VIRTUAL UNIVERSITY OF PAKISTAN

Fundamentals of Front End Development

CS202

Course Handouts

GRADUP

Practice. Pass. Graduate.

thegradup.com

Week1

Lecture 1

- Topic-001 Introduction Front end Development
- Topic-002 URL-Uniform Resource Locator
- Topic-003 HTTP Basics
- Topic-004 Webserver, Services and Agents

Lecture 2

- Topic-005 Domain Hosting, FTP
- Topic-006 Fundamentals of HTML
- Topic-007 HTML Basics
- Topic-008 HTML Elements and Attributes

Lecture 3

- Topic-009 HTML Elements and Attributes_2
- Topic-010 Heading and Paragraph
- Topic-011 HTML Styling
- Topic-012 HTML Quotations

Week 2

Lecture 4

- Topic-013 HTML Computer Code
- Topic-014 HTML Comments
- Topic-015 HTML Links-The Target Attribute
- Topic-016 HTML Images

Lecture 5

- Topic-017 HTML Table
- Topic-018 HTML Table Handling
- Topic-019 HTML List

© Virtual University of Pakistan

Lecture 6

Topic-021 HTML Layout

Topic-022 HTML Iframe

Topic-023 HTML Form

Topic-024 HTML Colors

Week 3

Lecture 7

Topic-025 HTML Heads

Topic-026 HTML Entities and Symbol

Topic-027 HTML Encoding (Character Sets)

Topic-028 HTML URL Encoding

Lecture 8

Topic-029 HTML and XHTML

Topic-030 Introduction to CSS

Topic-031 CSS Syntax

Topic-032 CSS Selectors

Lecture 9

Topic-033 CSS Insertion

Topic-034 CSS Background

Topic-035 CSS Text and Fonts

Topic 036 CSS Text and Fonts_2

Week 4

Lecture 10

Topic-037 CSS Links

Topic-038 CSS Lists

Topic-039 CSS Tables

Topic-040 CSS Box Model

Lecture 11

Topic-041 CSS Border and Outline

Topic-042 CSS Margin and Padding

Topic-043 CSS Dimension

Topic-044 CSS Display

Lecture 12

Topic-045 CSS Positioning & Floats

Topic-046 CSS Positioning & Floats_2

Topic-047 CSS Align

Topic-048 CSS Combinators

Week 5

Lecture 13

Topic-049 CSS Pseudo-Class

Topic-050 CSS Pseudo Elements

Topic-051 CSS Class

Topic-052 CSS Navigation Menu

Lecture 14

Topic-053 CSS Image Opacity

Topic-054 CSS Image Sprites

Topic-055 CSS Media Types

Topic-056 CSS Attribute Selector

Lecture 15

Topic-057 Fundamentals of JavaScript

Topic-058 JavaScript Syntax

Topic-059 JavaScript Statements

© Virtual University of Pakistan

Week 6

Lecture 16

Topic-061 JavaScript Variables & Operators

Topic-062 JavaScript Variables & Operators_2

Topic-063 JavaScript Functions

Topic-064 JavaScript Objects

Lecture 17

Topic-065 JavaScript Scope

Topic-066 JavaScript Events

Topic-067 JavaScript Strings

Topic-068 JavaScript Numbers

Lecture 18

Topic-069 JavaScript Number Methods

Topic-070 JavaScript Math

Topic-071 JavaScript Date - a

Topic-072 JavaScript Date - b

Week 7

Lecture 19

Topic-073 JavaScript Arrays

Topic-074 JavaScript Arrays (Cont.…)

Topic-075 Arrays Methods

Topic-076 JavaScript Arrays Methods

Lecture 20

Topic-077 JavaScript Comparisons

Topic-078 JavaScript Conditions

Topic-079 JavaScript Switch Statement

Lecture 21

Topic-081 JavaScript While Loop

Topic-082 JavaScript Break and Continue

Topic-083 JavaScript Datatypes

Topic-084 JavaScript Type Conversion

Week 8

Lecture 22

Topic-085 JavaScript RegEXP

Topic-086 JavaScript Hoisting

Topic-087 JavaScript Error Handling

Topic-088 JavaScript Debugging

Lecture 23

Topic-089 JavaScript Best Practices - a

Topic-090 JavaScript Best Practices - b

Topic-091 JavaScript Common Mistakes

Topic-092 JavaScript Common Mistakes (Cont.…)

Lecture 24

Topic-093 JavaScript Performance

Topic-094 JavaScript Reserve Words

Topic-095 JavaScript Global Functions and Properties

Topic-096 JavaScript Window Object

Week 9

Lecture 25

Topic-097 JavaScript and Forms

Topic-098 JavaScript Case Study

Topic-099 AJAX Introduction

Topic-100 AJAX Sending Request - a

Lecture 26

Topic-101 AJAX Sending Request - b

Topic-102 AJAX Receiving Response - a

Topic-103 AJAX Receiving Response - b

Topic-104 AJAX Case Study

Lecture 27

Topic-105 jQuery Introduction

Topic-106 jQuery Syntax

Topic-107 jQuery Selectors

Topic-108 jQuery Traversing

Week 10

Lecture 28

Topic-109 jQuery Events - a

Topic-110 jQuery Events - b

Topic-111 jQuery Effects - a

Topic-112 jQuery Effects - b

Lecture 29

Topic-113 jQuery Effects - c

Topic-114 jQuery Effects - d

Topic-115 jQuery Animation

Topic-116 XML Element

Lecture 30

Topic-117 jQuery Chaining

Topic-118 jQuery HTML - a

Topic-119 jQuery HTML - b

© Virtual University of Pakistan

Week 11

Lecture 31

Topic-121 jQuery CSS - b

Topic-122 jQuery Dimensions

Topic-123 jQuery Traversing - a

Topic-124 jQuery Traversing - b

Lecture 32

Topic-125 jQuery Traversing - c

Topic-126 jQuery AJAX - a

Topic-127 jQuery AJAX - b

Topic-128 jQuery noConflict

Lecture 33

Topic-129 jQuery Case Study

Topic-130 XML Introduction

Topic-131 XML Usage

Topic-132 XML Tree

Week 12

Lecture 34

Topic-133 XML Syntax

Topic-134 HTML5 Canvas Gradients

Topic-135 XML Attributes

Topic-136 XML NameSpaces

Lecture 35

Topic-137 XML DTD

Topic-138 XML Schema

Topic-139 XSLT (extensible Style sheet Language Transformation)

© Virtual University of Pakistan

Lecture 36

Topic-141 XPath

Topic-142 XLink & XPointer

Topic-143 JSON Introduction

Topic-144 JSON Syntax

Week 13

Lecture 37

Topic-145 JSON Case Study

Topic-146 HTML5 Introduction

Topic-147 HTML5 Elements

Topic-148 HTML5 Semantics – a

Topic-149 HTML5 Semantics - b

Lecture 38

Topic-150 HTML4 to HTML5 Migration

Topic-151 HTML5 Coding Conventions - a

Topic-152 HTML5 Coding Conventions – b

Topic-153 HTML5 Canvas

Topic-154 HTML5 Canvas Coordinates

Lecture 39

Topic-155 HTML5 Canvas Text

Topic-156 HTML5 Canvas Images

Topic-157 HTML5 SVG

Topic-158 HTML5 Multimedia

Topic-159 HTML5 Video

Week 14

Lecture 40

Topic-160 CSS3 Multi-Column Layout

Topic-161 HTML5 Plug-ins

Topic-162 HTML5 Geolocation

Topic-163 HTML5 Drag/Drop

Topic-164 HTML5 Local Storage

Lecture 41

Topic-165 HTML5 App Cache

Topic-166 HTML5 Web Workers

Topic-167 HTML5 SSE

Topic-168 HTML5 Case Study

Topic-169 CSS3 Introduction

Lecture 42

Topic-170 CSS3 Round Corners

Topic-171 CSS3 Broader Image

Topic-172 CSS3 Backgrounds – a

Topic-173 CSS3 Backgrounds - b

Topic-174 CSS3 Colors

Week 15

Lecture 43

Topic-175 CSS3 Gradients - a

Topic-176 CSS3 Gradients – b

Topic-177 CSS3 Shadow Effects

Topic-178 CSS3 Text Effects and Fonts

Topic-179 CSS3 Transforms

Lecture 44

Topic-180 CSS3 Transitions

Topic-182 Responsive Design Testing

Topic-183 CSS3-Parallax Background Implementation

Topic-184 Responsive Web Design (RWD) Introduction

Lecture 45

Topic-185 Responsive Web Design (RWD) The Viewport

Topic-186 Responsive Web Design (RWD) Grid-View

Topic-187 Responsive Web Design (RWD) Media Queries

Topic-188 Responsive Web Design (RWD) Images

Topic-189 Responsive Web Design (RWD) Videos

CS202 - Fundamentals of Front-End Development

1. Topic 1 - Introduction to Front End Development

Front-end development, also known as client-side development is the practice of producing HTML, CSS and JavaScript for a website or web application so that a user can see and interact with them directly.

Front End Languages: “Front end languages” lives in your browser.

- HTML
- CSS
- JavaScript

Front End Development

Front end web development is NOT design. The front-end developer stands between the designer on one end and the back-end developer on the other.

As a Front-end developer, you have to:

- Make sure the content is presented as desire.
- All user interactions are handled.

2. Topic 2 - URL: Uniform Resource Locator

URL stands for Uniform Resource Locator. It is the global address of documents and other resources on the World Wide Web. A URL is one type of Uniform Resource Identifier (URI).

Uniform Resource Identifier (URI) is the generic term for all types of names and addresses that refer to objects on the World Wide Web.

The term "Web address" is a synonym for a URL that uses the HTTP or HTTPS protocol. Here are few examples:

- https://mydomain.com/resource2.html
- ftp://mydomain.com/resource3.html

Parts of a URL

1. The first part of the URL is called a protocol identifier and it indicates what protocol to use.
2. The second part is called a resource location name and it specifies the IP address or the domain name where the resource is located.
3. The third part is optional; it’s the resource name, typically a file name on server.

Dynamic URL

A dynamic URL is the address of a dynamic Web page. Dynamic URLs often contain the characters like: ?, &, %, +, =, $, cgi-bin

There is a fourth optional part for dynamic URL. It contains the parameters sent to the dynamic page, in key=value pair.

?fname=Asim&job=Software%20Engineer

URL Encoding

URLs can only be sent over the Internet using the ASCII character-set.

Since URLs often contain characters outside the ASCII set, the URL has to be converted into a valid ASCII format.

URL encoding replaces unsafe ASCII characters with a "%" followed by two hexadecimal digits. URLs cannot contain spaces. URL encoding normally replaces a space with a plus (+) sign or with %20.

3. Topic 3 - HTTP Basic

HTTP

HTTP stands for Hypertext Transfer Protocol. It's an application-level protocol, works as a request- response protocol between a client and server over web.

HTTP is the foundation of data communication for the World Wide Web. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text.

HTTP Requests

Two commonly used methods for a request-response between a client and server are: GET and POST

1. GET - Requests data from a specified resource, has limited length and can cached by client.
2. POST - Submits data to be processed to a specified resource, has no data limit and never cached.

HTTPS

It is Hypertext Transfer Protocol Secure. Or HTTP over SSL It's a secure version of HTTP. All the communications between client and server are encrypted and authenticated.

4. Topic 4 - Web Server, Services, And Agents

Web Server

A Web server is a program that uses HTTP (Hypertext Transfer Protocol) to serve the files that form Web pages to users, in response to their requests, by HTTP clients. Dedicated computers and appliances may be referred to Web servers as well.

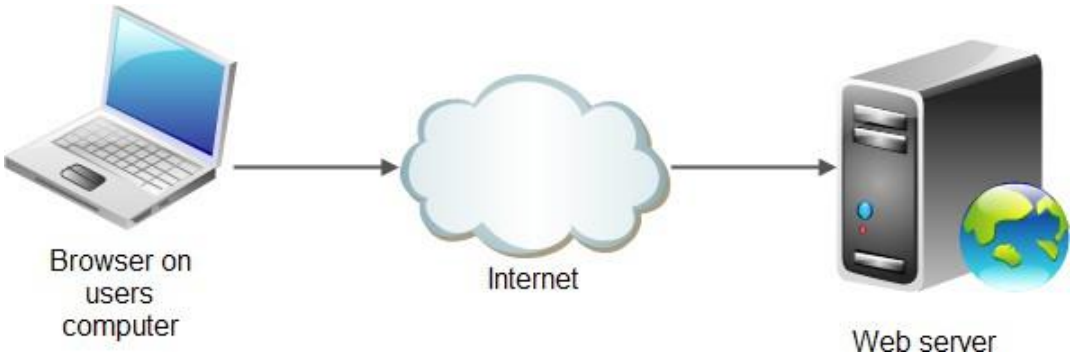


Figure 1: Web Servers

Application Server

An application server is a software framework that provides both facilities to create web applications and a server environment to run them. It acts as a set of components accessible to the developer through an API defined by the platform itself.

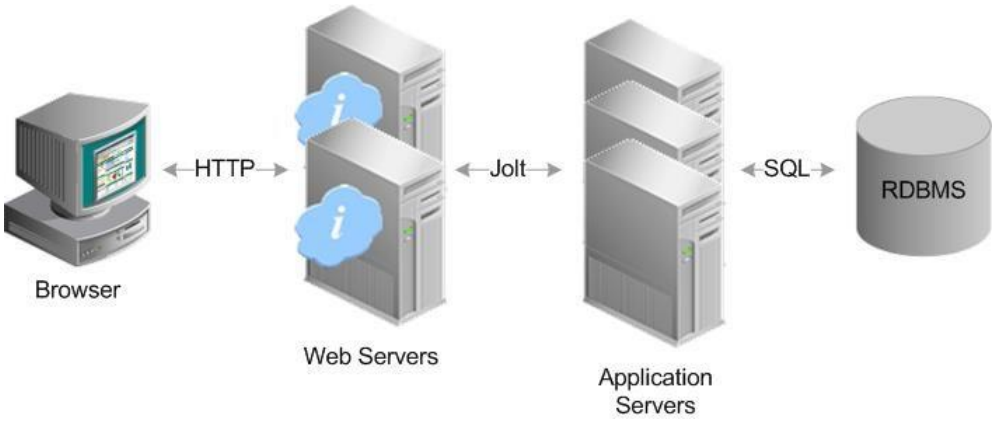


Figure 2: Application Servers

Web Services

Web services are client and server applications that communicate over the World Wide Web (WWW) using Hypertext Transfer Protocol (HTTP). Typically, Web Service resides on a Web Server and use XML to communicate messages.

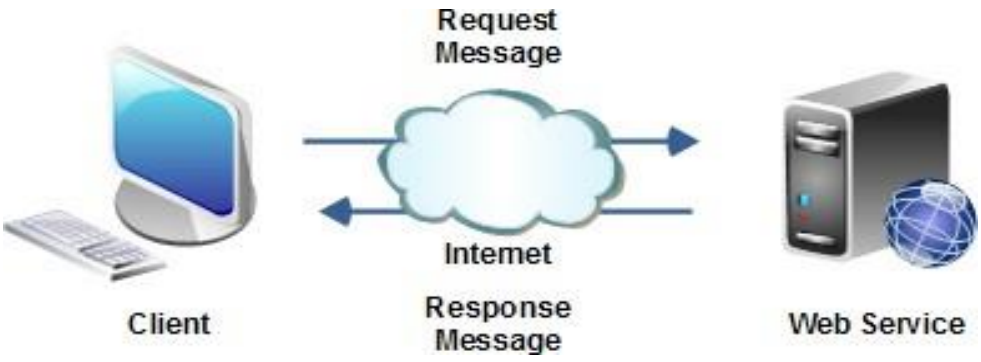


Figure 3: Web Servers

Web Agents / User Agents

These are the Software applications that are used by users to communicate to a Web server to retrieve resources or receive data. Web Browser is a typical example of Web/User Agent application.

5. Topic 5 - Domain, Hosting, FTP

Domain

A domain name is a unique name for your web site.

- Google.com
- Microsoft.com
- TechnologyWisdom.com
- Facebook.com

It is that part of the Uniform Resource Locator (URL) that tells a domain name server using the domain name system (DNS), where to forward a request for a Web page. The domain name is mapped to an IP address (which represents a physical point on the Internet). See here some examples of Domain Name.

- www. google.com.pk
- facebook.com
- validate.ecp.gov

Parts of Domain Name

1. **TLD: Top Level Domain**

www.google.**com**

facebook.**com**

validate.ecp.**gov**

2. **Second Level Domain**

Must be unique on the Internet and registered with one of the ICANN-accredited registrars for specific TLD.

www.**google**.com

facebook.com

validate.**ecp**.gov

3. **Third Level Domain**

www.google.com

facebook.com (no sub domain)

validate.ecp.gov

Hosting

An Internet hosting service is a service that runs Internet servers, allowing organizations and individuals to serve content on the Internet.

FTP

The File Transfer Protocol (FTP) is a standard network protocol used to transfer computer files from one host to another host over a TCP-based network, such as the Internet.

FTP is simple and secure

File is split into small data packets. Client sends a packet to destination after opening an authenticated connection. FTP server checks the packet, if data is correct, it sends request for next packet, till the file is completely received.

FTP Client

FTP Client is typically software that allows us to communicate with a server to transfer files using FTP protocol.

6. Topic 6 - HTML: Hyper Text Markup Language

Introduction

HTML stands for Hyper Text Markup Language. It is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology used to create web pages, as well as to create user interfaces for mobile and web applications. Web browsers can read HTML files and render them into visible or audible web pages.

7. Topic 7 – HTML Basics

HTML Documents

All HTML documents must start with a type declaration: <!DOCTYPE html>

See the following example:

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h1>Lorem ipsum</h1>
<p>Dummy Text. </p>
```

```
</body>

</html>
```

The HTML document itself begins with <html> and ends with </html>.

```
<!DOCTYPE html>

<html>

<head>

</head>

<body>

<h1>Lorem ipsum</h1>

<p>Dummy Text. </p>

</body>

</html>
```

The visible part of the HTML document is between <body> and</body>.

```
<!DOCTYPE html>

<html>

<head>

</head>

<body>

<h1>Lorem ipsum</h1>

<p>Dummy Text. </p>

</body>

</html>
```

The <head> tag, which is placed before the <body> tag, is optional and is used to define metadata elements such as <meta>.

```
<!DOCTYPE html>

<html>

<head>
```

```
</head>
<body>
<h1>Lorem ipsum</h1>
<p>Dummy Text. </p>
</body>
</html>
```

HTML Headings

HTML headings are defined with the <h1> to <h6> tags.

HTML Paragraphs

HTML paragraphs are defined with the <p> tag:

```
<p>My Paragraph. </p>
<p>Another Paragraph. </p>
```

HTML Images

HTML images are defined with the <img> tag. The source file (src), alternative text (alt), and size (width and height) are provided as attributes:

```
<img src=“image.jpg" alt=“Alternate text” width="104” height=“142“>
```

8. Topic 8 – HTML Elements and Attributes

HTML Elements

HTML elements are written with a start tag, with an end tag, with the content in between:

```
<tagname>content</tagname>
```

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

```
<p>Dummy Text</p>
```

Nested HTML Elements

HTML elements can be nested (elements can contain elements). All HTML documents consist of nested HTML elements. This example contains 4 HTML elements:

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).

```
<!DOCTYPE html>
```

```
<html>

<body>

<h1>My Heading</h1>

<p>My 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>Dummy Text</h1>

<p>Lorem ipsum dolor</p>

</body>
```

The <h1> element defines a heading. It has a start tag <h1> and an end tag </h1>. The element content is: My Heading.

```
<h1>My Heading</h1>
```

The element defines a paragraph. It has a start tag and an end tag . The element content is: My paragraph.

```
<p>My paragraph. </p>
```

Don't Forget the End Tag

Some HTML elements will display correctly, even if you forget the end tag:

```
<html>

<body>

<p>My paragraph

</body>

</html>
```

Empty HTML Element

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 you need to make your document readable by XML parsers, you should close all HTML elements.

Use Lowercase Tags

HTML tags are not case sensitive: <P> means the same as <p>. The HTML5 standard does not require lowercase tags, but W3C recommends lowercase in HTML4, and demands lowercase for stricter document types.

9. Topic 9 – HTML Elements and Attributes_2

HTML Attributes

- HTML elements can have attributes
- Attributes provide additional information about an element
- Attributes are always specified in the start tag
- Attributes come in name/value pairs like: name="value"

The lang Attribute

The document language can be declared in the <html> tag. The language is declared in the lang attribute. Declaring a language is important for accessibility applications (screen readers) and search engines:

```
<!DOCTYPE html>

<html lang="en-US">

<body>

<h1>My Heading</h1>

<p>My paragraph. </p>

</body>

</html>
```

The title Attribute

HTML paragraphs are defined with the <p> tag. In this example, the <p> element has a title attribute. The value of the attribute is “paragraph":

```
<p title="paragraph">
```

The href Attribute

HTML links are defined with the <a> tag. The link address is specified in the href attribute:

Example

```
<a href="">This is a link</a>
```

Size Attributes

HTML images are defined with the <img> tag.

The filename of the source (src), and the size of the image (width and height) are all provided as attributes:

Example

<img src=“google.jpg" width="104" height="142">

The alt Attribute

The alt attribute specifies an alternative text to be used, when an HTML element cannot be displayed. The value of the attribute can be read by "screen readers". This way, someone "listening" to the webpage, i.e. a blind person, can "hear" the element.

<img src=“google.jpg“ alt=“google” width="104“ height="142">

Always Use Lowercase Attributes

The HTML5 standard does not require lower case attribute names.

Always Quote Attribute Values

The HTML5 standard does not require quotes around attribute values. The href attribute, demonstrated above, can be written as:

<a href=>

Single or Double Quotes?

Double style quotes are the most common in HTML, but single style can also be used. In some situations, when the attribute value itself contains double quotes, it is necessary to use single quotes:

<p title=‘Smith "ShotGun" Jack'>

10. Topic 10 – Heading and Paragraph

HTML Headings

Headings are defined with the <h1> to <h6> tags. <h1> defines the most important heading. <h6> defines the least important heading.

Headings Are Important

Use HTML headings for headings only. Don't use headings to make text BIG or bold. Search engines use your headings to index the structure and content of your web pages. Users skim your pages by its headings.

It is important to use headings to show the document structure. In HTML, h1 headings should be main headings, followed by h2 headings, then the less important h3, and so on.

HTML Horizontal Rules

The <hr> tag creates a horizontal line in an HTML page. The hr element can be used to separate content. See the following example:

```
<p>First paragraph. </p>

<hr>

<p>Second paragraph. </p>

<hr>

<p>Third paragraph. </p>
```

The HTML <head> Element

The HTML <head> element has nothing to do with HTML headings. The HTML <head> element contains Meta data. Meta data are not displayed. The HTML <head> element is placed between the <html> tag and the <body> tag:

```
<!DOCTYPE html>

<html>

<head>

<title>My HTML</title>

<meta charset="UTF-8">

</head>

<body>
```

The HTML <title> Element

The HTML <title> element is Meta data. It defines the HTML document's title. The title will not be displayed in the document, but might be displayed in the browser tab.

```
<!DOCTYPE html>

<html>

<head>

<title>My HTML</title>

<meta charset="UTF-8">

</head>

<body>
```

The HTML <meta> Element

The HTML <meta> element is also meta data. It can be used to define the character set, and other information about the HTML document.

HTML Paragraphs

The HTML element defines a paragraph. See the examples below:

My Paragraph

Second Paragraph</p

HTML Display

You cannot be sure how HTML will be displayed. Large or small screens and resized windows will create different results. With HTML, you cannot change the output by adding extra spaces or extra lines in your HTML code.

The browser will remove extra spaces and extra lines when the page is displayed. Any number of spaces, and any number of new lines, counts as only one space.

HTML Line Breaks

The HTML <br> element defines a line break. Use <br> if you want a line break (a new line) without starting a new paragraph.

11. Topic 11 – HTML Styling

Every HTML element has a default style (background color is white and text color is black). Changing the default style of an HTML element, can be done with the style attribute. This example changes the default background color from white to light-grey:

<body style="background-color:lightgrey">

<h1>heading</h1>

paragraph.

</body>

The HTML Style Attribute

The HTML style attribute has the following syntax: style="property:value"

HTML Text Color

The color property defines the text color to be used for an HTML element:

<h1 style="color:blue">This is a heading</h1>

This is a paragraph

HTML Fonts

The font-family property defines the font to be used for an HTML element. See the examples below:

```
<h1 style="font-family:verdana">This is a heading</h1>

<p style="font-family:courier">This is a paragraph.</p>
```

HTML Text Size

The font-size property defines the text size to be used for an HTML element:

```
<h1 style="font-size:300%">This is a heading</h1>

<p style="font-size:160%">This is a paragraph. </p>
```

HTML Text Alignment

The text-align property defines the horizontal text alignment for an HTML element:

```
<h1 style="text-align:center">Centered Heading</h1>

<p>This is a paragraph. </p>
```

HTML Formatting Elements

- In the previous chapter, you learned about HTML styling, using the HTML style attribute.
- HTML also defines special elements, for defining text with a special meaning.
- HTML uses elements like <b> and <i> for formatting output, like bold or italic.
- Formatting elements were designed to display special types of text:
    - Bold text
    - Important text
    - Italic text
    - Emphasized text
    - Marked text
    - Small text
    - Deleted text
    - Inserted text
    - Subscripts
    - Superscripts

HTML Bold and Strong Formatting

The HTML <b> element defines bold text, without any extra importance.

```
<p>This text is normal. </p>

<p><b>This text is bold</b>. </p>
```

HTML Italic and Emphasized Formatting

The HTML <i> element defines italic text, without any extra importance.

 This text is normal.

 This text is italic.

HTML Italic and Emphasized Formatting

The HTML element defines emphasized text, with added semantic importance.

This text is normal.

This text is emphasized.

HTML Small Formatting

The HTML <small> element defines small text:

<h2>HTML <small>Small</small> Formatting</h2>

HTML Marked Formatting

The HTML <mark> element defines marked or highlighted text:

<h2>HTML <mark>Marked</mark> Formatting</h2>

HTML Deleted Formatting

The HTML <del> element defines deleted (removed) of text.

My favorite color is ~~blue~~ red.

HTML Subscript Formatting

The HTML <sub> element defines subscripted text.

This is _{subscripted} text.

HTML Superscript Formatting

The HTML <sub> element defines subscripted text.

This is ^{superscripted} text.

12. Topic 12 – HTML Quotations

HTML <q> for Short Quotations

The HTML <q> element defines a short quotation. Browsers usually insert quotation marks around the <q> element.

HTML <blockquote> for Long Quotations

The HTML <blockquote> element defines a quoted section. Browsers usually indent <blockquote> elements.

Here is a quote from WWF's website:

<blockquote cite="">

For 50 years, WWF has been protecting the future of nature.

</blockquote>

HTML <abbr> for Abbreviations

The HTML <abbr> element defines an abbreviation or an acronym. Marking abbreviations can give useful information to browsers, translation systems and search-engines.

HTML <address> for Contact Information

The HTML <address> element defines contact information (author/owner) of a document or article. The element is usually displayed in italic. Most browsers will add a line break before and after the element.

<address>

Written by Jon Doe.

Visit us at:

Example.com

Box 564, Disneyland

USA

</address>.

HTML <cite> for Work Title

The HTML <cite> element defines the title of a work. Browsers usually displays <cite> elements in italic.

<cite>The Scream</cite> by Edward Munch. Painted in 1893.

HTML <bdo> for Bi-Directional Override

The HTML <bdo> element defines bi-directional override. If your browser supports bdo, the text will be written from right to left.

<bdo dir="rtl">This text will be written from right to left</bdo>

13. Topic 13 – HTML Computer Code

HTML Computer Code Formatting

Normally, HTML uses variable letter size, and variable letter spacing. This is not wanted when displaying examples of computer code. The <kbd>, <samp>, and <code> elements all support fixed letter size and spacing.

HTML Keyboard Formatting

The HTML <kbd> element defines keyboard input:

```
<p>To open a file, select:</p>

<p><kbd>File | Open...</kbd></p>
```

HTML Sample Formatting

The HTML <samp> element defines a computer output:

```
<samp>

demo.example.com

login: Apr 12 09:10:17Linux 2.6.10grsec+gg3+e+fhs6b+nfs+gr0501

</samp>
```

HTML Code Formatting

The HTML <code> element defines programming code:

```
<code>

var person = { firstName:"John", lastName:"Doe", age:50, eyeColor:"blue" }

</code>
```

HTML Variable Formatting

The HTML <var> element defines a mathematical variable:

```
<p>Einstein wrote:</p>

<p><var>E = m c<sup>2</sup></var></p>
```

14. Topic 14 – HTML Comments

HTML Comment Tags

You can add comments to your HTML source by using the following syntax:

```
<!-- Write your comments here -->
```

Comments in HTML are not displayed by the browser, but they can help document your HTML. With comments you can place notifications and reminders in your HTML. Try these examples and see the differences:

```
<!—This is a comment-->

<p>This is a paragraph.</p>

<!-- Remember to add more information here -->
```

Comments are also great for debugging HTML, because you can comment out HTML lines of code, one at a time, to search for errors:

<!—Do not display

<img border="0" src="pic_mountain.jpg" alt="Mountain">

-->

Conditional Comments

You might stumble upon conditional comments in HTML:

<!--[if IE 8]>

....some HTML here....

<![endif]

Software Program Tags

HTML comments tags can also be generated by various HTML software programs.

For example <!--webbot bot--> tags wrapped inside HTML comments by FrontPage and Expression Web. As a rule, let these tags stay, to help support the software that created them.

15. Topic 15 – HTML Links

HTML Links-Hyperlinks

HTML links are hyperlinks. A hyperlink is a text or an image you can click on, and jump to another document.

HTML Links – Syntax

In HTML, links are defined with the <a> tag:

The href attribute specifies the destination address. The link text is the visible part.

Clicking on the link text, will send you to the specified address.

This is a Link to Google Local Links

The example above used an absolute URL (A full web address). A local link (link to the same web site) is specified with a relative URL (without https://www.....)

HTML Images

HTML Links - Colors and Icons

When you move the mouse cursor over a link, two things will normally happen:

- The mouse arrow will turn into a little hand.
- The color of the link element will change.

By default, links will appear as this in all browsers:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

Try the following examples:

<style>

a:link {color:#000000; background-color:transparent; text-decoration:none}

a:visited {color:#000000; background-color:transparent; text-decoration:none}

a:hover {color:#ff0000; background-color:transparent; text-decoration:underline}

a:active {color:#ff0000; background-color:transparent; text-decoration:underline}

</style>

HTML Links - The target Attribute

The target attribute specifies where to open the linked document. Let see an example that will open the linked document in a new browser window or in a new tab.

Visit Google!

HTML Links - Image as Link

In HTML, it is common to use images as links. As in the following example:

<img src="smiley.gif" alt="HTML tutorial" style="width:42px;height:42px;border:0">

HTML Links - The id Attribute

The id attribute can be used to create bookmarks inside HTML documents. Bookmarks are not displayed in any special way. They are invisible to the reader. Add an id attribute to any <a> element:

Useful Tips Section

Then create a link to the <a> element (tips):

View Tips

Create a link to the <a> element (tips) from another page:

Let See Tips

16. Topic 16 – HTML Images

- In HTML, images are defined with the <img> tag.
- The <img> tag is empty, it contains attributes only, and does not have a closing tag.
- The src attribute defines the url (web address) of the image

<img src="url" alt="some_text">

The alt Attribute

- The alt attribute specifies an alternate text for the image, if it cannot be displayed.
- The value of the alt attribute should describe the image in words.

HTML Screen Readers

- Screen readers are software programs that can read what is displayed on a screen.
- Used on the web, screen readers can "reproduce" HTML as text-to-speech, sound icons, or braille output.
- Screen readers are used by people who are blind, visually impaired, or learning disabled.

Image Size - Width and Height

- You can use the style attribute to specify the width and height of an image.
- The values are specified in pixels (use px after the value)

<img src="html5.gif" alt="HTML5 Icon" style= "width:128px; height:128px">

Width and Height or Style?

- Both, the width, the height, and the style attributes, are valid in the latest HTML5 standard.
- We suggest you use the style attribute. It prevents styles sheets from changing the default size of images

```
<html>
<head>
<style>
img { width:100%; }
</style>
</head>
<body>
<img src="html5.gif" alt="HTML5 Icon" style="width:128px;height:128px">
<img src="html5.gif" alt="HTML5 Icon" width="128" height="128">
</body>
</html>
```

- If not specified, the browser expects to find the image in the same folder as the web page.
- However, it is common on the web, to store images in a sub-folder, and refer to the folder in the image name:

Note: If a browser cannot find an image, it will display a broken link icon:

Images on Another Server

- Some web sites store their images on image servers.
- Actually, you can access images from any web address in the world:

```
<img src="">
```

Using an Image as a Link

It is common to use images as links:

```
<a href="url">

<img src="logo.png" alt="logo">

</a>
```

Image Maps

For an image, you can create an image map, with clickable areas.

Image Floating

You can let an image float to the left or right of a paragraph.

```
<p>

<img src="smiley.gif" alt="Smiley face" style="float:left;width:42px;height:42px">

A paragraph with an image.

</p>
```

17. Topic 17 – HTML Table

You can also create a table in an html document:

Number	First Name	Last Name	Points
1	Eve	Jackson	94
2	John	Doe	80

3	Adam	Johnson	67
4	Jill	Smith	50

Defining HTML Tables

- Tables are defined with the <table> tag.
- Tables are divided into table rows with the <tr> tag.
- Table rows are divided into table data with the <td> tag.
- Row can also be divided into table headings with the <th> tag.

```
<table style="width:100%">

<tr>

<td>Jill</td>

<td>Smith</td>

<td>50</td>

</tr>

<tr>

<td>Eve</td>

<td>Jackson</td>

<td>94</td>

</tr>

</table>
```

An HTML Table with a Border Attribute

- If you do not specify a border for the table, it will be displayed without borders.
- A border can be added using the border attribute.

```
<table border="1" style="width:100%">

<tr><td>Jill</td>

<td>Smith</td>

<td>50</td>
```

```
<td>Eve</td>
<td>Jackson</td>
<td>94</td>
</tr></table>
```

An HTML Table with Collapsed Borders

If you want the borders to collapse into one border, add CSS border-collapse

```
table, th, td {
border: 1px solid black; border-collapse: collapse;
}
th,td {
padding: 15px;
}
```

HTML Table Headings

Table headings are defined with the <th> tag.

By default, all major browsers display table headings as bold and centered:

```
<table style="width:100%">
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Points</th>
</tr><tr>
<td>Eve</td>
<td>Jackson</td>
<td>94</td>
</tr>
</table>
```

An HTML Table with Border Spacing

Border spacing specifies the space between the cells.

To set the border spacing for a table, use the CSS border-spacing property:

```
table {
border-spacing: 5px;
}
```

18. Topic 18 – HTML Tables Handling

Table Cells that Span Many Columns

To make a cell span more than one column, use the colspan attribute

```
<table style="width:100%">
<tr><th>Name</th>
<th colspan="2">Telephone</th> </tr>
<tr>
<td>Bill Gates</td>
<td>555 77 854</td>
<td>555 77 855</td>
</tr>
</table>
```

Table Cells that Span Many Rows

To make a cell span more than one row, use the rowspan attribute

```
<table style="width:100%">
<tr>
<th>Name:</th>
<td>Bill Gates</td>
</tr>
<tr>
<th rowspan="2">Telephone:</th>
<td>555 77 854</td>
```

```
</tr>
<tr>
<td>555 77 855</td>
</tr>
</table>
```

An HTML Table with a Caption

To add a caption to a table, use the <caption> tag:

```
<table style="width:100%">
<caption>Monthly savings</caption>
<tr>
<th>Month</th>
<th>Savings</th>
</tr>
<tr>
<td>January</td>
<td>$100</td>
</tr>
<tr>
<td>February</td>
<td>$50</td>
</tr></table>
```

Different Styles for Different Tables

- Most of the examples we show use a style attribute (width="100%") to define the width of each table.
- This makes it easy to define different widths for different tables.
- The styles in the <head> section, however, define a style for all tables in a page.
- To define a special style for a special table, add an id attribute to the table

```
<table id="t01">
<tr>
```

```
<th>Firstname</th>
<th>Lastname</th>
<th>Points</th>
</tr><tr>
<td>Eve</td>
<td>Jackson</td>
<td>94</td>
</tr>
</table>
```

A different style for this table:

```
table#t01 { width: 100%;
background-color: #f1f1c1;
}
```

Try more styles

```
table#t01 tr:nth-child(even) {
background-color: #eee;
}
table#t01 tr:nth-child(odd) {
background-color: #fff;
}
table#t01 th {
color: white;
background-color: black;
}
```

19. Topic 19 – HTML Lists

Unordered HTML Lists

- An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

- The list items will be marked with bullets (small black circles).

Coffee

Tea

Milk

Unordered HTML Lists - The Style Attribute

A style attribute can be added to an unordered list, to define the style of the marker:

Style	Description
list-style-type:disc	The list items will be marked with bullets (default)
list-style-type:circle	The list items will be marked with circles

list-style-type:square	The list items will be marked with squares
list-style-type:none	The list items will not be marked

Disc:

<ul style="list-style-type:disc">

Coffee

Tea

Milk

Circle:

<ul style="list-style-type:circle">

Coffee

Milk

Square:

<ul style="list-style-type:square">

Coffee

Tea

Milk

None:

<ul style="list-style-type:none">

Coffee

Tea

Milk

Ordered HTML Lists

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag. The list items will be marked with numbers.

Coffee

Tea

Milk

Numbers:

<ol type="1">

Coffee

Tea

Milk

```
</ol>
```

Upper Case:

```
<ol type="A">
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ol>
```

Lower Case:

```
<ol type="a">
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ol>
```

Roman Upper Case:

```
<ol type="I">
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ol>
```

Roman Lower Case:

```
<ol type="i">
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ol>
```

HTML Description Lists

- A description list, is a list of terms, with a description of each term.

- The <dl> tag defines a description list.
- The <dt> tag defines the term (name), and the <dd> tag defines the data (description).

Horizontal Lists

HTML lists can be styled in many different ways with CSS. One popular way, is to style a list to display horizontally:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul#menu li {
display:inline;
}
</style>
</head>
<body>
<h2>Horizontal List</h2>
<ul id="menu">
<li>HTML</li>
<li>CSS</li>
<li>JavaScript</li>
<li>PHP</li>
</ul>
</body>
</html>
```

20. Topic 20 – HTML Blocks

HTML Block Elements and Inline Elements

Most HTML elements are defined as block level elements or inline elements. Block level elements normally start (and end) with a new line, when displayed in a browser.

Examples: <h1>, <p>, <ul>, <table>

Inline elements are normally displayed without line breaks. Examples: <b>, <td>, <a>, <img>

The HTML Element

The HTML element is a block level element that can be used as a container for other HTML elements. The element has no special meaning. It has no required attributes, but style and class are common.

Because it is a block level element, the browser will display line breaks before and after it. When used together with CSS, the element can be used to style blocks of content.

The HTML Element

The HTML element is an inline element that can be used as a container for text. The element has no special meaning. It has no required attributes, but style and class are common.

Unlike , which is formatted with line breaks, the element does not have any automatic formatting. When used together with CSS, the element can be used to style parts of the text:

```
<h1>My <span style="color:red">Important</span> Heading</h1>
```

21. Topic 21 – HTML Layout

HTML layout is the basic structure of a web page. We use html elements to define the arrangement of the content on webpage.

HTML Layout using Elements

```
<body>

<div id="header">

<h1>City Gallery</h1>

</div>

<div id="nav"> London<br> Paris<br> Tokyo<br>

</div>

<div id="section">

<h1>London</h1>

<p> London is the capital city of England. </p>

<p> Standing on the River Thames, </p>

</div>

<div id="footer">

Copyright©W3Schools.com
```

```
</div>
</body>
```

CSS:

Try this style sheet with this html layout:

```
<style> #header {
background-color:black; color:white;
text-align:center; padding:5px;
}
#nav {
line-height:30px;
background-color:#eeeeee;
height:300px;
width:100px;
float:left;
padding:5px;
}
#section {
width:350px;
float:left;
padding:10px;
}
#footer {
background-color:black;
color:white;
clear:both;
text-align:center;
padding:5px;
```

```
}
</style>
```

HTML Layout Using Tables

Layout can be achieved using the <table> element, because table elements can be styled with CSS:

```
<body>
<table class="lamp">
<tr>
<th>
<img src="/images/lamp.jpg" alt="Note" style="height:32px;width:32px">
</th>
<td>
```

The table element was not designed to be a layout tool.

```
</td></tr>
</table></body>
```

CSS

Try this CSS code with for this layout:

```
<style>
table.lamp {
width:100%;
border:1px solid #d4d4d4;}
table.lamp th, td {
padding:10px;}
table.lamp th {
width:40px;}
</style>
```

22. Topic 22 – HTML Iframe

The <iframe> tag specifies an inline frame. An inline frame is used to embed another document within the current HTML document.

iframe Syntax

The syntax for adding an iframe is:

```
<iframe src="URL"></iframe>
```

iframe - Set Height and Width

Use the height and width attributes to specify the size. The attribute values are specified in pixels by default, but they can also be in percent (like "80%").

```
<iframe src="demo_iframe.htm" width="200" height="200"></iframe>
```

Iframe - Remove the Border

By default, an iframe has a black border around it. To remove the border, add the style attribute and use the CSS border property:

```
<iframe src="demo_iframe.htm" style="border:none"></iframe>
```

Use iframe as a Target for a Link

An iframe can be used as the target frame for a link. The target attribute of the link must refer to the name attribute of the iframe:

```
<iframe src="demo_iframe.htm" name="iframe_a"></iframe>

<p><a href="" target="iframe_a">google

</a></p>
```

23. Topic 23 – HTML Forms

The <form> Element

HTML forms are used to collect user input. The <form> element defines an HTML form:

```
<form>

Form elements

</form>
```

Form Elements

HTML forms contain form elements. Form elements are different types of input elements, checkboxes, radio buttons, submit buttons, and more.

The <input> Element

The <input> element is the most important form element. The <input> element has many variations, depending on the type attribute. Here are the types used in this chapter:

- **Text Input**

  <input type="text"> defines a one-line input field for text input:

```
<form>
First name:<br>
<input type="text" name="firstname"><br>
Last name:<br>
<input type="text" name="lastname">
</form>
```

- **Radio Button Input**

  <input type="radio"> defines a radio button. Radio buttons let a user select ONE of a limited number of choices:

```
<form>
<input type="radio" name="sex" value="male" checked>Male
<br>
<input type="radio" name="sex" value="female">Female
</form>
```

- **The Submit Button**

  <input type="submit"> defines a button for submitting a form to a form-handler. The form- handler is typically a server page with a script for processing input data. The form-handler is specified in the form's action attribute:

```
<form action="action_page.php">
First name:<br>
<input type="text" name="firstname" value="Mickey">
<br>
Last name:<br>
<input type="text" name="lastname" value="Mouse">
<br><br>
<input type="submit" value="Submit">
```

```
</form>
```

- **The Action Attribute**

  The action attribute defines the action to be performed when the form is submitted. The common way to submit a form to a server, is by using a submit button. Normally, the form is submitted to a web page on a web server.

  In the example above, a server-side script is specified to handle the submitted form:

  <form action="action_page.php">

  If the action attribute is omitted, the action is set to the current page.

  When to Use GET?

  You can use GET (the default method):

  If the form submission is passive (like a search engine query), and without sensitive information, use this method.

  When you use GET, the form data will be visible in the page address: action_page.php?firstname=Mickey&lastname=Mouse

  When to Use POST?

  You should use POST:

  If the form is updating data, or includes sensitive information (password), use this method. The POST method offers better security because the submitted data is not visible in the page address.

  The Name Attribute

  To be submitted correctly, each input field must have a name attribute. This example will only submit the "Last name" input field:

```
<form action="action_page.php">

First name:<br>

<input type="text" value="Mickey">

<br>

Last name:<br>

<input type="text" name="lastname" value="Mouse">

<br><br>

<input type="submit" value="Submit">

</form>
```

  Grouping Form Data with <fieldset>

The <fieldset> element groups related data in a form. The <legend> element defines a caption for the <fieldset> element:

```
<form action="action_page.php">

<fieldset>

<legend>Personal information:</legend> First name:<br>

<input type="text" name="firstname“ value="Mickey"> Last name:<br>

<input type="text" name="lastname" value="Mouse“><br>

<input type="submit" value="Submit"></fieldset>

</form>
```

24. Topic 24 – HTML Colors

Colors are displayed combining RED, GREEN, and BLUE light. Colors in HTML can be specified as:

1. Hexadecimal colors
2. RGB colors
3. Color names

1. **Hexadecimal Colors:** Supported in all major browsers

A hexadecimal color is specified with: #RRGGBB, hexadecimal integers. The Values are between 00 and FF.

