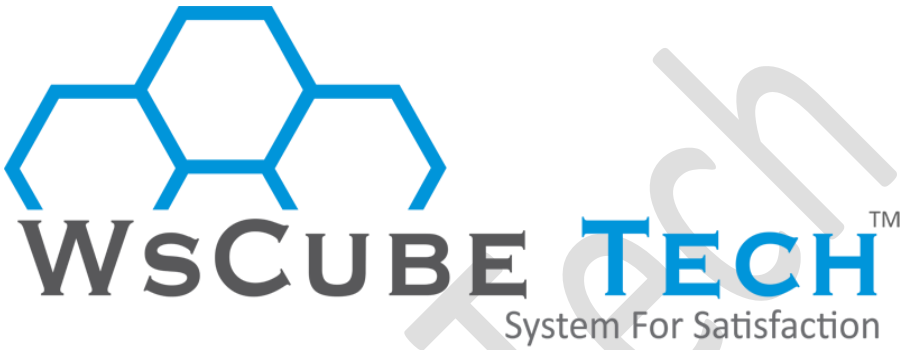
WsCube Tech

1st Floor, Laxmi Tower, Bhaskar Circle, Ratanada,
Jodhpur.
Mobile : +91-92696-98122 , 85610-89567



HTML5 CUBE

Building Static Websites with HTML5

BY **WsCube Tech**

HTML5 was released on 22 January 2008, with a major update and "W3C Recommendation" status in October 2014. Its goals were to improve the language with support for the latest multimedia and other new features; to keep the language both easily readable by humans and consistently understood by computers and devices such as web browsers, parsers, etc., without XHTML's rigidity; and to remain backward-compatible with older software.

HTML5 Semantic Elements

Semantic HTML elements clearly describe it's meaning in a human and machine readable way. Elements such as <header>, <footer> and <article> are all considered semantic because they accurately describe the purpose of the element and the type of content that is inside them.

Declaring that the document contains HTML5 mark-up with the HTML5 doctype
<!DOCTYPE html>

Declaring the character set with the <meta charset>
<meta charset="UTF-8">

<article> :- An article should make sense on its own, and it should be possible to read it independently from the rest of the web site.
For example an <article> element can be used for Post, Blog, and Newspaper Article.

Example :-

```
<article>
  <h1>WsCube Tech</h1>
  <p>For Us our Students is our top priority. Welcome to WsCube Tech</p>
</article>
```

<aside> :- The <aside> element defines some content aside from the content it is placed in (like a sidebar).

Example :-

```
<aside>
  <h1>WsCube Tech</h1>
  <p>For Us our Students is our top priority. Welcome to WsCube Tech </p>
</aside>
```

<header> :- The <header> element specifies a header for a document or section. It should be used as a container for introductory content.

Example :-

```
<header>
  <h1>WsCube Tech</h1>
  <p>For Us our Students is our top priority. The training programme and curriculum has  designed in such a smart way that the students could get familiar with industrial professionalism  since the beginning of the training and till the completion of the curriculum.</p>
</header>
```

<footer> :- The <footer> element specifies a footer for a document or section. It should contain information about its containing element.

Example :-

```
<footer>
  <p>For Us our Students is our top priority. The training programme and curriculum has designed in such a smart way that the students could get familiar with industrial professionalism since the beginning of the training and till the completion of the curriculum.</p>
</footer>
```

<nav> :- The <nav> element defines a set of navigation links.

Example :-

```
<nav>
  <a href="home"> Home </a> |
 <a href="home"> about </a> |
 <a href="home"> gallery</a> |
 <a href="home"> courses </a> |
 <a href="home"> contact </a> |
</nav>
```

<figure> and <figcaption> :- The purpose of a figure caption is to add a visual explanation to an image.

Example :-

```
<figure>
   <img src="images/image.png">
   <figcaption>image</figcaption>
</figure>
```

<summary> :- Defines a summary for the <details> element.

<details> :- <details> represents a widget from which the user can obtain additional information or controls on-demand.

Example :-

```
<details>
 <summary>WsCube Tech</summary>
  <p>For Us our Students is our top priority. The training programme and curriculum has designed in such a smart way that the students could get familiar with industrial professionalism since the beginning of the training and till the completion of the curriculum.</p>
</details>
```

<section> :- <section> Defines a section of a document, such as header, footer etc.

Example :-

```
<section>
 <h1>Welcome to WsCube Tech</h1>
  <p>Wscube tech</p>
</section>
```

<datalist> :- Represents a set of pre-defined options for an <input> element.

Example :-

```
<datalist id="list">
   <option value="php">
   <option value="java">
   <option value="html">
</datalist>
```

<mark> :- <section> represents text highlighted for reference purposes.

Example :-

```
<p>Welcome to <mark>Wscube</mark> tech</p>
```

<source> :- Defines alternative media resources for the media elements like <audio> or <video>.

<audio> :- Embeds a sound, or an audio stream in an HTML document.

Example :-

```
<audio controls="controls" src="demo.mp3"></audio>

OR

<audio controls="controls">
```

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

<video> :- Represents a time and/or date.

Example :-

```
<video controls="controls" src="demo.mp4">
</video>

OR

<video controls="controls">
  <source src="demo.mp4" type="video/mp4">
  <source src="demo.ogv" type="video/ogg">
</video>
```

<meter> :- <meter> represents a scalar measurement within a known range.

Example :-

```
<p>HTML: <meter value="0.85">85%</meter> </p>
<p>CSS: <meter value="8" min="0" max="10">8 out of 10</meter> </p>
<p>HTML5: <meter low="60" high="80" max="100" value="85">Very High</meter> </p>
```

<output> :- <optput> represents the result of a calculation.

Example :-

```
<output name="result" for="a b"></output>
```

HTML5 introduces new input types for forms.

Seacrh :- search type, define a search field (Google search, bing search).

Range :- range type, define a range control (like a limitation control).

Example :-

```
<input type="range" min="0" max="10">
```

email :- used for input fields that should contain an e-mail address.

Example :-

```
<input type="email" name="email">
```

password :- used for input fields that should contain a password.

Example :-

```
<input type="password" name="password">
```

Date :- It used for input fields that should contain a date.

Example :-

```
<input type="date" name="registeration">
```

datetime-local :- A date and time input field, with no time zone.

Example :-

```
<input type="datetime-local" name="bday">
```

Url :- It used for input fields that should contain a URL address.

Example :-

```
<input type="url" name="home_page_url">
```

Month :- It allows the user to select a month and year.

Example :-

```
<input type="month" name="month">
```

Color :- It sed for input fields that should contain a color.

Example :-

```
<input type="color" name="color">
```

tel :- It used for input fields that should contain a telephone number.

Example :-

```
<input type="tel" name="telephone">
```

Week :- It allows the user to select a week and year

Example :-

```
<input type="week" name="week">
```

number :- number type, define a numeric input field.

Example :-

```
<input type="number" name="mobile_number">
```

Time :- Its allows the user to select a time (no time zone).

Example :-

```
<input type="time" name="time">
```

HTML5 introduce new attributes.

placeholder :- Placeholder attribute specifies a short hint that describes the expected value of an input field.

Choices :-
1. user define

Example :-

```
<input  type="text" name="name" placeholder="Enter Name" >
```

align :- Horizontally text align .

Choices :-
1. left
2. Right
3. center

Example :-

```
<div align="center" >WsCube Tech</div>
```

background :- place an image in background.

Choices :-
1. url

Example :-

```
<div align="center" background="images/demo.jpg" >WsCube Tech</div>
```

bgcolor :- place a color in background.

Choices :-
1. color name
2. color code
3. RGB value

Example :-

```
<div bgcolor="#F0F0F4" >WsCube Tech</div>
```

hidden :- Specifies whether element should be visible or not.

Choices :-
1. hidden

Example :-

```
<div hidden>WsCube Tech</div>
```

id :- names an element should for use with css.

Choices :-
1. user define

Example :-

```
<div id="name" >WsCube Tech</div>
```

style :- Specifies an inline css.

Choices :-
1. CSS

Example :-

```
<div style="width:100px;" >WsCube Tech</div>
```

title :- pop-up title when hover an element.

Choices :-
1. user define

Example :-

```
<div title="name" >WsCube Tech</div>
```

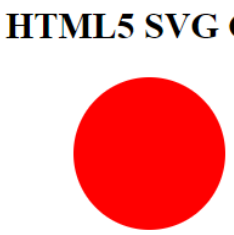
SVG

SVG (Scalable Vector Graphics) is a language for describing 2D-graphics and graphical applications in XML and the XML is then rendered by an SVG viewer.
SVG is mostly useful for vector type diagrams like Pie charts, Two-dimensional graphs in an X,Y coordinate system etc.
Most of the web browsers can display SVG just like they can display PNG, GIF, and JPG. Internet Explorer users may have to install the Adobe SVG Viewer to be able to view SVG in the browser.

Circle :- HTML5 version of an SVG example which would draw a circle using <circle> tag.

Example :-

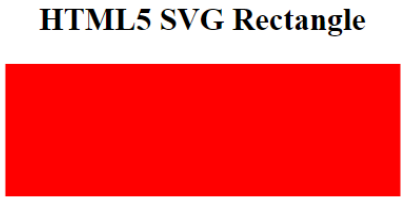
```
<html>
  <head>
        <style>
        #svgelem {
        position: relative;
        left: 50%;
        -webkit-transform: translateX(-20%);
        -ms-transform: translateX(-20%);
        transform: translateX(-20%);
        }
        </style>
        <title>SVG</title>
        <meta charset = "utf-8" />
  </head>
  <body>
        <h2 align = "center">HTML5 SVG Circle</h2>
        <svg id = "svgelem" height = "200">
        <circle id = "redcircle" cx = "50" cy = "50" r = "50" fill = "red" />
        </svg>
  </body>
</html>
```



Rectangle :- HTML5 version of an SVG example which would draw a rectangle using <rec> tag.

Example :-

```
<!DOCTYPE html>
<html>
  <head>
        <style>
        #svgelem {
        position: relative;
        left: 50%;
        -webkit-transform: translateX(-50%);
        -ms-transform: translateX(-50%);
        transform: translateX(-50%);
        }
        </style>
        <title>SVG</title>
        <meta charset = "utf-8" />
  </head>
  <body>
        <h2 align = "center">HTML5 SVG Rectangle</h2>
        <svg id = "svgelem" height = "200" >
        <rect id = "redrect" width = "300" height = "100" fill = "red" />
        </svg>
  </body>
</html>
```

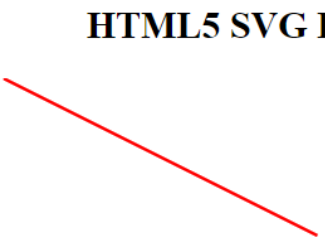


Line :- HTML5 version of an SVG example which would draw a line using <line> tag.

Example :-

```
<!DOCTYPE html>
<html>
  <head>
        <style>
        #svgelem {
        position: relative;
        left: 50%;
        -webkit-transform: translateX(-50%);
```

```
        -ms-transform: translateX(-50%);
        transform: translateX(-50%);
        }
        </style>
        <title>SVG</title>
         <meta charset = "utf-8" />
  </head>
  <body>
        <h2 align = "center">HTML5 SVG Line</h2>
        <svg id = "svgelem" height = "200" >
        <line x1 = "0" y1 = "0" x2 = "200" y2 = "100"
        style = "stroke:red;stroke-width:2"/>
        </svg>
  </body>
</html>
```

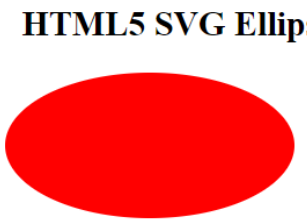


Ellipse :- HTML5 version of an SVG example which would draw a ellipse using <ellipse> tag.

Example :-

```
<!DOCTYPE html>
<html>
  <head>
        <style>
        #svgelem {
        position: relative;
        left: 50%;
        -webkit-transform: translateX(-40%);
        -ms-transform: translateX(-40%);
        transform: translateX(-40%);
        }
        </style>
        <title>SVG</title>
        <meta charset = "utf-8" />
  </head>
  <body>
        <h2 align = "center">HTML5 SVG Ellipse</h2>
```

```
        <svg id = "svgelem" height = "200" >
        <ellipse cx = "100" cy = "50" rx = "100" ry = "50" fill = "red" />
        </svg>
  </body>
</html>
```



Polygon :- HTML5 version of an SVG example which would draw a Polygon using <polygon> tag.

Example :-

```
<html>
  <head>
        <style>
        #svgelem {
        position: relative;
        left: 50%;
        -webkit-transform: translateX(-50%);
        -ms-transform: translateX(-50%);
        transform: translateX(-50%);
        }
        </style>
        <title>SVG</title>
        <meta charset = "utf-8" />
  </head>
  <body>
        <h2 align = "center">HTML5 SVG Polygon</h2>
        <svg id = "svgelem" height = "200" >
        <polygon  points = "20,10 300,20, 170,50" fill = "red" />
        </svg>
  </body>
</html>
```

Polyline:- HTML5 version of an SVG example which would draw a Polyline using <polyline> tag.

Example :-

```
<!DOCTYPE html>
<html>
  <head>
        <style>
        #svgelem {
        position: relative;
        left: 50%;
        -webkit-transform: translateX(-20%);
        -ms-transform: translateX(-20%);
        transform: translateX(-20%);
        }
    </style>
        <title>SVG</title>
        <meta charset = "utf-8" />
  </head>
  <body>
        <h2 align = "center">HTML5 SVG Polyline</h2>
        <svg id = "svgelem" height = "200">
        <polyline points = "0,0 0,20 20,20 20,40 40,40 40,60" fill = "red" />
        </svg>
  </body>
</html>
```

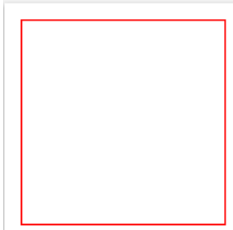
Canvas

HTML5 element <canvas> gives you an easy and powerful way to draw graphics using JavaScript. It can be used to draw graphs, make photo compositions or do simple (and not so simple) animations.
Here is a simple <canvas> element which has only two specific attributes width and height plus all the core HTML5 attributes like id, name and class, etc.

<canvas> :-

Example :-

```
<!DOCTYPE HTML>
<html>
  <head>
        <style>
        #mycanvas{border:1px solid red;}
        </style>
  </head>
  <body>
        <canvas id = "mycanvas" width = "100" height = "100"></canvas>
  </body>
</html>
```



The Rendering Context

The <canvas> is initially blank, and to display something, a script first needs to access the rendering context and draw on it.
The canvas element has a DOM method called getContext, used to obtain the rendering context and its drawing functions. This function takes one parameter, the type of context2d.

Following is the code to get required context along with a check if your browser supports <canvas> element –

```
var canvas  = document.getElementById("mycanvas");
if (canvas.getContext) {
  var ctx = canvas.getContext('2d');
  // drawing code here
} else {
  // canvas-unsupported code here
}
```

Note :- The latest versions of Firefox, Safari, Chrome and Opera all support for HTML5 Canvas but IE8 does not support canvas natively.
You can use ExplorerCanvas to have canvas support through Internet Explorer. You just need to include this JavaScript as follows :-
<!--[if IE]><script src = "excanvas.js"></script><![endif]-->

Canvas - Drawing Bezier Curves :-

beginPath() :- This method resets the current path.

moveTo(x, y) :- This method creates a new subpath with the given point.

closePath() :- This method marks the current subpath as closed, and starts a new subpath with a point the same as the start and end of the newly closed subpath.

fill() :- This method fills the subpaths with the current fill style.

stroke() :- This method strokes the subpaths with the current stroke style.

bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y) :- This method adds the given point to the current path, connected to the previous one by a cubic Bezier curve with the given control points.

The x and y parameters in bezierCurveTo() method are the coordinates of the end point. cp1x and cp1y are the coordinates of the first control point, and cp2x and cp2y are the coordinates of the second control point.

