lang* attribute is the XHTML replacement for the *lang* attribute. The value of the *xml:lang* attribute should be an ISO-639 country.

Generic Attributes

Here›s a table of some other attributes that are readily usable with many of the HTML tags.

Attribute	Options	Function
align	right, left, center	Horizontally aligns tags
valign	top, middle, bottom	Vertically aligns tags within an HTML element.
bgcolor	numeric, hexidecimal, RGB values	Places a background color behind an element
background	URL	Places a background image behind an element
id	User Defined	Names an element for use with Cascading Style Sheets.
class	User Defined	Classifies an element for use with Cascading Style Sheets.
width	Numeric Value	Specifies the width of tables, images, or table cells.
height	Numeric Value	Specifies the height of tables, images, or table cells.
title	User Defined	“Pop-up” title of the elements.

We will see related examples as we will proceed to study other HTML tags. For a complete list of HTML Tags and related attributes.

1.3 HTML STRUCTURE

KEY NOTES

Perception

Mastering the HTML structure is fundamental for effective web development. It ensures that content is organized logically, accessibility considerations are addressed, and visual presentation is consistent. By adhering to best practices in structuring HTML documents, developers create web pages that are well-organized, functional, and user-friendly.

DOCTYPE

The <!DOCTYPE html> declaration is used to inform a website visitor's browser that the document being rendered is an HTML document. While not actually an HTML element itself, every HTML document should being with a DOCTYPE declaration to be compliant with HTML standards.

Html Element

An HTML element is an individual component of an HTML document or web page, once this has been parsed into the Document Object Model. HTML is composed of a tree of HTML nodes, such as text nodes. Each node can have HTML attributes specified. Nodes can also have content, including other nodes and text. Many HTML nodes represent semantics, or meaning. For example, the title node represents the title of the document.

Learning Objectives

- Nested HTML Elements
- Empty HTML Elements
- Head Element
- Body Element

Perception

The structure of an HTML document forms the foundation of how content is organized and presented on a web page. It comprises a series of hierarchical elements and tags that dictate the layout, arrangement, and semantic meaning of the content. Understanding the HTML structure is essential for creating well-structured and accessible web pages.

At the core of the HTML structure is the root element, <html>, which encapsulates the entire document. This element acts as a container for all other HTML elements and tags. Within the <html> element, two primary sections play distinct roles: the <head> and the <body>.

The <head> section contains metadata about the document, providing information that is not directly visible to users but is essential for the browser and search engines. One crucial element within the <head> is the <title> tag, which defines the title of the webpage displayed in the browser's title bar or tab. Other meta information, such as character encoding declarations and linked stylesheets, are often included here.

The <body> section houses the visible content of the webpage. It encompasses various HTML elements that structure the content logically and aesthetically. Headings, denoted by <h1>, <h2>, <h3>, and so on, establish a hierarchical structure for titles and subtitles. Paragraphs are defined using the <p> element, separating text into meaningful blocks.

Links are integral to web navigation, and the <a> (anchor) element creates hyperlinks. The href attribute within the <a> tag specifies the destination URL, allowing users to navigate to other pages or resources. Images are displayed using the <img> element, with the src attribute indicating the image file source and the alt attribute providing alternative text for screen readers and when images cannot be displayed.

Lists organize information, and HTML provides both ordered (<ol>) and unordered (<ul>) list types. List items within these lists are represented by the <li> element. For more complex content organization, HTML tables (<table>) offer a way to structure data into rows (<tr>) and cells (<td>) or header cells (<th>).

Forms facilitate user input and interaction. The <form> element creates a form, while input fields such as text boxes, checkboxes, and radio buttons are established using the <input> element. The type attribute within the <input> tag specifies the type of input field.

Comments (<!-- Comment goes here -->) can be added within the HTML code to provide explanations or notes for other developers and collaborators. These comments do not affect the rendered output and are solely for human understanding.

DOCTYPE

The <!DOCTYPE> declaration must be the very first thing in your HTML document, before the <html> tag. The <!DOCTYPE> declaration is not an HTML tag; it is an instruction to the web browser about what version of HTML the page is written in. In HTML 4.01, the <!DOCTYPE> declaration refers to a DTD, because HTML 4.01 was based on SGML. The DTD specifies the rules for the markup language, so that the browsers render the content correctly.

The HTML <doctype> tag is used for specifying which version of HTML the document is using. This is referred to as the document type declaration (DTD).

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML doctype Tag</title>
</head>
<body>
```

doctype declaration <doctype> is mentioned at the starting of every HTML document.

</body>

</html>

This will produce following result:

doctype declaration <doctype> is mentioned at the starting of every HTML document.

Declaration

HTML 4.01 has 3 possible doctypes: HTML 4 Strict, HTML 4 Transitional, and HTML 4 Frameset. Every HTML document you create should have one of these three DTDs.

HTML 4 Strict

This document type includes all HTML elements except those that have been deprecated, and those that appear in frameset documents.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"https://www.w3.org/TR/html4/strict.dtd">

HTML 4 Transitional

This document type includes all HTML elements including those that have been deprecated.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"https://www.w3.org/TR/html4/loose.dtd">;

HTML 4 Frameset

This document type includes all HTML elements in the transitional DTD as well as those in framed document.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"https://www.w3.org/TR/html4/frameset.dtd">

HTML 5 Declaration

In HTML5 there is only one declaration i.e.

<!DOCTYPE html>

Html Element

An HTML element usually consists of a start tag and end tag, with the content inserted in between:

<tagname>Content goes here...</tagname>

The HTML element is everything from the start tag to the end tag:

My first paragraph.

Nested HTML Elements

HTML elements can be nested (elements can contain elements).

All HTML documents consist of nested HTML elements.

This example contains four HTML elements:

Example

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Validation is the process of establishing documentary evidence demonstrating that a procedure, process, or activity carried out in testing and then production maintains the desired level of compliance at all stages.

Example Explained

The <html> element defines the whole document.

It has a start tag <html> and an end tag </html>.

The element content is another HTML element (the <body> element).

```
<html>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

The <body> element defines the document body.

It has a start tag <body> and an end tag </body>.

The element content is two other HTML elements (<h1> and).

```
<body>
<h1>My First Heading</h1>
```

My first paragraph.
</body>

The <h1> element defines a heading.
It has a start tag <h1> and an end tag </h1>.
The element content is: My First Heading.
<h1>My First Heading</h1>
The element defines a paragraph.
It has a start tag and an end tag .
The element content is: My first paragraph.
My first paragraph.
Do Not Forget the End Tag
Some HTML elements will display correctly, even if you forget the end tag:
Example

```
<html>
<body>

<p>This is a paragraph
<p>This is a paragraph

</body>
</html>
```

Empty HTML Elements

HTML elements with no content are called empty elements.

 is an empty element without a closing tag (the <br> tag defines a line break).
Empty elements can be "closed" in the opening tag like this: <br />.
HTML5 does not require empty elements to be closed. But if you want stricter **validation**, or if you need to make your document readable by XML parsers, you must close all HTML elements properly.

Use Lowercase Tags

HTML tags are not case sensitive: <P> means the same as .

The HTML5 standard does not require lowercase tags, but W3C recommends lowercase in HTML, and demands lowercase for stricter document types like XHTML.

Head Element

HTML Head tag is the first child of HTML Element. It Contains all information related to webpage, css links, scripts and meta tags. The HTML <head> element provides general information (metadata) about the document, including its title and links to its scripts and style sheets.

The HTML <head> Elemen

The <head> element is a container for metadata (data about data) and is placed between the <html> tag and the <body> tag. HTML metadata is data about the HTML document. Metadata is not displayed.

Metadata typically define the document title, character set, styles, links, scripts, and other meta information.

The following tags describe metadata: <title>, <style>, <meta>, <link>, <script>, and <base>.

The HTML <title> Element

The <title> element defines the title of the document, and is required in all HTML/XHTML documents.

The <title> element:

- defines a title in the browser tab
- provides a title for the page when it is added to favorites
- displays a title for the page in search engine results

A simple HTML document:

Example

```
<!DOCTYPE html>
<html>

<head>
  <title>Page Title</title>
</head>

<body>
The content of the document.....
</body>

</html>
```

The HTML <style> Element

The <style> element is used to define style information for a single HTML page:

Example

```
<style>
  body {background-color: powderblue;}
  h1 {color: red;}
  p {color: blue;}
</style>
```

The HTML <link> Element

The <link> element is used to link to external style sheets:

Example

```
<link rel="stylesheet" href="mystyle.css">
```

The HTML <meta> Element

The <meta> element is used to specify which character set is used, page description, keywords, author, and other metadata.

Metadata is used by browsers (how to display content), by search engines (keywords), and other web services.

Define the character set used:

```
<meta charset="UTF-8">
```

Define a description of your web page:

```
<meta name="description" content="Free Web tutorials">
```

Define keywords for search engines:

```
<meta name="keywords" content="HTML, CSS, XML, JavaScript">
```

Define the author of a page:

```
<meta name="author" content="John Doe">
```

Refresh document every 30 seconds:

```
<meta http-equiv="refresh" content="30">
```

Example of <meta> tags:

```
<meta charset="UTF-8">
<meta name="description" content="Free Web tutorials">
<meta name="keywords" content="HTML,CSS,XML,JavaScript">
<meta name="author" content="John Doe">
```

The HTML <script> Element

The <script> element is used to define client-side JavaScripts.

This JavaScript writes "Hello JavaScript!" into an HTML element with id="demo":

Example

```
<script>
function myFunction {
   document.getElementById("demo").innerHTML = "Hello JavaScript!";
}
</script>
```

The HTML <base> Element

The <base> element specifies the base URL and base target for all relative URLs in a page:

Example

```
<base href="https://www.w3schools.com/images/" target="_blank">
```

Language Attribute.

The lang attribute specifies the language of the element's content. Common examples are "en" for English, "es" for Spanish, "fr" for France and so on. The lang attribute is part of the Global Attributes, and can be used on any HTML element.

Some French text in a paragraph:

```
<p lang="fr">Ceci est un paragraphe.</p>
```

The lang Attribute specifies the natural language of the content of a web page. An attribute on the html tag sets the language for all the text on the page. If part of the page uses text in a different language, you can add a language attribute with a different value to the element that surrounds that content.

Body Element

The <body> element contains the entire content of a webpage. It must be the second element inside of the parent <html> element, following only the <head> element.

<!DOCTYPE html> <html> <head> . . . </head> <body> <!-- Page content goes here --> </body> </html>

The <body> element may be the most important HTML element there is. The contents of the <body> element are what actually get displayed to the user visiting your web page or viewing your document. And yet, it is very simple. The <body> must be a direct child of <html>, it must come after <head> (if <head> is present, which it doesn't have to be), there can be only one of it, and … that's it.

body { font-family: "Adobe Garamond Premier Pro", Garamond, "Linux Libertine", Baskerville, Georgia, serif; font-weight: normal: font-size: 18px; color:

#111111; background-color: #fefefe; padding: 1em; width: 95vw; max-width: 950px; margin: 0 auto; }

The example shows the basic structure of a webpage, including the body block.

```
<!DOCTYPE html>
<html>
  <head>
    <title>The beauty of life</title>
  </head>
  <body>
    <h1>The beauty of life</h1>
    <p>This morning I woke up feeling different...</p>
  </body>
</html>
```

Targeting the <body>

In content management systems, it is a common practice to place a list of content-specific CSS classes into the <body> element. This allows designers and JavaScript developers to target pages based on specific content attributes, such as category, tags, or content type.

<body class=”post post-single post-template-default tag-html tag-element tag-document category-markup category-html post-123 with-comments social-sharing-on logged-in”>

These classes are added programmatically by different components and plugins, and provide an easy way for developers to target styling or functionality at highly specific sets of pages (and if they never get used, they aren’t causing any problems).



TECHNIQUE 1

USING HTML IN WEB PAGE

HTML is the programming language that powers the web. And like any language, once you master it, you can begin to create your own content, whether that's simple websites or complex web applications. This course provides an in-depth look at the essentials: the syntax of HTML and best practices for writing and editing your code. Senior staff author James Williamson reviews the structure of a typical HTML document, and shows how to section pages and format your content with HTML.

- □ We already practice this technique
- □ This technique would not be suitable for our operation
- □ We practiced this in the past and need to jumpstart
- □ We should implement this technique



TECHNIQUE 2

HTML TAGS

This is a list of tags used in the HTML language. Each tag starts with a tag opener (a less than sign) and ends with a tag closer (a greater than sign). Many tags have corresponding closing tags which identical except for a slash after the tag opener. (For example, the TITLE tag). Some tags take parameters, called attributes. The attributes are given after the tag, separated by spaces. Certain attributes have an effect simply by their presence, others are followed by an equals sign and a value. (See the Anchor tag, for example). The names of tags and attributes are not case sensitive: they may be in lower, upper, or mixed case with exactly the same meaning. (In this document they are generally represented in upper case.) Currently HTML documents are transmitted without the normal SGML framing tags, but if these are included parsers will ignore them.

- □ We already practice this technique
- □ This technique would not be suitable for our operation
- □ We practiced this in the past and need to jumpstart
- □ We should implement this technique



TECHNIQUE 3

WHY MARKUP USE IN HTML

HTML markup consists of several key components, including those called tags (and their attributes), character-based data types, character references and entity references. HTML tags most commonly come in pairs like <h1> and </h1>, although some represent empty elements and so are unpaired, for example <img>. The first tag in such a pair is the start tag, and the second is the end tag (they are also called opening tags and closing tags). Another important component is the HTML document type declaration, which triggers standards mode rendering.

- ☐ We already practice this technique
- ☐ This technique would not be suitable for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 4

CREATING DOCUMENT STRUCTURE IN HTML

Document structure in HTML refers to the organization and arrangement of elements within an HTML document. Proper document structure is essential for creating well-formed, accessible, and semantically meaningful web pages. It helps browsers and assistive technologies interpret the content correctly and provides a clear hierarchy for both developers and users. The basic components of HTML document structure include the <!DOCTYPE> declaration, the <html> element, the <head> section, and the <body> section.

- ☐ We already practice this technique
- ☐ This technique would not be suitable for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique

SUMMARY

- Hypertext Markup Language (HTML) serves as the fundamental building block of web pages, providing a structured framework for organizing and displaying content on the internet.
- Links are integral to the web experience, and HTML's <a> element is employed to create hyperlinks. The "href" attribute within the <a> tag defines the URL the link points to, enabling users to navigate to other web pages or resources.
- HTML forms the cornerstone of web development by structuring content in a standardized manner. Its simple yet powerful syntax allows developers to create a wide range of web experiences, from basic text and images to complex forms and interactive interfaces.
- The structure of an HTML document forms the foundation of how content is organized and presented on a web page. It comprises a series of hierarchical elements and tags that dictate the layout, arrangement, and semantic meaning of the content.
- The <head> section contains metadata about the document, providing information that is not directly visible to users but is essential for the browser and search engines. One crucial element within the <head> is the <title> tag, which defines the title of the webpage displayed in the browser's title bar or tab.
- Links are integral to web navigation, and the <a> (anchor) element creates hyperlinks. The href attribute within the <a> tag specifies the destination URL, allowing users to navigate to other pages or resources.
- The tag offers a way to structure your text into different paragraphs. Each paragraph of text should go in between an opening and a closing tag as shown below in the example:
- Horizontal lines are used to visually break up sections of a document. The <hr> tag creates a line from the current position in the document to the right margin and breaks the line accordingly.
- HTML is a language, which makes it possible to present information (e.g. scientific research) on the Internet. What you see when you view a page on the Internet is your browser's interpretation of HTML. To see the HTML code of a page on the Internet, simply click "View" in the top menu of your browser and choose "Source".
- HTML is an abbreviation of "HyperText Mark-up Language" - which is already more than you need to know at this stage. However, for the sake of good order, let us explain in greater detail.
- HTML file names should end in the suffix ".html" or ".htm". Whichever suffix you decide to use, please talk to your team members and use the same suffix on all your project html pages. Otherwise it is hard to remember which is which and your project will likely have more broken links in it. Example:myhomepage.html

KNOWLEDGE CHECK

1. The common element which describe the web page, is?

a. heading
b. paragraph
c. list
d. All of these

2. HTML stands for?

a. Hyper Text Markup Language
b. High Text Markup Language
c. Hyper Tabular Markup Language
d. None of these

3. Which of the following tag is used to mark a begining of paragraph ?

a. <TD>
b.
c. <P>
d. <TR>

4. From which tag descriptive list starts ?

a. <LL>
b. <DD>
c. <DL>
d. <DS>

5. Correct HTML tag for the largest heading is

a. <head>
b. <h6>
c. <heading>
d. <h1>

6. Which HTML tag is used to create a hyperlink?

a. <link>
b. <a>
c. <hyperlink>
d. <url>

7. What is the purpose of the <img> tag in HTML?

a. Create a paragraph of text
b. Embed a video
c. Display an image
d. Create a table

8. Which HTML element is used to define the structure of an HTML document?

a. <head>
b. <body>
c. <main>
d. <html>

REVIEW QUESTIONS

1. Provide examples of some commonly used HTML tags and their purposes.
2. What is the role of the <!DOCTYPE> declaration in an HTML document?
3. Why is it important to choose appropriate names for files and folders in web development?
4. How are HTML attributes used to provide additional information about HTML elements?
5. Explain the role of the <body> element in defining the visible content of a webpage.

Check Your Result

1. (d) 2. (a) 3. (c) 4. (c) 5. (d)
6. (b) 7. (c) 8. (d)

REFERENCES

1. Berners-Lee, Tim; Fischetti, Mark (2000). Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor. San Francisco: Harper.
2. Charles McCathie Nevile (14 December 2017). "HTML 5.2 is done, HTML 5.3 is coming". World Wide Web Consortium. Retrieved 15 December 2017.
3. Cimpanu, Catalin (29 May 2019). "Browser vendors Win War with W3C over HTML and DOM standards". ZDNet. Archived from the original on 29 May 2019. Retrieved 29 May 2019.
4. Document Object Model (DOM) Level 3 Core Specification», A. Le Hors, P. Le Hégaret, et al. (eds.), 7 April 2004, W3C Recommendation. Available at http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/.
5. Namespaces in XML 1.0 (third edition)», T. Bray, D. Hollander, A. Layman, R. Tobin, H. S. Thompson, 8 December 2009 Available at: http://www.w3.org/TR/2009/REC-xml-names-20091208
6. Spiesser, J., Kitchen, L.: Optimization of HTML automatically generated by WYSIWYG programs. In: 13th International Conference on World Wide Web, pp. 355—364. WWW '04. ACM, New York, NY (New York, NY, U.S., May 17–20, 2004)

CHAPTER 2

CSS

2.1 BASICS OF CSS

KEY NOTES

Definition

Cascading Style Sheets (CSS) is a vital tool for web developers, allowing them to separate content from design and control the visual presentation of HTML elements. Through the use of selectors, declarations, and specific rule application, CSS enables the creation of consistent and attractive user interfaces across various platforms and devices. Its evolution with CSS3 has brought about advanced features that contribute to the dynamic and interactive nature of today's web applications.

The Cascade Part of CSS

The cascade part of CSS means that more than one stylesheet can be attached to a document, and all of them can influence the presentation.

Benefits of CSS

Using CSS modularizes your websites in much the same way that programmers have been modularizing software code for year. Modular code in your website allows for greater design consistency and easier maintenance. CSS is the new standard and has been now for several years. It's time to separate the structure of HTML and the presentation of CSS.

Learning Objectives

- Discuss about CSS versions
- Identify CSS3 modules
- Elaborate the strategy of CSS box model

Definition

Cascading Style Sheets, commonly known as CSS, is a fundamental technology in web development that enhances the presentation and layout of HTML documents. It provides a powerful mechanism for separating the content and structure of a webpage from its visual design. CSS allows web designers and developers to control how elements are displayed on a page, enabling consistent and visually appealing user experiences across different devices and screen sizes.

CSS operates on the principle of cascading, where styles are applied in a hierarchical manner, allowing for both global and specific styling. This separation of content and presentation fosters cleaner and more maintainable code. The CSS rule set consists of selectors and declarations. A selector targets one or more HTML elements, while declarations define the styling properties to be applied to those elements.

Selectors can range from basic element selectors (like "p" for paragraphs) to more complex class selectors (like ".header" for header elements) and ID selectors (like "#main-content" for specific content areas). This flexibility enables precise targeting of elements for styling.

Declarations within CSS include properties and values. Properties specify the aspect of an element to be styled, such as color, font size, margin, padding, and more. Values are assigned to properties and determine how the selected elements will appear. For example, "color: blue;" would change the text color of a selected element to blue.

CSS also supports inheritance and specificity. Inheritance means that properties set on a parent element can be inherited by its child elements, reducing the need for repetitive styling. Specificity comes into play when conflicting styles are applied to the same element. The most specific style rule takes precedence, preventing unintended styling conflicts.

External stylesheets are a commonly used method to apply CSS to HTML documents. By linking an external stylesheet in the HTML document's head section, the same styles can be applied across multiple pages, promoting consistent branding and design. Additionally, inline styles and internal styles within the HTML document are alternatives to external stylesheets, although they are typically used for specific cases.

CSS3, the latest version of CSS at the time of my last update in September 2021, introduced a wide range of advanced features, including animations, transitions, media queries for responsive design, and flexible layouts with tools like Flexbox and Grid. These features empower developers to create modern, dynamic, and visually appealing web applications.



Attribute is a specification that defines a property of an object, element, or file.

The "Cascade" Part of CSS

When you use multiple styles they have a cascading effect on one another: the first style defined for an element is loaded by the browser; if another style affects the same element, the new definition will override the first only if its **attributes** are the same, otherwise the new definition will be added to the first definition, thus creating a combined style for the element.

For example, consider a style that defines the text color for a paragraph (<p>) as black. Later you define another style that affects the <p> tag with a border of one pixel. The resulting style for the <p> tag now includes both a black text color and a border around the paragraph one-pixel thick. Thus, these styles cascade onto one another and are combined into a single style.



Browsers that support CSS have a built-in cascading order of importance (from most to least):

- User-defined styles — Some browsers allow for users to define their own styles. For example, a color-blind user may specify specific colors for hyperlinks and always use that in the browser to override styles on each Web page.

- Inline styles
- Embedded styles
- Linked styles
- Imported styles
- Default browser styles

Who Creates and Maintains CSS?

CSS was invited by Håkon Wium Lie on October 10, 1994 and maintained through a group of people within the W3C called the CSS Working Group. The CSS Working Group creates documents called specifications. When a specification has been discussed and officially ratified by W3C members, it becomes a recommendation.

Module is a part of a program. Programs are composed of one or more independently developed modules that are not combined until the program is linked.

These ratified specifications are called recommendations because the W3C has no control over the actual implementation of the language. Independent companies and organizations create that software.

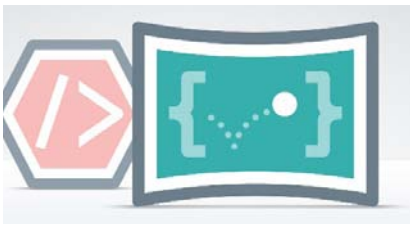
NOTE − The World Wide Web Consortium, or W3C is a group that makes recommendations about how the Internet works and how it should evolve.

CSS Versions

Cascading Style Sheets, level 1 (CSS1) was came out of W3C as a recommendation in December 1996. This version describes the CSS language as well as a simple visual formatting model for all the HTML tags.

CSS2 was became a W3C recommendation in May 1998 and builds on CSS1. This version adds support for media-specific style sheets e.g. printers and aural devices, downloadable fonts, element positioning and tables.

CSS3 was became a W3C recommendation in June 1999 and builds on older versions CSS. It has divided into documentations is called as Modules and here each **module** having new extension features defined in CSS2.



CSS3 Modules

CSS3 Modules are having old CSS specifications as well as extension features.

- Selectors
- Box Model
- Backgrounds and Borders
- Image Values and Replaced Content
- Text Effects
- 2D/3D Transformations
- Animations
- Multiple Column Layout
- User Interface

Browser is an application program that provides a way to look at and interact with all the information on the World Wide Web.

Benefits of CSS

Another of CSS's boons is that you define things *once*, making it far more efficient than defining everything in HTML on every page. This means:

- Pages download faster, sometimes by as much as 50%
- You have to type less code, and your pages are shorter and neater.
- The look of your site is kept consistent throughout all the pages that work off the same stylesheet.
- Updating your design and general site maintenance are made much easier, and errors caused by editing multiple HTML pages occur far less often.

Well-authored CSS also improves the accessibility of web content, allowing access through myriad devices (handheld PDAs for example) and ensuring that web users with disabilities are still able to receive it. It also eliminates the need for browser-specific hacks and tags, which means your site has a better chance of working across all major **browsers**.

Initially vaguely intimidating, CSS is a well-designed, elegant language. It is hugely important for the future of web design, and has been pivotal in helping designers move away from the problematic, hack-ridden days of presentational HTML tags like <font>, and allowed us to return to using logical, structural elements which make the sites more accessible.



All that, and there are dozens of powerful extra formatting options and possibilities available through stylesheet commands that are not possible through normal HTML. You will see these later on when we get on to things like backgrounds, spacing, layers and borders.

CSS Box Model

The CSS Box Model is a fundamental concept in web development that defines the properties and dimensions of an HTML element. It represents an element as a rectangular box, encompassing the content, padding, border, and margin. Understanding the box model is crucial for creating consistent layouts and managing spacing in web design.

The four main components of the CSS Box Model are:

Content: This is the innermost part of the box and contains the actual content of the element, such as text, images, or other nested elements. The width and height of the content area are determined by the width and height properties.

Padding: Padding is the space between the content and the border. It provides visual breathing room and can be adjusted using the padding property. Padding can have individual values for each side (top, right, bottom, left), or a single value that applies to all sides.

Border: The border surrounds the padding and content areas, providing a visible boundary for the element. It is defined using properties like border-width, border-style, and border-color. Like padding, borders can also be adjusted individually or as a whole.

Margin: The margin is the outermost area of the box and provides space between elements. It helps control the spacing between adjacent elements. The margin property determines how much space surrounds an element on the outside. Similar to padding, margins can also be set individually or uniformly.

It's important to note that the total width and height of an element include the content, padding, border, and margin. This means that when specifying the width or height of an element, you need to consider all these components. To calculate the total space an element occupies, you add up the content width/height, padding, border, and margin.

For example:

```
/* Applying box model properties */
.box {
  width: 200px;
  height: 150px;
  padding: 20px;
  border: 2px solid #333;
  margin: 10px;
}
```

In the example above, if the content width of .box is 200px and the padding is 20px on all sides, the total width of the element becomes 240px (200px content + 20px padding on both sides). Adding the 2px border on each side, the total width becomes 244px. Including the 10px margin on all sides, the total width of the space occupied by the element becomes 264px.

Understanding the CSS Box Model allows web developers and designers to precisely control spacing and layout, ensuring that elements are positioned and sized as intended. It's a cornerstone concept that plays a crucial role in creating visually appealing and responsive web layouts.

THEORY

2.2 CSS SYNTAX

KEY NOTES

Definition

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts −

Selector − A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.

Property - A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be *color*, *border* etc.

Value - Values are assigned to properties. For example, *color* property can have value either *red* or *#F1F1F1* etc.

CSS Selectors

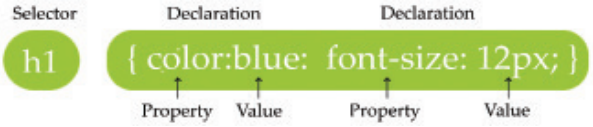
Selectors are where most of the power in CSS lies. Selectors allow the author to specifically target an element to apply any given property to. Many different selector types can be combined for the precision application of styles.

Learning Objectives

- Explain grouping selectors
- Discuss about CSS comments

Definition

A CSS rule-set consists of a selector and a declaration block:



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

In the following example all <p> elements will be center-aligned, with a red text color:

Example

```
p {
    color: red;
    text-align: center;
}
```

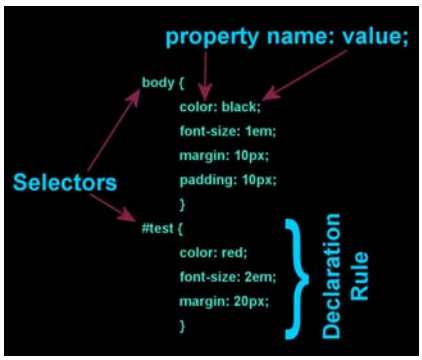
Though easy to use, CSS is a very specific standard. This means that if you do not get the grammar right the browser won't know what to do. Either your style sheet will not work at all, or it will work in unexpected ways. Above you have seen what the parts of a style sheet are, and an example of how they are put together. Referring to the examples above, here are some simple rules to follow to get your style sheet right every time.



- Every statement must have a selector and a declaration. The declaration comes immediately after the selector and is contained in a pair of curly braces.
- The declaration is one or more properties separated by semicolons.
- Each property has a property name followed by a colon and then the value for that property. There are many different types of values, but any given property can only take certain values as set down in the specification.
- Sometimes a property can take a number of values, as in the font-family. The values in the list should be separated by a comma and a space.
- Sometimes a value will have a unit as well as the actual value, as in the 1.3em. You must not put a space between the value and its unit.
- As with HTML, white space can be used to make your style sheet easier to read and write.

CSS Selectors

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.



The Element Selector

The element selector selects elements based on the element name.

You can select all <p> elements on a page like this (in this case, all <p> elements will be center-aligned, with a red text color):

Example

```
p {
    text-align: center;
    color: red;
}
```

Id Selector is a way to select only the element with the specified id , and apply styles to that element.

The id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element should be unique within a page, so the **id selector** is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

The style rule below will be applied to the HTML element with id=”para1”:

Example

```
#para1 {
    text-align: center;
    color: red;
}
```

Note: An id name cannot start with a number!

The class Selector

The class selector selects elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the name of the class.

In the example below, all HTML elements with class=”center” will be red and center-aligned:

Example

```
.center {
    text-align: center;
    color: red;
}
```

You can also specify that only specific HTML elements should be affected by a class.

In the example below, only <p> elements with class=”center” will be center-aligned:

Example

```
p.center {
    text-align: center;
    color: red;
}
```

HTML elements can also refer to more than one class.

In the example below, the <p> element will be styled according to class=”center” and to class=”large”:

Example

```
<p class="center large">This paragraph refers to two classes.</p>
```

Note: A class name cannot start with a number!

Grouping Selectors

If you have elements with the same style definitions, like this:

```
h1 {
    text-align: center;
    color: red;
}

h2 {
    text-align: center;
    color: red;
}

p {
    text-align: center;
    color: red;
}
```

It will be better to group the selectors, to minimize the code.

To group selectors, separate each selector with a comma.

In the example below we have grouped the selectors from the code above:

Example

```
h1, h2, p {
    text-align: center;
    color: red;
}
```

CSS Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

A CSS comment starts with /* and ends with */. Comments can also span multiple lines:

Example

```
p {
    color: red;
    /* This is a single-line comment */
    text-align: center;
}

/* This is
a multi-line
comment */
```

2.3 TYPES OF CSS STYLES

KEY NOTES

Definition

CSS (Cascading Style Sheets) is a language used to apply visual styling to HTML elements, enhancing the appearance and layout of web pages. CSS styles control aspects such as colors, fonts, spacing, positioning, and more.

Stylesheet definitions created with CSS can be inserted into an HTML document in a few different ways. There are three types of stylesheets:

- Inline styles
- Embedded styles
- External styles

Inline Style

Inline style sheet within a tag. Applies only to that particular occurrence of that tag.

Embedded Styles

Internal (also called Embedded) style sheet is defined within the head section of a page. Applies to that page only.

External Styles

External style sheet defined in a separate, hence external, file. Applies to all pages that link to the external style sheet.

Learning Objectives

- Understand websites code
- Discuss about attributes

Definition

To add CSS styles to your website, you can use three different ways to insert the CSS. You can Use an "*External Stylesheet*", an "*Internal Stylesheet*", or in "*Inline Style*". The benefit for using each depends on what you are doing with the Style. The following table explains the difference between them.

Table 2.1: Different CSS style linking

Internal Stylesheet	An internal stylesheet holds the CSS code for the webpage in the head section of the particular file. This makes it easy to apply styles like classes or id's in order to reuse the code. The downside of using an internal stylesheet is that changes to the internal stylesheet only effect the page the code is inserted into.

External Stylesheet	The External Stylesheet is a .css file that you link your website to. This makes it so that what ever you change in the .css sheet, will effect every page in your website. This prevents you from having to make many code changes in each page. This is for "*global*" site changes.
Inline Styles	The Inline style is specific to the tag itself. The inline style uses the HTML "*style*" attribute to style a specific tag. This is not recommended, as every CSS change has to be made in every tag that has the inline style applied to it. The Inline style is good for one an individual CSS change that you do not use repeatedly through the site.

Inline Styles

Inline styles are styles that are written directly in the tag in the HTML document. Inline styles affect only the specific tag they are applied to. Here is an example of an inline style applied to a standard link, or anchor, tag:

Web page is a document commonly written in HyperText Markup Language (HTML) that is accessible through the Internet or other network using an Internet browser.

```
<a href="/index.html" style="text-decoration: none;">
```

This CSS rule would turn the standard underline text decoration off of this one link. It would not, however, change any other link on the page. This is one of the limitations of inline styles. Since they only change on specific item, you would need to litter your HTML with these styles to achieve an actual page design. That is not a best practice. In fact, it is one step removed from the days of "font" tags and the mixture of structure and style in **web pages**.



Inline styles also have very high specificity. This makes them very hard to overwrite with other, non-inline styles. For

example, if you want to make a site responsive and change how an element looks at certain breakpoints by using media queries, inline styles on an element will make this very hard to do.

Ultimately, inline styles are really only appropriate when used very sparingly.

Creating an Inline Stylesheet

Use inline stylesheets when you want to apply a style to a single occurrence of an element.

Inline stylesheets are declared within individual tags and affect those tags only. Inline stylesheets are declared with the *style* attribute.

Example:

```
<p style="color:gray">This text will be gray.
</p>
```

In this example, we are using the stylesheet command **color** to denote that the text in a paragraph will be gray.

Output:

```
This text will be gray.
```

Website code refers to the underlying programming languages and markup used to create and structure the content, design, and functionality of a website.

Embedded Styles

Embedded styles are styles that are embedded in the head of the document. Embedded styles affect only the tags on the page they are embedded in. When creating a stylesheet internally in the web page, you will need to use the <style></style> HTML tags in the Head section of your webpage. All the code for the Internal CSS stylesheet is contained between the <head></head> section of your websites code. Below is an example of what an Internal stylesheet looks like.

```
<head>

  <style type="text/css">
    h1 {color:blue;}
    h2 {color:red;}
    p {color:green;}
  </style>

</head>
```



Creating an Internal Stylesheet

Use an internal stylesheet when you want an HTML document to have a unique style. An internal stylesheet is defined using the <style> tag and goes in the head section of an HTML document.

The <style> tag specifies the content type of a stylesheet with its **type attribute which should be set to "text/**css».

*Syntax***:**

```
<style type="text/css">
styles go here
</style>
```

*Example***:**

```
<html>
<head>
<style type="text/css">
p {color: green}
</style>
</head>
<body>
<p>
The text in this paragraph will be green.
</p>
<p>
This paragraph too.
</p>
</body>
</html>
```

The above stylesheet definition specifies that all text declared with the <p> tag should be green.

*Output***:**

```
The text in this paragraph will be
green.

This paragraph too.
```

NOTE**:** There are some old browsers still in use that do not support styles. Such browsers will ignore the <style> tag and will display the content within it on a webpage. You can prevent this by placing HTML comments within the <style> tag:

*Example***:**

```
<html>
<head>
<style type="text/css">
<!--
p {color: green;}
-->
</style>
</head>
<body>
<p>
The text in this paragraph will be green.
</p>
</body>
</html>
```

External Style Sheets

Most websites today use external style sheets. External styles are styles that are written in a separate document and then attached to various web documents. External style sheets can affect any document they are attached to, which means that if you have a 20-page website where each page uses the same style sheet (this is typically how it is done), you can make a visual change to every one of those pages by simply editing that style sheet.

This makes long term site management much easier.

The downside to external style sheets is that they require pages to fetch and load these external files. Not every page will use every style in the CSS sheet, so many pages will load a much larger CSS page than that actually need.

While it is true that there is a performance hit for external CSS files, it can certainly be minimized. Realistically, CSS files are just text files, so they are generally not very large to begin with. If you entire site uses 1 CSS file, you also get the benefit of that document being cached after it is initially loaded. This means that there could be a slight performance hit on the first page some visits, but subsequent pages will use the cached CSS file, so any hit would be negated.

Interface refers to a boundary or connection point between different systems, modules, or components. It defines how different entities can interact with each other and what operations or methods they can use to communicate and exchange information.



Creating an External Stylesheet

Use an external stylesheet when you want to apply one style to many pages. If you make one change in an external stylesheet, the change is universal on all the pages where the stylesheet is used.

An external stylesheet is declared in an external file with a .css extension. It is called by pages whose interface it will affect. External stylesheets are called using the <link> tag which should be placed in the head section of an HTML document. This tag takes three attributes.

Attributes of the <link> tag:

- rel - When using an external stylesheet on a webpage, this attribute takes the value «stylesheet»
- type - When using an external stylesheet on a webpage, this attribute takes the value «text/css»
- href - Denotes the name and location of the external stylesheet to be used.

Example:

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="style1.css"
/>
</head>
<body>
<p>
The text in this paragraph will be blue.
</p>
</body>
</html>
```

Output:

The text in this paragraph will be blue

The code from style1.css:

```
p {color:blue}
```

NOTE: The <style> tag is NOT used in an external stylesheet, and neither are HTML comments.



TECHNIQUE 5

MULTIPLE STYLESHEETS

You can have multiple stylesheet definitions on one page. For example, an internal stylesheet and an external stylesheet on one page or an inline stylesheet and an internal stylesheet on one page.

If a property has been set for the same tag in different stylesheet definitions on the same page, the definition that will be used will be from the most specific stylesheet. That is, the stylesheet that has the highest priority.

- □ We already practice this technique
- □ This technique would not be suitable for our operation
- □ We practiced this in the past and need to jumpstart
- □ We should implement this technique



TECHNIQUE 6

BODY TAG

The Body tag has properties and values that control the appearance of the page. The following code example code has the more popular properties and values

```
BODY {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 10pt;
background-color: #3333FF;
color: #000033;
}
```

- □ We already practice this technique
- □ This technique would not be suitable for our operation
- □ We practiced this in the past and need to jumpstart
- □ We should implement this technique



TECHNIQUE 7

CREATING "PRINT" STYLESHEETS

People will, sooner or later, print your Web page for their own use. If you don't create a print stylesheet, the browser's default will determine what the page looks like, and it usually isn't pretty. Do your users the courtesy of giving them a proper print stylesheet. They dramatically increase the usability of your site.

- □ We already practice this technique
- □ This technique would not be suitable for our operation
- □ We practiced this in the past and need to jumpstart
- □ We should implement this technique



TECHNIQUE 8

CSS POSITIONING STRATEGY

CSS positioning is a technique used to control the layout and positioning of HTML elements on a web page. There are several positioning strategies available in CSS, each serving different purposes. Understanding these strategies is essential for creating well-structured and visually appealing layouts.

- □ We already practice this technique
- □ This technique would not be suitable for our operation
- □ We practiced this in the past and need to jumpstart
- □ We should implement this technique

SUMMARY

- Cascading Style Sheets, commonly known as CSS, is a fundamental technology in web development that enhances the presentation and layout of HTML documents. It provides a powerful mechanism for separating the content and structure of a webpage from its visual design.
- CSS operates on the principle of cascading, where styles are applied in a hierarchical manner, allowing for both global and specific styling. This separation of content and presentation fosters cleaner and more maintainable code.
- Declarations within CSS include properties and values. Properties specify the aspect of an element to be styled, such as color, font size, margin, padding, and more. Values are assigned to properties and determine how the selected elements will appear.
- External stylesheets are a commonly used method to apply CSS to HTML documents. By linking an external stylesheet in the HTML document's head section, the same styles can be applied across multiple pages, promoting consistent branding and design.
- The CSS Box Model is a fundamental concept in web development that defines the properties and dimensions of an HTML element. It represents an element as a rectangular box, encompassing the content, padding, border, and margin.
- Cascading style sheets (CSS) is a standard defined by the World Wide Web Consortium that offers designers more flexibility and accuracy when defining the appearance of text and formats than standard HTML.
- To add CSS styles to your website, you can use three different ways to insert the CSS. You can use an «external stylesheet», an «internal stylesheet», or in «inline style». The benefit for using each depends on what you are doing with the Style.
- Inline styles are styles that are written directly in the tag in the HTML document. Inline styles affect only the specific tag they are applied to.
- Embedded styles are styles that are embedded in the head of the document. Embedded styles affect only the tags on the page they are embedded in.

KNOWLEDGE CHECK

1. The CSS links properties are

a. :link, :visited, :hover, :active

b. :link, :visit, :hover, :active

c. :link, :visited, :over, :active

d. :link, :visited, :hover, :active, :inactive

2. Internet Explorer uses ..... property to create transparent images.

a. -moz-opacity:x
b. filter: alpha(opacity=x)
c. Both of the above
d. None of the above

3. The different ways to associate styles with a HTML document is/are

a. Embedded CSS with <style> element
b. Inline CSS with style attribute.
c. External CSS with <link> element.
d. All of the above

4. Which of the following is correct CSS syntax for using font property?

a. .....
b. .....
c. .....
d. None of the above

5. In CSS tables, the possible values for the caption-side property can have the following values.

a. top, bottom, left or right
b. top, bottom, center, left or right
c. top or bottom
d. left or right

6. The ..... property indicates whether a cell without any content should have a border displayed.

a. blank-cells
b. empty-cells
c. nocontent-cells
d. noborder-cells

7. The ..... specifies whether a border should be solid, dashed line, doted line, double line, groove etc.

a. border-layout
b. border-decoration
c. border-style
d. border-weight

8. What does the CSS property "opacity" control?

a. Element's background color
b. Element's text color
c. Element's visibility and transparency
d. Element's border width

REVIEW QUESTIONS

1. What does CSS stand for? Where in an HTML document is the correct place to refer to an external style sheet?
2. Which HTML tag is used to define an internal style sheet? Mention.
3. What is the correct CSS syntax for making all the elements bold?
4. Which CSS property controls the text size? Explain.
5. How do you insert a comment in a CSS file? Describe.

Check your Result

1. (a) 2. (b) 3. (d) 4. (c) 5. (a)
6. (b) 7. (c) 8. (c)

REFERENCES

1. Cascading Style Sheets Cascading Style Sheets, PhD thesis, by Håkon Wium Lie – provides an authoritative historical reference of CSS
2. Cascading Style Sheets: Designing for the Web (2005) by Håkon Wium Lie and Bert Bos,
3. Dan Cederholm (2009): Web Standards Solutions, The Markup and Style Handbook, Friends of Ed,
4. Jeffrey Zeldman (2009): Designing With Web Standards, New Riders, ISBN 978-0321616951 (paperback) (book's companion site)
5. Keith Schengili-Roberts (2003): Core CSS, 2nd Edition, Prentice Hall, ISBN 0-13-009278-9
6. Kynn Bartlett: Teach Yourself CSS in 24 Hours, 2nd Edition (2006), Sams Publishing, ISBN 978-0672329067
7. Meyer, Eric A. (2006). Cascading Style Sheets: The Definitive Guide, Third Edition. O'Reilly Media, Inc.
8. On the Analysis of Cascading Style Sheets, Pierre Geneves, Nabil Layaida, and Vincent Quint, Proceedings of the 21st International Conference on World Wide Web (WWW'12), pp. 809–818, 2012.
9. The Zen of CSS Design (2005) (co-authored by CSS Zen Garden Owner, Dave Shea, and Molly E. Holzschlag), ISBN 0-321-30347-4

CHAPTER 3

JAVASCRIPT

3.1 BASICS OF JAVASCRIPT

KEY NOTES

Definition

Cascading Style Sheets (CSS) is a vital tool for web developers, allowing them to separate content from design and control the visual presentation of HTML elements. Through the use of selectors, declarations, and specific rule application, CSS enables the creation of consistent and attractive user interfaces across various platforms and devices. Its evolution with CSS3 has brought about advanced features that contribute to the dynamic and interactive nature of today's web applications.

The Cascade Part of CSS

The cascade part of CSS means that more than one stylesheet can be attached to a document, and all of them can influence the presentation.

Benefits of CSS

Using CSS modularizes your websites in much the same way that programmers have been modularizing software code for year. Modular code in your website allows for greater design consistency and easier maintenance. CSS is the new standard and has been now for several years. It's time to separate the structure of HTML and the presentation of CSS.

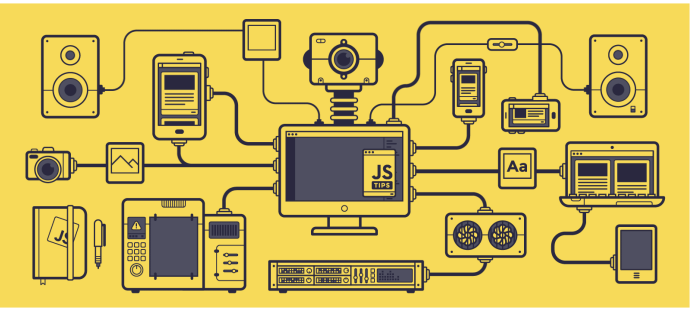
Learning Objectives

- Discuss about CSS versions
- Identify CSS3 modules
- Elaborate the strategy of CSS box model

Definition

JavaScript ("JS" for short) is a full-fledged dynamic programming language that, when applied to an HTML document, can provide dynamic interactivity on websites. It was invented by Brendan Eich, co-founder of the Mozilla project, the Mozilla Foundation, and the Mozilla Corporation.

JavaScript is incredibly versatile. You can start small, with carousels, image galleries, fluctuating layouts, and responses to button clicks. With more experience you'll be able to create games, animated 2D and 3D graphics, comprehensive database-driven apps, and much more!



Application Programming Interface (API) define how software components should interact, enabling developers to access and use functionalities provided by other applications, libraries, or platforms without needing to understand their internal workings.

JavaScript itself is fairly compact yet very flexible. Developers have written a large variety of tools on top of the core JavaScript language, unlocking a vast amount of extra functionality with minimum effort. These include:

- Browser Application Programming Interfaces (APIs) — APIs built into web browsers, providing functionality like dynamically creating HTML and setting CSS styles, collecting and manipulating a video stream from the user's webcam, or generating 3D graphics and audio samples.
- Third-party APIs to allow developers to incorporate functionality in their sites from other content providers, such as Twitter or Facebook.
- Third-party frameworks and libraries you can apply to your HTML to allow you to rapidly build up sites and applications.

The Java Connection

Understandably, JavaScript's connection with Java is regularly misunderstood. They are not the same thing.

- Java, created by Sun Microsystems, is a full computer programming language like C++, suitable for writing complete, large-scale programs.
- JavaScript, on the other hand, was created by » Netscape. It was based to some degree on Java — the syntax of the code is very similar — but it is very rarely used for anything outside of a browser. It was actually originally to be called ‹Live Script›, but Java›s increasing popularity at the time made Netscape change the name for marketing reasons.

Web servers handle incoming requests from clients (usually web browsers) and respond by delivering the requested resources, such as HTML pages, images, videos, and more.

A scripting language can be thought of as a lightweight programming language; one that can be interpreted by a browser without needing to be *compiled* first. The script is actually just some commands that the browser has to do.

The two share many similarities. Most prominent of these is that they are both forms of Object-Oriented Programming, or OOP. This means that you work with small objects that are combined together to form larger objects.



You may have heard of Java being used alongside HTML through things called *applets*. These small-scale applications can be embedded into pages for very advanced effects, but they are not entirely practical, due to large file sizes and limited extra utility. JavaScript on the other hand, can be *very* useful.

Versions

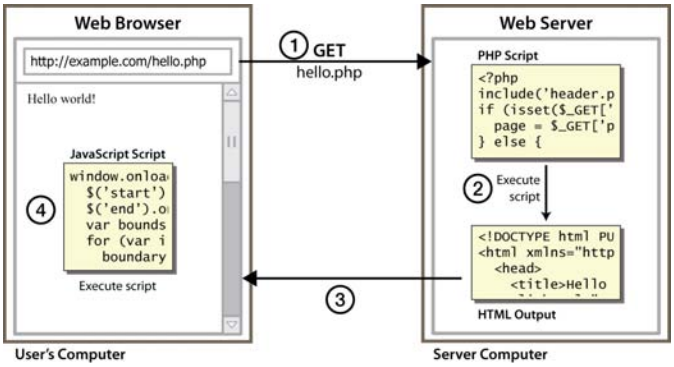
Like seemingly every tool we use to create our sites, JavaScript has been arriving in versions, ever since version 1.0 which turned up in Netscape Navigator 2. In subsequent versions JavaScript graduated through versions 1.1, 1.2 and 1.3. Microsoft made an attempt at JS 1.0 support in Internet Explorer 3, but it was bugged to bits and was very unreliable. Microsoft dubbed their version “JScript”. The latest generation of **browsers** have decent support for JavaScript 1.3.

Coders' dismay at the incompatibilities wrought by the two browsers' different levels of support eventually lead to a standardized version of JavaScript, sometimes called *ECMAScript*, after its standardizers, the ECMA. Modern browsers like IE8, Firefox and Safari have good support for this standard, and a lot of work has gone on recently to ensure that all browsers are operating on the same DOM.

Client-side JavaScript

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.



The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.

The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.

JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

Advantages of JavaScript

The merits of using JavaScript are −

- *Less server interaction*− You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- *Immediate feedback to the visitors*− They do not have to wait for a page reload to see if they have forgotten to enter something.
- *Increased interactivity*− You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- *Richer interfaces*− You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Limitations of JavaScript

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features −

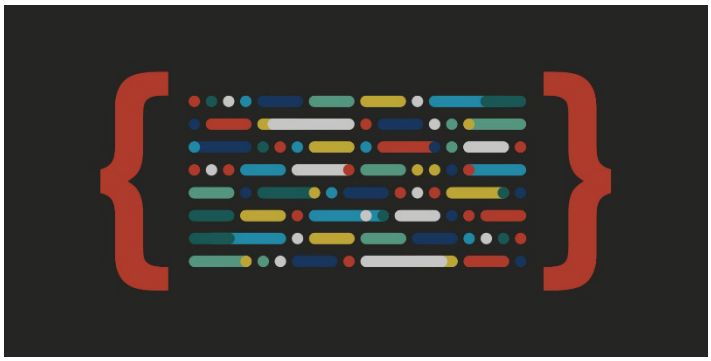
- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript does not have any multithreading or **multiprocessor** capabilities.

Multiprocessor is a computer system with two or more central processing units (CPUs), with each one sharing the common main memory as well as the peripherals.

Once again, JavaScript is a lightweight, interpreted programming language that allows you to build interactivity into otherwise static HTML pages.

JavaScript Development Tools

One of major strengths of JavaScript is that it does not require expensive development tools. You can start with a simple text editor such as Notepad. Since it is an interpreted language inside the context of a web browser, you do not even need to buy a compiler.



To make our life simpler, various vendors have come up with very nice JavaScript editing tools. Some of them are listed here −

- *Microsoft FrontPage* − Microsoft has developed a popular HTML editor called FrontPage. FrontPage also provides web developers with a number of JavaScript tools to assist in the creation of interactive websites.
- *Macromedia Dreamweaver MX* − Macromedia Dreamweaver MX is a very popular HTML and

JavaScript editor in the professional web development crowd. It provides several handy prebuilt JavaScript components, integrates well with databases, and conforms to new standards such as XHTML and XML.

- *Macromedia HomeSite 5* − HomeSite 5 is a well-liked HTML and JavaScript editor from Macromedia that can be used to manage personal websites effectively.

3.2 DOM MANIPULATION AND EVENTS

KEY NOTES

Definition

The Document Object Model (DOM) is a programming interface for HTML documents. Meaning, it provides a way to use code (JavaScript, for example) to access and modify the structure, style, and content of an HTML page.

In the DOM, all HTML elements are defined as objects. Each of these objects has a set of properties and methods. A property is a value that you can get or set (like changing the content of an HTML element). A method is an action you can do (like add or deleting an HTML element).

DOM and JavaScript

A Document object represents the HTML document that is displayed in that window. The Document object has various properties that refer to other objects which allow access to and modification of document content.

The Event Object

The event object contains a number of properties that describe the event that occurred.

Event Phases

Indicates which phase of the event flow is currently being evaluated. The number is represented by 4 constants:

0. NONE
1. CAPTURING_PHASE - The event flow is in capturing phase
2. AT_TARGET - The event flow is in target phase, i.e. it is being evaluated at the event target
3. BUBBLING_PHASE - The event flow is in bubbling phase

Learning Objectives

- Access the DOM
- Explain important data types

Definition

The Document Object Model (DOM) is a programming interface for HTML and XML documents. It represents the page so that programs can change the document structure, style and content. The DOM represents the document as nodes and objects. That way, programming languages can connect to the page.

A Web page is a document. This document can be either displayed in the browser window, or as the HTML source. But it is the same document in both cases. The Document Object Model (DOM) represents that same document so it can be manipulated. The DOM is an object-oriented representation of the web page, which can be modified with a scripting language such as JavaScript.

Scripting languages are commonly used for automating tasks, manipulating data, creating dynamic web content, and more. They are often favored for their ease of use and rapid development capabilities.

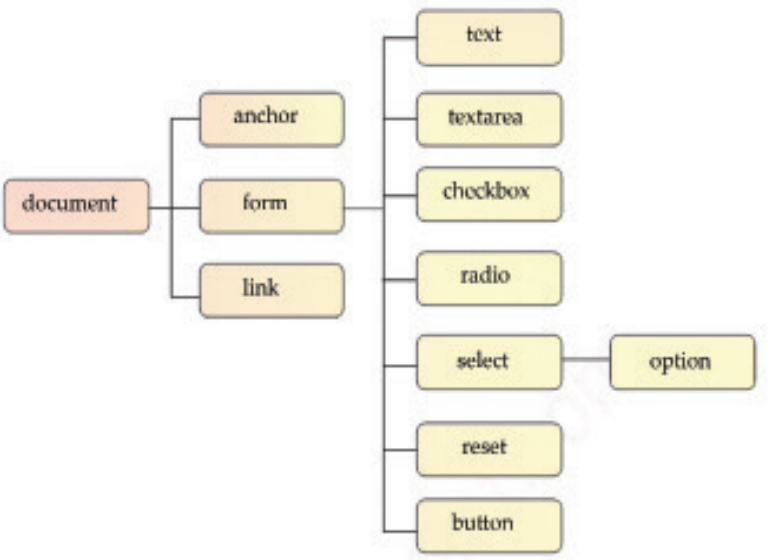


Figure 3.1: Properties of document object.

The W3C DOM and WHATWG DOM standards are implemented in most modern browsers. Many browsers extend the standard, so care must be exercised when using them on the web where documents may be accessed by various browsers with different DOMs.

For example, the standard DOM specifies that the getElementsByTagName method in the code below must return a list of all the <P> elements in the document:

```
var paragraphs = document.getElementsByTagName("P");
// paragraphs[0] is the first <p> element
// paragraphs[1] is the second <p> element, etc.
alert(paragraphs[0].nodeName);
```

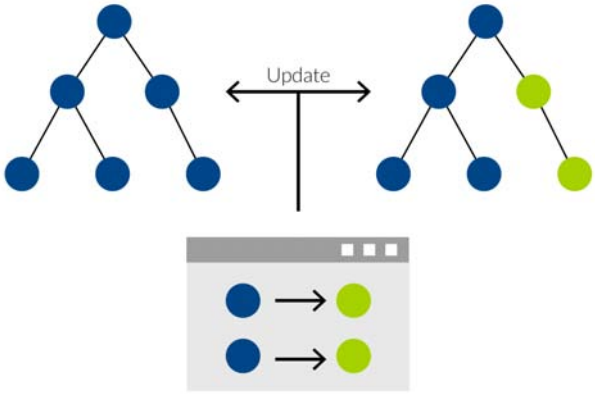
All of the properties, methods, and events available for manipulating and creating web pages are organized into objects (e.g., the document object that represents the document itself, the table object that implements the special HTMLTableElement DOM interface for accessing HTML tables, and so forth). This documentation provides an object-by-object reference to the DOM implemented in Gecko-based browsers.

DOM and JavaScript

The short example above, like nearly all of the examples in this reference, is JavaScript. That is to say, it's *written* in JavaScript, but it *uses* the DOM to access the document and

its elements. The DOM is not a programming language, but without it, the JavaScript language would not have any model or notion of web pages, HTML documents, XML documents, and their component parts (e.g. elements). Every element in a document—the document as a whole, the head, tables within the document, table headers, text within the table cells—is part of the document object model for that document, so they can all be accessed and manipulated using the DOM and a scripting language like JavaScript.

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991.



In the beginning, JavaScript and the DOM were tightly intertwined, but eventually they evolved into separate entities. The page content is stored in the DOM and may be accessed and manipulated via JavaScript, so that we may write this approximative equation:

API (HTML or XML page) = DOM + JS (scripting language)

The DOM was designed to be independent of any particular programming language, making the structural representation of the document available from a single, consistent API. Though we focus exclusively on JavaScript in this reference documentation, implementations of the DOM can be built for any language, as this **Python** example demonstrates:

```
# Python DOM example
import xml.dom.minidom as m
doc = m.parse("C:\\Projects\\Py\\chap1.xml");
doc.nodeName # DOM property of document object;
p_list = doc.getElementsByTagName("para");
```

Accessing the DOM

You do not have to do anything special to begin using the DOM. Different browsers have different implementations of the DOM,

and these implementations exhibit varying degrees of conformance to the actual DOM standard, but every web browser uses some document object model to make web pages accessible to script.

When you create a script–whether it is inline in a <script> element or included in the web page by means of a script loading instruction–you can immediately begin using the API for the document or window elements to manipulate the document itself or to get at the children of that document, which are the various elements in the web page. Your DOM programming may be something as simple as the following, which displays an alert message by using the alert() function from the window object, or it may use more sophisticated DOM methods to actually create new content, as in the longer example below.