- #0000FF (blue)
- #FF0000 (red)

2. **RGB Colors**

These are supported in all major browsers. An RGB color value is specified with: rgb(red, green, blue). The values are between 0 and 255.

- rgb(0,0,255) // blue
- rgb(255,0,0) // red

3. **Color Names**

All major browsers also support 140 standard color names. You can see more example online.

HTML Colors

The combination of red, green and blue values from 0 to 255 gives a total of more than 16 million different colors: (256 x 256 x 256)

Most modern monitors are capable of displaying at least 16384 different colors.

25. Topic 25 – HTML Head

The HTML <head> Element

The <head> element is a container for Meta data (data about data). HTML Meta data is data about the HTML document. Metadata is not displayed. Meta data typically define document title, styles, links, scripts, and other Meta information. The following tags describes Meta data: <title>,

<style>, <meta>, <link>, <script>, and <base>.

Omitting <html> and <body>

In the HTML5 standard, the <html> tag, the <body> tag, and the <head> tag can be omitted. The following code will validate as HTML5:

```
<!DOCTYPE html>

<head>

<title>Page Title </title>

</head>

<h1>This is a heading. </h1>

<p>This is a paragraph. </p>
```

The <html> element is the document root. It is the recommended place for specifying the page language:

```
<!DOCTYPE html>

<html lang="en-US">
```

Declaring a language is important for accessibility applications (screen readers) and search engines. Omitting <html> and <body> can crash badly written DOM and XML software. Finally, omitting <body> can produce errors in older browsers (IE9).

Omitting <head>

In the HTML5 standard, the <head> tag can also be omitted. By default, browsers will add all elements before <body>, to a default <head> element. You can reduce the complexity of HTML, by omitting the <head> tag.

```
<!DOCTYPE html>

<html>

<title>Page Title</title>

<body>

<h1>This is        a heading</h1>

<p>This is a       paragraph. </p>

</body>
```

The HTML <title> Element

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

The <title> element

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

The HTML <style> Element

The <style> element is used to define style information for an HTML document. Inside the <style> element you specify how HTML elements should render in a browser.

```
<style>

body {background-color:yellow;} p {color:blue;}

</style>
```

The HTML <link> Element

The <link> element defines the page relationship to an external resource. The <link> element is most often used to link to style sheets:

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

<meta> Element

The <meta> element is used to specify page description, keywords, author, and other metadata. Meta data is used by browsers (how to display content), by search engines (keywords), and other web services.

Define keywords for search engines:

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

Define a description of your web page:

```
<meta name="description" content="Free Web tutorials on HTML and CSS">
```

Define the character set

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

Define the author of a page:

```
<meta name="author" content="Hege Refsnes">
```

Refresh document every 30 seconds:

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

The HTML <script> Element

The <script> element is used to define client-side JavaScripts. The script that writes Hello JavaScript! Into an HTML element with the id="demo"), looks like:

<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:

<base href="" target="_blank">

26. Topic 26 – HTML Entities and Symbols

Some characters are reserved in HTML. If you use the less than (<) or greater than (>) signs in your text, the browser might mix them with tags. Character entities are used to display reserved characters in HTML. A character entity looks like this:

&entity_name;

OR

&#entity_number;

To display a less than (<) sign we must write:

< or <

Non-Breaking Space

A common character entity used in HTML is the non-breaking space ( ). Remember that browsers will always truncate spaces in HTML pages. If you write 10 spaces in your text, the browser will remove 9 of them. To add real spaces to your text, you can use the   character entity.

Combining Diacritical Marks

A diacritical mark is a "glyph" added to a letter. Some diacritical marks, like grave (`) and acute (´) are called accents. Diacritical marks can appear both above and below a letter, inside a letter, and between two letters.

Diacritical marks can be used in combination with alphanumeric characters, to produce a character that is not present in the character set (encoding) used in the page.

HTML Symbol Entities

HTML entities were described in the previous chapter. Many mathematical, technical, and currency symbols, are not present on a normal keyboard. To add these symbols to an HTML page, you can use an HTML entity name.

If no entity name exists, you can use an entity number; a decimal (or hexadecimal) reference.

Example

I will display €

I will display €

I will display €

Result

I will display €

I will display €

I will display €

Some Mathematical Symbols Supported by HTML

Char	Number	Entity	Description
∀	∀	∀	FOR ALL
∂	∂	∂	PARTIAL DIFFERENTIAL
∃	∃	∃	THERE EXISTS
∅	∅	∅	EMPTY SETS
∇	∇	∇	NABLA
∈	∈	∈	ELEMENT OF
∉	∉	∉	NOT AN ELEMENT OF
∋	∋	∋	CONTAINS AS MEMBER
∏	∏	∏	N-ARY PRODUCT

∑	∑	∑	N-ARY SUMMATION

Some Greek Letters Supported by HTML

Char	Number	Entity	Description
A	Α	Α	GREEK CAPITAL LETTER ALPHA
B	Β	Β	GREEK CAPITAL LETTER BETA
Γ	Γ	Γ	GREEK CAPITAL LETTER GAMMA
Δ	Δ	Δ	GREEK CAPITAL LETTER DELTA
E	Ε	Ε	GREEK CAPITAL LETTER EPSILON
Z	Ζ	Ζ	GREEK CAPITAL LETTER ZETA
Char	Number	Entity	Description
A	Α	Α	GREEK CAPITAL LETTER ALPHA
B	Β	Β	GREEK CAPITAL LETTER BETA
Γ	Γ	Γ	GREEK CAPITAL LETTER GAMMA

27. Topic 27 – HTML Encoding (Character Set)

What is Character Encoding?

ASCII was the first character encoding standard (also called character set). It defines 127 different alphanumeric characters that could be used on the internet. ASCII supported numbers (0-9), English letters (A-Z), and some special characters like! $ + - ( ) @ < > .

ANSI (Windows-1252) was the original Windows character set. It supported 256 different character codes. SO-8859-1 was the default character set for HTML 4. It also supported 256 different character codes. Because ANSI and ISO was limited, the default character encoding was changed to UTF-8 in HTML5.

UTF-8 (Unicode) covers almost all of the characters and symbols in the world.

The HTML charset Attribute

To display an HTML page correctly, a web browser must know the character set used in the page. This is specified in the <meta> tag:

For HTML4:

```
<meta http-equiv="Content-Type" content="text/html;charset=ISO-8859-1">
```

For HTML5:

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

The ASCII Character Set

- ASCII uses the values from 0 to 31 (and 127) for control characters.
- ASCII uses the values from 32 to 126 for letters, digits, and symbols.
- ASCII does not use the values from 128 to 255.

The ANSI Character Set (Windows-1252)

- ANSI is identical to ASCII for the values from 0 to 127.
- ANSI has a proprietary set of characters for the values from 128 to 159.
- ANSI is identical to UTF-8 for the values from 160 to 255.

The ISO-8859-1 Character Set

- 8859-1 is identical to ASCII for the values from 0 to 127.
- 8859-1 does not use the values from 128 to 159.
- 8859-1 is identical to UTF-8 for the values from 160 to 255.

The UTF-8 Character Set

- UTF-8 is identical to ASCII for the values from 0 to 127.
- UTF-8 does not use the values from 128 to 159.
- UTF-8 is identical to both ANSI and 8859-1 for the values from 160 to 255.
- UTF-8 continues from the value 256 with more than 10.000 different characters.

28. Topic 28 – HTML URL Encoding

HTML Uniform Resource Locators

A URL is another word for a web address. A URL can be composed of words (google.com), or an Internet Protocol (IP) address (192.68.20.50). Most people enter the name when surfing, because names are easier to remember than numbers.

URL

Web browsers request pages from web servers by using a URL. When you click on a link in an HTML page, an underlying <a> tag points to an address on the web. A Uniform Resource Locator (URL) is used to address a document (or other data) on the web.

A web address, like http://www.htmllectures.com/html/default.asp

Follows these syntax rules: scheme://host.domain:port/path/filename

- **scheme** - defines the type of Internet service (most common is http)
- **host** - defines the domain host (default host for http is www)
- **domain** - defines the Internet domain name (google.com)
- **port** - defines the port number at the host (default for http is 80)
- **path** - defines a path at the server (If omitted: the root directory of the site)
- **filename** - defines the name of a document or resource

Common URL Schemes

cheme	Short for	Used for
http	HyperText Transfer Protocol	Common web pages. Not encrypted
https	Secure HyperText Transfer Protocol	Secure web pages. Encrypted
ftp	File Transfer Protocol	Downloading or uploading files
file		A file on your computer

URL Encoding

URLs can only be sent over the Internet using the ASCII character-set. Since URLs often contain characters outside the ASCII set, the URL has to be converted into ASCII. URL encoding converts characters into a format that can be transmitted over the Internet.

URL encoding replaces non-ASCII characters with a "%" followed by hexadecimal digits. URLs cannot contain spaces. URL encoding normally replaces a space with a plus (+) sign, or %20.

29. Topic 29 – HTML and XHTML

What Is XHTML?

- XHTML stands for EXtensible HyperText Markup Language.
- XHTML is almost identical to HTML.
- XHTML is stricter than HTML.
- XHTML is HTML defined as an XML application.
- XHTML is supported by all major browsers.
- **Why XHTML?**
- Many pages on the internet contain "bad" HTML.
- This HTML code works fine in most browsers (even if it does not follow the HTML rules):

See the following example:

```
<html>

<head>

<title>This is bad HTML</title>

<body>

<h1>Bad HTML

<p>This is a paragraph

</body>
```

- Today's market consists of different browser technologies. Some browsers run on computers, and some browsers run on mobile phones or other small devices. Smaller devices often lack the resources or power to interpret "bad" markup.
- If you want to study XML, please read our XML tutorial.
- By combining the strengths of HTML and XML, XHTML was developed.
- XHTML is HTML redesigned as XML.
- XML is a markup language where documents must be marked up correctly (be "well- formed").
- By combining the strengths of HTML and XML, XHTML was developed.
- XHTML is HTML redesigned as XML.

The Most Important Differences from HTML:

Document Structure

- XHTML DOCTYPE is mandatory
- The xmlns attribute in <html> is mandatory
- <html>, <head>, <title>, and <body> are mandatory

XHTML Element

- XHTML elements must be properly nested
- XHTML elements must always be closed
- XHTML elements must be in lowercase
- XHTML documents must have one root element
- Attribute names must be in lower case
- Attribute values must be quoted
- Attribute minimization is forbidden

<!DOCTYPE> Is Mandatory

- An XHTML document must have an XHTML DOCTYPE declaration.
- A complete list of all the XHTML Doctypes is found in our HTML Tags Reference.
- The <html>, <head>, <title>, and <body> elements must also be present, and the xmlns attribute in <html> must specify the xml namespace for the document.

In HTML, some elements can be improperly nested within each other, like this:

```
<b><i>This text is bold and italic</b></i>
```

In XHTML, all elements must be properly nested within each other, like this:

```
<b><i>This text is bold and italic</i></b>
```

XHTML Elements Must Always Be Closed

This is wrong:

```
<p>hi, everyone

<p>How are you?
```

This is correct:

```
<p>hi, everyone</p>

<p>How are you? </p>
```

Empty Elements Must Also Be Close

Wrong:

```
A break: <br>

A horizontal rule: <hr>

An image: <img src="happy.gif" alt="Happy face">
```

Correct:

A break:

A horizontal rule: <hr />

An image:

XHTML Elements Must Be in Lower Case

Wrong:

<BODY>

<P>Hi, Everyone</P>

</BODY>

Correct:

<body>

Hi, Everyone

</body>

XHTML Attribute Names Must Be in Lower Case

Wrong:

<table WIDTH="100%">

Correct:

<table width="100%">

Attribute Values Must Be Quoted

Wrong:

<table width=100%>

Correct:

<table width="100%">

How to Convert from HTML to XHTML

- Add an XHTML <!DOCTYPE> to the first line of every page
- Add an xmlns attribute to the html element of every page
- Change all element names to lowercase
- Close all empty elements
- Change all attribute names to lowercase
- Quote all attribute values

Introduction to CSS

30. Topic 30 – Introduction to CSS

What is CSS?

CSS stands for Cascading Style Sheets. CSS defines how HTML elements are to be displayed. A style sheet language that is used for describing the presentation of a document written in a markup language.

- CSS defines the ways in which HTML elements are to be displayed
- CSS styles were added to HTML 4.0 to solve a problem
- CSS is used in development of sites where every element of site needed font and style
- CSS styles are normally stored in an external file with .css extension, which is included in the main HTML file.

CSS Solved a Big Problem.

- HTML was never intended to contain tags for formatting a document.
- HTML was intended to define the content of a document, like:
  - <h1> Hi, Everyone </h1>
  - How are you.
- When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers.
- Development of large websites. Where fonts and color information were added to every single page, became a long and expensive process.
- To solve this problem the World Wide Web Consortium (w3C) created CSS.
- In HTML 4.0, all formatting could (and should!) be removed from the HTML document, and stored in a separate CSS file.

You can change the style of entire site by just editing the CSS file.

31. Topic 31 –CSS Syntax

CSS Syntax consists of a selector and a declaration block.

Here is a style which is defined for h1 (first heading in HTML)

```
h1 {

color: blue;

font-size: 12px;

}
```

h1 is a selector, HTML element you want to style values inside the curly braces are declarations.

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

Color: blue; and Font-size:12px; are two declarations for Selector

Color and font-size are two property names in declarations whereas Blue and 12px are two values in declarations. Property name and values are separated by colon.

CSS Comments are used to explain the CSS codes which you wrote to define the colors and style for html elements.

CSS comments starts with /* and ends with */

32. Topic 32 – CSS Selectors

CSS selectors are patterns used to select the element(s) you want to style. Some important CSS selectors are given here:

Selector	Example	Example description	CSS
.class	.intro	Selects all elements with class="intro"	1
#id	#firstname	Selects the element with id="firstname"	1
element	p	Selects all <p> elements	1
element>element	div > p	Selects all <p> elements where the parent is a element	2
element1~element2	p ~ ul	Selects every element that are preceded by a <p> element	3
[attribute]	[target]	Selects all elements with a target attribute	2
[attribute=value]	[target=_blank]	Selects all elements with target="_blank"	2
[attribute|=value]	[lang|=en]	Selects all elements with a lang attribute value starting with "en"	2

Note: in the table below, the "CSS" column indicates in which CSS version the property is defined (CSS1, CSS2, or CSS3).

33. Topic 33 – CSS Insertion

CSS can be used by including external style sheet, internal style sheet or defining inline styles.

1. External Style Sheet

Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the head section.

Example:

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

2. Internal Style Sheet

An internal style sheet may be used if one single page has a unique style. Internal styles are defined within the <style> element, inside the head section of an HTML page.

Example:

```
<head>
<style>
Body {background-color:linen;}
H1 {color:maroon; margin-left:40px;}
</style>
</head>
```

3. Inline CSS

An inline style may be used to apply a unique style for a single element. To use inline styles, add the style attribute to the relevant tag. The style attribute can contain any CSS property. See an example below:

```
<h1 style="color:blue;margin-left:30px;">This is a heading.</h1>
```

This example shows how to change the color and the left margin of a <h1> element.

34. Topic 34 – CSS Background

CSS background property is used to give a background style to HTML elements. Some of major background properties are given below:

Background Color

The background-color property specifies the background color of an element.

Examples:

```
h1 {background-color: #6495ed;}
p {background-color: #e0ffff;}
```

div {background-color: #b0c4de;}

In this example, the <h1>, <p>, and elements have different background colors

Background Image

The background-image property specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element. The background image for a page can be set like this:

body {background-image: url("paper.gif");}

Background Image - Repeat Horizontally or Vertically

The background-image property repeats an image both horizontally and vertically.

Background Image - Set position and no-repeat

Showing the image only once is specified by the background-repeat property.

Background Image - Set position and no-repeat

Let say we want to change the position of the image, so that it does not disturb the text or other elements too much. The position of the image can be specified by the background-position property.

In order to shorten the code, the shorthand property for background is simply "background"

35. Topic 35 – CSS Text and Fonts

The CSS color property is used to set the color of the text. With CSS, a color is most often specified by:

- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"
- a color name - like "red"

The default color for a page is defined in the body selector.

Text Transformation

In CSS, The text-transform property is used to specify uppercase and lowercase letters in a text. It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word. Example is give below:

- p.uppercase {text-transform: uppercase;}
- p.lowercase {text-transform: lowercase;}
- p.capitalize {text-transform: capitalize;}

Text Indentation

In CSS, the text-indent property is used to specify the indentation of the first line of a text or paragraph.

p {text-indent: 50px;}

CSS Font

CSS font properties define the font family, boldness, size, and the style of a text.

CSS Font Families

In CSS, there are two types of font family names:

- **Generic family** – a group of font families with a similar look (like "Serif" or "Monospace")
- **Font family** – a specific font family (like "Times New Roman" or "Arial")

The table below can help you to understand font families more clearly:

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

Font Family

The font family of a text is set with the font-family property. The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font. Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Note: If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman ". See the example here:

More than one font family is specified in a comma-separated list:

```
p {
font-family: "Times New Roman", Times, serif;
}
```

36. Topic 36 – CSS Text and Fonts_2

In CSS, the font-style property is mostly used to specify italic text. This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported).

See the examples below for font styles:

```
p.normal { font-style: normal; }

p.italic { font-style: italic; }

p.oblique { font-style: oblique; }
```

Font Size

In CSS, the font-size property is used to sets the size of the text.

Explanation: Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs. The font-size value can be an absolute or relative size.

❖ **Absolute size:**
- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

❖ **Relative size:**
- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.

Set Font Size with Pixels: Setting the text size with pixels gives you full control over the text size. See the example below:

```
h1 {font-size: 40px;}

h2 {font-size: 30px;}

p {font-size: 14px;}
```

Set Font Size with Em

In CSS, to allow users to resize the text (in the browser menu), many developers use em instead of pixels. The em size unit is recommended by the W3C.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px. The size can be calculated from pixels to em using this formula: pixels/16=em

Set Font Size with Pixels

Setting the text size with pixels gives you full control over the text size

Use a Combination of Percent and Em

The solution that works in all browsers, is to set a default font-size in percent for the <body> element. See the example below:

```
body {font-size: 100%;}

h1 {font-size: 2.5em;}

h2 {font-size: 1.875em;}

p {font-size: 0.875em;}
```

37. Topic 37 – CSS Links

Styling Links

Links can be styled with any CSS property. Below example is used to give a specific color to all the link text in file.

```
a {color: #FF0000;}
```

Furthermore, In addition, links can be styled differently depending on what state they are in. The four links states are:

The four links states are:

- **a:link** – a normal, unvisited link
- **a:visited** – a link the user has visited
- **a:hover** – a link when the user mouses over it
- **a:active** – a link the moment it is clicked

Furthermore, there are also some of the other common ways to style links.

Text Decoration

In CSS, this property is mostly used to remove underlines from links. See the example below:

- **a:link** {text-decoration: none;} – to remove the link decoration
- **a:visited** {text-decoration: none;} – to remove the visited link's decoration
- **a:hover** {text-decoration: underline;} – underline the link text while mouse over
- **a:active** {text-decoration: underline;} – underline the active link

Background Color

In CSS, the background-color property specifies the background color for links. See example below

```
a:link {background-color: #B2FF99;}
```

38. Topic 38 – CSS Lists

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker

Lists in HTML

In HTML, there are two types of lists:

- unordered lists (<ul>) - the list items are marked with bullets
- ordered lists (<ol>) - the list items are marked with numbers or letters

In CSS, the type of list item marker is specified with the list-style-type property. Some examples are given below:

```
ul.a {list-style-type: circle;}

ul.b {list-style-type: square;}

ol.c { list-style-type: upper-roman;}

ol.d {list-style-type: lower-alpha;}
```

Output:

Example of unordered lists:

- Coffee
- Tea
- Coca Cola

- Coffee
- Tea
- Coca Cola

Example of ordered lists:

I. Coffee
II. Tea
III. Coca Cola

a. Coffee
b. Tea
c. Coca Cola

An Image as the List Item Marker

To specify an image as the list item marker, use the list-style-image property.

```
ul {list-style-image: url('sqpurple.gif');}
```

List – Shorthand Propert

The list-style property is a shorthand property. It is used to set all the list properties in one declaration.

```
ul {list-style: square inside url('sqpurple.gif');}
```

39. Topic 39 – CSS Tables

The look of an HTML table can be greatly improved with CSS.

Table Borders

To specify table borders in CSS, use the border property. The example below specifies a black border for <table>, <th>, and <td> elements

```
table, th, td {border: 1px solid black;}
```

CSS Tables

Notice that the table in the example above has double borders. This is because both the table and the <th>/<td> elements have separate borders. To display a single border for the table, use the border-collapse property. The border-collapse property sets whether the table borders are collapsed into a single border or separated:

```
table {border-collapse: collapse;}

table, th, td {border: 1px solid black;}
```

Firstname	Lastname
Peter	Griffin
Lois	Griffin

Note: If a !DOCTYPE is not specified, the border-collapse property can produce unexpected results in IE8 and earlier versions.

Table Width and Height

Width and height of a table is defined by the width and height properties. The example below sets the width of the table to 100%, and the height of the <th> element to 50px:

```
table{ width:100%;}

th { height: 50px;}
```

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300
Cleveland	Brown	$250

Horizontal Text Alignment

The text-align property sets the horizontal alignment, like left, right, or center. By default, the text in <th> elements are center-aligned and the text in <td> elements are left-aligned. The following example left-aligns the text in <th> elements:

```
th {text-align: left;}
```

Vertical Text Alignment

The vertical-align property sets the vertical alignment, like top, bottom, or middle. By default, the vertical alignment of text in a table is middle (for both <th> and <td> elements). Example given below sets the vertical text alignment to bottom for <td> elements:

```
td { height: 50px; vertical-align: bottom;}
```

Table Padding

To control the space between the border and content in a table, use the padding property on <td> and <th> elements.

```
td {padding: 15px;}
```

Table Color

We can specify using CSS the color of the borders, and the text and background color of <th> elements.

```
table, td, th {border: 1px solid green;}

th {background-color: green; color: white;}
```

40. Topic 40 – CSS Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout. The CSS box model is essentially a box that wraps around HTML elements and it consists of: margins, borders, padding, and the actual content.

The box model allows us to add a border around elements, and to define space between elements. The image below illustrates the box model:



Explanation the different parts of the box model:

- Content - The content of the box, where text and images appear
- Padding - Clears an area around the content. The padding is transparent
- Border - A border that goes around the padding and content
- Margin - Clears an area outside the border. The margin is transparent.

Check here the CSS code for Box model:

```
div {width: 300px; padding: 25px; border: 25px solid navy; margin: 25px;}
```

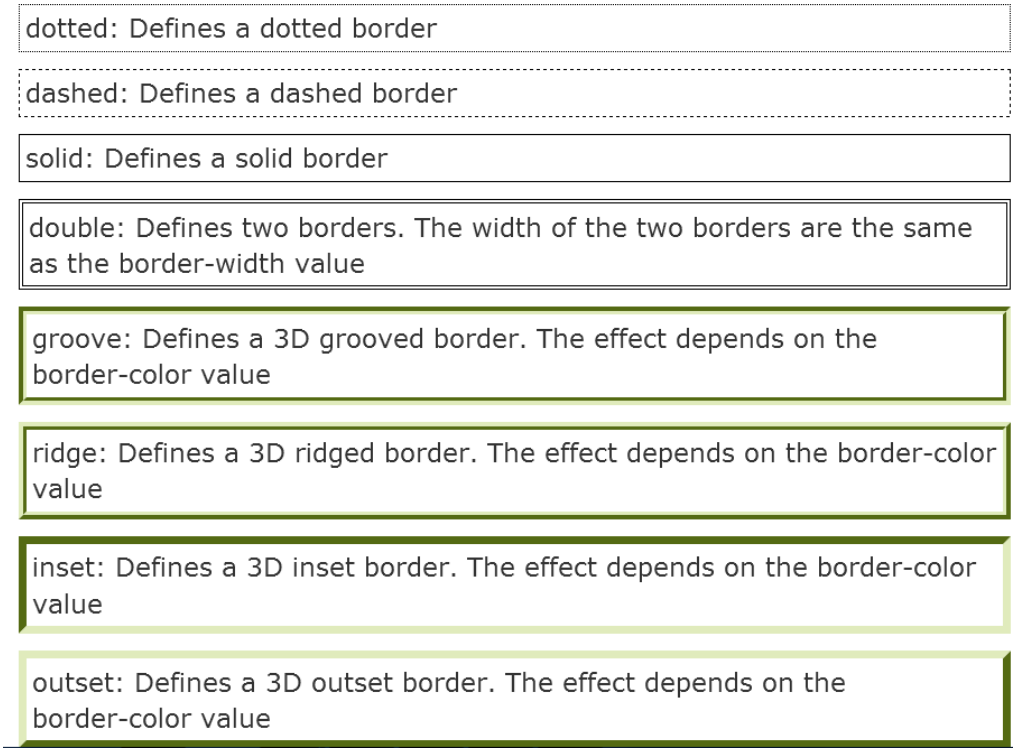
Width and Height of an Element

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works. Let's style a element to have a total width of 350px.

```
div {width: 320px; padding: 10px; border: 5px solid gray; margin: 0; }
```

41. Topic 41 – CSS Border and Outline

The border-style property specifies what kind of border to display. You can check below the border-style values with demonstration:



Border Width

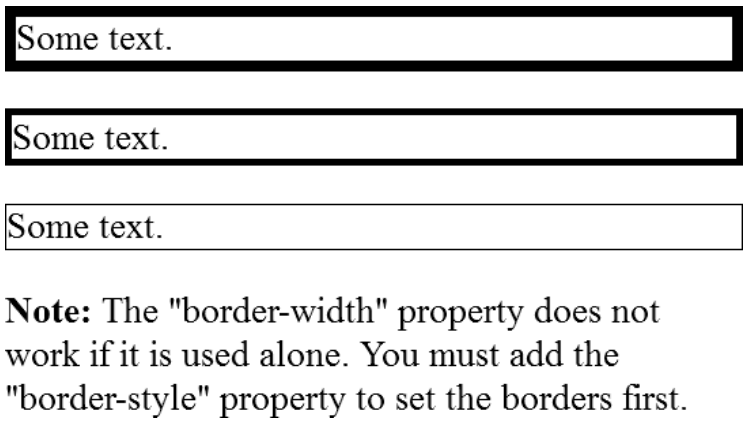
The border-width property is used to set the width of the border. The width is set in pixels, or by using one of the three pre-defined values: thin, medium, or thick.

Note: The "border-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.

```
p.one {border-style:solid; border-width:5px;}

p.two { border-style: solid; border-width: medium;}
```

Output:



Border Color

The border-color property is used to set the color of the border. The color can be set by:

- Name - specify a color name, like "red"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- Hex - specify a hex value, like "#ff0000"

Furthermore, you can also set the border color to "transparent". If the border color is not set it is inherited from the color property of the element.

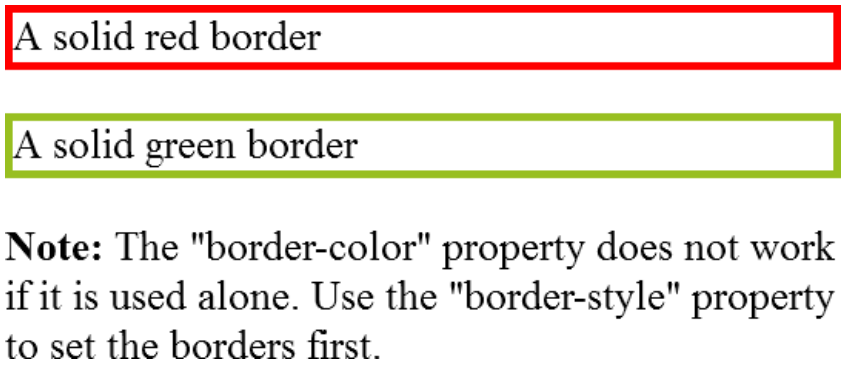
Note: The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.

See an example below:

```
p.one{border-style: solid; border-color: red; }

p.two {border-style: solid; border-color: #98bf21; }
```

Output:



Border - Individual sides

In CSS, it is possible to specify different borders for different sides. In an example below, different border style has been specified for the different sides of the text in a paragraph.

```
p {

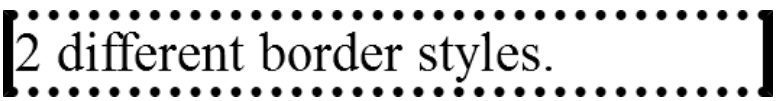
border-top-style: dotted;

border-right-style: solid;

border-bottom-style: dotted;

border-left- style: solid;}
```

Output:



Border - Shorthand property

As you have seen, there are many properties to consider when dealing with borders. To shorten the code, it is also possible to specify all the individual border properties in one property. This is called a shorthand property.

The border property is shorthand for the following individual border properties:

- border-width
- border-style (required)
- border-color

CSS Outlines

An outline is a line that is drawn around elements (outside the borders) to make the element "stand out". In CSS, the outline properties specify the style, color, and width of an outline. However, the outline property is different from the border property. We can define the outline property as stated below:

```
p {border: 1px solid red; outline: green dotted thick; }
```

You can also try the code for practice purposes:

```
p {border: 1px solid red;}

p.dotted {outline-style: dotted;}

p.dashed {outline-style: dashed;}

p.solid {outline-style: solid;}

p.double {outline-style: double;}

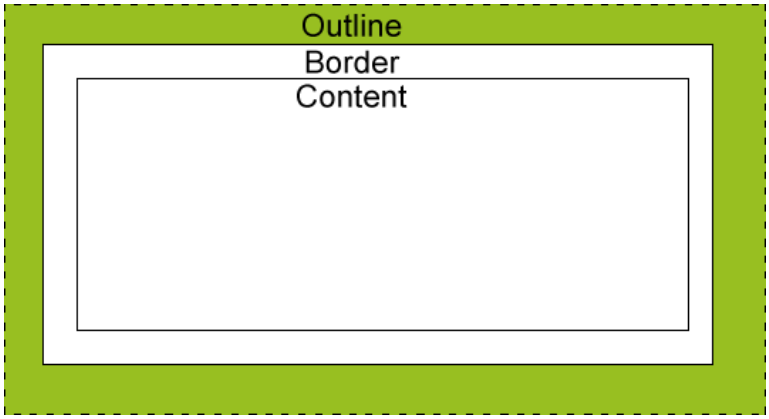
p.groove {outline-style: groove;}

p.ridge {outline-style: ridge;}

p.inset {outline-style: inset;}

p.outset {outline-style: outset;}
```

Remember that, the outline is not a part of an element's dimensions; the element's total width and height is not affected by the width of the outline.



42. Topic 42 – CSS Margin and Padding

CSS Margin

The CSS margin properties define the space around elements. The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent. The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once. Some possible values for margin properties are below:

Value	Description
auto	The browser calculates a margin
length	Specifies a margin in px, pt, cm, etc. Default value is 0px
%	Specifies a margin in percent of the width of the containing element
inherit	Specifies that the margin should be inherited from the parent element

Margin - Individual sides

In CSS, it is possible to specify different margins for different sides of an element. Try this code for a paragraph:

```
p { margin-top: 100px; margin-bottom: 100px; margin-right: 150px; margin-left: 50px; }
```

Margin - Shorthand property

To shorten the code, it is possible to specify all the margin properties in one property. This is called a shorthand property, the "margin" will be used as a short hand property for all margins.

```
p { margin: 100px 50px; }
```

The margin property can have from one to four values.

margin: 25px 50px 75px 100px;

- top margin is 25px
- right margin is 50px
- bottom margin is 75px
- Left margin is 100px

CSS Padding

The CSS padding properties define the space between the element border and the element content. It clears an area around the content (inside the border) of an element. The padding is affected by the background color of the element.

The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property can also be used, to change all padding(s) at once.

Padding - Individual sides

In CSS, it is possible to specify different padding for different sides. Try the code below:

```
p {
padding-top: 25px;
padding-right: 50px;
padding-bottom: 25px;
padding-left: 50px;
}
```

Padding - Shorthand property

To shorten the code, it is possible to specify all the padding properties in one property. This is called its shorthand property. For all the padding properties, the shorthand property is "padding"

```
p {
padding: 25px 50px; }
```

More examples

padding: 25px 50px 75px 100px;

- top padding is 25px

- right padding is 50px
- bottom padding is 75px
- left padding is 100px

padding: 25px;

- all four paddings are 25px

padding: 25px 50px 75px;

- top padding is 25px
- right and left paddings are 50px
- bottom padding is 75px

margin: 25px 50px 75px;

- top and bottom paddings are 25px
- right and left paddings are 50px

43. Topic 43 – CSS Dimension

The CSS dimension properties allow you to control the height and width of an element. Try the code below to set the dimension of an image:

```
img { width: 200px; }
```

Also try this code:

```
p {min-height: 100px; background-color: yellow;}
```

44. Topic 44 – CSS Display

The display property specifies if/how an element is displayed, and the visibility property specifies if an element should be visible or hidden.

Hiding an Element - display: none or visibility: hidden

Hiding an element can be done by setting the display property to "none" or the visibility property to "hidden". However, notice that these two methods produce different results:

Hiding an Element - display: none or visibility: hidden

Visibility: hidden hides an element, but it will still take up the same space as before. The element will be hidden, but still affect the layout. See example below:

```
h1.hidden {visibility: hidden;}
```

Hiding an Element - display: none or visibility: hidden

Display: none hides an element, and it will not take up any space. The element will be hidden, and the page will be

```
h1.hidden {display: none;}
```

CSS Display - Block and Inline Elements

A block element is an element that takes up the full width available, and has a line break before and after it. Examples of block elements:

- <h1>
- <p>
-
-

An inline element only takes up as much width as necessary, and does not force line breaks. Examples of inline elements:

-
- <a>

Changing How an Element is displayed

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow web standards. The example, given below displays <li> elements as inline elements:

```
li {display: inline;}
```

45. Topic 45 – CSS Positioning and Floats

Positioning

The CSS positioning properties allow you to position an element. It can also place an element behind another, and specify what should happen when an element's content is too big. Elements can be positioned using the top, bottom, left, and right properties.

However, these properties will not work unless the position property is set first. They also work differently depending on the positioning method. There are four different positioning methods.

1. Static
2. Fixed
3. Relative
4. Absolute

1. Static Positioning

HTML elements are positioned static by default. A static positioned element is always positioned according to the normal flow of the page.

Static positioned elements are not affected by the top, bottom, left, and right properties.

2. Fixed Positioning

An element with a fixed position is positioned relative to the browser window, and will not move even if the window is scrolled.

p.pos_fixed { position: fixed; top: 30px; right: 5px; }

Fixed positioned elements are removed from the normal flow. The document and other elements behave like the fixed positioned element does not exist. Fixed positioned elements can overlap other elements.

3. **Relative Positioning**

A relative positioned element is positioned relative to its normal position. Examples are given below:

h2.pos_left {position: relative; left: -20px;}

h2.pos_right {position: relative; left: 20px;}

The content of relatively positioned elements can be moved and overlap other elements, but the

reserved space for the element is still preserved in the normal flow.

h2.pos_top {position: relative; top: -50px;}

Relatively positioned elements are often used as container blocks for absolutely positioned elements.

4. **Absolute Positioning**

An absolute position element is positioned relative to the first parent element that has a position other than static. If no such element is found, the containing block is <html>:

h2 {

position: absolute;

left: 100px;

top: 150px;

}

Absolutely positioned elements are removed from the normal flow. The document and other elements behave like the absolutely positioned element does not exist. Absolutely positioned elements can overlap other elements.

Overlapping Elements

When elements are positioned outside the normal flow, they can overlap other elements.

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the other.)

An element can have positive and negative stack order.

img {

```
position: absolute;

left: 0px;

top: 0px;

z-index: -1;

}
```

46. Topic 46 – CSS Positioning and Floats_2

CSS Float

With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it. Float is often used with images, but it is also useful when working with layouts.

How Elements Float?

Elements are floated horizontally; this means that an element can only be floated left or right, not up or down. A floated element will move as far to the left or right as it can. Usually this means all the way to the left or right of the containing element.

The elements after the floating element will flow around it. The elements before the floating element will not be affected. If an image is floated to the right, a following text flows around it, to the left. Try the code given below:

```
img {

float: right;

}
```

Floating Elements Next to Each Other

If you place several floating elements after each other, they will float next to each other if there is room.

```
. thumbnail {

float: left;

width: 110px;

height: 90px;

margin: 5px;

}
```

Turning off Float - Using Clear

Elements after the floating element will flow around it. To avoid this, use the clear property. The clear property specifies which sides of an element other floating elements are not allowed.

47. Topic 47 – CSS Aligns

Aligning Block Elements

A block element is an element that takes up the full width available, and has a line break before and after it.

Examples of block elements:

- <h1>
- <p>
-

Center Aligning Using the margin Property

Block elements can be center-aligned by setting the left and right margins to "auto". Setting the left and right margins to auto specifies that they should split the available margin equally. The result is a centered element. Try this example code:

```
. center {
Margin-left: auto;
margin-right: auto;
width: 70%;
background-color: #b0e0e6;
}
```

Left and Right Aligning Using the position Property

One method of aligning elements is to use absolute positioning. See an example below:

```
. right {
position: absolute;
right: 0px;
width: 300px;
background-color: #b0e0e6;
}
```

Cross Browser Compatibility Issues

When aligning elements like this, it is always a good idea to predefine margin and padding for the <body> element. This is to avoid visual differences in different browsers. There is problem with IE8 and earlier, whrn using the position property.

If a container element (in our case) has a specified width, and the! DOCTYPE declaration is missing, IE8 and earlier versions will add a 17px margin on the right side. This seems to be space reserved for a scrollbar. Always set the !DOCTYPE declaration when using the position property.

```
body { margin: 0; padding: 0; }

.container { position: relative; width: 100%;}

.right { position: absolute; right: 0px; width: 300px; background-color: #b0e0e6; }
```

Left and Right Aligning Using the float Property

One method of aligning elements is to use the float property. Example is given below:

```
.right { float: right; width: 300px; background-color: #b0e0e6; }
```

48. Topic 48 – CSS Combinators

A combinator is something that explains the relationship between the selectors.

CSS Combinators

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator. There are four different combinators in CSS3:

1. Descendant Selector
2. Child Selector
3. Adjacent Sibling Selector
4. General Sibling Selector

1. Descendant Selector

The descendant selector matches all elements that are descendants of a specified element. The example given below selects all <p> elements inside elements:

```
div p {

background-color: yellow;

}
```

2. Child Selector

The child selector selects all elements that are the immediate children of a specified element. The example which is given below selects all <p> elements that are immediate children of a element:

```
div > p {

background-color: yellow;

}
```

3. Adjacent Sibling Selector

The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

This code selects the <p> elements that are placed immediately after elements:

```
div + p {
background-color: yellow;
}
```

4. General Sibling Selector

The general sibling selector selects all elements that are siblings of a specified element.

The example code given below selects all <p> elements that are siblings of elements:

```
div ~ p {
background-color: yellow;
}
```

49. Topic 49 – CSS Pseudo-Class

What are Pseudo-classes?

A pseudo-class is used to define a special state of an element. For example, it can be used to:

- Style an element when a user mouse is over it.
- Style visited and unvisited links differently.

The syntax of pseudo-classes:

```
selector:pseudo-class { property:value; }
```

Anchor Pseudo-classes

Links can be displayed in different ways:

/* unvisited link */

```
a:link {
color: #FF0000;
}
```

/* visited link */

```
a:visited {
```

color: #00FF00;

}

/* mouse over link */

a:hover {

color: #FF00FF;

}

/* selected link */

a:active {

color: #0000FF;

}

Pseudo-classes and CSS Classes

Pseudo-classes can be combined with CSS classes.

a.highlight:hover { color: #ff0000; }

When you hover over the link with class highlight, it will change color.

CSS - The: first-child Pseudo-class

The: first-child pseudo-class matches a specified element that is the first child of another element.

In the following example, the selector matches any <p> element that is the first child of any element. See the examples given below:

p: first-child {color: blue;}

In the following example, the selector matches the first <i> element in all <p> elements:

p i: first-child {color: blue;}

In the following example, the selector matches all <i> elements in <p> elements that are the first child of another element:

p: first-child i {color: blue;}

CSS - The: lang Pseudo-class

The: lang pseudo-class allows you to define special rules for different languages.

Note: IE8 supports the: lang pseudo-class only if a <!DOCTYPE> is specified.

In the example below, the: lang class defines the quotation marks for <q> elements with lang="no":

```
<html>
<head>
<style> q:lang(no) {
quotes: "~" "~";
}
</style>
</head>
```

50. Topic 50 – Pseudo-Elements

What are Pseudo-Elements?

A CSS pseudo-element is used to style specified parts of an element. For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

Syntax

The syntax of pseudo-elements:

```
selector::pseudo-element {
property:value;
}
```

The ::first-line Pseudo-element

The :: first-line pseudo-element is used to add a special style to the first line of a text.

The :: first-line pseudo-element can only be applied to block elements. Format the first line of the text in all <p> elements:

```
p::first-line { color: #ff0000; font-variant: small-caps; }
```

The ::first-line Pseudo-element

The following properties apply to the ::first-line pseudo-element:

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing

- vertical-align
- text-transform
- line-height
- clear

The ::first-letter Pseudo-element

The ::first-letter pseudo-element is used to add a special style to the first letter of a text.

The :: first-line pseudo-element can only be applied to block elements. Format the first line of the text in all <p> elements:

```
p::first-letter { color: #ff0000; font-size: xx-large; }
```

The following properties apply to the ::first-letter pseudo- element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

Pseudo-elements and CSS Classes

Pseudo-elements can be combined with CSS classes. The example below will display the first letter of paragraphs with class="intro", in red and in a larger size.

```
p.intro::first-letter { color: #ff0000; font-size:200%; }
```

Multiple Pseudo-elements

Several pseudo-elements can also be combined.

Multiple Pseudo-elements

Let see an example, with the first letter of a paragraph in red, in an xx-large font size. The rest of the first line will be blue, an in small-caps. The rest of the paragraph will be default font size and color.

```
p::first-letter {color: #ff0000; font-size: xx-large; }

p::first-line { color: #0000ff; font-variant: small-caps; }
```

The ::before pseudo-element can be used to insert some content before the content of an element. The following example inserts an image before each <h1> element:

```
h1::before { content: url(smiley.gif); }
```

CSS - The ::after Pseudo-element

The ::after pseudo-element can be used to insert some content after the content of an element. The following example inserts an image after each <h1> element:

```
h1::after {content: url(smiley.gif); }
```

CSS - The ::selection Pseudo-element

The ::selection pseudo-element matches the portion of an element that is selected by a user. The following example makes the selected text red on a yellow background:

These properties can be applied to ::selection: color, background, cursor, and outline.

```
::selection {color: red; background: yellow;}
```

51. Topic 51 – CSS Class

In CSS, class is a type of selector. The .class selector styles all elements with the specified class attribute value.

Class Syntax

```
.class {
css declarations;
}
```

Example

```
.imp {color:blue; font-size:14px}
<div class="imp"> … </div>
<span class="imp"> … </span>
<a class="imp"> … </a>
```

The Class Selector

You can also specify that only specific HTML elements should be affected by a class. See the example code below:

```
p.imp {color:blue; font-size:14px}
<div class="imp"> … </div>
<span class="imp"> … </span>
```

```
<p class="imp"> … </p>
```

You can use alpha numeric characters for a class name. Do NOT start a class name with a number.

52. Topic 52 – CSS Navigation Menu

Navigation Bars

Having easy-to-use navigation is important for any web site. With CSS you can transform boring HTML menus into good-looking navigation bars.

Navigation Bar = List of Links

- A navigation bar needs standard HTML as a base.
- Here we will build the navigation bar from a standard HTML list.

A navigation bar is basically a list of links, so using the <ul> and <li> elements make perfect sense.

So, let's start to create a navigation bar. Define in HTML the code:

```
<ul>

<li><a href="default.asp">Home</a></li>

<li><a href="news.asp">News</a></li>

<li><a href="contact.asp">Contact</a></li>

<li><a href="about.asp">About</a></li>

</ul>
```

Now let's remove the bullets and the margins and padding from the list

```
ul { list-style-type: none; margin: 0; padding: 0;}
```

list-style-type: none - Removes the bullets. A navigation bar does not need list markers. Setting margins and padding to 0 to remove browser default settings. This code is the standard code used in both vertical and horizontal navigation bars.