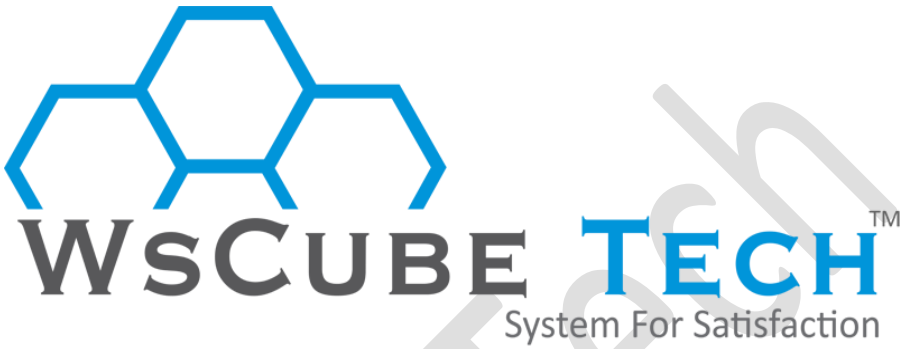
Example :-

```
<!DOCTYPE HTML>
<html>
  <head>
        <style>
        #test {
        width: 100px;
        height:100px;
        margin: 0px auto;
        }
        </style>
        <script type = "text/javascript">
        function drawShape() {
        // get the canvas element using the DOM
              var canvas = document.getElementById('mycanvas');
        // Make sure we don't execute when canvas isn't supported
        if (canvas.getContext) {
        // use getContext to use the canvas for drawing
```

```
        var ctx = canvas.getContext('2d');
        ctx.beginPath();
        ctx.moveTo(75,40);
          ctx.bezierCurveTo(75,37,70,25,50,25);
          ctx.bezierCurveTo(20,25,20,62.5,20,62.5);
          ctx.bezierCurveTo(20,80,40,102,75,120);
          ctx.bezierCurveTo(110,102,130,80,130,62.5);
          ctx.bezierCurveTo(130,62.5,130,25,100,25);
          ctx.bezierCurveTo(85,25,75,37,75,40);
        ctx.fill();
        } else {
        alert('You need Safari or Firefox 1.5+ to see this demo.');
        }
        }
        </script>
  </head>
  <body id = "test" onload = "drawShape();">
        <canvas id = "mycanvas"></canvas>
  </body>
</html>
```

WsCube Tech

1st Floor, Laxmi Tower, Bhaskar Circle, Ratanada,
Jodhpur.
Mobile : +91-92696-98122 , 85610-89567



CSS3 CUBE

Building Attractive Websites with CSS3

BY **WsCube Tech**

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. CSS3 is a latest standard of css earlier versions(CSS2). The main difference between css2 and css3 is follows –

1. Media Queries
2. Namespaces
3. Selectors Level 3
4. Color

CSS3 modules
CSS3 is collaboration of CSS2 specifications and new specifications, we can called this collaboration is module. Some of the modules are shown below –

1. Selectors
2. Box Model
3. Backgrounds
4. Image Values and Replaced Content
5. Text Effects
6. 2D Transformations
7. 3D Transformations
8. Animations
9. Multiple Column Layout
10. User Interface

CSS3 Rounded corners are used to add special colored corner to body or text by using the border-radius property.A simple syntax of rounded corners is as follows −

```
#rcorners {
   border-radius: 60px/15px;
   background: #FF0000;
   padding: 20px;
   width: 200px;
   height: 150px;
}
```

The possible values for Rounded corners :-

border-radius :- Use this element for setting four boarder radius property
border-top-left-radius :- Use this element for setting the boarder of top left corner
border-top-right-radius :- Use this element for setting the boarder of top right corner
border-bottom-right-radius :- Use this element for setting the boarder of bottom right corner
border-bottom-left-radius :- Use this element for setting the boarder of bottom left corner

Example :-

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

```
    <style>
      #rcorners1 {
        border-radius: 25px;
        background: #8AC007;
        padding: 20px;
        width: 200px;
        height: 150px;
      }
      #rcorners2 {
        border-radius: 25px;
        border: 2px solid #8AC007;
        padding: 20px;
        width: 200px;
        height: 150px;
      }
    </style>
  </head>

  <body>
    <p id = "rcorners1">Rounded corners!</p>
    <p id = "rcorners2">Rounded corners!</p>
  </body>
</html>
```

Each Corner property

We can specify the each corner property as shown below example :−

```
<html>
  <head>
    <style>
      #rcorners1 {
        border-radius: 15px 50px 30px 5px;
        background: #a44170;
        padding: 20px;
        width: 100px;
        height: 100px;
        Float : left;
      }
      #rcorners2 {
        border-radius: 15px 50px 30px;
        background: #a44170;
        padding: 20px;
        width: 100px;
        height: 100px;
        Float : left;
      }
      #rcorners3 {
```

```
            border-radius: 15px 50px;
            background: #a44170;
            padding: 20px;
            width: 100px;
            height: 100px;
            Float : left;
         }
      </style>
   </head>

   <body>
      <p id = "rcorners1"></p>
      <p id = "rcorners2"></p>
      <p id = "rcorners3"></p>
   </body>
<body>
```

CSS Border image property is used to add image boarder to some elements.you don't need to use any HTML code to call boarder image.

Syntax example of boarder :−

```
#borderimg {
   border: 10px solid transparent;
   padding: 15px;
}
```

The most commonly used values are shown below −

border-image-source :- Used to set the image path
border-image-slice :- Used to slice the boarder image
border-image-width :- Used to set the boarder image width
border-image-repeat :- Used to set the boarder image as rounded, repeated and stretched

Example :-

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

```
    <style>
      #borderimg1 {
        border: 10px solid transparent;
        padding: 15px;
        border-image-source: url(/css/images/border.png);
        border-image-repeat: round;
        border-image-slice: 30;
        border-image-width: 10px;
      }
      #borderimg2 {
        border: 10px solid transparent;
        padding: 15px;
        border-image-source: url(/css/images/border.png);
        border-image-repeat: round;
        border-image-slice: 30;
        border-image-width: 20px;
      }
      #borderimg3 {
        border: 10px solid transparent;
        padding: 15px;
        border-image-source: url(/css/images/border.png);
        border-image-repeat: round;
        border-image-slice: 30;
        border-image-width: 30px;
      }
    </style>
  </head>

  <body>
    <p id = "borderimg1">This is image boarder example.</p>
    <p id = "borderimg2">This is image boarder example.</p>
    <p id = "borderimg3">This is image boarder example.</p>
  </body>
</html>
```

Multi Background

CSS Multi background property is used to add one or more images at a time without HTML code, We can add images as per our requirement.
Syntax example of multi background images :−

```
#multibackground {
   background-image: url(/css/images/logo.png), url(/css/images/border.png);
   background-position: left top, left top;
   background-repeat: no-repeat, repeat;
   padding: 75px;
}
```

The most commonly used values are shown below :-

background :- Used to setting all the background image properties in one section
background-clip :- Used to declare the painting area of the background
background-image :- Used to specify the background image
background-origin :- Used to specify position of the background images
background-size :- Used to specify size of the background images

Size of Multi background

Multi background property is accepted to add different sizes for different images.

Syntax example :−

```
#multibackground {
   background: url(/css/imalges/logo.png) left top no-repeat, url(/css/images/boarder.png) right bottom no-repeat,
url(/css/images/css.gif) left top repeat;
   background-size: 50px, 130px, auto;
}
```

CSS3 has Supported additional color properties as follows −

1. RGBA colors
2. HSL colors
3. HSLA colors
4. Opacity

RGBA stands for **Red Green Blue Alpha**.It is an extension of CSS2,Alpha specifies the opacity of a color and parameter number is a numerical between 0.0 to 1.0.

Syntax example of RGBA :−

```
#d1 {background-color: rgba(255, 0, 0, 0.5);}
#d2 {background-color: rgba(0, 255, 0, 0.5);}
```

```
#d3 {background-color: rgba(0, 0, 255, 0.5);}
```

HSL stands for **hue, saturation, lightness**.Here Huge is a degree on the color wheel, saturation and lightness are percentage values between 0 to 100%.

Syntax example of HSL :-

```
#g1 {background-color: hsl(120, 100%, 50%);}
#g2 {background-color: hsl(120, 100%, 75%);}
#g3 {background-color: hsl(120, 100%, 25%);}
```

HSLA stands for **hue, saturation, lightness and alpha**. Alpha value specifies the opacity as shown RGBA.

syntax example of HSLA :−

```
#g1 {background-color: hsla(120, 100%, 50%, 0.3);}
#g2 {background-color: hsla(120, 100%, 75%, 0.3);}
#g3 {background-color: hsla(120, 100%, 25%, 0.3);}
```

opacity is a thinner paints need black added to increase opacity.

Syntax example of opacity :−

```
#g1 {background-color:rgb(255,0,0);opacity:0.6;}
#g2 {background-color:rgb(0,255,0);opacity:0.6;}
#g3 {background-color:rgb(0,0,255);opacity:0.6;}
```

CSS3 supported to add shadow to text or elements.Shadow property has divided as follows −

1. Text shadow
2. Box Shadow
3. Text shadow

CSS3 supported to add shadow effects to text. Following is the example to add shadow effects to text −

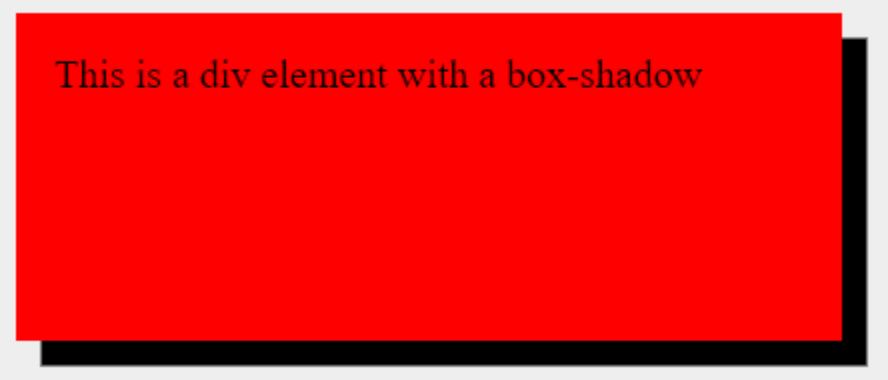
```
<html>
  <head>
    <style>
      h1 {
        text-shadow: 2px 2px;
      }
      h2 {
        text-shadow: 2px 2px red;
      }
      h3 {
        text-shadow: 2px 2px 5px red;
      }
      h4 {
```

```
        color: white;
        text-shadow: 2px 2px 4px #000000;
      }
      h5 {
        text-shadow: 0 0 3px #FF0000;
      }
      h6 {
        text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;
      }
      p {
        color: white;
        text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;
      }
    </style>
  </head>

  <body>
    <h1>WsCube Tech</h1>
    <h2>WsCube Tech</h2>
    <h3>WsCube Tech</h3>
    <h4>WsCube Tech</h4>
    <h5>WsCube Tech</h5>
    <h6>WsCube Tech</h6>
    <p>WsCube Tech</p>
  </body>
</html>
```

box shadow

Used to add shadow effects to elements, Following is the example to add shadow effects to element.

```
<html>
  <head>
    <style>
      div {
        width: 300px;
        height: 100px;
        padding: 15px;
        background-color: red;
        box-shadow: 10px 10px;
      }
    </style>
  </head>

  <body>
    <div>This is a div element with a box-shadow</div>
  </body>
</html>
```



CSS3 contained several extra features, which is added later on.

1. text-overflow
2. word-wrap
3. word-break

There are following most commonly used property in CSS3 −

text-align-last :- Used to align the last line of the text
text-emphasis :- Used to emphasis text and color
text-overflow :- used to determines how overflowed content that is not displayed is signaled to users
word-break :- Used to break the line based on word
word-wrap :- Used to break the line and wrap onto next line

Text-overflow

The text-overflow property determines how overflowed content that is not displayed is signaled to users.

Example :-

```
<html>
  <head>
    <style>
      p.text1 {
        white-space: nowrap;
        width: 500px;
        border: 1px solid #000000;
        overflow: hidden;
        text-overflow: clip;
      }
p.text2 {
        white-space: nowrap;
        width: 500px;
```

```
        border: 1px solid #000000;
        overflow: hidden;
        text-overflow: ellipsis;
      }

    </style>
  </head>

  <body>
     <b>Original Text:</b>

    <p>
      wscube tech originated from the idea that there exists a class of
      readers who respond better to online content and prefer to learn new
      skills at their own pace from the comforts of their drawing rooms.
    </p>
    <b>Text overflow:clip:</b>
    <p class = "text1">
      wscube tech originated from the idea that there exists
      a class of readers who respond better to online content and prefer
      to learn new skills at their own pace from the comforts of their
      drawing rooms.
    </p>
<b>Text overflow:ellipsis</b>

    <p class = "text2">
      wscube tech originated from the idea that there exists
      a class of readers who respond better to online content and
      prefer to learn new skills at their own pace from the comforts
      of their drawing rooms.
    </p>

        </body>
</html>
```

CSS3 Word Breaking

Used to break the line, following code shows the sample code of word breaking.

```
<html>
  <head>
    <style>
      p.text1 {
        width: 140px;
        border: 1px solid #000000;
        word-break: keep-all;
      }
      p.text2 {
        width: 140px;
        border: 1px solid #000000;
```

```
        word-break: break-all;
      }
    </style>
  </head>

  <body>

    <b>line break at hyphens:</b>
    <p class = "text1">
      WsCube Tech originated from the idea that there exists a
      class of readers who respond better to online content and prefer
      to learn new skills at their own pace from the comforts of
      their drawing rooms.
    </p>

    <b>line break at any character</b>

    <p class = "text2">
      WsCube Tech originated from the idea that there exists a
      class of readers who respond better to online content and
      prefer to learn new skills at their own pace from the comforts
      of their drawing rooms.
    </p>

  </body>
</html>
```

CSS word wrapping
Word wrapping is used to break the line and wrap onto next line.
syntax example :−

```
p {
  word-wrap: break-word;
}
```

Web fonts are used to allows the fonts in CSS, which are not installed on local system.

Different web fonts formats

TrueType Fonts (TTF) :- TrueType is an outline font standard developed by Apple and Microsoft in the late 1980s, It became most common fonts for both windows and MAC operating systems.

OpenType Fonts (OTF) :- OpenType is a format for scalable computer fonts and developed by Microsoft.

The Web Open Font Format (WOFF) :- WOFF is used for develop web page and developed in the year of 2009. Now it is using by W3C recommendation.

SVG Fonts/Shapes :- SVG allow SVG fonts within SVG documentation. We can also apply CSS to SVG with font face property.

Embedded OpenType Fonts (EOT) :- EOT is used to develop the web pages and it has embedded in webpages so no need to allow 3rd party fonts

```
@font-face {
   font-family: myFirstFont;
   src: url(/css/font/SansationLight.woff);
}
 div {
   font-family: myFirstFont;
}
```

Fonts **description**
The following list contained all the fonts description which are placed in the @font-face rule −

Font-family :-	Used to defines the name of font
Src :-	Used to defines the URL
Font-stretch :-	Used to find, how font should be stretched
Font-style :-	Used to defines the fonts style
Font-weight :-	Used to defines the font weight(boldness)

2D transforms are used to re-change the element structure as translate, rotate, scale, and skew.
The following table has contained common values which are used in 2D transforms −

- **matrix(n,n,n,n,n,n) :-** Used to defines matrix transforms with six values
- **translate(x,y) :-** Used to transforms the element along with x-axis and y-axis
- **translateX(n) :-** Used to transforms the element along with x-axis
- **translateY(n) :-** Used to transforms the element along with y-axis
- **scale(x,y) :-** Used to change the width and height of element
- **scaleX(n) :-** Used to change the width of element
- **scaleY(n) :-** Used to change the height of element
- **rotate(angle) :-** Used to rotate the element based on an angle
- **skewX(angle) :-** Used to defines skew transforms along with x axis
- **skewY(angle) :-** Used to defines skew transforms along with y axis

Rotate 20 degrees
Box rotation with 20 degrees angle as shown below

```
div#myDiv {
   /* IE 9 */
   -ms-transform: rotate(20deg);

   /* Safari */
   -webkit-transform: rotate(20deg);

   /* Standard syntax */
```

```
  transform: rotate(20deg);
}
```

Skew X axis

Box rotation with skew x-axis as shown below −

```
div#skewDiv {
  /* IE 9 */
  -ms-transform: skewX(20deg);

  /* Safari */
  -webkit-transform: skewX(20deg);

  /* Standard syntax */
  transform: skewX(20deg);
}
```

Skew Y axis
Box rotation with skew y-axis as shown below −

```
div#skewDiv {
  /* IE 9 */
  -ms-transform: skewY(20deg);

  /* Safari */
  -webkit-transform: skewY(20deg);

  /* Standard syntax */
  transform: skewY(20deg);
}
```

Matrix transform
Box rotation with Matrix transforms as shown below −

```
div#myDiv1 {
  /* IE 9 */
  -ms-transform: matrix(1, -0.3, 0, 1, 0, 0);

  /* Safari */
  -webkit-transform: matrix(1, -0.3, 0, 1, 0, 0);

  /* Standard syntax */
  transform: matrix(1, -0.3, 0, 1, 0, 0);
}
```

Matrix transforms with another direction.

```
div#myDiv2 {
   /* IE 9 */
   -ms-transform: matrix(1, 0, 0.5, 1, 150, 0);

   /* Safari */
   -webkit-transform: matrix(1, 0, 0.5, 1, 150, 0);

   /* Standard syntax */
   transform: matrix(1, 0, 0.5, 1, 150, 0);
}
```

Using with 3d transforms, we can move element to x-axis, y-axis and z-axis, Below example clearly specifies how the element will rotate.