```
<body onload="window.alert('Welcome to my home page!')
```

Aside from the <script> element in which the JavaScript is defined, this JavaScript sets a function to run when the document is loaded (and when the whole DOM is available for use). This function creates a new H1 element, adds text to that element, and then adds the H1 to the tree for this document:

```
<html>
  <head>
    <script>
      // run this function when the document is loaded
      window.onload = function() {

        // create a couple of elements in an otherwise empty HTML page
        var heading = document.createElement("h1");
        var heading_text = document.createTextNode("Big Head!");
        heading.appendChild(heading_text);
        document.body.appendChild(heading);
      }
    </script>
  </head>
  <body>
  </body>
</html>
```

Important Data Types

This reference tries to describe the various objects and types in as simple a way as possible. But there are a number of different data types being passed around the API that you should be aware of. For the sake of simplicity, syntax examples in this API reference typically refer to nodes as elements, to arrays of nodes as nodeLists (or simply elements), and to attribute nodes simply as attributes.

The following table briefly describes these data types.

document	When a member returns an object of type document (e.g., the **ownerDocument** property of an element returns the document to which it belongs), this object is the root document object itself.

element	element refers to an element or a node of type element returned by a member of the DOM API. Rather than saying, for example, that the document.createElement() method returns an object reference to a node, we just say that this method returns the element that has just been created in the DOM. element objects implement the DOM Element interface and also the more basic Node interface, both of which are included together in this reference.
nodeList	A nodeList is an array of elements, like the kind that is returned by the method document.getElementsByTagName(). Items in a nodeListare accessed by index in either of two ways: • list.item(1) • list[1] These two are equivalent. In the first, **item()** is the single method on the nodeList object. The latter uses the typical array syntax to fetch the second item in the list.
attribute	When an attribute is returned by a member (e.g., by the **createAttribute()** method), it is an object reference that exposes a special (albeit small) interface for attributes. Attributes are nodes in the DOM just like elements are, though you may rarely use them as such.
namedNodeMap	A namedNodeMap is like an array, but the items are accessed by name or index, though this latter case is merely a convenience for enumeration, as they are in no particular order in the list. A namedNodeMap has an item() method for this purpose, and you can also add and remove items from a namedNodeMap.

DOM Interfaces

This guide is about the objects and the actual *things* you can use to manipulate the DOM hierarchy. There are many points where understanding how these work can be confusing. For example, the object representing the HTML form element gets its **name** property from the HTMLFormElement interface but its **className** property from the HTMLElementinterface. In both cases, the property you want is simply in that form object.

But the relationship between objects and the interfaces that they implement in the DOM can be confusing, and so this section attempts to say a little something about the actual **interfaces** in the DOM specification and how they are made available.

Interfaces and Objects

Many objects borrow from several different interfaces. The table object, for example, implements a specialized HTML Table Element Interface, which includes such

methods as createCaption and insertRow. But since it's also an HTML element, table implements the Element interface described in the DOM element. And finally, since an HTML element is also, as far as the DOM is concerned, a node in the tree of nodes that make up the object model for an HTML or XML page, the table element also implements the more basic Node interface, from which Element derives.

When you get a reference to a table object, as in the following example, you routinely use all three of these interfaces interchangeably on the object, perhaps without knowing it.

```
var table = document.getElementById("table");
var tableAttrs = table.attributes; // Node/Element interface
for (var i = 0; i < tableAttrs.length; i++) {
  // HTMLTableElement interface: border attribute
  if(tableAttrs[i].nodeName.toLowerCase() == "border")
    table.border = "1";
}
// HTMLTableElement interface: summary attribute
table.summary = "note: increased border";
```

Core Interfaces in the DOM

This section lists some of the most commonly-used interfaces in the DOM. The idea is not to describe what these APIs do here but to give you an idea of the sorts of methods and properties you will see very often as you use the DOM.

Document and window objects are the objects whose interfaces you generally use most often in DOM programming. In simple terms, the window object represents something like the browser, and the document object is the root of the document itself. Element inherits from the generic Node interface, and together these two interfaces provide many of the methods and properties you use on individual elements. These elements may also have specific interfaces for dealing with the kind of data those elements hold, as in the tableobject example in the previous section.

The following is a brief list of common APIs in web and XML page scripting using the DOM.

- document.getElementById(id)
- document.getElementsByTagName(name)
- document.createElement(name)
- parentNode.appendChild(node)
- element.innerHTML
- element.style.left
- element.setAttribute
- element.getAttribute

- element.addEventListener
- window.content
- window.onload
- window.dump
- window.scrollTo

Testing the DOM API

This document provides samples for every interface that you can use in your own web development. In some cases, the samples are complete HTML pages, with the DOM access in a <script> element, the interface (e.g, buttons) necessary to fire up the script in a form, and the HTML elements upon which the DOM operates listed as well. When this is the case, you can cut and paste the example into a new HTML document, save it, and run the example from the browser.

HTML is the standard markup language for creating Web pages. HTML stands for Hyper Text Markup Language.

There are some cases, however, when the examples are more concise. To run examples that only demonstrate the basic relationship of the interface to the HTML elements, you may want to set up a test page in which interfaces can be easily accessed from scripts. The following very simple web page provides a <script> element in the header in which you can place functions that test the interface, a few **HTML** elements with attributes that you can retrieve, set, or otherwise manipulate, and the web user interface necessary to call those functions from the browser. You can use this test page or create a similar one to test the DOM interfaces you are interested in and see how they work on the browser platform. You can update the contents of the test() function as needed, create more buttons, or add elements as necessary.

```
<html>
  <head>
    <title>DOM Tests</title>
    <script type="application/javascript">
    function setBodyAttr(attr, value){
      if (document.body) eval('document.body.'+attr+'="'+value+'"');
      else notSupported();
    }
    </script>
  </head>
  <body>
    <div style="margin: .5in; height: 400;">
      <p><b><tt>text</tt></b></p>
      <form>
        <select onChange="setBodyAttr('text',
        this.options[this.selectedIndex].value);">
```

```
          <option value="black">black
          <option value="darkblue">darkblue
        </select>
        <p><b><tt>bgColor</tt></b></p>
        <select onChange="setBodyAttr('bgColor',
        this.options[this.selectedIndex].value);">
          <option value="white">white
          <option value="lightgrey">gray
        </select>
        <p><b><tt>link</tt></b></p>
        <select onChange="setBodyAttr('link',
        this.options[this.selectedIndex].value);">
          <option value="blue">blue
          <option value="green">green
        </select>  <small>
        <a href="http://www.brownhen.com/dom_api_top.html" id="sample">
         (sample link)</a></small><br>
      </form>
      <form>
        <input type="button" value="version" onclick="ver()" />
      </form>
    </div>
  </body>
</html>
```

To test a lot of interfaces in a single page-for example, a "suite" of properties that affect the colors of a web page-you can create a similar test page with a whole console of buttons, textfields, and other HTML elements. The following screenshot gives you some idea of how interfaces can be grouped together for testing.



Figure 3.2: Sample DOM Test Page.

In this example, the dropdown menus dynamically update such DOM-accessible aspects of the web page as its background color (bgColor), the color of the hyperlinks (aLink), and color of the text (text). However you design your test pages, testing the interfaces as you read about them is an important part of learning how to use the DOM effectively.

The Event Object

The event object is created when the event first happens; it travels with the event on its journey through the DOM. The function that we assign as a callback to an event listener is passed the event object as its first argument. We can use this object to access a wealth of information about the event that has occurred:

- type (string): This is the name of the event.
- target (node): This is the DOM node where the event originated.
- currentTarget (node): This is the DOM node that the event callback is currently firing on.
- bubbles (boolean): This indicates whether this is a "bubbling" event.
- preventDefault (function): This prevents any default behavior from occurring that the user agent (i.e. browser) might carry out in relation to the event (for example, preventing a click event on an <a> element from loading a new page).
- stopPropagation (function): This prevents any callbacks from being fired on any nodes further along the event chain, but it does not prevent any additional callbacks of the same event name from being fired on the current node.
- stopImmediatePropagation (function): This prevents any callbacks from being fired on any nodes further along the event chain, including any additional callbacks of the same event name on the current node.
- cancelable (boolean): This indicates whether the default behavior of this event can be prevented by calling the event.preventDefault method.
- defaultPrevented (boolean): This states whether the preventDefault method has been called on the event object.
- isTrusted (boolean): An event is said to be "trusted" when it originates from the device itself, not synthesized from within JavaScript.
- eventPhase (number): This number represents the phase that the event is currently in: none (0), capture (1), target (2) or bubbling (3). We'll go over event phases next.
- timestamp (number): This is the date on which the event occurred.

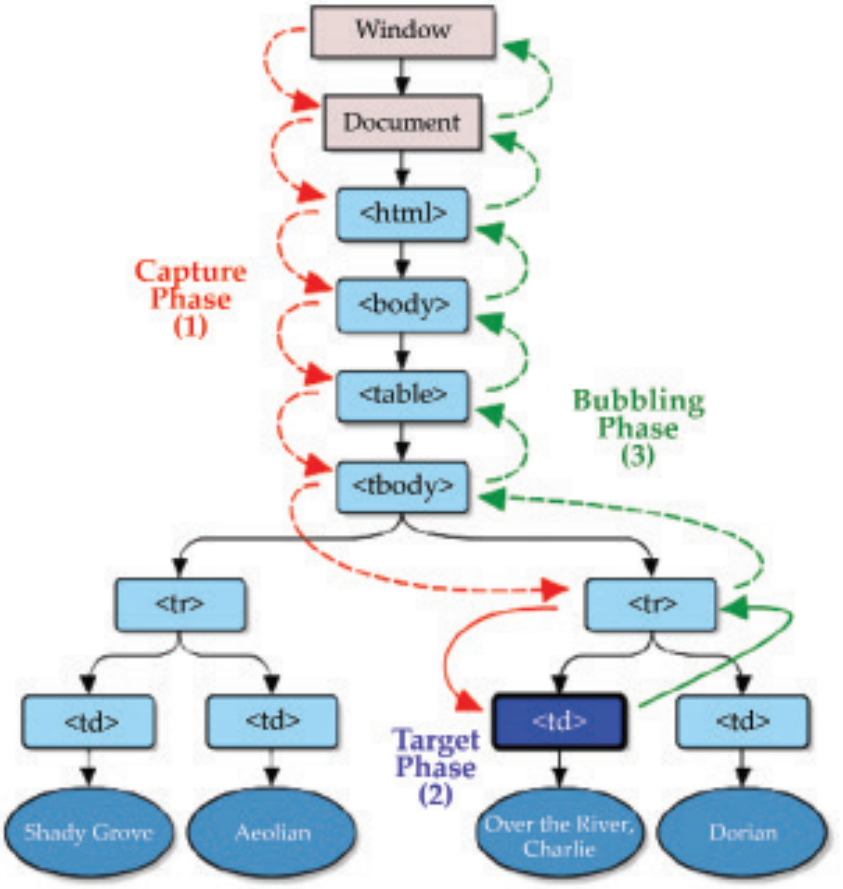
Many other properties can be found on the event object, but they are specific to the type of event in question. For example, mouse events will include clientX and clientY properties on the event object to indicate the location of the pointer in the viewport.

It is best to use your favorite browser's **debugger** or a console.log to look more closely at the event object and its properties.

Event Phases

When a DOM event fires in your app, it does not just fire once where the event originated; it embarks on a journey of three phases. In short, the event flows from the

document's root to the target (i.e. capture phase), then fires on the event target (target phase), then flows back to the document's root (bubbling phase).



Capture Phase

The first phase is the capture phase. The event starts its journey at the root of the document, working its way down through each layer of the DOM, firing on each node until it reaches the event target. The job of the capture phase is to build the propagation path, which the event will travel back through in the bubbling phase.

As mentioned, you can listen to events in the capture phase by setting the third argument of addEventListener to true. You could potentially prevent any clicks from firing in a certain element if the event is handled in the capture phase.

```
var form = document.querySelector('form');

form.addEventListener('click', function(event) {
  event.stopPropagation();
}, true); // Note: 'true'
```

If you are unsure, listen for events in the bubbling phase by setting the useCapture flag to false or undefined.

Target Phase

An event reaching the target is known as the target phase. The event fires on the target node, before reversing and retracing its steps, propagating back to the outermost document level.

In the case of nested elements, mouse and pointer events are always targeted at the most deeply nested element. If you have listened for a click event on a element, and the user actually clicks on a <p> element in the div, then the <p> element will become the event target. The fact that events "bubble" means that you are able to listen for clicks on the (or any other ancestor node) and still receive a callback once the event passes through.

Bubbling Phase

After an event has fired on the target, it does not stop there. It bubbles up (or propagates) through the DOM until it reaches the document's root. This means that the same event is fired on the target's parent node, followed by the parent's parent, continuing until there is no parent to pass the event onto.

Think of the DOM as an onion and the event target as the core of the onion. In the capture phase, the event drills into the onion through each layer. When the event reaches the core, it fires (the target phase), and then reverses, working its way back up through each layer (the propagation phase). Once the event has returned to the surface, its journey is over.

Bubbling is useful. It frees us from listening for an event on the exact element it came from; instead, we listen on an element further up the DOM tree, waiting for the event to reach us. If events did not bubble, we would have to, in some cases, listen for an event on many different elements to ensure that it is caught.

The majority of, but not all, events bubble. When events do not bubble, it is usually for a good reason.

Stopping Propagation

Interrupting the path of the event at any point on its journey (i.e. in the capture or bubbling phase) is possible simply by calling the stopPropagation method on the event object. Then, the event will no longer call any listeners on nodes that it travels through on its way to the target and back to the document.

```
child.addEventListener('click', function(event) {
 event.stopPropagation();
});

parent.addEventListener('click', function(event) {
 // If the child element is clicked
 // this callback will not fire
});
```

Calling event.stopPropagation() will not prevent any additional event listeners from being called on the current target if multiple listeners for the same event exist. If you wish to prevent any additional listeners from being called on the current node, you can use the more aggressive event.stopImmediatePropagation() method.

```
child.addEventListener('click', function(event) {
 event.stopImmediatePropagation();
});

child.addEventListener('click', function(event) {
 // If the child element is clicked
 // this callback will not fire
});
```

Prevent The Browser's Default Behavior

The browser has default behaviors that will respond when certain events occur in the document. The most common event is a link being clicked. When a click event occurs on an <a> element, it will bubble up to the document level of the DOM, and the browser will interpret the href attribute and reload the window at the new address.

In Web applications, developers usually want to manage the navigation themselves, without causing the page to refresh. To do this, we need to prevent the browser's default response to clicks and instead do our own thing. To do this, we call event.preventDefault().

```
anchor.addEventListener('click', function(event) {
  event.preventDefault();
  // Do our own thing
});
```

We can prevent many other default behaviors in the browser. For example, we could prevent presses of the space bar from scrolling the page in an HTML5 game, or we could prevent clicks from selecting text.

Calling event.stopPropagation() here will only prevent callbacks that are attached

further down the propagation chain from being fired. It will not prevent the browser from doing its thing.

Custom DOM Events

The browser is not the only thing that is able to trigger DOM events. We can create our own custom events and dispatch them on any element in the document. This type of event would behave just the same as a regular DOM event.

```
var myEvent = new CustomEvent("myevent", {
  detail: {
    name: "Wilson"
  },
  bubbles: true,
  cancelable: false
});

// Listen for 'myevent' on an element
myElement.addEventListener('myevent', function(event) {
  alert('Hello ' + event.detail.name);
});

// Trigger the 'myevent'
myElement.dispatchEvent(myEvent);
```

Event listener is a programming construct used in web development to listen for and respond to specific events that occur in a web page or application.

Synthesizing "untrusted" DOM events on elements (for example, click) to simulate user interaction is also possible. This can be useful when testing DOM-related libraries.

Note the following:

- The CustomEvent API is not available in IE 8 and below.
- The Flight framework from Twitter makes use of custom events to communicate between modules. This enforces a highly decoupled, modular architecture.

Delegate Event Listeners

Delegate event listeners are a more convenient and performant way to listen for events on a large number of DOM nodes using a single event listener. For example, if a list contains 100 items that all need to respond to a click event in a similar way, then we could query the DOM for all of the list items and attach an event listener to each one. This would result in 100 separate event listeners. Whenever a new item is added to the list, the click event listener would have to be added to it. Not only does this risk getting expensive, but it is tricky to maintain.

Delegate event listeners can make our lives a lot easier. Instead of listening for the click event on each element, we listen for it on the parent <ul> element. When an <li> is clicked, then the event bubbles up to the <ul>, triggering the callback. We can identify which <li> element has been clicked by inspecting the event.target. Below is a crude example to illustrate:

```
var list = document.querySelector('ul');

list.addEventListener('click', function(event)
  var target = event.target;

  while (target.tagName !== 'LI') {
    target = target.parentNode;
    if (target === list) return;
  }

  // Do stuff here
```

This is better because we have only the overhead of a single event listener, and we no longer have to worry about attaching a new event listener when an item is added to the list. The concept is pretty simple but super-useful.

Use an event delegate JavaScript library, such as FT Lab's ftdomdelegate. If you are using jQuery, you can seamlessly use event delegation by passing a selector as the second parameter to the .on() method.

```
// Not using event delegation
$('li').on('click', function(){});

// Using event delegation
$('ul').on('click', 'li', function(){});
```

Useful Events

LOAD

The load event fires on any resource that has finished loading (including any dependent resources). This could be an image, style sheet, script, video, audio file, document or window.

```
image.addEventListener('load', function(event) {
  image.classList.add('has-loaded');
});
```

ONBEFOREUNLOAD

window.onbeforeunload enables developers to ask the user to confirm that they want to leave the page. This can be useful in applications that require the user to save changes that would get lost if the browser's tab were to be accidentally closed.

```
window.onbeforeunload = function() {
  if (textarea.value != textarea.defaultValue) {
    return 'Do you want to leave the page and discard changes?';
  }
};
```

Note that assigning an onbeforeunload handler prevents the browser from caching the page, thus making return visits a lot slower. Also, onbeforeunload handlers must be synchronous.

STOPPING WINDOW BOUNCE IN MOBILE SAFARI

At the Financial Times, we use a simple event.preventDefaulttechnique to prevent mobile Safari from bouncing the window when it is scrolled.

```
document.body.addEventListener('touchmove', function(event) {
 event.preventDefault();
});
```

Be warned that this will also prevent any native scrolling from working (such as overflow: scroll). To allow native scrolling on a subset of elements that need it, we listen for the same event on the scrollable element and set a flag on the event object. In the callback at the document level, we decide whether to prevent the default behavior of the touch event based on the existence of the isScrollable flag.

```
// Lower down in the DOM we set a flag
scrollableElement.addEventListener('touchmove', function(event) {
 event.isScrollable = true;
});

// Higher up the DOM we check for this flag to decide
// whether to let the browser handle the scroll
document.addEventListener('touchmove', function(event) {
 if (!event.isScrollable) event.preventDefault();
});
```

Manipulating the event object is not possible in IE 8 and below. As a workaround, you can set properties on the event.target node.

RESIZE

Listening to the resize event on the window object is super-useful for complex responsive layouts. Achieving a layout with CSS alone is not always possible. Sometimes JavaScript has to help us calculate and set the size of elements. When the window is resized or the device's orientation changes, then we would likely need to readjust these sizes.

```
window.addEventListener('resize', function() {
  // update the layout
});
```

It is recommended to use a debounced callback to normalize the callback rate and prevent extreme thrashing in the layout.

TRANSITIONEND

Today we use CSS to power the majority of transitions and animations in our applications. Sometimes, though, we still need to know when a particular animation has finished.

```
el.addEventListener('transitionEnd', function() {
 // Do stuff
});
```

Note the following:

- If you are using @keyframe animations, use the animationEndevent name, instead of transitionEnd.
- Like a lot of events, transitionEnd bubbles. Remember either to call event. stopPropagation() on any descendant transition events or to check the event. target to prevent callback logic from running when it's not supposed to.
- Event names are still widely vendor-prefixed (for example, webkitTransitionEnd, msTransitionEnd, etc). Use a library such as Modernizr to get the event name's correct prefix.

ANIMATIONITERATION

The animationiteration event will fire every time a currently animating element completes an iteration. This is useful if we want to stop an animation but not midway through.

```
function start() {
  div.classList.add('spin');
}

function stop() {
  div.addEventListener('animationiteration', callback);

  function callback() {
    div.classList.remove('spin');
    div.removeEventListener('animationiteration', callback);
  }
}
```

ERROR

If an error occurs when a resource loads, we might want to do something about it, especially if our users are on a flaky connection. The Financial Times uses this event to detect any images that might have failed to load in an article and instantly hide them. Because the "DOM Level 3 Events" specification has redefined the error event to "not bubble," we can handle the event in one of two ways.

```
imageNode.addEventListener('error', function(event) {
  image.style.display = 'none';
});
```

Unfortunately, addEventListener does not address all use cases.

3.3 CONDITIONALS AND VARIABLE

Key Notes

Fundamental

Referring to an action that takes place only if a specific condition is met. Conditional expressions are one of the most important components of programming languages because they enable a program to act differently each time it is executed, depending on the input. Most programming languages use the word if for conditional expressions.

Conditional statements let your Puppet code behave differently in different situations. They are most helpful when combined with facts or with data retrieved from an external source.

Conditionals

Conditional statement is a set of rules performed if a certain condition is met. Below are some examples of conditional statements in programming.

Variables

Variable substitutions are a flexible way to adjust configuration based on your variables and the context of your deployment. You can often tame the number and complexity of your variables by breaking them down into simple variables and combining them together using expressions.

Learning Objectives

- Understand JavaScript variable scope
- Discuss about JavaScript variable names

Fundamental

Conditionals and variables are essential concepts in JavaScript, a versatile and widely used programming language. They empower developers to create dynamic and responsive applications by making decisions based on data and storing information for later use.

Conditionals are programming structures that allow developers to execute different blocks of code based on certain conditions. The most common type of conditional statement is the "if statement." It evaluates a given condition and, if the condition is true, executes a specified block of code. For example:

```
if (temperature > 30) {
  console.log("It's hot outside!");
} else {
  console.log("It's not too hot.");
}
```

In this code snippet, if the variable temperature is greater than 30, the first block of code will be executed, printing "It's hot outside!" Otherwise, the second block will be executed, printing "It's not too hot."

Operands are the objects that are manipulated and operators are the symbols that represent specific actions.

Developers can also use "else if" to handle multiple conditions and the "switch" statement for more complex branching logic.

On the other hand variables are used to store and manage data in JavaScript. They provide a way to label and reference values, making it easier to work with information throughout a program. Variables can hold various types of data, such as numbers, strings, booleans, and objects. To declare a variable, you use the var, let, or const keyword:

```
let name = "Alice";
const age = 30;
var isStudent = true;
```

In this example, name, age, and isStudent are variables that hold a string, number, and boolean value, respectively.

Variables can be updated and manipulated as needed:

```
let count = 0;
count = count + 1;
```

Additionally, variables can be combined with conditionals to create more dynamic behavior:

```
if (score >= passingScore) {
  console.log("Congratulations! You passed.");
} else {
  console.log("Sorry, you didn't pass.");
}
```

Here, the variable score is compared with passingScore using an "if" statement to determine whether the user has passed or not.

Conditionals and variables are fundamental concepts in JavaScript that enable developers to create flexible and interactive applications. Conditionals allow for decision-making based on data conditions, while variables facilitate the storage and manipulation of data, enhancing the dynamic nature of JavaScript programs. Understanding and effectively using conditionals and variables is crucial for building functional and responsive web applications.

Conditionals

The conditional (ternary) operator is the only JavaScript operator that takes three **operands**. This operator is frequently used as a shortcut for the if statement.

Operating system (OS) manages computer hardware resources, provides services to software applications, and allows users to interact with and control the computer system.

Often the result of a play may depend on the value of a variable, fact (something learned about the remote system), or previous task result. In some cases, the values of variables may depend on other variables. Further, additional groups can be created to manage hosts based on whether the hosts match other criteria.

The When Statement

Sometimes you will want to skip a particular step on a particular host. This could be something as simple as not installing a certain package if the operating system is a particular version, or it could be something like performing some cleanup steps if a filesystem is getting full. This is easy to do with the *when* clause, which contains a raw Jinja2 expression without double curly braces. It's actually pretty simple:

```
tasks:
  - name: "shut down Debian flavored systems"
    command: /sbin/shutdown -t now
    when: ansible_os_family == "Debian"
    # note that Ansible facts and vars like ansible_os_family can be used
    # directly in conditionals without double curly braces
```

You can also use parentheses to group conditions:

```
tasks:
  - name: "shut down CentOS 6 and Debian 7 systems"
    command: /sbin/shutdown -t now
    when: (ansible_distribution == "CentOS" and ansible_distribution_major_version == "6") or
          (ansible_distribution == "Debian" and ansible_distribution_major_version == "7")
```

Multiple conditions that all need to be true (a logical 'and') can also be specified as a list:

```
tasks:
  - name: "shut down CentOS 6 systems"
    command: /sbin/shutdown -t now
    when:
      - ansible_distribution == "CentOS"
      - ansible_distribution_major_version == "6"
```

A number of Jinja2 "filters" can also be used in when statements, some of which are unique and provided. Suppose we want to ignore the error of one statement and then decide to do something conditionally based on success or failure:

```
tasks:
  - command: /bin/false
    register: result
    ignore_errors: True

  - command: /bin/something
    when: result|failed

  # In older versions of ansible use |success, now both are valid but succeeded uses the correct tense.
  - command: /bin/something_else
    when: result|succeeded

  - command: /bin/still/something_else
    when: result|skipped
```

Note that was a little bit of foreshadowing on the 'register' statement.

As a reminder, to see what facts are available on a particular system, you can do:

```
ansible hostname.example.com -m setup
```

Tip: Sometimes you'll get back a variable that's a string and you'll want to do a math operation comparison on it. You can do this like so:

```
tasks:
  - shell: echo "only on Red Hat 6, derivatives, and later"
    when: ansible_os_family == "RedHat" and ansible_lsb.major_release|int >= 6
```

Variables defined in the playbooks or inventory can also be used. An example may be the execution of a task based on a variable's boolean value:

```
vars:
  epic: true
```

Then a conditional execution might look like:

```
tasks:
    - shell: echo "This certainly is epic!"
      when: epic
```

or:

```
tasks:
    - shell: echo "This certainly isn't epic!"
      when: not epic
```

If a required variable has not been set, you can skip or fail using Jinja2's *defined* test. For example:

```
tasks:
    - shell: echo "I've got '{{ foo }}' and am not afraid to use it!"
      when: foo is defined

    - fail: msg="Bailing out. this play requires 'bar'"
      when: bar is undefined
```

This is especially useful in combination with the conditional import of vars files (see below). As the examples show, you do not need to use *{{ }}* to use variables inside conditionals, as these are already implied.

Loops and Conditionals

Combining *when* with *with_items* (see Loops), be aware that the *when* statement is processed separately for each item. This is by design:

Conditional statement, often referred to as a "conditional" or "if statement," is a fundamental programming construct used to make decisions and control the flow of a program based on certain conditions.

```
tasks:
    - command: echo {{ item }}
      with_items: [ 0, 2, 4, 6, 8, 10 ]
      when: item > 5
```

If you need to skip the whole task depending on the loop variable being defined, used the *|default* filter to provide an empty iterator:

```
- command: echo {{ item }}
  with_items: "{{ mylist|default([]) }}"
  when: item > 5
```

If using *with_dict* which does not take a list:

```
- command: echo {{ item.key }}
  with_dict: "{{ mydict|default({}) }}"
  when: item.value > 5
```

Loading in Custom Facts

It's also easy to provide your own facts if you want. To run them, just make a call to your own custom fact gathering module at the top of your list of tasks, and variables returned there will be accessible to future tasks:

```
tasks:
    - name: gather site specific fact data
      action: site_facts
    - command: /usr/bin/thingy
      when: my_custom_fact_just_retrieved_from_the_remote_system == '1234'
```

Applying 'when' to roles and includes

Note that if you have several tasks that all share the same conditional statement, you can affix the conditional to a task include statement as below. All the tasks get evaluated, but the conditional is applied to each and every task:

```
- include: tasks/sometasks.yml
  when: "'reticulating splines' in output"
```

Note:

In versions prior to 2.0 this worked with task includes but not playbook includes. 2.0 allows it to work with both.

Or with a role:

```
- hosts: webservers
  roles:
     - { role: debian_stock_config, when: ansible_os_family == 'Debian' }
```

You will note a lot of 'skipped' output by default when using this approach on systems that do not match the criteria. Read up on the 'group_by' module in the About Modules docs for a more streamlined way to accomplish the same thing.

Conditional Imports

Sometimes you will want to do certain things differently in a playbook based on certain criteria. Having one playbook that works on multiple platforms and OS versions is a good example.

As an example, the name of the Apache package may be different between CentOS and Debian, but it is easily handled with a minimum of syntax:

```
---
- hosts: all
  remote_user: root
  vars_files:
    - "vars/common.yml"
    - [ "vars/{{ ansible_os_family }}.yml", "vars/os_defaults.yml" ]
  tasks:
  - name: make sure apache is started
    service: name={{ apache }} state=started
```

As a reminder, the various YAML files contain just keys and values:

```
---
# for vars/CentOS.yml
apache: httpd
somethingelse: 42
```

How does this work? If the operating system was 'CentOS', the first file would try to import would be 'vars/CentOS.yml', followed by '/vars/os_defaults.yml' if that file did not exist. If no files in the list were found, an error would be raised. On Debian, it would instead first look towards 'vars/Debian.yml' instead of 'vars/CentOS.yml', before falling back on 'vars/os_defaults.yml'. Pretty simple.

To use this conditional import feature, you'll need facter or ohai installed prior to running the playbook, but you can of course push this out if you like:

```
# for facter
ansible -m yum -a "pkg=facter state=present"
ansible -m yum -a "pkg=ruby-json state=present"

# for ohai
ansible -m yum -a "pkg=ohai state=present"
```

The approach to configuration – separating variables from tasks, keeps your playbooks from turning into arbitrary code with ugly nested ifs, conditionals, and so on - and results in more streamlined & auditable configuration rules – especially because there are a minimum of decision points to track.

Command is a directive to a computer program acting as an interpreter of some kind, in order to perform a specific task.

Selecting Files and Templates Based On Variables

Sometimes a configuration file you want to copy, or a template you will use may depend on a variable. The following construct selects the first available file appropriate for the variables of a given host, which is often much cleaner than putting a lot of if conditionals in a template.

The following example shows how to template out a configuration file that was very different between, say, CentOS and Debian:

```
- name: template a file
  template: src={{ item }} dest=/etc/myapp/foo.conf
  with_first_found:
    - files:
       - {{ ansible_distribution }}.conf
       - default.conf
      paths:
       - search_location_one/somedir/
       - /opt/other_location/somedir/
```

Register Variables

Often in a playbook it may be useful to store the result of a given **command** in a variable and access it later. Use of the command module in this way can in many ways eliminate the need to write site specific facts, for instance, you could test for the existence of a particular program.

The 'register' keyword decides what variable to save a result in. The resulting variables can be used in templates, action lines, or *when* statements. It looks like this (in an obviously trivial example):

```
- name: test play
  hosts: all

  tasks:

      - shell: cat /etc/motd
        register: motd_contents

      - shell: echo "motd contains the word hi"
        when: motd_contents.stdout.find('hi') != -1
```

As shown previously, the registered variable's string contents are accessible with the 'stdout' value. The registered result can be used in the "with_items" of a task if it is converted into a list (or already is a list) as shown below. "stdout_lines" is already available on the object as well though you could also call "home_dirs.stdout.split()" if you wanted, and could split by other fields:

```
- name: registered variable usage as a with_items list
  hosts: all

  tasks:

      - name: retrieve the list of home directories
        command: ls /home
        register: home_dirs

      - name: add home dirs to the backup spooler
        file: path=/mnt/bkspool/{{ item }} src=/home/{{ item }} state=link
        with_items: "{{ home_dirs.stdout_lines }}"
        # same as with_items: "{{ home_dirs.stdout.split() }}"
```

As shown previously, the registered variable's string contents are accessible with the 'stdout' value. You may check the registered variable's string contents for emptiness:

```
- name: check registered variable for emptiness
  hosts: all

  tasks:

      - name: list contents of directory
        command: ls mydir
        register: contents

      - name: check contents for emptiness
        debug: msg="Directory is empty"
        when: contents.stdout == ""
```

Variables

While automation exists to make it easier to make things repeatable, all of your systems are likely not exactly alike.

On some systems you may want to set some behavior or configuration that is slightly different from others.

Also, some of the observed behavior or state of remote systems might need to influence how you configure those systems. (Such as you might need to find out the IP address of a system and even use it as a configuration value on another system).

You might have some templates for configuration files that are mostly the same, but slightly different based on those variables. Variables are how we deal with differences between systems.

Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container.

Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the **var** keyword as follows.

```
<script type="text/javascript">
   <!--
      var money;
      var name;
   //-->
</script>
```

You can also declare multiple variables with the same **var** keyword as follows −

```
<script type="text/javascript">
   <!--
      var money, name;
   //-->
</script>
```

Storing a value in a variable is called variable initialization. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

For instance, you might create a variable named money and assign the value 2000.50 to it later. For another variable, you can assign a value at the time of initialization as follows.

```
<script type="text/javascript">
   <!--
      var name = "Ali";
      var money;
      money = 2000.50;
   //-->
</script>
```

Note − Use the var keyword only for declaration or initialization, once for the life of any variable name in a document. You should not re-declare same variable twice.

JavaScript is untyped language. This means that a JavaScript variable can hold a value of any data type. Unlike many other languages, you do not have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

JavaScript Variable Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

- *Global Variables* − A global variable has global scope which means it can be defined anywhere in your JavaScript code.
- *Local Variables* − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Within the body of a function, a local variable takes precedence over a global variable with the same name. If you declare a local variable or function parameter with the same name as a global variable, you effectively hide the global variable. Take a look into the following example.

```
<html>
   <body onload = checkscope();>
      <script type = "text/javascript">
         <!--
            var myVar = "global"; // Declare a global variable
            function checkscope( ) {
               var myVar = "local";  // Declare a local variable
               document.write(myVar);
            }
         //-->
      </script>
   </body>
</html>
```

This produces the following result −

