# 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:

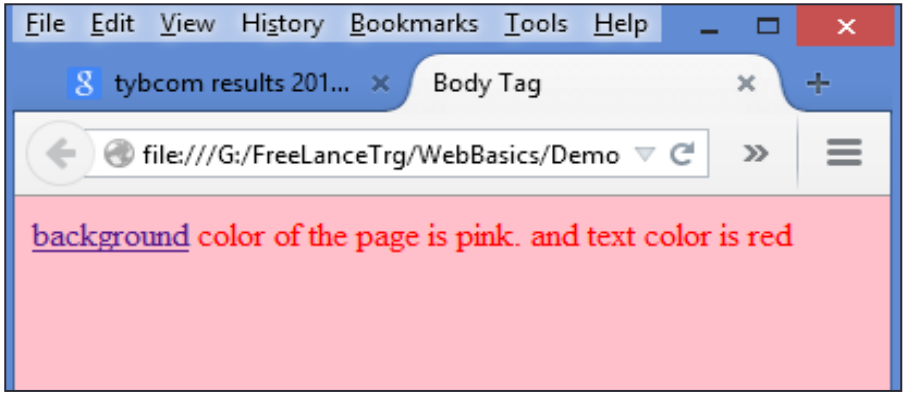
```
<!DOCTYPE HTML>
<html>
        <head>
            <title>Title of page</title>
        </head>
        <body>
                This is my first homepage. <b>This text is bold</b>
        </body>
</html>
```

- 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.).

# HTML Body Section

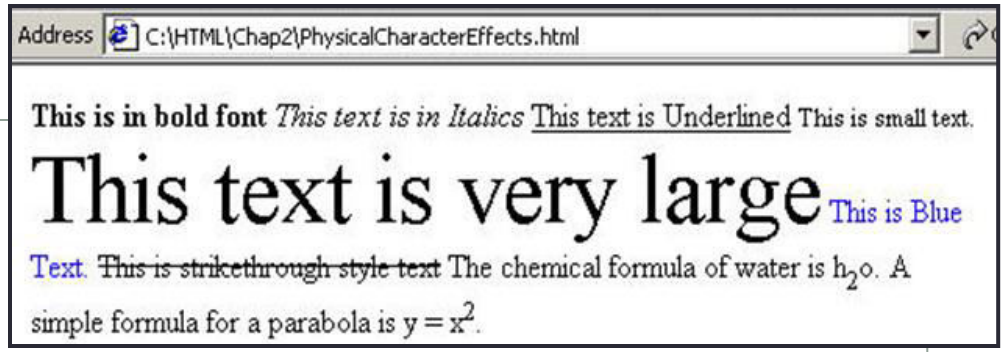
- <body> Element:
  - Each document can have at most one <body> element.
  - Attributes used in <body> element are:
    - BGCOLOR: Gives a background color to HTML page.
    - BACKGROUND: Use to specify images to the BACKGROUND.
    - TEXT: Specifies text color throughout the browser window
    - LINK, 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">
    - <body background="symbol.gif" text="red" link="blue" bgproperties="fixed">

```
<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>
```



# Physical Character Effects

- Bold Font: …
- Italic: …
- Underline: <u>…</u>
- Strikethrough: <strike> or <s>
- Fixed width font: <tt> - normal text in fixed-width font
- Subscript: <sub>
- Superscript: <sup>




```
<html>
<body>
<b>This is in bold font</b>
<i>This text is in Italics</i>
<u>This text is Underlined</u>
<small>This is small text.</small>
<font size=7>This text is very large</font>
<font color="Blue">This is Blue Text.</font>
<strike>This is strikethrough style text</strike>
The chemical formula of water is h<sub>2</sub>o.
A simple formula for a parabola is y = x<sup>2</sup>.
</body>
</html>
```

# Document (Body) Contents

- Body Text
  - HTML truncates spaces in your text.
  - Use <br> to insert new lines.
  - Use <p> tag to create paragraphs.
  - Use  to hold division or a section in an HTML document
  - Use  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 -->

# Logical Character Effects

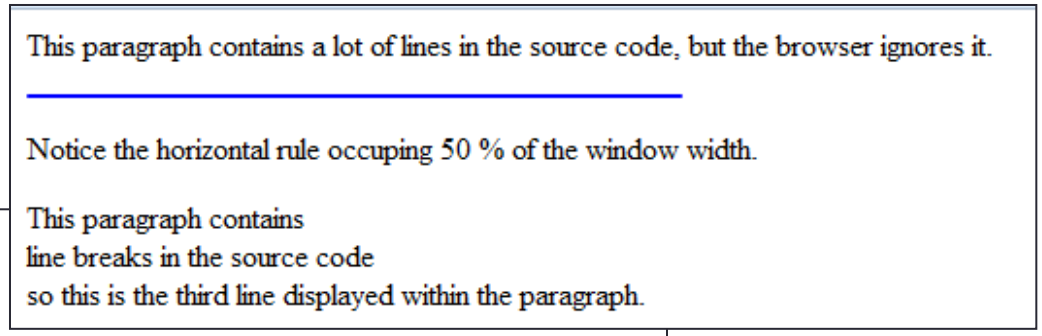
- Heading Styles:
  - **<hn>.....</hn>**
    - Value of n can range from **1 to 6**
  - <h1 align="center">This is level 1 heading</h1>

**Using Heading styles**

```
<html>
<head><title>This is the first html page</title>
<body>This is the first html page created
<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>
</head>
</html>
```

# Horizontal Lines in a Web Page

- <hr> - horixontal 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>
```

1. INDIA
2. SRILANKA

  - 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>
  - I-Uppercase Roman letters
  - i-Lowercase Roman letters
  - 1-Standard numbers, default

# Bulleted 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

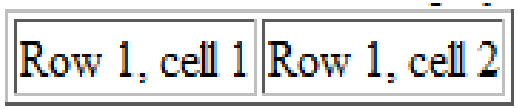
- Images are added into a document using <img> tag.
- Attributes of <img> tag are:
  - alt: Alternative text to display for non-graphical browsers.
  - align: Image horizontal alignment. Valid values: left, right, center.
  - width/Height: Sets the width and height of the image.
  - src: Specifies the image path or source.

```
<IMG src="home.png“  height="100“  width="200"  >
```

# Table

- Use <table> tag to create a table.
- Table Attributes:
  - Border: <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 border="1">
        <td>Row 1, cell 1</td>
        <td>Row 1, cell 2</td>
    </tr>
</table>
```



# Table Data

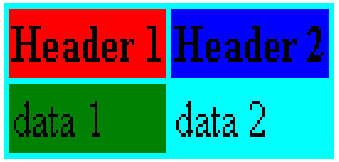
- An HTML table has two kinds of cells:
  - Header Cells <th>: Contain header information (created with <th> element) ; The text is bold and centered.
  - Standard Cells <td>: Contain data (created with the td element) ; The text is regular and left-aligned.

```
<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">
```

```
<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>
```



# How Does a Hyperlink Work?

- Hyperlinks access resources on the internet.
- Create a link with <a href=“”> (anchor)

Login Here
Hello, Welcome to My Site
I have some older information about this subject.

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

- Hyperlinks in Lists Items & Table Elements

```
<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>
```

- mumbai
- pune
- nasik

```
<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>
```

| team | points | grade |
|---|---|---|
| mumbai | 90 | a |
| pune | 86 | b |
| nasik | 80 | c |

# HTML Forms : Types of Form Fields

```
<form method="get/post" action="URL">
        Field definitions
</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 Field

- **Single Line Text Fields**: 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”>
```

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

# Hidden 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"/>

# Password Fields

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

# File Selector Dialog Box

- <input type="file" name="name" size="width of field" value="value">

```
<FORM>
   Please select file that you want to upload:
   <INPUT name="file" type="file"> <BR>
   <INPUT type="submit" >
</FORM>
```

# Action 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"/>

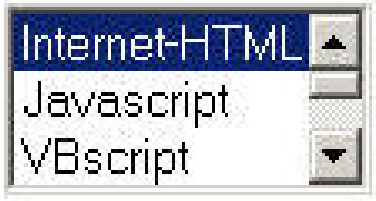
# Drop-Down List

<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.

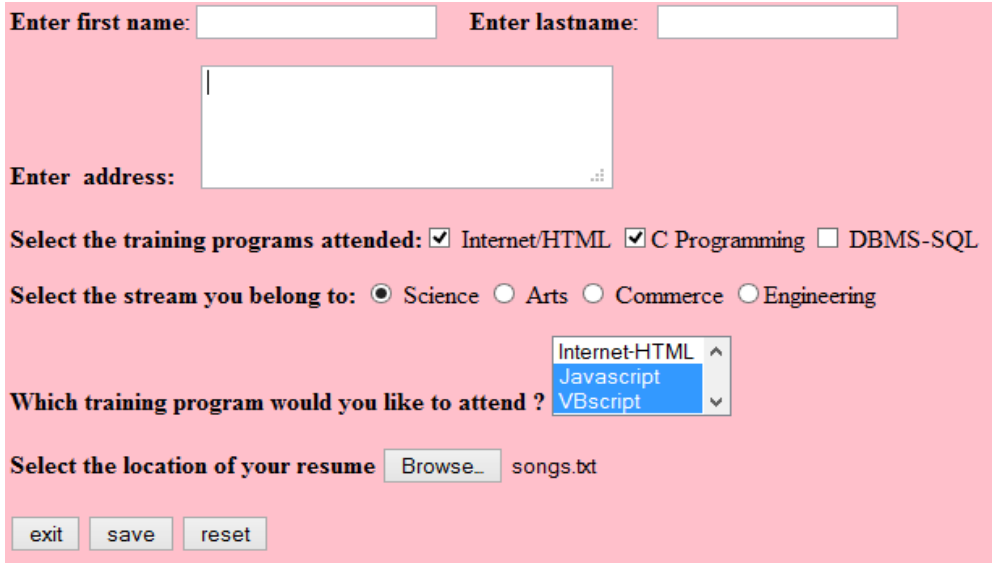
Eg:

```
<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>
<option value="xm">XML</option>
<option value="jv">JAVA</option>
<option value="jsp">jsp</option>
</select>
</form>
```



# Complete example

```
<html><head><title>form examples</title></head>
<body bgcolor="pink">
<form name="form1" action="store.html" method="post">
<p>
<strong>Enter first name</strong>: <input name="username">    
<strong>Enter lastname</strong>:   <input  maxlength="30" name="surname"></p>
<p><strong>Enter  address:</strong>    
<textarea name="addr" rows="3" readonly value="sjdshd"></textarea>
<br><br>
<strong>Select the training programs attended:</strong>
<input type="checkbox" value="internet/html" name="internet-html"> Internet/HTML
<input type="checkbox" checked value="c programming" name="c-programming">C Programming
<input type="checkbox" value="dbms-sql" name="dbms-sql"> DBMS-SQL </p>
<p><strong>Select the stream you belong to:</strong>
<input type="radio" value="science" name="s-grp"> Science
<input type="radio" value="arts" name="s-grp"> Arts
<input type="radio" value="commerce" name="s-grp"> Commerce
<input type="radio" value="oth2" name="s-grp">Engineering </p>
<strong>Which training program would you like to attend ?</strong>
 <select size="5" multiple name="pref">
    <option value="ih"   selected>Internet-HTML
    <option value="js">Javascript
    <option   value="vbs">VBscript
    <option value="as">ASP
    <option value="xm">XML
    <option value="jv">JAVA
    <option value="jsp">jsp</option>
</select> <br>
<input type="button" value="exit" name="but">
<input type="submit" value="save">
 <input type="reset" value="reset">
</form></body></html>
```



# 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

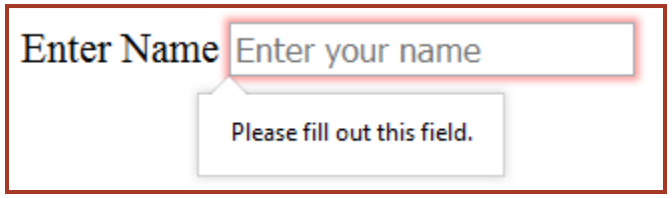
# HTML5 Attributes for <input>

- A Form is one of the most basic and essential feature of any web site
  - HTML5 brings 13 new input types
  - 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