Vertical Navigation Bar

To build a vertical navigation bar we only need to style the <a> element, in addition to the code we have.

```
a {display: block; width: 60px;}
```

Explanation:

- display: block - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width
- width: 60px - Block elements take up the full width available by default. We want to specify a 60px width

Horizontal Navigation Bar

There are two ways to create a horizontal navigation bar. Using inline or floating list items.

Both methods work fine, but if you want the links to be the same size, you have to use the floating method.

One way to build a horizontal navigation bar is to specify the <li> elements as inline, in addition to the "standard" code we have:

```
li {display: inline;}
```

Explanation:

display: inline; - By default, <li> elements are block elements. Here, we remove the line breaks before and after each list item, to display them on one line.

Floating List Items

Here links have different widths. For all the links to have an equal width, float the <li> elements and specify a width for the <a> elements.

```
li {float: left; }

a { display: block; width: 60px; }
```

Explanation

- width: 60px - Since block elements take up the full width available, they cannot float next to each other. We specify the width of the links to 60px.
- float: left - use float to get block elements to slide next to each other
- display: block - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width

CSS Image Opacity

53. Topic 53 – CSS Image Opacity

CSS Image Opacity / Transparency

Creating transparent images with CSS is easy. The CSS opacity property is a part of the CSS3 recommendation.

Creating a Transparent Image

The CSS3 property for transparency is opacity. First will show you how to create a transparent image with CSS:

```
img {opacity: 0.4; filter: alpha(opacity=40); /* For IE8 and earlier */ }
```

Explanation:

IE9, Firefox, Chrome, Opera, and Safari use the property opacity for transparency. The opacity property can take a value from 0.0 - 1.0. A lower value makes the element more transparent.

IE8 and earlier use:

filter:alpha(opacity=x)

The x can take a value from 0 - 100. A lower value makes the element more transparent.

Image Transparency - Hover Effect

```
img { opacity: 0.4; filter: alpha(opacity=40); /* For IE8 and earlier */ }

img:hover { opacity: 1.0; filter: alpha(opacity=100); /* For IE8 and earlier */ }
```

Text in Transparent Box

First, we create a element (class="background") with a background image, and a border. Then we create another (class="transbox") inside the first . The have a background color, and a border - the div is transparent.

Text in Transparent Box

Inside the transparent , we add some text inside a <p> element.

54. Topic 54 – CSS Image Sprites

What is Image Sprites?

An image sprite is a collection of images put into a single image. A web page with many images can take a long time to load and generates multiple server requests. Using image sprites will reduce the number of server requests and save bandwidth.

See this Example:

Instead of using three separate images, we use this single image ("img_navsprites.gif"):



With CSS, we can show just the part of the image we need. In the following example the CSS specifies which part of the "img_navsprites.gif" image to show:"):

```
#home {width: 46px; height: 44px; background: url(img_navsprites.gif) 0 0;}
```

Explanation:

- <img id="home" src="img_trans.gif"> - Only defines a small transparent image because the src attribute cannot be empty. The displayed image will be the background image we specify in CSS
- width: 46px; height: 44px; - Defines the portion of the image we want to use
- background: url(img_navsprites.gif) 0 0; - Defines the background image and its position (left 0px, top 0px)

Create a Navigation List

We want to use the sprite image ("img_navsprites.gif") to create a navigation list. We will use an HTML list, because it can be a link and also supports a background image.

- #navlist {position: relative;} - position is set to relative to allow absolute positioning inside it
- #navlist li {margin:0; padding:0; list-style: none; position: absolute; top:0;} - margin and padding is set to 0, list-style is removed, and all list items are absolute positioned
- #navlist li, #navlist a {height:44px; display: block;} - the height of all the images are 44px

To start the position and style for each specific part:

- #home {left:0px; width:46px;} - Positioned all the way to the left, and the width of the image is 46px
- #home {background:url(img_navsprites.gif) 0 0;} - Defines the background image and its position (left 0px, top 0px)
- #prev {left:63px; width:43px;} - Positioned 63px to the right (#home width 46px + some extra space between items), and the width is 43px.
- #prev {background:url('img_navsprites.gif') -47px 0;} - Defines the background image 47px to the right (#home width 46px + 1px line divider)
- #next {left:129px; width:43px;} - Positioned 129px to the right (start of #prev is 63px + #prev width 43px + extra space), and the width is 43px.
- #next {background:url('img_navsprites.gif') -91px 0;} - Defines the background image 91px to the right (#home width 46px + 1px line divider + #prev width 43px + 1px line divider)

Image Sprites - Hover Effect

Now we want to add a hover effect to our navigation list. Our new image ("img_navsprites_hover.gif") contains three navigation images and three images to use for hover effects:



Because this is one single image, and not six separate files, there will be no loading delay when a user hovers over the image. We only add three lines of code to add the hover effect:

- #home a:hover { background: url('img_navsprites_hover.gif') 0 -45px; }
- #prev a:hover { background: url('img_navsprites_hover.gif') -47px -45px; }
- #next a:hover { background: url('img_navsprites_hover.gif') -91px -45px; }

CSS Media Types

55. Topic 55 – CSS Media Types

Media Types

Some CSS properties are designed for a specific type of media. For example the "voice-family" property is designed

For example, the "font-size" property can be used for both screen and print media, but perhaps with different values. A document usually needs a larger font-size on a screen than on paper, and sans-serif fonts are easier to read on the screen, while serif fonts are easier to read on paper.

The @media Rule

The @media rule makes it possible to define different style rules for different media types in the same stylesheet. The CSS in the example below tells the browser to display a 17 pixels Verdana font on the screen. But if the page is printed, it will be in a blue 14 pixels Georgia font:

```
@media screen {
p {
font-family: verdana, sans-serif;
font-size: 17px;
}
}
```

56. Topic 56 – CSS Attribute Selectors

Style HTML Elements with Specific Attributes

It is possible to style HTML elements that have specific attributes or attribute values.

1. **CSS [attribute] Selector**

The [attribute] selector is used to select elements with a specified attribute. The following example selects all <a> elements with a target attribute:

```
a[target] {
background-color: yellow;
}
```

2. **CSS [attribute=value] Selector**

The [attribute=value] selector is used to select elements with a specified attribute and value. The example, which is given below selects all <a> elements with a target="_blank" attribute:

```
a[target="_blank"] { background-color: yellow; }
```

3. **CSS [attribute~=value] Selector**

The [attribute~=value] selector is used to select elements with an attribute value containing a specified word. The example, which is given below selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower":

[title~="flower"] { border: 5px solid yellow; }

4. **CSS [attribute|=value] Selector**

The [attribute|=value] selector is used to select elements with the specified attribute starting with the specified value. The following example selects all elements with a class attribute value that begins with "top":

[class|="top"] { background: yellow; }

Note: The value has to be a whole word, either alone, like class="top", or followed by a hyphen(-), like class="top-text"!

5. **CSS [attribute^=value] Selector**

The [attribute^=value] selector is used to select elements whose attribute value begins with a specified value. The following example selects all elements with a class attribute value that begins with "top":

[class^="top"] {background: yellow; }

6. **CSS [attribute$=value] Selector**

The [attribute$=value] selector is used to select elements whose attribute value ends with a specified value. The following example selects all elements with a class attribute value that ends with "test":

[class$="test"] { background: yellow; }

7. **CSS [attribute*=value] Selector**

The [attribute*=value] selector is used to select elements whose attribute value contains a specified value. The following example selects all elements with a class attribute value that contains "te":

[class*="te"] { background: yellow; }

Styling Forms

The attribute selectors can be useful for styling forms without class or ID

input[type="text"] {width: 150px; display: block; margin-bottom: 10px; background- color: yellow; }

input[type="button"] { width: 120px; margin-left: 35px; display: block;}

57. Topic 57 – Fundamentals of JavaScript

JavaScript is the programming language of HTML and the Web. It is a small and lightweight language. Programming makes computers do what you want them to do. JavaScript is easy to learn.

JavaScript contains a standard library of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements. Core JavaScript can be extended for a variety of purposes by supplementing it with additional objects. Let’s discuss the fundamentals of JavaScript.

Simple JavaScript code in HTML Document:

```
<!DOCTYPE html>

<html>

<body>

<h1>My First JavaScript</h1>

<button type="button" onclick="document.getElementById('demo').innerHTML = Date()"> Click me to display Date and Time.</button>

<p id="demo"></p>

</body>

</html>
```

JavaScript Can Change HTML Content

One of many HTML methods is getElementById(). Let's use the method to "find" an HTML element (with id="demo"), and changes the element content (innerHTML) to "Hello JavaScript":

```
<!DOCTYPE html>

<html>

<body>

<h1>My First JavaScript</h1>

<button type="button" onclick="document.getElementById('demo').innerHTML = Date()"> Click me to display Date and Time.</button>

<p id="demo"></p>

</body>

</html>
```

JavaScript Can Change HTML Attributes

Let's change an HTML image, by changing the src attribute of an <img> tag:

```
<script>

function changeImage() {

var image = document.getElementById('myImage'); if (image.src.match("bulbon")) {

image.src = "pic_bulboff.gif";

} else {
```

```
image.src = "pic_bulbon.gif";}}

</script>
```

Explanation

Get two images for a bulb ON and OFF, give them the name as per specified in code and try. After clicking on image, the bulb will turn ON and OFF.

JavaScript Can Change HTML Styles (CSS)

Changing the style of an HTML element, is a variant of changing an HTML attribute. Try the JS script given below:

```
<script>

function myFunction() { var x, text;

// Get the value of the input field with id="numb" x = document.getElementById("numb").value;

// If x is Not a Number or less than one or greater than 10 if (isNaN(x) || x < 1 || x > 10) { text = "Input not valid“; } else { text = "Input OK"; }

document.getElementById("demo").innerHTML = text;}

</script>
```

58. Topic 58 – JavaScript Syntax

JavaScript Programs:

A computer program is a list of "instructions" to be "executed" by the computer. In a programming language, these program instructions are called statements, and we have already discussed that, JavaScript is a programming language.

Script statements are separated by semicolon. See some examples below:

```
var x = 5;

var y = 6;

var z = x + y;
```

JavaScript Statements

JavaScript statements are composed of Values, Operators, Expressions, Keywords, and Comments.

JavaScript Values

The JavaScript syntax defines two types of values:

- Fixed values
- Variable values.

Whereas, fixed values are called literals. Variable values are called variables.

JavaScript Literals

The most important rules for writing fixed values are:

Numbers are written with or without decimals such as:

- 10.50
- 1001

Strings are text, written within double or single quotes:

- "John Doe"
- 'John Doe'

Expressions can also represent fixed values:

- 5 + 6
- 5 * 10

JavaScript Variables

In a programming language, variables are used to store data values.

- JavaScript uses the var keyword to define variables.
- An equal sign is used to assign values to variables.

Check out some examples given below:

In the example which is given below, x is defined as a variable. Then, x is assigned (given) the value 8:

- var x;
- x = 8;

JavaScript Operators

JavaScript uses an assignment operator (=) to assign values to variables:

- var x = 5;
- var y = 6;

JavaScript uses arithmetic operators (+ - * /) to compute values:

- (5+6)*10;
- var x = (5 + 6) * 10;

JavaScript Keywords

JavaScript keywords are used to identify actions to be performed. The var keyword tells the browser to create a new variable:

- var x = 5 + 6;
- var y = x * 10;

JavaScript Comments

Not all JavaScript statements are "executed". Code after double slashes // or between /* and */ is treated as a comment. Comments are ignored, and will not be executed:

- var x = 5; // will be executed
- // var x = 6; will NOT be executed

JavaScript is Case Sensitive

All JavaScript identifiers are case sensitive. The variables lastName and lastname, are two different variables.

- lastName = "Doe";
- lastname = "Peterson";

JavaScript and Camel Case

Historically, programmers have used three ways of joining multiple words into one variable name:

Hyphens:

- first-name
- last-name
- master-card

Underscore:

- first_name
- last_name
- master_card

Camel Case:

- FirstName
- LastName
- MasterCard

In programming languages, especially in JavaScript, camel case often starts with a lowercase letter:

- firstName, masterCard

JavaScript Character Set

JavaScript uses the Unicode character set. Unicode covers (almost) all the characters, punctuations, and symbols in the world.

59. Topic 59 – JavaScript Statements

In language, a statement tells the browser what to do. The statements are executed, one by one, in the same order as they are written. Most JavaScript programs contain many JavaScript statements. Here is a JavaScript Statement:

```
document.getElementById("demo").innerHTML = “Hi, Everyone.";
```

Explanation: This statement tells the browser to write "Hello Everyone." inside an HTML element with id="demo".

Let’s discuss some general elements in JavaScript statements

Semicolons(;)

Semicolons in JavaScript separate JavaScript statements. Add a semicolon at the end of each executable statement. Let evaluate the role of semicolons in JS through the example given below:

```
a= 5;

b = 6;

c = a + b;
```

When separated by semicolons, multiple statements on one line are allowed:

```
a = 5; b = 6; c = a + b;
```

JavaScript White Space

JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.

JavaScript Line Length and Line Breaks

For best readability, programmers often like to avoid code lines longer than 80 characters. If a JavaScript statement does not fit on one line, the best place to break it is after an operator:

```
document.getElementById("demo").innerHTML = "Hello Dolly.";
```

JavaScript Code Blocks

JavaScript statements can be grouped together in code blocks, inside curly brackets {...}. The purpose of code blocks is to define statements to be executed together.

60. Topic 60 – JavaScript Comments

JavaScript comments can be used to explain JavaScript code, and to make it more readable. Furthermore, you can also use the JS comments to prevent execution, when testing alternative code.

- Single Line Comments: Single line comments start with //. Any text between // and the end of the line, will be ignored by JavaScript (will not be executed).
- Multi-line Comments: Multi-line comments start with /* and end with */. Any text between /* and */ will be ignored by JavaScript.

Using Comments to Prevent Execution

Using comments to prevent execution of code, is suitable for code testing. Adding // in front of a code line changes the code lines from an executable line to a comment.

61. Topic 61 – JavaScript Variables & Operators

JavaScript Variables: JavaScript variables are containers for storing data values.

In the example, given below, x, y, and z, are variables:

```
var x = 5; var y = 6; var z = x + y;
```

From the example above, you can expect:

- x stores the value 5
- y stores the value 6
- z stores the value 11

JavaScript Identifiers

All JavaScript variables must be identified with unique names and these unique names are called identifiers. Identifiers can be short names (like x and y), or more descriptive names (age, sum, totalVolume). The general rules for constructing names for variables (unique identifiers) are:

- Names can contain letters, digits, underscores, and dollar signs.
- Names must begin with a letter
- Names can also begin with $ and _
- Names are case sensitive (y and Y are different variables)
- Reserved words (like JavaScript keywords) cannot be used as names

The Assignment Operator

In JavaScript, the equal sign (=) is an "assignment" operator, not an "equal to" operator. This is different from algebra.

JavaScript Data Types

JavaScript variables can hold numbers like 100, and text values like "John Doe". In programming, text values are called text strings. JavaScript can handle many types of data but for now, just think of numbers and strings:

Strings are written inside double or single quotes. Numbers are written without quotes. If you put quotes around a number, it will be treated as a text string.

```
Var pi=3.14;

Var person="John Doe";

Var answer = 'Yes I am!';

var anum = '3.14';
```

Declaring (Creating) JavaScript Variables

Creating a variable in JavaScript is called "declaring" a variable. You declare a JavaScript variable with the var keyword:

```
var carName;
```

After the declaration, the variable is empty (it has no value). To assign a value to the variable, use the equal sign:

```
carName = "Volvo";
```

See the example:

```
<p id="demo"></p>

<script>

var carName = "Volvo"; document.getElementById("demo").innerHTML = carName;

</script>
```

One Statement, Many Variables

You can declare many variables in one statement. Start the statement with var and separate the variables by comma:

```
var person = "John Doe", carName = "Volvo", price = 200;
```

Value = undefined

In computer programs, variables are often declared without a value. The value can be something that has to be calculated, or something that will be provided later, like user input.

A variable declared without a value will have the value undefined. The variable carName will have the value undefined after the execution of this statement:

var carName;

Note: If you re-declare a JavaScript variable, it will not lose its value.

62. Topic 62 – JavaScript Variables & Operators_2

JavaScript Operators

As with algebra, you can do arithmetic with JavaScript variables, using operators like = and +

See these Examples:

```
var x = 5 + 2 + 3;

var x = 5;        // assign the value 5 to x

var y = 2;        // assign the value 2 to y

var z = x + y;        // assign the value 7 to z (x + y)
```

JavaScript String Operators

In JavaScript, the + operator can also be used to add (concatenate) strings. See the example below:

```
txt1 = "John";

txt2 = "Doe";

txt3 = txt1 + " " + txt2;
```

The += assignment operator can also be used to add (concatenate) strings:

```
txt1 = "What a very ";

txt1 += "nice day";
```

Adding Strings and Numbers

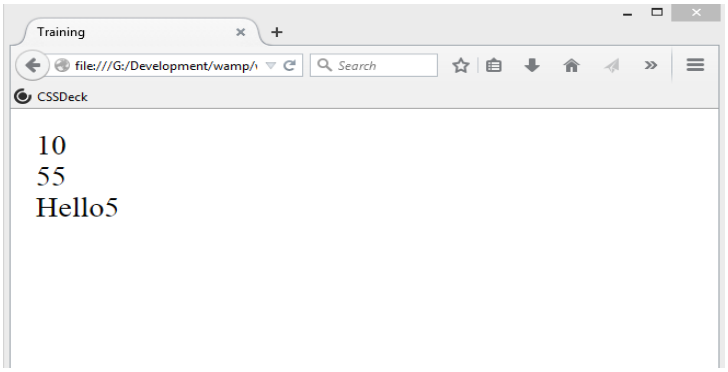
Adding two numbers will return the sum, but adding a number and a string will return a string:

```
x = 5 + 5;

y = "5" + 5;

z= "Hello" + 5;
```

Output



63. Topic 63 – JavaScript Functions

What is JavaScript Function?

A JavaScript function is a block of code designed to perform a particular task. The function is executed when "something" invokes it (calls it).

JavaScript Function Syntax

A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses (). Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

function name (parameter1, parameter2, parameter3) { code to be executed }

Explanation:

Function parameters are the names listed in the function definition. Function arguments are the real values received by the function when it is invoked.

Function Invocation

The code inside the JS function will execute when "something" invokes (calls) the function:

- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self-invoked)

Function Return

When JavaScript reaches a return statement, the function will stop executing. If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

Why Functions? (Importance of Functions)

You can reuse code: Define the code once, and use it many times. You can use the same code many times with different arguments, to produce different results. See the example below:

function toCelsius(fahrenheit) { return (5/9) * (fahrenheit-32);

}

document.getElementById("demo").innerHTML = toCelsius(32);

The () Operator Invokes the Function

Using the example above, toCelsius refers to the function object, and toCelcius() refers to the function result.

Accessing a function without () will return the function definition:

Function toCelsius(fahrenheit)

{ return (5/9) * (fahrenheit-32); }

document.getElementById("demo").innerHTML = toCelsius;

Functions Used as Variables

In JavaScript, you can use functions the same way as you use variables. See the example below: You can use:

var text = "The temperature is " + toCelsius(32) + " Centigrade"; Instead of:

var x = toCelsius(32);

var text = "The temperature is " + x + " Centigrade";

64. Topic 62 – JavaScript Objects

Real Life Objects, Properties, and Methods

Object properties can be primitive values, other objects, and functions. An object method is an object property containing a function definition. JavaScript objects are containers for named values, called properties and methods.

In a real life, a car is an object. A car has properties like weight and color, and methods like start and stop.

All cars have the same methods, but the methods are performed at different times.

JavaScript Objects

You have already learned that JavaScript variables are containers for data values. This code assigns a simple valuye (Fiat) to a variable named car:

```
var car = " Fiat";
```

Objects are variables too but objects can contain many values. This codde assigns many valyes (Fiat, 500, white) to a vairbale named car:

```
var car = {type: " Fiat", model:500, color: " white"};
```

Objects Properties

The name:values pairs (in JavaScript objects) are called properties.

Property	Property Value
firstName	John
lastName	Doe
Age	50
eyeColor	blue

Objects Methods

Methods are actions that can be performed on objects. Methods are stored in properties as function definitions.

Property	Property Value
firstName	John
lastName	Doe

Age	50
eyeColor	blue
fullName	Function() {return this.firstName+ " "+this.lastName;}

Objects Definition

You can define and creare a Javascript object with an object literal. For example:

Var person = {firstName: "John", lastName: "Doe", age:50, eyeColor: "blue"};

Spaces and line breaks are not important. An object definition can span multiple lines. For example:

Var person = {

firstName: "John",

lastName: "Doe",

age:50,

eyeColor: "blue"

};

Accessing Object Properties

We can access object properties in two ways:

objectName.propertyName

Example:

Person.lastName

or

objectName[propertyName]

example:

person["lastName"];

Accessing Object Methods

You can access an object with the following syntax

objectName.methodName()

Example:

Name=person.fullName();

If you can access the fullName property, without (), it will return the function definition: example

Name = person.fullName;

Do not declare Strings, Numbers, and Booleans as object because when a Javascrips variale is dec;ared with the keyword "new", the variable is created as an object:

Example

- var x= new String(); //declares x as a String object
- var y= new Number(); //declares y as a Num object
- var z = new Boolean(); //declares z as a Boolean object

65. Topic 65 – JavaScript Scope

Scope is the set of variables, objects, and functions you have access to. In JavaScript, objects and functions are also variables. JavaScript has function scope: The scope changes inside functions.

Local JavaScript Variables

Variables declared within a JavaScript function, become LOCAL to the function. Local variables have local scope: They can only be accessed within the function. Local variables are created when a function starts, and deleted when the function is completed. See the example below:

```
//code here cannot use variable userName

function myFunction() {

var userName = “Tariq";

// code here can use variable userName

}
```

Global JavaScript Variables

A variable declared outside a function, becomes GLOBAL. A global variable has global scope: All scripts and functions on a web page can access it. The global scope is the complete JavaScript environment.

Automatically Global

If you assign a value to a variable that has not been declared, it will automatically become a GLOBAL variable.

The Lifetime of JavaScript Variables: The lifetime of a JavaScript variable starts when it is declared.

- Local variables are deleted when the function is completed.
- Global variables are deleted when you close the page.

- Function arguments (parameters) work as local variables inside functions.
- In HTML, the global scope is the window object: All global variables belong to the window object.

66. Topic 66 – JavaScript Events

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

HTML Events

- An HTML event can be something the browser does, or something a user does.
- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

Often, when events happen, you may want to do something. JavaScript lets you execute code when events are detected. HTML allows event handler attributes, with JavaScript code, to be added to HTML elements.

See the example below:

With single quotes:

```
<some-HTML-element some-event='some JavaScript'>
```

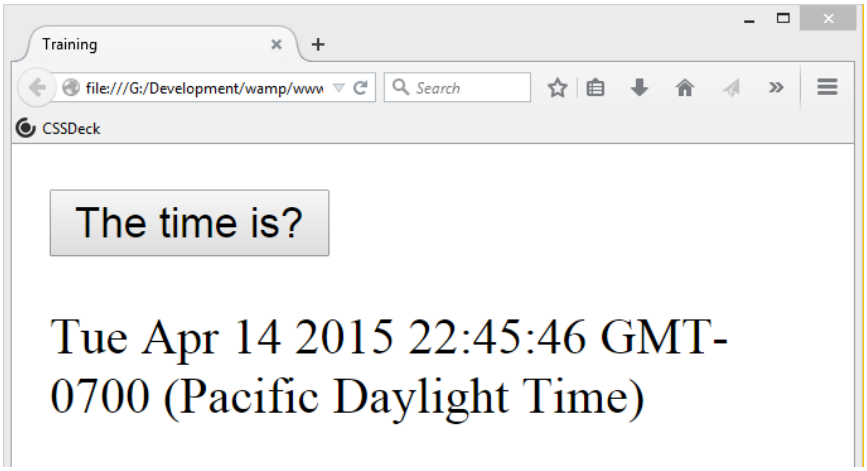
With double quotes:

```
<some-HTML-element some-event="some JavaScript">
```

In the following example, an onclick attribute (with code), is added to a button element:

```
<!DOCTYPE html>
<html>
<body>
<button
onclick="getElementById('demo').innerHTML=Date()">The time is?</button>
<p id="demo"></p>
</body>
</html>
```

Output in Browser:

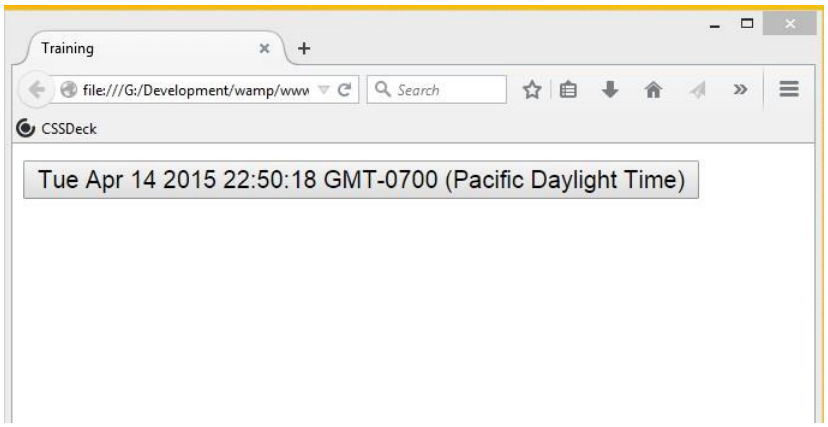


Here, the JavaScript code changes the content of the element with id="demo".

In the next example, the code changes the content of its own element (using this.innerHTML):

```
<!DOCTYPE html>
<html>
<body>
<button onclick="this.innerHTML=Date()">The time is?</button>
</body>
</html>
```

Output in Browser:



Furthermore, you can search online for HTML events with practical demonstration.

67. Topic 67 – JavaScript Strings

A JavaScript string simply stores a series of characters like "Mike Slough". A string can be any text inside quotes. You can use single or double quotes. For example:

var carname = "Volvo XC60"; (with double quotes)

var carname = 'Volvo XC60'; (with double quotes)

Furthermore, you can use quotes inside a string, as long as they don't match the quotes surrounding the string. See this example:

Var answer = "It's alright";

Var answer = "He is called 'Johnny'";

var answer = 'He is called "Johnny"';

String Length

The length of a string is found in the built in property length. See the example below:

Var txt ="ABCDEFGHIJKLMNOPQRSTUVWXYZ";

var sln = txt.length;

Special Characters: because strings must be written within quotes, JavaScript will misunderstand this string:

var y = "We are the so-called "Vikings" from the north."

Explanation

The string will be chopped to "We are the so-called ". The solution to avoid this problem is to use the \ escape character. The backslash escape character turns special characters into string characters. Checkout the example below:

Var x = 'It\'s alright';

var y = "We are the so-called \"Vikings\" from the north. “

The escape character (\) can also be used to insert other special characters in a string.

Breaking Long Code Lines

For best readability, programmers often like to avoid code lines longer than 80 characters. If a JavaScript statement does not fit on one line, the best place to break it is after an operator. For further clarification, see the examples below:

Example

document.getElementById("demo").innerHTML = “hi,how are you?.";

You can also break up a code line within a text string with a single backslash:

Example

document.getElementById("demo").innerHTML = “hi \ How are you?";

The safest (but a little slower) way to break a long string is to use string addition:

Example

```
document.getElementById("demo").innerHTML = “Hi," + “How are you?”;
```

Example

You cannot break up a code line with a backslash:

```
document.getElementById("demo").innerHTML = \ " Hi, How are you?";
```

Strings Can be Objects

Normally, JavaScript strings are primitive values, created from literals:

```
var firstName = “Mike"
```

But strings can also be defined as objects with the keyword new. Check out an example below:

```
var x = “Mike";

var y = new String(“Mike");

// typeof x will return string

// typeof y will return object
```

When using the == equality operator, equal strings look equal:

```
var x = “Mike";

var y = new String(“Mike");

// (x == y) is true because x and y have equal values
```

When using the === equality operator, equal strings are not equal, because the === operator expects equality in both type and value.

```
var x = “Mike";

var y = new String(“Mike");
```

String Properties and Methods

Primitive values, like “Mike Slough", cannot have properties or methods (because they are not objects). But with JavaScript, methods and properties are also available to primitive values, because JavaScript treats primitive values as objects when executing methods and properties.

68. Topic 68 – JavaScript Numbers

JavaScript numbers can be written with, or without decimals. See in example below:

```
var x = 34.00;      // A number with decimals

var y = 34;         // A number without decimals
```

Furthermore, extra-large or extra small numbers can be written with scientific (exponent) notation:

```
var x = 123e5;      // 12300000

var y = 123e-5; // 0.00123
```

JavaScript Numbers are Always 64-bit Floating Point

Unlike many other programming languages, JavaScript does not define different types of numbers, like integers, short, long, floating-point etc.

JavaScript numbers are always stored as double precision floating point numbers, following the international IEEE 754 standard.

This format stores numbers in 64 bits, where the number (the fraction) is stored in bits 0 to 51, the exponent in bits 52 to 62, and the sign in bit 63:

Value (aka Fraction/Mantissa)	Exponent	Sign
52 bits (0 - 51)	11 bits (52 - 62)	1 bit (63)

Precision

Integers (numbers without a period or exponent notation) are considered accurate up to 15 digits. See the example below:

```
var x = 999999999999999;

// x will be 999999999999999 var y = 9999999999999999;

// y will be 10000000000000000
```

The maximum number of decimals is 17, but floating-point arithmetic is not always 100% accurate:

```
var x = 0.2 + 0.1;

// x will be 0.30000000000000004
```

To solve the problem, it helps to multiply and divide:

```
var x = (0.2 * 10 + 0.1 * 10) / 10;  // x will be 0.3
```

Hexadecimal

```
var x = 0xFF;// x will be 255
```

Infinity

Infinity (or -Infinity) is the value JavaScript will return if you calculate a number outside the largest possible number. See the example below:

```
var myNumber = 2;

while (myNumber != Infinity) {     // Execute until Infinity myNumber = myNumber * myNumber;

}
```

Division by 0 (zero) also generates Infinity:

```
var x = 2 / 0;         // x will be Infinity

var y = -2 / 0;        // y will be –Infinity
```

Infinity is a number: typeOf Infinity returns number.

```
typeof Infinity;     // returns "number"
```

NaN - Not a Number

NaN is a JavaScript reserved word indicating that a value is not a number. Trying to do arithmetic with a non-numeric string will result in NaN (Not a Number):

You can use the global JavaScript function isNaN() to find out if a value is a number.

```
var x = 100 / "Apple";

isNaN(x);// returns true because x is Not a Number
```

Numbers Can be Objects

Normally JavaScript numbers are primitive values created from literals:

```
var x = 123;
```

But numbers can also be defined as objects with the keyword new:

```
var y = new Number (123)
```

When using the == equality operator, equal numbers look equal:

```
var x = 500;

var y = new Number (500); // (x == y) is true because x and y have equal values
```

When using the === equality operator, equal numbers are not equal, because the === operator expects equality in both type and value.

```
var x = 500;

var y = new Number (500); // (x === y) is false because x and y have different types
```

Objects cannot be compared:

```
var x = new Number (500);

var y = new Number (500);

// (x == y) is false because objects cannot be compared
```

Number Properties and Methods

Primitive values (like 3.14 or 2014), cannot have properties and methods (because they are not objects). But, with JavaScript, methods and properties are also available to primitive values, because JavaScript treats primitive values as objects when executing methods and properties.

69. Topic 69 – JavaScript Numbers Methods

Number Methods

JavaScript number methods are methods that can be used on numbers. In JavaScript, all number methods return a new value. They do not change the original variable. In the table below, you can see some number methods with their appropriate description:

Method	Description
toString()	Returns a number as a string
toExponential()	Returns a string, with a number rounded and written using exponential notation.
toFixed()	Returns a string, with a number rounded and written with a specified number of decimals.
toPrecision()	Returns a string, with a number written with a specified length
valueOf()	Returns a number as a number

Converting Variables to Numbers

- Number()
- parseInt()
- parseFloat()

These methods are not number methods, but global JavaScript methods. Let's illustrate them briefly:

Number ()

Number (), can be used to convert JavaScript variables to numbers. See some examples below:

```
x = true; Number(x);        // returns 1

x = false; Number(x);       // returns 0

x = new Date(); Number(x);          // returns 1404568027739

x = "10"; Number(x);        // returns 10

x = "10 20"; Number(x); // returns NaN
```

parseInt()

parseInt() parses a string and returns a whole number. Spaces are allowed. Only the first number is returned. If the number cannot be converted, NaN (Not a Number) is returned.

- parseInt("10"); // returns 10
- parseInt("10.33"); // returns 10
- parseInt("10 20 30"); // returns 10
- parseInt("10 years"); // returns 10
- parseInt("years 10"); // returns NaN

parseFloat()

parseFloat() parses a string and returns a number. Spaces are allowed. Only the first number is returned. If the number cannot be converted, NaN (Not a Number) is returned.

- parseFloat("10");   // returns 10
- parseFloat("10.33");   // returns 10.33
- parseFloat("10 20 30"); // returns 10
- parseFloat("10 years"); // returns 10
- parseFloat("years 10"); // returns NaN

JavaScript Math

70. Topic 70 – JavaScript Math

To understand the JavaScript Math, we have to evaluate some basic mathematical objects in programming.

The Math Object

The Math object allows you to perform mathematical tasks. It includes several mathematical methods. Check out the example below:

```
Math.random();    // returns a random number
```

Math.min() and Math.max() can be used to find the lowest or highest value in a list of arguments:

```
Math.min(0, 150, 30, 20, -8);        // returns -8
```

You can find below some meth objects:

- **Math.random()**

Math.random() returns a random number between 0 and 1 Math.random();

- **Math.round()**

Math.round() rounds a number to the nearest integer:

```
Math.round(4.7); // returns 5

Math.round(4.4); // returns 4
```

- **Math.ceil()**

Math.ceil() rounds a number up to the nearest integer:

```
Math.ceil(4.4); // returns 5
```

- **Math.floor()**

Math.floor() rounds a number down to the nearest integer:

```
Math.floor(4.7); // returns 4
```

- **MathConstants**

JavaScript provides 8 mathematical constants that can be accessed with the Math object:

```
Math.E;    // returns Euler's number

Math.PI    // returns PI

Math.SQRT2                  // returns the square root of 2

Math.SQRT1_2 // returns the square root of 1/2

Math.LN2                    // returns the natural logarithm of 2

Math.LN10                   // returns the natural logarithm of 10

Math.LOG2E       // returns base 2 logarithm of E
```

Math.LOG10E // returns base 10 logarithm of E

Furthermore, you can check online several Math Object Methods

71. Topic 71 – JavaScript Date - a

In JavaScript, The Date object lets you work with dates (years, months, days, hours, minutes, seconds, and milliseconds). Let's discuss how to deal with date object in JavaScript:

JavaScript DateFormats:

A JavaScript date can be written as a string:

Wed Apr 15 2015 23:17:51 GMT-0700 (Pacific Daylight Time)

Or as a number:

1429165071958

Dates written as numbers, specifies the number of milliseconds since January 1, 1970, 00:00:00.

Displaying Dates

we use a script to display dates inside a <p> element with id="demo":

```
<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = Date();

</script>
```

Creating Date Objects

The Date object lets us work with dates. A date consists of a year, a month, a day, an hour, a minute, a second, and milliseconds.

Date objects are created with the new Date() constructor. There are following ways of initiating a date:

```
new Date()

new Date(milliseconds)

new Date(dateString) // var d = new Date("October 13, 2014 11:13:00");

new Date(year, month, day, hours, minutes, seconds, milliseconds)
```

- Using new Date(number), creates a new date object as zero time plus the number. Zero time is 01 January 1970 00:00:00 UTC. The number is specified in milliseconds:

```
<script>
```

```
Var d = new Date(86400000);
document.getElementById("demo").innerHTML = d;
</script>
```

- Using new Date(7 numbers), creates a new date object with the specified date and time:

The 7 numbers specify the year, month, day, hour, minute, second, and millisecond, in that order:

```
<script>
var d = new Date(99,5,24,11,33,30,0);
document.getElementById("demo").innerHTML = d;
</script>
```

- Variants of the example above let us omit any of the last 4 parameters:

```
<script>
var d = new Date(99,5,24);
document.getElementById("demo").innerHTML = d;
</script>
```

Date Methods

When a Date object is created, a number of methods allow you to operate on it. Date methods allow you to get and set the year, month, day, hour, minute, second, and millisecond of objects, using either local time or UTC (universal, or GMT) time. When you display a date object in HTML, it is automatically converted to a string, with the toString() method.

```
<p id="demo"></p>
<script>
d = new Date(); document.getElementById("demo").innerHTML = d.toString();
</script>
```

- The toUTCString() method converts a date to a UTC string (a date display standard).

```
<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.toUTCString();
</script>
```

- The toDateString() method converts a date to a more readable format:

```
<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.toDateString();
</script>
```

72. Topic 72 – JavaScript Date - b

Date Formats

There are generally three types of valid JavaScript date formats:

- ISO Dates
- Long Dates
- Short Dates

Let's discuss each format in detail below:

ISO Date Format: The ISO 8601 syntax (YYYY-MM-DD) is the newest (and preferred) JavaScript date format.

```
var d = new Date("2015-03-25"); // YYYY-MM-DD
var d = new Date("2015-03"); // YYYY-MM
var d = new Date("2015"); // YYYY
var d = new Date("2015-03-25T12:00:00");
```

The T in the date string, between the date and time, indicates UTC time

Long Date Format: Long dates are most often written with a "MMM DD YYYY" syntax.

```
var d = new Date("Mar 25 2015");
```

year, month, and day can be in any order:

```
var d = new Date("25 Mar 2015")
```

And, month can be written in full (January), or abbreviated (Jan). Commas are ignored. Names are case insensitive

Short Date Format: Short dates are most often written with an "MM/DD/YYYY" syntax. Either "/" or "-" can be used as a separator.

```
var d = new Date("03/25/2015");
var d = new Date("03-25-2015");
```

JavaScript will also accept "YYYY/MM/DD"

Month is written before day in all Short date and ISO date formats.

Date Get Methods

Get methods are used for getting a part of a date. In the following table, you can see some methods with description:

Method	Description
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)

JavaScript Date Set Methods

Set methods are used for setting a part of a date. In the following table, some date set methods are given:

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)

Date Get/Set Methods

Let’s use Set and Get methods to add some days to a date object. See the example below: Let’s add 4 days to current date.

```
<script>
var d = new Date(); d.setDate( d.getDate() + 4 );
document.getElementById("demo").innerHTML = d;
</script>
```

String to Date

If you have a valid date string, you can use the Date.parse() method to convert it to milliseconds.

Date.parse() returns the number of milliseconds between the date and January 1, 1970. See a script below:

```
<script>
var msec = Date.parse("March 21, 2012");
var d = new Date(msec);
document.getElementById("demo").innerHTML = d;
</script>
```

Date Comparison

Date objects can easily be compared using standard comparison operators. See a detailed example below:

```
var today, anotherday, result;
today = new Date();
anotherday = new Date("03/25/2016");
if (anotherday > today) {
result = "Today is before March 25, 2016.“;
} else {
result = "Today is after March 25, 2016.“;
}
```

73. Topic 73 – JavaScript Arrays

What is an Array?

An array is a special variable, which can hold more than one value at a time. If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
var car1 = "Toyota";
```

var car2 = "Honda";

var car3 = "BMW";

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

JavaScript Arrays

In JavaScript, arrays are used to store multiple values in a single variable. In the example below, we will use a script to display arrays inside a element with id="demo":

<script>

Var cars = ["Toyota", "Honda", "BMW"]; document.getElementById("demo").innerHTML = cars;

</script>

Creating an Array

using an array literal is the easiest way to create a JavaScript Array.

Syntax:

var array-name = [item1, item2, ...];

See an example here:

var cars = ["Toyota", "Honda", "BMW"];

Using the JavaScript Keyword new:

The following example also creates an Array, and assigns values to it:

var cars = new Array("Toyota", "Honda", "BMW");

Access the Elements of an Array:

You refer to an array element by referring to the index number. This statement accesses the value of the first element in cars:

var name = cars[0];

This statement modifies the first element in cars:

cars[0] = “Audi";

Access the Elements of an Array

JavaScript variables can be objects. Arrays are special kinds of objects. Because of this, you can have variables of different types in the same Array. You can have objects in an Array. You can have functions in an Array. You can have arrays in an Array:

```
myArray[0] = Date.now;

myArray[1] = myFunction;

myArray[2] = myCars;
```

Arrays are Objects:

Arrays are a special type of objects. The typeof operator in JavaScript returns "object" for arrays. But, JavaScript arrays are best described as arrays.

Arrays:

```
var person = [“Adil", “Saeed", 46];
```

Arrays use numbers to access its "elements". In this example, person[0] returns John.

Object:

```
var person = {firstName:“Adil", lastName:“Saeed", age:46};
```

Objects use names to access its "members". In this example, person.firstName returns Adil.

Array Properties and Methods

The real strength of JavaScript arrays are the built-in array properties and methods:

```
var x = cars.length;

var y = cars.sort;
```

The length Property

The length property of an array returns the length of an array (the number of array elements).

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.length; // returns 4
```

Adding Array Elements

The easiest way to add a new element to an array is to use the length property:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits[fruits.length] = "Lemon"; // adds a new element (Lemon) to fruits
```

Adding elements with high indexes can create undefined "holes" in an array.

Looping Array Elements

The best way to loop through an array, is using a "for" loop: var index;

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

for (index = 0; index < fruits.length; index++) {

text += fruits[index];

}
```

74. Topic 74 – JavaScript Arrays (Cont…)

Many programming languages support arrays with named indexes. Arrays with named indexes are called associative arrays (or hashes). JavaScript does not support arrays with named indexes. In JavaScript, arrays use numbered indexes. See the example below:

```
var person = []; person[0] = "Adil"; person[1] = "Anwar";

person[2] = 46;

var x = person.length;

// person.length will return 3 var y = person[0];

// person[0] will return "Adil"
```

What is the Difference between Arrays and Objects?

In JavaScript:

- Arrays use numbered indexes.
- Objects use named indexes.

When to Use Arrays? When to use Objects?

- JavaScript does not support associative arrays.
- You should use objects when you want the element names to be strings (text).
- You should use arrays when you want the element names to be numbers.

Avoid new Array()

There is no need to use the JavaScript's built-in array constructor new Array(). Use [] instead.

```
var points = new Array();  // Bad

var points = []; // Good
```

These two different statements both create a new array containing 6 numbers:

```
var points = new Array(40, 100, 1, 5, 25, 10) // Bad

var points = [40, 100, 1, 5, 25, 10];          // Good
```

The new keyword complicates your code and produces nasty side effects.

How to Recognize an Array?

How do I know if a variable is an array?

The problem is that the JavaScript operator typeof returns "object"

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

typeof fruits; // typeof returns object
```

The typeof operator returns object because a JavaScript array is an object. To solve this problem you can create isArray() function:

```
function isArray(myArray)

{

return myArray.constructor.toString().indexOf("Array") > -1;

}
```

75. Topic 75 –Array Methods

The strength of JavaScript arrays lies in the array methods. Let's discuss JavaScript array methods in details:

- **Converting Arrays to Strings**

In JavaScript, all objects have the valueOf() and toString() methods.

The valueOf() method is the default behavior for an array. It returns an array as a string. See an example here:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"]; document.getElementById("demo").innerHTML = fruits.valueOf();
```

Output

The valueOf() method returns an array as a comma separated string.

Banana,Orange,Apple,Mango

- **Popping and Pushing**

When you work with arrays, it is easy to remove elements and add new elements. Popping items out of an array, or pushing items into an array:

Popping

The pop() method removes the last element from an array:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.pop(); // Removes the last element ("Mango") from fruits
```

Pushing

The push() method adds a new element to an array (at the end):

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.push("Kiwi"); // Adds a new element ("Kiwi") to fruits
```

- **Shifting Elements**

Shifting is equivalent to popping, working on the first element instead of the last. The shift() method removes the first element of an array, and "shifts" all other elements one place up. See an example here:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.shift(); // Removes the first element "Banana" from fruits
```

- **Changing Elements**

Array elements are accessed using their index number. Lets discuss with detailed example:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits[0] = "Kiwi";

// Changes the first element of fruits to "Kiwi“

// ["Kiwi", "Orange", "Apple", "Mango"]

fruits[fruits.length] = "Kiwi";

// Appends "Kiwi" to fruit

// ["Banana", "Orange", "Apple", "Mango", "Kiwi" ]
```

- **Deleting Elements**

Since JavaScript arrays are objects, elements can be deleted by using the JavaScript operator delete.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

delete fruits[0];

// Changes the first element in fruits to undefined

// [undefined, "Orange", "Apple", "Mango"]
```

- **Splicing an Array**

The splice() method can be used to add new items to an array. See this example:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.splice(2, 1, "Lemon", "Kiwi");
// ["Banana", "Orange", "Lemon", "Kiwi", "Mango"]
```

The first parameter (2) defines the position where new elements should be added.