```
local
```

JavaScript Variable Names

While naming your variables in JavaScript, keep the following rules in mind.

- You should not use any of the JavaScript reserved keywords as a variable name. These keywords are mentioned in the next section. For example, break or boolean variable names are not valid.
- JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or an underscore character. For example, 123test is an invalid variable name but _123test is a valid one.
- JavaScript variable names are case-sensitive. For example, Name and name are two different variables.

JavaScript Reserved Words

A list of all the reserved words in JavaScript are given in the following table. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

3.4 PAIR PROGRAMMING

KEY NOTES

Concept

Pair programming is an Agile technique originating from Extreme programming (XP) in which two developers team together and work on one computer. The two people work together to design, code and test user stories. Ideally, the two people would be equally skilled and would each have equal time at the keyboard.

Pair programming uses the four eyes principal, ensuring that two sets of eyes review the code that is being produced even when there is a division of labor. While one person writes test cases using test-driven development techniques, for example, the other person may write code to get the tests to pass. Typically, the two developers will turns coding or reviewing, checking each other's work as they go.

Origins

People have started pair programming long before it is called so. Fred Brooks announced "Fellow graduate student Bill Wright and I first tried pair programming when I was a grad student (1953-1956). We produced 1500 lines of defect-free code; it ran correctly first try." In the early 1980s, Larry Constantine reported observing "Dynamic Duos", producing code fater and more bug free. In 1998, Temple University professor John Nosek was the first to run an empirical study on the efficacy of pair programmers. In the late 1990s/early 2000s, pair programming practice is brought to forefront.

Pairing Mechanics

With pair programing, one of the pair, called the driver, types at the computer or writes down a design. The other partner, called the navigator, has many jobs. One of these is to observe the work of the driver—looking for tactical and strategic defects in the driver's work.

Learning Objectives

- Understand pairing strategies
- Identify expected benefits

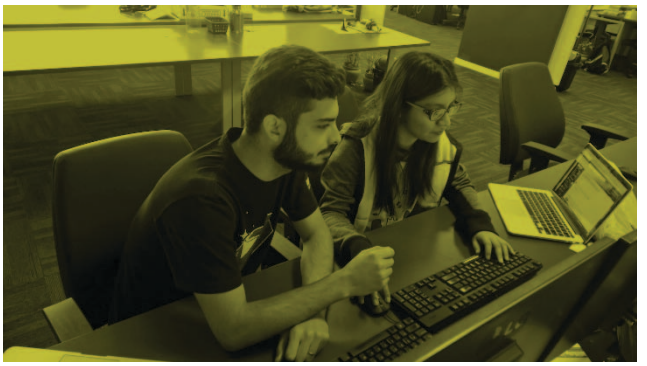
Concept

Pair programming consists of two programmers sharing a single workstation (one screen, keyboard and mouse among the pair). The programmer at the keyboard is usually called the "driver", the other, also actively involved in the programming task but focusing more on overall direction is the "navigator"; it is expected that the programmers swap roles every few minutes or so.

All code to be sent into production is created by two people working together at a single computer. Pair programming increases software quality without impacting time to deliver. It is counter intuitive, but 2 people working at a single computer will add as much functionality as two working separately except that it will be much higher in quality. With increased quality comes big savings later in the project.

- The best way to pair program is to just sit side by side in front of the monitor. Slide the key board and mouse back and forth. Both programmers concentrate on the code being written.

- Pair programming is a social skill that takes time to learn. You are striving for a cooperative way to work that includes give and take from both partners regardless of corporate status. The best pair programmers know when to say "let's try your idea first." Do not expect people to be good at it from the start. It helps if you have someone on your team with experience to show everyone what it should feel like.



In that style, actively sharing the input devices can be problematic: except for a few multiplayer games, it's hard to find software out there that supports two people typing and clicking around at the same time. To avoid confusion and unnecessary use of the backspace key, pairs naturally agree on who gets to use the keyboard and mouse at any given time, and those arrangements can vary widely depending on a number of factors.

To us, an extremely important part of such agreements is what to do when a programmer is not typing. The term "navigator" (as the opposite of "driver"), used to describe that role, likely comes from rallying – where it's usual to have a navigator sitting next to the pilot or driver handing out coordinates and calling attention to troublesome spots along the way.

Much like in racing though, the driver gets a disproportionate share of the credit and attention, and often, the mechanics of good navigation have been somewhat neglected.

Origins

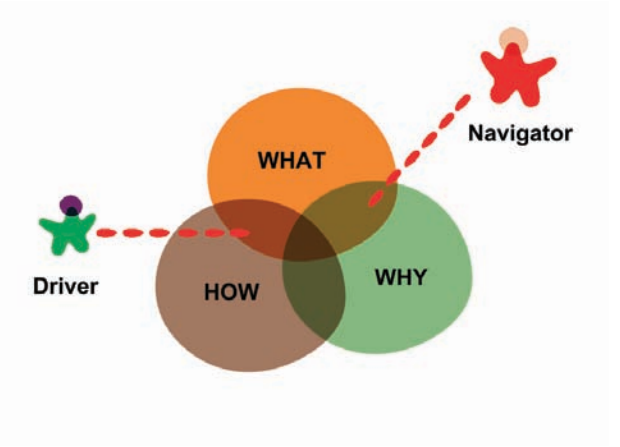
The names of various celebrities have been invoked in an attempt to imbue pair programming with an aura of necessity if not sanctity; anecdotes of John Von Neummann, Fred Brooks, Jerry Weinberg, Richard Gabriel or Edsger Dijkstra using the practice are fascinating but sometimes hard to substantiate. However, the following timeline of verifiable sources does suggest that pair programming, in its modern form, has been around since well before the Agile movement:

- 1992: "Dynamic Duo" is the term coined by Larry Constantine, reporting on a visit to Whitesmiths Inc., a compiler vendor started by P.J. Plauger, one of the implementors of C: "At each terminal were two programmers! Of course, only one programmer was actually cutting code at each keyboard, but the others were peering over their shoulders." Whitesmiths existed from 1978 to 1988.

- 1993: "The benefits of collaboration for student programmers" by Wilson et al. is one early empirical study indicating benefits of pairing for programming tasks specifically. Posterior studies are more abundant and driven by the desire to "validate" pair programming after it had already gained popularity through Extreme Programming.
- 1995: the pattern "Developing in Pairs" is given a brief description, in Alexandrian pattern form, in Jim Coplien's chapter "A Generative Development-Process Pattern Language" from the first patterns book, "Pattern Languages of Program Design".
- 1998: in "Chrysler goes to Extremes", the earliest article about Extreme Programming, pair programming is presented as one of the core practices of the C3 team; it is later described formally as one of XP's original "twelve practices"
- 2000: (or earlier) - the roles of Driver and Navigator are introduced to help explain pair programming; the earliest known reference is a mailing list posting; note however that the reality of these roles has been disputed, for instance Sallyann Bryant's article "Pair programming and the mysterious role of the navigator"
- 2002: "Pair Programming Illuminated", by Laurie Williams and Robert Kessler, is the first book devoted exclusively to the practice and discusses its theory, practice and the various studies up to that date
- 2003: an anonymous article on the C2 Wiki describes Ping-Pong Programming, a moderately popular variant which marries pairing with test-driven development.

Pairing Mechanics

Pairing involves having two programmers working at a single workstation. One programmer "drives," operating the keyboard, while the other "navigates," watching, learning, asking, talking, and making suggestions. In theory, the driver focuses on the code at hand: the syntax, semantics, and algorithm. The navigator focuses less on that, and more on a level of abstraction higher: the test they are trying to get to pass, the technical task to be delivered next, the time elapsed since all the tests were run, the time elapsed since the last repository commit, and the quality of the overall design. The theory is that pairing results in better designs, fewer bugs, and much better spread of knowledge across a development team, and therefore more functionality per unit time, measured over the long term.



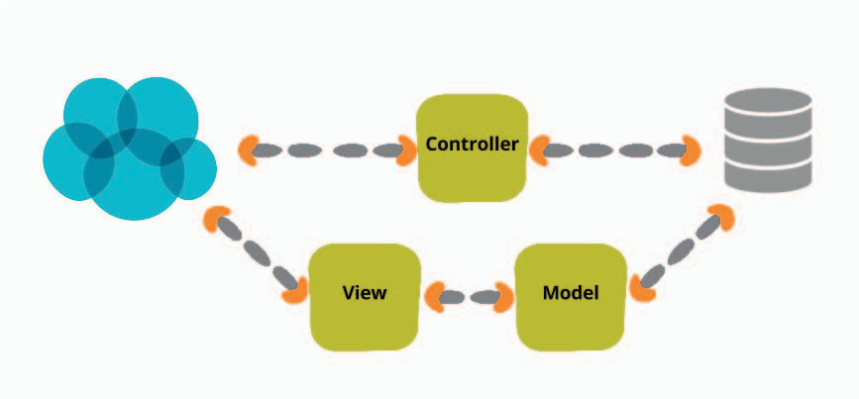
Spreading Knowledge

Certainly as a mentoring mechanism, pairing is hard to beat. If pairs switch off regularly (as they should), pairing spreads several kinds of knowledge throughout the team with great efficiency: codebase knowledge, design and architectural knowledge, feature and problem domain knowledge, language knowledge, development platform knowledge, framework and tool knowledge, refactoring knowledge, and testing knowledge. There is not much debate that pairing spreads these kinds of knowledge better than traditional code reviews and less formal methods. So what productivity penalty, if any, do you pay for spreading knowledge so well?

Productivity refers to the measure of how efficiently and effectively tasks are completed and goals are achieved within a given timeframe.

Pairing and Productivity

Research results and anecdotal reports seem to show that short-term productivity might decrease modestly (about 15%), but because the code produced is so much better, long-term productivity goes up. And certainly it depends on how you measure productivity, and over what term. In an agile context, productivity is often measured in running, tested features actually delivered per iteration and per release. If a team measures productivity in lines of code per week, they may indeed find that pairing causes this to drop (and if that means fewer lines of code per running, tested feature, that's a good thing!).



Productivity and Staff Turnover

Proponents of pairing claim that if you measure productivity across a long enough term to include staff being hired and leaving, pairing starts to show even more value. In many mainstream projects, expertise tends to accumulate in "islands of knowledge." Individual programmers tend to know lots of important things that the other programmers do not know as well. If any of these islands leaves the team, the project may be delayed badly or worse. Part of the theory of pairing is that by spreading many kinds of knowledge so widely within a team, management reduces their exposure to this constant threat of staff turnover. In Extreme Programming, they speak of the Truck Number: the number of team members that would need to be hit by a truck to kill the project. Extreme Programming projects strive to keep the Truck Number as close as possible to the total team size. If someone leaves, there are usually several others to take his or her place. It's not that there is no specialization, but certainly everyone knows more about all of what is going on. If you measure productivity in terms of features delivered over several releases by such a team, it should be higher than if pairing does not occur.

Pairing Strategies

In by-the-book Extreme Programming, all production code is written by pairs. Many non-XP agile teams do not use pairing at all. But there is lots of middle ground between no pairing and everyone pairing all the time. Try using pairing when mentoring new hires, for extremely high-risk tasks, at the start of a new project when the design is new, when adopting a new technology, or on a rotating monthly or weekly basis. Programmers who prefer to pair might be allowed to, while those who do not are allowed not to. The decision to use code reviews instead of any pairing at all is popular, but we do not know of any reason not to at least experiment with pairing. There is no reasonable evidence that it hurts a team or a project, and there is increasing evidence that it is a helpful best practice.

Skill Levels

As suggested above one of the major issues preventing effective pairing is passivity. When used simultaneously with test-driven development, one variant called "ping-pong programming" encourages more frequent switching of roles: one programmer writes a failing unit test, then passes the keyboard to the other who writes the corresponding code, then goes on to a new test. This variant can be used purely for pedagogic purposes, or by already experienced programmers as a playful variant.

Ping-pong programming is a collaborative software development technique often used in agile and pair programming environments.

- Beginner:
- able to participate as navigator, in particular to intervene appropriately
- able to participate as driver, in particular to explain code while writing it
- Intermediate
- can tell the right moment to give up the keyboard and switch roles
- can tell the right moment to "steal" the keyboard and switch roles
- Advanced
- able to "drop in" when another pair has been working on a task and pick up the navigator role smoothly

Signs of Use

- the room's furniture and workstations are set up so as to encourage pairing (in teams new or hostile to pairing, obvious mistakes are tolerated, such as desks with too little room for two chairs)
- the room's noise level is controlled: the muted conversations from several simultaneous pairs create a background hum but do not rise to the level where they would disturb anyone's work
- if, on entering the room, you spot any programmer wearing an audio headset, take that as a "negative" sign - not only is pairing probably not practiced in the team but the conditions for successful adoptions are likely not met

Expected Benefits

- increased code quality: "programming out loud" leads to clearer articulation of the complexities and hidden details in coding tasks, reducing the risk of error or going down blind alleys
- better diffusion of knowledge among the team, in particular when a developer unfamiliar with a component is pairing with one who knows it much better
- better transfer of skills, as junior developers pick up micro-techniques or broader skills from more experienced team members
- large reduction in coordination efforts, since there are N/2 pairs to coordinate instead of N individual developers
- improved resiliency of a pair to interruptions, compared to an individual developer: when one member of the pair must attend to an external prompt, the other can remains focused on the task and can assist in regaining focus afterwards

Potential Costs

While empirical studies have yet to yield definite results on either benefits or costs, a commonly cited best-case estimate of 15% overhead is claimed for systematic pairing, relative to individual work; this overhead, it is claimed (again with some empirical support, though not entirely conclusive), is compensated by gains in code quality which usually entails significant maintenance penalties down the road.

3.5 DEBUGGING

KEY NOTES

Definition

Debugging, in computer programming and engineering, is a multistep process that involves identifying a problem, isolating the source of the problem, and then either correcting the problem or determining a way to work around it. The final step of debugging is to test the correction or workaround and make sure it works. Key things that you can use the debugger to determine are:

The flow of the program (what happens next on a line by line basis)

- The creation of variables
- The data being stored in each variable
- The entering/leaving of functions
- The calculations that are made
- The entering of IF statements or ELSE statements
- The LOOPING of code, etc.

The Debugging Process

The debugging process include:

- Reproduce the problem.
- Describe the bug. Try to get as much input from the user to get the exact reason.
- Capture the program snapshot when the bug appears. Try to get all the variable values and states of the program at that time.
- Analyze the snapshot based on the state and action. Based on that try to find the cause of the bug.
- Fix the existing bug, but also check that any new bug does not occur.

Common Debugging Tools

A debugger or debugging tool is a computer program that is used to test and debug other programs (the "target" program).

Learning Objectives

- Use a JavaScript debugger
- Explain useful tips for developers

Definition

Debugging is the process of detecting and removing of existing and potential errors (also called as 'bugs') in a software code that can cause it to behave unexpectedly or crash. To prevent incorrect operation of a software or system, debugging is used to find and resolve bugs or defects. When various subsystems or modules are tightly coupled, debugging becomes harder as any change in one module may cause more bugs to appear in another. Sometimes it takes more time to debug a program than to code it.

The Debugging Process

To debug a program, user has to start with a problem, isolate the source code of the problem, and then fix it. A user of a program must know how to fix the problem as knowledge about problem analysis is expected. When the bug is fixed, then the software is ready to use. Debugging tools (called debuggers) are used to identify coding errors at various development stages. They are used to reproduce the conditions in which error has occurred, then examine the program state at that time and locate the cause. Programmers can trace the program execution step-by-step by evaluating the value of variables and stop the execution wherever required to get the value of variables or reset the program variables. Some programming language packages provide a debugger for checking the code for errors while it is being written at run time.

Integrated Development Environment (IDE) is a software application that provides comprehensive tools and features to facilitate the development of software applications.

In software development, debugging involves locating and correcting code errors in a computer program. Debugging is part of the software testing process and is an integral part of the entire software development lifecycle. The debugging process starts as soon as code is written and continues in successive stages as code is combined with other units of programming to form a software product. In a large program that has thousands and thousands of lines of code, the debugging process can be made easier by using strategies such as unit tests, code reviews and pair programming.

Once an error has been identified, it is necessary to actually find the error in the code. At this point, it can be useful to look at the code's logging and use a stand-alone debugger tool or the debugging component of an integrated development environment (IDE). Invariably, the bugs in the functions that get most use are found and fixed first. In some cases, the module that presents the problem is obvious, while the line of code itself is not. In that case, unit tests -- such as JUnit and xUnit, which allow the programmer to run a specific function with specific inputs -- can be helpful in debugging.

The standard practice is to set up a "breakpoint" and run the program until that breakpoint, at which time program execution stops. The debugging component of an IDE typically provides the programmer with the capability to view memory and see variables, run the program to the next breakpoint, execute just

the next line of code, and, in some cases, change the value of variables or even change the contents of the line of code about to be executed.

Common Debugging Tools

Debugging is the process of finding and resolving of defects that prevent correct operation of computer software or a system.

Source code analyzers, which include security, common code errors and complexity analyzers, can also be helpful in **debugging**. A complexity analyzer can find modules that are so intricate as to be hard to understand and test. Some tools can actually analyze a test run to see what lines of code were not executed, which can aid in debugging. Other debugging tools include advanced logging and simulators that allow the programmer to model how an app on a mobile device will display and behave. Some tools, especially open source tools and scripting languages, do not run in an IDE and require a more manual approach to debugging. Such techniques include dropping values to a log, extensive "print" statements added during code execution or hard-coded "wait" commands that simulate a breakpoint by waiting for keyboard input at specific times.

The use of the word *bug* as a synonym for *error* originated in engineering. The term's application to computing and the inspiration for using the word *debugging* as a synonym for *troubleshooting* has been attributed to Admiral Grace Hopper, a pioneer in computer programming, who was also known for her dry sense of humor. When an actual bug (a moth) got caught between electrical relays and caused a problem in the U.S. Navy's first computer, Admiral Hopper and her team "debugged" the computer and saved the moth. It now resides in the Smithsonian Museum.

Use a JavaScript Debugger

A debugger is an application that places all aspects of script execution under the control of the programmer. Debuggers provide fine-grained control over the state of the script through an interface that allows you to examine and set values as well as control the flow of execution. Once a script has been loaded into a debugger, it can be run one line at a time or instructed to halt at certain breakpoints. Once execution is halted, the programmer can examine the state of the script and its variables in order to determine if something is amiss. You can also watch variables for changes in their values.

Useful Tips for Developers

You can keep the following tips in mind to reduce the number of errors in your scripts and simplify the debugging process −

- Use plenty of comments. Comments enable you to explain why you wrote the script the way you did and to explain particularly difficult sections of code.
- Always use indentation to make your code easy to read. Indenting statements also makes it easier for you to match up beginning and ending tags, curly braces, and other HTML and script elements.
- Write modular code. Whenever possible, group your statements into functions. Functions let you group related statements, and test and reuse portions of code with minimal effort.
- Be consistent in the way you name your variables and functions. Try using names that are long enough to be meaningful and that describe the contents of the variable or the purpose of the function.
- Use consistent syntax when naming variables and functions. In other words, keep them all lowercase or all uppercase; if you prefer Camel-Back notation, use it consistently.
- Test long scripts in a modular fashion. In other words, do not try to write the entire script before testing any portion of it. Write a piece and get it to work before adding the next portion of code.
- Use descriptive variable and function names and avoid using single-character names.
- Watch your quotation marks. Remember that quotation marks are used in pairs around strings and that both quotation marks must be of the same style (either single or double).
- Watch your equal signs. You should not used a single = for comparison purpose.
- Declare variables explicitly using the var keyword.



TECHNIQUE 9

ECLIPSE JAVASCRIPT DEVELOPMENT TOOLS

The Eclipse JavaScript Development Tools provide Eclipse Platform plug-ins that implement an IDE supporting the development of JavaScript applications and JavaScript within web applications. It adds a JavaScript project type and perspective to the Eclipse Workbench as well as a number of views, editors, wizards, and builders.

- ☐ We already practice this technique
- ☐ This technique would not be suitable for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 10

JAVASCRIPT IN MOZILLA FIREFOX

JavaScript is a core feature of many websites, especially modern interactive websites. Disabling it can be done for speed and bandwidth concerns, security, or even to ensure that the site works with a screen reader for visually impaired users. Unfortunately, not all websites have a developed site without JavaScript enabled. Because of this, disabling JavaScript in Mozilla 24.0 is typically only recommended for advanced users, though it's easy to turn back on. There are two main ways to enable JavaScript in Firefox: using add-ons, which gives users controls not designed by Mozilla, or manually configuring the browser.

- ☐ We already practice this technique
- ☐ This technique would not be suitable for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 11

COOKIES AND JAVASCRIPT

Cookies are used by websites to store information that allows for a more consistent user experience and to help users navigate websites. Cookies also personalize a website user's experience by storing information about a user's activity on a particular site and storing that information for future visits. JavaScript is a web programming language commonly used for enhancing user interfaces by allowing web programmers to make web-pages more interactive. Knowing how to enable cookies and JavaScript will allow you to make the most of your web browsing.

- □ We already practice this technique
- □ This technique would not be suitable for our operation
- □ We practiced this in the past and need to jumpstart
- □ We should implement this technique



TECHNIQUE 12

ERROR HANDLING AND DEBUGGING

Error handling and debugging are crucial aspects of programming in JavaScript, as they help developers identify and resolve issues in their code, leading to more reliable and efficient applications. Debugging tools enable you to inspect the values of variables at different points in the code, helping you understand how data changes during execution.

- □ We already practice this technique
- □ This technique would not be suitable for our operation
- □ We practiced this in the past and need to jumpstart
- □ We should implement this technique

SUMMARY

- JavaScript is incredibly versatile. You can start small, with carousels, image galleries, fluctuating layouts, and responses to button clicks. With more experience you'll be able to create games, animated 2D and 3D graphics, comprehensive database-driven apps, and much more!
- A scripting language can be thought of as a lightweight programming language; one that can be interpreted by a browser without needing to be compiled first. The script is actually just some commands that the browser has to do.
- Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.
- The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.
- Conditionals and variables are essential concepts in JavaScript, a versatile and widely used programming language.
- Conditionals are programming structures that allow developers to execute different blocks of code based on certain conditions.
- Conditionals allow for decision-making based on data conditions, while variables facilitate the storage and manipulation of data, enhancing the dynamic nature of JavaScript programs.
- Pair programming consists of two programmers sharing a single workstation (one screen, keyboard and mouse among the pair). The programmer at the keyboard is usually called the "driver", the other, also actively involved in the programming task but focusing more on overall direction is the "navigator"; it is expected that the programmers swap roles every few minutes or so.
- Debugging is the process of detecting and removing of existing and potential errors (also called as 'bugs') in a software code that can cause it to behave unexpectedly or crash.

KNOWLEDGE CHECK

1. Why so JavaScript and Java have similar name?

a. JavaScript is a stripped-down version of Java

b. JavaScript's syntax is loosely based on Java's

c. They both originated on the island of Java

d. None of the above

2. When a user views a page containing a JavaScript program, which machine actually executes the script?

a. The User's machine running a Web browser
b. The Web server
c. A central machine deep within Netscape's corporate offices
d. None of the above

3. ……. JavaScript is also called client-side JavaScript.

a. Microsoft
b. Navigator
c. LiveWire
d. Native

4. ……. JavaScript is also called server-side JavaScript.

a. Microsoft
b. Navigator
c. LiveWire
d. Native

5. What are variables used for in JavaScript Programs?

a. Storing numbers, dates, or other values
b. Varying randomly
c. Causing high-school algebra flashbacks
d. None of the above

6. Which keyword is used to declare a variable in JavaScript?

a. new
b. var
c. let
d. const

7. What is the purpose of the "getElementById" method in JavaScript?

a. It creates a new HTML element.
b. It retrieves an element's value.
c. It selects an element based on its class name.
d. It retrieves an element by its unique ID.

8. Which function is used to display a dialog box with a message and an "OK" button in JavaScript?

a. confirm()
b. alert()

c. prompt()
d. log()

REVIEW QUESTIONS

1. What is JavaScript? How to use external JavaScript file?
2. What is DOM? What is the use of document object?
3. How to create function in JavaScript? Explain.
4. What is a potential pitfall with using typeof bar === "object" to determine if bar is an object? How can this pitfall be avoided?
5. What is the significance of, and reason for, wrapping the entire content of a JavaScript source file in a function block?

Check your Result

1. (b) 2. (a) 3. (b) 4. (c) 5. (a)
6. (b) 7. (d) 8. (b)

REFERENCES

1. Effective JavaScript: 68 Specific Ways to Harness the Power of JavaScript by David Herman (2012)
2. Eloquent JavaScript: A Modern Introduction to Programming by Marijn Haverbeke (2018)
3. Head First JavaScript Programming: A Brain-Friendly Guide by Eric T. Freeman and Elisabeth Robson (2014)
4. High Performance JavaScript: Build Faster Web Application Interfaces by Nicholas C. Zakas (2010)
5. Learning React: Functional Web Development with React and Redux by Alex Banks and Eve Porcello (2017)
6. Learning React: Modern Patterns for Developing React Apps by Alex Banks and Eve Porcello (2020)
7. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics by Jennifer Niederst Robbins (2018)
8. Mastering JavaScript Functional Programming: Build robust and maintainable software using JavaScript by Federico Kereki (2016)
9. Node.js Design Patterns: Master best practices to build modular and scalable server-side web applications by Mario Casciaro (2016)
10. Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries by Eric Elliott (2014)

11. React Design Patterns and Best Practices: Build easy to scale modular applications using the most powerful components and design patterns by Michele Bertoli (2017)
12. Vue.js: Up and Running: Building Accessible and Performant Web Apps by Callum Macrae (2018)

CHAPTER 4

PHP

4.1 CONCEPT OF PHP

KEY NOTES

Definition

PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

Common uses of PHP

PHP is a general-purpose scripting language that is especially suited to server-side web development, in which case PHP generally runs on a web server. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on websites or elsewhere.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible −

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

Learning Objective

- Define pros and cons of PHP
- Describe how to PHP Environment Setup

Definition

PHP stands for PHP: Hypertext Preprocessor, with that PHP standing for Personal HomePage [Tools]. This type of acronym is known as a retronym. Originally, in 1994, the language was designed as a small set of binaries used to collect some basic site traffic data. In 1997, the parser was rewritten by two Israelis and the name was changed to the current acronym — it being determined that hypertext preprocessor was a decidedly more acceptable name in the business world.

It is an open-source language, used primarily for dynamic web content and server-side applications. PHP is often pointed to as the main competitor with the following:

- Microsoft's C# — Visual Basic.NET — ASP family
- Sun's Java — JSP
- Macromedia's ColdFusion
- CGI — Perl

PHP has many open-source libraries included with the core build, and many more are readily available. Extensions exist to help it interface with a number of systems, including IRC, a number of compression formats, and Windows API. Other extensions exist to let PHP generate file formats on-the-fly, such as a popular extension that allows it to create Adobe Flash® movies.

Since version 3, PHP has integrated object oriented features. Version 5 built substantially on this limited functionality, and the language now has robust object oriented capabilities, including interfaces, exceptions, destructions, and abstracts.

PHP reached wide-spread popularity with version 4, released in 2000. In 2004, version 5 debuted, and it is now considered one the top languages used for server-side scripting.

No doubt much of its popularity is due to its relative ease to learn, and its notorious looseness. Arrays and variables in PHP are able to hold any type of object, variables need not be declared, and the syntax is remarkably simple.

Unlike many languages, such as C# or Perl, which have primarily a following of more generalist programmers, many PHP programmers know no other language. This occasionally causes it to be dismissed as a lesser language, but its growing popularity and the many robust and efficient sites built using it as a structure seem to dispel this myth.

PHP has occasionally been criticized for what are viewed by some as security flaws, in comparison to languages such as ASP. A lack of easily understandable error messages, a sometimes overly robust configuration file, and an obviously incomplete set of built-in functions are also pointed to as areas which could use marked improvement.

Development continues apace, however, and with each successive build, the langugage appears to address more and more of the concerns raised by its open-source community.

Common uses of PHP

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Robust configuration file is a structured and well-designed file that contains settings, parameters, and options used to configure software applications, systems, or services.

Characteristics of PHP

The PHP language revolves around the central theme of practicality. PHP is about providing the programmer with the necessary tools to get the job done in a quick and efficient fashion. Five important characteristics make PHP's practical nature possible:

- Simplicity
- Efficiency
- Familiarity
- Security
- Flexibility

Simplicity

A PHP script can consist of 10,000 lines or one line: whatever you need to get the job done. There is no need to include libraries, special compilation directives, or anything of the sort. The PHP engine simply begins executing the code after the first **escape sequence** (). If the code is syntactically correct, it will be executed exactly as it is displayed.

Escape sequence is a sequence of characters that does not represent itself when used inside a character or string literal, but is translated into another character or a sequence of characters that may be difficult or impossible to represent directly.

Efficiency

Efficiency is an extremely important consideration for working in a multiuser environment such as the WWW. PHP 4.0 introduced resource allocation mechanisms and more pronounced support for object-oriented programming, in

addition to session management features. Reference counting has also been introduced in the latest version, eliminating unnecessary memory allocation.

Familiarity

Programmers from many backgrounds will find themselves already accustomed to the PHP language. Many of the language's constructs are borrowed from C and Perl, and in many cases PHP code is almost indistinguishable from that found in the typical C or Pascal program. This minimizes the learning curve considerably.

Security

PHP provides developers and administrators with a flexible and efficient set of security safeguards. These safeguards can be divided into two frames of reference: system level and application level.

System-Level Security Safeguards

PHP furnishes a number of security mechanisms that administrators can manipulate,providing for the maximum amount of freedom and security when PHP is properly configured. PHP can be run in what is known as safe mode, which can limit users' attempts to exploit the PHP implementation in many important ways. Limits can also be placed on maximum execution time and memory usage, which if not controlled can have adverse affects on server performance. Much as with a cgi-bin folder, administrators can also place restrictions on the locations in which users can view and execute PHP scripts and use PHP scripts to view guarded server information, such as the passwd file.

Application-Level Security Safeguards

Several trusted data encryption options are supported in PHP's predefined function set. PHP is also compatible with many third-party applications, allowing for easy-integration with secure ecommerce technologies. Another advantage is that the PHP source code is not viewable through the browser because the script is completely parsed before it is sent back to the requesting user. This benefit of PHP's server-side architecture prevents the loss of creative scripts to users at least knowledgeable enough to execute a 'View Source'. Because PHP is an embedded language, it is extremely flexible towards meeting the needs of the developer. Although PHP is generally touted as being used in conjunction solely with HTML, it can also be integrated alongside languages like JavaScript, WML, XML, and many others. Additionally, as with most other mainstream languages, wisely planned PHP applications can be easily expanded as needed. Browser dependency is not an issue because PHP scripts are compiled entirely on the server side before being sent to the user. In fact, PHP scripts can be sent to just about any kind of device containing a browser, including cell phones, personal digital assistant (PDA) devices, pagers, laptops, not to mention the traditional PC. People who want to develop shell-based applications can also execute PHP from the command

line. Since PHP contains no server-specific code, users are not limited to a specific and perhaps unfamiliar Web server. Apache, Microsoft IIs, Netscape Enterprise Server, Stronghold, and Zeus are all fair game for PHP's server integration. Because of the various platforms that these servers operate on, PHP is largely platform independent, available for such platforms as UNIX, Solaris, FreeBSD, and Windows 95/98/NT. Finally, PHP offers access to external components, such as Enterprise Java Beans and Win32 COM objects. These newly added features put PHP in the big league, truly enabling developers to scale PHP projects upward and outward as need be.

Flexibility

The open source development strategy has gained considerable notoriety in the software industry. The prospect of releasing source code to the masses has resulted in undeniably positive outcomes for many projects, perhaps most notably Linux, although the success of the Apache project has certainly been a major contributor in proving the validity of the open source ideal. The same holds true for the developmental history of PHP, as users worldwide have been a huge factor in the advancement of the PHP project.

Personal Digital Assistant (PDA) is a computer that fits in the palm of your hand to help collect such information as contacts, appointments, files, and programs.

"Hello World" Script in PHP

To get a feel for PHP, first start with simple PHP scripts. Since "Hello, World!" is an essential example, first we will create a friendly little "Hello, World!" script.

As mentioned earlier, PHP is embedded in HTML. That means that in amongst your normal HTML (or XHTML if you're cutting-edge) you'll have PHP statements like this

```
<html>

  <head>
    <title>Hello World</title>
  </head>

  <body>
    <?php echo "Hello, World!";?>
  </body>
```

</html>

It will produce following result

Hello, World!

If you examine the HTML output of the above example, you'll notice that the PHP code is not present in the file sent from the server to your Web browser. All of the PHP present in the Web page is processed and stripped from the page; the only thing returned to the client from the Web server is pure HTML output.

All PHP code must be included inside one of the three special markup tags ATE are recognized by the PHP Parser.

<?php PHP code goes here ?>

<? PHP code goes here ?>

<script language="php"> PHP code goes here </script>

A most common tag is the <?php...?> and we will also use the same tag in our tutorial.

From the next chapter we will start with PHP Environment Setup on your machine and then we will dig out almost all concepts related to PHP to make you comfortable with the PHP language.

Pros and Cons of PHP

As with all programming languages, there are a number of pros and cons to the Hypertext Preprocessor (PHP) scripting language that affect how suitable it is for various applications. PHP advantages include that it's a free, open-source programming language, relatively easy to learn and flexible. PHP also supports a range of types of database, making it a good choice for applications that need to communicate with databases. Cons of PHP include that it's ineffective at producing desktop applications, relatively slow compared to some other languages and has poor support for handling errors.

Programming language usually refers to high-level languages, such as BASIC, C, C++, COBOL, FORTRAN, Ada, and Pascal. Each language has a unique set of keywords (words that it understands) and a special syntax for organizing program instructions.

There are many pros and cons of PHP, as there are with any **programming language**. One of the main PHP advantages is that it is open source. This means it's free to use and is constantly being improved by a large number of people rather than a single company. The developers behind PHP have created an extensive online resource of all the language's functions,

including examples of how to use them, which makes it easier to learn PHP than to learn the alternatives. PHP is such a popular programming language that there also are a number of online support groups to help speed up the learning process.

Another advantage of PHP is that it's a flexible programming language. A lot of programming languages in 2011 are designed to be object-oriented, which is a convention in programming that helps to organize data more efficiently. Programmers often debate when object-oriented programming is required. PHP supports object-oriented programming but also can function without it.

PHP is highly effective at accessing and communicating with various database types. This makes it perfect for web-based scripts, such as website content management systems, which need to store data in a database. The programming language supports a number of database types, which is one of the most important of the pros and cons of PHP.

One of the main disadvantages of PHP is that it is generally not suitable for making desktop applications. This is why it's essential to weigh the pros and cons of PHP in relation to a particular project before deciding whether to use it. Also, compared to other languages, PHP applications tend to run slower. Whether this is an issue for a particular project depends on the processing power required to execute it.

Among the most frequently talked about pros and cons of PHP is its error handling. PHP error handling is traditionally considered poor when compared to other programming languages, mainly because of PHP's development history and its support for older functions. This means it can take more time to discover why a particular piece of code isn't working as expected.

Different Types of PHP Tools

PHP is a web development scripting language that is utilized to create dynamic, text-based website pages. The acronym has developed over time and is now known to refer to the language's use as a PHP Hypertext Preprocessor. The system can be used on the majority of server units and is available without charge for use on all of the major operating systems (OSs). To assist in the development of website pages through the scripting language, several software-based PHP tools have been created to provide more control to developers. These tools include designers, viewers, debuggers, and security encoding programs, among others.

Created to simplify the development process when utilizing the scripting language, PHP tools assist in common tasks, such as editing, encoding, and debugging program code. These applications exist with the core aim of making the development process of PHP scripts as quick and easy as possible. The dense code of standard PHP scripts can make identifying and addressing errors a time-consuming task. PHP tools have been developed to streamline this task along with other operations associated with compiling and editing the script.

In terms of common usage when looking at PHP tools, certain applications are more commonplace than others. In general PHP development, a debugger is one of the key applications utilized. A PHP debugger gives full control of script execution to the programmer. Use of this tool allows the developer to run code one line at a time to isolate problem areas.

One of the biggest concerns of developers is theft of script and unauthorized code modifications; to counteract this risk, a PHP encoder can be used. An encoding tool has the ability to change source code into PHP-specific byte codes. Carrying out this encoding prevents unauthorized access to the code from external sources. An additional benefit to encoding is the operational acceleration that can be gained through running a series of PHP byte codes in comparison to standard script.

Visual PHP tools are also available in the form of designer applications and "what you see is what you get" (WYSIWYG) tools. Designer applications provide a visual aid to search through hierarchical PHP code structure, functions, and objects. To assist in the creation of dynamic web pages, WYSIWYG PHP tools render the PHP code into a visual representation of the script. This allows the developer to view an approximation of the final page during development.

Visual representation is to make complex or abstract concepts more understandable and accessible to humans by presenting them in a visually intuitive manner.

PHP Environment Setup

We have set up the PHP Programming environment on-line, so that you can compile and execute all the available examples on line. It gives you confidence in what you are reading and enables you to verify the programs with different options. Feel free to modify any example and execute it on-line.

```
<html>
  <head>
    <title>Online PHP Script Execution</title>
  </head>

  <body>

    <?php
      echo "<h1>Hello, PHP!</h1>";
    ?>
```

```
</body>
</html>
```

For most of the examples given in this tutorial, you will find a Try it option in our website code sections at the top right corner that will take you to the online compiler. So just make use of it and enjoy your learning.

In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

Web Server: PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server.

Database: PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.

PHP Parser: In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser. This tutorial will guide you how to install PHP parser on your computer.

Internet Information Services (IIS), formerly known as Internet Information Server, is a web server software developed by Microsoft for hosting websites, applications, and services on the Windows operating system.

PHP Parser Installation

Before you proceed it is important to make sure that you have proper environment setup on your machine to develop your web programs using PHP.

Type the following address into your browser's address box.

```
http://127.0.0.1/info.php
```

If this displays a page showing your PHP installation related information then it means you have PHP and Webserver installed properly. Otherwise you have to follow given procedure to install PHP on your computer.

This section will guide you to install and configure PHP over the following four platforms:

- PHP Installation on Linux or Unix with Apache
- PHP Installation on Mac OS X with Apache
- PHP Installation on Windows NT/2000/XP with IIS
- PHP Installation on Windows NT/2000/XP with Apache

4.2 FUNCTIONS OF PHP

KEY NOTES

Definition

A function is a self-contained block of code that performs a specific task. PHP has a huge collection of internal or built-in functions that you can call directly within your PHP scripts to perform a specific task, like gettype(), print_r(), var_dump, etc.

Data Types in PHP

The values assigned to a PHP variable may be of different data types including simple string and numeric types to more complex data types like arrays and objects.

PHP supports total eight primitive data types: Integer, Floating point number or Float, String, Booleans, Array, Object, resource and NULL.

Variables in PHP

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Learning Objective

- How to Create PHP function
- Describe PHP functions with parameters
- Define PHP functions returning value

Definition

PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.

You already have seen many functions like fopen() and fread() etc. They are built-in functions but PHP gives you option to create your own functions as well.

PHP also allows you to define your own functions. It is a way to create reusable code packages that perform specific tasks and can be kept and maintained separately form main program. Here are some advantages of using functions:

- *Functions reduces the repetition of code within a program*: Function allows you to extract commonly used block of code into a single component. Now can you can perform the same task by calling this function wherever you want without having to copy and paste the same block of code again and again.
- *Functions makes the code much easier to maintain*: Since a function created once can be used many times, so any changes made inside a function automatically implemented at all the places without touching the several files.
- *Functions makes it easier to eliminate the errors*: When the program is subdivided into functions, if any error occur you know exactly what function

causing the error and where to find it. Therefore, fixing errors becomes much easier.

- *Functions can be reused in other application*: Because a function is separated from the rest of the script, it's easy to reuse the same function in other applications just by including the php file containing those functions.

There are two parts which should be clear to you −

- Creating a PHP Function
- Calling a PHP Function

In fact you hardly need to create your own PHP function because there are already more than 1000 of built-in library functions created for different area and you just need to call them according to your requirement.

Library functions are a key component of programming because they save time and effort, promote code reusability, and ensure consistent and reliable implementations of common tasks.

Creating PHP Function

It is very easy to create your own PHP function. Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it. Following example creates a function called writeMessage() and then calls it just after creating it.

Note that while creating a function its name should start with keyword function and all the PHP code should be put inside { and } braces as shown in the following example below:

```
<html>

  <head>
    <title>Writing PHP Function</title>
  </head>

  <body>

    <?php
      /* Defining a PHP Function */
      function writeMessage() {
           echo "You are really a nice person, Have a nice
time!";
      }
```

```
      /* Calling a PHP Function */
      writeMessage();
   ?>

   </body>
</html>
```

This will display following result

```
You are really a nice person, Have a nice time!
```

PHP Functions with Parameters

PHP gives you option to pass your parameters inside a function. You can pass as many as parameters your like. These parameters work like variables inside your function. Following example takes two integer parameters and add them together and then print them.

```
<html>

  <head>
    <title>Writing PHP Function with Parameters</title>
  </head>

  <body>

    <?php
      function addFunction($num1, $num2) {
        $sum = $num1 + $num2;
        echo "Sum of the two numbers is : $sum";
      }

      addFunction(10, 20);
    ?>

  </body>
</html>
```

This will display following result

```
Sum of the two numbers is: 30
```

Passing Arguments by Reference

It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.

Any changes made to an argument in these cases will change the value of the original variable. You can pass an argument by reference by adding an ampersand to the variable name in either the function call or the function definition.

Following example depicts both the cases.

```
<html>

  <head>
    <title>Passing Argument by Reference</title>
  </head>

  <body>

    <?php
      function addFive($num) {
        $num += 5;
      }

      function addSix(&$num) {
        $num += 6;
      }

      $orignum = 10;
      addFive( $orignum );

      echo "Original Value is $orignum<br />";

      addSix( $orignum );
      echo "Original Value is $orignum<br />";
    ?>
```

```
  </body>
</html>
```

This will display following result

```
Original Value is 10
Original Value is 16
```

PHP Functions Returning Value

A function can return a value using the return statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code.

You can return more than one value from a function using return array(1,2,3,4).

Following example takes two integer parameters and add them together and then returns their sum to the calling program. Note that return keyword is used to return a value from a function.

```
<html>

  <head>
    <title>Writing PHP Function which returns value</title>
  </head>

  <body>

    <?php
      function addFunction($num1, $num2) {
        $sum = $num1 + $num2;
        return $sum;
      }
      $return_value = addFunction(10, 20);

      echo “Returned value from the function : $return_value”;
    ?>

  </body>
</html>
```

This will display following result

```
Returned value from the function: 30
```

Setting Default Values for Function Parameters

You can set a parameter to have a default value if the function's caller doesn't pass it.

Following function prints NULL in case use does not pass any value to this function.

```
<html>

  <head>
    <title>Writing PHP Function which returns value</title>
  </head>

  <body>

    <?php
      function printMe($param = NULL) {
        print $param;
      }

      printMe("This is test");
      printMe();
    ?>

  </body>
</html>
```

This will produce following result

```
This is test
```

Dynamic Function Calls

It is possible to assign function names as strings to variables and then treat these variables exactly as you would the function name itself. Following example depicts this behavior.

```
<html>

  <head>
    <title>Dynamic Function Calls</title>
```

```
</head>

<body>

  <?php
    function sayHello() {
      echo "Hello<br />";
    }

    $function_holder = "sayHello";
    $function_holder();
  ?>

  </body>
</html>
```

This will display following result −

```
Hello
```

Data Types in PHP

PHP provides eight types of values, or data types. Four are scalar (single-value) types: integers, floating-point numbers, strings, and booleans. Two are compound (collection) types: arrays and objects. The remaining two are special types: resource and NULL. Numbers, booleans, resources, and NULL are discussed in full here, while strings, arrays, and objects are big enough topics.

Integers

Integers are whole numbers, like 1, 12, and 256. The range of acceptable values varies according to the details of your platform but typically extends from -2,147,483,648 to +2,147,483,647. Specifically, the range is equivalent to the range of the long data type of your C compiler. Unfortunately, the C standard doesn't specify what range that long type should have, so on some systems you might see a different integer range.

Integer literals can be written in decimal, octal, or hexadecimal. Decimal values are represented by a sequence of digits, without leading zeros. The sequence may begin with a plus (+) or minus (-) sign. If there is no sign, positive is assumed. Examples of decimal integers include the following:

1998

-641

+33

Octal numbers consist of a leading 0 and a sequence of digits from 0 to 7. Like decimal numbers, octal numbers can be prefixed with a plus or minus. Here are some example octal values and their equivalent decimal values:

```
 0755     // decimal 493
+010      // decimal 8
```

Octal number is a base-8 numeral system that uses digits from 0 to 7 to represent values. It is similar to the more common decimal (base-10) system that we use every day, which uses digits 0 to 9.

Hexadecimal values begin with 0x, followed by a sequence of digits (0-9) or letters (A-F). The letters can be upper- or lowercase but are usually written in capitals. Like decimal and octal values, you can include a sign in hexadecimal numbers:

```
 0xFF       // decimal 255
0x10        // decimal 16
-0xDAD1     // decimal -56017
```

If you try to store a too-large integer in a variable, it will automatically be turned into a floating-point number.

Use the is_int( ) function (or its is_integer( ) alias) to test whether a value is an integer:

```
if (is_int($x)) {
  // $x is an integer
}
```

Floating-Point Numbers

Floating-point numbers (often referred to as real numbers) represent numeric values with decimal digits. Like integers, their limits depend on your machine's details. PHP floating-point numbers are equivalent to the range of the double data type of your C compiler. Usually, this allows numbers between 1.7E-308 and 1.7E+308 with 15 digits of accuracy. If you need more accuracy or a wider range of integer values, you can use the BC or GMP extensions. See Appendix B for an overview of the BC and GMP extensions.

Floating point is derived from the fact that there is no fixed number of digits before and after the decimal point; that is, the decimal point can float. There are also representations in which the number of digits before and after the decimal point is set, called fixed-pointrepresentations.

PHP recognizes floating-point numbers written in two different formats. There's the one we all use every day:

```
 3.14
0.017
-7.1
```

but PHP also recognizes numbers in scientific notation:

```
 0.314E1     // 0.314*101, or 3.14
```

```
17.0E-3      // 17.0*10-3, or 0.017
```

Floating-point values are only approximate representations of numbers. For example, on many systems 3.5 is actually represented as 3.4999999999. This means you must take care to avoid writing code that assumes floating-point numbers are represented completely accurately, such as directly comparing two floating-point values using ==. The normal approach is to compare to several decimal places:

```
if (int($a * 1000) == int($b * 1000)) {
 // numbers equal to three decimal places
```

Use the is_float( ) function (or its is_real( ) alias) to test whether a value is a floating point number:

```
if (is_float($x)) {
   // $x is a floating-point number
}
```

Strings

Because strings are so common in web applications, PHP includes core-level support for creating and manipulating strings. A string is a sequence of characters of arbitrary length. String literals are delimited by either single or double quotes:

```
 'big dog'
"fat hog"
```

Variables are expanded within double quotes, while within single quotes they are not:

```
 $name = "Guido";
echo "Hi, $name\n";
echo 'Hi, $name';
Hi, Guido
Hi, $name
```

Double quotes also support a variety of string escapes, as listed in Table 4.1.

Table 4.1: Escape sequences in double-quoted strings.

Escape sequence	Character represented
\"	Double quotes
\n	Newline
\r	Carriage return
\t	Tab
\\	Backslash
\$	Dollar sign

\{	Left brace
\}	Right brace
\[	Left bracket
\]	Right bracket
\0 through \777	ASCII character represented by octal value
\x0 through \xFF	ASCII character represented by hex value

A single-quoted string only recognizes \\ to get a literal backslash and \' to get a literal single quote:

```
$dos_path = 'C:\\WINDOWS\\SYSTEM';
$publisher = 'Tim O\'Reilly';
echo "$dos_path $publisher\n";
C:\WINDOWS\SYSTEM Tim O'Reilly
```

To test whether two strings are equal, use the == comparison operator:

```
if ($a == $b) { echo "a and b are equal" }
```

Use the is_string( ) function to test whether a value is a string:

```
if (is_string($x)) {
   // $x is a string
}
```

PHP provides operators and functions to compare, disassemble, assemble, search, replace, and trim strings, as well as a host of specialized string functions for working with HTTP, HTML, and SQL encodings. Because there are so many string-manipulation functions.

Booleans

A boolean value represents a "truth value"—it says whether something is true or not. Like most programming languages, PHP defines some values as true and others as false. Truth and falseness determine the outcome of conditional code such as:

```
if ($alive) { ... }
```

In PHP, the following values are false:

- The keyword false
- The integer 0
- The floating-point value 0.0
- The empty string ("") and the string "0"
- An array with zero elements
- An object with no values or functions
- The NULL value

Any value that is not false is true, including all resource values

PHP provides true and false keywords for clarity:

```
$x = 5;        // $x has a true value
$x = true;     // clearer way to write it
$y = "";       // $y has a false value
$y = false;    // clearer way to write it
```

Use the is_bool( ) function to test whether a value is a boolean:

```
if (is_bool($x)) {
  // $x is a boolean
}
```

Arrays

An array holds a group of values, which you can identify by position (a number, with zero being the first position) or some identifying name (a string):

```
$person[0] = "Edison";
$person[1] = "Wankel";
$person[2] = "Crapper";
$creator['Light bulb'] = "Edison";
$creator['Rotary Engine'] = "Wankel";
$creator['Toilet'] = "Crapper";
```

The array( ) construct creates an array:

```
$person = array('Edison', 'Wankel', 'Crapper');
$creator = array('Light bulb'   => 'Edison',
                 'Rotary Engine' => 'Wankel',
                 'Toilet'       => 'Crapper');
```

There are several ways to loop across arrays, but the most common is a foreach loop:

```
foreach ($person as $name) {
  echo "Hello, $name\n";
}
foreach ($creator as $invention => $inventor) {
  echo "$inventor created the $invention\n";
}
Hello, Edison
Hello, Wankel
Hello, Crapper
```

```
Edison created the Light bulb
Wankel created the Rotary Engine
Crapper created the Toilet
```

You can sort the elements of an array with the various sort functions:

```
sort($person);
// $person is now array('Crapper', 'Edison', 'Wankel')

asort($creator);
// $creator is now array('Toilet'       => 'Crapper',
//                       'Light bulb'   => 'Edison',
//                       'Rotary Engine' => 'Wankel');
```

Use the is_array( ) function to test whether a value is an array:

```
if (is_array($x)) {
  // $x is an array
}
```

There are functions for returning the number of items in the array, fetching every value in the array, and much more.

Objects

PHP supports object-oriented programming (OOP). OOP promotes clean modular design, simplifies debugging and maintenance, and assists with code reuse.

Object-Oriented Programming (OOP) provides a structured way to model, design, and implement complex systems by encapsulating data and behavior into self-contained units.

Classes are the unit of object-oriented design. A class is a definition of a structure that contains properties (variables) and methods (functions). Classes are defined with the class keyword:

```
class Person {
 var $name = '';

 function name ($newname = NULL) {
  if (! is_null($newname)) {
   $this->name = $newname;
  }
  return $this->name;
```

```
  }
}
```

Once a class is defined, any number of objects can be made from it with the new keyword, and the properties and methods can be accessed with the -> construct:

```
 $ed = new Person;
$ed->name('Edison');
printf("Hello, %s\n", $ed->name);
$tc = new Person;
$tc->name('Crapper');
printf("Look out below %s\n", $tc->name);
Hello, Edison
Look out below Crapper
```

Use the is_object( ) function to test whether a value is an object:

```
 if (is_object($x)) {
   // $x is an object
}
```

Resources

Many modules provide several functions for dealing with the outside world. For example, every database extension has at least a function to connect to the database, a function to send a query to the database, and a function to close the connection to the database. Because you can have multiple database connections open at once, the connect function gives you something by which to identify that connection when you call the query and close functions: a resource.

Resources are really integers under the surface. Their main benefit is that they're garbage collected when no longer in use. When the last reference to a resource value goes away, the extension that created the resource is called to free any memory, close any connection, etc. for that resource:

```
$res = database_connect( );   // fictitious function
database_query($res);
$res = "boo";                 // database connection automatically closed
```

The benefit of this automatic cleanup is best seen within functions, when the resource is assigned to a local variable. When the function ends, the variable's value is reclaimed by PHP:

```
 function search ( ) {
  $res = database_connect( );
```

```
 $database_query($res);
}
```

When there are no more references to the resource, it's automatically shut down.

That said, most extensions provide a specific shutdown or close function, and it's considered good style to call that function explicitly when needed rather than to rely on variable scoping to trigger resource cleanup.

Use the is_resource( ) function to test whether a value is a resource:

```
if (is_resource($x)) {
   // $x is a resource
}
```

NULL

There's only one value of the NULL data type. That value is available through the case-insensitive keyword NULL. The NULL value represents a variable that has no value (similar to Perl's undef or Python's None):

```
$aleph = "beta";
$aleph = null;      // variable's value is gone
$aleph = Null;      // same
$aleph = NULL;      // same
```

Use the is_null( ) function to test whether a value is NULL—for instance, to see whether a variable has a value:

```
if (is_null($x)) {
   // $x is NULL
}
```

Variables in PHP

Variables in PHP are represented by a dollar sign followed by the name of the variable. The variable name is case-sensitive.

Variables are always assigned by value. That is to say, when you assign an expression to a variable, the entire value of the original expression is copied into the destination variable. This means, for instance, that after assigning one variable's value to another, changing one of those variables will have no effect on the other.

Variables in PHP are identifiers prefixed with a dollar sign ($). For example:

```
$name
$Age
```

```
$_debugging
$MAXIMUM_IMPACT
```

A variable may hold a value of any type. There is no compile- or runtime type checking on variables. You can replace a variable's value with another of a different type:

```
$what = "Fred";
$what = 35;
$what = array('Fred', '35', 'Wilma');
```

Explicit syntax, the programmer explicitly specifies the details of how a particular operation or task should be performed.

There is no explicit syntax for declaring variables in PHP. The first time the value of a variable is set, the variable is created. In other words, setting a variable functions as a declaration. For example, this is a valid complete PHP program:

```
$day = 60 * 60 * 24;
echo "There are $day seconds in a day.\n";
```

```
There are 86400 seconds in a day.
```

A variable whose value has not been set behaves like the NULL value:

```
if ($uninitialized_variable === NULL) {
  echo "Yes!";
}
```

```
Yes
```

Variable Variables

You can reference the value of a variable whose name is stored in another variable. For example:

```
$foo = 'bar';
$$foo = 'baz';
```

After the second statement executes, the variable $bar has the value "baz".

Variable References

In PHP, references are how you create variable aliases. To make $black an alias for the variable $white, use:

```
$black =& $white;
```

The old value of $black is lost. Instead, $black is now another name for the value that is stored in $white:

```
$big_long_variable_name = "PHP";
```

```
$short =& $big_long_variable_name;
$big_long_variable_name .= " rocks!";
print "\$short is $short\n";
print "Long is $big_long_variable_name\n";
$short is PHP rocks!
Long is PHP rocks!
$short = "Programming $short";
print "\$short is $short\n";
print "Long is $big_long_variable_name\n";
$short is Programming PHP rocks!
Long is Programming PHP rocks!
```

After the assignment, the two variables are alternate names for the same value. Unsetting a variable that is aliased does not affect other names for that variable's value, though:

```
$white = "snow";
$black =& $white;
unset($white);
print $black;
snow
```

Functions can return values by reference

```
function &ret_ref() {     // note the &
  $var = "PHP";
  return $var;
}
$v =& ret_ref();          // note the &
```

Variable Scope

The scope of a variable, which is controlled by the location of the variable's declaration, determines those parts of the program that can access it. There are four types of variable scope in PHP: local, global, static, and function parameters.

Local Scope

A variable declared in a function is local to that function. That is, it is visible only to code in that function (including nested function definitions); it is not accessible outside the function. In addition, by default, variables defined outside a function (called global variables) are not accessible inside the function. For example, here's a function that updates a local variable instead of a global variable:

```
function update_counter ( ) {
 $counter++;
}
$counter = 10;
update_counter( );
echo $counter;
10
```

The $counter inside the function is local to that function, because we haven't said otherwise. The function increments its private $counter, whose value is thrown away when the subroutine ends. The global $counter remains set at 10.

Only functions can provide local scope. Unlike in other languages, in PHP you can't create a variable whose scope is a loop, conditional branch, or other type of block.

Global Scope

Variables declared outside a function are global. That is, they can be accessed from any part of the program. However, by default, they are not available inside functions. To allow a function to access a global variable, you can use the global keyword inside the function to declare the variable within the function. Here's how we can rewrite the update_counter( ) function to allow it to access the global $counter variable:

```
function update_counter ( ) {
   global $counter;
   $counter++;
}
$counter = 10;
update_counter( );
echo $counter;
11
```

A more cumbersome way to update the global variable is to use PHP's $GLOBALS array instead of accessing the variable directly:

```
function update_counter ( ) {
   $GLOBALS[counter]++;
}
$counter = 10;
update_counter( );
echo $counter;
11
```

Static Variables

A static variable retains its value between calls to a function but is visible only within that function. You declare a variable static with the static keyword. For example:

```
function update_counter ( ) {
 static $counter = 0;
 $counter++;
 echo “Static counter is now $counter\n”;
}
$counter = 10;
update_counter( );
update_counter( );
echo “Global counter is $counter\n”;
Static counter is now 1
Static counter is now 2
Global counter is 10
```

Function Parameters

A function definition can have named parameters:

```
function greet ($name) {
  echo “Hello, $name\n”;
}
greet(“Janet”);
Hello, Janet
```

Function parameters are local, meaning that they are available only inside their functions. In this case, $name is inaccessible from outside greet( ).

Garbage Collection

PHP uses reference counting and copy-on-write to manage memory. Copy-on-write ensures that memory isn’t wasted when you copy values between variables, and reference counting ensures that memory is returned to the operating system when it is no longer needed.

To understand memory management in PHP, you must first understand the idea of a symbol table . There are two parts to a variable—its name (e.g., $name), and its value (e.g., “Fred”). A symbol table is an array that maps variable names to the positions of their values in memory.

When you copy a value from one variable to another, PHP doesn’t get more memory for a copy of the value. Instead, it updates the symbol table to say “both

of these variables are names for the same chunk of memory." So the following code doesn't actually create a new array:

```
$worker = array("Fred", 35, "Wilma");
$other = $worker;                    // array isn't copied
```

If you then modify either copy, PHP allocates the memory and makes the copy:

```
$worker[1] = 36;                     // array is copied, value changed
```

By delaying the allocation and copying, PHP saves time and memory in a lot of situations. This is copy-on-write.

Each value pointed to by a symbol table has a reference count, a number that represents the number of ways there are to get to that piece of memory. After the initial assignment of the array to $worker and $worker to $other, the array pointed to by the symbol table entries for $worker and $other has a reference count of 2. In other words, that memory can be reached two ways: through $worker or $other. But after $worker[1] is changed, PHP creates a new array for $worker, and the reference count of each of the arrays is only 1.

[1]It is actually 3 if you are looking at the reference count from the C API, but for the purposes of this explanation and from a user-space perspective, it is easier to think of it as 2.

When a variable goes out of scope (as a function parameter or local variable does at the end of a function), the reference count of its value is decreased by one. When a variable is assigned a value in a different area of memory, the reference count of the old value is decreased by one. When the reference count of a value reaches 0, its memory is freed. This is reference counting.

Reference counting is the preferred way to manage memory. Keep variables local to functions, pass in values that the functions need to work on, and let reference counting take care of freeing memory when it's no longer needed. If you do insist on trying to get a little more information or control over freeing a variable's value, use the isset( ) and unset( ) functions.

To see if a variable has been set to something, even the empty string, use isset( ):

```
$s1 = isset($name);                  // $s1 is false
$name = "Fred";
$s2 = isset($name);                  // $s2 is true
```

Use unset( ) to remove a variable's value:

```
$name = "Fred";
unset($name);                        // $name is NULL
```

4.3 PHP FILE HANDLING

KEY NOTES

Definition

File handling in PHP refers to the ability to read, write, manipulate, and manage files on the server or local file system using PHP scripts. PHP provides a range of functions and features for performing various file-related operations, making it versatile for tasks such as reading configuration files, logging data, uploading files, and more.

Error & Exception Handling

Error handling is the process of catching errors raised by your program and then taking appropriate action. If you would handle errors properly then it may lead to many unforeseen consequences. Exceptions are used to change the normal flow of a script if a specified error occurs.

Expressions and Operators

Expressions that result in Boolean values such as comparison operations can be combined to form more complex expressions. This is handled through logical operators. These operators are used for the logical combinations AND, OR, XOR, and logical NOT. For AND combinations, the & or && logical operators are used.

Learning Objective

- Define how to using die() function
- Describe custom error handling function
- Explain exceptions handling

Definition

File handling is an important part of any web application. You often need to open and process a file for different tasks.

PHP supports file handling which is used to read, write and append data in file. This chapter will demonstrate to you following PHP functions related to files:

- File Opening
- File Reading
- File Writing
- File Closing

File Opening in PHP

fopen() function is used to open a file in PHP. Its required two arguments, first the file name and then file opening mode.

The file may be opened in one of the following modes:	
r	Read only. Starts at the beginning of the file
r+	Read/Write. Starts at the beginning of the file
w	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist
a	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist
a+	Read/Append. Preserves file content by writing to the end of the file
x	Write only. Creates a new file. Returns FALSE and an error if file already exists
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists

Example:

```
<?php
$fileName = "/doc/myFile.txt";
$fp = fopen($fileName,"r");
if( $fp == false )
{
  echo ( "Error in opening file" );
  exit();
}
?>
```

If an attempt to open a file fails then fopen returns a false value, otherwise it returns a file pointer which is used for further reading or writing of that file.

File Reading in PHP

Once a file is opened using fopen() function then it can be read by a function called fread(). This function requires two arguments. The file pointer and length in bytes of the file must be expressed.

The files size can be calculated using the filesize() function which takes the file name as its argument and returns the size of the file in bytes.

Example:

```
<?php
$fileName = "/doc/myFile.txt";
$fp = fopen($fileName,"r");
```

```
if( $fp == false )
{
  echo ( “Error in opening file” );
  exit();
}

$fileSize = filesize( $fileName );
$fileData = fread( $fp, $fileSize );
?>
```

Reading a File Line by Line

The fgets() function is used to read a single line from a file and after a call to this function the file pointer has moved to the next line.

Example:

```
<?php
$fileName = “/doc/myFile.txt”;
$fp = fopen($fileName,”r”);
if( $fp == false )
{
  echo ( “Error in opening file” );
  exit();
}

while(!feof($fp))
{
  echo fgets($fp). “<br>”;
}
?>
```

File Writing in PHP

Using the PHP fwrite() function, a new file can be written or text can be appended to an existing file. This function requires two arguments, first specifying a file pointer and the string of data that is to be written. Optionally a third integer argument can be used to specify the length of the data to write. If the third argument is used, file writing would will stop after the specified length has been reached.

Example:

```
<?php
$fileName = "/doc/myFile.txt";
$fp = fopen($fileName,"w");
if( $fp == false )
{
  echo ( "Error in opening file" );
  exit();
}
fwrite( $fp, "This is a sample text to write\n" );
?>
```

Closing a File in PHP

In PHP it is not system critical to close all your files after using them because, file will auto close after PHP code finishes execution.

You can close file, using fclose() function.

Example:

```
<?php
$fileName = "/doc/myFile.txt";
$fp = fopen($fileName,"w");
if( $fp == false )
{
  echo ( "Error in opening file" );
  exit();
}

//some code to be executed

fclose( $fp );
?>
```

Error handling refers to the anticipation, detection, and resolution of programming, application, and communications errors. Specialized programs, called error handlers, are available for some applications.

The fclose() function requires a file pointer as its argument and then returns true when the closure succeeds or false if it fails.

Error & Exception Handling

Error handling is the process of catching errors raised by your program and then taking appropriate action. If you would handle errors properly then it may lead to many unforeseen consequences.

Its very simple in PHP to handle an errors.

Using die() function

While writing your PHP program you should check all possible error condition before going ahead and take appropriate action when required.

```
<?php
  if(!file_exists("/tmp/test.txt")) {
    die("File not found");
  }else {
    $file = fopen("/tmp/test.txt","r");
    print "Opend file sucessfully";
  }
  // Test of the code here.
?>
```

This way you can write an efficient code. Using above technique you can stop your program whenever it errors out and display more meaningful and user friendly message.

Defining Custom Error Handling Function

You can write your own function to handling any error. PHP provides you a framework to define error handling function.

This function must be able to handle a minimum of two parameters (error level and error message) but can accept up to five parameters (optionally: file, line-number, and the error context) –

Syntax

```
error_function(error_level,error_message, error_file,error_line,error_context);
```

Sr.No	Parameter & Description
1	**error_level** Required - Specifies the error report level for the user-defined error. Must be a value number.

2	**error_message** Required - Specifies the error message for the user-defined error
3	**error_file** Optional - Specifies the file name in which the error occurred
4	**error_line** Optional - Specifies the line number in which the error occurred
5	**error_context** Optional - Specifies an array containing every variable and their values in use when the error occurred

Possible Error levels

These error report levels are the different types of error the user-defined error handler can be used for. These values cab used in combination using | operator

Sr.No	Constant & Description	Value
1	**.E_ERROR** Fatal run-time errors. Execution of the script is halted	1
2	**E_WARNING** Non-fatal run-time errors. Execution of the script is not halted	2
3	**E_PARSE** Compile-time parse errors. Parse errors should only be generated by the parser.	4
4	**E_NOTICE** Run-time notices. The script found something that might be an error, but could also happen when running a script normally	8
5	**E_CORE_ERROR** Fatal errors that occur during PHP's initial start-up.	16
6	**E_CORE_WARNING** Non-fatal run-time errors. This occurs during PHP's initial start-up.	32
7	**E_USER_ERROR** Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()	256

8	**E_USER_WARNING** Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()	512
9	**E_USER_NOTICE** User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()	1024
10	**E_STRICT** Run-time notices. Enable to have PHP suggest changes to your code which will ensure the best interoperability and forward compatibility of your code.	2048
11	**E_RECOVERABLE_ERROR** Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())	4096
12	**E_ALL** All errors and warnings, except level E_STRICT (E_STRICT will be part of E_ALL as of PHP 6.0)	8191

All the above error level can be set using following PHP built-in library function where level cab be any of the value defined in above table.

```
int error_reporting ( [int $level] )
```

Following is the way you can create one error handling function:

```
<?php
  function handleError($errno, $errstr,$error_file,$error_line) {
    echo "<b>Error:</b> [$errno] $errstr - $error_file:$error_line";
    echo "<br />";
    echo "Terminating PHP Script";

    die();
  }
?>
```

Once you define your custom error handler you need to set it using PHP built-in library set_error_handler function. Now lets examine our example by calling a function which does not exist.

