**Html elements:**

<p> - paragraph tag

<br> - break tag breaks the line(empty new line)

<hr> - horizontal line -------------------

<!--> - comments

<link> - defines the relationship b/w a document and external resource

<link rel=”stylesheet” type=”text/css” href=”mystyle.css”>

<meta> - defines metadata about an html

Formatting elements:

<b> - bold text

<i> - italics text

<small> - smaller text

<strong> - important text

<sub> - subscripted text

<sup> - superscripted text

Attributes:

Class, id (unique), style (css)

Title - specifies extra information about an element (displayed as tool tip)

Html Links:

<a>

Html images

<img src=”smiley.gif” alt =”smiley face” width=”42” height = “42”> #single tag

Source alternative text width height of image

HTML division:

<div> - defines a division or a section in an HTML document. Used to group block-elements to format them with the CSS.

HTML lists:

Unordered list , Ordered list, Definition list

<UL type=”circle”>

<LI>red</LI> **.** red

<LI>yellow</LI> **.** yellow

<LI>blue</LI> **.** blue

</UL>

Ordered List

<OL type=”1”>

<LI>red</LI> 1. red

<LI>yellow</LI> 2. Yellow

<LI>blue</LI> 3. blue

</OL>

Definition list:

This contains a series of terms and definitions, and would typically be used in a glossary.

<dl>

<dt>Sump</dt>

<dd>A place where water completely fills the cave passage</dd>

<dt>Rift</dt>

<dd>A vertical fracture in the rock, created by geological stress</dd>

<dt>Abseil</dt>

<dt>Rappel</dt>

<dd>To descend a rope using a device to control speed </dd>

</dl>

Sump

A place where water completely fills the cave passage

Rift

A vertical fracture in the rock, created by the geological stress

Abseil

Rappel

To descend a rope using a device to control speed.

HTML Table:

border - thickness of border

cellspacing - space between cells

cellpadding - cell contents and cell walls space

<table border=”1” cellspacing=”2” cellpadding=”2”>

<tr>

<td></td>

<td></td>

<td> </td>(table data)

</tr>(table row)

</table>

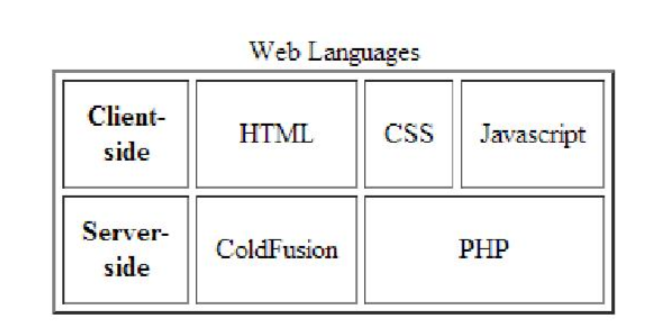
* <td> &nbsp; </td> - for empty lines ( “;” don’t forget)
* To bold <td> use <th></th> table header
* <caption> right below <table> and above first <tr>
* Aligning the table content: left, centre, right
* Column spanning uneven no. of cells in a table

<table **border=”2” cellspacing=”2” cellpadding=”10” width=”30%” align=”left”**>

***<caption>Web language</caption>***

<tr>

<th **width=”40%”** align=”center”>client-side</th>

<td **align=”center**”> HTML</td>

<td align=”center”>CSS</td>

<td align=”center”>JavaScript</td>

</tr>

<th align = “center”> Server-side</th>

<td align =”center”> ColdFusion</td>

<td align = “center” colspan=”2”> PHP</td>

</tr>

</table>

* <th align=”center” rowspan=”2”> web languages</th> row java html

Span php css

**HTML forms:**

<form>

<button>

<input type=”submit”>

<input type=”reset”>

Checkboxes

**Radio buttons**

<input type= ”radio” **name=”color”** value=”red”> Red color <br>

<input type= “radio” **name=”color”** value=”blue”> Blue color <br>

Menus

**Textfield**

<input type = “text” name=”sample\_text\_field”>

**Textarea**

<textarea name=”sample\_textarea”>Sample textarea</textarea>

Text input

<form>

First name: <input type=”text” name=”firstname”><br>

Last name: <input type=”text” name=”lastname”>

</form>

**Dropdown lists (Select Options )**

<select name=”day”>

<option value=”Friday” selected>Friday</option>

<option value=”Saturday”> Saturday </option>

<option value=”Sunday”>Sunday</option>

</select>

**Multiple Selections:**

<select name=”day” **multiple**>

Buttons:

Submit button: <input type=”submit” name=”submit” value=”Submit Form”>

Reset button: < input type=”reset” name=”reset” value=”Reset Form”>

Hidden Input: provides a way to pass data to server without displaying it to users. Used for tracking codes, keys, other information not pertinent to the users but helpful to the websites overall.

<input type=”hidden” name=”tracking\_code” value=”abc\_123”>

File Input: To allow users to add a file to a form.

<input type=”file” name=”file>

Organizing form elements:

Label

<label **for=”username”** > Username </label>

<input type=”text” name=”username**” id=”username”**>

<label><input type=”radio” name=”day” value=”Friday” checked> Friday </label>

Fieldset: fieldsets group form controls and labels into organized sections, much like a div

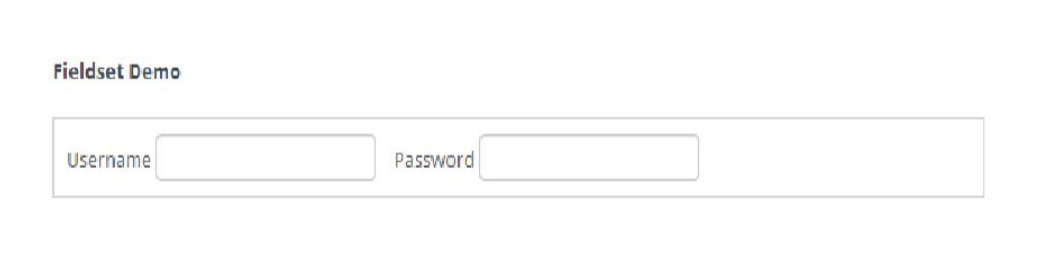
<filedset>

<label for=”username”>Username</label>

<input type=”text” name=”username” id=”username”>

<label for=”password”> Password</label>

<input type=”text” name=”password” id=”password”>

</fieldset>

CSS:

SYNTAX: Selector { property:value; property:value; ….}

P { color:blue; text-align:left; }

Grouping selectors p1,h1,table

Id selector: #p1

Class selector: p.classname or simply .classname [p.center, tr.center ]

Element selector: p

Inserting CSS:

External css ( A .css file)

<head>

<link rel=”stylesheet” type=”text/css” href=”mystyle.css”>

</head>

(Or)

<head>

<style type=”text/css”>

@import url(mystyle.css);

</style>

</head>

Internal CSS:

<head>

<style>

.left{

Color:blue;

Text-align:left;

Font-family:”calibri”;

