

---

# HTML, CSS, JAVASCRIPT, BOOTSTRAP

---

**HTML 4 + HTML 5 + CSS 2 + CSS 3 + JavaScript 5 +  
JavaScript 6 & 7 + Bootstrap**

Complete HTML, CSS, JS, Bootstrap - Udemy Course

<https://www.udemy.com/ui-technologies-indepth/>

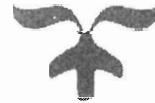
**Coupon Code: HARSHAWEB650**

Complete Angular 2.4.5.6 - Udemy Course

<https://www.udemy.com/complete-angular-indepth-easy/>

**Coupon Code: HARSHAANGULAR650**

**by Harsha Vardhan (UI Expert)**



<https://www.udemy.com/courses/search/?q=harsha>

## Contents

<b>HTML, CSS, JavaScript, Bootstrap .....</b>	<b>16</b>
<b>Fundamentals .....</b>	<b>16</b>
Application .....	16
Client .....	16
Browser .....	16
Web application .....	17
Http .....	17
<b>HTML 4 &amp; 5 .....</b>	<b>18</b>
<b>    HTML Basics .....</b>	<b>18</b>
Introduction to HTML .....	18
HTML Versions .....	18
Tag .....	18
Syntax of HTML Program .....	19
Steps to Prepare First Example in HTML .....	19
1. Installing Visual Studio Code .....	19
2. Create HTML Program .....	25
3. Execute the HTML Program .....	28
<b>    Understanding HTML Syntax .....</b>	<b>29</b>
<html> .....	29
<head> .....	30
<body> .....	30
<title> .....	30
Attributes .....	30
DOCTYPE .....	31
<b>    Basic Tags in HTML .....</b>	<b>31</b>
Headings .....	31
Paragraphs .....	33
Line Breaks .....	33
<b>    Text Formatting Tags .....</b>	<b>34</b>
Bold .....	34
Italic .....	34
Underline .....	35
Strikeout .....	35

---

Superscript .....	36
Subscript .....	36
<b>Images.....</b>	<b>37</b>
Images .....	37
<b>Hyperlinks.....</b>	<b>38</b>
Hyperlinks.....	38
<b>List tags .....</b>	<b>42</b>
Unordered List .....	42
Ordered List .....	43
Definition List .....	44
<b>Tables.....</b>	<b>45</b>
Table Tags.....	45
<b>Miscellaneous Tags.....</b>	<b>48</b>
IFrame .....	48
HTML Entities.....	50
Meta .....	51
<b>Forms .....</b>	<b>52</b>
Form .....	52
Input Tag .....	53
TextBox.....	54
Password TextBox.....	54
CheckBox.....	55
Checked Attribute .....	55
RadioButton .....	55
File Browse Button.....	56
Reset Button .....	57
Submit Button.....	57
Name Attribute.....	58
Login Form.....	58
Registration Form.....	58
Post Submission .....	59
Image Submit Button .....	60
General Button.....	61
Hidden Field .....	61
Color .....	61
Date .....	62
Time .....	62

Datetime-Local.....	63
Month.....	63
Week.....	63
Search.....	64
Number .....	64
Range.....	65
Email.....	65
Url .....	66
<b>Maxlength Attribute .....</b>	<b>66</b>
<b>Value Attribute .....</b>	<b>66</b>
<b>Readonly Attribute.....</b>	<b>67</b>
<b>Disabled Attribute .....</b>	<b>67</b>
<b>Tabindex Attribute .....</b>	<b>68</b>
<b>ID Attribute .....</b>	<b>68</b>
<b>Placeholder Attribute .....</b>	<b>69</b>
<b>Autofocus Attribute .....</b>	<b>69</b>
<b>Required Attribute .....</b>	<b>70</b>
<b>Pattern Attribute.....</b>	<b>70</b>
<b>Min and Max Attributes .....</b>	<b>71</b>
<b>Step Attribute .....</b>	<b>71</b>
<b>Novalidate Attribute.....</b>	<b>72</b>
<b>Multiple Attribute .....</b>	<b>72</b>
<b>AutoComplete Attribute .....</b>	<b>73</b>
<b>Button Tag .....</b>	<b>73</b>
<b>Fieldset .....</b>	<b>74</b>
<b>Legend.....</b>	<b>75</b>
<b>Label .....</b>	<b>76</b>
<b>DropdownList.....</b>	<b>76</b>
<b>Option Groups.....</b>	<b>77</b>
<b>ListBox .....</b>	<b>78</b>
<b>Selected Attribute .....</b>	<b>79</b>
<b>Textarea.....</b>	<b>79</b>
<b>&lt;div&gt; and &lt;span&gt;.....</b>	<b>80</b>
<b>DIV .....</b>	<b>80</b>
<b>Span .....</b>	<b>80</b>
<b>Advanced Tags.....</b>	<b>81</b>
<b>Horizontal Ruler.....</b>	<b>81</b>

Audio .....	81
Video .....	82
Details and Summary.....	83
DataList.....	83
ProgressBar .....	85
Article.....	85
<b>Semantic Tags.....</b>	<b>86</b>
Header.....	86
Nav.....	86
Section .....	87
Footer.....	87
Aside.....	87
<b>Storage .....</b>	<b>88</b>
Local Storage.....	88
Session Storage.....	89
<b>Geo Location.....</b>	<b>91</b>
<b>Web Workers .....</b>	<b>92</b>
<b>Drag and Drop .....</b>	<b>94</b>
<b>Canvas.....</b>	<b>97</b>
Creating Canvas Container.....	97
Get context of canvas .....	97
strokeStyle .....	97
lineWidth .....	97
strokeRect .....	97
fillStyle .....	97
fillRect .....	98
Example on Canvas .....	98
<b>SVG.....</b>	<b>98</b>
Syntax to create SVG container .....	98
Example on SVG .....	98
<b>Deprecated Tags / Attributes .....</b>	<b>99</b>
Deprecated / Removed Tags in HTML 5.....	99
Deprecated / Removed Attributes in HTML 5 .....	100
<b>CSS 2 &amp; 3 .....</b>	<b>101</b>
Fundamentals of CSS.....	101
Introduction to CSS .....	101

---

CSS Basic Selectors.....	101
First Example on CSS.....	102
Example on ID Selector.....	102
CSS - Properties .....	102
Colors.....	103
color.....	103
background-color.....	103
Types of colors .....	104
Font Styles .....	106
font-family.....	106
font-size.....	107
font-weight.....	108
font-style .....	108
Text Styles.....	109
letter-spacing .....	109
word-spacing .....	110
line-height .....	111
text-decoration .....	112
text-transform .....	113
text-align .....	114
text-indent.....	115
Span.....	117
Background Image .....	117
background-image.....	117
background-attachment .....	118
Lists .....	119
list-style-type for <ul> .....	119
list-style-type for <ol> .....	121
list-style-image .....	124
DIV.....	124
<div> tag .....	124
"width" and "height" properties.....	125
float .....	126
clear .....	128
Box Model .....	129
Understanding Box Model.....	129
border-style.....	129

---

border-width .....	129
border-color .....	130
border - shortcut.....	131
border - sides.....	132
margin .....	133
margin - sides .....	134
margin - shortcut .....	136
padding .....	137
padding - sides .....	138
padding - shortcut .....	139
<b>Advanced CSS Properties.....</b>	<b>139</b>
"opacity" property.....	139
display .....	140
visibility.....	141
overflow .....	142
<b>Types of Style Sheets .....</b>	<b>143</b>
1. Internal Style Sheet / Embedded Style Sheet.....	143
2. External Style Sheet.....	144
3. Inline Style Sheet.....	145
<b>CSS Selectors .....</b>	<b>145</b>
Tag Selector.....	146
ID Selector .....	146
Class Selector.....	147
Compound Selector.....	148
Grouping Selector.....	148
Child Selector .....	149
Direct Child Selector .....	150
Attribute Selector .....	150
Hover Selector .....	151
Focus Selector.....	152
Universal Selector.....	152
CSS Style Precedence .....	153
<b>CSS Realtime Examples .....</b>	<b>155</b>
Table Styles .....	155
Hyperlink Styles .....	156
Menubar .....	157
Login Form.....	158

---

<b>Advanced CSS .....</b>	<b>160</b>
Border Radius.....	160
Shadow.....	161
Transitions.....	162
Transformations .....	166
1. Rotate Transformation .....	166
2. Skew Transformation .....	167
3. Scale Transformation .....	168
4. Translate Transformation.....	169
Multiple Transformations.....	169
transform-origin .....	170
Animations.....	171
<b>Static Page Template .....</b>	<b>173</b>
Example on Static Page Template.....	173
Example on Page Navigation using Static Page Template.....	175
<b>Responsive Web Design .....</b>	<b>179</b>
What is Responsive Web Design.....	179
Media Queries .....	179
View port meta tag.....	180
Example on Responsive Web Design.....	181
<b>Bootstrap .....</b>	<b>188</b>
<b>Fundamentals of Bootstrap.....</b>	<b>188</b>
Introduction to Bootstrap .....	188
Preparing Environment for Bootstrap .....	188
"container" class .....	190
"container-fluid" class.....	191
Colors.....	192
Text Colors .....	192
Background Colors .....	193
Text.....	195
Display Headings .....	195
Text Alignment .....	196
Text Styles .....	198
Lead .....	199
Visibility .....	201
Grid System.....	201
Understanding the Columns .....	201