```
<?php
  error_reporting( E_ERROR );
```

```
        function handleError($errno, $errstr,$error_file,$error_
line) {
        echo "<b>Error:</b> [$errno] $errstr - $error_file:$error_
line";
          echo "<br />";
          echo "Terminating PHP Script";

          die();
        }

      //set error handler
      set_error_handler("handleError");

      //trigger error
      myFunction();
    ?>
```

Exceptions Handling

PHP 5 has an exception model similar to that of other programming languages. Exceptions are important and provides a better control over error handling.

Lets explain there new keyword related to exceptions:

- *Try*: A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown".
- *Throw*: This is how you trigger an exception. Each "throw" must have at least one "catch".
- *Catch*: A "catch" block retrieves an exception and creates an object containing the exception information.

Programming languages provide a set of rules and syntax that programmers use to write code, which is then translated into machine-readable instructions that computers can execute.

When an exception is thrown, code following the statement will not be executed, and PHP will attempt to find the first matching catch block. If an exception is not caught, a PHP Fatal Error will be issued with an "Uncaught Exception ...

- An exception can be thrown, and caught ("catched") within PHP. Code may be surrounded in a try block.

- Each try must have at least one corresponding catch block. Multiple catch blocks can be used to catch different classes of exceptions.
- Exceptions can be thrown (or re-thrown) within a catch block.

Example

Following is the piece of code, copy and paste this code into a file and verify the result.

```
<?php
  try {
    $error = 'Always throw this error';
    throw new Exception($error);

    // Code following an exception is not executed.
    echo 'Never executed';
  }catch (Exception $e) {
    echo 'Caught exception: ',  $e->getMessage(), "\n";
  }

  // Continue execution
  echo 'Hello World';
?>
```

In the above example $e->getMessage function is used to get error message. There are following functions which can be used from Exception class.

- getMessage() − message of exception
- getCode() − code of exception
- getFile() − source filename
- getLine() − source line
- getTrace() − n array of the backtrace()
- getTraceAsString() − formated string of trace

Creating Custom Exception Handler

You can define your own custom exception handler. Use following function to set a user-defined exception handler function.

```
string set_exception_handler ( callback $exception_handler )
```

Here exception_handler is the name of the function to be called when an uncaught exception occurs. This function must be defined before calling set_exception_handler().

Example

```
<?php
  function exception_handler($exception) {
    echo “Uncaught exception: “ , $exception->getMessage(), “\n”;
  }

  set_exception_handler(‘exception_handler’);
  throw new Exception(‘Uncaught Exception’);

  echo “Not Executed\n”;
?>
```

Expressions and Operators

An expression is a bit of PHP that can be evaluated to produce a value. The simplest expressions are literal values and variables. A literal value evaluates to itself, while a variable evaluates to the value stored in the variable. More complex expressions can be formed using simple expressions and operators.

An operator takes some values (the operands) and does something (for instance, adds them together). Operators are written as punctuation symbols—for instance, the + and - familiar to us from math. Some operators modify their operands, while most do not.

Table 4.2 summarizes the operators in PHP, many of which were borrowed from C and Perl. The column labeled “P” gives the operator’s precedence; the operators are listed in precedence order, from highest to lowest. The column labeled “A” gives the operator’s associativity, which can be L (left-to-right), R (right-to-left), or N (non-associative).

Table 4.2: PHP operators

P	A	Operator	Operation
19	N	new	Create new object
18	R	[	Array subscript
17	R	!	Logical NOT
	R	~	Bitwise NOT
	R	++	Increment
	R	--	Decrement
	R	(int), (double), (string), (array), (object)	Cast
	R	@	Inhibit errors
16	L	*	Multiplication

	L	/	Division
	L	%	Modulus
15	L	+	Addition
	L	-	Subtraction
	L	.	String concatenation
14	L	<<	Bitwise shift left
	L	>>	Bitwise shift right
13	N	<, <=	Less than, less than or equal
	N	>, >=	Greater than, greater than or equal
12	N	==	Value equality
	N	!=, <>	Inequality
	N	===	Type and value equality
	N	!==	Type and value inequality
11	L	&	Bitwise AND
10	L	^	Bitwise XOR
9	L	|	Bitwise OR
8	L	&&	Logical AND
7	L	||	Logical OR
6	L	?:	Conditional operator
5	L	=	Assignment
	L	+=, -=, *=, /=, .=, %=, &=, |=, ^=, ~=, <<=, >>=	Assignment with operation
4	L	and	Logical AND
3	L	xor	Logical XOR
2	L	or	Logical OR
1	L	,	List separator

Number of Operands

Most operators in PHP are binary operators; they combine two operands (or expressions) into a single, more complex expression. PHP also supports a number of unary operators, which convert a single expression into a more complex expression. Finally, PHP supports a single ternary operator that combines three expressions into a single expression.

Operator Precedence

The order in which operators in an expression are evaluated depends on their relative precedence. For example, you might write:

```
2 + 4 * 3
```

As you can see in Table 4.2, the addition and multiplication operators have different precedence, with multiplication higher than addition. So the multiplication happens

before the addition, giving 2 + 12, or 14, as the answer. If the precedence of addition and multiplication were reversed, 6 * 3, or 18, would be the answer.

To force a particular order, you can group operands with the appropriate operator in parentheses. In our previous example, to get the value 18, you can use this expression:

```
(2 + 4) * 3
```

It is possible to write all complex expressions (expressions containing more than a single operator) simply by putting the operands and operators in the appropriate order so that their relative precedence yields the answer you want. Most programmers, however, write the operators in the order that they feel makes the most sense to programmers, and add parentheses to ensure it makes sense to PHP as well. Getting precedence wrong leads to code like:

```
$x + 2 / $y >= 4 ? $z : $x << $z
```

This code is hard to read and is almost definitely not doing what the programmer expected it to do.

One way many programmers deal with the complex precedence rules in programming languages is to reduce precedence down to two rules:

- Multiplication and division have higher precedence than addition and subtraction.
- Use parentheses for anything else.

Operator Associativity

Associativity defines the order in which operators with the same order of precedence are evaluated. For example, look at:

```
2 / 2 * 2
```

The division and multiplication operators have the same precedence, but the result of the expression depends on which operation we do first:

```
2/(2*2)     // 0.5
(2/2)*2     // 2
```

The division and multiplication operators are left-associative; this means that in cases of ambiguity, the operators are evaluated from left to right. In this example, the correct result is 2.

Implicit Casting

Many operators have expectations of their operands—for instance, binary math operators typically require both operands to be of the same type. PHP's variables can store integers, floating-point numbers, strings, and more, and to keep as much of the type details away from the programmer as possible, PHP converts values from one type to another as necessary.

The conversion of a value from one type to another is called casting. This kind of implicit casting is called type juggling in PHP. The rules for the type juggling done by arithmetic operators are shown in Table 4.3.

Table 4.3: Implicit casting rules for binary arithmetic operations

Type of first operand	Type of second operand	Conversion performed
Integer	Floating point	The integer is converted to a floating-point number
Integer	String	The string is converted to a number; if the value after conversion is a floating-point number, the integer is converted to a floating-point number
Floating point	String	The string is converted to a floating-point number

Some other operators have different expectations of their operands, and thus have different rules. For example, the string concatenation operator converts both operands to strings before concatenating them:

```
3 . 2.74     // gives the string 32.74
```

You can use a string anywhere PHP expects a number. The string is presumed to start with an integer or floating-point number. If no number is found at the start of the string, the numeric value of that string is 0. If the string contains a period (.) or upper- or lowercase e, evaluating it numerically produces a floating-point number. For example:

```
"9 Lives" - 1;              // 8 (int)
"3.14 Pies" * 2;            // 6.28 (float)
"9 Lives." - 1;             // 8 (float)
"1E3 Points of Light" + 1;  // 1001 (float)
```

Arithmetic Operators

The **arithmetic operators** are operators you'll recognize from everyday use. Most of the arithmetic operators are binary; however, the arithmetic negation and arithmetic assertion operators are unary. These operators require numeric values, and non-numeric values are converted into numeric values by the rules. The arithmetic operators are:

Arithmetic operators take numerical values (either literals or variables) as their operands and return a single numerical value. The standard arithmetic operators are addition (+), subtraction (-), multiplication (*), and division (/).

Addition (+)

The result of the addition operator is the sum of the two operands.

Subtraction (-)

The result of the subtraction operator is the difference between the two operands; i.e., the value of the second operand subtracted from the first.

Multiplication (*)

The result of the multiplication operator is the product of the two operands. For example, 3 * 4 is 12.

Division (/)

The result of the division operator is the quotient of the two operands. Dividing two integers can give an integer (e.g., 4/2) or a floating-point result (e.g., 1/2).

Modulus (%)

The modulus operator converts both operands to integers and returns the remainder of the division of the first operand by the second operand. For example, 10 % 6 is 4.

Arithmetic negation (-)

The arithmetic negation operator returns the operand multiplied by -1, effectively changing its sign. For example, -(3 - 4) evaluates to 1. Arithmetic negation is different from the subtraction operator, even though they both are written as a minus sign. Arithmetic negation is always unary and before the operand. Subtraction is binary and between its operands.

Arithmetic assertion (+)

The arithmetic assertion operator returns the operand multiplied by +1, which has no effect. It is used only as a visual cue to indicate the sign of a value. For example, +(3 - 4) evaluates to -1, just as (3 - 4) does.

String Concatenation Operator

Manipulating strings is such a core part of PHP applications that PHP has a separate string concatenation operator (.). The concatenation operator appends the righthand operand to the lefthand operand and returns the resulting string. Operands are first converted to strings, if necessary. For example:

```
 $n = 5;
$s = ‘There were ‘ . $n . ‘ ducks.’;
// $s is ‘There were 5 ducks’
```

Autoincrement and Autodecrement Operators

In programming, one of the most common operations is to increase or decrease the value of a variable by one. The unary autoincrement (++) and autodecrement (--) operators provide shortcuts for these common operations. These operators are unique in that they work only on variables; the operators change their operands' values as well as returning a value.

There are two ways to use autoincrement or autodecrement in expressions. If you put the operator in front of the operand, it returns the new value of the operand (incremented or decremented). If you put the operator after the operand, it returns the original value of the operand (before the increment or decrement). Table 4.4 lists the different operations.

Table 4.4: Autoincrement and autodecrement operations

Operator	Name	Value returned	Effect on $var
$var++	Post-increment	$var	Incremented
++$var	Pre-increment	$var + 1	Incremented
$var--	Post-decrement	$var	Decremented
--$var	Pre-decrement	$var - 1	Decremented

These operators can be applied to strings as well as numbers. Incrementing an alphabetic character turns it into the next letter in the alphabet. As illustrated in Table 4.5, incrementing "z" or "Z" wraps it back to "a" or "Z" and increments the previous character by one, as though the characters were in a base-26 number system.

Table 4.5: Autoincrement with letters

Incrementing this	Gives this
"a"	"b"
"z"	"aa"
"spaz"	"spba"
"K9"	"L0"
"42"	

Comparison Operators

As their name suggests, comparison operators compare operands. The result is always either true, if the comparison is truthful, or false, otherwise.

Operands to the comparison operators can be both numeric, both string, or one numeric and one string. The operators check for truthfulness in slightly different ways based on the types and values of the operands, either using strictly numeric comparisons or using lexicographic (textual) comparisons. Table 4.6 outlines when each type of check is used.

Table 4.6: Type of comparision performed by the comparision operators

First operand	Second operand	Comparison
Number	Number	Numeric
String that is entirely numeric	String that is entirely numeric	Numeric
String that is entirely numeric	Number	Numeric
String that is not entirely numeric	Number	Lexicographic
String that is entirely numeric	String that is not entirely numeric	Lexicographic
String that is not entirely numeric	String that is not entirely numeric	Lexicographic

One important thing to note is that two numeric strings are compared as if they were numbers. If you have two strings that consist entirely of numeric characters and you need to compare them lexicographically, use the strcmp( ) function.

The comparison operators are:

Equality (==)

If both operands are equal, this operator returns true; otherwise, it returns false.

Identical (===)

If both operands are equal and are of the same type, this operator returns true; otherwise, it returns false. Note that this operator does not do implicit type casting. This operator is useful when you don't know if the values you're comparing are of the same type. Simple comparison may involve value conversion. For instance, the strings "0.0" and "0" are not equal. The == operator says they are, but === says they are not.

Inequality (!= or <>)

If both operands are not equal, this operator returns true; otherwise, it returns false.

Not identical (!==)

If both operands are not equal, or they are not of the same type, this operator returns true; otherwise, it returns false.

Greater than (>)

If the lefthand operator is greater than the righthand operator, this operator returns true; otherwise, it returns false.

Greater than or equal to (>=)

If the lefthand operator is greater than or equal to the righthand operator, this operator returns true; otherwise, it returns false.

Less than (<)

If the lefthand operator is less than the righthand operator, this operator returns true; otherwise, it returns false.

Less than or equal to (<=)

If the lefthand operator is less than or equal to the righthand operator, this operator returns true; otherwise, it returns false.

Bitwise Operators

The bitwise operators act on the binary representation of their operands. Each operand is first turned into a binary representation of the value, as described in the bitwise negation operator entry in the following list. All the bitwise operators work on numbers as well as strings, but they vary in their treatment of string operands of different lengths. The bitwise operators are:

Bitwise negation (~)

The bitwise negation operator changes 1s to 0s and 0s to 1s in the binary representations of the operands. Floating-point values are converted to integers before the operation takes place. If the operand is a string, the resulting value is a string the same length as the original, with each character in the string negated.

Bitwise AND (&)

The bitwise AND operator compares each corresponding bit in the binary representations of the operands. If both bits are 1, the corresponding bit in the result is 1; otherwise, the corresponding bit is 0. For example, 0755 & 0671 is 0651. This is a bit easier to understand if we look at the binary representation. Octal 0755 is binary 111101101, and octal 0671 is binary 110111001. We can the easily see which bits are on in both numbers and visually come up with the answer:

```
  111101101
& 110111001
  ---------
  110101001
```

The binary number 110101001 is octal 0651. You can use the PHP functions bindec( ), decbin( ), octdec( ), and decoct( ) to convert numbers back and forth when you are trying to understand binary arithmetic.

Here's a tip: split the binary number up into three groups. 6 is binary 110, 5 is binary 101, and 1 is binary 001; thus, 0651 is 110101001.

If both operands are strings, the operator returns a string in which each character is the result of a bitwise AND operation between the two corresponding characters in the operands. The resulting string is the length of the shorter of the two operands; trailing

extra characters in the longer string are ignored. For example, “wolf” & “cat” is “cad”.

Bitwise OR (|)

The bitwise OR operator compares each corresponding bit in the binary representations of the operands. If both bits are 0, the resulting bit is 0; otherwise, the resulting bit is 1. For example, 0755 | 020 is 0775.

If both operands are strings, the operator returns a string in which each character is the result of a bitwise OR operation between the two corresponding characters in the operands. The resulting string is the length of the longer of the two operands, and the shorter string is padded at the end with binary 0s. For example, “pussy” | “cat” is “suwsy”.

Bitwise XOR (^)

The bitwise XOR operator compares each corresponding bit in the binary representation of the operands. If either of the bits in the pair, but not both, is 1, the resulting bit is 1; otherwise, the resulting bit is 0. For example, 0755 ^ 023 is 776.

If both operands are strings, this operator returns a string in which each character is the result of a bitwise XOR operation between the two corresponding characters in the operands. If the two strings are different lengths, the resulting string is the length of the shorter operand, and extra trailing characters in the longer string are ignored. For example, “big drink” ^ “AA” is “#(“.

Left shift (<<)

The left shift operator shifts the bits in the binary representation of the lefthand operand left by the number of places given in the righthand operand. Both operands will be converted to integers if they aren’t already. Shifting a binary number to the left inserts a 0 as the rightmost bit of the number and moves all other bits to the left one place. For example, 3 << 1 (or binary 11 shifted one place left) results in 6 (binary 110).

Note that each place to the left that a number is shifted results in a doubling of the number. The result of left shifting is multiplying the lefthand operand by 2 to the power of the righthand operand.

Right shift (>>)

The right shift operator shifts the bits in the binary representation of the lefthand operand right by the number of places given in the righthand operand. Both operands will be converted to integers if they aren’t already. Shifting a binary number to the right inserts a 0 as the leftmost bit of the number and moves all other bits to the right one place.

The rightmost bit is discarded. For example, 13 >> 1 (or binary 1101) shifted one place right results in 6 (binary 110).

Logical Operators

Logical operators provide ways for you to build complex logical expressions. Logical operators treat their operands as Boolean values and return a Boolean value. There are both punctuation and English versions of the operators (|| and or are the same operator). The logical operators are:

Logical AND (&&, and)

The result of the logical AND operation is true if and only if both operands are true; otherwise, it is false. If the value of the first operand is false, the logical AND operator knows that the resulting value must also be false, so the righthand operand is never evaluated. This process is called short-circuiting, and a common PHP idiom uses it to ensure that a piece of code is evaluated only if something is true. For example, you might connect to a database only if some flag is not false:

```
$result = $flag and mysql_connect( );
```

The && and and operators differ only in their precedence.

Logical OR (||, or)

The result of the logical OR operation is true if either operand is true; otherwise, the result is false. Like the logical AND operator, the logical OR operator is short-circuited. If the lefthand operator is true, the result of the operator must be true, so the righthand operator is never evaluated. A common PHP idiom uses this to trigger an error condition if something goes wrong. For example:

```
$result = fopen($filename) or exit( );
```

The || and or operators differ only in their precedence.

Logical XOR (xor)

The result of the logical XOR operation is true if either operand, but not both, is true; otherwise, it is false.

Logical negation (!)

The logical negation operator returns the Boolean value true if the operand evaluates to false, and false if the operand evaluates to true.

Casting Operators

Although PHP is a weakly typed language, there are occasions when it's useful to consider a value as a specific type. The casting operators, (int) , (float), (string), (bool), (array), and (object), allow you to force a value into a particular type. To use a casting operator, put the operator to the left of the operand. Table 4.7 lists the casting operators, synonymous operands, and the type to which the operator changes the value.

Table 4.7: PHP casting operators

Operator	Synonymous operators	Changes type to
(int)	(integer)	Integer
(float)	(real)	Floating point
(string)		String
(bool)	(boolean)	Boolean
(array)		Array
(object)		Object

Casting affects the way other operators interpret a value, rather than changing the value in a variable. For example, the code:

```
$a = "5";
$b = (int) $a;
```

assigns $b the integer value of $a; $a remains the string "5". To cast the value of the variable itself, you must assign the result of a cast back into the variable:

```
$a = "5"
$a = (int) $a; // now $a holds an integer
```

Not every cast is useful: casting an array to a numeric type gives 1, and casting an array to a string gives "Array" (seeing this in your output is a sure sign that you've printed a variable that contains an array).

Casting an object to an array builds an array of the properties, mapping property names to values:

```
 class Person {
  var $name = "Fred";
  var $age  = 35;
}
$o = new Person;
$a = (array) $o;
print_r($a);
Array
(
   [name] => Fred
   [age] => 35
)
```

You can cast an array to an object to build an object whose properties correspond to the array's keys and values. For example:

```
$a = array('name' => 'Fred', 'age' => 35, 'wife' => 'Wilma');
```

```
$o = (object) $a;
echo $o->name;
Fred
```

Keys that aren't valid identifiers, and thus are invalid property names, are inaccessible but are restored when the object is cast back to an array.

Assignment Operators

Assignment operators store or update values in variables. The autoincrement and autodecrement operators we saw earlier are highly specialized assignment operators—here we see the more general forms. The basic assignment operator is =, but we'll also see combinations of assignment and binary operations, such as += and &=.

Assignment

The basic assignment operator (=) assigns a value to a variable. The lefthand operand is always a variable. The righthand operand can be any expression—any simple literal, variable, or complex expression. The righthand operand's value is stored in the variable named by the lefthand operand.

Because all operators are required to return a value, the assignment operator returns the value assigned to the variable. For example, the expression $a = 5 not only assigns 5 to $a, but also behaves as the value 5 if used in a larger expression. Consider the following expressions:

```
$a = 5;
$b = 10;
$c = ($a = $b);
```

The expression $a = $b is evaluated first, because of the parentheses. Now, both $a and $b have the same value, 10. Finally, $c is assigned the result of the expression $a = $b, which is the value assigned to the lefthand operand (in this case, $a). When the full expression is done evaluating, all three variables contain the same value, 10.

Assignment with Operation

In addition to the basic assignment operator, there are several assignment operators that are convenient shorthand. These operators consist of a binary operator followed directly by an equals sign, and their effect is the same as performing the operation with the operands, then assigning the resulting value to the lefthand operand. These assignment operators are:

Plus-equals (+=)

Adds the righthand operand to the value of the lefthand operand, then assigns the result to the lefthand operand. $a += 5 is the same as $a = $a + 5.

Minus-equals (-=)

Subtracts the righthand operand from the value of the lefthand operand, then assigns the result to the lefthand operand.

Divide-equals (/=)

Divides the value of the lefthand operand by the righthand operand, then assigns the result to the lefthand operand.

Multiply-equals (*=)

Multiplies the righthand operand with the value of the lefthand operand, then assigns the result to the lefthand operand.

Modulus-equals (%=)

Performs the modulus operation on the value of the lefthand operand and the righthand operand, then assigns the result to the lefthand operand.

Bitwise-XOR-equals (^=)

Performs a bitwise XOR on the lefthand and righthand operands, then assigns the result to the lefthand operand.

Bitwise-AND-equals (&=)

Performs a bitwise AND on the value of the lefthand operand and the righthand operand, then assigns the result to the lefthand operand.

Bitwise-OR-equals (|=)

Performs a bitwise OR on the value of the lefthand operand and the righthand operand, then assigns the result to the lefthand operand.

Concatenate-equals (.=)

Concatenates the righthand operand to the value of the lefthand operand, then assigns the result to the lefthand operand.

Miscellaneous Operators

The remaining PHP operators are for error suppression, executing an external command, and selecting values:

Error suppression (@)

Some operators or functions can generate error messages. The error suppression operator is used to prevent these messages from being created.

Execution (`...`)

The backtick operator executes the string contained between the backticks as a shell command and returns the output. For example:

```
$listing = `ls -ls /tmp`;

echo $listing;
```

Conditional (?:)

The conditional operator is, depending on the code you look at, either the most overused or most underused operator. It is the only ternary (three-operand) operator and is therefore sometimes just called the ternary operator.

The conditional operator evaluates the expression before the ?. If the expression is true, the operator returns the value of the expression between the ? and :; otherwise, the operator returns the value of the expression after the :. For instance:

```
<a href="<?= $url ?>"><?= $linktext ? $linktext : $url ?></a>
```

If text for the link $url is present in the variable $linktext, it is used as the text for the link; otherwise, the URL itself is displayed.



TECHNIQUE 13

WRITE THE PHP SCRIPTS

PHP is a server scripting language used to make web pages interactive. It became widely popular due to its ease of use, interactivity within web pages, and integration with HTML. Think of what happens when a page is edited on this website. Behind this process are many, perhaps hundreds, of PHP scripts controlling how web pages change based on a variety of circumstances. This article will teach you how to write a few very simple PHP scripts so that you can get a basic understanding of how PHP works.

- ☐ We already practice this technique
- ☐ This techniques would suitable or not for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 14

OPEN A PHP FILE

PHP is a scripting language utilized most often by web developers in order to offer dynamic content, which means that the content on the web page can change based on the parameters set in the PHP code. You could make this as simple as changing the background image based on the user's time of day. Or, it could be as complex as creating a publication workflow that allows different users to see a submissions based on where that submission is in the process. You can easily view PHP files on both Mac and Windows computers with a code editing software. You can also run your PHP file using a server you pay for, or locally on your computer through XAMPP.

- ☐ We already practice this technique
- ☐ This techniques would suitable or not for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 15

WHY PHP

PHP remains the most widespread and popular server-side programming language on the web. It is installed by most web hosts, has a simple learning curve, close ties with the MySQL database, and an excellent collection of libraries to cut your development time. PHP may not be perfect, but it should certainly be considered for your next web application. Both Yahoo and Facebook use it with great success.

- □ We already practice this technique
- □ This techniques would suitable or not for our operation
- □ We practiced this in the past and need to jumpstart
- □ We should implement this technique



TECHNIQUE 16

ARRAYS IN PHP

Arrays are fundamental data structures in PHP that allow you to store and manage multiple values under a single variable name. PHP supports various types of arrays, each with its own characteristics and use cases. Arrays can store values of the same data type or a mix of different data types.

- □ We already practice this technique
- □ This techniques would suitable or not for our operation
- □ We practiced this in the past and need to jumpstart
- □ We should implement this technique

SUMMARY

- PHP programmers know no other language. This occasionally causes it to be dismissed as a lesser language, but its growing popularity and the many robust and efficient sites built using it as a structure seem to dispel this myth.
- The PHP language revolves around the central theme of practicality. PHP is about providing the programmer with the necessary tools to get the job done in a quick and efficient fashion.
- PHP is highly effective at accessing and communicating with various database types. This makes it perfect for web-based scripts, such as website content management systems, which need to store data in a database.
- PHP is a web development scripting language that is utilized to create dynamic, text-based website pages. The acronym has developed over time and is now known to refer to the language's use as a PHP Hypertext Preprocessor.
- PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.
- Integers are whole numbers, like 1, 12, and 256. The range of acceptable values varies according to the details of your platform but typically extends from -2,147,483,648 to +2,147,483,647. Specifically, the range is equivalent to the range of the long data type of your C compiler.
- PHP provides operators and functions to compare, disassemble, assemble, search, replace, and trim strings, as well as a host of specialized string functions for working with HTTP, HTML, and SQL encodings. Because there are so many string-manipulation functions.
- PHP uses reference counting and copy-on-write to manage memory. Copy-on-write ensures that memory isn't wasted when you copy values between variables, and reference counting ensures that memory is returned to the operating system when it is no longer needed.
- File handling is an important part of any web application. You often need to open and process a file for different tasks.
- PHP supports file handling which is used to read, write and append data in file.
- Error handling is the process of catching errors raised by your program and then taking appropriate action. If you would handle errors properly then it may lead to many unforeseen consequences.
- An expression is a bit of PHP that can be evaluated to produce a value. The simplest expressions are literal values and variables.
- The arithmetic operators are operators you'll recognize from everyday use. Most of the arithmetic operators are binary; however, the arithmetic negation and arithmetic assertion operators are unary.

- The bitwise negation operator changes 1s to 0s and 0s to 1s in the binary representations of the operands. Floating-point values are converted to integers before the operation takes place. If the operand is a string, the resulting value is a string the same length as the original, with each character in the string negated.

KNOWLEDGE CHECK

1. PHP files have a default file extension of.

a. .html
b. .xml
c. .php
d. .ph

2. A PHP script should start with ___ and end with ___:

a. < php >
b. < ? php ?>
c. < ? ? >
d. < ?php ? >

3. Which of the following type specifier is invalid in print f ( ) functions

a. % a
b. % b
c. % c
d. % d

4. Which of the following function can assign the output to a string variable

a. echo ( )
b. print ( )
c. print f ( )
d. s print f ( )

5. Which of the following data type is not seal or datetype supported by PHP

a. Array
b. String
c. Float
d. Boolean

6. Which of the following is used to start a PHP block?

a. <?
b. <%=

c. <?php

d. {}

7. Which function is used to output text in PHP?

a. print

b. echo

c. write

d. output

8. Which of the following variables stores information submitted by a user through an HTML form?

a. $_POST

b. $_GET

c. $_REQUEST

d. All of the above

REVIEW QUESTIONS

1. What is the concept of PHP?
2. What are the characteristics of PHP?
3. What are the functions of PHP?
4. How are data types used in PHP?
5. How does PHP handle file handling?
6. What is error and exception handling in PHP?

Check Your Result

1 (c)	2 (d)	3 (a)	4 (d)	5 (a)
6. (c)	7. (b)	8. (d)		

REFERENCES

1. Lerdorf, Rasmus (2007-04-26). "PHP on Hormones" (mp3). The Conversations Network. Retrieved 2009-06-22.
2. Lerdorf, Rasmus (1995-06-08). "Announce: Personal Home Page Tools (PHP Tools)". Newsgroup: comp.infosystems.www.authoring.cgi. Retrieved 2006-09-17.
3. Rasmus Lerdorf (Dec 16, 2013). "Re: Flexible function naming". Newsgroup: php.internals. Retrieved December 26, 2013.
4. Kerner, Sean Michael (2008-02-01). "PHP 4 is Dead—Long Live PHP 5". InternetNews. Retrieved 2008-03-16.
5. Trachtenberg, Adam (2004-07-15). "Why PHP 5 Rocks!". O'Reilly. Retrieved 2008-02-22.
6. Christian Seiler; Dmitry Stogov (2008-07-01). "Request for Comments: Lambda functions and closures". php.net. Retrieved 2014-07-07.
7. Darryl Patterson (5 August 2004). "Simplify Business Logic with PHP DataObjects - O'Reilly Media". ibm.com. Retrieved 16 December 2014.
8. Pawel Krawczyk (2013). "So what are the "most critical" application flaws? On new OWASP Top 10". IPSec.pl. Retrieved 2015-04-15.

CHAPTER 5

OVERVIEW OF AJAX

5.1 BASIC CONCEPT OF AJAX

KEY NOTES

Concept

Ajax is a set of Web development techniques using many Web technologies on the client side to create asynchronous Web applications. With Ajax, Web applications can send data to and retrieve from a server asynchronously (in the background) without interfering with the display and behavior of the existing page.

JavaScript

JavaScript often abbreviated as JS, is a high-level, dynamic, weakly typed, object-based, multi-paradigm, and interpreted client-side programming language. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production.

Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

Using AJAX in Web Development

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Learning Objectives

- Learn about browser support
- Focus on impact of ajax
- Able to prepare for the use of AJAX in Web Development

Concept

Ajax refers to JavaScript and XML, technologies that are widely used for creating dynamic and asynchronous web content. While Ajax is not limited to JavaScript and XML technologies, more often than not they are used together by web applications. The focus of this tutorial is on using JavaScript based Ajax functionality in JavaServer Faces web applications.

JavaScript is a dynamic scripting language for web applications. It allows users to add enhanced functionality to user interfaces and allows web pages to interact with clients asynchronously. JavaScript runs mainly on the client side (as in a browser) and thereby reduces server access by clients.

When a JavaScript function sends an asynchronous request from the client to the server, the server sends back a response that is used to update the page's Document Object Model (DOM). This response is often in the format of an XML document. The term Ajax refers to this interaction between the client and server. The server response need not be in XML only; it can also be in other formats, such as JSON. This tutorial does not focus on the response formats.

Ajax enables asynchronous and partial updating of web applications. Such functionality allows for highly responsive web pages that are rendered in near real time. Ajax-based web applications can access server and process information and can also retrieve data without interfering with the display and rendering of the current web page on a client (such as a browser).

Some of the advantages of using Ajax are as follows:

- Form data validation in real time, eliminating the need to submit the form for verification
- Enhanced functionality for web pages, such as user name and password prompts
- Partial update of the web content, avoiding complete page reloads

AJAX stands for Asynchronous JavaScript and XML. Asynchronous means that you can make request to a server via Hypertext Transfer Protocol (HTTP), and process the other data while waiting for the response. It is not a new programming language but it is a group of interrelated web development techniques used on client side to create interactive web applications. It is a way of using existing standards such as JavaScript and XML to create interactive web applications. Browser based applications and platform independent applications use Ajax.

With AJAX, we can move data from web page to server and web applications can send data to and receive data from server without interfering with existing page and without making the user wait while reloading the whole page. This is accomplished through a kind of HTTP request. Web pages which do not use AJAX must reload the entire page to change the contents of the web page. There are many web applications running on Ajax technology such as Gmail, Yahoo, Facebook, Twitter and YouTube

etc. Ajax uses XHTML for content, CSS for presentation and Document Object Model (DOM) for content display.

To learn Ajax you should have basic understanding of the following:

- HTML/XHTML
- JavaScript
- CSS
- DOM (Document Object Model)

Ajax technology has some advantages and disadvantages as mentioned below:

Advantages:

- Ajax allows web applications to interact with data on the **web browser**.
- Ajax can be used to handle several complex web applications, for example Google map.
- Ajax can be used for creating rich web based applications like desktop applications.
- Ajax can be used to update the page data without reloading the whole page. It reduces server load.
- Faster retrieval of the data.
- Ajax is easy to learn and understand. It is based on JavaScript and other technologies such as XML, CSS etc.
- The code is light weight and not very complex.
- Ajax is much responsive, whole page data can be transferred at a time.
- It allows easier navigation and form validation to get proper validation.
- Reduce the traffic level between client and server and also reduce the time consuming on both side responses.

Web browser is a software application for retrieving, presenting and traversing information resources on the World Wide Web.

Disadvantages:

- It is open source. Anyone can view the source code written for Ajax.
- It increases design and development time.
- ActiveX requests are supported only in IE5 and IE6 of old version. This feature is now available in latest version in few browsers.

- Ajax is not fully integrated with any browser. It has browser compatibility issues.
- It is harder to bookmark updated page content.
- The browsers which do not have JavaScript cannot use the application.
- It has low latency connection to the server.
- Debugging is difficult.
- There is problem with browser back button when using Ajax enabled pages.
- It is less secure because JavaScript code is open source. Unauthorized user/ hacker can use code for inferring server side weaknesses.

Characteristics of an Ajax Application

Ajax refers to set of technology and techniques that make web pages to be more interactive like desktop applications. Different from classic web applications, Ajax applications have the following characteristics.

- Ajax makes boundary between web page and application. In classic web application, web page is rendered by browser to display the result and in Ajax application, the server sends HTML page to browser that allows making page to be interactive.
- With Ajax, a web page feels like a desktop application. It responds fast and updates the page by fetching the data from the server in the background.
- In classic web applications web servers serve HTML pages. Some of the pages are static while others are created dynamically at server side. In Ajax applications servers do not need to convert data into HTML page. The server sends data directly to the client.
- Important characteristic of Ajax is in its first letter "A" – Asynchronous. Asynchronous means the user can interact with the application while the browser is communicating with the server.
- Ajax is scripting based approach. Developers need to understand JavaScript and write code in JavaScript. JavaScript is hard to develop, maintain, test and debug.
- An Ajax application reduces network load and provides support for both synchronous and asynchronous communication.
- If user wants to create a secure application and if the browser does not support **JavaScript** then we cannot use Ajax.

Browser Support

Not all browsers support Ajax. Here are some browsers which support Ajax:

- Mozilla Firefox 1.0 and above
- Microsoft Internet Explorer 5 and above

- Safari 1.2 and above
- Netscape Navigator 7 and above

JavaScript is a high-level, dynamic, weakly typed, object-based, multi-paradigm, and interpreted client-side programming language. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production.

What can Ajax do?

Ajax plays an important role in making web 2.0 promises a reality. Some of the features of web 2.0 are:

- We can use the Web as a platform.
- Software can be delivered as a service.
- Cost effective
- Architecture with completely user interaction.

Google map is considered as one of the popular AJAX based applications which allows user to zoom in and zoom out and scroll the map in four directions like a desktop application. User can drag the map on the screen using the mouse pointer and double clicking the center.

Gmail uses Ajax for spell check, new mail check and other functions. Yahoo's Flickr uses Ajax for text editing and tagging interface. Yahoo front page also uses Ajax field. Windows live is application of Ajax technique. It has features such as search, maps, news, email integration, instant messenger etc.

Impact of Ajax on User

Existing Ajax applications can be divided into two types such as

- Partially AJAXed –used to provide functionalities like in.-flickr
- Fully AJAXed –used for functionalities as well as user interface such as -Google Maps and Windows Live.

The fully AJAXed applications are different from traditional web applications. In these applications

- A web page breaks down
- Surfing site or using applications is not just clicking links and loading pages,
- URL remains unchanged in address bar, forward, backward, reload buttons are rendered meaningless.

Using AJAX in Web Development

Using AJAX in web development allows you to create more dynamic and interactive web applications by updating parts of a page without requiring a full page reload. Here's a step-by-

step guide on how to use AJAX in web development: Set Up Your Project: Begin by creating a new HTML file and adding the necessary elements (like buttons, forms, and containers) that you plan to update using AJAX.

Include JavaScript Libraries: Include the JavaScript libraries you intend to use. While you can use the native fetch API, many developers use libraries like jQuery, Axios, or the Fetch API for easier handling of AJAX requests.

For example, with jQuery:

```
<script src=”https://code.jquery.com/jquery-3.6.0.min.js”></script>
```

Write the JavaScript Code: Write the JavaScript code that handles the AJAX requests and updates the page content.

```
$(document).ready(function() {
  // Attach a click event to a button
  $(‘#load-button’).click(function() {
    // Make an AJAX GET request to a server endpoint
    $.ajax({
      url: ‘data.json’, // URL to fetch data from
      method: ‘GET’,
      dataType: ‘json’, // Expected response data type
      success: function(data) {
        // Update a container with the fetched data
        $(‘#content-container’).html(data.content);
      },
      error: function(xhr, status, error) {
        console.error(‘Error:’, status, error);
      }
    });
  });
});
```

Create a Server Endpoint: You need a server endpoint that responds to AJAX requests. This can be a simple script (e.g., PHP, Node.js) that returns data in JSON or XML format.

For example, in PHP:

```
<?php
$response = array(‘content’ => ‘Updated content from the server’);
header(‘Content-Type: application/json’);
echo json_encode($response);
```

?>

HTML Markup: Create the HTML markup that triggers the AJAX request. This can be a button or any other element with an appropriate event listener.

<button id=”load-button”>Load Content</button>

Initial content

Testing and Debugging: Open your HTML file in a web browser and test the AJAX functionality. Use the browser’s developer tools to monitor network requests and debug any issues.

Remember that AJAX is just one part of a larger web development process. It’s important to consider factors like error handling, security, and performance optimization when using AJAX in your projects. Additionally, modern web development often involves using more advanced techniques like Single Page Applications (SPAs) and frameworks like React, Angular, or Vue.js, which provide more sophisticated ways of handling dynamic content without traditional page reloads.

5.2 RICH INTERNET APPLICATIONS

KEY NOTES

Definition

Rich Internet application is a Web application that has many of the characteristics of desktop application software, typically delivered by way of a site-specific browser, a browser plug-in, an independent sandbox, extensive use of JavaScript, or a virtual machine.

Web service

Web service a standardized way of integrating web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone.

Server Controls

Controls are small building blocks of the graphical user interface, which include text boxes, buttons, check boxes, list boxes, labels, and numerous other tools. Using these tools, the users can enter data, make selections and indicate their preferences.

Learning Objectives

- Define the features of ajax.
- Describe the architecture of microsoft ajax applications.
- Discuss about ajax server architecture.

Definition

Ajax is a way of developing richer and more interactive web applications .These applications are referred to as Rich Internet Applications (RIA). RIA are the **web applications** which provides functionality of the desktop applications. The primary difference between RIA and traditional web applications is quality of interaction between user and the interface. RIA uses controls to improve user's interaction by allowing better error management, efficient interactions and user experience.

RIAs fall into two basic categories:

1. Object oriented based approaches: Java and .Net
2. Scripting based approaches: Ajax and Flash

Features of RIAs are as follows:

- Installation is not required. Only Ajax enabled browsers are required.
- Compatible with existing HTML infrastructure and content.
- Easy to try without the need of additional software.
- Lower development and deployment costs.
- Reduces the load on the server and reduction in network traffic.

Features of Ajax

The main feature of Ajax is it make web-page faster, ajax allow reload only important part of web-page not complete website or web-page.

Web application is a client–server software application in which the client (or user interface) runs in a web browser. Common web applications include webmail, online retail sales, online auctions, wikis, instant messaging services and many other functions.



Important Features of Ajax

- It make web page faster.
- Independent of server technology.
- Increase the Performance of web page.
- No need to pushing on a submit button and reloading the complete website.
- No need to reload the whole page only some part of page is reloaded which is required to reload.
- Apart from obtaining the XMLHTTP object, all processing is same for all browser types, because JavaScript is used.
- Using ajax develop faster and more interactive web applications.
- Not require to completely reload page due to this server use less bandwidth.

Use Microsoft Ajax Features

Web forms applications that use Ajax features offer the following features:

- Familiar interactive UI elements such as progress indicators, tooltips, and pop-up windows.
- Improved efficiency for Web Forms application, because significant parts of a Web page's processing can be performed in the browser.

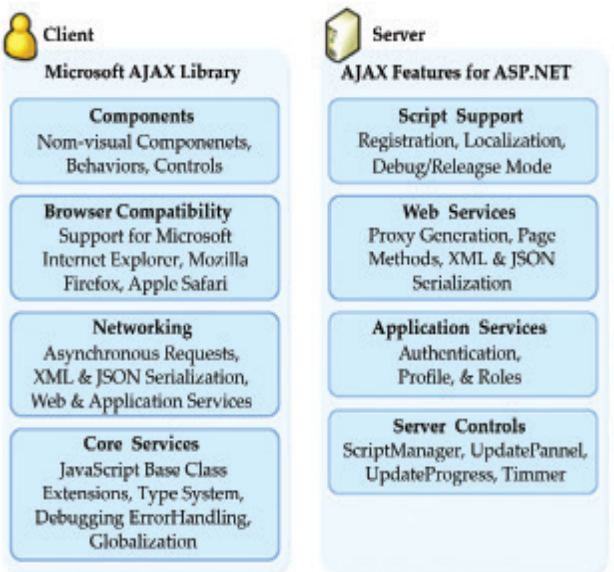
- Partial-page updates that refresh only the parts of the Web page that have changed.
- Client integration with ASP.NET application services for forms authentication, roles, and user profiles.
- Auto-generated proxy classes that simplify calling Web service methods from client script.
- The ability to customize server controls to include client capabilities.
- Support for the most popular browsers, which includes Microsoft Internet Explorer, Mozilla Firefox, and Apple Safari.

Architecture of Microsoft Ajax Applications

Web forms application simplifies the process of creating dynamic and interactive web applications by abstracting much of the low-level HTML, JavaScript, and server-side coding.

A Microsoft Ajax Web application consists of either a client-only solution or a client and server solution. A client-only solution uses Microsoft Ajax Library but does not use any ASP.NET server controls. For instance, an **HTML** can include script elements that reference the Microsoft Ajax Library .js files. The Microsoft Ajax Library allows Ajax applications to perform all processing on the client. A client and server solution consists of using both the Microsoft Ajax Library and ASP.NET server controls.

The following illustration shows the functionality that is included in the client-script libraries and server components that are included with ASP.NET.



The illustration shows the functionality of the client-based Microsoft Ajax Library, which includes support for creating client components, browser compatibility, and networking and core services. The illustration also shows the functionality of server-based Microsoft Ajax features, which include script support, Web services, application services, and server controls.

Microsoft Ajax Client Architecture

The client architecture includes libraries for component support, browser compatibility, networking, and core services.

Components

Client components enable rich behaviors in the browser without postbacks. Components fall into three categories:

- Components, which are non-visual objects that encapsulate code.
- Behaviors, which extend the behavior of existing DOM elements.
- Controls, which represent a new DOM element that has custom behavior.

The type of component that you use depends on the type of client behavior you want. For example, a watermark for an existing text box can be created by using a behavior that is attached to the text box.

Browser Compatibility

The browser compatibility layer provides Microsoft Ajax scripting compatibility for the most frequently used browsers. This enables you to write the same script regardless of which supported browser you are targeting.

Networking

The networking layer handles communication between script in the browser and Web-based services and applications. It also manages asynchronous remote method calls. In many scenarios, such as partial-page updates that use the UpdatePanel control, the networking layer is used automatically and does not require that you write any code.

The networking layer also provides support for accessing server-based forms authentication, role information, and profile information in client script. This support is also available to Web applications that are not created by using ASP.NET, as long as the application has access to the Microsoft Ajax Library.

Core Services

The Ajax client-script libraries in ASP.NET consist of JavaScript (.js) files that provide features for object-oriented development. The object-oriented features included in the Microsoft Ajax client-script libraries enable a high level of consistency and modularity in client scripting.

The following core services are part of the client architecture:

- Object-oriented extensions to JavaScript, such as classes, namespaces, event handling, inheritance, data types, and object serialization.
- A base class library, which includes components such as string builders and extended error handling.
- Support for JavaScript libraries that are either embedded in an assembly or are provided as standalone JavaScript (.js) files. Embedding JavaScript libraries in an assembly can make it easier to deploy applications and can help solve versioning issues.

Debugging and Error Handling

The core services include the Sys.Debug class, which provides methods for displaying objects in readable form at the end of a Web page. The class also shows trace messages, enables you to use assertions, and lets you break into the debugger. An extended Error object API provides helpful exception details with support for release and debug modes

Globalization

The Ajax server and client architecture in ASP.NET provides a model for localizing and globalizing client script. This enables you to design applications that use a single code base to provide UI for many locales (languages and cultures). *For example*, the Ajax architecture enables JavaScript code to format Date or Number objects automatically according to culture settings of the user's browser, without requiring a postback to the server.

Ajax Server Architecture

The server pieces that support Ajax development consist of ASP.NET Web server controls and components that manage the UI and flow of an application. The server pieces also manage serialization, validation, and control extensibility. There are also ASP.NET Web services that enable you to access ASP.NET application services for forms authentication, roles, and user profiles.

Script Support

Ajax features in ASP.NET are commonly implemented by using client script libraries that perform processing strictly on the client. You can also implement Ajax features by using server controls that support scripts sent from the server to the client. You can also create custom client script for your ASP.NET applications. In that case, you can also use Ajax features to manage your custom script as static .js files (on disk) or as .js files embedded as resources in an assembly.

Ajax features include a model for release and debug modes. Release mode provides error checking and exception handling that is optimized for performance, with minimized script size. Debug mode provides more robust debugging features,

such as type and argument checking. ASP.NET runs the debug versions when the application is in debug mode. This enables you to throw exceptions in debug scripts while minimizing the size of release code.

Script support for Ajax in ASP.NET is used to provide two important features:

- The Microsoft Ajax Library, which is a type system and a set of JavaScript extensions that provide namespaces, inheritance, interfaces, enumerations, reflection, and additional features.
- Partial-page rendering, which updates regions of the page by using an asynchronous postback.

Localization

The Microsoft Ajax architecture builds on the foundation of the ASP.NET 2.0 localization model. It provides additional support for localized .js files that are embedded in an assembly or that are provided on disk. ASP.NET can serve localized client scripts and resources automatically for specific languages and regions.

Web Services

With Ajax functionality in an ASP.NET Web page, you can use client script to call both ASP.NET Web services (.asmx) and **Windows Communication Foundation** (WCF) services (.svc). The required script references are automatically added to the page, and they in turn automatically generate the Web service proxy classes that you use from client script to call the Web service. You can also access ASP.NET Web services without using Microsoft Ajax server controls (*for example,* if you are using a different Web development environment). To do so, in the page, you can manually include references to the Microsoft Ajax Library, to script files, and to the Web service itself. At run time, ASP.NET generates the proxy classes that you can use to call the services.

Windows Communication Foundation is a framework for building service-oriented applications. Using WCF, you can send data as asynchronous messages from one service endpoint to another. A service endpoint can be part of a continuously available service hosted by IIS, or it can be a service hosted in an application.

Application Services

Application services in ASP.NET are built-in Web services that are based on ASP.NET forms authentication, roles, and user profiles. These services can be called by client script in an Ajax-enabled Web page, by a Windows client application, or by a WCF-compatible client.

Server Controls

Ajax server controls consist of server and client code that integrate to produce rich client behavior. When you add an Ajax-enabled control to an ASP.NET Web page, the page automatically sends supporting client script to the browser for Ajax functionality. You can provide additional client code to customize the functionality of a control, but this is not required.

The following list describes the most frequently used Ajax server controls.

Control	Description
ScriptManager	Manages script resources for client components, partial-page rendering, localization, globalization, and custom user scripts. The ScriptManager control is required in order to use the UpdatePanel, UpdateProgress, and Timer controls. However, the ScriptManager control is not required when creating a client-only solution.
UpdatePanel	Enables you to refresh selected parts of the page, instead of refreshing the whole page by using a synchronous postback.
UpdateProgress	Provides status information about partial-page updates in UpdatePanel controls.
Timer	Performs postbacks at defined intervals. You can use the Timer control to post the whole page, or use it together with the UpdatePanel control to perform partial-page updates at a defined interval.

You can also create custom ASP.NET server controls that include Ajax client behaviors. Custom controls that enhance the capabilities of other ASP.NET Web controls are referred to as *extender control*s.

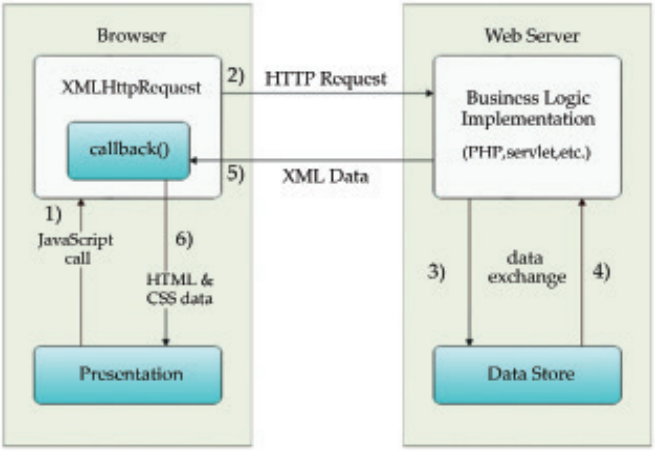
How does AJAX work?

AJAX calls are *asynchronous,* meaning they are made behind the scenes, independently from each other and the site as a whole. When a browser makes an AJAX call to the server, it is not stuck waiting for a response, halting all of the site's functionality. Instead, the web service on the server will send the data back to the browser once the task is completed, where client-side scripts will process the response and deliver it to the user.

For example: If you want to give a movie on Netflix a star rating, you can click the rating, it will show up immediately, and the rating gets stored in your profile, all without any other changes to the page. That's AJAX in action.

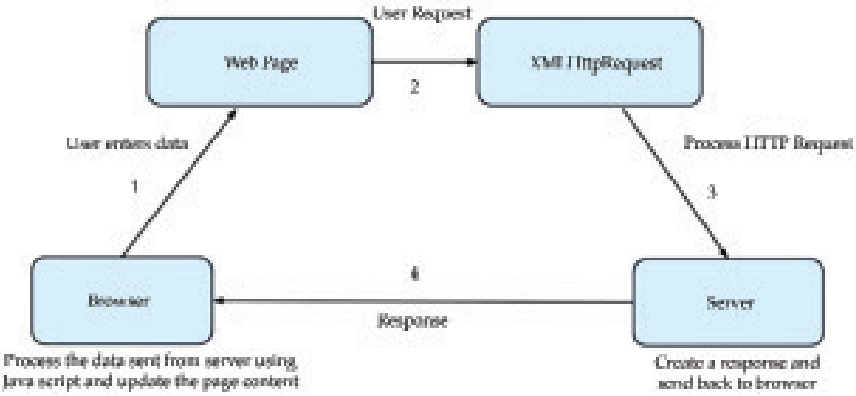
How AJAX works

AJAX communicates with the server using XMLHttpRequest object. Let's try to understand the flow of ajax or how ajax works by the image displayed below.



As you can see in the above example, XMLHttpRequest object plays an important role.

- User sends a request from the UI and a javascript call goes to XMLHttpRequest object.
- HTTP Request is sent to the server by XMLHttpRequest object.
- Server interacts with the database using JSP, PHP, Servlet, ASP.net etc.
- Data is retrieved.
- Server sends XML data or JSON data to the XMLHttpRequest callback function.
- HTML and CSS data is displayed on the browser.



As shown in the figure, when user enters some data through the browser, the appropriate web page will be open. When the user requests from the browser it creates XMLHttpRequest object. When a request goes to server it processes the HttpRequest and creates response and send data back to the browser. It processes the data sent from the server using java script and update the page content.

5.3 TECHNOLOGIES USED IN AJAX

Key Notes

Use Technologies in AJAX

AJAX (Asynchronous JavaScript and XML) involves a combination of various technologies that work together to enable asynchronous communication between the client and the server. While AJAX originally included XML as a data format, JSON (JavaScript Object Notation) is more commonly used today due to its simplicity and efficiency.

Document Object Model

Document Object Model (DOM) is a cross-platform and language-independent application programming interface that treats an HTML, XHTML, or XML document as a tree structure wherein each node is an object representing a part of the document. The objects can be manipulated programmatically and any visible changes occurring as a result may then be reflected in the display of the document.

Cascading Style Sheets

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media.

Learning Objectives

- Discuss about ajax operation.
- Focus on simple ajax application

Use Technologies in AJAX

AJAX cannot work independently. It is used in combination with other technologies to create interactive webpages. Before describe ajax, ajax is not a technology but group of inter-related technologies. Following technologies are used in Ajax.

As describe earlier, ajax is not a technology but group of inter-related technologies. AJAX technologies includes:

- JavaScript
- DOM
- CSS
- XMLHttpRequest
- HTML/XHTML and CSS

JavaScript

JavaScript is an object based scripting language used to make request to the web server. JavaScript function is called when event occurs in a page. It is loosely typed and cross-platform programming language. It is used to create interactive web sites. It is mainly

used for client side validation, displaying date and time, displaying popup windows and dialog boxes etc. It has become common in server side programming and creation of desktop applications. It provides dynamic behavior to the application.

DOM

It stands for Document Object Model which is representation of XML and HTML documents. It allows interaction with web page using java script and changes the structure of the document. It provides API for changing content, structure and style. It binds JavaScript and other scripting languages with HTML and other markup languages. It is also used to display dynamic data and interacting with data. It is used for manipulating XML and HTML documents.

CSS

It stands for Cascading Style Sheets. It is used for describing the presentation of documents written in markup language. It defines how to display contents and styles of HTML elements. Its new feature added to HTML where users can create style sheets. Styles are saved in external.css file which allows changing the layout of pages in the web site such as layout, colors, fonts, height, width, background images etc. It enables to design web site in a new way. It is a must when working with web design.

XMLHttpRequest

It is used to interact with client and server. It plays important role in Ajax web development technique. It can be used by JavaScript, VBScript and other scripting languages to manipulate data to and from server using HTTP making connection between client and server side. It is used by many websites to create dynamic web applications. With Ajax, JavaScript communicates with server using XMLHttpRequest object to send request and get response from the server without refreshing the page.

Syntax for creating XMLHttpRequest is as follows:

Variable=new XMLHttpRequest ();

HTML/XHTML and CSS

These technologies are used for displaying content and style. It is mainly used for presentation

Steps of AJAX Operation

It has following stages:

- User visits the page by clicking the targeted link or typing **URL** in the web browser. When the page is loaded, the Ajax engine is also initialized. User interacts with Ajax engine to interact with web server.
- User requests from web browser and receive the response back. This can be

done by an AJAX application which recalls the HTTP transaction which sends request to the server, server responds to the client and client and server close the connection.

- The client renders the document which may include some JavaScript. Communication with server takes place asynchronously to the user.
- JavaScript code creates XMLHttpRequest object which requests document from the web server.
- XMLHttpRequest indicates that data has arrived which is often called a callback. Ajax engine calls the callback functions to receive server response. Data is exchanged with server without reloading the page.
- The callback function updates the browser using DOM objects, HTML and CSS data. The callback function processes the data and displays it. Ajax applications are terminated once the request is completed.

Uniform Resource Locator (URL) is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.

XMLHttpRequest

It is used to interact with client and server. It plays an important role in Ajax web development technique. It can be used by JavaScript, VBScript and other scripting languages to manipulate data to and from server using HTTP making connection between client and server side.

- It is used by many websites to create dynamic web applications.
- It communicates with server in the background.
- It updates the page with new data without reloading the page.
- It request and receive the data from web server.

Creating XMLHttpRequest object is as follows:

All modern browsers such as IE7, Firefox, Chrome, and Safari have built in XMLHttpRequest object.

variable variable_name = new XMLHttpRequest ();

OR

Old versions of Internet Explorer such as IE5 and IE6 use ActiveX objects as shown below:

variable variable_name= new ActiveXObject (“Microsft. XMlHTTP”);

The following small example determines which object to use and how to create it:

```
If(window.XMLHttpRequest)
{
request=new XMLHttpRequest();
}
else if(window.ActiveXObject)
{
request=new ActiveXObject("Microsoft.XMLHTTP");
}
```

XMLHttpRequest object methods:

- open (method,URL):

  open (method, URL, async), open (method, URL, async, userName), open (method, URL, async, username, password), Specifies the method, URL and other attributes of the request.

  Method represents types of the request such as GET, POST or HEAD and other Http methods such as PUT and DELETE.

  URL Specifies location or path of the file async specifies true (asynchronous) or false(synchronous), "true" means script processes after the send() method without waiting for response and "false" means script waits for response before continuing the script processing.

- send(string): It is used to send the request.
- String: Use only POST method request.
- setRequestHeader(label, value): It is used to set label/value to HTTP header.
- getResponseHeader(headerName): It returns value of the specific HTTP header
- getAllResponseHeaders(): It returns complete set of HTTP headers.
- abort(): It cancels the request.

XMLHttpRequest object Properties

- readyState: It defines current state or holds the status of the XMLHttpRequest. It has some possible values from 0 to 4.
  - 0: The request is not initialized.
  - 1: The request has been setup. Server connection established.
  - 2: The request has been sent.
  - 3: The request processing.
  - 4: The request is finished and response is ready.
- Onreadystatechange: It stores the name of the function that will process

the response from the server and will be called automatically each time the readyState property changes.

- Status: It returns status as a number. e.g. 404: page not found 200: OK
- responseText: It returns returned data as string of characters.
- responseXml: It returns data as XML file. It returns XML document object of data from server.
- statusText: It returns the status as a string. (e.g. "Not Found" or "OK").

Ajax Security

- JavaScript code is open source. Unauthorized user/hacker can use code for inferring server side weaknesses and can easily penetrate into the system.
- It is combination of both JavaScript and XML which have security issues that Ajax helps to facilitate.
- The client code can be easily modified by attacker because it is open source code.
- The Ajax engine uses JavaScript to capture and transfer user commands into function calls which are sent in plain visible text to the server which may reveal database details.
- JavaScript code is downloaded from server side and executed at client side which compromises the client by mal-intended code.
- All the client side validation must be backed up by server side validation.
- Ajax applications use the same server side security schemes of regular web applications.
- JavaScript code is open source. Unauthorized user/hacker can use code for inferring server side weaknesses and can easily penetrate into the system.
- It is combination of both JavaScript and XML which have security issues that Ajax helps to facilitate.
- The client code can be easily modified by attacker because it is open source code.
- The Ajax engine uses JavaScript to capture and transfer user commands into function calls which are sent in plain visible text to the server which may reveal database details.
- JavaScript code is downloaded from server side and executed at client side which compromises the client by mal-intended code.
- All the client side validation must be backed up by server side validation.
- Ajax applications use the same server side security schemes of regular web applications.

Ajax Issues

Ajax applications are difficult to maintain, test and debug.

- It is harder to bookmark updated page content.
- Developers should know the knowledge of technical skills such as JavaScript, CSS, HTML and DOM.
- The browsers which do not have JavaScript cannot use the application.
- Old browsers does not support XMLHttpRequest object. IFrame will help because IFrame supports modern browsers.
- JavaScript technology is incompatible with some applications and browsers.
- Complexity is increased because server side developers need to understand presentation logic at client side and server side.
- It is less secure because JavaScript code is open source and is visible to user/hacker .So JavaScript code can raise security problem. Since data is sent in plain text, so it is extremely vulnerable to hacker attacks.

Complexity characterises the behaviour of a system or model whose components interact in multiple ways and follow local rules, leading to non-linearity, randomness, collective dynamics, hierarchy, and emergence.

Example

Now let us see some Simple Hello World Example of Ajax usage.

Listing 1: ajaxhelloworld.htm

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function callme()
{

var a;

    if (window.XMLHttpRequest)
    {// If the browser if IE7+FirefoxChromeOperaSafari
      a=new XMLHttpRequest();

    }
   else
    {//If browser is IE6, IE5
      a=new ActiveXObject("Microsoft.XMLHTTP");
    }
```

```
a.onreadystatechange=function()
{

  if (a.readyState==4 && a.status==200)
  {
    document.getElementById("myDiv").innerHTML=a.responseText;
   }
}

a.open("POST","mrbool.txt",true);
a.send();
} // fun1() close
</script>
</head>
<body>

<div id="myDiv" style="width: 300px; height: 30px;">Click on the button below</div>
<button type="button" onclick="callme()">Change Content</button>

</body>
</html>
```

What is Ajax and where is it Used in Technology?

