WE HELP YOU SHAPE YOUR FUTURE.

1.1 Overview & Basic Tags

HTML5 is the newest version of Hyper Text Markup Language. The first web browser was introduced in 1993 and the name was **MOSAIC**. The development of **MOSAIC** was at the **NCSA** (National Center for Supercomputing Applications). Later it was discontinued to development on 7th of January 1997. Still the people were using the nonstandard version of **HTML**.

The First Draft of **HTML5** Was announced in January 2008. And amazingly **HTML5** has a broad browser support. Though the **HTML5** is still under developing phase. And a lot of organizations are working and planning for the development of **HTML5**.

We can't expect the HTML5 may be the future of Web Designing, but we can say that this is the present of Web designing. Before the development of **HTML5**, we were in compulsion to work in Photoshop and Flash application, but with the development of **HTML5**, these affords has been reduced. Many more long script codes can be done with a simple tagging. As we can use **<details>** and **<summary>** tag for show and hide function of Java Script. We need not to put a long affords to code this thing. Apart from this feature we can use the 3D image with **<canvas>**, the special designed paragraph with **<article>** and many more.

Application of HTML

As mentioned before, HTML is one of the most widely used language over the web. I'm going to list few of them here:

- **Web pages development** - HTML is used to create pages which are rendered over the web. Almost every page of web is having html tags in it to render its details in browser.
- **Internet Navigation** - HTML provides tags which are used to navigate from one page to another and is heavily used in internet navigation.
- **Responsive UI** - HTML pages now-a-days works well on all platform, mobile, tabs, desktop or laptops owing to responsive design strategy.

- **Offline support** HTML pages once loaded can be made available offline on the machine without any need of internet.
- **Game development**- HTML5 has native support for rich experience and is now useful in gaming development arena as well.

What You Can do With Html

There are lot more things you can do with HTML.

- You can publish documents online with text, images, lists, tables, etc.
- You can access web resources such as images, videos or other HTMLdocument via hyperlinks.
- You can create forms to collect user inputs like name, e-mail address, comments, etc.
- You can include images, videos, sound clips, flash movies, applications and other HTML documents directly inside an HTML document.
- You can create offline version of your website that work without internet.
- You can find the current location of your website's visitor.

Html Documents

Creating Your First HTML Webpage

These tags should be placed underneath each other at the top of every HTML page that you create.

<!DOCTYPE html> — This tag specifies the language you will write on the page. In this case, the language is HTML 5.

<html> — This tag signals that from here on we are going to write in HTML code.

<head> — This is where all the metadata for the page goes — stuff mostly meant for search engines and other computer programs.

<body> — This is where the content of the page goes.

Inside the <head> tag, there is one tag that is always included: <title>, but there are others that are just as important:

<title> This is where we insert the page name as it will appear at the top of the browser window or tab.

<meta> This is where information about the document is stored: character encoding, name (page context), description. Let's try out a basic <head> section:

<head>

<title>My First Webpage</title>

<meta charset="UTF-8">

<meta name="description" content="This field contains information about your page. It is usually around two sentences long.">.

<meta name="author" content="Conor Sheils">

</head>

An **HTML element** is defined by a starting tag. If the element contains other content, it ends with a closing tag, where the element name is preceded by a forward slash as shown below with few tags −

Start Tag	Content	End Tag
	This is paragraph content.	
<h1>	This is heading content.	</h1>
	This is division content.	

So here .... **is an HTML element,** <h1>...</h1> **is another HTML element. There are some HTML elements which don't need to be closed, such as** <img.../>**,** <hr /> **and**
 elements. These are known as void elements **or Empty HTML Elements.**

An HTML comment begins with <!--, and ends with -->, as shown in the example below: You can also comment out part of your HTML code for debugging purpose, as shown here:

Example

<!-- Hiding this image for testing

<img src="images/smiley.png" alt="Smiley">

-->

Images

Images in HTML are inline elements that can be placed within a paragraph. To add an image, use the <img> tag along with the src attribute to specify the location of the image.

Use a <figure> element to mark up a photo in a document.

The <figure> element can also contain a <figcaption>:

```
<figure>
   <img src="/media/examples/elephant-660-480.jpg"
        alt="Elephant at sunset">
   <figcaption>An elephant at sunset</figcaption>
</figure>
```

Hyperlinks

A link ("anchor") is a small span of text that will direct you to a different section in the page, or to a different page. To create a link, you will need to specify where you would like the user to be directed to when the link is clicked by specifying the href attribute.

Setting the Targets for Links

The target attribute tells the browser where to open the linked document. There are four defined targets, and each target name starts with an underscore(_) character:

- _blank — Opens the linked document in a new window or tab.
- _parent — Opens the linked document in the parent window.

- _self — Opens the linked document in the same window or tab as the source document. This is the default, hence it is not necessary to explicitly specify this value.
- _top — Opens the linked document in the full browser window.

For example:

```
<a href="https://www.google.com">A link to Google</a>
```

To create a link to a different section in the same page, you will need to use a hash sign along with the element ID to where you would like the browser to jump to. For example:

<a href="#faq">Click here to read the Frequently Asked Questions

<h3 id="faq">Frequently asked questions</h3>

The first rule about fight club is that you do not talk about fight club.

<h3 id="faq">Frequently asked questions</h3>

The first rule about fight club is that you do not talk about fight club.

Creating Horizontal Rules

You can use the <hr> tag to create horizontal rules or lines to visually separate content sections on a web page. Like
, the <hr> tag is also an empty element.

<hr size=”3” color=”#cfecc” />

1.2 Lists

HTML provides a way to create both an ordered list (with elements counting up, 1, 2, 3...) and an unordered list with bullets instead of numbers. Lists are a good way to formalize a list of items and let the HTML styling do the work for you.

Ordered Lists

Here is an example of how to create an ordered list:

 Here is a list of ordered items:

```
<ol>
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ol>
```

Ordered lists have a “type” attribute which defines the numbering convention to use.