Methods of 3D transforms
Below methods are used to call 3D transforms −

- **matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n) :-** Used to transforms the element by using 16 values of matrix
- **translate3d(x,y,z) :-** Used to transforms the element by using x-axis,y-axis and z-axis
- **translateX(x) :-** Used to transforms the element by using x-axis
- **translateY(y) :-** Used to transforms the element by using y-axis
- **translateZ(z) :-** Used to transforms the element by using y-axis
- **scaleX(x) :-** Used to scale transforms the element by using x-axis
- **scaleY(y) :-** Used to scale transforms the element by using y-axis
- **scaleY(y) :-** Used to transforms the element by using z-axis
- **rotateX(angle) :-** Used to rotate transforms the element by using x-axis
- **rotateY(angle) :-** Used to rotate transforms the element by using y-axis
- **rotateZ(angle) :-** Used to rotate transforms the element by using z-axis

X-axis 3D transforms
The following an example shows the x-axis 3D transforms.

```
-webkit-transform: rotateX(150deg);

/* Safari */
transform: rotateX(150deg);

/* Standard syntax */
}
```

Y-axis 3D transforms

The following an example shows the y-axis 3D transforms −

```
div#yDiv {
   -webkit-transform: rotateY(150deg);
```

```
   /* Safari */
   transform: rotateY(150deg);

   /* Standard syntax */
}
```

Z-axis 3D transforms
The following an example shows the Z-axis 3D transforms −

```
div#zDiv {
   -webkit-transform: rotateZ(90deg);

   /* Safari */
   transform: rotateZ(90deg);

   /* Standard syntax */
}
```

Animation is process of making shape changes and creating motions with elements.

@keyframes
Keyframes will control the intermediate animation steps in CSS3.

```
@keyframes animation {
  from {background-color: pink;}
  to {background-color: green;}
}
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: animation;
  animation-duration: 5s;
}
```

CSS3 supported multi columns to arrange the text as news paper structure.
Some of most common used multi columns properties as shown below −

column-count :- Used to count the number of columns that element should be divided.
column-fill :- Used to decide, how to fill the columns.
column-gap :- Used to decide the gap between the columns.
column-rule :- Used to specifies the number of rules.
rule-color :- Used to specifies the column rule color.
rule-style :- Used to specifies the style rule for column.
rule-width :- Used to specifies the width.
column-span :- Used to specifies the span between columns.

Example :-

```
.multi {
     /* Column count property */
   -webkit-column-count: 4;
   -moz-column-count: 4;
   column-count: 4;

   /* Column gap property */
   -webkit-column-gap: 40px;
   -moz-column-gap: 40px;
   column-gap: 40px;

   /* Column style property */
   -webkit-column-rule-style: solid;
   -moz-column-rule-style: solid;
   column-rule-style: solid;
 }
```

The user interface property allows you to change any element into one of several standard user interface elements.

Some of the common properties which are using in css3 User interface.

Box-sizing :- Allows to users to fix elements on area in clear way.
Resize :- Used to resize elements which are on area.

Example of resize property
Resize property is having three common values as shown below −

1. horizontal
2. vertical
3. both

Using of both value in resize property in css3 user interface −

```
div {
   border: 2px solid;
   padding: 20px;
   width: 300px;
   resize: both;
   overflow: auto;
 }
```

CSS3 Outline offset
Out line means draw a line around the element at outside of border.

```
div {
```

```
    margin: 20px;
    padding: 10px;
    width: 300px;
    height: 100px;
    border: 5px solid pink;
    outline: 5px solid green;
    outline-offset: 15px;
  }
```

Box sizing property is using to change the height and width of element.

Since css2, the box property has worked like as shown below −

width + padding + border = actual visible/rendered width of an element's box

height + padding + border = actual visible/rendered height of an element's box

Means when you set the height and width, it appears little bit bigger then given size cause element boarder and padding added to the element height and width.

CSS3 box sizing property

```
.div1 {
   width: 300px;
   height: 100px;
   border: 1px solid blue;
   box-sizing: border-box;
 }
```

What is Gradients?
Gradients displays the combination of two or more colors as shown below −

Types of gradients

1. Linear Gradients(down/up/left/right/diagonally)
2. Radial Gradients

Linear gradients
Linear gradients are used to arrange two or more colors in linear formats like top to bottom.

Top to bottom

```
<html>
  <head>
    <style>
      #grad1 {
        height: 100px;
        background: -webkit-linear-gradient(pink,green);
        background: -o-linear-gradient(pink,green);
```

```
        background: -moz-linear-gradient(pink,green);
        background: linear-gradient(pink,green);
      }
    </style>
  </head>

  <body>
    <div id = "grad1"></div>
  </body>
</html>
```

Left to right

```
<html>
  <head>
    <style>
      #grad1 {
        height: 100px;
        background: -webkit-linear-gradient(left, red , blue);
        background: -o-linear-gradient(right, red, blue);
        background: -moz-linear-gradient(right, red, blue);
        background: linear-gradient(to right, red , blue);
      }
    </style>
  </head>

  <body>
    <div id = "grad1"></div>
  </body>
</html>
```

Diagonal
Diagonal starts at top left and right button.

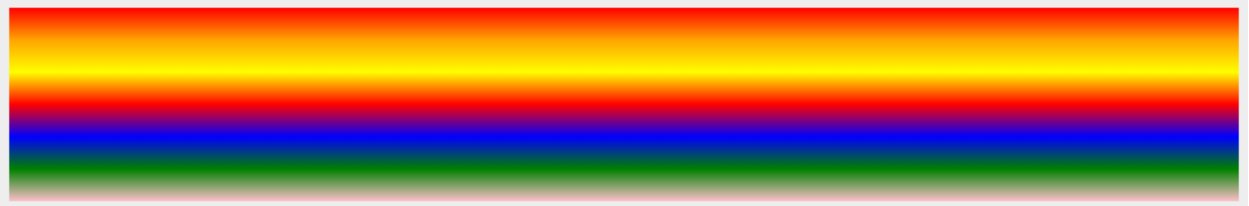
Example :-

```
#grad1 {
  height: 100px;
  background: -webkit-linear-gradient(left top, red , blue);
  background: -o-linear-gradient(bottom right, red, blue);
  background: -moz-linear-gradient(bottom right, red, blue);
  background: linear-gradient(to bottom right, red , blue);
}
```

Multi color

Example :-

```
#grad2 {
  height: 100px;
  background: -webkit-linear-gradient(red, orange, yellow, red, blue, green,pink);
  background: -o-linear-gradient(red, orange, yellow, red, blue, green,pink);
  background: -moz-linear-gradient(red, orange, yellow, red, blue, green,pink);
  background: linear-gradient(red, orange, yellow, red, blue, green,pink);
}
```

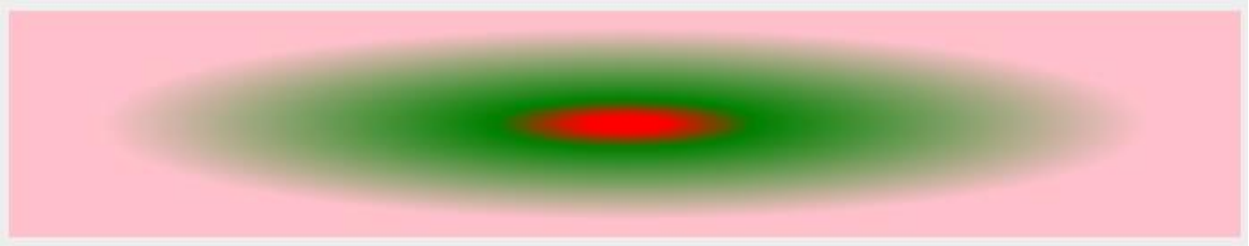


CSS3 Radial Gradients
Radial gradients appears at center.

Example :-

```
#grad1 {
  height: 100px;
  width: 550px;
  background: -webkit-radial-gradient(red 5%, green 15%, pink 60%);
  background: -o-radial-gradient(red 5%, green 15%, pink 60%);
  background: -moz-radial-gradient(red 5%, green 15%, pink 60%);
  background: radial-gradient(red 5%, green 15%, pink 60%);
```

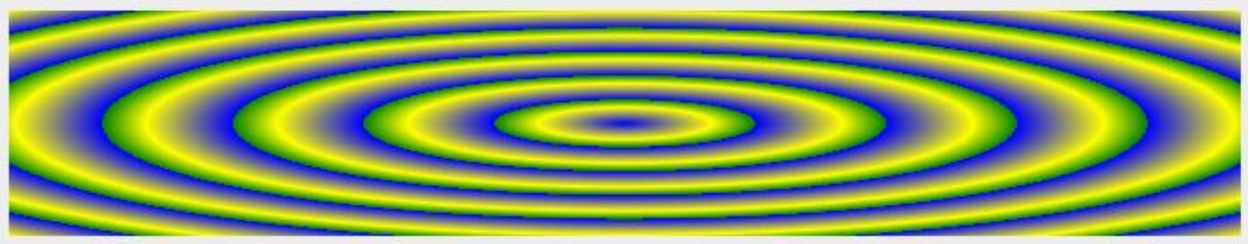
```
}
```



CSS3 Repeat Radial Gradients

Example :-

```
#grad1 {
  height: 100px;
  width: 550px;
  background: -webkit-repeating-radial-gradient(blue, yellow 10%, green 15%);
  background: -o-repeating-radial-gradient(blue, yellow 10%, green 15%);
  background: -moz-repeating-radial-gradient(blue, yellow 10%, green 15%);
  background: repeating-radial-gradient(blue, yellow 10%, green 15%);
}
```



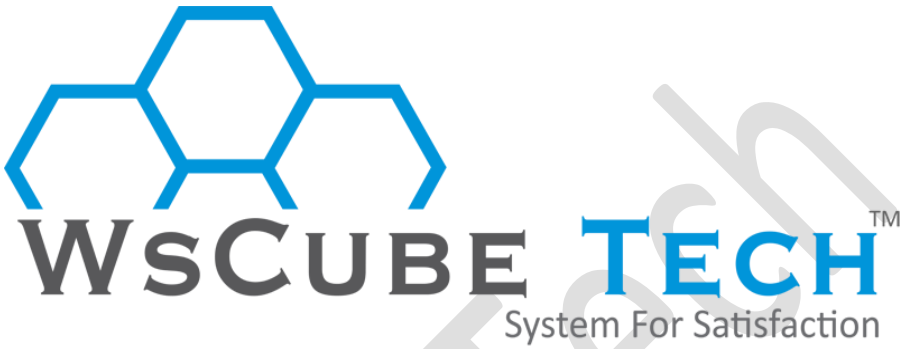
Media queries

Media queries is for different style rules for different size devices such as mobiles, desktops, etc.,

```
body {
  background-color: lightpink;
}
@media screen and (max-width: 420px) {
  body {
    background-color: lightblue;
  }
}
```

WsCube Tech

1st Floor, Laxmi Tower, Bhaskar Circle, Ratanada,
Jodhpur.
Mobile : +91-92696-98122 , 85610-89567



JAVASCRIPT CUBE

Building Interactive Website with JAVASCRIPT

BY **WsCube Tech**

What is JavaScript?
JavaScript and Java are completely different languages, both in concept and design.
JavaScript is an object-based scripting language which is lightweight and cross-platform.

JavaScript is not a compiled language, but it is a translated language. The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.
JavaScript is one of the 3 languages all web developers must learn:

1. HTML to define the content of web pages
2. CSS to specify the layout of web pages
3. JavaScript to program the behavior of web pages

Web pages are not the only place where JavaScript is used. Many desktop and server programs use JavaScript. Node.js is the best known. Some databases, like MongoDB and CouchDB, also use JavaScript as their programming language.

Application of JavaScript

JavaScript is used to create interactive websites. It is mainly used for:

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

Example :-

```
<script>
document.write("Hello JavaScript");
</script>
```

Warning for Non-JavaScript Browsers

If you have to do something important using JavaScript, then you can display a warning message to the user using
<noscript> tags.

Syntax

```
<noscript>
        Sorry...JavaScript is needed to go ahead.
</noscript>
```

3 Places to put JavaScript code :-

1. Between the body tag of html
2. Between the head tag of html
3. In .js file (external javaScript)

The script tag specifies that we are using JavaScript.
The text/javascript is the content type that provides information to the browser about the data.
The document.write() function is used to display dynamic content through JavaScript. We will learn about document object in detail later.

Alert :-

Displayed the dynamic content using JavaScript. Let's see the simple example of JavaScript that displays alert dialog box.

Example :-

```
<script type="text/javascript">
 alert("Hello Javatpoint");
</script>
```

Let's see the same example of displaying alert dialog box of JavaScript that is contained inside the head tag.

Example :-

```
<html>
        <head>
        <script type="text/javascript">
                        alert("Hello Javatpoint");
        </script>
        </head>
        <body>
        </body>
</html>
```

External JavaScript file

We can create external JavaScript file and embed it in many html page.
It provides code re usability because single JavaScript file can be used in several html pages.
An external JavaScript file must be saved by .js extension. It is recommended to embed all JavaScript files into a single file. It increases the speed of the webpage.

Example :-

Let's create an external JavaScript file that prints Hello Javatpoint in a alert dialog box.

```
//message.js

function msg(){
 alert("Hello Javatpoint");
}
```

Let's include the JavaScript file into html page. It calls the JavaScript function on button click.

```
//index.html

<html>
<head>
        <script type="text/javascript" src="message.js"></script>
</head>
<body>
        <p>Welcome to JavaScript</p>
        <form>
<input type="button" value="click" onclick="msg()"/>
```

```
        </form>
</body>
</html>
```

JavaScript Variable

A JavaScript variable is simply a name of storage location. There are two types of variables in JavaScript :

1. local variable
2. global variable

There are some rules while declaring a JavaScript variable (also known as identifiers).
Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.
After first letter we can use digits (0 to 9), for example value1.

JavaScript variables are case sensitive, for example x and X are different variables.

```
//Correct JavaScript variables
var x = 10;
var _value="sonoo";

//Incorrect JavaScript variables
var  123=30;
var *aa=320;
```

Example :-

```
<script>
  var x = 10;
  var y = 20;
  var z=x+y;
  document.write(z);
</script>
```

JavaScript local variable :-

A JavaScript local variable is declared inside block or function. It is accessible within the function or block only.

Example :-

```
<script>
function abc(){
var x=10; //local variable
}
</script>

Or,
```

```
<script>
If(10<13){
var y=20; //JavaScript local variable
}
</script>
```

JavaScript global variable :-

A JavaScript global variable is declared outside the function or declared with window object. It can be accessed from any function.
Let's see the simple example of global variable in JavaScript.

Example :-

```
<script>

var data=200; //gloabal variable

function a(){
document.writeln(data);
}

function b(){
document.writeln(data);
}
a(); //calling JavaScript function
b();
</script>
```

Declaring JavaScript global variable within function
To declare JavaScript global variables inside function, you need to use window object.

Example :-

```
window.value=90;
```

Now it can be declared inside any function and can be accessed from any function.

Example :-

```
function m(){
window.value=100;//declaring global variable by window object
}

function n(){
alert(window.value);//accessing global variable from other function
}
```

javascript concatenate (+)

concatenation, or **concat** is a term that describes combining a string, text, or other data in a series without any gaps. In programming languages, an operator is **used** to denote **concatenation**

var a=”hello”
var b=”Wscubetech”
var c=a+b
document.write(c)
hello Wscubetech

JavaScript String concat() Method

The concat() method is used to join two or more strings.
This method does not change the existing strings, but returns a new string containing the text of the joined strings.

var c=a.concat(b)
document.write(c)
hello Wscubetech

Javascript Data Types :-

JavaScript provides different data types to hold different types of values. There are two types of data types in JavaScript.
Primitive data type
Non-primitive (reference) data type
JavaScript is a dynamic type language, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use var here to specify the data type. It can hold any type of values such as numbers, strings etc.