Ajax is not a programming language or a tool, but a concept. Ajax is a client-side script that communicates to and from a server/database without the need for a postback or a complete page refresh. The best definition you have read for Ajax is "the method of exchanging data with a server, and updating parts of a web page without reloading the entire page." Ajax itself is mostly a generic term for various JavaScript techniques used to connect to a web server dynamically without necessarily loading multiple pages. In a more narrowly-defined sense, it refers to the use of XmlHttpRequest objects to interact with a web server dynamically via JavaScript.

Benefits of Ajax

There are four main benefits of using Ajax in web applications:

- *Callbacks*: Ajax is used to perform a callback, making a quick round trip to and from the server to retrieve and/or save data without posting the entire page back to the server. By not performing a full postback and sending all form data to the server, network utilization is minimized and quicker operations occur. In sites and locations with restricted bandwidth, this can greatly improve network performance. Most of the time, the data being sent to and from the server is minimal. By using callbacks, the server is not required to process all form elements. By sending only the necessary data, there is limited processing on the server. There is no need to process all form elements, process the ViewState, send images back to the client, or send a full page back to the client.
- *Making Asynchronous Calls*: Ajax allows you to make asynchronous calls to

a web server. This allows the client browser to avoid waiting for all data to arrive before allowing the user to act once more.

- *User-Friendly*: Because a page postback is being eliminated, Ajax enabled applications will always be more responsive, faster and more user-friendly.
- *Increased Speed*: The main purpose of Ajax is to improve the speed, performance and usability of a web application. A great example of Ajax is the movie rating feature on Netflix. The user rates a movie and their personal rating for that movie will be saved to their database without waiting for the page to refresh or reload. These movie ratings are being saved to their database without posting the entire page back to the server.

Technical Aspects of Ajax

Ajax callbacks can be done by instantiating an XMLHttpRequest object in the client-side JavaScript. The XMLHttpRequest object can be used to directly call server-side objects like pages and web services. These pages and web services will either save and/or return data.

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

Ajax was originally an acronym for Asynchronous JavaScript and XML. "Asynchronous" means that multiple events are happening independently of one another. Once a client initializes an Ajax callback to the server, the client will not need to wait for a response and can continue to use the web application while the request is being processed. Once done, the server will send a response back to the client and the client will process it as necessary.

What Advances have Been Made to Ajax?

JavaScript is the client-side programming language and **XML** is a markup language to define data. JSON is another markup language to define data. JSON (JavaScript Object Notation) is much easier to use with JavaScript than XML. When it comes to Ajax and JavaScript, JSON Web Services are replacing XML Web Services.

Another major advance to JavaScript and Ajax is the JavaScript object library called jQuery. This free, open-source software is a wrapper around JavaScript. jQuery is used to easily

write client-side JavaScript to navigate and manipulate a page and make asynchronous Ajax callbacks. By using jQuery and JSON Web Services, Ajax callbacks have become standard programming practices for designing and developing web applications.

The Ajax Control Toolkit is a suite of controls created by Microsoft that is integrated into Visual Studio and can be dragged and dropped onto web forms just like html and server controls. These controls are intended to be used for Ajax callbacks. However, they can also be used as normal client and or server controls. *For example,* Asp.Net does not come with the Tabs controls. However, the Ajax Control Toolkit does. The Tab control can postback to the server just like server controls.

Where Ajax should be used?

Ajax should be used anywhere in a web application where small amounts of information could be saved or retrieved from the server without posting back the entire pages. A good example of this is data validation on save actions. Another example would be to change the values in a drop down list-box based on other inputs, such as state and college list boxes. When the user selects a state, the college list box will repopulate with only colleges and universities in that state.

Another great example is when the client needs to save or retrieve session values from the server, based on a user preference such as the height, width or position of an object. Adjusting the width could make a callback to the server to set the session variable for the new width. This way, whenever the page is refreshed, the server can adjust the object's width based on this session variable. Otherwise, the object would go back to its initial default width.

Other features include text hints and autocomplete text boxes. The client types in a couple of letters and a list of all values that start with those letters appear below. A callback is made to a web service that will retrieve all values that begin with these characters. This is a fantastic feature that would be impossible without Ajax and is also part of the Ajax Control Toolkit.

Segue recently used Ajax to support a client application that had problems due to limited bandwidth and page size. The combination caused the application to take too long to retrieve data and display it on the page. Sometimes, the web server would simply not have the resources to handle the request and timeout. The best solution for this issue was Ajax.

To solve this problem, we created JSON Web Services on the web server in order to retrieve the details about the selected item. The JSON web service would retrieve the data and convert into JSON and return a JSON string.

Instead of posting back to the server, the client would call the web service when an item was selected from the list box. We used jQuery to make an asynchronous Ajax call to the web service. Once the client retrieved the data back from the web service, more client side processing was done to display the information on the page. The time it took to display the details on the page after the item was selected was instantaneous.

There was no page flicker, refresh or postback involved.

Simple Ajax Application

We have created a simple AJAX application that illustrates the basic concepts. This application is a *Pizza Ordering Form*. There are various inputs in this ordering form, the first of which is the phone number of the customer. The Pizza Company keeps a database of all the customers who have ordered before. As soon as the customer types in his / her phone number a call is made to the server to search the database for that phone number and extract all the relevant information on that client. That information is then used to auto fill the form.

Here is the table that has been created in the database:

Phone	Last Name	First Name	Street	City	State	Zip
407-896-0001	Mouse	Mickey	123 Cheddar St.	Orlando	FL	32801
407-896-0002	Duck	Donald	387 Eider Dn.	Orlando	FL	32801
407-896-0003	Mouse	Minnie	482 Brie Ln.	Orlando	FL	32801

XML Requests to the Server

XML is neither necessary nor required to make requests to the server. In fact it is less efficient than sending data in plain text as name-value pairs. The only compelling reason to use the XML format in sending requests is that the script on the server side does not accept data in any other format.

Here is an example of how to send a request in XML format:

Let us say that the server only accepts a person's name in this format:

```
<name>
  <firstName> Mickey </firstName>
  <lastName> Mouse </lastName>
</name>
```

Then these steps have to be followed to send a person's name to the server:

```
function callServer()
{
  // Get the first and last name from the form
  var firstName = document.getElementById("firstName").value;
  var lastName = document.getElementById("lastName").value;
```

```
  // Create the XML String
  var xmlStr = "<name>" +
            "<firstName>" + escape(firstName) + "</firstName>" +
               "<lastName>" + escape(lastName) + "</lastName>" +
               "</name>";

  // Build the URL to connect to
  var url = "/scripts/accessNames.php";

  // Open a connection to the server
  xhr.open ("POST", url, true);

  // Tell the server that you are sending XML
  xhr.setRequestHeader ("Content-Type", "text/xml");

  // Delegate a function to handle the response
  xhr.onreadystatechange = updatePage();

  // Send the request
  xhr.send (xmlStr);
}
```

XML Responses from the Server

While it is easy to send a request as name-value pairs to the server, no matter in what format the response comes back it has to be parsed on the client side. It makes sense to send the response in as unambiguous way as possible. This where the strength of XML lies.

If the response came back in the following format you would have no problem understanding what it meant:

```
<students>
   <student>
     <name> Mickey Mouse </name>
     <major> Dance </major>
     <gpa> 3.5 </gpa>
```

```
  </student>
  <student>
    <name> Donald Duck </name>
    <major> Song </major>
    <gpa> 3.2 </gpa>
  </student>
</students>
```

There are two ways in which you can treat the XML response from the server:

- As plain text that is in XML format
- XML document represented as a DOM Document object

If you treat the XML response as plain text your code in the *function updatePage()* to accept the response would be of the form:

```
var response = request.responseText;
```

where *response* would be in the form of a long string with no spacings like so:

```
<students><student><name>Mickey Mouse</name>
<major> Dance </major><gpa>3.5</gpa></student>
<student><name>Donald Duck</name><major>Song</major>
<gpa>3.2</gpa></student></students>
```

The string has been split into lines for readability. You would then parse the string using the various String functions in JavaScript like *split()* and extract the information.

On the other hand if you treat the XML response as DOM document object then JavaScript provides utilities to parse the response more efficiently. The following example shows how:

```
function updatePage()
{
 if ((request.readyState == 4) && (request.status == 200))
 {
   var xmlDoc = request.responseXML;

   // Get all student elements as an array
   var student = xmlDoc.getElementsByTagName («student»);

   // Iterate through the array and retreive the student info
   for (var i = 0; i < student.length; i++)
   {
```

```
    var name = student[i].childNodes[0].value;
    var major = student[i].childNodes[1].value;
    var gpa = student[i].childNodes[2].value;

    // Now you can do something with this data
   }
  }
}
```



TECHNIQUE 17

CONVENIENCE METHODS OF AJAX

If you are just making a simple request and if we do not care about error handling jQuery provides several "convenience methods" that let us use an abbreviated syntax. Each of these methods takes a URL, an optional data object, and an optional callback for handling a successful request.

- ☐ We already practice this technique
- ☐ This techniques would suitable or not for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 18

AJAX ASSIGNMENT

Asynchronous JavaScript and XML commonly referred AJAX is a web development technique or a set of web development techniques mostly used for building interactive applications for the Web that process user requests immediately. It is a technique for creating fast and dynamic web pages. AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page. Ajax applications do not require installation of a plug-in; it works directly with a Web.

- ☐ We already practice this technique
- ☐ This techniques would suitable or not for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 19

MAKE AN ACCORDION MENU IN JQUERY

A menu with the default configuration, disabled items and nested menus. A list is transformed, adding theming, mouse and keyboard navigation support. Try to tab to the menu then use the cursor keys to navigate.

- ☐ We already practice this technique
- ☐ This techniques would suitable or not for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 20

HANDLING AJAX RESPONSES

Handling AJAX responses is a crucial aspect of building dynamic and interactive web applications. Once an AJAX request is made to the server, you need to process the server's response and update the web page's content accordingly. Inside the success callback, process the server's response. This usually involves extracting data from the response and updating the page's content.

- ☐ We already practice this technique
- ☐ This techniques would suitable or not for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique

SUMMARY

- Ajax refers to JavaScript and XML, technologies that are widely used for creating dynamic and asynchronous web content. While Ajax is not limited to JavaScript and XML technologies, more often than not they are used together by web applications. The focus of this tutorial is on using JavaScript based Ajax functionality in JavaServer Faces web applications.
- Google map is considered as one of the popular AJAX based applications which allows user to zoom in and zoom out and scroll the map in four directions like a desktop application.
- Ajax is a way of developing richer and more interactive web applications .These applications are referred to as Rich Internet Applications (RIA).
- Ajax features in ASP.NET are commonly implemented by using client script libraries that perform processing strictly on the client
- JavaScript is an object based scripting language used to make request to the web server. JavaScript function is called when event occurs in a page.
- Using AJAX in web development allows you to create more dynamic and interactive web applications by updating parts of a page without requiring a full page reload.
- Additionally, modern web development often involves using more advanced techniques like Single Page Applications (SPAs) and frameworks like React, Angular, or Vue.js, which provide more sophisticated ways of handling dynamic content without traditional page reloads.
- The Ajax server and client architecture in ASP.NET provides a model for localizing and globalizing client script. This enables you to design applications that use a single code base to provide UI for many locales (languages and cultures).
- JavaScript is the client-side programming language and XML is a markup language to define data. JSON is another markup language to define data. JSON (JavaScript Object Notation) is much easier to use with JavaScript than XML.

KNOWLEDGE CHECK

1. What is AJAX?

a. is a program

b. is a country name

c. is a football club name

d. None of these

2. What sever support AJAX?

a. SMTP
b. WWW
c. HTTP
d. None of these

3. AJAX Stands for:

a. Asynchronous Javascript and XML
b. Abstract JSON and XML
c. Another Java Abstraction for X-Windows
d. Another Java and XML Library

4. What makes Ajax unique? Ajax

a. It works as a stand-alone Web-development tool.
b. It works the same with all Web browsers.
c. It uses C++ as its programming language.
d. It makes data requests asynchronously.

5. What does the XMLHttpRequest object accomplish in Ajax?

a. It's the programming language used to develop Ajax applications.
b. It provides a means of exchanging structured data between the Web server and client.
c. It provides the ability to asynchronously exchange data between Web browsers and a Web server.
d. It provides the ability to mark up and style the display of Web-page text.

6. Of the following technologies, which one provides the ability to dynamically interact with Web page layout? Ajax

a. JavaScript.
b. XML.
c. Document Object Model.
d. HTML.

7. What is the first step in creating a Joke of the Day application?

a. Create the application's text file.
b. Write the application's HTML.
c. Turn the HTML application into an Ajax application.
d. Upload the application to a Web server.

8. Ajax stands for Synchronous JavaScript and XML. | Ajax

a. True
b. False

9. Ajax is a programming language.

a. True

b. False

10. ActiveX can be disabled via browser settings.

a. True

b. False

REVIEW QUESTIONS

1. Explain the characteristics of an ajax application.
2. What do you understand rich internet applications?
3. Discuss about features of ajax.
4. How does ajax work?
5. How to technologies used in AJAX?
6. Discuss about ajax security and ajax issues.
7. What is ajax and where is it used in technology?
8. Explain the XML requests to the server.

Check your Result

1. (a)	2. (c)	3. (d)	4. (c)	5. (b)
6. (b)	7. (b)	8. (b)	9. (b)	10. (a)

REFERENCES

1. APACHE: XAP: Overview. http://ineuhator.apache.org/xap/indcx.phpOcid=14- html, Nov 2006
2. BACKSASE: Ajax in the Enterprise. 2007 IBer071 BERGER, Kenneth: Introducing Adobe Dreamweaver CS3. http://www.adobe.cord dev-netAlreamweaver/artieles/introducing_dreamweaver_es3.html, Mar 2007
3. BERRAIOLO, Jon: When Does Ajax Make Business Sense. Jun 2007 fGa1061 GALBRAITII, Ben: 2006 Survey Results. Sep 2006
4. BOJANIC, Peter: The Joy of XUL. 2007 1Cha0 I I CIIAMPEON, Steve: JavaScript: How Did We Get Here? Apr 2001
5. CRANE, Dave ; PASCARELLO, Eric ; JAMES, Darren: Ajax in action - Das Entwick-krbuch jiir das Web 2,12, 1st Miinchen : Addison-Wesley. 2006
6. DEB, Brijesh: Rich Internet Applications - A Look Into Available Technology Choices. In: fax magazine (2006), Jun. — http://jaxmagmetfitrionlinc-artikelt psecomid, 828, nodeid, 147.html
7. DOMENIC, Marc: Rich Internet Applications and AJAX - Selecting the best product. Nov 2005
8. DOWNES, Jonathan: Reverse Alta. May 2006
9. ER R A, Alex ; B AUER I N I, Steve: Adobe and Mozilla Foundation to Open Source Flash Player Scripting Engine. Nov 2007
10. GETAHEAD: DWR: Easy AJAX for JAVA. http://getahead.org/dwrioverviewidwr, kiib061 GIBBS, Matt: ASP.NET Alias Powers the AJAX-Style Sites You have Been Waiting For. In: MSDN Magazine Vol. 21 (2006). Jul Google Web Toolkit Product Overview. http://code.google.com/webtool-kit/overview.hunl, 2007
11. GOODMAN, Robert ; BUSTELO, Leugim: AJAX Toolkit Framework. http://www.eclipse.oreatf/dernos/webinar/AJAX Toolkit Framcwork.ppt, Nov 2006
12. William: Laszlo: An Open Source Framework for Rich Internet Applica-tions. hup://todayjava.nct/pub/a/today/2005/03/22/laszlo.html, Mar 2005

CHAPTER 6

ESSENTIAL OF WEB API

6.1 FUNCTIONALITY OF APPLICATION PROGRAMING INTERFACE

KEY NOTES

Concept

APIs empower developers to create more powerful applications by leveraging the capabilities of other services and systems without having to reinvent the wheel. Whether it's integrating maps into a navigation app, enabling social media sharing, or processing payments securely, APIs have become the building blocks of modern software, enabling a rich ecosystem of interconnected applications.

Web APIs

Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices.

Implementations

A single API can have multiple implementations (or none, being abstract) in the form of different libraries that share the same programming interface. The separation of the API from its implementation can allow programs written in one language to use a library written in another.

Learning Objectives

- API in object-oriented languages
- API libraries and frameworks
- API and protocols
- API sharing and reuse via virtual machine
- Web use to share content
- APIs and Copyrights

Concept

An Application Programming Interface (API) is a set of protocols, functions, and tools that allows different software applications to communicate with each other. APIs serve as intermediaries that enable developers to access certain features or data of an application, service, or platform without needing to understand the underlying complexities. APIs play a crucial role in modern software development by promoting modularity, interoperability, and reusability of code.

APIs provide a well-defined structure for how different software components interact. They specify the methods and data formats that developers should use when making requests to access a service's functionality. APIs are commonly used to facilitate interactions between different software systems, such as web applications, mobile apps, and third-party services.

APIs can take various forms, including web APIs (also known as HTTP APIs) that are accessed over the internet, library APIs that offer functions within a programming language, and operating system APIs that provide interaction with underlying hardware and software resources.

Web APIs, in particular, have gained immense popularity. They expose endpoints that developers can access through HTTP requests (such as GET, POST, PUT, DELETE). These endpoints return data usually in formats like JSON or XML. This enables developers to integrate external functionalities into their applications, such as retrieving weather data, social media posts, or performing online payments.

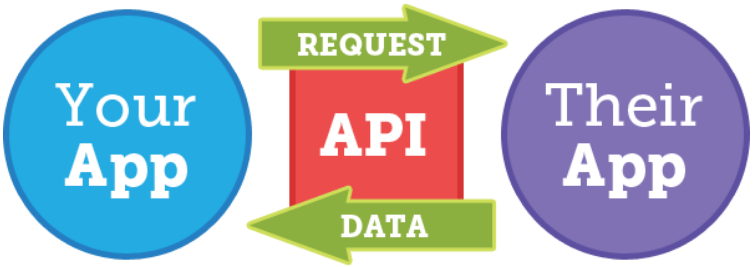
By providing a standardized way for different applications to communicate and collaborate, APIs offer numerous benefits. They allow developers to build on existing tools and services, saving time and effort. This concept of "building on top of" is known as "API economy," where organizations provide APIs to foster innovation and drive business growth. This approach promotes specialization and distributed development, leading to improved software quality and faster development cycles.

Language used

An API can be:

- Language-dependent, meaning it is only available by using the syntax and elements of a particular language, which makes the API more convenient to use.
- Language-independent, written so that it can be called from several programming languages. This is a desirable feature for a service-oriented API that is not bound to a specific process or system and may be provided as remote procedure calls or web services. For example, a website that allows users to review local restaurants is able to layer their reviews over maps taken from Google Maps, because Google Maps has an API that facilitates this

functionality. Google Maps' API controls what information a third-party site can use and how they can use it.



The term API may be used to refer to a complete interface, a single function, or even a set of APIs provided by an organization. Thus, the scope of meaning is usually determined by the context of usage.

Detailed explanation

An API may describe the ways in which a particular task is performed. In procedural languages like C language the action is usually mediated by a function call. Hence the API usually includes a description of all the functions/routines it provides. For instance: the math.h include file for the C language contains the definition of the function prototypes of the mathematical functions available in the C language library for mathematical processing (usually called libm). This file describes how to use the functions included in the given library: the function prototype is a signature that describes the number and types of the parameters to be passed to the functions and the type of the return value.



The behavior of the functions is usually described in more details in a human readable format in printed books or in electronic formats like the man pages: e.g. on Unix systems the command man 3 sqrt will present the signature of the function sqrt in the form:

SYNOPSIS

```
#include <math.h>
double sqrt(double X);
float sqrtf(float X);
```

DESCRIPTION

DESCRIPTION

sqrt computes the positive square root of the argument. ...

RETURNS

On success, the square root is returned. If X is real and

positive...

That means that the function returns the square root of a positive floating point number (single or double precision) as another floating point number. Hence the API in this case can be interpreted as the collection of the include files used by the C language and its human readable description provided by the man pages.

Documentation

Many program development environments provide the documentation associated with an API in some digital format,

e.g. perl comes with the tool perldoc:

$ perldoc -f sqrt

sqrt EXPR

sqrt #Return the square root of EXPR. If EXPR is omitted,

returns

#square root of $_. Only works on non-negative

operands, unless

#you've loaded the standard Math::Complex module.

python comes with the tool pydoc:

$ pydoc math.sqrt

Help on built-in function sqrt in math:

math.sqrt = sqrt(...)

sqrt(x)

Return the square root of x.

ruby comes with the tool ri:

$ ri Math::sqrt

-- Math::sqrt

Math.sqrt(numeric) => float

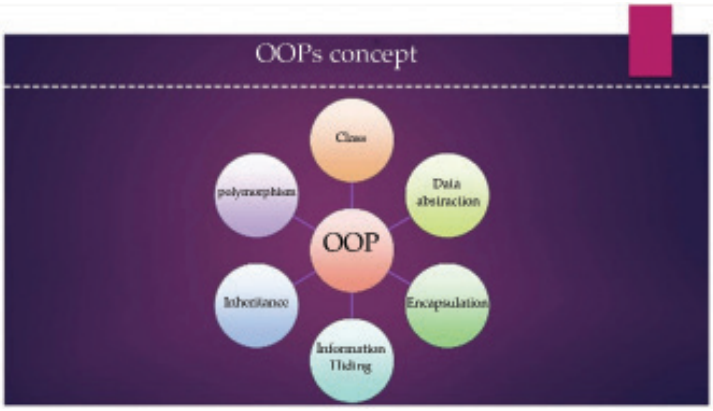
Returns the non-negative square root of _numeric_.

Java comes with the documentation organized in HTML pages (JavaDoc format), while Microsoft distributes the API documentation for its languages (Visual C++, C#, Visual Basic, F#, etc...) embedded in Visual Studio's help system.

API in object-oriented languages

In object-oriented languages, an API usually includes a description of a set of class definitions, with a set of behaviors associated with those classes. This abstract concept is associated with the real functionality exposed, or made available, by the classes that are implemented in terms of class methods (or more generally by all its public components hence all public methods, but also possibly including any internal entity made public, like fields, constants, nested objects, enums...). The API in this case can be conceived as the totality of all the methods publicly exposed by the classes (usually called the class interface). This means that the API prescribes the methods by which one interacts with/handles the objects derived from the class definitions. More generally, one can see the API as the collection of all the kinds of objects one can derive from the class definitions, and their associated possible behaviors. Again: the use is mediated by the public methods, but in this interpretation, the methods are seen as a technical detail of how the behavior is implemented. For instance: a class representing a Stack can simply expose publicly two methods push() (to add a new item to the stack), and pop() (to extract the last item, ideally placed on top of the stack).

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.



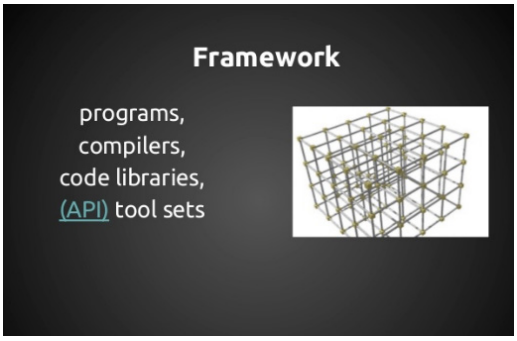
In this case the API can be interpreted as the two methods pop() and push(), or, more generally, as the idea that one can use an item of type Stack that implements the behavior of a stack: a pile exposing its top to add/remove elements. The second

interpretation appears more appropriate in the spirit of object orientation. This concept can be carried to the point where a class interface in an API has no methods at all, but only behaviors associated with it. For instance, the Java language and Lisp (programming language) API include the interface Serializable, which is a marker interface that requires that each class that implements it should behave in a serialized fashion. This does not require to have any public method, but rather requires that any class that implements it to have a representation that can be saved (serialized) at any time (this is typically true for any class containing simple data and no link to external resources, like an open connection to a file, a remote system, or an external device). Similarly the behavior of an object in a concurrent (multi-threaded) environment is not necessarily determined by specific methods, belonging to the interface implemented, but still belongs to the API for that Class of objects, and should be described in the documentation. In this sense, in **object-oriented languages**, the API defines a set of object behaviors, possibly mediated by a set of class methods. In such languages, the API is still distributed as a library. For example, the Java language libraries include a set of APIs that are provided in the form of the JDK used by the developers to build new Java programs. The JDK includes the documentation of the API in JavaDoc notation. The quality of the documentation associated with an API is often a factor determining its success in terms of ease of use.

Object-oriented language was primarily designed to reduce complexity in typical procedural languages through data binding and encapsulation techniques.

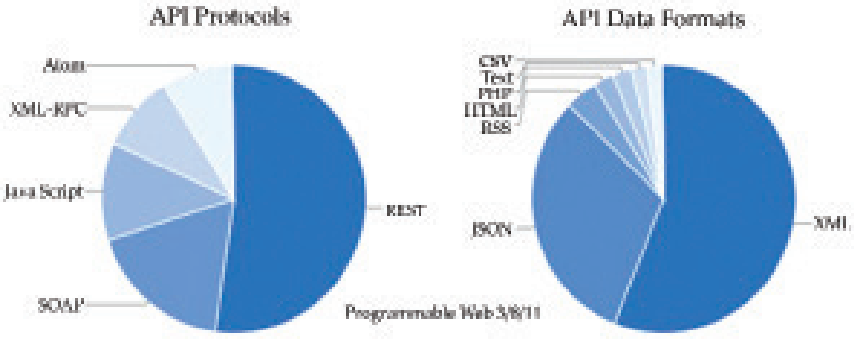
API libraries and frameworks

An API is usually related to a software library: the API describes and prescribes the expected behavior while the library is an actual implementation of this set of rules. A single API can have multiple implementations (or none, being abstract) in the form of different libraries that share the same programming interface. An API can also be related to a software framework: a framework can be based on several libraries implementing several APIs, but unlike the normal use of an API, the access to the behavior built into the framework is mediated by extending its content with new classes plugged into the framework itself. Moreover the overall program flow of control can be out of the control of the caller, and in the hands of the framework via inversion of control or a similar mechanisms.



API and protocols

An API can also be an implementation of a protocol. In general the difference between an API and a protocol is that the protocol defines a standard way to exchange requests and responses based on a common transport and agreeing on a data/message exchange format, while an API (not implementing a protocol) is usually implemented as a library to be used directly: hence there can be no transport involved (no information physically transferred from/to some remote machine), but rather only simple information exchange via function calls (local to the machine where the elaboration takes place) and data is exchanged in formats expressed in a specific language. When an API implements a protocol it can be based on proxy methods for remote invocations that underneath rely on the communication protocol. The role of the API can be exactly to hide the detail of the transport protocol. E.g.: RMI is an API that implements the JRMP protocol or the IIOP as RMI-IIOP. Protocols are usually shared between different technologies (system based on given computer programming languages in a given operating system) and usually allow the different technologies to exchange information, acting as an abstraction/mediation level between the two worlds. While APIs can be specific to a given technology: hence the APIs of a given language cannot be used in other languages, unless the function calls are wrapped with specific adaptation libraries.



Object API and protocols

An object API can prescribe a specific object exchange format, an object exchange protocol can define a way to transfer the same kind of information in a message

sent to a remote system. When a message is exchanged via a protocol between two different platforms using objects on both sides, the object in a programming language can be transformed (marshalled and unmarshalled) in an object in a remote and different language: so, e.g., a program written in Java invokes a service via SOAP or IIOP written in C# both programs use APIs for remote invocation (each locally to the machine where they are working) to (remotely) exchange information that they both convert from/to an object in local memory. Instead when a similar object is exchanged via an API local to a single machine the object is effectively exchanged (or a reference to it) in memory: e.g. via the memory allocated by a single process, or among multiple processes using shared memory or other sharing techologies like tuple spaces.

Shared memory is memory that may be simultaneously accessed by multiple programs with an intent to provide communication among them or avoid redundant copies.



API Sharing and Reuse via Virtual Machine

Some languages like those running in a virtual machine (e.g. .NET CLI compliant languages in the Common Language Runtime and JVM compliant languages in the Java Virtual Machine) can share APIs. In this case the virtual machine enables the language interoperation thanks to the common denominator of the virtual machine that abstracts from the specific language using an intermediate bytecode and its language binding. Hence this approach maximizes the code reuse potential for all the existing libraries and related APIs.

Web APIs

When used in the context of web development, an API is typically defined as a set of Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages, which is usually in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format. While "Web API" is virtually a synonym for web service, the recent

trend (so-called Web 2.0) has been moving away from **Simple Object Access Protocol** (SOAP) based services towards more direct Representational State Transfer (REST) style communications. Web APIs allow the combination of multiple services into new applications known as mashups.

Simple Object Access Protocol (SOAP) is a protocol specification for exchanging structured information in the implementation of web services in computer networks. Its purpose is to induce extensibility, neutrality and independence.



Web use to share content

The practice of publishing APIs has allowed web communities to create an open architecture for sharing content and data between communities and applications. In this way, content that is created in one place can be dynamically posted and updated in multiple locations on the web.

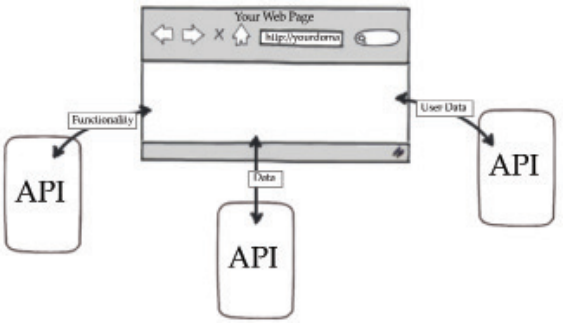


1. Photos can be shared from sites like Flickr and Photobucket to social network sites like Facebook and MySpace.
2. Content can be embedded, e.g. embedding a presentation from SlideShare on a LinkedIn profile.
3. Content can be dynamically posted. Sharing live comments made on Twitter with a Facebook account, for example, is enabled by their APIs.
4. Video content can be embedded on sites which are served by another host.
5. User information can be shared from web communities to outside applications, delivering new functionality to the web community that shares its user data via an open

API. One of the best examples of this is the Facebook Application platform. Another is the Open Social platform.

Implementations

The POSIX standard defines an API that allows a wide range of common computing functions to be written in a way such that they may operate on many different systems (Mac OS X, and various Berkeley Software Distributions (BSDs) implement this interface); however, making use of this requires re-compiling for each platform. A compatible API, on the other hand, allows compiled object code to function without any changes to the system implementing that API. This is beneficial to both software providers (where they may distribute existing software on new systems without producing and distributing upgrades) and users (where they may install older software on their new systems without purchasing upgrades), although this generally requires that various software libraries implement the necessary APIs as well.



Apple Inc. has shown less concern, breaking compatibility or implementing an API in a slower "emulation mode"; this allows greater freedom in development, at the cost of making older software obsolete. Among Unix-like operating systems, there are many related but incompatible operating systems running on a common hardware platform (particularly Intel 80386-compatible systems). There have been several attempts to standardize the API such that software vendors may distribute one binary application for all these systems; however, to date, none of these have met with much success. The Linux Standard Base is attempting to do this for the Linux platform, while many of the BSD Unixes, such as FreeBSD, NetBSD, and OpenBSD, implement various levels of API compatibility for both backward compatibility (allowing programs written for older versions to run on newer distributions of the system) and cross-platform compatibility (allowing execution of foreign code without recompiling).

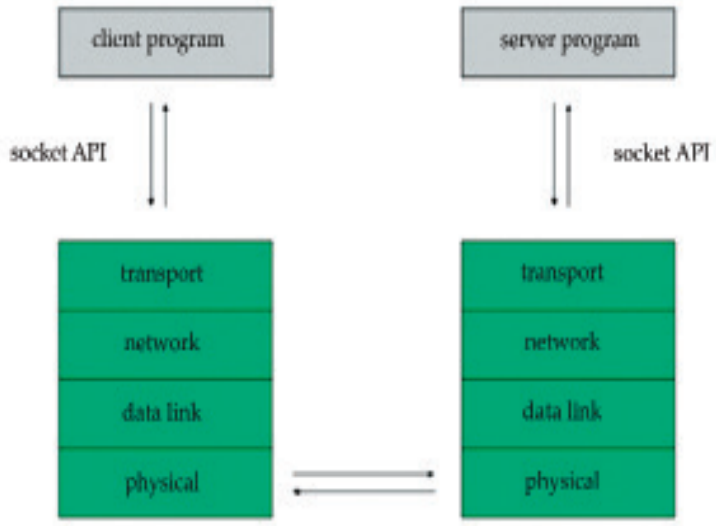
Release policies

The two options for releasing API are:

1. Protecting information on APIs from the general public. For example, Sony used to make its official PlayStation 2 API available only to licensed PlayStation developers. This enabled Sony to control who wrote PlayStation 2 games. This

gives companies quality control privileges and can provide them with potential licensing revenue streams.

2. Making APIs freely available. For example, Microsoft makes the Microsoft Windows API public, and Apple releases its APIs Carbon and Cocoa, so that software can be written for their platforms. A mix of the two behaviors can be used as well.



APIs and Copyrights

In 2010 Oracle sued Google for having distributed a new implementation of Java embedded in the Android operating system. The litigation concerns also the fact that Google did not acquire any permission to reproduce the Java API. Instead a similar permission was given to the OpenJDK project. However, Hon. William Alsup ruled on May 31, 2012, that APIs cannot be copyrighted.

API examples

- ASPI for SCSI device interfacing
- Carbon and Cocoa for the Macintosh
- DirectX for Microsoft Windows
- EHLLAPI
- Java APIs
- ODBC for Microsoft Windows
- OpenAL cross-platform sound API
- OpenCL cross-platform API for general-purpose computing for CPUs & GPUs
- OpenGL cross-platform graphics API
- OpenMP API that supports multi-platform shared memory multiprocessing

programming in C, C++ and Fortran on many architectures, including Unix and Microsoft Windows platforms.

- Simple DirectMedia Layer (SDL)
- Talend integrates its data management with BPM from Bonita Open Solution

6.2 API USES

KEY NOTES

Basics

An API (Application Programming Interface) is a set of rules and protocols that allows different software applications to communicate and interact with each other. APIs enable developers to access specific functionalities or data from a service, library, or platform without needing to understand the underlying implementation details.

Operating systems

An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs. All computer programs, excluding firmware, require an operating system to function.

Remote APIs

The Remote API to access your application's data store from an app running on your local machine, or from a local interactive Remote API shell.

Use in Web APIs

Web APIs are the defined interfaces through which interactions happen between an enterprise and applications that use its assets.

Leanings Objectives

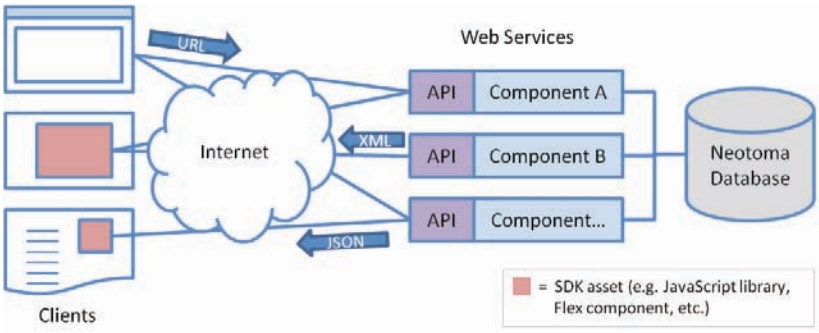
- Basics use of API
- API library and framework
- Web API in ASP.NET

Basics

Basically, an API is a set of protocols, subroutines and tools to develop software programs. Like a graphical interface, the objective of an API is to make the developer's work easier. To achieve this, an API provides remote calls, object classes, data structures or variables, which are all the blocks required for a developer to use certain technologies in his application, reducing the cognitive load to which he is exposed. The maintenance and development of applications that use APIs, becomes more expensive and time consuming, without proper documentation. This, should contain all the relevant information about the services available in the API, as well as the way they work, so the developer knows what the expected response when interacting with it is.

Despite the three components introduced being different from each other, there is a strong connection between an API, library and framework. An API describes a behavior while a library contains all the implementation of the rules used in this behavior.

Regarding the API-framework relation, an API is the public side of a framework, which means, from all the methods implemented in a framework, only the ones relevant to the user will be presented in an API.



Library

A library is composed by a set of rules used by an application, this means the library contains all the implementation for the expected behavior. Example given, when a programmer wants to develop an application, he can use a library to make the system calls instead of implementing them. The code available in the library is independent from the application itself and is structured in a way that it can be used and reused by multiple applications, without these being connected between each other. These libraries, allow the code to be shared by multiple developers, which makes it easy when developing new software projects.

Framework

Last but not least, a framework is composed by a group of libraries that allows to execute a bigger operation. These frameworks can be seen as development platforms, which should be adapted to the developer's requirements. In the end, a framework can be seen as the skeleton of the application, for example, considering a team that is developing a website for a company, by using a framework, they can focus their efforts on specific code for the website itself instead of all the handling and management of the requests.

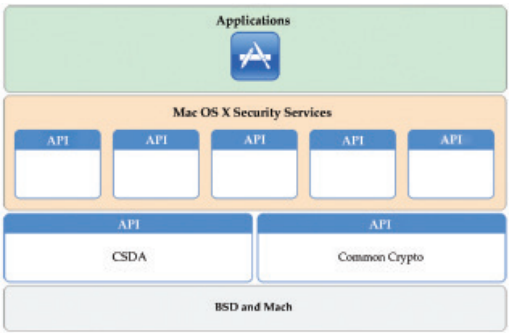
Operating systems

An API can specify the interface between an application and the operating system. POSIX, for example, specifies a set of common APIs that aim to enable an application written for a POSIX conformant operating system to be compiled for another POSIX conformant operating system. Linux and Berkeley Software Distribution are examples of operating systems that implement the POSIX APIs. Microsoft has shown a strong commitment to a backward-compatible API, particularly within their Windows API (Win32) library, so older applications may run on newer versions of Windows using an executable-specific setting called "Compatibility Mode". An API differs from an

application binary interface (ABI) in that an API is source code based while an ABI is binary based. For instance, POSIX provides APIs, while the Linux Standard Base provides an ABI.

- Software developers have demonstrated that many different programs can take advantage of a number of similar modules.

For example, word processors, spreadsheets, presentation software, and database software, although they accomplish quite different tasks, all need to be able to permit the user to save work in progress into a data file, with the user being able to select the disk drive, folder name, file name, and in some cases file format.



Every version of the Macintosh OS and every version of Windows has included, as part of the operating system, a module that provides that functionality.

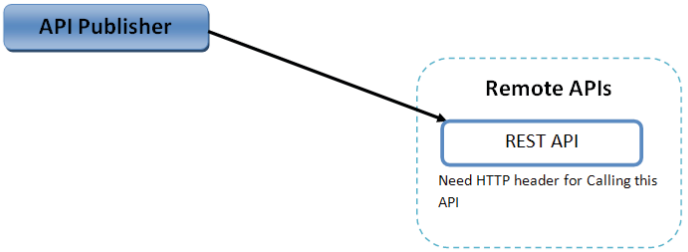
- Software developers writing application programs do not have to write their own module that performs such functions. Instead, they write their software to use the operating system's module.
- Some operating systems provide modules that support graphical user interfaces, other operating systems provide modules that support command line user interfaces, and some provide both.

Remote APIs

Remote APIs allow clients connecting from a separate tier to communicate with the Alfresco content application server. Remote APIs are based on web services and RESTful and CMIS protocols, and are language agnostic, allowing you to develop against these APIs using a range of languages including Java, PHP, Ruby, and .NET.

There are several Remote APIs available, including:

- **Web services** - object-oriented API using SOAP and supporting a range of content services including authentication, query, node creation and update, access control, and actions
- **RESTful** - HTTP-based resource oriented interfaces used by the Surf framework and Alfresco Share
- **CMIS** - a standard that defines web services and REST-based bindings for working with CMIS compliant repositories



A remote API allowing to control a simulation (or the simulator itself) from an external application or a remote hardware (e.g. real robot, remote computer, etc.).

The remote API is composed by approximately one hundred specific functions and one generic function that can be called from a C/C++ application, a Python script, a Java application, a Matlab/Octave program, an Urbi script, or a Lua script. The remote API functions are interacting with via socket communication in a way that reduces lag and network load to a great extent.

All this happens in a hidden fashion to the user. The remote API can let one or several external applications interact with V-REP in a synchronous* or asynchronous* way (asynchronous by default), and even remote control of the simulator is supported (e.g. remotely loading a scene, starting, pausing or stopping a simulation for instance).

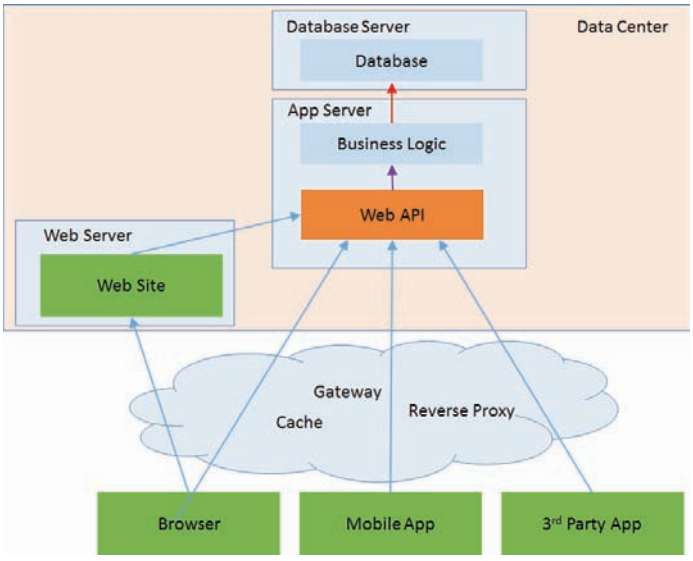
The remote API functionality comes in 2 separate entities that are interacting via socket communication:

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

- The client side (i.e. your application): the remote API on the client side is available for many different programming languages. Currently following languages are supported: C/C++, Python, Java, Matlab, Octave, Urbi and Lua. Bindings for other languages can easily be created by yourself. Instructions on how to enable the remote API on the client side are given here.
- The server side: the remote API on the server side is implemented via a plugin that should be loaded by default: v_repExtRemoteApi.dll, libv_repExtRemoteApi.dylib or libv_repExtRemoteApi.so. The plugin project files are located in installation directory, in the "programming" folder. Instructions on how to enabled the remote API on the server side are given here.

Use in Web APIs

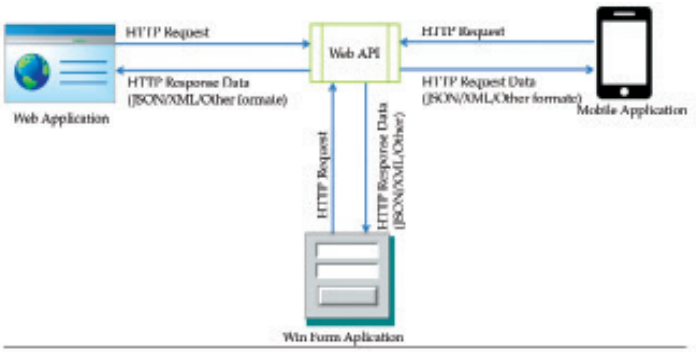
An API approach is an architectural approach that revolves around providing programmable interfaces to a set of services to different applications serving different types of consumers. When used in the context of web development, an API is typically defined as a set of Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages, which is usually in an **Extensible Markup Language (XML)** or JavaScript Object Notation (JSON) format. An example might be a shipping company API that can be added to an eCommerce-focused web site, to facilitate ordering shipping services and automatically include current shipping rates, without the site developer having to enter the shipper's rate table into a web database. While "web API" historically has been virtually synonymous for web service, the recent trend (so-called Web 2.0) has been moving away from Simple Object Access Protocol (SOAP) based web services and service-oriented architecture (SOA) towards more direct representational state transfer (REST) style web resources and resource-oriented architecture (ROA). Part of this trend is related to the Semantic Web movement toward Resource Description Framework (RDF), a concept to promote web-based ontology engineering technologies. Web APIs allow the combination of multiple APIs into new applications known as mashups. In the social media space, web APIs have allowed web communities to facilitate sharing content and data between communities and applications. In this way, content that is created in one place can be dynamically posted and updated in multiple locations on the web.



Web API as the name suggests, is an API over the web which can be accessed using HTTP protocol. It is a concept and not a technology. We can build Web API using different technologies such as Java, .NET etc. For example, Twitter's REST APIs provide programmatic access to read and write data using which we can integrate twitter's capabilities into our own application.

Web API in ASP.NET

The ASP.NET Web API is an extensible framework for building HTTP based services that can be accessed in different applications on different platforms such as web, windows, mobile etc. It works more or less the same way as ASP.NET MVC web application except that it sends data as a response instead of html view. It is like a webservice or WCF service but the exception is that it only supports HTTP protocol.



ASP.NET Web API Characteristics:

- ASP.NET Web API is an ideal platform for building RESTful services.
- ASP.NET Web API is built on top of ASP.NET and supports ASP.NET request/response pipeline
- ASP.NET Web API maps HTTP verbs to method names.
- ASP.NET Web API supports different formats of response data. Built-in support for JSON, XML, BSON format.
- ASP.NET Web API can be hosted in IIS, Self-hosted or other web server that supports .NET 4.0+.
- ASP.NET Web API framework includes new HttpClient to communicate with Web API server. HttpClient can be used in ASP.MVC server side, Windows Form application, Console application or other apps.

