# HTML

## What is HTML?

#### HTML(Hyper Text Markup Language)

* + is a language for describing web pages.
  + not a programming language
  + uses markup tags to describe web pages.

#### Most Web documents are created using HTML.

* + Documents are saved with extension .html or .htm.

#### Markup?

* + Markup Tags are strings in the language surrounded by a < and a > sign.
  + Opening tag: <html> Ending tag: </html>
  + Not case sensitive.
  + Can have attributes which provide additional information about HTML elements on your page.

Example

* + - <body bgcolor="red">
    - <table border="0">

## HTML

#### An HTML document appears as follows:

|  |  |  |
| --- | --- | --- |
|  | DOCTYPE tells type, version,  language of particular document. |  |
| <!DOCTYPE HTML>  <html> |  |
| <head>  <title>Title of page</title>  </head>  <body>  This is my first homepage. <b>This text is bold</b>  </body>  </html> | | |

* + **HTML** **Head** **Section**: contain information about the document. The browser does not display this information to the user. Following tags can be in the head section: <base>, <link>, <meta>, <script>,

<style>, and <title>.

* + **HTML** **Body** **Section**: defines the document’s body. Contains all the contents of the document (like text,

images, colors, graphics, etc.).

* + Each document can have at most one <body> element

## Document (Body) Contents

#### Body Text

* + - * HTML truncates spaces in your text.
      * Use <br> to insert new lines.
      * Use <p> tag to create paragraphs.
      * Use <div> to hold division or a section in an HTML document
      * Use <span> as an inline container to mark up a part of a text, or a part of a document

#### Comments in HTML Document

* + - * Increase code readability; Ignored by the browser.
      * Example of HTML comment: <!-- This is a Sample HTML Comment -->

#### <div>…</div>: Creates divisions in Web pages. Can be used to set the alignment/class

for an entire section of the page. Eg:

* + - * <div align=“center”> This text is at the centre of the browser window </div>

## Attributes

#### HTML attributes are special words which provide additional information about the elements or attributes are the modifier of the HTML element.

* Eg of attributes used in <body> element :
  + BGCOLOR: Gives a background color to HTML page.
  + BACKGROUND: Use to specify images to the BACKGROUND.
  + TEXT: Specifies text color throughout the browser window
  + LINK, ALINK, VLINK: Used to specify link color, active link color & visited link color

#### Examples:

* + <body text=“red”> OR <body text=“#FF0000”>
  + <body link=“red” alink=“blue” vlink=“purple”>
  + <body bgcolor=“black”> OR <body bgcolor=“#000000”>
  + <body background[=“http://www.mysite.edu/img1.gif](http://www.mysite.edu/img1.gif)”>

<HTML>

<HEAD>

<TITLE>Body Tag</TITLE>

</HEAD>

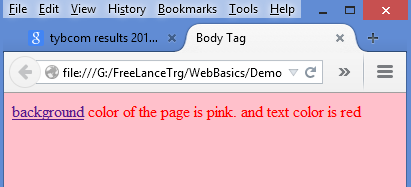
<BODY BGCOLOR="pink" text="red" alink="green" link="yellow">

<a href="body.html">background</a>

color of the page is pink. and text color is red

</BODY>

</HTML>



## Formatting tags

|  |  |  |
| --- | --- | --- |
|  |  | |
| <b>This is in bold font</b><br> |  |  |
| <i>This text is in Italics</i><br>  <u>This text is Underlined</u><br>  <small>This is small text.</small><br>  <font size=7>This text is very large</font><br>  <font color=”Blue”>This is Blue Text.</font><br>  <strike>This is strikethrough style text</strike><br> The chemical formula of water is h<sub>2</sub>o.<br>  A simple formula for a parabola is y = x<sup>2</sup>.<br> Applying some <mark>markup</mark> here<br>  Applying <em>emphasis</em><br>  <tt>teletype text</tt> | |  |

* Bold Font: <b>…</b>
* Italic: <i>…</i>
* Underline: <u>…</u>
* Strikethrough: <strike> or <s>
* Subscript: <sub>
* Superscript: <sup>

**7**

## Layout tags

#### Heading Styles:

##### <hn>… </hn>

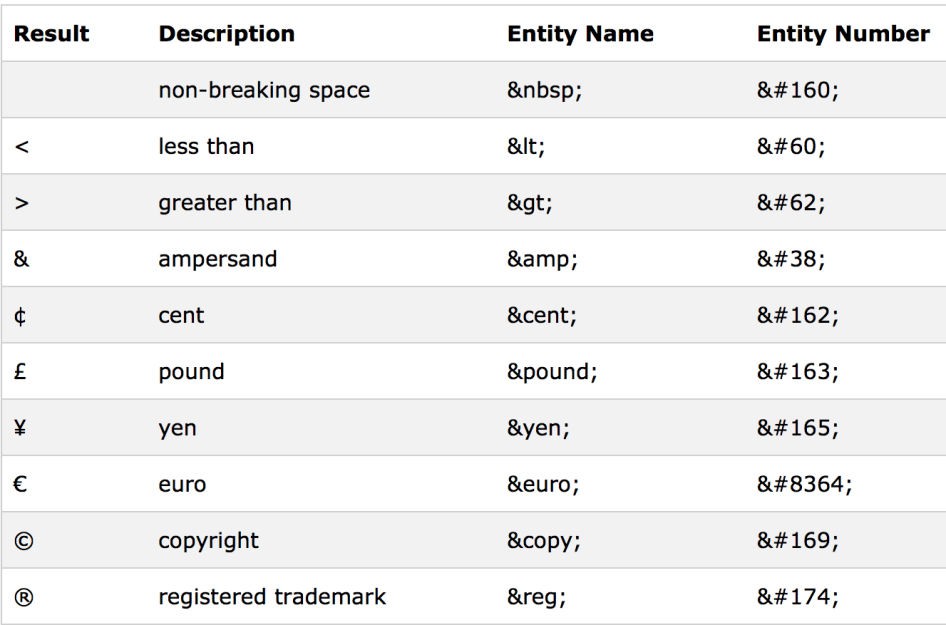
* + - Value of n can range from **1** **to** **6**
  + <h1 align=“center”>This is level 1 heading</h1>

|  |  |  |
| --- | --- | --- |
|  |  | |
| <html>  <head>  <title>Header Demo</title>  </head> |  |  |
| <body>  This is Regular body  <h1 align="left">This is level 1 heading</h1>  <h2 align="right">This is level 2 heading</h2>  <h3 align="center">This is level 3 heading</h3>  </body>  </html> | |  |

## Special Characters in HTML

#### Character Entities

* + Comprise following parts:
    - Ampersand (&),
    - Entity name or a #
    - Character code
    - Semicolon (;)
  + Included in HTML page using:
    - To display “>” symbol, use character code 62 i.e. &#62 or &gt;
    - For space, use &nbsp;

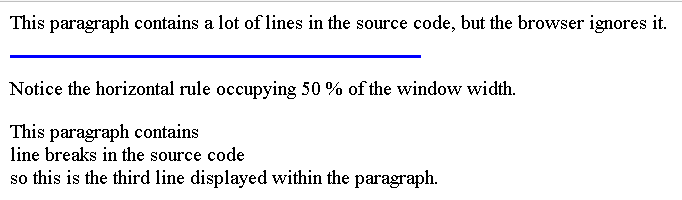


## Horizontal Lines in a Web Page

#### <hr> - horizontal Rule. Attributes:

* + Size: Line thickness <hr size=“5”>
  + Width: Line width either in pixels or % of browser window
    - <hr width=“100”> or <hr width=“60%”>
  + Align: Alignment values can be left, center or right <hr align=“center”>
  + Color: to display colored horizontal lines. Eg <hr color=“red”>

<body> <p>



This paragraph contains a lot of lines in the source code,

but the browser ignores it. </p>

<hr size="2" width="50%" color="blue" align = "left" >

<p>Notice the horizontal rule occupying 50 % of the window width.

<p>This paragraph contains <br> line breaks in the

source code <br>

so this is the third line displayed within the paragraph.

</body>

## Numbered List (Ordered List)

#### Automatically generate numbers in front of each item in the list

* + Number placed against an item depends on the location of the item in the list.
  + Example:

<body>

<ol>

<li>INDIA</li>

<li>SRILANKA</li>

</ol>

</body>



* + To start an ordered list at a number other than 1, use **START** attribute:

Example : <ol start=11>



#### Select the type of numbering system with the **type** attribute. Example:

* A-Uppercase letters. <OL TYPE=A>
* a-Lowercase letters. <OL TYPE=a>

<ol type="a">

<li>INDIA</li>

<li>SRILANKA</li>

</ol>

* I-Uppercase Roman letters



* i-Lowercase Roman letters
* 1-Standard numbers, default

## Bu leted List (Unordered List)

* Example:

<ul>

<li>Latte</li>

<li>Expresso</li>

<li>Mocha</li>

</ul>



* To change the type of bullet used throughout the list, place the TYPE attribute in the

<UL> tag at the beginning of the list.

* + DISC (default) gives bullet in disc form.
  + SQUARE gives bullet in square form.
  + CIRCLE gives bullet in circle form.



<ul>

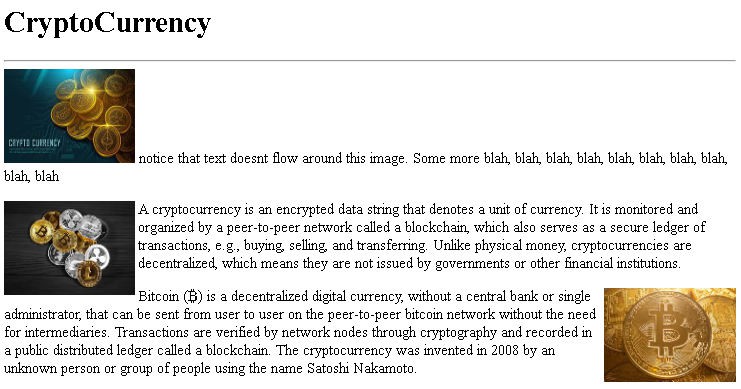
<li type=‘disc’>Latte</li>

<li type=‘square’>Expresso</li>

<li type=‘circle’>Mocha</li>

</ul>

## Adding Image



#### Images are added into a document using <img> tag.

* Attributes :
  + alt: holds a text description of the image; its optional, but is incredibly useful for accessibility - screen

readers read this description out to their users so they know what the image means

* + align: Image horizontal alignment; and also flow of text around image. Valid values: left, right
  + width/Height: Sets the width and height of the image.
  + src: Specifies the image path or source.

<h1>CryptoCurrency</h1><hr>

<img src="crypto.jfif" height="100" width="140"> notice that text doesnt flow aro ...

<p><img src="bitcoin2.jfif" align="left" height="100" width="140">

<p>A cryptocurrency is an encrypted data string that denotes a unit of currency.

...

<p><img src="bitcoin1.jfif" align="right" height="100" width="140">

<p>Bitcoin (₿) is a decentralized digital currency, without a central bank or

single administrator, that can be sent from user to user on the peer-to-peer ...

## Table

#### Use <table> tag to create a table.

* Table Attributes:
  + Border: applies border to table. Eg : <table border=“2”>………</table>
  + Align: defines the horizontal alignment of the table element.
    - Values of the align attribute are right, left and center. <table align=“center”>
  + Width: defines the width of the table element.
    - <table width=“75%”>………</table>
    - <table width=“400”>………</table>
  + Example:

<table border="1">

<tr>

<td>Row-1, cell-1</td>

<td>Row-1, cell-2</td>

</tr>

</table>



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

* + The <caption> tag must be inserted immediately after the <table> tag

<table style="width:100%">

<caption>Monthly savings</caption>

<tr><th>Month</th><th>Savings</th></tr>

…

</table>

## Table Data

#### An HTML table has two kinds of cells:

* + Header Cells <th>: Contain header information; The text is bold and centered.
  + Standard Cells <td>: Contain data; The text is regular and left-aligned.

|  |  |  |
| --- | --- | --- |
|  | |  |
| <table>  <tr> <th>Column1 Header</th> <th>Column2 Header</th></tr>  <tr> <td>Cell 1,1</td> <td>Cell 1,2</td> </tr>  <tr> <td>Cell 2,1</td> <td>Cell 2,2</td> </tr> |  |  |
| </table> | |  |

* You can insert a bgcolor attribute in a <table>, <td>, <th> or <tr> tag to set the color of the particular element.



<table bgcolor="cyan">

<tr bgcolor="blue">

<th bgcolor="red">Header 1</th><th>Header 2</th>

</tr>

<tr>

<td bgcolor="green">data 1</td><td>data 2</td>

</tr>

</table>

## Ce l Spanning

#### colspan="number of columns"

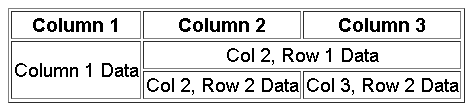
* + By default, the number of columns in a table is defined by the number of table data cells appearing

in the table row that contains the most data.

* + Ideally, place the same number of data cells in each table row. If a table row does not contain the requisite number of table cells, then it will essentially be in 'error' and will be displayed with a missing cell.

#### rowspan="number of rows"

* + Forces a table cell to span the number of rows specified by the given value.



<table border=1>

<tr>

<th>Column 1</th><th>Column 2</th><th>Column 3</th>

</tr>

<tr>

<td rowspan=2>Column 1 Data</td>

<td align=center colspan=2>Col 2, Row 1 Data</td>

</tr>

<tr>

<td>Col 2, Row 2 Data</td><td>Col 3, Row 2 Data</td>

</tr>

</table>

## Hyperlink

#### Hyperlinks access resources on the internet.

* Create a link with <a href=“”> (anchor)

>

|  |  |  |
| --- | --- | --- |
|  |  | |
| * <a href[="http://www.mysite.com/login.htm](http://www.mysite.com/login.htm)">Login Here</ * Hello, Welcome to <a href="welcome.htm">My Site</a> | a |  |
| * I have some <a href="<http://www.state.edu/info/info.htm>"> older information</a> about this   subject. | |  |



<ul>

<li><a href='home.html'>mumbai</a></li>

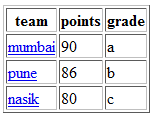
<li><a href='home.html'>pune</a></li>

<li><a href='home.html'>nasik</a></li>

</ul>

<table border=1>

<tr><th>team<th>points<th>grade</tr>



<tr> <td><a href='home.html'>mumbai</a></td><td>90</td><td>a</td> </tr>

<tr> <td><a href='home.html'>pune</a></td><td>86</td><td>b</td> </tr>

<tr> <td><a href='home.html'>nasik</a></td><td>80</td><td>c</td> </tr>

</table>

## Use of Image as a Hyperlink

#### Images used as hyperlinks:

* + - <A href="image1.html"><img src="home.gif"></a>
  + Images contained within a table:



<table align="center" border="2">

<tr><th>Product</th><th>Cost</th><th>Image</th></tr>

<tr>

<td>Pencil</td>

<td>$8</td>

<td><img src="pencil.gif"></td>

</tr>

<tr>

<td>Brush</td>

<td>$15</td>

<td><img src="paintbrush.gif"></td>

</tr>

<tr>

<td>Pin</td>

<td>$3</td>

<td><img src="pin.gif"></td>

</tr>

</table>

## HTML Forms for User Input

#### Data Submission using a Form

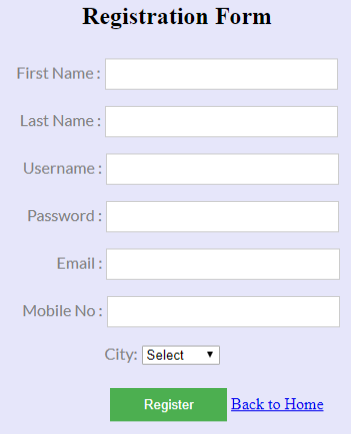
<form method=“get/post” action=“URL”>

*Field* *definitions*

</form>

* HTML forms are used to accept of user input.
* A form contains form elements.
* Form elements are elements that allow users to enter information in a form.
* Define a form with the <form> tag.</form>

#### <input> tag is used to create form input fields.



* Type attribute of <input> tag specifies the field type
  + Single line text box <input type=“text”>
  + Password field <input type=“password”>
  + Hidden field <input type=“hidden”>
  + Radio button <input type=“radio”>
  + Checkbox <input type=“checkbox”>
  + File selector dialog box <input type=“file”>
  + Button <input type=“button”>
  + Submit/Reset <input type=“submit/reset”>
* <textarea>
* <select>
* <button>
* Text fields: **Single** **Line** : used to type letters, numbers, etc. in a form.
* <INPUT TYPE=”type” NAME=”name” SIZE=”number” VALUE=”value” maxlength=n>
* Eg:

<form>

First name: <input type="text" name="firstname“ value=“fname> Last name:<input type="text" name="lname">

</form>

#### Text Area (Multiple Line Text Field)

* A text area can hold an unlimited number of characters. Text renders in a fixed-width font (usually Courier).
* You can specify text area size with cols and rows attributes.

<textarea name=“name” rows=“10” cols=“50” [disabled] [readonly]> Default-Text

</textarea>

<textarea name=“address” rows=5 cols=10>

Please write your address

</textarea>

<textarea rows=“4” cols=“20”>

#### Password

* <input type=“password” name=“name” size=n value=“value” [disabled]>
* Eg: Enter the password:<input type=“password” name=“passwd” size=20 value=“abc”>

#### Check box - Lets you select one or more options from a limited number of choices.

<input type=“checkbox” name=“name” value=“value” [checked] [disabled]> Checkbox Label

* + Content of value attribute is sent to the form's action URL.

<input type=“checkbox” name=“color1” value=“0”/>Red

<input type=“checkbox” name=“color3” value=“1” checked/>Green



#### Radio Buttons

* + <input type=“radio” name=“name” value=“value” [checked] [disabled]> Radio Button Label
  + *Content* *of* *the* *value* *attribute* *is* *sent* *to* *the* *form's* *action* *URL*.



<form>

<input type="radio" name=“gender" value="male"/>Male<br>

<input type="radio" name=“gender" value="female"/> Female

</form>

* Hidden form field - Allows to pass information between forms:
  + <input type=“hidden” name=“name” size=“n” value=“value”/>
  + <input type=“hidden” name=“valid\_user” size=“5” value=“yes”/>

## Form elements

#### File Selector Dialog Box

* + <input type=“file” name=“name” size=“width of field” value=“value”>

<FORM>

Select file to upload:

<INPUT name=“file” type=“file”> <BR>

<INPUT type=“submit” >

</FORM>



#### Buttons:

* + To add a button to a form use:
  + <input type=“button” name=“btnCalculate” value=“Calculate”/>
  + To submit the contents of the form use:
  + <input type=“submit” name=“btnSubmit” value=“Submit”/>
  + To reset the field contents use:
  + <input type=“reset” name=“btnReset” value=“Reset”/>
  + You can also create button using <button> tag ; defines a push button. Inside this element you can put

content, such as text or images.

* + <button type="button">Click Me!</button>

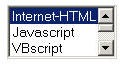
## Drop-Down List

<select name=“name” multiple=“true/false” size=n [disabled]>

<option [selected] [disabled] [value]>Option 1</option>…

</select>

* + - Multiple: States if multiple element selection is allowed.
    - Size: Number of visible elements.
    - Disabled: States if the option is to be disabled after it first loads.



<form>

<select multiple size="3" name="pref">

<option value="ih" selected>Internet-HTML</option>

<option value="js">Javascript</option>

<option value="vbs">VBscript</option>

<option value="as">ASP</option>

</select>

</form>

#### Forms with labels

<form>

<label for=“uname">User name : </label>

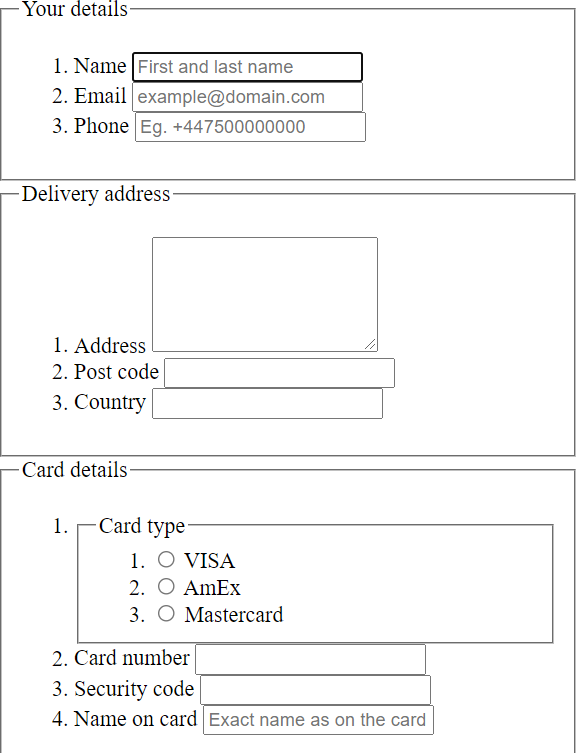


<input id=“uname" name=“username">

<button>Submit</button>

</form>

#### Group related elements in a form



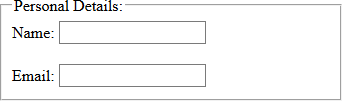
<form>

<fieldset>

<legend>Personal Details:</legend> Name: <input type="text"><br><br> Email: <input type="text"><br>

</fieldset>

</form>



* An HTML iframe is used to display a web page within a web page.
  + specifies an inline frame.
  + Use the height and width attributes to specify the size of the iframe
  + With CSS, you can also change the size, style and color of the iframe's border Eg <iframe src=“” style="border:none;“>
  + Alternatively use the frameborder attribute : 1 (yes) or 0 (no)



<html>

<body>

<p>Some HTML text before iframe display</p>

<iframe src="numberedList.html"

width="20%" height="200">

Your browser doesn’t support inline frames.

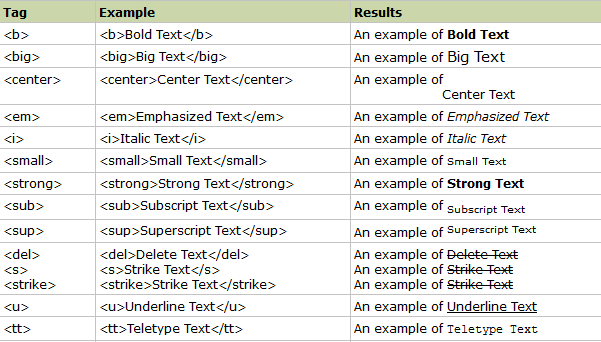
</iframe>

<p> Some text after iframe display </p>

</body>

</html>

Formatting tags



# HTML5

## What's new in HTML5?

#### HTML5 offers new enhanced set of tags

* + New Content Tags : <nav>,<section>,<header>,<article>,<aside>,<summary>
  + New Media Tags : <video>,<audio>
  + New Dynamic drawing : <canvas>graphic tag
  + New form controls, like calendar, date, time, email, url, search

#### Support for JavaScript APIs

* + Canvas element for 2D drawing API
  + Video and audio APIs
  + APIs to support offline storages
  + The Drag & Drop APIs
  + The Geolocation API
  + Web workers, WebSQL etc

#### The DOCTYPE tells the browser which type and version of document to expect.

* + The DOCTYPE announcement is not a HTML label; it is a guideline to the web program about what variant of HTML the page is composed in.

<!DOCTYPE html>

## HTML5 Attributes for <input>

#### A Form is one of the most basic and essential feature of any web site

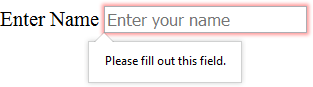
* + HTML5 brings 13 new input types and 14 new attributes
  + HTML5 introduces these data types via the <input type=”\_NEW\_TYPE\_HERE\_”/> format

#### **Placeholder** - A placeholder is a textbox that hold a text in lighter shade when there is no value and not focused

* + <input id="first\_name" placeholder=“This is a placeholder">
  + Once the textbox gets focus, the text goes off and you shall input your own text

#### **AutoFocus** - Autofocus is a Boolean attribute of form field that make browser set focus on it when a page is loaded

* + <input id ="Text2" type="text" autofocus/>
* **Required** - A "Required Field" is a field that must be filled in with value before submission of a form
* <input name="name" type="text" required />



* **Email** - Checks whether the string entered by the user is valid email id or not.
  + <input id="email" name="email" type="email" />

#### **Search** - used for search fields (behaves like a regular text field).

* + <input id="mysearch" type="search" />

#### **Tel** - used for input fields that should contain a telephone number.

* + <input type="tel" name="usrtel">
  + <input type="tel" name="phone“ pattern**="[2-9][0-9]{2}-[0-9]{3}-[0-9]{4}“** title="North American format: XXX-XXX-XXXX">

#### **url** - is used for input fields that should contain a URL address.

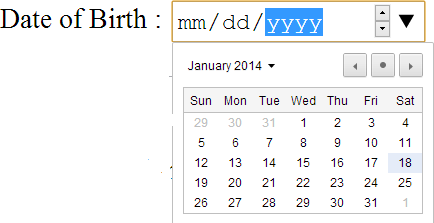
* + Depending on browser support, the url field can be automatically validated when submitted.

#### **color** – displays a color palette

* **Number** - used for input fields that should contain a numeric value.
  + Min and max parameters provided to limit the values.
  + Browser will treat it as simple textfield if it doesn’t support this type.
  + <input id="movie" type="number" value="0"/>
  + <input type="number“ min="0“ max="50“ step="2“ value="6">

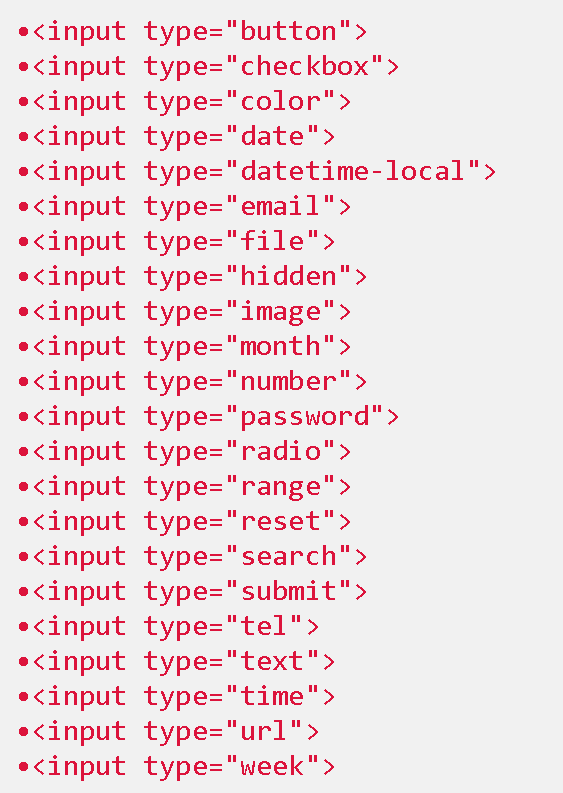
#### **Range** - used for input fields that should contain a value within a range

* + Browser will treat it as simple textfield if it doesn’t support this type
  + <input id="test" type="range"/>



* + <input type="range" min="1" max="20" value="0">

#### **Date** - used for input fields that should contain a date.

* + Depending on browser support, a date picker can show up in the input field.
  + <input id="meeting" type="date“ />
* **month** - Selects month and year
* **week** - Selects week and year
* **time** - Selects time (hour and minute)
* **datetime** - Selects time, date, month and year
* **datetime-local** - Selects time, date, month and year (local time)
* required: Specifies whether a form field needs to be filled in before the form can be submitted.
* minlength and maxlength: Specifies the minimum and maximum length of textual data

(strings).

* min and max: Specifies the minimum and maximum values of numerical input types.
* type: Specifies whether the data needs to be a number, an email address, or some other specific preset type.
* pattern: Specifies a regular expression that defines a pattern the entered data needs to

follow.

<form>

<label for="uname">User Name</label>

<input id="uname" name="uname" required pattern="[a-zA-Z]\*">

<button>Submit</button>

</form>

* **Pattern** : specifies a JavaScript regular expression for the field’s value to be checked against. pattern makes it easy for us to implement specific validation for product codes, invoice numbers, and so on.

<label>Product Number:</label>

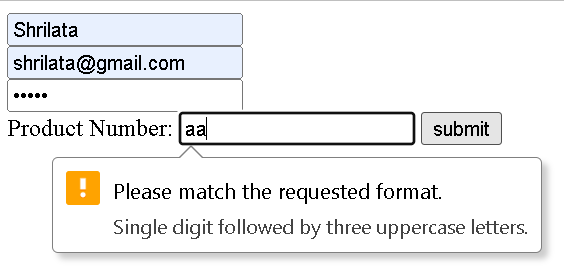
<input type="text"

name="product" pattern="[0-9][A-Z]{3}"

title="Single digit followed by three uppercase letters.“

/>

* + The title attribute **specifies** **extra** **information** **about** **an** **element**. The information is most often shown as a tooltip text when the mouse moves over the element.



#### Until now, there has never been a standard for playing audio on a web page.

* + Today, most audio is played through a audio plugin (like Microsoft Windows Media player, Microsoft Silverlight ,Apple QuickTime and the famous Adobe Flash).
  + However, not all browsers have the same plugins.
  + HTML5 the audio element to play sound files, or an audio stream.
  + Other properties like auto play, loop, preload area also available



<audio controls>

<source src="vincent.mp3" type="audio/mpeg"/>

<source src="vincent.ogg" type="audio/ogg"/>

</audio>

#### Today, most videos are shown through a plugin (like Flash). However, not all browsers have the same plugins.

* + HTML5 provides <video> element to include video
  + Supported video formats for the video element : Ogg, MP4, WebM, .flv, .avi
  + Attributes : width, height, poster, autoplay, controls, loop, src

<video controls="controls" width="640" height="480" src=“bunny.mp4" />

Your browser does not support the video element.