To count using numbers, use type="1":

```
<p>Here is a list of ordered items:</p>
<ol type="1">
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ol>
```

To count using uppercase letters, use type="A":

```
<p>Here is a list of ordered items:</p>
<ol type="A">
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ol>
```

To count using lowercase letters, use type="a":

```
<p>Here is a list of ordered items:</p>
<ol type="a">
  <li>First item</li>
```

```
    <li>Second item</li>
    <li>Third item</li>
  </ol>
```

To count using uppercase roman numerals, use type="I":

```
  <p>Here is a list of ordered items:</p>
  <ol type="I">
    <li>First item</li>
    <li>Second item</li>
    <li>Third item</li>
  </ol>
```

To count using lowercase roman numerals, use type="i":

```
  <p>Here is a list of ordered items:</p>
  <ol type="i">
    <li>First item</li>
    <li>Second item</li>
    <li>Third item</li>
  </ol>
```

Unordered lists

Here is an example of how to create an unordered list:

```
  <p>Here is a list of unordered items:</p>
  <ul>
    <li>First item</li>
    <li>Second item</li>
```

```
    <li>Third item</li>
  </ul>
```

To change the list style attributes, we can use the CSS attribute called list-style-type. The available types are:

- disc
- circle
- square
- none

Here is an example of the disc list style type:

```
<p>Here is a list of unordered items:</p>
<ul style="list-style-type: disc">
<li>First item</li>
<li>Second item</li>
<li>Third item</li>
</ul>
```

Here is an example of the circle list style type:

```
<p>Here is a list of unordered items:</p>
<ul style="list-style-type: circle">
<li>First item</li>
<li>Second item</li>
<li>Third item</li>
</ul>
```

Here is an example of the square list style type:

```
<p>Here is a list of unordered items:</p>
```

```
<ul style="list-style-type: square">
<li>First item</li>
<li>Second item</li>
<li>Third item</li>
</ul>
```

Here is an example of the none list style type:

```
<p>Here is a list of unordered items:</p>
<ul style="list-style-type: none">
<li>First item</li>
<li>Second item</li>
<li>Third item</li>
</ul>
```

1.3 Table

- Tables can be used to compare two or more items in tabular form layout.
- Tables are used to create databases.

An HTML table is defined with the “table” tag. Each table row is defined with the “tr” tag. A table header is defined with the “th” tag. By default, table headings are bold and centered. A table data/cell is defined with the “td” tag.

- In HTML Table will be created by using <table>table data goes here..</table> tag.
- We know that table contains Rows and Columns, those are defined with tr and td.
- <tr> stands for Table Row which is used to make a Row.
- <td> stands for Table Data that is used to make a Column.
- Table heading can be defined by using <th>Name</th>
- Cellpadding and cell spacing is used to adjust the white space in table cell.

- Cell spacing defines the width of the border. cell spacing="0" cellpadding="15"
- Cellpadding represents the distance between cell borders and the content within.
- <caption> Books Information</caption> tag will serve as a title and show at the top of the table.

The <thead> element structures the headings in your table and this tells browsers what e.g. each column contains. The <tbody> element structures all of the content, so that the browser knows what the actual content of the table is. Using the same example as before, the <thead> and the <tbody> elements are to be used like this:

```
<table border="1" width="100%">
<caption> Table Example</caption>
<thead>
<tr>
<td>Row 1, cell 1</td>
<td>Row 1, cell 2</td>
<td>Row 2, cell 3</td>
</tr>
</thead>
<tbody>
<tr>
<td>Row 2, cell 1</td>
<td>Row 2, cell 2</td>
<td>Row 2, cell 3</td>
</tr>
<tr>
<td>Row 3, cell 1</td>
<td>Row 3, cell 2</td>
<td>Row 3, cell 3</td>
</tr>
```

</tbody>

</table>

1.4 Forms

An HTML form is a section of a document containing normal content plus some controls

Checkboxes, radio buttons, menus, text fields etc.

- Every form in a document is contained in a FORM tag –
- The FORM tag specifies the action that takes place when the form is submitted

Difference between GET and POST

GET

1. Show the user data in Url
2. The length of a URL is limited (about 3000 characters)
3. Never use GET to send sensitive data!

POST

1. Never Show the user data in Url
2. The length of a URL is unlimited
3. Always use POST to send sensitive data!

Following common attributes: -

autocomplete - string to indicate which autocomplete functionality to apply to the input, commonly set to on to allow for autocompletion

autofocus - boolean to indicate if the input should be focused automatically (on page load)

disabled - boolean to indicate if the input should be disabled **form -** the ID of the <form> the input is a member, defaults to the nearest form containing the input

list - the ID of a <datalist> element that provides a list of suggested values, not widely supported at the moment

name - the name of the input, used as an indicator on submitted data **required** - boolean

to indicate if the value is required before the <form> can be submitted **tabindex** - number

to indicate which order the inputs should receive focus when the user

hits TAB

type - string to indicate which type of input the element represents.

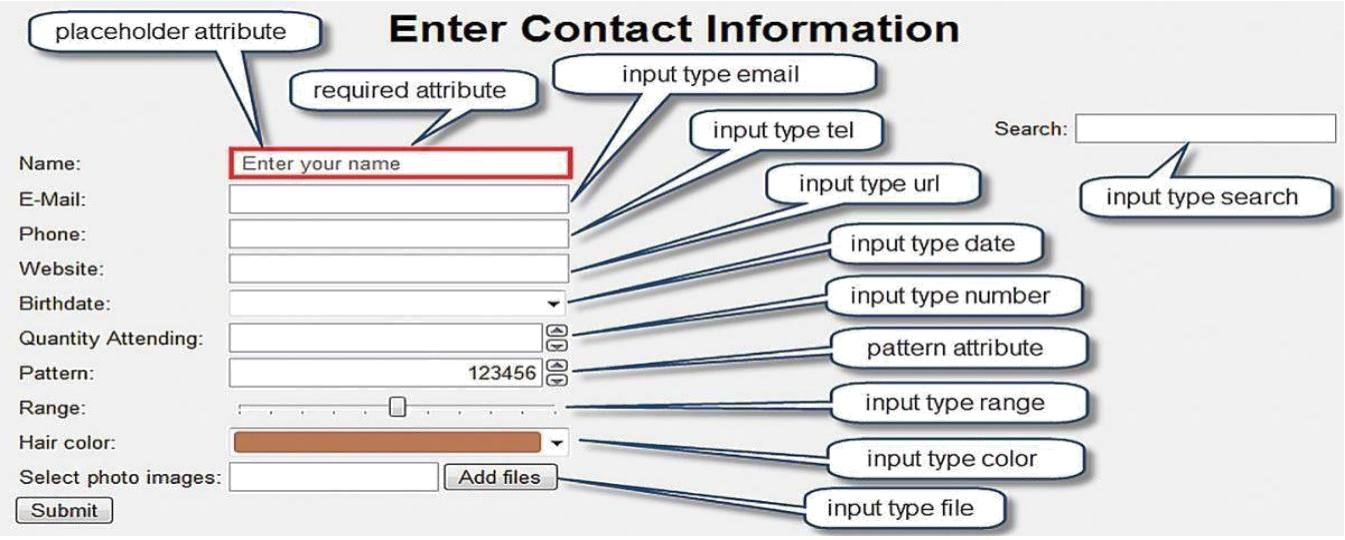
Form element's attributes:

Attribute	Description
name	Specifies the name of the form.
action	Specifies the URL of the program or script on the web server that will be used for processing the information submitted via form.
method	Specifies the HTTP method used for sending the data to the web server by the browser. The value can be either get (the default) and post.
target	Specifies where to display the response that is received after submitting the form. Possible values are _blank and .
enctype	Specifies how the form data should be encoded when submitting the form to the server. Applicable only when the value of the method attribute is .