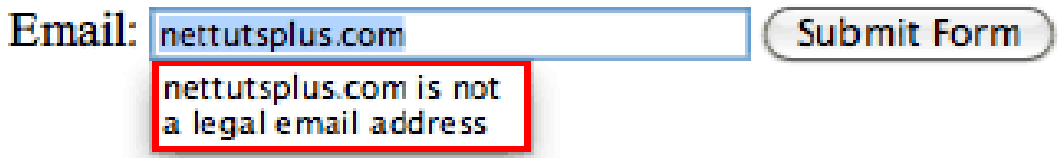
- **AutoFocus** - Autofocus is a Boolean attribute of form field that make browser set focus on it when a page is loaded
  - <input id ="Text2" type="text" autofocus/>

- **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 />

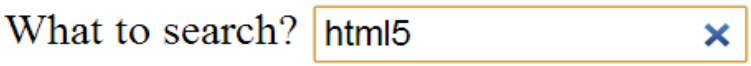


# New Form elements

- **Email** - This field is used to check 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" />

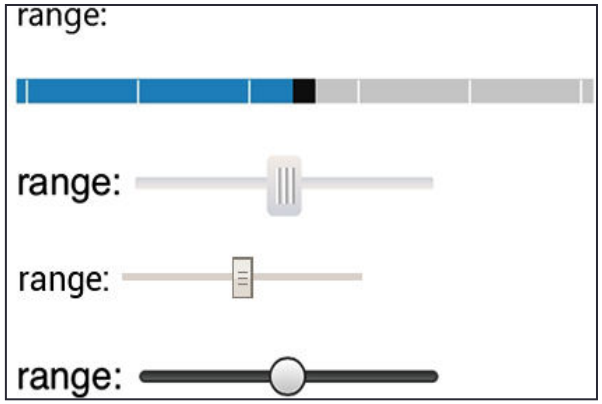



- **Tel** - used for input fields that should contain a telephone number.
  - <input type="tel" name="usrtel">
- **url** - is used for input fields that should contain a URL address.
  - Depending on browser support, the url field can be automatically validated when submitted.
- **color** – displays a color palette

# New Form elements

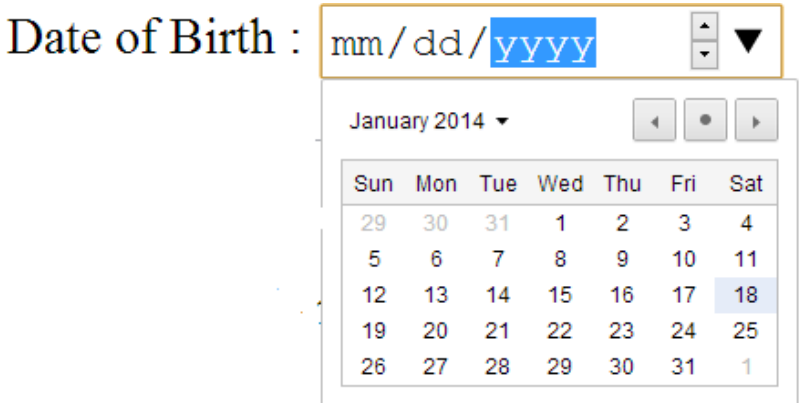
- **Number** - used for input fields that should contain a numeric value.
  - Min and max parameters provided to limit the values.
  - Browser will treat it as simple textfield if it doesn't support this type.
  - <input id="movie" type="number" value="0"/>
  - <input type="number“ min="0“ max="50“ step="2“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

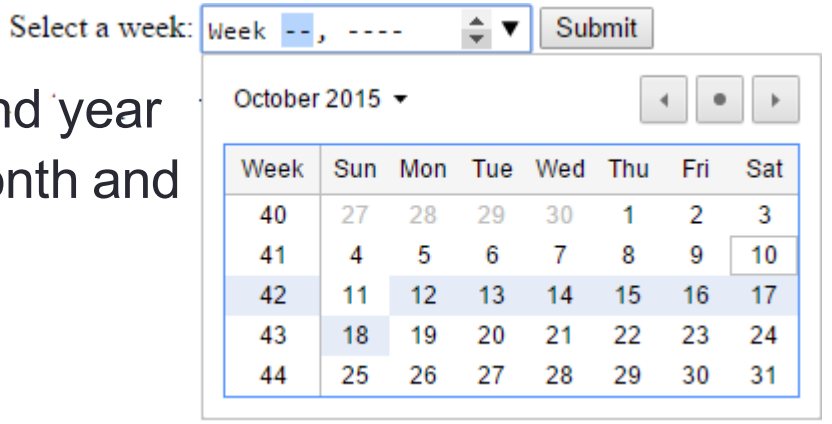


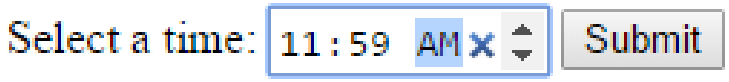
# New Form elements

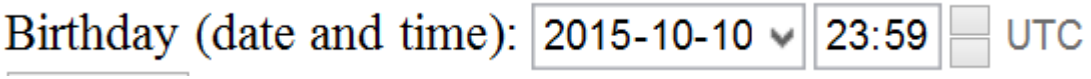
- **Date** - used for input fields that should contain a date.
  - Depending on browser support, a date picker can show up in the input field.
  - <input id="meeting" type="date“ />




- **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)



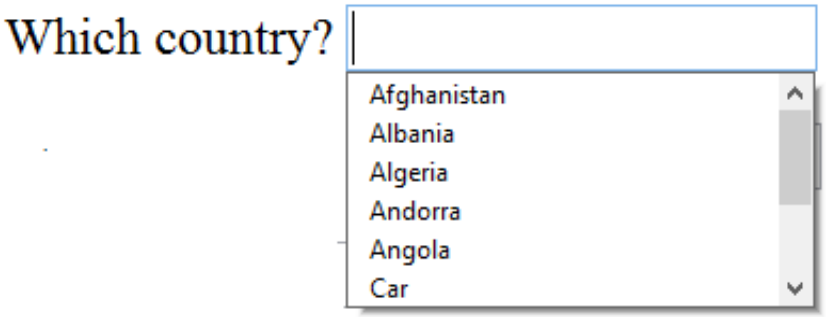



# New Form elements

- Data List - specifies a list of options for an input field. The list is created with option elements inside the datalist.
  - seems like type-ahead auto suggest textbox as you can see in Google search box

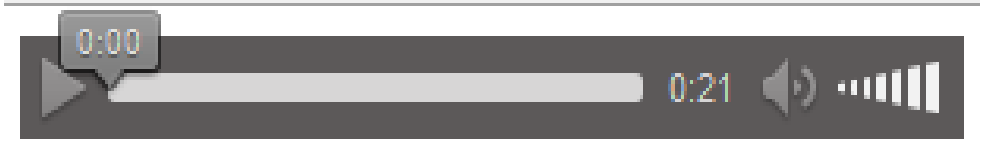
```
<input id="country_name"
        name="country_name"
        type="text“
        list="country" />

<datalist id="country">
<option value="Australia">
<option value="Austria">
<option value="Algeria">
<option value="Andorra">
<option value="Angola">
</datalist>
```



# Audio

- 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 specifies a standard way to include audio, with the audio element; The audio element can play sound files, or an audio stream.
  - Currently, there are 3 supported formats for the audio element: Ogg Vorbis, MP3, .webm and WAV
  - Other properties like auto play, loop, preload area also available

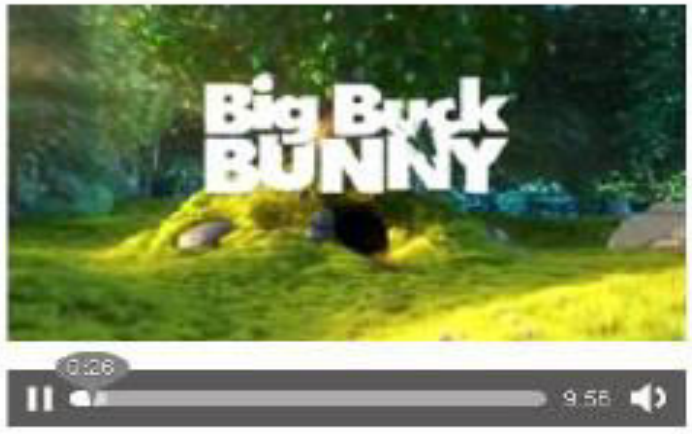
```
<audio controls>
<source src="vincent.mp3" type="audio/mpeg"/>
<source src="vincent.ogg" type="audio/ogg"/>
</audio>
```



# Video

- Until now, there hasn't been a standard for showing video on a web page.
  - Today, most videos are shown through a plugin (like Flash). However, not all browsers have the same plugins.
  - HTML5 specifies a standard way to include video with the video element
  - 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>
```

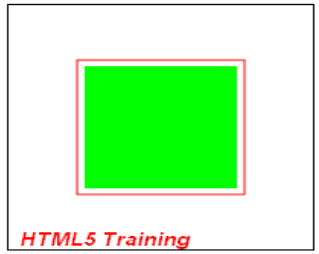


# Canvas

- 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

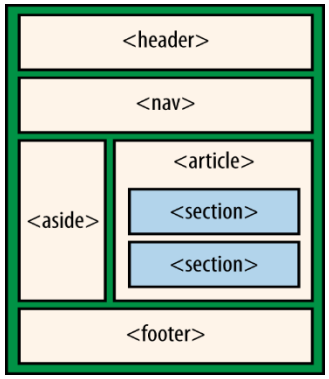
```
<canvas id="myCanvas" width="200" height="100">
</canvas>
```

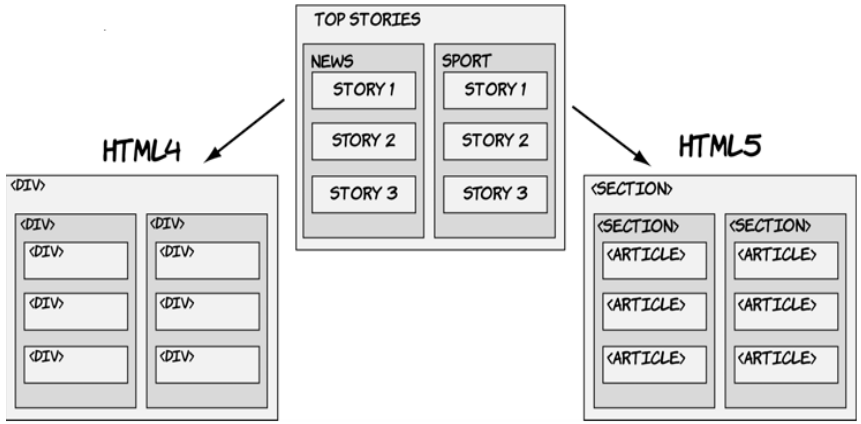
```
<canvas id="myCanvas"></canvas>
<script type="text/javascript">
  var canvas=document.getElementById('myCanvas');
  var ctx=canvas.getContext('2d');
  ctx.fillStyle='#FF0000';
  ctx.fillRect(0,0,80,100);
</script>
```




# New Semantic Elements

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



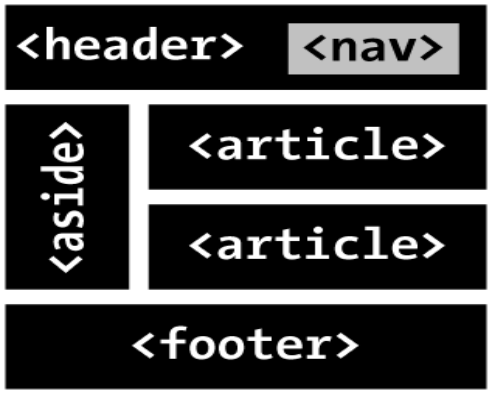




- **<article>:** element represents a self-contained composition in a document, page, application, or site that is intended to be independently distributable or reusable
  - Eg a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment
- **<nav>**: Represents a major navigation block. It groups links to other pages or to parts of the current page.
- **<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

# New Semantic Elements

- **<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.




Fat footer on the w3c site

# What is CSS?

- Cascading Styles Sheets - a way to style and present HTML.
  - HTML deals with content and structure, stylesheet deals with formatting and presentation of that document.
  - Allows to control the style and layout of multiple Web pages all at once.
- Example:

```
<font color="#FF0000" face="Verdana, Arial, Helvetica, sans-serif">
<strong>This is text</strong></font>
```

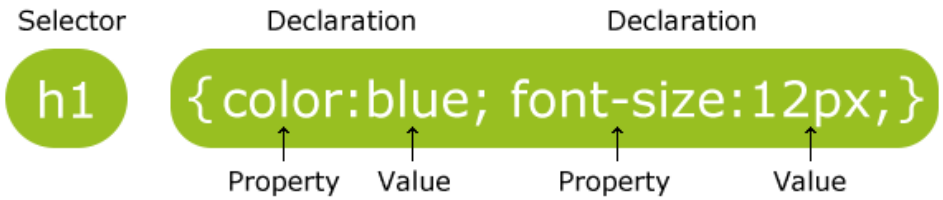
Messy HTML

```
<p class="MyStyle">My CSS styled text</p>
…….
<style>
   .myStyle {
       font-family: Verdana, Arial, Helvetica, sans-serif; font-weight: bold; color: #FF0000; }
</style>
```

- Why CSS?
  - saves time
  - Pages load faster
  - Easy maintenance
  - Superior styles to HTML

# CSS Syntax

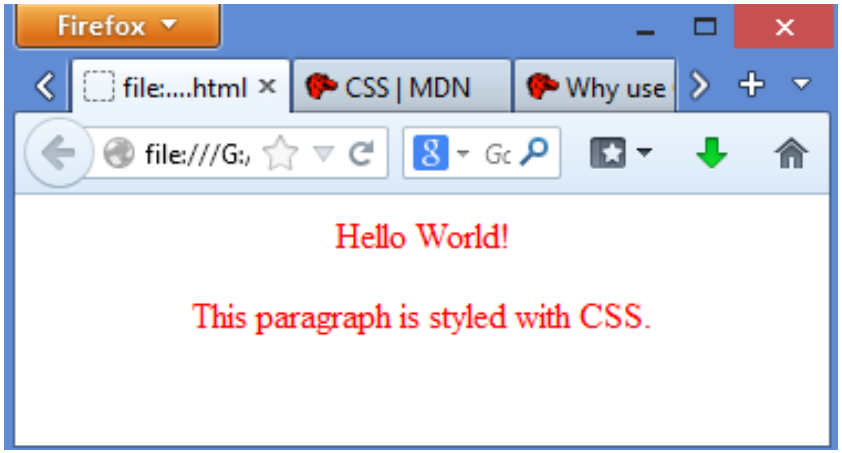
- A CSS rule has two parts: a selector, and one or more declarations:




- "HTML tag" { "CSS Property" : "Value" ; }

```
<head>
<style>
p
 {
   color:red;
   text-align:center;
 }
</style>
</head>
<body>
<p>Hello World!</p>
<p>This paragraph is styled with CSS.</p>
</body>
```

Selects all the paragraphs in HTML document and gives them style



# 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.
  - Example in previous slide

- 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.
  - This is a paragraph.
  - A new background and font color with inline CSS

# Demo: Link Style Sheet

```
<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>
```

```
body {   background: black;
         color:green
      }
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_ex2.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>
```

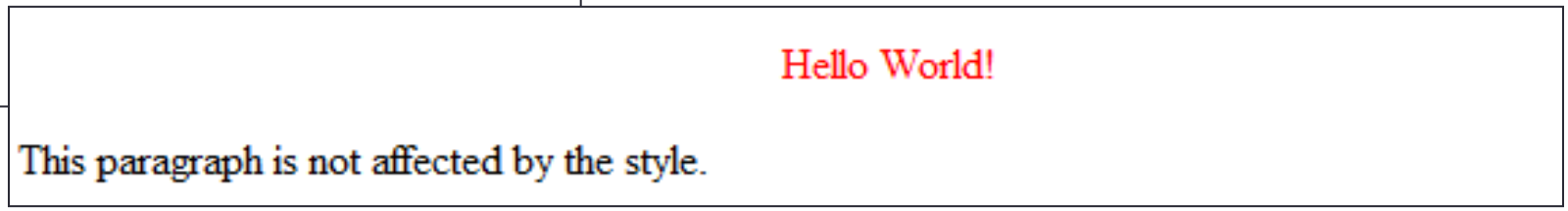
# Types of selectors

- HTML selectors
  - 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

# ID Selector & Style Sheet Classes

```
<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>
```

```
<html> <head>
<style>
#para1 {
text-align:center;
color:red;
}
</style> </head>
<body>
<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the
style.</p>
</body>
</html>
```



# 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

# 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; }
```

This is some text in a paragraph.

```
<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>
```

Hello, world!

```
<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>
```

# 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

- Where to Write JavaScript?
  - Head Section
  - Body Section
  - External File

```
<html>
<head>
<script type="text/javascript">
  function message() {
        alert("Hello World")
   }
</script>
</head>
<body onload="message()">
</body>
</html>
```

```
<html>
<head></head>
<body >
<script language="javascript">
            document.write("Hello World!")
</script>
</body>
</html>
```

```
<head>
<script src="common.js">
      <!– no javascript statements →
   </ script>
</head>
<body>
   <script>
       document.write("display value of a variable"+msg)
   </script>
</body>
```

```
//common.js file contents
var msg
msg="<h1>in external file</h1>"
```

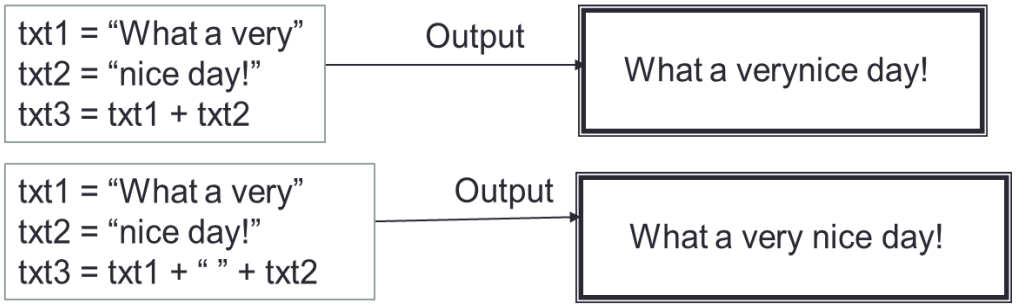
# Data Types in JavaScript

- 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:

- Arithmetic Operators ( + , -, * , / , %)
- Assignment Operators(=,+=,-=,*=,/=,%=)
- Comparison Operators (==,!=,<,<=,>,>=)
- Boolean Operators(&&,||,!)
- Bitwise Operators(&,|,!,^,<<,>>,>>>)
- String Operators(=,+,+=)



# Control Structures and Loops

- JavaScript supports the usual control structures:
  - the conditionals:
    - if,
    - if...else
    - If … else if … else
    - Switch

```
if(condition) {
      statement 1
} else {
       statement 2
}
```

```
if(a>10) {
document.write("Greater than 10")
} else {
document.write("Less than 10")
}
```

```
document.write( (a>10) ? "Greater than 10" : "Less than 10" );
```

  - iterations:
    - for
    - while

```
switch (variable) {
      case outcome1 :{
            //stmts for outcome 1
             break; }
      case outcome2 :{
            //stmts outcome 2
              break; }
       default: {
             //none of the outcomes is chosen
          }
}
```

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

```
for(var i=0;i<10;i++)
{
        document.write("Hello");
}
```

```
while(condition) {
      statements
}
```

```
while(i<10) {
     document.write("Hello");
       i++;
}
```

# JavaScript Functions

```
function myFunction (arg1, arg2, arg3)
{
        statements
          return
}
```

Calling the function :

```
myFunction( “abc”, “xyz”, 4 )
myFunction()
```

```
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
```

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

```
var area = function (radius) {
      return Math.PI * radius * radius;
  };
alert(area(5));          // => 78.5
```

# Global and Local Variables

```
<script language="Javascript">
  var companyName="TechnoFlo"
  function f(){
    var employeeName="Henry"
    document.write("Welcome to "+companyName+ ", " +employeeName)
  }
</script>
```

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

# Predefined Functions

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

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

- isNaN : Evaluates an argument to determine if it is “NaN” (not a number)
  - isNaN (testValue), where testValue is the value you want to evaluate

```
isNaN(0)              //false
isNaN('123')          //false
isNaN('Hello')        //true
```

- Parseint and parsefloat
  - Returns a numeric value for string argument.
  - parseInt (str)
  - parseFloat (str)

```
parseInt("3 blind mice")          // => 3
parseFloat(" 3.14 meters")        // => 3.14
parseInt("-12.34")                // => -12
parseInt("0xFF")                  // => 255
parseInt("0xff")                  // => 255
parseInt("-0XFF")                 // => -255
parseFloat(".1")                  // => 0.1
parseInt("0.1")                   // => 0
parseInt(".1")                    // => NaN: integers can't start with "."
parseFloat("$72.47");   // => NaN: numbers can't start with "$"
```

parseInt() interprets any number beginning with “0x” or “0X” as hexadecimal no

# Working with Predefined Core Objects

# String Objects

- Creating a string object:
  - var myString = new String(“characters”)
  - var myString = “fred”

- Properties of a string object:
  - length: returns the number of characters in a string.

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

- 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));

# String functions

- 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
```

# String functions

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

```
var str = "zero one two three four";
var arr = str.split(" ");
for(i = 0; i < str.length; i++){  document.write(“<br>” + arr[i]); }
```

Output :

zero
one
two
three
four

```
var myString = “Anderson,Smith,Johnson,Washington”
var myArray = myString.split(“,”)
var itemCount = myArray.length // result: 4
```

- Complete Example:

```
var s = "hello, world"          // Start with some text.
s.charAt(0)                     // => "h": the first character.
s.charAt(s.length-1)            // => "d": the last character.
s.substring(1,4)                // => "ell": the 2nd, 3rd and 4th characters.
s.slice(1,4)                    // => "ell": same thing
s.slice(-3)                     // => "rld": last 3 characters
s.indexOf("l")                  // => 2: position of first letter l.
s.lastIndexOf("l")              // => 10: position of last letter l.
s.indexOf("l", 3)               // => 3: position of first "l" at or after 3
s.split(", ")                   // => ["hello", "world"] split into substrings
s.replace("h", "H")             // => "Hello, world": replaces all instances
s.toUpperCase()                 // => "HELLO, WORLD"
```

# 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 |

# Array

- An array is data structure for storing and manipulating ordered collections of data.
- An array can be created in several ways.
  - Eg1: Regular: ----------→
  - 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];

```
var cars=new Array();
cars[0]="Spark";
cars[1]="Volvo";
cars[2]="BMW";
```

- Deleting an array element eliminates the index from the list of accessible index values
  - delete is a unary operator that attempts to delete the **object property** or **array element** specified
  - This does not reduce array's length

```
myArray.length// result: 5
delete myArray[2]
myArray.length// result: 5
myArray[2] // result: undefined
```

# 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”
```

```
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”
```

```
var arrayText = myArray.join(“,”)
```

```
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"]
```

# Creating New Objects

1. **Using Object Initializers**
   - Syntax : objName = {property1:value1, property2:value2, … }
   - person = { "name ":"amit", "age":23};
   - myHonda = {color:“red”, wheels:4, engine:{cylinders:4, size:2}}
2. **Using Constructors**
   - Define the object type by writing a constructor function.
   - Create an instance of the object with new.

```
function car(make, model, year) {
  this.make = make
  this.model = model
  this.year = year
}
…..
mycar = new car( “Ford” , “Mustang” , 2013)
```

```
function person(name, age) {
    this.name = name
    this.age = age
}
ken = new person( “Ken” , 33 )
```

```
function car(make, year, owner) {
  this.make = make
  this.year = year
  this.owner = owner
}
car1 = new car( “Mazda”, 1990, ken )
```

# Creating New Objects (Contd.)

- Accessing properties
- Defining methods
  - Example:

car2.owner.name

car1.make = “corvette”

```
obj.methodName = function_name
obj.methodName(params)
```

```
function car(make, model, year, owner) {
        this.make = make;
        this.model = model;
        this.year = year;
        this.displayCar = displayCar;
}
function displayCar() {
        document.writeln( “A beautiful” + this.year
         + “ ” + this.make + “ ” + this.model
 }
….
car1.displayCar();   car2.displayCar()
```