---

---

Grid System with Responsive Web Design .....	206
Jumbotron .....	211
Jumbotron .....	211
Images.....	212
Image Shapes .....	212
Image Alignment .....	213
Image Fluid .....	214
Tables.....	215
Basic Table .....	215
Borderless Table .....	217
Bordered Table .....	219
Striped Table .....	221
Hover Table .....	223
Table Background Colors.....	225
Table Header Background Colors.....	228
Table Small .....	230
Table Responsive .....	232
Alerts .....	234
Alerts.....	234
Buttons .....	236
Button Colors .....	236
Button Outline .....	237
Button Sizes.....	239
Button Groups.....	240
Button Vertical Groups .....	241
Button DropDown .....	242
Badges .....	243
Basic Badges.....	243
Badge Colors .....	244
Pill Badges .....	245
Progress Bar.....	246
Basic Progress Bar .....	246
Progress Bar Colors .....	247
Pagination .....	249
Basic Pagination.....	249
Pagination Size.....	250
Pagination Alignment .....	251

---

---

<b>Breadcrumbs</b> .....	252
<b>Breadcrumbs</b> .....	252
<b>List Groups</b> .....	253
<b>Basic List Groups</b> .....	253
<b>List Group Colors</b> .....	254
<b>Cards</b> .....	256
<b>Basic Cards</b> .....	256
<b>Card Colors</b> .....	257
<b>Card Images</b> .....	259
<b>Card Groups</b> .....	260
<b>Tooltip</b> .....	262
<b>Tooltip</b> .....	262
<b>Popover</b> .....	263
<b>Popover</b> .....	263
<b>Collapsible</b> .....	264
<b>Basic Collapsible</b> .....	264
<b>Link Collapsible</b> .....	265
<b>Accordion</b> .....	266
<b>Forms</b> .....	268
<b>Inline Form</b> .....	268
<b>Stacked Form</b> .....	270
<b>Form Grid</b> .....	272
<b>Horizontal Form Grid</b> .....	274
<b>Input Groups</b> .....	277
<b>Form Validation</b> .....	278
<b>Navigation Menu</b> .....	281
<b>Basic Navigation Menu</b> .....	281
<b>Navigation Menu Alignment</b> .....	282
<b>Vertical Navigation Menu</b> .....	283
<b>Tabs</b> .....	285
<b>Pills</b> .....	286
<b>Tabs with DropDown</b> .....	288
<b>Navigation Bar</b> .....	289
<b>Basic Navigation Bar</b> .....	289
<b>NavBar Collapsible</b> .....	292
<b>NavBar DropDown</b> .....	293
<b>NavBar Search</b> .....	295

---

NavBar FixedTop.....	297
NavBar Sticky Top.....	299
Scrollspy.....	301
Carousel.....	303
Carousel.....	303
Modal.....	306
Modal.....	306
<b>JavaScript 5, 6 &amp; 7 .....</b>	<b>309</b>
<b>Fundamentals.....</b>	<b>309</b>
<b>Introduction to JavaScript.....</b>	<b>309</b>
<b>Syntax of JavaScript.....</b>	<b>309</b>
<b>console.log().....</b>	<b>309</b>
<b>Steps to Prepare First Example in JavaScript.....</b>	<b>310</b>
1. Installing Visual Studio Code .....	310
2. Create JavaScript Program .....	310
3. Execute the JavaScript Program .....	313
<b>Variables.....</b>	<b>314</b>
<b>Operators.....</b>	<b>315</b>
<b>Arithmetical Operators.....</b>	<b>315</b>
<b>Assignment Operators.....</b>	<b>316</b>
<b>Increment and Decrement Operators.....</b>	<b>317</b>
<b>Relational Operators.....</b>	<b>317</b>
<b>Logical Operators .....</b>	<b>318</b>
<b>Concatenation Operator.....</b>	<b>318</b>
<b>Control Statements .....</b>	<b>319</b>
<b>if.....</b>	<b>319</b>
<b>Switch-case .....</b>	<b>324</b>
<b>While .....</b>	<b>324</b>
<b>Do-While .....</b>	<b>325</b>
<b>For.....</b>	<b>326</b>
<b>Functions.....</b>	<b>327</b>
<b>Arguments and Return.....</b>	<b>329</b>
<b>Recursion .....</b>	<b>330</b>
<b>Arrays.....</b>	<b>331</b>
<b>Steps for development of arrays.....</b>	<b>331</b>
<b>Object Oriented Programming in JavaScript.....</b>	<b>333</b>

Introduction to Objects .....	333
Types of Object Oriented Programming (OOP) languages.....	334
Creating Objects .....	334
Object Literals.....	334
Constructor Function .....	335
Object.Keys .....	336
JSON.....	337
Stringify .....	337
Parse.....	338
Object Array Literal.....	339
Object Array .....	340
Prototype.....	340
Inheritance.....	341
<b>Data Types .....</b>	<b>342</b>
String .....	343
typeof.....	343
undefined vs null .....	344
== and ===.....	345
String Function.....	345
ToString Function.....	346
Number Function.....	346
ParseInt Function.....	348
ParseFloat Function .....	349
+ Unary Operator .....	350
toFixed() function .....	350
<b>String Functions .....</b>	<b>351</b>
toUpperCase() .....	351
toLowerCase() .....	352
length .....	352
charAt() .....	353
charCodeAt() .....	353
substr() .....	353
indexOf().....	354
replace() .....	355
split() .....	356
trim().....	356
concat() .....	357

<b>Date Functions</b> .....	357
new Date() .....	357
toLocaleDateString().....	358
toLocaleTimeString().....	358
getTime() .....	358
getDay() .....	359
getDate().....	359
getMonth().....	360
getFullYear().....	360
getHours().....	361
getMinutes().....	361
getSeconds().....	362
getMilliseconds() .....	362
Creating a Custom Date:.....	362
<b>Advanced</b> .....	363
Closures .....	363
Hoisting.....	364
<b>DOM</b> .....	365
<b>I) Window</b> .....	366
1. location.href.....	367
2. navigator.userAgent .....	367
3. navigator.screenX.....	367
4. navigator.screenY .....	367
5. alert().....	367
6. confirm() .....	368
7. print() .....	368
8. setTimeout().....	369
9. setInterval().....	369
10. scrollTo().....	370
11. open() .....	370
<b>II) Document</b> .....	371
1. title .....	371
2. head.....	371
3. body.....	371
4. images .....	371
5. links .....	372
6. URL .....	372

document.write() .....	372
document.getElementById() .....	373
document.getElementsByName() .....	373
document.getElementsByTagName() .....	373
document.getElementsByClassName() .....	374
document.querySelectorAll() .....	374
document.querySelector() .....	375
<b>III) Element.....</b>	<b>375</b>
<b>tagName .....</b>	<b>375</b>
<b>id .....</b>	<b>376</b>
<b>innerHTML.....</b>	<b>376</b>
<b>innerText .....</b>	<b>376</b>
<b>style .....</b>	<b>377</b>
<b>parentElement .....</b>	<b>377</b>
<b>children .....</b>	<b>378</b>
<b>scrollTop .....</b>	<b>378</b>
<b>setAttribute() .....</b>	<b>379</b>
<b>getAttribute() .....</b>	<b>379</b>
<b>removeAttribute() .....</b>	<b>380</b>
<b>attributes .....</b>	<b>380</b>
<b>hasAttribute() .....</b>	<b>381</b>
<b>focus() .....</b>	<b>381</b>
<b>click() .....</b>	<b>382</b>
<b>remove().....</b>	<b>382</b>
<b>document.createElement() .....</b>	<b>383</b>
<b>appendChild() .....</b>	<b>383</b>
<b>addEventListener() .....</b>	<b>383</b>
<b>Introduction to Events .....</b>	<b>383</b>
<b>List of Events.....</b>	<b>384</b>
<b>"Click" event.....</b>	<b>384</b>
<b>"Dblclick" event .....</b>	<b>385</b>
<b>"Mouseover" and "Mouseout" events .....</b>	<b>386</b>
<b>"Mousemove" event.....</b>	<b>387</b>
<b>"Keyup" event .....</b>	<b>388</b>
<b>"Keypress" event .....</b>	<b>389</b>
<b>"Focus" and "Blur" events .....</b>	<b>391</b>
<b>"Change" event.....</b>	<b>392</b>

"Contextmenu" event .....	395
"Cut", "Copy", "Paste" events.....	395
Login - Example.....	397
Add, Subtract, Multiply, Divide Example.....	397
Validations .....	399
Regular Expressions .....	400
Types of JavaScript.....	402
1. Internal JavaScript .....	402
2. External JavaScript.....	402
JavaScript - New Features .....	403
Introduction to JavaScript 6.....	403
Class-based OOP .....	404
Classes.....	404
Classes.....	405
Constructors.....	407
Methods .....	408
Inheritance.....	409
Method Overriding.....	410
Misc. Concepts.....	412
Set and Get Methods.....	412
Default Arguments.....	414
Arrow Functions.....	414
Let.....	415
Const .....	416
Rest (...) Operator .....	416
Destructuring .....	417
Multiline Strings .....	419
String Interpolation .....	419
Reading Elements from Array .....	420

## HTML, CSS, JavaScript, Bootstrap

### FUNDAMENTALS

#### Application

- It is a program that runs based on the operating system.
- Program is a collection of statements (instructions) to the system.
- For example, a statement instructs the computer to store a value. Another statement instructs the computer to do addition of numbers.
- **Examples:** Notepad, MS Word, Google, Facebook, Flipkart etc.