}

</style>

</head>

Inline style: ( should use very sparingly)

<p style=”color:blue; text-align:left”> This is a paragraph </p>

Style strength: **inline >> internal >> external**

Provide power to **external use !important**

In .css

H3{

Color : blue !important;

}

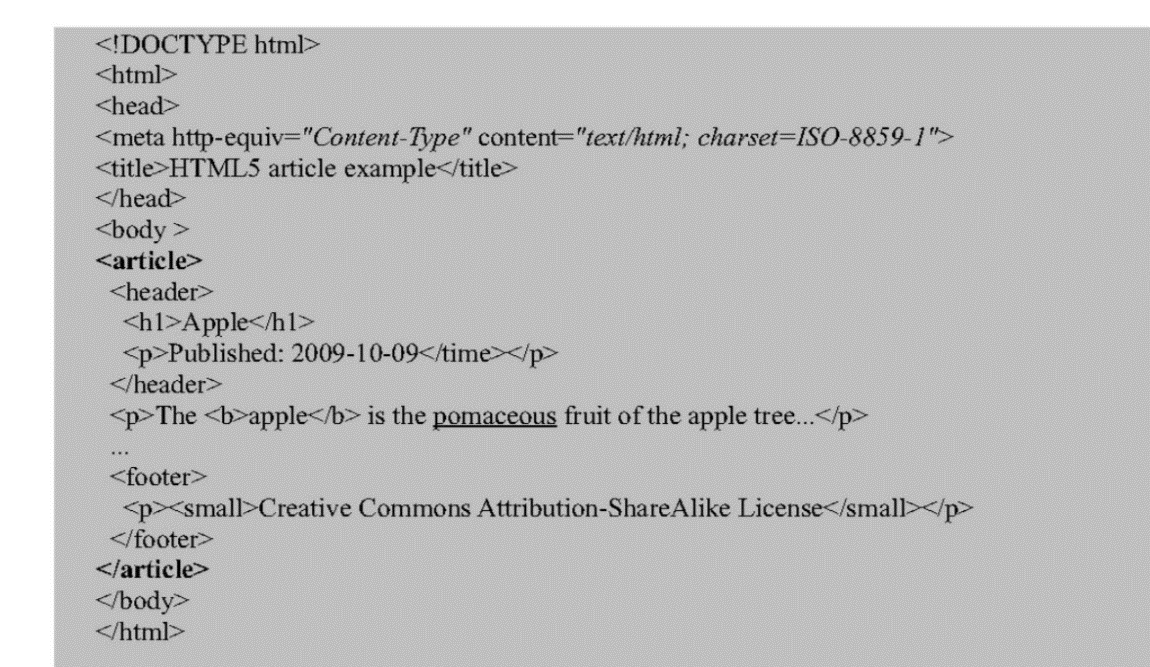
HTML – 5

<!DOCTYPE html>

HTML 5 tags

Structural elements:

<article> - contains either actual content or contains several different section elements or one or more article elements within it.



<section> - represents a generic section of a document.

<article>

<section>

<h4>What We Do</h4>

<p> We protect sharks….</p>

</section>

<section>

<h4>News Items</h4>

<p> Man eats shark….</p>

</section>

</article>

<section id=”penguins”> <img src=”Penguins.jpg” height=”550px” width=”800px”/></section>

<a href=”#penguins”>Penguins Image</a>

When we click on Penguins image link, page will be navigated to section id penguins.

<header>

<article>

<header>

<h1> Header of the article</h1>

<p>HTML5 makes life easy</p>

</header>

</article>

<footer>

<html>

<body>

<footer>

<h4> Footer example: </h4>

<address>

Spec Enterprises<br>

Bengaluru, Karnataka

</address>

</footer>

</body>

</html>

<nav> - defines the set of the navigation links.

<nav>

<a href=”/html/”>SAP</a> |

<a href=”/css/”>JAVA</a> |

<a href=”/js/” > ORACLE APPS</a> |

<a href=”/jquery/”>ASP>NET</a>

</nav>

<aside> - defines some content aside from the content it is placed in.

Used for some effects like pull quotes or sidebars, for advertising for groups of nav elements.

<body>

<p> My family and I visited The wonderla this summer.</p>

<aside>

<h4> Wonderla</h4>

<p> The wonderla is like a water world</p>

</aside>

</body>

<figure> - used to specify contents like illustrations, diagrams, photos, code, listings etc.

To include an image in center of page

<!DOCTYPE html>

<html>

<body>

<figure align=”center”>

<img scr=”tulip.jpg” alt=”The amazing flower” width=”300” height=”250”>

</figure>

</body>

</html>

<figcaption> - defines caption for a <figure> element, should be placed as first or last child.

<!DOCTYPE html>

<html>

<body>

<figure align=”center”>

<img scr=”tulip.jpg” alt=”The amazing flower” width=”300” height=”250”>

**<figcaption> Figure : A Flower</figcaption>**

</figure>

</body>

</html>

Semantic elements:

* <i> and <b> got semantic meaning in HTML5 though operation remained same.
* *<em>* **<strong>** **<mark>** (text highlight) remain almost the same.
* <small> - to denote side-comments, copy rights.

New Semantic Elements:

Time:

Attributes:

Datetime - represents a machine readable date/time of time.

<time datetime=”1905”>year 1905</time>

<time datetime=”1905-11-13”>13th November 1905</time>

Taking time input from user using JavaScript:

<!DOCTYPE>

<html>

<head>

<script type=”text/javascript”>

function format(){

var timeformat = document.getElementById(“userTime”).value;

var hour = timeformat.substring(0,2);

var min = timeformat.substring(3,5);

var time;

if (hour == 12){

time = hour + “-“ +min + “ “ + “PM”;

}

else if (hour >12){

time = hour – 12 + “-“ + “min” + “ “ + “PM”;

}

else{

time = hour + “-“ + min + “ “ + “AM”;

}

alert(time);

}

</script>

</head>

<body>

<form>

<input type=”time” id=”userTime” min=”00:00:00” onblur=”format()”/>

</form>

</body>

</html>

<mark> - for indicating text as marked or highlighted for reference purposes.

<p> This is a <mark> marked</mark> text </p>

**Media Elements:**

**<video>** - to specify video.

Attribute

Src - specifies the URL of the video file.

Autoplay - play as soon as loaded

Loop - loop

Controls - whether to display controls or not

Preload - how to load video while loading page

Muted - whether to mute or not

Poster - the url to the image you want to display till video is loaded

Height , width - of the video player

**Subtags:**

**<source>**

**Events:** abort, canplay, canplaythrough, Emptied, Ended, error, pause, play, seeked, stalled, volumechange

<video autoplay=”autoplay” controls>

<source src=”/media/video.ogg” type=”video/ogg”/>

<source src = “/media/video.mp4” type=”video/mp4”>

Your browser does not support HTML5 video.

</video>

**<audio>**

<audio autoplay = “autoplay” controls>

<source src=”/html5/audio.ogg” type=”audio/ogg”/>

Your browser does not support the <audio> element.