# Examples : Using Object Initializers

```
// Example 1
var myFirstObject = {};
myFirstObject.firstName = "Andrew";
myFirstObject.lastName = "Grant";
console.log(myFirstObject.firstName);
```

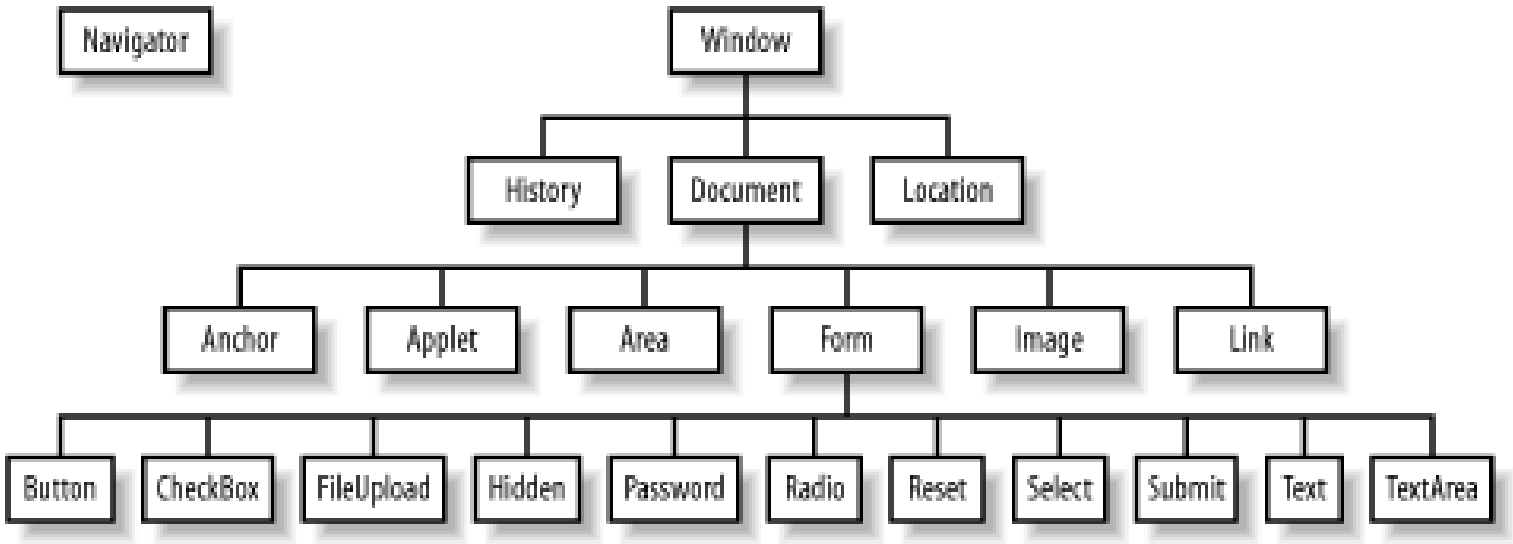
```
// Example 2
var mySecondObject = {
    firstName: "Andrew",
    lastName: "Grant"
};
console.log(mySecondObject.firstName);
```

```
// Example 3
var myThirdObject = new Object();
myThirdObject.firstName = "Andrew";
myThirdObject.lastName = "Grant";
console.log(myThirdObject.firstName);
```

```
var myFirstObject = {};
myFirstObject.firstName = "Andrew";
console.log(myFirstObject.firstName);
myFirstObject.firstName = "Monica";
console.log(myFirstObject.firstName);
myFirstObject["firstName"] = "Catie";
console.log(myFirstObject["firstName"]);
```

```
//Adding Methods to Objects
var person= {
  name: "Andrew",
  age: 21,
  info: function () {
    console.log("Name" + this.name );
    console.log("Age" + this.age );
   }
};
person.info();
for (var prop in person) {
     console.log(person[prop]);
}
```

# JavaScript Document Object Model



# 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”)

- window.open(*URL,name,specs*)
  - URL : Specifies the URL of the page to open. If no URL is specified, a new window with about:blank is opened
  - Name : Specifies the target attribute or the name of the window.
  - Specs : comma-separated list of items.

```
myWindow=window.open('','','width=200,height=100');
myWindow.document.write("<p>This is 'myWindow'</p>");
myWindow.focus();
```

example opens an about:blank page in a new browser window:

# setInterval and setTimeout methods

```
<body>
<input type="text" id="clock" size="35" />
<script language=javascript>
var int=self.setInterval("clock()",50)
function clock() {
   var ctime=new Date()
   document.getElementById("clock").value=ctime
 }
</script>
<button onclick="int=window.clearInterval(int)">Stop interval</button>
</body>
```

```
<head> <script type="text/javascript">
function timedMsg()  {
    var t=setTimeout("alert('5 seconds!')",5000)
}
</script> </head>
<body> <p>Click on the button. An alert box will be displayed after 5 seconds.</p>
<form>
<input type="button" value="Display timed alertbox!“ onClick="timedMsg()">
</form>
</body>
```

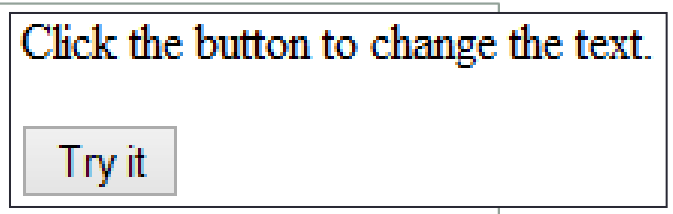
# Document Object

- 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

| | |
|---|---|
| document.anchors | Returns a collection of all the anchors in the document |
| document.baseURI | Returns the absolute base URI of a document |
| document.cookie | Returns all name/value pairs of cookies in the document |
| document.forms | Returns a collection of all the forms in the document |
| document.getElementById() | Returns the element that has the ID attribute with the specified value |
| document.getElementsByName() | Accesses all elements with a specified name |
| document.getElementsByTagName() | Returns a NodeList containing all elements with the specified tagname |
| document.images | Returns a collection of all the images in the document |
| document.lastModified | Returns the date and time the document was last modified |
| document.links | Returns a collection of all the links in the document |
| document.referrer | Returns the URL of document that loaded current document |
| document.title | Sets or returns the title of the document |
| document.URL | Returns the full URL of the document |
| document.write() | Writes HTML expressions or JavaScript code to a document |
| document.writeln() | Same as write(), but adds a newline character after each statement |

# Examples : Modifying content

```
<body>
<p id="p1">Click the button to change the text.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction() {
   document.getElementById("p1").innerHTML="Hello World";
};
</script></body>
```

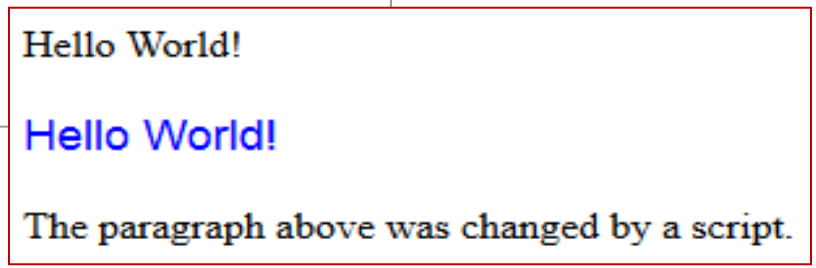




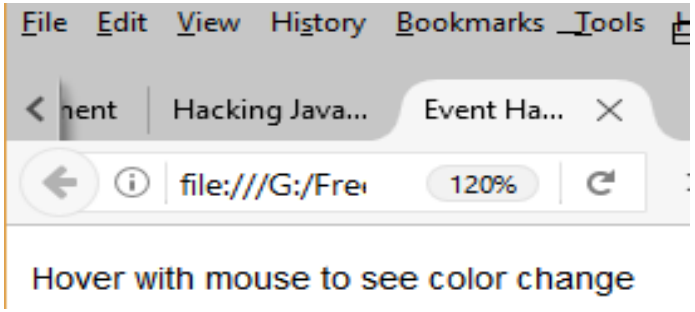
```
<body>
The title of the document is:
<script>
document.write(document.title);
document.title="another title"  //change the title
</script>
</body>
```

# Example : Modifying styles

```
<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>
</html>
```



# Mouse events

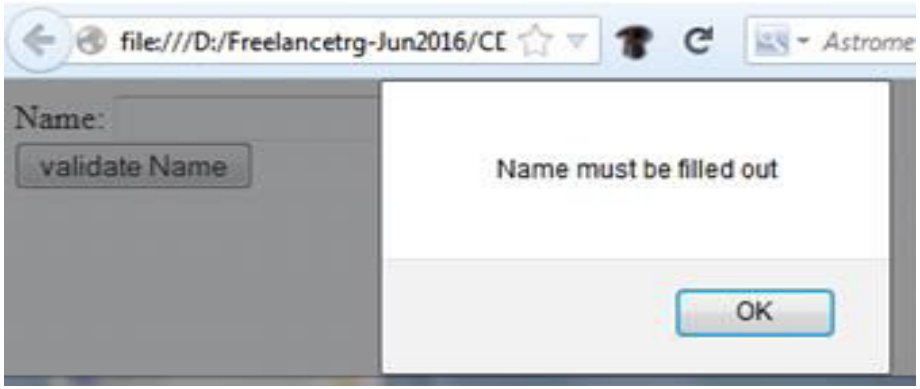



```
<SCRIPT>
function changeColor(para){
   para.style.color="blue";
   para.style.backgroundColor = "lightgray";
   para.style.font = "italic bold 30px arial,serif";
}
function revertColor(para){
    para.style.color="black";
    para.style.backgroundColor = "white";
    para.style.font = "12px arial,serif";
}
</SCRIPT>
<BODY>
   <p id="p1" onmouseover="changeColor(this)"
          onmouseout="revertColor(this)">Hover with mouse to see color change</p>
</BODY>
```

***Hover with mouse to see color change***

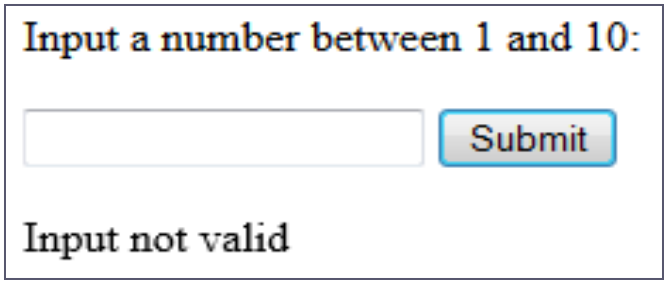
# Form validation

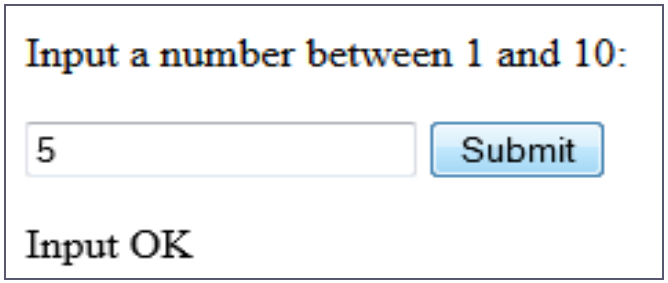
```
<html>
<head>
<script >
function validate(){
    var x=document.getElementById("fname").value;
    if (x == null || x == "") {
        alert("Name must be filled out");
        return false;
    }
}
</script>
</head>
<body>
<form id="form1"  onsubmit="return validate()">
Name: <input type="text" name="fname" id="fname" /><br />
<input type="submit" value="validate Name"  />
</form>
</body></html>
```