<form>	It defines an HTML form to enter inputs by the used side.
<input>	It defines an input control.
<textarea>	It defines a multi-line input control.
<label>	It defines a label for an input element.
<fieldset>	It groups the related element in a form.
<legend>	It defines a caption for a <fieldset> element.
<select>	It defines a drop-down list.
<optgroup>	It defines a group of related options in a drop-down list.
<option>	It defines an option in a drop-down list.
<button>	It defines a clickable button.

Latest Input Types in HTML5

HTML5 introduces several new <**input**> **types** like email, date, time, color, range, and so on. To improve the user experience and to make the forms more interactive. However, if a browser failed to recognize these new **input types**, it will treat them like a normal text box.



1.5 Google Map, Video, Audio & iframe

Embedding Video

<video src = "foo.mp4" width = "300" height = "200" controls> Your browser does not support the <video> element.

</video>

Video Attribute Specification

The HTML5 video tag can have a number of attributes to control the look and feel and various functionalities of

the control –

Sr.No.	Attribute & Description
1	**autoplay** This Boolean attribute if specified, the video will automatically begin to play back as soon as it can do so without stopping to finish loading the data.

2	**autobuffer** This Boolean attribute if specified, the video will automatically begin buffering even if it's not set to automatically play.
3	**controls** If this attribute is present, it will allow the user to control video playback, including volume, seeking, and pause/resume playback.
4	**height** This attribute specifies the height of the video's display area, in CSS pixels.
5	**Loop** This Boolean attribute if specified, will allow video automatically seek back to the start after reaching at the end.
6	**preload** This attribute specifies that the video will be loaded at page load, and ready to run. Ignored if autoplay is present.
7	**poster**
	This is a URL of an image to show until the user plays or seeks. **Src**
8	The URL of the video to embed. This is optional; you may instead use the <source> element within the video block to specify the video to embed.

Embed Audio

HTML5 supports <audio> tag which is used to embed sound content in an HTML or XHTML document as follows.

```
<audio src = "foo.wav" controls autoplay>
  Your browser does not support the <audio> element.
</audio>
```

The current HTML5 draft specification does not specify which audio formats browsers should

support in the audio tag. But most commonly used audio formats are **ogg, mp3** and **wav**.

You can use <source> tag to specify media along with media type and many other attributes. An audio element allows multiple source elements and browser will use the first recognized format.

Iframe

The iframe in HTML stands for Inline Frame. The" iframe" tag defines a rectangular region within the document in which the browser can display a separate document, including scrollbars and borders. An inline frame is used to embed another document within the current HTML document.

```
 <iframe width="560" height="315" src="https://www.youtube.com/embed
/owsfdh4gxyc" frameborder="0" allowfullscreen></iframe>
```

Sr.No	Attribute & Description
1	**src** This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL. For example, src = "/html/top_frame.htm" will load an HTML file available in html directory.
2	**name** This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.
3	**frameborder** This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).
4	**marginwidth** This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth = "10".
5	**marginheight** This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example marginheight = "10".
6	**height** This attribute specifies the height of <iframe>.
7	**scrolling** This attribute controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example scrolling = "no" means it should not have scroll bars.

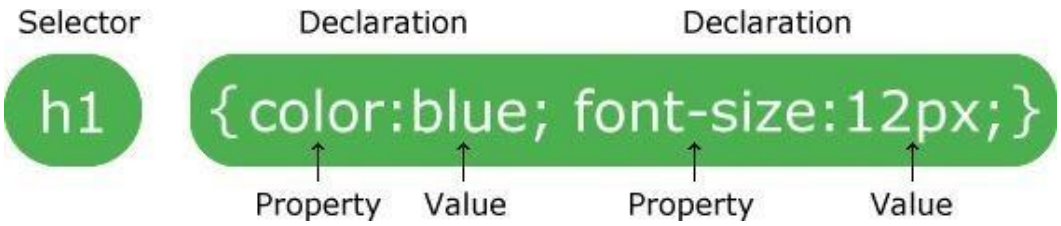
2.1 Overview

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

CSS Syntax

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



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

The declaration block contains one or more declarations separated by semicolons. Each declaration includes a CSS property name and a value, separated by a colon.

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

2.3 Types of CSS3

Three Ways to Insert CSS

- External CSS
- Internal CSS
- Inline CSS

- **Below code is an example of 'inline CSS', where the styles are defined inside the individual tags. <!-- css.html -->**

```
<!DOCTYPE html>

<html>

<head>

<title>CSS Tutorial</title>

</head>

<body>

<h3 style="color:blue"> Heading 1 </h3>

<h3 style="color:blue"> Heading 3 </h3>

<h3 style="color:blue"> Heading 3 </h3>

</body>

</html>
```

In the above code, we have three 'headings' with font-color as 'blue'. Suppose, we want to change the color to red, then we must go to to individual 'h3' tag and then change the color. This is easy in this case, but if we have 100 headings in 5 different 'html' files, then this process is not very handy.

Embedded CSS

In the below code, the style is embedded inside the 'style' tag as shown in Lines 8-17. Here, we have defined two classes i.e. 'h3_blue (Lines 21-23)' and 'h3_red (Lines 26-28)'. Then, the selectors at Lines 9 and 13 targets the class 'h3_blue' & 'h3_red', and change the color to blue and red respectively. In this chapter, we will discuss the selectors (e.g. h3.h3_blue) in more details.

Note:

- **In CSS, the comments are written between /* and */.**

CSS has three parts,

– Selectors e.g. p, h3.h3_blue

– Properties e.g. color

– Values of properties e.g. red

```
<!-- css.html --> <!DOCTYPE
html>
 <html>
 <head>
 <title>CSS Tutorial</title>
 <style type="text/css">
 h3.h3_blue{ /*change color to
 blue*/ color: blue;
}
 h3.h3_red{ /*change color to
 red*/ color:red;
 }
</style>
 </head>
 <body>
 <h3 class='h3_blue'>  Heading  1 </h3>
<h3  class='h3_blue'>  Heading  3 </h3>
<h3  class='h3_blue'>  Heading  3 </h3>
<h3 class='h3_red'> Heading 1 </h3>
 <h3 class='h3_red'> Heading 3 </h3>
 <h3 class='h3_red'> Heading 3 </h3>
 </body>
```

```
</html>
```

External CSS

We can write the ‘CSS’ code in different file and then import the file into ‘html’ document as shown in this section.