The second parameter (1) defines how many elements should be removed.

The rest of the parameters ("Lemon" , "Kiwi") define the new elements to be added.

- **Sorting an Array**

The sort() method sorts an array alphabetically.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.sort();
// Sorts the elements of fruits
// ["Apple", "Banana", "Mango", "Orange"]
```

- **Reversing an Array**

The reverse() method reverses the elements in an array. You can use it to sort an array in descending order too.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.sort();
// Sorts the elements of fruits
// ["Apple", "Banana", "Mango", "Orange"]
fruits.reverse();
// Reverses the order of the elements
// ["Orange", "Mango", "Banana", "Apple"]
```

76. Topic 76 – JavaScript Array Methods

Sorting an Array

The sort() method sorts an array alphabetically.

```
var points = [40,100,5,25];
```

```
points.sort(); // [100,40,25,5]
```

The Compare Function

The purpose of the compare function is to define an alternative sort order.

The compare function should return a negative, zero, or positive value, depending on the arguments.

```
var points = [40, 100, 1, 5, 25, 10];

points.sort(function(a, b){return a-b});

function(a, b){return a-b}
```

When the sort() function compares two values, it sends the values to the compare function, and sorts the values according to the returned (negative, zero, positive) value.

```
function(a, b)

{

return a-b

}
```

When comparing 40 and 100, the sort() method calls the compare function(40,100).

The function calculates 40-100, and returns -60 (a negative value). The sort function will sort 40 as a value lower than 100.

Joining Arrays

The concat() method creates a new array by concatenating two arrays. Here is an example with more details:

```
var myGirls = ["Aisha", "Meryam"];

var myBoys = ["Bilal", "Umer" , "Ali"];

var myChildren = myGirls.concat(myBoys);

// Concatenates (joins) myGirls and myBoys arrays
```

Slicing an Array

The slice() method slices out a piece of an array into a new array.

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];

var citrus = fruits.slice(1, 3); // ["Orange", "Lemon"]

var citrus = fruits.slice(2); // ["Lemon", "Apple", "Mango"]
```

77. Topic 77 – JavaScript Comparison

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values. Major comparison operators are given in the table below:

For x = 5

Operator	Description	Comparing	Returns
==	equal to	x == 8	false
		x == 5	true
===	equal value and equal type	x === "5"	false
		x === 5	true
!=	not equal	x != 8	true
!==	not equal value or not equal type	x !== "5"	true
		x !== 5	false
>	greater than	x > 8	false
<	less than	x < 8	true
>=	greater than or equal to	x >= 8	false
<=	less than or equal to	x <= 8	true

How Can it be Used?

Comparison operators can be used in conditional statements to compare values and take action depending on the result:

```
if (age < 18) text = "Too young";
```

Logical Operators

Logical operators are used to determine the logic between variables or values. See the table below:

For x = 6 and y = 3

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x == 5 || y == 5) is false
!	not	!(x == y) is true

Conditional (Ternary) Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

variablename = (condition) ? value1:value2

Example

var vo = (age < 18) ? "Too young":"Old enough";

If the variable age is a value below 18, the value of the variable vo will be "Too young", otherwise the value of vo will be "Old enough “.

Comparing Different Types:

Comparing data of different types may give unexpected results.

When comparing a string with a numeric constant, JavaScript will treat the number as a string when doing the comparison. The result of this is commonly not the same as a numeric comparison.

When comparing two strings, "2" will be greater than "12", because (alphabetically) 1 is less than 2.

To secure a proper result, variables should be converted to the proper type before comparison. See an example below:

age = Number(age); if (isNaN(age)) {

vo = "Error in input";

} else {

vo = (age < 18) ? "Too young" : "Old enough";

}

JavaScript Bitwise Operators

Bit operators work on 32-bit numbers. Any numeric operand in the operation is converted into a 32-bit binary number. The result is converted back to a JavaScript number.

```
x = 5 & 1;
```

The result in x: 1

78. Topic 77 – JavaScript Condition

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- if: Use if to specify a block of code to be executed, if a specified condition is true
- else: Use else to specify a block of code to be executed, if the same condition is false
- else if : Use else if to specify a new condition to test, if the first condition is false
- switch: Use switch to specify many alternative blocks of code to be executed

The if Statement

Use the if statement to specify a block of JavaScript code to be executed if a condition is true.

```
if (condition) {
// if the condition is true
}
```

A detailed example is given here:

```
Make a "Good day" greeting if the hour is less than 18:00: if (hour < 18) {
greeting = "Good day";
}
```

The result of greeting will be:

Good day

The else Statement

Use the else statement to specify a block of code to be executed if the condition is false.

```
if (condition) {
// if the condition is true
} else {
```

```
// if the condition is false
}
```

See the following example:

If the hour is less than 18, create a "Good day" greeting, otherwise "Good evening":

```
if (hour < 18) {
greeting = "Good day";
} else {
greeting = "Good evening";
}
```

The else if Statement

Use the else if statement to specify a new condition if the first condition is false

Syntax

```
if (condition1) {
// if condition1 is true
} else if (condition2) {
// if the condition1 is false and condition2 is true
} else {
// if the condition1 is false and condition2 is false
}
```

See this example:

If time is less than 10:00, create a "Good morning" greeting, if not, but time is less than 20:00, create a "Good day" greeting, otherwise a "Good evening":

```
if (time < 10) {
greeting = "Good morning";
} else if (time < 20) { greeting = "Good day";
} else {
greeting = "Good evening";
```

}

79. Topic 79 – Switch Statement in JavaScript

What is a Switch Statement?

The switch statement is used to perform different actions based on different conditions. Use the switch statement to select one of many blocks of code to be executed. Check syntax here:

Syntax

switch(expression) {

case n1:

code block break;

case n2:

code block break;

default:

default code block

}

This is how it works:

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.

As in this example,

The getDay() method returns the weekday as a number between 0 and 6.

(Sunday=0, Monday=1, Tuesday=2 ..)

Use the weekday number to calculate current weekday name.

Here is another example:

switch (new Date().getDay()) {

case 0:

day = "Sunday";

break;

case 1:

day = "Monday";

break;

case 2:

day = "Tuesday";

break;

The break Keyword

When the JavaScript code interpreter reaches a break keyword, it breaks out of the switch block. This will stop the execution of more code and case testing inside the block.

The default Keyword

The default keyword specifies the code to run if there is no case match.

Common Code and Fall-Through

Sometimes, in a switch block, you will want different cases to use the same code, or fall-through to a common default.

- multiple cases can share same code block.
- default case does not have to be the last case.

Example:

switch (new Date().getDay()) {

case 1:

case 2:

case 3:

default:

text = “Working Day";

break;}

80. Topic 80 – JavaScript for Loop

JavaScript for Loop

Loops can execute a block of code a number of times. Loops are handy, if you want to run the same code over and over again, each time with a different value. Often this is the case when working with arrays:

Instead of writing:

text += cars[0] + "<br>";

```
text += cars[1] + "<br>";

text += cars[2] + "<br>";

text += cars[3] + "<br>";

text += cars[4] + "<br>";

text += cars[5] + "<br>";
```

You can write:

```
for (i = 0; i < cars.length; i++)

{ text += cars[i] + "<br>";

}
```

Different Kinds of Loops

Here are different kinds of loops used in this language:

1. for - loops through a block of code a number of times
2. for/in - loops through the properties of an object
3. while - loops through a block of code while a specified condition is true
4. do/while - also loops through a block of code while a specified condition is true

The For Loop

The for loop is often the tool you will use when you want to create a loop. Check the Syntax here:

Syntax

```
for (statement 1; statement 2; statement 3) {

// code block to be executed

}
```

Statement 1 is executed before the loop starts.

Statement 2 defines the condition for running the loop.

Statement 3 is executed each time after the loop.

```
for ( i = 0 ; i < 5 ; i++ ) {

text += "The number is " + i + "<br>";

}
```

Output

The number is 0

The number is 1

The number is 2

The number is 3

The number is 4

Statement 1

Normally you will use statement 1 to initiate the variable used in the loop (i = 0). This is not always the case, JavaScript doesn't care. Statement 1 is optional.

You can initiate many values in statement 1 (separated by comma)

```
for ( i = 0, len = cars.length, text = "“ ; i < len ; i++ ) {
text += cars[i] + "<br>";
}
```

Statement 2

Often statement 2 is used to evaluate the condition of the initial variable. This is not always the case, JavaScript doesn't care. Statement 2 is also optional. If statement 2 returns true, the loop will start over again, if it returns false, the loop will end.

Statement 3

Often statement 3 increases the initial variable.

This is not always the case, JavaScript doesn't care, and statement 3 is optional.

Statement 3 can do anything like negative increment (i--), positive increment (i = i + 15), or anything else. Statement 3 can also be omitted.

Check an example here:

```
var i = 0;
var len = cars.length; for (; i < len; ) {
text += cars[i] + "<br>"; i++;
}
```

The For/In Loop

The JavaScript for/in statement loops through the properties of an object. See an example here:

```
var person = {fname:"John", lname:"Doe", age:25};
var text = ""; var x;
```

```
for (x in person) { text += person[x];
}
```

Output

John Doe 25

81. Topic 81 – JavaScript While Loop

While Loop

Loops can execute a block of code as long as a specified condition is true. The while loop in JS loops through a block of code as long as a specified condition is true. Check out syntax here:

```
while (condition) {
// code block to be executed
}
```

In the following example, the code in the loop will run, over and over again, as long as a variable

(i) is less than 10:

```
while (i < 10) {
text += "The number is " + i; i++; }
```

Output

The	number	is	0
The	number	is	1
The	number	is	2
The	number	is	3
The	number	is	4
The	number	is	5
The	number	is	6
The	number	is	7
The	number	is	8
The	number	is	9

The Do/While Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

```
do {
// code block to be executed
```

```
}

while (condition);
```

The example below uses a do/while loop. The loop will always be executed at least once, even if the condition is false:

```
do {

text += "The number is " + i; i++;

}

while (i < 10);
```

Output

The	number	is	0
The	number	is	1
The	number	is	2
The	number	is	3
The	number	is	4
The	number	is	5
The	number	is	6
The	number	is	7
The	number	is	8
The	number	is	9

Comparing For and While

Note that a while loop is much the same as a for loop, with statement 1 and statement 3 omitted.

JavaScript Break & Continue

82. Topic 82 – JavaScript Break and Continue

The break statement "jumps out" of a loop. The continue statement "jumps over" one iteration in the loop.

The Break Statement

We already know that break can be used to "jump out" of a switch() statement. The break statement can also be used to jump out of a loop and proceed with code after loop code block.

```
for (i = 0; i < 10; i++) {

if (i === 3) {

break;

}}
```

```
text += "The number is " + i + "<br>";
}
```

Output

The number is 0

The number is 1

The number is 2

The Continue Statement

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This example skips the value of 3:

```
for (i = 0; i < 10; i++) {
if (i === 3) {
continue;
}
text += "The number is " + i + "<br>";
}
```

A loop which will skip the step where i = 3.

Output

The	number	is	0
The	number	is	1
The	number	is	2
The	number	is	3
The	number	is	4
The	number	is	5
The	number	is	6
The	number	is	7
The	number	is	8
The	number	is	9

83. Topic 83 – JavaScript Data Type

In JavaScript there are 5 different data types that can contain values:

- Number
- Boolean
- Object
- Function

There are 3 types of objects:

- Object
- Date
- Array

And 2 data types that cannot contain values:

- null
- undefined

The typeof Operator

You can use the typeof operator to find the data type of a JavaScript variable.

```
typeof "John”      // Returns string
typeof 3.14        // Returns number
typeof NaN         // Returns number
typeof false       // Returns boolean
typeof [1,2,3,4]   // Returns object
typeof {name:'John', age:34}        // Returns object
typeof new Date()// Returns object
typeof function () {}        // Returns function
typeof myCar       // Returns undefined (if myCar is not declared)
typeof null        // Returns object
```

- The data type of NaN is number
- The data type of an array is object
- The data type of a date is object
- The data type of null is object
- The data type of an undefined variable is undefined

The Data Type of typeof

The typeof operator is not a variable. It is an operator. Operators (+ - * /) do not have any data type. But, the typeof operator always returns a string containing the type of the operand.

The constructor Property

The constructor property returns the constructor function for all JavaScript variables. "John".constructor // Returns function String() { … }

(3.14).constructor // Returns function Number() { … }

false.constructor // Returns function Boolean() { … }

[1,2,3,4].constructor // Returns function Array() { … }

{name:'John', age:34}.constructor // Returns function Object() { … }

new Date().constructor // Returns function Date() { … }

function () {}.constructor // Returns function Function(){ … }

You can check the constructor property to find out if an object is of a crtain type (contains the word of your desired type):

```
function isArray(mArr) {
return mArr.constructor.toString().indexOf("Array") > -1;
}
```

84. Topic 84 – JavaScript Type Conversion

JavaScript variables can be converted to a new variable and another data type:

- By the use of a JavaScript function
- Automatically by JavaScript itself

Converting Numbers to Strings

The global method String() can convert numbers to strings. It can be used on any type of numbers, literals, variables, or expressions

```
String(x)
String(123)
String(100 +23)
```

The Number method toString() does the same. See an example below:

```
var x = 123;
x.toString();
(123).toString();
```

Method	Description
toExponential()	Returns a string, with a number rounded and written using exponential notation.
toFixed()	Returns a string, with a number rounded and written with a specified number of decimals.
toPrecision()	Returns a string, with a number written with a specified length

Converting Booleans to Strings

The global method String() can convert booleans to strings.

```
String(false)        // returns "false"
String(true)         // returns "true“
```

The Boolean method toString() does the same.

```
false.toString()     // returns "false"
true.toString()      // returns "true"
```

Converting Dates to Strings

The global method String() can convert dates to strings.

```
String(Date())
// returns Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe Daylight Time)
```

The Date method toString() does the same.

See an example here:

```
Date().toString()
// returns Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe Daylight Time)
```

Converting Strings to Numbers

The global method Number() can convert strings to numbers. Strings containing numbers (like "3.14") convert to numbers (like 3.14). Empty strings convert to 0.

Anything else converts to NaN (Not a number).

```
Number("3.14") // returns 3.14

Number(" ")        // returns 0 Number("")  // returns 0

Number("99 88") // returns NaNstrings convert to 0.
```

The Unary + Operator

The unary + operator can be used to convert a variable to a number:

```
var y = "5";  //y is a string

var x = + y; // x is a number
```

If the variable cannot be converted, it will still become a number, but with the value NaN (Not a number):

```
var y = "John"; // y is a string

var x = + y;       // x is a number (NaN)
```

Converting Booleans to Numbers

The global method Number() can also convert booleans to numbers.

```
Number(false)    // returns 0

Number(true)     // returns 1
```

Converting Dates to Numbers

The global method Number() can be used to convert dates to numbers.

```
d = new Date();

Number(d)
```

// returns something like 1404568027739, time in milliseconds but as number. The date method getTime() does the same.

Automatic Type Conversion

When JavaScript tries to operate on a "wrong" data type, it will try to convert the value to a "right" type.

The result is not always what you expect

```
5 + null // returns 5 because null is converted to 0

"5" + null // returns "5null" because null is converted to "null"

"5" + 1 // returns "51" because 1 is converted to "1"

"5" - 1      // returns 4 because "5" is converted to 5
```

Automatic String Conversion

JavaScript automatically calls the variable's toString() function when you try to "output" an object or a variable.

```
document.getElementById("demo").innerHTML = myVar;

// if myVar = {name:"Fjohn"} // toString converts to "[object Object]"

// if myVar = [1,2,3,4]   // toString converts to "1,2,3,4"

// if myVar = new Date() // toString converts to "Fri Jul 18 2014 09:08:55 GMT+0200"
```

Numbers and booleans are also converted, but this is not very visible:

```
// if myVar = 123 // toString converts to "123"

// if myVar = true // toString converts to "true"

// if myVar = false          // toString converts to "false"
```

For more details, you can search online for JavaScript Type Conversion Table, which shows the result of converting different JavaScript values to Number, String, and Boolean.

85. Topic 85 – JavaScript RegExp

JavaScript Regular Expressions

A regular expression is a sequence of characters that forms a search pattern. The search pattern can be used for text search and texts replace operations. When you search for data in a text, you can use this search pattern to describe what you are searching for.

A regular expression can be a single character, or a more complicated pattern. Regular expressions can be used to perform all types of text search and texts replace operations.

Syntax

```
/pattern/modifiers;
```

Example:

```
var patt = /text/i
```

Using String Methods

In JavaScript, regular expressions are often used with the two string methods: search() and replace(). The search() method uses an expression to search for a match, and returns the position of the match. The replace() method returns a modified string where the pattern is replaced.

Using String search() and replace() With String

The search and replace methods can also accept a string as search argument. The string argument will be converted to a regular expression.

Modifiers can be used to perform case-insensitive more global searches

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

Brackets are used to find a range of characters

Expression	**Description**
[abc]	Find any of the characters between the brackets
[0-9]	Find any of the digits between the brackets
(x|y)	Find any of the alternatives separated with |

Metacharacters are characters with a special meaning

Metacharacter	**Description**
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning or at the end of a word
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Quantifiers define quantities

Quantifier	Description
n+	Matches any string that contains at least one n
n*	Matches any string that contains zero or more occurrences of n
n?	Matches any string that contains zero or one occurrences of n

Using the RegExp Object

In JavaScript, the RegExp object is a regular expression object with predefined properties and methods.

RegExp test()

The test() method is a RegExp expression method. It searches a string for a pattern, and returns true or false, depending on the result.

The following example searches a string for the character "e":

```
var patt = /e/;

patt.test("The best things in life are free!");
```

Since there is an "e" in the string, the output of the code above will be: True

RegExp exec()

The exec() method is a RegExp expression method.

It searches a string for a specified pattern, and returns the found text. If no match is found, it returns null.

The following example searches a string for the character "e":

```
/e/.exec("The best things in life are free!");
```

Since there is an "e" in the string, the output of the code above will be: e

86. Topic 86 – JavaScript Hoisting

What is Hoisting?

Hoisting is JavaScript's default behavior of moving declarations to the top.

JavaScript Declarations are Hoisted

In JavaScript, a variable can be declared after it has been used. In other words; a variable can be used before it has been declared. Check this example:

```
x = 5; // Assign 5 to x

elem = document.getElementById("demo");

// Find an element elem.innerHTML = x;

// Display x in the element

var x; // Declare x
```

Here is another example:

```
var x; // Declare x

x = 5; // Assign 5 to x

elem = document.getElementById("demo");

// Find an element

elem.innerHTML = x;

// Display x in the element
```

To understand this, you have to understand the term "hoisting".

Hoisting is JavaScript's default behavior of moving all declarations to the top of the current scope (to the top of the current script or the current function).

JavaScript Initializations are Not Hoisted

JavaScript only hoists declarations, not initializations. See given example:

```
var x = 5; // Initialize x

var y = 7; // Initialize y

elem = document.getElementById("demo");

// Find an element elem.innerHTML = x + " " + y;

// Display x and y as 57
```

Declare Your Variables At the Top!

Hoisting is an unknown or overlooked behavior of JavaScript. If a developer doesn't understand hoisting, programs may contain bugs (errors). To avoid bugs, always declare all variables at the beginning of every scope.

87. Topic 87 – JavaScript Error Handling

JavaScript Errors - Throw and Try to Catch

- The **try** statement lets you test a block of code for errors.
- The **catch** statement lets you handle the error.
- The **throw** statement lets you create custom errors.
- The **finally** statement lets you execute code, after try and catch, regardless of the result.

Errors Will Happen!

When executing JavaScript code, different errors can occur. Errors can be coding errors made by the programmer, errors due to wrong input, and other unforeseeable things.

The JavaScript statements **try** and **catch** come in pairs:

```
try {
Block of code to try
}
catch(err) {
Block of code to handle errors
}
```

Try this example:

```
<script> try {
adddlert("Welcome guest!");
}
catch(err) {
document.getElementById("demo").innerHTML = err.message;
}
</script>
```

JavaScript Throws Errors

When an error occurs, JavaScript will normally stop, and generate an error message. The technical term for this is: JavaScript will throw an error.

The throw Statement

The **throw** statement allows you to create a custom error. The technical term for this is: throw an exception. The exception can be a JavaScript String, a Number, a Boolean or an Object.

```
throw "Too big";            // throw a text
throw 500;          // throw a number
```

If you use throw together with, try and catch, you can control program flow and generate custom error messages.

Input Validation Example

Let's create code to examine input. If the value is wrong, an exception (err) is thrown. The exception (err) is caught by the catch statement and a custom error message is displayed.

Let say x is an input from user. Expected to be a number between 5 and 10.

```
try {
if (x == "") throw "empty";
if (isNaN(x)) throw "not a number"; x = Number(x);
if(x < 5) throw "too low";
if(x > 10) throw "too high";
} catch(err) {
message = "Input is " + err;
}
```

The finally Statement

The **finally** statement lets you execute code, after try and catch, regardless of the result.

```
try {
Block of code to try
}
catch(err) {
Block of code to handle errors
}
finally {
Block of code to be executed regardless of the try / catch result
}
```

Error Handling

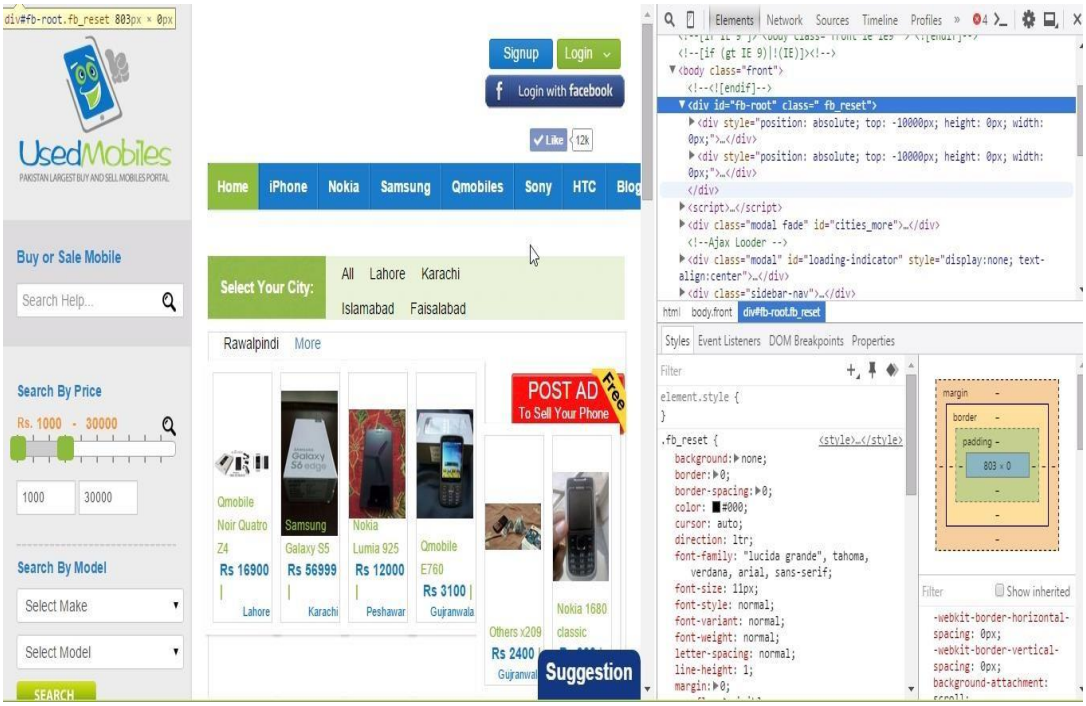
- try to execute a code
- throw exception based on desired condition
- catch exception/errors
- execute some code finally

88. Topic 88 – JavaScript Debugging

Your code might contain syntax errors, or logical errors, that are difficult to diagnose. So, It is difficult to write JavaScript code without a debugger. Often, when JavaScript code contains errors, nothing will happen. There are no error messages, and you will get no indications where to search for errors.

JavaScript Debuggers

Searching for errors in programming code is called code debugging. Debugging is not easy. But fortunately, all modern browsers have a built-in debugger. Built-in debuggers can be turned on and off, forcing errors to be reported to the user. To understand debugger more clearly, lets take a look to the image below:



With a debugger, you can also set breakpoints (places where code execution can be stopped), and examine variables while the code is executing.

The console.log() Method

If your browser supports debugging, you can use console.log() to display JavaScript values in the debugger window. For example:

```
<script>

a = 5;
```

```
b = 6;

c = a + b; console.log(c);

</script>
```

Setting Breakpoints

In the debugger window, you can set breakpoints in the JavaScript code. At each breakpoint, JavaScript will stop executing, and let you examine JavaScript values. After examining values, you can resume the execution of code (typically with a play button).

The debugger Keyword

The debugger keyword stops the execution of JavaScript, and calls (if available) the debugging function. This has the same function as setting a breakpoint in the debugger. If no debugging is available, the debugger statement has no effect. See the example below:

With the **debugger** turned on, this code will stop executing before it executes the third line.

```
var x = 15 * 5;

debugger;

document.getElementbyId("demo").innerHTML = x;
```

Major Browsers' Debugging Tools

Normally, you activate debugging in your browser with F12, and select "Console" in the debugger menu.

Google Chrome

- Open the browser.
- From the menu, select tools.
- From tools, choose developer tools.
- Finally, select Console.

Firefox Firebug

- Open the browser.
- Go to the web page: https://www.getfirebug.com
- Follow the instructions how to: install Firebug

Internet Explorer

- Open the browser.
- From the menu, select tools.
- From tools, choose developer tools.
- Finally, select Console.

- Open the browser.
- Go to the webpage: https://dev.opera.com
- Follow the instructions how to add a Developer Console button to your toolbar.

Safari Firebug

- Open the browser.
- Go to the webpage:
- Follow the instructions how to install Firebug Lite.

Safari Develop Menu

- Go to Safari, Preferences, Advanced in the main menu.
- Check "Enable Show Develop menu in menu bar".
- When the new option "Develop" appears in the menu: Choose "Show Error Console".

JavaScript Best Practices – a

89. Topic 89 – JavaScript Best Practice -a

Avoid Global Variables

Minimize the use of global variables. This includes all data types, objects, and functions. Global variables and functions can be overwritten by other scripts. Use local variables instead, and learn how to use closures.

Always Declare Local Variables

All variables used in a function should be declared as local variables. Local variables must be declared with the var keyword, otherwise they will become global variables.

Declarations on Top

It is a good coding practice to put all declarations at the top of each script or function.

This will:

- Give cleaner code
- Provide a single place to look for local variables
- Make it easier to avoid unwanted (implied) global variables
- Reduce the possibility of unwanted re-declarations

```
// Declare at the beginning

var firstName, lastName, price, discount, fullPrice;

// Use later firstName = "John";

lastName = "Doe"; price = 19.90;

discount = 0.10;
```

```
fullPrice = price * 100 / discount;
```

This also goes for loop variables:

```
// Declare at the beginning

var i;

// Use later

for (i = 0; i < 5; i++) {

….

}
```

Initialize Variables

It is a good coding practice to initialize variables when you declare them.

This will:

- Give cleaner code
- Provide a single place to initialize variables
- Avoid undefined values

Never Declare Number, String, or Boolean Objects

Always treat numbers, strings, or booleans as primitive values. Not as objects. Declaring these types as objects, slows down execution speed, and produces nasty side effects.

```
var x = "John";

var y = new String("John");

(x === y) // is false because x is a string and y is an object.
```

Or even worse:

```
var x = new String("John");

var y = new String("John");

(x == y) // is false because you cannot compare objects.
```

Don't Use new Object()

- Use {} instead of new Object()
- Use "" instead of new String()
- Use 0 instead of new Number()
- Use false instead of new Boolean()
- Use [] instead of new Array()

- Use /()/ instead of new RegExp()
- Use function (){} instead of new function()

Check this example:

```
var x1 = {};          // new object

var x2 = "";          // new primitive string

var x3 = 0;           // new primitive number

var x4 = false;       // new primitive boolean

var x5 = [];          // new array object

var x6 = / ()/;                 // new regexp object

var x7 = function () {};     // new function object
```

Beware of Automatic Type Conversions

Beware that numbers can accidentally be converted to strings or NaN (Not a Number). JavaScript is loosely typed. A variable can contain different data types, and a variable can change its data type. See the example below:

```
var x = "Hello";    // typeof x is a string

x = 5;       // changes typeof x to a number
```

When doing mathematical operations, JavaScript can convert numbers to strings.

```
var x= 5+7;

// x.valueOf() is 12, typeof x is a number

var x= 5+"7";

// x.valueOf() is 57, typeof x is a string

var x= "5"+7;

// x.valueOf() is 57, typeof x is a string

var x= 5-7;

// x.valueOf() is -2, typeof x is a number

var x= 5-"7";

// x.valueOf() is -2, typeof x is a number

var x= "5"-7;

// x.valueOf() is -2, typeof x is a number
```

```
var x= 5-"x";

// x.valueOf() is NaN, typeof x is a number
```

Beware of Automatic Type Conversions

Subtracting a string from a string, does not generate an error but returns NaN (Not a Number).

Example

```
"Hello" - "Dolly"  // returns NaN
```

90. Topic 90 – JavaScript Best Practice -b

Use === Comparison

The == comparison operator always converts (to matching types) before comparison. The === operator forces comparison of values and type.

```
0 == "";          // true

1 == "1";         // true

1 == true;        // true

0 === "";         // false

1 === "1";        // false

1 === true;       // false
```

Use Parameter Defaults

If a function is called with a missing argument, the value of the missing argument is set to undefined.

Undefined values can break your code. It is a good habit to assign default values to arguments.

```
function myFunction(x, y) {

if (y === undefined) {

y = 0;

}}
```

End Your Switches with Defaults

End Your Switches with Defaults. Even if you think its not needed. See the example below:

```
switch (new Date().getDay()) {

case 0:
```

```
day = "Sunday"; break;

case 1:

day = "Monday"; break;

case 2:

day = "Tuesday"; break;

case 3:

day = "Wednesday"; break;

case 4:

day = "Thursday"; break;

case 5:

day = "Friday"; break;

case 6:

day = "Saturday"; break;

default:

day = "Unknown";

}
```

Avoid Using eval()

The eval() function is used to run text as code. In almost all cases, it should not be necessary to use it. Because it allows arbitrary code to be run, it also represents a security problem.

91. Topic 91 – JavaScript Common Mistakes

Accidentally Using the Assignment Operator

javaScript programs may generate unexpected results if a programmer accidentally uses an assignment operator (=), instead of a comparison operator (==) in an if statement. See the examples below:

- This if statement returns false (as expected) because x is not equal to 10:

```
var x = 0;
if (x == 10)
```

- This if statement returns true (maybe not as expected), because 10 is true:

```
var x = 0;
if (x = 10)
```

- This if statement returns false (maybe not as expected), because 0 is false:

```
var x = 0;
```

```
if (x = 0)
```

Expecting Loose Comparison

In regular comparison, data type does not matter. This if statement returns true:

```
var x = 10;
var y = "10";
if (x == y)
```

In strict comparison, data type does matter. This if statement returns false:

```
var x = 10;
var y = "10";
if (x === y)
```

Switch Statement

It is a common mistake to forget that switch statements use strict comparison. This case switch will display an alert:

```
var x = 10; switch(x) {
case 10: alert("Hello");
}
```

This case switch will not display an alert:

```
var x = 10; switch(x) {
case "10": alert("Hello");
}
```

Confusing Addition & Concatenation

Addition is about adding numbers. Concatenation is about adding strings. In JavaScript both operations use the same + operator. When adding two variables, it can be difficult to anticipate the result:

```
var x = 10;
var y = 5;
var z = x + y;       // the result in z is 15
var x = 10;
var y = "5";
```

```
var z = x + y;        // the result in z is "105"
```

Misunderstanding Floats

All numbers in JavaScript are stored as 64-bits Floating point numbers (Floats). All programming languages, including JavaScript, have difficulties with precise floating-point values.

```
var x = 0.1;

var y = 0.2;

var z = x + y          // the result in z will not be 0.3

if (z == 0.3)         // this if test will fail
```

To solve the it helps to multiply and divide:

```
var z = (x * 10 + y * 10) / 10;        // x will be 0.3
```

Breaking a JavaScript String

JavaScript will allow you to break a statement into two lines:

```
var x = "Hello World!";
```

But, breaking a statement in the middle of a string will not work:

```
var x = "Hello World!";
```

You must use a "backslash" if you must break a string:

```
var x = "Hello \ World!";
```

Misplacing Semicolon

Because of a misplaced semicolon, this code block will execute regardless of the value of x:

```
if (x == 19);

{

// code block

}
```

Breaking a Return Statement

It is a default JavaScript behavior to try closing a statement automatically at the end of a line:

```
function myFunction(a) {

var power = 10;
```

```
return a * power;
}
```

Example:

```
function myFunction(a) {
var
power = 10; return a * power;
}
```

Example:

```
function myFunction(a) {
var power = 10; return
a * power;
}
```

The function will return undefined!

As parameter with return is optional and return itself is a complete statement.

92. Topic 92 – JavaScript Common Mistakes (Cont…)

Accessing Arrays with Named Indexes

Many programming languages support arrays with named indexes. Arrays with named indexes are called associative arrays (or hashes). JavaScript does not support arrays with named indexes. in JavaScript, arrays use numbered indexes:

Example:

```
var person = [];
person[0] = "John";
person[1] = "Doe";
person[2] = 46;
var x = person.length;      // person.length will return 3
var y = person[0];          // person[0] will return "John"
```

In JavaScript, objects use named indexes.

If you use a named index, when accessing an array, JavaScript will redefine the array to a standard object. After the automatic redefinition, array methods and properties will produce undefined or incorrect results.

Example:

```
var person = [];

person["firstName"] = "John";

person["lastName"] = "Doe";

person["age"] = 46;

var x = person.length; // person.length will return 0

var y = person[0]; // person[0] will return undefined
```

Ending an Array Definition with a Comma Incorrect:

```
points = [40, 100, 1, 5, 25, 10,];
```

Some JSON and JavaScript engines will fail, or behave unexpectedly.

Correct:

```
points = [40, 100, 1, 5, 25, 10];
```

Ending an Object Definition with a Comma

Incorrect:

```
person = {firstName:"John", lastName:"Doe", age:46,}
```

Some JSON and JavaScript engines will fail, or behave unexpectedly.

Correct:

```
person = {firstName:"John", lastName:"Doe", age:46}
```

Undefined is Not Null

With JavaScript, null is for objects, undefined is for variables, properties, and methods. To be null, an object has to be defined, otherwise it will be undefined. If you want to test if an object exists, this will throw an error if the object is undefined.

Incorrect:

```
if (myObj !== null && typeof myObj !== "undefined")
```

Because of this, you must test typeof() first:

Correct:

```
if (typeof myObj !== "undefined" && myObj !== null)
```

Expecting Block Level Scope

JavaScript does not create a new scope for each code block. It is true in many programming languages, but not true in JavaScript.

93. Topic 93 – JavaScript Performance

Reduce Activity in Loops

Loops are often used in programming.

Each statement in a loop, including the for statement, is executed for each iteration of the loop. Search for statements or assignments that can be placed outside the loop.

Bad Code:

```
for (i = 0; i < arr.length; i++) {
....
}
```

Better Code:

```
x = arr.length;
for (i = 0; i < x; i++) {
....
}
```

Reduce DOM Access

Accessing the HTML DOM is very slow, compared to other JavaScript statements. If you expect to access a DOM element several times, access it once, and use it as a local variable. See this example:

```
obj = document.getElementById("demo");
obj.innerHTML = "Hello";
....
obj.innerHTML = "Its done";
```

Reduce DOM Size

Keep the number of elements in the HTML DOM small. This will always improve page loading, and speed up rendering (page display), especially on smaller devices.

Avoid Unnecessary Variables

Don't create new variables if you don't plan to save values. Often you can replace code like this:

```
var fullName = firstName + " " + lastName;

document.getElementById("demo").innerHTML = fullName;
```

With this:

```
document.getElementById("demo").innerHTML = firstName + " " + lastName
```

Delay JavaScript Loading

Putting your scripts at the bottom of the page body, lets the browser load the page first.

While a script is downloading, the browser will not start any other downloads. In addition all parsing and rendering activity might be blocked.

An alternative is to use defer="true" in the script tag. The defer attribute specifies that the script should be executed after the page has finished parsing, but it only works for external scripts.

If possible, you can add your script to the page by code, after the page has loaded.

```
<script>

window.onload = downScripts;

function downScripts() {

var element = document.createElement("script");

element.src = "myScript.js";

document.body.appendChild(element);

}

</script>
```

Avoid Using with

Avoid using the **with** keyword. It has a negative effect on speed. It also clutters up JavaScript scopes. The with keyword is not allowed in strict mode.

94. Topic 94 – JavaScript Reserved Words

In JavaScript, some identifiers are reserved words and cannot be used as variables or function names.

JavaScript Standards

- All modern browsers fully support ECMAScript 3 (ES3, the third edition of JavaScript from 1999).
- ECMAScript 4 (ES4) was never adopted. ECMAScript 5 (ES5, released in Dec 2009).
- ECMAScript 6 (ES6) was released in June 2015, and is the latest official version of JavaScript.

Remember that, In JavaScript you cannot use these reserved words as variables, labels, or function names:

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

Java Reserved Words

JavaScript is often used together with Java. You should avoid using some Java objects and properties as JavaScript identifiers:

getClass	java	JavaArray
javaClass	JavaObject	JavaPackage

Windows Reserved Words

JavaScript can be used outside HTML. It can be used as the programming language in many other applications. In HTML you must (for portability you should) avoid using the name of HTML and Windows objects and properties.

HTML Event Handlers

In addition, you should avoid using the name of all HTML event handlers.

Nonstandard JavaScript

In addition to reserved words, there are also some nonstandard keywords used in some JavaScript implementations.

One example is the const keyword used to define variables. Some JavaScript engines will treat const as a synonym to var. Other engines will treat const as a definition for read-only variables.

Const is an extension to JavaScript. It is supported by the JavaScript engine used in Firefox and Chrome. But it is not a part of the JavaScript standards ES3 or ES5. Do not use it.

95. Topic 95 – JavaScript Global Function and Properties

The JavaScript global properties and functions can be used with all the built-in JavaScript objects.

- **Infinity**

  A numeric value that represents positive/negative infinity

  Infinity is displayed when a number exceeds the upper or lower limit of the floating point numbers.

- **NaN**

  The NaN property represents "Not-a-Number" value. This property indicates that a value is not a legal number.

- **undefined**

  Indicates that a variable has not been assigned a value.

- **eval()**

  Evaluates a string and executes it as if it was script code. If the argument is an expression, eval() evaluates the expression. If the argument is one or more JavaScript statements, eval() executes the statements. See the example below:

  ```
  var x = 10; var y = 20;

  var a = eval("x * y");      // 200

  var b = eval("2 + 2");      // 4

  var c = eval("x + 17");    // 27
  ```

- **eval()**

You can execute any JavaScript statement using eval()

Not recommended to use it unless you have no other choice.

96. Topic 96 – JavaScript Window Object

Window Object

The window object represents an open window in a browser. If a document contain frames (<iframe> tags), the browser creates one window object for the HTML document, and one additional window object for each frame. All global JavaScript objects, functions, and variables automatically become members of the window object.

There is no public standard that applies to the Window object, but all major browsers support it. see some examples below:

```
function myfunc() {
…
}
window.myfunc();
```

Or myfunc();

Example

```
window.document.getElementById("idname");
```

Is same as:

```
document.getElementById("idname");
```

Example

- window.open() - open a new window
- window.close() - close the current window
- window.moveTo() - move the current window
- window.resizeTo() - resize the current window

Properties and Functions in Windows Object

We have several properties and functions available in Windows object to perform different functions. Important ones are here:

- **setTimeOut()**

The setTimeout() method calls a function or evaluates an expression after a specified number of milliseconds.

Syntax

```
setTimeout(function,millisec,param1,param2,...);
```

Or

window.setTimeout(function,millisec,param1, param2,...);

Example

setTimeout(function(){ alert("Hello"); }, 3000);

Example

function myFunc() {

// do something;

…

// call again

setTimeout(myFunc, 3000);

}

myFunc();

Example

function myFunc() {

// do something;

… i++;

// call again based on condition

if (i < 10) {

setTimeout(myFunc, 3000);

}

}

- **clearTimeout()**

The clearTimeout() method clears a timer set with the setTimeout() method. The ID value returned by setTimeout() is used as the parameter for the clearTimeout() method. See the example given below:

var id = 0;

function myFunc() {

// do something;

```
…
// call again
id = setTimeout(myFunc, 3000);
}
clearTimeout(id);
```

- **setInterval()**

  The setInterval() method calls a function or evaluates an expression at specified intervals (in milliseconds).

```
function myFunc() {
// do something;
…
}
setInterval(myFunc, 3000);
```

- **clearInterval()**

  The setInterval() method will continue calling the function until clearInterval() is called, or the window is closed.

  The ID value returned by setInterval() is used as the parameter for the clearInterval() method. See the example below:

```
var id = 0;
function myFunc() {
// do something;
…
}
id = setInterval(myFunc, 3000);
clearInterval(id);
```

Window Object

There are also many other function and properties in Window Object that can be used as require.

- alert()
- close()
- scrollBy()
- scrollTo()

Go online to search more examples for windows object function and properties.

97. Topic 97 – JavaScript and Forms

JavaScript can access Forms using Document Object Model (DOM)

Accessing Forms

```
<form id="myForm" name="myForm">

<input type="text" name=" fieldname">

</form>

document.forms["myForm"]["fieldname"].value
```

Or

```
formObj = document.getElementById("myForm");

x = formObj.fieldname.value;
```

Form Validation

Data validation is the process of ensuring that input is clean, correct, and useful. Typical validation tasks are:

- has the user filled in all required fields?
- has the user entered a valid data?

Most often, the purpose of data validation is to ensure correct input to a computer application. Validation can be defined by many different methods, and deployed in many different ways.

- **Server-side validation** is performed by a web server, after input has been sent to the server.
- **Client-side validation** is performed by a web browser, before input is sent to a web server.

If a form field (fname) is empty, this function alerts a message, and returns false, to prevent the form from being submitted.