#### Client

- It is a machine or device (desktop, laptop, tablet, phone or Smart TV), which can access the data from server.
- The device which is used by the user is called as "client".

#### Browser

- It is software (tool) installed on the client, to see the output of the web page. User gives input in the browser and gets the output in the browser.
- The browser sends a request to server to get web page.
- Browser provides navigation among web pages.
- Browser executes the html, css, javascript programs and displays corresponding output to the user.
- Browsers are developed by different companies. For example, "Chrome" browser was developed by "Google" company.

#### Important browsers:

- Google Chrome
- Mozilla Firefox
- Microsoft Internet Explorer
- Microsoft Edge

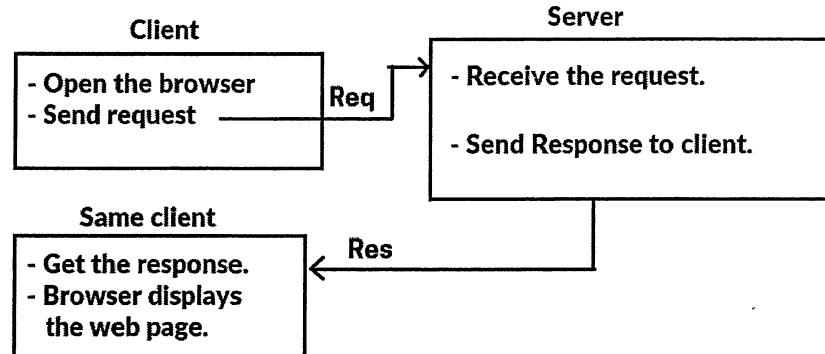
- Apple Safari
- Opera

### Web application

- It is a set of programs that are running on the browser to interact with the user.
- It is responsible to store information temporarily.
- It is responsible to interact with server i.e. sending request to server and get response from server.

### Http

- Http stands for "Hypertext Transfer Protocol".
- It is a protocol, which provides a set of rules to send request to server and get response from server.
- Http protocol defines "request format" and "response format".
- Http protocol defines HTTP status codes, Http Content Types etc.



# HTML 4 & 5

## HTML Basics

### Introduction to HTML

- HTML stands for “Hypertext Markup Language”.
- “Hypertext” means “the text that can be transferred from internet server to internet client”.
- HTML is a markup language. The markup language is a language, which syntax will be in the form of tags.
- HTML is used to design web pages. That means HTML is used to create elements (such as headings, paragraphs, icons, menus, logos, images, textboxes, buttons etc.) in the web pages.
- HTML is easy language to understand.
- HTML is “client side language”. That means the html code executes on the client (browser).
- HTML is supported by all the browsers such as Google Chrome, Mozilla Firefox, Microsoft Internet Explorer, Safari, Opera and other browsers.
- HTML is used developed by “Tim Berners-Lee” and maintained by “W3C” (World Wide Web).
- HTML is used in all real web sites today.
- The file extension should be “.html”.
- HTML is the interpreter-based language. That means the HTML code will be converted into machine language in line-by-line manner. Browser interprets HTML code.
- HTML is not a case sensitive language. That means you can write the html code in either upper case or lower case.

### HTML Versions

- HTML 1.0 : Nov 1991
- HTML 2.0 : Mar 1995
- HTML 3.0 : Jan 1997
- HTML 4.0 : Dec 1997
- HTML 5.0 : Oct 2014
- HTML 5.1 : Nov 2016
- HTML 5.2 : Dec 2017

### Tag

- A tag is a keyword, enclosed within “<” and “>” in HTML language.
- Tag is instruction / command to browser.
- Tag is used to display some specific output in the web page.

- Syntax: <tag>

### Types of tags

- Tags are two types:
  1. **Paired tags:** Contains opening tag and ending tag. The opening tag specifies starting point of the output. The closing tag specifies end point of the output. Ex: <h1> hello </h1>
  2. **Unpaired tags:** Contains single tag only (no separate ending tag). Ex: <hr>

### Syntax of HTML Program

- Every html program should have the following syntax:

```
<html>
  <head>
    Non Content here
  </head>
  <body>
    Content here
  </body>
</html>
```

- The <html> tag represents starting and ending point of the HTML program. The <html> tag contains two child tags, those are <head> and <body>
- The <head> tag represents non-content information of the web page. The information that doesn't appear in the web page is called as "non-content".
- The <body> tag represents content information of the web page. The information that appears in the web page is called as "content".

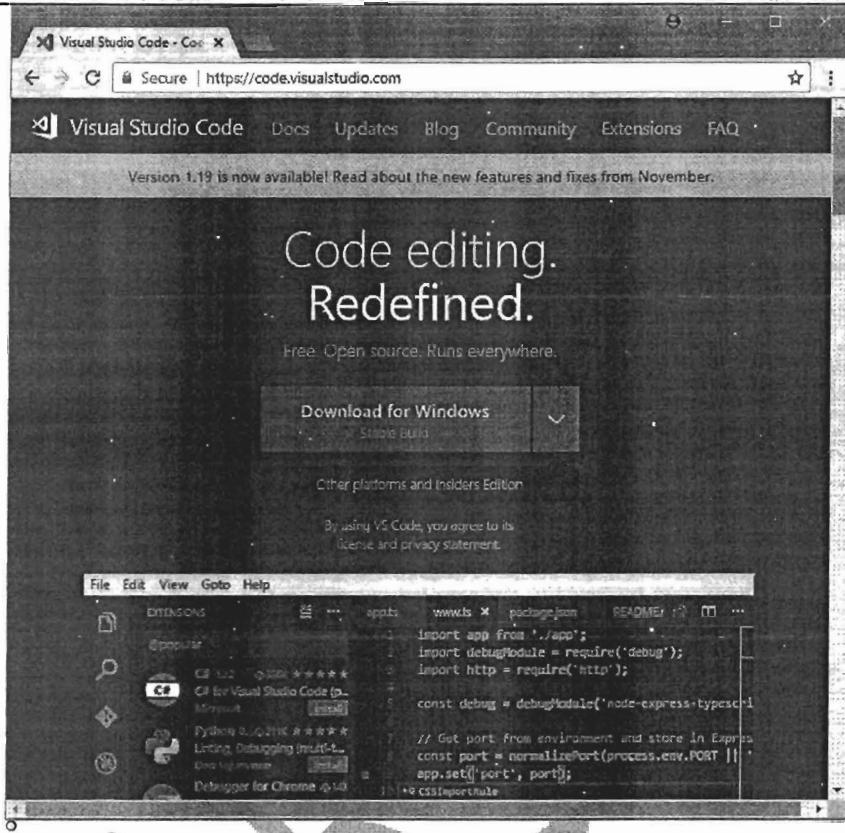
### Steps to Prepare First Example in HTML

1. Installing Visual Studio Code
2. Creating HTML Program
3. Executing HTML Program

#### 1. Installing Visual Studio Code

"Visual Studio Code" is the recommended editor for html, css, javascript and many other languages / frameworks.

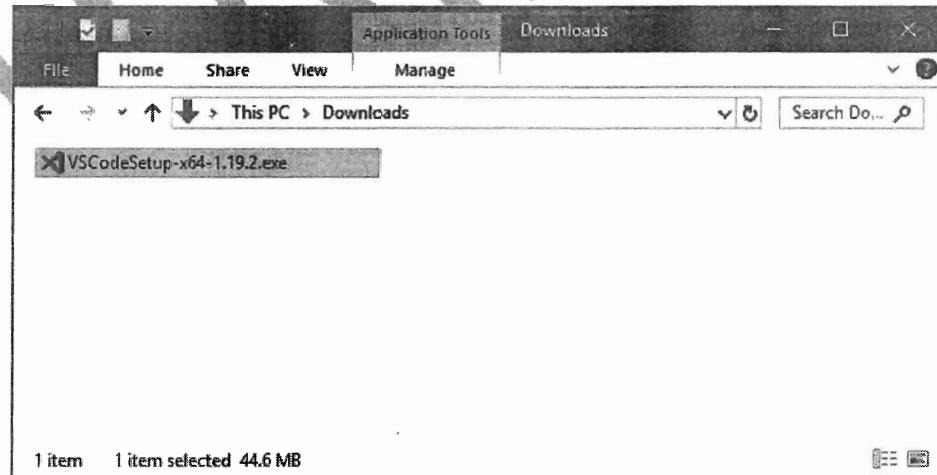
Go to <https://code.visualstudio.com>



Click on "Download for Windows".