</video>

#### A canvas is a rectangle in your web page within which you can use JavaScript to draw shapes

* + Canvas can be used to represent something visually in your browser like Simple Diagrams, Fancy user interfaces, Animations, Charts and graphs, Embedded drawing applications, Working around CSS limitations
  + The canvas element has several methods for drawing paths, boxes, circles, characters, and adding images.
  + The canvas element has no drawing abilities of its own. All drawing must be done inside a JavaScript

<canvas id="myCanvas"></canvas>

<script type="text/javascript">

var canvas=document.getElementById('myCanvas'); var ctx=canvas.getContext('2d'); ctx.fillStyle='#FF0000';

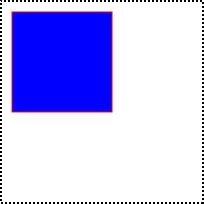
ctx.fillRect(0,0,80,100);

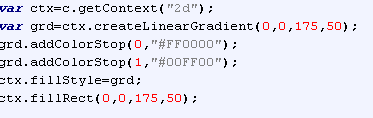
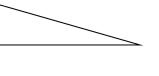
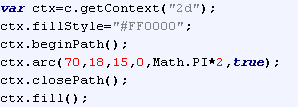
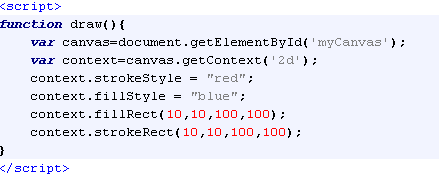
</script>

<canvas id="myCanvas" width="200" height="100">

</canvas>

Canvas examples

* 

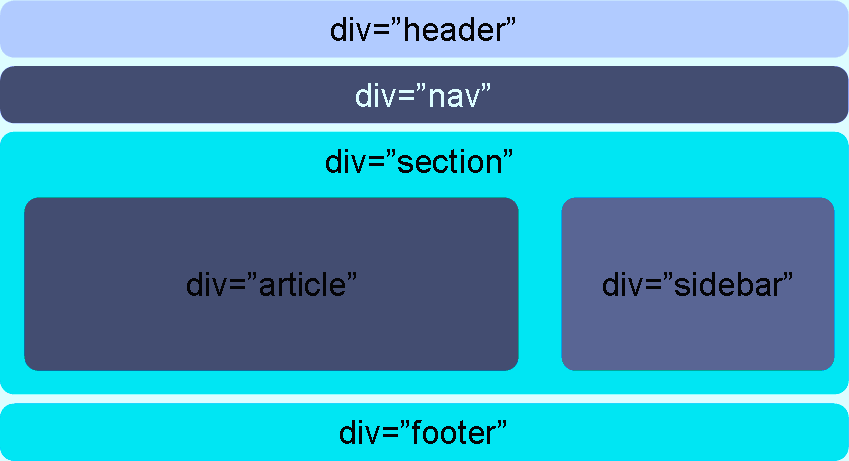
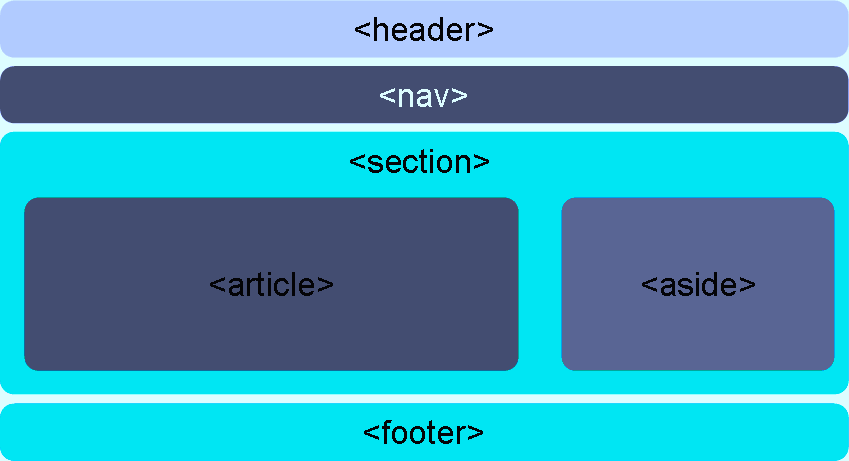


Laying out a page with HTML5

* Most HTML 4 pages include a variety of common structures, such as headers, footers and columns
* It’s common to mark them up using div

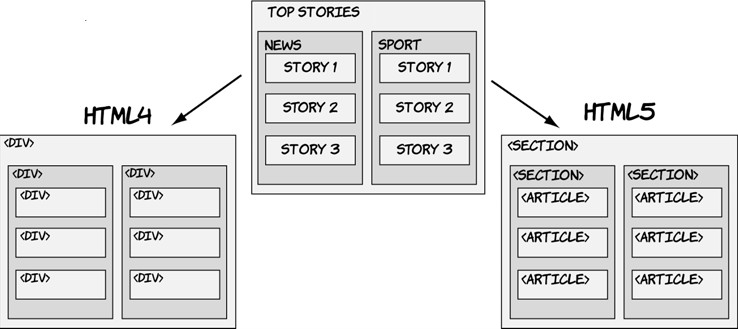
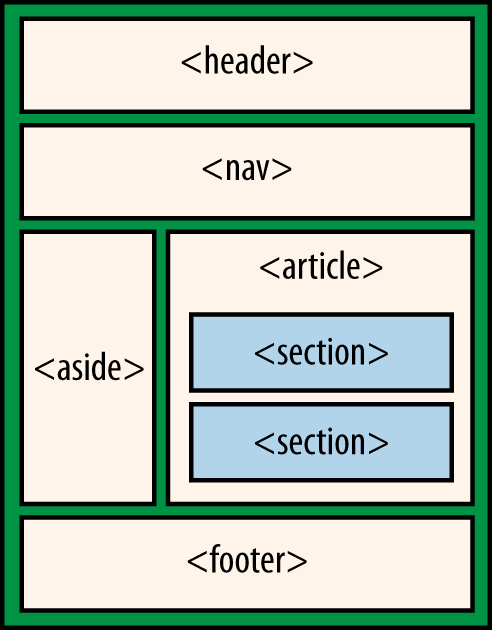
elements, giving each a descriptive id or class

* HTML 5 addresses this issue by introducing new elements for representing each of these different sections
* Elements that make it much easier to structure pages

## New Semantic Elements

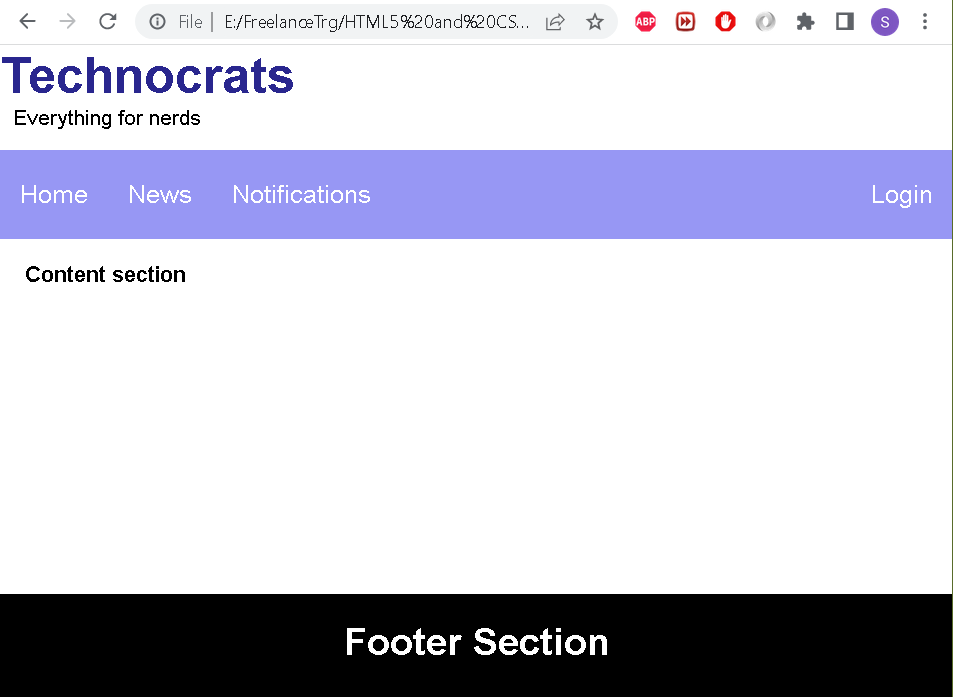
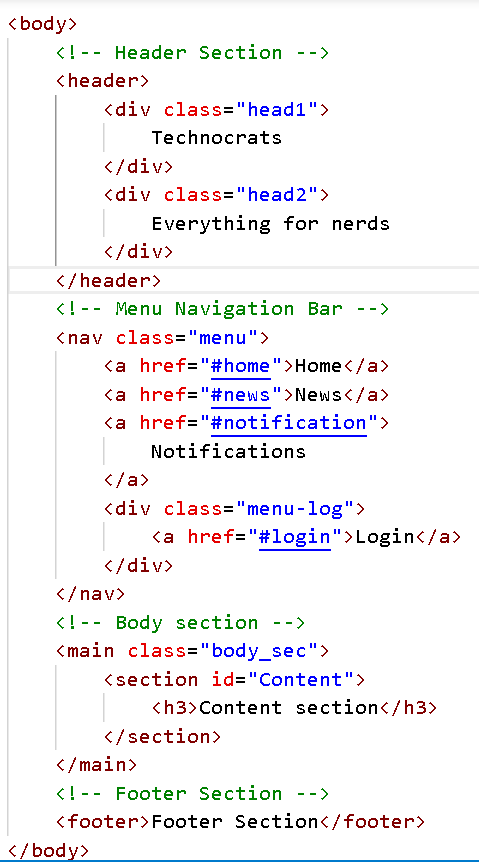
#### **<section>** **:** can be used to thematically group content, typically with a heading.



* + **<article>:** element represents a self-contained composition in a document, page, application, or site that is intended to be independently distributable or reusable
    - Eg a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment

#### **<nav>**: Represents a major navigation block. It groups links to other pages or to parts of

the current page.



blocking elements2.html

## New Semantic Elements

* + **<Header>:** tag specifies a header for a document or section. Can also be used as a heading of an blog entry or news article as every article has its title and published date and time
  + **<aside>:** The "aside" element is a section that somehow related to main content, but it can be separate from that content header and footer element in an article.
  + **<footer>:** Similarly to "header" element, "footer" element is often referred to the footer of a web page.
    - However, you can have a footer in every section, or every article too

#### **<figure>:** The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

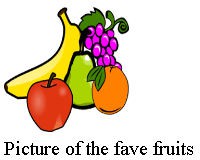
* + - This element can optionally contain a figcaption element to denote a caption for the figure.

<figure>

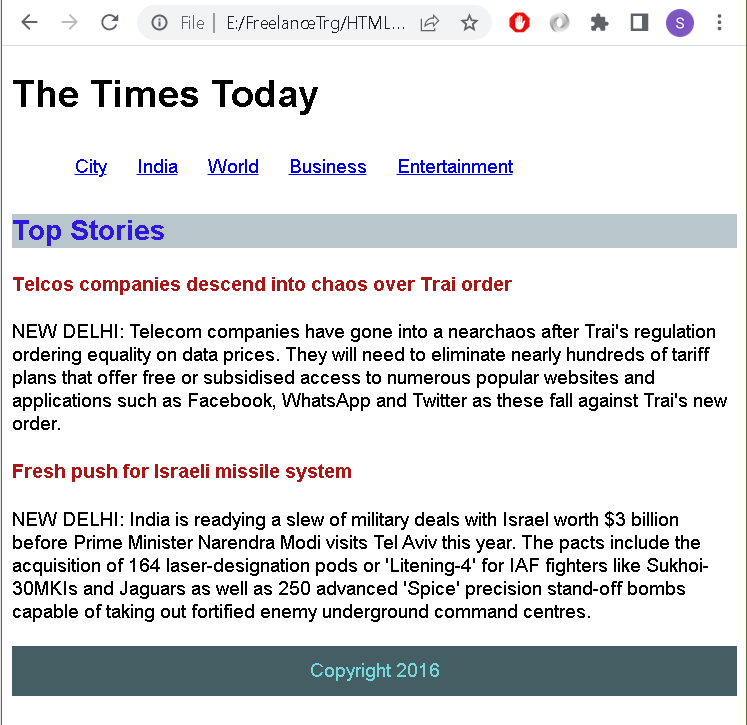
<img src="fruits.png" alt="Picture of the fave fruits" height="100" width="100">

<figcaption>Picture of the fave fruits</figcaption>

</figure>



sectionarticle-eg



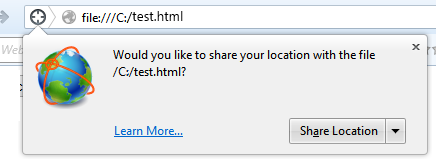
## Geo Location API

#### geolocation is best described as the determination of the geographic position of a person, place, or thing

* Geolocation API is used to locate a user's position
* It also keeps the track of as they move around, always with the user's consent
* The API is device-agnostic; it doesn't care how the browser determines location, so long as clients can request and receive location data in a standard way
  + The underlying mechanism might be via GPS, wifi, or simply asking the user to enter their location manually
  + Since any of these lookups is going to take some time, the API is asynchronous; you pass it a callback method

whenever you request a location

* Since the nature of the API exposes the user’s location, it could compromise their privacy.
* So the user’s permission to attempt to obtain the geolocation information must be sought before proceeding



## Using the Geolocation API

* Test for the presence of the geolocation object:

if (navigator.geolocation) // check for Geolocation support

console.log('Geolocation is supported!');

else console.log('Geolocation is not supported for this Browser/OS version yet.');

* Obtain the geolocation object

var geolocation = navigator.geolocation;

* Geolocation Methods
* getCurrentPosition() : retrieves the current geographic location of the user.
* watchPosition() : retrieves periodic updates about the current geographic location of the device.
* clearWatch() : cancels an ongoing watchPosition call.

function getLocation() {

var geolocation = navigator.geolocation; geolocation.getCurrentPosition(showLocation, errorHandler);

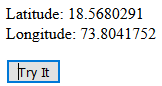
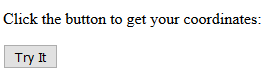
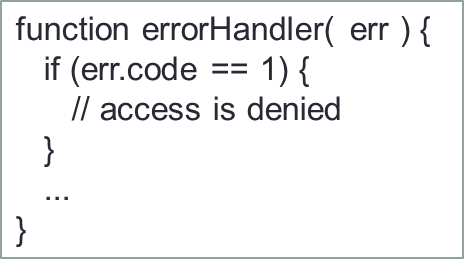
}

showLocation and errorHandler are callback methods which would be used to get actual position

#### getCurrentPosition() method is called asynchronously with an object **Position** which stores the complete location information.

* + - The **Position** object specifies the current geographic location of the device

We need to catch any error and handle it gracefully



function showLocation( position ) {

var latitude = position.coords.latitude; var longitude = position.coords.longitude;

...

}

<body>

<p id="demo">Click the button to get your coordinates:</p>

<button onclick="getLocation()">Try It</button>

<script>

var x=document.getElementById("demo"); function getLocation() {

if (navigator.geolocation) navigator.geolocation.getCurrentPosition(showPosition);

else{

x.innerHTML="Geolocation is not supported by this browser.";

}

function showPosition(position) {

x.innerHTML = "Latitude: " + position.coords.latitude +

"<br />Longitude: " + position.coords.longitude;

}

</script>

# (CSS) Cascading Style Sheets

## What is CSS?

#### Cascading Styles Sheets - a way to style and present HTML.

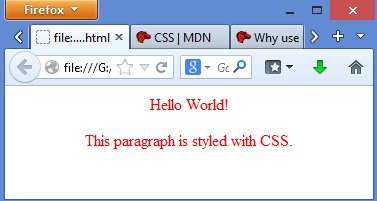
* + HTML deals with content & structure, stylesheet deals with formatting & presentation of that document.
  + Allows to control the style and layout of multiple Web pages all at once.

#### Why CSS?

* + saves time
  + Pages load faster
  + Easy maintenance
  + Superior styles to HTML
* A CSS rule has two parts: a selector, and one or more declarations:
* "HTML tag" { "CSS Property" : "Value" ; }
* The selector is normally the HTML element you want to style.
* Example:

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

h3{ color: white; }



<head>

<style>

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

</style>

</head>

<body>

<p>Hello World!</p>

<p>This paragraph is styled with CSS.</p>

</body>

## Three Ways to Insert CSS

* Embedded Style Sheets
  + Style defined between <STYLE>..</STYLE> tags. <STYLE> tags appear either in <HEAD> section or between </HEAD> and <BODY> tags.
* Linked Style Sheets
  + Separate files (extn .CSS) that are linked to a page with the <LINK> tag. Are referenced with a URL. Placing a single <LINK> tag within the <HEAD> tags links the page that needs these styles.

<head>

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

</head>

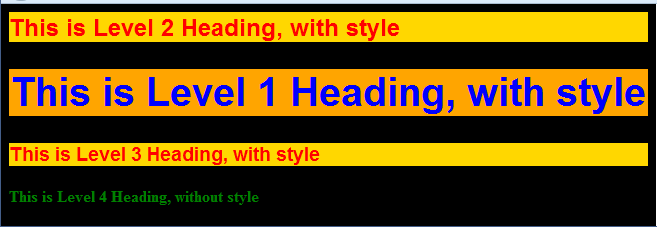
* Inline Style Sheets
  + Only applies to the tag contents that contain it. Used to control a single tag element. Tag inherits

style from its parent. Egs.

* + <p style="color:sienna;margin-left:20px">This is a paragraph.</p>
  + <p style="background: blue; color: white;">A new background and font color with inline CSS</p>

## Demo: Link Style Sheet

body { background: black; color:green



<html>

<head>

**<link** **rel=stylesheet** **href=“linked\_ex.css” type=“text/css”>**

</head>

<body>

<h2>This is Level 2 Heading, with style</h2>

<h1>This is Level 1 Heading, with style</h1>

<h3>This is Level 3 Heading, with style</h3>

<h4>This is Level 4 Heading, without style</h4>

</body>

</html>

}

h1 { background: orange; font-family: Arial, Impact; color: blue;

font-size:30pt; text-align: center

}

h2, h3 { background: gold;

font-family: Arial, Impact; color:red }

linked\_ex.css

## Inline Style Sheet

* All style attribute are specified in the tag it self. It gives desired effect on that tag only.

Doesn’t affect any other HTML tag.

<body style="background: white; color:green">

<h2 style="background: gold; font-family: Arial, Impact; color:red"> This is Level 2 Heading, with style</h2>

<h1 style="background: orange; font-family: Arial, Impact; color: blue;font-size:30pt; text-align:

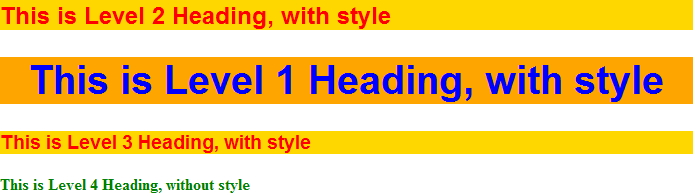
center">This is Level 1 Heading, with style</h1>

<h3 style="background: gold; font-family: Arial, Impact;color:red"> This is Level 3 Heading, with style</h3>

<h4>This is Level 4 Heading, without style</h4>

<h1>This is again Level 1 heading with default styles</h1>

</body>



## Multiple Style Sheets

#### If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet.

* + For example, an external style sheet has these properties for the h3 selector, & an internal style sheet

has these:

H3 { text-align:right; font-size:20pt;

}

H3 { color:red;

text-align:left; font-size:8pt;

}

* + If the page with the internal style sheet also links to the external style sheet the properties for h3 will be:
  + color:red;

The color is inherited from the external style sheet and the text- alignment and the font-size is replaced by the internal style sheet.

text-align:right; font-size:20pt;

* Grouping Selectors
  + In style sheets there are often elements with the same style.
    - H1 { color:green; }
    - h2 { color:green; }
    - P { color:green; }
  + To minimize the code, you can group selectors by separating each selector with a comma. Example
    - h1,h2,p { color:green; }

## Types of selectors

#### HTML selectors <tag>

* + Used to define styles associated to HTML tags. – already seen!!!!

#### Class selectors (.)

* + Used to define styles that can be used without redefining plain HTML tags.

#### ID selectors (#)

* + Used to define styles relating to objects with a unique id

<head>

<style> #para1 {

text-align:center; color:red;

}

</style>

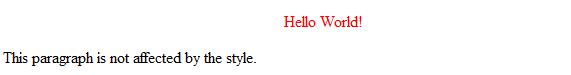
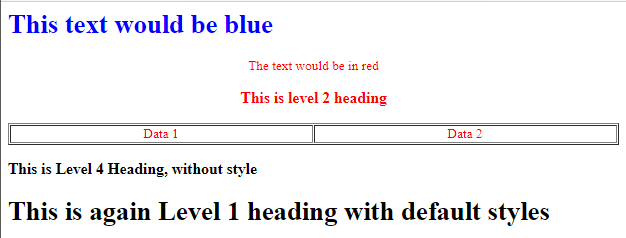
</head>

<body>

<p id="para1">Hello World!</p>

<head>

<style>



H1.myClass {color: blue}

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

</style>

<body style="background: white; color:green">

<H1 class="myClass">This text would be blue</H1>

<p class="myOtherClass">The text would be in red</P>

<H3 class="myOtherClass">This is level 2 heading</H3>

<table class=myotherClass border width=100%>

<td>Data 1</td><td>Data 2</td>

</table>

<h3>This is Level 4 Heading, without style</h3>

<h1>This is again Level 1 heading with default styles</h1>

<p>This paragraph is not affected by the style.</p>

</body>

51

## class Selector

#### The class selector is used to specify a style for a group of elements.

* + The class selector uses the HTML class attribute, and is defined with a ".“
  + Can also affect only specific HTML elements. Ex. all p elements with class="center" will be center-

aligned (see below)

|  |  |  |
| --- | --- | --- |
| <head> | |  |
| <style>  .center { text-align:center; }  </style>  </head> |  |  |
| <body>  <h1 class="center">Center-aligned heading</h1>  <p class="center">Center-aligned paragraph.</p>  </body> | |  |

|  |  |  |
| --- | --- | --- |
| <style>  p.center { text-align:center; }  </style> |  |  |
|  |  |
| </head>  <body>  <h1 class="center">This heading will not be affected</h1>  <p class="center">This paragraph will be center-aligned.</p> | |  |

## CSS Box Model

#### All HTML elements can be considered as boxes.

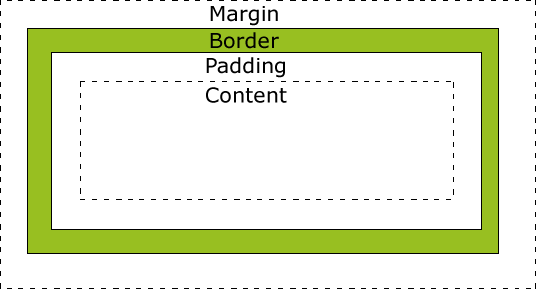
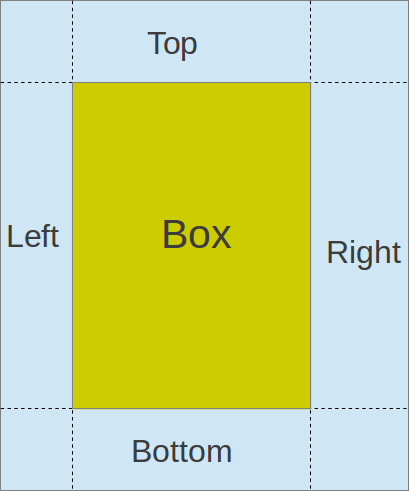
* The CSS box model is essentially a box that wraps around HTML elements, and it consists of:
  + Margin - Clears an area around the border. The margin does not have a background color, it is

completely transparent

* + Border - A border that goes around the padding and content. The border is affected by the background color of the box
  + Padding - Clears an area around the content. The padding is affected by the background color of the box
  + Content - The content of the box, where text and images appear

<h1>Header 1</h1>

<p> paragraph 1</p>



## CSS Border

#### The border property is a shorthand for the following individual border properties: border- width, border-style (required), border-color



p { border: 5px solid red; }



<style type="text/css">

.box {

width: 100px; height: 100px; border-color: Blue; border-width: 2px; border-style: solid;

}

</style>

<div class="box"> Hello, world!

</div>

<div class="box" style="border-style: dashed;">Dashed</div>

<div class="box" style="border-style: dotted;">Dotted</div>

<div class="box" style="border-style: double;">Double</div>

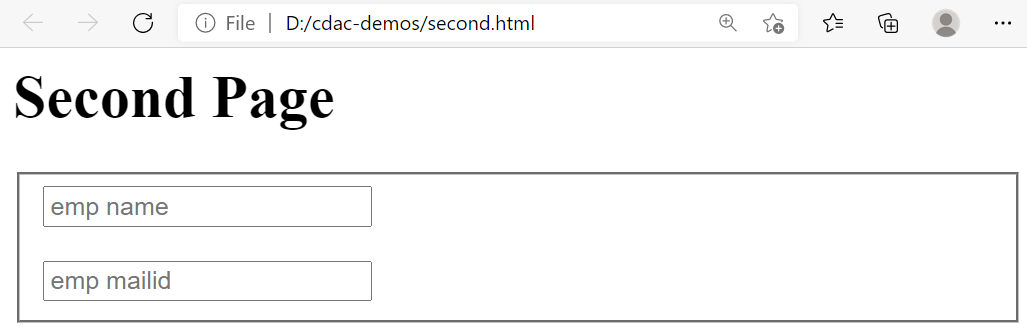
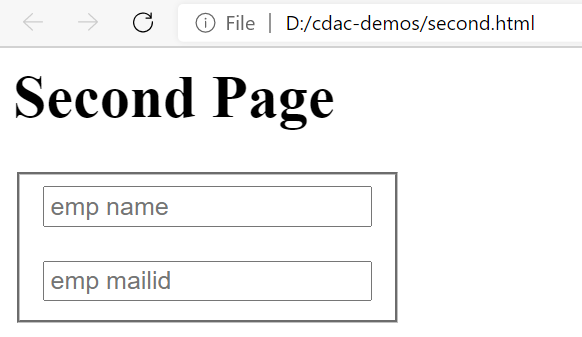
<div class="box" style="border-style: groove;">Groove</div>

<div class="box" style="border-style: inset;">Inset</div>

<div class="box" style="border-style: outset;">Outset</div>

<div class="box" style="border-style: ridge;">Ridge</div>

<div class="box" style="border-style: solid;">Solid</div>



<form>

<fieldset>

<input type="text" placeholder="emp name"><br><br>

<input type="text" placeholder="emp mailid"><br>

</fieldset>

</form>

<html>

<head>

<style>

fieldset{width:100px}

</style>

</head>

<body>

<h1> Second Page</h1>

<form>

<fieldset>

<input type="text" placeholder="emp name">

<input type="text" placeholder="emp mailid">

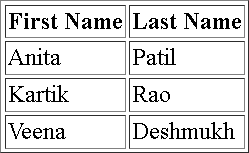
</fieldset>

</form>

</body>

</html>

## Co lapsing borders on table



table, th, td {

border: 1px solid black; border-collapse: collapse;

}

<table border="1">

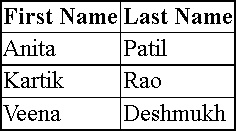
<tr><th>First Name</th><th>Last Name</th></tr>

<tr><td>Anita</td><td>Patil</td></tr>

<tr><td>Kartik</td><td>Rao</td></tr>

<tr><td>Veena</td><td>Deshmukh</td></tr>

</table>



<head>

<style>

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

</style>

</head>

<body>

<table border="1">

<tr><th>First Name</th><th>Last Name</th></tr>

<tr><td>Anita</td><td>Patil</td></tr>

<tr><td>Kartik</td><td>Rao</td></tr>

<tr><td>Veena</td><td>Deshmukh</td></tr>

</table>

CSS3 border

* CSS 3 defines “border radius”, giving developers the possibility to make rounded

corners on their elements.

<style> #rcorners1 {

border-radius: 25px; background: #8AC007; padding: 20px;

width: 100px;

height: 100px;

}

#rcorners2 {

border-radius: 25px; border: 2px solid #8AC007; padding: 20px;

width: 100px; height: 100px;

}



#rcorners3 {

border-radius: 25px; background: url(paper.gif); background-position: left top; background-repeat: repeat; padding: 20px;

width: 100px; height: 100px;

}

</style>

<p id="rcorners1">Rounded corners!</p>

<p id="rcorners2">Rounded corners!</p>

<p id="rcorners3">Rounded corners!</p>

# CSS Styling

## CSS Background

* CSS background properties are used to define the background effects of an element.
* CSS properties used for background effects:

|  |  |
| --- | --- |
| [background-color](http://www.w3schools.com/cssref/pr_background-color.asp) | Sets the background color of an element |
| [background-image](http://www.w3schools.com/cssref/pr_background-image.asp) | Sets the background image for an element |
| [background-position](http://www.w3schools.com/cssref/pr_background-position.asp) | Sets the starting position of a background image |
| [background-repeat](http://www.w3schools.com/cssref/pr_background-repeat.asp) | Sets how a background image will be repeated |

* Example:

*With* *CSS,* *a* *color* *is* *specified* *by:*

* *a* *HEX* *value* *-* *like* *"#ff0000"*
* *an* *RGB* *value* *-* *like* *"rgb(255,0,0)"*
* *a* *color* *name* *-* *like* *"red"*
  + div {background-color:#b0c4de;}
  + body {background-image:url('paper.gif');}
  + body {background-image:url('gradient2.png');background-repeat:repeat-x;}
  + body {background-image:url('img\_tree.png'); background-repeat:no-repeat;

background-position:right top; }

#### The background-repeat property sets if/how a background image will be repeated.

* + By default, (repeat) : a [background-image](https://www.w3schools.com/cssref/pr_background-image.asp) is repeated both vertically and horizontally.
  + no-repeat : The background-image is not repeated. The image will only be shown once

## Demo : CSS Background

body {

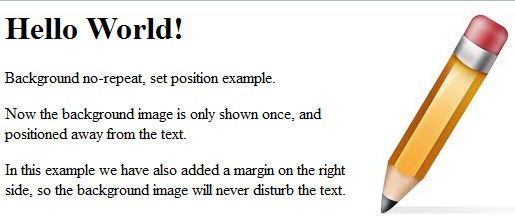
background-image: url("img\_tree.gif"), url("img\_flwr.gif"); background-color: #cccccc;

}

body {

background: #00ff00 url("smiley.gif") no-repeat fixed center;

}



<html>

<head><style> body {

background-image:url('img\_tree.png'); background-repeat:no-repeat; background-position:right top;

margin-right:200px;

}

</style> </head>

<body>

<h1>Hello World!</h1>

<p>Background no-repeat, set position example.</p>

<p>Now the background image is only shown once, and positioned away from the text.</p>

<p>In this example we have also added a margin on the right side, so the background image will never disturb the text.</p>

</body></html>

CSS Text

* CSS Text Properties

|  |  |
| --- | --- |
| [color](http://www.w3schools.com/cssref/pr_text_color.asp) | Sets the color of text |
| [direction](http://www.w3schools.com/cssref/pr_text_direction.asp) | Specifies the text direction/writing direction |
| [letter-spacing](http://www.w3schools.com/cssref/pr_text_letter-spacing.asp) | Increases or decreases the space between characters in a text |
| [text-align](http://www.w3schools.com/cssref/pr_text_text-align.asp) | Specifies the horizontal alignment of text |
| [text-decoration](http://www.w3schools.com/cssref/pr_text_text-decoration.asp) | Specifies the decoration added to text |
| [text-indent](http://www.w3schools.com/cssref/pr_text_text-indent.asp) | Specifies the indentation of the first line in a text-block |
| [text-shadow](http://www.w3schools.com/cssref/css3_pr_text-shadow.asp) | Specifies the shadow effect added to text |
| [text-transform](http://www.w3schools.com/cssref/pr_text_text-transform.asp) | Controls the capitalization of text |
| [white-space](http://www.w3schools.com/cssref/pr_text_white-space.asp) | Specifies how white-space inside an element is handled |
| [word-spacing](http://www.w3schools.com/cssref/pr_text_word-spacing.asp) | Increases or decreases the space between words in a text |

<style>

h1 {text-decoration:overline;}

h2 {text-decoration:line-through;} h3 {text-decoration:underline;}

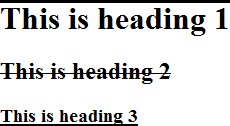
</style>

<body>

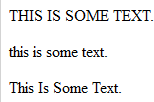
<h1>This is heading 1</h1>

<h2>This is heading 2</h2>

<h3>This is heading 3</h3>



## Demo :



<style>

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

</style>

<body>

<p class="uppercase">This is some text.</p>

<p class="lowercase">This is some text.</p>

<p class="capitalize">This is some text.</p>

</body>

<head> <style>



h1 {text-align:center;color:#00ff00;} p.date {text-align:right;}

p.main {text-align:justify;} p.ex {color:rgb(0,0,255);} p.indent {text-indent:50px;}

</style> </head>

<body>

<h1>Hello World!</h1>

<p class="date"> Sep 2013</p>

<p class="main indent">The CSS property text-align corresponds to the attribute align used in old versions of HTML. Text can either be aligned to the left, to the right or centred. In addition to this, the value justify will stretch each line so that both the right and left margins are straight. You know this layout from for example newspapers and magazines. </p>

<p class="ex">The property text-decoration makes it is possible to add different "decorations" or "effects" to text. </p>

</body>

## CSS : Styling Fonts

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

|  |  |
| --- | --- |
| [font-family](http://www.w3schools.com/cssref/pr_font_font-family.asp) | Specifies the font family for text |
| [font-size](http://www.w3schools.com/cssref/pr_font_font-size.asp) | Specifies the font size of text |
| [font-style](http://www.w3schools.com/cssref/pr_font_font-style.asp) | Specifies the font style for text |
| [font-variant](http://www.w3schools.com/cssref/pr_font_font-variant.asp) | Specifies if a text should be displayed in a small-caps font |
| [font-weight](http://www.w3schools.com/cssref/pr_font_weight.asp) | Specifies the weight of a font |

* Example:
  + p.normal {font-weight:normal;}
  + p{font-family:"Times New Roman", Times;}
  + p.italic {font-style:italic;}
  + h1 {font-size:40px;}
  + p.small { font-variant:small-caps; }

# JAVASCRIPT

## Overview

#### JavaScript is Netscape's cross-platform, object-based scripting language

* + JavaScript code is embedded into HTML pages
  + It is a lightweight programming language
  + Client-side JavaScript extends the core language by supplying objects to control a browser and its Document Object Model

#### Why use Javascript?

* + Provides HTML designers a programming tool :
  + Puts dynamic text into an HTML page
  + Reacts to events
  + Reads and writes to HTML elements :
  + Can be used to perform Client side validation

|  |  |
| --- | --- |
| * The <SCRIPT> tag | <SCRIPT>  JavaScript statements …  </SCRIPT> |
| <html> |
| <head> </head>  <body>  <script type="text/javascript"> document.write("<H1>Hello World!</H1>"); alert(“some message”);  console.log(“some message”);  </script>  </body></html> | |

## Embedding JavaScript in HTML

<html>

<head>

<script type="text/javascript"> function message() {

alert(“Hello World")

}

</script>

</head>

<body onload="message()">

</body>

</html>

#### Where to Write JavaScript?

* + Head Section
  + Body Section
  + External File

<html>

<head></head>

<body >

<script language=“javascript">

document.write(“Hello World!“)

</script>

</body>

</html>

|  |  |  |
| --- | --- | --- |
| <head>  <script **src="common.js"**> | |  |
| <!– no javascript statements   </ script>  </head> | **//common.js** **file** **contents**  var msg  msg=“<h1>in external file</h1>" | |
| <body>  <script>  document.write("display value of a variable”+msg)  </script>  </body> | |  |

## Data Types in JavaScript

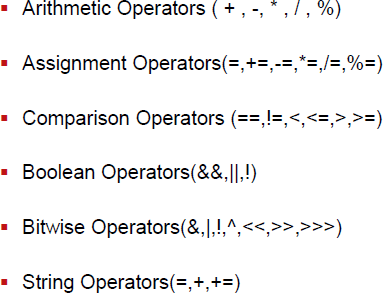
#### JavaScript is a free-form language. Need not declare all variables, classes, and methods

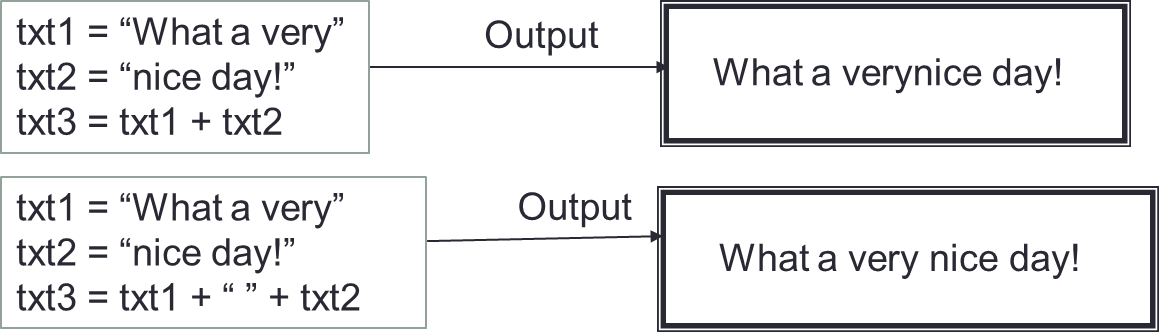
* Variables in JavaScript can be of type:
  + Number (4.156, 39)
  + String (“This is JavaScript”)
  + Boolean (true or false)
  + Null (null)  usually used to indicate the absence of a value

#### Defining variables. var variableName = value

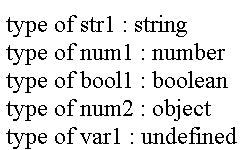
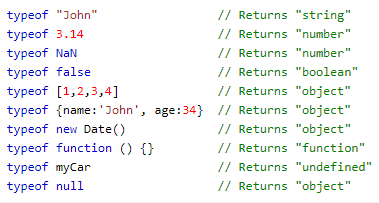
* JavaScript variables are said to be un-typed or loosely typed
  + letters of the alphabet, digits 0-9 and the underscore (\_) character and is case-sensitive.
  + Cannot include spaces or any other punctuation characters.
  + First character of name must be either a letter or the underscore character.
  + No official limit on the length of a variable name, but must fit within a line.