# Example

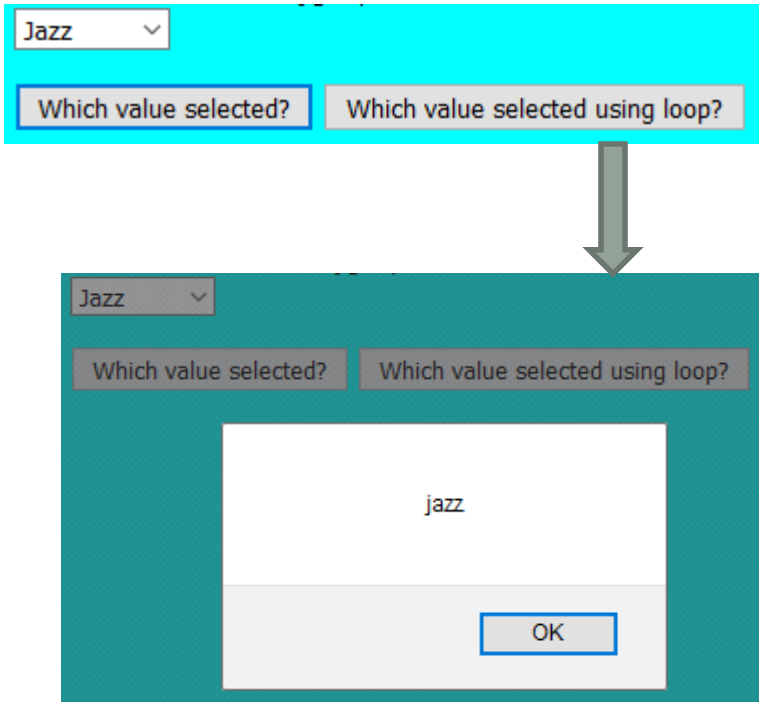
```
<body>
< Input a number between 1 and 10:</p>
<input id="numb">
<button type="button" onclick="myFunction()">Submit</button>
<p id="demo"></p>
<script>
function myFunction() {
    var x, text;
   x = document.getElementById("numb").value;
   if (isNaN(x) || x < 1 || x > 10) {
        text = "Input not valid";
    } else {
        text = "Input OK";
    }
    document.getElementById("demo").innerHTML = text;
}
</script>
</body>
```



# Example

```
<SCRIPT>
function valSelected1(){
  var sel = document.getElementById("musicTypes");
  alert(sel.value);       // prints value, not text
  var opt = sel.options[sel.selectedIndex];
  alert(opt.text);      //option.text prints text
}
function valSelected3(){
  var sel = document.getElementById("musicTypes");   var opt;
    for ( var i = 0, len =; i < sel.options.length; i++ ) {
        opt = sel.options[i];
        if ( opt.selected == true ) {   break;   }
    }
    alert(opt.value);
}
</SCRIPT>
<FORM NAME="selectForm">
  <SELECT name="musicTypes" id="musicTypes">
    <OPTION VALUE="rnb" SELECTED> R&B </OPTION>
    <OPTION VALUE="jazz"> Jazz </OPTION>
    <OPTION VALUE="blues"> Blues </OPTION>
</SELECT>
<INPUT TYPE="button" VALUE="value selected?"onClick="valSelected1()">
<INPUT TYPE="button" VALUE="value selected using loop?"onClick="valSelected3()">
</FORM>
</BODY>
```



# AJAX & 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.

```
var obj = {
    "name":"Amit",
    "age": 37.5,
    "married": true,
    "address":{ "city":"Pune" , "state":"Mah" },
    "hobbies":["swimming","reading","music"]
}
```

- 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, 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

# Demo

```
<script language="javascript">
var JSONObject = {  "name" : "Amit",
        "address"  : "B-123 Bangalow",
        "age"  : 23,
        "phone"   : "011-4565763",
        "MobileNo"  : 0981100092
  };
var str =
"<h2><font color='blue'>Name </font>::“        +JSONObject.name+"</h2>“ +
 "<h2><font color='blue'>Address </font>::“   + JSONObject.address+"</h2>“ +
"<h2><font color='blue'>Age </font>::“           +JSONObject.age+"</h2>“ +
"<h2><font color='blue'>Phone No </font>::“  +JSONObject.phone+"</h2>“ +
"<h2><font color='blue'>Mobile No </font>::“  + JSONObject.MobileNo+"</h2>“;

document.write(str);
</script>
```

**Name ::Amit**

**Address ::B-123 Bungalow**

**Age ::23**

**Phone No ::011-4565763**

**Mobile No ::981100092**

# Demo

```
<script >
var students = {
    "Students": [
            { "Name": "Amit Goenka",
              "Major": "Physics"
            },
            { "Name": "Smita Pallod",
             "Major": "Chemistry"
            },
            { "Name": "Rajeev Sen",
              "Major": "Mathematics"
            }] }
var i=0
document.writeln("students.Students.length : " + students.Students.length);
for(i=1;i<students.Students.length+1;i++)   {
    document.writeln("<b>Name : </b>" + students.Students[i].Name + "    ");
    document.writeln("<b>Majoring in : </b>" + students.Students[i].Major);
    document.writeln("</br>");
}
</script>
```

students.Students.length : 3
**Name :** Amit Goenka **Majoring in :** Physics
**Name :** Smita Pallod **Majoring in :** Chemistry
**Name :** Rajeev Sen **Majoring in :** Mathematics

## 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)

# 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

```
if (window.XMLHttpRequest) { // Mozilla, Safari, IE7+ ...
    xhr = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE 6 and older
    xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
```

- Step 2: Make the request

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

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

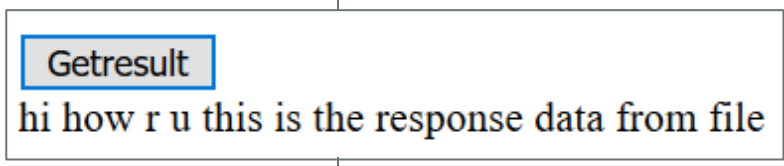
- Step 3 : Attach callback function to xhr object

```
httpRequest.onreadystatechange = function(){
      // process the server response
};
```

# Ajax Demo

```
<script >
var xhr;
function getData(){
   getHTTPRequestObject();
   if(xhr){
      xhr.open("GET", "Sample.txt", true);
      xhr.send();
      xhr.onreadystatechange = function(){
         if(xhr.readyState == 4 &&  xhr.status == 200){
               document.getElementById("lblresult").innerHTML=xhr.responseText;
          }
       }  //end of callback function
}}
function getHTTPRequestObject() {
       if (window.XMLHttpRequest) { // Mozilla, Safari, IE7+ ...
             xhr = new XMLHttpRequest();
       } else if (window.ActiveXObject) { // IE 6 and older
            xhr = new ActiveXObject("Microsoft.XMLHTTP");
  }}
</script> <body>
    <input type="button" onclick="getData()"  value="Getresult"/>
    <div id="lblresult"></div>
</body>
```

```
//sample.txt
hi how r u this is the
response data from file
```



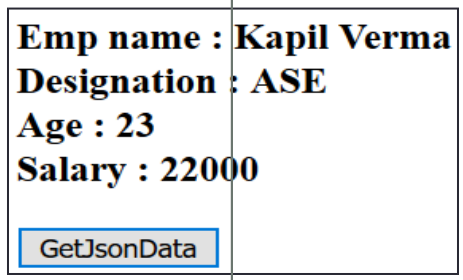
# AJAX Demo with XML

```
<script language="javascript" type="text/javascript">
var xmlhttp;
function getData(){
  getHTTPRequestObject();
  if(xmlhttp){
     xmlhttp.open("GET", "Employee.xml", true);
    xmlhttp.send();
    xmlhttp.onreadystatechange = function(){
        if(xmlhttp.readyState == 4 && ixmlhttp.status == 200){
            xmlDoc=xmlhttp.responseXML;
            x=xmlDoc.getElementsByTagName("EmpName");
            document.write("<h3>--- Emp names ----</h3><br>");
            for (i=0;i<x.length;i++){
                document.write(x[i].childNodes[0].nodeValue  + "<br>");
             }
    } }
</script>
<body>
<input type="button" onclick=" getData()" value="Getresult"/>
<div id="lblresult"></div>
</body>
</html>
```

```
<Employees>
  <Employee>
    <empid>1001</empid>
    <EmpName>Vipul</EmpName>
    <Desig>Software Analyst</Desig>
  </Employee>
 <Employee>
  <Empid>1002</Empid>
    <EmpName>Vivek</EmpName>
    <Desig>Software Analyst</Desig>
  </Employee>
</Employees>
```

# AJAX Demo with JSON

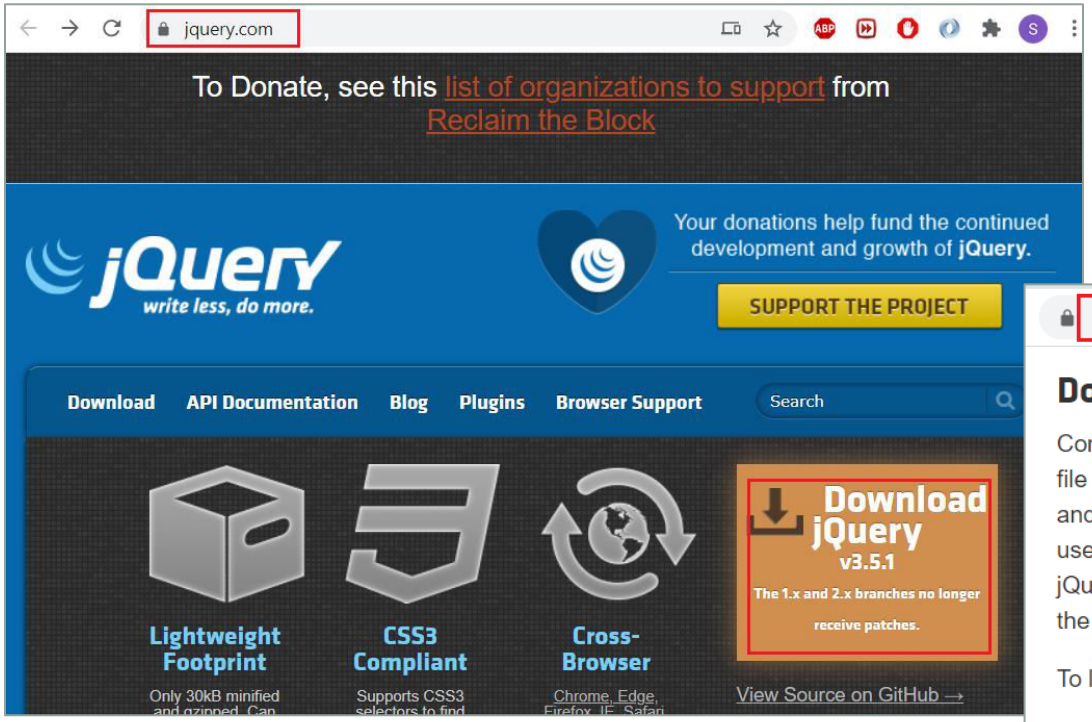
```
<script >
 var xmlhttp;
 function getData(){
   getHTTPRequestObject();
   if(xmlhttp){
      xmlhttp.open("GET", "EmpJSONData.txt", true);
     xmlhttp.onreadystatechange = function(){
       if(xmlhttp.readyState == 4 && xmlhttp.status == 200){
            var obj = JSON.parse(xmlhttp.responseText);
            var displaytext = "";
            displaytext += "Emp name : " + obj.name + "<br>" +
                        "Designation : " + obj.desig + "<br>" +
                        "Age : " + obj.age + "<br>" +
                        "Salary : " + obj.sal;
           document.getElementById("lblres").innerHTML = displaytext;
       }
} }
</script><body>
<h3 id="lblres">Result</h3>
<input type="button" id="btngetjsondata" onclick="getData()"  value="GetData">
</body>
```

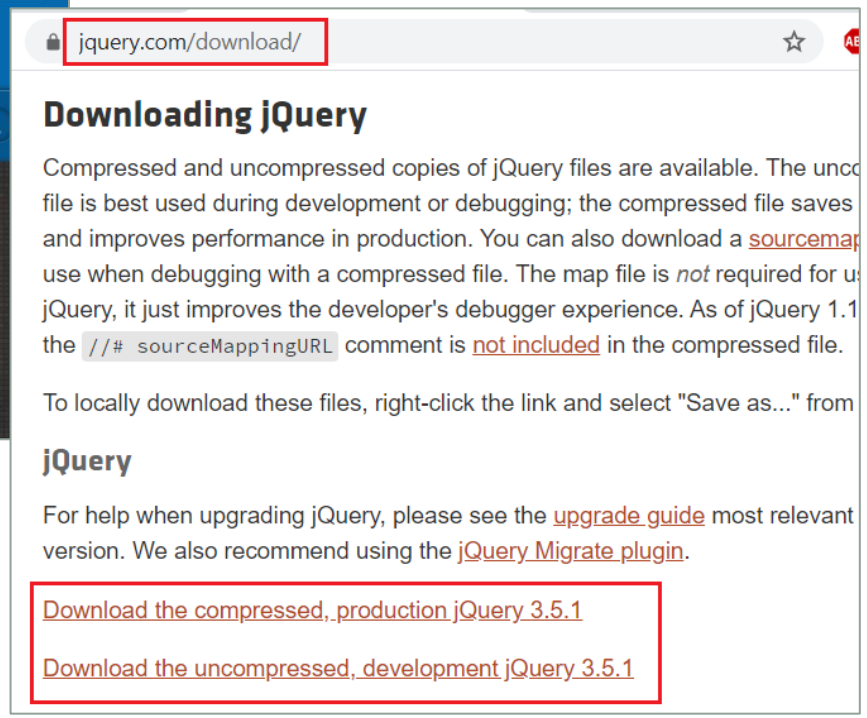



```
{ "name":"Kapil Verma",
  "desig":"ASE",
  "age":23,
  "sal":22000
}
```