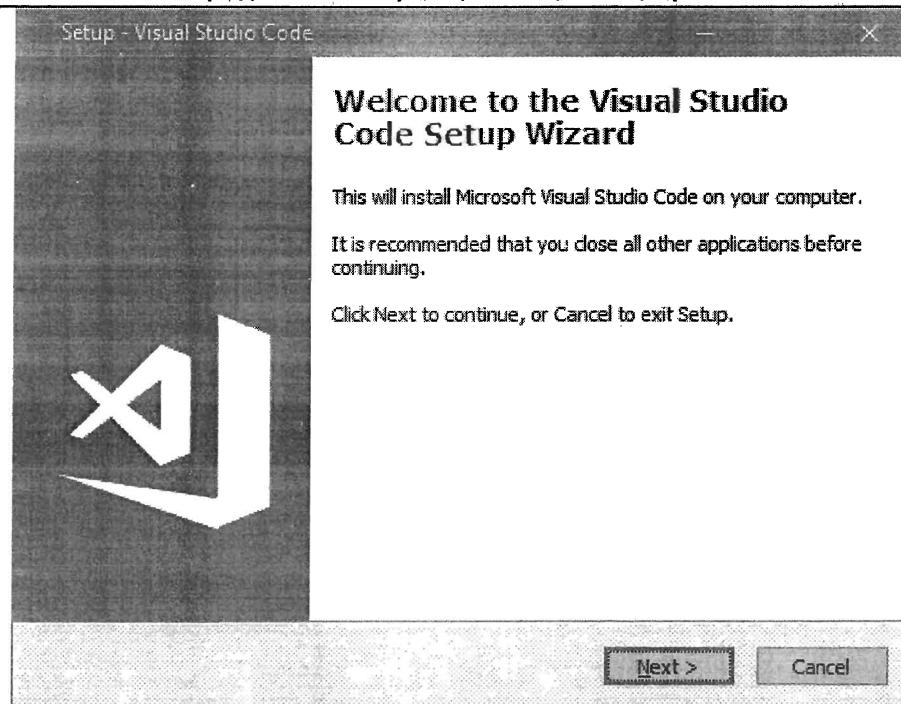
**Note:** The version number may be different at your practice time.

Go to "Downloads" folder; you can find "VSCodeSetup-x64-1.19.2.exe" file.

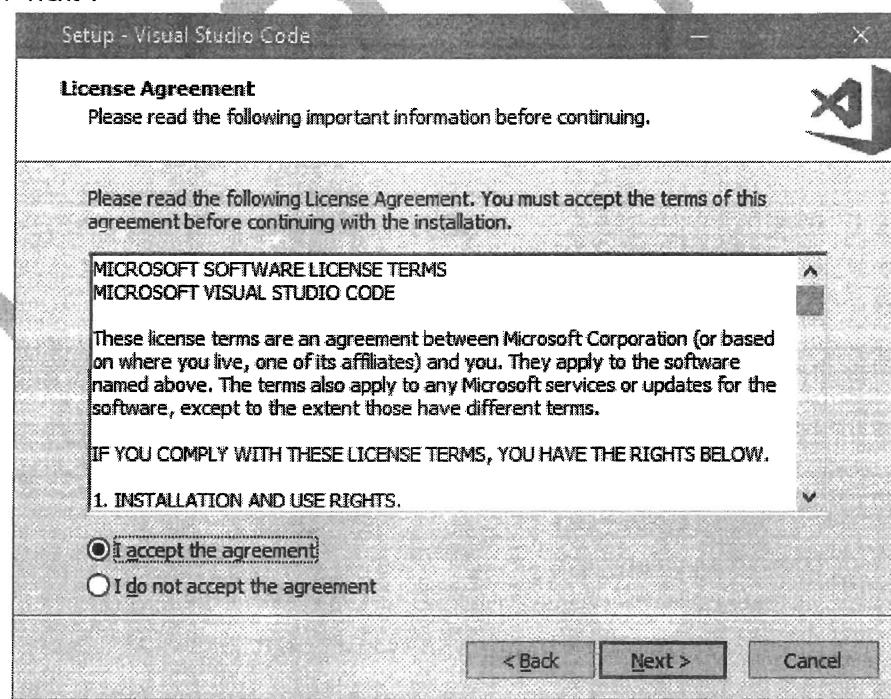


Double click on "VSCodeSetup-x64-1.19.2.exe" file.

Click on "Yes".

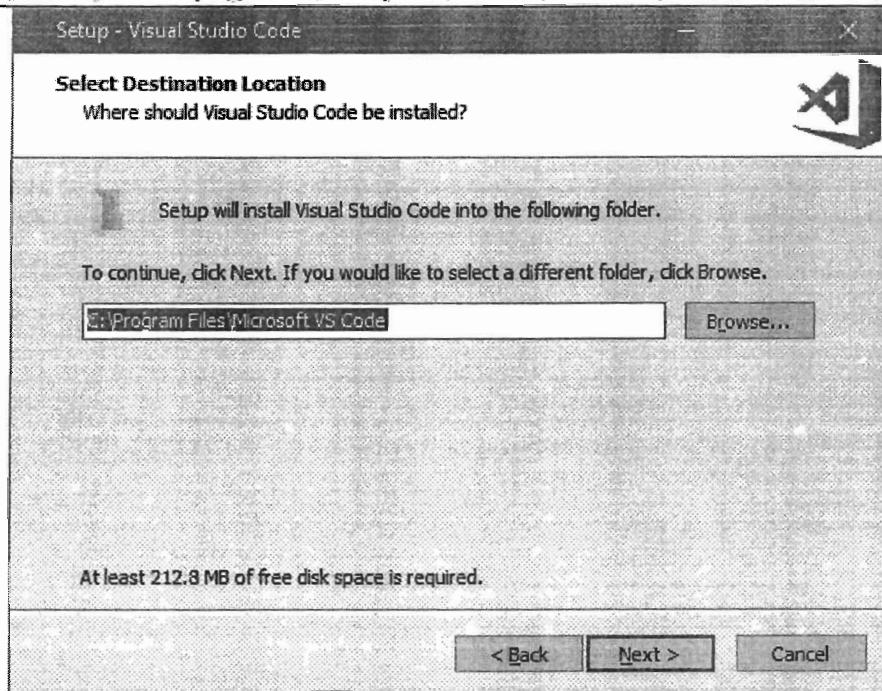


Click on “Next”.

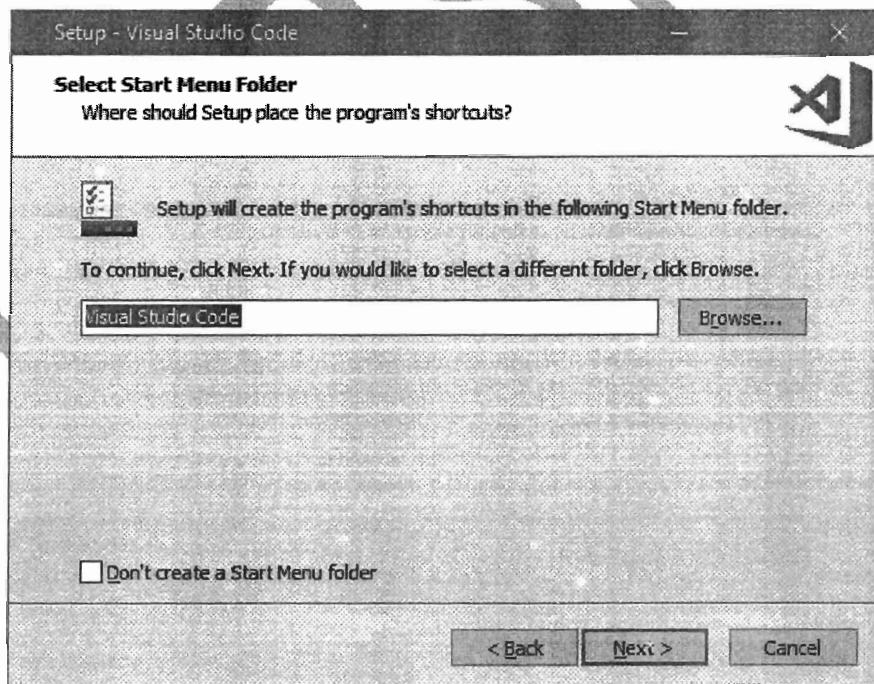


Click on “I accept the agreement”.

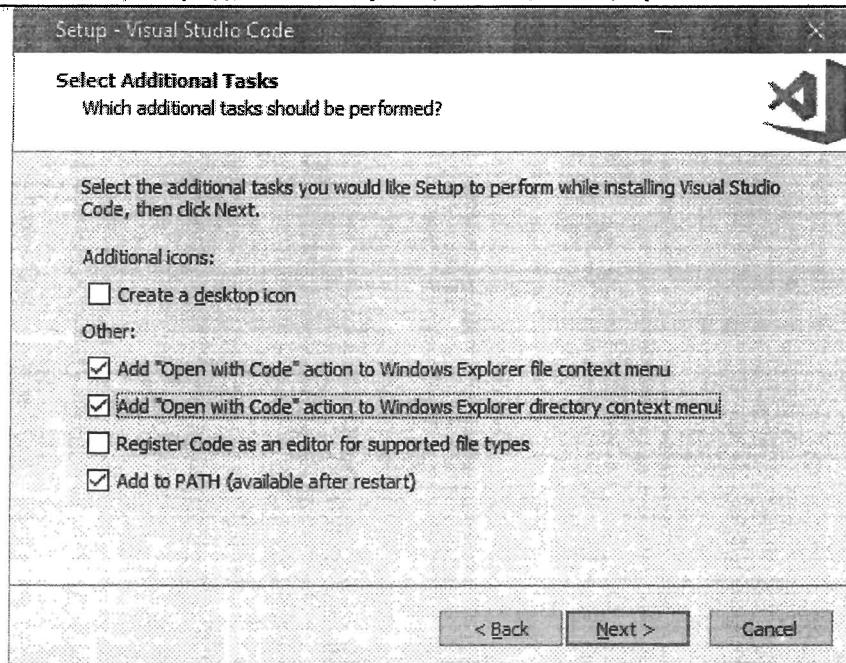
Click on “Next”.