Javascript operators:

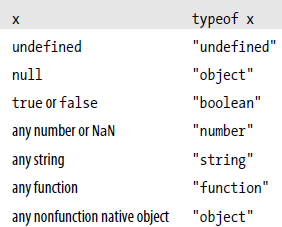




Typeof Operator



|  |  |  |
| --- | --- | --- |
| typeof | undefinedvariable | “undefined” |
| typeof | 33 | “number” |
| typeof | “abcdef” | “string” |
| typeof | true | “boolean” |
| typeof | null | “object” |



<SCRIPT LANGUAGE="JavaScript">

var num1=20 var str1="abc" var bool1=true var num2=null var var1;

document.write("type of str1 : "+typeof(str1)+"<BR>") document.write("type of num1 : "+typeof(num1)+"<BR>") document.write("type of bool1 : "+typeof(bool1)+"<BR>") document.write("type of num2 : "+typeof(num2)+"<BR>") document.write("type of var1 : "+typeof(var1)+"<BR>")

</SCRIPT>

## Control Structures and Loops

#### JavaScript supports the usual control structures:

* + the conditionals:

if(a>10) { document.write(“Greater than 10”)

} else {

document.write(“Less than 10”)

}

if(condition) { statement 1

} else {

statement 2

}

* + - if,
    - if...else
    - If … else if … else
    - Switch

while(condition) { statements

}

for(var i=0;i<10;i++)

{

document.write(“Hello”);

}

for( [initial expression;][condition;][increment expression] ) { statements

}

switch (variable) { case outcome1 :{

//stmts for outcome 1 break; }

case outcome2 :{

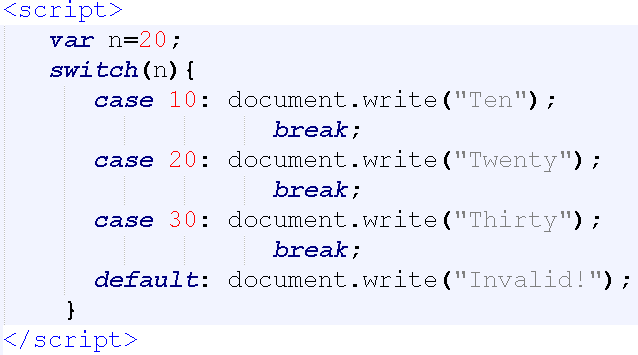
//stmts outcome 2

break; } default: {

//none of the outcomes is chosen

}

}



document.write( (a>10) ? “Greater than 10” : “Less than 10” );

* + iterations:
    - for
    - while

while(i<10) { document.write(“Hello”); i++;

}

## JavaScript Functions

function myFunction (arg1, arg2, arg3) {

statements

return

}

function area(w1, w2, h) { var area=(w1+w2)\*h/2; alert(area+" sq ft");

}

area(2,3,7); //calling the function

function diameter(radius){ return radius \* 2;

}

var d=diameter(5); //calling the function

Calling the function : myFunction( “abc”, “xyz”, 4 ) myFunction()

* **Function** **expressions** **-** functions are assigned to variables

var myFunction = function() { statements

}

var area = function (radius) {

return Math.PI \* radius \* radius;

};

alert(area(5)); // => 78.5

##### Global and Local Variables

* Variables that exist only inside a function are called Local variables - they can’t be changed by main code or other functions
* Within the body of a function, a local variable takes precedence over a global variable with the same name.

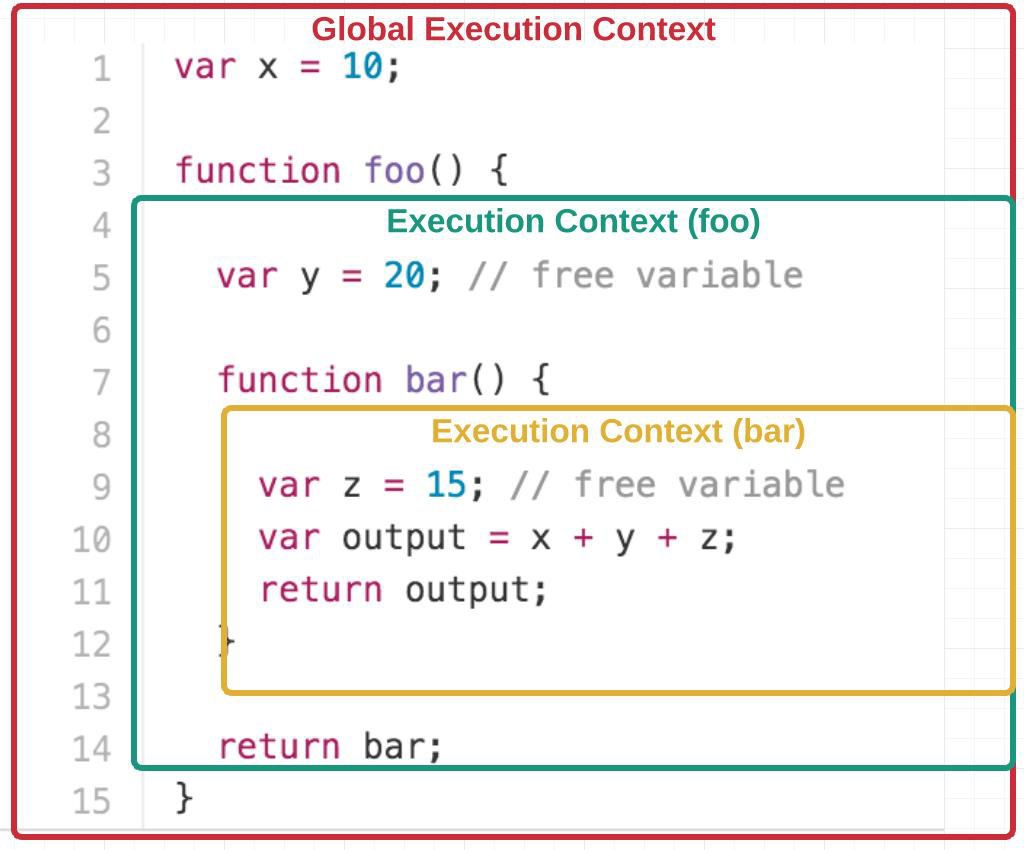
|  |  |  |
| --- | --- | --- |
|  |  |  |
| <script language=“Javascript”> | Global Variable |  |
| var companyName=“TechnoFlo”  function f(){ Local Variable  var empName=“Henry”  document.write(“Welcome to ”+companyName+ ”, “ +empName)  }  </script> | | |

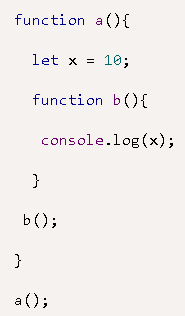
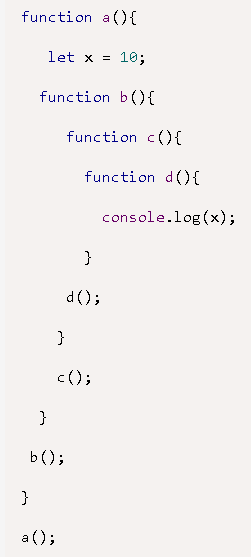
* Variables that exist throughout the script are called Global variables - Their values can be changed anytime in the code and even by other functions

## Closures

#### The closure is an inner function which always has access to the variables and parameters of its outer function, even when the outer function has returned.

* + - The closure has three scope chains: it has access to its own scope (variables defined between its curly brackets), it has access to the outer function’s variables, and it has access to the global variables.





Calling inner function

* + - * Focus on lines 10 and 11.

1. function sayHello() {

2.

3.

4.

var say = function() {

console.log(hello);

}

5.

6.

7.

8.

9.

// Local variable that goes into closure

var hello = 'Hello, world!'; return say;

}

1. var sayHelloClosure = sayHello();
2. sayHelloClosure(); // ‘Hello, world!’
   * + - At line number 10 we are done with the execution of **function** **sayHello()** and the entire body of **function** **say()** is returned and stored in **var** **sayHelloClosure**, due to the line 7 **return** **say**.

###### The return statement does not execute the inner function – function is executed only when followed by (), but rather the return statement returns the reference to the function as a function in JavaScript is also an object.

* + - * We can access the variable hello which is defined in **function** **sayHello()** through **function**

**say()**

* + - * This is closure in action - that is inner function (say()) can have access to the outer function variables as well as all the global variables.
      * closure is created when a child function keep the environment of the parent scope even after the parent function has already executed
        1. <html>
        2. <body>
        3. <script>
        4. function outer() {
        5. var i = 1;
        6. function inner(){
        7. return i++;

8. }

9. return inner;

10. }

11. var getInner = outer(); 12.

1. console.log(getInner());
2. console.log(getInner());
3. console.log(getInner());
4. </script>
5. </body>
6. </html>
   * Focus on lines 11 to 15.
   * At line number 11 we are done with the execution of **function** **outer()** and the entire body of **function** **inner()** is returned and stored in **var** **get\_func\_inner**, due to the line



9 **return** **inner**.

###### The return statement does not execute the inner function – function is executed only when followed by (), but rather the return statement returns the reference to the function as a function in JavaScript is also an object.

**output**

we used a parameter function rather than a default one. Note even when we are done with the execution of **outer(5)** we can access the **outer\_arg** variable from the inner function. And on execution of inner function produce the summation



<script>

console.log("--------example-3 ")

function outer(outer\_arg) {

function inner(inner\_arg) { return outer\_arg + inner\_arg;

}

return inner;

}

var getInner = outer(5);

console.log(getInner(4)); console.log(getInner(3));

</script>

of **outer\_arg** and **inner\_arg** as desired.

output

#### isFinite: evaluates an argument to determine if it is a finite number.

* If needed, the parameter is first converted to a number.

isFinite (number) //where number is the number to evaluate

var a = isFinite(123) + "<br>"; //true var b = isFinite(-1.23) + "<br>"; //true var d = isFinite(0) + "<br>"; //true var e = isFinite("123") + "<br>"; //true var f = isFinite("Hello") + "<br>"; //false

#### isNaN : Evaluates an argument to determine if it is “NaN” (not a number)

isNaN(0) isNaN('123')

isNaN('Hello')

//false

//false

//true

isNaN('2005/12/12') //true

* Parseint and parsefloat
  + Returns a numeric value for string argument.
  + parseInt (str)

parseInt("3 blind mice") parseFloat(" 3.14 meters")

// => 3

// => 3.14

* + parseFloat (str)

#### **const** : from JS 1.5 onwards.- to define constants

* + Eg :

const myBirthday = '18.04.1982';

myBirthday = '01.01.2001';

// error, can't reassign the constant!

const LANGUAGES = ['Js', 'Ruby', 'Python', 'Go']; LANGUAGES = "Javascript"; // shows error.

LANGUAGES.push('Java’); // Works fine.

console.log(LANGUAGES); // ['Js', 'Ruby', 'Python', 'Go', 'Java']

#### **let** : to define block-scoped variables; can be used in four ways:

* + as a variable declaration like var; in a for or for/in loop, as a substitute for var;
  + as a block statement, to define new variables and explicitly delimit their scope
  + to define variables that are scoped to a single expression.

let a = 50; let b = 100; if (true) {

let a = 60; var c = 10;

console.log(a/c); // 6

console.log(b/c); // 10

}

console.log(c); // 10

console.log(a); // 50

* + Eg : let message = 'Hello!';
  + Eg : let user = 'John', age = 25, message = 'Hello';

if (true) {

let a = 40; console.log(a); //40

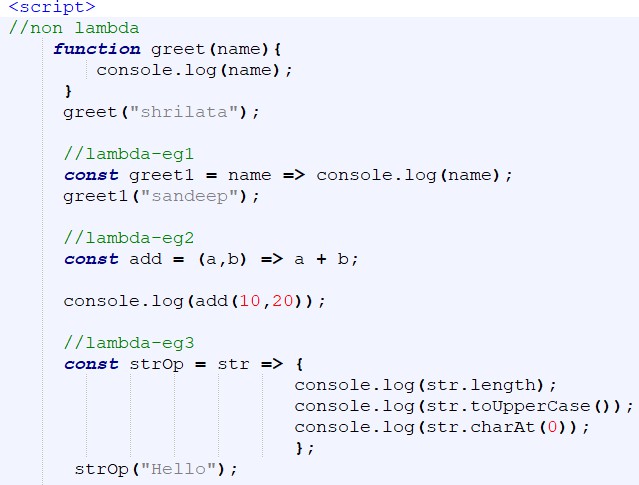
}

console.log(a); // undefined

#### Arrow functions : allows you to create functions in a cleaner way compared to regular functions

* + is a compact alternative to a traditional [function expression](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/function), but is limited and can't be used in all

situations.



# Predefined Core Objects

**79**

## String Objects

#### Creating a string object:

* + var myString = new String(“characters”)
  + var myString = “fred”

•

•

•

•

“Lincoln”.length // result = 7 “Four score”.length // result = 10 “One\ntwo”.length // result = 7 “”.length // result = 0

#### Properties of a string object:

* + length: returns the number of characters in a string.

#### String Object methods:

* + charAt(index) : returns the character at a specified position.
  + Eg : var str = "Hello world!";
    - str.charAt(0); //returns H
    - str.charAt(str.length-1)); //returns !
  + concat() : joins two or more strings
  + stringObject.concat(stringX,stringX,...,stringX)
  + Eg: var str1="Hello "; var str2="world!";

document.write(str1.concat(str2));

#### indexOf () : returns the position of the first occurrence of a specified string value in a string.

* + index values start their count with 0.
  + If no match occurs within the main string, the returned value is -1.
  + string.indexOf( searchString [, startIndex])

Eg : var str="Hello world, welcome";

str.indexOf("Hello"); //returns 0

str.indexOf("wor")); //returns 6

str.indexOf("e",5); //returns 14

#### toLowerCase() / toUpperCase()

Eg: var str="Hello World!"; str.toLowerCase() //returns hello world str.toUpperCase() //returns HELLO WORLD

* slice( startIndex [, endIndex])
  + Extracts a part of a string and returns the extracted part in a new string

Eg : var str=“Hello World"; str.slice(6) //returns World str.slice(0,1) //returns H

#### split(“delimiterCharacter”[, limitInteger]) - Splits a string into array of strings

* + string.split(“delimiterCharacter”[, limitInteger])

Output :

zero one two three four

var str = "zero one two three four"; var arr = str.split(" ");

for(i = 0; i < str.length; i++){ document.write(“<br>” + arr[i]); }

var myString = “Anderson,Smith,Johnson,Washington”

var myArray = myString.split(“,”)

var itemCount = myArray.length // result: 4

* + Complete Example:

var s = "hello, world" s.charAt(0) s.charAt(s.length-1) s.substring(1,4) s.slice(1,4)

s.slice(-3)

s.indexOf("l") s.lastIndexOf("l") s.indexOf("l", 3)

s.split(", ")

s.replace("h", "H") s.toUpperCase()

// Start with some text.

// => "h": the first character.

// => "d": the last character.

// => "ell": the 2nd, 3rd and 4th characters.

// => "ell": same thing

// => "rld": last 3 characters

// => 2: position of first letter l.

// => 10: position of last letter l.

// => 3: position of first "l" at or after 3

// => ["hello", "world"] split into substrings

// => "Hello, world": replaces all instances

// => "HELLO, WORLD"

## String property : Prototype

#### A prototype is a property or method that becomes a part of every new object created after the

prototype items have been added.

* + Sometimes you want to add new properties/methods to all existing objects of a given type.
  + For strings, as an example, you may want to define a new method for converting a string into a new type

of HTML font tag not already defined by JavaScript’s string object.

* + A function definition (makeItHot()) accumulates string data to be returned to the object when the function

is invoked as the object’s method.

* + The this keyword extracts the object making the call, which you convert to a string for concatenation with the rest of the strings to be returned.

#### prototype

* + Allows you to add properties and methods to an object.
  + Syntax : object.prototype.name=value

function makeItHot() {

return "<FONT COLOR='red'>" + this.toString() + "</FONT>“

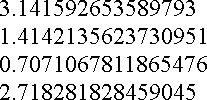
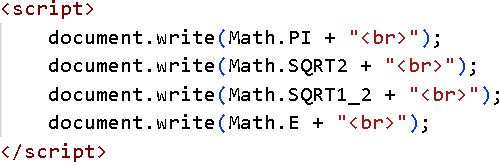
}

String.prototype.hot = makeItHot

document.write("<H1>This site is on " + "FIRE".hot() + "!!</H1>")

Math Properties

|  |  |  |
| --- | --- | --- |
| **Property** | **Value** | **Description** |
| Math.E | 2.718281828459045091 | Euler’s constant |
| Math.LN | 0.6931471805599452862 | Natural log of 2 |
| Math.LN10 | 2.302585092994045901 | Natural log of 10 |
| Math.LOG2E | 1.442695040888963387 | Log base-2 of E |
| Math.LOG10E | 0.4342944819032518167 | Log base-10 of E |
| Math.PI | 3.141592653589793116 | PI |
| Math.SQRT1\_2 | 0.7071067811865475727 | Square root of 0.5 |
| Math.SQRT2 | 1.414213562373095145 | Square root of 2 |



hods)

|  |  |  |
| --- | --- | --- |
| **Method** **syntax** | **Returns** | Math Objects (Met |
| Math.abs*(*val) | Absolute value of *val* |
|  |
| Math.acos*(*val) | Arc cosine (in radians) of *val* | |
| Math.asin*(*val) | Arc sine (in radians) of *val* | |
| Math.atan*(*val) | Arc tangent (in radians) of *val* | |
| Math.atan2*(*val1, val2) | Angle of polar coordinates *x* and *y* | |
| Math.ceil*(*val) | Returns the value of x rounded up to its nearest integer | |
| Math.cos*(*val) | Cosine of *val* | |
| Math.exp*(*val) | Euler’s constant to the power of *val* | |
| Math.floor*(*val) | Next integer less than or equal to *val* | |
| Math.max*(*val1, val2) | The greater of *val1* or *val2* | |
| Math.min*(*val1, val2) | The lesser of *val1* or *val2* | |
| Math.pow*(*val1, val2) | *Val1* to the *val2* power | |
| Math.random() | Random number between 0 and 1 | |
| Math.round*(*val) | returns the nearest integer: N+1 when *val* >= n.5; otherwise N | |
| Math.sin*(*val) | Sine (in radians) of *val* | |
| Math.sqrt*(*val) | Square root of *val* | |
| Math.tan*(*val) | Tangent (in radians) of *val* | |
| Math.log(val) | Natural logarithm (base e) of val | |

Date

#### Date object allows the handling of date and time information.

* + All dates are in milliseconds from January 1, 1970, 00:00:00.
  + Dates before 1970 are invalid dates.

#### There are different ways to define a new instance of the date object:

var d = new Date()

//Current date

var d = new Date(milliseconds) var d = new Date(dateString)

var d = new Date(year, month, day, hours, minutes, seconds, milliseconds)

<script>

var d=new Date();



document.write(d);

</script>

var d = new Date(86400000);

var d = new Date(99,5,24,11,33,30,0);

## Date Object - Methods

* getDate( ) Date of the month (1 - 31)
* getDay( ) Day of the week (0 - 6, 0-Sunday)
* getMonth( ) The month (0 - 11, 0 - Jan.)
* getFullYear( ) The year (4 digits)
* getHours( ) Hour of the day (0 - 23)
* getMinutes( ) Minutes (0 - 59)
* getSeconds( ) Seconds (0 - 59)
* getTime( ) Milliseconds since 1/1/1970
* getTimezoneOffset( ) Offset between local time and GMT

#### setDate(dayValue) 1-31

#### setHours(hoursValue) 0-23

#### setMinutes(minutesValue) 0-59

#### setMonth(monthValue) 0-11

#### setSeconds(secondsValue) 0-59

* setTime(timeValue) >=0
* setYear(yearValue) >=1970
* valueOf() returns number of millisecond since 1 jan 1970
* Date.now() same as valueOf()

## Array

#### An array is data structure for storing and manipulating ordered collections of data.

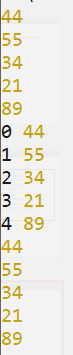
* An array can be created in several ways.
  + Eg1: Regular

var cars=new Array(); cars[0]="Spark";

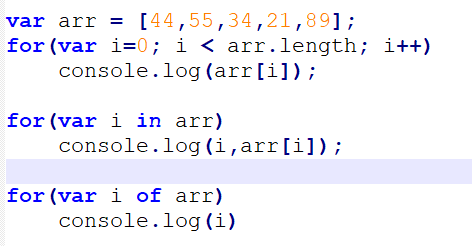
cars[1]="Volvo";

cars[2]="BMW";

* + Eg 2: Condensed: var cars=new Array("Spark","Volvo","BMW");
  + Eg 3: Literal: var cars=["Spark","Volvo","BMW"];
  + Eg 4: var matrix = [[1,2,3], [4,5,6], [7,8,9]];



* + Eg 5 : var sparseArray = [1,,,,5];
* Iterating thru array



## Array Object Methods

* + arrayObject.reverse()
  + arrayObject.slice(startIndex, [endIndex])
  + arrayObject.join(separatorString) : array contents will be joined and placed into arrayText by

using the comma separator“

* + arrayObject.push(): add one or more values to the end of an array

arrayObject.slice(startIndex [, endIndex])

//Returns: Array

var solarSys = new Array (“Mercury”,”Venus”,”Earth”,”Mars”,”Jupiter”,”Saturn”)

var nearby = solarSys.slice(1,4)

// result: new array of “Venus”, “Earth”, “Mars”



var names = ["Andrew","Monica","Catie","Jenna"]; var joined\_arr = names.join("|") console.log(joined\_arr)

console.log("joinedarr type :",typeof(joined\_arr))

arrayObject.concat(array2)

var a1 = new Array(1,2,3)

var a2 = new Array(“a”,”b”,”c”)

var a3 = a1.concat(a2)

// result: array with values 1,2,3,”a”,”b”,”c”

a = []; // Start with an empty array

a.push("zero") // Add a value at the end. a = ["zero"]

a.push("one", "two") // Add two more values. a = ["zero", "one", "two"]

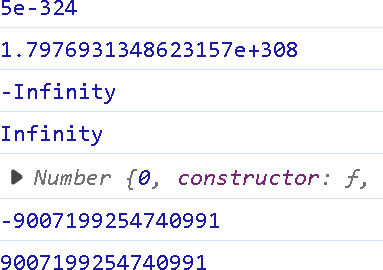
#### The **Number** object is a wrapper object allowing you to work with numerical values.

* + A Number object is created using the Number() constructor or as a literal
  + JavaScript creates Number objects when a variable is set to a number value, for example **var** **num** **=**

**255.336**;. It is seldom necessary to create Number objects explicitly.

|  |  |
| --- | --- |
| **Property** | **Description** |
| constructor | Returns the function that created JavaScript's Number prototype |
| MAX\_VALUE | Returns the largest number possible in JavaScript |
| MIN\_VALUE | Returns the smallest number possible in JavaScript |
| NEGATIVE\_INFINITY | Represents negative infinity (returned on overflow) |
| NaN | Represents a "Not-a-Number" value |
| POSITIVE\_INFINITY | Represents infinity (returned on overflow) |
| prototype | Allows you to add properties and methods to an object |
| [Number.MAX\_SAFE\_INTEGER](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/MAX_SAFE_INTEGER) | The maximum safe integer in JavaScript (253 - 1). |

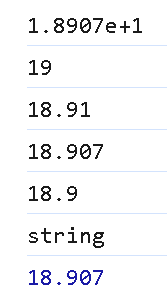
## Number Methods and properties



console.log(Number.MIN\_VALUE) console.log(Number.MAX\_VALUE) console.log(Number.NEGATIVE\_INFINITY) console.log(Number.POSITIVE\_INFINITY) console.log(Number.prototype) console.log(Number.MIN\_SAFE\_INTEGER)

console.log(Number.MAX\_SAFE\_INTEGER)

|  |  |
| --- | --- |
| **Method** | **Description** |
| isFinite() | Checks whether a value is a finite number |
| isInteger() | Checks whether a value is an integer (new in ES6) |
| isNaN() | Checks whether a value is Number |
| isSafeInteger() | Checks whether value can be safely represented in JavaScript. |
| toExponential(x) | Converts a number into an exponential notation |
| toFixed(x) | Formats a number with x numbers of digits after the decimal point |
| toPrecision(x) | Formats a number to x length |
| valueOf() | Returns the primitive value of a number |
| toString() | Converts a number to a string |
| parseFloat()/parseInt() | Same as global counterparts |

let numObj = 5.123456

t.

|  |  |
| --- | --- |
| let num = new Number('18.907');  console.log(num.toExponential()); //Converts a number into an Exponential notation. console.log(num.toFixed());  console.log(num.toFixed(2)); //Rounds up a number to x digits after the decimal.  console.log(num.toPrecision());  console.log(num.toPrecision(3)); //Rounds up a number to a length of x digits. console.log(typeof num.toString()); //toString() Returns a String value of a number | |
| object.  console.log(num.valueOf()); //Returns the primitive value of the Number objec |  |
|  |  |

console.log(numObj.toPrecision()) // logs '5.123456' console.log(numObj.toPrecision(5)) // logs '5.1235' console.log(numObj.toPrecision(2)) // logs '5.1' console.log(numObj.toPrecision(1)) // logs '5'

let numObj = 53.123456

console.log(numObj.toPrecision()) // logs '5.123456' console.log(numObj.toPrecision(5)) // logs '53.123' console.log(numObj.toPrecision(2)) // logs '53' console.log(numObj.toPrecision(1)) // logs '5e+1'

## Boolean Object

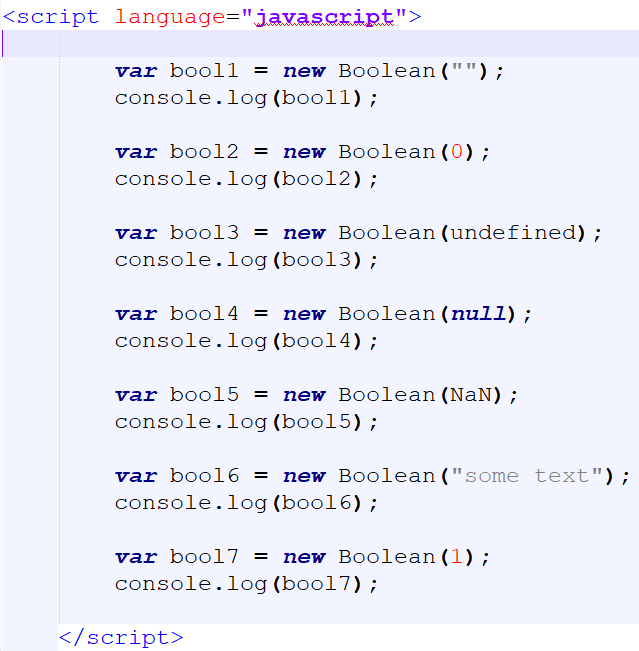
#### The Boolean object represents two values, either "true" or "false".

* The following syntax creates a boolean object. var val = new Boolean(value);
  + If value parameter is omitted or is 0, -0, null, false, NaN, undefined, or the empty string (""), the

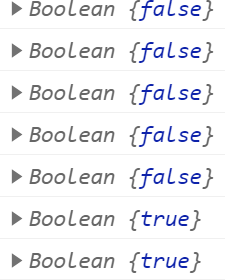
object has an initial value of false.

* + Everything else is true; including any object, an empty array ([]), or the string "false", create an object with an initial value of true.
  + Ie, Everything With a "Value" is True and Everything Without a "Value" is False

#### Boolean Methods:



* + toString() : Returns a string of either "true" or "false" depending upon the value of the object.
  + valueOf() : Returns the primitive value of the Boolean object.



## User defined Objects

### Objects can be created in several ways:

* 1. Using the Object() constructor:
  2. Using Object.create() method:
  3. Using the bracket's syntactic sugar. eg : var b = {};
  4. Using a function constructor
  5. Using class (ES6)

### Using Object Initializers

* Syntax : objName = {property1:value1, property2:value2, … }
* Eg



## Examples : Using Object Initializers



### Using function Constructors

function person(name, age) { this.name = name this.age = age

}

p1 = new person( "Ken" , 33 ) console.log(p1.age, p1.name)

function car(make, year, owner) { this.make = make

this.year = year this.owner = owner

}

c1 = new car( "Tiago", 2022, p1 ) console.log(c1.make, c1.year, c1.owner.name)

#### Accessing properties

var name = p1.name c1.make = "Baleno"

var age = c1.owner.age

console.log(name, age, c1.make) //Ken 33 Baleno

* You **cannot** add a new property to an existing object constructor
  + Eg : Car.price = null  gives error

#### The **prototype** property allows you to add new properties to object constructors:

function Person(name, age){ this.name = name this.age = age

this.display = function(){

console.log(this.name, this.age)

}

}

p1 = new Person("Soha", 30) p1.display() //Soha 30

car.prototype.model = "Tiago" car.prototype.price = 730000 console.log(c1.make, c1.model, c1.price)

//Tata Tiago 730000

Only modify your own prototypes. Never modify the prototypes of standard JavaScript objects.

c2 = new car("Maruti",2020,p1)

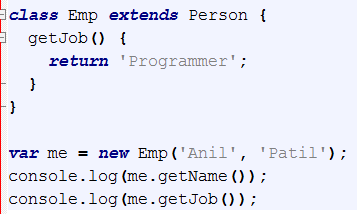
c2.model = "Suzuki"

c2.price = 500000

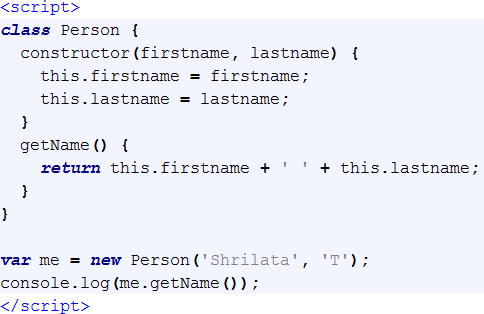
console.log(c2.make, c2.model, c2.price)

//Maruti Suzuki 500000

* Defining methods



constructor method is always defined with the name "constructor"



Classes can have methods, which defined as functions, albeit without needing to use the function keyword.

class Car { constructor(brand) {

this.carname = brand;

}

present() {

return 'I have a ' + this.carname;

}

}

class Model extends Car {

constructor(brand, mod) { super(brand); this.model = mod;

}

show() {

return this.present() + ', it is a ' + this.model;

}

}

let myCar = new Model("Ford", "Mustang"); console.log(myCar.show()); //I have a Ford, it is a Mustang

Inheritance ensures code reuse.

class Animal { speak() {

console.log("Animal makes a sound");

}

}

class Dog extends Animal { speak() {

console.log("Dog barks");

Polymorphism allows objects to behave differently using the same interface

}

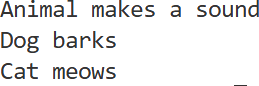
}

class Cat extends Animal { speak() {

console.log("Cat meows");

const animal = new Animal() animal.speak()

}



} const dog = new Dog();

dog.speak();

const cat = new Cat();

cat.speak();

#### Encapsulation wraps up data and information under a single unit.

* + In OOP, it means binding together the data and the functions that manipulate them together in a class

class Person{

constructor(name, age) { this.name = name; this.age = age;

}

getDetails() {

return `${this.name} is ${this.age} years old.`;

}

}

const p1 = new Person("Soha", 30); console.log(p1.getDetails()); //Soha is 30 years old.

## Different ways to achieve encapsulation in JavaScript?

#### [JavaScript](https://www.javascript.com/) is a loosely typed language and does not have traditional access modifiers like private, protected, and public which are common in other object-oriented programming languages.

* However, we can achieve encapsulation in JavaScript through the following methods:
  + Closures
  + Constructor functions

function Person(name) { let \_name = name;

this.getName = function () {

return \_name;

};

this.setName = function (newName) {

\_name = newName;

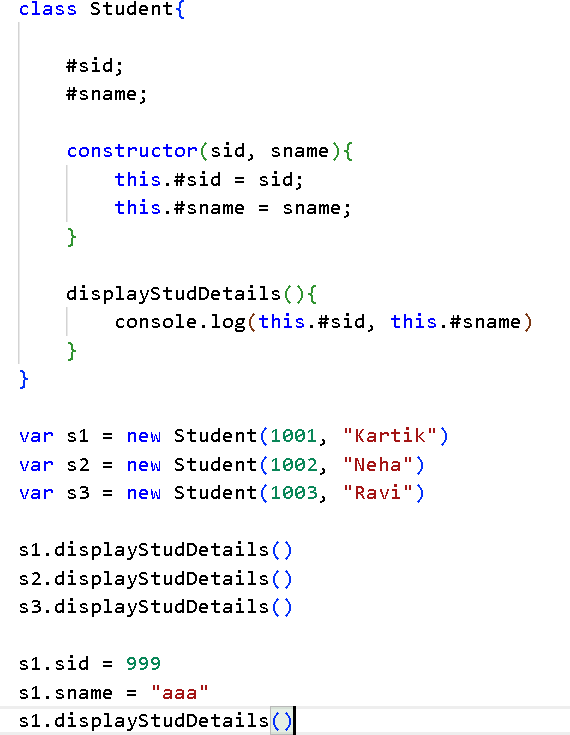
};

}

const person = new Person('John Doe'); console.log(person.getName()); // outputs ‘John Doe’ person.setName('Jane Doe'); console.log(person.getName()); // outputs ‘Jane Doe’ console.log(person.\_name) //undefined

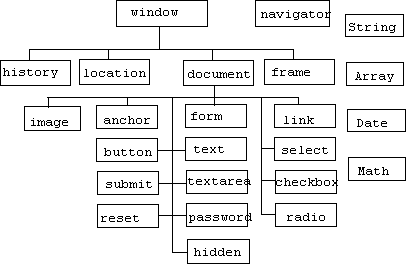
* + Class syntax





The class syntax supports the creation of private properties and methods with the **#** symbol, although this is a proposed feature that may not be widely supported yet.