</audio>

**Embed:**

<Embed> - represents a container for external application or interactive content.

Attributes:

Height and width in pixels

Src URL

Type MIME type

Ex: To embed videlan VLC Media Player plugin in google chrome

<embed type=’application/x-vlc-plugin’ pluginspage=’http://www.videolan.org’ version=’VideoLAN.VLCPlugin2’ width=’800’ height=’600’ id=’vlc’ loop=’yes’ autoplay=’yes’ target=’sourceLocation(can be remote or local)’> </embed>

HTML 5 FORM and FORM attributes:

Form attributes:

Placeholder - grayed out text for reference.

<input type=”text” name=”user\_name” id=”user\_name” placeholder=”atleast 3 characters”>

Autofocus - focuses the field when page is rendered.

<input type=”text” name=”first\_name” id=”first\_name” autofocus>

Autocomplete - basing on previous input

<input type=”text” name=”tracking-code” id=”tracking-code” autocomplete=”off”>

Required - compulsory field

<input type=”text” id=”first-name” name=”first-name” required>

Pattern - it specifies a javascript regular expression for the field’s value to be checked against.

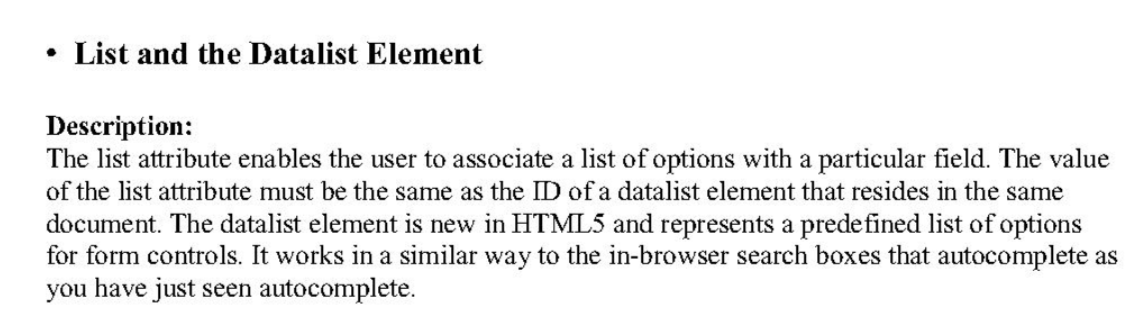
Ex: pattern for checking a single digit followed by 3 Upper Case Letters.

<label> Product Number:

<input pattern=”[0-9][A-Z]{3}” name=”product” type=”text” title=”single digit followed by three upper case letters.”/>

</label>

List and datalist:



<!DOCTYPE html>

<form name=”myForm” action=”/” method = “get”>

<label> Your favorite ice cream:

<datalist id = “icecream”>

<option value=”va”> Vanilla </option>

<option value=”st”> Strawberry</option>

<option value= “ac” >Almond Carnival</option>

<!--….-->

</datalist>

If other, please specify:

<input type =”text” name=”icecream” list=”icecream”>

</label>

</form>

Form ( attribute not tag)

The form attribute is used to associate an input, select, or textarea element with a form (known as its form owner) The primary use case for this is that input buttons that are placed within tables can now be associated with a form.

<input type=”button” name=”sort-l-h” form=”sort”>

**Formaction:**

Specifies the file or application that will submit the form. It has the same effect as the **action attribute** on the form element and can only be used with a submit or image button (type=”submit” or type=”image”). When the form is submitted it looks for Formaction then proceeds for action attribute.

<input type=”submit” value=”Submit” formaction=”abc.html”>

**Formenctype: how the data is encoded with the post method type.**

<input type=”submit” value=”Submit” formenctype=”application/x-www-form-urlencoded”>

**Formmethod : specifies the HTTP method (GET, POST, PUT, DELETE) == method attribute**

<input type=”submit” value=”Submit” formmethod=”POST”>

**Formtarget: specifies the target window for the form results. == target**

<input type=”submit” value=”Submit” formtarget=”\_self”>

**HTML 5 Form New Input Types**

**Search: search in amazon/flipkart …….**

Old: <input type=”text” name=”search”>

New: <input type=”search” name=”search”>

**Email:**

<input type=”email” name=”email” required>

**URL:**

<input type=”url” name=”url” required>

**TEL: telephone number validation**

<input type=”tel” name=”tel” id=”tel” required>

**Number:**

Age: <input type=”number” min=”5” max=”18” step=”1” value=”9” name=”age”>

**Dates:**

<input id=”dob” name=”dob” type=”date”>

<input id=”startdate” name=”startdate” min=”2012-01-01” max=”2013-01-01” type=”date”>

We can manipulate the date format based on your requirement using javascript and use required format in your backend.

<!DOCTYPE html>

<html>

<head>

<script type=”text/javascript”>

Function format(){

Var fulldate = document.getElementById(“fulldate”).value;

Var year = fulldate.substring(0, 4);

Var month = fulldate.substring(5,7);

Var date = fulldate.substring(8);

Var date = date + “-“ + month + “-“ + year;

Alert (date);

}

</script></head>

<body>

<!-- Date format implementation using HTML 5 -->

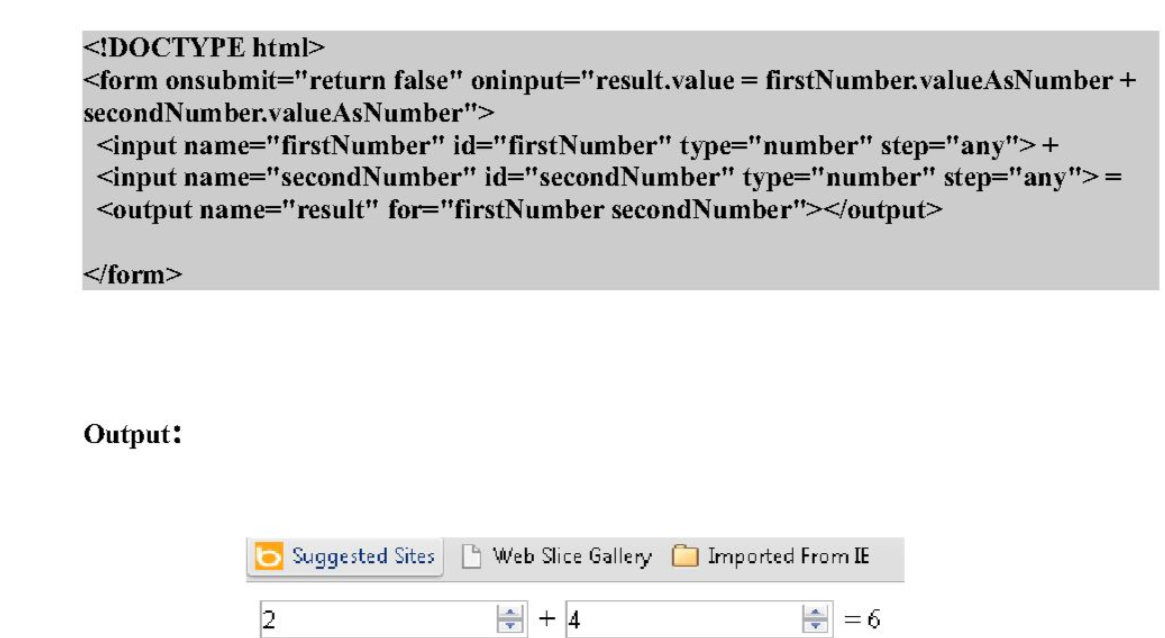
HTML 5 is amazing !!!!!!!!!!!