Click on "Next".



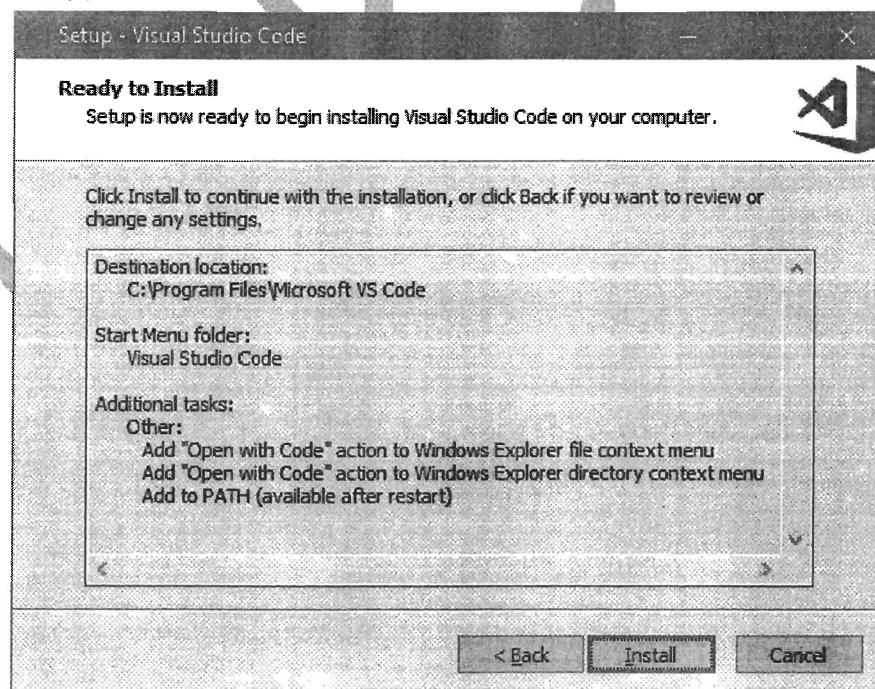
Click on "Next".



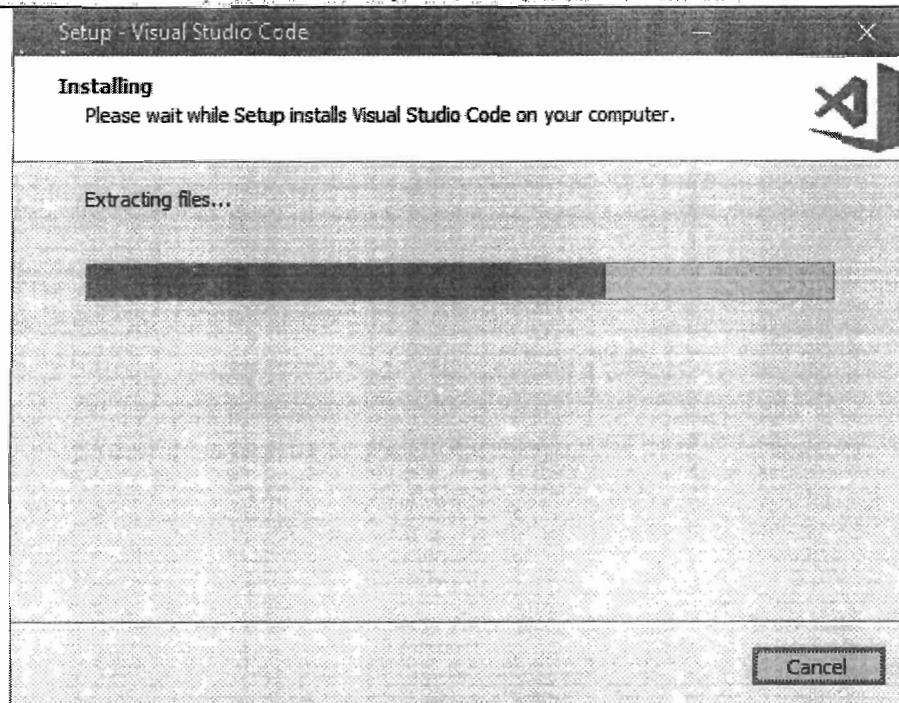
Check the checkbox "Add Open with Code action to Windows Explorer file context menu".

Check the checkbox "Add Open with Code action to Windows Explorer directory context menu".

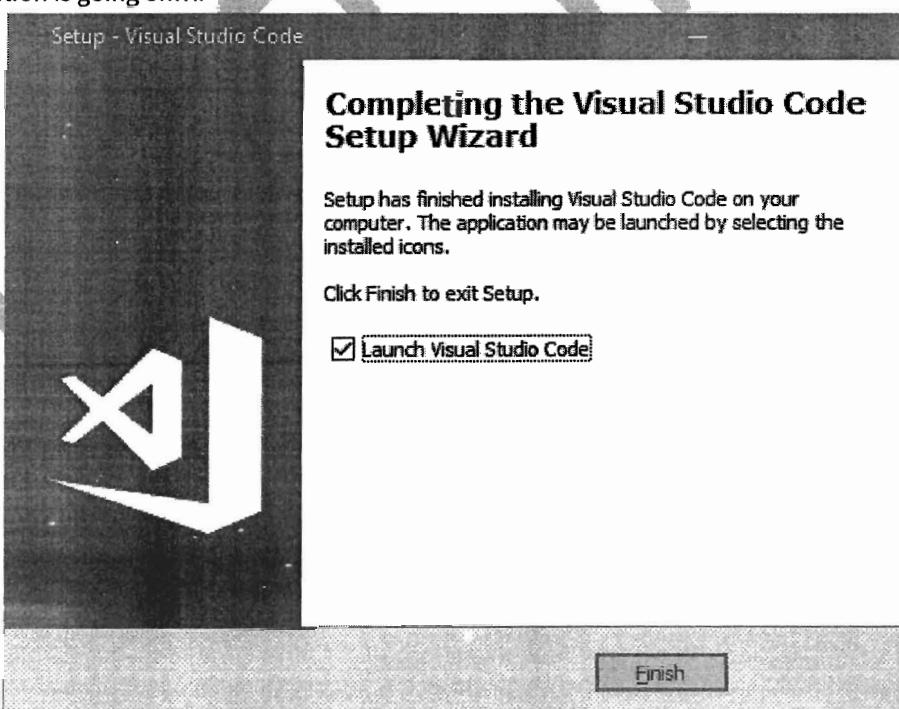
Click on "Next".



Click on "Install".



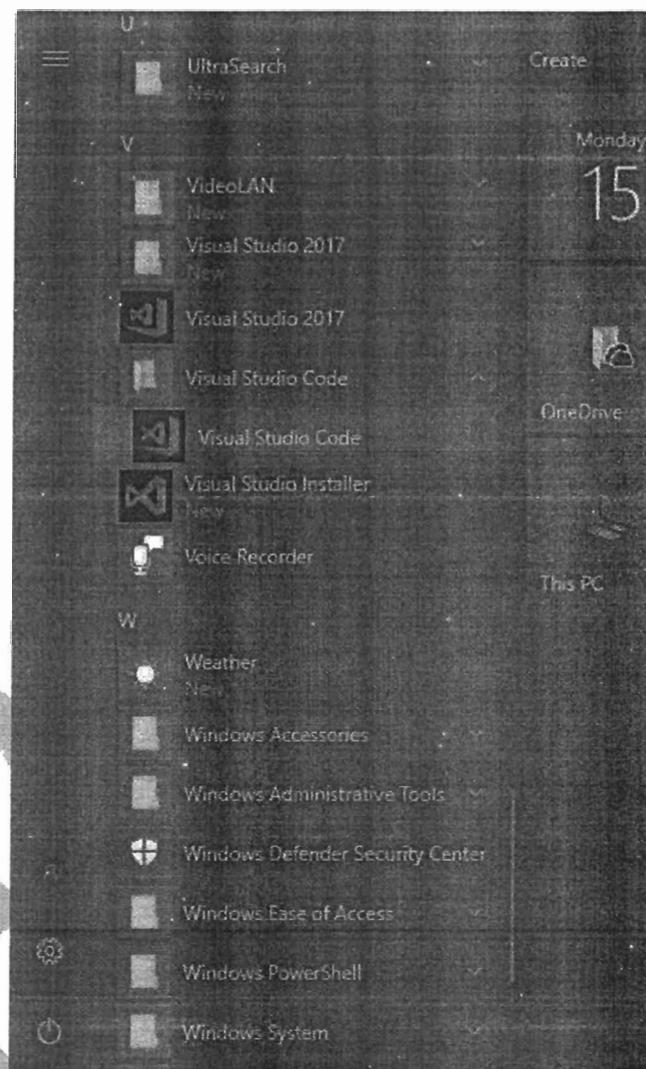
Installation is going on....