# 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. DOM manipulation is by no means straightforward at the base level, and that's where jQuery comes in. It 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

```
<html>
<head>
<title>Test jQuery</title>
<script src="../scripts/jquery-3.5.1.min.js"></script>
</head>
<body>
   <!-- bidy of HTML -->
</body>
</html>
```

- 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).

```
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.1.1.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
```

# Using jQuery

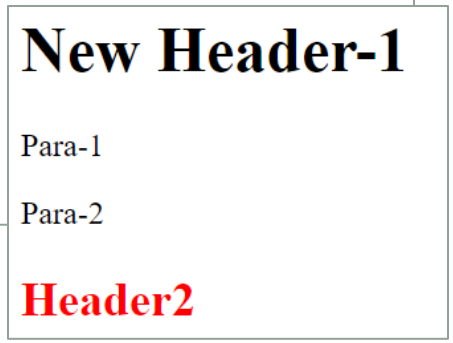
```
<html>
<head>
<title>Test jQuery</title>
<script type="text/javascript"  src="jquery-3.5.1.js"></script>
<script type="text/javascript">
$(document).ready(function()  {
      alert('Hi');
 });
</script>
</head>
<body>
     Welcome to jQuery
</body>
</html>
```

```
$(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()

```
<html>
<head>
<title>Test jQuery</title>
<script src="../scripts/jquery-3.5.1.min.js"></script>
<script>
$(document).ready(function(){
        $("h2").css("color","red");
        jQuery("h1").html("New Header-1");
    });
</script>
</head>
<body>
     <h1>jQuery Enabled</h1>
    <p>Para-1</p>
    <p>Para-2</p>
    <h2>Header2</h2>
</body>
</html>
```



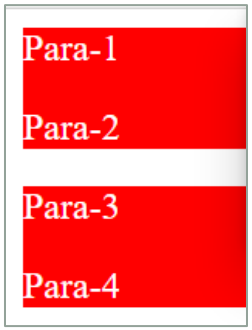
# Selectors

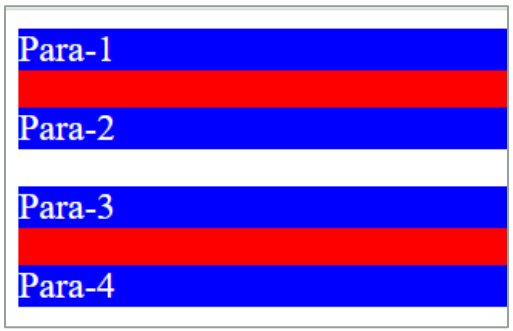
- **Selecting by Tag Name:**
- Selecting single tag takes the following syntax
  - $('p') – selects all <p> elements
  - $('a') – selects all <a> elements
- To reference multiple tags, use the ( , ) to separate the elements
  - $('p, a, span') - selects all paragraphs, anchors and span elements

- **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

```
<style type="text/css">
.redDiv{background-color:red; color:white;}
</style>
<script src="..\Scripts\jquery-3.5.1.js"></script>
<script>
$(document).ready(function() {
    var paragraphs = $('p');
    alert(paragraphs.length);   //4
    paragraphs.css('background-color','blue');
    paragraphs.each(function(){
       alert($(this).html());
    });
    var collections = $('div,p');
    alert(collections.length);    //6
    var bodydivs = $('body div');
    alert(bodydivs.length);    //2
});
</script>
</head>
<body>
<div class="redDiv" >
<p>Para-1</p>
<p>Para-2</p>
</div>
<div  class="redDiv">
<p>Para-3</p>
<p>Para-4</p>
</div>
</body>
```



# 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 "

```
<script src="..\Scripts\jquery-3.5.1.js"></script>
<script>
$(document).ready(function()
{
    alert($('#testDiv').html());
    $('#testDiv').html("Changed Text in Div!!");
    $("#p1").hide();
});
</script>
</head>
<body>
<div id="testDiv">Test Div</div>
<p id="p1"> Para-1</p>
<p>Para-2</p>
</body>
```

Test Div

Para-1

Para-2

Changed Text in Div!!

Para-2

# 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  tags with class="myclass"

```
<script>
$(document).ready(function(){
    $(".bold").css("font-weight", "bold");
});
</script>
</head>
<body>
<ul>
   <li class="bold">Test 1</li>
   <li>Test 2</li>
   <li class="bold">Test 3</li>
</ul>
</body>
```

- **Test 1**
- Test 2
- **Test 3**

```
<style type="text/css">
.blueDiv{background-color:lightblue; color:darkblue;}
.redDiv{background-color:orange; color:red;}
</style>
<script src="..\Scripts\jquery-3.5.1.js"></script>
<script>
$(document).ready(function(){
    var collection = $('.blueDiv');
    collection.css('border','5px solid blue');
    collection.css('padding','10px');
});
</script>
</head>
<body>
<div class="blueDiv">
<p>para-1</p>
</div>
<div class="redDiv">
<p>para-2</p>
</div>
<div class="blueDiv">
<p>para-3</p>
</div>
</body>
```

para-1

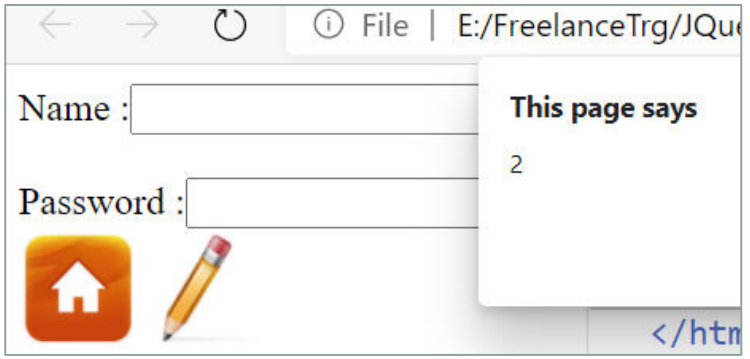
para-2

para-3

# 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

```
<script src="..\Scripts\jquery-3.5.1.js"></script>
<script>
$(document).ready(function()
{
    var list = $('div[title="first"],img[height]' );
    alert(list.length);   //2
});
</script>
</head>
<body>
<div title="first">
Name :<input type="text"/>
</div><br>
<div title="second" >
Password :<input type="password"/>
</div>
<img src="a.png" height="50" width="50">
<img src="b.png"  width="50">
</body>
```



# 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"

```
<script src="..\Scripts\jquery-3.5.1.js"></script>
<script>
$(document).ready(function()
{
    var inputs = $(':input');
    alert($(inputs[2]).val());  //Chennai
    $( ":text" ).css({ background: "yellow", border: "3px red solid" });
});
</script>
</head>
<body>
Name : <input id="txtName" type="text" value="Karthik"><br>
Age : <input id="txtAge" type="text" /><br>
City :
<select id="city">
<option value="Bangalore">Bangalore</option>
<option value="Chennai" selected="selected">Chennai</option>
<option value="Mumbai">Mumbai</option>
</select><br>
</body>
```

# Basic 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>
```

| | | |
|---|---|---|
| *TD #0* | *TD #1* | *TD #2* |
| TD #3 | TD #4 | TD #5 |
| TD #6 | TD #7 | TD #8 |

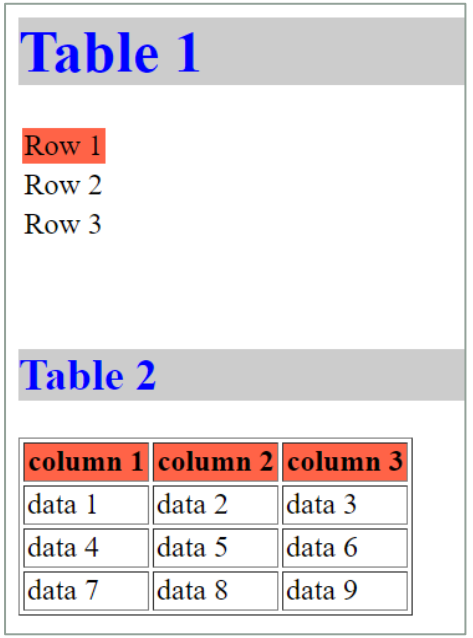
```
<script type="text/javascript">
$(document).ready(function() {
    $('tr:odd').css('background-color','tomato');
    $('tr:even').css('background-color','bisque');
});
</script>
<table border=1 cellspacing=5 cellpadding=5>
  <th>column 1<th>column 2<th>column 3
  <tr><td>data 1</td><td>data 2</td><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>
```

| column 1 | column 2 | column 3 |
|---|---|---|
| data 1 | data 2 | data 3 |
| data 4 | data 5 | data 6 |
| data 7 | data 8 | data 9 |
| data 10 | data 11 | data 12 |

# 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 can match more than one; one for each parent.
- :header - Selects all elements that are headers, like h1, h2, etc

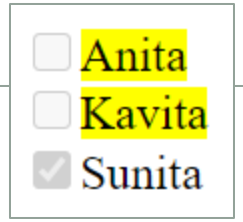
```
<script>
$(document).ready(function() {
    $( ":header" ).css({ background: "#ccc", color: "blue" });
    $('tr:first-child').css('background-color','tomato');
});
</script>
</head>
<body>
<h1>Table 1</h1>
<table>
  <tr><td>Row 1</td></tr>
  <tr><td>Row 2</td></tr>
  <tr><td>Row 3</td></tr>
</table>
<br/><br/>
<h2>Table 2</h2>
<table border=1>
<th>column 1<th>column 2<th>column 3
<tr><td>data 1</td><td>data 2</td><td>data 3
```



# Basic Filters continued

- :not - Selects all elements that do not match the given selector.
  - Eg : $("p:not(.intro)") - Select all <p> elements except those with class="intro"
  - Eg : $('a:not(div.important a, a.nav)'); - // Selects anchors that do not reside within 'div.important' or have the class 'nav'

```
<body>
 <div><input type="checkbox"><span>Anita</span></div>
 <div><input type="checkbox"><span>Kavita</span></div>
 <div><input type="checkbox" checked="checked"><span>Sunita</span></div>
 <script>
   $( "input:not(:checked) + span" ).css( "background-color", "yellow" );
   $( "input").attr( "disabled", "disabled" );
</script>
 </body>
```



# 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

```
  <style>
  span {color: blue; }
  </style>
</head>
<body>
  <ul>
    <li>John</li>
    <li>Karl</li>
    <li>Brandon</li>
  </ul> <hr>
  <ul><li>Sam</li></ul> <hr>
  <ul>
    <li>Glen</li>
    <li>Tane</li>
  </ul>
 <script>
$( "ul li:nth-child(2)" ).append( "<span> - 2nd!</span>" );
</script>
```

- John
- Karl - 2nd!
- Brandon

- Sam

- Glen
- Tane - 2nd!

```
 <style>
   span {color: #008;}
   span.sogreen { color: green; font-weight: bolder;}
   span.solast {text-decoration: line-through;}
</style>
<script src="..\Scripts\jquery-3.5.1.min.js"></script>
</head>
<body>
<div> <span>John,</span><span>Karl,</span><span>Brandon</span></div>
<div> <span>Glen,</span> <span>Tane,</span><span>Ralph</span> </div>
<script>
$( "div span:first-child" ) .css( "text-decoration", "underline" )
     .hover(function() {
          $( this ).addClass( "sogreen" );
        }, function() {
            $( this ).removeClass( "sogreen" );
       });
$( "div span:last-child" ).css({ color:"red", fontSize:"80%" })
.hover(function() {
      $( this ).addClass( "solast" );
   }, function() {
      $( this ).removeClass( "solast" );
    });
$("div span:nth-child(2)").css("background-color","yellow");
</script>
```