<form>

<input type=”date” id=”fulldate” onblur=”format()” />

</form>

**HTML 5 interesting Form elements for Web application**

**Output :** this element represents the result of a calculation or user action

**Progress:** represents the progress of a task.

<html> <body>

Downloading Progress:

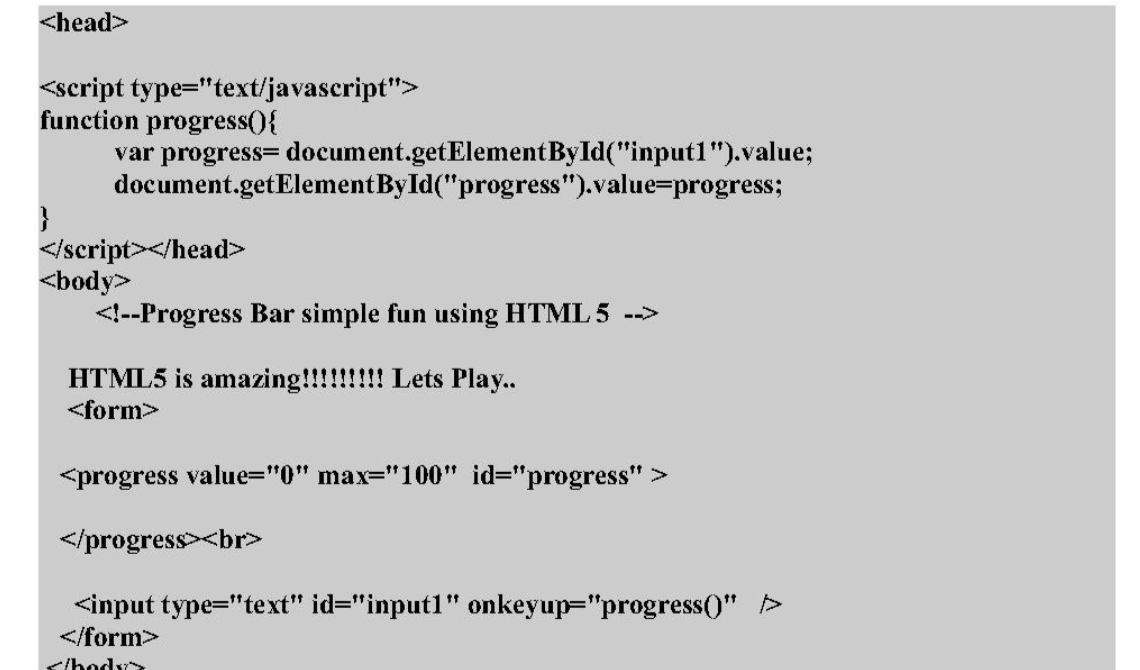
<progress value=”22” max=”100”></progress> <br>

Downloading progress:

<progress value=”0.8”> </progress>

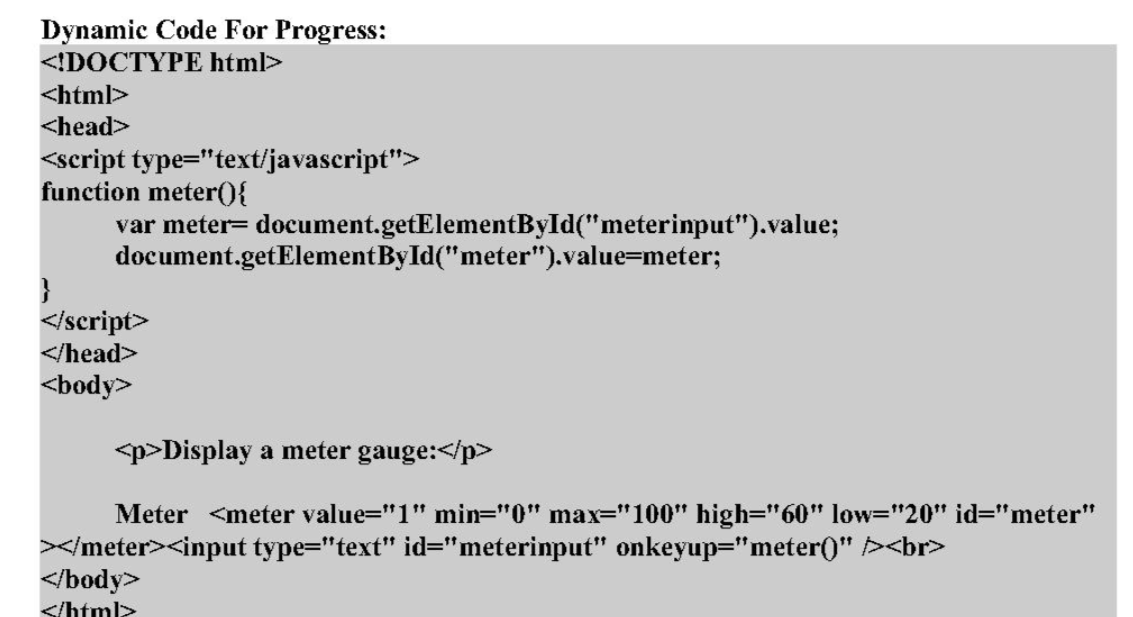
</body></html>

Dynamic code:



**Meter: fractional completion like percentage completion in a social networking profile**

**Dynamic code for meter:**

****

**Details and Summary:**

**<details> -** specifies additional details that user can view or hide on demand.

<**summary> -** to provide the summary.