## JavaScript Document Object Model



**Navigator** **Object** **Hierarchy**

## Window Object Methods

#### alert(message)

* + window.alert(“Display Message”)

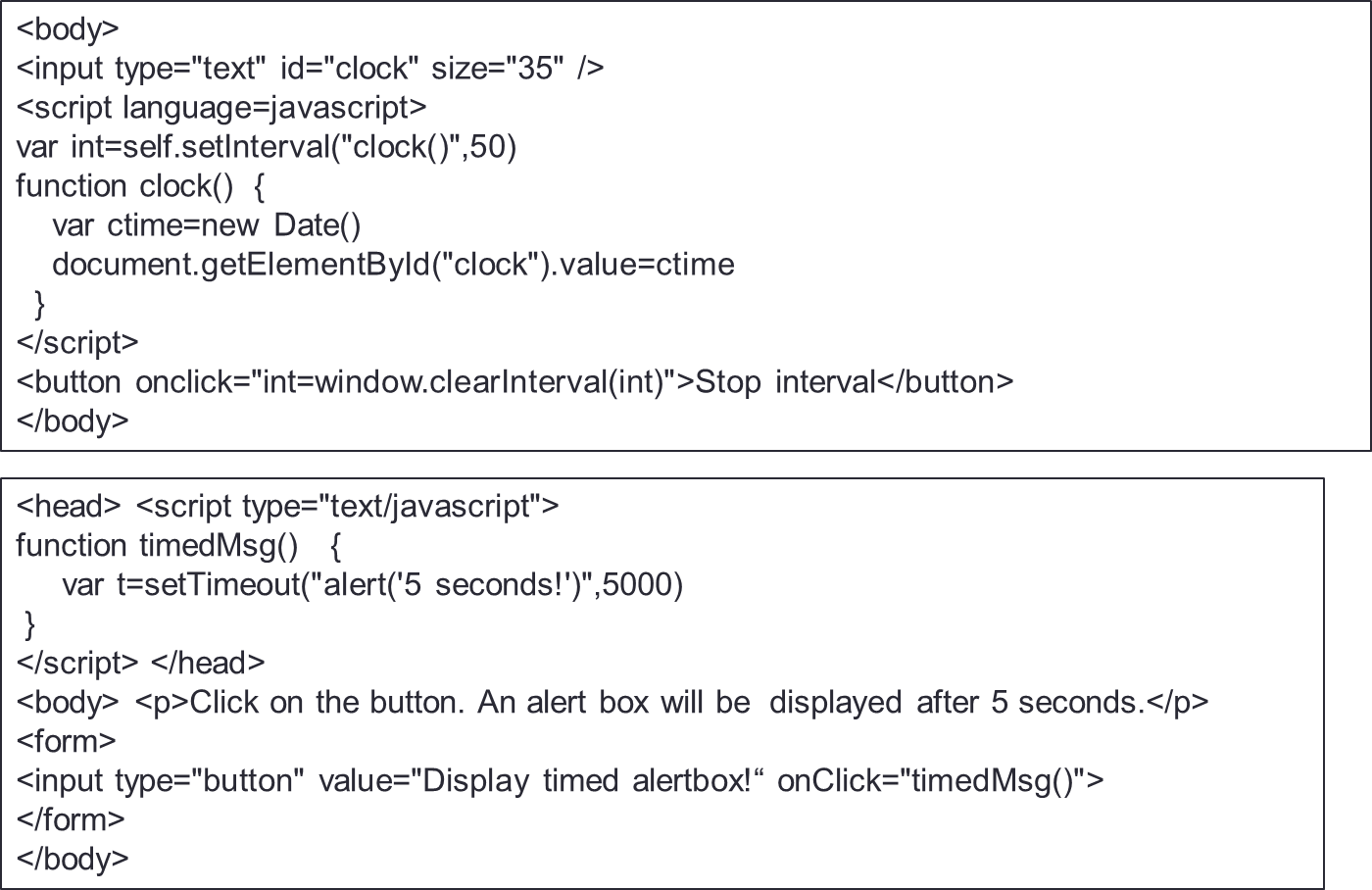
#### confirm(message)

* + window.confirm(“Exit Application ?”)

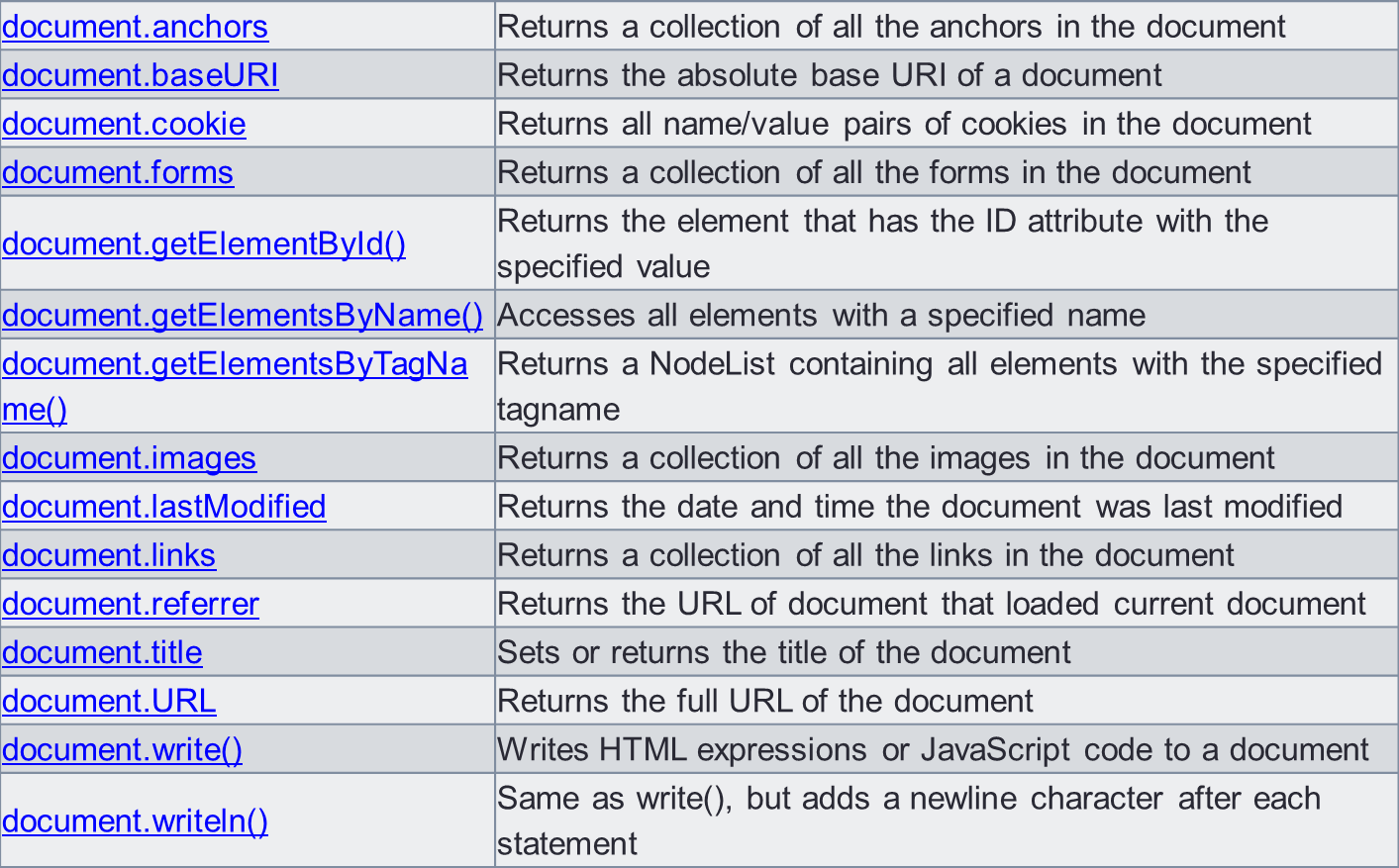
#### prompt(message,[defaultReply])

* + var input=window.prompt(“Enter value of X”)

## setInterval and setTimeout methods

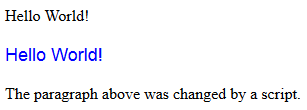


* + When an HTML document is loaded into a web browser, it becomes a document object; root node of the HTML document and owns all other nodes



|  |  |  |
| --- | --- | --- |
| <html>  <body>  <p id="intro">Hello World!</p>  <p>This example demonstrates the <b>getElementById</b> method!</p>  <script>  x=document.getElementById("intro");  document.write("<p>The text from the intro paragraph: " + **x.innerHTML** + "</p>");  </script> | |  |
| </body>  </html> |  |  |
|  |  | |

<body>



<p id="p1">Hello World!</p>

<p id="p2">Hello World!</p>

<script>

document.getElementById("p2").style.color = "blue"; document.getElementById("p2").style.fontFamily = "Arial"; document.getElementById("p2").style.fontSize = "larger";

</script>

<p>The paragraph above was changed by a script.</p>

</body>

## Examples

|  |  |  |
| --- | --- | --- |
| <script>  function f1(){ var str = "";  var plist = document.getElementsByName("c1"); for(i=0;i<plist.length;i++){  if(plist[i].checked)  str = plist[i].value + "&nbsp;&nbsp;" + str ;  }  document.getElementById("s1").innerHTML = "Skills : " + str;  }  </script>  </head>  <body>  <H1> Welcome to Javascript </H1> Skills:  <input type="checkbox" name="c1" value="java">Java  <input type="checkbox" name="c1" value="JS">JavaScript  <input type="checkbox" name="c1" value="JSP">JSP | |  |
| <input type="button" value="click" onclick="f1()"><br>  <span id="s1"></span>  </body> |  |  |
|  |  | |

<html>

<head>

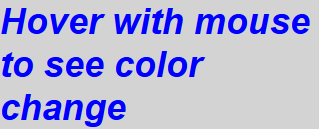
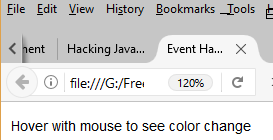
<SCRIPT>

function changeColor(){

var para = document.getElementById("p1"); para.style.color="blue"; para.style.backgroundColor = "lightgray"; para.style.font = "italic bold 30px arial,serif";

## Mouse events

}



function revertColor(){

var para = document.getElementById("p1"); para.style.color="black"; para.style.backgroundColor = "white"; para.style.font = "12px arial,serif";

}

</SCRIPT>

</head>

<body>

<div>

<p id="p1" onmouseover="changeColor()" onmouseout="revertColor()" onclick="f1()" >

Hover with mouse to see color change

</p>

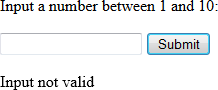
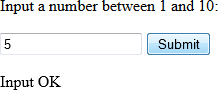
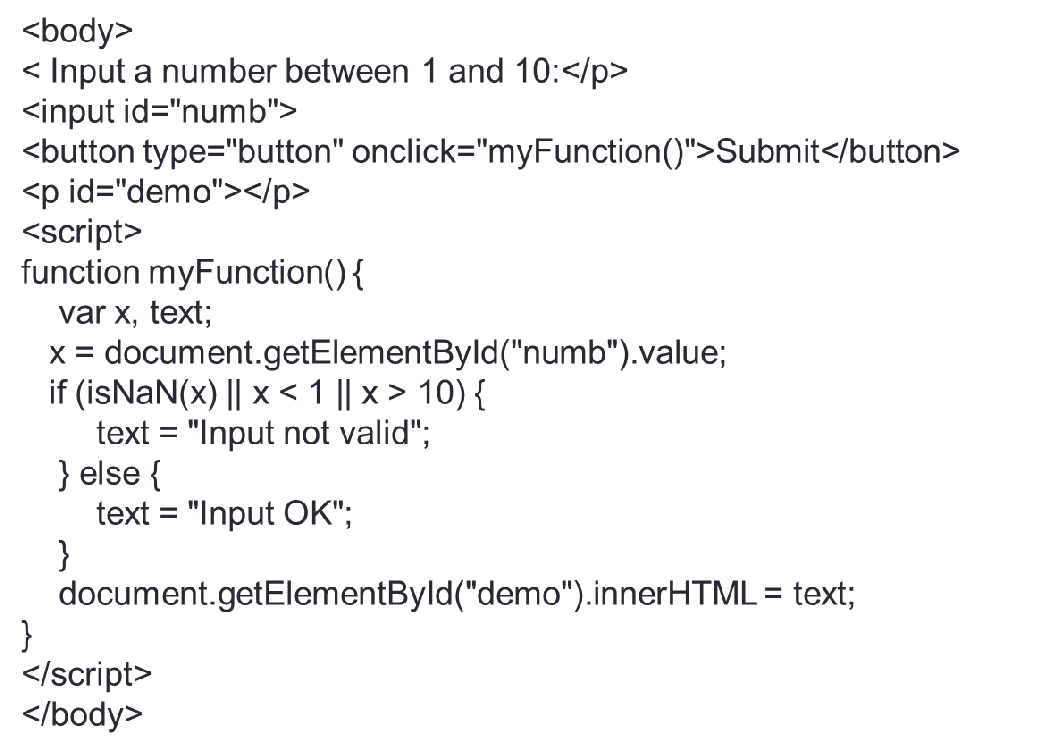
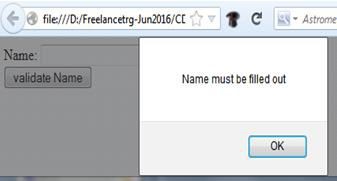
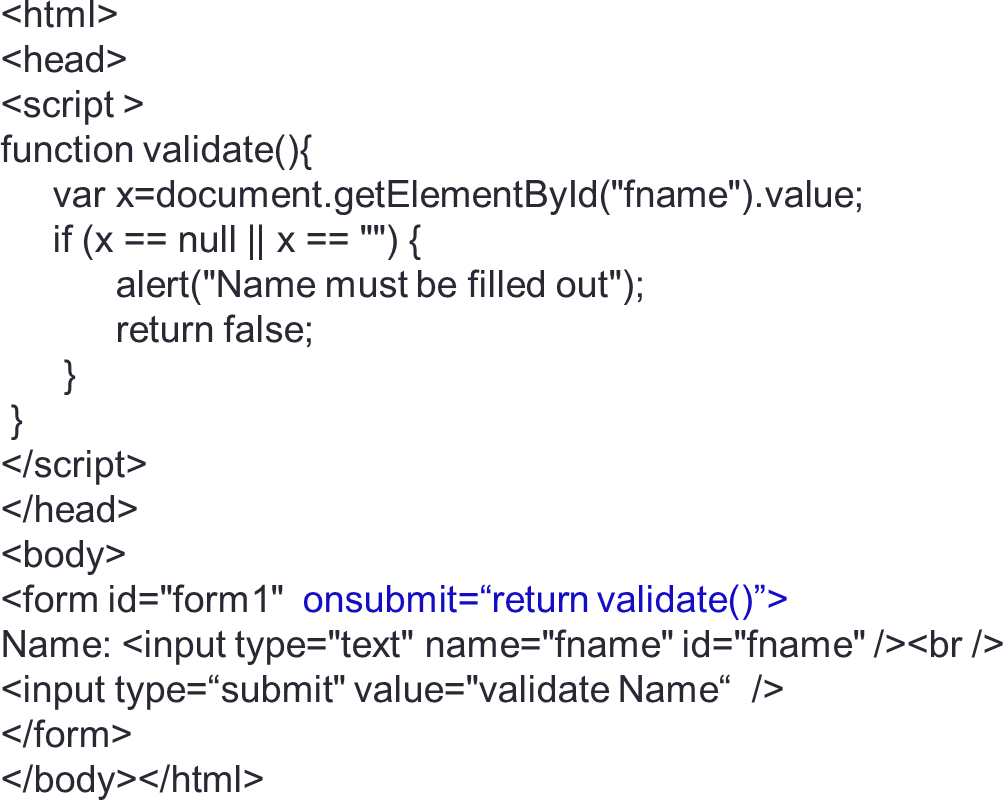
</div>

</body></html>

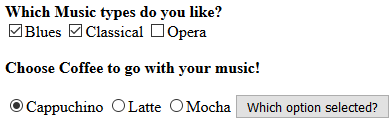
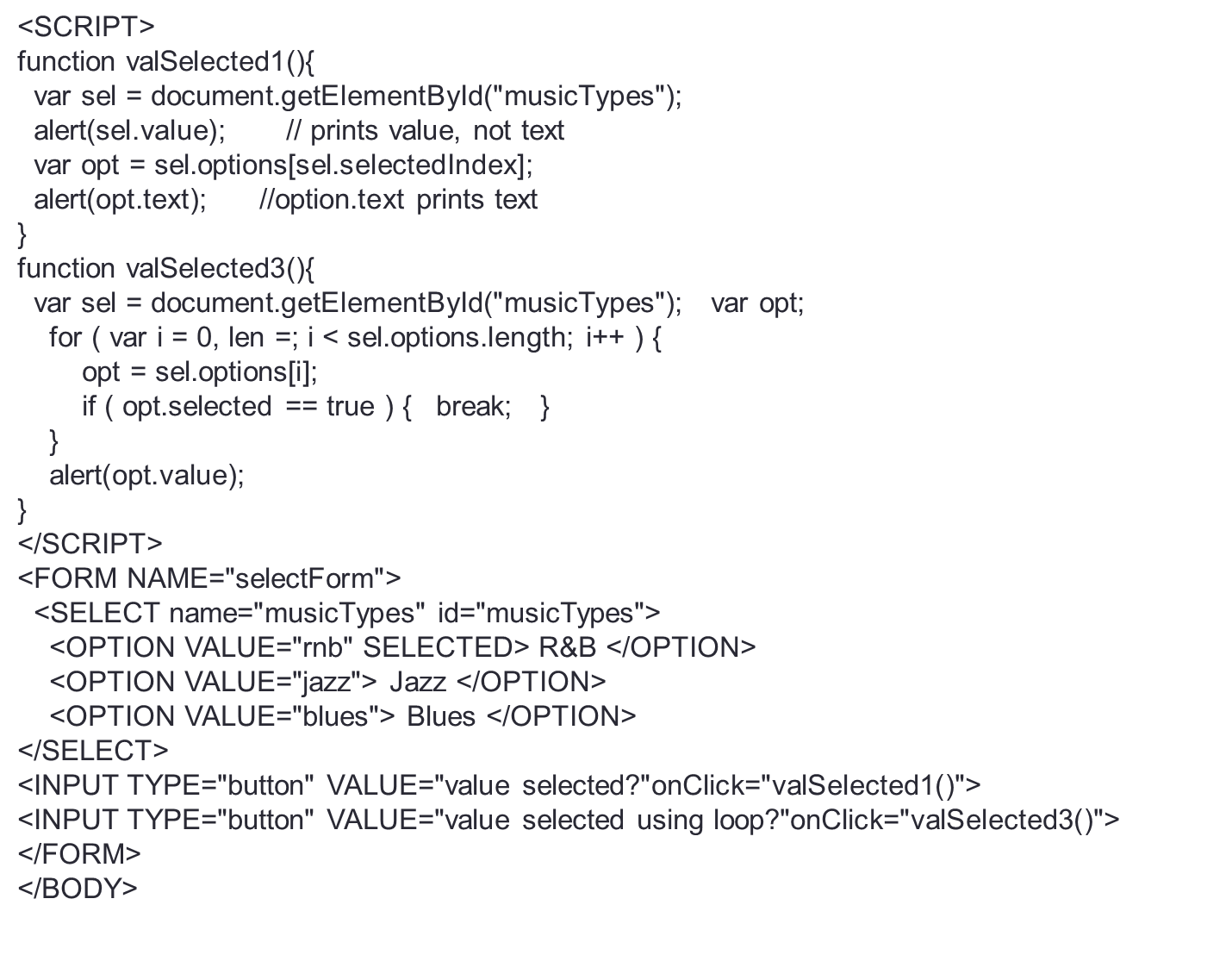
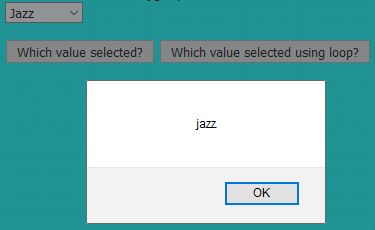
DOM manipulation using javascript

|  |  |  |
| --- | --- | --- |
|  |  |  |
| <!DOCTYPE html>  <html>  <body>  <p>This is para-1</p>  <p>This is para-2</p>  <p>This is para-3</p> |  |  |
| <button onclick="myFunction()">Change!!</button>  <script>  function myFunction() {  var nodelist = document.getElementsByTagName("p");  var i;  for (i = 0; i < nodelist.length; i++) { nodelist[i].style.backgroundColor = "yellow";  }  }  </script>  </body>  </html> | | |

Form validation



Example



Example

<SCRIPT>

function valSelected(){



var radio = document.getElementsByName(“coffee");

for(var i = 0; i < radio.length; i++){

if(radio[i].checked) console.log("coffee selected : " + radio[i].value);

}

var checklist = document.getElementsByClassName("c1"); for(i=0; i<checklist.length; i++){

if (checklist[i].checked == true) console.log("Music selected : " + checklist[i].value);

}

</SCRIPT>

<FORM NAME="selectForm">

<B>Which Music types do you like?</B>

<input type="checkbox" class="c1" id="c1" value="blues">Blues</input>

<input type="checkbox" class="c1" id="c2" value="classical">Classical</input>

<input type="checkbox" class="c1" id="c3" value="opera">Opera</input>

<b>Choose Coffee to go with your music!</b><br>

<INPUT TYPE="radio" name="coffee" id="coffee" VALUE="cappuchino">Cappuchino</input>

<INPUT TYPE="radio" name="coffee" id="coffee" VALUE="latte">Latte</input>

<INPUT TYPE="radio" name="coffee" id="coffee" VALUE="Mocha">Mocha</input>

<INPUT TYPE="button" VALUE="Which option selected?" onClick="valSelected()">

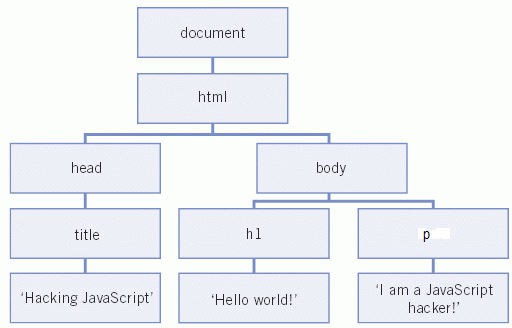
</FORM>

#### The Document object, its Element objects, and Text objects are all Node objects. Node defines the following important properties:

* + parentNode : node that is the parent of this one, or null for nodes like the Document object that have no

parent.

* + childNodes : read-only NodeList that is a representation of a Node’s child nodes.
  + firstChild, lastChild: first & last child nodes of a node; null if node has no children.
  + nextSibling, previousSibling : The next and previous sibling node of a node.
  + nodeValue : textual content of a Text or Comment node.
  + nodeName : tag name of an Element, converted to uppercase.
  + nodeType : kind of node this is.



|  |  |
| --- | --- |
| **Node** **type** | **value** |
| Document | 9 |
| Element | 1 |
| Text | 3 |
| Comments | 8 |

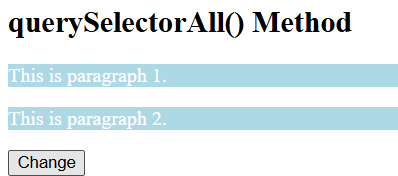
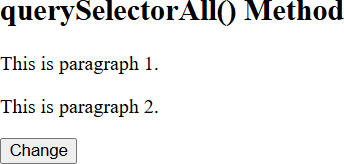
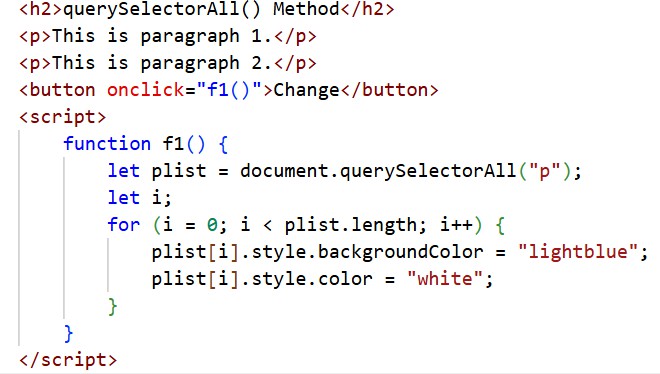
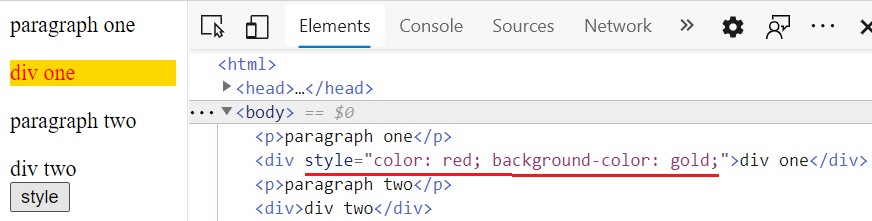
#### The **querySelector()** method returns the first element that matches one or more CSS selectors. If no match is found, it returns null.

* + Its equivalent to getElementById()
  + However, querySelector() and querySelectorAll() are newer methods; with these we are free to target elements based on any CSS selector, thus we have more flexibility.
  + Syntax : var ele = document.querySelector(selector);
  + ele – First matching element or null (if no element matches the selectors)
  + selector – one or more CSS selectors, such as "#fooId", ".fooClass” etc
  + Eg : let e = document.querySelector("p"); //selects the first paragraph in html doc

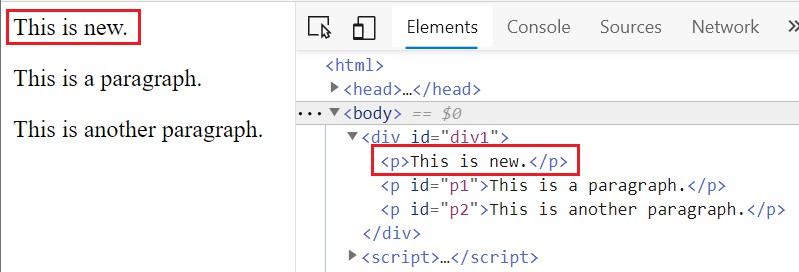
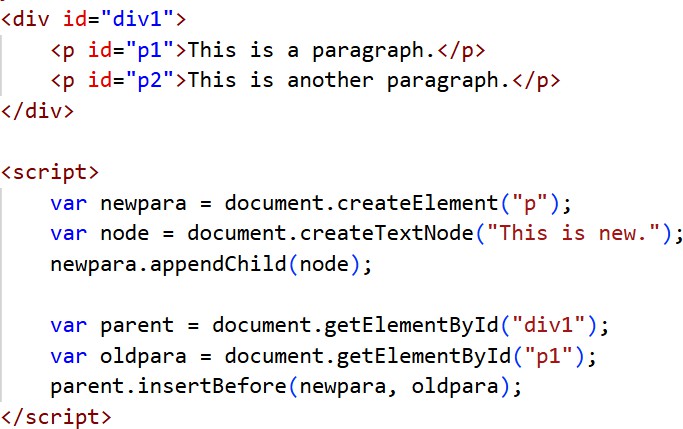
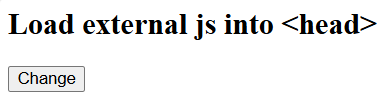
#### **querySelectorAll()** returns a NodeList representing a list of the document's elements that match the specified group of selectors.

* + Syntax : elementList = parentNode.querySelectorAll(selectors);
  + Eg : To obtain a NodeList of all of the <p> elements in the document:
  + const matches = document.querySelectorAll("p");
  + Eg To return a list of all <div> elements within the document with a class of either note or alert:
  + const matches = document.querySelectorAll("div.note, div.alert");

|  |  |  |
| --- | --- | --- |
|  |  | |
| <p>paragraph one</p>  <div>div one</div>  <p>paragraph two</p>  <div>div two</div>  <input type="button" value="style" onclick="f1()">  <script> |  |  |
| function f1() {  var firstDiv = document.querySelector('div'); firstDiv.style.color = 'red'; firstDiv.style.backgroundColor = 'gold’;  }  </script> | |  |



* **createElement()** : create new Element nodes
* **createTextNode()** : Creates text nodes
* **cloneNode()** : returns a new copy of the node
* **appendChild()** or **insertBefore()** : insert new node into the document
* **removeChild():** removes a node from the document tree
  + this method isn’t invoked on the node to be removed but on parent of that node
  + Eg : To remove the node n from the document : n.parentNode.removeChild(n);



<body>

The title of the document is:

<div id="div1">

<p id="p1">This is a paragraph.</p>

<p id="p2">This is another paragraph.</p>

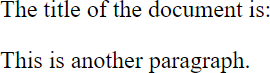
</div>

<script>

var parent = document.getElementById("div1"); var child = document.getElementById("p1"); parent.removeChild(child);

</script>

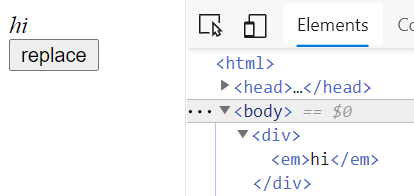
this method isn’t invoked on the node to be removed but on parent of that node



#### The **replaceChild()** method replaces a child element with another one belonging to the parent element that calls this method.

* In this example the child element <strong> belonging to the <div> parent element is replaced with a newly created <em> tag.

<div>



<strong>hello</strong>

</div>

<input type="button" value="replace" onclick="fReplace()">

<script>

function fReplace(){

var em = document.createElement('em');

var strong = document.querySelector('strong'); var div = document.querySelector('div'); em.textContent = 'hi';

div.replaceChild(em, strong);

}

</script>

#### A regular expression is an object that describes a pattern of characters.

* + Its matched against a text string, when you perform searches & replacements
  + Perform client-side data validations or any other extensive text entry parsing
  + RegExp objects may be created either with the RegExp() constructor or using a special literal syntax.
  + regular expression literals are specified as characters within a pair of slash (/)

var re = / / var re = / /g

var re = /web/i var re = /web/gi

simple pattern to match the space character

matching a string on a global basis

a case-insensitive match

expression is both case-insensitive and global

* + Eg

str = "I love JavaScript!"; regexp = /love/;

alert( str.search(regexp) ); // 2

|  |  |
| --- | --- |
| [i](https://www.w3schools.com/jsref/jsref_regexp_i.asp) | Perform case-insensitive matching |
| [g](https://www.w3schools.com/jsref/jsref_regexp_g.asp) | Perform a global match (find all matches  rather than stopping after the first match) |
| [m](https://www.w3schools.com/jsref/jsref_regexp_m.asp) | Perform multiline matching |

#### \d Numeral: Find any single digit 0 through 9

* /\d\d\d/ matches “212” and “415” but not “B17”

#### \D Non-numeral: Find any non-digit

* /\D\D\D/ matches “ABC” but not “212” or “B17”

#### \s Single White Space: Find any single space character

* /over\sbite/ matches “over bite” but not “overbite” or “over bite”

#### \S Single Non-White Space:

* /over\Sbite/ matches “over-bite” but not “overbite” or “over bite”

#### \w Letter, Numeral, or Underscore:

* /A\w/ matches “A1” and “AA” but not “A+”

#### \W Not letter, Numeral, or Underscore:

* /A\W/ matches “A+” but not “A1” and “AA”

#### “.” Any Character Except Newline:

* /…/ matches “ABC”, “1+3”, “A 3” or any 3 characters

#### […] Character Set: any character in the specified character set

* /[AN]BC/ matches “ABC” and “NBC”

#### [^…] Negated Character Set: any character not in the specified character set

* /[^AN]BC/ matches “BBC” and “CBC” but not “ABC” or “NBC”

### Positional Metacharacters

#### “^” - At the beginning of a string or line

* /^Fred/ matches “Fred is OK” but not “I’m with Fred” or “Is Fred here?”

#### “$” - At the end of a string or line

* /Fred$/ matches “I’m with Fred” but not “Fred is OK” or “Is Fred here?”

#### “\*” - Zero or More Times:

* + /Ja\*vaScript/ matches “JvaScript”, “JavaScript” & “JaaavaScript” but not “JovaScript”

#### “?” - Zero or One Time:

* + /Ja?vaScript/ matches “JvaScript” or “JavaScript” but not “JaaavaScript”

#### “+” - One or More Times:

* + /Ja+vaScript/ matches “JavaScript” or “JaavaScript” but not “JvaScript”

#### {n} - Exactly n Times:

* + /Ja{2}vaScript/ matches “JaavaScript” but not “JvaScript” or “JavaScript”

#### {n,} - N or More Times:

* + /Ja{2,}vaScript/ matches “JaavaScript” or “JaaavaScript” but not “JavaScript”

#### {n,m} - At Least n, At Most m Times:

* + /Ja{2,3}vaScript/ matches “JaavaScript” or “JaaavaScript” but not “JavaScript”

## Regular Expression Object

#### Create Regular Expression:

regExpObject = /pattern/ [g | i | gi] regExpObject = new RegExp(pattern, flag)

* + Eg: re = new RegExp( "pushing", "g" );
  + Eg: var zipcode = new RegExp("\\d{6}", "g");
* Methods that use regular expressions

|  |  |
| --- | --- |
| **Method** | **Description** |
| exec() | executes a search for a match in a string. It returns an array of information or null on a mismatch. |
| test() | tests for a match in a string. It returns true or false |
| match() | executes a search for a match in a string. It returns an array of information or null on a mismatch. |
| search() | tests for a match in a string; returns the index of the match, or -1 if search fails |
| replace() | executes a search for a match in a string, and replaces the matched substring with a  replacement substring. |
| split() | uses a reg exp or a fixed string to break a string into an array of substrings. |

var pattern = /java/g;

var text = "JavaScript is more fun than java!"; var result = pattern.exec(text)

console.log(result) //["java", index: 28, input: "JavaScript is more fun than java!"]

When there’s a "g" flag, then str.match returns an array of all matches. There are no additional properties in that array, and parentheses do not create any elements.

With no “g” flag, looks for the first match only.