Example :-

```
var a=40;//holding number
var b="Rahul";//holding string
```

JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

Data Type	Description
String	represents sequence of characters e.g. "hello"

Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

JavaScript non-primitive data types

The non-primitive data types are as follows:

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

JavaScript Operators :-

JavaScript operators are symbols that are used to perform operations on operands. For example:
var sum=10+20;
Here, + is the arithmetic operator and = is the assignment operator.
There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

JavaScript Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

Operator	Description	Example
==	Is equal to	10==20 = false

===	Identical (equal and of same type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

JavaScript Bitwise Operators

The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows:

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10

<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2

JavaScript Logical Operators

The following operators are known as JavaScript logical operators.

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

JavaScript Assignment Operators

The following operators are known as JavaScript assignment operators.

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30

-=	Subtract and assign	var a=20; a-=10; Now a = 10
*=	Multiply and assign	var a=10; a*=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0

JavaScript Special Operators

The following operators are known as JavaScript special operators.

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
Delete	Delete Operator deletes a property from the object.
In	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
New	creates an instance (object)

Typeof	checks the type of object.
Void	it discards the expression's return value.
Yield	checks what is returned in a generator by the generator's iterator.

JavaScript If-else :-

The JavaScript if-else statement is used to execute the code whether condition is true or false. There are three forms of if statement in JavaScript.

1. If Statement
2. If else statement
3. If else if statement

If Statement

```
if (condition1) {
  //  block of code to be executed if condition1 is true
}
```

Example :-

```
<script>
var a=20;
if(a>10){
document.write("value of a is greater than 10");
}
</script>
```

If else Statement

```
if (condition1) {
  //  block of code to be executed if condition1 is true
}

 else {
  //  block of code to be executed if the condition1 is false and condition2 is false
}
```

Example :-

```
<script>
var a=20;
if(a%2==0){
document.write("a is even number");
}
else{
document.write("a is odd number");
}
</script>
```

If else if Statement

```
if (condition1) {
  //  block of code to be executed if condition1 is true
}
 else if (condition2) {
  //  block of code to be executed if the condition1 is false and condition2 is true
}
 else {
  //  block of code to be executed if the condition1 is false and condition2 is false
}
```

Example :-

```
<script>
Var marks = 75;
if (marks > 90)
{
 greeting = "A Grade";
}
else if (marks > 80)
{
  greeting = "B Grade";
}
else
{
  greeting = "C Grade";
}
<script>
```

JavaScript Switch :-

The JavaScript switch statement is used to execute one code from multiple expressions. It is just like else if statement that we have learned in previous page. But it is convenient than if..else..if because it can be used with numbers, characters etc.

Syntax :-

```
switch (expression)
{
  case condition 1: statement(s)
  break;
  case condition 2: statement(s)
  break;
  case condition n: statement(s)
  break;
  default: statement(s)
}
```

Example :-

```
<html>
  <body>
        <script type = "text/javascript">
        <!--
        var grade = 'A';
        document.write("Entering switch block<br />");
        switch (grade) {
        case 'A': document.write("Good job<br />");
        break;

        case 'B': document.write("Pretty good<br />");
        break;

        case 'C': document.write("Passed<br />");
        break;

        case 'D': document.write("Not so good<br />");
        break;

        case 'F': document.write("Failed<br />");
        break;

        default:  document.write("Unknown grade<br />")
        }
        document.write("Exiting switch block");
        //-->
         </script>
        <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

JavaScript Loops :-

The JavaScript loops are used to iterate the piece of code using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.
There are four types of loops in JavaScript.

1. for loop
2. while loop
3. do-while loop
4. for-in loop

JavaScript For loop :-

The JavaScript for loop iterates the elements for the fixed number of times. It should be used if number of iteration is known.

Syntax :-

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

Example :-

```
var i;
var text="";
for (i = 0; i <= 10; i++) {
  text= text + i + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

JavaScript Nested For loop :-

Syntax :-

```
for (statement 1; statement 2; statement 3) {
 for (statement 1; statement 2; statement 3) {
   // code block to be executed
 }
}
```

Example :-

```
var i;
Var j;
for (i = 0; i <= 10; i++) {
 for (i = 0; i <= 10; i++) {
  text += "*<br>";
 }
}
```

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

JavaScript While loop :-

The JavaScript while loop iterates the elements for the infinite number of times. It should be used if number of iteration is not known.

Syntax :-

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

Example :-

```
var i;
var text="";
while (i<=10)
{
  text += 2+"*"+i+"="+2*i+ "<br>";
  i++;
}
document.getElementById("demo").innerHTML = text;
```

JavaScript do while loop :-

The JavaScript do while loop iterates the elements for the infinite number of times like while loop. But, code is executed at least once whether condition is true or false.

Syntax :-

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

Example :-

```
do
{
  text += 2+"*"+i+"="+2*i+ "<br>";
  i++;
}
while (i<=10);

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

JavaScript Functions :-

JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.
Advantage of JavaScript function
There are mainly two advantages of JavaScript functions.

Code reusability: We can call a function several times so it save coding.
Less coding:- It makes our program compact. We don't need to write many lines of code each time to perform a common task.

JavaScript Function Syntax :-

```
function functionName([arg1, arg2, ...argN]){
 //code to be executed
}
```

Example :-

```
<script>
function msg(){
alert("hello! this is message");
}
</script>
<input type="button" onclick="msg()" value="call function"/>
```

JavaScript Function Arguments :-

We can call function by passing arguments.

Example :-

```
<script>
function getcube(number){
alert(number*number*number);
}
</script>
<form>
<input type="button" value="click" onclick="getcube(4)"/>
</form>
```

Function with Return Value :-

We can call function that returns a value and use it in our program.

Example :-

```
<script>
function getInfo(){
return "hello javatpoint! How r u?";
```

```
}
</script>
<script>
document.write(getInfo());
</script>
```

JavaScript Events :-

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.
When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.
Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.

Onclick - The user clicks an HTML element
Onmouseover - The user moves the mouse over an HTML element
Onmouseout - The user moves the mouse away from an HTML element
Onkeydown - The user pushes a keyboard key
Onload - The browser has finished loading the page
Onblur-Triggers when the window loses focus
Ondblclick- Triggers on a mouse double-click
Onchange- Triggers when an element changes
Onfocus-Triggers when the window gets focus
Onkeypress- Triggers when a key is pressed and released
Onkeyup- Triggers when a key is released
Onscroll- Triggers when an element's scrollbar is being scrolled
Onselect- Triggers when an element is selected
Onsubmit- Triggers when a form is submitted

Onclick function :-

script run on some event, such as when a user clicks somewhere.

Example :-

```
<html>
  <head>
        <script type = "text/javascript">
        <!--
        function sayHello() {
        alert("Hello World")
        }
        //-->
        </script>
  </head>
  <body>
        <input type = "button" onclick = "sayHello()" value = "Say Hello" />
```

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

Six other escape sequences are valid in JavaScript :-

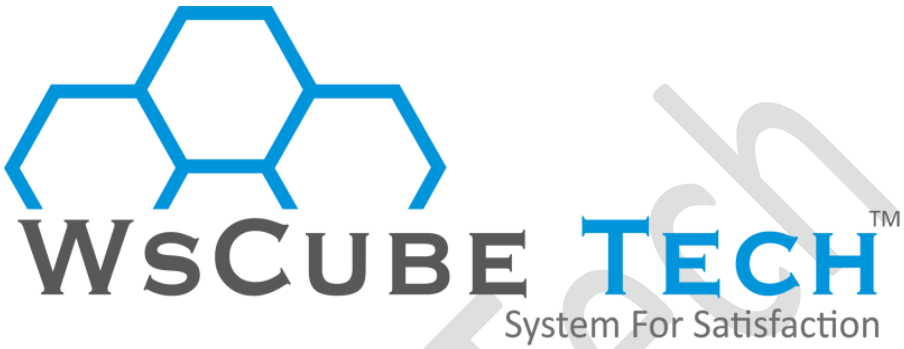
\b - Backspace
\f - Form Feed
\n - New Line
\r - Carriage Return
\t - Horizontal Tabulator
\v - Vertical Tabulator

How to print selected div instead complete page

```
<script type="text/javascript">
        function printDiv() {
        var printContents = document.getElementById('Your printable div').innerHTML;
        var originalContents = document.getElementById('body').innerHTML;
        document.getElementById('body').innerHTML = printContents;
        window.print();
        document.body.innerHTML = originalContents;
        }
</script>
```

WsCube Tech

1st Floor, Laxmi Tower, Bhaskar Circle, Ratanada,
Jodhpur.
Mobile : +91-92696-98122 , 85610-89567



JQUERY CUBE

Building Interactive Websites with JQUERY

BY **WsCube Tech**

What is Jquery ?

jQuery is a fast and concise JavaScript library created by John Resig in 2006. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

Here is the list of important core features supported by jQuery

1. **DOM manipulation :−** The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called Sizzle.
2. **Event handling :−** The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
3. **AJAX Support :−** The jQuery helps you a lot to develop a responsive and featurerich site using AJAX technology.
4. **Animations :−** The jQuery comes with plenty of built-in animation effects which you can use in your websites.
5. **Lightweight :−** The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).
6. **Cross Browser Support :−** The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
7. **Latest Technology :−** The jQuery supports CSS3 selectors and basic XPath syntax.

There are two ways to use jQuery.

1. **Local Installation −** You can download jQuery library on your local machine and include it in your HTML code.
2. **CDN Based Version −** You can include jQuery library into your HTML code directly from Content Delivery Network.

Example :-

```
<html>
  <head>
    <title>The jQuery Example</title>
    <script type = "text/javascript" src = "/jquery/jquery-2.1.3.min.js"></script>

        OR

     <script src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

    <script type = "text/javascript">
      $(document).ready(function() {
        document.write("Hello, World!");
      });
    </script>
  </head>

  <body>
    <h1>Hello</h1>
  </body>
</html>
```

How to Call a jQuery Library Functions?

As almost everything, we do when using jQuery reads or manipulates the document object model (DOM), we need to make sure that we start adding events etc. as soon as the DOM is ready.
If you want an event to work on your page, you should call it inside the $(document).ready() function. Everything inside it will load as soon as the DOM is loaded and before the page contents are loaded.
To do this, we register a ready event for the document as follows −

```
$(document).ready(function() {
   // do stuff when DOM is ready
});
```

Using Multiple Libraries

You can use multiple libraries all together without conflicting each others. For example, you can use jQuery and MooTool javascript libraries together.

Objects

JavaScript supports Object concept very well. You can create an object using the object literal as follows −

```
var emp = {
   name: "Zara",
   age: 10
};
```

Arrays

You can define arrays using the array literal as follows −

```
var x = [];
var y = [1, 2, 3, 4, 5];
```

Functions

A function in JavaScript can be either named or anonymous. A named function can be defined using function keyword as follows −

```
function named(){
   // do some stuff here
}
```

Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variable will have only two scopes.

1. **Global Variables −** A global variable has global scope which means it is defined everywhere in your JavaScript code.

2. **Local Variables −** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Example :-

```
var myVar = "global";     // ==> Declare a global variable

function ( ) {
   var myVar = "local";   // ==> Declare a local variable
   document.write(myVar); // ==> local
}
```

Built-in Functions

JavaScript comes along with a useful set of built-in functions. These methods can be used to manipulate Strings, Numbers and Dates.
Following are important JavaScript functions −

1. **charAt() :-** Returns the character at the specified index.
2. **concat()** :- Combines the text of two strings and returns a new string.
3. **forEach() :-** Calls a function for each element in the array.
4. **indexOf() :-** Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.
5. **length()** :- Returns the length of the string.
6. **pop()** :- Removes the last element from an array and returns that element.
7. **push() :-** Adds one or more elements to the end of an array and returns the new length of the array.
8. **reverse() :-** Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
9. **sort() :-** Sorts the elements of an array.
10. **substr() :-** Returns the characters in a string beginning at the specified location through the specified number of characters.
11. **toLowerCase() :-** Returns the calling string value converted to lower case.
12. **toString() :-** Returns the string representation of the number's value.
13. **toUpperCase() :-** Returns the calling string value converted to uppercase.

The $() factory function

jQuery selectors start with the dollar sign and parentheses − $(). The factory function $() makes use of following three building blocks while selecting elements in a given document −

- **Tag Name :-** Represents a tag name available in the DOM. For example $('p') selects all paragraphs <p> in the document.

- **Tag ID :-** Represents a tag available with the given ID in the DOM. For example $('#some-id') selects the single element in the document that has an ID of some-id.
- **Tag Class :-** Represents a tag available with the given class in the DOM. For example $('.some-class') selects all elements in the document that have a class of some-class.

All the above items can be used either on their own or in combination with other selectors. All the jQuery selectors are based on the same principle except some tweaking.

NOTE :− The factory function $() is a synonym of jQuery() function. So in case you are using any other JavaScript library where $ sign is conflicting with some thing else then you can replace $ sign by jQuery name and you can use function jQuery() instead of $()

Example :-

Following is a simple example which makes use of Tag Selector. This would select all the elements with a tag name p.

```
<html>
  <head>
    <title>The jQuery Example</title>
    <script type = "text/javascript"
       src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>

    <script type = "text/javascript" language = "javascript">
       $(document).ready(function() {
          $("p").css("background-color", "yellow");
       });
    </script>
  </head>

  <body>
    <div>
       <p class = "myclass">This is a paragraph.</p>
       <p id = "myid">This is second paragraph.</p>
       <p>This is third paragraph.</p>
    </div>
  </body>
</html>
```

How to Use Selectors?

The selectors are very useful and would be required at every step while using jQuery. They get the exact element that you want from your HTML document.

Following table lists down few basic selectors and explains them with examples.

- **Name :-** Selects all elements which match with the given element Name.

- **#ID :-** Selects a single element which matches with the given ID.
- **.Class :-** Selects all elements which match with the given Class.
- **Universal (*) :-** Selects all elements available in a DOM.
- **Multiple Elements x, y, z :-** Selects the combined results of all the specified selectors x, y or z.

Selectors Examples

Similar to above syntax and examples, following examples would give you understanding on using different type of other useful selectors −

- $('*') :- This selector selects all elements in the document.
- $("p > *") :- This selector selects all elements that are children of a paragraph element.
- $("#specialID") :- This selector function gets the element with id="specialID".
- $(".specialClass") :- This selector gets all the elements that have the class of specialClass.
- $("li:not(.myclass)") :- Selects all elements matched by <li> that do not have class = "myclass".
- $("a#specialID.specialClass") :- This selector matches links with an id of specialID and a class of specialClass.
- $("p a.specialClass") :- This selector matches links with a class of specialClass declared within <p> elements.
- $("ul li:first") :- This selector gets only the first <li> element of the <ul>.
- $("#container p") :- Selects all elements matched by <p> that are descendants of an element that has an id of container.
- $("li > ul") :- Selects all elements matched by <ul> that are children of an element matched by
- $("strong + em") :- Selects all elements matched by that immediately follow a sibling element matched by .
- $("p ~ ul") :- Selects all elements matched by <ul> that follow a sibling element matched by <p>.
- $("code, em, strong") :- Selects all elements matched by <code> or or .
- $("p strong, .myclass") :- Selects all elements matched by that are descendants of an element matched by <p> as well as all elements that have a class of myclass.
- $(":empty") :- Selects all elements that have no children.
- $("p:empty") :- Selects all elements matched by <p> that have no children.

Event

We have the ability to create dynamic web pages by using events. Events are actions that can be detected by your Web Application.

Following are the examples events :−

- A mouse click
- A web page loading

- Taking mouse over an element
- Submitting an HTML form
- A keystroke on your keyboard, etc.
- When these events are triggered, you can then use a custom function to do pretty much whatever you want with the event. These custom functions call Event Handlers.

Binding Event Handlers

Using the jQuery Event Model, we can establish event handlers on DOM elements with the bind() method as follows −

```
<html>
  <head>
    <title>The jQuery Example</title>
    <script type = "text/javascript"
       src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>