**JAVA SCRIPT**

<script>

alert(‘Hello’);

</script>

Inclusion 2 types:

1. Writing directly in b/w <script>
2. Writing in separate .js file.

**<script src=”myfile.js”> </script>**

**Variables:**

**var variableName = data;**

ar x = “George”; //string

var c = ‘a’ ; //char

var I = 5; //int

var f = 4.5; //float

var b= true; //Boolean

var x = new Array(2,4,6,3); //array of integers

<script>

var x=”George”;

alert (‘Name is ‘ + x );

x = 24; // javascript is **type independent**

alert ( ‘ Age is ‘ + x );

</script>

**Events: onclick, onsubmit, onchange, onselect**

**Onclick:**

<button onclick=”alert(‘You clicked on me’)” > Click Me </button> // alert is method is called

**Onsubmit:**

<form action=”” method=”post” onsubmit=”alert(‘You are about to submit the form’)”>

<input type=”text”>

<input type=”submit”>

</form>

**Onchange : event occurs whenever content of particular element changes.**

<body>

<select onchange=”alert(‘you have selected an item’)”>

<option selected = “selected”> Select Item </option>

<option > Jam </option>

<option> Sauce </option>

<option> cream </option>

</select>

</body>

**Onselect : event occurs when text of an input tag is selected.**

<input type=”text” onselect=”alert(‘Text selected’)”>

**Functions**

**function** myFunction(arg1, arg2…….){

var value=10; //local declaration 🡪 local variable

return value;

}

<html>

<head>

<script>

var pi=3.14; //global variable

**function calculateArea(radius)**{

alert(‘area of circle is ‘ + (pi\*radius\*radius));

}

</script>

</head>

<body>

**<button onclick=”calculateArea(5)”>Find Area</button>**

</body>

</html>

**Conditional Operators:**

<html>

<head>

<script>

Function compare(x,y){

If (x>y){

Alert(x+ ‘ is greater than ‘ + y);

}

else if (x<y){

alert (y+ ‘ is greater than ‘ + x);

}

Else{

alert (‘both the numbers are equal’);

}

}

</script>

</head>

<body>

<button type=”button” onclick=”compare(5,7)”>Click Me</button>

</body>

</html>

**For loop:**

<html> <head>

<script>

**var x = new Array(2,4,6);**

var sum =0;

for ( var I =0 ; **i<x.length**; i++){

sum = sum+x[i];

}

alert(**‘**Sum of numbers is**’** + sum);

</script>

</head>

<body></body></html>

**PopUp boxes:**

In javaScript we can create small pop-up window to either **show messages or get input from user**.

Alertbox, Confirm box, prompt box.

Alertbox - to show messages as pop-up.

Confirm box - used for confirmations.

It returns either true or false when we click on Ok and Cancel respectively.

Syntax: Event=”return confirm(Message)”;

When we write return the Boolean value returned will determine whether the form will be submitted or not.

<form action=”/myservlet.do”>

<input type=”text”>

<input type=”submit” **onclick=”return confirm(‘Are you sure you want to submit?’)**”>

</form>

**Prompt Box - when we want to take input from user. //return type string**

**Syntax: prompt(Message, default value); //default value is optional**

<html>

<head> </head>

<script>

Function add(){

Var x=prompt(“Enter First Number”);

Var y=prompt(“Enter Second Number”);

Alert(‘The sum is ‘ + (parseInt(x) + parseInt(y)));

}

</script>

<body>

<button onclick=”add()”>Calculate</button>

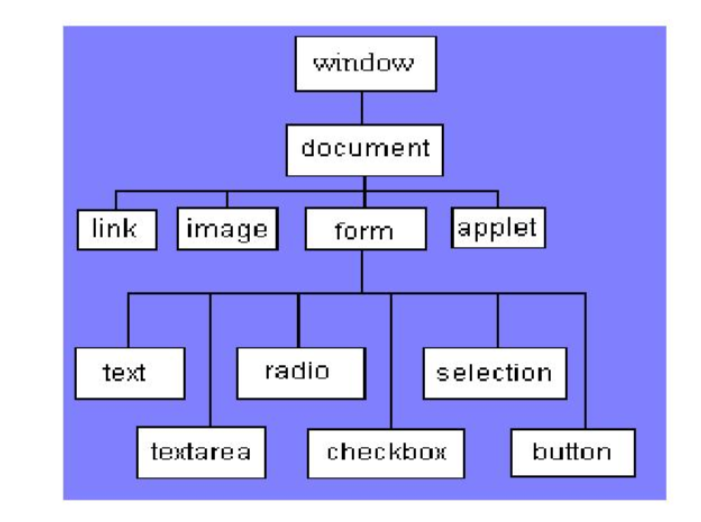
</body>

</html>

**Document Object Model:**

When a browser loads a web page, it creates the Document object Model of the page.

In java script function we can access the elements of the page either by traversing through the elements or by using some of the methods of these objects.



**getElementById:**

<script>

Function sayHello(){

Var x=document.getElementById(“name”);

Alert(‘Hello ‘ + x.value);

}

</script>

<body>

<form name=”myForm” action=”/register.do”>

<input name=”firstName” id=”name”>

<br>

<input type=”submit” onclick=”sayHello()”>

</form>

</body>

(Or)

Var x = document.forms[“myForm”][“name”].value

(or)

Var x = document.myForm.firstName.value

**getElementsByName: to get all elements with given name**

<script>

Function printItems(){

Var x= document.getElementsByName(“name”);

Var items=””;

For (var i=0; i<x.length; i++){

If(x[i].checked)

Items=items+x[i].value+” “;

}

Alert(‘Items selected ‘ + items);

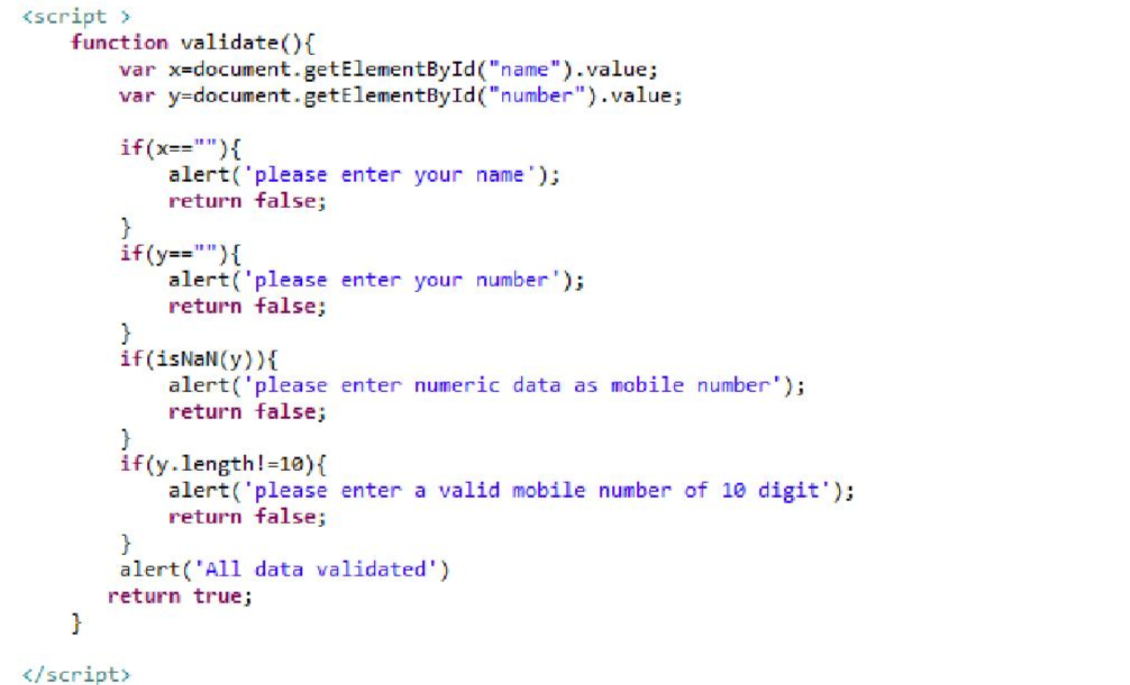
}

</script>

**getElementsByTagName:**

var x=document.getElementsByTagName(“input”);