In this way, we can manage the files easily.

- **The ‘CSS’ code is saved in the file ‘my_css.css’ which is saved inside the folder ‘asset/css’.**

```
/* asset/css/my_css.css */
h3.h3_blue{ color: blue;
}
h3.h3_red{ color:red;
}
```

```
<!DOCTYPE html>
<head>
<title>CSS Tutorial</title>
<link rel="stylesheet" type="text/css" href="asset/css/my_css.css">
</head>
<body>
<h3 class='h3_blue'> Heading 1 </h3>
<h3 class='h3_blue'> Heading 3 </h3>
<h3 class='h3_blue'> Heading 3 </h3>
<h3 class='h3_red'> Heading 1 </h3>
<h3 class='h3_red'> Heading 3 </h3>
<h3 class='h3_red'> Heading 3 </h3>
</body>
```

```
</html>
```

Internal CSS

The internal style is defined inside the <style> element, inside the head section.

```
<html>

<head>

<style> body {

background-color: linen;

}

h1 { color: maroon; margin-left: 40px; }

</style>

</head>

<body>

<h1>This is a heading</h1>

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

</body>

</html>
```

Basic CSS Selectors

There are three types of selectors in CSS,

- **Element :** can be selected using it's name e.g. 'p', 'div' and 'h1' etc.
- **Class :** can be selected using '.className' operator e.g. '.h3_blue'.
- **ID :** can be selected using '#idName' e.g. '#my_para'.

```
<!-- css.html -->
```

```
<!DOCTYPE html>
<html>
<head>
<title>CSS Selectors</title>
<link rel="stylesheet" type="text/css" href="asset/css/my_css.css">
</head>
<body>
<h3>CSS Selectors</h3>
<p class='c_head'> Paragraph with class 'c_head' </p>
<p id='i_head'> Paragraph with id 'i_head' </p>
</body>
</html>
```

- **Below code shows the example of different selectors, and the output is shown in**

```
/* asset/css/my_css.css */
/*element selection*/ h3 {
color: blue;
}
/*class selection*/
.c_head{
font-family: cursive; color: orange;
}
/*id selection*/ #i_head{
font-variant: small-caps; color: red
```

```
}
```

Output:

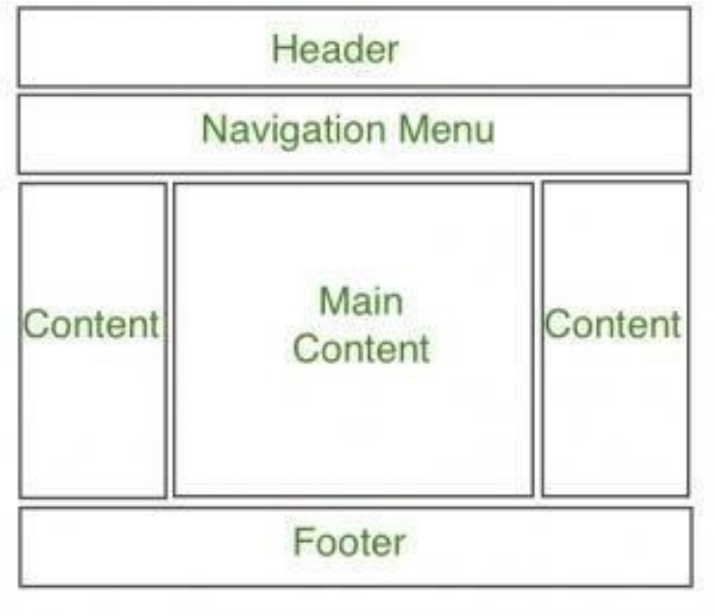
2.3 CSS Properties

Property	Syntax	Description/possible values
size	20%	size = 20%
	20px	20 pixel
	2em	2*font-size
	2mm, 2cm, 2in	2 mm, cm and inch
color	names	e.g. red, blue, green
	hex code (#RRGGBB or #RGB)	#FFF000 or #FFF
	rgb(num, num, num)	rgb(0, 0, 255) or rgb(20%, 10%, 70%)
link	a:link	a:link {color: red}
	a:hover	
	a:visited	
	a:active	
Font	font-family	serif, cursive
	font-style	normal, italic, oblique
	font-variant	normal, small-caps
	font-weight	normal, bold, bolder, lighter, 100-900
	font-size	10px, small, medium, large etc.
Text	color	red, #FFF
	letter-spacing	10px
	word-spacing	10 px
	text-align	right, left, center
	text-decoration	underline, overline, line-through, none
	text-transform	capitalize, uppercase, lowercase, none
	white-space	pre, normal, nowrap
	text-shadow	text-shadow:5px 5px 8px red;
Image	border	'1px', or '1px solid blue'
	height, width	100px, 20%
Border	border-style	solid, dashed, dotted, double, none etc.
	border-top-style	
	border-bottom-style	
	border-left-style	
	border-right-style	
	border-width	4px, 4pt
	border-bottom-width	similarly use 'top', 'left', 'right'
	border (shortcut)	1px solid blue'
Margin	margin, margin-left etc.	
Padding	padding (top, bottom, left, right)	'10px 10px 2px 2px' or '10px 2px'
	padding-right, padding-left etc.	

2.4 Layout in CSS3

A website can be divided into various sections comprising of header, menus, content and footer based on which there are many different layout designs available for developer. Different layouts can be created by using div tag and use CSS property to style it.

The most common structure of website layout is given below:



Header section contains a website logo, a search bar and profile of user. The navigation menu contains link to various categories of articles available and content section is divided into 3 parts(columns) with left and right sidebar containing links to other articles and advertisements whereas the main content section is the one containing this article, then at the bottom there is a footer section which contains address, links, contacts etc.

3.1 Introduction to Bootstrap

First developed by Twitter, Bootstrap is by now used for anything from developing web applications to WordPress themes. Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web. Bootstrap includes a responsive, mobile first fluid grid system that appropriately Bootstrap is a front-end framework that helps you build mobile responsive websites more quickly and easily. scales up to 12 columns as the device or viewport size increases. It includes predefined classes for easy layout options, as well as powerful mixins for generating more semantic layouts.

To load Bootstrap, you would need to load it from somewhere. We can load the Bootstrap library from a CDN by running the following <link> tag:

Where to Get Bootstrap?

There are two ways to start using Bootstrap on your own web site.