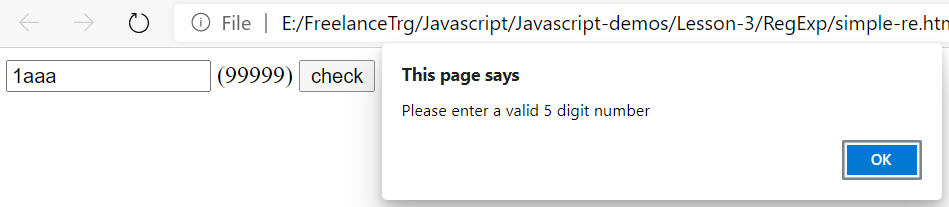
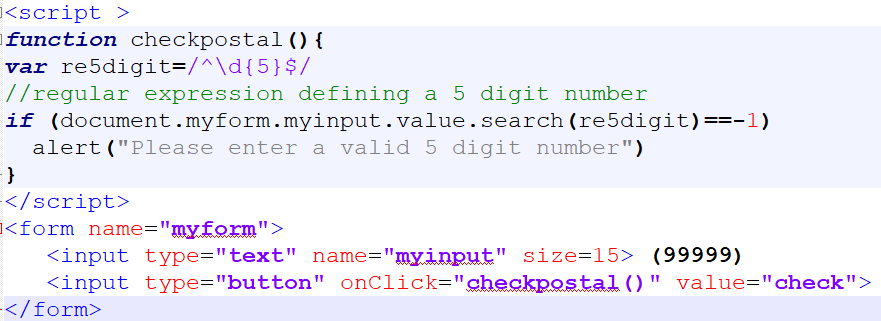
let str = "HO-Ho-ho!";

let result = str.match( /ho/ig ); //global search

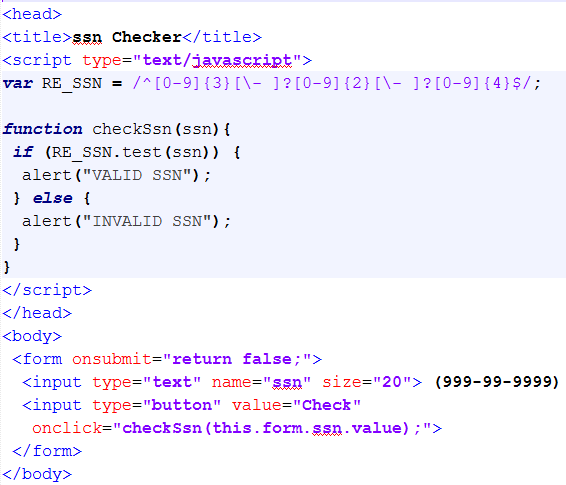
alert( result ); // HO, Ho, ho (all matches, case-insensitive)

s1 = "how are you all doing"; var re = new RegExp("o","g");

console.log(s1.replace(re,"z")); //hzw are yzu all dzing



Demo

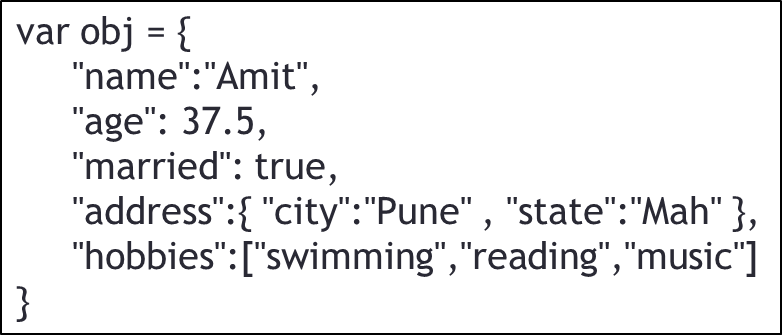


# JSON

## JSON (JavaScript Object Notation)

#### JSON is a simple and easy to read and write data exchange format.

* + It is easy for humans to read and write and easy for machines to parse and generate.
  + It is based on a subset of the JavaScript, Standard ECMA-262
  + JSON is a text format that is completely language independent; can be used with most of the modern programming languages.
  + The filename extension is .json
  + JSON Internet Media type is application/json
  + It’s popular and implemented in countless projects worldwide, for those don’t like XML, JSON is a very

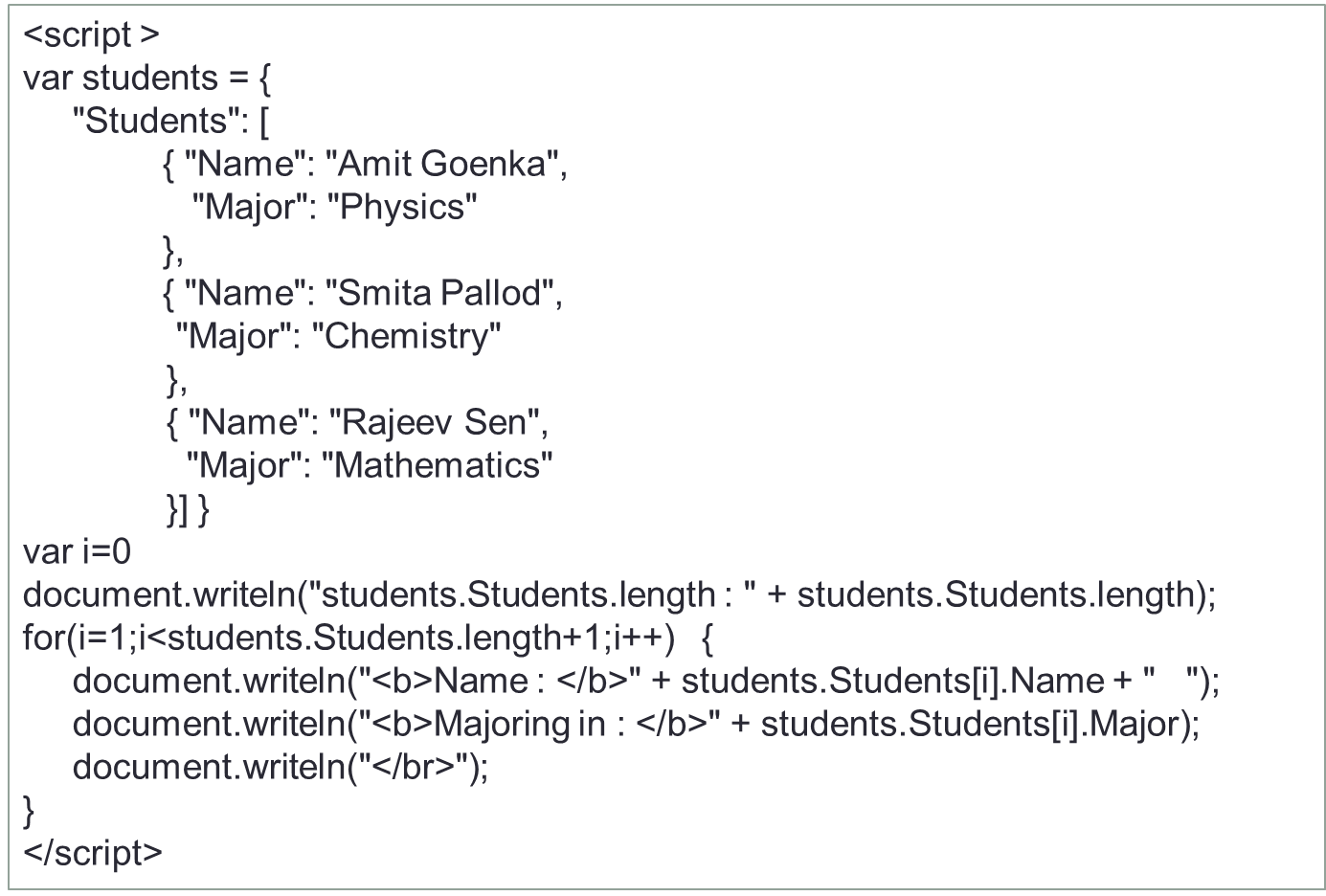
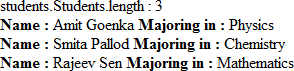
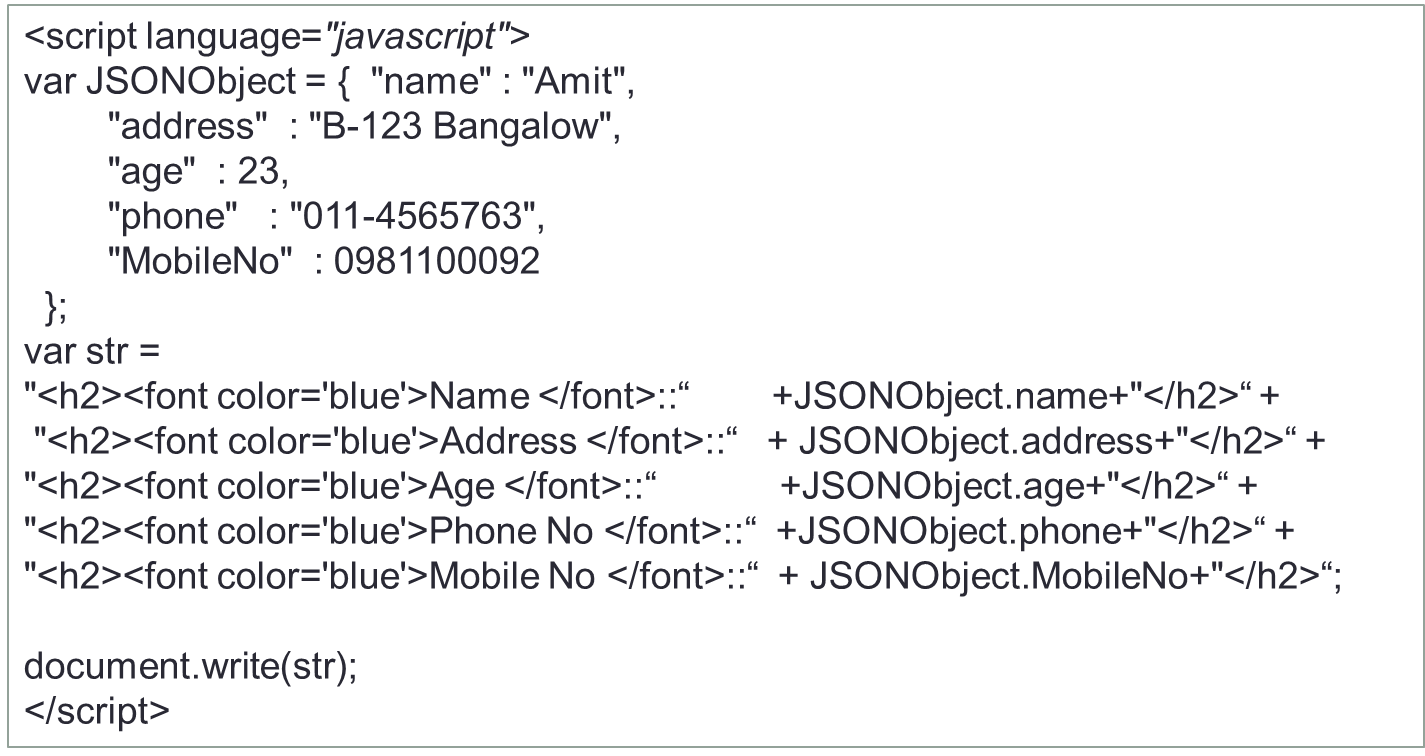
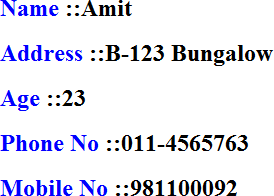
good alternative solution.

#### Values supported by JSON

* + Strings : double-quoted Unicode, with backslash escaping
  + Numbers:
    - double-precision floating-point format in JavaScript
  + Booleans : true or false
  + Objects: an unordered, comma-separated collection of key:value pairs enclosed in [curly braces](http://en.wikipedia.org/wiki/Curly_brace), with the ':' character separating the key and the value; the keys must be strings and should be distinct from each other
  + Arrays : an ordered, comma-separated sequence of values enclosed in square brackets; the values do

not need to be of the same type

* + Null : A value that isn't anything



## Serializing Objects

#### serialization is the process of converting an object’s state to a string from which it can later be

restored.

* + Sometimes we receive a raw JSON string, and we need to convert it to an object. And when we want to send a JavaScript object across the network, we need to convert it to JSON (a string) before sending.
  + **JSON.parse():** Accepts a JSON string as a parameter, and returns the corresponding JavaScript object.
  + **JSON.stringify():** Accepts an object as a parameter, and returns the equivalent JSON string.

o = {x:1, y:{z:[false,null,""]}};

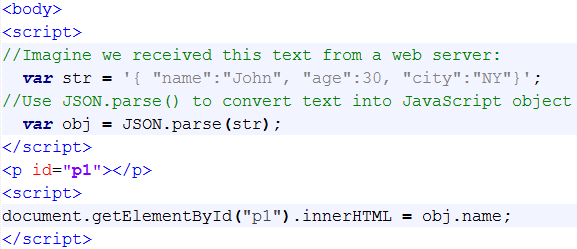
s = JSON.stringify(o); p = JSON.parse(s); console.log(p.x);

// Define a test object

// s is '{"x":1,"y":{"z":[false,null,""]}}'

// p is a deep copy of o

//1



typeof(obj) returns object

## What is Ajax ?

#### “Asynchronous JavaScript And XML”

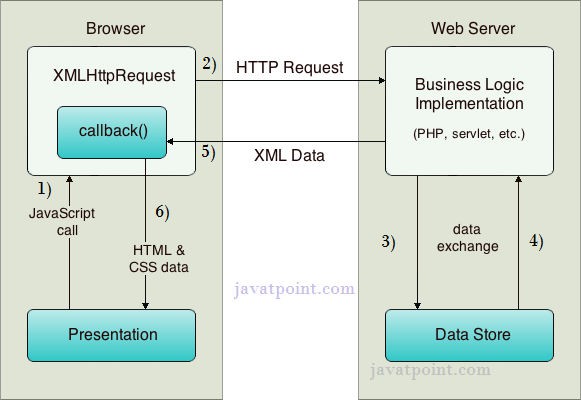
* + AJAX is not a programming language, but a technique for making the user interfaces of web applications more responsive and interactive
  + It provide a simple and standard means for a web page to communicate with the server without a complete page refresh.

### Why Ajax?

#### Intuitive and natural user interaction

* + No clicking required. Call can be triggered on any event
  + Mouse movement is a sufficient event trigger

#### "Partial screen update" replaces the "click, wait, and refresh" user interaction model

* + Only user interface elements that contain new information are updated (fast response)
* The rest of the user interface remains displayed as it is without interruption (no loss of operational context)

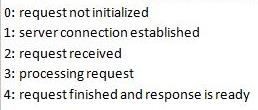
AJAX architecture

## XMLHttpRequest

#### JavaScript object - XMLHttpRequest object for asynchronously exchanging the XML data between the client and the server

* XMLHttpRequest Methods
  + open(“method”, “URL”, syn/asyn) : Assigns destination URL, method, mode
  + send(content) : Sends request including string or DOM object data
  + abort() : Terminates current request
  + getAllResponseHeaders() : Returns headers (labels + values) as a string
  + getResponseHeader(“header”) : Returns value of a given header
  + setRequestHeader(“label”,”value”) : Sets Request Headers before sending

#### XMLHttpRequest Properties

* + Onreadystatechange : Event handler that fires at each state change
  + readyState values – current status of request
  + Status : HTTP Status returned from server: 200 = OK
  + responseText : get the response data as a string
  + responseXML : get the response data as XML data

## Creating an AJAX application

* Step 1: Get an instance of XHR object

xhr = new XMLHttpRequest();

* Step 2: Make the request

xhr.open('GET', ['http://www.example.org/some.file',](http://www.example.org/some.file%27) true); xhr.send(null);

xhr.open("POST", "AddNos.jsp");

xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded"); xhr.send("tno1=100&tno2=200");

* Step 3 : Attach callback function to xhr object

function loadDoc() {

var xhttp = new XMLHttpRequest(); xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) { document.getElementById("demo").innerHTML = this.responseText;

}

};

xhttp.open("GET", "ajax\_info.txt", true); xhttp.send();

}

<script > var xhr;

function getData(){

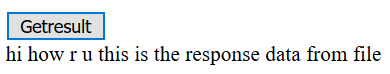
xhr = new XMLHttpRequest();

if(xhr){

xhr.open("GET", "Sample.txt", true); xhr.send();

## Ajax Demo

//sample.txt



hi how r u this is the

response data from file

xhr.onreadystatechange = function(){

if(xhr.readyState == 4 && xhr.status == 200){

document.getElementById("lblresult").innerHTML=xhr.responseText;

}

} //end of callback function

}}

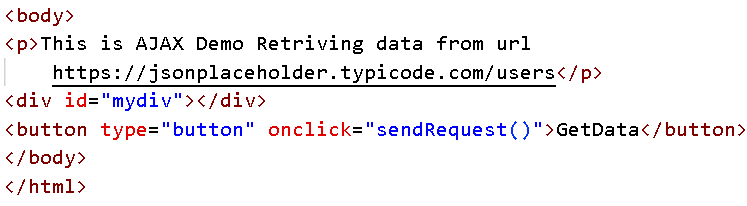
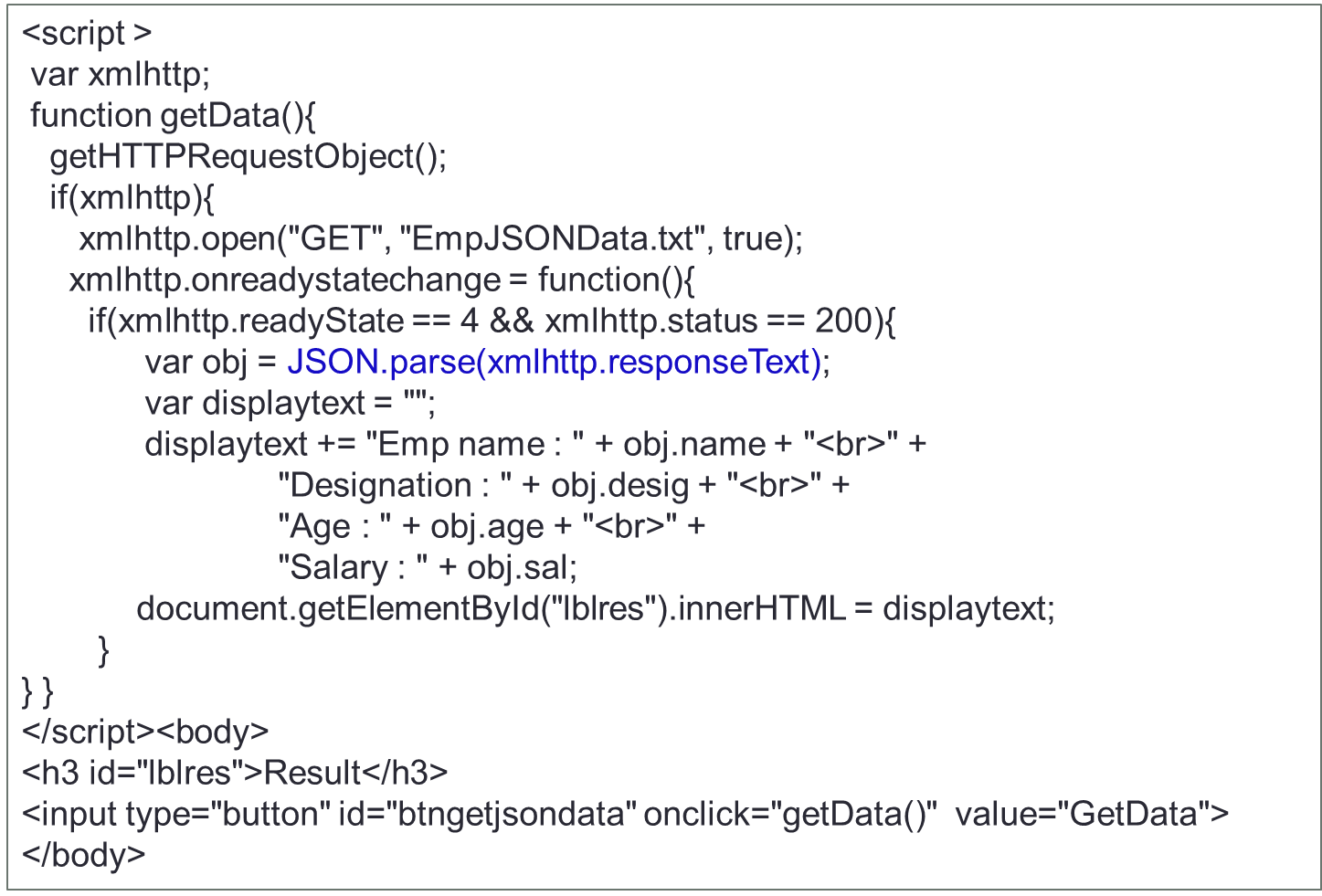
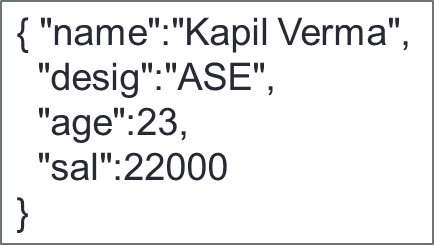
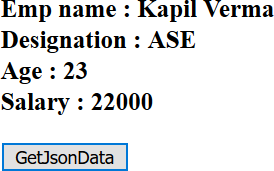
</script> <body>

<input type="button" onclick="getData()" value="Getresult"/>

<div id="lblresult"></div>

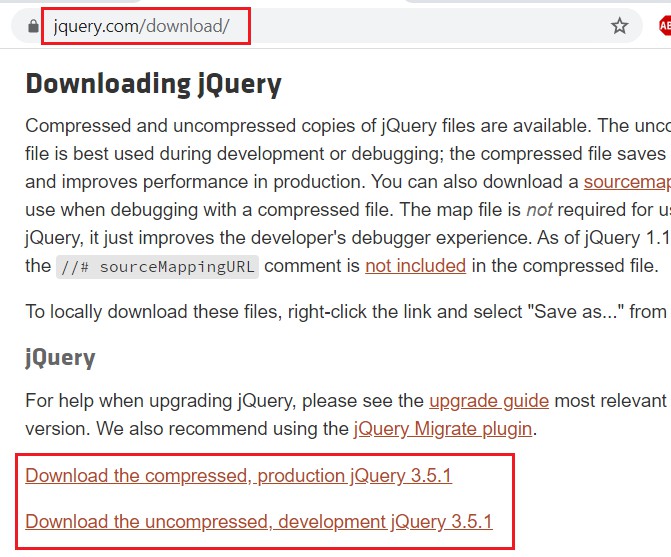
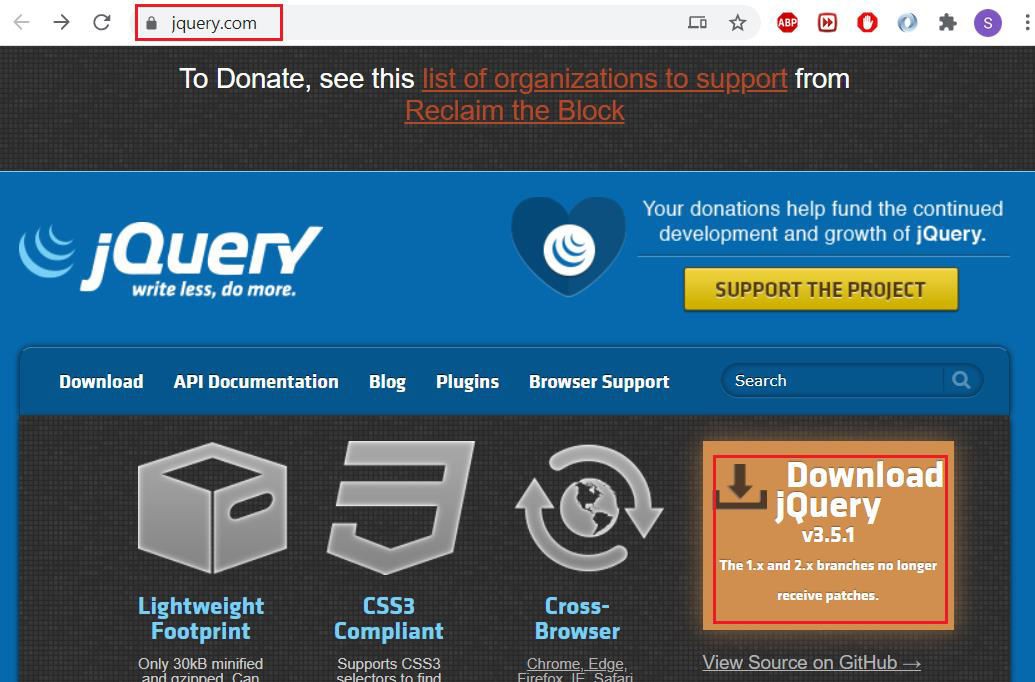
</body>

AJAX Demo with JSON



# JQUERY

## Software



* **Choosing** **a** **Text** **Editor**
  + Text editors that support jQuery include Brackets, Sublime Text, Kwrite, Gedit, Notepad++, PSPad,

or TextMate.

## jQuery Introduction

#### jQuery is a lightweight, cross browser and feature-rich JavaScript library which is used to manipulate DOM

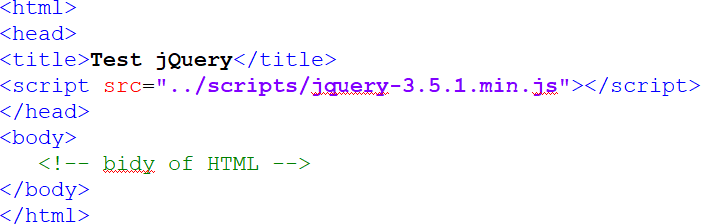
* + Originally created by John Resig in early 2006.
  + The jQuery project is currently run and maintained by a distributed group of developers as an open-source project.

#### Why jQuery

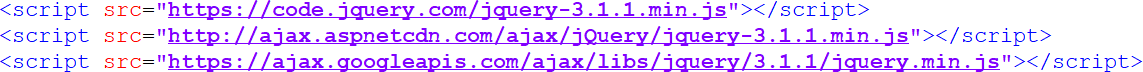
* + JavaScript is great for a lot of things especially manipulating the DOM but it’s pretty complex stuff. jQuery abstracts away a lot of the complexity involved in dealing with the DOM, and makes creating effects super easy.
  + It can locate elements with a specific class
  + It can apply styles to multiple elements
  + It solves the cross browser issues
  + It supports method chaining
  + It makes the client side development very easy

## Including jQuery in HTML Document

* jQuery library can be included in a document by linking to a local copy or to one of the versions available from public servers.
* Eg : include a local copy of the jQuery library



* Eg : include the library from a publicly available repository
  + There are several well-known public repositories for jQuery; these repositories are also known as Content Delivery Networks (CDNs).



## Using jQuery

<html>

<head>

<title>Test jQuery</title>

<script type="text/javascript" src="jquery-3.5.1.js"></script>

<script type="text/javascript">

$(document).ready(function() {

alert('Hi');

});

</script>

</head>

<body>

Welcome to jQuery

</body>

</html>

//shortcut…

$(function() {

// jQuery code

});

Introduction to selectors

#### jQuery uses same CSS selectors used to style html page to manipulate elements

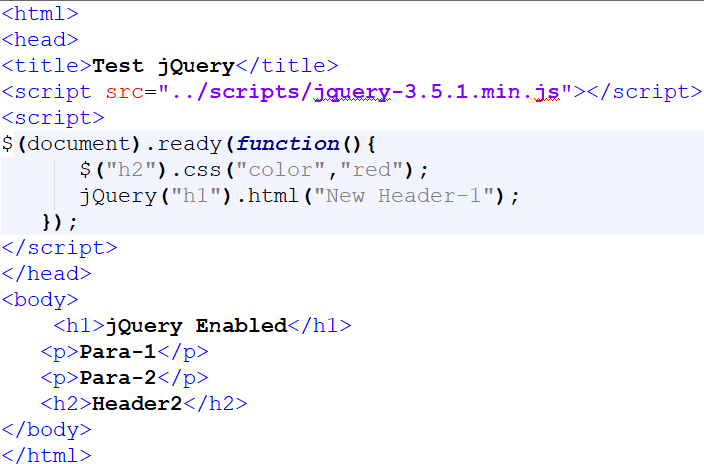
* + CSS selectors select elements to add style to those elements where as jQuery selectors select

elements to add behavior to those elements.

* + Selectors allow page elements (Single or Multiple) to be selected.

#### Selector Syntax

* + $(selectorExpression)



* + jQuery(selectorExpression)
  + $(selector).action()

## Selectors

### Selecting by Tag Name:

#### Selecting single tag takes the following syntax

* + $(‘p’) – selects all <p> elements
  + $(‘a’) – selects all <a> elements

#### To reference multiple tags, use the ( , ) to separate the elements

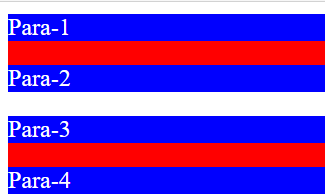
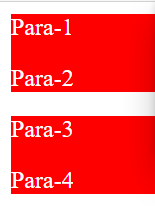
* + $(‘p, a, span’) - selects all paragraphs, anchors and span elements

### Selecting Descendants

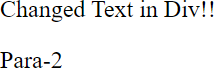
#### $(‘ancestor descendant’) - selects all the descendants of the ansector

* + $(‘table tr’) - Selects all tr elements that are the descendants of the table element
* Descendants can be children, grand children etc of the designated ancestor element.

## Demo

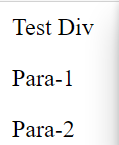
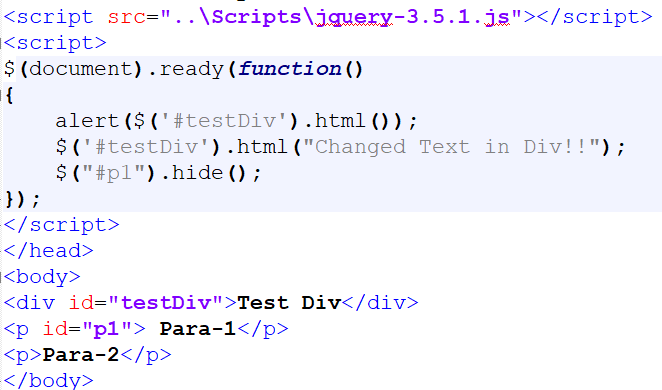


Selecting by Element ID



#### It is used to locate the DOM element very fast.

* Use the # character to select elements by ID
  + $("#first") — selects the element with id="first"
  + $(‘#myID’) – selects the element with id=" myID "



## Selecting Elements by Class Name

#### Use the ( . ) character to select elements by class name

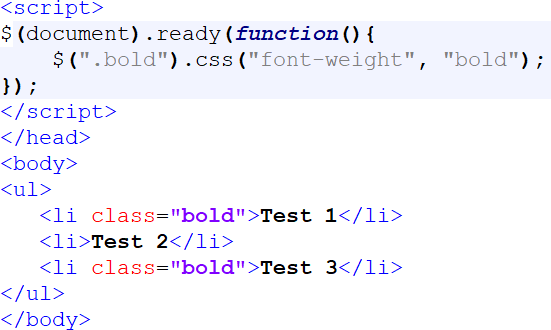
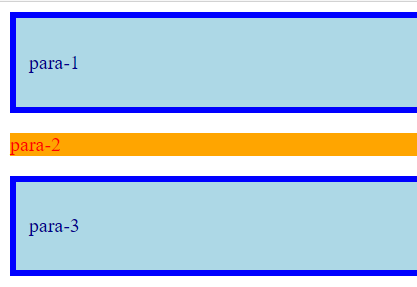
* + $(".intro") — selects all elements with class="intro"

#### To reference multiple tags, use the ( , ) character to separate class name.

* + $(‘.blueDiv, .redDiv’) - selects all elements containing class blueDiv and redDiv

#### Tag names can be combined with elements name as well.

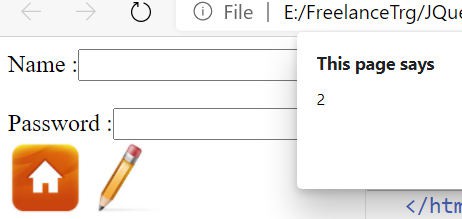
* + $(‘div.myclass’) – selects only those <div> tags with class=“myclass”



## Selecting by attribute values

#### Use brackets [attribute] to select on attribute name and/or attribute value

* + $(‘a[title]’) - selects all anchor elements that have a title attribute
  + $(‘a[title=“trainer”]’) – selects all <a> elements that have a “trainer” title attribute value



## Selecting by input elements

#### To select input elements of type : <input>:

* + $('input[type="text"]').css("background", "yellow");

#### To select all input elements

* + $(‘:input’) - Selects all form elements (input, select, textarea, button).
  + $(‘:input[type=“radio”]’) – selects all radio buttons
  + $(":text") - All input elements with type="text"
  + $(":password") - All input elements with type="password"
  + $(":radio") - All input elements with type="radio"
  + $(":checkbox") - All input elements with type="checkbox"
  + $(":submit") - All input elements with type="submit"
  + $(":reset") - All input elements with type="reset"
  + $(":button") - All input elements with type="button"
  + $(":file") - All input elements with type="file"



## Filters

#### The index-related selectors (:eq(), :lt(), :gt(), :even, :odd) filter the set of elements that have matched the expressions that precede them.