**Validations:**

****

<body>

<form action =”/register.do” onsubmit=”return validate()”>

Name : <input type =”text” id=”name”> <br>

Mobile no: <input type=”text” id=”number”><br>

<input type=”submit” value=”submit”>

</form>

</body>

**Radio button validation:**

<script>

Function checkRadio(){

Var x=document.getElementByName(“gender”)

For (var i=0; i<x.length; i++){

If(x[i].checked)

Return true;

}

Alert(‘please select gender’);

Return false;

}

**Select box validaton**

<script>

Function checkRadio(){

Var x = document.myForm.country;

If(x.selectedIndex==0){

Alert(‘please select country’);

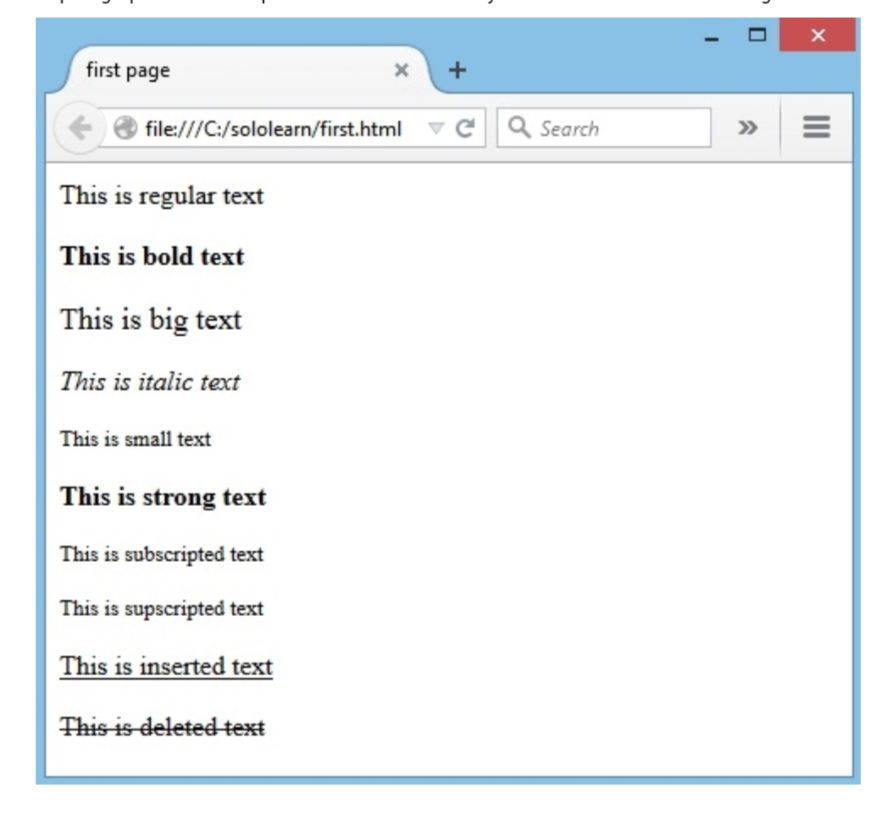
Return false;

}

Alert (x.value);

Return true;

} </script>

Text Formatting:

Paragraph - <p></p>

Bold -<b></b>

Big - <big>

Italic - <i></i>

Small - <small>

Strong - <strong>

Subscript - <sub>

Superscript - <sup>

Inserted - <ins>

Deleted - <del>

<hr> -horizontal line

<br> - line break

<!-- comments -->

<span> single line separator

<div> multiline separator

**HTML Attributes:**

Align - to align to left, right, center

Width, height - 50px, 50%,

Images:

<img src=”image.jpg” height=”” width=”” alt=”” **border=”1px”**/>

Links:

<a href=”-----” target**=”\_blank”**> link </a>

\_blank – to open in a new window or tab.

List:

<ol> <li> </li> <li> </li> </ol>

<ul> <li> </li> <li> </li> </ul>

**Table:**

<table **align=”center” border=”2”**>

<tr>

<td **bgcolor=”red”**> </td>

<td **colspan=”2”**> </td>

</tr>

</table>

**Block level:** <h1> <form> <li> <ol> <ul> <p> <pre> <table> <div>

**Inline level:** <b> <a> <strong> <img> <input> <em> <span>

* **Inline elements cannot any block level elements**

Other types of elements:

APPLET - embedded java applet

IFRAME - Inline frame

INS - inserted text

MAP - image map

OBJECT - embedded object

SCRIPT - script with an html element

FORM:

<form action=”http://.........’>

<form action=”url” method=”GET”>

<input type=”**text**” name=”username” >

<input type=”**password**” name=”password”>

<input type=”**radio/checkbox**” name=”gender” value=”male”>

<input type**=”radio/checkbox**” name=”gender” value=”female”>

<input type=”submit” value=”Submit”>

<**textarea** name=”message”> <**/textarea**>

**Color: #FFF or #FFFFFF or (255,255,255)**

**<body bgcolor=”#000099”>**

**<font color=”#FFFF00”> </font>**

**Frames:**

**<iframe height=”150” width=”300” src=”http://www.......” > </iframe>**

**HTML 5:**

**Content Models:**

**Metadata:** <base>, <link>, <meta>, <noscript>, <script>, <style>, <title>

**Embedded:** <audio>, <video>, <canvas>, <iframe>, <img>, <math>, <object>, <svg>

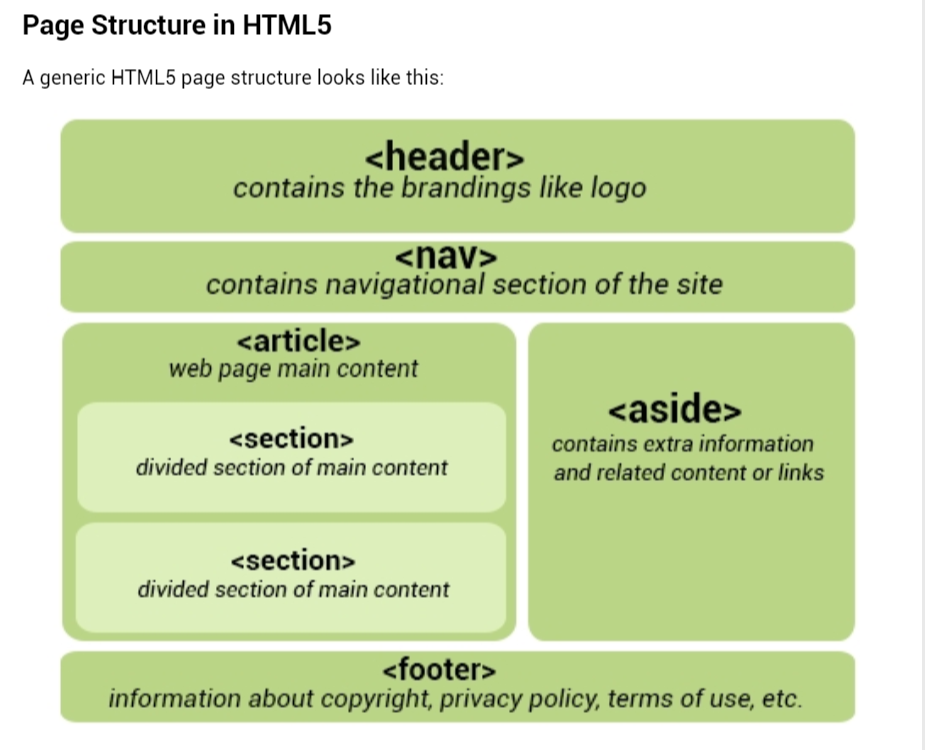
**Interactive:** <a>, <audio>, <video>, <button>, <details>, <embed>, <iframe>, <img>, <input>, <label>, <object>, <select >, <textarea>

**Heading:** <h1> <h2> <h3> <h4> <h5> <h6> <hgroup>

**Phrasing:** <img> <span> <strong> <label> <br/> <small> <sub> ……..

**Flow content: all**

**Sectioning content:** <article> <aside> <nav> <section>



<header> </header>

<footer> </footer>

<nav>

<ul>

<li> <a href=”#”>Home</a></li>

<li> <a href=”#”>Services</a></li>

</ul>

</nav>

Article, Section and Aside

<article>

<section>

</section>

<aside>

</aside>

</article>

**Audio:**

<audio src=”audio.mp3” controls autoplay loop> </audio>  
<audio controls>

<source src=”audio.mp3” **type=”audio/mpeg”**> //ogg // mp3

Similarly Video

**Progress:**

Status: **<progress min=”0” max=”100” value=”35”> </progress>**

**Web Storage API:**

**Storing a value:**

localStorage.**setItem**(“key1”, “value1”);

alert(localStorage.**getItem**(“key1”));

localStorage**.removeItem**(“key1”));

localStorage.**clear();**

**Geolocation:**

**navigator.geolocation.getCurrentPosition();**

**Drag and drop API**

**Svg - scalable vector graphics //animate for animation**

**Canvas + JS to draw graphics**

**HTML5 forms**

<label>

<form autocomplete=off>

<input type=”email” placeholder=”email.@example.com autofocus required

**HTML5 added several new input types:**  
- color  
- date  
- datetime  
- datetime-local  
- email  
- month  
- number  
- range  
- search  
- tel  
- time  
- url  
- week

**New input attributes in HTML5:**  
- autofocus  
- form  
- formaction  
- formenctype  
- formmethod  
- formnovalidate  
- formtarget  
- height and width  
- list  
- min and max  
- multiple  
- pattern (regexp)  
- placeholder  
- required  
- step

<input id=”mysearch” name=”searchItem” type=”search”>

CSS

H1 { color : orange; }

Selector property value

Type selectors - p, h1, h2, div

Id selectors - #id

Class selectors - .classname

Descendant selectors - #intro .first em

For intro as id, consider class with first as its name then in it find em tag

/\* css comments \*/

Working with text:

* font-family : Arial, “Helvetica Neue”, sans-serif;
* font-size : small, larger, medium, **xx-small, 20px, 20em** (em=pixels/16);
* font-style : normal, italic, **oblique;**
* font-weight : lighter, bolder, bold, **100(thin), 900(thick);**
* font-variant : normal**, small-caps, inherit;**
* color : **#00FF00, rgb(0, 255, 0)**
* text-align : left, right, center, **justify;**
* vertical-align : top, middle, bottom;

vertical-align : **baseline sub super -10**

**inline subscript superscript -10 px or pt or cm or %**

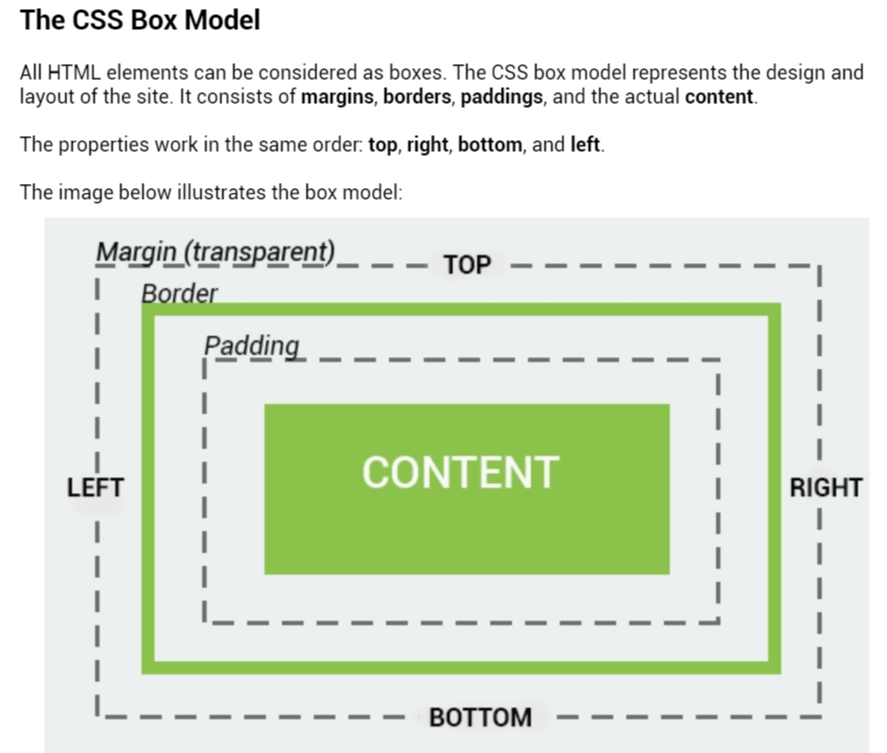
if having any **<div>** tags then **display:inline-table** and **display:table-cell** are used along with **vertical-align** for proper results.