You can:

- Download Bootstrap from getbootstrap.com
- Include Bootstrap from a CDN

Bootstrap CDN

If you don't want to download and host Bootstrap yourself, you can include it from CDN (Content Delivery Network).

MaxCDN provides CDN support for Bootstrap's CSS and JavaScript. You must also include jQuery:

Max CDN

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap. min.css">

<!-- jQuery library -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
```

```
<!-- Latest compiled JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
```

3.2 Containers

Bootstrap also requires a containing element to wrap site contents.

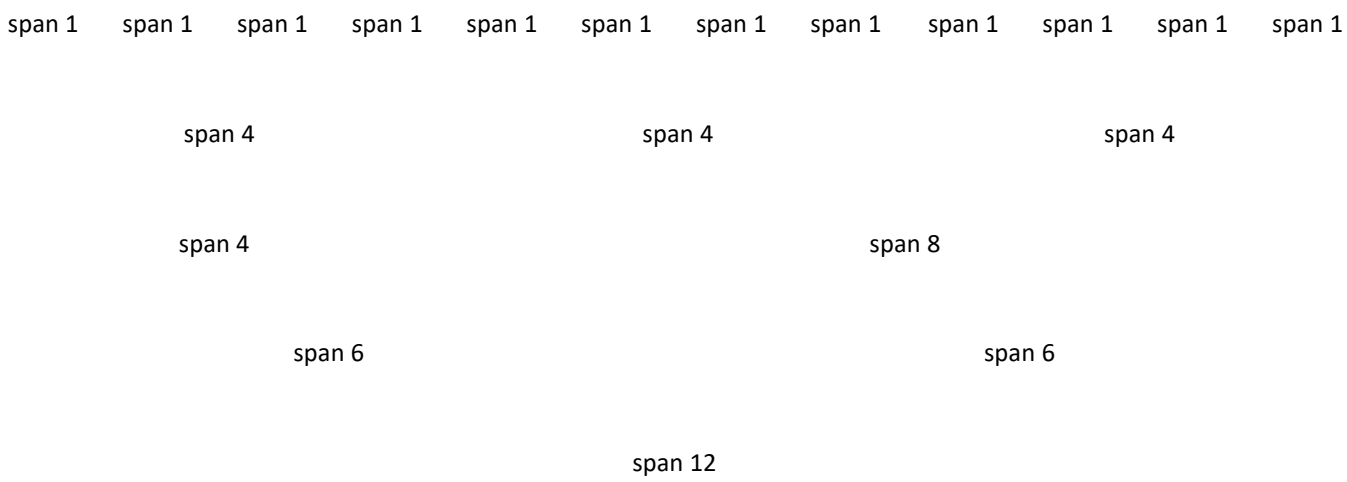
There are two container classes to choose from:

1. The .container class provides a responsive **fixed width container**
2. The .container-fluid class provides a **full width container**, spanning the entire width of the viewport

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

The Bootstrap grid system has four classes:

- xs (for phones - screens less than 768px wide)

- sm (for tablets - screens equal to or greater than 768px wide)
- md (for small laptops - screens equal to or greater than 992px wide)
- lg (for laptops and desktops - screens equal to or greater than 1200px wide)

3.3 Bootstrap Exercises with Classes

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container">
  <h2>Alerts</h2>
  <div class="alert alert-success">
    <strong>Success!</strong> This alert box could indicate a successful or positive action.
  </div>
  <div class="alert alert-info">
    <strong>Info!</strong> This alert box could indicate a neutral informative change or action.
  </div>
  <div class="alert alert-warning">
    <strong>Warning!</strong> This alert box could indicate a warning that might need attention.
  </div>
```

```
  <div class="alert alert-danger">
    <strong>Danger!</strong> This alert box could indicate a dangerous or potentially negative action.
  </div>
</div>
</body>
</html>
```

Success! This alert box indicates a successful or positive action.

Info! This alert box indicates a neutral informative change or action.

Warning! This alert box indicates a warning that might need attention.

Danger! This alert box indicates a dangerous or potentially negative action.

Glyphicons

Bootstrap provides 260 glyphicons from the Glyphicons Halflings set.

Glyphicons can be used in text, buttons, toolbars, navigation, forms, etc.

Here are some examples of glyphicons:

Envelope glyphicon:

Print glyphicon:

Search glyphicon:

Download glyphicon:

Button Styles

Bootstrap provides different styles of buttons:

Basic Default Primary Success Info Warning Danger Link

To achieve the button styles above, Bootstrap has the following classes:

- .btn
- .btn-default
- .btn-primary
- .btn-success

- .btn-info
- .btn-warning
- .btn-danger
- .btn-link

3.4 JSON

JSON is a format for storing and transporting data. JSON is often used when data is sent from a server to a web page.

What is JSON?

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight data interchange format
- JSON is language independent
- JSON is "self-describing" and easy to understand

JSON Syntax Rules

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

JSON Arrays

JSON arrays are written inside square brackets. An array can contain

objects: "employees":[
 {"firstName":"John", "lastName":"Doe"},
 {"firstName":"Anna", "lastName":"Smith"},
 {"firstName":"Peter", "lastName":"Jones"}
]

Functions for Working with JSON

This section will look at two methods for stringifying and parsing JSON. Being able to convert JSON from

object to string and vice versa is useful for transferring and storing data.

The JSON.stringify() function converts an object to a JSON string.

`JSON.parse()` to convert the string into a JavaScript object.

‘

Chapter 4 JAVASCRIPT & jQuery

4.1 Introduction to JavaScript and Basic Syntax

What is JavaScript?

JavaScript, often abbreviated as JS, is a high-level, just-in-time compiled, object-oriented programming language that conforms to the ECMAScript specification. JavaScript has curly- bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. JavaScript is the Programming Language for the Web. JavaScript can update and change both HTML and CSS. JavaScript can calculate, manipulate and validate data.

Why is it called JavaScript?

When JavaScript was created, it initially had another name: “LiveScript”. But Java was very popular at that time, so it was decided that positioning a new language as a “younger brother” of Java would help. But as it evolved, JavaScript became a fully independent language with its own specification called ECMAScript and now it has no relation to Java at all.

What can JavaScript Do?

- JavaScript can dynamically modify an HTML page
- JavaScript can react to user input
- JavaScript can validate user input

- JavaScript can be used to create cookies (yum!)
- JavaScript is a full-featured programming language
- JavaScript user interaction does not require any communication with the server.

Application of JavaScript

- JavaScript is used to create interactive websites. It is mainly used for:
- Client-side validation and Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box)
- Displaying clocks etc.