* + They narrow the set down based on the order of the elements within this matched set.
  + Eg, if elements are first selected with a class selector (.myclass) and four elements are returned, these elements are given indices 0 through 3

#### eq() - Select the element at index n within the matched set.

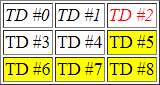
* + Eg : $("p:eq(1)") - Select the second <p> element
  + Eg : $("element:eq(0)") is same as $(‘element:first-child’)

#### $(‘element:odd’) and $(‘element:even’) selects odd and even positions respectively. 0 based indexing

* + Odd returns (1,3,5…) and Even returns (0,2,4…)

#### :gt() and lt() - Select all elements at an index > or < index within the matched set

* + Eg : $("tr:gt(3)") : Select all <tr> elements after the 4 first
  + Eg : $("tr:lt(4)") : Select the 4 first <tr> elements



<table border="1">

<tr><td>TD #0</td><td>TD #1</td><td>TD #2</td></tr>

<tr><td>TD #3</td><td>TD #4</td><td>TD #5</td></tr>

<tr><td>TD #6</td><td>TD #7</td><td>TD #8</td></tr>

</table>

<script>

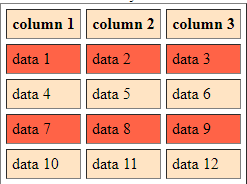
$( "td:eq( 2 )" ).css( "color", "red" );

//$( "tr:first" ).css( "font-style", "italic" ); //is same as below line

$( "tr:eq(0)" ).css( "font-style", "italic" );

$( "td:gt(4)" ).css( "backgroundColor", "yellow" );

</script>



<script type="text/javascript">

$(document).ready(function() {

$('tr:odd').css('background-color','tomato');

$('tr:even').css('background-color','bisque');

});

</script>

<table border=1 cellspacing=5 cellpadding=5>

<th>column 1<th>column 2<th>column 3

<tr><td>data 1</td><td>data 2</td><td>data 3

<tr><td>data 4<td>data 5<td>data 6

<tr><td>data 7<td>data 8<td>data 9

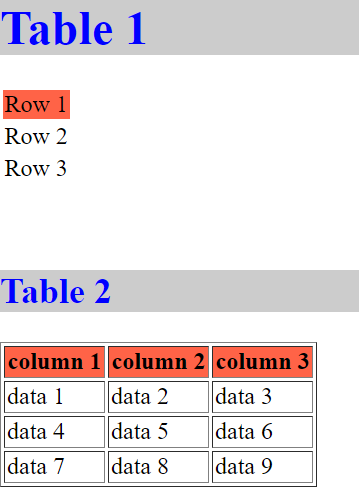
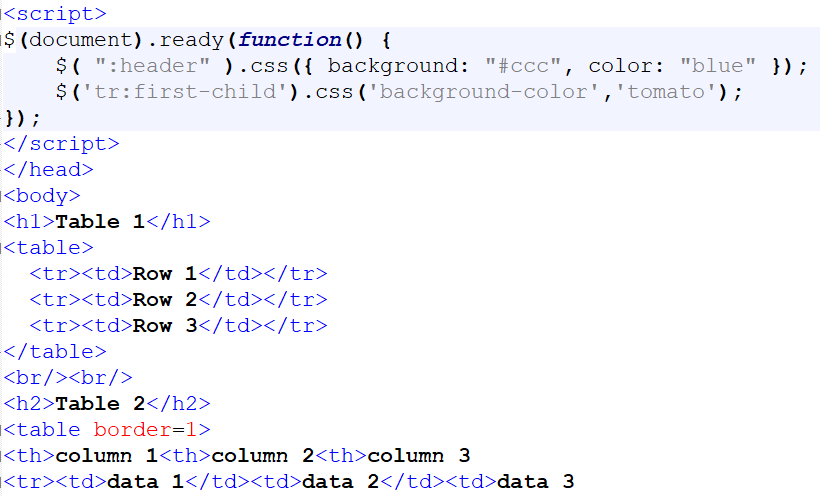
<tr><td>data 10<td>data 11<td>data 12

</table>

## Basic Filters continued

#### :first , :last - Selects the first/last matched element.

* + :first is equivalent to :eq( 0 ) and :lt(1). This matches only a single element, whereas, :[first-child](http://api.jquery.com/first-child-selector/) can match more than one; one for each parent.
* :header - Selects all elements that are headers, like h1, h2, etc



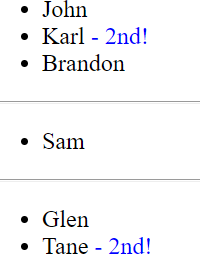
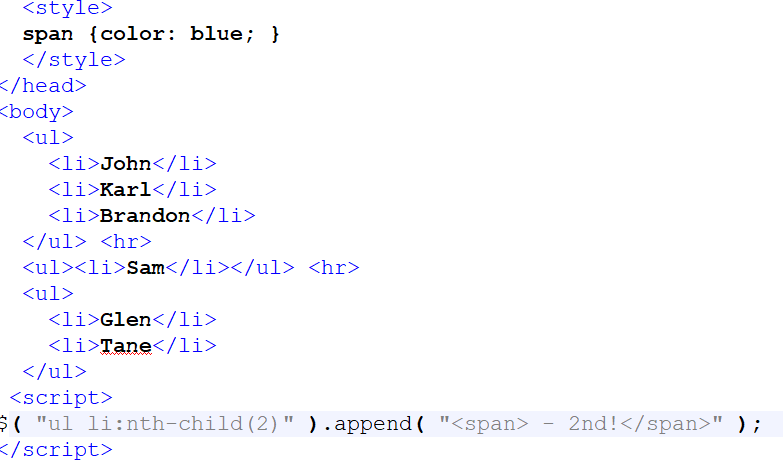
## Child Filter

#### $(‘element:first-child’) and $(‘element:last-child’) selects the first child & last child of its parent.

* + $(‘span:first-child’) returns the span which is a first child for all the groups

#### :nth-child() - Selects all elements that are the nth-child of their parent. 1-based indexing

* + Eg : $("p:nth-child(3)") - Select each <p> element that is the third child of its parent







## Content Filters

#### :contains() will select elements that match the contents.

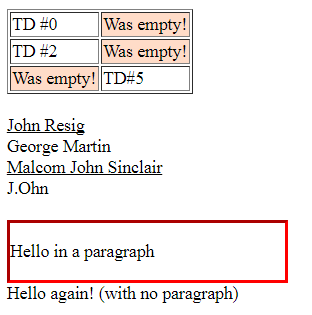
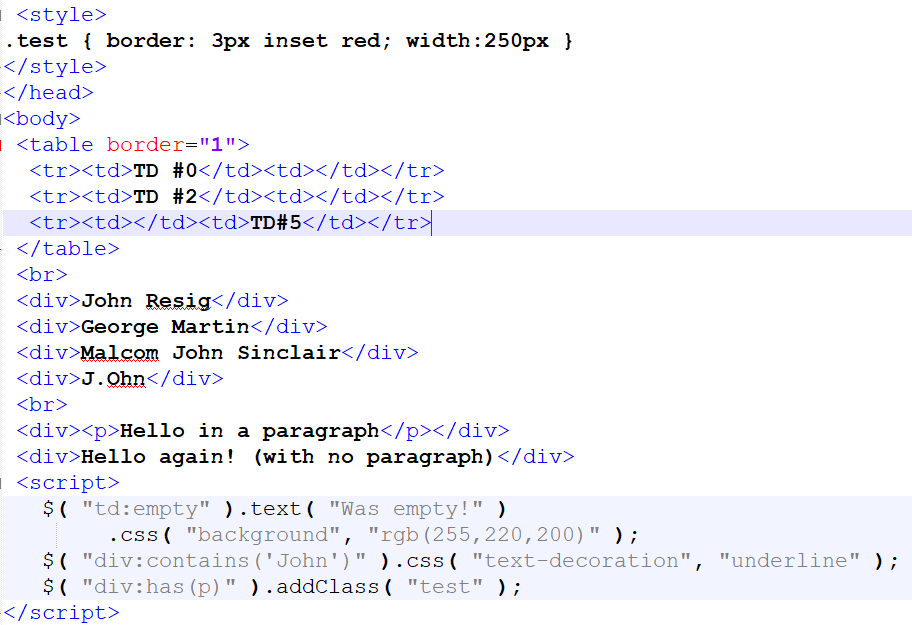
* + $(‘div:contains(“hello”)’) - selects div’s which contains the text hello (match is case sensitive)

#### :empty - Select all elements that have no children (incl text nodes)

* :has : Selects elements which contain at least one element that matches the specified selector.
  + Eg : $("p:has(span)") - Select all <p> elements that have a <span> element inside of them
  + Eg : $("div:has(p,span,li)").css("border","solid red"); - Select all <div> elements that have at least one of the given elements inside

#### :parent - Select all elements that have at least one child node (either an element or text).

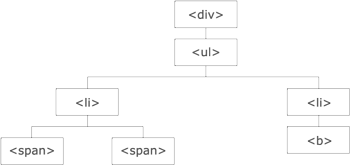
* + Eg : $("td:parent") - Select all <td> elements with children, including text



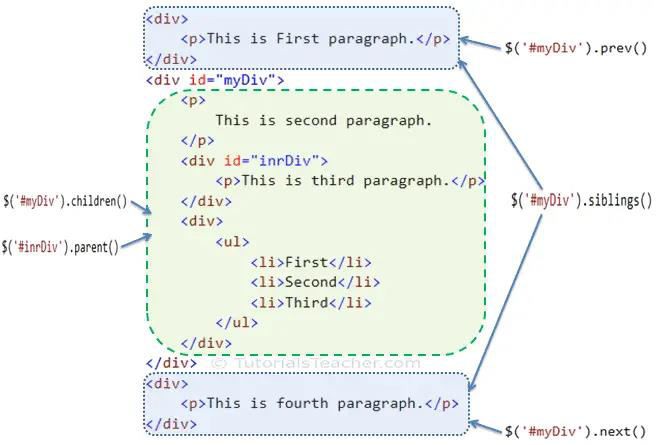
## jQuery Traversing

#### 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.

Most of the DOM Traversal Methods do not modify the jQuery DOM object and they are used to filter out elements from a document based on given conditions

* + - The <div> 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 <div>
    - The left <li> element is the **parent** of <span>, **child** of <ul> and a **descendant** of <div>
    - The <span> element is a **child** of the left <li> and a **descendant** of <ul> and <div>
    - And so on….

<https://api.jquery.com/category/traversing/>

|  |  |
| --- | --- |
| **Methods** | **Description** |
| children() | Get all the child elements of the specified element(s) |
| each() | Iterate over specified elements and execute specified call back  function for each element. |
| find() | Get all the specified child elements of each specified element(s). |
| first() | Get the first occurrence of the specified element. |
| next() | Get the immediately following sibling of the specified element. |
| parent() | Get the parent of the specified element(s). |
| prev() | Get the immediately preceding sibling of the specified element. |
| siblings() | Get the siblings of each specified element(s) |

**jQuery** **methods** **for** **traversing** **DOM** **elements**

## jQuery Traversing -> filtering

#### .eq() - Reduce the set of matched elements to the one at the specified index.

* + Eg : $("p").eq(1).css("background-color","yellow") - Select the second <p> element (index number 1)

#### .filter() - Reduce the set of matched elements to those that match the selector or pass the function's test

* + Eg : $("p").filter(".intro") - Return all <p> elements with class "intro“
* .first() / last()
  + Eg : $("div p").first() - Select first <p> element inside first <div> element

#### .has() - Reduce the set of matched elements to those that have a descendant that matches the selector or DOM element.

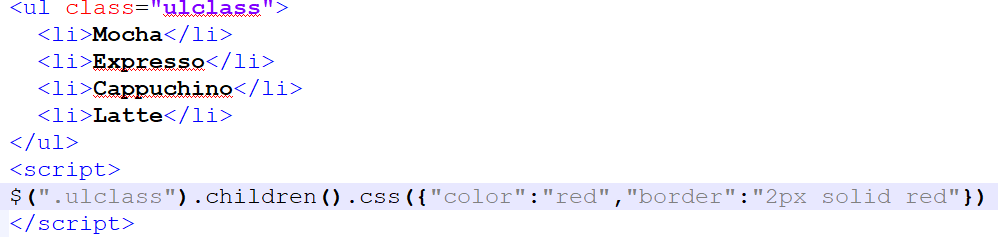
* + Eg : $("p").has("span") - Return all <p> elements that have <span> element inside

## jQuery Traversing -> Tree Traversal

#### .children() - Returns all direct children of the selected element

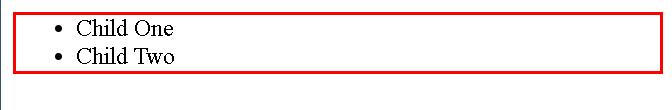
* + Eg : $("ul").children().css({"color":"red","border":"2px solid red"}) - Return elements that are direct children of

<ul>



* .find() - Returns descendant elements of the selected element
* .next() / prev() - Returns the next / previous sibling element of the selected element
* nextAll() - returns all next sibling elements of the selected element
* parent() - Returns the direct parent element of the selected element

<body>



<div class="great-grand-parent">

<div class="grand-parent">

<ul class="parent">

<li class="child-one">Child One</li>

<li class="child-two">Child Two</li>

</ul>

</div>

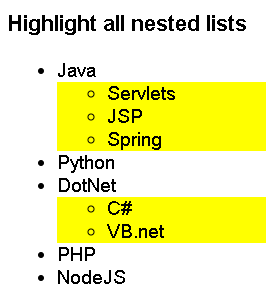
</div>

<script>

$( ".child-two" ).parent().css( "border", "2px solid red" );

</script>

<body>



<h3>Highlight all nested lists</h3>

<ul>

<li>Java

<ul>

<li>Servlets</li>

<li>JSP</li>

<li>Spring</li>

</ul>

</li>

<li>Python</li>

<li>DotNet

<ul>

<li>C#</li>

<li>VB.net</li>

</ul>

</li>

<li>PHP</li>

<li>NodeJS</li>

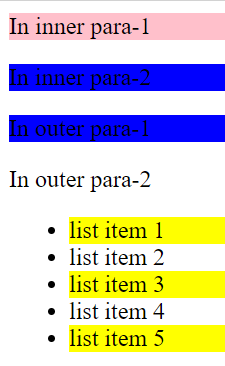
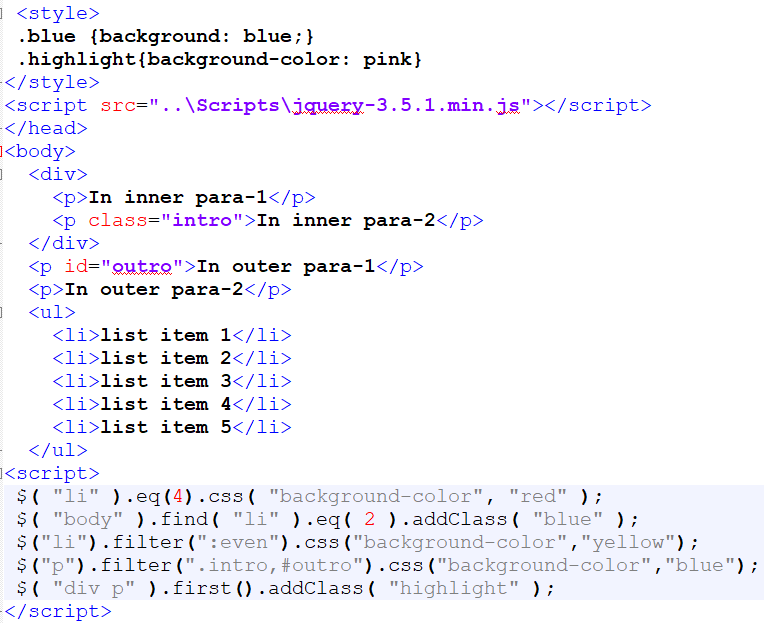
</ul>

<script>

* children() - returns all the direct children of the matched element.
* find() - returns all the descendant elements of the matched element.

$( "ul" ).find( "li ul" ).css( "background-color", "yellow" );

</script>



## Method Chaining

#### Chaining is a good way to avoid selecting elements more than once. Eg:

* + $("div").fadeOut();
  + $("div").css("color", "red");
  + $("div").text("hello world");

#### Instead of doing that and running $(“div”) three times, you could do this:

* + $("div").fadeOut().css("color", "red").text("hello world");

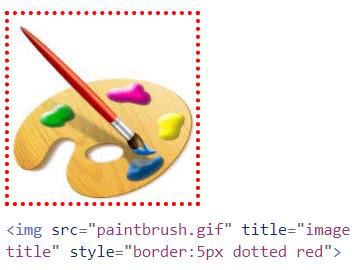
# ATTRIBUTES AND TEMPLATES

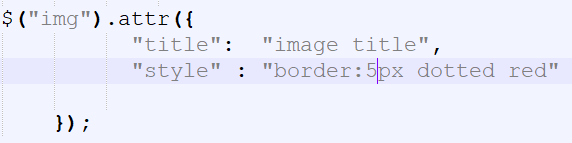
## Working with Attributes

#### Object attributes can be used using attr():

* + var val = $(‘#customDiv’).attr(‘title’); - Retrieves the title attribute value

#### .attr(attributeName,value) : accesses an object’s attributes and modifies the values.

* + $(‘img’).attr(‘title’,’Image title’); - changes the title attribute value to Image title.
* To modify multiple attributes, pass JSON object.



* You can also remove attributes entirely using .removeAttr().

## jQuery Templates

#### Template is a form, model, sample or predefined shape used to provide consistent look and feel to information presented on a website

* + jQuery Templates are client-side based templates.
  + JQuery templating let you render JSON to HTML elements.

#### The benefits of using jQuery Templates are:

* + Easily convert JSON object to HTML without need for parsing
  + Reusability
  + Rendered and cached on client-side
  + Templates are written in pure HTML with Template Tags and simple jQuery code for magic to happen
  + Maximize the separation of UI and DATA
  + jQuery Templates is an official plugin for the jQuery Library. We need to add reference to the jQuery

Templates library.

* + <script src=<http://ajax.microsoft.com/ajax/jquery.templates/beta1/jquery.tmpl.min.js> />

<script>

$(document).ready(function() { var studList = [

{ Name: "Anil", Surname: "Patil", Grade: 'A' },

{ Name: "Soha", Surname: "Kumari", Grade: 'B' },

{ Name: "Arnav", Surname: "Rao", Grade: 'A' },

{ Name: "Vanita", Surname:"Kapoor", Grade: 'B'}

];

$("#studTemplate").tmpl(studList).appendTo("#students");

});

</script>

<script id="studTemplate" type="text/html">

<tr><td>${Name}</td><td>${Surname}</td>

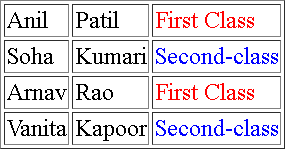
{{if Grade=='A'}}

(<td style='color:red'>First Class</td>)

{{else}}

(<td style='color:blue'>Second-class</td>)

{{/if}}

</tr>

</script>

<body>

<table border="1" id="students"></table>

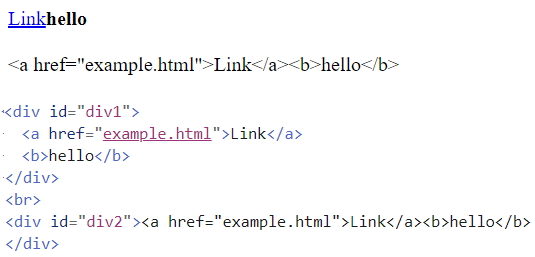
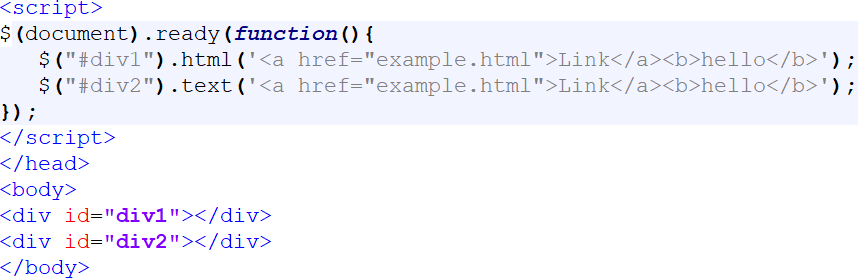
</body>

</html>

## Getting Content

* 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

$.html() treats the string as HTML, $.text() treats the content as text



## Adding and Removing Nodes

* In traditional approach adding and removing nodes is tedious.
* To insert nodes four methods available:
* Appending adds children at the end of the matching elements
  + .append()
  + .appendTo()

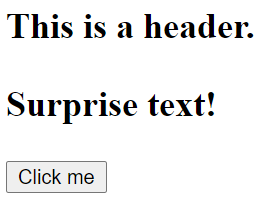
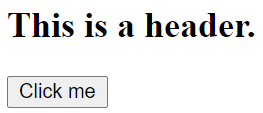
#### Prepending adds children at the beginning of the matching elements

* + .prepend()
  + .prependTo()

#### To wrap the elements use .wrap()

* + Eg: $(“p”).wrap(“<h1></h1>”) //wraps “p” in <h1> tags

#### To remove nodes from an element use .remove()



<script>

$(document).ready(function() {

$("button").click(function(){

$("h2").append("<p>Surprise text!</p>");

});

});

</script>

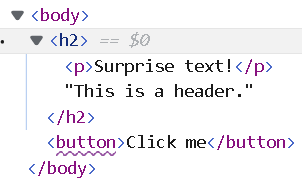
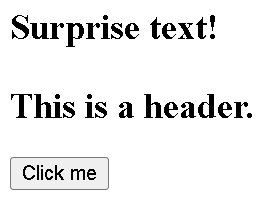
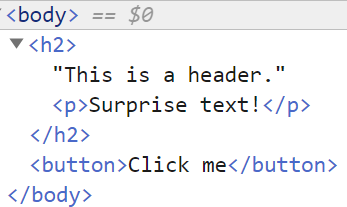
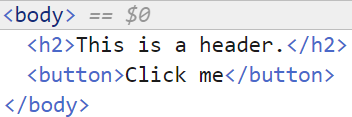
</head>

<body>

<h2>This is a header.</h2>

<button>Click me</button>

</body>



<script>

$(document).ready(function() {

$("button").click(function(){

$("h2").prepend("<p>Surprise text!</p>");

});

});

</script>

<script>



#### $(document).ready(function() {

$('button').click(function() {

//append can be a element, but can be simple text too

#### $('#hello').append(' world!');

});

});

</script>

</head>

<body>

#### <p id="hello">Hello</p>

<button>Click me</button>

</body>

<script>

$(document).ready(function() {

$("#btn").click(function(){

$("h2").remove();

});

});

</script>

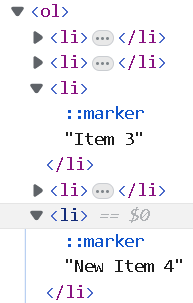
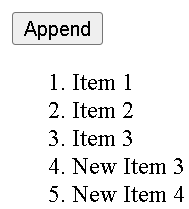
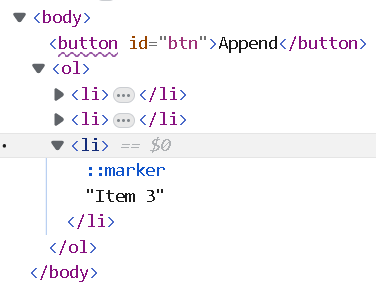
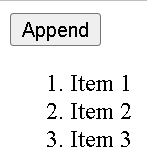
<body>

#### <button id="btn">Remove</button><br>

<h2>Press the button to remove this heading</h2>

</body>

<script type="text/javascript">



$(document).ready(function(){ var i=3;

$("#btn").click(function(){

$("ol").append("<li>New Item " + i++ + "</li>");

});

});

</script>

</head>

<body>

#### <button id="btn">Append</button>

<ol>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

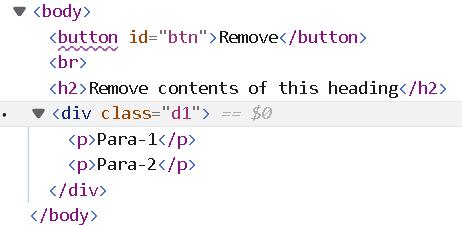
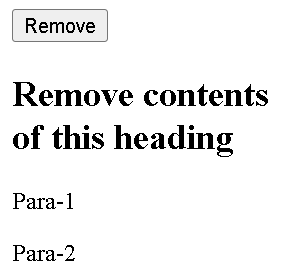
</ol>

</body>

|  |  |  |
| --- | --- | --- |
| <script>  $(document).ready(function() {  $("#btn").click(function(){  $("div").wrap($("<h3 style='text-align:center;color:blue'></h3>"));  }); | |  |
| });  </script>  <body>  <div>Hello</div>  <button id="btn">Wrap</button><br>  </body> |  |  |
|  | |  |



<script>



$(document).ready(function() {

$("#btn").click(function(){

$("h2").empty();

$(".d1").empty();

});

});

</script>

<body>

<button id="btn">Remove</button><br>

<h2>Remove contents of this heading</h2>

<div class="d1">

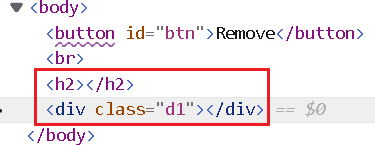
<p>Para-1</p>

<p>Para-2</p>

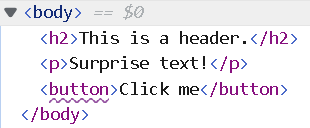
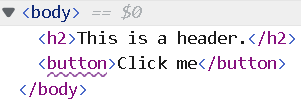
</div>

</body>

This method removes not only child (and other descendant) elements, but also any text within the set of matched elements.

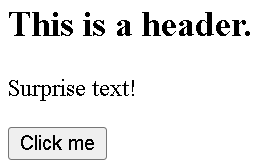


## Other DOM manipulation methods



#### Methods that allow us to insert new content outside an existing element:

* + .after() : Insert content after each element in the set of matched elements.
  + .before() : Insert content before each element in the set of matched elements.
  + .insertAfter() : Insert every element in the set of matched elements after the target.
  + .insertBefore() : Insert every element in the set of matched elements before the target.
  + .replaceAll() : Replace each target element with the set of matched elements.
  + .replaceWith() : Replace each element in the set of matched elements with the provided new content and return the set of elements that was removed.



<script>

$(document).ready(function() {

$("button").click(function(){

$("h2").after("<p>Surprise text!</p>

});

});

</script>

</head>

<body>

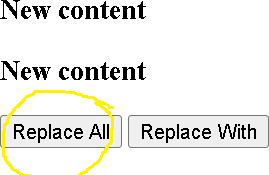
<h2>This is a header.</h2>

<button>Click me</button>

</body>

");

<script>



$(document).ready(function() {

$("#btn1").click(function(){

$("<h3>New content</h3>").replaceAll("p");

});

$("#btn2").click(function(){

$("p.p2").replaceWith("<h3>New heading</h3>");

});

});

</script>

</head>

<body>

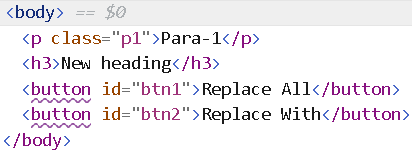
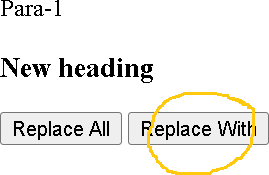
<p class="p1">Para-1</p>

<p class="p2">Para-2</p>

<button id="btn1">Replace All</button>

<button id="btn2">Replace With</button>

</body>



## Modifying Styles

#### .css() function is used to modify an object’s style

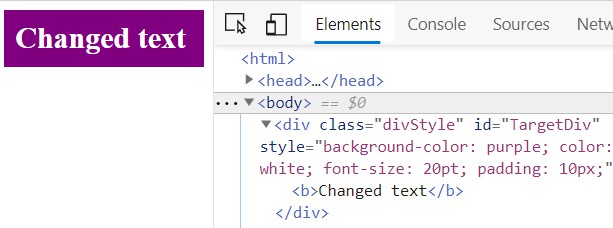
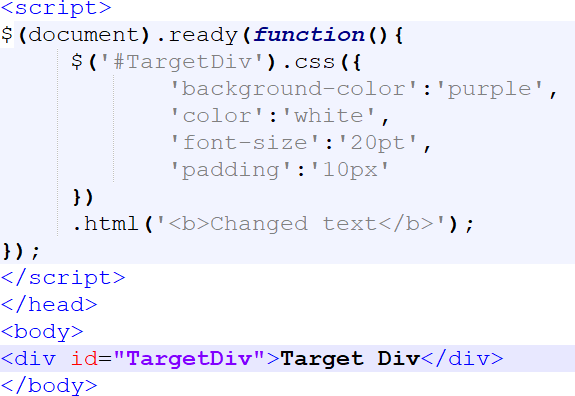
$('div').css({

“color”:”red”,

“font-weight”:”bold”

});

* + $('div').css('color','red');
* Multiple styles can be modified by passing a JSON Object



* The jQuery library defines a number of utility functions that are namespaced by jQuery/$

and that don’t operate on a jQuery object.

* Some of them are:
  + **$.each()** : is used to iterate over arrays and objects



$.each([ 33,44,55 ], function( index, value ) { console.log( index + ": " + value );

});

$.each(['Dia', 'Ria','Nia'], function( index, value ) { console.log( index + ": " + value );

});

var person = { "name": "shrilata",

"email": "[shri@gmail.com"](mailto:shri@gmail.com),

};

$.each( person, function( key, value ) { console.log( key + ": " + value ); });

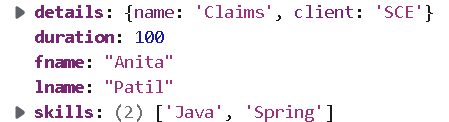
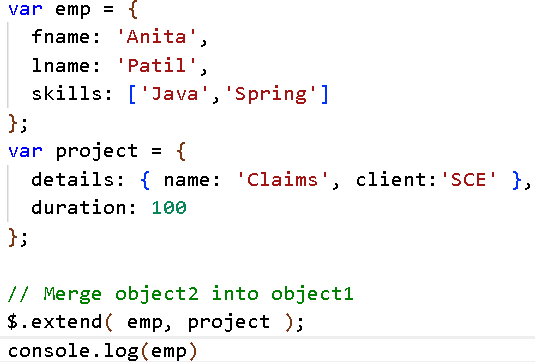
* + **$.isArray(** **obj** **)** : determine whether the argument is an array.
  + **$.inArray()** : is used to returns a value's index in an array, or -1 if the value is not in the array.

console.log($.isArray([1,2,3])) //true

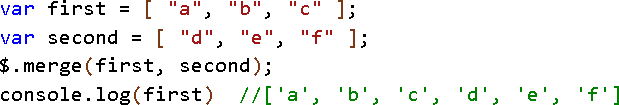
var arr = [22,33,44,55];

console.log($.inArray(44,arr)) //2

* **$.extend(** **target,** **object1** **[,** **objectN** **]** **)** : Merge the contents of two or more objects together into the first object.



* **$.merge(** **first,** **second** **)** **:** merges the contents of two arrays together into the first array.



## Working with Classes

#### The four methods for working with css class attributes are

* .addClass() : adds one or more classes to the class attribute of each element.
  + $(‘p’).addClass(‘classOne’);
  + $(‘p’).addClass(‘classOne classTwo’);

#### .hasClass() : returns true if the selected element has a matching class

* + if($('p').hasClass('classOne')) { //perform operation}

#### removeClass() remove one or more classes

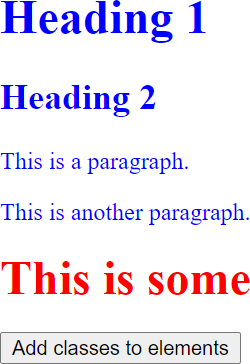
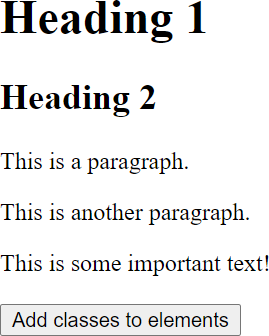
* + $(‘p’).removeClass(‘classOne classTwo’);

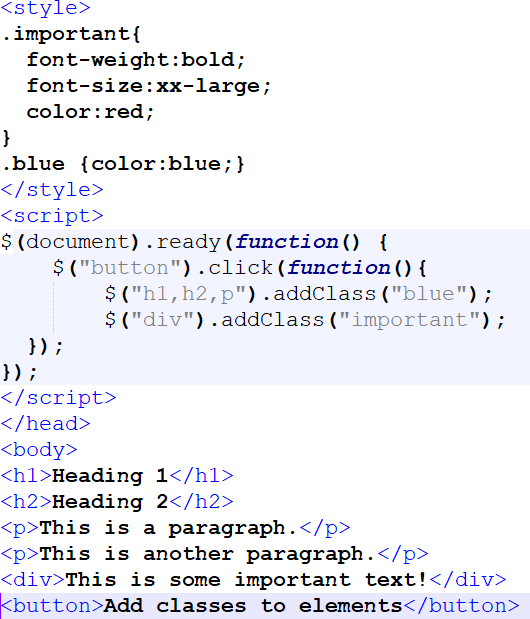
#### To remove all class attributes for the matching selector

* + $(‘p’).removeClass();

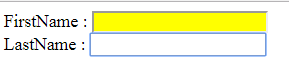
#### .toggleClass() : alternates adding or removing a class based on the current presence or absence of the class.

* + $(‘#targetDiv’).toggleClass('highlight');





## Working with Classes (Contd)



jQuery Event Model Benefits

#### Events notify a program that a user performed some type of action

* jQuery Events
  + click()

$("img").mouseover(function () {

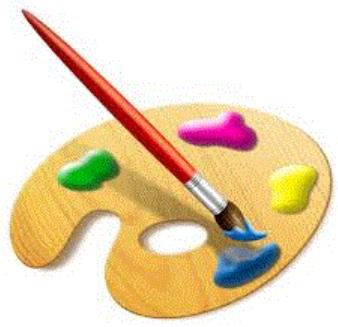
$(this).css("opacity", "0.3");

});

$("img").mouseout(function () {

$(this).css("opacity", "1.0");

});

* + blur()
  + focus()
  + dblclick()
  + mousedown()
  + mouseup()
  + mouseover()
  + keydown()
  + keypress()
  + hover() : hover() method takes two functions and is a combination of the mouseenter() and mouseleave() methods.

$("#p1").hover(function(){ alert("You entered p1!");

},

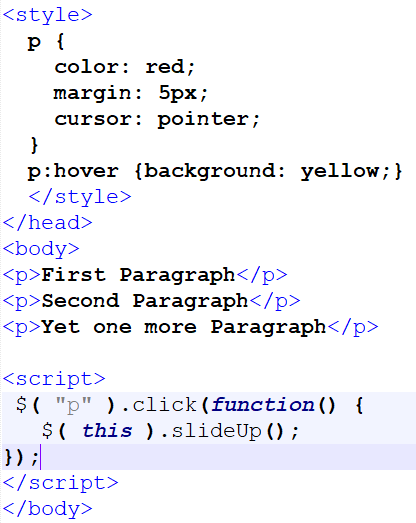
function(){

alert("Bye! You now leave p1!");

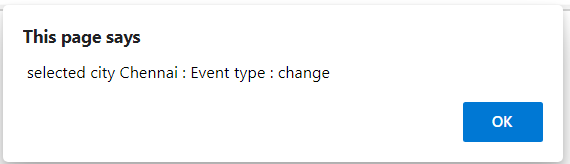
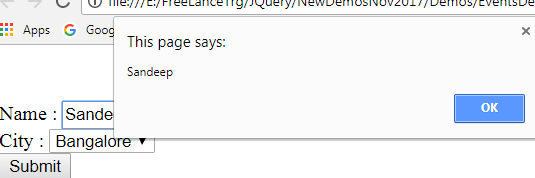
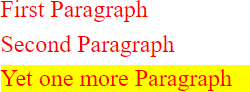
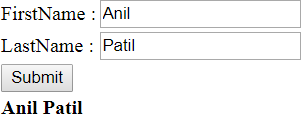
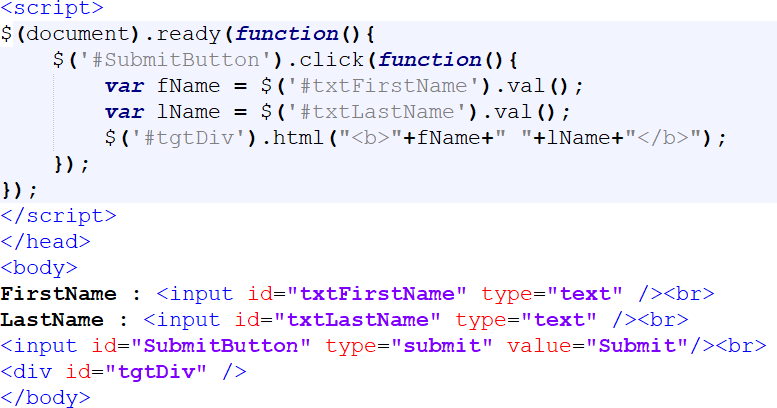
});

## Handling Click Events

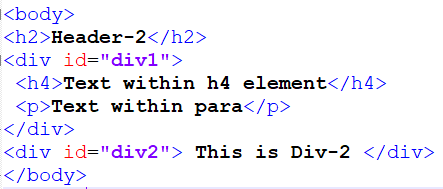
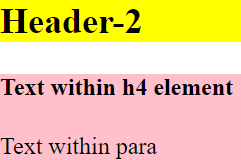
#### .click(handler([eventObject])) is used to listen for a click event or trigger a click event on an element



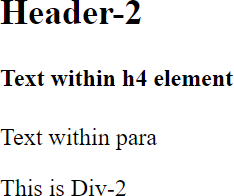
* + $(‘#submitButton’).click(function() { alert(‘Clicked’) });



## Using on() and off()



* The on() method attaches one or more event handlers for the selected elements.