Aspect of API Documentation

API documentation is a crucial aspect of any well-designed API. It provides developers with the information they need to understand how to use the API effectively and integrate it into their applications. Clear and comprehensive documentation can make the difference between a successful API adoption and confusion for developers. Here's an overview of what API documentation should include:

- **Introduction:**
  - Brief overview of what the API does and its main functionalities.
  - Explanation of why the API exists and how it benefits developers.
- **Getting Started:**

  - Instructions on how to obtain API credentials (API keys, authentication tokens, etc.).
  - Basic example of making a simple API request.
- **Authentication and Authorization:**
  - Detailed instructions on how to authenticate with the API.
  - Explanation of different authentication methods (OAuth, API keys, JWT, etc.).
  - Information on required headers, tokens, and how to manage access rights.
- **Endpoints and Requests:**
  - List of available API endpoints and their purpose.
  - Explanation of HTTP methods (GET, POST, PUT, DELETE) and when to use them.
  - Description of the required parameters and request payloads for each endpoint.
  - Example requests and responses.
- **Response Format:**
  - Explanation of the response format (JSON, XML, etc.).
  - Description of the different fields in the response and their meanings.
  - Information on error responses and status codes.
- **Rate Limiting and Usage Policies:**
  - Details about rate limiting, throttling, and usage quotas.
  - Guidelines on how to handle rate-limiting errors.
- **Data Models and Schemas:**
  - Definition of data structures used in requests and responses.
  - Code snippets or sample payloads that demonstrate data structure.
- **Code Examples:**
  - Comprehensive examples of API requests and responses in different programming languages.
  - Illustrations of common use cases to help developers understand how to implement API calls.
- **Best Practices:**
  - Recommendations on how to structure API requests and responses.
  - Guidelines for error handling, error messages, and graceful degradation.
- **Advanced Features:**
  - Documentation for more complex or specialized features of the API.
  - Explanation of any optional query parameters or headers.
- **Versioning:**

  - Explanation of versioning strategies and how to specify the API version.
  - Guidelines for managing backward compatibility.
- **FAQ and Troubleshooting:**
  - Answers to frequently asked questions.
  - Solutions to common issues and troubleshooting tips.
- **Changelog:**
  - Record of changes, updates, and improvements to the API over time.
  - Information on deprecated endpoints or features.
- **Contact Information:**
  - How to get in touch with the API provider's support or developer relations team.
- **Terms of Use and Legal Information:**
  - Information about usage policies, terms of service, and any legal considerations.

Remember that well-organized and easily searchable documentation is key. It should provide developers with the information they need quickly and efficiently, making it easier for them to integrate your API into their projects.

6.3 API TESTING

KEY NOTES

Strategy

API testing is entirely different from GUI testing and mainly concentrates on the business logic layer of the software architecture. This testing won't concentrate on the look and feel of an application.

Start API testing

The first part of API testing involves setting up a testing environment, with the required set of parameters around the API. This involves configuring the database and server for the application's requirements.

Learning Objectives

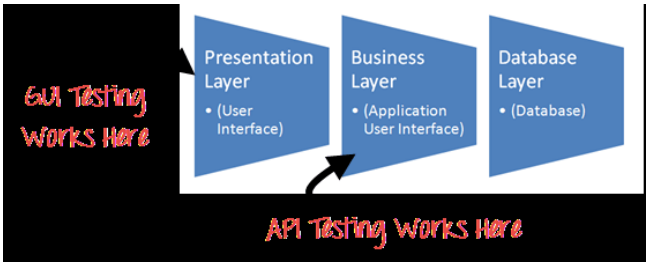
- Set-up of API Test environment
- Common Tests performed on API's
- Challenges with API Testing
- Advantages of API Testing

Strategy

API testing is a type of software testing that involves testing application programming interfaces (APIs) directly and as part of integration testing to determine if they meet expectations for functionality, reliability, performance, and security. Since APIs lack a GUI, API testing is performed at the message layer. API testing is now considered critical for automating testing because APIs now serve as the primary interface to application logic and because GUI tests are difficult to maintain with the short release cycles and frequent changes commonly used with Agile software development and DevOps).

The API Testing is performed for the system, which has a collection of API that ought to be tested. During Testing, a test of following things is looked at.

- Exploring boundary conditions and ensuring that the test harness varies parameters of the API calls in ways that verify functionality and expose failures.
- Generating more value added parameter combinations to verify the calls with two or more parameters.
- Verifying the behaviour of the API which is considering the external environment conditions such as files, peripheral devices, and so forth.
- Verifying the Sequence of API calls and check if the API's produce useful results from successive calls.



Instead of using standard user inputs (keyboard) and outputs, in API Testing, you use software to send calls to the API, get output, and note down the system's response.

API Testing requires an application to interact with API. In order to test an API, you will need to

- Use Testing Tool to drive the API
- Write your own code to test the API

Set-up of API Test environment

- API testing is different than other testing types as GUI is not available, and yet you are required to setup initial environment that invokes API with required set of parameters and then finally examines the test result.
- Hence, Setting up a testing environment for API testing seems a little complex.
- Database and server should be configured as per the application requirements.
- Once the installation is done, API Function should be called to check whether that API is working.

Common Tests performed on API's

- Return Value based on input condition - The return value from the API's are checked based on the input condition.
- Verify if the API's does not return anything.
- Verify if the API triggers some other event or calls another API. The Events output should be tracked and verified.
- Verify if the API is updating any data structure.

Challenges with API Testing

An API is an interface intended to be interpreted by a computer. Since the interface boundary has to be technology agnostic, the inputs and outputs are normally text-formatted messages. Text messages may be easy for a human to read in order to interpret the inputs and outputs, but in-depth knowledge of the internals of the application is required to sufficiently test it.

This is the difference between black-box testing and white-box testing. In black-box testing, the tester is only concerned with the functionality of the overall application,

exercising only the inputs and checking the outputs.

In white-box testing, the tester is much more concerned with internal operations of the application, exercising the individual paths the data can travel through the application.

API testing normally includes only the white-box testing approach.

Another difficulty is that testers are used to working from a set of provided business requirements. However, business requirements normally describe use cases at a fairly high level, and do not go down to the level of individual application functions.

It is not unusual that business analysts are not even aware of low level detail and the amount of granularity that everything has to be broken down into, and are not aware that software is able to be tested at this level.

Typically, project managers are not concerned with assigning time specifically to developing a rich and detailed API, let alone testing it. Paradoxically, if everything is done correctly, the API is the only place where business rules are coded and enforced. The end-user GUI only displays results and any error messages provided by the API.

Sadly, because of the above problems, developers are often left with testing their own code, if there is any testing at this level at all. And due to time pressures, unit tests often only glance at the positive tests – the happy path – and more in depth testing is left until later in the project.

Of course, looking at only the positive paths through the application is not really testing, but only checking that the software fulfils business requirements under the most optimal conditions.

The **Graphical user interface (GUI)** usually exposes only part of the overall functionality and by design testers are not able to go really deep into the application code, especially when it comes to testing all the negative test cases. This leaves the internals of the application possibly exposed to all sorts of breakage and attacks by end-users, accidental or malicious.

Graphical user interface (GUI) is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation.

Lastly, there are situations where the API is the final product – a public API. A simple Internet search will reveal there are thousands of public APIs for anything from GPS and mapping solutions to locating a radio station that is playing your favorite

song right now. However, these APIs expose functionality that the developer hopes or thinks other developers/ users will want. In this case, the tester's role needs also to include a user advocate role, to make sure the offering is complete – everything that any user may need is all there. This is in addition to the multitude of hats, the tester already wears: functional testing, security testing, load testing, etc.

Advantages of API Testing

Putting more effort into API testing leads to a much healthier final product. Ensuring that all data access (read and write) goes only through the API significantly simplifies security and compliance testing and thereby certification, since there is only one interface.

Ensuring that all the required business rules are being enforced at the API tier allows time for much more complete user-experience tests once the UI is released, and not having to concentrate on testing every single business rule and path through the application near the end of the project. Ensuring that the API offers complete functionality allows for easy future expansion of the application as new business needs arise.

Start API testing

Once you've set up your API testing environment, make an API call right away to make sure nothing is broken before you go forward to start your more thorough testing.

You can start combining your application data with your API tests to ensure that the API performs as expected against possible known input configurations. See this REST Testing example for beginners to help with the visualization.

Next, you need to organize yourself around the API test. Start by asking yourself these questions:

- Who is your target audience? Who is your API consumer?
- What environment/s should the API typically be used?
- What aspects are you testing?
- What problems are we testing for?
- What are your priorities to test?
- What is supposed to happen in normal circumstances?
- What could potentially happen in abnormal circumstances?
- What is defined as a Pass or a Fail? What data is the desired output? What is the chain of events?
- What other APIs could this API interact with?
- Who on your team is in charge of testing what?

After you've created these testing boundaries and requirements, you need to decide what you want to test your API for. These are some of the common kinds of API testing:

- Functionality testing — the API technically works.
- Usability testing — the API is easy to work with.
- Reliability testing — the API can be consistently connected to and lead to consistent results.
- Load testing — the API can handle a large amount of calls.
- Creativity testing — the API can handle being used in different ways.
- Security testing — the API has defined security requirements including authentication, permissions and access controls. See some API security tips for protectiving vital data.
- Proficiency testing — the API increases what developers are able to do.
- API documentation testing — also called discovery testing, the API documentation easily guides the user.

Tests will surely vary but here are common API test examples:

- Checking API return values based on the input condition.
- Verifying if the API doesn't return anything at all or the wrong results.
- Verifying if the API triggers some other event or calls another API.
- Verifying if the API is updating any data structures.



TECHNIQUE 21

API DOCUMENTATION

API documentation describes what services an API offers and how to use those services, aiming to cover everything a client would need to know to use the API. Documentation is crucial for the development and maintenance of applications that use the API. API documentation is traditionally found in documentation files, but can also be found in social media such as blogs, forums, and Q&A websites. Traditional documentation files are often presented via a documentation system, such as Javadoc or Pydoc, that has a consistent appearance and structure. However, the types of content included in the documentation differs from API to API

- ☐ We already practice this technique
- ☐ This techniques would suitable or not for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 22

TESTING REST API MANUALLY

Rest stands for Representational State Transfer. It is an architectural style and an approach for communication used in the development of Web Services. REST has become a logical choice for building APIs. It enables users to connect and interact with cloud services efficiently.

- ☐ We already practice this technique
- ☐ This techniques would suitable or not for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 23

ACCESSING APP ENGINE WITH REMOTE API

The Remote API library allows any Python client to access services available to App Engine applications. For example, if your App Engine application uses Google Cloud Data-store or Google Cloud Storage, a Python client could access those storage resources using the Remote API. You can use the Remote API to access your application's data store from an app running on your local machine, or from a local interactive Remote API shell. The Remote API interacts with real services, so this access does use quota and billable resources.

- ☐ We already practice this technique
- ☐ This techniques would suitable or not for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 24

API LIFECYCLE MANAGEMENT

API Lifecycle Management refers to the process of managing an API from its conception through design, development, testing, deployment, maintenance, and eventual retirement. Effectively managing the entire lifecycle ensures that the API remains reliable, secure, and useful to its users.

- ☐ We already practice this technique
- ☐ This techniques would suitable or not for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique

SUMMARY

- An Application Programming Interface (API) is a set of protocols, functions, and tools that allows different software applications to communicate with each other.
- APIs provide a well-defined structure for how different software components interact. They specify the methods and data formats that developers should use when making requests to access a service's functionality.
- Web APIs, in particular, have gained immense popularity. They expose endpoints that developers can access through HTTP requests (such as GET, POST, PUT, DELETE).
- The API in this case can be conceived as the totality of all the methods publicly exposed by the classes (usually called the class interface).
- API documentation is a crucial aspect of any well-designed API. It provides developers with the information they need to understand how to use the API effectively and integrate it into their applications.
- An API is usually related to a software library: the API describes and prescribes the expected behavior while the library is an actual implementation of this set of rules.
- An object API can prescribe a specific object exchange format, an object exchange protocol can define a way to transfer the same kind of information in a message sent to a remote system.
- A compatible API, on the other hand, allows compiled object code to function without any changes to the system implementing that API.
- Basically, an API is a set of protocols, subroutines and tools to develop software programs. Like a graphical interface, the objective of an API is to make the developer's work easier.
- An API approach is an architectural approach that revolves around providing programmable interfaces to a set of services to different applications serving different types of consumers.

KNOWLEDGE CHECK

1. In order to transfer data between a database and an application written in the Java programming language, the JDBC API provides which of these methods?

a. Methods on the ResultSet class for retrieving SQL SELECT results as Java types.

b. Methods on the PreparedStatement class for sending Java types as SQL statement parameters.

c. Methods on the CallableStatement class for retrieving SQL OUT

parameters as Java types.

d. All mentioned above

2. Which of the following component of HTTP request is used to identify the resource on server?

a. VERB

b. URI

c. HTTP Version

d. Request Header

3. What are the tools used for API testing?

a. SoapUI Pro

b. PostMan

c. Alertsite API monitoring

d. All mentioned above

4. Which of the following is a best practice for designing a secure RESTful web service?

a. No sensitive data in URL - Never use username, password or session token in URL , these values should be passed to Web Service via POST method.

b. Restriction on Method execution - Allow restricted use of methods like GET, POST, DELETE. GET method should not be able to delete data.

c. Both of the above.

d. None of the above.

5. Request Body - Contains Message content or Resource representation.

a. True

b. False

6. What is the purpose of API documentation?

a. To showcase the aesthetic design of the API.

b. To provide information about the internal implementation of the API.

c. To guide developers on how to use the API and its functionalities.

d. To provide marketing material for the API.

7. Which HTTP method is typically used for retrieving data from an API?

a. POST

b. GET

c. PUT

d. DELETE

8. What is the role of authentication in API usage?

a. It ensures that the API is free of errors.

b. It helps prevent unauthorized access to the API's functionalities.
c. It controls the speed at which API requests are made.
d. It enhances the visual design of the API.

REVIEW QUESTIONS

1. Define the role of API in object-oriented languages.
2. Discuss on API libraries and frameworks.
3. Give an overview on API and protocols
4. What do you understand by the remote APIs?
5. Identify the challenges in API testing.

Check Your Results

1. (d)	2. (a)	3. (d)	4. (c)	5. (a)
6. (c)	7. (b)	8. (b)		

REFERENCES

1. Benslimane, D.; Dustdar, S.; Sheth, A. (2008). "Services Mashups: The New Generation of Web Applications". IEEE Internet Computing. 10 (5): 13–15. doi:10.1109/MIC.2008.110.
2. Bloch, Joshua (2008). "Effective Java (2nd edition)" (http://java.sun.com/docs/books/effective/). Addison-Wesley. pp. 259–312.
3. Microsoft (October 2001). "Run Older Programs On Windows XP" (http://www.microsoft.com/windowsxp/using/helpandsupport/ learnmore/appcompat.mspx). Microsoft. p. 4. .
4. Mulloy, Brian. Web API Design - Crafting Interfaces that Developers Love (PDF). apigee. p. 11.
5. Niccolai, James (2008-04-23), "So What Is an Enterprise Mashup, Anyway?" (http://www.pcworld.com/businesscenter/article/145039/ so_what_is_an_enterprise_mashup_anyway.html), PC World, "Dynamic Community content via APIs". October 26, 2009.
6. Stoughton, Nick (April 2005). "Update on Standards" (https://db.usenix.org/publications/login/2005-04/openpdfs/standards2004.pdf) (PDF). USENIX. . Retrieved 2009-06-04.

CHAPTER 7

COLDFUSION MX

7.1 CONCEPT OF COLDFUSION MX

KEY NOTES

Definition

ColdFusion MX is a powerful web application server that lets you create robust sites and applications without a long learning curve. ColdFusion MX does not require coding in traditional programming languages (for example, C/C++, Java, XML), although it supports these traditional programming languages.

Flash Remoting Services

The Flash Remoting service of ColdFusion MX, ColdFusion developers can work together with Macromedia Flash MX designers to build Flash UIs for ColdFusion applications. Building Flash UIs requires the separation of UI code from business logic code. User interface controls are built in Flash MX, while business logic is built in ColdFusion.

Web Services

A web services are any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service.

Learning Objective

- Define native XML handling
- Describe simplified security in ColdFusion MX

Definition

ColdFusion is one of the easiest-touse programming languages, yet it can build the most powerful, extensible Web applications in the world. ColdFusion's elegant simplicity is appreciated by both novice programmers who want a fast start on building real-world applications and experienced developers coming from complicated Web technologies such as Active Server Pages.

ColdFusion is a simple language to learn and use primarily because it employs a tag-based syntax, which makes the transition between HTML (the language that your Web browser uses to render content) and CFML (**ColdFusion Markup Language**, the language that ColdFusion uses to generate that content from dynamic data) within your code very natural. Take, for example, the following HTML listing of companies (no CFML is in this code):

ColdFusion Markup Language more commonly known as CFML, is a scripting language for web development that runs on the JVM, the .NET framework, and Google App Engine.

```
<table>
   <tr>
     <td>13</td>
     <td>The Very Big Corporation of America</td>
   </tr>
   <tr>
     <td>14</td>
     <td>Ma's Homemade Pies</td>
   </tr>
   <tr>
     <td>15</td>
     <td>Shecky Records</td>
   </tr>
</table>
```

By using ColdFusion's tag-based syntax, you can easily generate this table dynamically from data stored in your database (the CFML is boldfaced in the following example), as follows:

```
<cfquery name="GetCcompanies" datasource="MyDatabase">
SELECT
  CompanyID,
  CompanyName
FROM
  Company
</cfquery>

<table>
  <cfoutput query="GetCompanies">
   <tr>
     <td>#CompanyID#</td>
     <td>#CompanyName#</td>
   </tr>
  </cfoutput>
</table>
```

Basically, ColdFusion performs a query against the MyDatabase datasource (an object that connects to and communicates with a database) and names the data result set (the data that results from performing the query) GetCompanies. Then, for each row in the GetCompanies query, ColdFusion outputs the HTML markup used to format an HTML table row containing two table cells. The first table cell contains the value of CompanyID, and the second table cell contains the value of CompanyName from the current row in the result set.

Flash Remoting Services

For anyone who has succeeded in interfacing Flash movies with ColdFusion, MX's new Flash Remoting Services are a true godsend. What used to require a good bit of knowledge in Flash ActionScripting (the language of Flash), plus hours or days of laboring in Flash and ColdFusion MX, now takes only a basic understanding of ActionScript and a few lines of code to link a Flash movie with ColdFusion. If you're new to ColdFusion, you'll still be amazed at how easy Flash Remoting makes creating Flash movies that use the functionality built into your ColdFusion applications.

Where earlier versions of Flash were basically crude animation tools that could be forced into exchanging data with ColdFusion, Flash Remoting turns the Flash MX authoring environment into a true client-application development environment that is designed from the ground up to work directly with ColdFusion MX. Flash MX applications communicate with ColdFusion by using a new and very efficient binary format named AMF (Action Message Format), which is not only faster but requires only a fraction of the bandwidth previously taken up by XMLbased data exchanges.

Flash Remoting is able to handle complex datatypes, such as objects, structures, arrays, and recordsets.

Flash Remoting enables you to develop applications with truly interactive graphical user interfaces. The best that you could do before Flash Remoting came along was to create an overly complicated DHTML convolution that either cached data locally or made constant page requests from ColdFusion each time that the user touched a form control.

Web Services

A **Web service** exposes a software component to remote systems in a platform-independent manner, meaning that any

application server platform capable of consuming web services can make use of the web services created with ColdFusion MX. ColdFusion MX truly rang the bell with its implementation of Web services: To create a Web service in MX, you simply set the Access attribute of a ColdFusion component function to Remote.

You need to pay attention to a few details, of course, but creating a Web service is a natural extension to ColdFusion's component functionality. Whenever a ColdFusion component is accessed as a Web service, the remote consumer of the Web service receives the result in the native format of the consumer's platform, so .NET consumers receive ColdFusion arrays as .NET arrays, and Java consumers receive them as Java arrays.

Some incompatibilities do exist between certain ColdFusion complex data types and specific consumer platforms, but these hurdles can be cleared by exchanging complex data as XML documents that the consumer may then transform as needed.

Native XML Handling

XML stands for eXtensible Markup Language, and it provides a way to structure data and encode it with additional information using a plain text format that can be used by most modern application servers and applications. Data interchange with XML was limited to one of the following two options in ColdFusion:

Use WDDX as an interchange format: This was a good start, but it was limited because you were locked in to a data-centric flavor of XML.

Use a COM object or third-part custom tag library: Some good libraries were on the market, but they still are no match for the capability to natively use XML and XML objects.

Now, by using ColdFusion MX, all you need do to create an XML object is use the new CFXML tag, as follows:

```
<cfxml variable="XmlObj">
<my-xml-tag>
   <my-child-tag />
</my-xml-tag>
</cfxml>
```

And that's only the beginning of an impressive array of XML handling features; in fact, the only feature missing from ColdFusion's XML implementation is the capability to validate against a DTD or XMLSchema document, which can be easily remedied by using COM.

ColdFusion MX also natively handles XSLT transformations, so you can transform structured XML documents into virtually any type of content. MX's XPath capabilities enable you to query an XML object and extract data structures that match search criteria. After you parse an XML document into an **XML** object, you can refer to its data elements by using the same ColdFusion syntax used for handling arrays and structures, so your learning curve remains relatively small.

Simplified Security

ColdFusion MX makes security easier to manage in two ways: application security by using the new CFLOGIN tag and the new J2EE-integrated Sandbox Security.

Application Security

ColdFusion MX introduces CFLOGIN, which solves two problems at once. First, it's easy to use and understand, so it takes less development time than a roll-your-own solution. Second, and perhaps more important, CFLOGIN integrates directly with J2EE security, which is a proven security framework that can integrate with other J2EE products. Integrating with J2EE security means that a user can log in to a ColdFusion Web site and have that same login honored by a call to an EJB (Enterprise JavaBean) or a JSP page.

XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

Sandbox Security

Available only in the Enterprise version of ColdFusion MX (although Professional has a similar feature, named Resource Security), Sandbox Security is a great feature for shared hosts or people running a department with multiple applications, because it enables a site administrator to restrict what resources the code on his server can access. The administrator can, for example, restrict the code in one directory from accessing specific tags and datasources, and he can restrict code in another directory from accessing a completely different set of tags and datasources.

Database Integration

Database integration is a fundamental aspect of building dynamic web applications using ColdFusion MX. ColdFusion MX provides powerful tools and features for connecting to databases, executing queries, and manipulating data. Here's a step-by-step overview of how you can integrate databases with ColdFusion MX:

1. Database Connection

ColdFusion uses Data Source Names (DSNs) to connect to databases. Create a DSN to establish a connection between ColdFusion and your database server.

You can create DSNs using the ColdFusion Administrator or define them in the application code.

2. Querying the Database:

Use the <cfquery> tag to execute SQL queries against the database.

Example:

```
<cfquery name=”myQuery” datasource=”yourDSN”>
  SELECT * FROM yourTable
</cfquery>
```

3. Displaying Query Results:

Use <cfoutput> to display the results of your query.

Example:

```
<cfoutput query=”myQuery”>
  #myQuery.columnName#
</cfoutput>
```

4. Parameterized Queries:

To prevent SQL injection attacks, use parameterized queries.

Example:

```
<cfquery name=”paramQuery” datasource=”yourDSN”>
    SELECT * FROM yourTable WHERE id = <cfqueryparam value=”#url.id#” cfsqltype=”cf_sql_integer”>
</cfquery>
```

5. Data Manipulation:

Use <cfquery> to execute INSERT, UPDATE, DELETE statements.

Example:

```
<cfquery datasource=”yourDSN”>
  INSERT INTO yourTable (column1, column2) VALUES (‘value1’, ‘value2’)
</cfquery>
```

6. Transactions:

Wrap multiple queries in a transaction using <cftransaction> to ensure data consistency.

Example:

```
<cftransaction>
```

```
  <cfquery datasource=”yourDSN”>
    <!-- First query -->
  </cfquery>
  <cfquery datasource=”yourDSN”>
    <!-- Second query -->
  </cfquery>
</cftransaction>
```

7. Error Handling:

Implement error handling using <cftry> and <cfcatch> to manage database errors gracefully.

Example:

```
<cftry>
  <cfquery datasource=”yourDSN”>
    <!-- Your query -->
  </cfquery>
  <cfcatch type=”Database”>
    <!-- Handle database error -->
  </cfcatch>
</cftry>
```

8. Stored Procedures:

Call stored procedures using the <cfstoredproc> tag.

Example:

```
<cfstoredproc procedure=”yourProcedure” datasource=”yourDSN”>
  <cfprocparam type=”IN” value=”paramValue” cfsqltype=”cf_sql_varchar”>
  <cfprocparam type=”OUT” variable=”outputValue” cfsqltype=”cf_sql_integer”>
</cfstoredproc>
```

Remember to replace placeholders like yourDSN, yourTable, columnName, etc., with your actual database information and structure. ColdFusion MX’s database integration capabilities make it convenient to work with databases, execute queries, and manage data within your web applications.

7.2 ELEMENTS OF CFML

KEY NOTES

Definition

The basic elements of CFML, including tags, functions, constants, variables, expressions, and CFScript, make it a powerful tool for developing interactive web applications.

Comments

Commenting in ColdFusion is different most other languages. The comment is very much like the HTML comment, but that extra - is super important. Don't forget it. The ColdFusion comment doesn't validate in HTML either, so be sure to not to use the ColdFusion comment on strictly HTML pages.

ColdFusion Components

ColdFusion components, also known as CFCs, bring some object-like behavior to ColdFusion. In their simplest implementation, components are formal containers for interrelated functions created by using CFFUNCTION. By encapsulating all functions related to a specific process or entity in a single component.

Data types

ColdFusion variables can hold many different data types. A data type indicates what kind of information is being stored. ColdFusion variables can store data types, such as numbers, text strings, dates, times, and Boolean values, just to name a few.

Learning Objective

- Identify ColdFusion tags
- Describe how to create custom tags

Definition

CFML is a dynamic application development tool with many of the features of a programming language, including functions, expressions, variables and constants, and flow-control constructs, such as if-then and loops. CFML also has a "language within a language," CFScript, which enables you to use a syntax similar to JavaScript for many operations.

These elements and other basic CFML entities such as

- comments,
- data types,
- escape characters,
- and reserved words

Comments

ColdFusion comments have a similar format to HTML comments. However, they use three dash characters instead of two; for example:

```
<!--- This is a ColdFusion Comment. Browsers do not receive it. --->
```

The ColdFusion server removes all ColdFusion comments from the page before returning it to the web server. As a result, the page that a user's browser receives does not include the comment, and users cannot see the comment even if they view the page source.

You can embed CFML comments in begin tags (not just tag bodies), functions calls, and variable text in number signs. ColdFusion ignores the text in comments such as the following:

```
<cfset MyVar = var1 <!--- & var2 --->>
<cfoutput>#Dateformat(now() <!---, "dddd, mmmm yyyy" --->)#</cfoutput>
```

This technique can be useful if you want to temporarily comment out parts of expressions or optional attributes or arguments. You can also nest comments, as the following example shows:

```
<!--- disable this code
<!--- display error message --->
<cfset errormessage1="Oops!">
<cfoutput>
#errormessage1#
</cfoutput>
--->
```

This is useful if you want to temporarily disable a section of code while you test your application. You can embed comments within comments, however, you should use this technique carefully.

Note: You cannot embed comments inside a tag name or function name, such as <cf_My<!--- New --->CustomTag>.

You also cannot embed comments inside strings, as in the following example: IsDefined("My<!--- New ---
>Variable").

Tags

ColdFusion tags tell the ColdFusion server that it must process information. The ColdFusion server only processes tag contents; it returns text outside of ColdFusion to the web server unchanged. ColdFusion provides a wide variety of built-in tags and lets you create custom tags.

Tag syntax

ColdFusion tags have the same format as HTML tags. They are enclosed in angle brackets (< and >) and can have zero or more named attributes. Many ColdFusion tags have bodies; that is, they have beginning and end tags with text to be processed between them. For example:

```
<cfoutput>
    Hello #YourName#! <br>
</cfoutput>
```

Assignment statement in programming is used to assign a value to a variable. It's a fundamental operation that allows you to store data in variables for further manipulation and use.

Other tags, such as cfset and cfhttp, never have bodies; all the required information goes between the beginnings (<) character and the ending (>) character, as in the following example:

```
<cfset YourName="Bob">
```

Note: The cfset tag differs from other tags in that it has neither a body nor arguments. Instead, the tag encloses an assignment statement that assigns a value to a variable. The cfset tag can also call a function without assigning a value to a result variable.

Sometimes, although the tag can have a body, you do not need to put anything in it because the attributes specify all the required information. You can omit the end tag and put a forward slash character before the closing (>) character, as in the following example:

<cfprocessingdirective pageencoding="euc-jp" />

In most cases, you specify tag attributes directly in the tag using the format attributeName="attributeValue", as the preceding example shows. However, as an alternative, you can put all the attributes in a structure and specify the structure in a single attributeCollection attribute, using the following format:

<tagname attributeCollection="#structureName#">

When you use this format for all built-in ColdFusion tags except cfmodule, the tag must have only the attributeCollection attribute. This format is useful when you use dynamic arguments, where the number and values of the arguments to a tag can vary based on processing results.

The following example shows this usage:

<!--- Configure dynamic attribute variables. --->

<cfparam name="theURL" default="http://www.adobe.com">

<cfparam name="resolveURL" default="yes">

<!--- Code that dynamically changes values for attributes might go here. --->

<!--- Create an arguments structure using variables. --->

<cfset myArgs=StructNew()>

```
<cfset myArgs.url="#theURL#">
<!--- Include a user name and password only if they are available. --->
<cfif IsDefined("username")>
<cfset myArgs.username="#username#">
</cfif>
<cfif IsDefined("password")>
<cfset myArgs.password="#password#">
</cfif>
<cfset myArgs.resolveURL="#resolveURL#">
<cfset myArgs.timeout="2">
<!--- Use the myArgs structure to specify the cfhttp tag attributes. --->
<cfhttp attributeCollection="#myArgs#">
<cfoutput>
#cfhttp.fileContent#
</cfoutput>
```

Note: The attributeCollection attribute used in the cfmodule tag and when calling custom tags directly is different from the attributeCollection attribute for all other tags. In the cfmodule tag and in custom tags, you can mix the attributeCollection attribute and explicit custom tag attributes. Also, in the cfmodule tag, the attributeCollection attribute cannot contain the name and template attributes. You must specify these attributes directly in the cfmodule tag.

You can use the attributeCollection attribute in all tags except the following:

cfargument	cfelseif	cflogout	cfset
cfbreak	cffunction	cfloop	cfsilent
cfcase	cfif	cfparam	cfswitch
cfcatch	cfimport	cfprocessingdirective	cftry
cfcomponent	cfinterface	cfproperty	
cfdefaultcase	cflogin	cfrethrow	
cfelse	cfloginuser	cfreturn	

Built-in Tags

Built-in tags make up the heart of ColdFusion. These tags have many uses, including the following:

- Manipulating variables
- Creating interactive forms
- Accessing and manipulating databases
- Displaying data

- Controlling the flow of execution on the ColdFusion page
- Handling errors
- Processing ColdFusion pages
- Managing the CFML application framework
- Manipulating files and directories
- Using external tools and objects, including Verity collections, COM, Java, and CORBA objects, and executable programs
- Using protocols, such as mail, http, ftp, and pop

Custom Tags

ColdFusion lets you create custom tags. You can create two types of custom tags:

- CFML custom tags that are ColdFusion pages
- CFX tags that you write in a programing language such as Java or C++

Custom tags can encapsulate frequently used business logic or display code. These tags enable you to place frequently used code in one place and call it from many places. Custom tags also let you abstract complex logic into a single, simple interface. They provide an easy way to distribute your code to others; you can even distribute encrypted versions of the tags to prevent access to the tag logic.

CFML Custom Tags

When you write a custom tag in CFML, you can take advantage of all the features of the ColdFusion language, including all built-in tags and even other custom tags. CFML custom tags can include body sections and end tags. Because they are written in CFML, you do not need to know a programming language such as Java. CFML custom tags provide more capabilities than **user-defined functions**, but are less efficient.

User-defined functions are a function provided by the user of a program or environment, in a context where the usual assumption is that functions are built into the program or environment.

CFX Tags

CFX tags are ColdFusion custom tags that you write in a programming language such as Java or C++. These tags can take full advantage of all the tools and resources provided by these languages, including their access to runtime environments. CFX tags also generally execute faster than CFML custom tags

because they are compiled. CFX tags can be cross-platform, but are often platform-specific, for example if they take advantage of COM objects or the Windows API.

Functions

Functions typically manipulate data and return a result. You can also create user-defined functions (UDFs), sometimes referred to as custom functions.

Functions have the following general form:

functionName([argument1[, argument2]]...)

Some functions, such as the Now function take no arguments. Other functions require one or more comma-separated arguments and can have additional optional arguments. All ColdFusion functions return a value. For example, Round(3.14159) returns the value 3.

Built-in Functions

ColdFusion built-in functions perform a variety of tasks, including, but not limited to, the following:

- Creating and manipulating complex data variables, such as arrays, lists, and structures
- Creating and manipulating queries
- Creating, analyzing, manipulating, and formatting strings and date and time values
- Evaluating the values of dynamic data
- Determining the type of a variable value
- Converting data between formats
- Performing mathematical operations
- Getting system information and resources

For alphabetical and categorized lists of ColdFusion functions.

You use built-in functions throughout ColdFusion pages. Built-in functions are frequently used in a cfset or cfoutput tag to prepare data for display or further use. For example, the following line displays today's date in the format October 24, 2007:

<cfoutput>#DateFormat(Now(), "mmmm d, yyyy")#</cfoutput>

Note that this code uses two nested functions. The Now function returns a ColdFusion date-time value representing the current date and time. The DateFormat function takes the value returned by the Now function and converts it to the desired string representation.

Functions are also valuable in CFScript scripts. ColdFusion does not support ColdFusion tags in CFScript, so you must use functions to access ColdFusion functionality in scripts.

User-Defined Functions

You can write your own functions, user-defined functions (UDFs). You can use these functions in ColdFusion expressions or in CFScript. You can call a user-defined function anywhere you can use a built-in CFML function. You create UDFs using the cffunction tag or the CFScript function statement. UDFs that you create using the cffunction tag can include ColdFusion tags and functions. UDFs that you create in CFScript can only include functions. You can create stand-alone UDFs or encapsulate them in a ColdFusion component.

User-defined functions let you encapsulate logic and operations that you use frequently in a single unit. This way, you can write the code once and use it multiple times. UDFs ensure consistency of coding and enable you to structure your CFML more efficiently. Typical user-defined functions include mathematical routines, such as a function to calculate the logarithm of a number; string manipulation routines, such as a function to convert a numeric monetary value to a string such as "two dollars and three cents"; and can even include encryption and decryption routines.

ColdFusion Components

ColdFusion components encapsulate multiple, related, functions. A ColdFusion component is essentially a set of related user-defined functions and variables, with additional functionality to provide and control access to the component contents. ColdFusion components can make their data private, so that it is available to all functions (also called methods) in the component, but not to any application that uses the component.

ColdFusion components have the following features:

- They are designed to provide related services in a single unit.
- They can provide web services and make them available over the Internet.
- They can provide ColdFusion services that Flash clients can call directly.
- They have several features that are familiar to object-oriented programmers, including data hiding, inheritance, packages, and introspection.

Constants

The value of a constant does not change during program execution. Constants are simple scalar values that you can use within expressions and functions, such as "Robert Trent Jones" and 123.45. Constants can be integers, real numbers, time and date values, Boolean values, or text strings. ColdFusion does not allow you to give names to constants.

Variables

Variables are the most frequently used operands in ColdFusion expressions. Variable values can be set and reset, and can be passed as attributes to CFML tags. Variables can

be passed as parameters to functions, and can replace most constants. ColdFusion has a number of built-in variables that provide information about the server and are returned by ColdFusion tags.

The following two characteristics classify a variable:

- The scope of the variable, which indicates where the information is available and how long the variable persists
- The data type of the variable's value, which indicates the kind of information a variable represents, such as number, string, or date

Variable Scopes

The following table describes ColdFusion variable scopes:

Scope	Description
Variables (local)	The default scope for variables of any type that are created with the cfset and cfparam tags. A local variable is available only on the page on which it is created and any included pages.
Form	The variables passed from a form page to its action page as the result of submitting the form.
URL	The parameters passed to the current page in the URL that is used to call it.
Attributes	The values passed by a calling page to a custom tag in the custom tag's attributes. Used only in custom tag pages.
Caller	A reference, available in a custom tag, to the Variables scope of the page that calls the tag. Used only in custom tag pages.
ThisTag	Variables that are specific to a custom tag, including built-in variables that provide information about the tag. Used only in custom tag pages. A nested custom tag can use the cfassociate tag to return values to the calling tag's ThisTag scope.
Request	Variables that are available to all pages, including custom tags and nested custom tags, that are processed in response to an HTTP request. Used to hold data that must be available for the duration of one HTTP request.
CGI	Environment variables identifying the context in which a page was requested. The variables available depend on the browser and server software.
Cookie	Variables maintained in a user's browser as cookies.
Client	Variables that are associated with one client. Client variables let you maintain state as a user moves from page to page in an application and are available across browser sessions.
Session	Variables that are associated with one client and persist only as long as the client maintains a session.

Application	Variables that are associated with one, named, application on a server. The Application.cfc initialization code or the cfapplication tag name attribute specifies the application name.
Server	Variables that are associated with the current ColdFusion server. This scope lets you define variables that are available to all your ColdFusion pages, across multiple applications.
Flash	Variables sent by an Adobe Flash movie to ColdFusion and returned by ColdFusion to the movie.
Arguments	Variables passed in a call to a user-defined function or ColdFusion component method.
This	Variables that are declared inside a ColdFusion component or in a cffunction tag that is not part of a ColdFusion component.
function local	Variables that are declared in a user-defined function and exist only while the function executes.

Expressions

Concatenation operator is used to combine or join strings together. In ColdFusion, the concatenation operator is represented by the plus sign (+). It allows you to combine two or more strings into a single string.

ColdFusion expressions consist of operands and operators. Operands are comprised of constants and variables, such as "Hello" or MyVariable. Operators, such as the string concatenation operator (&) or the division operator (/) are the verbs that act on the operands. ColdFusion functions also act as operators. The simplest expression consists of a single operand with no operators. Complex expressions consist of multiple operands and operators. For example, the following statements are all ColdFusion expressions:

12

MyVariable

(1 + 1)/2

"father" & "Mother"

Form.divisor/Form.dividend

Round(3.14159)

Data Types

ColdFusion is considered typeless because you do not explicitly specify variable data types.

However, ColdFusion data, the constants and the data that variables represent, do have data types, which correspond to the ways the data is stored on the computer.

ColdFusion data belongs to the following type categories:

Category	Description and Types
Simple	Represents one value. You can use simple data types directly in ColdFusion expressions. ColdFusion simple data types are: • strings A sequence of alphanumeric characters enclosed in single or double quotation marks, such as "This is a test." • integers A sequence of numbers written without quotation marks, such as 356. • real numbers, such as -3.14159 • Boolean values Use True, Yes, or 1 for true and False, No, or 0 for false. Boolean values are not case-sensitive. • date-time values ColdFusion supports a variety of data formats.
Complex	A container for data. Complex variables generally represent more than one value. ColdFusion built-in complex data types are: • arrays • structures • queries
Binary	Raw data, such as the contents of a GIF file or an executable program file.
Object	COM, CORBA, Java, web services, and ColdFusion Component objects: Complex objects that you create and access using the cfobject tag and other specialized tags.

Flow Control

ColdFusion provides several tags that let you control how a page gets executed. These tags generally correspond to programming language flow control statements, such as if, then, and else. The following tags provide ColdFusion flow control:

Tags	Purpose
cfif, cfelseif, cfelse	Select sections of code based on whether expressions are True or False.
cfswitch, cfcase, cfdefaultcase	Select among sections of code based on the value of an expression. Case processing is not limited to True and False conditions.

cfloop, cfbreak	Loop through code based on any of the following values: entries in a list, keys in a structure or external object, entries in a query column, an index, or the value of a conditional expression.
cfabort, cfexit	End processing of a ColdFusion page or custom tag.

cfif, cfelseif, and cfelse

The cfif, cfelseif, and cfelse tags provide if-then-else conditional processing, as follows:

- The cfif tag tests a condition and executes its body if the condition is True.
- If the preceding cfif (or cfelseif) test condition is False, the cfelseif tag tests another condition and executes its body if that condition is True.
- The cfelse tag can optionally follow a cfif tag and zero or more cfelseif tags. Its body executes if all the preceding tags' test conditions are False.

The following example shows the use of the cfif, cfelseif, and cfelse tags. If the value of the type variable is "Date," the date displays; if the value is "Time," the time displays; otherwise, both the time and date display.

```
<cfif type IS "Date">
    <cfoutput>#DateFormat(Now())#</cfoutput>
<cfelseif type IS "Time">
    <cfoutput>#TimeFormat(Now())#</cfoutput>
<cfelse>
    <cfoutput>#TimeFormat(Now())#, #DateFormat(Now())#</cfoutput>
</cfif>
```

cfswitch, cfcase, and cfdefaultcase

The cfswitch, cfcase, and cfdefaultcase tags let you select among different code blocks based on the value of an expression. ColdFusion processes these tags as follows:

- The cfswitch tag evaluates an expression. The cfswitch tag body contains one or more cfcase tags and optionally includes cfdefaultcase tag.
- Each cfcase tag in the cfswitch tag body specifies a value or set of values. If a value matches the value determined by the expression in the cfswitch tag, ColdFusion runs the code in the body of the cfcase tag and then exits the cfswitch tag. If two cfcase tags have the same condition, ColdFusion generates an error.
- If none of the cfcase tags match the value determined by the cfswitch tag, and the cfswitch tag body includes a cfdefaultcase tag, ColdFusion runs the code in the cfdefaultcase tag body.

Note: Although the cfdefaultcase tag does not have to follow all cfcase tags, it is good programming practice to put it at the end of the cfswitch statement.

The cfswitch tag provides better performance than a cfif tag with multiple cfelseif tags, and is easier to read. Switch processing is commonly used when different actions are required based on a string variable such as a month or request identifier.

The following example shows switch processing:

```
<cfoutput query = "GetEmployees">
<cfswitch expression = #Department#>
    <cfcase value = "Sales">
        #FirstName# #LastName# is in <b>Sales</b><br><br>
    </cfcase>
    <cfcase value = "Accounting">
        #FirstName# #LastName# is in <b>Accounting</b><br><br>
    </cfcase>
    <cfcase value = "Administration">
        #FirstName# #LastName# is in <b>Administration</b><br><br>
    </cfcase>
    <cfdefaultcase>#FirstName# #LastName# is not in Sales,
        Accounting, or Administration.<br>
    </cfdefaultcase>
</cfswitch>
</cfoutput>
```

cfloop and cfbreak

The cfloop tag loops through the tag body zero or more times based on a condition specified by the tag attributes. The cfbreak tag exits a cfloop tag.

cfloop: The cfloop tag provides the following types of loops:

Loop type	Description
Index	Loops through the body of the tag and increments a counter variable by a specified amount after each loop until the counter reaches a specified value.
Conditional	Checks a condition and runs the body of the tag if the condition is True.
Query	Loops through the body of the tag once for each row in a query.
List, file, or array	Loops through the body of the tag once for each entry in a list, each line in a file, or each item in an array.
Collection	Loops through the body of the tag once for each key in a ColdFusion structure or item in a COM/DCOM object.

The following example shows a simple index loop:

```
<cfloop index = "LoopCount" from = 1 to = 5>

The loop index is <cfoutput>#LoopCount#</cfoutput>.<br>
</cfloop>
```

The following example shows a simple conditional loop. The code does the following:

- Sets up a ten-element array with the word “kumquats” in the fourth entry.
- Loops through the array until it encounters an array element containing “kumquats” or it reaches the end of the array.
- Prints out the value of the Boolean variable that indicates whether it found the word kumquats and the array index at which it exited the loop.

```
<cfset myArray = ArrayNew(1)>
<!--- Use ArraySet to initialize the first ten elements to 123 --->
<cfset ArraySet(myArray, 1, 10, 123)>
<cfset myArray[4] = "kumquats">

<cfset foundit = False>
<cfset i = 0>
<cfloop condition = "(NOT foundit) AND (i LT ArrayLen(myArray))">
    <cfset i = i + 1>
    <cfif myArray[i] IS "kumquats">
        <cfset foundit = True>
    </cfif>
</cfloop>
<cfoutput>
i is #i#<br>
foundit is #foundit#<br>
</cfoutput>
```

Note: You can get an infinite conditional loop if you do not force an end condition. In this example, the loop is infinite if you omit the <cfset i = i + 1> statement. To end an infinite loop, stop the ColdFusion application server.

cfbreak: The cfbreak tag exits the cfloop tag. You typically use it in a cfif tag to exit the loop if a particular condition occurs. The following example shows the use of a cfbreak tag in a query loop:

```
<cfloop query="fruitOrder">
    <cfif fruit IS "kumquat">
        <cfoutput>You cannot order kumquats!<br></cfoutput>
        <cfbreak>
    </cfif>
    <cfoutput>You have ordered #quantity# #fruit#.<br></cfoutput>
</cfloop>
```

cfabort and cfexit

The cfabort tag stops processing of the current page at the location of the cfabort tag. ColdFusion returns to the user or calling tag everything that was processed before the cfabort tag. You can optionally specify an error message to display. You can use the cfabort tag as the body of a cfif tag to stop processing a page when a condition, typically an error, occurs.

The cfexit tag controls the processing of a custom tag, and can only be used in ColdFusion custom tags.

Character Case

ColdFusion is case-insensitive. For example, the following all represent the cfset tag: cfset, CFSET, CFSet, and even cfsEt. However, you should get in the habit of consistently using the same case rules in your programs; for example:

- Develop consistent rules for case use, and stick to them. If you use lowercase characters for some tag names, use them for all tag names.

- Always use the same case for a variable. For example, do not use both myvariable and MyVariable to represent the same variable on a page.

Follow these rules to prevent errors on application pages where you use both CFML and case-sensitive languages, such as JavaScript.

Special Characters

The double-quotation marks (“), single-quotation mark (‘), and number sign (#) characters have special meaning to ColdFusion. To include any of them in a string, double the character; for example, use ## to represent a single # character. The need to escape the single- and double-quotation marks is context-sensitive. Inside a double-quoted string, you do not need to escape single-quotation mark (apostrophe) characters. Inside a single-quoted string, you do not escape double-quotation mark characters. The following example illustrates escaping special characters, including the use of mixed single- and doublequotation marks:

```
<cfset mystring = "We all said ""Happy birthday to you.""">
<cfset mystring2 = 'Then we said "How old are you now?"'>
<cfoutput>
    #mystring#<br>
    #mystring2#<br>
    Here is a number sign: ##
</cfoutput>
```

The output looks like this:

```
We all said "Happy birthday to you."
Then we said "How old are you now?"
Here is a number sign: #
```

Reserved Words

As with any programming tool, you cannot use just any word or name for ColdFusion variables, UDFs and custom tags. You must avoid using any name that can be confused with a ColdFusion element. In some cases, if you use a word that ColdFusion uses—for example, a built-in structure name—you can overwrite the ColdFusion data. The following list indicates words you must not use for ColdFusion variables, user-defined function names, or custom tag names. While some of these words can be used safely in some situations, you can prevent errors by avoiding them entirely.

- Built-in function names, such as Now or Hash
- Scope names, such as Form or Session
- Any name starting with cf. However, when you call a CFML custom tag directly, you prefix the custom tag page name with cf_.
- Operators, such as NE or IS
- The names of any built-in data structures, such as Error or File

- The names of any built-in variables, such as RecordCount or CGI variable names
- CFScript language element names such as for, default, or continue

Data structures are fundamental concepts in computer science and programming that allow you to organize and store data in a way that optimizes access, manipulation, and storage efficiency.

You must also not create form field names ending in any of the following, except to specify a form field validation rule using a hidden form field name.

- integer
- _float
- _range
- _date
- _time
- _eurodate

Because ColdFusion is not case-sensitive, all of the following are reserved words: IS, Is, iS, and is.

CFScript

CFScript is a language within a language. CFScript is a scripting language that is similar to JavaScript but is simpler to use. Also, unlike JavaScript, CFScript only runs on the ColdFusion server; it does not run on the client system. A CFScript script can use all ColdFusion functions and all ColdFusion variables that are available in the script's scope.

CFScript provides a compact and efficient way to write ColdFusion logic. Typical uses of CFScript include:

- Simplifying and speeding variable setting
- Building compact flow control structures
- Encapsulating business logic in user-defined functions

The following sample script populates an array and locates the first array entry that starts with the word "key". It shows several of the elements of CFScript, including setting variables, loop structures, script code blocks, and function calls. Also, the code uses a cfoutput tag to display its results. Although you can use CFScript for output, the cfoutput tag is usually easier to use.

```
<cfscript>
strings = ArrayNew(1);
strings[1]="the";
strings[2]="key to our";
strings[4]="idea";
for( i=1 ; i LE 4 ; i = i+1 )
{
    if(Find("key",strings[i],1))
        break; }
</cfscript>
<cfoutput>Entry #i# starts with "key"</cfoutput><br>
```

7.3 EXTENDING COLDFUSION PAGES WITH CFSCRIPT

KEY NOTES

Basics

ColdFusion MX offers a server-side scripting language, CFScript, that provides ColdFusion functionality in script syntax. This JavaScript-like language gives developers the same control flow as ColdFusion, but without tags.

CFScript Language

The Cfscript tag is a powerful addition to the CFML language allowing developers to express their applications using a Javascript-esq syntax notation.

Using CFScript Statements

CFScript includes the following types of statements:

- Assignment statements and functions
- Conditional processing statements
- Looping statements

Learning Objective

- Discuss how to compare tags and CFScript

Basics

CFScript, short for ColdFusion Script, is a scripting language that offers a streamlined and programmatic way to develop web applications and dynamic websites using the ColdFusion platform. Unlike traditional ColdFusion Markup Language (CFML), which employs tags resembling HTML, CFScript employs a more concise syntax reminiscent of JavaScript. With CFScript, developers can write server-side code that interacts with databases, handles user input, and generates dynamic content.

At its core, CFScript is designed to enhance coding efficiency and readability. It allows developers to express logic, control structures, and data manipulation using a more familiar syntax similar to other scripting languages. This includes constructs like variables, functions, conditional statements, loops, and arrays. By providing an alternative to tag-based coding, CFScript empowers developers with a more code-centric approach, facilitating rapid development and easier maintenance.

Variables are fundamental components of CFScript, enabling storage and manipulation of data. Declarations follow the "var" keyword and can hold various data types, such as strings, numbers, and arrays. Functions, akin to methods or procedures in other programming languages, are used to group code into reusable

units. ColdFusion provides a range of built-in functions for tasks like database queries, string manipulation, and date formatting.

Control structures in CFScript include if-else statements for conditional branching and looping constructs like "for" and "while" loops for repetitive tasks. These structures facilitate decision-making and iteration within the code. Arrays, ordered collections of data, are useful for storing and manipulating multiple values under a single variable.

Interacting with databases is a crucial aspect of web development, and CFScript offers features for database connectivity. Developers can use functions like "QueryNew" to create new query objects, "QueryAddRow" to populate them, and "QueryExecute" to execute SQL commands. This allows dynamic data retrieval and modification for seamless integration with web applications.

The combination of these features enables developers to build dynamic web applications efficiently. ColdFusion scripts can be embedded within CFML templates, allowing for a mix of both tag-based and script-based approaches. However, CFScript truly shines in scenarios where code readability and familiarity with scripting languages play a crucial role.

CFScript is a scripting language within the ColdFusion ecosystem that offers a concise and programmatic way to develop dynamic web applications. By providing a more code-centric approach with familiar constructs, such as variables, functions, control structures, and database interactions, CFScript enhances coding efficiency, readability, and maintenance. It empowers developers to create powerful server-side logic, enabling the creation of feature-rich web applications.

Comparing tags and CFScript

The following examples show how you can use CFML tags and CFScript to do the same thing. Each example takes data submitted from a form and puts it in a structure; if the form does not have a last name and department field, it displays a message.

Using CFML tags

```
<cfif IsDefined("Form.submit")>
    <cfif (Form.lastname NEQ "") AND (Form.department NEQ "")>
        <cfset employee=structnew()>
        <cfset employee.firstname=Form.firstname>
        <cfset employee.lastname=Form.lastname>
        <cfset employee.email=Form.email>
        <cfset employee.phone=Form.phone>
        <cfset employee.department=Form.department>
        <cfoutput>
            Adding #Form.firstname# #Form.lastname#<br>
        </cfoutput>
    <cfelse>
        <cfoutput>
            You must enter a Last Name and Department.<br>
        </cfoutput>
    </cfif>
</cfif>
```

Using CFScript

```
<cfscript>
    if (IsDefined("Form.submit")) {
        if ((Form.lastname NEQ "") AND (Form.department NEQ "")) {
            employee=StructNew();
            employee.firstname=Form.firstname;
            employee.lastname=Form.lastname;
            employee.email=Form.email;
            employee.phone=Form.phone;
            employee.department=Form.department;
            WriteOutput("Adding #Form.firstname# #Form.lastname# <br>");
            }
        else
            WriteOutput("You must enter a Last Name and Department.<br>");
        }
</cfscript>
```

The CFScript Language

The syntax of the CFScript language.