    <script type = "text/javascript" language = "javascript">
       $(document).ready(function() {
          $('div').bind('click', function( event ){
             alert('Hi there!');
          });
       });
    </script>

    <style>
       .div{ margin:10px;padding:12px; border:2px solid #666; width:60px;}
    </style>
  </head>

  <body>
    <p>Click on any square below to see the result:</p>

    <div class = "div" style = "background-color:blue;">ONE</div>
    <div class = "div" style = "background-color:green;">TWO</div>
    <div class = "div" style = "background-color:red;">THREE</div>
  </body>
</html>
```

Syntax :−

selector.bind(eventType[, eventData], handler)

Following is the description of the parameters −

eventType − A string containing a JavaScript event type, such as click or submit. Refer to the next section for a complete list of event types.

eventData − This is optional parameter is a map of data that will be passed to the event handler.
handler − A function to execute each time the event is triggered.

Removing Event Handlers

Typically, once an event handler is established, it remains in effect for the remainder of the life of the page. There may be a need when you would like to remove event handler.

jQuery provides the unbind() command to remove an exiting event handler.

The syntax of unbind() :−

selector.unbind(eventType, handler)

or

selector.unbind(eventType)

Following is the description of the parameters −

eventType − A string containing a JavaScript event type, such as click or submit. Refer to the next section for a complete list of event types.
handler − If provided, identifies the specific listener that's to be removed.

- **blur :-** Occurs when the element loses focus.
- **change :-** Occurs when the element changes.
- **click :-** Occurs when a mouse click.
- **dblclick :-** Occurs when a mouse double-click.
- **error :-** Occurs when there is an error in loading or unloading etc.
- **focus :-** Occurs when the element gets focus.
- **keydown :-** Occurs when key is pressed.
- **keypress :-** Occurs when key is pressed and released.
- **keyup :-** Occurs when key is released.
- **load :-** Occurs when document is loaded.
- **mousedown :-** Occurs when mouse button is pressed.
- **mouseenter :-** Occurs when mouse enters in an element region.
- **mouseleave :-** Occurs when mouse leaves an element region.
- **mousemove :-** Occurs when mouse pointer moves.
- **mouseout :-** Occurs when mouse pointer moves out of an element.
- **mouseover :-** Occurs when mouse pointer moves over an element.
- **mouseup :-** Occurs when mouse button is released.
- **resize :-** Occurs when window is resized.
- **scroll :-** Occurs when window is scrolled.
- **select :-** Occurs when a text is selected.
- **submit :-** Occurs when form is submitted.
- **unload :-** Occurs when documents is unloaded.

The Event Object

The callback function takes a single parameter; when the handler is called the JavaScript event object will be passed through it.

The event object is often unnecessary and the parameter is omitted, as sufficient context is usually available when the handler is bound to know exactly what needs to be done when the handler is triggered, however there are certain attributes which you would need to be accessed.

The Event Attributes

- **altKey :-** Set to true if the Alt key was pressed when the event was triggered, false if not. The Alt key is labeled Option on most Mac keyboards.
- **ctrlKey :-** Set to true if the Ctrl key was pressed when the event was triggered, false if not.
- **data :-** The value, if any, passed as the second parameter to the bind() command when the handler was established.
- **keyCode :-** For keyup and keydown events, this returns the key that was pressed.
- **metaKey :-** Set to true if the Meta key was pressed when the event was triggered, false if not. The Meta key is the Ctrl key on PCs and the Command key on Macs.
- **pageX :-** For mouse events, specifies the horizontal coordinate of the event relative from the page origin.
- **pageY :-** For mouse events, specifies the vertical coordinate of the event relative from the page origin.
- **relatedTarget :-** For some mouse events, identifies the element that the cursor left or entered when the event was triggered.
- **screenX :-** For mouse events, specifies the horizontal coordinate of the event relative from the screen origin.
- **screenY :-** For mouse events, specifies the vertical coordinate of the event relative from the screen origin.
- **shiftKey :-** Set to true if the Shift key was pressed when the event was triggered, false if not.
- **target :-** Identifies the element for which the event was triggered.
- **timeStamp :-** The timestamp (in milliseconds) when the event was created.
- **type :-** For all events, specifies the type of event that was triggered (for example, click).
- **which :-** For keyboard events, specifies the numeric code for the key that caused the event, and for mouse events, specifies which button was pressed (1 for left, 2 for middle, 3 for right).

```
<html>
  <head>
    <title>The jQuery Example</title>
    <script type = "text/javascript"
      src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>

    <script type = "text/javascript" language = "javascript">
      $(document).ready(function() {
        $('div').bind('click', function( event ){
          alert('Event type is ' + event.type);
          alert('pageX : ' + event.pageX);
          alert('pageY : ' + event.pageY);
          alert('Target : ' + event.target.innerHTML);
        });
      });
    </script>
```

```
    <style>
      .div{ margin:10px;padding:12px; border:2px solid #666; width:60px;}
    </style>
  </head>

  <body>
    <p>Click on any square below to see the result:</p>

    <div class = "div" style = "background-color:blue;">ONE</div>
    <div class = "div" style = "background-color:green;">TWO</div>
    <div class = "div" style = "background-color:red;">THREE</div>
  </body>
</html>
```

The Event Methods

There is a list of methods which can be called on an Event Object −

preventDefault() :- Prevents the browser from executing the default action.

isDefaultPrevented() :- Returns whether event.preventDefault() was ever called on this event object.

stopPropagation() :- Stops the bubbling of an event to parent elements, preventing any parent handlers from being notified of the event.

isPropagationStopped() :- Returns whether event.stopPropagation() was ever called on this event object.

stopImmediatePropagation() :- Stops the rest of the handlers from being executed.

isImmediatePropagationStopped() :- Returns whether event.stopImmediatePropagation() was ever called on this event object.

Event Manipulation Methods

Following table lists down important event-related methods −

bind(type, [data], fn) :- Binds a handler to one or more events (like click) for each matched element. Can also bind custom events.

off(events [, selector] [, handler(eventObject)]) :- This does the opposite of live, it removes a bound live event.

hover(over, out) :- Simulates hovering for example moving the mouse on, and off, an object.

on(events [, selector] [, data], handler) :- Binds a handler to an event (like click) for all current − and future − matched element. Can also bind custom events.

one(type, [data], fn) :- Binds a handler to one or more events to be executed once for each matched element.

ready(fn) :- Binds a function to be executed whenever the DOM is ready to be traversed and manipulated.

trigger(event, [data]) :- Trigger an event on every matched element.

triggerHandler(event, [data]) :- Triggers all bound event handlers on an element.

unbind([type], [fn]) :- This does the opposite of bind, it removes bound events from each of the matched elements.

Event Helper Methods

jQuery also provides a set of event helper functions which can be used either to trigger an event to bind any event types mentioned above.

Trigger Methods

Following is an example which would triggers the blur event on all paragraphs −

```
$("p").blur();
```

Binding Methods

Following is an example which would bind a click event on all the −

```
$("div").click( function () {
   // do something here
});
```

List of all the Event in jQuery

blur( ) :- Triggers the blur event of each matched element.

Example :-

```
$("input").blur(function(){
    $(this).css("background-color", "green");
});
```

The function is executed when the form field loses focus:

blur(fn) :- Bind a function to the blur event of each matched element.
click( ) :- Triggers the click event of each matched element.

Example :-

```
$("p").click(function(){
  $(this).hide();
});
```

The following example says: When a click event fires on a <p> element; hide the current <p> element.

click(fn) :- Binds a function to the click event of each matched element.
change( ) :- Triggers the change event of each matched element.

Example :-

```
$("input").change(function(){
  alert("The text has been changed.");
});
```

Alert a text when an <input> field is changed.

change(fn) :- Binds a function to the change event of each matched element.
dblclick( ) :- Triggers the dblclick event of each matched element.

Example :-

```
$("input").change(function(){
  alert("The text has been changed.");
});
```

Alert a text when an <input> field is changed.

dblclick(fn) :- Binds a function to the dblclick event of each matched element.
error( ) :- Triggers the error event of each matched element.

Example :-

```
$("img").error(function(){
  $("img").replaceWith("<p>Error loading image!</p>");
});
```

If the image element encounters an error, replace it with a text.

Note:- The error() method was deprecated in jQuery version 1.8, and removed in version 3.0.

error(fn) :- Binds a function to the error event of each matched element.
focus( ) :- Triggers the focus event of each matched element.

Example :-

```
$("input").focus(function()
{
  $("span").css("display", "inline").fadeOut(2000);
});
```

Attach a function to the focus event. The focus event occurs when the <input> field gets focus.

- **focus(fn) :-** Binds a function to the focus event of each matched element.
- **keydown( ) :-** Triggers the keydown event of each matched element.
- **keydown(fn) :-** Bind a function to the keydown event of each matched element.
- **keypress( ) :-** Triggers the keypress event of each matched element.
- **keypress(fn) :-** Binds a function to the keypress event of each matched element.
- **keyup( ) :-** Triggers the keyup event of each matched element.
- **keyup(fn) :-** Bind a function to the keyup event of each matched element.
- **load(fn) :-** Binds a function to the load event of each matched element.

- **mousedown(fn) :-** Binds a function to the mousedown event of each matched element.
- **mouseenter(fn) :-** Bind a function to the mouseenter event of each matched element.
- **mouseleave(fn) :-** Bind a function to the mouseleave event of each matched element.
- **mousemove(fn) :-** Bind a function to the mousemove event of each matched element.
- **mouseout(fn) :-** Bind a function to the mouseout event of each matched element.
- **mouseover(fn) :-** Bind a function to the mouseover event of each matched element.
- **mouseup(fn) :-** Bind a function to the mouseup event of each matched element.
- **resize(fn) :-** Bind a function to the resize event of each matched element.
- **scroll(fn) :-** Bind a function to the scroll event of each matched element.
- **select( ) :-** Trigger the select event of each matched element.
- **select(fn) :-** Bind a function to the select event of each matched element.
- **submit( ) :-** Trigger the submit event of each matched element.
- **submit(fn) :-** Bind a function to the submit event of each matched element.
- **unload(fn) :-** Binds a function to the unload event of each matched element.

It is used for selecting HTML elements and performing some action on the element(s).

Hide, Show, Toggle, Slide, Fade, and Animate.

```
$(document).ready(function(){
   $("#hide").click(function(){
        $("p").hide(1000);
    });
    $("#show").click(function(){
       $("p").show(2000);
   });
});
```

1. fadeIn()
2. fadeOut()
3. fadeToggle()
4. fadeTo()

Methods Description

fadeIn() :- The jQuery fadeIn() method is used to fade in a hidden element. $(selector).fadeIn(speed,callback);

fadeOut() :- The jQuery fadeOut() method is used to fade out a visible element. $(selector).fadeOut(speed,callback);

fadeToggle() :- The jQuery fadeToggle() method toggles between the fadeIn() and fadeOut() methods.

$(selector).fadeToggle(speed,callback);

fadeTo() The jQuery fadeTo() method allows fading to a given opacity (value between 0 and 1)

$(selector).fadeTo(speed,opacity,callback);

```
$("button").click(function(){
    $("#div1").fadeIn();
    $("#div2").fadeIn("slow");
    $("#div3").fadeIn(3000);
})

$(document).ready(function(){
   $("button").click(function(){
   $("#div1").fadeOut();
   $("#div2").fadeOut("slow");
   $("#div3").fadeOut(3000);
 });
});

$(document).ready(function(){
      $("button").click(function(){
          $("#div1").fadeToggle();
          $("#div2").fadeToggle("slow");
          $("#div3").fadeToggle(3000);
       });
});

$(document).ready(function(){
     $("button").click(function(){
         $("#div1").fadeTo("slow",0.15);
         $("#div2").fadeTo("slow",0.4);
         $("#div3").fadeTo("slow",0.7);
      });
});
```

1. slideDown()
2. slideUp()
3. slideToggle()

Methods Description

slideDown()
The jQuery slideDown() method is used to slide down an element.

$(selector).slideDown(speed,callback);

slideUp()

The jQuery slideUp() method is used to slide up an element.

$(selector).slideUp(speed,callback);

slideToggle()
The jQuery slideToggle() method toggles between the slideDown() and slideUp() methods.

$(selector).slideToggle(speed,callback);

Get Attribute Value

The attr() method can be used to either fetch the value of an attribute from the first element in the matched set or set attribute values onto all matched elements.

Example :-

Following is a simple example which fetches title attribute of tag and set value with the same value −

```
<html>
  <head>
    <title>The jQuery Example</title>
    <script type = "text/javascript"
      src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>

    <script type = "text/javascript" language = "javascript">
      $(document).ready(function() {
        var title = $("em").attr("title");
        $("#divid").text(title);
      });
    </script>
  </head>

  <body>
    <div>
      <em title = "Bold and Brave">This is first paragraph.</em>
      <p id = "myid">This is second paragraph.</p>
      <div id = "divid"></div>
    </div>
  </body>
</html>
```

Set Attribute Value

The attr(name, value) method can be used to set the named attribute onto all elements in the wrapped set using the passed value.

Example :-

Following is a simple example which set src attribute of an image tag to a correct location −

```
<html>
  <head>
    <title>The jQuery Example</title>
    <base href="https://www.tutorialspoint.com" />
    <script type = "text/javascript"
      src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>

    <script type = "text/javascript" language = "javascript">
      $(document).ready(function() {
        $("#myimg").attr("src", "/jquery/images/jquery.jpg");
      });
    </script>
  </head>

  <body>
    <div>
      <img id = "myimg" src = "/images/jquery.jpg" alt = "Sample image" />
    </div>
  </body>
</html>
```

Applying Styles

The addClass(classes) method can be used to apply defined style sheets onto all the matched elements. You can specify multiple classes separated by space.

Example :-
Following is a simple example which sets class attribute of a para <p> tag −

```
<html>
  <head>
    <title>The jQuery Example</title>
    <script type = "text/javascript"
      src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>

    <script type = "text/javascript" language = "javascript">
      $(document).ready(function() {
        $("em").addClass("selected");
        $("#myid").addClass("highlight");
      });
    </script>

    <style>
```

```
      .selected { color:red; }
      .highlight { background:yellow; }
    </style>
  </head>

  <body>
    <em title = "Bold and Brave">This is first paragraph.</em>
    <p id = "myid">This is second paragraph.</p>
  </body>
</html>
```

Attribute Methods

Following table lists down few useful methods which you can use to manipulate attributes and properties −

- **attr(properties) :-** Set a key/value object as properties to all matched elements.
- **attr(key, fn) :-** Set a single property to a computed value, on all matched elements.
- **removeAttr(name) :-** Remove an attribute from each of the matched elements.
- **hasClass(class) :-** Returns true if the specified class is present on at least one of the set of matched elements.
- **removeClass(class) :-** Removes all or the specified class(es) from the set of matched elements.
- **toggleClass(class) :-** Adds the specified class if it is not present, removes the specified class if it is present.
- **html( ) :-** Get the html contents (innerHTML) of the first matched element.
- **html(val) :-** Set the html contents of every matched element.
- **text( ) :-** Get the combined text contents of all matched elements.
- **text(val) :-** Set the text contents of all matched elements.
- **val( ) :-** Get the input value of the first matched element.
- **val(val) :-** Set the value attribute of every matched element if it is called on <input> but if it is called on <select> with the passed <option> value then passed option would be selected, if it is called on check box or radio box then all the matching check box and radiobox would be checked.

Examples :-

```
$("#myID").attr("custom")
```
This would return value of attribute custom for the first element matching with ID myID.

```
$("img").attr("alt", "Sample Image")
```
This sets the alt attribute of all the images to a new value "Sample Image".

```
$("input").attr({ value: "", title: "Please enter a value" });
```
Sets the value of all <input> elements to the empty string, as well as sets The jQuery Example to the string Please enter a value.

```
$("a[href^=https://]").attr("target","_blank")
```
Selects all links with an href attribute starting with https:// and set its target attribute to _blank.