```
function validateForm() {

var x = document.forms["myForm"]["fname"].value;

if (x == null || x == "") {

alert("Name must be filled out");

return false;

}

}
```

98. Topic 98 – JavaScript Case Study

See here HTML Form Example

```
<form name="myForm" action="demo_form.asp" onsubmit="return
validateForm()" method="post">
Name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
```

HTML Form Validation

HTML form validation can be performed automatically by the browser.

If a form field (fname) is empty, the required attribute prevents this form from being submitted.

HTML Form Example

```
<form action="demo_form.asp" method="post">
<input type="text" name="fname" required>
<input type="submit" value="Submit">
</form>
```

HTML Constraint Validation

HTML5 introduced a new HTML validation concept called constraint validation. HTML constraint validation is based on:

- HTML Input Attributes
- CSS Pseudo Selectors
- DOM Properties and Methods

AJAX Introduction

99. Topic 99 – AJAX Introduction

AJAX: Asynchronous JavaScript and XML

AJAX stands for Asynchronous JavaScript and XML. AJAX is about sending and receiving data from a server without reloading the whole page. 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. You can see here some **Real-life Examples:**

- Gmail
- Facebook

- Youtube
- Google Maps

Nowadays many more web applications use AJAX to update content without reloading the complete page.

- AJAX is Based on Internet Standards

AJAX is based on internet standards, and uses a combination of:

- XMLHttpRequest object (to retrieve data from a web server)
- JavaScript/DOM (to display/use the data)

Google Suggest

AJAX was made popular in 2005 by Google, with Google Suggest. When you start typing in Google's search box, a JavaScript sends the letters off to a server and the server returns a list of suggestions.

- AJAX is based on existing standards.
- Allow you to update page without reload
- Allow improving webpage performance

100. Topic 100 – AJAX Sending Request - a

Sending a Request

The keystone of AJAX is the XMLHttpRequest object. The XMLHttpRequest object is used to exchange data with a server.

XMLHttpRequest Object

All modern browsers support the XMLHttpRequest object.

The XMLHttpRequest object is used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

XMLHttpRequest Object

All modern browsers (Chrome, IE7+, Firefox, Safari, and Opera) have a built-in XMLHttpRequest object.

Create an XMLHttpRequest Object

Syntax for creating an XMLHttpRequest object:

```
variable = new XMLHttpRequest();
```

Old versions of Internet Explorer (IE5 and IE6) use an ActiveX Object:

```
variable = new ActiveXObject("Microsoft.XMLHTTP");
```

See and example here:

```
if (window.XMLHttpRequest) {
```

```
xhttp = new XMLHttpRequest();

} else {

// code for IE6, IE5

xhttp = new ActiveXObject("Microsoft.XMLHTTP");}
```

Send a Request to Server

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object.

open()

- open(method, url, async)
- method: the type of request: GET or POST
- url: the server (file) location
- async: true (asynchronous) or false (synchronous)
- xhttp.open("GET", "filename.html", true);

send()

- send(): Sends the request to the server (used for GET)
- send(string): Sends the request to the server (used for POST)

GET or POST?

GET is simpler and faster than POST, and can be used in most cases.

Post is used when:

- A cached file is not an option (update a file or database on the server).
- Sending large amount of data to the server (POST has no size limitations).
- Sending user input (which can contain unknown characters), POST is more robust and secure than GET.

Here is an example:

A simple GET request:

```
xhttp.open("GET", "demo.php", true);

xhttp.send();
```

If you want to send information with the GET method, add the information to the URL:

```
xhttp.open("GET", "demo.php?fname=Asim&lname=Adeel", true);

xhttp.send();
```

POST Requests

```
xhttp.open("POST", "demo.php", true);

xhttp.send("fname=Henry&lname=Ford");
```

Request Header

- setRequestHeader(header, value)
- Adds HTTP headers to the request
- header: specifies the header name
- value: specifies the header value

POST Requests

A POST request:

```
xhttp.open("POST", "demo.php", true);

xhttp.setRequestHeader(

"Content-type",

"application/x-www-form-urlencoded");

xhttp.send("fname=Henry&lname=Ford");
```

The url - A File On a Server

The url parameter of the open() method, is an address to a file on a server.

```
xhttp.open("GET", "demo.php", true);
```

The file can be any kind of file, like .txt and .xml, or server scripting files like .asp and

.php (which can perform actions on the server before sending the response back).

101. Topic 101 – AJAX Sending Request - b

Asynchronous - True or False?

AJAX stands for Asynchronous JavaScript and XML, and for the XMLHttpRequest object to behave as AJAX, the async parameter of the open() method has to be set to true:

```
xhttp.open("GET", "info.php", true);
```

With AJAX, the JavaScript does not have to wait for the server response.

- execute other scripts while waiting for server response.
- deal with the response when the response ready.

Async=true

When using async=true, specify a function to execute when the response is ready in the onreadystatechange event. See the following example:

xhttp.onreadystatechange = function() {

if (xhttp.readyState == 4 && xhttp.status == 200) { document.getElementById("demo").innerHTML = xhttp.responseText;

}}

xhttp.open("GET", “info.txt", true); xhttp.send();

Async=false

To use async=false, change the third parameter in the open() method to false:

xhttp.open("GET", “info.txt", false);

JavaScript will NOT continue to execute, until the server response is ready. See the example given below:

xhttp.open("GET", “info.txt", false); xhttp.send();

document.getElementById("demo").innerHTML=xhttp.responseText;

102. Topic 102 – AJAX Receiving Response - a

Server Response

To get the response from a server, use the responseText or responseXML property of the XMLHttpRequest object.

AJAX - Server Response

responseText get the response data as a string responseXML get the response data as XML data

onreadystatechange event

When a request to a server is sent, we want to perform some actions based on the response. The onreadystatechange event is triggered every time the readyState changes.

onreadystatechange Property

onreadystatechange Stores a function (or the name of a function) to be called automatically each time the readyState property changes.

readyState Property

readyState holds the status of the XMLHttpRequest. Changes from 0 to 4:

0: request not initialized

1: server connection established 2: request received

3: processing request

4: request finished and response is ready

status Property

status 200: "OK"

404: Page not found

Ready State

When readyState is 4 and status is 200, the response is ready for us to read and process. See an example here:

```
var xhttp = new XMLHttpRequest(); xhttp.onreadystatechange = function() {
if (xhttp.readyState == 4 && xhttp.status == 200) {
document.getElementById("test").innerHTML = xhttp.responseText;}}
xhttp.send();
```

103. Topic 103 – AJAX Receiving Response - b

Using a Callback Function

A callback function is a function passed as a parameter to another function.

If you have more than one AJAX task on your website, you should create ONE standard function for creating the XMLHttpRequest object, and call this for each AJAX task. The function call should contain the URL and what to do on onreadystatechange (which is probably different for each call). Here is an example:

```
function myFunc(cFunc) {
var xhttp = new XMLHttpRequest(); xhttp.onreadystatechange = function() {
if (xhttp.readyState == 4 && xhttp.status == 200) { cFunc(xhttp);
}}}
```

The responseText Property

If the response from the server is not XML, use the responseText property. The responseText property returns the response as a string, and you can use it accordingly.

```
document.getElementById("demo").innerHTML = xhttp.responseText;
```

The responseXML Property

If the response from the server is XML, and you want to parse it as an XML object, use the responseXML property.

```
xmlDoc = xhttp.responseXML; txt = "";
x = xmlDoc.getElementsByTagName(“BookTitle"); for (i = 0; i < x.length; i++) {
```

```
txt += x[i].childNodes[0].nodeValue + "<br>";

}

document.getElementById("demo").innerHTML = txt;
```

104. Topic 104 – AJAX Case Study

The responseText Property

You have become familiarized with important methods and properties relevant to AJAX. You have been given code snippets to understand each one of them.

Now, you will be given an overview of working with AJAX using an example. This example shows an implementation of an autocomplete feature. A user types something in an input field and at that time an event is fired in the backend using AJAX. A list is displayed to the user based on what the user has typed so far, which contains words that can possibly be the ones the user wants to type.

You can understand the complete example in the video lesson. Some of the important components in the example code are given below:

- There is an HTML code with an <input> field and an empty to display autocomplete suggestions.
- loadDoc() is a function, which is invoked as result of oninput() event in the input field. An XMLHttpRequest object is created in this function and a request is sent with POST method. You will note that the keyword the user has typed in the input field is passed in send() to be sent to the server.
- When a response is received the loadCountries() function is called, which populates response text in the empty .
- A php file with the server-side logic shows how the data is retrieved from a database and a response is readied as a formatted unordered list <ul>.
- Each list <li> element in the response has an onclick() event associated with it, so that when the user selects any of the options, it displays the search box.

105. Topic 105 – jQuery Introduction

What is jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.

The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

Why jQuery is important?

There are lots of other JavaScript frameworks on web, but jQuery seems to be the most popular, and also the most extendable.

Key Value

Crossbrowser Compatible.

jQuery will run exactly the same in all major browsers.

Adding jQuery to HTML

- Download the jQuery library from jQuery.com
- Include jQuery from a CDN (Content Delivery Network), like Google

Downloading jQuery

Two versions of jQuery available:

- Production version - minified and compressed.
- Development version - for testing and development (uncompressed and readable code)

The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag:

```
<head>

<script type="text/javascript" src="jquery-1.11.3.min.js"></script>

</head>
```

- OR -

```
<head>

<script src="jquery-1.11.3.min.js"></script>

</head>
```

jQuery CDN

Both Google and Microsoft host jQuery. Faster Load Time due to cached copy.

Google CDN

```
<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

</head>
```

Microsoft CDN

<head>

<script src=""></script>

</head>

106. Topic 106 – jQuery Syntax

With jQuery you select (query) HTML elements and perform "actions" on them. The jQuery syntax is tailor made for selecting HTML elements and performing some action on the element(s).

Basic syntax is:

$(selector).action()

A $ sign to define/access jQuery

A (selector) to "query (or find)" HTML elements

A jQuery action() to be performed on the element(s)

Check some examples:

- $(this).hide() - hides the current element.
- $("p").hide() - hides all <p> elements.
- $(".test").hide() - hides all elements with class="test".
- $("#test").hide()- hides the element with id="test".

The Document Ready Event

To prevent any jQuery code from running before the document is finished loading. Make sure all jQuery methods/calls goes inside a document ready event like:

$(document).ready(function(){

// jQuery methods go here...

});

The Document Ready Event

It is good practice to wait for the document to be fully loaded and ready before working with it.

The Document Ready Event

The jQuery team has also created a shorter method for the document ready event:

$(function(){

// jQuery methods go here...

});

```
$(document).ready(function(){

// jQuery methods go here...

});
```

107. Topic 107 – jQuery Selectors

jQuery Selectors

jQuery selectors are one of the most important parts of the jQuery library. Selectors in jQuery allow you to select and manipulate HTML element(s). jQuery selectors are used to select HTML elements based on their id, classes, types, attributes, values of attributes.

All selectors in jQuery start with the dollar sign and parentheses.

```
$()
```

The element Selector

The jQuery element selector selects elements based on the element name. You can select all <p> elements on a page like this:

```
$("p")
```

When a user clicks on a button, all <p> elements will be hidden:

```
$(document).ready(function(){

$("button").click(function(){

$("p").hide();

});

});
```

The #id Selector

The jQuery #id selector uses the id attribute of an HTML tag to find the specific element. An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.

The #id Selector

To find an element with a specific id, write a hash character, followed by the id of the HTML element:

```
$("#test")
```

When a user clicks on a button, the element with id="test" will be hidden:

```
$(document).ready(function(){

$("button").click(function(){
```

```
$("#test").hide();
});
});
```

The .class Selector

The jQuery class selector finds elements with a specific class. To find elements with a specific class, write a period character, followed by the name of the class:

```
$(".test")
```

When a user clicks on a button, the elements with class="test" will be hidden:

```
$(document).ready(function(){
$("button").click(function(){
$(".test").hide();
});
});
```

Here are some more examples of jQuery Selectors

Syntax	Description
$("*")	Selects all elements
$(this)	Selects the current HTML element
$("p.intro")	Selects all <p> elements with class="intro"
$("p:first")	Selects the first <p> element
$("ul li:first")	Selects the first element of the first

$("ul li:first-child")	Selects the first element of every

Syntax	Description
$("[href]")	Selects all element with an href attribute
$("a[target='_blank']")	Selects all <a> elements with a target attribute value equal to" _blank".
$("a[target != _blank']")	Selects all <a> elements with a target attribute value NOT equal to" _blank".

Syntax	Description
$(":button ")	Selects all <button> elements and <input> elements of type=": button"
$("tr:even")	Selects all even <tr> elements
$("tr:odd")	Selects all odd <tr> elements

Also, you can search online for jQuery Selectors examples and details.

Functions In a Separate File

If your website contains a lot of pages, and you want your jQuery functions to be easy to maintain, you can put your jQuery functions in a separate .js file. See an example here:

```
<head>
<script src="">
</script>
<script src="my_jquery_functions.js"></script>
</head>
```

108. Topic 108 – jQuery Traverse

jQuery Traverse

109. Topic 109 – jQuery Events - a

jQuery is customized to respond to events in an HTML page.

What are Events?

All the different visitor's actions that a web page can respond to are called events. An event represents the precise moment when something happens.

- moving a mouse over an element
- selecting a radio button
- clicking on an element

Here are some common DOM events:

Mouse Events	Keyboard Events	Form Events	Document/window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

jQuery Syntax for Event Methods

In jQuery, most DOM events have an equivalent jQuery method. To assign a click event to all paragraphs on a page, you can do this:

```
$("p").click();
```

The next step is to define what should happen when the event fires. You must pass a function to the event:

```
$("p").click(function(){

// action goes here!!

});
```

110. Topic 110 – jQuery Events - b

Commonly Used jQuery Event Methods

Document ready()

The $(document).ready() method allows us to execute a function when the document is fully loaded.

```
$(document).ready();

$(document).ready(function(){

// action goes here

});
```

click()

The click() method attaches an event handler function to an HTML element. The function is executed when the user clicks on the HTML element. See the below example.

Lets hide the current p element, when a click event fires on a <p> element.

```
$("p").click(function(){
$(this).hide();
});
```

dblclick()

The dblclick() method attaches an event handler function to an HTML element. The function is executed when the user double-clicks on the HTML element. Understand this through example:

```
$("p").dblclick(function(){
$(this).hide();
});
```

mouseleave()

```
$("#p1").mouseleave(function(){
alert("Bye! You now leave p1!");
});
```

mousedown()

```
$("#p1").mousedown(function(){
alert("Mouse down over p1!");
});
```

mouseup()

```
$("#p1").mouseup(function(){
alert("Mouse up over p1!");
});
```

hover()

The hover() method takes two functions and is a combination of the mouseenter() and mouseleave() events.

```
$("#p1").hover(function(){
alert("You entered p1!");
},
function(){
alert("Bye! You now leave p1!");
});
```

focus()

The event fires when the form field gets focus.

```
$("input").focus(function(){
$(this).css("background-color", "#cccccc");
});
```

blur()

```
$("input").blur(function(){
$(this).css("background-color", "#ffffff");
});
```

The on() Method

The on() method attaches one or more event handlers for the selected elements.

Attach a click event to a <p> element:

```
$("p").on("click", function(){
$(this).hide();
});
```

We can attach multiple event handlers to an element using on().

```
$("p").on({
mouseenter: function(){
$(this).css("background-color", "lightgray");
},
mouseleave: function(){
```

```
$(this).css("background-color", "lightblue");
},
click: function(){
$(this).css("background-color", "yellow");
}
});
```

01- jQuery Effects – a

jQuery Effects Method

jQuery provide easy to use methods for creating animation effects. Here are some common jQuery Effects Methods:

Method	Description
fadeIn()	Fades in the selected elements
fadeOut()	Fades out the selected elements
fadeToggle()	Toggles between the fadeIn() and fadeOut() methods
hide()	Hides the selected elements

jQuery fadeIn() Method

The fadeIn() method gradually changes the opacity, for selected elements, from hidden to visible (fading effect). Hidden elements will not be displayed at all (no longer affects the layout of the page).

This method is often used together with the fadeOut() method.

Check here jQuery fadeIn() Method syntax:

```
$(selector).fadeIn(speed,easing,callback)
```

speed:

Optional. Specifies the speed of the fading effect. Default value is 400 milliseconds Possible values:

- Milliseconds
- "slow"
- "fast"

easing:

Optional. Specifies the speed of the element in different points of the animation. Default value is "swing"

Possible values:

- "swing" - slower at the start/end, but faster in the middle
- "linear" - moves in a constant speed

callback:

Optional. A function to be executed after the fadeIn() method is completed.

jQuery fadeOut() Method

The fadeOut() method gradually changes the opacity, for selected elements, from visible to hidden (fading effect).

Note: Hidden elements will not be displayed at all (no longer affects the layout of the page).

jQuery fadeOut() Syntax

```
$(selector).fadeOut(speed,easing,callback)
```

jQuery fadeToggle() Method

The fadeToggle() method fadeOut the selected elements if its visible or fadeIn if its not.

Syntax

```
$(selector).fadeToggle  (speed,easing,callback)
```

jQuery fadeTo() Method

The jQuery fadeTo() method allows fading to a given opacity.

Syntax

```
$(selector).fadeTo(speed,opacity,callback)
```

02- jQuery Effects – b

jQuery hide/show Methods

- hide()
- show()
- toggle()

jQuery hide() Method

The hide() method hides the selected elements.

Syntax

$(selector).hide(speed,callback)

jQuery show() Method

The show() method show the selected elements.

Syntax

$(selector).show(speed,callback)

jQuery toggle() Method

The toggle() method hides the selected elements.

Syntax

$(selector).toggle(speed,callback)

jQuery Sliding Methods: There are following jQuery Sliding Methods.

- slideDown()
- slideUp()
- slideToggle()

Get here more details for each method:

jQuery slideDown() Method

The slideDown() method slides down and show the selected elements.

Syntax

$(selector).slideDown(speed,callback)

jQuery slideUp() Method

The slideUp() method slide up and hide the selected elements.

Syntax

$(selector).slideUp(speed,callback)

jQuery slideToggle() Method

The slideToggle() method toggles between the slideDown() and slideUp() methods.

Syntax

$(selector).slideToggle(speed,callback)

03- jQuery Effects – c

jQuery Callback Function

A callback function is executed after the current effect is 100% finished. JavaScript statements are executed line by line. However, with effects, the next line of code can be run even though the effect is not finished. This can create errors.

To prevent this, you can create a callback function.

A callback function is executed after the current effect is finished.

jQuery Callback Function Syntax

$(selector).hide(speed,function(){ ... });

The example below has a callback parameter that is a function that will be executed after the hide effect is completed:

Example with Callback

```
$("button").click(function(){
$("p").hide("slow", function(){ alert("The paragraph is now hidden");
                    });
                    });
```

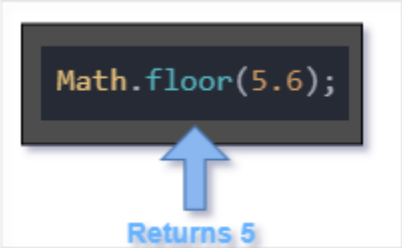
04- jQuery Effects – d

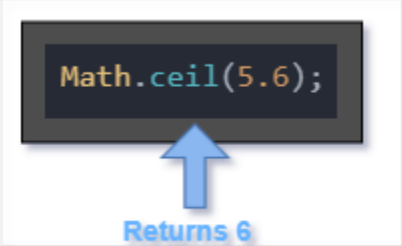
The jQuery Math Object

You may have observed that some new methods have been used in the example described in lesson# 113, such as floor() and random(). These methods are part of the jQuery Math object. The jQuery Math object is used to perform mathematical operations on numeric values. This object comes with a number of properties. You will have an overview of a few of them in this lesson.

Math floor() and ceil() methods

The floor() method *rounds down* a given number to its nearest integer. However, ceil() method *rounds up* a given



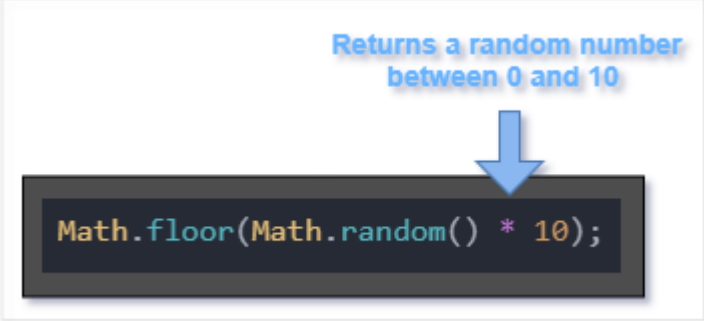


Math random() method

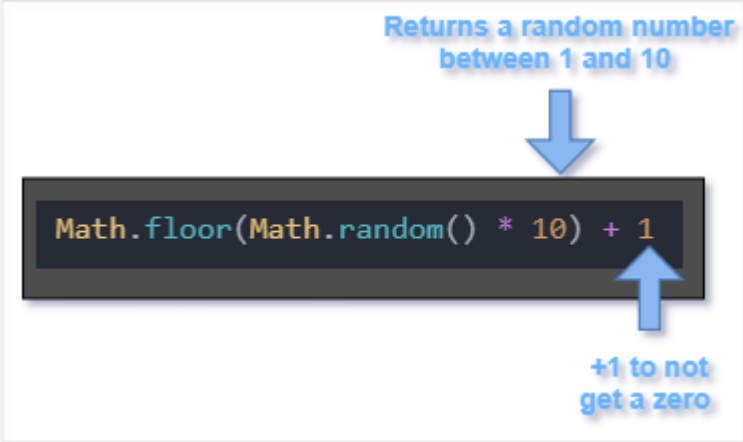
This method returns a random number between the specified range of 0 and 1. However, 0 is inclusive within the range while 1 is not.



If you want a random integer within a specified range other than 0 to 1, you can take product of the number returned by random() with the end range. Use floor() to round down that number. See the example.



If you want to specify a range starting from 1 but not 0 then you can write it as:


Returns a random number between 1 and 10
Math.floor(Math.random() * 10) + 1
+1 to not get a zero

05- jQuery Animate

jQuery Effects – Animation

The jQuery animate() method lets you create custom animations. Check out syntax here:

```
$(selector).animate({params},speed,callback);
```

All HTML elements have a static position, and cannot be moved. To animate, remember to first set the CSS position property of the element to relative, fixed, or absolute!

```
$("button").click(function(){
$("div").animate({left:  '500px'});
                    });
```

jQuery animate()

You can manipulate multiple properties at the same time to create an animation. See the example below:

```
$("button").click(function(){
$("div").animate({ left: '500px',
opacity: '0.5', height: '250px', width:
'250px'
                    });
                    });
```

It is also possible to define relative values. Just use += or -= in front of the value.

```
$("button").click(function(){
$("div").animate({ left: '500px', height: ' +=100px', width: ' +=100px'
                    });
                    });
```

jQuery animate() - Using Pre-defined Values

You can even specify a property's animation value as "show", "hide", or "toggle":

```
$("button").click(function(){
$("div").animate({ height: 'toggle'});
```

});

jQuery animate() - Queue Functionality

By default, jQuery comes with queue functionality for animations.

For multiple animate() calls, jQuery creates an "internal" queue with these method calls. Then it runs the animate calls ONE by ONE. See an example here:

```
$("button").click(function(){ var div = $("div");
div.animate({height: '300px', opacity: '0.4'}, "slow");
div.animate({width: '300px', opacity: '0.8'}, "slow");
div.animate({height: '100px', opacity: '0.4'}, "slow");
div.animate({width: '100px', opacity: '0.8'}, "slow");
});
```

jQuery stop() Method

The jQuery stop() method is used to stop an animation or effect before it is finished. The stop() method works for all jQuery effect functions, including sliding, fading and custom animations.

Syntax:

```
$(selector).stop(stopAll,goToEnd);
```

stopAll: For all queue animations,

default: false gotToEnd:

To complete current animation,

default: false

06- XML Elements

What is an XML Element?

An XML element is everything from (including) the element's start tag to (including) the element's end tag.

An element can contain other elements, text, or attributes.

```
<bookstore>
<book category="CHILDREN">
```

<title> Harry Potter </title>

<author> J K. Rowling </author>

<price> 29.99 </price>

</book>

<book category="WEB">

<title> Learning XML </title>

<author> Erik T. Ray </author>

<price> 39.95 </price>

</book>

</bookstore>

Empty XML Element?

<element></element>

Self closing: <element />

The two forms above produce identical results in an XML parser.

Empty elements do not have any content, but they can have the attribtues.

XML Naming Rules?

- Case-sensitive
- Must start with a letter or underscore
- Cannot start with the letters xml (or XML, or Xml, etc)
- Can contain letters, digits, hypoens, underscopres, and periods
- Cannot contain spaces

Any name can be used, no words are reserved (except xml).

Best Naming Practices?

- Create descriptive names, like this <person>, <firstname>, <lastname>.
- Create short and simple names, like this: <book_title>, Not like this: <the_title_of_the_book>.
- Avoid "-" , if you name something “first-name”, some software may think you want to subtract "name" from "first".
- Avoid " . " , if you name something “first.name”, some software may think you want to subtract "name" is property of the object "first".
- Avoid " : ", colons are reserved for namespaces.

- Non-English letters like eoa are perfectly legal in XML, but you may face problems if your software doesn't support them.
- XML documents often have a corresponding database. A good practice is to use the naming rules of your database for the elements in the XML documents.
- They are no naming styles defined for XML elements.

But it is good to know commonly used naming styles.

Style	Example	Description
Lower case	<firstname>	All letters lower case
Upper case	<FIRSTNAME>	All letters upper case
Underscore	<first_name>	Underscore separates words
Pascal case	<FirstName>	Uppercase first letter in each word
Camel Case	<firstName>	Uppercase first letter in each word except the first

Example

Let say we have a message xml, that our software use to handle messages:

<message>

<to> Kalsoom </to>

<from> Faisal </from>

<body> Did you read the book Beautiful Eyes? </body>

</message>

Let's add sine extra information to it:

<message>

<date> 11 Nov 2015 </date>

<to> Kalsoom </to>

<from> Faisal </from>

<body> Did you read the book Beautiful Eyes? </body>

XML is Extensibe

One of the beauties of XML is that it can be extended without breaking applications.

07- jQuery Chaining

You can chain together actions/methods in jQuery. Chaining allows us to run multiple jQuery methods (on the same element) within a single statement.

jQuery Method Chaining

To chain an action, you simply append the action to the previous action. The following example chains together the css(), slideUp(), and slideDown() methods. The "p1" element first changes to red, then it slides up, and then it slides down:

```
$("#p1").css("color","red").slideUp(2000).slideDown(2000);
```

When chaining, the line of code could become quite long. However, jQuery is not very strict on the syntax; you can format it like you want, including line breaks and indentations.

This also works just fine:

```
$("#p1").css("color", "red")
.slideUp(2000)
.slideDown(2000);
```

08- jQuery HTML -a

jQuery – Get/Set Content and Attributes

jQuery contains powerful methods for changing and manipulating HTML elements and attributes.

jQuery DOM Manipulation

One very important part of jQuery is the possibility to manipulate the DOM. jQuery comes with a bunch of DOM related methods that make it easy to access and manipulate elements and attributes.

DOM = Document Object Model

The DOM defines a standard for accessing HTML and XML documents.

```
<html>
<body>
<h1>Heading</h1>
<p>some text here
<ul><li>list element-1</li>
<li>list element-2</li>
</ul></p>
<p>some more text</p>
</body>
</html>
```

Getting/Setting Content

- **text()**
- **html()**
- **val()**

text()

html()

Sets or returns the content of selected elements (including HTML markup)

val()

Sets or returns the value of form fields

Get Example

```
$("#btn1").click(function() { alert("Text: " +
$("#test").text());
                    });
$("#btn2").click(function(){ alert("HTML: " +
$("#test").html());
                    });
```

Set Example

```
$("#btn1").click(function(){
$("#test1").text("Hello world!");
                    });
$("#btn2").click(function(){
$("#test2").html("<b>Hello world!</b>");
                    });
```

The jQuery **attr()** method is used to get and set attribute values. Check out example below:

```
<a id="link2" href="link-url">some link</a>
$("button").click(function(){
alert($("#link2").attr("href"));
                    });
```

Callback function

The functions text(), html(), val() and attr() all comes with a call back function.

The callback function has two parameters: index of the current element in the list of elements selected and the original value. See an example below:

```
$("#btn1").click(function(){
                    $("#test1").text(function(i, origText) {
                      return "Old: " + origText + " New: Good Work!";
```

```
    });
});
```

Example Modified

```
$("#btn1").click(function(){
$("#test1").text(function(i, origText)
{
if (origText=="nice")
{
return "very nice";
                    } else {
                      }
                            return
                            "make it
                            nice";

                });
                });
```

Callback function for attr()

attr() call back function works in similar way.

```
<a id="link2" href="link-url">some link</a>
$("button").click(function(){
$("#link2").attr("href", function(i, origValue){ return origValue + "/new";
                });
                });
<a id="link2" href="link-url/new">some link</a>
```

09- jQuery HTML - b

Adding New HTML Content

- append()
- prepend()
- after()

- before()

© Virtual University of Pakistan

append() Inserts content at the end of the selected elements

prepend() Inserts content at the beginning of the selected elements

after() Inserts content after the selected elements

before() Inserts content before the selected elements

 some paragraph text

 some paragraph text **append()**

prepend() some paragraph text

 some paragraph text **after()**

before() **<**p> some paragraph text

Example

```
$("p").append("Some appended text");
$("p").prepend("Some prepended text");
$("p").after("<p>Some text text</p>");
$("p").before("Some text before");
```

Removing Elements/Content

- **remove()** Removes the selected element (and its child elements).
- **empty()** Only removes the child elements and content from the selected element.

jQuery remove() Method

The jQuery remove() method removes the selected element(s) and its child elements.

```
$("#div1").remove();
$("#div1").empty();
```

Filter the Elements to be Removed

The jQuery remove() method also accepts one parameter, which allows you to filter the elements to be removed. The parameter can be any of the jQuery selector syntaxes.

This example removes all elements with class="test" and class="demo":

10- jQuery CSS - a

jQuery Manipulating CSS

jQuery has several methods for CSS manipulation.

- **addClass()** - Adds one or more classes to the selected elements
- **removeClass()** - Removes one or more classes from the selected elements
- **toggleClass()** - Toggles between adding/removing classes from the selected elements
- **css()** - Sets or returns the style attribute

Example Stylesheet

```
important {
font-weight: bold;
font-size: xx-large;
                    }
.blue {
color: blue;
                    }
```

addClass() Method

Adds one or more classes to one or more selected Element.

```
$("button").click(function(){
                      $("h1, h2, p").addClass("blue");
                     $("div").addClass("important");
                    });
```

You can also specify multiple classes within the addClass() method:

```
$("button").click(function(){
                $("#div1").addClass("important  blue");
                    });
```

removeClass() Method

Remove a specific class attribute from selected elements.

```
$("button").click(function(){
$("h1, h2, p").removeClass("blue");
                    });
```

toggleClass() Method

This method toggles between adding/removing classes from the selected elements.

```
$("button").click(function(){
$("h1, h2, p").toggleClass("blue");
                    });
```

11- jQuery CSS – b

The css() method sets or returns one or more style properties for the selected elements.

Return a CSS Property

To return the value of a specified CSS property, use the following syntax:

```
css("propertyname");
```

The following example will return the background-color value of the FIRST matched element:

```
$("p").css("background-color");
```

Set a CSS Property

You can also set a specified CSS property, using css method.

Set a CSS Property

To set a specified CSS property, use the following syntax:

```
css("propertyname","value");
```

The following example will set the background-color value for ALL matched elements:

```
$("p").css("background-color",  "yellow");
```

Set Multiple CSS Properties

You can also set multiple specified CSS properties using same css method.

To set multiple CSS properties, use the following syntax:

```
css({"propertyname":"value", "propertyname":"value",
"propertyname":"value",...});
```

The following example will set a background-color and a font-size for ALL matched elements:

```
$("p").css({"background-color": "yellow", "font-size": "200%"});
```

12- jQuery Dimension

With jQuery, it is easy to work with the dimensions of elements and browser window.

Here are jQuery Dimension Methods

- width()
- height()
- innerWidth()
- innerHeight()
- outerWidth()
- outerHeight()

jQuery width() and height() Methods

- The width() method sets or returns the width of an element (excludes padding, border and margin).
- The height() method sets or returns the height of an element (excludes padding, border and margin).

The following example returns the width and height of a specified element:

```
$("button").click(function(){ var txt = "";

txt += "Width: " + $("#div1").width() + "</br>";

txt += "Height: " + $("#div1").height();

$("#div1").html(txt);

                    });
```

jQuery innerWidth() and innerHeight() Methods

- The innerWidth() method returns the width of an element (includes padding).
- The innerHeight() method returns the height of an element (includes padding).

The following example returns the inner-width/height of a specified element:

```
$("button").click(function(){ var txt = "";

txt += "Inner width: " + $("#div1").innerWidth(); txt += " Inner height: " +

    $("#div1").innerHeight();

$("#div1").html(txt);

                    });
```

The outerWidth() method returns the width of an element (includes padding and border).

The outerHeight() method returns the height of an element (includes padding and border).

The following example returns the outer-width/height of a specified element:

```
$("button").click(function(){ var txt = "";
txt += "Outer width: " + $("#div1").outerWidth(); txt += " Outer height: " +
    $("#div1").outerHeight();
$("#div1").html(txt);
                        });
```

jQuery More width() and height()

We can also set width and height of a specified element.

The following example sets the width and height of a specified element:

```
$("button").click(function(){
$("#div1").width(500).height(500);
                    });
```

jQuery More width() and height()

Lets get the width and height of the document (the HTML document) and window (the browser viewport).

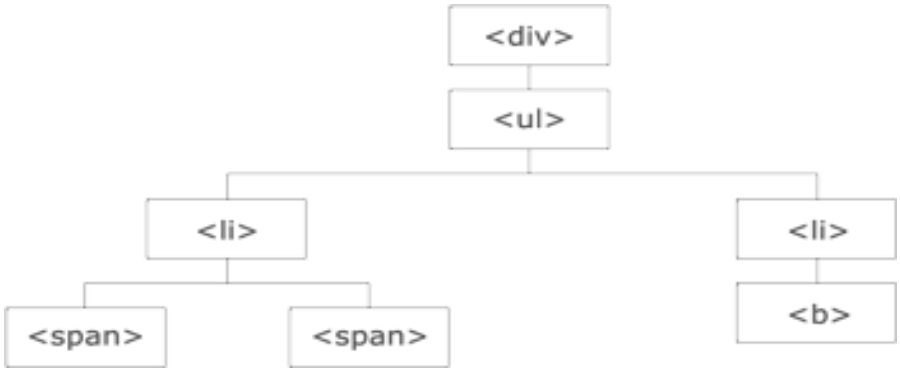
```
$("button").click(function(){
var txt = "";
txt += "Doc width/height: " + $(document).width(); txt += "x" + $(document).height() +
"\n";
txt += "Window width/height: " + $(window).width(); txt += "x" + $(window).height();
alert(txt);
                    });
```

13- jQuery Traversing - a

What is Traversing?

jQuery traversing, which means "move through", are used to "find" (or select) HTML elements based on their relation to other elements. Start with one selection and move through that selection until you reach the elements you desire.

Let's understand this with a a family tree.



<b>

child, parent, ancestor, descendants, siblings, current

Traversing the DOM

jQuery provides several methods that allows us to traverse the DOM. The largest category of traversal methods is tree-traversal.

14- jQuery Traversing - b

jQuery Traversing – Ancestors

An ancestor is a parent, grandparent, great-grandparent, and so on.

With jQuery you can traverse up the DOM tree to find ancestors of an element.

Traversing Up the DOM Tree

Three useful jQuery methods for traversing up the DOM tree are:

parent () parents () parentsUntil()

The parent () method returns the direct parent element of the selected element. This method only traverses a single level up the DOM tree.

The following example returns the direct parent element of each elements:

```
$(document). ready (function () {

$("span"). parent ();

                    });
```



jQuery parents() Method

The parents() method returns all ancestor elements of the selected element, all the way up to the document's root element (<html>).

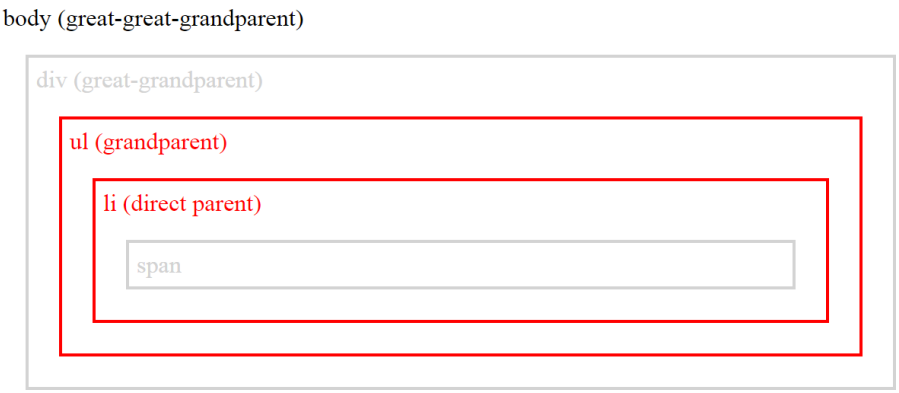
The following example returns all ancestors of all elements:

```
$(document).ready(function(){
$("span").parents();
});
```

jQuery parentsUntil() Method

The parentsUntil() method returns all ancestor elements between two given arguments. The following example returns all ancestor elements between a and a element:

```
$(document).ready(function(){
$("span").parentsUntil("div");
});
```



jQuery Traversing – Descendants

A descendant is a child, grandchild, great-grandchild, and so on. With jQuery you can traverse down the DOM tree to find descendants of an element.

Traversing Down the DOM Tree

Two useful jQuery methods for traversing down the DOM tree are:

- children()
- find()

jQuery children() Method

The children() method returns all direct children of the selected element. This method only traverse a single level down the DOM tree. The following example returns all elements that are direct children of each elements:

```
$(document).ready(function(){
$("div").children();
});
```

jQuery find() Method

The find() method returns descendant elements of the selected element, all the way down to the last descendant. The following example returns all elements that are descendants of :

```
$(document).ready(function(){
$("div").find("span");
});
```

15- jQuery Traversing - c

jQuery Traversing – Siblings

Siblings share the same parent. With jQuery you can traverse sideways in the DOM tree to find siblings of an element.

Traversing Sideways in The DOM Tree

There are many useful jQuery methods for traversing sideways in the DOM tree.

Traversing Sideways in The DOM Tree siblings()

- **next()**
- **nextAll()**
- **nextUntil()**
- **prev()**
- **prevAll()**
- **prevUntil()**

jQuery siblings() Method

The siblings() method returns all sibling elements of the selected element. The following example returns all sibling elements of <h2>:

```
$(document).ready(function(){
$("h2").siblings();
});
```

jQuery next() Method

The next() method returns the next sibling element of the selected element. The following example returns the next sibling of <h2>:

```
$(document).ready(function(){
$("h2").next();
});
```

jQuery nextAll() Method

The nextAll() method returns all next sibling elements of the selected element. The following example returns all next sibling elements of <h2>:

```
$(document).ready(function(){
$("h2").nextAll();
});
```

jQuery nextUntil() Method

The nextUntil() method returns all next sibling elements between two given arguments. The following example returns all sibling elements between a <h2> and a <h6> element:

```
$(document).ready(function(){
$("h2").nextUntil("h6");
});
```

jQuery prev(), prevAll() & prevUntil() Methods

The prev(), prevAll() and prevUntil() methods work just like the next(), nextAll(), nextUntil() methods but with reverse functionality, they return previous sibling elements.

16- jQuery AJAX - a

What is AJAX?

AJAX = Asynchronous JavaScript and XML.

In short; AJAX is about loading data in the background and display it on the webpage, without reloading the whole page.

jQuery and AJAX?

jQuery provides several methods for AJAX functionality. With the jQuery AJAX methods, you can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post.

jQuery load() Method

The jQuery load() method is a simple, but powerful AJAX method. The load() method loads data from a server and puts the returned data into the selected element. Check the syntaxes below:

$(selector).load(URL,data,callback);

URL: location of resource to load

data: querystring key/value pairs, optional

callback: name of callback function, optional

Here is another example:

$("#div1").load("resource_name.htm");

jQuery load() callback

The optional callback parameter specifies a callback function to run when the load() method is completed. The callback function can have different parameters.

- **responseTxt** - contains the resulting content if the call succeeds
- **statusTxt** - contains the status of the call
- **xhr** - contains the XMLHttpRequest object

jQuery load() Method

It is also possible to add a jQuery selector to the URL parameter. See these examples:

$("#div1").load("resource_name.htm #p1");

$("#div1").load("resource_name.htm #p1");

<h1>jQuery AJAX resource file</h1>

This is some text in a paragraph.

text for p1 paragraph.

text for p2 paragraph.

17- jQuery AJAX - b

HTTP Request: GET vs. POST

Two commonly used methods for a request-response between a client and server are: GET and POST.

- GET - Requests data from a specified resource
- POST - Submits data to be processed to a specified resource

jQuery $.get() Method

The $.get() method requests data from the server with an HTTP GET request.

jQuery $.get() Syntax

```
$.get(URL,callback);
```

The required URL parameter specifies the URL you wish to request.

The optional callback parameter is the name of a function to be executed if the request succeeds. The following example uses the $.get() method to retrieve data from a file on the server:

```
$("button").click(function(){
$.get("test.php", function(data, status){ alert("Data: " + data + "\nStatus: " + status);
                    });
                    });
```

jQuery $.post() Method

The $.post() method requests data from the server using an HTTP POST request.

jQuery $.post() Syntax

```
$.post(URL,data,callback);
```

The required URL parameter specifies the URL you wish to request. The optional data parameter specifies some data to send along with the request. The optional callback parameter is the name of a function to be executed if the request succeeds.

```
$("button").click(function(){
$.post("test_post.php",
                    {
name: "Asim Adeel", city: "Islamabad"
                    },
function(data, status){
alert("Data: " + data + "\nStatus: " + status);
                    });
                    });
```

18- jQuery noConflict

jQuery noConflict() Method

jQuery uses the $ sign as a shortcut for jQuery. There are many other popular JavaScript frameworks like: Angular, Backbone, Ember, Knockout, and more.

What if other JavaScript frameworks also use the $ sign as a shortcut?

If two different frameworks are using the same shortcut, one of them might stop working.

The noConflict() method releases the hold on the $ shortcut identifier, so that other scripts can use it. Lets understand **noConflict() Method with an example:**

```
$.noConflict(); jQuery(document).ready(function() {
jQuery("button").click(function() { jQuery("p").text("jQuery is still
working!");
                    });
                    });
```

Custom Shortcut

You can also create your own shortcut very easily. The noConflict() method returns a reference to jQuery, that you can save in a variable, for later use.