Where to Put your Scripts

- Scripts can be placed in the HEAD or in the BODY

–In the HEAD, scripts are run before the page is displayed

–In the BODY, scripts are run as the page is displayed

- **In the HEAD is the right place to define functions and variables that are used by scripts within the BODY**

JavaScript can be inserted into documents by using the SCRIPT tag

```
<html>
<head>
<title>Hello World in JavaScript</title>
</head>
<body>
<script type="text/javascript"> document.write("Hello World!");
</script>
```

</body>

</html>

External Scripts

- Scripts can also be loaded from an external file
- This is useful if you have a complicated script or set of subroutines that are used in several different documents.

<script src="myscript.js"></script>

4.2 JavaScript Variables

- JavaScript has variables that you can declare with the optional var keyword
- Variables declared within a function are local to that function
- Variables declared outside of any function are global variables var myname ="Pat Morin";

JavaScript is dynamically typed, which means you do not have to declare the type of your variables in your code.

let num = 5;

let myString = "Hello"; var interestRate = 0.25;

After ES2015, we now have two new variable containers: let and const. Now we shall look at both of them one by one. The variable type **Let** shares lots of similarities with var but unlike var it has scope constraints. To know more about them visit let vs var. Let's make use of let variable:

// let variable

let x; // undefined let name = 'Mukul';

/ can also declare multiple vlaues let a=1,b=2,c=3;

/ assignment let a = 3;

a = 4; // works same as var.

Const is another variable type assigned to data whose value cannot and will not change throught the script.

// const variable const name = 'Mukul';

name = 'Mayank'; // will give Assignment to constant variable error.

Variable Scope in Javascript

Scope of a variable is the part of the program from where the variable may directly be accessible.
In JavaScript, there are two types of scopes:

1. **Global Scope** – Scope outside the outermost function attached to Window.
2. **Local Scope** – Inside the function being executed.

Let's look at the code below. We have a global variable defined in first line in global scope. Then we have a local variable defined inside the function fun().

let globalVar = "This is a global

variable"; function fun() {

let localVar = "This is a local variable";

console.log(globalVar); console.log(localVar);

}

fun();

When we execute the function fun(), the output shows that both global as well as local variables are accessible inside the function as we are able to console.log them. This shows that inside the function we have access to both global variables (declared outside the function) and local variables (declared inside the function).Let's move the console.log statements outside the function and put them just after calling the function.

let globalVar = "This is a global variable"; function

```
fun() {
  let localVar = "This is a local variable";
}

fun(); console.log(globalVar);
console.log(localVar);
```

Javascript Operators

JavaScript includes operators as in other languages. An operator performs some operation on single or multiple operands (data value) and produces a result. For example 1 + 2, where + sign is an operator and 1 is left operand and 2 is right operand. + operator adds two numeric values and produces a result which is 3 in this case.

JavaScript includes following categories of operators.

1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators
4. Assignment Operators
5. Conditional Operators

Arithmetic Operators

Arithmetic operators are used to perform mathematical operations between numeric operands.

Operator	Description
+	Adds two numeric operands.
-	Subtract right operand from left operand
*	Multiply two numeric operands.
/	Divide left operand by right operand.
%	Modulus operator. Returns remainder of two operands.

Ternary Operator

JavaScript includes special operator called ternary operator :? that assigns a value to a variable based

on some condition. This is like short form of if-else condition.tax:

<condition> ? <value1> : <value2>;

Ternary operator starts with conditional expression followed by ? operator. Second part (after ? and before : operator) will be executed if condition turns out to be true. If condition becomes false then third part (after :) will be executed.

Example: Ternary operator var a = 10, b = 5;
var c = a > b? a : b; // value of c would be 10 var d = a > b? b : a; // value of d would be 5

Comparison Operators

== — Equal to

= — Equal value and equal type

!= — Not equal

!== — Not equal value or not equal type

> — Greater than

< — Less than

>= — Greater than or equal to

<= — Less than or equal to

? — Ternary operator

Logical Operators

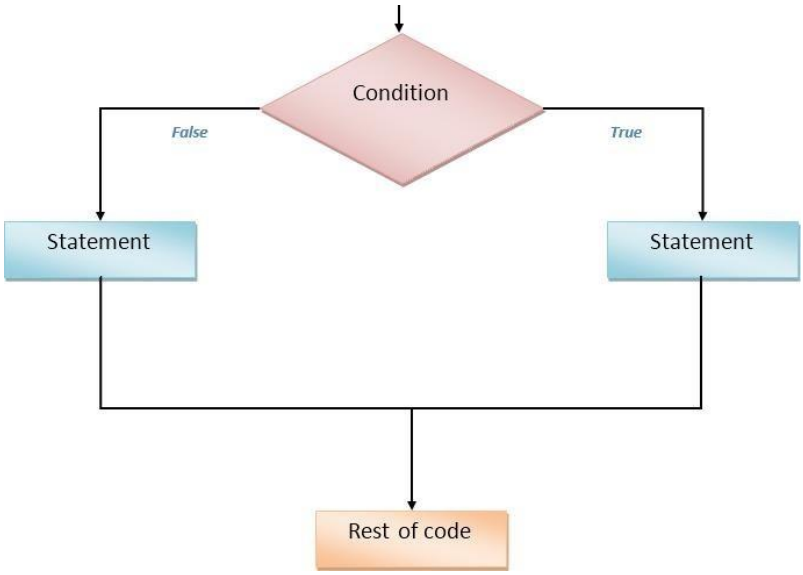
& — Logical and

|| — Logical or

! — Logical not

4.3 Conditional Statements

Conditional statements are used to decide the flow of execution based on different conditions. If a condition is true, you can perform one action and if the condition is false, you can perform another action.



Different Types of Conditional Statements

1. If statement
2. If…Else statement
3. If…Else If…Else statement

If statement

Syntax: if (condition){

lines of code to be executed if condition is true

}

<html>

```
<head>
  <title>IF Statments!!!</title>
  <script type="text/javascript">
    var age = prompt("Please enter your age");
    if(age>=18)
    document.write("You are an adult <br />");
    if(age<18)
    document.write("You are NOT an adult <br />");
  </script>
</head>
<body>
</body>
</html>
```

If...Else statement

Syntax:

```
if (condition){
lines of code to be executed if the condition is true
}else{
lines of code to be executed if the condition is false
}
```

```
<html>
<head>
```

```
    <title>If...Else Statments!!!</title>
    <script type="text/javascript">
      / Get the current hours var hours = new
      Date().getHours(); if(hours<12)
      document.write("GoodMorning!!!<br />");
      else

      document.write("Good Afternoon!!!<br />");
    </script>
  </head>
  <body>
  </body>
  </html>
```

If...Else If...Else

statement Syntax:

```
if (condition1){
lines of code to be executed if condition1 is true
}else if(condition2){
lines of code to be executed if condition2 is true
}

else{
```

lines of code to be executed if condition1 is false and condition2 is false

}

Switch

If you're going to be doing a large number of tests, it makes sense to use a switch statement instead of nested ifs. Switches in javascript are quite powerful, allowing evaluations on both the switch and the case.

```
var x=5; switch (x) {

case 1: alert('x is equal to 1!'; break; case 2:

alert('x is equal to 2!'; break; case 5: alert('x

is equal to 5!'; break; default: alert("x isn't

1, 2 or 5!");

}
```

Note that if you omit the break statement that ALL of the code to the end of the switch statement will be executed. So if x is actually equal to 5 and there is no break statement, an alert for "x is equal to 5" will appear as well as an alert for "x isn't 1,2, or 5!".