```
$("a").removeAttr("target")
```
This would remove target attribute of all the links.

Find Elements by Index

Example :-

```
<html>
  <head>
    <title>The JQuery Example</title>
    <script type = "text/javascript"
      src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>

    <script type = "text/javascript" language = "javascript">
      $(document).ready(function() {
        $("li").eq(2).addClass("selected");
      });
    </script>

    <style>
      .selected { color:red; }
    </style>
  </head>

  <body>
    <div>
      <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>
        <li>list item 6</li>
      </ul>
    </div>
  </body>
</html>
```

Filtering out Elements

The filter(selector) method can be used to filter out all elements from the set of matched elements that do not match the specified selector(s). The selector can be written using any selector syntax.

Example :-

```
<html>
  <head>
    <title>The JQuery Example</title>
    <script type = "text/javascript"
      src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>

    <script type = "text/javascript" language = "javascript">
      $(document).ready(function() {
        $("li").filter(".middle").addClass("selected");
      });
    </script>

    <style>
      .selected { color:red; }
    </style>
  </head>

  <body>
    <div>
      <ul>
        <li class = "top">list item 1</li>
        <li class = "top">list item 2</li>
        <li class = "middle">list item 3</li>
        <li class = "middle">list item 4</li>
        <li class = "bottom">list item 5</li>
        <li class = "bottom">list item 6</li>
      </ul>
    </div>
  </body>
</html>
```

Locating Descendant Elements

The find(selector) method can be used to locate all the descendant elements of a particular type of elements. The selector can be written using any selector syntax.

Example :-

Following is an example which selects all the elements available inside different <p> elements −

```
<html>
  <head>
    <title>The JQuery Example</title>
    <script type = "text/javascript"
      src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>

    <script type = "text/javascript" language = "javascript">
```

```
    $(document).ready(function() {
      $("p").find("span").addClass("selected");
    });
  </script>

  <style>
    .selected { color:red; }
  </style>
 </head>

 <body>
  <p>This is 1st paragraph and <span>THIS IS RED</span></p>
  <p>This is 2nd paragraph and <span>THIS IS ALSO RED</span></p>
 </body>
</html>
```

JQuery DOM Filter Methods

Following table lists down useful methods which you can use to filter out various elements from a list of DOM elements −

- **eq(index) :-** Reduce the set of matched elements to a single element.
- **filter(selector) :-** Removes all elements from the set of matched elements that do not match the specified selector(s).
- **filter(fn) :-** Removes all elements from the set of matched elements that do not match the specified function.
- **is(selector) :-** Checks the current selection against an expression and returns true, if at least one element of the selection fits the given selector.
- **map(callback) :-** Translate a set of elements in the jQuery object into another set of values in a jQuery array (which may, or may not contain elements).
- **not(selector) :-** Removes elements matching the specified selector from the set of matched elements.
- **slice(start, [end]) :-** Selects a subset of the matched elements.

JQuery DOM Traversing Methods

Following table lists down other useful methods which you can use to locate various elements in a DOM −

- **add(selector) :-** Adds more elements, matched by the given selector, to the set of matched elements.
- **andSelf( ) :-** Add the previous selection to the current selection.
- **children([selector]) :-** Get a set of elements containing all of the unique immediate children of each of the matched set of elements.
- **closest(selector) :-** Get a set of elements containing the closest parent element that matches the specified selector, the starting element included.
- **contents( ) :-** Find all the child nodes inside the matched elements (including text nodes), or the content document, if the element is an iframe.
- **end( ) :-** Revert the most recent 'destructive' operation, changing the set of matched elements to its previous state.
- **find(selector) :-** Searches for descendant elements that match the specified selectors.

- **next([selector]) :-** Get a set of elements containing the unique next siblings of each of the given set of elements.
- **nextAll([selector]) :-** Find all sibling elements after the current element.
- **offsetParent( ) :-** Returns a jQuery collection with the positioned parent of the first matched element.
- **parent([selector]) :-** Get the direct parent of an element. If called on a set of elements, parent returns a set of their unique direct parent elements.
- **parents([selector]) :-** Get a set of elements containing the unique ancestors of the matched set of elements (except for the root element).
- **prev([selector]) :-** Get a set of elements containing the unique previous siblings of each of the matched set of elements.
- **prevAll([selector]) :-** Find all sibling elements in front of the current element.
- **siblings([selector]) :-** Get a set of elements containing all of the unique siblings of each of the matched set of elements.

Apply CSS Properties

This is very simple to apply any CSS property using JQuery method css(PropertyName, PropertyValue).

Syntax –

```
selector.css( PropertyName, PropertyValue );
```

Here you can pass PropertyName as a javascript string and based on its value, PropertyValue could be string or integer.

Example :-

```
<html>
  <head>
    <title>The jQuery Example</title>
    <script type = "text/javascript"
      src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>
    <script type = "text/javascript" language = "javascript">
      $(document).ready(function() {
        $("li").eq(2).css("color", "red");
      });
    </script>
  </head>

  <body>
    <div>
      <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>
        <li>list item 6</li>
      </ul>
    </div>
```

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

Apply Multiple CSS Properties

You can apply multiple CSS properties using a single JQuery method CSS({key1:val1, key2:val2....). You can apply as many properties as you like in a single call.

syntax –

```
selector.css( {key1:val1, key2:val2....keyN:valN})
```

Here you can pass key as property and val as its value as described above.

Example :-

```
<html>
  <head>
    <title>The jQuery Example</title>
    <script type = "text/javascript"
      src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>

    <script type = "text/javascript" language = "javascript">
      $(document).ready(function() {
        $("li").eq(2).css({"color":"red", "background-color":"green"});
      });
    </script>
  </head>

  <body>
    <div>
      <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>
        <li>list item 6</li>
      </ul>
    </div>
  </body>
</html>
```

Setting Element Width & Height

The width(val) and height(val) method can be used to set the width and height respectively of any element.

Example :-

```
<script type = "text/javascript" language = "javascript">
        $(document).ready(function() {
                $("div:first").width(100);
                $("div:first").css("background-color", "blue");
        });
</script>
```

JQuery CSS Methods

Following table lists down all the methods which you can use to play with CSS properties −

- **css(name) :-** Return a style property on the first matched element.
- **css(name, value) :-** Set a single style property to a value on all matched elements.
- **css(properties) :-** Set a key/value object as style properties to all matched elements.
- **height(val) :-** Set the CSS height of every matched element.
- **height( ) :-** Get the current computed, pixel, height of the first matched element.
- **innerHeight( ) :-** Gets the inner height (excludes the border and includes the padding) for the first matched element.
- **innerWidth( ) :-** Gets the inner width (excludes the border and includes the padding) for the first matched element.
- **offset( ) :-** Get the current offset of the first matched element, in pixels, relative to the document.
- **offsetParent( ) :-** Returns a jQuery collection with the positioned parent of the first matched element.
- **outerHeight([margin]) :-** Gets the outer height (includes the border and padding by default) for the first matched element.
- **outerWidth([margin]) :-** Get the outer width (includes the border and padding by default) for the first matched element.
- **position( ) :-** Gets the top and left position of an element relative to its offset parent.
- **scrollLeft(val) :-** When a value is passed in, the scroll left offset is set to that value on all matched elements.
- **scrollLeft( ) :-** Gets the scroll left offset of the first matched element.
- **scrollTop(val) :-** When a value is passed in, the scroll top offset is set to that value on all matched elements.
- **scrollTop( ) :-** Gets the scroll top offset of the first matched element.
- **width(val) :-** Set the CSS width of every matched element.
- **width( ) :-** Get the current computed, pixel, width of the first matched element.

Content Manipulation

The html( ) method gets the html contents (innerHTML) of the first matched element.

Syntax :-

```
selector.html( )
```

Example :-

```
<html>
  <head>
    <title>The jQuery Example</title>
    <script type = "text/javascript"
      src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>
```

```
    <script type = "text/javascript" language = "javascript">
      $(document).ready(function() {
        $("div").click(function () {
          var content = $(this).html();
          $("#result").text( content );
        });
      });
    </script>

    <style>
      #division{ margin:10px;padding:12px; border:2px solid #666; width:60px;}
    </style>
  </head>

  <body>
    <p>Click on the square below:</p>
    <span id = "result"> </span>

    <div id = "division" style = "background-color:blue;">
      This is Blue Square!!
    </div>
  </body>
</html>
```

DOM Element Replacement

You can replace a complete DOM element with the specified HTML or DOM elements. The replaceWith(content) method serves this purpose very well.

syntax :-

```
selector.replaceWith( content )
```

Example :-

```
<script type = "text/javascript" language = "javascript">
        $(document).ready(function() {
                $("div").click(function () {
                        $(this).replaceWith("<h1>JQuery is Great</h1>");
                });
        });
</script>
```

Removing DOM Elements

There may be a situation when you would like to remove one or more DOM elements from the document. JQuery provides two methods to handle the situation.

The empty( ) method remove all child nodes from the set of matched elements where as the method remove(expr) method removes all matched elements from the DOM.

syntax :−

```
selector.remove( [ expr ])

or

selector.empty( )
```

Example :-

```
<script type = "text/javascript" language = "javascript">
      $(document).ready(function() {
         $("div").click(function () {
               $(this).remove( );
         });
      });
  </script>
```

Inserting DOM Elements

There may be a situation when you would like to insert new one or more DOM elements in your existing document. JQuery provides various methods to insert elements at various locations.

The after(content) method insert content after each of the matched elements where as the method before(content) method inserts content before each of the matched elements.

Syntax :−

```
selector.after( content )

or

selector.before( content )
```

Example :-

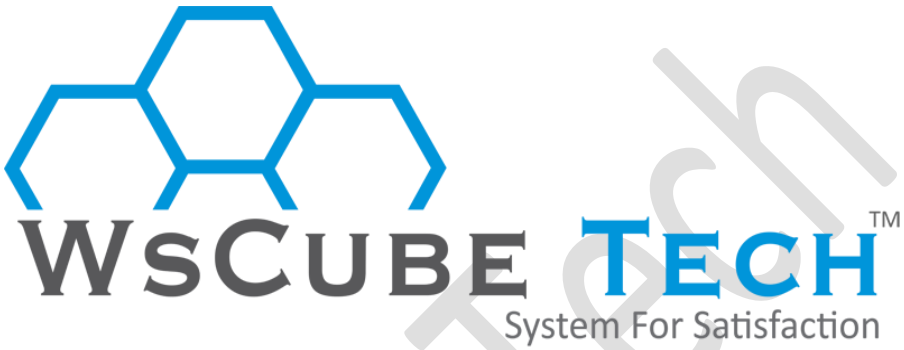
```
    <script type = "text/javascript" language = "javascript">
       $(document).ready(function() {
          $("div").click(function () {
             $(this).before('<div class="div"></div>' );
          });
       });
    </script>
```

DOM Manipulation Methods

- **after(content) :-** Insert content after each of the matched elements.
- **append(content) :-** Append content to the inside of every matched element.
- **appendTo(selector) :-** Append all of the matched elements to another, specified, set of elements.
- **before(content) :-** Insert content before each of the matched elements.
- **clone(bool) :-** Clone matched DOM Elements, and all their event handlers, and select the clones.
- **clone( ) :-** Clone matched DOM Elements and select the clones.
- **empty( ) :-** Remove all child nodes from the set of matched elements.
- **html(val) :-** Set the html contents of every matched element.
- **html( ) :-** Get the html contents (innerHTML) of the first matched element.
- **insertAfter(selector) :-** Insert all of the matched elements after another, specified, set of elements.
- **insertBefore(selector) :-** Insert all of the matched elements before another, specified, set of elements.
- **prepend(content) :-** Prepend content to the inside of every matched element.
- **prependTo(selector) :-** Prepend all of the matched elements to another, specified, set of elements.
- **remove(expr) :-** Removes all matched elements from the DOM.
- **replaceAll(selector) :-** Replaces the elements matched by the specified selector with the matched elements.
- **replaceWith(content) :-** Replaces all matched elements with the specified HTML or DOM elements.
- **text(val) :-** Set the text contents of all matched elements.
- **text( ) :-** Get the combined text contents of all matched elements.
- **wrap(elem) :-** Wrap each matched element with the specified element.
- **wrap(html) :-** Wrap each matched element with the specified HTML content.
- **wrapAll(elem) :-** Wrap all the elements in the matched set into a single wrapper element.
- **wrapAll(html) :-** Wrap all the elements in the matched set into a single wrapper element.
- **wrapInner(elem) :-** Wrap the inner child contents of each matched element (including text nodes) with a DOM element.
- **wrapInner(html) :-** Wrap the inner child contents of each matched element (including text nodes) with an HTML structure.

WsCube Tech

1st Floor, Laxmi Tower, Bhaskar Circle, Ratanada,
Jodhpur.
Mobile : +91-92696-98122 , 85610-89567



BOOTSTRAP

Building Responsive Websites with BOOTSTRAP

BY **WsCube Tech**

What is Twitter Bootstrap?

Bootstrap is a sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development. It uses HTML, CSS and Javascript.

History

Bootstrap was developed by Mark Otto and Jacob Thornton at Twitter. It was released as an open source product in August 2011 on GitHub.
Why use Bootstrap?

- **Mobile first approach :-** Since Bootstrap 3, the framework consists of Mobile first styles throughout the entire library instead of in separate files.
- **Browser Support :-** It is supported by all popular browsers.
- **Easy to get started :-** With just the knowledge of HTML and CSS anyone can get started with Bootstrap. Also the Bootstrap official site has a good documentation.
- **Responsive design :-** Bootstrap's responsive CSS adjusts to Desktops,Tablets and Mobiles.

- Provides a clean and uniform solution for building an interface for developers.
- It contains beautiful and functional built-in components which are easy to customize.
- It also provides web based customization.
- And best of all it is an open source.

Bootstrap 3 is mobile-first

Bootstrap 3 is designed to be responsive to mobile devices. Mobile-first styles are part of the core framework.
To ensure proper rendering and touch zooming, add the following <meta> tag inside the <head> element:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
The initial-scale=1 part sets the initial zoom level when the page is first loaded by the browser

File structure

P R E C O M P I L E D B O O T S T R A P
Once the compiled version Bootstrap is downloaded, extract the ZIP file, and you will see the following file/directory structure:

As you can see there are compiled CSS and JS (bootstrap.*), as well as compiled and minified CSS and JS (bootstrap.min.*). Fonts from Glyphicons are included, as is the optional Bootstrap theme.

BOOTSTRAP SOURCE CODE

If you downloaded the Bootstrap source code then the file structure would be as follows:

- The files under less/, js/, and fonts/ are the source code for Bootstrap CSS, JS, and icon fonts (respectively).
- The dist/ folder includes everything listed in the precompiled download section above.
- docs-assets/, examples/, and all *.html files are Bootstrap documentation.

HTML Template

A basic HTML template using Bootstrap would look like as this

```
<!DOCTYPE html>
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
  <h1>My First Bootstrap Page</h1>
  <p>This is some text.</p>
</div>

</body>
</html>
```

Media Queries

Media query is a really fancy term for "conditional CSS rule". It simply applies some CSS based on certain conditions set forth. If those conditions are met, the style is applied.

Media Queries in Bootstrap allow you to move, show and hide content based on viewport size. Following media queries are used in LESS files to create the key breakpoints in the Bootstrap grid system.
Occasionally these are expanded to include a max-width to limit CSS to a narrower set of devices.

```
/* Extra small devices (phones, less than 768px) */
/* No media query since this is the default in Bootstrap */

/* Small devices (tablets, 768px and up) */ @media (min-width: @screen-sm-min) { ... }

/* Medium devices (desktops, 992px and up) */ @media (min-width: @screen-md-min) { ... }

/* Large devices (large desktops, 1200px and up) */ @media (min-width: @screen-lg-min) { ... }
@media (max-width: @screen-xs-max) { ... }
@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max) { ... } @media (min-width: @screen-md-min) and (max-width: @screen-md-max) { ... } @media (min-width: @screen-lg-min) { ... }
```

Media queries have two parts, a device specification and then a size rule. In the above case, the following rule is set

```
@media (min-width: @screen-sm-min) and (max-width: @screen-sm-max) { ... }
```

For all devices no matter what kind with min-width: @screen-sm-min if the width of the screen gets smaller than @screen-sm-max, then do something.

Grid Classes
The Bootstrap grid system has four classes:

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

Grid options

The following table summarizes aspects of how Bootstrap grid system works across multiple devices:

Extra small devices Phones (<768px) Small devices Tablets (≥768px) Medium devices Desktops (≥992px) Large devices Desktops (≥1200px)

	Extra small devices Phones (<768px)	**Small devices Tablets (≥768px)**	**Medium devices Desktops (≥992px)**	**Large devices Desktops (≥1200px)**
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints	Collapsed to start, horizontal above breakpoints	Collapsed to start, horizontal above breakpoints
Max container width	None (auto)	750px	970px	1170px
Class prefix	**col-xs-**	**.col-sm-**	**.col-md-**	**.col-lg-**
# of columns	**12**	**12**	**12**	**12**
Max column width	Auto	60px	78px	95px

Gutter width	30px (15px on each side of a column)	30px (15px on each side of a column)	30px (15px on each side of a column)	30px (15px on each side of a column)
Nestable	Yes	Yes	Yes	Yes
Offsets	Yes	Yes	Yes	Yes
Column ordering	Yes	Yes	Yes	Yes

BASIC GRID STRUCTURE

```
<div class="row">
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
</div>
<div class="row">
  <div class="col-md-8">.col-md-8</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-6">.col-md-6</div>
  <div class="col-md-6">.col-md-6</div>
</div>
```

Example :-

Fluid container

Turn any fixed-width grid layout into a full-width layout by changing your outermost .container to .container-fluid.