John,Karl,Brandon
Glen, Tane,Ralph

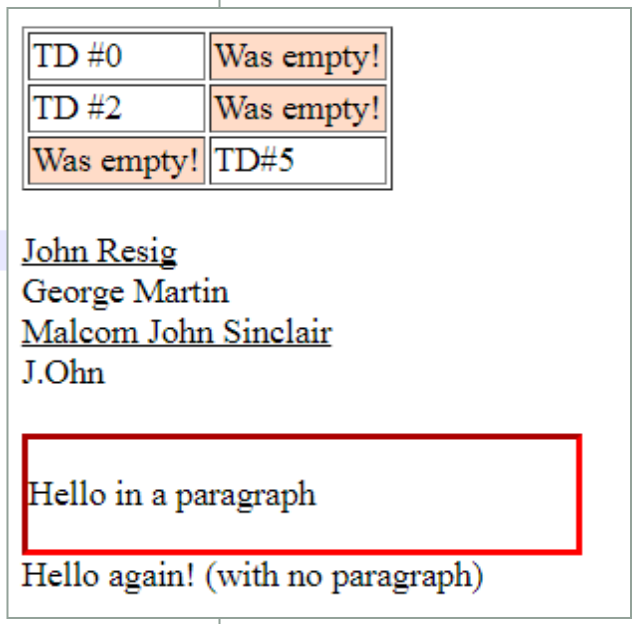
John,Karl,Brandon
Glen, Tane,Ralph

John,Karl,Brandon
Glen, Tane,~~Ralph~~

# 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  element inside of them
  - Eg : $("div:has(p,span,li)").css("border","solid red"); - Select all  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

```
 <style>
.test { border: 3px inset red; width:250px }
</style>
</head>
<body>
 <table border="1">
  <tr><td>TD #0</td><td></td></tr>
  <tr><td>TD #2</td><td></td></tr>
  <tr><td></td><td>TD#5</td></tr>
 </table>
 <br>
 <div>John Resig</div>
 <div>George Martin</div>
 <div>Malcom John Sinclair</div>
 <div>J.Ohn</div>
 <br>
 <div><p>Hello in a paragraph</p></div>
 <div>Hello again! (with no paragraph)</div>
 <script>
   $( "td:empty" ).text( "Was empty!" )
       .css( "background", "rgb(255,220,200)" );
   $( "div:contains('John')" ).css( "text-decoration", "underline" );
   $( "div:has(p)" ).addClass( "test" );
</script>
```



# 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“

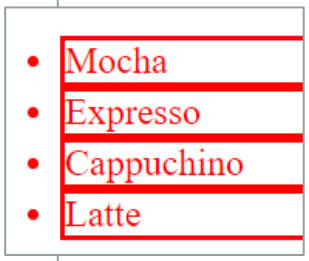
- .first() / last()
  - Eg : $("div p").first() - Select first <p> element inside first  element

- .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  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>

```
<ul class="ulclass">
  <li>Mocha</li>
  <li>Expresso</li>
  <li>Cappuchino</li>
  <li>Latte</li>
</ul>
<script>
$(".ulclass").children().css({"color":"red","border":"2px solid red"})
</script>
```




- .find() - Returns descendant elements of the selected element
  - Eg : $("ul").find("span").css({"color":"red","border":"2px solid red"}) - Return all  elements that are descendants of <ul>
- .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

```
 <style>
  .blue {background: blue;}
  .highlight{background-color: pink}
</style>
<script src="..\Scripts\jquery-3.5.1.min.js"></script>
</head>
<body>
  <div>
    <p>In inner para-1</p>
    <p class="intro">In inner para-2</p>
  </div>
  <p id="outro">In outer para-1</p>
  <p>In outer para-2</p>
  <ul>
    <li>list item 1</li>
    <li>list item 2</li>
    <li>list item 3</li>
    <li>list item 4</li>
    <li>list item 5</li>
  </ul>
<script>
 $( "li" ).eq(4).css( "background-color", "red" );
 $( "body" ).find( "li" ).eq( 2 ).addClass( "blue" );
 $("li").filter(":even").css("background-color","yellow");
 $("p").filter(".intro,#outro").css("background-color","blue");
 $( "div p" ).first().addClass( "highlight" );
</script>
```

In inner para-1

In inner para-2

In outer para-1

In outer para-2

- list item 1
- list item 2
- list item 3
- list item 4
- list item 5

# 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");**

# Iterating through Nodes

- .each(function(index,Element)) is used to iterate through jQuery objects.
  - jQuery 3 also allows to iterate over the DOM elements of a jQuery collection using the for...of loop.

```
<script>
$(document).ready(function(){
    $("li").each(function(){
        console.log($(this).text());
    });
    $("li").each (function (index){
        console.log(index+"="+$(this).text());
    });
    $("li").each (function (index,element){
       console.log(index+" : "+$(element).text());
    });
    $("li").each (function (index){
        this.title = "Index = "+index;
    });
});
</script>
</head>
<body>
<ul>
   <li>listitem-1</li>
   <li>listitem-2</li>
   <li>listitem-3</li>
</ul>
```

```
listitem-1
listitem-2
listitem-3
0=listitem-1
1=listitem-2
2=listitem-3
0 : listitem-1
1 : listitem-2
2 : listitem-3
```

```
▼<ul>
  ▶<li title="Index = 0">…</li>
  ▶<li title="Index = 1">…</li>
  ▶<li title="Index = 2">…</li>
```

- listitem-1
- listitem-2
- listitem-3

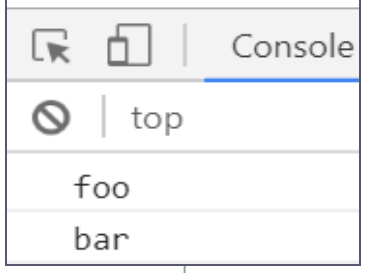
```
var i = 0;
for(var li of $("li")) {
    li.id = 'li-' + i++;
}
```

```
▼<ul>
  ▶<li id="li-0">…</li>
  ▶<li id="li-1">…</li>
  ▶<li id="li-2">…</li>
  </ul>
```

```
<script>
$(document).ready(function(){
    var result = "";
    $('div.One,div.Two,div.Three').each(function(index) {
        result+=index+" "+$(this).text()+"<br/>";
    });
    $('#OutputDiv').html(result);
    //try with element
    $("li").each(function(){
       console.log($(this).text())
    });
});
</script>
</head>
<body>
<ul>
   <li>foo</li>
   <li>bar</li>
</ul>
<div class="One"></div>
<div class="Two">Second Div</div>
<div class="Three">Third Div</div>
------------- output from jQuery --------------
<div id="OutputDiv" />
</body>
```

- foo
- bar

Second Div
Third Div
---------- output
0
1 Second Div
2 Third Div



# 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.

```
$("img").attr({
          "title":  "image title",
          "style" : "border:5px dotted red"

    });
```

```
<img src="paintbrush.gif" title="image
title" style="border:5px dotted red">
```

- You can also remove attributes entirely using .remo

```
<script>
$(document).ready(function(){
        var result = "";
        $('div').each(function(index){
            //raw DOM object
            this.title="Custom title";

            //using jQuery Object
            //$(this).attr('title','Modified title');
        });
        //using JSON to Modify multiple attributes
        $('div').attr({
                    title:'JSON Title',
                    style:'font-size:20pt;background-color:bisque;'
                  }).css('color','brown')
                    .css('text-transform','uppercase');

});
</script>
</head>
<body>
<div class="One">First Div</div>
<div class="Two">Second Div</div>
<div class="Three">Third Div</div>
</body>
```


attributes-demo.html

FIRST DIV
SECOND DIV
THIRD DIV

Elements Console Sources Network

```
<html>
▶ <head>…</head>
··· ▼ <body> == $0
    <div class="One" title="JSON Title"
    style="font-size: 20pt; background-color:
    bisque; color: brown; text-transform:
    uppercase;">First Div</div>
    <div class="Two" title="JSON Title"
    style="font-size: 20pt; background-color:
    bisque; color: brown; text-transform:
    uppercase;">Second Div</div>
```

# 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

```
<script>
$(document).ready(function(){
    $("#div1").html('<a href="example.html">Link</a><b>hello</b>');
    $("#div2").text('<a href="example.html">Link</a><b>hello</b>');
});
</script>
</head>
<body>
<div id="div1"></div>
<div id="div2"></div>
</body>
```

Linkhello

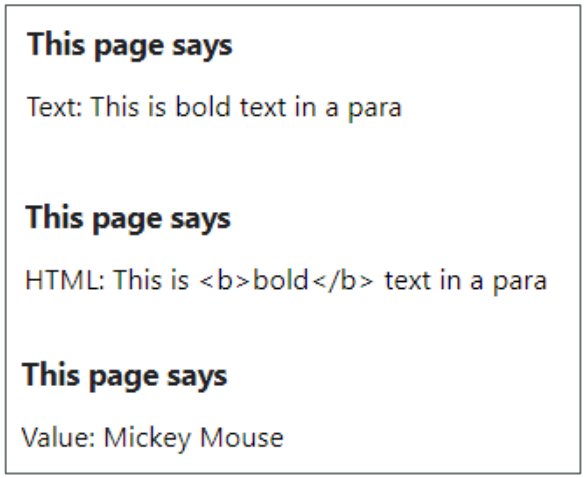
<a href="example.html">Link</a>hello

```
<div id="div1">
  <a href="example.html">Link</a>
  <b>hello</b>
</div>
<br>
<div id="div2"><a href="example.html">Link</a><b>hello</b>
</div>
```

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

# Getting Content

```
<script type="text/javascript">
    $(document).ready(function(){
        $("#btn1").click(function(){
            alert("Text: " + $("#test").text());
        });
        $("#btn2").click(function(){
            alert("HTML: " + $("#test").html());
        });
        $("#btn3").click(function(){
            alert("Value: " + $("#test1").val())
        });
});
</script>
</head>
<body>
<p id="test">This is <b>bold</b> text in a para</p>
<input id="test1" value="Mickey Mouse"><br>
<button id="btn1">Button1</button>
<button id="btn2">Button2</button>
<button id="btn3">Button3</button>
```




```
var input = $( 'input[type="text"]' );
input.val( 'new value' );
input.val();    // returns 'new value'
```

# 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(){
  $("#btn1").click(function(){
    $("p").prepend(" <b>Prepended text</b>.");
    $("h2").wrap("<header></header>");
  });
  $("#btn2").click(function(){
    $("ol").append("<li>Appended item</li>");
  });
  $("#btn3").click(function(){
    $( "h2" ).text("h2 position changed")
             .appendTo( $( "p" ) )
  });
});
</script>
</head>
<body>
 <h2>Greetings</h2>
 <p>This is a paragraph.</p>
 <p>This is another paragraph.</p>
 <ol>
   <li>List item 1</li>
   <li>List item 2</li>
   <li>List item 3</li>
 </ol>
 <button id="btn1">Prepend text</button>
 <button id="btn2">Append list items</button>
 <button id="btn3">AppendTo text</button>
```

## Greetings

This is a paragraph.

This is another paragraph.

1. List item 1
2. List item 2
3. List item 3

Prepend text | Append list items | AppendTo text

## Greetings

**Prepended text**.This is a paragraph.

**Prepended text**.This is another paragraph.

1. List item 1
2. List item 2
3. List item 3

Prepend text | Append list items | AppendTo text

## Greetings

**Prepended text**.This is a paragraph.

**Prepended text**.This is another paragraph.

1. List item 1
2. List item 2
3. List item 3
4. Appended item
5. Appended item

Prepend text | Append list items | AppendTo text

**Prepended text**.This is a paragraph.

## h2 position changed

**Prepended text**.This is another paragraph.

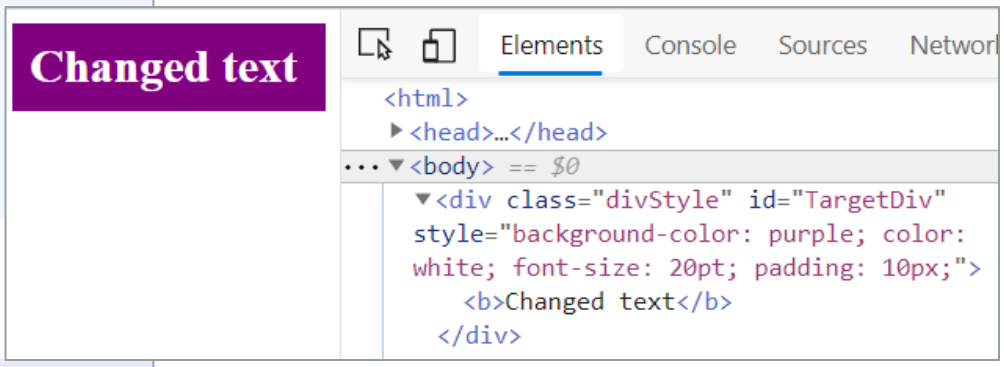
## h2 position changed

# Modifying Styles

- .css() function is used to modify an object's style
  - $('div').css('color','red');
- Multiple styles can be modified by passing a JSON Object