4.4 Loops

Loops are part of most programming languages. They allow you to execute blocks of code desired number of times with different values:

You have several parameters to create loops:

- **for** — The most common way to create a loop in JavaScript
- **while** — Sets up conditions under which a loop executes

- **do while** — Similar to the **while** loop but it executes at least once and performs a check at the end to see if the condition is met to execute again
- **break** —Used to stop and exit the cycle at certain conditions
- **continue** — Skip parts of the cycle if certain conditions are met

For Loop if you want to run the same code over and over again, each time with a different value.

Syntax:

```
for(initializer; condition; iteration)
{
  // Code to be executed
}
```

- Initializer: Initialize a counter variable to start with
- Condition: specify a condition that must evaluate to true for next iteration
- Iteration: increase or decrease counter

```
<script>
for (i=1; i<=5; i++) {
 document.write(i + "<br/>")
}
</script>
```

The **while** loop loops through a block of code as long as a specified condition is true.

Syntax

```
while (condition) {
  // code block to be executed
}
```

```
<script> var i = 1;
while(i <= 5) {
```

```
    document.write("<p>The number is " + i + "</p>");

    i++;

}
</script>
```

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.

Syntax

```
do {
  // code block to be executed
}
while (condition);
```

```
var i = 1; do {
   document.write("<p>The number is " + i + "</p>");
    i++;
} while(i <= 5);
</script>
```

4.5 Arrow Functions

JavaScript conforms to the ECMAScript (ES) standards. The ES6, or ECMAScript 2015 specifications, had introduced some of the revolutionary specifications for JavaScript, like Arrow Functions, Classes, Rest and Spread operators, Promises, let and const, etc.

ARROW FUNCTION SYNTAX

As we know, an ES5 function has the following syntax:

```
function square(a) {
```

return a * a;

}

IN ES6, WE CAN WRITE THE SAME FUNCTION WITH ONLY ONE LINE OF CODE:

let square = (a) => { return a * a; }

Furthermore, if the function body has only one statement that it returns, we can skip curly braces {} and the return statement:

let square = (a) => a * a

ALSO, IF THE FUNCTION TAKES ONLY ONE PARAMETER WE CAN EVEN SKIP THE BRACES () AROUND IT:

let square = a => a * a

ON THE OTHER HAND, IF THE FUNCTION DOES NOT TAKES ANY PARAMETERS WE MAY WRITE IT LIKE THIS:

let results = () => { /* ...some statements */ };

HERE IS AN EXAMPLE OF A SIMPLE ADDITION FUNCTION IN ES5

var sum = function (num1, num2) { return num1 + num2;

};

console.log(sum(7,12));

We can rewrite this function using fat arrows => in a couple of steps.

- 1. Delete the function keyword to the left of the function arguments
- 2. Insert a fat arrow to the right of the function arguments

YOUR FIRST ARROW FUNCTION

var sum = (num1, num2) => { return num1 + num2;

};

console.log(sum(7, 12));

4.6 Arrays

We have learned that a variable can hold only one value, for example var i = 1, we can assign only one literal value to i. We cannot assign multiple literal values to a variable i. To overcome this problem, JavaScript provides an array.

An array is a special type of variable, which can store multiple values using special syntax.

Every value is associated with numeric index starting with 0. The following figure illustrates how an array stores values.

An array is a special variable, which can hold more than one value at a time and You access an array element by referring to the index number.

Example: Declare and Initialize JS Array

```
var stringArray = ["one", "two", "three"]; var numericArray = [1, 2, 3,
4]; var decimalArray = [1.1, 1.2, 1.3];
var booleanArray = [true, false, false, true];
var mixedArray = [1, "two", "three", 4];
```

Array Methods

concat() — Join several arrays into one

indexOf() — Returns the first position at which a given element appears in an array

join() — Combine elements of an array into a single string and return the string

lastIndexOf() — Gives the last position at which a given element appears in an array

pop() — Removes the last element of an array **push() —** Add a new element at the end

reverse() — Sort elements in a descending order

shift() — Remove the first element of an array

slice() — Pulls a copy of a portion of an array into a new

array **sort() —** Sorts elements alphabetically

splice() — Adds elements in a specified way and

position **toString() —** Converts elements to strings

unshift() —Adds a new element to the beginning

valueOf() — Returns the primitive value of the specified object

Pop and Shift: Removing Elements From Arrays in Javascript

You can remove elements from the end of an array in javascript using pop() and from the start of an array using shift(). Both of these functions return the removed elements.

/ **Array containing initial elements.** var fruits = ['apple', 'orange',

'kiwi']; alert(fruits.pop());

4.7 Timing Events

The `window` object allows execution of code at specified time intervals.

These time intervals are called timing events.

The two key methods to use with JavaScript are:

- `setTimeout(`*function, milliseconds*`)`
  Executes a function, after waiting a specified number of milliseconds.
- `setInterval(`*function, milliseconds*`)`
  Same as setTimeout(), but repeats the execution of the function continuously.

`window.setTimeout(`*function*, *milliseconds*`);`

The `window.setTimeout()` method can be written without the window prefix.

The first parameter is a function to be executed.

The second parameter indicates the number of milliseconds before execution.

What is an Event?

JavaScript events are actions that happen on their own. The events are happening all the time, JavaScript simply listens for the ones you specify, and then you can take some action. For example all of the following are events.