```
<div class="container-fluid">
  <div class="row">
  </div>
</div>
```

Example: Mobile and desktop

<!-- Stack the columns on mobile by making one full-width and the other half-width -->

```
<div class="row">
  <div class="col-xs-12 col-md-8">.col-xs-12 .col-md-8</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>

<!-- Columns start at 50% wide on mobile and bump up to 33.3% wide on desktop -->
<div class="row">
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>

<!-- Columns are always 50% wide, on mobile and desktop -->
<div class="row">
  <div class="col-xs-6">.col-xs-6</div>
  <div class="col-xs-6">.col-xs-6</div>
</div>
```

Bootstrap Grid System Example: Medium and Large Device

We have seen the basic grid system in Example: Stacked-to-horizontal. Here we had used 2 divs and gave them the 50%/50% split at the medium viewport width:

```
<div class=”col-md-6” >.....</div>
<div class=”col-md-6” >.....</div>
```

But at large your design could really be better as a 33%/66%. So what we’re going to do is set it up to change the column widths at the breakpoint:

```
<div class=”col-md-6 col-lg-4” >.....</div>
<div class=”col-md-6 col-lg-4” >.....</div>
```

Now Bootstrap is going to say “at the medium size, I look at classes with md in them and use those. At the large size, I look at classes with the word lg in them and use those. In this case, our 2 divs will go from a 50%/50% split and then up to a 33%/66%.

Bootstrap Grid System Example: Mobile, Tablet, Desktops
We have seen an example for Medium and Large Device. Now let us take it to another level, where we would want to change it for the extra small phone size as well. Say we want to add the option for the columns to be split 25%/75% for tablets, we go like this:

```
<div class="col-sm-3 col-md-6 col-lg-4">....</div>
<div class="col-sm-9 col-md-6 col-lg-8">....</div>
```

Now this gives us 3 different column layouts at each point. On a phone, it will be 25% on the left, and 75% on the right. On a tablet, it will be 50%/50% again, and on a large viewport, it will be 33%/66%. 3 different layouts for each of the 3 responsive sizes.

Bootstrap Images

Bootstrap provides three classes that can be used to apply some simple styles to images:

- **.img-rounded:** adds border-radius:6px to give the image rounded corners.
- **.img-circle:** makes the entire image round by adding border-radius:500px.
- **.img-thumbnail:** adds a bit of padding and a gray border.

Text alignment

```
<p class="text-left">Left aligned text on all viewport sizes.</p>
<p class="text-center">Center aligned text on all viewport sizes.</p>
<p class="text-right">Right aligned text on all viewport sizes.</p>

<p class="text-sm-left">Left aligned text on viewports sized SM (small) or wider.</p>
<p class="text-md-left">Left aligned text on viewports sized MD (medium) or wider.</p>
<p class="text-lg-left">Left aligned text on viewports sized LG (large) or wider.</p>
<p class="text-xl-left">Left aligned text on viewports sized XL (extra-large) or wider.</p>
```

Text transform

```
<p class="text-lowercase">Lowercased text.</p>
<p class="text-uppercase">Uppercased text.</p>
<p class="text-capitalize">CapiTaliZed text.</p>
```

Contextual Colors and Backgrounds
Bootstrap also has some contextual classes that can be used to provide "meaning through colors".
The classes for text colors are: .text-muted, .text-primary, .text-success, .text-info, .text-warning, and .text-danger:

Example :-

This text is muted.

This text is important.
This text indicates success.
This text represents some information.
This text represents a warning.
This text represents danger.
Try it Yourself »

The classes for background colors are:.bg-primary, .bg-success, .bg-info, .bg-warning, and .bg-danger:

Example :-

This text is important.
This text indicates success.
This text represents some information.
This text represents a warning.
This text represents danger.

Responsive images

Bootstrap 3 allows to make the images responsive by adding a class .img-responsive to the <img>tag. This class applies max-width: 100%; and height: auto; to the image so that it scales nicely to the parent element.

```
<img src="..." class="img-responsive" alt="Responsive image">
```

Containers

Use class .container to wrap a page's content and easily center the content's as shown below.

```
<div class="container">
...
</div>
```

Note that, due to padding and fixed widths, containers are not nestable by default. Take a look at bootstrap.css file:

Bootstrap Tables

Bootstrap provides a clean layout for building tables. Some of the table elements supported by Bootstrap are:

Tag Description

<table> : Wrapping element for displaying data in a tabular format
<thead> : Container element for table header rows (<tr>) to label table columns
<tbody> : Container element for table rows (<tr>) in the body of the table
<tr> : Container element for a set of table cells (<td> or <th>) that appears on a single row
<td> : Default table cell
<th> : Special table cell for column (or row, depending on scope and placement) labels. Must be used within a <thead>
<caption> : Description or summary of what the table holds.

Basic Table

If you want a nice, basic table style with just some light padding and horizontal dividers, add the base clasS of .table to any table as shown in the following example

Optional Table Classes

Along with the base table markup and the .table class, there are a few additional classes that you can use to style the markup. Following sections will give you a glimpse of all these classes.

STRIPED TABLE

By adding the .table-striped class, you will get stripes on rows within the <tbody>.

```
<table class="table table-striped">
  <table>
```

BORDERED TABLE

By adding the .table-bordered class, you will get borders surrounding every element and rounded corners around the entire table.

```
<table class="table table-bordered"></table>
```

HOVER TABLE

By adding the .table-hover class, a light gray background will be added to rows while the cursor hovers over them.

```
<table class="table table-bordered"></table>
```

CONDENSED TABLE

By adding the .table-condensed class, row padding is cut in half to condense the table. as seen in the following example. This is useful if you want denser information.

```
<table class="table table-condensed"></table>
```

Contextual classes

The Contextual classes shown in following table will allow you to change the background color of your table rows or individual cells.

Class Description

.active : Applies the hover color to a particular row or cell

.**success :** Indicates a successful or positive action

.**warning :** Indicates a warning that might need attention

.danger : Indicates a dangerous or potentially negative action

These classes can be applied to <tr>, <td> or <th>.

Responsive tables

By wrapping any .table in .table-responsive class, you will make the table scroll horizontally up to small devices (under 768px). When viewing on anything larger than 768px wide, you will not see any difference in these tables.

```
<div class="table-responsive">
        <table class="table">
        </table>
</div>
```

Bootstrap Forms

In this chapter we will study how to create forms with ease using Bootstrap. Bootstrap makes it easy with the simple HTML markup and extended classes for different styles of forms.

Form Layout

Bootstrap provides you with following types of form layouts:

- Vertical (default) form
- Inline form
- Horizontal form

VERTICAL OR BASIC FORM

The basic form structure comes with Bootstrap; individual form controls automatically receive some global styling. To create a basic form do the following:

- Add a role form to the parent <form> element.
- Wrap labels and controls in a with class .form-group. This is needed for optimum spacing.

```
<form role="form">
 <div class="form-group">
 </div>
</form>
```

INLINE FORM

To create a form where all of the elements are inline, left aligned and labels are alongside, add the class .form-inline to the <form> tag.

```
<form class="form-inline" role="form">
        <div class="form-group">
          <label class="sr-only" for="name">Name</label>
                <input type="text" class="form-control" id="name"                placeholder="Enter Name">
          </div>
</form>
```

- By default Inputs, selects, and textareas have 100% width in Bootstrap. You need to set a width on the form controls when using inline form.
- Using the class .sr-only you can hide the labels of the inline forms.

HORIZONTAL FORM

Horizontal forms stands apart from the others not only in the amount of markup, but also in the presentation of the form. To create a form that uses the horizontal layout, do the following:

- Add a class of .form-horizontal to the parent <form> element.
- Wrap labels and controls in a with class .form-group.
- Add a class of .control-label to the labels.

```
<form class="form-horizontal" role="form">
    <div class="form-group"></div>
</form>
```

INPUTS

The most common form text field is the input—this is where users will enter most of the essential form data. Bootstrap offers support for all native HTML5 input types: text, password, datetime, datetime-local, date, month, time, week, number, email, url, search, tel, and color. Proper type declaration is required to make Inputs fully styled.

```
<form role="form">
<div class="form-group">
<label for="name">Label</label>
<input type="text" class="form-control" placeholder="Text input">
</div>
</form>
```

TEXTAREA

The textarea is used when you need multiple lines of input. Change rows attribute as necessary (fewer rows = smaller box, more rows = bigger box).

```
<form role="form">
<div class="form-group">
<label for="name">Text Area</label>
<textarea class="form-control" rows="3"></textarea>
</div>
</form>
```

CHECKBOXES AND RADIOS

Checkboxes and radio buttons are great when you want users to choose from a list of preset options.

- When building a form, use checkbox if you want the user to select any number of options from a list. Use radio if you want to limit the user to just one selection.
- Use .checkbox-inline or .radio-inline class to a series of checkboxes or radios for controls appear on the same line.

The following example demonstrates both (default and inline) types:

```
<label for="name">Example of Default Checkbox and radio button </label>
<div class="checkbox">
<label><input type="checkbox" value="">Option 1</label>
```

```
</div>
<div class="checkbox">
<label><input type="checkbox" value="">Option 2</label>
</div>
<div class="radio">
<label>
<input type="radio" name="optionsRadios" id="optionsRadios1" value="option1" checked> Option 1
</label>
</div>
<div class="radio">
<label>
<input type="radio" name="optionsRadios" id="optionsRadios2" value="option2">
Option 2 - selecting it will deselect option 1
</label>
</div>
```

SELECTS

A select is used when you want to allow the user to pick from multiple options, but by default it only allows one.

- Use <select> for list options with which the user is familiar, such as states or numbers.
- Use multiple="multiple" to allow the user to select more than one option. The following example demonstrates both (select and multiple) types:

```
<form role="form">
<div class="form-group">
<label for="name">Select list</label>
<select class="form-control">
<option>1</option>
<option>2</option>
<option>3</option>
<option>4</option>
<option>5</option>
</select>

<label for="name">Mutiple Select list</label>
<select multiple class="form-control">
<option>1</option>
<option>2</option>
<option>3</option>
<option>4</option>
<option>5</option>
</select>
</div>
</form>
```

Static control

Use the class .form-control-static on a <p>, when you need to place plain text next to a form label within a horizontal form.

```
<form class="form-horizontal" role="form">
<div class="form-group">
<label class="col-sm-2 control-label">Email</label>
<div class="col-sm-10">
<p class="form-control-static">email@example.com</p>
</div>
</div>
<div class="form-group">
<label for="inputPassword" class="col-sm-2 control-label">Password</label>
<div class="col-sm-10">
<input type="password" class="form-control" id="inputPassword" placeholder="Password">
</div>
</div>
</form>
```

Form Control Sizing

You can set heights and widths of forms using classes like .input-lg and .col-lg-* respectively.

```
 <form role="form">
<div class="form-group">
<input class="form-control input-lg" type="text" placeholder=".input-lg">
</div>
</form>
```

Bootstrap Buttons

This chapter will discuss about how to use Bootstrap button with examples. Anything that is given a class of .btn will inherit the default look of a gray button with rounded corners. However Bootstrap provides some options to style buttons, which are summarized in the following table:

Class	Description
Btn	Default/ Standard button.
Btn-primary	Provides extra visual weight and identifies the primary action in a set of buttons.
Btn-success	Indicates a successful or positive action.
Btn-info	Contextual button for informational alert messages.

Btn-warning	Indicates caution should be taken with this action.
Btn-danger	Indicates a dangerous or potentially negative action.
Btn-link	Deemphasize a button by making it look like a link while maintaining button behavior.

Button Size

The following table summarizes classes used to get buttons of various sizes:

Class	Description
.btn-lg	This makes button size large.
.btn-sm	This makes button size small.
.btn-xs	This makes button size with extra small.
.btn-block	This creates block level buttons—those that span the full width of a parent.

Bootstarp Responsive Utilities

Bootstrap provides some handful helper classes, for faster mobile-friendly development. These can be used for showing and hiding content by device via media query combined with large, small, and medium devices.

Use these sparingly and avoid creating entirely different versions of the same site. Responsive utilities are currently only available for block and table toggling.

Classes	Devices
.visible-xs	Extra small (less than 768px) visible
.visible-sm	Small (up to 768 px) visible

.visible-md	Medium (768 px to 991 px) visible
.visible-lg	Larger (992 px and above) visible
.hidden-xs	Extra small (less than 768px) hidden
.hidden-sm	Small (up to 768 px) hidden
.hidden-md	Medium (768 px to 991 px) hidden
.hidden-lg	Larger (992 px and above) hidden

Bootstrap Glyphicons

Where to find Glyphicons?

Now that we have downloaded Bootstrap 3.x version and understand its directory structure from chapter Environment Setup, glyphicons can be found within the fonts folder. This contains following files:

- glyphicons-halflings-regular.eot
- glyphicons-halflings-regular.svg
- glyphicons-halflings-regular.ttf
- glyphicons-halflings-regular.woff

Associated CSS rules are present within bootstrap.css and bootstrap-min.css files within css folder ofdist folder. You can see the available glyphicons at this link : Glyphicons List

Usage

To use the icons, simply use the following code just about anywhere in your code. Leave a space between the icon and text for proper padding.

```
<span class="glyphicon glyphicon-search"></span>
```

Bootstrap Dropdowns

Dropdown menus are toggleable, contextual menus for displaying links in a list format. This can be made interactive with the dropdown JavaScript plugin.
To use dropdown, just wrap the dropdown menu within the class .dropdown. Following example demonstrates a basic dropdown menu:

```
<div class="dropdown">
        <button type="button" class="btn dropdown-toggle" id="dropdownMenu1" data-toggle="dropdown">
        Topics
```

```
            <span class="caret"></span>
        </button>
        <ul class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu1">
            <li role="presentation">
            <a role="menuitem" tabindex="-1" href="#">Java</a>
          </li>
            <li role="presentation">
            <a role="menuitem" tabindex="-1" href="#">Data Mining</a>
          </li>
            <li role="presentation">
            <a role="menuitem" tabindex="-1" href="#"> Data Communication/Networking
            </a>
          </li>
            <li role="presentation" class="divider"></li>
            <li role="presentation">
            <a role="menuitem" tabindex="-1" href="#">Separated link</a>
          </li>
        </ul>
</div>
```

OPTIONS

ALIGNMENT

Align the dropdown menu to right by adding the class .pull-right to .dropdown-menu. Following example demonstrates this:

```
<div class="dropdown">
        <button type="button" class="btn dropdown-toggle" id="dropdownMenu1" data-
toggle="dropdown">Topics
            <span class="caret"></span>
        </button>
        <ul class="dropdown-menu pull-right" role="menu" aria-labelledby="dropdownMenu1">
            <li role="presentation">
            <a role="menuitem" tabindex="-1" href="#">Java</a>
          </li>
            <li role="presentation">
            <a role="menuitem" tabindex="-1" href="#">Data Mining</a>
          </li>
            <li role="presentation">
            <a role="menuitem" tabindex="-1" href="#"> Data Communication/Networking
            </a>
          </li>
            <li role="presentation" class="divider"></li>
            <li role="presentation">
            <a role="menuitem" tabindex="-1" href="#">Separated link</a>
          </li>
        </ul>
```

```
</div>
```

Bootstrap Button Groups

Button groups allow multiple buttons to be stacked together on a single line. This is useful when you want to place items like alignment buttons together. You can add on optional JavaScript radio and checkbox style behavior with Bootstrap Button Plugin.
Following table summarizes the important classes Bootstrap provides to use button groups:

Class	Description	Code Sample
.btn-group	This class is used form a basic button group. Wrap a series of buttons with class .btn in.btn-group.	 <button type="button" class="btn btn-default">Button1</button> <button type="button" class="btn btn-default">Button2</button>
.btn-toolbar	This helps to combine sets of into a for more complex components.	 ... ...

.btn-group-lg, .btn-group-sm, .btn-group-xs	These classes can be applied to button group instead of resizing each button.	... ...
.btn-group-vertical	This class make a set of buttons appear vertically stacked rather than horizontally.	 ...

Bootstrap Button Dropdowns

This chapter will discuss about how to add dropdown menu to buttons using Bootstrap classes. To add a dropdown to a button, simply wrap the button and dropdown menu in a .btn-group. You can also use to act as an indicator that the button is a dropdown.