$( "#dataTable tbody tr" ).on( "click", function() {

console.log( $( this ).text() );

});

# ANIMATIONS

## Showing and Hiding Elements

#### To set a duration and a callback function

* + - show(duration, callback)
    - duration is the amount of time taken (in milliseconds), and callback is a callback function jQuery will

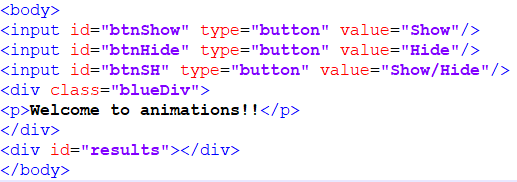
call when the transition is complete.

#### The corresponding version of hide( )

* + - hide(duration, callback)

#### To toggle an element from visible to invisible or the other way around with a specific speed and a callback function, use this form of toggle( )

* + - toggle(duration, callback)



## jQuery Sliding Effects

#### The jQuery slide methods slide elements up and down.

* jQuery has the following slide methods:
  + $(selector).slideDown(speed,callback) : *Display* *the* *matched* *elements* *with* *a* *sliding* *motion*
  + $(selector).slideUp(speed,callback) : *Hide* *the* *matched* *elements* *with* *a* *sliding* *motion.*
  + $(selector).slideToggle(speed,callback)
* The speed parameter can take the following values: "slow", "fast", "normal", or milliseconds.
* The callback parameter is the name of a function to be executed after the function

completes.

$("#flip").click(function(){

$("#panel").slideDown();

});

## jQuery Fading Effects

#### The jQuery fade methods gradually change the opacity for selected elements.

* jQuery has the following fade methods:
  + $(selector).fadeIn(speed,callback)
  + $(selector).fadeOut(speed,callback)
  + $(selector).fadeTo(speed,opacity,callback)

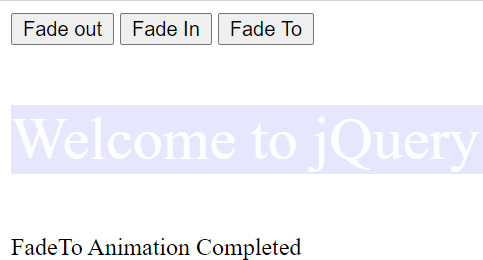
#### The speed parameter can take the following values: "slow", "fast", "normal", or milliseconds.

* + The opacity parameter in the fadeTo() method allows fading to a given opacity.
  + The callback parameter is the name of a function to be executed after the function completes.

$('.blueDiv').fadeTo(1000,0.1,function(){

$('#results').text('FadeTo Animation Completed');

});



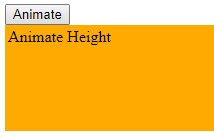
## Creating Custom Animation

#### Custom animation can be created in jQuery with the animate() function

* + animate(params, duration, callback)
    - params contains the properties of the object you’re animating, such as CSS properties, duration is the optional

time in milliseconds that the animation should take and callback is an optional callback function.

<style type="text/css">



#content { background-color:#ffaa00; width:300px; height:30px; padding:3px; }

</style>

<script type="text/javascript">

$(document).ready(function() {

$("#animate").click(function() {

$("#content").animate({"height": "100px", "width": "350px"}, "slow");

});

});

</script>

</head>

<body>

<input type="button" id="animate" value="Animate"/>

<div id="content">Animate Height</div> </body>

## Jquery plugins

#### A jQuery plugin is a method that we use to extend jquery’s prototype object.

* + It is a piece of code written in a JavaScript file that enables all jQuery objects to inherit any methods that need to be added.

#### Plugin is used to work with a collection of elements.

* + In fact, we can consider each method that comes with jQuery core as a plugin, eg .css(), .hide(),

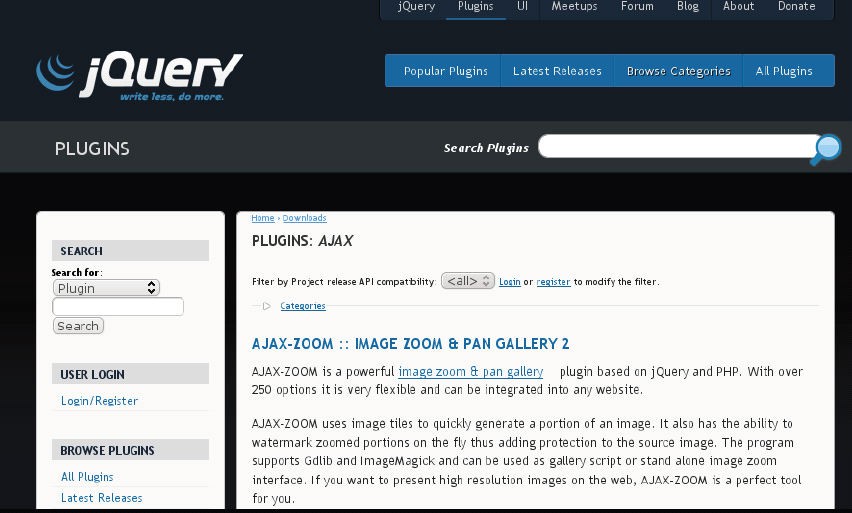
.fadeout(), .addClass() etc

* There are a large number of ready to use plugins available which you can download from repository link at <https://plugins.jquery.com/tag/jquery/>
  + This is official jQuery plug-in repository, where jQuery plug-in developers submit their plug-ins.
  + We can look for the best plug-ins from this repository.

#### We can create our own custom plugins; jQuery comes with all the necessary hooks to build plugins easily

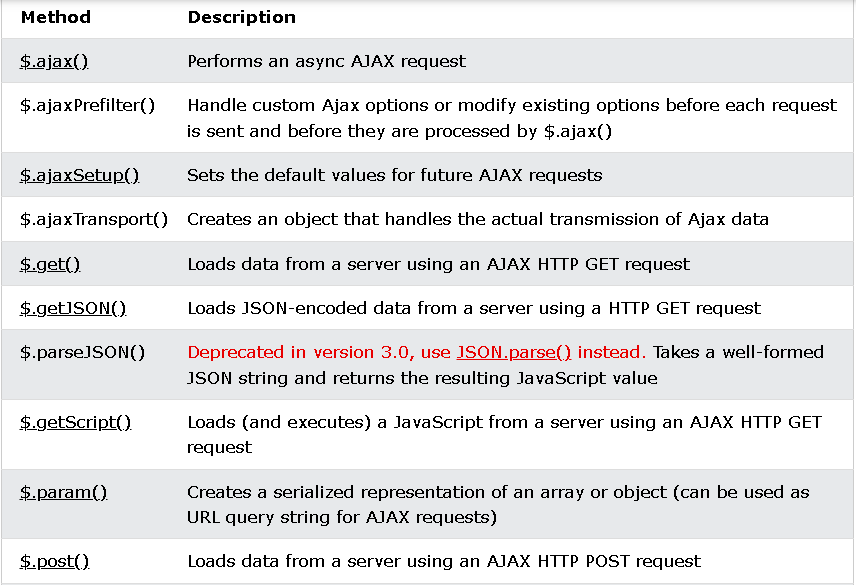
* + A plug-in is piece of code written in a standard JavaScript file.
  + These files provide useful jQuery methods which can be used along with jQuery library methods.

## plugins.jquery.com



jQuery Ajax features

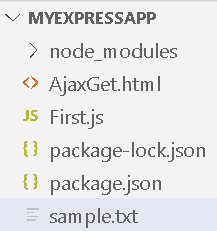
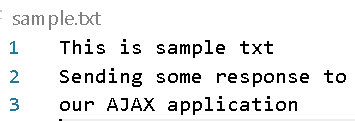
* jQuery provides several functions that can be used to send and receive data



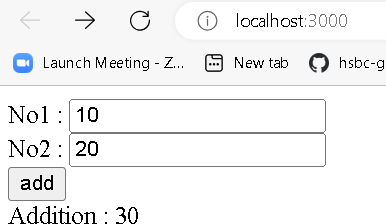
## Using get()

#### $.get( url [, data ] [, success ] [, dataType ] ) : retrieve data from a server using GET

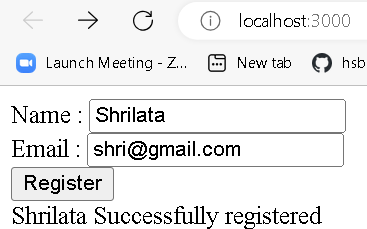
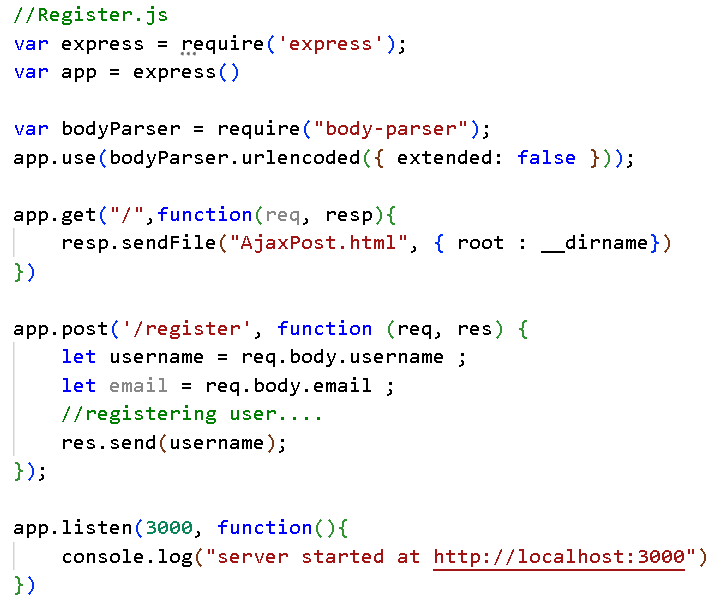
* + datatype can be html, xml, json



$.get( url [, data ] [, success ] [, dataType ] )



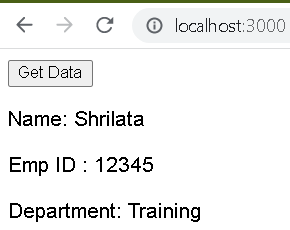
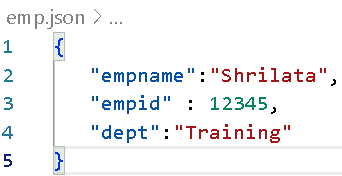
## Using post()



* $.post( url [, data ] [, success ] [, dataType ]) : Send data to the server using a HTTP POST request.

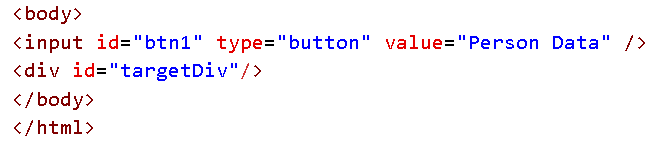
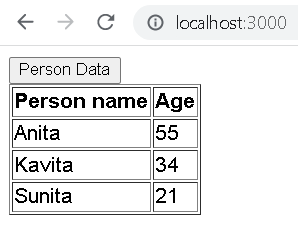
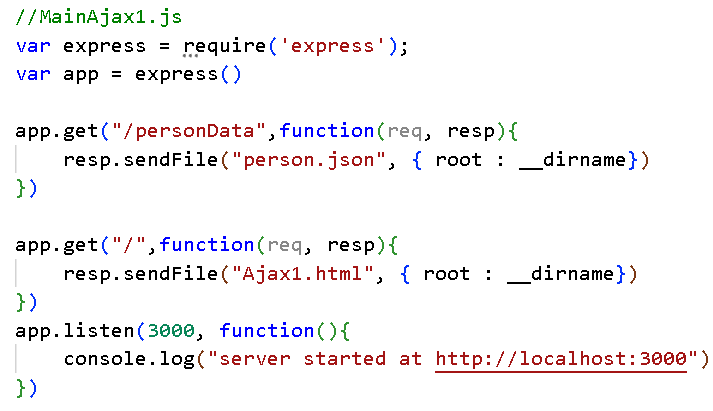
## Getting JSON data

#### $.getJSON( url [, data ] [, success ] ) can retrieve data from a server

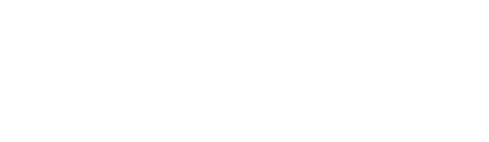
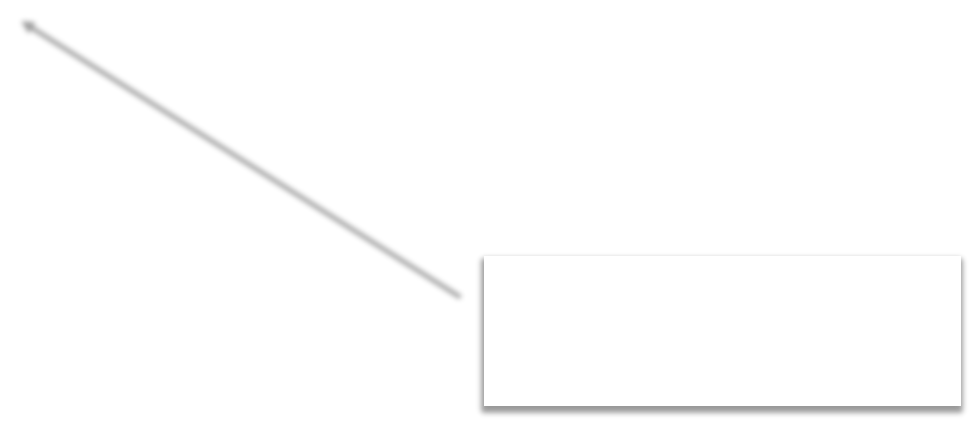
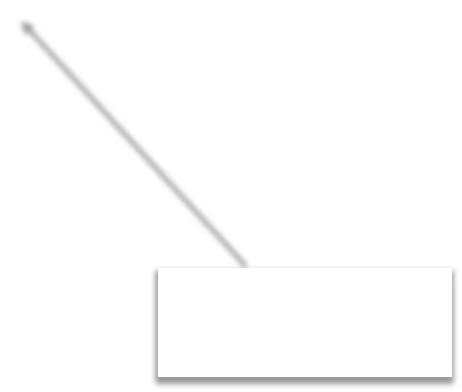
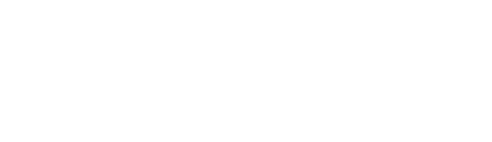
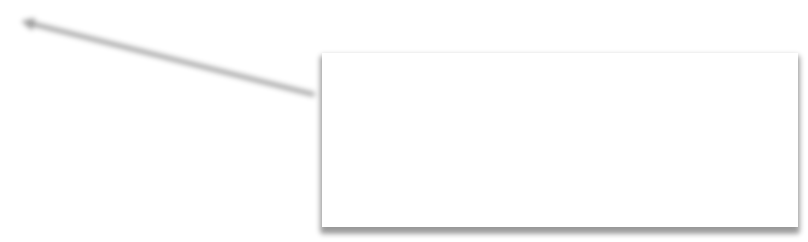
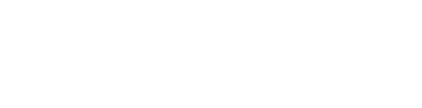
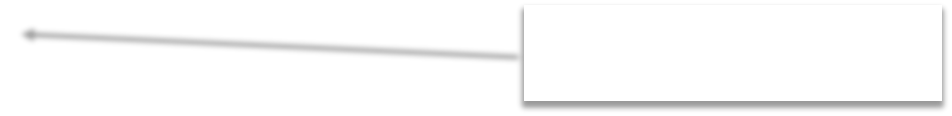
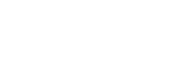
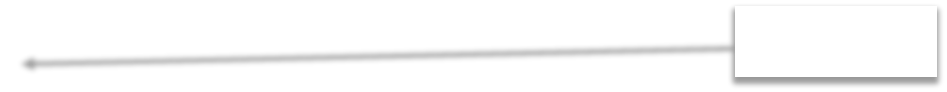
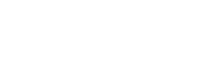
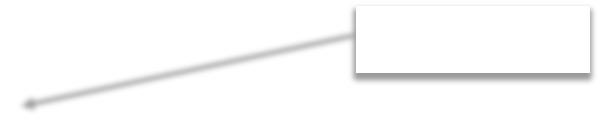


Using ajax() function • ajax() function is configured by assigning

values to JSON properties



### jQuery.ajax({



type: "GET|POST|DELETE|PUT",

url: url,

data: ‘{param:”value”}’, dataType:"text|html|json|jsonp|script|xml“, complete: function(){…..},

success: success\_callback, error: error\_callback

Request Type Page URL

Data to be sent in the body of the request.

Specifies the type of data expected in the response and how that data should be processed by jQuery.

});

Callback function representing success

function to be called when the request finishes (after success and error callbacks are executed)

## Loading HTML content from server

#### $(selector).load(url,data,callback) allows HTML content to be loaded from a server and added into DOM object.

* + $("#targetDiv").load('GetContents.html');

#### A selector can be added after the URL to filter the content that is returned from the calling load().

* + $("#targetDiv").load('GetContents.html #Main');

#### Data can be passed to the server using load(url,data)

* + $('#targetDiv').load('Add.aspx',{firstNumber:5,secondNumber:10})
* load () can be passed a callback function

$(‘#targetDiv’).load(‘Notfound.html’, function (res,status,xhr) { if (status == “errror”) {

alert(xhr.statusText);

}

});

## Using get(), getJSON() & post()

#### $.get(url,data,callback,datatype) can retrieve data from a server.

* + datatype can be html, xml, json

$.get('GetContents.html',function(data){

$('#targetDiv').html(data);

},'html');

#### $.getJSON(url,data,callback) can retrieve data from a server.

$.getJSON('GetContents.aspx,{id:5},function(data){

$('#targetDiv').html(data);

* $.pos}t)(;url,data,callback,datatype) can post data to a server and retrieve results.

## Using ajax() function

* ajax() function is configured by assigning values to JSON properties

$.ajax({

url: “employee.asmx/GetEmployees",

data : null,

contentType: "application/json; charset=utf-8",

datatype: ‘json’,

success: function(data,status,xhr){

//Perform success operation

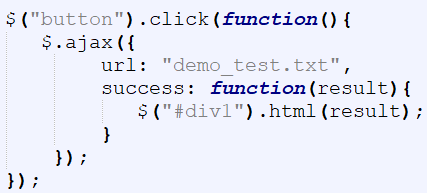
},

error: function(xhr,status,error) {

//show error details

}

});



# BOOTSTRAP 5

Pre-reqs:

* HTML
* CSS
* JavaScript

## Introduction

#### Bootstrap is an opensource frontend framework developed by Twitter.

* + It is the most popular ***HTML,*** ***CSS,*** ***and*** ***JavaScript*** framework for developing responsive, mobile first web sites.
  + Bootstrap is a free and open source collection of tools for creating websites and web applications.
  + Bootstrap contains a set of CSS- and HTML-based templates for styling forms, elements, buttons, navigation, typography, and a range of other UI components.
  + It also comes with optional JavaScript plugins to add interactivity to components.
* Bootstrap is promoted as being **One** **framework,** **every** **device**.
  + This is because websites built with Bootstrap will automatically scale between devices — whether the device is a mobile phone, tablet, laptop, desktop computer, screen reader, etc.

#### Responsive web design is about creating web sites which automatically adjust themselves

to look good on all devices, from small phones to large desktops.

* + Developers can then create a single design that works on any kind of device: mobiles, tablets, smart TVs, and PCs

## Where to Get Bootstrap 5?

#### There are three ways to start using Bootstrap 5 on your own web site.

* + Download Bootstrap 5 from getbootstrap.com : [https://getbootstrap.com/docs/5.0/getting-](https://getbootstrap.com/docs/5.0/getting-started/download/)

[started/download/](https://getbootstrap.com/docs/5.0/getting-started/download/)

* + Using CDN (Content Delivery Network).
  + The <https://getbootstrap.com/docs/5.0/getting-started/introduction/> page gives CDN links for CSS and js files
  + Install bootstrap vis package managers like NPM or YARN

<link href=["https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css](https://cdn.jsdelivr.net/npm/bootstrap%405.0.2/dist/css/bootstrap.min.css)" rel="stylesheet"

integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"

crossorigin="anonymous">

<script src=["https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"](https://cdn.jsdelivr.net/npm/bootstrap%405.0.2/dist/js/bootstrap.bundle.min.js) integrity="sha384- MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"

crossorigin="anonymous">

</script>

The attributes integrity and crossorigin have been added to CDN links to implement Subresource Integrity (SRI). It is a security feature that enables you to mitigate the risk of attacks originating from compromised CDNs, by ensuring that the files your website fetches (from a CDN or anywhere) have been delivered without unexpected or malicious modifications. It works by allowing you to provide a cryptographic hash that a fetched file must match.

## Create First Web Page With Bootstrap 5

#### Bootstrap 5 requires a containing element to wrap site contents.

* + There are two container classes to choose from:
    - The .container class provides a responsive fixed width container
    - The .container-fluid class provides a full width container, spanning the entire width of the viewport

To ensure proper rendering and touch zooming, add this <meta> tag



* + - * 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 part sets the initial zoom level when the page is first loaded by the browser.

## Bootstrap Container

#### Bootstrap container is basically used in order to create a centered area that lies within the page and generally deals with the margin of the content and the behaviors that are responsible for the layout.

* + It contains the grid system (row elements, which in turn are the container of columns).

#### There are two container classes in Bootstrap:

* + .container: provides a fixed width container with responsiveness. It will not take the complete width of its viewport.
  + .container-fluid: provides a full width container of the viewport and its width will change (expand or

shrink) on different screen sizes.



<body>

<div class="container">

<h1>Container</h1>

</div>

</body>

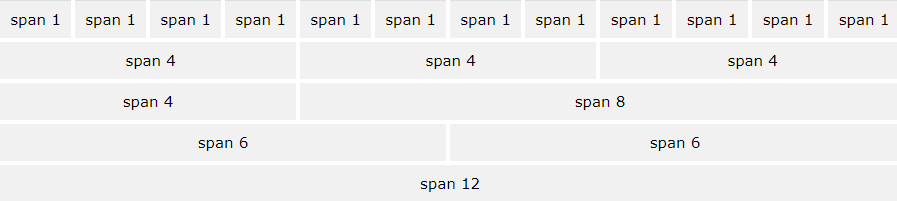
## Bootstrap Grid System

#### Bootstrap **grid** **system** allows to properly house the website's content.

* + Grid system divides the screen into columns―up to 12 in each row. (rows are infinite) ; that can be

used to create various types of layouts.

* + The column widths vary according to the size of screen they're displayed in.
  + Hence, Bootstrap's grid system is **responsive**, as the columns resize themselves dynamically when the size of browser window changes.
  + If you do not want to use all 12 columns individually, you can group the columns together to create wider columns



If the sum of the cols in your row doesn’t get to 12, then they don’t fill the whole row. And if it goes beyond

12 then it will move to the next line (it will only display the sum of the first elements <=12 on the first line).

## Building a Basic Grid

#### Recommended: place all the rows and columns inside a container to ensure proper alignment and padding.

* + Container creates a fixed width container in the browser window, while container-fluid creates a full-width fluid container.
  + Hence, it is a good practice to wrap all the contents within a container.
  + To create a container in our HTML page: <div class="container">…</div>
  + Next, create a row inside a container, then start creating the columns.
  + Bootstrap has a class row for creating rows:

<div class="container">

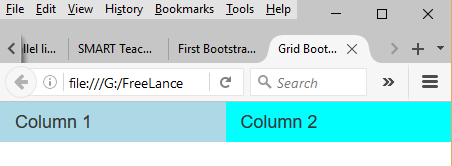
<div class="row">

//add desired number of cols here

</div>

</div>

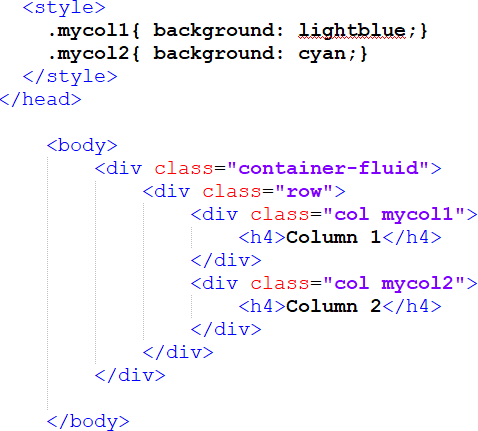
## Grid Classes



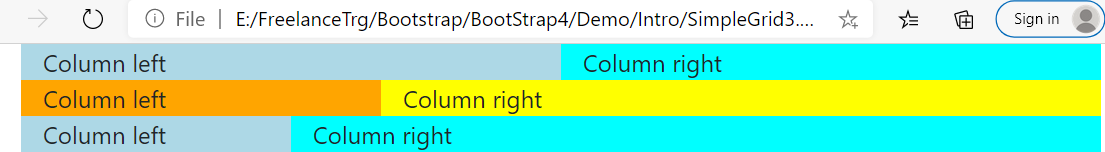
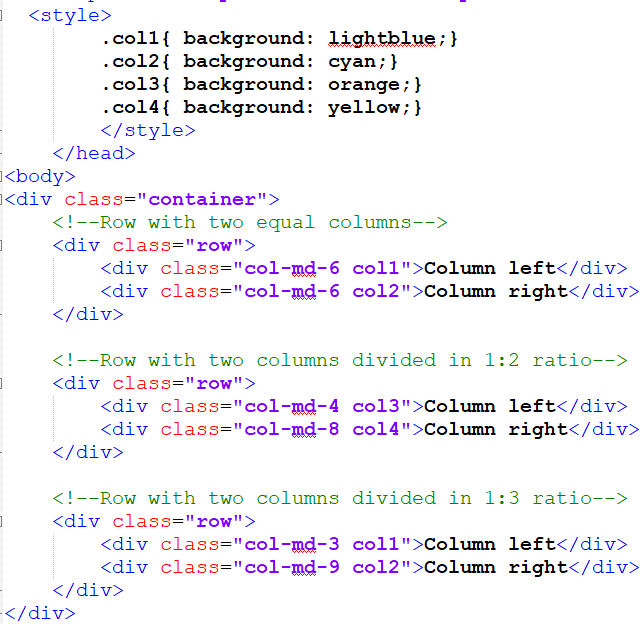
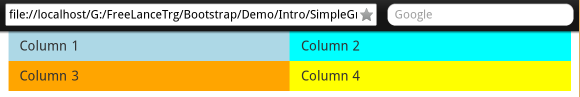
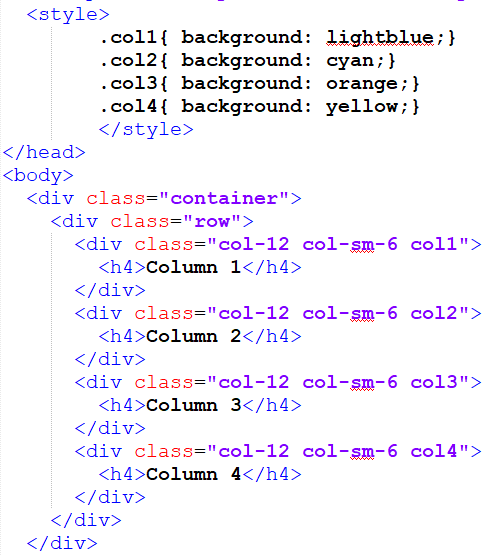
#### The Bootstrap 5 grid system has 6 classes:

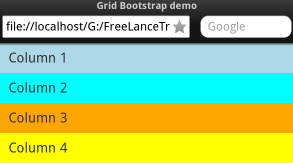
* + .col-xs (extra small devices - screen width less than 576px)
  + .col-sm- (small devices - screen width equal to or greater than 576px)
  + .col-md- (medium devices - screen width equal to or greater than 768px)
  + .col-lg- (large devices - screen width equal to or greater than 992px)
  + .col-xl- (xlarge devices - screen width equal to or greater than 1200px)
  + .col-xxl(Extra extra large- screen width equal to or greater than 1400px)

#### Example



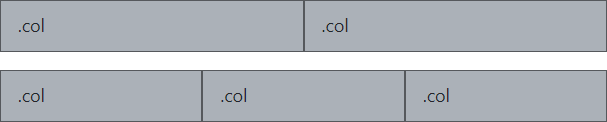
Building a Basic Grid





## Bootstrap Auto-layout Columns

* You can create equal width columns for all devices (xs, sm, md, lg, and xl) through simply using the class .col, without specifying any column number.





# MISC COMPONENTS

## Bootstrap Typography

#### Typography ? The style and appearance of printed matter.

* + Typography refers to the various styles present in Bootstrap style sheets which define how various text elements will appear on the web page.
  + It shows how certain text elements are rendered when we use Bootstrap without including the style classes.
  + Typography includes colors, fonts, headings, text alignment, background color, etc

#### HTML uses default font and style to create headings, paragraphs, lists and other inline elements.

* + Bootstrap overrides default and provides consistent styling across browsers for common typographic elements.
  + Eg, Bootstrap provides its own style for all six standard heading levels
  + Bootstrap 5 uses a default font-size of 16px, and its line-height is 1.5.



## Typography

#### Bootstrap 5 styles HTML headings (<h1> to <h6>) with a bolder font-weight and an increased font-size

<h1>h1. Bootstrap heading</h1>

<h2>h2. Bootstrap heading</h2>

<h3>h3. Bootstrap heading</h3>

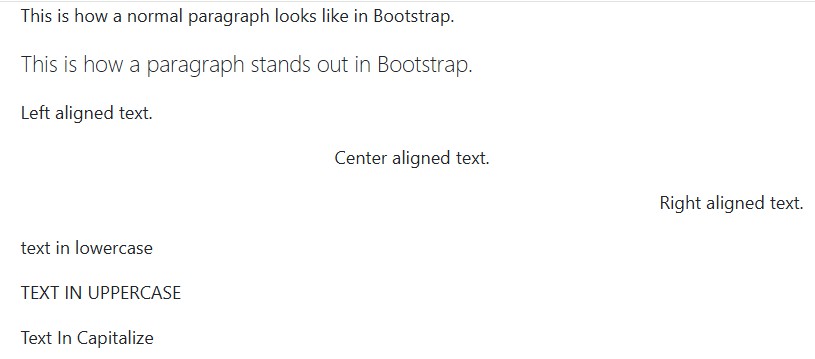
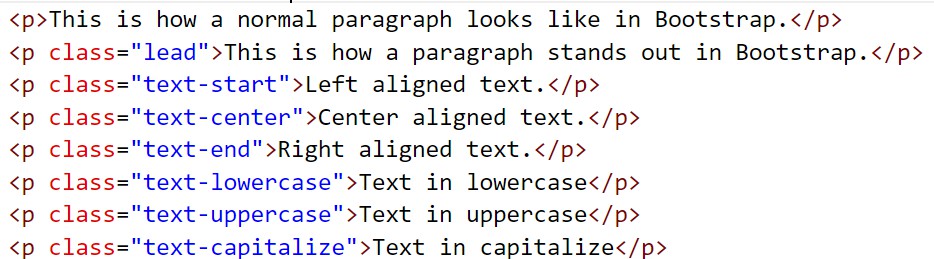
<h4>h4. Bootstrap heading</h4>

<h5>h5. Bootstrap heading</h5>

<h6>h6. Bootstrap heading</h6>



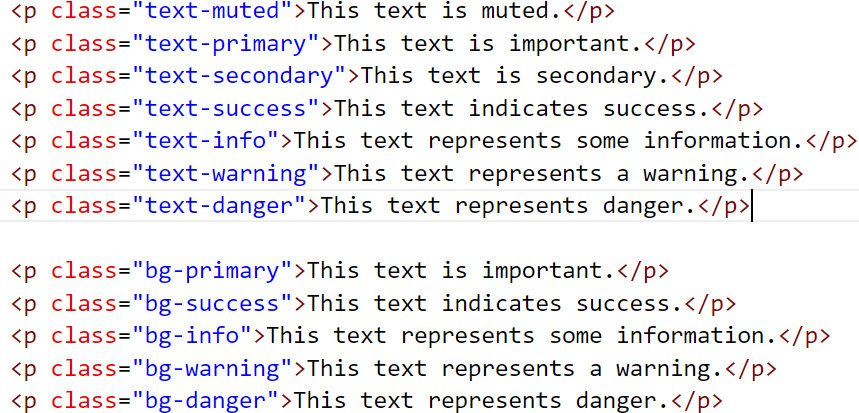
* You can make a content stand out by adding one of several classes.
* You can also transform the text to lowercase, uppercase or make them capitalize.



## Working with content : Text Coloring

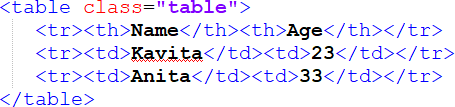
* Colors are the powerful method of conveying important information in website design.
* Bootstrap has handful of emphasis utility classes that can be used for this purpose such as

showing success message in green color, warning or error message in red color, etc.

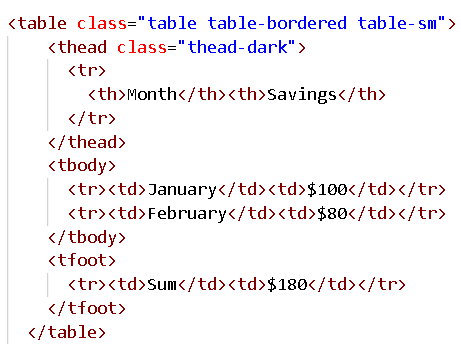
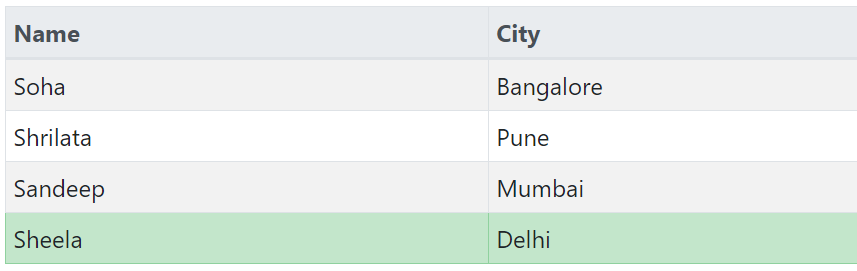
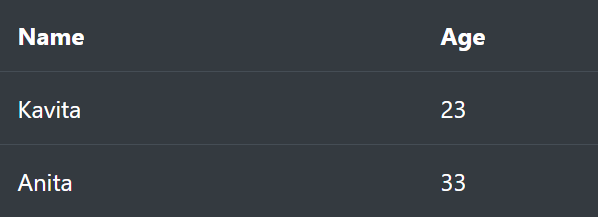
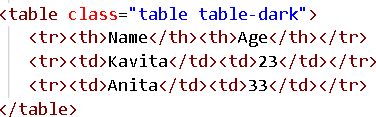


* Bootstrap provides an efficient layout to build elegant tables
  + You can create tables with basic styling that has horizontal dividers and small cell padding, by just

adding the Bootstrap's class .table to the <table> element.