Identifying CFScript

You enclose CFScript regions inside and tags. No other CFML tags are allowed inside a cfscript region. The following lines show a minimal script:

<cfscript>

a = 2;

</cfscript>

Variables

CFScript variables can be of any ColdFusion type, such as numbers, strings, arrays, queries, and objects. The CFScript code can read and write any variables that are available in the page that contains the script. This includes all common scope variables, such as session, application, and server variables.

Expressions and Operators

CFScript supports all CFML expressions. CFML expressions include operators (such as +, -, EQ, and so on), as well as all CFML functions. You can use several comparison operators in CFScript only, not in CFML tags. (You can also use the corresponding CFML operators in CFScript.) The following table lists the CFScript-only operators and the equivalent operator that you can use in CFML tags or CFScript:

CFScript operator	CFML operator	CFScript operator	CFML operator
==	EQ	!=	NEQ
<	LT	<=	LTE
>	GT	>=	GTE

Statements

CFScript supports the following statements:

assignment	for-in	try-catch
function call	while	function (function definition)
if-else	do-while	var (in custom functions only)
switch-case-default	break	return (in custom functions only)
for	continue	

The following rules apply to statements:

- You must put a semicolon at the end of a statement.
- Line breaks are ignored. A single statement can cross multiple lines.
- White space is ignored. For example, it does not matter whether you precede a semicolon with a space character.
- Use curly braces to group multiple statements together into one logical statement unit.
- Unless otherwise indicated, you can use any ColdFusion expression in the body of a statement.

Statement Blocks

Curly brace characters ({ and }) group multiple CFScript statements together so that they are treated as a single unit or statement. This enables you to create code blocks in conditional statements, such as the following:

```
if(score GT 0) {
        result = “positive”;
        Positives = Positives + 1;
}
```

In this example, both assignment statements are executed if the score is greater than 0. If they were not in the code block, only the first line would execute. You do not have to put brace characters on their own lines in the code. For example, you could put the open brace in the preceding example on the same line as the if statement, and some programmers use this style. However, putting at least the ending brace on its own line makes it easier to read the code and separate out code blocks.

Comments

CFScript has two forms of comments: single line and multiline.

A single line comment begins with two forward slashes (//) and ends at the line end; for example:

```
//This is a single-line comment.
```

//This is a second single-line comment.

A multiline comment starts with a /* marker and continues until it reaches a */ marker; for example:

/*This is a multiline comment.
You do not need to start each line with a comment indicator.
This is the last line in the comment. */

The following rules apply to comments:

1. Comments do not have to start at the beginning of a line. They can follow active code on a line. For example, the following line is valid:

   MyVariable = 12; // Set MyVariable to the default value.

2. The end of a multiline comment can be followed on the same line by active code. For example, the following line is valid, although it is poor coding practice:

   End of my long comment */ foo = "bar";

3. You can use multiline format for a comment on a single line, for example:

   /*This is a single line comment using multiline format. */

- You cannot nest /* and */ markers inside other comment lines.
- CFML comments (<!--- and --->) do not work in CFScript.

Reserved Words

In addition to the names of ColdFusion functions and words reserved by ColdFusion expressions (such as NOT, AND, IS, and so on), the following words are reserved in CFScript. Do not use these words as variables or identifiers in your scripting code:

break	default	function	switch
case	do	if	try
catch	else	in	var
continue	for	return	while

Differences from JavaScript

Although CFScript and JavaScript are similar, they have several key differences. The following list identifies CFScript features that differ from JavaScript:

- CFScript uses ColdFusion expressions, which are neither a superset nor a subset of JavaScript expressions. In particular, ColdFusion expressions do not support bitwise operators, and the ColdFusion MOD or % operator operates differently from the corresponding JavaScript % operator: In ColdFusion, the operator

does integer arithmetic and ignores fractional parts. ColdFusion expressions also support the EQV, IMP, CONTAINS, and DOES NOT CONTAIN operators that are not supported in JavaScript.

- Variable declarations (var keyword) are only used in user-defined functions and threads.
- CFScript is case-insensitive.
- All statements end with a semicolon, and line breaks in the code are ignored.
- Assignments are statements, not expressions, and therefore cannot be used in situations that require the assignment operation to be evaluated.
- JavaScript objects, such as Window and Document, are not available.
- Only the ColdFusion server processes CFScript. There is no client-side CFScript.

CFScript Limitation

You cannot include ColdFusion tags in CFScript. However, you can include cfscript blocks inside other ColdFusion tags, such as cfoutput.

CFScript functional equivalents to ColdFusion tags

Tag	CFScript equivalent
`cfset`	Direct assignment, such as Myvar=1;
`cfoutput`	`WriteOutput` function
`cfif, cfelseif, cfelse`	`if` and `else` statements
`cfswitch, cfcase, cfdefaultcase`	`switch, case,` and `default` statements
Indexed `cfloop`	`for` loops
Conditional `cfloop`	`while` loops and `do while` loops
Structure `cfloop`	`for in` loop. (There is no equivalent for queries, lists, or objects.)
`cfbreak`	`break` statement. CFScript also has a `continue` statement that has no equivalent CFML tag.
`cftry, cfcatch`	`try` and `catch` statements
`cfcookie`	Direct assignment of Cookie scope memory-only variables. You cannot use direct assignment to set persistent cookies that are stored on the user's system.
`cfobject`	`CreateObject` function

For example, the following example loops through a query in CFScript:

```
...
<cfscript>
// Loop through the qGetEmails RecordSet
```

```
for (x = 1; x LTE qGetEmails.RecordCount; x=x+1) {
  This_id = qGetEmails.Emails_id[x];
  This_Subject = qGetEmails.Subject[x];
  This_RecFrom = qGetEmails.RecFrom[x];
  This_SentTo = qGetEmails.SentTo[x];
  This_dReceived = qGetEmails.dReceived[x];
   This_Body = qGetEmails.Body[x];
... // More code goes here.
}
</cfscript>
```

Using CFScript Statements

The uses CFScript statements:

- Using assignment statements and functions
- Using conditional processing statements
- Using looping statements

Using Assignment Statements and Functions

CFScript assignment statements are the equivalent of the cfset tag. These statements have the following form:

lval = expression;

lval is any ColdFusion variable reference; for example:

x = “positive”;

y = x;

a[3]=5;

structure.member=10;

ArrayCopy=myArray;

You can use ColdFusion function calls, including UDFs, directly in CFScript. For example, the following line is a valid CFScript statement:

StructInsert(employee,”lastname”,FORM.lastname);

Switch statement is a control structure in programming that allows you to efficiently evaluate an expression and select from a list of possible actions based on the value of that expression.

Using Conditional Processing Statements

CFScript includes the following conditional processing statements:

- if and else statements, which serve the same purpose as the cfif, cfelseif, and cfelse tags
- switch, case, and default statements, which are the

equivalents of the cfswitch, cfcase, and cfdefaultcase tags

Using if and else statements

The if and else statements have the following syntax:

if(expr) statement [else statement]

In its simplest form, an if statement looks like this:

```
if(value EQ 2700)
    message = "You've reached the maximum";
```

A simple if-else statement looks like the following:

```
if(score GT 1)
    result = "positive";
else
    result = "negative";
```

CFScript does not include an elseif statement. However, you can use an if statement immediately after an else statement to create the equivalent of a cfelseif tag, as the following example shows:

```
if(score GT 1)
    result = "positive";
else if(score EQ 0)
    result = "zero";
else
    result = "negative";
```

As with all conditional processing statements, you can use curly braces to enclose multiple statements for each condition, as follows:

```
if(score GT 1) {
    result = "positive";
    message = "The result was positive.";
    }
else {
    result = "negative";
    message = "The result was negative.";
    }
```

Using switch and case statements

The switch statement and its dependent case and default statements have the following syntax:

```
switch (expression) {
        case constant: [case constant:]... statement(s) break;
        [case constant: [case constant:]... statement(s) break;]...
        [default: statement(s)] }
```

Use the following rules and recommendations for switch statements:

- You cannot mix Boolean and numeric constant values in a switch statement.
- Each constant value must be a constant (that is, not a variable, a function, or other expression).
- Multiple case constant: statements can precede the statement or statements to execute if any of the cases are true.

This lets you specify several matches for one code block.

- No two constant values can be the same.
- The statements following the colon in a case statement block do not have to be in braces. If a constant value equals the switch expression, ColdFusion executes all statements through the break statement.
- The break statement at the end of the case statement tells ColdFusion to exit the switch statement. ColdFusion does not generate an error message if you omit a break statement. However, if you omit it, ColdFusion executes all the statements in the following case statement, even if that case is false. In nearly all circumstances, this is not what you want to do.
- You can have only one default statement in a switch statement block. ColdFusion executes the statements in the default block if none of the case statement constants equals the expression value.
- The default statement does not have to follow all switch statements, but it is good programming practice to do so. If any switch statements follow the default statement you must end the default block code with a break statement.
- The default statement is not required. However, you should use one if the case constants do not include all possible values of the expression.
- The default statement does not have to follow all the case statements; however, it is good programming practice to put it there.

ColdFusion is a battle-proven and high-performing application server that makes web development easy for every coder.

The following switch statement takes the value of a name variable:

- If the name is John or Robert, it sets both the male

variable and the found variable to True.

- If the name is Mary, it sets the male variable to False and the found variable to True.
- Otherwise, it sets the found variable to False.

```
switch(name) {
    case "John": case "Robert":
        male=True;
        found=True;
        break;
    case "Mary":
        male=False;
        found=True;
        break;
    default:
        found=False;
} //end switch
```

Using Looping Statements

CFScript provides a richer selection of looping constructs than those supplied by CFML tags. It enables you to create efficient looping constructs similar to those in most programming and scripting languages. CFScript provides the following looping constructs:

- For
- While
- Do-while
- For-in

CFScript also includes the continue and break statements that control loop processing. The following sections describe these types of loops and their uses.

Using for loops

The for loop has the following format:

for (initial-expression; test-expression; final-expression) statement

The initial-expression and final-expression can be one of the following:

- A single assignment expression; for example, x=5 or loop=loop+1
- Any ColdFusion expression; for example, SetVariable(“a”,a+1)
- Empty

The test-expression can be one of the following:

- Any ColdFusion expression; for example:

  A LT 5

  index LE x

  status EQ “not found” AND index LT end

- Empty

When ColdFusion executes a for loop, it does the following:

1. Evaluates the initial expression.
2. Evaluates the test-expression.
3. If the test-expression is False, exits the loop and processing continues following the statement.

If the test-expression is True:

a. Executes the statement (or statement block).

b. Evaluates the final-expression.

c. Returns to Step 2.

For loops are most commonly used for processing in which an index variable is incremented each time through the loop, but it is not limited to this use.

The following simple for loop sets each element in a 10-element array with its index number.

```
for(index=1;
    index LTE 10;
    index = index + 1)
    a[index]=index;
```

The following, more complex, example demonstrates two features:

- 5The use of curly braces to group multiple statements into a single block.
- An empty condition statement. All loop control logic is in the statement block.

This example shows one important issue that you must remember when creating loops: you must always ensure that the loop ends. If this example lacked the else if statement, and there was no “key” in the array, ColdFusion would loop forever or until a system error occurred; you would have to stop the server to end the loop

The example also shows two issues with index arithmetic: in this form of loop you must make sure to initialize the index, and you must keep track of where the index is incremented. In this case, because the index is incremented at the top of the loop, you must initialize it to 0 so it becomes 1 in the first loop.

```
<cfscript>
strings=ArrayNew(1);
ArraySet(strings, 1, 10, "lock");
strings[5]="key";
indx=0;
for( ; ; ) {
    indx=indx+1;
    if(Find("key",strings[indx],1)) {
        WriteOutput("Found key at " & indx & ".<br>");
        break;
        }
    else if (indx IS ArrayLen(strings)) {
        WriteOutput("Exited at " & indx & ".<br>");
        break;
        }
}
</cfscript>
```

Using while loops

The while loop has the following format:

```
while (expression) statement
```

The while statement does the following:

1. Evaluates the expression.
2. If the expression is True, it does the following:
   a. Executes the statement, which can be a single semicolon-terminated statement or a statement block in curly braces.
   b. Returns to step 1.

If the expression is False, processing continues with the next statement.

The following example uses a while loop to populate a 10-element array with multiples of five.

```
a = ArrayNew(1);
loop = 1;
while (loop LE 10) {
    a[loop] = loop * 5;
    loop = loop + 1;
    }
```

As with other loops, you must make sure that at some point the while expression is False and you must be careful to check your index arithmetic.

Using do-while loops

The do-while loop is like a while loop, except that it tests the loop condition after executing the loop statement block. The do-while loop has the following format:

do statement while (expression);

The do while statement does the following:

1 Executes the statement, which can be a single semicolon-terminated statement or a statement block in curly braces.

2 Evaluates the expression.

3 If the expression is true, it returns to step 1.

If the expression is False, processing continues with the next statement.

The following example, like the while loop example, populates a 10-element array with multiples of 5:

```
a = ArrayNew(1);
loop = 1;
do {
a[loop] = loop * 5;
loop = loop + 1;
}
while (loop LE 10);
```

Because the loop index increment follows the array value assignment, the example initializes the loop variable to 1 and tests to make sure that it is less than or equal to 10. The following example generates the same results as the previous two examples, but it increments the index before assigning the array value. As a result, it initializes the index to 0, and the end condition tests that the index is less than 10.

```
a = ArrayNew(1);
loop = 0;
do {
    loop = loop + 1;
    a[loop] = loop * 5;
}
while (loop LT 10);
```

The following example loops through a query:

```
<cfquery ... name="myQuery">
... sql goes here...
</cfquery>
<cfscript>
if (myQuery.RecordCount gt 0) {
  currRow=1;
  do {
    theValue=myQuery.myField[CurrRow];
    currRow=currRow+1;
  } while (currRow LTE myQuery.RecordCount);
}
</cfscript>
```

Using for-in loops

The for-in loop loops over the elements in a ColdFusion structure. It has the following format:

```
for (variable in structure) statement
```

The variable can be any ColdFusion identifier; it holds each structure key name as ColdFusion loops through the structure. The structure must be the name of an existing ColdFusion structure. The statement can be a single semicolon terminated statement or a statement block in curly braces. The following example creates a structure with three elements. It then loops through the structure and displays the name and value of each key. Although the curly braces are not required here, they make it easier to determine the contents of the relatively long WriteOutput function. In general, you can make structured control flow, especially loops, clearer by using curly braces.

Break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

```
myStruct=StructNew();
myStruct.productName="kumquat";
mystruct.quality="fine";
myStruct.quantity=25;
for (keyName in myStruct) {
    WriteOutput("myStruct." & Keyname & " has the value: " &
            myStruct[keyName] &"<br>");
}
```

Using continue and Break Statements

The continue and break statements enable you to control the processing inside loops:

- The continue statement tells ColdFusion to skip to the beginning of the next loop iteration.
- The **break statement** exits the current loop or case statement.

Using continue: The continue statement ends the current loop iteration, skips any code following it in the loop, and jumps to the beginning of the next loop iteration. For example, the following code loops through an array and display's each value that is not an empty string:

```
for ( loop=1; loop LE 10; loop = loop+1) {
    if(a[loop] EQ "") continue;
    WriteOutput(loop);
}
```

(To test this code snippet, you must first create an array, a, with 10 or more elements, some of which are not empty strings.)

In general, the continue statement is particularly useful if you loop over arrays or structures and you want to skip processing for array elements or structure members with specific values, such as the empty string.

Using break: The break statement exits the current loop or case statement. Processing continues at the next CFScript statement. You end case statement processing blocks with a break statement. You can also use a test case with a break statement to prevent infinite loops, as shown in the following example. This script loops through an array and prints out the array indexes that contain the value key. It uses a conditional test and a break statement to make sure that the loop ends when at the end of the array.

```
strings=ArrayNew(1);
ArraySet(strings, 1, 10, "lock");
strings[5]="key";
strings[9]="key";
indx=0;
for( ; ; ) {
    indx=indx+1;
    if(Find("key",strings[indx],1)) {
        WriteOutput("Found a key at " & indx & ".<br>");
        }
    else if (indx IS ArrayLen(strings)) {
        WriteOutput("Array ends at index " & indx & ".<br>");
        break;
    }
}
```



TECHNIQUE 25

CREATING VARIABLES

You create most ColdFusion variables by assigning them values. (You must use the ArrayNew function to create arrays.) Most commonly, you create variables by using the cfset tag. You can also use the cfparam tag, and assignment statements in CFScript. Tags that create data objects also create variables. For example, the cfquery tag creates a query object variable. ColdFusion automatically creates some variables that provide information about the results of certain tags or operations. ColdFusion also automatically generates variables in certain scopes, such as Client and Server. ColdFusion generates an error when it tries to use a variable before it is created. This can happen, for example, when processing data from an incompletely filled form. To prevent such errors, test for the variable's existence before you use it.

- ☐ We already practice this technique
- ☐ This techniques would suitable or not for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 26

WORKING WITH DYNAMIC DATA

A web application page is different from a static web page because it can publish data dynamically. This can involve querying databases, connecting to LDAP or mail servers, and leveraging COM, DCOM, CORBA, or Java objects to retrieve, update, insert, and delete data at run time—as your users interact with pages in their browsers. For ColdFusion developers, the term data source can refer to a number of different types of structured content accessible locally or across a network. You can query websites, LDAP servers, POP mail servers, and documents in a variety of formats. Most commonly though, a database drives your applications, and for this discussion a data source means the entry point from ColdFusion to a database.

- ☐ We already practice this technique
- ☐ This techniques would suitable or not for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 27

MACROMEDIA FLASH INTEGRATION

Macromedia Flash (now Adobe Flash) integration in ColdFusion MX brought together the dynamic capabilities of ColdFusion web applications with the interactive and multimedia-rich features of Flash content. This integration allowed developers to create more engaging and visually appealing web applications.

- ☐ We already practice this technique
- ☐ This techniques would suitable or not for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique

SUMMARY

- ColdFusion is one of the easiest-touse programming languages, yet it can build the most powerful, extensible Web applications in the world.
- Flash Remoting enables you to develop applications with truly interactive graphical user interfaces. The best that you could do before Flash Remoting came along was to create an overly complicated DHTML convolution that either cached data locally or made constant page requests from ColdFusion each time that the user touched a form control.
- CFML is a dynamic application development tool with many of the features of a programming language, including functions, expressions, variables and constants, and flow-control constructs, such as if-then and loops.
- CFX tags are ColdFusion custom tags that you write in a programming language such as Java or C++.
- ColdFusion MX provides powerful tools and features for connecting to databases, executing queries, and manipulating data.
- CFScript, short for ColdFusion Script, is a scripting language that offers a streamlined and programmatic way to develop web applications and dynamic websites using the ColdFusion platform.
- Control structures in CFScript include if-else statements for conditional branching and looping constructs like “for” and “while” loops for repetitive tasks.
- CFScript enhances coding efficiency, readability, and maintenance. It empowers developers to create powerful server-side logic, enabling the creation of feature-rich web applications.
- A Web service exposes a software component to remote systems in a platform-independent manner, meaning that any application server platform capable of consuming web services can make use of the web services created with ColdFusion MX.

KNOWLEDGE CHECK

1. Websites without a database attached to them have which of the following characteristics?

a. The ability to generate data via SQL.

b. The inability to use a browser to display Web pages.

c. Static information using HTML or JavaScript.

d. The need to use TCP/IP as the network protocol.

2. XML is which of the following?

a. A scripting language that states the rules for tagging elements.

b. A programming language that states the rules for tagging elements.
c. A programming language that allows the creation of customized tags.
d. A scripting language that allows the creation of customized tags.

3. Features of XML include which of the following?

a. The tags are used to describe the appearance of the content.
b. Addresses the structuring and manipulation of the data involved.
c. Governs the display of information in a Web browser.
d. All of the above.

4. Which of the following is a valid type of function javascript supports?

a. named function
b. anonymous function
c. Both of the above.
d. None of the above.

5. Which built-in method sorts the elements of an array?

a. changeOrder(order)
b. order()
c. sort()
d. None of the above.

6. What does ColdFusion primarily enable you to develop?

a. Desktop applications
b. Mobile apps
c. Web applications
d. Operating systems

7. In ColdFusion, which of the following is used for server-side scripting?

a. HTML
b. CSS
c. CFML
d. JavaScript

8. Which ColdFusion tag is used to output dynamic content to a web page?

a. <output>
b. <display>
c. <write>
d. <cfoutput>

REVIEW QUESTIONS

1. What is ColdFusion?
2. Discuss how to compare cold fusion with other programming languages.
3. Define the elements of CFML.
4. Describe the ColdFusion components.
5. Explain the ColdFusion data types.
6. Write the ColdFusion pages with CFML scripting.

Check Your Result

1 (c)	2 (d)	3 (b)	4 (c)	5 (c)
6. (c)	7. (c)	8. (d)		

REFERENCES

1. Auerbach, Jon; Kerber, Ross (January 30, 1998). "Massachusetts Rises Despite Passing of High-Tech Giants". Wall Street Journal. Retrieved 2015-11-02.
2. Consuming a Web Service in ASP.NET Tutorial Archived December 23, 2008, at the Wayback Machine.. Digital Colony (2007-08-23). Retrieved on 2013-07-21.
3. G. Rosu. CS422 - Programming Language Design (Fall 2010). University of Illinois at urbana-Champaign (UIUC), USA.
4. Hilwa, Al (January 2015). "Turning Up the Heat on Mobile Application Development with ColdFusion 11" (PDF). IDC White Paper. Retrieved 2015-11-02.
5. Maurizio Gabbrielli and Simone Martini Programming Languages: Principles and Paradigms. Springer-Verlag London Limited 2010.
6. Metz, Cade (October 9, 2014). "Beef up your browser". PC Mag. Retrieved 2015-11-02.
7. The Haskell 2010 Language Report. Retrieved 2011-12-07. Most Haskell implementations extend the Haskell 2010.

CHAPTER 8

WEB GRAPHICS

8.1 BASICS OF WEB GRAPHICS

KEY NOTES

Definition

Web graphics in any websites is as significant as the content of the site. An excellent designed graphics can give better and creative ideas to customer of what they are looking for. Web graphics helps designers to enhance the website design by adding colors, visual appeals and helps in providing artistic professional touch to their creativity. The websites that do not have web graphics do not appeal to visitors. Successes of web graphics depend upon the effective and efficient placement on the websites. The appropriate placement of graphics not only attracts end-users but also makes website well turned-out.

Purpose of graphics in web design:

- Graphics add visual/aesthetic appeal to the information on the web page, helping to maintain viewer interest and attention
- Help to create visual structure for information and links on the page
- Communicate or explain ideas visually

Pixels- A pixel (a word invented from "picture element") is the basic unit of programmable color on a computer display or in a computer image.

Resolution- Resolution is the number of pixels (individual points of color) contained on a display monitor, expressed in terms of the number of pixels on the horizontal axis and the number on the vertical axis.

Learning Objectives

- Understand graphics and file size
- Explain uses of graphics

Definition

Web graphics are visual representations used on a Web site to enhance or enable the representation of an idea or feeling, in order to reach the Web site user. Graphics may entertain, educate, or emotionally impact the user, and are crucial to strength of branding, clarity of illustration, and ease of use for interfaces.

Examples of graphics include maps, photographs, designs and patterns, family trees, diagrams, architectural or engineering blueprints, bar charts and pie charts, typography, schematics, line art, flowcharts, and many other image forms.



Web development is a broad term for the work involved in developing a web site for the Internet (World Wide Web) or an intranet (a private network).

Graphic designers have many tools and technologies at their disposal for everything from print to **Web development**, and W3C provides many of the underlying formats that can be used for the creation of content on the open Web platform.

Pixels

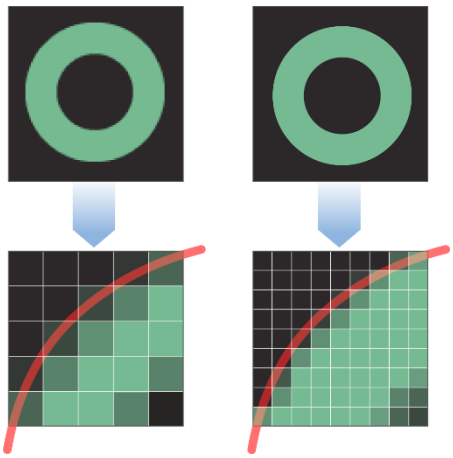
Computer based images are comprised of thousands of tiny dots of color called pixels. These are the small dots you see if you put your face too close to your television or computer screen. Each digital image is comprised of thousands or millions of individual pixels, each with its own color. When these groups of pixels are viewed as a whole, we see the entire image. Most computer monitors have display sizes of 800x600 pixels, 1024x768 pixels, or 1280x1024 pixels.

Resolution

Resolution indicates the number of pixels that are displayed per inch for an image (or pixels per centimeter). Most computer monitors display at resolutions of 72 pixels per inch or 96 pixels per inch. If an image is 72 pixels wide, then it will display as 1 inch big on a monitor that is set to a resolution of 72 pixels/inch.

It is important to remember that for computer graphics, a higher resolution (pixels/inch) does not mean a higher quality image. What is important is the total size, in pixels, of the image.

Graphics for print and graphics for display on computer monitors use different resolutions. While monitors are constrained by the 72 or 96 pixel per inch limitation, standard deskjet printers currently print in the range of 4800x1200 dpi (dots per inch) and pictures in magazines typically print at a resolution of 150 dpi. Unlike computer graphics, with images for print, a higher resolution will yield a higher quality image. Therefore, if you intend to print an image, you want to use a higher resolution than you would for a web graphic to achieve a higher quality print.



So, if you are scanning an image for use on a web page, what resolution should you use? For display on a computer monitor, **scanning** an image at a higher resolution will simply yield an image that displays larger on the screen, since more total pixels are captured. For example, a 3 inch by 3 inch picture scanned at a resolution of 72 pixels per inch will be 216 pixels wide and 216 pixels high and display as a 3 inch by 3 inch picture when displayed on the computer monitor. A 3 inch by 3 inch picture scanned at a resolution of 144 pixels per inch will be 432 pixels wide and 432 pixels high and display as a 6 inch by 6 inch picture when displayed on most computer monitors.

Scanning is reading a text quickly in order to find specific information, e.g. figures or names.

Therefore, the answer to the question of what resolution to scan images at really depends on the size of the original picture and how big you want the image to appear on the web page. Think about the total number of pixels you are capturing instead of the pixels per inch. It is important to remember, though, that

it never hurts to capture an image onto the computer that is larger than you need it (this is true for digital camera settings as well). You can always use a graphics program to reduce the size of an image or crop it without losing quality. If you try to make it larger using a graphics program, however, the image will lose quality. A smaller image means that the image contains fewer pixels, and a larger image has more pixels. You can always throw away pixels that you have, but you cannot add pixels that you don't have!

Graphics and File Size

Web designers are always caught in a constant compromise between image quality and file size. Generally speaking, the higher the quality of the image, the greater the file size will be. Also, the larger the image, the larger the file size will be.

It is important to consider the file size of images when using them on a web page. Images with large file sizes take longer to load. Long load times can be an annoyance to your audience and may render pages essentially unusable for those with slow Internet connections.

Uses of Graphics

Graphics are used for everything from enhancing the appearance of Web pages to serving as the presentation and user interaction layer for full-fledged Web Applications.

Different use cases for graphics demand different solutions, thus there are several different technologies available. Photographs are best represented with PNG, while interactive line art, data visualization, and even user interfaces need the power of SVG and the Canvas API. CSS exists to enhance other formats like HTML or SVG. WebCGM meets the needs for technical illustration and **documentation** in many industries.

Documentation is a set of documents provided on paper, or online, or on digital or analog media, such as audio tape or CDs.

Guidelines for effective use of web graphics are:

- Graphics should fit in with the purpose, organization, and style of the page. They should enhance the design, structure, or informative content of the web page without distracting attention. As much as you may want to add an image to a page because it shows off some fancy new effect you learned in the graphics program, think first about whether it fits in or not. You may want

to keep these types of graphics in your personal portfolio under the heading, "Cool Effects I Know in My Graphics Program."

- Avoid using graphics with large file sizes that add to the load time of the page. Also, consider the cumulative file size of all images on the page. Excessive "page weight" caused by poor image use can result in slow load times for pages.



- Graphics should help to guide the viewers' focus to the important content on the page. Using visually strong graphic elements on a page can be useful in directing viewers' attention and providing structure for the page. Be careful, though. Strong graphic elements can also pull attention away from central content or compete with one another on the page. This results in the page appearing overly busy or cluttered and makes it difficult to read. Remember, if you attempt to emphasize everything then nothing ends up standing out.
- Avoid repetitive use of overly bright or potentially "obnoxious" images. These types of "eye catching" images may be attractive at first, but after the novelty wears off, they may cause viewers to lose patience with the site. While an animation of a dancing monkey may seem interesting and funny at first, over time it may become annoying and drive viewers away.
- Avoid the use of graphics to convey textual content information. While it may be tempting from a design standpoint to use images for textual information because of the greater number of design options such as font choices and text effects, graphic based text presents a variety of problems: Images of text cannot be resized like true text, so users with poor vision are unable to resize it to meet their visual needs; images require much longer to download than text; users can't search for images of text using their browser's *find* feature; and search engines are better able to index websites that contain actual text, well-structured with HTML.
- Provide textual equivalent alternatives for graphic content. Remember that not all users are able to see the images on your web page. Whenever images are used, it is important to provide equivalent content or descriptions of the image in a textual format. The most common way this is done is to provide

descriptive text using the ALT attribute. This is especially important where images are used as navigation buttons or links.

- When using text in graphics, make sure there is sufficient contrast between the text and the background color. Design graphic elements so that users can easily distinguish the text from its background. Also, be sure to avoid color combinations (reds and greens for example) that color blind users will not be able to distinguish apart from one another.

8.2 GRAPHIC FILE FORMATS

KEY NOTES

Definition

Graphic file formats are diverse and cater to a range of needs, from static images to complex animations, and from print quality to online sharing. Choosing the right format depends on factors like image type, usage scenario, desired quality, and compatibility with software and platforms. Understanding the characteristics of different formats empowers creators to make informed decisions when selecting the most suitable format for their specific projects.

GIF Graphics

The CompuServe Information Service popularized the Graphic Interchange Format (gif) in the 1980s as an efficient means to transmit images across data networks. Gif files incorporate a "lossless" compression scheme to keep file sizes at a minimum without compromising quality. However, gif files are 8-bit graphics and thus can only accommodate 256 colors.

JPEG Graphics

The other graphic file format commonly used on the web to minimize graphics file sizes is the Joint Photographic Experts Group (jpeg) compression scheme. Unlike gif graphics, jpeg images are full-color images that dedicate at least 24 bits of memory to each pixel, resulting in images that can incorporate 16.8 million colors.

PNG Graphics

Portable Network Graphic (png) is an image format developed by a consortium of graphic software developers as a nonproprietary alternative to the gif image format.

Learning Objectives

- Discuss about native file formats
- Identify other file formats

Perception

Graphic file formats are standardized ways of encoding and storing visual information, such as images, illustrations, and graphics, in digital form. These formats play a vital role in various applications, from web design and digital art to scientific visualization and multimedia content creation. Each graphic file format has distinct characteristics that determine its suitability for specific purposes, including image quality, compression, transparency, and support for animation.

Raster image formats, also known as bitmap formats, store images as a grid of individual pixels, where each pixel contains color and brightness information. Common raster formats include JPEG, PNG, GIF, and BMP. JPEG (Joint Photographic Experts Group) is widely used for photographs due to its efficient compression that maintains good image quality. PNG (Portable Network Graphics) offers lossless compression and

supports transparency, making it suitable for images with sharp edges and transparent backgrounds. GIF (Graphics Interchange Format) is known for its support of simple animations and low-color images, making it suitable for icons and simple graphics. BMP (Bitmap) is a basic format that stores pixel data without compression and is used primarily on Windows systems.

Vector image formats, on the other hand, represent images using mathematical equations that define lines, curves, and shapes. This allows vector graphics to be resized without losing quality. Formats like SVG (Scalable Vector Graphics) are popular for web design and interactive graphics because they can adapt to various screen sizes without pixelation. Adobe Illustrator's AI format is another example of a vector format that preserves complex design information and is suitable for professional graphic design projects.

In addition to these, there are specialized formats for specific purposes. TIFF (Tagged Image File Format) is favored in professional photography and publishing due to its ability to retain high-quality images without compression loss. RAW formats, like Canon's CR2 and Nikon's NEF, store unprocessed sensor data from digital cameras, offering photographers more control during post-processing. DICOM (Digital Imaging and Communications in Medicine) is crucial in medical imaging, facilitating the exchange and storage of medical images such as X-rays and MRIs.

Furthermore, multimedia formats encompass both audio and video components. Formats like MP4, AVI, and MOV integrate video and audio data, while formats like WAV and MP3 store audio only. Multimedia formats are essential for video streaming, video editing, and audio playback applications.

JPEG (Joint Photographic Experts Group)

The jpeg image format allows up to 16 million colors and is the best choice for image with many colors or color gradations (especially photographs).



Figure 8.1: Save all photographs for the web as JPEGs. Make sure to set the quality high enough to avoid JPEG noise which is comprised of small rectangles called artifacts.

Jpeg is a "lossy" format which means that each time the image is saved and compressed, image information is lost and quality is degraded. Jpeg images allow

for various levels of compression. Low compression means high image quality, but large file size. High compression means lower image quality, but smaller file size.

GIF (Graphic Information Format)

Gif is a "lossless" image format, meaning that the quality of the image is not degraded through **compression**. The limitation of gif images is that the color palette includes only 256 colors. Therefore, the gif format is a good choice for simpler graphics with a smaller number of solid colors. The gif format also allows for transparent backgrounds, which allows the background color of the web page to show behind the graphic.

Compression is a reduction in the number of bits needed to represent data.

GIF File Compression

The gif file format uses a relatively basic form of file compression (Lempel Ziv Welch, or lzw) that squeezes out inefficiencies in data storage without losing data or distorting the image. The lzw compression scheme is best at compressing images with large fields of homogeneous color, such as logos and diagrams. It is much less efficient at compressing complicated "photographic" pictures with many colors and complex textures.

Dithering

Full-color photographs can contain an almost infinite range of color values; gif images can contain no more than 256 colors. The process of reducing many colors to 256 or fewer is called dithering. With dithering, pixels of two colors are juxtaposed to create the illusion that a third color is present. Dithering a photographic image down to 256 colors produces an unpleasantly grainy image. In the past this technique was necessary to create images that would look acceptable on 256-color computer screens, but with today's full-color displays there is seldom any need to dither an image. If you need a wider range of colors than the gif format can handle, try using your image editor to save the image in both jpeg and png formats (described below), compare the resulting file sizes and image qualities, and pick the best balance of file size and image quality.



Figure 8.2: Dithering a full-color photograph (left) to a 256-color image (right) results in an image with lots of visual noise and harsh transitions between pixels of different colors. Luckily, such color dithering compromises are now mostly a thing of the past, since most users now have "true color" monitors that can display thousands or millions of colors.

Improving GIF Compression

You can take advantage of the characteristics of lzw compression to improve its efficiency and thereby reduce the size of your gif graphics. The strategy is to reduce the number of colors in your gif image to the minimum number necessary and to remove colors that are not required to represent the image. A gif graphic cannot have more than 256 colors, but it can have fewer. Images with fewer colors will compress more efficiently under lzw compression. For example, when creating gif graphics in Photoshop, do not save every file automatically with 256 colors. A simple gif image may look fine at 8, 16, or 32 colors, and the file size savings can be substantial. For maximum efficiency in gif graphics, use the minimum number of colors that gives you a good visual result.

Interlaced GIF

Interlacing is a method of encoding a bitmap image such that a person who has partially received it sees a degraded copy of the entire image.

The conventional (noninterlaced) gif graphic downloads one line of pixels at a time from top to bottom, and browsers display each line of the image as it gradually builds on the screen. In interlaced gif files the image data is stored in a format that allows browsers that support this feature to build a low-resolution version of the full-sized gif picture on the screen while the file is downloading. Many people find the "fuzzy-to-sharp" animated effect of interlacing visually appealing, but the most important benefit of interlacing is that it gives the user a preview of the full picture while the picture downloads into the browser.

Interlacing is best for larger gif images such as illustrations of 200 × 100 pixels or greater. Interlacing is a poor choice for

small gif graphics such as navigation bars, buttons, and icons. These small graphics will load onto the screen much faster if you keep them in conventional (noninterlaced) gifformat. In general, **interlacing** has no significant effect on the file size of average gif graphics.

Transparent GIF

The gif format allows you to pick colors from the color lookup table of the gif to be transparent. You can use image-editing software such as Photoshop (and many shareware utility programs) to select colors in a gif graphic's color palette to become transparent. Usually the color selected for transparency is the background color in the graphic. Unfortunately, the transparent property is not selective; if you make a color transparent, every pixel in the graphic that shares that color will also become transparent, which can cause unexpected results.

Adding transparency to a gif graphic can produce disappointing results when the image contains anti-aliasing. If you use an image-editing program like Photoshop to create a shape set against a background color, Photoshop will smooth the shape by inserting pixels of intermediate colors along the shape's boundary edges. This smoothing, or anti-aliasing, improves the look of screen images by softening what would otherwise look like jagged edges. The trouble comes when you set the background color to transparent and then use the image on a Web page against a different background color. The anti-aliased pixels in the image will still correspond to the original background color. In the example below, when we change the background color from white to transparent (letting the gray web page background show through), an ugly white halo appears around the graphic.

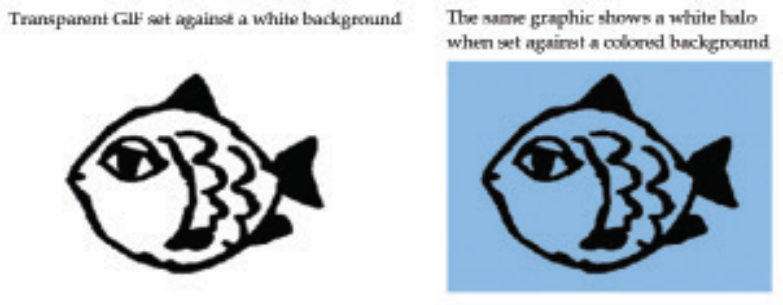


Figure 8.3: The dreaded "white halo" in transparent GIF graphics.

The same problem exists with printing. Most browsers do not print background colors, and a transparent gif anti-aliased against a colored background will not blend smoothly into the white of the printed page.

PNG (Progressive Network Graphics)

PNG is a newer image format that provides a combination of many features of both jpeg and gif. The PNG format allows for use of millions of colors as well as providing

the ability to have transparent backgrounds. The one drawback to PNG's is that the format is not supported by all web browsers, particularly older browsers.



Figure 8.4: PNGs can have fully and partially transparent areas. A gray and white checkerboard is the traditional symbol for transparency. The logo on the right is just three white letters and two semi-transparent white letters on top of a blue background.

PNG graphics were designed specifically for use on web pages, and they offer a range of attractive features, including a full range of color depths, support for sophisticated image transparency, better interlacing, and automatic corrections for display monitor gamma. PNG images can also hold a short text description of the image's content, which allows Internet search engines to search for images based on these embedded text descriptions.

PNG supports full-color images and can be used for photographic images. However, because it uses lossless compression, the resulting file is much larger than with lossy jpeg compression. Like gif, PNG does best with line art, text, and logos—images that contain large areas of homogenous color with sharp transitions between colors. Images of this type saved in the PNG format look good and have a similar or even smaller file size than when saved as gifs. However, widespread adoption of the PNG format has been slow. This is due in part to inconsistent support in web browsers. In particular, Internet Explorer does not fully support all the features of PNG graphics. As a result, most images that would be suitable for PNG compression use the gif format instead, which has the benefit of full and consistent browser support.

Native File Formats

Native file formats are the formats that the graphics editing programs use to save the graphic project files in. The native file formats are larger in file size and contain all of the information that allows the image to be edited. Examples of native file formats are .psd for Photoshop and .png for Fireworks.



Other File Formats

Computer graphics may also be present in a number of other different formats such as .bmp, .tiff, and .eps. These formats have their particular uses, such as for print graphics, but are not suitable for display on the web. To use one of these types of files on the web, open it in your graphics program and then save it as either a jpeg, gif, or png.

8.3 GRAPHICS AS CONTENT

KEY NOTES

Definition

Computer Graphic is the discipline of producing picture or images using a computer which include modeling, creation, manipulation, storage of geometric objects, rendering, converting a scene to an image, the process of transformations, rasterization, shading, illumination, animation of the image, etc.

Computer Graphics has been widely used in graphics presentation, paint systems, computer-aided design (CAD), image processing, simulation, etc. From the earliest text character images of a non-graphic mainframe computers to the latest photographic quality images of a high resolution personal computers, from vector displays to raster displays, from 2D input, to 3D input and beyond, computer graphics has gone through its short, rapid changing history. From games to virtual reality, to 3D active desktops, from unobtrusive immersive home environments, to scientific and business, computer graphics technology has touched almost every concern of our life.

Graphics serve a number of purposes as elements of content—along with and complementary to text content:

- Illustrations
- Diagrams
- Quantitative data
- Analysis and causality
- Integration

Learning Objectives

- Discuss about graphic communication on the web
- write the basic need of Visual Communication Principles in graphics.

Definition

Good diagrams and interesting illustrations are visually arresting and create documents that are distinct and memorable. Illustrations can convey complex quantitative or spatial information quickly, transcending language barriers, and combine with text to complement many styles of information gathering and learning. Personal computing and the rise of the Internet as a publishing medium have liberated information publishing from physical constraints: new tools have increased our ability to create information graphics, and the web provides a full-color multimedia medium unbounded by the costs of publishing and distributing paper.

Purposes of graphics as elements of content are:

- Illustrations: Graphics can show you things, bringing pieces of the world into your document
- Diagrams: Quantitative graphics and process diagrams can explain concepts visually

- Quantitative data: Numeric charts can help explain financial, scientific, or other data
- Analysis and causality: Graphics can help take apart a topic or show what caused it
- Integration: Graphics can combine words, numbers, and images in a comprehensive explanation.



Illustrations

Photographs and illustrations bring the world into your document in a concise and unforgettable way that saves that proverbial "thousand words" of explanation. Why describe something when you can simply—and more memorably—show it? The web is a color medium with an enormous range of display colors, and color graphics on web pages are almost the equal of fine printing on paper, especially when graphics are reproduced at roughly the same size as they might appear in print.

Diagrams and Quantitative Data

The resolution limitations of the computer screen (72–90 pixels per inch) versus print (300+ dots per inch) begin to show when you create complex charts and diagrams for the computer page. Most chart graphics work as well on the screen as on paper, but the label **typography** in charts and diagrams must be optimized for display on web screens. The visual detail of complex diagrams converted from print uses is often surprisingly good, but always check the typography within the figure, as labels may become illegible in the conversion process.

Typography is the art and technique of arranging type to make written language legible, readable, and appealing when displayed.



Integrated Visual Presentations

Multifaceted information graphics often integrate quantitative charts, three-dimensional illustrations, and extensive captioning in free-form layouts that become visual narratives capable of explaining complex concepts and natural phenomena.

Graphic Communication on the Web

A chart or diagram is an implicit promise to the user that you will make a complex world easier to understand. The advice on graphic communications is the same as on written communications:

- Trust the reader's intelligence. Don't dumb down your material on the supposition that web users are somehow fundamentally different kinds of people from print readers and have no interest in complexity. Regular readers of web sites may once have been distinguishable from other publishing audiences, but now everyone reads the web.
- Respect the medium. The readers are the same as in print media, but the web has a different profile of strengths and weaknesses. Take advantage of the web's enormous capabilities to communicate complex color visuals without the expense of printing and physical distribution.
- Tell the truth as you understand it. Distorting quantitative data is not just a failure to communicate; it is a betrayal of the reader's trust.
- Do not cherry-pick your data. If you are making a case with visual evidence, do not process and edit your visuals so heavily that the audience has no choice but to accept your point of view. Trust your audience enough to give them the data: let them look at the same higher-resolution images or ambiguous results that you saw and decide the issue for themselves.
- Be bold and substantial. A serious interest in visual communication does not require that you use only small, mousy graphics in pale colors. Visual evidence

cannot become persuasive if no one ever notices it. Just do not ever try to wow an audience with bright graphics to make up for thin content.

Visual Communication Principles in Graphics

Visual communication principles are essential in graphics to ensure effective and impactful visual content that effectively conveys messages, engages audiences, and communicates information accurately. These principles serve as guidelines for creating visually appealing and meaningful graphics that resonate with viewers. Here's why visual communication principles are a basic need in graphics:

- **Clarity and Understanding**: Visual communication principles help ensure that the graphics are clear, organized, and easy to understand. By applying concepts like hierarchy, contrast, and simplicity, designers can structure their graphics in a way that guides viewers' attention and makes the intended message clear.
- **Engagement and Attention**: Graphics that adhere to visual communication principles are more likely to capture viewers' attention and keep them engaged. Elements like contrast, color, and visual hierarchy can draw the eye to important information and create a visually pleasing experience.
- **Simplification of Complex Ideas**: Complex concepts can be difficult to explain using text alone. Visual communication principles allow designers to distill complex ideas into simpler visual representations, making the content more accessible and digestible.
- **Memorability**: Well-designed graphics are more memorable than plain text. Applying principles like repetition, alignment, and color can help create a consistent visual style that sticks in the viewers' minds.
- **Conveying Emotions and Tone**: Graphics have the power to evoke emotions and convey tone effectively. Visual communication principles enable designers to choose colors, fonts, and imagery that align with the intended emotional response or overall tone of the content.
- **Universal Understanding**: Visual elements often transcend language barriers, making graphics a powerful tool for reaching diverse audiences. Visual communication principles help ensure that the graphics communicate universally, allowing people from different cultures and backgrounds to understand the content.
- **User Experience (UX) and Accessibility**: Designing graphics with visual communication principles in mind improves the overall user experience. Proper use of color, contrast, and typography enhances readability, and considerations for accessibility (such as color contrast for users with visual impairments) ensure inclusivity.
- **Branding and Consistency**: Visual communication principles play a key role in maintaining branding consistency across graphics. By adhering to a set of

design guidelines, graphics can reinforce brand identity and create a cohesive visual presence.

- **Effective Data Visualization**: When creating graphs, charts, and infographics, visual communication principles are crucial for presenting data accurately and clearly. Proper use of scales, labels, and visual cues ensures that the information is conveyed accurately to the audience.
- **Creative Expression and Artistry**: While principles provide guidelines, they also offer room for creativity and artistry. Designers can use principles as building blocks to craft unique and aesthetically pleasing graphics that still communicate effectively.

Visual communication principles are the foundation upon which successful graphics are built. They enable designers to create content that not only looks good but also effectively communicates messages, ideas, and information to a wide range of audiences. Whether creating marketing materials, educational resources, or multimedia presentations, understanding and applying these principles is fundamental to achieving the desired impact with visual content.



TECHNIQUE 28

ARCHIVING YOUR WEB SITE GRAPHICS

Always save a copy of your original graphics files, and make it a standard practice to create separate new files every time you make significant changes to an image, such as resizing it or changing the file format. After the close of a project all photos and artwork should be kept and stored at their full original resolution and in a format that does not compromise the image quality of the files through "lossy" image compression, as in JPEG. We prefer to archive every image generated in a project. Many small 8-bit GIF or JPEG illustrations on the finished Web page, for example, start out as much larger high-resolution files in Photoshop format. We save all the intermediate pieces, not just the original and final files. This will save you a lot of time if you later change your mind about the best file format for a graphic or need to modify it. If you have archived the full-color Photoshop version of the graphic, you can easily create a new version in a different format. If you save only the final GIFs, you will have lost your full-color version. If you save only the final JPEGs, you will no longer have images without compression artifacts, and recompressing an image that already has JPEG compression noise usually yields poor results.

- □ We already practice this technique
- □ This technique would not be suitable for our operation
- □ We practiced this in the past and need to jumpstart
- □ We should implement this technique



TECHNIQUE 29

BROWSER-SAFE COLORS

The inconsistencies between how different browsers and operating systems display colors is a nightmarish mess. To make sure images look the same no matter what a viewer uses to look at them, the limited yet relatively Web safe palette of 216 colors was born. Select the "Web 216 palette" from the list of swatches and choose your colors from amongst those provided. For the ultimate in Web color safety, try the really safe palette.

- ☐ We already practice this technique
- ☐ This technique would not be suitable for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 30

GRAPHIC TEXT

Graphic typography in gif and jpeg graphics is invisible to screen readers. Search engines don't load graphics as they scan and index web pages, but even if they did, a web graphic is merely a collection of differently colored pixels arranged over an area of the screen, and text that is contained in a graphic is simply differently colored pixels. As a result, software cannot make intelligent use of those graphics by, for example, reading the text aloud. Additionally, browsers cannot enlarge graphic text as elegantly as with regular text. The only option is to enlarge the image, which results in pixilated, fuzzy text that is difficult to read. In addition, graphic text cannot be recolored or styled by the user's browser to make reading easier.

- ☐ We already practice this technique
- ☐ This technique would not be suitable for our operation
- ☐ We practiced this in the past and need to jumpstart
- ☐ We should implement this technique



TECHNIQUE 31

STORYTELLING WITH GRAPHICS

Storytelling with graphics is a dynamic and engaging method of conveying narratives, ideas, and information through visual elements. It marries the art of storytelling with the impact of visuals, resulting in a compelling and memorable experience for the audience. This approach involves the strategic use of images, illustrations, diagrams, and other graphical elements to guide viewers through a narrative arc, captivate their attention, and facilitate the understanding of complex concepts.

- □ We already practice this technique
- □ This technique would not be suitable for our operation
- □ We practiced this in the past and need to jumpstart
- □ We should implement this technique

SUMMARY

- Web graphics are visual representations used on a Web site to enhance or enable the representation of an idea or feeling, in order to reach the Web site user.
- Resolution indicates the number of pixels that are displayed per inch for an image (or pixels per centimeter).
- Web designers are always caught in a constant compromise between image quality and file size. Generally speaking, the higher the quality of the image, the greater the file size will be.
- Graphics are used for everything from enhancing the appearance of Web pages to serving as the presentation and user interaction layer for full-fledged Web Applications.
- Graphic file formats are standardized ways of encoding and storing visual information, such as images, illustrations, and graphics, in digital form. These formats play a vital role in various applications, from web design and digital art to scientific visualization and multimedia content creation.
- Common raster formats include JPEG, PNG, GIF, and BMP. JPEG (Joint Photographic Experts Group) is widely used for photographs due to its efficient compression that maintains good image quality.
- Vector image formats, on the other hand, represent images using mathematical equations that define lines, curves, and shapes. This allows vector graphics to be resized without losing quality.
- Gif is a "lossless" image format, meaning that the quality of the image is not degraded through compression. The limitation of gif images is that the color palette includes only 256 colors.
- The gif format allows you to pick colors from the color lookup table of the gif to be transparent. You can use image-editing software such as Photoshop (and many shareware utility programs) to select colors in a gif graphic's color palette to become transparent.
- Visual communication principles are essential in graphics to ensure effective and impactful visual content that effectively conveys messages, engages audiences, and communicates information accurately.
- Visual communication principles play a key role in maintaining branding consistency across graphics. By adhering to a set of design guidelines, graphics can reinforce brand identity and create a cohesive visual presence.
- Native file formats are the formats that the graphics editing programs use to save the graphic project files in. The native file formats are larger in file size and contain all of the information that allows the image to be edited.

KNOWLEDGE CHECK

1. Which devices provides positional information to the graphics system?

a. Input devices
b. Output devices
c. Pointing devices
d. Both a and c

2. The number of pixels stored in the frame buffer of a graphics system is known as

a. Resolution
b. Depth
c. Resalution
d. Only a

3. In graphical system, the array of pixels in the picture are stored in

a. Memory
b. Frame buffer
c. Processor
d. All the above

4. Pixel mask means

a. A string containing only 1;s
b. A string containing only 0's
c. A string containing 1 and 0
d. A string containing 0 and 0

5. The maximum number of points that can be displayed without overlap on a CRT is referred as

a. Picture
b. Resolution
c. Persistence
d. Neither b nor c

6. …… stores the picture information as a charge distribution behind the phosphor-coated screen.

a. Cathode ray tube
b. Direct-view storage tube
c. Flat panel displays
d. 3D viewing device.

7. The process of digitizing a given picture definition into a set of pixel-intensity for storage in the frame buffer is called

a. Rasterization
b. Encoding
c. Scan conversion
d. True color system

8. A line drawn in the background color is

a. Visible
b. Invisible
c. Visible or Invisible
d. Only b

9. Which type of graphic format is best suited for images with sharp edges and transparent backgrounds?

a. JPEG
b. GIF
c. PNG
d. BMP

10. What is the primary advantage of using vector graphics on the web?

a. They have a smaller file size.
b. They support animation.
c. They maintain image quality at any size.
d. They have a greater color depth.