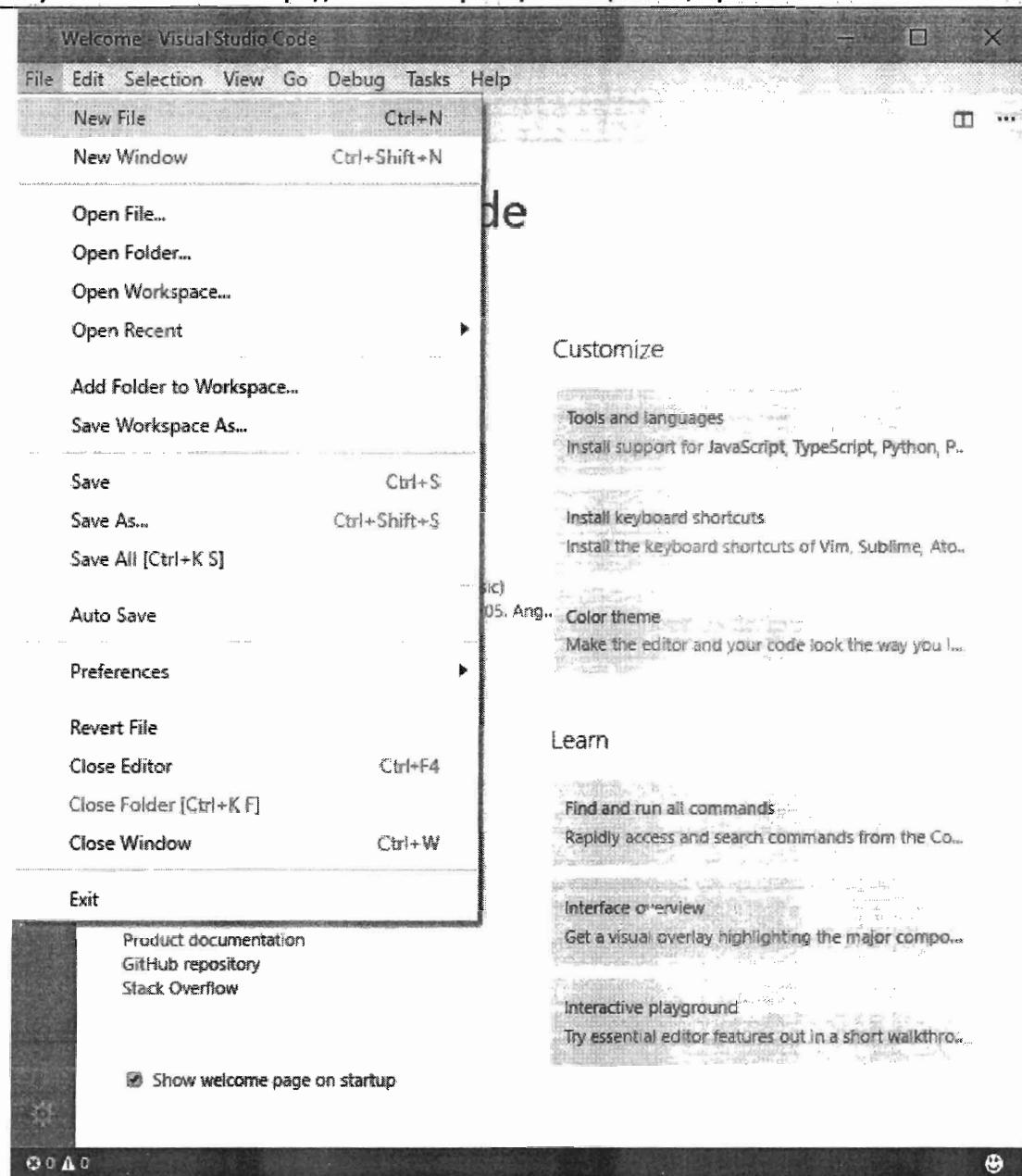
Click on "Finish".

## 2. Create HTML Program

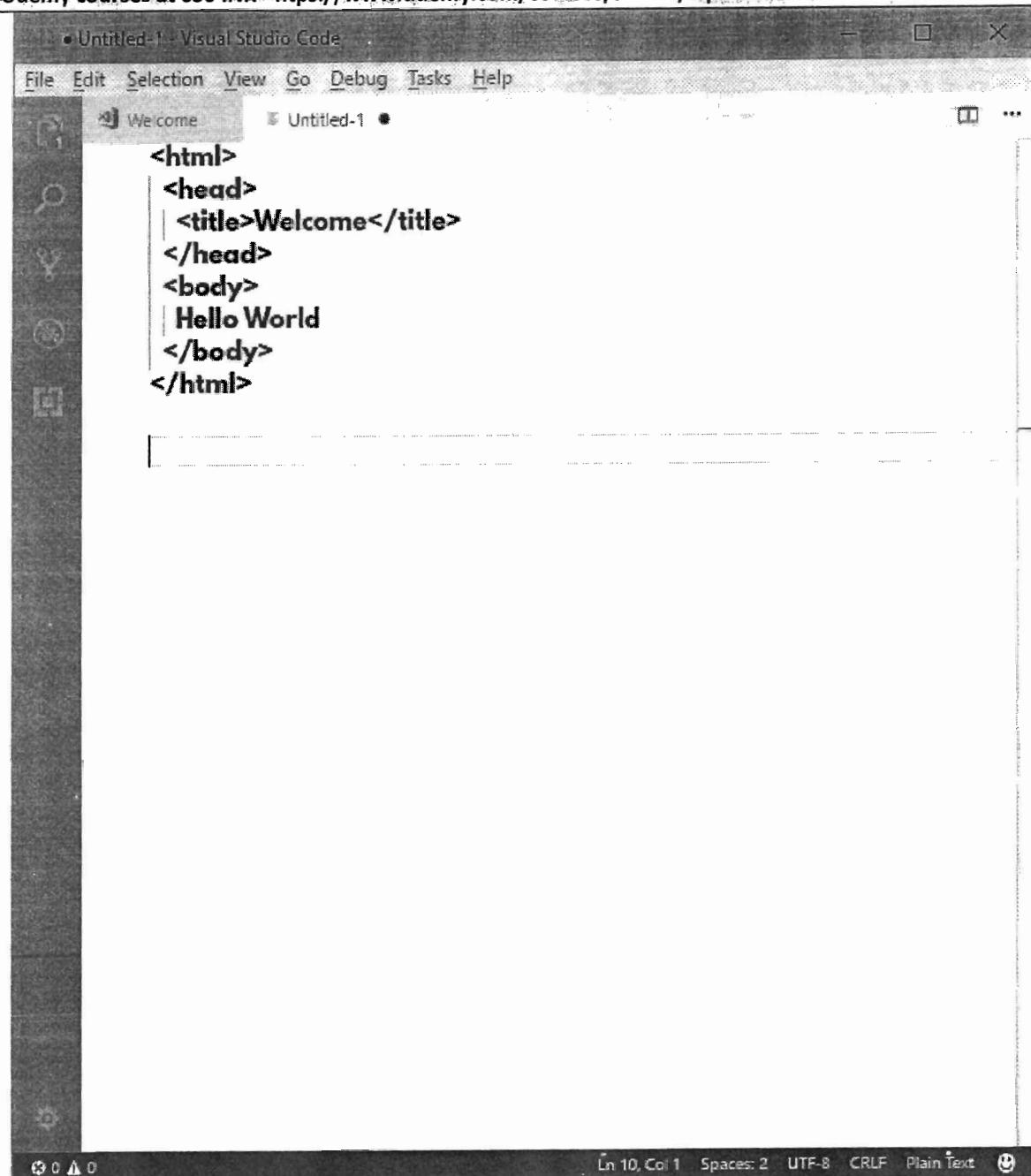
- Open “Visual Studio Code”, by clicking on “Start” – “Visual Studio Code”.



- Visual Studio Code opened.