* + The .table-striped class adds zebra-stripes to a table
  + The .table-bordered class adds borders on all sides of the table and cells
  + The .table-sm class makes a table more compact by cutting cell padding in half
  + The .table-dark class create inverted version of this table, i.e. table with light text on dark backgrounds

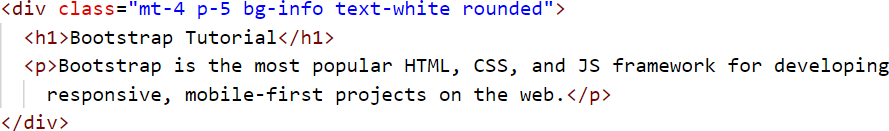


#### A jumbotron indicates a big box for calling extra attention to some special content or information.

* + Jumbotrons are no longer supported in Bootstrap 5. However, you can use a <div> element and add

special helper classes together with a color class to achieve the same effect

* + Tip: Inside a jumbotron you can put nearly any valid HTML, including other Bootstrap elements/classes.



## images

#### To add images on the webpage use element <img> , it has **three** classes to apply simple

styles to images.

* + .rounded: This adds rounded corners to our image.
  + .rounded-circle: This turns our images into a circle.
  + .img-thumbnail: This adds borders to our image thus shaping it into a thumbnail.

#### Alignment : These classes decide the floating attribute of images:

* + .float-end: This floats our image to the right.
  + .float-start: This floats it to the left.
  + Center an image by adding the utility classes .mx-auto (margin:auto) and .d-block (display:block) to the image

<img src="taj.jpg" class="rounded-circle">

<img src="taj.jpg" class="rounded">

<img src="taj.jpg" class="img-thumbnail">

<img src="taj.jpg" class="float-end rounded-circle">

<p>… </p>





# BOOTSTRAP ICONS

## Bootstrap Icons

#### Bootstrap 5 has released a new SVG icon library which was undertaken by the co-founder of Bootstrap Mark Otto.

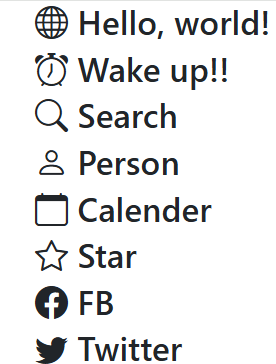
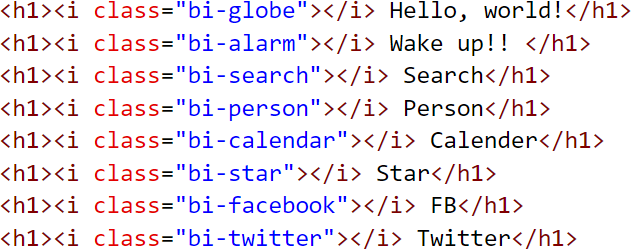
<https://icons.getbootstrap.com/>

#### Bootstrap Icons are SVGs, so you can include them into your HTML in a few ways depending on how your project is setup.

* The simplest way to include Bootstrap icons in a web page is using the CDN link.
  + This link points to a CSS file that has all the necessary classes to generate font icons.



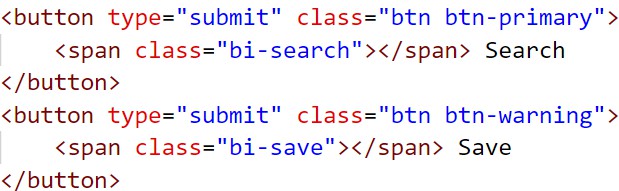
* + To use Bootstrap icons in your code you'll require an <i> tag with an individual icon class **.bi-\*** applied on it. The general syntax for using Bootstrap icons is:
  + <i class="bi-class-name"></i> Where class-name is the name of the icon class, e.g. search, person



## Using Icons in Bootstrap 5

* Placing icons inside buttons:





* Similarly, you can place icons inside the navs, forms, tables, paragraphs or anywhere

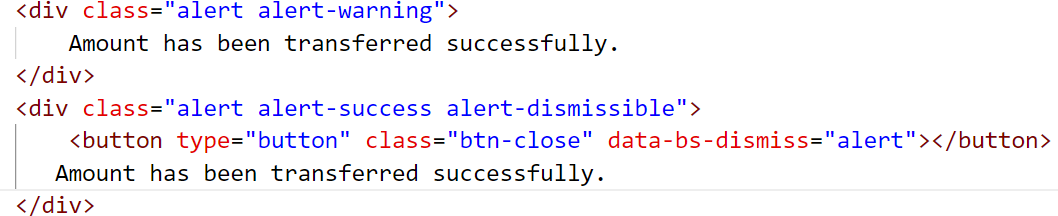
you want

## Alerts

#### Bootstrap comes with a very useful component for displaying alert messages in various sections of our website

* + You can use them for displaying a success message, a warning message etc
  + These messages can be annoying to visitors, hence they should have dismiss functionality added to give visitors the ability to hide them.
  + Alerts are created with the **.alert** class, followed by one of the contextual classes like .alert-success,

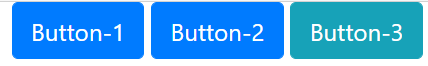
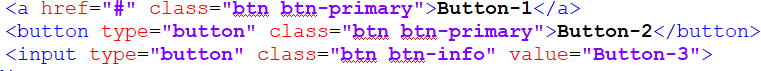
.alert-info



contextual classes for alert messages:

alert-success alert-info alert-danger alert-warning

#### Its easy to convert an a, button, or input element into a fancy bold button in Bootstrap; just



have to add the btn class

* You can also create outline buttons by replacing the button modifier classes
* Buttons come in various color options:
  + btn-primary for dark blue



* + btn-secondary for dark gray
  + btn-success for green
  + btn-info for teal blue



<button type="button" class="btn btn-primary btn-lg"> Large</button>

<button type="button" class="btn btn-primary"> Default</button>

<button type="button" class="btn btn-primary btn-sm btn-success"> Small</button>

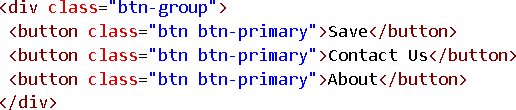
* + btn-warning for orange
  + btn-danger for red
  + btn-dark for dark
  + And in various sizes:

o btn-lg for large buttons

o btn-sm for small buttons

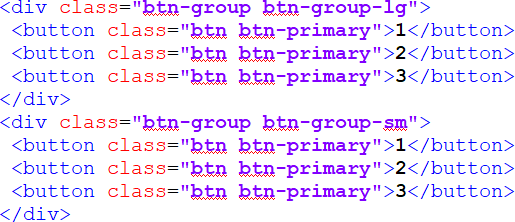
* Button groups allow multiple buttons to be stacked together
  + To create a button groups just wrap a series of buttons with .btn class in a <div> element and apply the class .btn-group on it





* + Use the class .btn-group-vertical to create a vertical button group
  + <div class="btn-group-vertical">
  + Use class .btn-group-lg for a large button group or .btn-group-sm for a small button group





<span class="badge bg-primary">Primary</span>

<span class="badge bg-secondary">Secondary</span>

<span class="badge bg-success">Success</span>

<span class="badge bg-danger">Danger</span>

#### Badges are generally used to indicate some valuable information on the web pages such as important heading, warning messages, notification counter, etc.

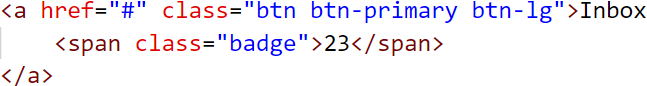


* + They are mostly used to display numbers such as unread items, notifications,

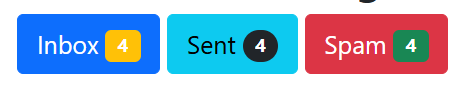
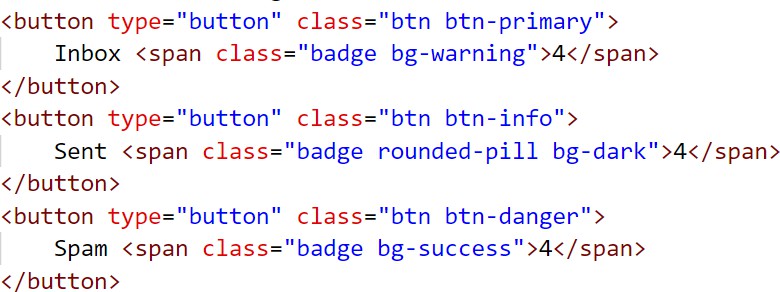
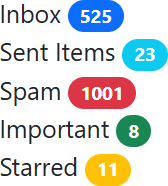
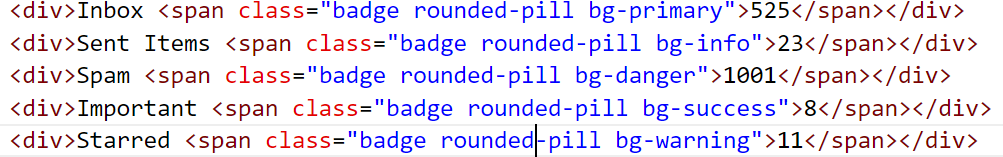
and so on, rather than text

* + Use the .badge class together with a contextual class (like .bg-secondary) within <span> elements to create badges.
  + Badges are **self-collapsing** components, ie when there's no content inside the badge it will not be visible on the website





**.rounded-pill**: for creating rounded badges



## Progress bar

#### A progress bar is a visual representation to show the progress of a task or an operation.

* + Used in many web applications to show the progress of action to users.
  + Bootstrap offers different progress bars, depending upon multiple parameters like color, height, shape, or

style.

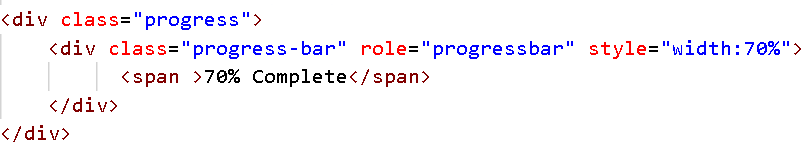
* + To create a default progress bar, use the **.progress** class with a **.progress-bar** nested inside
  + To create progress bars with different colors and styles, use utility (contextual) classes

|  |  |  |
| --- | --- | --- |
| <div class="progress">  <div class="progress-bar" style="width:70%"></div> | |  |
| <div> |  |  |
|  |  | |

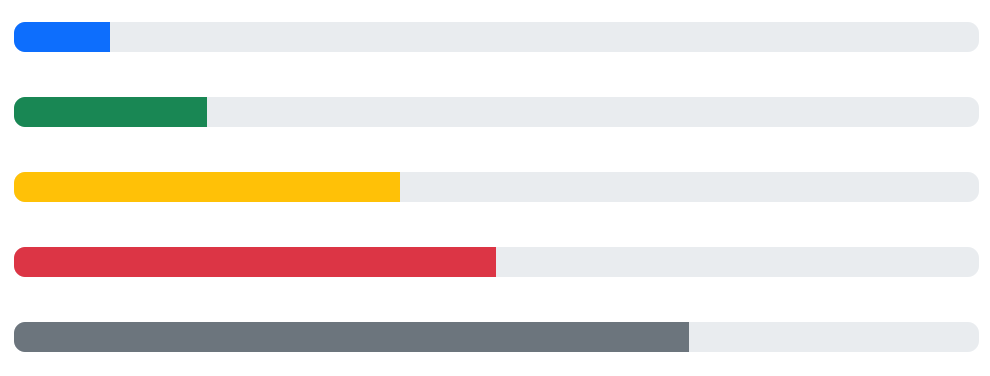
#### You can include a label that displays a percentage value on the progress bar (to explicitly state, in percentage terms, the completeness of the task).

* + To display a label on the progress bar, insert text as contents of the <div> element that has the

.progress-bar class applied.



<div class="progress"> <!-- Blue -->



<div class="progress-bar" style="width:10%"></div>

</div>

<div class="progress"> <!-- Green -->

<div class="progress-bar bg-success" style="width:20%"></div>

</div>

<div class="progress"> <!-- Orange -->

<div class="progress-bar bg-warning" style="width:40%"></div>

</div>

<div class="progress"> <!-- Red -->

<div class="progress-bar bg-danger" style="width:50%"></div>

</div>

<div class="progress"> <!-- Grey -->

<div class="progress-bar bg-secondary" style="width:70%"></div>

</div>

## Progress bar

Pagination

#### Pagination are those numbered links we find either below a webpage or on top of a webpage.

* + They are used to select pages between pages of a website.

#### Pagination is used to enable navigation between pages in a website.

* + The pagination used in Bootstrap has a large block of connected links that are hard to miss and are easily scalable.

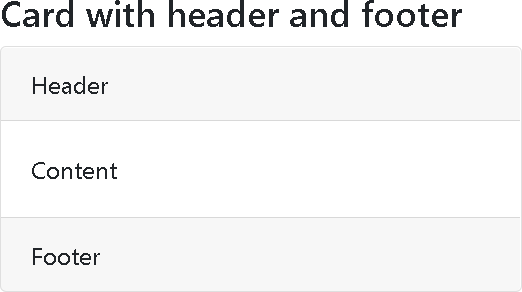
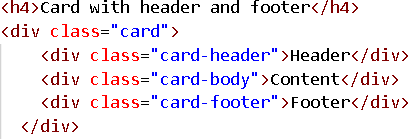
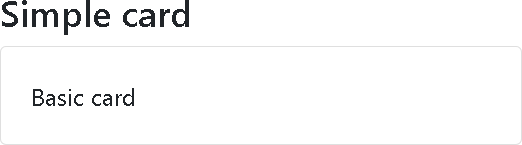
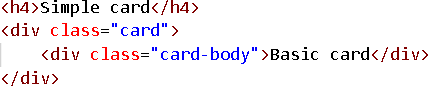
#### To create a basic pagination, add the .pagination class to an <ul> element.

* Then add the .page-item to each <li> element and a .page-link class to each link inside <li>
  + The .pagination class is used to specify pagination on a list group.
  + The .page-item class is used to specify each pagination item in the group.
  + The .page-link class is used to specify the link in the pagination item.

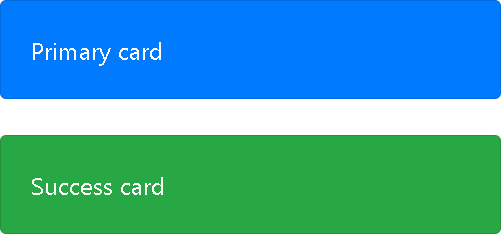
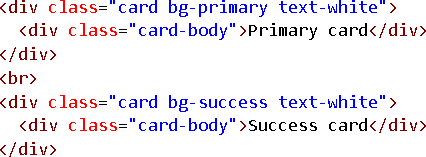
|  |  |  |
| --- | --- | --- |
|  |  | |
| <div class="container">  <h2>Pagination</h2>  <p>The .pagination class provides pagination links:</p>  <ul class="pagination"> |  |  |
| <li class="page-item"><a href="#" class="page-link">1</a></li>  <li class="page-item"><a href="#" class="page-link">2</a></li>  <li class="page-item"><a href="#" class="page-link">3</a></li>  <li class="page-item"><a href="#" class="page-link">4</a></li>  <li class="page-item"><a href="#" class="page-link">5</a></li>  </ul>  </div> | |  |

#### A card in bootstrap is a bordered box with some padding around its content

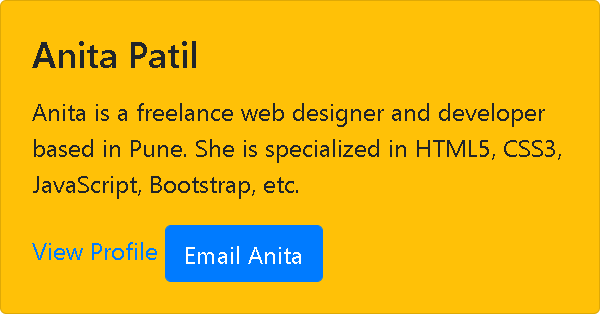
* + It includes options for headers and footers, a wide variety of content, contextual background colors, and powerful display options.
  + If you’re familiar with Bootstrap 3, cards replace old panels, wells, and thumbnails.
  + Cards support a wide variety of content, including text, list groups, links, and more within a box with rounded corners
  + The card div also can have optional parts like header, footer etc.
  + A basic card is created with the .card class. Content inside the card has a .card-body class

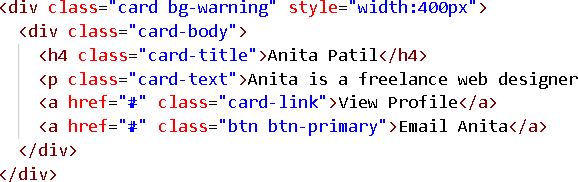


#### To add a background color the card, use contextual classes (.bg-primary, .bg-success, .bg- info, .bg-warning, .bg-danger, .bg-secondary, .bg-dark and .bg-light.



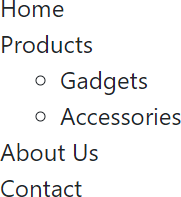
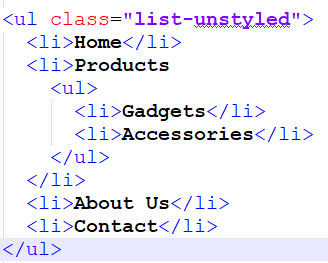
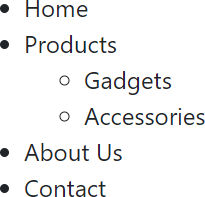
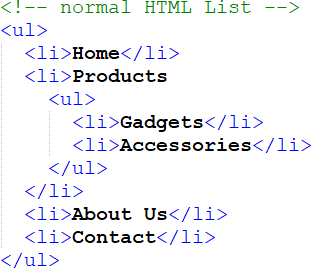
* Titles, text, and links
  + Use .card-title to add card titles to any heading element.
  + The .card-text class is used to remove bottom margins for a <p> element if it is the last child (or the only one) inside .card-body.
  + The .card-link class adds a blue color to any link, and a hover effect.





#### Unstyled Ordered and Unordered Lists

* + Sometimes you might need to remove the default styling form the list items. You can do this by simply applying the class .list-unstyled to the respective <ul> or <ol> elements



#### If you want to create a horizontal menu using ordered or unordered list you need to place all

list items in a single line i.e. side by side.

* + You can do this by simply applying the class .list-inline to the respective <ul> or <ol>, and the class

.list-inline-item to the <li> elements.

<ul class="list-inline">

<li class="list-inline-item">Home</li>

<li class="list-inline-item">Products</li>

<li class="list-inline-item">About Us</li>

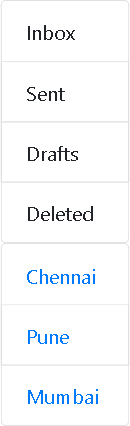
<li class="list-inline-item">Contact</li>

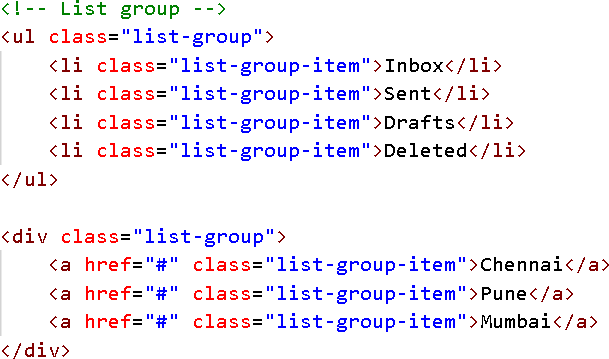
</ul>



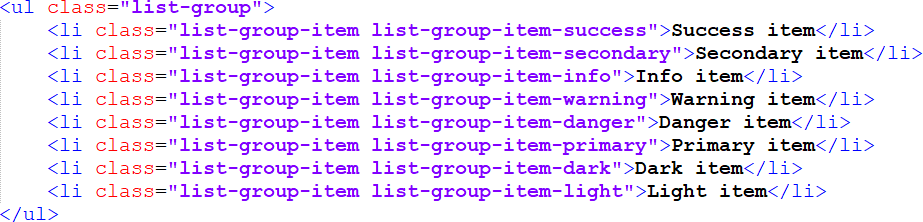
#### List group is used for creating lists, like a list of useful resources/ list of recent activities.

* + You can also use it for a complex list of large amounts of textual content.
  + Add class list-group to a <ul> or <div> element to make its children appear as a list.
  + The children can be li or a element, depending on your parent element choice.
  + The child should always have the class list-groupitem.





* + - We can also apply various colors to each list item by adding list-group-item-\* classes along with list-group-item.

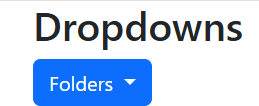


## Dropdowns

#### A dropdown is a toggleable menu allowing user to choose one value from a predefined list

* + .dropdown class indicates a dropdown menu.
  + To open the dropdown menu, use a button or a link with a class of .dropdown-toggle and the data-bs-

toggle="dropdown" attribute.

* + Add the .dropdown-menu class to a <ul> element to actually build the dropdown menu.



# BOOTSTRAP NAVIGATION BAR

## Bootstrap 5 Navs

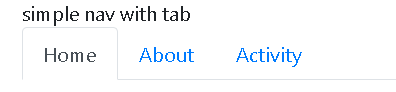
* If you want to create a simple horizontal menu, add the .nav class to a **<ul>/<div>/<nav>**

element, followed by .nav-item for each <li>/<a> and add the .nav-link class to their links

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| <div> | Simple nav without bootstrap | nav/a </div> | |  |
| <nav>  <a  <a | href="#">Home</a> href="#">About</a> | |  |  |
| <a href="#">Activity</a>  </nav> | | | |  |

|  |  |  |  |
| --- | --- | --- | --- |
| <div> | Simple nav with nav/a </div> |  |  |
| <nav class="nav">  <a href="#" class="nav-item nav-link">Home</a>  <a href="#" class="nav-item nav-link">About</a>  <a href="#" class="nav-item nav-link">Activity</a>  </nav>  <div> Simple nav with ul</div>  <ul class="nav"> | |  |  |
| <li><a href="#" class="nav-item nav-link">Home</a></li>  <li><a href="#" class="nav-item nav-link">About</a></li>  <li><a href="#" class="nav-item nav-link">Activity</a></li>  </ul> | | |  |

* Add the class **.nav-tabs** to the basic nav to generate a tabbed navigation.



<!-- simple nav with tab -->

<nav class="nav nav-tabs">

<a href="#" class="nav-item nav-link active">Home</a>

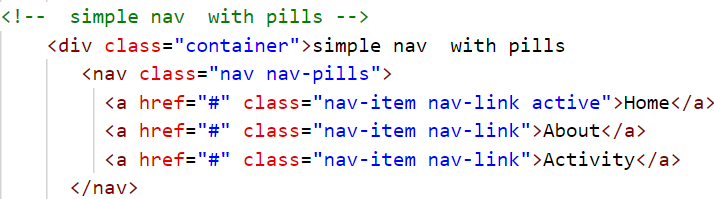
<a href="#" class="nav-item nav-link">About</a>

<a href="#" class="nav-item nav-link">Activity</a>

</nav>

* Similarly, you can create pill based navigation by adding the class **.nav-pills** on the basic

nav instead of class .nav-tabs



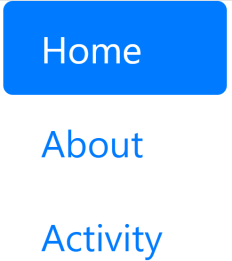
* Vertically stack these pills by attaching an additional class flex-column

<!-- simple nav with pills and stacked with flex -->

<nav class="nav nav-pills flex-column">

...

</nav>



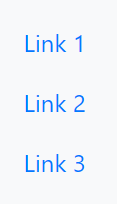
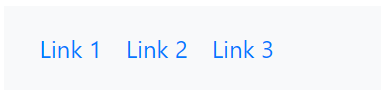
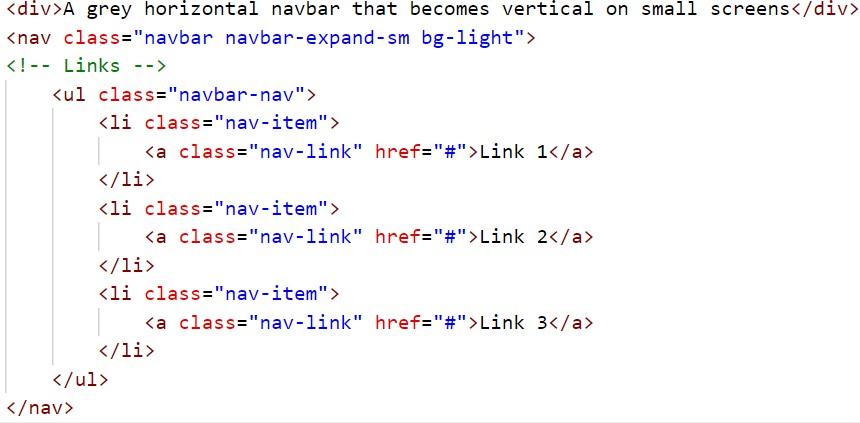
* + The .nav class is for simple navigation. Whereas .nav-bar is intended for the main page navigation on top of the page and has a whole bunch of styling attributes available.
  + First build a <div> or <nav> element, with class navbar.
    - A standard navigation bar is created with the .navbar class, followed by a responsive collapsing class:

.navbar-expand-xxl|xl|lg|md|sm (becomes vertical on small screens).

<div class="navbar">

</div>

* + - To add links inside the navbar, use a <ul> element with class .navbar-nav
    - Then add the usual <li> and <a> elements with a .nav-item and a .nav-link class

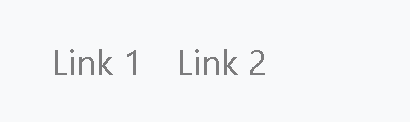
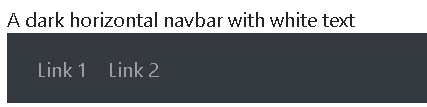
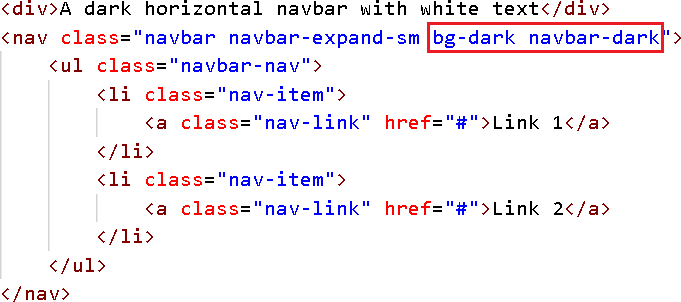


* Use any of the .bg-color classes to change the background color of the navbar (.bg-primary,

.bg-success, .bg-info, .bg-warning, .bg-danger, .bg-secondary, .bg-dark and .bg-light)

* Tip: Add a white text color to all links in the navbar with the .navbar-dark class, or use the

.navbar-light class to add a black text color.



<nav class="navbar navbar-expand-sm bg-light navbar-light">

* The .navbar-brand class is used to highlight the brand/logo/project name of your page



<!-- navbar with image logo -->

<nav class="navbar navbar-dark bg-secondary">

<a href="#" class="navbar-brand">

<img src="paws.png" height="28" alt="CoolBrand">

</a>

</nav>

<!-- navbar with text logo -->

<nav class="navbar navbar-dark bg-secondary">

<a class="navbar-brand">Pawsome</a>

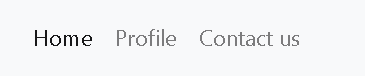
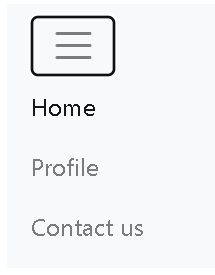
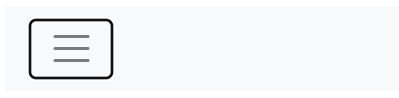
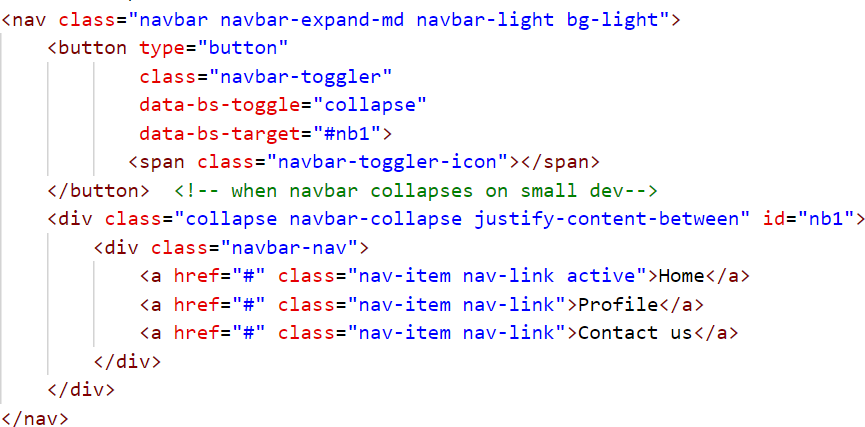
</nav>

## Co lapsing The Navigation Bar

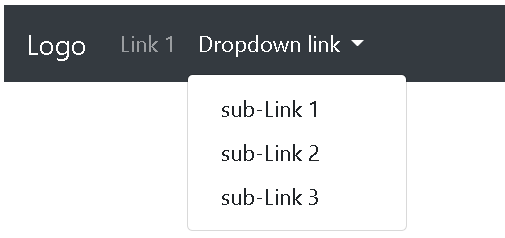
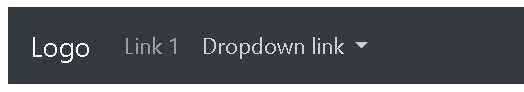
* Very often, especially on small screens, you want to hide the navigation links and replace them with a button that should reveal them when clicked on.
* To create a collapsible navigation bar, use a button with class="navbar-toggler", data-bs-toggle="collapse"

and data-bs-target="#thetarget".

* Then wrap the navbar content (links, etc) inside a div element with class="collapse navbar-collapse", followed by an id that matches the data-target of the button: "thetarget".



Navbar With Dropdown

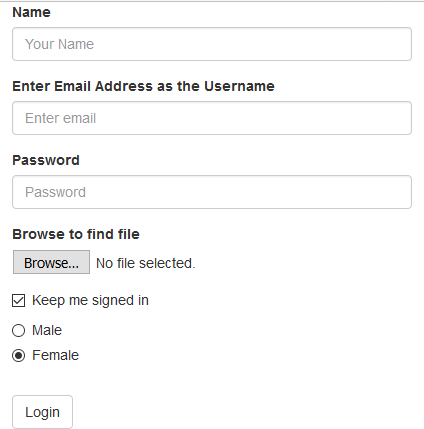


# FORMS

## Creating Forms

#### By default, forms are aligned vertically as the default value for display is block and is set to 100%.

* We can use additional classes to change this default placement
  + There are no classes to make a layout form but we can use some classes to change the alignment of forms.



* + Syntax:

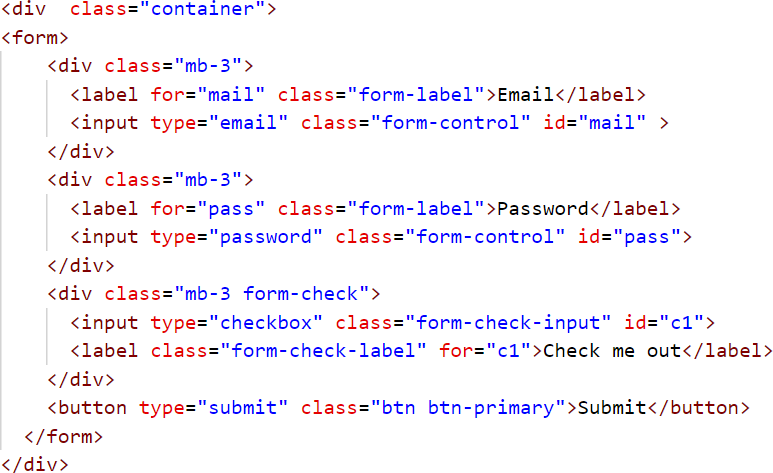


add a .form-label class to each label element to ensure correct padding

## Creating Forms : Vertical Form Layout

#### The form controls in this layout are stacked with left-aligned labels on the top.

add a .form-label class to each label element to ensure correct padding

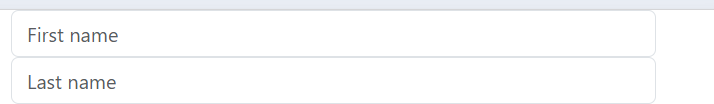
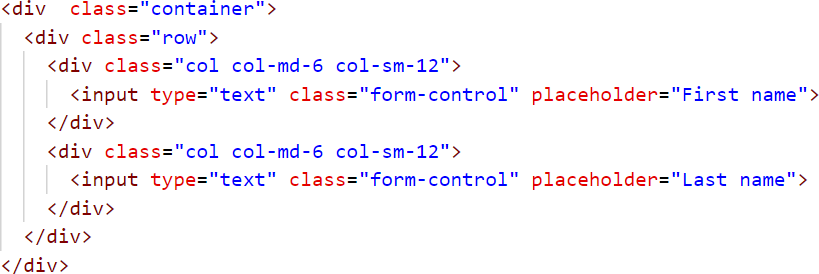


Add class .form-control get proper form styling

## Inline Forms

#### In an inline form, form elements to appear side by side.

* + - Add class .row and .col



## Bootstrap theme

#### bootstrap.css is the core css for BootStrap that defines all the style for various controls/components, where as bootstrap-theme.css defines the themes (gradient/animation) for buttons, dropdown, menu, navbar, progressbar, panels.

* + - Most of the times adding bootstrap.css is enough for bootstrap to work, but for gradient/animation, you should use bootstrap-theme.css