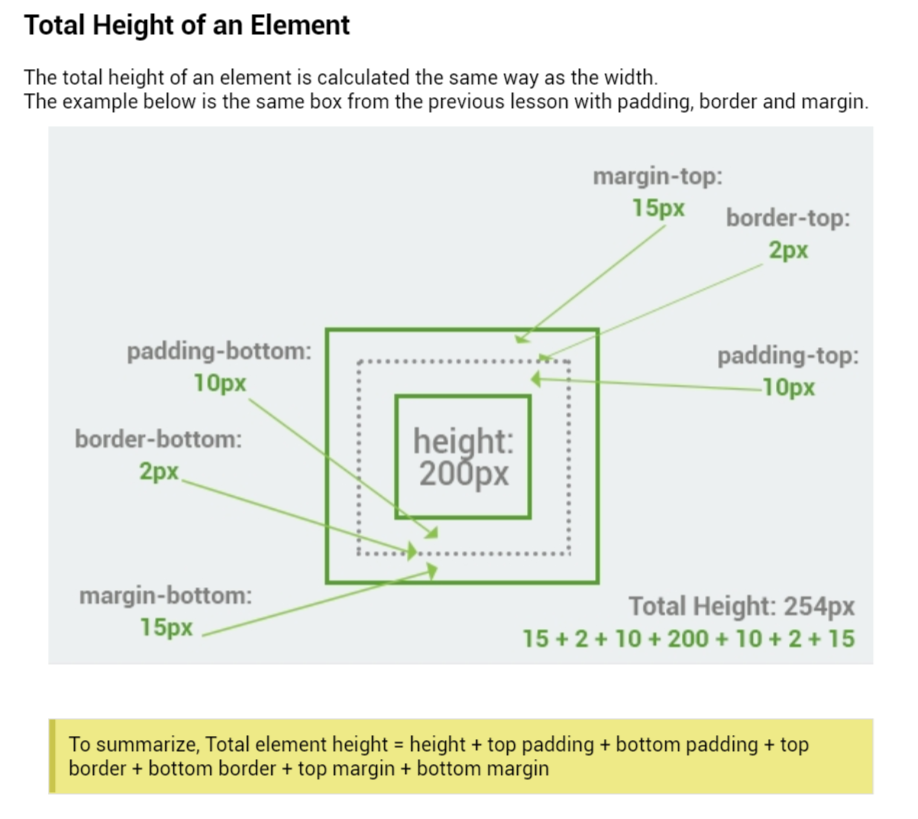
* text-decoration : none, inherit, overline, underline, line-through, blink
* text-indent : 60 (px, pt, cm, em,…), % or inherit
* text-shadow : 5px 2px 4px grey;

x-coordinate y-coordinate blur radius shadow color

units can be px, cm, mm, in, pc, pt etc….

* text-transform : capitalize, uppercase, lowercase;
* letter-spacing : normal, 5px, -1.5em.. ;
* word-spacing : normal, 30px, -20em.. ;
* white-space : nowrap, pre, pre-line, pre-wrap;
  + nowrap -- **continues all** in a single line (no matter what!)
  + pre -- wrap only on **line breaks** and **whitespaces**
  + pre-line -- considers **only line breaks**
  + pre-wrap -- at **end of box field, line break**, **whitespaces** (all valid)





* **Every HTML element is a CSS box object.**

**Borders:**

* Border : 5px solid green;
  + - * Width style color
* Border-style : none dotted dashed double grove ridge insert outset hidden
  + Border-right-style
  + Border-top-style
  + Border-left-style
  + Border-bottom-style
* Border-width : 5px, 20px, 12em;
  + Border-left/right/top/bottom-width
* Border-color : blue, #0000FF, rgb(0, 153, 0);
* **Padding : 10px;**
* Width : 10px, 10% ;
* Height : 10px, 10%;
  + Min-width
  + Min-height
  + Max-width
  + Max-height

**Background:**

* Background-color : #C0C0C0, rgb(135, 206, 235), LightGreen;
* Background-image : url(“css\_logo.png”);
* Background-repeat : repeat-x, repeat-y, no-repeat, inherit
  + - * + Horizontally repeat vertical no repeat parent
* Background-attachment: fixed, scroll, inherit
  + - * + BgImage Fixed bgImage scrolls parents
* Background-size
* Background-clip

**List style property:**

* List-style : square outside none;
  + - * Type position image
  + List-style-type : circle, lower-alpha, square, decimal, disc;
  + List-style-position : inside, outside
  + List -style-image : url(“logo.jpg”);

**Table styling:**

* Border-collapse : separate;
* Border-spacing : 20px 40px;
* Caption-side : top;
* Empty-cells : show / hide;
* **Table-layout**  : auto / fixed;

**Links styling:**

* a:link : a:link{ color:red} : for unvisited links
* a:visited : a.visited{} : for visited links
* a:active : a:active{} : on clicking it
* a.hover : a.hover{} : on mouse over it

**Customizing the Mouse cursor:**

* **cursor:help**
* **cursor:default**
* **cursor:crosshair**
* **cursor:pointer**

**JAVA SCRIPT**

**Document.write(“Hello World”);**

External javascript - <script **src=”demo.js**”></script>

Javascript comments - **// line comments /\* block comments \*/**

**var x = 100; - document.wrtie(x);**

**Data types:**

* Number
* Float
* Char
* String
* Boolean
* Array

Eval (“10 \* 20 + 8” )

Evaluates (string ) 🡪 208

“String” \* 5 🡪 NaN (Not a Number)

“ In JS % (mod) is used not only on integers but **also on Floating point numbers**

5.0 == 5 🡪 true (Equality)

**5.0 === 5 🡪 false (identical)**

10 !== ‘10’ 🡪 true (Not identical)

&& || ! are the logical operators

Var isAdult = (age < 18 ) ? “Too young” : “Old Enough”; (Ternary operator)

“20” + “58” 🡪 “2058” “hello” + “world” 🡪 “hello world”

If else if else switch for while do while continue break same as C

**Functions:**

**function functionName(){**

**//code;**

**}**

**functionName(); //functionCall**

* If a function is called with **missing arguments**, the missing values are set to **undefined,** which indicated that a variable has not been assigned a value.
* If you **do not return** anything from a function, **undefined** will be returned.

**Dialogue boxes:**

alert (“ Do you really want to leave this page?”);

**var user = prompt** (“Please enter your name”);

**var result = confirm**(“Do you really want to exit”);

Objects:

**Single object creation:**

Var person = {

Name: “john”, age:31, favColor: “green”, height: 183

};

Var x = person.age; person.age**.length**

Var y = person[‘age’]

**Multiple object (instances) creation:**

function person (name, age, color){

this.name = name;

this.age = age;

this.favColor = color;

}

Var p1 = new person ( “John”, 42, “green” );

Var p2 = new person ( “Amy” , 21, “red” );

Document.write( p1.age );

This.changeName = function (name){

This.name = name;

}

p.changeName(“John”);

**var courses = new Array (“HTML”, “CSS”, “JS” );**

**document.write(courses[10]) 🡪 undefined**

**var courses = new Array(3); note () not []**

**courses [0] = “HTML”;**

**var courses2 = [“js”, “c++”];**

**c = courses.concat(courses2);**

**courses.length**

**Associative arrays:**

Var person = [];

**Person[“name”] = “John”;**

**Person[“age”] = 46;**

Document.write(Person[“age”]);

The math object:

Math.PI;

Abs(x)

Ceil() exp() floor() log() max() min(a,b ,c …) pow(x,y) random() sqrt() tan()

setInterval(myAlert, 3000); // calls myAlert function every 3 seconds.