```
var jq = $.noConflict(); jq(document).ready(function(){

jq("button").click(function(){ jq("p").text("jQuery is still
working!");
                    });
                    });
```

Another Usage

To keep using the $ shortcut, you can pass the $ sign in as a parameter to the ready method. This allows you to access jQuery using $, inside this function - outside of it, you will have to use "jQuery“

```
$.noConflict();
jQuery(document).ready(function($){

$("button").click(function(){
$("p").text("jQuery is still working!");
                    });
```

});

19- jQuery CaseStudy

Lets take a real life example.

jQuery AutoComplete

Lets update out AJAX auto complete implementation to jQuery AJAX.

20- Extensible Markup Language

Introduction

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable. It is defined by the W3C's XML 1.0 Specification and by several other related specifications, all of which are free open standards. Here are some main features of XML:

- XML is heavily used as a format for document storage and processing, both online and offline.
- XML is extensible, because it only specifies the structural rules of tags. No specification on tags them self.
- It allows validation using schema languages such as XSD and Schematron, which makes effective unit-testing, firewalls, acceptance testing, contractual specification and software construction easier.
- The hierarchical structure is suitable for most (but not all) types of documents.
- It is platform-independent, thus relatively immune to changes in technology.
- XML files are text files, which can be managed by any text editor.
- XML is very simple, because it has less than 10 syntax rules.
- XML tags are not predefined. You must define your own tags
- XML can be used to create new internet languages
- XML is a markup language much like HTML
- XML was designed to describe data.
- XML is not a replacement for HTML

21- XML Usage

How Can XML be used?

XML language is used in many aspects of web development. XML is often used to separate data from presentation.

Separates Data from Presentation

XML does not carry any information about how to be displayed. Same XML data can be used in many different presentation scenarios.

Complement to HTML

In many HTML applications, XML is used to store or transport data, while HTML is used to format and display the same data. When displaying data in HTML, you should not have to edit the HTML file when the data changes. The data can be stored in separate XML files.

Transaction Data

Thousands of XML formats exist, in many different industries, to describe day-to-day data transactions.

- Stocks and Shares
- Financial transactions
- News information
- Weather services etc.

22- XML Tree

XML documents have a hierarchical structure and can conceptually be interpreted as a tree structure, called an XML tree. XML documents must contain a root element (one that is the parent of all other elements). All elements in an XML document can contain sub elements, text and attributes. The tree represented by an XML document starts at the root element and branches to the lowest level of elements.

XML Tree Structure

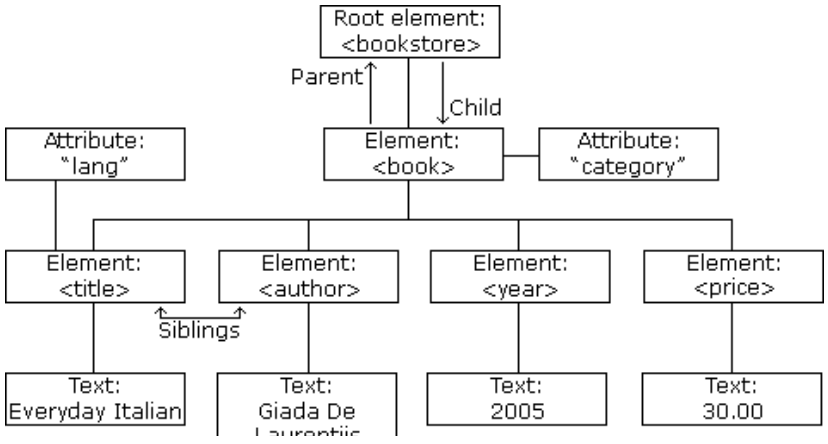
<root>

<child>

<subchild>. .....</subchild>

</child>

</root>



See this XML document example:

<?xml version="1.0" encoding="UTF-8"?>

<bookstore>

<book category="cooking">

<author>Giada De Laurentiis</author>

<title lang="en">Everyday Italian</title>

<price>30.00</price>

<year>2005</year>

</book>

<bookstore>

Self-Describing Syntax

XML uses a much self-describing syntax. A prolog defines the XML version and the character encoding:

<?xml version="1.0" encoding="UTF-8"?>

23- XML Syntax

XML Syntax Rules

The syntax rules of XML are very simple and logical. The rules are easy to learn, and easy to use.

Must Have a Root Element

XML documents must contain one root element that is the parent of all other elements.

<root>

<child>

<subchild>. ..... </subchild>

</child>

</root>

The XML Prolog

The XML prolog is optional. If it exists, it must come first in the document.

<?xml version="1.0" encoding="UTF-8"?>

UTF-8 is the default character encoding for XML documents. To avoid errors, you should specify the encoding used, or save your XML files as UTF-8.

All Elements Must Have a Closing Tag

In HTML, some elements do not have to have a closing tag. In XML, it is illegal to omit the closing tag. All elements must have a closing tag.

XML Tags are Case Sensitive

XML tags are case sensitive. The tag <Name> is different from the tag <name>. Opening and closing tags must be written with the same case.

<Message>Hi</message>

<message>Hi</Message>

<message>Hi</message>

XML Elements Must be Properly Nested

In HTML, you might see improperly nested elements:

bold and italic text

In XML, all elements must be properly nested within each other:

bold and italic text

XML Attribute Values Must be Quoted

XML elements can have attributes in name/value pairs just like in HTML. In XML, the attribute values must always be quoted. See the following example:

<note date=12/11/2015>

<to>Asim</to>

<from>Faisal</from>

</note>

<note date="12/11/2015">

<to>Asim</to>

<from>Faisal</from>

</note>

Entity References

Some characters have a special meaning in XML.

Like "<" inside an XML element, will generate an error. This will

generate an XML error:

<message>if salary < 1000 then</message>

To avoid this error, replace the "<" character with an entity reference:

<message>if salary < 1000 then</message> There are

5 pre-defined entity references in XML:

<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

Comments in XML

You can use Comments in XML. The syntax for writing comments in XML is similar to that of HTML.

<!-- This is a comment -->

White-space is preserved in XML

XML does not truncate multiple white-spaces in a document (while HTML truncates multiple white-spaces to one single white-space).

XML Stores New Line as LF

Windows applications store a new line as: carriage return and line feed (CR+LF). Unix

and Mac OSX uses LF.

Old Mac systems uses CR. XML stores a

new line as LF. **Well Formed XML**

XML documents that conform to the syntax rules are said to be "Well Formed" XML document.

24- HTML5 Canvas Gradients

Canvas Gradients

Shapes on the canvas are not limited to solid colors.

Gradients can be used to fill rectangles, circles, lines, text, etc.

Draw Gradients

There are two different types of gradients.

Linear

createLinearGradient(x,y,x1,y1)

Radial/Circular

createRadialGradient(x,y,x1,y1,r1)

once we have a gradient object, we must add two or more color stops.

addColorStop()

The addColorStop() method specifies the color stops, and its position along the gradient.

Gradient positions can be between 0 to 1.

addColorStop(pos,color);

Draw Linear Gradients

```
Var c = document.getElementById (“myCanvas");
Var ctx = c.getContext("2d");
//Create Gradient
Var grd = ctx.createLinearGradient(0,0,200,0);
Grd.addColorStop(0, "yellow");
Grd.addColorStop(1, "blue");
//Fill with gradient
Ctx.fillStyle=grd;
Ctx.fillRect(10,10,150,80);
```

Draw Circular Gradients

```
Var c = document.getElementById (“myCanvas");
Var ctx = c.getContext("2d");
//Create Gradient
Var grd = ctx.createRadialGradient(75,50,5,90,60,100);
Grd.addColorStop(0, "red");
Grd.addColorStop(1, "white");
//Fill with gradient
Ctx.fillStyle=grd;
Ctx.fillRect(0,0,200,80);
```

25- XML Attributes

XML elements can have attributes, just like HTML. Attributes are designed to contain data or information related to a specific element that may not be part of the data itself.

```
<bookstore>
        <book category="CHILDREN">
        <title>Harry Potter</title>
```

<author>J K. Rowling</author>

<price>29.99</price>

</book>

</bookstore>

XML Attributes Must be Quoted

Attribute values must always be quoted. Either single or double quotes can be used.

<person gender="female">
<person gender='female'>

See an example here:

<gangster name='George "Shotgun" Ziegler'>

you can use character entities too:

<gangster name="George "Shotgun" Ziegler">

XML Elements vs. Attributes

When to make a data as sub-element or attribute

<person gender="male">

<firstname>Asim</firstname>

<lastname>Adeel</lastname>

</person>

<person>

<gender>male</gender>

<firstname>Asim</firstname>

<lastname>Adeel</lastname>

</person>

There are no rules about when to use attributes or when to use elements. Better option is to use element to hold data. See the examples:

Example:

<person dateofbirth="11 Feb 1976">

<firstname>Asim</firstname>

<lastname>Adeel</lastname>

</person>

Example:

<person>

<dateofbirth>11 Feb 1976</dateofbirth>

<firstname>Asim</firstname>

<lastname>Adeel</lastname>

</person>

Example:

<person>

<dateofbirth>

<date>11</date>

<month>Feb</month>

<year>1976</year>

</dateofbirth>

<firstname>Asim</firstname>

<lastname>Adeel</lastname>

</person>

Avoid XML Attributes?

Problems using attributes:

- attributes cannot contain multiple values
- attributes cannot contain tree structures
- attributes are not easily expandable (for future changes)

XML Attributes for Metadata

Best usage of attributes is to use them to store Metadata about the element data.

<person **id="101"**>

<dateofbirth>11 Feb 1976</dateofbirth>

<firstname>Asim</firstname>

<lastname>Adeel</lastname>

</person>

The id attribute above is metadata.

26- XML Namespace

XML Namespaces provide a method to avoid element name conflicts.

Name Conflicts

In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.

This XML carries HTML table information:

<table>

<tr>

<td>Apples</td>

<td>Bananas</td>

</tr>

</table>

This XML carries information about a table (a piece of furniture):

<table>

<name>Coffee Table</name>

<width>80</width>

<length>120</length>

</table>

Name Conflicts

Both contain a <table> element, but the elements have different content and meaning. If we have to use these 2 xml in same application, it will raise a conflict.

Solution:Using a Prefix

Name conflicts in XML can easily be avoided using a name prefix.

<h:table>

<h:tr>

<h:td>Apples</h:td>

<h:td>Bananas</h:td>

</h:tr>

</h:table>

<f:table>

<f:name>Coffee Table</f:name>

<f:width>80</f:width>

<f:length>120</f:length>

XML Namespaces - The xmlns Attribute

When using prefixes in XML, a namespace for the prefix must be defined. The namespace is defined by the xmlns attribute in the start tag of an element.

XML Namespaces - The xmlns Attribute

The namespace declaration has the following syntax.

xmlns:prefix="URI"

```
<root>
<h:table xmlns:h="http://www.fruits.com/table">
<h:tr>
<h:td>Apples</h:td>
<h:td>Bananas</h:td>
</h:tr>
</h:table>
<f:table xmlns:f="http://www.furniture.com/table">
<f:name>Coffee Table</f:name>
<f:width>80</f:width>
<f:length>120</f:length>
</f:table>
</root>
```

When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace.

The namespace URI is not used by the parser to look up information. The purpose is to give the namespace a unique name. Many author use the namespace as a pointer to a web page containing namespace information.

Default Namespaces

Defining a default namespace for an element saves us from using prefixes in all the child elements. Check below the **Default Namespace Syntax:**

xmlns="namespaceURI"

Example:

```
<table xmlns="http://www.xyz.com/table">
<tr>
<td>Apples</td>
<td>Bananas</td>
</tr>
</table>
```

Namespaces in Real Use

XSLT is an XML language that can be used to transform XML documents into other formats, like HTML.

27- XML DTD

DTD stands for **Document Type Definition**. It defines legal building blocks of an XML document.

- An XML document with correct syntax is called "Well Formed".
- An XML document validated against a DTD is "Well Formed" and "Valid".

Valid XML Documents

A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a DTD. See this example:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE note SYSTEM "Note.dtd">

<note>

<to>Asim</to>

<from>Faisal</from>

<heading>Reminder</heading>

<body>Don't forget to send files!</body>

</note>
```

The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements.

Using DTD for Entity Declaration

A doctype declaration can also be used to define special characters and character strings, used in the document.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE note [

<!ENTITY nbsp " ">

<!ENTITY writer "Faisal Nisar">

<!ENTITY copyright "Copyright: Virtual University">]>

<note>

<to>Asim</to>

<from>Faisal</from>

<heading>Reminder</heading>

<body>Don't forget to send files!</body>

<footer>&writer; &copyright;</footer>

</note>
```

With a DTD, independent groups of people can agree on a standard for interchanging data. With a DTD, you can verify that the data you receive from an unknown source is valid.

28- XML Schema

XML Schema is an XML-based alternative to DTD.

- An XML Schema describes the structure of an XML document, just like a DTD.
- An XML document validated against an XML Schema is both "Well Formed" and "Valid".

See the following examples:

```
<xs:element name="note">
<xs:complexType>
<xs:sequence>
<xs:element name="to" type="xs:string"/>
<xs:element name="from" type="xs:string"/>
<xs:element name="heading" type="xs:string"/>
<xs:element name="body" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

Example

<xs:element name="note">

```
<xs:complexType>
<xs:sequence>
< xs:element name="to" type="xs:string"/>
.....
<xs:sequence>
<xs:complexType >
<xs:element>
```

<xs:element name="note"> defines the element called "note"

Example:

<xs:element name="note">

<xs:complexType>

<xs:sequence>

< xs:element name="to" type="xs:string"/>

……

<xs:sequence>

<xs:complexType >

<xs:element>

<xs:complexType > the "note" element is a complex type

XML Schemas are Powerful than DTD

- Written in XML
- Extensible to additions
- Support data types
- Support namespaces

Why XML Schema?

With XML Schema, your XML files can carry a description of its own format. Independent groups of people can agree on a standard for interchanging data.

You can verify data.

XML Schemas Support Data Types

```
<xs:element name=“books">
<xs:complexType>
<xs:sequence>
<xs:element name="title" type="xs:string"/>
<xs:element name=“author" type="xs:string"/>
<xs:element name="pub_date" type="xs:date"/>
<xs:element name="price" type="xs:decimal"/>
</xs:complexType>
</xs:sequence>
</xs:element>
```

XML Schemas Support Data Types

- It is easier to describe document content
- It is easier to define restrictions on data
- It is easier to validate the correctness of data
- It is easier to convert data between different data types

XML Schemas use XML Syntax

```
<xs:element name="note">
<xs:complexType>
<xs:sequence>
<xs:element name="to" type="xs:string"/>
<xs:element name="from" type="xs:string"/>
<xs:element name="heading" type="xs:string"/>
<xs:element name="body" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

XML Schemas use XML Syntax

- You don't have to learn a new language
- You can use your XML editor to edit your Schema files
- You can use your XML parser to parse your Schema files
- You can manipulate your Schemas with the XML DOM
- You can transform your Schemas with XSLT

29- XSLT (eXtensible Style Sheet Language Transformation)

What is XSLT?

XSLT (eXtensible Stylesheet Language Transformations) is the recommended style sheet language for XML. XSLT uses XPath to find information in an XML document. With XSLT you can add/remove elements and attributes to or from the output file. You can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.

How it Works?

XSLT uses XPath to define parts of the source document that should match one or more predefined templates. When a match is found, XSLT will transform the matching part of the source document into the result document.

30- HTML5 Audio

© Virtual University of Pakistan

112- XML Elements

What is an XML Element?

An XML element is everything from (including) the element's start tag to (including) the element's end tag. An element can contain other elements, text or attributes.

Empty XML Elements

An element with no content is said to be empty.

<element></element> Self closing:

<element />

The two forms above produce identical results in an XML parser. Empty elements do

not have any content, but they can have attributes!

XML Naming Rules

XML elements must follow some naming rules.

- Case-sensitive
- Must start with a letter or underscore
- Cannot start with the letters xml (or XML, or Xml, etc)
- Can contain letters, digits, hyphens, underscores, and periods
- Cannot contain spaces

Any name can be used; no words are reserved (except xml).

Best Naming Practices

Create descriptive names, like this: <person>, <firstname>, <lastname>.

Create short and simple names, like this: <book_title> not like this: <the_title_of_the_book>.

Avoid "-"

If you name something "first-name", some software may think you want to subtract "name" from "first".

Avoid "."

If you name something "first.name", some software may think that "name" is a property of the object "first".

Avoid ":"

Colons are reserved for namespaces.

Non-English letters like éòá are perfectly legal in XML, but you may face problems if your software doesn't support them.

XML documents often have a corresponding database. A good practice is to use the naming rules of your database for the elements in the XML documents.

Naming Styles

There are no naming styles defined for XML elements. But, it's good to know commonly used naming styles.

Style	Example	Description
Lower case	<firstname>	All letters lower case
Upper case	<FIRSTNAME>	All letters upper case
Underscore	<first_name>	Underscore separates words
Pascal case	<FirstName>	Uppercase first letter in each word
Camel case	<firstName>	Uppercase first letter in each word except the first

XML Elements are Extensible

XML elements can be extended to carry more information. Let say we have a message xml that our software use to handle messages:

```
<message>
    <to>Kamran</to>
    <from>Faisal</from>
    <body>Did you read the book Beautiful Eyes?</body>
</message>
```

Lets add some extra information to it:

```
<message>
    <date>11 Nov 2015</date>
    <to>Kalsoom</to>
    <from>Faisal</from>
    <body>Did you read the book Beautiful Eyes?</body>
```

XML is Extensible

One of the beauties of XML: It can be extended without breaking applications.

113- XML XPath

What is XPath?

XPath (the XML Path language) is a language for finding information in an XML document.

- XPath is a syntax for defining parts of an XML document

- XPath uses path expressions to navigate in XML documents
- XPath contains a library of standard functions
- XPath is a major element in XSLT
- XPath is also used in XQuery, XPointer and Xlink
- XPath is a W3C recommendation

XPath Path Expressions

XPath uses path expressions to select nodes or node-sets in an XML document. These path expressions look very much like the expressions you see when you work with a traditional computer file system.

Today XPath expressions can also be used in JavaScript, Java, XML Schema, PHP, Python, C and C++, and lots of other languages.

XPath is used in XSLT

XPath is a major element in the XSLT standard. Without XPath knowledge you will not be able to create XSLT documents.

Check **XPath Expression** below:

XPath Expression	Result
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[last()-1]	Selects the last but one book element that is the child of the bookstore element
/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element

//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='en']	Selects all the title elements that have a "lang" attribute with a value of "en"
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
/bookstore/book[price>35.00]/title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

123- JSON – Introduction JSON Topic 143

JSON stands for **JavaScript Object Notation**. JSON is a lightweight data-interchange format. JSON

Let's discuss the difference between JSON and XML through the examples below:

XML Example

```
<friends>
        <friend>
            <firstName>Asim</firstName>
            <lastName>Adeel</lastName>
        </friend>
        <friend>
            <firstName>Tanweer</firstName>
            <lastName>Khan</lastName>
        </friend>
</friends>
```

JSON Example

```
{"friends":[
   {"firstName":"Asim", "lastName":"Adeel"},
   {"firstName":"Tanweer", "lastName":"Khan"}
]}
```

The JSON format is syntactically identical to the code for creating JavaScript objects. So we can use standard JavaScript functions to convert JSON data into native JavaScript objects. See here a JSON Example:

```
var text = '{"firstName":"Asim" , "lastName":"Adeel" ,"phone":"3331234567"}';

var obj = JSON.parse(text);

obj. firstName;
obj. lastName;

obj.phone;
```

Like XML

- Both JSON and XML is "self describing" (human readable)
- Both JSON and XML is hierarchical (values within values)
- Both JSON and XML can be parsed and used by lots of programming languages
- Both JSON and XML can be fetched with an XMLHttpRequest

Unlike XML

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays

XML has to be parsed with an XML parser, JSON can be parsed by a standard JavaScript function.

Why JSON?

For AJAX applications, JSON is faster and easier than XML.

Using XML	Using JSON
• Fetch an XML document • Use the XML DOM to loop through the document • Extract values and store in variables	• Fetch a JSON string • JSON.Parse the JSON string

124- JSON Syntax Topic 144

The JSON syntax is a subset of the JavaScript syntax.

JSON Syntax Rules

JSON syntax is derived from JavaScript object notation syntax:

{"firstName":"Asim", "lastName":"Asdeel"}

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

JSON Values

JSON values can be:

- A number (integer or floating point)
- A string (in double quotes)
- A Boolean (true or false)
- An array (in square brackets)
- An object (in curly braces)
- null

JSON Objects

JSON objects are written inside curly braces. Just like JavaScript, JSON objects can contain multiple name/values pairs. See an example here:

{"firstName":"Asim", "lastName":"Asdeel"}

JSON Arrays

JSON arrays are written inside square brackets. Just like JavaScript, a JSON array can contain multiple objects.

Example

{"friends":[

{"firstName":"Asim", "lastName":"Adeel"},

{"firstName":"Tanweer", "lastName":"Khan"},

{"firstName":"Owais", "lastName":"Yousaf"}

]}

JSON Uses JavaScript Syntax

Because JSON syntax is derived from JavaScript object notation, very little extra software is needed to work with JSON within JavaScript.

With JavaScript you can create an array of objects and assign data to it. var friends = [

{"firstName":"Asim", "lastName":"Adeel"},

{"firstName":"Tanweer", "lastName":"Khan"},

{"firstName":"Owais", "lastName":"Yousaf"}

];

The first entry in the JavaScript object array can be accessed like this:

// returns Asim Adeel

friends[0].firstName + " " + friends[0].lastName;

Or:

// returns Asim Adeel

friends[0]["firstName"] + " " +friends[0] ["lastName"]

; Data can be modified like this:

friends[0].firstName = "Yasir";

Or:

JSON.parse()

The JavaScript function JSON.parse(text) can be used to convert a JSON text into a JavaScript object.

```
var obj = JSON.parse(text);
```

Old Browsers

```
var obj = eval ("(" + text + ")");
```

Topic 146

HTML5

Introduction

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and current version of the HTML standard. HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications. HTML5 is the latest version of HTML, latest take a look on HTML History.

Since the early days of the web, there have been many versions of HTML:

1989

WordlWideWeb invented by Tim Berners Lee.

1991

HTML invented by Tim Berners Lee

1993

HTML+ drafted by Dave Raggett

1995

HTML Working Group define HTML 2.0

1997

W3C Recommended HTML 3.2

1999

W3C Recommended HTML 4.01

2000

W3C Recommended XHTML 1.0

2004

WHATWG (Web Hypertext Application Technology Working Group) Formed.

2008

HTML5 WHATWG (Web Hypertext Application Technology Working Group) First Public Draft

2012

HTML5 WHATWG Living Standard

2014

HTML5 W3C Final Recommendation

HTML5

HTML5 is an enhanced version of HTML. It was published by W3C on 28 October 2014.

What is New in HTML5?

The DOCTYPE declaration for HTML5 is very simple:

```
<!DOCTYPE html>
```

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

See here HTML5 document example:

```
<!DOCTYPE html>
<html>
        <head>
        <meta charset="UTF-8">
        <title>Title of the document</title>
</head>
<body>
        Content of the document.....
    </body>
    </html>
```

New HTML5 Elements

The most interesting new elements are:

- New semantic elements like <header>, <footer>, <article>, and <section>.
- New form control attributes like number, date, time, calendar, and

range. The most interesting new elements are:

- New graphic elements: <svg> and <canvas>.
- New multimedia elements: <audio> and <video>.

New HTML5 API's (Application Programming Interfaces): HTML5 offers some set of

APIs, The most interesting new API's are:

- HTML Geolocation
- HTML Drag and Drop
- HTML Local Storage
- HTML Application Cache
- HTML Web Workers
- HTML SSE (Server Sent Events)

Elements Removed in HTML5

Some HTML4 Elements are also removed in HTML5.

Element	Replaced With
<acronym>	<abbr>
<applet>	<object>
<basefont>	CSS
<big>	CSS
	CSS
<dir>	
Element	Replaced With
<font>	CSS
<frame>	
<frameset>	
<noframes>	
<strike>	CSS
<tt>	CSS

New Elements in HTML5

Many new Elements introduced in HTML5, for better document structure. see here, some of new elements added in HTML5:

Tag	Description

<article>	Defines an article in the document
<aside>	Defines content aside from the page content
<bdi>	Defines a part of text that might be formatted in a different direction from other text
<details>	Defines additional details that the user can view or hide
<dialog>	Defines a dialog box or window

<figcaption>	Defines a caption for a <figure> element
<figure>	Defines self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for the document or a section
<header>	Defines a header for the document or a section
<main>	Defines the main content of a document

<mark>	Defines marked or highlighted text
<menuitem>	Defines a command/menu item that the user can invoke from a popup menu
<meter>	Defines a scalar measurement within a known range (a gauge)
<nav>	Defines navigation links in the document
<progress>	Defines the progress of a task

<mark>	Defines marked or highlighted text
<menuitem>	Defines a command/menu item that the user can invoke from a popup menu
<meter>	Defines a scalar measurement within a known range (a gauge)
<nav>	Defines navigation links in the document
<progress>	Defines the progress of a task
<rp>	Defines what to show in browsers that do not support ruby annotations
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography)
<ruby>	Defines a ruby annotation (for East Asian typography)
<section>	Defines a section in the document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time
<wbr>	Defines a possible line-break

New Form Elements

Tag	Description
<datalist>	Defines pre-defined options for input controls
<keygen>	Defines a key-pair generator field (for forms)
<output>	Defines the result of a calculation

New Input Types

New Input Types	New Input Attributes

• color • date • datetime • datetime-local • email • month • number • range • search	• autocomplete • autofocus • form • formaction • formenctype • formmethod • formnovalidate • formtarget • height and width • list
• tel • time • url • week	• min and max • multiple • pattern (regexp) • placeholder • required • step

HTML5 - New Attribute Syntax

HTML5 allows four different syntaxes for attributes.

HTML5 - New Attribute Syntax

- Empty

<input type="text" value="John" **disabled**>

- Unquoted

<input type="text" **value=John**>

- Double-quoted

<input type="text" **value="John Doe"**>

- Single-quoted

<input type="text" **value='John Doe'**>

HTML5 Graphics

Tag	Description

<canvas>	Defines graphic drawing using JavaScript
<svg>	Defines graphic drawing using SVG

New Media Elements

Tag	Description
<audio>	Defines sound or music content
<embed>	Defines containers for external applications (like plug-ins)
<source>	Defines sources for <video> and <audio>
<track>	Defines tracks for <video> and <audio>
<video>	Defines video or movie content

128- HTML5 Semantics

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Semantics is the study of the meanings of words and phrases in language. Semantic elements are elements with a meaning.

Examples of non-semantic elements:

 and

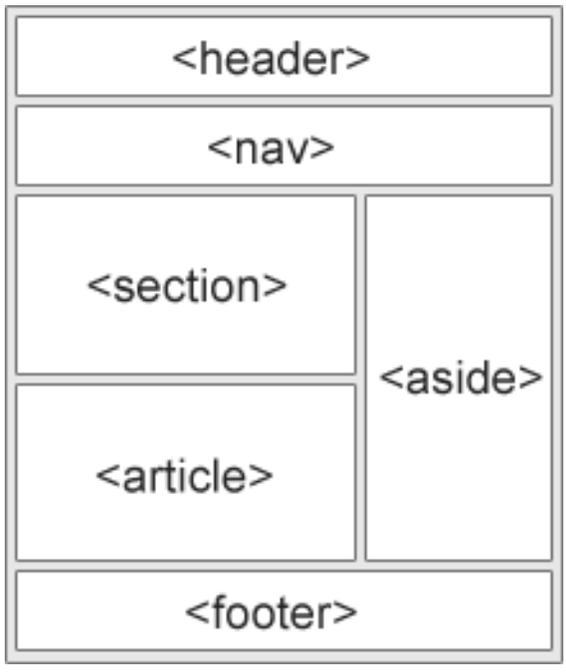
Examples of semantic elements:

<form>, <table>, and <img> It's common to use HTML code like:

```
<div id="nav">
<div class="header">
<div id="footer">
```

These are used to indicate the navigation, header, and footer in HTML5 document.

HTML5 offers new semantic elements to define different parts of a web page.



Here are some new Semantic Elements in HTML5:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>

HTML5 <section> Element

The <section> element defines a section in a document.

According to W3C's HTML5 documentation:

"A section is a thematic grouping of content, typically with a heading."

A Web site's home page could be split into sections for introduction, content, and contact information.

HTML5 <article> Element

The <article> element specifies independent, self-contained content.

An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.

Examples of where an <article> element can be used:

- Forum post
- Blog post
- Newspaper

article See the following

example:

```
<article>

<h1>Heading</h1>

<p>Lorem ipsum dolor sit amet</p>

<p>Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat  nulla pariatur.</p>

</article>
```

HTML5 <header> Element

The <header> element specifies a header for a document or section. The <header> element should be used as a container for introductory content.

HTML5 <footer> Element

The <footer> element specifies footer for a document or section. A <footer> element should contain information about its containing element.

HTML5 <nav> Element

The <nav> element defines a set of navigation links.

HTML5 <aside> Element

The <aside> element defines some content aside from the content it is placed in (like a sidebar). The aside content should be related to the surrounding content.

HTML5 <figure> and <figcaption> Elements

In books and newspapers, it is common to have captions with images. With HTML5, images and captions can be grouped together in <figure> elements.

Why Semantic HTML5 Elements?

With HTML4, developers used their own favorite attribute names to style page elements:

header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav,

...

Why Semantic HTML5 Elements?

This made it impossible for search engines to identify the correct web page content. With HTML5 elements like: <header> <footer> <nav> <section> <article>, this will become easier to identify.

According to the W3C, a Semantic Web:

"Allows data to be shared and reused across applications, enterprises, and communities."

129- HTML4 to HTML5 Migration

Migration from HTML4 to HTML5

Let see how to convert an existing HTML4 page into an HTML5 page, without destroying anything of the original content or structure.

Typical HTML4	Typical HTML5
	<header>
	<nav>
	<section>
	<article>
	<footer>

Change to HTML5 Doctype

HTML4 doctype:

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

HTML5 doctype:

<!DOCTYPE html>

HTML4:

```
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
```

HTML5:

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

Browser Support

HTML5 semantic elements are supported in all modern browsers. In addition, you can "teach" older browsers how to handle "unknown elements".

For Internet Explorer support:

```
<!--[if lt IE 9]>
  <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

Add CSS for HTML5 Semantic Elements

Review your existing CSS styles

Existing CSS styles:

```
div#header,div#footer,div#content,div#post  {
   border:1px solid grey;margin:5px;margin-
bottom:15px;padding:8px;background- color:white;
                }
div#header,div#footer {
                    color:white;background-color:#444;margin-bottom:5px;
                }
div#content {
                    background-color:#ddd;
                }
                div#menu ul {
                    margin:0;padding:
                    0;
                }
                div#menu ul li {
                    display:inline;
                    margin:5px;
```

```
}
```

Add CSS for HTML5 Semantic Elements

Duplicate with equal CSS styles for HTML5 semantic elements.

CSS for HTML5

```
header,footer,section,article  {
   border:1px                    solid     grey;margin:5px;margin-
bottom:15px;padding:8px;background- color:white;
                    }
header,footer {
                        color:white;background-color:#444;margin-bottom:5px;
                    }
section {
                        background-color:#ddd;
                    }
                    nav ul  {
                        margin:0;padding:
                        0;
                    }
nav ul li {
                        display:inline; margin:5px;
                    }
```

Change to HTML5 <header> and <footer>

Change the elements with id="header" and id="footer" to HTML5 semantic <header> and <footer> elements

```
<div id="header">
  <h1>Monday Times</h1>
</div>
.
.
<div id="footer">
  <p>&amp;copy; 2015 SomeSite. All rights reserved.</p>
```

```
</div>
```

```
  <h1>Monday Times</h1>
</header>
.
.
<footer>
  <p>&copy; 2014 W3Schools. All rights reserved.</p>
</footer>
```

Change to HTML5 <nav>

Change the element with id="menu" to an HTML5 semantic <nav> element.

```
<div id="menu">
                    <ul>
                      <li>News</li>
                      <li>Sports</li>
                      <li>Weather</li>
                    </ul>
</div>


            <nav>

                    <ul>
                      <li>News</li>
                      <li>Sports</li>
                      <li>Weather</li>
                    </ul>
```

Change to HTML5 <section>

Change the element with id="content" to an HTML5 semantic <section> element.

```
<div                                                    id="content“>
                    ….
</div>
<section>
                    ….
</section>
```

Change to HTML5 <article>

Change all element with class="post" to HTML5 semantic <article> elements.

```
<div class="post">
                    <h2>News Article</h2>
                    <p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum
hurum turum ipsum
                    lurum hurum turum.</p>
</div>
<article>

                    <h2>News Article</h2>
                    <p>Ipsum lurum hurum turum ipsum lurum hurum turum ipsum lurum hurum
turum ipsum
                    lurum hurum turum.</p>
</article>
```

<head> Tag

Finally you can remove the <head> tags. They are not needed in HTML5.

130a- HTML5 Coding Conventions

No certain coding style and syntax to use in HTML. With XHTML, developers were forced to write valid and "well-formed" code.

HTML5 is a bit sloppier when it comes to code validation. With HTML5, you must create your own Best Practice, Style Guide and Coding Conventions.

Be Smart and Future Proof

Keep a consistent style is the key. Using a well-formed "close to XHTML" syntax, can be smart.

Correct Document Type

Always declare the document type as the first line in your document:

<!DOCTYPE html>

If you want consistency with lower case tags, you can use:

<!doctype html>

Use Lower Case Element Names

HTML5 allows mixing uppercase and lowercase letters in element names. We recommend using lowercase element names.

<SECTION>

This is a paragraph.

</SECTION>

<Section>

** This is a paragraph. **

</SECTION>

<section>

This is a paragraph.

</section>

Close All HTML Elements

In HTML5, you don't have to close all elements (for example the element). We recommend closing all HTML elements.

Close All HTML Elements

Looking bad:

<section>

This is a paragraph.

This is a paragraph.

</section>

Looking good:

<section>

This is a paragraph.

This is a paragraph.

</section>

Close Empty HTML Elements

In HTML5, it is optional to close empty elements.

This is allowed:

<meta charset="utf-8">

This is also allowed:

<meta charset="utf-8" />

The slash (/) is required in XHTML and XML.

Use Lower Case Attribute Names

HTML5 allows mixing uppercase and lowercase letters in attribute names. We recommend using lowercase attribute names.

Quote Attribute Values

HTML5 allows attribute values without quotes. We recommend quoting attribute values.

This will not work, because the value contains spaces:

<table class=table striped>

This will work:

<table class="table striped">

Not good to mix style, so better to always use quotes.

Image Attributes

Always use the alt attribute with images. It is important when the image cannot be viewed.

<img src="html5.gif" alt="HTML5" style="width:128px;height:128px“>

Always define image size. It reduces flickering because the browser can reserve space for images before they are loaded.

<img src="html5.gif" alt="HTML5“ style="width:128px;height:128px">

130b- HTML5 Coding Conventions Avoid Long

Code Lines

When using an HTML editor, it is inconvenient to scroll right and left to read the HTML code. Try to avoid code lines longer than 80 characters.

Spaces and Equal Signs

Spaces around equal signs is legal:

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

But space-less is easier to read, and groups entities better together:

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

Blank Lines and Indentation

Do not add blank lines without a reason. For readability, add blank lines to separate large or logical code blocks. For readability, add 2 spaces of indentation. Do not use TAB.

Blank Lines and Indentation

Do not use unnecessary blank lines and indentation. It is not necessary to use blank lines between short and related items. It is not necessary to indent every element.

Omitting <html> and <body>?

In the HTML5 standard, the <html> tag and the <body> tag can be omitted.

```
<!DOCTYPE html>
<head>
        <title>Page Title</title>
</head>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

We do not recommend omitting the <html> and <body> tags.

<html> element is the root.

Declaring a language is important for accessibility applications (screen readers) and search engines. Possible in <html> element.

Omitting <head>?

In the HTML5 standard, the <head> tag can also be omitted. By default, browsers will add all elements before <body>, to a default <head> element.

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

```
<h1>This is a heading</h1>

<p>This is a paragraph.</p>

</body>


</html>
```

Meta Data

The <title> element is required in HTML5. Make the title as meaningful as possible. To ensure proper interpretation, and correct search engine indexing, both the language and the character encoding should be defined as early as possible in a document:

```
<!DOCTYPE html>

<html lang="en-US">

<head>

        <meta charset="UTF-8">

        <title>My Page Title</title>

</head>
```

HTML Comments

Short comments should be written on one line, with a space after <!-- and a space before -->:

```
<!-- This is a comment -->
```

Long comments, spanning many lines, should be written with <!-- and --> on separate lines:

```
<!--

 This is a long comment example. This is a long comment example. This is a long comment example.

 This is a long comment example. This is a long comment example. This is a long comment example.

-->
```

Long comments are easier to observe, if they are indented 2 spaces.

Loading Style Sheets

Use simple syntax for linking style sheets (the type attribute is not necessary).

Loading JavaScript

Use simple syntax for loading external scripts (the type attribute is not necessary).

<script src="myscript.js">

Use Lower Case File Names

Most web servers (Apache, Unix) are case sensitive about file names. Other web servers (Microsoft, IIS) are not case sensitive. If you move from a case insensitive, to a case sensitive server, even small errors will break your web. To avoid these problems, always use lower case file names (if possible).

Use Lower Case File Names

If you move from a case insensitive, to a case sensitive server, even small errors will break your web. To avoid these problems, always use lower case file names (if possible).

131- HTML5

Canvas What is HTML

Canvas?

The HTML <canvas> element is used to draw graphics, on the fly, via scripting (usually JavaScript). The <canvas> element is only a container for graphics. You must use a script to actually draw the graphics. Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

Canvas Examples

A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.

<canvas id="myCanvas" width="200" height="100"></canvas>

Always specify an id attribute (to be referred to in a script), and a width and height attribute to define the size of the canvas.

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">

</canvas>

Drawing on a Canvas

- Find the Canvas
- Create drawing object
- Draw on canvas

var can = document.getElementById("myCanvas");

Create a Drawing Object

var can = document.getElementById("myCanvas"); var ctx =

can.getContext("2d");

Draw on Canvas

```
var can = document.getElementById("myCanvas"); var ctx =
can.getContext("2d");
```

ctx.fillStyle = "#00FF00"; ctx.fillRect(0,0,200,100);

fillStyle()

Can be a CSS color, gradient or a pattern.

Drawing a Rectangle

fillRect(x,y,width,height)

<script>

var c = document.getElementById("myCanvas"); var ctx =

c.getContext("2d");

ctx.fillStyle = "#FF0000"; ctx.fillRect(0,0,200,100);

</script>

132- HTML5 Canvas Coordinates

Canvas Coordinates

The HTML canvas is a two-dimensional grid. The upper-left corner of the canvas has the coordinates (0,0)

Draw a Line

Move Current position to starting point. Draw line to a specific point. Use an ink method to display line in canvas.

moveTo(x,y) - defines the starting point of the line lineTo(x,y) -

defines the ending point of the line stroke() - display line in canvas

var c = document.getElementById("myCanvas"); var ctx =

c.getContext("2d");

ctx.moveTo(0,0); ctx.lineTo(200,100);

ctx.stroke();

133- HTML5 Canvas Gradients

Canvas Gradients

Gradients can be used to fill rectangles, circles, lines, text, etc. Shapes on the canvas are not limited to solid colors.

Draw Gradients

There are two different types of gradients: **Linear**

createLinearGradient(x,y,x1,y1) **Radial/Circular**

createRadialGradient(x,y,r,x1,y1,r1)

Once we have a gradient object, we must add two or more color stops. addColorStop()

The addColorStop() method specifies the color stops, and its position along the gradient. Gradient positions can be anywhere between 0 to 1.

addColorStop(pos,color);

Draw Linear Gradient

```
var c = document.getElementById("myCanvas"); var ctx =
c.getContext("2d");

// Create gradient
var grd = ctx.createLinearGradient(0,0,200,0);
grd.addColorStop(0,"yellow"); grd.addColorStop(1,"blue");

// Fill with gradient ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
```

Draw Circular Gradient

```
var c = document.getElementById("myCanvas"); var ctx =
c.getContext("2d");

// Create gradient
```

var grd = ctx.createRadialGradient(75,50,5,90,60,100);

grd.addColorStop(0,"red"); grd.addColorStop(1,"white");

// Fill with gradient ctx.fillStyle = grd;

ctx.fillRect(0,0,200,80);

134- HTML5 Canvas Text

Drawing Text

You can draw text on canvas too with a specified fonts and style. To draw text on a canvas, the most important property and methods are:

font - defines the font properties for the text fillText(text,x,y) - draws

"filled" text on the canvas strokeText(text,x,y) - draws text on the canvas

(no fill)

var c = document.getElementById("myCanvas"); var ctx =

c.getContext("2d");

ctx.font = "30px Arial"; ctx.fillText("Some

text",10,50);

Stroke Text

var c = document.getElementById("myCanvas"); var ctx =

c.getContext("2d");

ctx.font = "30px Arial"; ctx.strokeText("Some

text",10,50);

136- HTML5 SVG

SVG

- SVG stands for Scalable Vector Graphics
- SVG is used to define graphics for the Web
- SVG is a W3C recommendation

The HTML5 <svg> Element

The <svg> element is a container for SVG graphics. SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

SVG Circle

```
<!DOCTYPE html>
<html>
<body>
<svg width="500" height="500">
        <circle cx="200" cy="200" r="100"
        stroke="blue" stroke-width="5" fill="yellow"
        />
</svg>
</body>
</html>
```

SVG Rectangle

```
<svg width="400" height="100">
<rect x="100" y="100" width="300" height="100" style="fill:rgb(0,0,255);stroke-width:1;stroke:rgb(0,0,0)" />
</svg>
```

SVG Rounded Rectangle

```
<svg width="400" height="100">
<rect x="100" y="100" rx="20" ry="20" width="300" height="100" style="fill:rgb(0,0,255);stroke-width:1;stroke:rgb(0,0,0)" />
</svg>
```

SVG Polygon

```
<svg width="400" height="100">
        <polygon points="100,50 150,50 150,150 100,10"
        style="fill:lime;stroke:red;stroke-width:5;fill-
        rule:evenodd;" />
</svg>
```

SVG Star using Polygon

```
<svg width="400" height="100">

                    <polygon points="100,10 40,198 190,78 10,78 160,198"
```

```
style="fill:lime;stroke:green;stroke-width:5;fill-rule:evenodd;"  />
</svg>
```

SVG and Canvas

SVG is a language for describing 2D graphics in XML. Canvas draws 2D graphics, on the fly (with a JavaScript).

SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element.

In SVG, each drawn shape is remembered as an object. If attributes of an SVG object are changed, the browser can automatically re-render the shape.

Canvas is rendered pixel by pixel. In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

Canvas	**SVG**
• Resolution dependent • No support for event handlers • Poor text rendering capabilities • You can save the resulting image as .png or .jpg • Well suited for graphic-intensive games	• Resolution independent • Support for event handlers • Best suited for applications with large rendering areas (Google Maps) • Slow rendering if complex (anything that uses the DOM a lot will be slow) • Not suited for game applications

137- HTML5 Media

HTML5 Multimedia

What is Multimedia?

Multimedia comes in many different formats. It can be almost anything you can hear or see.

Examples: Pictures, music, sound, videos, records, films, animations, and more.

Browser Support

- text only, single color.
- colors, fonts and pictures.
- sounds, animations, and videos

HTML5 multimedia promises an easier future for multimedia.

Multimedia Formats

Multimedia elements (like sounds or videos) are stored in media files. The most common way to discover the type of a file, is to look at the file extension.

Video Formats

HTML5 standard support MP4, WebM, Ogg video formats

Format	Description
Ogg .ogg	Theora Ogg. Developed by the Xiph.Org Foundation.
WebM .webm	WebM. Developed by the web giants, Mozilla, Opera, Adobe, and Google.
MPEG-4 or MP4 .mp4	By the Moving Pictures Expert Group. Based on QuickTime. Commonly used in newer video cameras and TV hardware.

Audio Formats

HTML5 standard support MP3, WAV, Ogg audio formats.

Format	Description
WAV .wav	By IBM and Microsoft. Plays well on Win, Mac, and Linux operating systems.
Ogg .ogg	Ogg. By the Xiph.Org Foundation.
MP3 .mp3	MP3 files are actually the sound part of MPEG files. most popular. Combines good compression (small files) with high quality.

138- HTML5 Video

Playing Videos in HTML

Before HTML5, there was no standard for showing videos on a web page, and the videos could only be played with a plug-in (like flash). **The HTML5 <video>** element specifies a standard way to embed a video in a web page. See the following example:

```
<video width="320" height="240" controls>
    <source src="movie.mp4" type="video/mp4">
    <source src="movie.ogg" type="video/ogg">
```

Your browser does not support the video tag.

© Virtual University of Pakistan

</video>

In the above example,

- The controls attribute adds video controls, like play, pause, and volume.
- Width and Height are optional but recommended.
- Text between the <video> and </video> tags will only display in browsers that do not support the <video> element.
- Multiple <source> elements can link to different video files. The browser will use the first recognized format.

HTML <video> Autoplay

To start a video automatically use the autoplay attribute

<video width="320" height="240" autoplay>

<source src="movie.mp4" type="video/mp4">

<source src="movie.ogg" type="video/ogg"> Your browser does

not support the video tag.

</video>

Note: Does not work on iPad and iPhone.

HTML Video - Browser Support

Currently, there are 3 supported video formats for the <video> element: MP4, WebM, and Ogg.

HTML Video - Methods, Properties, and Events

HTML5 defines DOM methods, properties, and events for the <video> element. This allows you to load, play, and pause videos, as well as setting duration and volume.

Method	Description
addTextTrack()	Adds a new text track to the audio/video
canPlayType()	Checks if the browser can play the specified audio/video type
load()	Re-loads the audio/video element
play()	Starts playing the audio/video
pause()	Pauses the currently playing audio/video

HTML5 Some Video Properties

Property	Description
autoplay	Sets or returns whether the audio/video should start playing as soon as it is loaded
currentTime	Sets or returns the current playback position in the audio/video (in seconds)
duration	Returns the length of the current audio/video (in seconds)
ended	Returns whether the playback of the audio/video has ended or not
loop	Sets or returns whether the audio/video should start over again when finished
volume	Sets or returns the volume of the audio/video

HTML Video - Methods, Properties, and Events

There are also DOM events that can notify you when a video begins to play, is paused, etc.

HTML5 Some Video Events

Event	Description
ended	Fires when the current playlist is ended
error	Fires when an error occurred during the loading of an audio/video
loadeddata	Fires when the browser has loaded the current frame of the audio/video
pause	Fires when the audio/video has been paused
play	Fires when the audio/video has been started or is no longer paused
seeked	Fires when the user is finished moving/skipping to a new position in the audio/video
seeking	Fires when the user starts moving/skipping to a new position in the audio/video
volumechange	when the volume has been changed

139- HTML5 Audio

Before HTML5, there was no standard for playing audio files on a web page and the audio files could only be played with a plug-in (like flash).

The HTML5 <audio> element specifies a standard way to embed audio in a web page.

The HTML <audio> Element

To play an audio file in HTML, use the <audio> element.