REVIEW QUESTIONS

1. Mention what are the main language or platform used for web-design?
2. Explain how can you set an image as a background on web pages?
3. What are different types of file format? Explain.
4. What makes a good color scheme? Can you name a few examples of complementary, analogous and monochromatic color schemes?
5. When do you use JPEG compression and when would you prefer PNG instead?

Check your Result

1. (d)	2. (d)	3. (a)	4. (c)	5. (b)
6. (b)	7. (c)	8. (d)	9. (b)	10. (c)

REFERENCES

1. Coombs, Norman. Making Online Teaching Accessible, Jossey-Bass, 2010. “Making Online Teaching Accessible” is downloadable from Bookshare.
2. Cunningham, Katie. The Accessibility Handbook, O’Reilly Media, 2012.
3. Duckett, Jon. Accessible XHTML and CSS Web Sites Problem Design Solution, Wrox, 2005.
4. Felke-Morris, T. (2015). Web Development and Design Foundations with HTML & CSS. Pearson.
5. Hassell, Jonathan. Including your missing 20% by embedding web and mobile accessibility, BSI British Standards Institution, 2014.
6. Henry, Shawn Lawton. Just Ask: Integrating Accessibility Throughout Design, ET\ Lawton, 2007.
7. Horton, Sarah and Quesenbery, Whitney. A Web for Everyone, Rosenfeld Media, 2012.
8. Horton, Sarah and Quesenbery, Whitney. Universal Design for Web Accessibility, Rosenfeld Media, 2014.
9. Horton, Sarah. Access by Design: A Guide to Universal Usability for Web Designers, New Riders Publishing, 2005.
10. Hricko, Mary (Editor). Design and Implementation of Web-Enabled Teaching Tools, Idea Group Publishing, 2002.
11. Kline, Jeff. Strategic IT Accessibility: Enabling the Organization, Live Oak Book Company, 2011.
12. Lazar, Jonathan; Goldstein, Daniel; and Taylor, Anne . Ensuring Digital Accessibility through Process and Policy, Morgan Kaufmann, 2015.
13. Meyer, E. A., & Anderson, R. (2017). Web Graphics for Non-Designers. O’Reilly Media.
14. Shaffer, D. (2017). Web Development & Design Foundations with HTML5. Pearson.
15. Vise, Kristen. “An Interdisciplinary Approach to Graphic Design”. College of Liberal Arts. Archived from the original on 14 April 2021. Retrieved 13 October 2021.

CHAPTER 9

WEB DESIGN AND DEVELOPMENT

9.1 DEVELOPMENT IN WEB DESIGN

KEY NOTES

Concept

Web design has undergone transformative developments that have redefined the way websites are created, presented, and experienced. These advancements have been propelled by technological innovation, changing user preferences, and the evolution of design principles. One of the most significant developments is the shift towards responsive web design, a paradigm that acknowledges the diversity of devices used to access the internet.

Role of Web Designers

Web designers play a pivotal role in shaping the online presence of businesses, organizations, and individuals. Their responsibilities extend beyond aesthetics, encompassing user experience, functionality, and the overall success of a website.

Use of Web Design

Web design serves as a versatile tool that empowers individuals, businesses, organizations, and institutions to establish a digital presence, communicate effectively, engage users, and achieve their goals in the dynamic online landscape.

LEARNING OBJECTIVES

- Capable to choosing a web design tool
- Find the role of web designers
- Examine the difference between web designer and web developer
- related elements of web design

Concept

Web design is the process of planning, conceptualizing, and arranging content online. Today, designing a website goes beyond aesthetics to include the website's overall functionality. Web design also includes web apps, mobile apps, and user interface design. Web designing is the creation of websites and pages to reflect a company's brand and information and ensure a user-friendly experience. Appearance and design are incorporated as vital elements whether you're designing a website, mobile app or maintaining content on a web page. You can learn web designing to create a site for your own business or become a professional web designer creating sites for clients. Web design is what creates the overall look and feel when you're using a website. It's the process of planning and building the elements of your website, from structure and layout to images, colors, fonts and graphics. Web design has numerous components that work together to create the finished experience of a website, including graphic design, user experience design, interface design, search engine optimization (SEO) and content creation. These elements determine how a website looks, feels and works on various devices. Check out this in-depth guide on building a website, if you want to learn more about the whole process.



Web design is different from web development, which is the actual coding that makes a website work. When you're building a website, you need both web design and web development. Although you can find web designers who are also web and UX developers, these are distinct skill sets. Web designers take your ideas and turn them into a mockup that shows what your future website will look like. Web designers handle the creative part of designing a website. Web developers—also sometimes called engineers or coders—take the mockup your web designer made and translate it into a coding language so it can be displayed on the web. They make websites functional, which often means custom-coding widgets and other tools. A **user experience developer**, also known as a UX developer, is the one who makes your website user friendly. They

have technical skills as well as design skills and put them to work creating websites that attract and keep visitors.

User Experience Developer refer to a professional who specializes in creating user-centered and interactive web experiences by combining skills from both front-end development and user experience (UX) design.

Importance of Web Design

First impressions really matter. We can't stress this point enough: if you don't have a strong web presence, you're holding your brand back. Prospective customers who search the web for your brand and find nothing might think you've gone out of business. If they search and find something subpar, they'll get the impression that you don't care much about your company or product. Make every relationship that begins on your website a great one by getting your web design right. Now that you know what's what and who's who, let's look at some tell-tale signs of great web design and what distinguishes it from not so great web design.

Choosing a Web Design Tool

There are two main ways to design a website: using a desktop app or using a website builder. The tool you decide to use will vary greatly based on your team size, your budget, and the type of site you wish to build and its technical requirements.

1. **Desktop Apps.** Desktop apps require designers to create their design and send it to a development team who can then convert the design to code. The most popular desktop apps for designing websites are Photoshop and Sketch. Typically, this is the standard for large and/or complex websites because it allows the designer to focus on the overall look and feel, while all the technical challenges are transferred to the development team. Unfortunately, this process can be expensive and time-consuming because multiple resources, skill sets, and team members are required. To avoid involving a developer, it is beneficial to use a website builder to design a website with fewer technical requirements.
2. **Website Builders.** There are many website builders on the market today that offer a wide range of features and services. Wix, Squarespace, Webflow, and PageCloud, are just a few examples of popular website builders that vary in design capabilities, template options, price, and overall editing experience. Be sure to do your research, experiment with free trials, and determine which

platform best fits your website needs. Website builders create either adaptive or responsive websites, which offer different building experiences. These concepts will be discussed in more detail below so you can best understand which builders will work for you. If you don't know how to code, becoming familiar with the freedoms and limitations of various website design tools is essential. For example, although Wordpress is the most used website platform, it's not popular with visual designers because of its limited customization options. Before you start building a website, determine your website needs: Are you creating a photo gallery? How often will you update your site? Do you need a contact form? Choose a website builder that can help you effectively accomplish those goals.

Role of Web Designers

Web design identifies the goals of a website or webpage and promotes accessibility for all potential users. This process involves organizing content and images across a series of pages, integrating applications and other interactive elements.

The professionals who perform this process are called web designers, and their job includes the following duties:

- Selecting easy-to-read fonts
- Choosing attractive color schemes that also enable easy-to-read fonts
- Implementing a brand's identity into the colors, fonts and layout
- Creating a map of the website's structure to ensure intuitive navigation
- Placing images, logos, text, videos, applications and other elements
- Using coding languages, such as HTML and CSS, to create layouts and to style pages
- Making optimized versions of websites and pages both for desktop and mobile viewing

There are two common web design methods: adaptive and responsive design. In adaptive design, the website content is created using standard screen sizes as the frame for the layout. In responsive design, content moves dynamically according to the screen size.

Difference between Web Designer and Web Developer

Web Designer: A web designer is a person that makes the website looks good. They mainly refer to the splendiferous portion of the website as well as its usability. The main responsibility of Web Designers is to focus on the style and overall feel of the website using different software like Photoshop, Corel Draw, etc. to make the website more attractive.

There are different types of web designers as per their role:

- User Experience (UX) designer: UX is known as User Experience. Basically how a user feels and his/her demands getting fulfilled after using the software or automobile or any other gadget designed. In simple terms "is user able to use the product in an efficient manner, the way the developer has intended to use his artifact."
- User interface (UI) designer: User Interface (UI) Design is the creation of graphics, illustrations, and use of photographic art work and typography to enhance the display and layout of a digital product within its various device views. Interface elements consist of input controls (buttons, drop-down menus, and data fields), navigational components (search fields, slider, icons, tags), informational components (progress bars, notifications, message boxes).
- Visual designer: The Visual designer role is to aim for the completed product should look attractive. They are the mixture of User Interface and **Graphic designer**. Don't get confused between a Graphic designer and Visual designer.

Graphic designer is a creative professional who specializes in creating visual content to communicate messages, ideas, or information. They use a combination of typography, images, colors, and layout techniques to design materials for both print and digital media.



Web Developer: Web developers are generally called programmers. They take the design created by the web designers and convert it into a fully functioning website. They use different software and tools like Javascript, jQuery, Node.js, PHP, ASP. NET Python, etc. Their main aim is to build a smooth-running, well-functioning website. Web developers are responsible for

the collaboration with UX designers, UI designers, and visual designers to create web pages based on the design provided by the designer.

Based on their role, web developers can also be divided into three types:

- Front-end developer: The part of a website that user interacts with directly is termed as front end. It is also referred to as the 'client-side' of the application. It includes everything that users experience directly: text colors and styles, images, graphs and tables, buttons, colors, and navigation menu. HTML, CSS, and Javascript are the languages used for Front End development. The structure, design, behavior, and content of everything seen on browser screen when websites, web applications, or mobile apps are opened up, is implemented by front End developers.

Responsiveness and performance are two main objectives of the Front-End. The developer must ensure that the site is responsive i.e. it appears correctly on devices of all sizes no part of the website should behave abnormally irrespective of the size of the screen.

- Back-end developer: Backend is server-side of the website. It stores and arranges data, and also makes sure everything on the client-side of the website works fine. It is the part of the website that you cannot see and interact with. It is the portion of software that does not come in direct contact with the users. The parts and characteristics developed by backend designers are indirectly accessed by users through a front-end application. Activities, like writing APIs, creating libraries, and working with system components without user interfaces or even systems of scientific programming, are also included in the backend.
- Full stack developer: Full stack web developers have the ability to design complete web application and websites. They work on the frontend, backend, database and debugging of web application or websites.

Related Elements of Web Design

The web design process allows designers to adjust to any preferences and provide effective solutions. There are many standard components of every web design, including:

- Layout
- Images
- Visual hierarchy
- Color scheme
- Typography
- Readability
- Navigation

- Content
- Mobile

Layout: The layout of the website is how the material is displayed on a page. Choosing the layout is an essential task for the designer. It should be simple, intuitive and accessible. Web designers can use blank areas called white spaces to organize the elements of the site with grid-based designs to keep them in order. Designers can create specialized layouts for desktop screens and mobile devices. Mobile-friendly websites are a necessity because many visitors access websites on their cellphones or tablets. To ensure a website is ready for mobile visitors, the designer can use a responsive template that adapts to different screen sizes or a mobile-only look that will activate when a non-desktop device connects to the website. A consistent layout between supports contributes to the visitors' trust.

Images: Images are illustrations, graphics, photographs, icons and others used to provide supplementary information to the text. To create the effect desired, designers can pick images that complement each other and the brand that the website represents.

Visual hierarchy: Visual hierarchy is the order in which the user will process the information on the site. The designer creates it by applying a visual pattern to the website. The visual pattern is the way the design directs visitors' eyes and behaviors. For example, F-Patterns or Z-Patterns emphasize the top horizontal section of your site, where most designers place navigation and the brand's logo and sometimes a search box. These are elements that inspire user interaction and brand recognition.

Color scheme: The color scheme is a combination of colors that is in harmony with the brand and industry it represents. To achieve this, they will pick a dominant color and a few others to create a palette. A color palette can be monochromatic (different shades of the same color), analogous (colors close to each other) or complimentary. Designers also account for what colors users are more likely to be attracted to.

Typography: The typography is the style or font of the written content. Web designers pick one or a combination that is attractive and easy to read. To make the best choice, they should choose a font that corresponds to the target audience. Some sites may be better in serif fonts while others can use non-serif fonts, depending on the site's industry, purpose and typical user.

Readability: Readability is when the text of content is easy to see and read on a webpage. The text on the website should be readable because visitors usually spend little time on it and should find information quickly. The designers can achieve this by selecting an appropriate size and pixel for the text. The contrast between the text and the site's background colors also improves readability.

Navigation: The navigational elements are the tools allowing users to choose where they want to go within a website. They may be present in the header, body and footer of the website, depending on the site's layout and structure. These elements are essential as they direct visitors to the information they want as quickly as possible. Designers can choose a variety of navigation designs and layouts, such as using a

button that hides and reveal navigation menus. They can also incorporate one-click arrows and other buttons that direct users back to the top of a page, to a specific area of a page or another page entirely.

Content: Content is all of the information available on the website. It is a pivotal element because visitors want to get information quickly. When the website communicates clearly and grabs the readers' attention, it is more likely to convert them into consumers. The designer can achieve this by using the appropriate tone and provide the right information on the entire website, including the "About" and "Contact" pages.

Use of Web Design

Web designing is used for many important tasks and goals, including: Search engine optimization (SEO) is a method for improving the chances for a website to be found by search engines. Web design codes information in a way that search engines can read it. It can boost business because the site shows up on the top search result pages, helping people to find it.

Customer satisfaction: A professional web design impacts clients' satisfaction positively as it provides them the information they are looking for quickly. It helps the company build a positive relationship with the visitors by ensuring the navigation on its website is easy to understand, predictable and consistent.

Mobile responsiveness: Mobile responsiveness is the feature of a website that allows it to display on a mobile device and adapt its layout and proportions to be legible. Web design ensures sites are easy to view and navigate from mobile devices. When a website is well designed and mobile-responsive, customers can reach the business with ease.

Consistent branding: Branding refers to the promotion of a product with a unique design. Web design helps companies build or maintain a clear brand for their business. When a website expresses a business's brand consistently, it makes it easier to navigate and helps customers more clearly identify the visual elements of a brand as a specific company and its products or services.

Technical efficiency: This term refers to how productive a website can be in making a comfortable experience on a website. Designers can achieve this with clean coding that allows for quick loading times, functioning links and dynamic images and graphics. Web design services also fix those eventual glitches when they occur.

User experience optimization: Web designers run reports to understand the way people are interacting with a website all over the world. They determine which pages have more or less traffic and adapt the web design to optimize the user experience.

Conversion: Conversion happens when a visitor completes a desired action on the website. Attractive web design encourages visitors to stay long enough to be

converted into consumers. They will click on a call-to-action button, exchange valuable information and subscribe or buy a product.

Improve sales: Increasing the number of items sold or acquiring more active customers are objectives of a compelling website. As web design reaches targeted customers and search engines, it helps the business make conversions on their site and improve its sales.

9.2. WEB DESIGN ELEMENTS

Key Notes

Overview

Web design elements are the foundational building blocks used to create visually appealing, functional, and user-friendly websites. These elements work together to convey information, guide user interactions, and establish a cohesive design aesthetic. Understanding and utilizing these elements effectively is essential for crafting engaging and successful web experiences.

Visual Elements

Visual elements are fundamental components used in various forms of visual communication, including graphic design, web design, photography, and more. These elements serve as the building blocks that designers manipulate to convey messages, evoke emotions, and create visually appealing compositions.

Functional Elements

Functional elements are essential components that serve specific purposes and enhance the usability, interactivity, and overall functionality of a design or system. These elements are commonly used in various contexts, such as web design, user interfaces, software development, and industrial design. They play a crucial role in ensuring that users can effectively interact with and navigate through a product or environment.

Lookup for Good Web Design

Good web design encompasses a combination of visual aesthetics, user-centered functionality, and effective communication. Keep in mind that good web design is context-dependent and can vary based on the target audience, industry, and purpose of the website. Analyzing and learning from well-designed websites can provide valuable insights for your own design projects.

LEARNING OBJECTIVES

- Web design: what doesn't work?
- How to get Web Design Done

Overview

When designing a website it's important to consider both the site's appearance and functionality. Integrating these elements will maximize the site's overall usability and performance. Your site's usability includes elements such as an easy-to-navigate interface, appropriate use of graphics and images, well-written and well-placed text, and a color scheme. Your site's performance refers to its speed, ranking, search ability, and ability to capture your audience.

Visual Elements

Here's a quick overview of the elements you should consider while designing your website to make sure everything works well together. Each section will provide tips and tricks to help you get started. Written copy. Fundamentally, your website's appearance

and text go hand-in-hand. It's important to have your content writers and designers work together in order to create a cohesive design with balanced elements. Focus on creating chunks of text (using text blocks) in order to compliment your graphics and images.



Fonts. Choose a font that compliments your overall design. Font should pair with your color scheme, graphics, images, and strengthen the overall tone of your website. Tools like Canva's Font Combinator can help you find a perfect match for your font. Web design tools like PageCloud even include numerous font pairings within their app.

Colors. Colors are one of the most important elements to consider when designing a website. Keep in mind there are many misconceptions about the psychology of color, and it's more important to focus on colors that compliment your overall design and tone of your website. Align your color scheme with your brand and the messages you want to convey to your audience.

Layout. How you decide to arrange your content will have a dramatic impact on both the usability and functionality of your site. There are no specific rules to follow when choosing a layout, however, there are a few main principles to keep in mind. Make sure to consider the needs of your target audience and avoid using an over stimulating layout that might detract from the messages you want to convey.

Shapes. The use of graphic elements in web design can help seamlessly integrate text and images, and help with the site's overall appearance. Combining beautiful colors and shapes can help direct the attention of your sites visitors and contribute to your site's overall flow.

Spacing. Spacing is a key element to creating visually pleasing and easy to navigate websites. Every element in your design will incorporate spacing in one way or another. Appropriate use of whitespace is crucial in creating a design that perfectly balances

text, photos, and graphics. Keeping your spacing consistent can help your users navigate your website with ease. The concept of whitespace is definitely a priority of modern web designers.

Images and Icons. Amazing designs can communicate a lot of information in just a few seconds. This is made possible with the use of powerful images and icons. Choose images and icons that support and strengthen your message. A quick **Google search** for stock images and icons will generate thousands of options. To help simplify your search, here are a few favorites:

Google Search is a widely used search engine that allows users to search for a wide range of information on the internet. You can search for websites, articles, images, videos, and much more using keywords and phrases.

Free images and icons

- Pexels
- Unsplash
- IconMonstr Premium images and icons
- Shutterstock (Getty)

Videos. Integrating videos into web design is becoming increasingly popular amongst designers. When used properly, videos can help your users experience or understand a message that can't be properly conveyed through text or image. Keep in mind that like having a TV screen on in a restaurant, visitors' eyes will be drawn to moving images. Make sure your videos don't compete with or detract from other important elements.

Functional Elements

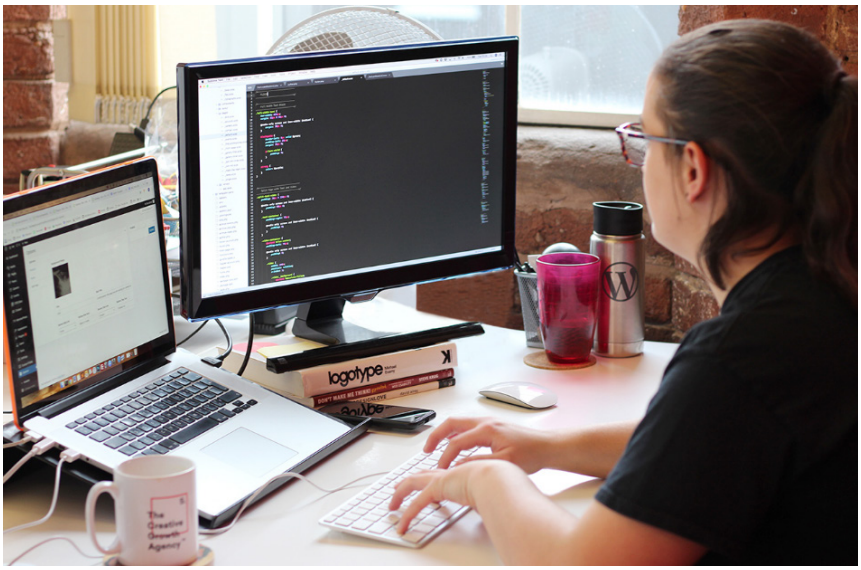
These functional elements are imperative to consider when designing your website. A website that functions properly is crucial for ranking highly on search engines, and giving your users the best possible experience.

Navigation. Your website's navigation is one of the main elements that determines whether your website is functioning properly. Depending on your audience, your navigation can serve multiple purposes: helping first time visitors discover what your site has to offer, giving easy access to your pages for returning visitors, and improving every visitor's overall experience. Check out these best practices for more tips on navigation.

User Interactions. Your site visitors have multiple ways of interacting with your site depending on their device (scrolling, clicking, typing, etc.). The best website designs simplify these interactions to give the user the sense that they are in control. Here are a few examples:

Never auto-play audio or videos

- Never underline text unless its clickable
- Make sure all forms are mobile-friendly
- Avoid pop ups
- Avoid scroll-jacking



Animations. There are tons of web animation techniques that can help your design grab visitor's attention, and allow your visitors to interact with your site by giving feedback. For example, adding "like" buttons or forms can keep your site's visitors engaged. If you're new to web design, we'd recommend keeping your animations simple to avoid developer intervention.

Speed. No one likes a slow website. Having to wait more than a few seconds for a page to load can quickly deter a visitor from remaining on or returning to your site. Regardless of how beautiful, if your site doesn't load quickly, it will not perform well in search (i.e. won't rank high on Google).

Top site builders typically compress your content for faster load times, however, there are no guarantees. Make sure to research which site builders will work best for the content you will have on your site. For example, PageCloud optimizes your images to ensure fast loading times for sites with large and/or multiple photos.

Site structure. A website's structure plays an important role in both user experience (UX) and search engine optimization (SEO). Your users should be able to easily navigate through your website without encountering any structural issues. If users are getting lost while attempting to navigate through your site, chances are "crawlers" are too. A crawler (or bot) is an automated program that searches through your website and can determine its functionality. Poor navigation can lead to a poor user experience and site ranking.

Cross-browser & cross-device compatibility. A great design should look polished on all devices and browsers (yes, even Internet Explorer). If you're building your site from scratch, we'd recommend using a cross-browser testing tool to make this tedious process faster and more efficient. On the other hand, if you're using a website building platform, the **cross-browser testing** is typically taken care of by the company's development team allowing you to focus on design.

Cross-browser testing is a critical process in web development that involves testing a website or web application across different web browsers and browser versions to ensure consistent functionality, appearance, and user experience.

Lookup for Good Web Design

Good web design isn't subjective. With other types of design, like illustration or sticker design, a lot of what constitutes "good" is up to the viewer's taste. With web design, the line between "good" and "not good" is much more defined. A well-designed website is a website that perfectly creates the experience your visitor is looking for.



Conversions can be anything, like signing up for a newsletter, making a purchase, opening an account or accessing more content on the website.

These include:

- Compelling use of negative space
- Clearly presented choices for the user (the fewer choices the user has, the less likely they are to become overwhelmed and confused)

- Obvious, clear calls to action
- Limited distractions and a well thought out user journey (i.e. using only images and text that are 100% relevant to the subject on the page, featuring only buttons that lead to desired actions and using font variations for emphasis and calls to action, not just for the sake of featuring different fonts)
- Responsive design (a design that resizes and reorients itself to the user's screen, making the website easy to use on any device: phone, tablet, laptop or desktop browser.
- Appropriately sized fonts that follow a hierarchy (see "Limited distractions")
- Relevant, high-quality content and images that hook your readers' attention
- A balance between the amount of text and images on each page (too much text can overwhelm a visitor, too little text can be equally disengaging)

If you don't think aesthetic design choices affect conversions, think again. Your website must be attractive—especially to your prospective user, so try to figure out what style is going to speak to them. Invest in eye-catching images that work with your brand. Stay away from obvious stock photography. Remaining true to your brand is key to successful website design. Even the most gorgeous website is useless if it doesn't match your brand.

Other building blocks of an effective web design are:

- Buttons
- Fonts
- Color palette
- The visual balance between your images and copy on each page

Of course, good web design isn't just utilitarian. Visitors like websites that are engaging and fit the brands' aesthetics. No matter how you achieve it, meshing an on-brand, engaging look with design elements that convert is how you win at web design.

Web design: what doesn't work?

We've gone over what good design is. Now let's talk a bit about what it isn't. As a general rule, visitors shouldn't have to do any work to use your website. The whole experience of using your website should be straightforward and intuitive.

Here are some examples: Clear calls to action are great web design; murky ones are bad web design. High contrast fonts are smart, effective web design; low contrast fonts that are hard to read are poor web design.

Here are a few other elements to avoid:

- Distracting images and backgrounds. As a general rule, stay away from tiled backgrounds. Though there are a few select instances where a tiled background could be a good choice, in most cases they're distracting.

- Non-responsive design. Nowadays your website simply needs to be mobile responsive.
- Unclear links and buttons. Visitors shouldn't have to hunt for links and buttons, they should be able to quickly see which images and pieces of text will take them to new pages or confirm their choices. Similarly, users should be able to clearly recognize fillable fields.
- Generic or irrelevant stock photos and filler text without valuable information.

Certain web design elements, like grid layouts, aren't inherently good or bad choices. They can be used in effective and ineffective ways, so taking care to do them right is a must. Another tricky web design element is animation. It's not 1999 anymore, you shouldn't have a comet tail trailing the user's cursor or make them scroll past a row of dancing hamsters to get to your content. But an animated exit pop-up that brings visitors' attention back to your site and gets them to convert? Yes.

How to get Web Design Done

Whew, getting web design right is a complex process and there are lots of things to look out for. Thankfully, you don't have to do it alone. When you're working with a professional web designer you can rely on their skill to get the perfect result. There are a few different ways you can get the website you need. The way that's right for you and your company depends on how complex of a website you need, how much you want to spend and how much of the work you can do yourself. You can work directly with a freelance web designer. Simply browse designer portfolios and pick the designer you like best and who fits the style and look you're going for. A freelance designer can customize an existing template or design a completely fresh template for your website. If you want your website built from scratch, you can work with a freelancer who has the skills to make it happen. Creating your own site with a template-driven website builder (like Wix or Squarespace) is also an option.

You'll need to be more hands on and if you're not a designer you will be limited to what these platforms offer, but you can always hire a designer to customize a template for you. You can also work with an agency that builds custom websites. Pick this option if you need to build a complex website from the ground up, but be aware that this end-to-end solution will cost you more. Be prepared to invest in high quality web design. Your web design could cost a few hundred to tens of thousands of dollars, depending on its complexity. What you'll spend is proportionate to what you need. So how do you know if you need a custom-built website or if a template will do you just fine? That all depends on what you need your website to do and what you plan for it. If scaling is part of your business plan, get a custom website. If you need it to be search engine friendly, if you need to adapt it to various business needs, if you have very specific ideas about the website that demand that it be built to your specifications, you need a custom website. When these aren't priorities for you and you don't have a large budget, a customized template is the way to go.

9.3 TYPES OF WEBSITE DESIGN: ADAPTIVE VS. RESPONSIVE

Key Notes

Observation

Website design is a multidisciplinary field that requires collaboration among designers, developers, content creators, and stakeholders to create a cohesive and impactful online presence. It blends creativity, user experience design, and technical implementation to craft websites that effectively communicate messages, deliver value, and engage users.

Adaptive Websites

Adaptive websites, also known as adaptive web design, refer to websites that adjust their layout and content presentation based on the characteristics of the user's device or screen size. Unlike responsive design, which uses fluid grids and flexible elements to adapt to different screen sizes, adaptive design involves creating distinct layouts for specific device types or breakpoints.

Responsive Websites

Responsive web design is an approach to designing and developing websites that aims to ensure optimal user experience across a wide range of devices and screen sizes. The goal is to create websites that adapt and respond to the user's device, whether it's a desktop computer, laptop, tablet, or smartphone.

LEARNING OBJECTIVES

- Find the basics of adaptive website builders
- Define the role of responsive website builders

Observation

Website design refers to the process of creating and structuring the visual and functional elements of a website. It involves the thoughtful arrangement of layout, color schemes, typography, imagery, and interactive elements to communicate a brand's message, engage users, and provide a seamless user experience. Effective website design aims to strike a balance between aesthetics and functionality, catering to both the visual appeal and the practical needs of users.

Understanding the pros and cons of adaptive and responsive websites will help you determine which website builder will work best for your website design needs.

You might come across articles online that talk about a whole bunch of different website design styles (fixed, static, fluid, etc.). However, in today's mobile-centric world, there are only two website styles to use to properly design a website: adaptive and responsive.

Adaptive Websites

Adaptive web design uses two or more versions of a website that are customized for specific screen sizes. Adaptive websites can be split into two main categories based upon how the site detects what size needs to be displayed:

1. Adapts based on device type. When your browser connects to a website, the HTTP request will include a field called "user-agent" that will inform the server about the type of device attempting to view the page. The adaptive website will know what version of the site to display based on what device is trying to reach it (i.e. desktop, mobile, tablet). Issues will arise if you shrink the browser window on a desktop because the page will continue to display the "desktop version" rather than shrinking to the new size.
2. Adapts based on browser width. Instead of using the "user-agent", the website uses media queries (a CSS feature that enables a webpage to adapt to different screen sizes) and breakpoints (certain width sizes) to switch between versions. So instead of having a desktop, tablet, and mobile version, you will have 1080px, 768px, and 480px width versions. This offers more flexibility when designing, and a better viewing experience as your website will adapt based on screen width.

Pros

- WYSIWYG editing (what you see is what you get)
- Custom designs are faster and easier to build without code
- Cross-browser and cross-device compatibility
- Fast-loading pages

Cons

- Websites that use "device-type" can look broken when viewed in a smaller browser window on a desktop
- Limitations on certain effects that only responsive sites can accomplish

Adaptive Website Builders

Wix and PageCloud are arguably the two best visual website builders on the market today. Both use an adaptive approach, meaning their drag-and-drop and WYSIWYG capabilities are second to none. You can build just about anything without having to write a single line of code. Wix has been around since 2006 and has since developed a wide range of features and templates to suit just about every business need. Today, it's considered one of the easiest tools for beginners. Wix has been around since 2006 and has since developed a wide range of features and templates to suit just about every business need. Today, it's considered one of the easiest tools for beginners.

Although it's hard to choose a winner in this category, here are few things to keep in mind:

- If you're looking for the most customizable experience, choose PageCloud.
- If you're looking for something really easy and don't have much design experience, choose Wix.
- If you're looking to work with a developer, choose PageCloud.
- If you want many template options, choose Wix.
- If you like shortcuts and the experience found in desktop publishing apps, choose PageCloud.

As both platforms offer free trials, we'd recommend trying them both before deciding.

Responsive Websites

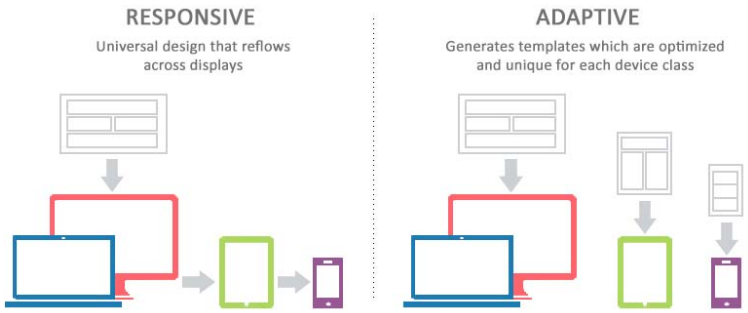
Responsive websites can use flexible grid layouts that are based on the percentage each element takes up in its container: if one element (e.g. a header) is 25% of its container, that element will stay at 25% no matter the change in screen size. Responsive websites can also use breakpoints to create a custom look at every screen size, but unlike adaptive sites that adapt only when they hit a breakpoint, responsive websites are constantly changing according to the screen size.

Pros

- Great experience at every screen size, regardless of the device type
- Responsive website builders are typically rigid which makes the design hard to "break"
- Tons of available templates to start from

Cons

- Requires extensive design and testing to ensure quality (when starting from scratch)
- Without accessing the code, custom designs can be challenging It's important to note that website builders can include both adaptive and responsive features. For example, PageCloud recently introduced a series of features that allow your content to act responsive even though the website itself is still adaptive.



Responsive Website Builders

Website builders are online platforms that allow individuals and businesses to create and design websites without the need for coding or technical expertise.

Tools like Square space offer responsive website builders, however, this means your editing experience is more limited. Creating a fluid responsive website is hard, and without knowing how to code, it is nearly impossible to build unique websites using responsive **website builders**.

This is where more complex web design tools, like Webflow and Froont, come into play. Here are some of the pros and cons to consider when looking to adopt one of these tools:

Pros

- Ability to create custom responsive sites without having to write code
- Unmatched control over every element on the page
- Ability to export code to host elsewhere

Cons

- Complex tools with steep learning curves
- Slower design process than adaptive website builders

9.4 PRINCIPLES OF DESIGN APPLIED TO WEBSITES

Key Notes

Principles

Effective web design goes beyond aesthetics and encompasses various principles that contribute to a user-friendly, visually appealing, and functional website. Effective web design requires a balance between creativity, functionality, and user-centered thinking.

Design with accessibility in mind, ensuring that all users, including those with disabilities, can access and interact with the content. This involves using proper markup, alt text for images, and clear navigation.

Functional Components of Web Design

Web design consists of several functional components that work together to create a cohesive and user-friendly website. These components encompass both the front-end (user interface) and back-end (server-side) aspects of web development. Keep in mind that the specific implementation of these components can vary based on the project's requirements, technology stack, and design approach.

LEARNING OBJECTIVES

- Prepare website layouts strategy
- Define the phenomena of website maintenance

Principles

One of the first parts of understanding what web design knows what good web design is—and how to achieve it. We can have a look at the principles of design for reference, a theory practiced by artists and designers which outline the visual qualities any composition should aim for. Applying these principles to web design can help beginner and advanced web designers alike achieve a site with a harmonious look and feel. Of course, these are not strict rules to follow, but rather guidelines to learn how we can apply the various elements of a website's design. Take it from Picasso, "learn the rules like a pro, so you can break them like an artist." Once you understand the goals of web design and become more comfortable with each website element, you can tweak the approach with a more creative touch.

Let's explore the following principles:

- Balance
- Contrast
- Emphasis
- Movement
- Rhythm
- Hierarchy

- White Space
- Unity

Balance. Visual balance means ensuring that none of the elements in a single composition are too overpowering. This can be applied to web design by drawing an imaginary line down the center of a webpage and arranging elements so that the visual weight is equal on both sides. There are two main ways to achieve balance on a website:

- Symmetrical balance is when the visual weight on both sides of that line is equal and arranged like a mirror image. When applied to your web design, this can evoke feelings of balance, beauty and consistency.
- Asymmetrical balance is when the visual weight is equal on both sides, but the composition and order of elements will vary (i.e., not a mirror image). A balanced asymmetrical design is considered a modern approach and will create a more dynamic experience for the viewer, while maintaining a harmonious composition.

Contrast. Contrast refers to arranging juxtaposing elements in a way that highlights their differences: dark and light, smooth and rough, large and small. When contrast is in the picture, it's dramatic and exciting qualities can captivate visitors as they scroll through your site.

Emphasis. The principle of emphasis reminds us that not all website elements are equal. Whether it's your logo, a CTA, or an image; if there's something on your webpage that visitors should notice first, applying the principle of emphasis with the use of bright color, animation, or size will ensure it's the dominant aspect of your composition.

Movement. When applied to web design, movement is what guides visitors from one element to the next. By controlling the size, direction, and order of elements on an individual web page's composition, you can direct the movement of the viewer's eye throughout your site.

Rhythm. Rhythm refers to the repetition of elements in order to create consistency, cohesiveness, or to amplify a certain message. Repeating characteristics such as your logo, brand colors and using the same typeface, will also strengthen your brand identity and presence on the web.

Hierarchy. Placing your business name at the bottom of your homepage is simply poor web design practice. Can you guess why? First-time visitors would have to scroll all the way down your site to know who you are. This is something we understand from the principle of hierarchy, which teaches us that the most important content should be placed in a prominent spot where visitors immediately see and interact with it.

White space. In art and design, any area of a composition that's void of visual elements is referred to as white space (psst: even when it's not actually white). This might not seem like a critical thing to pay attention to, but the conscious arrangement

of white space in web design will give the visual elements of a webpage room to breathe. It can also help achieve other goals in your composition, such as hierarchy, balance, emphasis, and more.

Unity. Unity is the culminating effect of all the individual elements you've added to your site, ideally resulting in one harmonious composition. The goal of unity in web design is to ensure that visitors will not get overwhelmed, confused, or turn away from your site. It might take a few tries to get it right, but once you achieve a unified web design, you can ensure that each aspect of your site plays a valuable role in its function and performance. This also means paying attention to what elements you include, where and how you position them, and if they are truly serving a purpose.

Website Layouts

Planning your website layout is like setting its foundation, since it will determine the arrangement and sequence of visual elements on each page of your website. This critical step in web design plays a role in a site's visual appearance, level of usability, and amplifying its message.

The best layout for your website can be determined by a variety of factors: the goals of your website, the message you want to convey to viewers, and what kind of content you'll include. While there's no "one-size-fits-all" solution, there are two major directions you can take:

- Layouts to accommodate your content: The layout you choose should be fitting for the type of content. For example, if you want a layout that will showcase products you might go for one that leaves ample space for highlighting images. A blog layout, on the other hand, will need to convey new information in an organized way.
- Common layouts: There are plenty of tried-and-tested website layouts out there. These tend to feel familiar to users, as they build on their existing expectations or past experiences of other websites. Since they may result in a more intuitive, easy-to-use interface, they can be great for beginners.



When designing a website of your own, you can use website templates in a wide range of categories to provide a solid infrastructure for your site's layout. If you want to design a layout from scratch, we recommend using **wireframes** to start. This process will allow you to draft out the layout of your website before the implementation process.

Wireframes serve as a crucial early-stage design tool used by designers, developers, and stakeholders to outline the placement of various elements on a webpage without focusing on colors, graphics, or detailed visuals.

Functional Components of Web Design

Website functionality essentially refers to how your website works; everything from its speed and ease of use, to what specific actions can be performed on it.

At the risk of dating myself, when I think about how speedy and efficient websites function today—compared to the '90s (one can hear AOL's dial up signal in the background), One is reminded that we've come a long way. Given the advancements across the industry of web design, it's in our best interest to utilize the modern tools available to guarantee our websites perform well, and are easy to use.

Let's go over the components of web design that will affect how your site functions:

- Navigation
- Speed
- SEO
- UX
- Adaptive design vs. responsive design

Navigation. A single web design can consist of multiple pages and items to be viewed or used by the visitor. Website navigation is what lets visitors find the webpages they need after they arrive at your site. Adding a navigation menu to your design is the best way to provide smooth navigation for visitors. A website's menu links to multiple items on your site and serves the user by helping them navigate between different pages and sections.

Depending on your site's design, you might choose between the following types of menus:

- Classic navigation menu: This popular menu type is placed in the website's header, presented as a horizontal list.
- Sticky menu: Also known as a fixed or floating menu, this menu stays put as visitors scroll down the site.

- Hamburger menu: A hamburger menu is an icon made up of three horizontal stripes that opens to a full menu once it's clicked.
- Dropdown menu: A menu in which a list of additional items opens up once visitors click on—or hover over—one of its items.
- Sidebar menu: A list of menu items located on the left or right side of a webpage.

Speed. Speed measures how quickly your site fully loads after a visitor enters it. No one, especially in our fast-paced generation, likes a slow website. It's got to be performing at high speed. In fact, studies show a 38% rise in bounce rates when a webpage's loading time surpasses 3 seconds. No matter how pretty the web design is, you can't compete unless you're providing visitors with the smoothest browsing experience. Many factors can affect the loading time of a page; some are related to a visitor's own device or internet connection, while others might be specific to the website they're attempting to browse. Today, there are proven practices and tools checking for auditing your site's performance and improving the loading speed of its pages.

SEO. SEO, or search engine optimization, is the process of optimizing a website so that it ranks well on search engines. Since it plays a big part of your website's success, we think it deserves to be included here with the functional elements. The more people who are able to find you on Google means the more visitors you'll have using your website.

Although SEO is an ongoing task, there are steps you can take to help your site's performance from the get-go. For example with Wix's SEO solutions, your site's built-in infrastructure will already support positive performance.

There are a number of common SEO practices you can take before hitting publish, too: include headings in your content, add alt text to images, use meta descriptions on relevant pages, and choose a domain name that represents your brand or business.

UX. Since the explosion of computer usage in the 1980s, the tech industry has constantly been exploring how human beings can better interact with technology. This is the practice of UX—or user experience—and when applied properly to web design, it can have a great impact on the user's journey.

The term UX is often used interchangeably with "user interface" design or "usability,", which are really , in reality, a part of the larger UX vision. While UX designers are concerned with these aspects of a product, they are also involved in looking at the big picture, finding ways to perfect and develop the products, branding, design, usability, and function.

The process of UX design is meant to ensure that a website encompasses high-quality interactions, content, products and services, by improving these seven factors:

- Useful
- Usable
- Findable

- Credible
- Desirable
- Accessible
- Valuable

Website Maintenance

The web design industry is one that's constantly introducing new features, tools and solutions. The down side of this quickly evolving world is that it requires you - and your website - to stay constantly up-to-date and implement a website maintenance plan. After you've completed your first design, you will eventually have to update your website to ensure the content is relevant, and the design is not obsolete. While seemingly futile, any outdated elements on your website can negatively impact your visitors' interactions, resulting in decreasing overall performance and sales.



Check in on your website at least once a month to make sure there are no bugs, everything works properly and that your information is current. When considering a redesign, think about the changes you can make to keep your web design relevant, improve it's ease of use, or amplify its performance. This might mean adding fresh visual content, an extra page, working on SEO, or performing an accessibility audit.



TECHNIQUE 32

USER EXPERIENCE (UX) DESIGN

User Experience (UX) design focuses on creating products, systems, and experiences that are user-centered, intuitive, and enjoyable for the people who use them. It encompasses a variety of disciplines and techniques that aim to understand users' needs, behaviors, and preferences in order to design effective and satisfying interactions.

- □ We already practice this technique
- □ This technique would not be suitable for our operation
- □ We practiced this in the past and need to jumpstart
- □ We should implement this technique



TECHNIQUE 33

WEB DESIGN AND SEO

Web design and search engine optimization (SEO) are two interconnected disciplines that together contribute to the success of a website. Web design focuses on creating visually appealing, user-friendly, and functional websites, while SEO aims to optimize the site's visibility and ranking in search engine results. An effective web design incorporates elements such as responsive layouts, intuitive navigation, and engaging visuals to enhance user experience. This includes using clean code, optimizing images, and ensuring fast loading times.

- □ We already practice this technique
- □ This technique would not be suitable for our operation
- □ We practiced this in the past and need to jumpstart
- □ We should implement this technique

SUMMARY

- Web design is the process of planning, conceptualizing, and arranging content online. Today, designing a website goes beyond aesthetics to include the website's overall functionality. Web design also includes web apps, mobile apps, and user interface design.
- Desktop apps require designers to create their design and send it to a development team who can then convert the design to code. The most popular desktop apps for designing websites are Photoshop and Sketch.
- Web design identifies the goals of a website or webpage and promotes accessibility for all potential users. This process involves organizing content and images across a series of pages, integrating applications and other interactive elements.
- A web designer is a person that makes the website looks good. They mainly refer to the splendiferous portion of the website as well as its usability.
- Web developers are generally called programmers. They take the design created by the web designers and convert it into a fully functioning website.
- The web design process allows designers to adjust to any preferences and provide effective solutions.
- The layout of the website is how the material is displayed on a page. Choosing the layout is an essential task for the designer. It should be simple, intuitive and accessible.
- Visual hierarchy is the order in which the user will process the information on the site. The designer creates it by applying a visual pattern to the website. The visual pattern is the way the design directs visitors' eyes and behaviors.
- Web designing is used for many important tasks and goals, including: Search engine optimization: Search engine optimization (SEO) is a method for improving the chances for a website to be found by search engines.
- A website's structure plays an important role in both user experience (UX) and search engine optimization (SEO). Your users should be able to easily navigate through your website without encountering any structural issues.
- Website design refers to the process of creating and structuring the visual and functional elements of a website. It involves the thoughtful arrangement of layout, color schemes, typography, imagery, and interactive elements to communicate a brand's message, engage users, and provide a seamless user experience.
- Adaptive web design uses two or more versions of a website that are customized for specific screen sizes.

KNOWLEDGE CHECK

1. What does CSS stand for in web design?

a. Computer Style Sheets
b. Central Style System
c. Cascading Style Sheets
d. Creative Style Solutions

2. What is the purpose of HTML in web design?

a. Handling server-side logic
b. Creating visual styles
c. Defining the structure and content of a webpage
d. Enhancing user interactions

3. What is the primary goal of responsive web design?

a. Creating visually stunning animations
b. Achieving a high search engine ranking
c. Designing for mobile devices only
d. Providing an optimal user experience across various devices

4. Which design principle focuses on arranging content elements based on their importance?

a. Responsive design
b. Visual hierarchy
c. Minimalism
d. Parallax scrolling

5. What does UX stand for in the context of web design?

a. User Experience
b. XML Syntax
c. Extended XHTML
d. XQuery Language

REVIEW QUESTIONS

1. How do you decide on a suitable web design tool?
2. What are some essential elements related to web design?
3. What aspects should you evaluate to determine the quality of a web design?

5. How do adaptive websites adjust to different devices compared to responsive websites?
6. What are the key functional components that make up a web design?

Check Your Result

1. (c) 2. (c) 3. (d) 4. (b) 5. (a)

REFERENCES

1. Barrell, Dylan. Agile Accessibility Explained: A practical guide to sustainable accessible software development, Amazon Digital Services, 2019.
2. Blanck, Peter. eQuality: The Struggle for Web Accessibility by Persons with Cognitive Disabilities, Cambridge Disability Law and Policy Series, 2015.
3. Byrne, Jim. 60 hot to touch Accessible Web Design tips – the tips no web developer can live without!, Jim Byrne, 2006, (ISBN: 978-1-4116-6729-7).
4. Chisholm, and May. Universal Design for Web Applications: Web Applications That Reach Everyone, O'Reilly Media, 2008.
5. Coolidge, Doner, Robertson, and Gray. Accessibility Toolkit - 2nd Edition, BCcampus, 2020.
6. Coombs, Norman. Making Online Teaching Accessible, Jossey-Bass, 2010. "Making Online Teaching Accessible" is downloadable from Bookshare.
7. Digital Education Strategies, The Chang School. What You Can Do to Remove Barriers on the Web, The Chang School, 2020.

INDEX

A

Ajax client behavior 186
Ajax Control Toolkit 196
Application Programming Interfaces (APIs) 64
Arithmetic operators 155, 168
Assignment operators 163
Asynchronous 173, 174, 176, 183, 185, 186, 191, 194, 196

B

Berkeley Software Distributions (BSDs) 216
Bitwise operators 159
Boolean 133, 134, 162
boolean variable 95
Bubbling Phase 79

C

Capture Phase 78
cascading effect 41
Cascading Style Sheet 188, 189
CFScript 244, 249, 250, 258, 260, 261, 262, 263, 264, 265, 266, 267, 269, 274, 275
class attribute 48
ColdFusion 237, 238, 239, 240, 241, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 260, 262, 263, 264, 265, 266, 268, 269, 270, 273, 275, 279
Computer programming languages 213
Conditionals 86, 88, 90
Corel Draw 310

D

debugger 77, 78, 96, 104, 105, 106
Debugging 104, 105, 106, 312
Delegate Event Listeners 81
dithering 289, 290
Document Object Model 174, 175, 188, 189, 204
Double-quotation marks (257)
Dynamic application 244

E

Efficiency 115, 117
Embedded styles 42, 51, 53
Error handling 143, 146, 147, 168
event object 69, 77, 78, 79, 80, 83, 84
Extensible Markup Language (XML) 214, 223
External styles 51, 55

F

File handling 143, 168
Floating-point number 131

G

GIF (Graphic Information Format) 289
Graphics 281, 282, 283, 284, 285, 287, 291, 294, 295, 299

H

Horizontal lines 7
HTML document 8, 11, 12, 24, 25, 26, 29
Hypertext Preprocessor 115, 116, 120, 121, 168
Hypertext Transfer Protocol (HTTP) 174, 214, 223

I

illustration 282, 284
Inline styles 42, 51, 52
integrated development environment (IDE) 105
Integrated object oriented 116
Interface design 308

J

Java language libraries 212
JavaScript 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 77, 82, 84, 86, 88, 94, 95, 96, 104, 106, 108, 109, 110, 111, 112
JavaScript Object Notation (JSON) 214, 223
JPEG (Joint Photographic Experts Group) 288

L

Logical operators 161

M

Macromedia Dreamweaver MX 67
Microsoft Ajax architecture 185
Microsoft FrontPage 67
Mobile-friendly 319

N

Native file formats 292
Nonbreaking space entity 9

O

Object-oriented languages 207, 211, 212, 236
Object-oriented programming (OOP) 135
Open-source community 116

P

Pairing 97, 99, 100, 101
Pair programming 97, 98, 99
Personal digital assistant (PDA) 118
Photoshop 309, 310, 334
PHP function 124, 143, 159, 168
pixels 281, 282, 283, 289, 290, 291, 295, 300, 303
PNG (Progressive Network Graphics) 291
Programming language 120, 121, 167
Prototype 209

R

Representational State Transfer (REST) 215
Reserved Word 257, 264
Resolution 281, 282, 303
Responsive websites 310, 323, 325, 336
Rich Internet Applications (RIA) 180

S

Sandbox Security 241
schematics 282
Search engine optimization (SEO) 308, 319, 333, 334
selector 46, 47, 48, 49
specifications 42, 43
String 117, 124, 132, 133, 134, 142, 145, 151, 152, 155, 156, 157, 159, 160, 161, 162, 165, 168, 169

stylesheet 39, 43, 44, 53, 54, 56, 57, 58, 63

T

Target Phase 79
textfields 76
typography 282, 295, 300

U

User-defined functions (UDFs) 249, 250
User experience design 308, 323
User-friendly experience 308
User interface design 308, 334

V

Variables 86, 89, 92, 94, 95
Visual designer 311

W

Web application 312, 320
Web browser 238, 278
Web design 307, 308, 310, 314, 316, 317, 321, 327, 333, 334
Web designer 307, 308, 310, 322, 334
Web Developer 310, 311
Web development 308, 320, 327
Web graphics 281, 282
Web service 239, 240
Website builder 309, 310, 322, 323
Windows Communication Foundation (WCF) 185
World Wide Web Consortium (W3C) 19

Theory, Practice and Techniques in Web Development

2nd Edition

Web development is a rapidly growing field that encompasses various skills and disciplines. It involves creating websites or web applications, and it plays a crucial role in shaping the online landscape. Web developers utilize programming languages such as HTML, CSS, and JavaScript to design and build user-friendly and visually appealing websites. The process of web development begins with understanding client requirements and gathering information about the target audience. This information is used to create wireframes or mockups, which provide a visual representation of the website's structure. Once the initial planning phase is complete, developers move on to coding the website using markup languages like HTML. HTML provides structure to the content by defining paragraphs, headers, lists, images, links, etc. Cascading Style Sheets (CSS) are applied alongside HTML to enhance the appearance of web pages. CSS controls elements' style attributes like color, font size, layout positioning - allowing designers to create responsive designs that look great on all devices. In addition to HTML and CSS, JavaScript is what brings interactivity to websites through algorithms executed by web browsers flawlessly. Using JavaScript libraries or frameworks like ReactJS or AngularJS streamlines development processes by providing pre-designed building blocks for complex functionalities. Web developers should also have an understanding of databases like MySQL or MongoDB for storing data within web applications dynamically. They need knowledge of server-side scripting using PHP, Python or Node.js as well as familiarity with content management systems (CMS) such as WordPress.

The book "Theory, Practice and Techniques in Web development" 2nd Design offers a comprehensive guide for individuals entering the field of web development. Focusing primarily on front-end aspects such as HTML authoring, graphic production, and media development, this resource provides insight into real-world projects through the design, planning, and construction process. While it does not delve into programming, scripting or server functions in depth, it lays the necessary foundation by introducing relevant terminology and technologies. Suitable for all skill levels from beginners seeking detailed explanations to professionals requiring quick reference materials, this book serves as an indispensable tool for acquiring skills needed to create dynamic and interactive web applications using various frameworks—making it an essential companion for students delving into the world of web development.

Éric Fournier is a Professor in the Department of Computer Science & Engineering. His expertise spans software engineering in mobility, modeling, testing, and ethnographic action research. He is also actively involved in advancing computer science education, focusing on new technologies, problem-based and work-integrated learning, as well as promoting academic integrity.

ISBN 978-1-98469-093-7

9 781984 690937 00000

BIBLIOTEX Digital Library

www.3ge-learning.com
email: info@3ge-learning.com