```
<div class="btn-group">
        <button type="button" class="btn btn-default dropdown-toggle" data-toggle="dropdown">
        Default <span class="caret"></span>
        </button>
        <ul class="dropdown-menu" role="menu">
           <li><a href="#">Action</a></li>
           <li><a href="#">Another action</a></li>
           <li><a href="#">Something else here</a></li>
        </ul>
</div>
```

Split Button Dropdowns

Split button dropdowns use the same general style as the dropdown button but add a primary action along with the dropdown. Split buttons have the primary action on the left and a toggle on the right that displays the dropdown.

```
<div class="btn-group">
        <button type="button" class="btn btn-default">Default</button>
        <button type="button" class="btn btn-default dropdown-toggle" data-toggle="dropdown">
```

```
                <span class="caret"></span>
                <span class="sr-only">Toggle Dropdown</span>
        </button>
        <ul class="dropdown-menu" role="menu">
          <li><a href="#">Action</a></li>
          <li><a href="#">Another action</a></li>
          <li><a href="#">Something else here</a></li>
                <li class="divider"></li>
          <li><a href="#">Separated link</a></li>
        </ul>
</div>
```

Checkboxes and radio addons

You can preappend or append radio buttons and checkboxes instead of text as demonstrated in the following example:

```
<div class="row">
        <div class="col-lg-6">
                <div class="input-group">
                <span class="input-group-addon">
                        <input type="checkbox">
                </span>
                <input type="text" class="form-control">
          </div><!-- /input-group -->
        </div><!-- /.col-lg-6 --><br>
        <div class="col-lg-6">
                <div class="input-group">
                <span class="input-group-addon">
                        <input type="radio">
                </span>
                <input type="text" class="form-control">
          </div><!-- /input-group -->
        </div><!-- /.col-lg-6 -->
</div><!-- /.row →
```

Button addons

You can even preappend or append buttons in input groups. Instead of .input-group-addon class, you'll need to use class .input-group-btn to wrap the buttons. This is required due to default browser styles that cannot be overridden.

Example :-

```
<div class="col-lg-6">
        <div class="input-group">
                <span class="input-group-btn">
                <button class="btn btn-default" type="button"> Go!</button>
          </span>
          <input type="text" class="form-control">
```

```
        </div><!-- /input-group -->
</div><!-- /.col-lg-6 --><br>
<div class="col-lg-6">
        <div class="input-group">
           <input type="text" class="form-control">
                <span class="input-group-btn">
                <button class="btn btn-default" type="button"> Go!</button>
           </span>
        </div><!-- /input-group -->
</div><!-- /.col-lg-6 -->
```

Bootstrap Navigation Elements

In this chapter we will discuss about how Bootstrap provides a few different options for styling navigation elements. All of them share the same markup and base class, .nav. Bootstrap also provides a helper class, to share markup and states. Swap modifier classes to switch between each style.

Tabular Navigation or Tabs

To create a tabbed navigation menu:

- Start with a basic unordered list with the base class of .nav
- Add class .nav-tabs.

Following example demonstrates this:

```
<p>Tabs Example</p>
<ul class="nav nav-tabs">
        <li class="active"><a href="#">Home</a></li>
        <li><a href="#">SVN</a></li>
        <li><a href="#">iOS</a></li>
        <li><a href="#">VB.Net</a></li>
        <li><a href="#">Java</a></li>
        <li><a href="#">PHP</a></li>
</ul>
```

PILLS Navigation

BASIC PILLS

To turn the tabs into pills, follow the same steps as above, use the class .nav-pills instead of .nav-tabs.

Example :-

```
<p>Tabs Example</p>
<ul class="nav nav-pills">
        <li class="active"><a href="#">Home</a></li>
        <li><a href="#">SVN</a></li>
        <li><a href="#">iOS</a></li>
        <li><a href="#">VB.Net</a></li>
        <li><a href="#">Java</a></li>
        <li><a href="#">PHP</a></li>
</ul>
```

PILLS WITH DROPDOWNS

do the same thing with pills, simply swap the .nav-tabs class with .nav-pills as shown in the following example.

```
<p>Pills With Dropdown Example</p>
<ul class="nav nav-pills">
        <li class="active"><a href="#">Home</a></li>
        <li><a href="#">SVN</a></li>
        <li><a href="#">iOS</a></li>
        <li><a href="#">VB.Net</a></li>
        <li class="dropdown">
                <a class="dropdown-toggle" data-toggle="dropdown" href="#"> Java <span class="caret"></span>
                </a>
                <ul class="dropdown-menu">
                <li><a href="#">Swing</a></li>
                <li><a href="#">jMeter</a></li>
                <li><a href="#">EJB</a></li>
                <li class="divider"></li>
                <li><a href="#">Separated link</a></li>
          </ul>
        </li>
        <li><a href="#">PHP</a></li>
</ul>
```

Bootstrap Navbar

The Navbar is a nice feature, and is one of the prominent features of Bootstrap sites. Navbars are responsive meta components that serve as navigation headers for your application or site. Navbars collapse in mobile views and become horizontal as the available viewport width increases. At its core, the navbar includes styling for site names and basic navigation.

Default navbar

To create a default navbar:

- Add classes .navbar, .navbar-default to the <nav> tag.
- Add role="navigation" to the above element, to help with accessibility.
- Add a header class .navbar-header to the element. Include an <a> element with classnavbar-brand. This will give the text a slightly larger size.
- To add links to the navbar, simply add an unordered list with a classes of .nav, .navbar-nav.

Example :-

```
<nav class="navbar navbar-default" role="navigation">
        <div>
                <ul class="nav navbar-nav">
                <li class="active"><a href="#">iOS</a></li>
```

```
                <li><a href="#">SVN</a></li>
                <li class="dropdown">
                <a href="#" class="dropdown-toggle" data-toggle="dropdown"> Java
                <b class="caret"></b>
                </a>
                        <ul class="dropdown-menu">
                                <li><a href="#">jmeter</a></li>
                                <li><a href="#">EJB</a></li>
                                <li><a href="#">Jasper Report</a></li>
                                <li class="divider"></li>
                                <li><a href="#">Separated link</a></li>
                                <li class="divider"></li>
                                <li><a href="#">One more separated link</a></li>
                        </ul>
                </li>
          </ul>
        </div>
</nav>
```

Responsive navbar

To add the responsive features to the navbar, the content that you want to be collapsed needs to be wrapped in a
 with classes .collapse, .navbar-collapse. The collapsing nature is tripped by a button that has a the class of .navbar-toggle and then features two data- elements. The first, data-toggle, is used to tell the JavaScript what to do with the button, and the second, data-target, indicates which element to toggle. Three with a class of .icon-bar create what I like to call the hamburger button. This will toggle the elements that are in the .nav- collapse . For this feature to work, you need to include the Bootstrap Collapse Plugin.

Example :-

```
<nav class="navbar navbar-default" role="navigation">
        <div class="navbar-header">
           <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#example-navbar-collapse">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
          </button>
        </div>
        <div class="collapse navbar-collapse" id="example-navbar-collapse">
                <ul class="nav navbar-nav">
                <li class="active"><a href="#">iOS</a></li>
                <li><a href="#">SVN</a></li>
                <li class="dropdown">
                        <a href="#" class="dropdown-toggle" data-toggle="dropdown"> Java <b class="caret"></b>
                        </a>
```

```
                    <ul class="dropdown-menu">
                        <li><a href="#">jmeter</a></li>
                        <li><a href="#">EJB</a></li>
                        <li><a href="#">Jasper Report</a></li>
                        <li class="divider"></li>
                        <li><a href="#">Separated link</a></li>
                        <li class="divider"></li>
                        <li><a href="#">One more separated link</a></li>
                    </ul>
                </li>
            </ul>
        </div>
</nav>
```

Bootstrap Labels

Labels are great for offering counts, tips, or other markup for pages. Use class .label to display labels as shown in the following example:

```
<h1>Example Heading <span class="label label-default">Label</span></h1>
<h2>Example Heading <span class="label label-default">Label</span></h2>
<h3>Example Heading <span class="label label-default">Label</span></h3>
<h4>Example Heading <span class="label label-default">Label</span></h4>
```

You can the appearance of the labels using the modifier classes label-default, label-primary, label-success, label-info, label-warning, label-danger as shown in the following example:

```
<span class="label label-default">Default Label</span>
<span class="label label-primary">Primary Label</span>
<span class="label label-success">Success Label</span>
<span class="label label-info">Info Label</span>
<span class="label label-warning">Warning Label</span>
<span class="label label-danger">Danger Label</span>
```

Bootstrap Badges

Badges are similar to labels; the primary difference is that the corners are more rounded.

Badges are mainly used to highlight new or unread items. To use badges just add to links, Bootstrap navs, and more.

Example :-

```
<a href="#">Mailbox <span class="badge">50</span></a>
```

When there are no new or unread items, badges will simply collapse via CSS's :empty selector provided no content exists within.

Active nav states

You can place badges in active states of pill and list navigations. You can achieve this by placing to active links, as demonstrated in the following example:

```
<h4>Example for Active State in Pill </h4>
<ul class="nav nav-pills">
        <li class="active"><a href="#">Home <span class="badge">42</span></a></li>
        <li><a href="#">Profile</a></li>
        <li><a href="#">Messages <span class="badge">3</span></a></li>
</ul>
<br>
<h4>Example for Active State in navigations</h4>
<ul class="nav nav-pills nav-stacked" style="max-width: 260px;">
        <li class="active">
                <a href="#">
                <span class="badge pull-right">42</span> Home
           </a>
        </li>
        <li><a href="#">Profile</a></li>
        <li>
                <a href="#">
                <span class="badge pull-right">3</span> Messages
           </a>
        </li>
</ul>
```

Bootstrap Alerts

Alerts provide a way to style messages to the user. They provide contextual feedback messages for typical user actions.

You can add an optional close icon to alert. For inline dismissal use the Alerts jQuery plugin.
You can add an basic alert by creating a wrapper and adding a class of .alert and one of the four contextual classes (e.g., .alert-success, .alert-info, .alert-warning, .alert-danger). The following example demonstrates this:

```
<div class="alert alert-success">Success! Well done its submitted.</div>
<div class="alert alert-info">Info! take this info.</div>
<div class="alert alert-warning">Warning ! Dont submit this.</div>
<div class="alert alert-danger">Error ! Change few things.</div>
```

Dismissal Alerts
To build a dismissal alert:

- Add an basic alert by creating a wrapper and adding a class of .alert and one of the four contextual classes (e.g., .alert-success, .alert-info, .alert-warning, .alert-danger)
- Also add optional .alert-dismissable to the above class.

- Add a close button.

Example :-

```
<div class="alert alert-success alert-dismissable">
        <button type="button" class="close" data-dismiss="alert" aria-hidden="true">
        &times;
        </button>
        Success! Well done its submitted.
</div>
<div class="alert alert-info alert-dismissable">
        <button type="button" class="close" data-dismiss="alert" aria-hidden="true">
        &times;
        </button>
        Info! take this info.
</div>
<div class="alert alert-warning alert-dismissable">
        <button type="button" class="close" data-dismiss="alert" aria-hidden="true">
        &times;
        </button>
        Warning ! Dont submit this.
</div>
<div class="alert alert-danger alert-dismissable">
        <button type="button" class="close" data-dismiss="alert" aria-hidden="true">
        &times;
        </button>
        Error ! Change few things.
</div>
```

Bootstrap Progress Bars

The purpose of progress bars is to show that assets are loading, in progress, or that there is action taking place regarding elements on the page.

Progress bars use CSS3 transitions and animations to achieve some of their effects. These features are not supported in Internet Explorer 9 and below or older versions of Firefox. Opera 12 does not support animations.

Default Progress Bar

To create a basic progress bar:

- Add a with a class of .progress.
- Next, inside the above , add an empty with a class of .progress-bar.
- Add a style attribute with the width expressed as a percentage. Say for example, style="60%"; indicates that the progress bar was at 60%.

Example :-

```
<div class="progress">
```

```
        <div class="progress-bar" role="progressbar" aria-valuenow="60" aria-valuemin="0" aria-valuemax="100" style="width: 40%;">
                <span class="sr-only">40% Complete</span>
        </div>
</div>
```

Alternate Progress Bar

To create a progress bar with different styles:

- Add a with a class of .progress.
- Next, inside the above , add an empty with a class of .progress-bar and classprogress-bar- * where * could be success, info, warning, danger.
- Add a style attribute with the width expressed as a percentage. Say for example, style="60%"; indicates that the progress bar was at 60%.

Example:-

```
<div class="progress">
        <div class="progress-bar progress-bar-success" role="progressbar" aria-valuenow="60" aria-valuemin="0" aria-valuemax="100" style="width: 90%;">
                <span class="sr-only">90% Complete (Sucess)</span>
        </div>
</div>
<div class="progress">
        <div class="progress-bar progress-bar-info" role="progressbar" aria-valuenow="60" aria-valuemin="0" aria-valuemax="100" style="width: 30%;">
                <span class="sr-only">30% Complete (info)</span>
        </div>
</div>
<div class="progress">
        <div class="progress-bar progress-bar-warning" role="progressbar" aria-valuenow="60" aria-valuemin="0" aria-valuemax="100" style="width: 20%;">
                <span class="sr-only">20%Complete (warning)</span>
        </div>
</div>
<div class="progress">
        <div class="progress-bar progress-bar-danger" role="progressbar" aria-valuenow="60" aria-valuemin="0" aria-valuemax="100" style="width: 10%;">
                <span class="sr-only">10% Complete (danger)</span>
        </div>
</div>
```

Bootstrap Media Object

These are abstract object styles for building various types of components (like blog comments, Tweets, etc) that feature a left- or right-aligned image alongside textual content. The goal of the media object is to make the code for developing these blocks of information drastically shorter.

The goal of media objects (light markup, easy extendability) is achieved by applying classes to some simple markup. There are two forms to the media object:

- .media: This class allows to float a media object (images, video, audio) to the left or right of a content block.
- .media-list: If you are preparing a list where the items will be part of an unordered list, use class. Useful for comment threads or articles lists.

Let us see an example below of default media object:

```
<div class="media">
	<a class="pull-left" href="#">
		<img class="media-object" src="/bootstrap/images/64.jpg" alt="Media Object">
	</a>
	<div class="media-body">
		<h4 class="media-heading">Media heading</h4>
		This is some sample text. This is some sample text. This is some sample text. This is some sample text. This is some sample text. This is some sample text. This is some sample text. This is some sample text.
	</div>
</div>
<div class="media">
	<a class="pull-left" href="#">
		<img class="media-object" src="/bootstrap/images/64.jpg" alt="Media Object">
	</a>
	<div class="media-body">
		<h4 class="media-heading">Media heading</h4>
		This is some sample text. This is some sample text. This is some sample text. This is some sample text. This is some sample text. This is some sample text. This is some sample text. This is some sample text.
		<div class="media">
		<a class="pull-left" href="#">
			<img class="media-object" src="/bootstrap/images/64.jpg" alt="Media Object">
		</a>
		<div class="media-body">
			<h4 class="media-heading">Media heading</h4>
			This is some sample text. This is some sample text. This is some sample text. This is some sample text. This is some sample text. This is some sample text. This is some sample text. This is some sample text.
		</div>
	  </div>
	</div>
</div>
```

Bootstrap List Group

In this article we will study about list group. Purpose of list group component is to render complex and customized content in lists. To get a basic list group:

- Add the class .list-group to element <ul>.

- Add class .list-group-item to <li>.

Example :-

```
<ul class="list-group">
<li class="list-group-item">Free Domain Name Registration</li>
<li class="list-group-item">Free Window Space hosting</li>
<li class="list-group-item">Number of Images</li>
<li class="list-group-item">24*7 support</li>
<li class="list-group-item">Renewal cost per year</li>
</ul>
```