```
<audio controls>
    <source src="sound.ogg" type="audio/ogg">
    <source src="sound.mp3" type="audio/mpeg">
    Your browser does not support the audio
    element.
</audio>
```

HTML Audio - Browser Support

Currently, there are 3 supported file formats for the <audio> element: MP3, Wav, and Ogg.

HTML Audio - Methods, Properties, and Events

HTML5 defines DOM methods, properties, and events for the <audio> element. This allows you to load, play, and pause audios, as well as setting duration and volume.

Method	Description
addTextTrack()	Adds a new text track to the audio/video
canPlayType()	Checks if the browser can play the specified audio/video type
load()	Re-loads the audio/video element
play()	Starts playing the audio/video
pause()	Pauses the currently playing audio/video

HTML5 Some Audio Properties

Property	Description
autoplay	Sets or returns whether the audio/video should start playing as soon as it is loaded
currentTime	Sets or returns the current playback position in the audio/video (in seconds)
duration	Returns the length of the current audio/video (in seconds)
ended	Returns whether the playback of the audio/video has ended or not
loop	Sets or returns whether the audio/video should start over again when finished
volume	Sets or returns the volume of the audio/video

HTML Audio - Methods, Properties, and Events

There are also DOM events that can notify you when an audio begins to play, is paused, etc.

HTML5 Some Audio Events

Event	Description
ended	Fires when the current playlist is ended
error	Fires when an error occurred during the loading of an audio/video
loadeddata	Fires when the browser has loaded the current frame of the audio/video
pause	Fires when the audio/video has been paused
play	Fires when the audio/video has been started or is no longer paused
seeked	Fires when the user is finished moving/skipping to a new position in the audio/video
seeking	Fires when the user starts moving/skipping to a new position in the audio/video
volumechange	when the volume has been changed

140- HTML5 Plug-ins

HTML Helpers

Helper applications are computer programs that extend the standard functionality of a web browser. Helper applications are also called plug-ins.

HTML Plug-ins

Examples of well-known plug-ins are Java applets. Plug-ins can be added to web pages with the <object> tag or the <embed> tag. Plug-ins can be used for many purposes: display maps, scan for viruses, verify your bank id, etc.

The <object> Element

The <object> element is supported by all browsers. The <object> element defines an embedded object within an HTML document.

```
<object width="100px" height="500px" data="example.swf"></object>

<object width="100%" height="500px" data="snippet.html"></object>

<object data="animage.jpeg"></object>
```

The <embed> Element

The <embed> element is also supported in all major browsers. The <embed> element also defines an embedded object within an HTML document.

Web browsers have supported the <embed> element for a long time. But it's not a part of HTML specification before HTML5.

```
<embed width="400" height="50" src="bookmark.swf">

<embed width="100%" height="500px" src="snippet.html">

<embed src="animage.jpeg">
```

141- HTML5 Geolocation

Locate the User's Position

The HTML Geolocation API is used to get the geographical position of a user. Since this can compromise user privacy, the position is not available unless the user approves it.

Geolocation is much more accurate for devices with GPS, like iPhone.

Using HTML Geolocation

Use the getCurrentPosition() method to get the user's position.

Example:

```
navigator.geolocation. getCurrentPosition(showPosition);

function showPosition(position) {

                str =    "Latitude: " +

                                                position.coords.latitude

                                + "<br>Longitude: " +
```

position.coords.longitude;

}

Handling Errors and Rejections

The second parameter of the getCurrentPosition() method is used to handle errors. It specifies a function to run if it fails to get the user's location.

See the following example:

navigator.geolocation.

getCurrentPosition(showPosition,

showError); Here is another example:

```
function showError(error) {
  switch(error.code) {
    case error.PERMISSION_DENIED: break;
    case error.POSITION_UNAVAILABLE:
                                        break
    ; case error.TIMEOUT: break;
    case error.UNKNOWN_ERROR: break;
  }
}
```

Displaying the Result in a Map

To display the result in a map, you need access to a map service that can use latitude and longitude, like Google Maps. See the example below:

var img_url = "http://maps.googleapis.com/maps/api/staticmap?center=" + **langitude** +","+ **longitude** + "&zoom=14&size=400x300&sensor=false";

Location-specific Information

Geolocation is also very useful for location-specific information.

- Up-to-date local information
- Showing Points-of-interest near the user
- Turn-by-turn navigation (GPS)

getCurrentPosition() Method - Return Data

The getCurrentPosition() method returns an object if it is successful. The latitude, longitude and accuracy properties are always returned.

Property	Description
coords.latitude	The latitude as a decimal number
coords.longitude	The longitude as a decimal number
coords.accuracy	The accuracy of position
coords.altitude	The altitude in meters above the mean sea level
coords.altitude Accuracy	The altitude accuracy of position

coords.heading	The heading as degrees clockwise from North
coords.speed	The speed in meters per second
timestamp	The date/time of the response

Other interesting Methods

watchPosition()

- Returns the current position of the user and continues to return updated position as the user moves (like the GPS in a car).

clearWatch()

- Stops the watchPosition() method.

142- HTML5 Drag/Drop

HTML5 Drag and Drop

Drag and drop is a part of the HTML5 standard. Drag and drop is a very common feature.

Make an Element Draggable

First of all: To make an element draggable, set the draggable attribute to true:

<img draggable="true">

What to Drag - ondragstart and setData()

Then, specify what should happen when the element is dragged. The ondragstart attribute calls a function, drag(event), that specifies what data to be dragged.

The dataTransfer.setData() method sets the data type and the value of the dragged data: function

drag(ev) {

ev.dataTransfer.setData("text", ev.target.id);

}

In this case, the data type is "text" and the value is the id of the draggable element ("drag1").

Where to Drop - ondragover

We need to take care of ondragover and ondrop events.

- The ondragover event specifies where the dragged data can be dropped.
- By default, data/elements cannot be dropped in other elements.
- To allow a drop, we must prevent the default handling of the

element. This is done by calling the event.preventDefault() method for the

ondragover event.

event.preventDefault()

Do the Drop - ondrop

When the dragged data is dropped, a drop event occurs. function drop(ev)

{

ev.preventDefault();

var data = ev.dataTransfer.getData("text"); ev.target.appendChild(document.getElementById(data));

}

Drag Drop

To Drag:

- Make element dragable
- Handle

ondragstart Where to

Drop:

- Handle ondragover
- Handle ondrop

© Virtual University of Pakistan

143- HTML5 Local Storage

HTML local storage: replacement of and better than cookies.

What is HTML Local Storage?

With local storage, web applications can store data locally within the user's browser. Before HTML5, application data had to be stored in cookies, included in every server request.

- More secure
- Large amounts of data (5mb) can be stored locally.
- Information is never transferred to the server.

HTML Local Storage Objects

HTML local storage provides two objects for storing data on the client:

- window.localStorage - stores data with no expiration date
- window.sessionStorage - stores data for one session (data is lost when the tab is

closed) Before using local storage, check browser support for localStorage and

sessionStorage:

```
if(typeof(Storage) !== "undefined") {
                    // Code for localStorage/sessionStorage.
} else {
                    // No Web Storage support..
                }
```

The localStorage Object

The localStorage object stores the data with no expiration date.

The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

Example:

```
// Store
localStorage.setItem("lastname",  "Adeel");
// Retrieve
var str = localStorage.getItem("lastname");
```

Example:

```
// Store
```

```
localStorage.lastname = "Adeel";

// Retrieve
```

```
var str = localStorage.lastname;

// To Remove localStorage.removeItem("lastname");
```

Note: Name/value pairs are always stored as strings. Remember to convert them to another format as needed.

The sessionStorage Object

The sessionStorage object is equal to the localStorage object, except that it stores the data for only one session.

144- HTML5 App Cache

With application cache it is easy to make an offline version of a web application, by creating a cache manifest file.

What is Application Cache?

HTML5 introduces application cache, which means that a web application is cached, and accessible without an internet connection. Application cache gives an application three advantages:

- Offline browsing
- Speed
- Reduced server

load See the example:

```
<!DOCTYPE HTML>

<html manifest="demo.appcache">

<body>

The content of the document.....

</body>

</html>
```

Cache Manifest Basics

To enable application cache, include the manifest attribute in the document's <html> tag:

```
<!DOCTYPE HTML>

<html manifest="demo.appcache">

        ...
```

Every page with the manifest attribute specified will be cached. If the manifest attribute is not specified, the page will not be cached. The recommended file extension for manifest file is: ".appcache"

The Manifest File

The manifest file is a simple text file, which tells the browser what to cache (and what to never cache).

The Manifest File

The manifest file has three sections:

- **CACHE MANIFEST** - Files listed under this header will be cached after they are downloaded for the first time
- **NETWORK** - Files listed under this header require a connection to the server, and will never be cached
- **FALLBACK** - Files listed under this header specifies fallback pages if a page is inaccessible

CACHE MANIFEST

The first line, CACHE MANIFEST, is required:

- CACHE MANIFEST
- /theme.css
- /logo.gif
- /main.js

NETWORK

The NETWORK section below specifies that the file "login.asp" should never be cached, and will not be available offline:

- NETWORK:
- login.asp

An asterisk can be used to indicate that all other resources/files require an internet

connection: NETWORK:

*

FALLBACK

The FALLBACK section below specifies that "offline.html" will be served in place of all files in the /html/ catalog, in case an internet connection cannot be established:

FALLBACK:

/html/ /offline.html

Once an application is cached, it remains cached until one of the following happens:

- The user clears the browser's cache
- The manifest file is modified
- The application cache is programmatically updated

Notes on Application Cache

Be careful with what you cache. Once a file is cached, the browser will continue to show the cached version, even if you change the file on the server. To ensure the browser updates the cache, you need to change the manifest file.

145- HTML5 Web Workers What is a Web Worker?

When executing scripts in an HTML page, the page becomes unresponsive until the script is finished. A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. Web worker runs in the background.

Check Web Worker Support

Before creating a web worker, check whether the user's browser supports it: if(typeof(Worker) !==

"undefined") {

// Yes! Web worker support!

// Some code.....

} else {

// Sorry! No Web Worker support..

}

Create a Web Worker File

Now, let's create our web worker in an external JavaScript. var i = 0;

function timedCount()

{ i = i + 1;

postMessage(i);

setTimeout("timedCount()",500);

}

timedCount();

```
var i = 0;
        function timedCount()
          { i = i + 1;
          postMessage(i);

          setTimeout("timedCount()",500);
        }
timedCount();
```

postMessage() method - is used to post a message back to the HTML page.

Web Worker Object

Now that we have the web worker file, we need to call it from an HTML page. Let's creates a new web worker object and runs the code in "test.js"

```
if(typeof(w) == "undefined") {
  w = new Worker("test.js");

}
```

Then we can send and receive messages from the web worker. Add an "onmessage" event listener to the web worker.

```
w.onmessage = function(event){
  document.getElementById("result").innerHTML = event.data;

};
```

When the web worker posts a message, the code within the event listener is executed. The data from the web worker is stored in **event.data**

Terminating a Web Worker

When a web worker object is created, it will continue to listen for messages (even after the external script is finished) until it is terminated. To terminate a web worker, and free browser/computer resources, use the terminate() method:

```
w.terminate();
```

Reuse the Web Worker

If you set the worker variable to undefined, after it has been terminated, you can reuse the variable:

w = undefined;

Web Workers and the DOM

Web workers do not have access to the following JavaScript objects:

- The window object
- The document object
- The parent object

146- HTML5 SSE

HTML5 Server-Sent Events

A server-sent event is when a web page automatically get updates from a server.

Server-Sent Events - One Way Messaging

This was also possible before, but the web page would have to ask if any updates were available. With server-sent events, the updates come automatically. See the example below:

```
var source = new

EventSource(“sse.php");

source.onmessage = function(event) {

  res =

  document.getElementById("result")

  res.innerHTML += event.data + "<br>";

};
```

- Create a new EventSource object, and specify the URL of the page sending the updates.
- Each time an update is received, the onmessage event occurs.

Check Server-Sent Events Support

It’s good to check for Server Support first. if(typeof(EventSource)

```
!== "undefined") {

        // Yes! Server-sent events support!

        // Some code.....

} else {

        // Sorry! No server-sent events support..

}
```

Server-Side Code Example

To make SSE work, you need a server capable of sending data updates (like PHP or ASP). The server-side event stream syntax is simple. Set the "Content-Type" header to "text/event-stream". Now you can start sending event streams.

The server-side event stream syntax is simple.

- Set the "Content-Type" header to "text/event-stream".
- Start sending event streams.

Code in PHP (sse.php):

```
<?php
        header('Content-Type: text/event-stream');
        header('Cache-Control: no-cache');
$time = date('r');
        echo "data: The server time is: {$time}\n\n";
        flush();
        ?>
<?php
```

- Set the "Content-Type" header to "text/event-stream"
- Specify that the page should not cache
- Output the data to send (Always start with "data: ")
- Flush the output data back to the web page

The EventSource Object

There are other events also available with EventSource object.

Events	Description
onopen	When a connection to the server is opened
onmessage	When a message is received
onerror	When an error occurs

==========---------- ==========

CSS3

CSS3 is the latest standard for CSS. It is divided into several separate documents called "modules". Each module adds new capabilities or extends features defined in CSS 2, preserving backward compatibility.

© Virtual University of Pakistan

CSS3 Modules

CSS3 has been split into "modules". It contains the "old CSS specification" (which has been split into smaller pieces). In addition, new modules are added. Some of the most important CSS3 modules are:

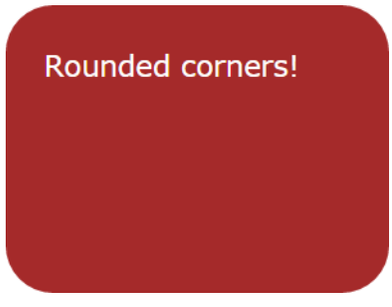
- Selectors
- Box Model
- Backgrounds and Borders
- Image Values and Replaced Content
- Text Effects
- 2D/3D Transformations
- Animations
- Multiple Column Layout
- User Interface

Most of the new CSS3 properties are implemented in modern browsers.

149- CSS3 Round Corners

CSS3 border-radius Property

With CSS3, you can give any element "rounded corners", by using the border-radius property. See below rounded corners for an element with a specified background color:



See below the CSS code for this element: #rcorners {

border-radius:

25px; background:

brown; padding:

20px; width:

200px; height:

150px;

```
}
```

CSS3 border-radius - Specify Each Corner

If you specify only one value for the border-radius property, this radius will be applied to all 4 corners. But you can specify each corner separately if you wish.

border-radius: 15px 50px 30px 5px

Four values: first value applies to top-left, second value applies to top-right, third value applies to bottom-right, and fourth value applies to bottom-left corner.

border-radius: 15px 50px 30px

Three values: first value applies to top-left, second value applies to top-right and bottom-left, and third value applies to bottom-right

border-radius: 15px 50px

Two values: first value applies to top-left and bottom-right corner, and the second value applies to top-right and bottom-left corner

150- CSS3 Border Images

The CSS3 border-image property allows you to specify an image to be used instead of the normal border around an element.

CSS3 border-image Property

The property has three parts:

- The image to use as the border
- Where to slice the image
- Define whether the middle sections should be repeated or

stretched Let's use the following image:



The border-image property takes the image and slices it into nine sections, like a tic-tac-toe board. It then places the corners at the corners, and the middle sections are repeated or stretched as you specify.

For border-image to work, the element also needs the border property set!

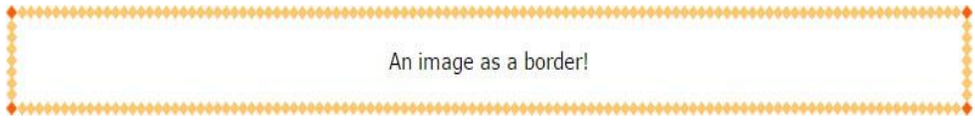
Example

```
#borderimg {
        border: 10px solid
        transparent; padding: 15px;
        border-image: url(border.png) 30 round;
```

}

border-image: url(border.png) 30 round;

Here, the middle sections of the image are repeated to create the border:

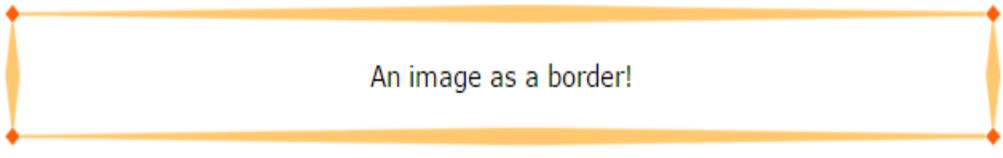


Example

```
#borderimg {
   border: 10px solid transparent;
   padding: 15px;
   border-image: url(border.png) 30 stretch;
}
```

border-image: url(border.png) 30 stretch;

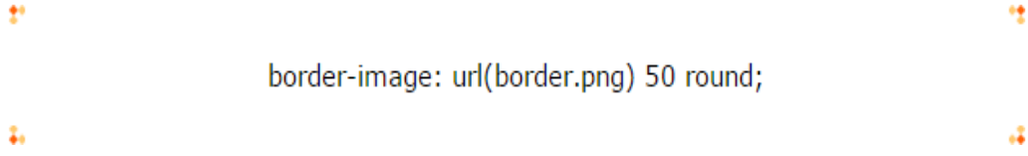
Here, the middle sections of the image are stretched to create the border:



CSS3 border-image - Different Slice Values

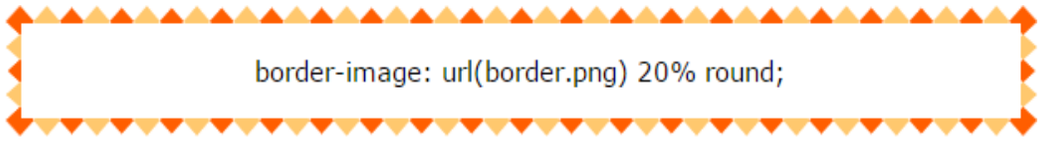
Different slice values completely changes the look of the

border: Example 1:



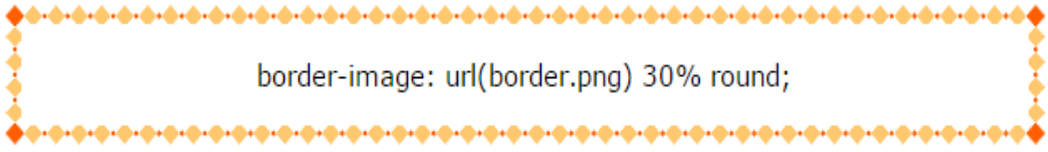
border-image: url(border.png) 50 round;

Example 2:



border-image: url(border.png) 20% round;

Example 3:



border-image: url(border.png) 30% round;

151a- CSS3 Backgrounds

CSS3 contains a few new background properties, which allow greater control of the background element.

- We can add multiple background images to one element.
- Some new CSS3 properties added too:
  - background-size
  - background-origin
  - background-clip

CSS3 Multiple Backgrounds

CSS3 allows you to add multiple background images for an element, through the background- image property.

The different background images are separated by commas, and the images are stacked on top of each other, where the first image is closest to the viewer.

Background Size

The CSS3 background-size property allows you to specify the size of background images.

Before CSS3, the size of a background image was the actual size of the image. CSS3 allows us to re-use background images in different contexts. The size can be specified in lengths, percentages, or by using one of the two keywords: contain or cover.

The two other possible values for background-size are, contain and cover.

The contain keyword scales the background image to be as large as possible (but both its width and its height must fit inside the content area).

background-size: **contain**;

The contain keyword scales the background image to be as large as possible (but both its width and its height must fit inside the content area).

background-size: **cover**;

The cover keyword scales the background image so that the content area is completely covered by the background image (both its width and height are equal to or exceed the content area). As such, some parts of the background image may not be visible in the background positioning area.

Define Sizes of Multiple Background Images

The background-size property also accepts multiple values for background size (using a comma- separated list), when working with multiple backgrounds.

Example

```
#example1 {
  background:
    url(img1.gif) left top no-repeat,
    url(img2.gif) right bottom no-repeat,
    url(img3.gif) left top repeat;
  background-size: 50px, 130px, auto;
}
```

Full Size Background Image

Let's make a background image that cover entire browser all the time. See this example: html {

```
  background: url(img.jpg) no-repeat center center
  fixed; background-size: cover;
}
```

CSS3 background-origin Property

The CSS3 background-origin property specifies where the background image is

positioned. The property takes three different values:

- **border-box** - the background image starts from the upper left corner of the border
- **padding-box** - (default) the background image starts from the upper left corner of the padding edge

CSS3 background-clip Property

The CSS3 background-clip property specifies the painting area of the background.

The property takes three different values:

- **border-box** - (default) the background is painted to the outside edge of the border
- **padding-box** - the background is painted to the outside edge of the padding
- **content-box** - the background is painted within the content box

152- CSS3 Colors

CSS supports color names, hexadecimal and RGB colors. CSS3 also introduces:

- RGBA colors
- HSL colors
- HSLA colors
- opacity

RGBA Colors

RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

An RGBA color value is specified with: rgba(red, green,

blue, alpha)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

HSL Colors

HSL stands for Hue, Saturation and Lightness. An HSL color value is specified with:

hsl(hue, saturation, lightness)

- Hue is a degree on the color wheel (from 0 to 360):
  - 0 (or 360) is red
  - 120 is green
  - 240 is blue
- Saturation is a percentage value: 100% is the full color.
- Lightness is also a percentage; 0% is dark (black) and 100% is white.

HSLA Colors

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color. An HSLA color value is specified with:

hsla(hue, saturation, lightness, alpha)

Here the alpha parameter defines the opacity. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

Opacity

The CSS3 opacity property sets the opacity for a specified RGB value. The opacity

153a- CSS3 Gradients

CSS3 gradients let you display smooth transitions between two or more specified colors.

- Reduce download time and bandwidth usage.
- Looks good when zoomed

Gradient Types

- Linear Gradients (goes down/ up/ left/ right/ diagonally)
- Radial Gradients (defined by their center)

CSS3 Linear Gradients

To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect. Here is syntax for linear gradient:

background: linear-gradient (direction, color-stop1, color-stop2, ...);

Using Angles

If you want more control over the direction of the gradient, you can define an angle, instead of the predefined directions (to bottom, to top, to right, to left, to bottom right, etc.).

Syntax

background: linear-gradient(angle, color-stop1, color-

stop2); here is a detailed example:

```
#gradient {
    background: -webkit-linear-gradient(180deg, red, blue); /* For Safari 5.1 to 6.0 */
    background: -o-linear-gradient(180deg, red, blue); /* For Opera 11.1 to 12.0 */
    background: -moz-linear-gradient(180deg, red, blue); /* For Firefox 3.6 to 15 */
    background: linear-gradient(180deg, red, blue);
}
```

The angle is specified as an angle between a horizontal line and the gradient line, going counter- clockwise. In other words, 0deg creates a bottom to top gradient, while 90deg generates a left to right gradient.

Using Multiple Color Stops

We can add multiple color stops to make multi color gradient. #gradient {

```
    background: -webkit-linear-gradient(red, green, blue); /* For Safari 5.1 to 6.0 */
```

```
background: -o-linear-gradient(red, green, blue); /* For Opera 11.1 to 12.0 */
background: -moz-linear-gradient(red, green, blue); /* For Firefox 3.6 to 15 */
background: linear-gradient(red, green, blue);
}
```

Using Transparency

CSS3 gradients also support transparency, which can be used to create fading effects. #grad {

```
background: linear-gradient (to right, rgba(255,0,0,0), rgba(255,0,0,1));
}
```

Repeating a linear-gradient

The repeating-linear-gradient() function is used to repeat linear gradients. #grad {

```
background: repeating-linear-gradient (red, yellow 10%, green 20%);
}
```

CSS3 Radial Gradients

A radial gradient is defined by its center. To create a radial gradient you must also define at least two color stops. Check here an **Example of Radial Gradient:**

background: radial-gradient(shape size at position, start-color, ..., last-color);

Radial Gradient - Differently Spaced Color Stops:

#grad {

```
background: radial-gradient(red 5%, green 15%, blue 60%);
}
```

Set Shape

The shape parameter defines the shape. It can take the value circle or ellipse. The default value is ellipse.

#grad {

```
background: radial-gradient(circle, red, yellow, green);
}
```

Size of Gradient

The size parameter defines the size of the gradient. It can take four values:

- closest-side
- farthest-side
- closest-corner
- farthest-corner

Repeating a radial-gradient

The repeating-radial-gradient() function is used to repeat radial gradients. #grad {

```
background: repeating-radial-gradient(red, yellow 10%, green 15%);
}
```

154- CSS3 Shadow Effects

Shadow Effects

With CSS3 you can add shadow to text and to elements.

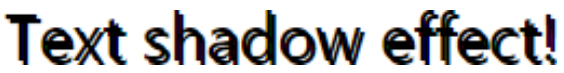
Using properties:

- text-shadow
- box-shadow

CSS3 Text Shadow

The CSS3 text-shadow property applies shadow to text. In its simplest use, you only specify the horizontal and vertical shadow:

```
h1 {
text-shadow: 2px 2px;
}
```



Add a color to the shadow:

```
h1 {
text-shadow: 2px 2px red;
}
```

Add a blur effect to the shadow: h1 {

```
    text-shadow: 2px 2px 5px red;
}
```

White text with black shadow

```
h1 {
    color: white;
    text-shadow: 2px 2px 4px #000000;
}
```

A red neon glow shadow:

```
h1 {
    text-shadow: 0 0 3px #FF0000;
}
```

Multiple Shadows

To add more than one shadow to the text, you can add a comma-separated list of shadows.

```
h1 {
    text-shadow: 0 0 3px #FF0000,
                 0 0 5px #0000FF;
}
```

CSS3 box-shadow Property

The CSS3 box-shadow property applies shadow to elements. In its simplest use, you only specify the horizontal shadow and the vertical shadow.

This is a yellow element with a black box-shadow

```
div {
    box-shadow: 10px 10px;
}
```

Add a color to the shadow:

```
div {
    box-shadow: 10px 10px grey;
}
```

Add a blur effect to the shadow:

```
div {
        box-shadow: 10px 10px 5px grey;
    }
```

155- Text Effects and Fonts

CSS3 Text

CSS3 contains several new text features.

- text-overflow
- word-wrap
- word-break

CSS3 Text Overflow

The CSS3 text-overflow property specifies how overflowed content that is not displayed should be signaled to the user.

text-overflow: clip:



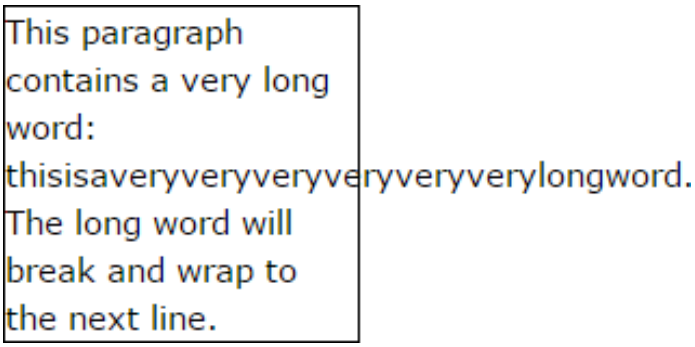
text-overflow: ellipsis:



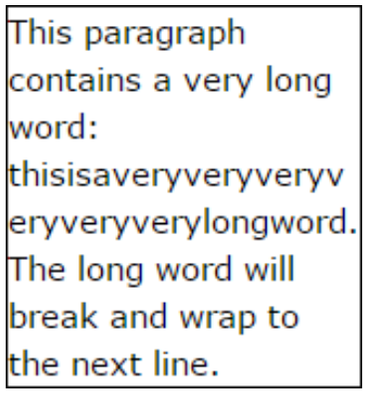
CSS3 Word Wrapping

The CSS3 word-wrap property allows long words to be able to be broken and wrap onto the next line.

If a word is too long to fit within an area, it expands outside:



The word-wrap property allows you to force the text to wrap - even if it means splitting it in the middle of a word.



Example:

```
p {
   word-wrap: break-word;
}
```

CSS3 Word Breaking

The CSS3 word-break property specifies line breaking rules: p.test1 {

```
   word-break: keep-all;
}
```

p.test2 {

```
   word-break: break-all;
}
```

CSS3 Web Fonts

With CSS3, web designers are no longer forced to use only web-safe fonts. It allows Web designers to use fonts that are not installed on the user's computer.

Define your fonts using the @font-face Rule.
Just include the font file on your web server.
And use these fonts normally as Web Safe fonts.

Different Font Formats

There are multiple formats of fonts:

TrueType Fonts (TTF)

TrueType is a font standard developed in the late 1980s, by Apple and Microsoft. TrueType is the most common font format for both the Mac OS and Microsoft Windows operating systems.

OpenType Fonts (OTF)

OpenType is a format for scalable computer fonts. It was built on TrueType, and is a registered trademark of Microsoft. OpenType fonts are used commonly today on the major computer platforms.

The Web Open Font Format (WOFF)

Format for use in web pages. Developed in 2009, and is now a W3C Recommendation. WOFF is essentially OpenType or TrueType with compression and additional metadata.

The goal is to support font distribution from a server to a client over a network with bandwidth constraints.

The Web Open Font Format (WOFF 2.0)

TrueType/OpenType font that provides better compression than WOFF 1.0.

SVG Fonts/Shapes

It allows SVG to be used as glyphs when displaying text. The SVG 1.1 specification define a font module that allows the creation of fonts within an SVG document.

You can also apply CSS to SVG documents, and the @font-face rule can be applied to text in SVG documents.

Embedded OpenType Fonts (EOT)

EOT fonts are a compact form of OpenType fonts designed by Microsoft for use as embedded fonts on web pages.

Using The Font You Want

In the CSS3 @font-face rule you must first define a name for the font, and then point to the font file.

```
@font-face {
    font-family: myFirstFont;
    src:
    url(sansation_light.woff);
}
div {
    font-family: myFirstFont;
}
```

156- CSS3 Transforms

A transformation is an effect that lets an element change shape, size and position. CSS3 supports 2D and 3D transformations. Here are some 2D transformation methods:

- translate()
- rotate()
- scale()
- skewX()
- skewY()
- matrix()

The translate() Method

The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

This example moves the element 50 pixels to the right, and 100 pixels down from its current position:

```
div {
        transform: translate(50px,100px);
    }
```

The rotate() Method

The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

- transform: rotate(20deg);
- transform: rotate(-20deg);

The scale() Method

The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).

- transform: scale(2,3);
- transform: scale(0.5,0.5);

The skewX() Method

The skewX() method skews an element along the X-axis by the given angle. transform:

```
skewX(20deg);
```

The skewY() Method

The skewY() method skews an element along the Y-axis by the given angle. transform:

```
skewY(20deg);
```

The skew() Method

The skew() method skews an element along the X and Y-axis by the given angles.

transform: skew(20deg, 10deg);

The matrix() Method

The matrix() method combines all the 2D transform methods into one. The matrix() method take six parameters, containing mathematic functions, which allows you to rotate, scale, move (translate), and skew elements:

transform: matrix(1, -0.3, 0, 1, 0, 0);

CSS3 3D Transforms

CSS3 allows you to format your elements using 3D transformations

too. 3D transformation methods:

- rotateX()
- rotateY()
- rotateZ()

The rotateX() Method

The rotateX() method rotates an element around its X-axis at a given degree. transform:

rotateX(150deg);

The rotateY() Method

The rotateY() method rotates an element around its Y-axis at a given degree. transform:

rotateY(130deg);

The rotateZ() Method

The rotateZ() method rotates an element around its Z-axis at a given degree. transform:

rotateZ(90deg);

CSS3 3D Transform Properties

Property	Description
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements

transform-style	Specifies how nested elements are rendered in 3D space
perspective	Specifies the perspective on how 3D elements are viewed
perspective-origin	Specifies the bottom position of 3D elements
backface-visibility	Defines whether or not an element should be visible when not facing the screen

3D Transform Methods

Function	Description
matrix3d (n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n)	Defines a 3D transformation, using a 4x4 matrix of 16 values
translate3d(x,y,z)	Defines a 3D translation
translateX(x)	Define 3D translation, using only the value for the X-axis
translateY(y)	Define 3D translation, using only the value for the Y-axis
translateZ(z)	Defines a 3D translation, using only the value for the Z-axis
scale3d(x,y,z)	Define 3D scale transformation
scaleX(x)	Define 3D scale transformation by giving a value for the X-axis
scaleY(y)	Define 3D scale transformation by giving a value for the Y-axis

scaleZ(z)	Define 3D scale transformation by giving a value for the Z-axis
rotate3d(x,y,z,angle)	Defines a 3D rotation
rotateX(angle)	Defines a 3D rotation along the X-axis
rotateY(angle)	Defines a 3D rotation along the Y-axis

157- CSS3 Transitions

CSS3 transitions allows us to change property values smoothly (from one value to another), over a given duration.

How to Use CSS3 Transitions?

To create a transition effect, you must specify two things:

- The CSS property you want to add an effect to
- The duration of the effect

Example:

We specified a transition effect for the width property, with a duration of 2 seconds: div {

```
width: 100px;
height: 100px;
background: red;
transition: width
2s;

}
```

Example:

We specified a transition effect for the width property, with a duration of 2 seconds: div {

```
width: 100px;
height: 100px;
background:
red;
-webkit-transition: width 2s; /* Safari
*/ transition: width 2s;
```

```
}
```

The transition effect will start when the specified CSS property (width) changes value. Transition effect can be applied to multiple properties.

```
div {
        transition: width 2s, height 4s;
}
```

Specify the Speed Curve of the Transition

The **transition-timing-function** property specifies the speed curve of the transition effect.

transition-timing-function values:

- ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- linear - specifies a transition effect with the same speed from start to end
- ease-in - specifies a transition effect with a slow start
- ease-out - specifies a transition effect with a slow end
- ease-in-out - specifies a transition effect with a slow start and end
- cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function

Delay the Transition Effect

The **transition-delay** property specifies a delay (in seconds) for the transition effect. div {

-webkit-transition-delay: 1s; /* Safari

*/ transition-delay: 1s;

}

Transition with Transformation

We can implement a transformation with a transitional effect too.

158- CSS3 Animations

CSS3 animations allows animation of most the HTML elements without using JavaScript or Flash!

What are CSS3 Animations?

An animation lets an element gradually change from one style to another. You can change as many CSS properties you want, as many times you want.

To use CSS3 animation, you must first specify keyframes for the animation. Keyframes hold what styles the element will have at certain time.

Example:

@keyframes anim1 {

from {background-color:

red;} to {background-color:

blue;}

}

You can specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain time.

Example:

```
div {
        width: 100px; height:
        100px; background-color:
        red; animation-name:
        anim1; animation-
        duration: 5s;
    }
```

To get an animation to work, you must bind the animation to an element. Example:

Example

```
div {
        width: 100px; height:
        100px; background-color:
        red; animation-name:
        anim1; animation-
        duration: 5s;
    }
```

If animation-duration is not set, there will be no animation effect as its default value is 0s.

Adding Steps in Animation

You can add percentage to define multiple keyframes. @keyframes anim2{

```
        0%   {background-color: red;}
        25%  {background-color: yellow;}
        50%  {background-color: blue;}
        100% {background-color: green;}
    }
```

Delay an Animation

The **animation-delay** property specifies a delay for the start of an animation. div {

width: 100px; height:

100px; background-color:

red;

```
animation-name:
anim1; animation-
duration: 4s;
animation-delay: 2s;

}
```

Run Animation multiple times

The **animation-iteration-count** property specifies the number of times an animation should run. div {

```
width: 100px; height: 100px;
background-color: red;
animation-name: anim1;
animation-duration: 4s;
animation-iteration-count:
5;

}
```

Reverse Animation or Alternate Cycles

The **animation-direction** property is used to let an animation run in reverse direction or alternate cycles.

```
div {
    width: 100px; height:
    100px; background-color:
    red; animation-name:
    anim1; animation-
    duration: 4s; animation-
    iteration-count: 5;
    animation-direction:
    reverse;
```

Speed Curve of the Animation

The **animation-timing-function** property specifies the speed curve of the animation. The animation-timing-function property can have the following values:

animation-timing-function values:

- ease - specifies an animation with a slow start, then fast, then end slowly (this is default)
- linear - specifies an animation with the same speed from start to end

- ease-in - specifies an animation with a slow start
- ease-out - specifies an animation with a slow end
- ease-in-out - specifies an animation with a slow start and end
- cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function

159- CSS3 Multi-column Layout

The CSS3 multi-column layout allows easy definition of multiple columns of text.

CSS3 Create Multiple Columns

The column-count property specifies the number of columns an element should be divided into. The following example will divide the text in the element into 3 columns div {

```
column-count: 3;
}
```

The following example will divide the text in the element into 3 columns div {

```
/* Chrome, Safari, Opera */
-webkit-column-count: 3;
/* Firefox */
-moz-column-count: 3;
column-count: 3;
}
```

Gap Between Columns

The **column-gap** property specifies the gap between the columns. The following example specifies a 40 pixels gap between the columns:

```
div {
-webkit-column-gap: 40px;
-moz-column-gap:
40px; column-gap:
40px;
}
```

Column Rules

The **column-rule-style** property specifies the style of the rule between

columns: div {

```
-webkit-column-rule-style: solid;
-moz-column-rule-style:
solid; column-rule-style:
solid;
}
```

The **column-rule-width** property specifies the width of the rule between

```
columns div {
-webkit-column-rule-width: 1px;
-moz-column-rule-width:
1px; column-rule-width:
1px;
}
```

The **column-rule-color** property specifies the color of the rule between

```
columns div {
-webkit-column-rule-color: blue;
-moz-column-rule-color:
blue; column-rule-color:
blue;
}
```

The **column-rule** property is a shorthand property for setting all the column-rule div {

```
-webkit-column-rule: 1px solid blue;
-moz-column-rule: 1px solid
blue; column-rule: 1px solid
blue;
}
```

Column Span

```
    -webkit-column-span:
    all; column-span: all;
}
```

Column Width

The **column-width** property specifies a suggested, optimal width for the columns. div {

```
-webkit-column-width:
100px; column-width:
100px;
```

}

RWD: Responsive Web Design

Introduction

Responsive web design (RWD) is an approach to web design aimed at crafting sites to provide an optimal viewing and interaction experience with a purpose of easy reading and navigation with a minimum of

resizing, panning, and scrolling on all devices. The concept of responsive web design is becoming more important as the amount of mobile traffic now accounts for more than half of total internet traffic.

Let's have some more discussion on Responsive Web Design:

Designing for the Best Experience for All Users

Web pages can be viewed using many different devices: desktops, tablets, and phones. Your web page should look good, and be easy to use, regardless of the device.

Designing For All

Web pages should not leave out information to fit smaller devices, but rather adapt its content to fit any device:

Design/Layout for Desktop

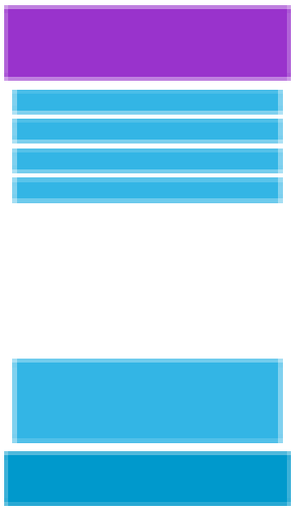


Design/Layout for Tablet





Design/Layout for Phone



It is called responsive web design when you use CSS and HTML to resize, hide, shrink, enlarge, or move the content to make it look good on any screen.

162- Responsive Web Design (RWD) - The Viewport

The Viewport

The viewport is the user's visible area of a web page. The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen. Before tablets and mobile phones, web pages were designed only for computer screens, and it was common for web pages to have a static design and a fixed size.

When using tablets and mobile phones, fixed size web pages are too large to fit the viewport. A quick fix is to scale down the whole webpage.

Setting the Viewport

HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.

You should include the following <meta> viewport element in all your web pages:

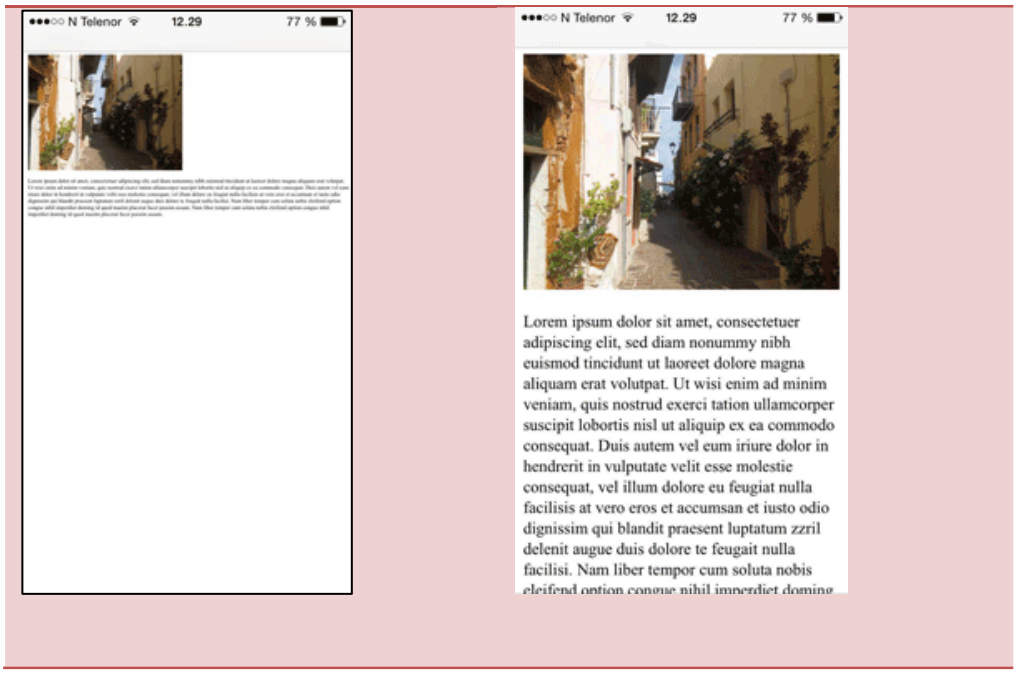
```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- A <meta> viewport element gives the browser instructions on how to control the page's dimensions and scaling.
- The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
- The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

You can see difference in a webpage with viewport define, when browsing using a mobile device.

Without the viewport meta tag **With the viewport meta tag**

© Virtual University of Pakistan



Size Content to The Viewport

Users are used to scroll websites vertically on both desktop and mobile devices - but not horizontally!

So, if the user is forced to scroll horizontally, or zoom out, to see the whole web page it results in a poor user experience.

Some Rules

1. Do NOT use large fixed width elements

For example, if an image is displayed at a width wider than the viewport it can cause the viewport to scroll horizontally.

Remember to adjust content to fit within the width of the viewport.

2. Do NOT let the content rely on a particular viewport width to render well

Since screen dimensions and width in CSS pixels vary widely between devices, content should not rely on a particular viewport width to render well.

3. Use CSS media queries to apply different styling for small and large screens

Setting large absolute CSS widths for page elements will cause the element to be too wide for the viewport, consider using relative width values, such as width: 100%.

Also, be careful of using large absolute positioning values. It may cause the element to fall outside the viewport on small devices.

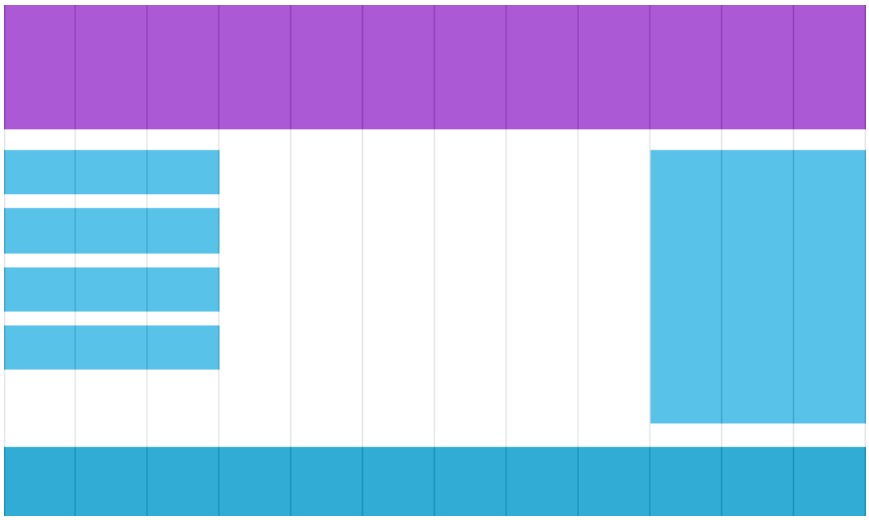
163- Responsive Web Design (RWD) Grid-View

What is a Grid-View?

Consider dividing your web page in a grid structure.



Using a grid-view is very helpful when designing web pages. It makes it easier to place elements on the page.



A responsive grid-view often has 12 columns, and has a total width of 100%, and will shrink and expand as you resize the browser window.

Building a Responsive Grid-View

All HTML elements have the box-sizing property set to border-box. This makes sure that the padding and border are included in the total width and height of the elements.

© Virtual University of Pakistan

```
* {
    box-sizing: border-box;
}
```

The following example shows a simple responsive web page, with two columns: Two columns responsive webpage:

```
.menu {
    width:
    25%; float:
    left;
}
.main {
    width:
    75%; float:
    left;
}
```

We want to use a responsive grid-view with 12 columns, to have more control over the web page. First we must calculate the percentage for one column: 100% / 12 columns = 8.33%.