- When the page loads
- When a user moves her mouse
- When the user clicks a button
- When the user scrolls the page
- When the user clicks a form field
- Every key press is an event

Example : Simple Program on onload() Event handler

```
<html>
    <head>
    <script
    type="text/javascript">
    function time() {
        var d = new Date();
        var ty = d.getHours() +
        ":"+d.getMinutes()+":"+d.getSeconds();
        document.frmty.timetxt.value=ty;
        setInterval("time()",1000)
    }
    </script>
    </head>
<body onload="time()">
    <center><h2>Displaying Time</h2>
        <form name="frmty">
            <input type=text name=timetxt size="8">
        </form>
```

```
    </center>
</body>
</html>
```

Event Handler	Description
onAbort	It executes when the user aborts loading an image.
onBlur	It executes when the input focus leaves the field of a text, textarea or a select option.
onChange	It executes when the input focus exits the field after the user modifies its text.
onClick	In this, a function is called when an object in a button is clicked, a link is pushed, a checkbox is checked or an image map is selected. It can return false to cancel the action.
onFocus	It executes when input focus enters the field by tabbing in or by clicking but not selecting input from the field.
onLoad	It executes when a window or image finishes loading.
onMouseOver	The JavaScript code is called when the mouse is placed over a specific link or an object.
onMouseOut	The JavaScript code is called when the mouse leaves a specific link or an object.
onReset	It executes when the user resets a form by clicking on the reset button.
onSubmit	It calls when the form is submitted.

4.8 JavaScript Form Validation

It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user. JavaScript provides facility to validate the form on the client-side so data processing will be faster than server-side validation. Most of the web developers prefer JavaScript form validation. Through JavaScript, we can validate name, password, email, date, mobile numbers and more fields.

JavaScript Form Validation Example

In this example, we are going to validate the name and password. The name can’t be empty

and password can’t be less than 6 characters long.

Here, we are validating the form on form submit. The user will not be forwarded to the next page until given values are correct.

```
<script>
function validateform(){
var name=document.myform.name.value;
var
password=document.myform.password.value;
if (name==null || name==""){
  alert("Name can't be
  blank"); return false;
}else if(password.length<6){
  alert("Password must be at least 6 characters
  long."); return false;
  } }
</script>
<body>
<form name="myform" method="post" action="" onsubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
```

```
</form>
```

4.9 jQuery Overview & Syntax

jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax. It is free, open-source software using the permissive MIT License. As of May 2019, jQuery is used by 73% of the 10 million most popular websites.

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. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

jQuery is a JavaScript Library.

jQuery greatly simplifies JavaScript programming.

jQuery is easy to learn.

Before you start studying jQuery, you should have a basic knowledge of:

- HTML
- CSS
- JavaScript

Downloading jQuery

There are several ways to start using jQuery on your web site. You can:

- Download the jQuery library from jQuery.com
- Include jQuery from a CDN, like Google

There are two versions of jQuery available for downloading:

- Production version - this is for your live website because it has been minified

and compressed

- Development version - this is for testingand development (uncompressed and readable code)

Both versions can be downloaded from jQuery.com.

The jQuery library is a single JavaScript file, and you reference it with the HTML `<script>` tag (notice that the `<script>` tag should be inside the `<head>` section):

```
<head>
<script src="jquery-3.4.1.min.js"></script>
</head>
```

Why jQuery?

There are lots of other JavaScript libraries out there, but jQuery is probably the most popular, and also the most extendable.

Many of the biggest companies on the Web use jQuery, such as:

- Google
- Microsoft
- IBM
- Netflix

jQuery Syntax

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)
-

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

You might have noticed that all jQuery methods in our examples, are inside a document ready event:

```
$(document).ready(function(){
  // jQuery methods go here...
});
```

The jQuery team has also created an even shorter method for the document ready event:

```
$(function(){
 // jQuery methods go here...
```

```
});
```

Get Content - text(), html(), and val()

Three simple, but useful, jQuery methods for DOM manipulation are:

- `text()` - Sets or returns the text content of selected elements
- `html()` - Sets or returns the content of selected elements (including HTML markup)
- `val()` - Sets or returns the value of form fields

jQuery Traversing

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.

The image below illustrates an HTML page as a tree (DOM tree). With jQuery traversing, you can easily move up (ancestors), down (descendants) and sideways (siblings) in the tree, starting from the selected (current) element. This movement is called traversing - or moving through - the DOM tree.

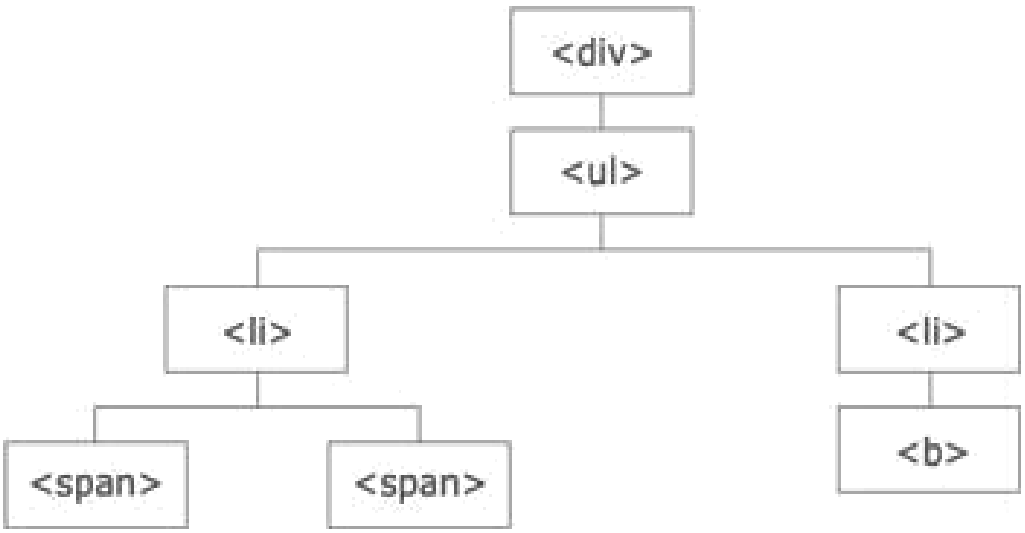


Illustration explained:

- The element is the **parent** of <ul>, and an **ancestor** of everything inside of it
- The <ul> element is the **parent** of both <li> elements, and a **child** of
- The left <li> element is the **parent** of , **child** of <ul> and a **descendant** of
- The element is a **child** of the left <li> and a **descendant** of <ul> and
- The two <li> elements are **siblings** (they share the same parent)
- The right <li> element is the **parent** of <b>, **child** of <ul> and a **descendant** of
- The <b> element is a **child** of the right <li> and a **descendant** of <ul> and

An ancestor is a parent, grandparent, great-grandparent, and so on.

A descendant is a child, grandchild, great-grandchild, and so on.

Siblings share the same parent.

Traversing the DOM

jQuery provides a variety of methods that allow us to traverse the DOM. The largest category of traversal methods are tree-traversal. The next chapters will show us how to travel up, down and sideways in the DOM tree.