```
$('div').css({
   "color":"red",
   "font-weight":"bold"
});
```

```
<script>
$(document).ready(function(){
      $('#TargetDiv').css({
            'background-color':'purple',
            'color':'white',
            'font-size':'20pt',
            'padding':'10px'
      })
      .html('<b>Changed text</b>');
});
</script>
</head>
<body>
<div id="TargetDiv">Target Div</div>
</body>
```



# 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');

```
<style>
.important{
  font-weight:bold;
  font-size:xx-large;
  color:red;
}
.blue {color:blue;}
</style>
<script>
$(document).ready(function() {
    $("button").click(function(){
        $("h1,h2,p").addClass("blue");
        $("div").addClass("important");
  });
});
</script>
</head>
<body>
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<div>This is some important text!</div>
<button>Add classes to elements</button>
```

# Heading 1

## Heading 2

This is a paragraph.

This is another paragraph.

This is some important text!

Add classes to elements

# Heading 1

## Heading 2

This is a paragraph.

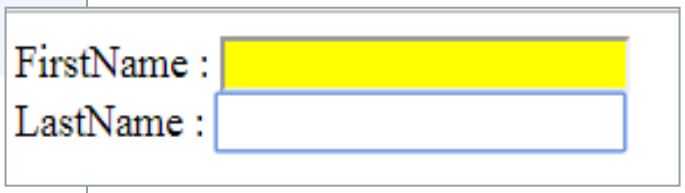
This is another paragraph.

This is some

Add classes to elements

# Working with Classes (Contd)

```
<style type="text/css">
.highlight{ background-color:yellow; };
</style>
<script>
  $(document).ready(function() {
    $('input[type="text"]').addClass('highlight');
    //$('#txtLastName').removeClass('highlight');
  });
  function FocusBlur(element){
    $(element).toggleClass('highlight');
  }
</script>
</head>
<body>
<table>
<tr>
<td>FirstName : </td>
<td><input id="txtFirstName" onFocus="FocusBlur(this)"
        onBlur="FocusBlur(this)" /></td>
</tr>
<tr>
<td>LastName : </td>
<td><input id="txtLastName" onFocus="FocusBlur(this)"
        onBlur="FocusBlur(this)"/></td>
</tr>
</table>
```



# jQuery Event Model Benefits

- Events notify a program that a user performed some type of action
- jQuery Events
  - click()
  - blur()
  - focus()
  - dblclick()
  - mousedown()
  - mouseup()
  - mouseover()
  - keydown()
  - keypress()
  - hover() : hover() method takes two functions and is a combination of the mouseenter() and mouseleave() methods.

```
$("img").mouseover(function () {
        $(this).css("opacity", "0.3");
 });
 $("img").mouseout(function () {
         $(this).css("opacity", "1.0");
 });
```

```
$("#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') });

```
<script>
$(document).ready(function(){
    $('#SubmitButton').click(function(){
        var fName = $('#txtFirstName').val();
        var lName = $('#txtLastName').val();
        $('#tgtDiv').html("<b>"+fName+" "+lName+"</b>");
    });
});
</script>
</head>
<body>
FirstName : <input id="txtFirstName" type="text" /><br>
LastName : <input id="txtLastName" type="text" /><br>
<input id="SubmitButton" type="submit" value="Submit"/><br>
<div id="tgtDiv" />
</body>
```

FirstName : Anil

LastName : Patil

Submit

**Anil Patil**

```
<style>
  p {
    color: red;
    margin: 5px;
    cursor: pointer;
  }
  p:hover {background: yellow;}
  </style>
</head>
<body>
<p>First Paragraph</p>
<p>Second Paragraph</p>
<p>Yet one more Paragraph</p>

<script>
 $( "p" ).click(function() {
   $( this ).slideUp();
});
</script>
</body>
```

First Paragraph

Second Paragraph

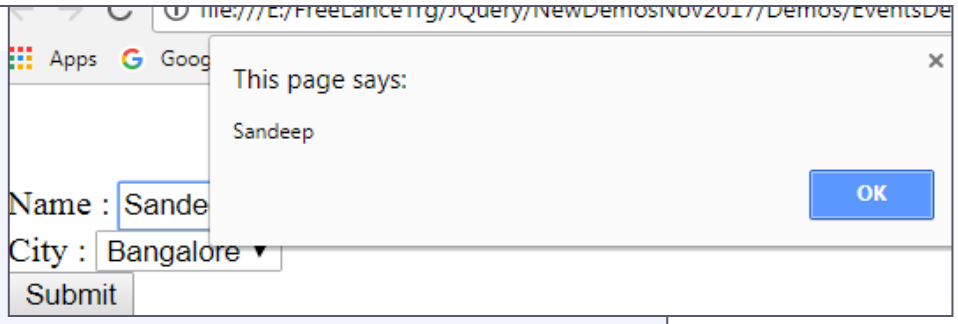
Yet one more Paragraph

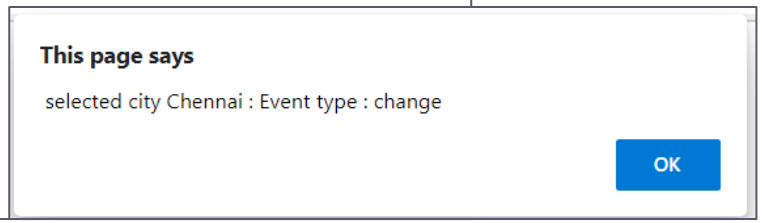
```
<style type="text/css">
.highlight{background-color:yellow;};
</style>
<script src="..\Scripts\jquery-3.5.1.js"></script>
<script>
        $(document).ready(function(){
            $('#targetDiv').mouseenter(function(){
                toggle(this);
            });
            $('#targetDiv').mouseleave(function(){
                toggle(this);
            });
            $('#targetDiv').mouseup(function(e){
                $(this).text('X : '+e.pageX+' Y :'+e.pageY );
            });
        });
        function toggle(element){
            $(element).toggleClass('highlight');
        }
</script>
</head>
<body>
<div id="targetDiv">Welcome to events</div>
</body>
```

Welcome to events

X : 123 Y :17

```
<script>
$(document).ready(function() {
    $('.testClass').change(function(){
        alert($(this).val());
    });
    $('#City').change(disp);

    function disp(e){
       alert(" selected city " + $(this).val() + " : Event type : " + e.type)
    }
});
</script>
</head>
<body><BR><BR>
Name : <input id="txtName" class="testClass" autofocus/>
City :
<select id="City" >
  <option value="Bangalore">Bangalore</option>
  <option value="Chennai" >Chennai</option>
  <option value="Mumbai">Mumbai</option>
</select><br>
<input type="submit" value="Submit"/></td>
</body>
```



# 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() );
});
```

```
<script>
$(document).ready(function() {
  $("#div1").on("click",function(){
     $(this).css("background-color","pink");
  });
  $("h2").on("mouseover mouseout",function(){
     $(this).toggleClass("intro");
  });
  $("#div2").on({
    mouseover: function(){
      $(this).css("background-color", "lightgray");
    },
    mouseout: function(){
      $(this).css("background-color", "lightblue");
    },
    click: function(){
      $(this).hide();
    }
  });
});
</script>
```

**Header-2**

**Text within h4 element**

Text within para

This is Div-2

**Header-2**

**Text within h4 element**

Text within para

```
<body>
<h2>Header-2</h2>
<div id="div1">
 <h4>Text within h4 element</h4>
 <p>Text within para</p>
</div>
<div id="div2"> This is Div-2 </div>
</body>
```

# 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)

```
<style type="text/css">
.blueDiv{
  background-color:blue;
  color:white;font-size:30pt;
  display:none
  }
</style>
<script src="..\Scripts\jquery-3.5.1.js"></script>
<script>
$(document).ready(function(){
   $('#btnShow').click(function(){
      //fast(200),normal(400) and slow(600) can also be s
      $('.blueDiv').show(5000,function(){
         $('#results').text('Show Animation Completed');
      });
    });
    $('#btnHide').click(function(){
      $('.blueDiv').hide(5000,function(){
        $('#results').text('Hide Animation Completed');
       });
    });
    $('#btnSH').click(function(){
      $('.blueDiv').toggle(5000,function(){
        $('#results').text('Animation Completed');
      });
    });
});
</script>
```


```
<body>
<input id="btnShow" type="button" value="Show"/>
<input id="btnHide" type="button" value="Hide"/>
<input id="btnSH" type="button" value="Show/Hide"/>
<div class="blueDiv">
<p>Welcome to animations!!</p>
</div>
<div id="results"></div>
</body>
```

# jQuery Sliding Effects

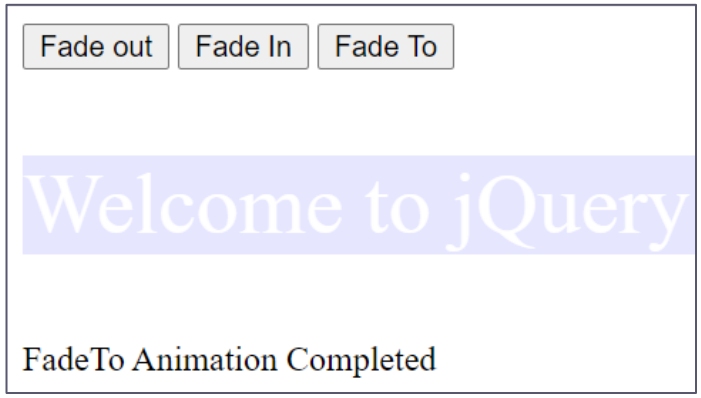
- The jQuery slide methods slide elements up and down.
- jQuery has the following slide methods:
  - $(selector).slideDown(speed,callback) : *Display the matched elements with a sliding motion*
  - $(selector).slideUp(speed,callback) : *Hide the matched elements with a sliding motion.*
  - $(selector).slideToggle(speed,callback)
- 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.

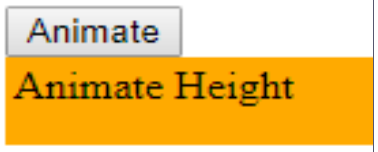
```
$('.blueDiv').fadeTo(1000,0.1,function(){
        $('#results').text('FadeTo Animation Completed');
});
```

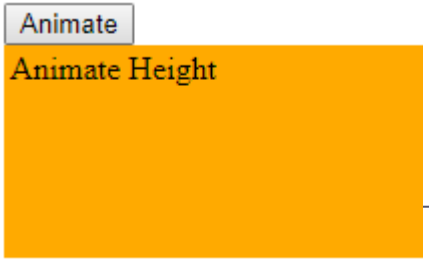


# Creating Custom Animation

- 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 Ajax features

- The jQuery library has a full suite of Ajax capabilities.
  - The functions and methods therein allow us to load data from the server without a browser page refresh.
  - $(selector).load() : Loads HTML data from the server
  - $.get() and $.post() : Get raw data from the server
  - $.getJSON() : Get / Post and return JSON data
  - $.ajax() : Provides core functionalityjQuery Ajax functions works with REST APIs, Webservices and more

# 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);
});
```

- $.post(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
}
});
```

```
$("button").click(function(){
    $.ajax({
        url: "demo_test.txt",
        success: function(result){
            $("#div1").html(result);
        }
     });
});
```