Then we make one class for each of the 12 columns, class="col-" and a number defining how many columns the section should span.

```
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
```

.col-11 {width: 91.66%;}

.col-12 {width: 100%;}

© Virtual University of Pakistan

All these columns should be floating to the left, and have a padding of

```
15px: [class*="col-"] {
  float: left;
  padding:
  15px;
  border: 1px solid red;
}
```

Each row should be wrapped in a . The number of columns inside a row should allways add up to 12.

```
<div class="row">
  <div class="col-3">...</div>
  <div class="col-9">...</div>
</div>
```

The columns inside a row are all floating to the left, and are therefore taken out of the flow of the page, and other elements will be placed as if the column does not exist. To prevent this, we will add a style that clears the flow. Try the code below:

CSS:

```
.row:after {
  content: "";
  clear: both;
  display:
  block;
}
```

164- Responsive Web Design (RWD) Media Queries

What is a Media Query?

Media query is a CSS technique introduced in CSS3. It uses the **@media** rule to include a block of CSS properties only if a certain condition is true.

If the browser window is smaller than 500px, the background color will change to light-blue: @media only

```
screen and (max-width: 500px) {
  body {
```

```
    background-color: lightblue;
}
```

```
}
```

Add a Breakpoint

We can add a breakpoint where certain parts of the design will behave differently on each side of the breakpoint.

For screen smaller than 768px, each column should have a width of 100%:

```
.col-1 {width: 8.33%;}  .col-2 {width: 16.66%;}

.col-3 {width: 25%;}

@media only screen and (max-width: 768px) {
        /* For mobile phones:
        */ [class*="col-"] {
            width: 100%;
        }
}
```

Mobile First

Mobile First means designing for mobile before designing for desktop or any other device (This will make the page display faster on smaller devices).

This means that we must make some changes in our CSS.

Instead of changing styles when the width gets smaller than 768px, we should change the design when the width gets larger than 768px. This will make our design Mobile First. See the example below:

Example:

```
/* For mobile phones: */ [class*="col-"] {
        width: 100%;
}

@media only screen and (min-width: 768px) {
        /* For desktop: */
        .col-1 {width: 8.33%;}   .col-2 {width: 16.66%;}
        .col-3 {width: 25%;}
```

Adding another Breakpoint

You can add as many breakpoints as you like. We will also insert a breakpoint between tablets and mobile phones.

We do this by adding one more media query (at 600px), and a set of new classes for devices larger than 600px (but smaller than 768px):

```
/* For mobile phones: */ [class*="col-"] {
    width: 100%;
}
```

Example

```
@media only screen and (min-width: 600px) {
    /* For tablets: */
    .col-m-1 {width: 8.33%;}
    .col-m-2 {width: 16.66%;}
    .col-m-3 {width: 25%;}
}
```

Example

```
@media only screen and (min-width: 768px) {
    /* For desktop: */
    .col-1 {width: 8.33%;}
    .col-2 {width: 16.66%;}
    .col-3 {width: 25%;}
}
```

Orientation: Portrait / Landscape

Media queries can also be used to change layout of a page depending on the orientation of the browser.

The web page will have a different background if the orientation is in landscape mode: @media only

```
screen and (orientation: landscape) {
    body {
        background-color: lightblue;
```

```
    }
}
```

165- Responsive Web Design - Images

Using the width Property

If the width property for an image is set to 100%, the image will be responsive and scale up and down.

```
img {
    width:
    100%;
    height: auto;
}
```

Here, the image can be scaled up to be larger than its original size. A better solution, in many cases, will be to use the max-width property instead.

If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size

Background Images

Background images can also respond to resizing and scaling. Let's review three different methods to resize background images.

1. If the **background-size** property is set to **"contain"**, the background image will scale, and try to fit the content area.

However, the image will keep its aspect ratio (the proportional relationship between the image's width and height)

2. If the **background-size** property is set to **"100% 100%"**, the background image will stretch to cover the entire content area.

3. If the **background-size** property is set to **"cover"**, the background image will scale to cover the entire content area.

Note that the "cover" value keeps the aspect ratio, and some part of the background image may be clipped.

Different Images for Different Devices

You can use media queries to display different images on different devices. See the example below:

```
/* For width smaller than 400px: */ body {
    background-image: url('img1.jpg');
```

```
}
/* For width 400px and larger: */
@media only screen and (min-width: 400px) { body {
    background-image: url('img2.jpg');
  }
}
```

Different Images for Different Devices

You can use the media query min-device-width, instead of min-width, which checks the device width, instead of the browser width. Then the image will not change when you resize the browser window:

```
/* For width smaller than 400px: */ body {
    background-image: url('img1.jpg');
}
/* For devices with width 400px and larger: */ @media only screen
and (min-device-width: 400px) { body {
    background-image: url('img2.jpg');
  }
}
```

HTML5 <picture> Element

HTML5 introduced the <picture> element, which lets you define more than one image. The <picture> element works similar to the <video> and <audio> elements.

You set up different sources, and the first source that fits the preferences is the one being used.

```
<picture>
    <source srcset="img1.jpg" media="(max-width: 400px)">
    <source srcset="img2.jpg">
    <img src="img3.jpg" alt="Flowers">
</picture>
```

- The **srcset** attribute is required, and defines the source of the image.
- The **media** attribute is optional, and accepts the media queries you find in CSS @media rule.
- You should also define an **<img>** element for browsers that do not support the <picture> element.

166- Responsive Web Design - Videos

Using the width Property

If the width property is set to 100%, the video player will be responsive and scale up and down video {

width:

100%;

height: auto;

}

Here, the video player can be scaled up to be larger than its original size. A better solution, in many cases, will be to use the max-width property instead.

If the max-width property is set to 100%, the video player will scale down if it has to, but never scale up to be larger than its original size.

167- Responsive Web Design (RDW) Frameworks

There are many existing CSS Frameworks that offer Responsive Design. They are free, and easy to use.

Bootstrap

The most popular Framework for Responsive Web Design is Bootstrap, it uses HTML, CSS and jQuery to make responsive web pages.

http://Getbootstrap.com

Foundation

Another popular Framework is Foundation, it uses HTML, CSS and jQuery to make responsive web pages.

http://foundation.zurb.co

m Skeleton

Another popular framework is Skeleton, it uses only CSS to make responsive web pages.

168- Bootstrap Introduction

What is Bootstrap?

Bootstrap is a free front-end framework for faster and easier web development. Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins. Bootstrap also gives you the ability to easily create responsive designs.

History of Bootstrap

Developed by Mark Otto and Jacob Thornton at Twitter, and released as an open source product in August 2011 on GitHub.

In June 2014 Bootstrap was the No.1 project on GitHub!

Why Use Bootstrap?

- **Easy to use:** Anybody with basic knowledge of HTML and CSS can start using Bootstrap
- **Responsive features:** Bootstrap's responsive CSS adjusts to phones, tablets, and desktops
- **Mobile-first approach:** In Bootstrap 3, mobile-first styles are part of the core framework
- **Browser compatibility:** Compatible with all modern browsers (Chrome, Firefox, IE, Safari, and Opera)

Where to Get Bootstrap?

There are two ways to start using Bootstrap on your own web site.

- Download Bootstrap
- Include from a CDN

Downloading Bootstrap

If you want to download and host Bootstrap yourself, go to **getbootstrap.com**, and follow the instructions there.

Bootstrap CDN

```
<!-- Latest compiled and minified CSS -->

<link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">

<!-- jQuery library -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

<!-- Latest compiled JavaScript -->

<script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
```

169- Bootstrap Grids

Bootstrap Grid System

Bootstrap's grid system allows up to 12 columns across the page. If you do not want to use all 12 columns individually, you can group the columns together to create wider columns.

Bootstrap's grid system is responsive, and the columns will re-arrange automatically depending on the screen size.

Grid Classes

Bootstrap Grid system has four classes:

- xs (for phones)
- sm (for tablets)
- md (for desktops)
- lg (for larger desktops)

These classes can be combined to create more dynamic and flexible layouts.

Basic Structure of a Bootstrap Grid

The following is a basic structure of a Bootstrap grid:

```
<div class="row">
        <div class="col-*-*"></div>
</div>
<div class="row">
        <div class="col-*-*"></div>
        <div class="col-*-*"></div>
        <div class="col-*-*"></div>
</div>
<div class="row">
</div>
```

First; create a row ( ). Then, add the desired number of columns (tags with appropriate .col-*-* classes).

```
<div class="row">
        <div class="col-*-*"></div>
        <div class="col-*-*"></div>
        <div class="col-*-*"></div>
</div>
```

Numbers in .col-*-* should always add up to 12 for each row.

170- Bootstrap Text/Typography

Bootstrap Defaults

Let's look at some HTML elements that will be styled a little bit different by Bootstrap than browser defaults.

Bootstrap's global default font-size is 14px, with a line-height of 1.428.

This is applied to the <body> and all paragraphs.

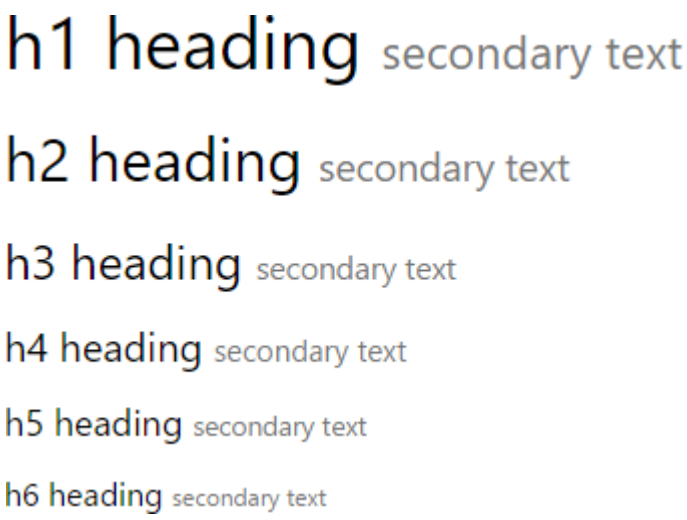
All <p> elements have a bottom margin that equals half their computed line-height (10px by default).

<h1> - <h6>

By default, Bootstrap will style the HTML headings (<h1> to <h6>) differently.

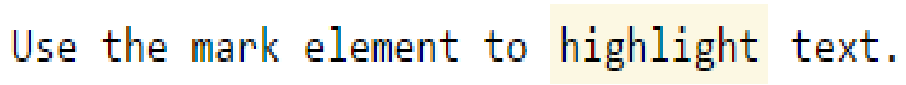
<small>

In Bootstrap the HTML <small> element is used to create a lighter, secondary text in any heading.



<mark>

Bootstrap will style the HTML <mark> element to highlight text.



<abbr>

The WHO was founded in 1948.

© Virtual University of Pakistan

<blockquote>

Bootstrap will style the HTML <blockquote> element in the following way:

> For 50 years, WWF has been protecting the future of nature. The world's leading conservation organization, WWF works in 100 countries and is supported by 1.2 million members in the United States and close to 5 million globally.
>
> — From WWF's website

To show the quote on the right, use the .blockquote-reverse class:

> For 50 years, WWF has been protecting the future of nature. The world's leading conservation organization, WWF works in 100 countries and is supported by 1.2 million members in the United States and close to 5 million globally.
>
> From WWF's website —

<dl>

Bootstrap will style the HTML <dl> element in the following way:

Coffee
- black hot drink

Milk
- white cold drink

<code>

Bootstrap will style the HTML <code> element in the following way:

Here we use code tag: `a = b + c` and `x = a - z`

<kbd>

Bootstrap will style the HTML <kbd> element in the following way:

Use `ctrl + p` to open the Print dialog box.

<pre>

Bootstrap will style the HTML <pre> element in the following way:

```
Text in a pre element
is displayed in a fixed-width
font, and it preserves
both      spaces and
line breaks.
```

Contextual Colors and Backgrounds

Bootstrap also has some contextual classes that can be used to provide "meaning through colors". The classes for text colors are:

.text-muted

.text-primary

.text-success

.text-info

.text-warning

.text-danger

The classes for background colors are:

.bg-primary

.bg-success

.bg-info

.bg-warning

.bg-danger

More Typography Classes

.lead

Makes a paragraph stand out

.text-lowercase

Indicates lowercased text

.text-uppercase

Indicates uppercased text

.text-capitalize

Indicates capitalized text

.list-inline

Places all list items on a single line

.pre-scrollable

Makes a <pre> element scrollable

171- Bootstrap Tables

Bootstrap Basic Table

A basic Bootstrap table has a light padding and only horizontal dividers. The **.table** class adds basic styling to a table.

Striped Rows

The .table-striped class adds zebra-stripes to a table

Bordered Table

The .table-bordered class adds borders on all sides of the table and cells

Hover Rows

The .table-hover class enables a hover state on table rows

Condensed Table

The .table-condensed class makes a table more compact by cutting cell padding in half.

Contextual Classes

Contextual classes can be used to color table rows (<tr>) or table cells (<td>)

The contextual classes that can be used are:

Class	Description
.active	Applies the hover color to the table row or table cell
.success	Indicates a successful or positive action
.info	Indicates a neutral informative change or action
.warning	Indicates a warning that might need attention
.danger	Indicates a dangerous or potentially negative action

Responsive Table

The .table-responsive class creates a responsive table. The table will then scroll horizontally on small devices (under 768px). When viewing on anything larger than 768px wide, there is no difference. See the example here:

```
<div class="table-responsive">
        <table class="table">
          ...
        </table>
      </div>
```

172- Bootstrap Images

Bootstrap Image Shapes

Bootstrap allows us to display images in different shapes.



Rounded Corners

The .img-rounded class adds rounded corners to an image (IE8 do not support rounded corners): class="img-rounded"

Circle

The .img-circle class shapes the image to a circle (IE8 do not support rounded corners): class="img-

circle"

Thumbnail

The .img-thumbnail class shapes the image to a thumbnail: class="img-

thumbnail”

Responsive Images

Images comes in all sizes. So do screens. Responsive images automatically adjust to fit the size of the screen. Create responsive images by adding

.img-responsive class to the <img> tag.

The image will then scale nicely to the parent element. The .img-responsive class applies max- width: 100%; and height: auto; to the image:

class="img-responsive"

Responsive Embeds

Bootstrap also let videos or slideshows scale properly on any device. Classes can be applied directly to <iframe>, <embed>, <video>, and <object> elements. By adding an .embed-responsive-item class to an <iframe> tag the video will scale nicely to the parent element.

The containing defines the aspect ratio of the video.

```
<!-- 16:9 aspect ratio -->
<div class="embed-responsive embed-responsive-16by9">
    <iframe class="embed-responsive-item" src="..."></iframe>
</div>


<!-- 4:3 aspect ratio -->
<div class="embed-responsive embed-responsive-4by3">
    <iframe class="embed-responsive-item" src="..."></iframe>
</div>
```

173- BS Jumbotron

Creating a Jumbotron

A jumbotron indicates a big box for calling extra attention to some special content or information. In Bootstrap a jumbotron is displayed as a grey box with rounded corners. It also enlarges the font

© Virtual University of Pakistan

sizes of the text inside it. Inside a jumbotron you can put nearly any valid HTML, including other Bootstrap elements/classes.

Use a element with class .jumbotron to create a jumbotron.

Place the jumbotron inside the

 if you want the jumbotron to NOT extend to the edge of the screen. See this example:

```
<div class="container">
        <div class="jumbotron">
          <h1>Title for you</h1>
          <p>any content here</p>
        </div>
        <p>This is another text.</p>
</div>
```

Jumbotron Outside Container

Place the jumbotron outside the if you want the jumbotron to extend to the screen edges.

Page Header

A page header is like a section divider. The **.page-header** class adds a horizontal line under the heading (+ adds some extra space around the element).

Use a element with class .page-header to create a page header

```
<div class="page-header">
        <h1>Example Page Header</h1>
</div>
```

174- Buttons and Button Groups

Button Styles

Bootstrap provides seven styles of buttons.

Button Styles

Bootstrap has the following classes for button styles:

.btn-default

.btn-primary

.btn-success

.btn-info

.btn-warning

.btn-danger

.btn-link

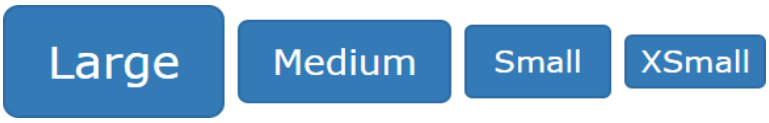


The button classes can be used on an <a>, <button>, or <input> element:

```
<a href="#" class="btn btn-info" role="button"> Link Button</a>
<button type="button" class="btn btn-info"> Button</button>
<input type="button" class="btn btn-info" value="Input Button">
<input type="submit" class="btn btn-info" value="Submit Button">
```

Button Sizes

Bootstrap provides four button sizes:



.btn-lg

.btn-md

.btn-sm

.btn-xs

Active/Disabled Buttons

A button can be set to an active (appear pressed) or a disabled (unclickable) state.

The class **.active** makes a button appear pressed, and the class **.disabled** makes a



© Virtual University of Pakistan

Button Groups

Bootstrap allows you to group a series of buttons together (on a single line) in a button group.

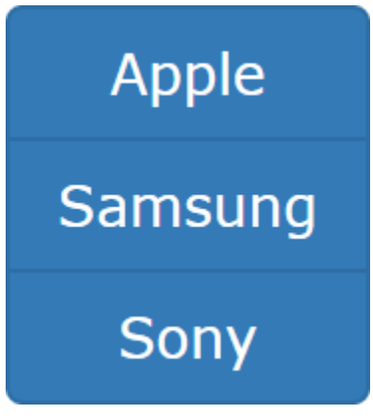
Use a element with class .btn-group to create a button group

```
<div class="btn-group">
        <button type="button" class="btn btn-primary"> Apple</button>
        <button type="button" class="btn btn-primary"> Samsung</button>
        <button type="button" class="btn btn-primary"> Sony</button>
</div>
```



Vertical Button Groups

Use the class .btn-group-vertical to create a vertical button group.



Justified Button Groups

To span the entire width of the screen, use the .btn-group-justified class.

Button Element

For <button> elements, you must wrap each button in a .btn-group class

```
<div class="btn-group btn-group-justified">
  <div class="btn-group">
```

```
<button type="button" class="btn btn-primary"> Apple</button>
</div>
<div class="btn-group">
<button type="button" class="btn btn-primary"> Samsung</button>
</div>
</div>
```

Nesting Button Groups & Dropdown Menus

You can nest button groups to create dropdown menus.

```
<div class="btn-group">
<button type="button" class="btn btn-primary">Apple</button>
<button type="button" class="btn btn-primary">Samsung</button>
<div class="btn-group">
<button type="button" class="btn btn-primary dropdown-toggle" data- toggle="dropdown">
Sony <span class="caret"></span></button>
<ul class="dropdown-menu" role="menu">
<li><a href="#">Tablet</a></li>
<li><a href="#">Smartphone</a></li>
</ul>
</div></div>
```

Split Button Dropdowns

```
<div class="btn-group">
<button type="button" class="btn btn-primary"> Sony</button>
<button type="button" class="btn btn-primary dropdown-toggle" data- toggle="dropdown">
<span class="caret"></span>
</button>
<ul class="dropdown-menu" role="menu">
```

```
<li><a href="#">Tablet</a></li>
<li><a href="#">Smartphone</a></li>
</ul>
</div>
```

175- Bootstrap Glyphicons

Glyphicons

Bootstrap includes 260 glyphs from the Glyphicons.com Halflings set. Glyphicons Halflings are made available for Bootstrap free of cost.

Syntax

A glyphicon is inserted with the following syntax:

```
<span class="glyphicon glyphicon-name"> </span>
```

The name part in the syntax above must be replaced with the proper name of the glyphicon. Check below example for **Bootstrap Glyph:**

```
<p>Envelope icon:
<span class="glyphicon glyphicon-envelope"> </span></p>
<p>Search icon:
<span class="glyphicon glyphicon-search"> </span></p>
<p>Print icon:
<span class="glyphicon glyphicon-print"> </span></p>
```

176- Bootstrap Progress Bars

Basic Progress Bar

A progress bar can be used to show a user how far along he/she is in a process. Bootstrap provides several types of progress bars. To create a default progress bar, add a .progress class to a

 element:

```
<div class="progress">
<div class="progress-bar" role="progressbar" aria-
valuenow="70" aria-valuemin="0" aria-valuemax="100"
style="width:70%">
</div>
</div>
```

A default progress bar in Bootstrap looks like this:



Contextual Progress Bars

Contextual classes can be used to provide "meaning through colors".

.progress-bar-success

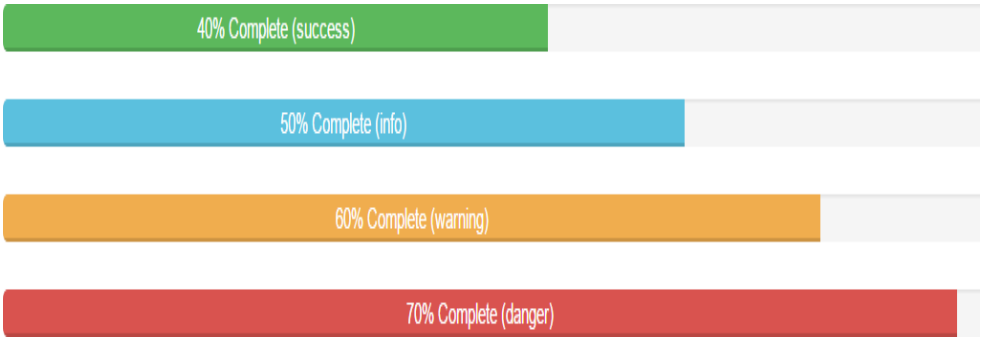
.progress-bar-info

.progress-bar-warning

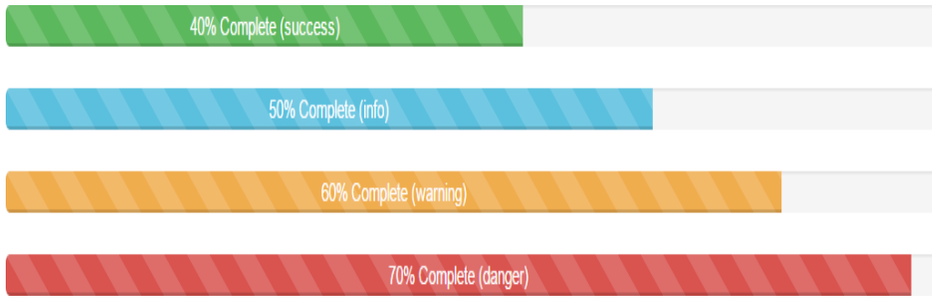
.progress-bar-danger

See the example below:

```
<div class="progress">
    <div class="progress-bar progress-bar-info" role="progressbar" aria-
    valuenow="50" aria-valuemin="0" aria-valuemax="100" style="width:50%">
     50% Complete (info)
    </div>
</div>
```



Striped Progress Bars



Animated Progress Bar

You can add class .active to animate the progress bar.

```
<div class="progress">
        <div class="progress-bar progress-bar-striped active" role="progressbar"
        aria-valuenow="40" aria-valuemin="0" aria-valuemax="100"
         style="width:40%"> 40%
        </div>
</div>
```

Stacked Progress Bars

Progress bars can also be stacked. Create a stacked progress bar by placing multiple bars into the same

<div

class="progress"> See the

example below:

```
<div class="progress">
        <div class="progress-bar progress-bar-success" role="progressbar"
         style="width:40%"> Free Space  </div>
 <div          class="progress-bar    progress-bar-warning"
               role="progressbar" style="width:10%">
         Warning  </div>
       <div class="progress-bar progress-bar-danger" role="progressbar"
         style="width:20%"> Danger  </div>
```

```
</div>
```



177- Bootstrap Pagination

Basic Pagination

If you have a web site with lots of pages, you may wish to add some sort of pagination to each page.

Basic Pagination

A basic pagination in Bootstrap looks like this:



To create a basic pagination, add the .pagination class to an <ul> element:

```
<ul class="pagination">
    <li><a href="#">1</a></li>
    <li><a href="#">2</a></li>
    <li><a href="#">3</a></li>
    <li><a href="#">4</a></li>
    <li><a href="#">5</a></li>
</ul>
```

Active State

The active state shows what is the current page:

```
<ul class="pagination">
    <li><a href="#">1</a></li>
    <li class="active"><a href="#">2</a></li>
    <li><a href="#">3</a></li>
    <li><a href="#">4</a></li>
```

```
                <li><a href="#">5</a></li>
</ul>
```

Disabled State

Add class .disabled if a link for some reason is disabled

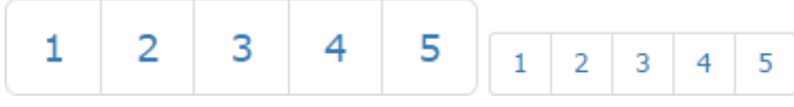
```
<ul class="pagination">
    <li><a href="#">1</a></li>
    <li class="disabled"><a href="#">2</a></li>
    <li><a href="#">3</a></li>
    <li><a href="#">4</a></li>
    <li><a href="#">5</a></li>
</ul>
```

Pagination Sizing

Pagination blocks can also be sized to a larger size or a smaller size.

Pagination Sizing

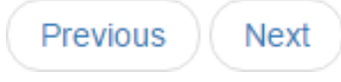
Add class **.pagination-lg** for larger blocks or **.pagination-sm** for smaller blocks.



Bootstrap Pager

Pager is also a form of pagination. Pager provides previous and next buttons (links). To create previous/next buttons, add the .pager class to an <ul> element:

```
<ul class="pager">
    <li><a href="#">Previous</a></li>
    <li><a href="#">Next</a></li>
</ul>
```



Align Buttons

Use the .previous and .next classes to align each button to the sides of the page

<ul class="pager">

© Virtual University of Pakistan

```
<li class="previous"><a href="#">Previous</a></li>
<li class="next"><a href="#">Next</a></li>
</ul>
```



178- Bootstrap List Groups

List Groups

Bootstrap offer classes to stylized Lists. List groups are a flexible and powerful component for displaying not only simple lists of elements, but complex ones with custom content.

Basic List

The most basic list group is an unordered list with list items. To create a basic list group, use an <ul> element with class .list-group, and <li> elements with class .list-group-item:

```
<ul class="list-group">
<li class="list-group-item">First item</li>
<li class="list-group-item">Second item</li>
<li class="list-group-item">Third item</li>
</ul>
```

List Group with Badges

You can also add badges to a list group. The badges will automatically be positioned on the right.

List Group with Badges

To create a badge, create a element with class .badge inside the list item:

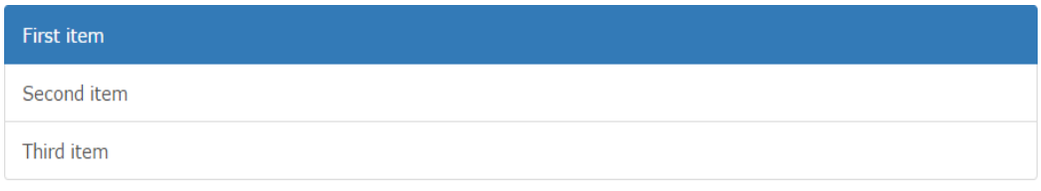
```
<ul class="list-group">
<li class="list-group-item"><span class="badge">12</span> New</li>
<li class="list-group-item"><span class="badge">5</span> Deleted</li>
<li class="list-group-item"><span class="badge">3</span> Warnings</li>
</ul>
```



List Group with Linked Items

The items in a list group can also be hyperlinks. To create a list group with linked items, use instead of <ul> and <a> instead of

```
<div class="list-group">
        <a href="#" class="list-group-item active">First item</a>
        <a href="#" class="list-group-item">Second item</a>
        <a href="#" class="list-group-item">Third item</a>
</div>
```



Disabled Item

To disable an item, add the .disabled class

```
<div class="list-group">
        <a href="#" class="list-group-item disabled"> First item</a>
        <a href="#" class="list-group-item"> Second item</a>
        <a href="#" class="list-group-item"> Third item</a>
</div>
```

Contextual Classes

Contextual classes can also be used to color list items. The classes for coloring list-items are:

.list-group-item-success

.list-group-item-info

.list-group-item-warning

© Virtual University of Pakistan

.list-group-item-danger

Custom Content

You can add nearly any HTML inside a list group item.

Bootstrap provides the classes .list-group-item-heading and .list-group-item-text which can be used as follows:

```
<div class="list-group">
    <a href="#" class="list-group-item active">
     <h4 class="list-group-item-heading"> First List Group Item Heading</h4>
     <p class="list-group-item-text"> List Group Item Text</p>
    </a></div>
```

179- Bootstrap Forms

Form controls automatically receive some global styling with Bootstrap.

Bootstrap's Default Settings

All textual <input>, <textarea>, and <select> elements with class

.form-control have a width of 100%.

Bootstrap Form Layouts

Bootstrap provides three types of form layouts:

- Vertical form (this is default)
- Horizontal form
- Inline form

Standard rules for all three form layouts:

- Always use <form role="form"> (helps improve accessibility for people using screen readers)
- Wrap labels and form controls in (needed for optimum spacing)
- Add class .form-control to all textual <input>, <textarea>, and <select> elements

Bootstrap Vertical Form (default)

Lets create a vertical form with two input fields, one checkbox, and a submit button.

Bootstrap Inline Form

In an inline form, all of the elements are inline, left-aligned, and the labels are alongside. This only applies to forms within viewports that are at least 768px wide!

Additional rule for an inline form:

- Add class .form-inline to the <form> element

Bootstrap Horizontal Form

A horizontal form stands apart from the other forms both in the amount of markup, and in the presentation of the form.

Additional rules for a horizontal form:

- Add class .form-horizontal to the <form> element
- Add class .control-label to all <label> elements

Use Bootstrap's predefined grid classes to align labels and groups of form controls in a horizontal layout.

180- Bootstrap Carousel

The Carousel Plugin

The Carousel plugin is a component for cycling through elements, like a carousel (slideshow). Plugins can be included individually (using Bootstrap's individual **"carousel.js"** file), **OR** all at once (using "bootstrap.js" or "bootstrap.min.js").

A typical carousal code will have following part:

- The outermost
- The "Indicators"
- The "Wrapper for slides"
- The "Left and right controls"

The outermost

- …

   The class="carousel" specifies that this contains a carousel.

- …

The .slide class adds a CSS transition and animation effect, which makes the items slide when showing a new item. Omit this class if you do not want this effect.

- <div id="myCarousel" class="carousel slide" **data-ride="carousel"**> …

The data-ride="carousel" attribute tells Bootstrap to begin animating the carousel immediately when the page loads.

The "indicators"

The indicators are the little dots at the bottom of each slide (which indicates how many slides there are in the carousel, and which slide the user is currently viewing).

<ol class="**carousel-indicators**">

<li **data-target="#myCarousel"**

data-slide-to="0" class="active">

<li **data-**

target="#myCarousel"

data-slide-to="1"> …

- The indicators are specified in an ordered list with class .carousel-indicators.
- The data-slide-to attribute specifies which slide to go to, when clicking on the specific dot.

The "Wrapper for slides"

This section contains the actual slides to display.

<div **class="carousel-inner"** role="listbox">

<img src="180-pic1.jpg" alt="Chania" width="460" height="345">

…

- The slides are specified in a with class .carousel-inner
- The content of each slide is defined in a with class .item. This can be text or images.
- The .active class needs to be added to one of the slides. Otherwise, the carousel will not be visible.

The "Left and right controls"

This is where we show next and previous slide links.

<a class="left carousel-control" href="#myCarousel" role="button"

data-slide="prev">

<< Previous

 …

The data-slide attribute accepts the keywords "prev" or "next", which alters the slide position relative to its current position.

Add Captions to Slides

Add within each to create a caption for each slide

The Carousel Plugin Classes

.carousel

Creates a carousel

.slide

Adds a CSS transition and animation effect when sliding from one item to the next. Remove this class if you do not want this effect

.carousel-indicators

Adds indicators for the carousel. These are the little dots at the bottom of each slide (which indicates how many slides there are in the carousel, and which slide the user are currently viewing)

.carousel-inner

Adds slides to the carousel

.item

Specifies the content of each slide

.left carousel-control

Adds a left button to the carousel, which allows the user to go back between the slides

.right carousel-control

Adds a right button to the carousel, which allows the user to go forward between the slides

.carousel-caption

Specifies a caption for the carousel

Carousel Methods

.carousel(options)

Activates the carousel with an option. See options above for valid values

.carousel("cycle")

Goes through the carousel items from left to right

.carousel("pause")

Stops the carousel from going through items

.carousel(number)

Goes to a specified item (zero-based: first item is 0, second item is 1, etc..)

Goes to the previous item

```
.carousel("next")
```

Goes to the next item

181- Responsive Design Testing

Responsive Testing

Testing responsive web designs is crucial as the user experience on mobile devices is different from desktops. It's not possible to test a design on all devices available in market.

Simplest way is to test responsiveness by resizing browser window. But this only caters for visual resizing.

You may also use several online testing services like:

- ResponsiveDesignChecker.com
- Mattkersley.com/responsive
- Ami.responsivedesign.i

s Mobile:

- Swipes
- Pinch to

Zoom Desktop:

- Hover
- Right Click

Another effective method for Responsive testing is to use Chrome DevTools.

Chrome DevTools -> **device mode** emulates mobile device experience.

Chrome Device Mode

Some of its main features are:

- Mobile device emulation
- Touch events emulation
- Media queries inspector
- Mobile network simulation

182- W3C Standards and Validations

W3C

The World Wide Web Consortium (W3C) is an international community that develops open standards to ensure the long-term growth of the World Wide Web.

W3C is led by Tim Berners Lee (Inventor of Web).

Core idea is to develop consensus on standards and keep improving them with community input.

W3C Validator

W3C offer an online tool to validate HTML, XHTML, SMIL, etc. documents against their standards and provide support in debugging to improve your web documents.

https://validator.w3.org/

183- Web Publishing Tools

Web Publishing

Web Publishing Tools are applications that are used to design and build websites, be as simple as a text editor or a feature rich web authoring package.

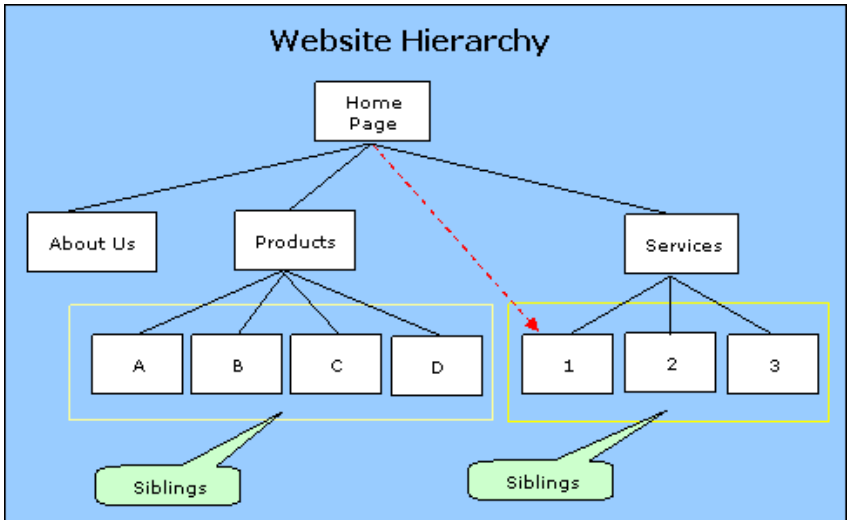
Also called a **Web Authoring Tools**

There are two types of Web Publishing Tools.

- Code Centric
- Page Centric or WYSIWYG (What You See Is What You Get)

There are some key points a good Web Publishing tool should have. Few of these key points are given below:

- It is important that each web page in the web adheres to a consistent design format.
- A good Web Publishing Application will provide functionality that supports and promotes website consistency. Provide ease for creation and use of common components.
- Like common headers, footers, graphics, font sets and backgrounds.
- Provide functionality to layout the overall structure of a website using a graphical user interface.



- Simple Web Publishing Tools provide the ability to generate web pages on a workstation but not the ability to automatically publish those pages to the Internet. An FTP application is require to copy web pages to online host.
- But a good Web Publishing Tool, also integrate FTP feature and directly publish pages on your host and keep track of pages updated, to update only those pages to the online server.
- Support Data Driven Website
- Support XML integration

Some Popular Tools

Let's review some popular Web Publishing Tools.

➢ Adobe PageMill

Adobe PageMill is a WYSIWYG Web Publishing Tool that allows you to view your HTML code as you construct your web page.

➢ NetObjects Fusion

NetObjects Fusion allows users to visually map out the website structure. It allows global changes to be made, automatically updates links, and constructs and organizes individual pages without using HTML or Dynamic HTML coding.

Fusion also allows you to publish your site easily to the web.

➢ Macromedia Dreamweaver

Macromedia Dreamweaver is both a Web Publishing Tool and HTML editor with WYSWYG support.

Provides drag-and-drop capabilities and supports Cascading Style Sheet (CSS) standards, Netscape Layers, and JavaScript.

The Dreamweaver 8 Web Publishing Application supports XML web services. Connections to data sources can be achieved by pointing a web page to an XML file or to the URL of an XML feed and then dragging and dropping the appropriate fields onto the page. It provides a platform and technology-independent development environment that supports PHP, J2EE, and Microsoft .NET.

184- Web Page Quality Issues

Web Quality

Let's discuss some points we should consider to keep our website implementation up to Quality.

Fonts

Use appropriate font style, weight and family.

- Do not overuse Bold text.
- Prefer Sans-serif fonts over Serif fonts.
- Prefer using web safe fonts over custom fonts.
- Use font style based on your website niche.

- Handle fonts and its size with CSS.

Colors and Backgrounds

Colors plays vital role in quality of your site.

- Only use dark colors, if you audience is of young age group.
- White background is preferred for corporate websites.
- Dark color text on white is more readable and also looks better.

Browser Compatibility

It is the biggest issue you must be aware of and handle.

- There are many versions of browser.
- Especially always remember to review your site on IE, as mostly IE present the page differently than other standard browsers.
- Prefer using HTML5
- Consider Using Frameworks like Bootstrap.

Speed

The best time of opening a website is 1 sec but it's ok to take up to 5 sec.

- You cannot control net speed but you can control the total size of your web page.
- Use CDN (Content Delivery Networks) where possible.
- Optimize Images.
- Avoid using custom fonts
- Test your site Speed using online tools.

SEO Friendly

Search Engine Optimization, in short SEO, plays vital role in online success of your website.

- Keep content easily accessible to users and robots.
- Use Sitemap to define your site structure
- Use Robot.txt file
- Interlink pages
- Define alt and roles with elements, where ever possible.

W3C Validation

Always validate your website with https://validator.w3.org/ and remove all reported issues.

185- Web Usability Issues

What is Web Usability?

Web Usability is the ease of using a website. It includes presentation of information on website, and making sure it's easily accessible to both desktop and mobile users.

Availability and Accessibility

Server Uptime: Make sure your site is always accessible over web, when a user tries to access it. Use Localize Server based on your target market.

Avoid Broken Links

Avoid Broken Links on your website. All links on your website should be

working. Tools you may use:

- https://validator.w3.org/checklink
- http://www.deadlinkchecker.com
- http://www.brokenlinkcheck.com

Responsive Web page

Make sure your website is tested for different sizes along with mobile interactions. You can Use Chrome -> Dev Tools -> device mode for testing Responsiveness of your web page.

Clarity of Content

Keep it simple, focus on what is important. Do not expect user to find things. Use proper headings and implement using proper header and paragraph tags. Present key content prominently and guide users as require to follow and explore your website.

Use Standards and Norms

Use popular standard and norms, like:

- Top menus.
- Standard footer links.
- Implement keys on forms, especially tabs.

Consistency

Keep it consistent. Do not keep on changing style or placement of components on different pages of your site, keep them consistent. It should be similar with style and color scheme.

User Interactions

Always handle user interactions with care. Indicate success or failure of user interaction with your site clearly.

Understand your audience

Understand your target audience and what they already know and why they are visiting your website, then base your website structure on that.

Good usability is not attained overnight. It requires thorough user research and an iterative approach of constant testing and refining.

186- HCI Considerations

HCI

Human Computer Interaction is the study, planning, and design of what happens when you and a computer work together.

As its name implies, HCI consists of three parts:

- The user
- The computer (or webpage)
- The ways they work together.

HCI Considerations for Frontend Developer

Building a webpage has no meaning if users are not going to use it. Real people will only use your website if it's easy to understand and useable.

Make sure to define all user interaction clearly.

Prominent headings for content on each page; use h1, h2 … tags. Always show clear respond (success or failure) to all user interactions in your page.

Use norms and standards. Mention Alt and Role attributes where applicable. Always test your site for Responsive behavior.

187- Search Engine Optimization (SEO)

What is Search engine optimization?

Search engine optimization (SEO) is the process of improving the ranking / visibility of a website in search engines. SEO is all about two things:

- What is on your site (On Site Optimization)
- How your site is linked / referred from other sites (Offsite Optimization)

Indexing in Search Engines

Web search engines (like Google) automatically add new web sites to their search index as they crawl the web. A new website needs to wait to get indexed.

Most search engines also invite you to submit your site.

- Google: http://www.google.com/addurl.html
- Yahoo: http://search.yahoo.com/info/submit.html
- Bing: http://www.bing.com/webmaster/SubmitSitePage.aspx

Note: Submitting your site does not guarantee indexing in most cases, but can speed up indexing process.

Black-hat and White-hat SEO

In simple words anything that is un-natural for your site promotion is considered as Black-hat SEO technique and vice versa.

SEO Key

Be natural and make sure your site is understandable by users and for robots too.

The content on your site and the HTML of your web pages are the most accessible and controllable search engine optimization (SEO) elements. You can win 75% of the SEO war if you can align your onsite optimization with your target.

On-Page Optimization Checklist

Lets discuss major on-page factors you can control to optimize your website.

- Content is high-quality, relevant, fresh and at least 500 words in length. Content should be accessible to bots too. Avoid duplicate content.
- Target search phrase is included in page headline and at least one sub-headline [use proper header h tag]. And repeated three to 10 times within body copy, don't over-do it. Keep reader in mind; make sure it is reader-friendly. Page includes relevant images and/or graphics that help illustrate the target search phrase. Captions for images and/or graphics include the target search phrase.
- Use the target phrase in image alt text (embedded in the image tag as alt="your keyword")
- Content and resources on the page are so good that visitors will want to share your link with others and post your link elsewhere.
- No misspellings or poor grammar.
- Inclusion of social media links and / or user discussion or reviews is a plus point. Pages with active visitor interaction are scored higher than static pages.
- If you are optimizing for specific country, state, city or regional names, be sure they are in your copy and perhaps in page footer.
- Include the target keyword/search phrase in the page URL if possible.
- **Title tags**: Every page should have an HTML title (enclosed in <title> </title> tags).
- The title copy should be unique, include the target keyword or target phrase, and must be 70 characters or less (any longer, and Google will truncate it anyway).
- **Meta description tag:** Search engines don't weigh Meta description in ranking a page, but it is still very important, it is the marketing copy for your page.
- The Meta description is enclosed in <meta name = "description" content=""> in head tag.
- The Meta description tag should be no longer than 160 characters, and should include your target phrase.
- Links from content: Use links to other pages on your site, or other resources. Links should be relevant to your page topic.
- Avoid using too many links. But having few internal links (to other pages on your site) is positive.
- Internal links also help search engine bots navigate and categorize your site.
- Setup Google Analytics.
- Setup your site on Google Webmaster Tools and Bing Webmaster Tools.
- Create sitemap.xml file
- Create Robots.txt file
- Keep JavaScript and CSS separate.
- Improve site speed.
- Validate your website using W3C Validator. url: https://validator.w3.org

189- Offsite SEO

Important part of Search Engine Optimization, Offsite SEO is about building credibility of your

Offsite Checklist

- Measure your inbound links and their value, either using Google Webmaster Tools or using different tools available. Scan your highest-authority inbound links for opportunity to get other similar links.
- Scan your competitors' highest-authority inbound links.
- Question yourself: Can you get those links too, or do they provide ideas for getting similar links?
- Inbound links from non-profit (.org) and education (.edu) sites are especially powerful.
- Build your inbound links gradually in natural order. Google algorithms will notice, and may penalize a quick accumulation of links.

Where to get inbound links?

- Business partners, clients, vendor's websites.
- Online press releases through paid press release services.
- Legitimate directories and local/regional web listings (yellow pages).
- Your own site blog and other relevant blogs.
- Thought leadership articles and whitepapers.
- Participating in relevant forums discussion and dropping your link.
- Social Bookmarking.
- Social Media popularity.
- Get reviews from other sites.
- Photo sharing.
- Video promotions.
- Participate in Answering sites.
- Document sharing.
- Infographic sharing.
- Mention your website link in email signature