- Go to "File" – "New File".
- Type the program as follows:



• Untitled-1 - Visual Studio Code

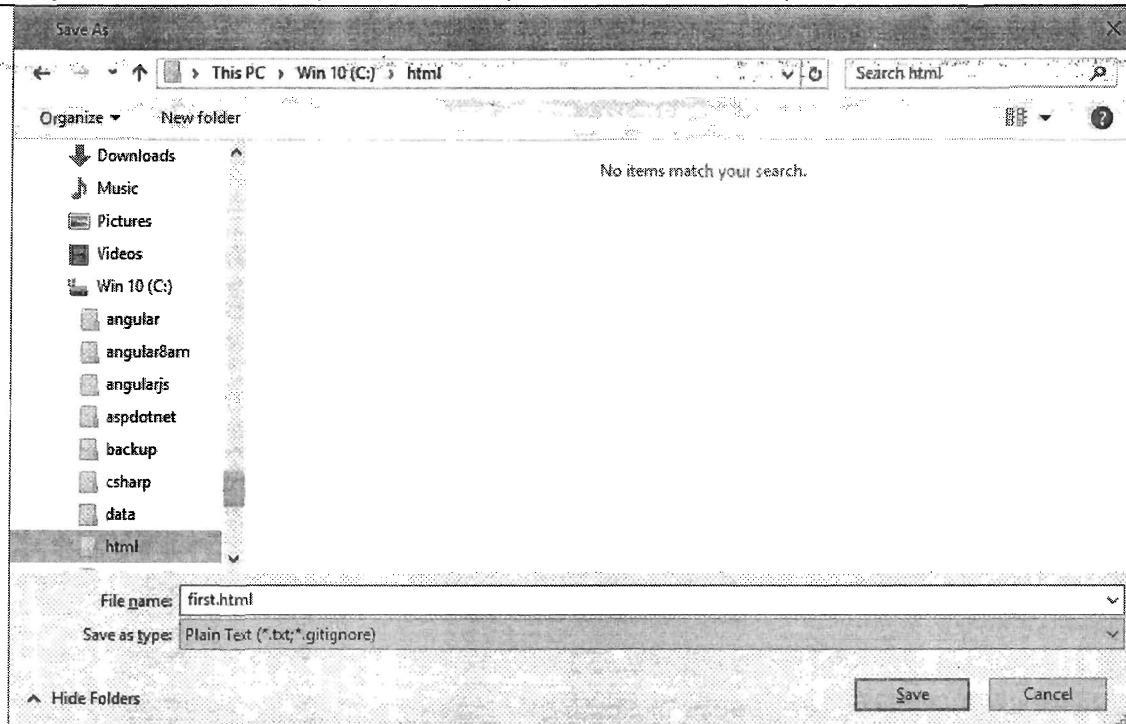
File Edit Selection View Go Debug Tasks Help

Untitled-1 •

```
<html>
<head>
| <title>Welcome</title>
</head>
<body>
| Hello World
</body>
</html>
```

Ln 10, Col 1 Spaces: 2 UTF-8 CRLF Plain Text

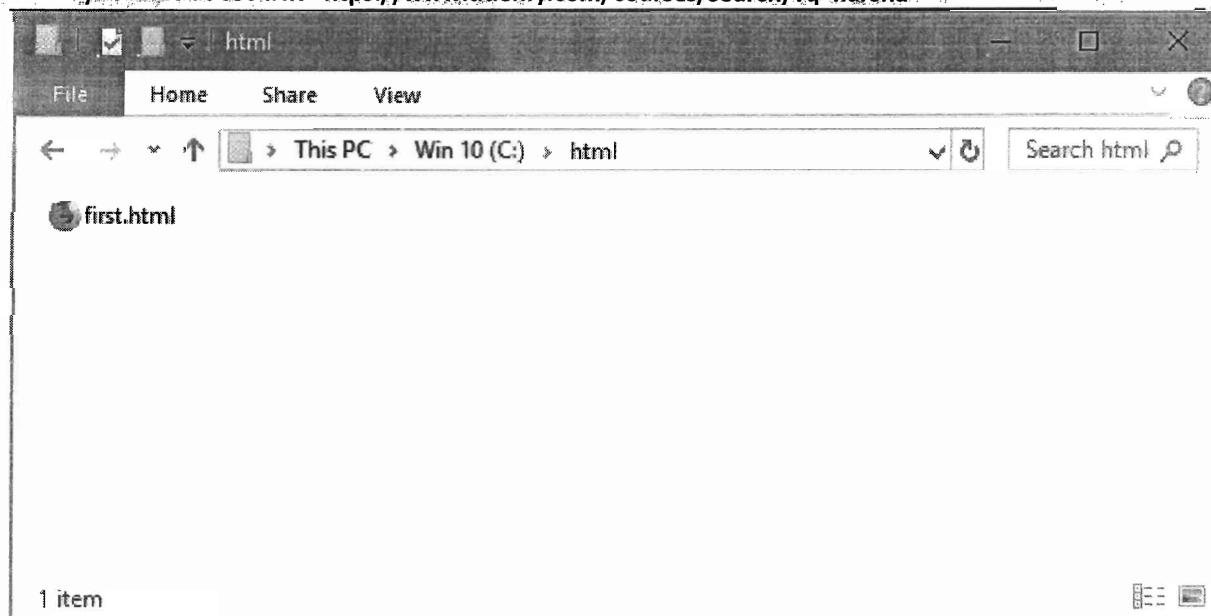
- Go to “File” menu – “Save” (or) Press Ctrl+S.



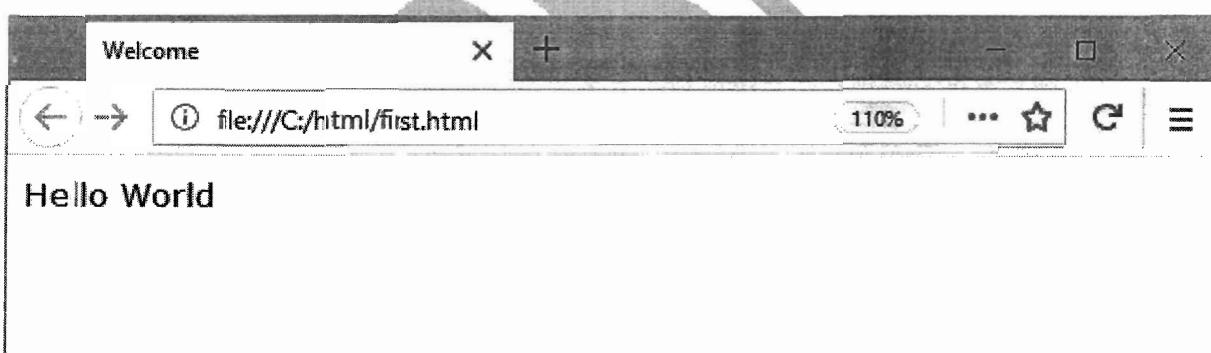
- Go to "c:\\" and click on "New folder". Enter the new folder name as "html".
- Select "c:\html" folder and enter the filename as "first.html".
- Click on "Save".
- Now the html file (c:\html\first.html) is ready.

### 3. Execute the HTML Program

- Go to "Computer" or "This PC" and go to "c:\html" folder.



- Double click on "first.html" (or) Right click on "first.html" and click on "Open With" – "Google Chrome".



**Output:** Hello World

## Understanding HTML Syntax

### **<html>**

- <html> tag represents starting and ending point of the html program. The <html> tag contains two child tags, those are <head> and <body>.

#### **Syntax:**

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

#### **Example:**

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

### **<head>**

- <head> tag represents non content information of the page. The information that doesn't appear in the web page is called as "non content".

#### **Syntax:**

```
<head> </head>
```

#### **Example:**

```
<head> </head>
```

### **<body>**

- <body> tag represents content information of the page. The information that appears inside the web page is called as "content".

#### **Syntax:**

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

#### **Example:**

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

### **<title>**

- <title> tag is used to specify the title of the web page that appears in the browser's title bar.
- <title> tag should be used in <head> tag only.
- <title> is a paired tag.

#### **Syntax:**

```
<title>Title here</title>
```

#### **Example:**

```
<title>My title</title>
```

### **Attributes**

- Attributes are the details about the tag (command).
- Every tag has its own set of attributes.
- Attribute contains a value; The value should be written inside the double quotes.

**Syntax:** <tag attribute="value"> </tag>

## DOCTYPE

- DOCTYPE is a directive in HTML, which tells the browser about the version of html that you are using in the web page.
- Various versions of html have various DOCTYPE's.

### 1. HTML 4

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  
"http://www.w3.org/TR/html4/strict.dtd">
```

### 2. XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

### 3. HTML 5

```
<!DOCTYPE html>
```

## Basic Tags in HTML

### Headings

#### **<h1>**

- It is used to create first level heading (main heading). The headings are displayed in very large size in the output. Every browser assigns a specific size for each level of heading. The size of heading is decided by the browsers, not fixed.
- It is a paired tag.

#### **Syntax:**

```
<h1>heading here</h1>
```

#### **Example:**

```
<h1>Heading 1 here</h1>
```

#### **<h2>**

- It is used to create second level heading (sub heading).
- It is a paired tag.

#### **Syntax:**

```
<h2>heading here</h2>
```

#### **Example:**

```
<h2>Heading 2 here</h2>
```

#### **<h3>**

- It is used to create third level heading (sub heading).
- It is a paired tag.

#### **Syntax:**

```
<h3>heading here</h3>
```

**Example:**

```
<h3>Heading 3 here</h3>
```

**<h4>**

- It is used to create third level heading (sub heading).
- It is a paired tag.

**Syntax:**

```
<h4>heading here</h4>
```

**Example:**

```
<h4>Heading 4 here</h4>
```

**<h5>**

- It is used to create third level heading (sub heading).
- It is a paired tag.

**Syntax:**

```
<h5>heading here</h5>
```

**Example:**

```
<h5>Heading 5 here</h5>
```

**<h6>**

- It is used to create third level heading (sub heading).
- It is a paired tag.

**Syntax:**

```
<h6>heading here</h6>
```

**Example:**

```
<h6>Heading 6 here</h6>
```

**Example on Headings:**

---

```
<html>
  <head>
    <title>Headings</title>
  </head>
  <body>
    <h1>Heading 1 here</h1>
    <h2>Heading 2 here</h2>
    <h3>Heading 3 here</h3>
    <h4>Heading 4 here</h4>
    <h5>Heading 5 here</h5>
    <h6>Heading 6 here</h6>
  </body>
</html>
```

## Paragraphs

- <p> tag is used to create a paragraph.
- Browsers display a line break, before and after each paragraph.
- Browsers display an empty line between paragraphs.
- It is a paired tag.

### Syntax:

```
<p>paragraph here</p>
```

### Example:

```
<p>Hello</p>
```

### Example on <p>

```
<html>
  <head>
    <title>Paragraphs</title>
  </head>
  <body>
    <h1>Paragraphs</h1>
    <p>Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.</p>
    <p>It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).</p>
    <p>There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable. If you are going to use a passage of Lorem Ipsum, you need to be sure there isn't anything embarrassing hidden in the middle of text. All the Lorem Ipsum generators on the Internet tend to repeat predefined chunks as necessary, making this the first true generator on the Internet. It uses a dictionary of over 200 Latin words, combined with a handful of model sentence structures, to generate Lorem Ipsum which looks reasonable. The generated Lorem Ipsum is therefore always free from repetition, injected humour, or non-characteristic words etc.</p>
  </body>
</html>
```

## Line Breaks

- <br> tag is used to create "line breaks".
- It moves the cursor to the next line.
- The content next to the <br> will be display in the next line.

- It is an unpaired tag. That means no closing tag for `<br>` tag.

**Syntax:**

```
<br>
```

**Example:**

```
<br>
```

**Example on <br>**

```
<html>
  <head>
    <title>Br</title>
  </head>
  <body>
    <h1>Br</h1>
    One<br>Two<br>Three
  </body>
</html>
```

## Text Formatting Tags

**Bold**

- `<b>` tag is used to display the text in bold.
- The text enclosed within the `<b>` tag will be display as bold.
- Use `<b>` tag to display important text that you want to highlight.
- It is a paired tag.

**Syntax:**

```
<b>bold text</b>
```

**Example:**

```
<b>Hello</b>
```

**Example of <b>**

```
<html>
  <head>
    <title>Bold</title>
  </head>
  <body>
    <h1>Bold</h1>
    This is normal text. <b>This is bold text.</b>
  </body>
</html>
```

**Italic**

- `<i>` tag is used to display the text in italic.
- The text enclosed within the `<i>` tag will be display as *italic*.

- It is a paired tag.

#### Syntax:

```
<i>italic text</i>
```

#### Example:

```
<i>Hello</i>
```

#### Example of <i>

```
<html>
  <head>
    <title>Italic</title>
  </head>
  <body>
    <h1>Italic</h1>
    This is normal text. <i>This is italic text</i>
  </body>
</html>
```

#### Underline

- <u> tag is used to display the text in underline.
- The text enclosed within the <u> tag will be underlined.
- Use <u> tag display important text.
- It is a paired tag.

#### Syntax:

```
<u>underline text</u>
```

#### Example:

```
<u>Hello</u>
```

#### Example on <u>

```
<html>
  <head>
    <title>Underline</title>
  </head>
  <body>
    <h1>Underline</h1>
    This is normal text. <u>This is underline text</u>
  </body>
</html>
```

#### Strikeout

- <strike> tag is used to display the text in strikeout.
- The <strike> tag to display un-important text.
- It is a paired tag.

**Syntax:**

```
<strike>strikeout text</strike>
```

**Example:**

```
<strike>Hello</strike>
```

**Example on <strike>**

```
<html>
  <head>
    <title>Strikeout</title>
  </head>
  <body>
    <h1>Strikeout</h1>
    This is normal text.
    <strike>This is strikeout text</strike>
  </body>
</html>
```

**Superscript**

- <sup> tag is used to display the text in superscript (The text appears a bit upper side of normal line).
- Use <sup> tag to display square or cube etc.
- It is a paired tag.

**Syntax:**

```
<sup>superscript text</sup>
```

**Example:**

```
<sup>Hello</sup>
```

**Example on <sup>**

```
<html>
  <head>
    <title>Superscript</title>
  </head>
  <body>
    <h1>Superscript</h1>
    1<sup>st</sup>
  </body>
</html>
```

**Subscript**

- <sub> tag is used to display the text in subscript (The text appears a bit bottom side of normal line).
- It is a paired tag.

**Syntax:**

```
<sub>subscript text</sub>
```

**Example:**

```
<sub>Hello</sub>
```

**Example on <sub>**

```
<html>
  <head>
    <title>Subscript</title>
  </head>
  <body>
    <h1>Subscript</h1>
    1<sub>st</sub>
  </body>
</html>
```

## Images

### Images

- <img> tag is used to display an image in the web page.
- It is strongly recommended to place the image file in the same folder, where the html file is present. For example, if the html file is present within "c:\html" folder, we have to place the image file either in the "c:\html" folder itself or in any subfolder of the "c:\html" folder.
- It is an unpaired tag.

#### Syntax:

```

```

#### Example:

```

```

#### Attributes:

##### 1. src:

- It is used to specify path of the image file. If the image file and html file both are in the same folder, no need to specify the full path of the image.

##### 2. width:

- It is used to specify width (horizontal size) of the image. It represents the value in the form of pixels. A pixel is a small "dot" (.) on the screen.

##### 3. height

- It is used to specify height (vertical size) of the image.

##### 4. title

- It is used to specify the tooltip (that appears when the user places mouse pointer on the image).

##### 5. alt

- It is used to specify the alternate text (that appears when the image is not loaded in the browser at run time). It is strongly recommended to use "alt" for every <img> tag in real-time, because, if the image is not loaded, at least the "alt" text appears to the user.

---

**Example on <img>**

```
<html>
  <head>
    <title>Img</title>
  </head>
  <body>
    <h1>Img</h1>
    
  </body>
</html>
```

**Note:** Place "img1.jpg" in the current folder.

## Hyperlinks

### Hyperlinks

- <a> tag is used to create a hyperlink. When the user clicks on the hyperlink, the specified web page or web site or file will be opened.
- The web basically contains links to other pages, so it is very common to use <a> tag in real-time.
- By default, every browser provides built-in style for each hyperlink, i.e. blue color + hand symbol for mouse cursor + underline. We can customize this style by using CSS.
- It is a paired tag.

#### Syntax:

```
<a href="target url here">link text here</a>
```

#### Example:

```
<a href="http://www.google.com">Google</a>
```

#### Attributes:

##### 1. href:

- It is used to specify the address of web page or web site that is to be opened when the user clicks on the hyperlink. It can contain address of an internet web site, local html file, local other type of file such as pdf or word document, image file etc.

##### 2. target="\_blank":

- It is used to open the target web page or web site in a separate browser tab.

---

**Example on <a>**

### Page1.html

```
<html>
  <head>
    <title>Page 1</title>
  </head>
  <body>
```

```
<h1>Page 1</h1>
<a href="page2.html">Go to page 2</a>
</body>
</html>
```

### Page2.html

```
<html>
<head>
  <title>Page 2</title>
</head>
<body>
  <h1>Page 2</h1>
  <a href="page1.html">Go to page 1</a>
</body>
</html>
```

### Example on internet links

---

```
<html>
<head>
  <title>Hyperlinks</title>
</head>
<body>
  <h1>Hyperlinks</h1>
  <a href="http://www.google.com">Google</a>
  <a href="http://www.facebook.com">Facebook</a>
  <a href="http://www.microsoft.com">Microsoft</a>
</body>
</html>
```

### Example on target

---

```
<html>
<head>
  <title>Hyperlinks</title>
</head>
<body>
  <h1>Hyperlinks</h1>
  <a href="http://www.google.com" target="_blank">Google</a>
</body>
</html>
```

### Example on file links

---

```
<html>
<head>
  <title>Hyperlinks</title>
</head>
<body>
  <h1>Hyperlinks</h1>
  <p> <a href="vodafone.jpg">Click here to open image file</a> </p>
  <p> <a href="Document1.docx">Click here to open doc file</a> </p>
  <p> <a href="Sample.pdf">Click here to open pdf file</a> </p>
```

```
<p> <a href="rain.mp3">Click here to open audio file</a> </p>
<p> <a href="trailer.mp4">Click here to open video file</a> </p>
</body>
</html>
```

**Note:** Place "vodafone.jpg", "document1.docx", "sample.pdf", "rain.mp3", "trailer.mp4" in "c:\html" folder.

### Example on <a> with <img>

```
<html>
  <head>
    <title>Hyperlinks</title>
  </head>
  <body>
    <h1>Hyperlinks</h1>
    <a href="http://www.vodafone.in">
      
    </a>
  </body>
</html>
```

**Note:** Place "vodafone.jpg" in the current folder.

### Example on <a> with internal links

```
<html>
  <head>
    <title>internal links</title>
  </head>
  <body>
    <a href="#first">India</a>
    <a href="#second">UK</a>
    <a href="#third">US</a>

    <h1 id="first">India</h1>
    <p>India, officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, the second-most populous country with over 1.2 billion people, and the most populous democracy in the world. Bounded by the Indian Ocean on the south, the Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it shares land borders with Pakistan to the west; China, Nepal, and Bhutan to the north-east; and Burma and Bangladesh to the east. In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; in addition, India's Andaman and Nicobar Islands share a maritime border with Thailand and Indonesia.</p>
    <h1 id="second">UK</h1>
    <p>The United Kingdom of Great Britain and Northern Ireland, commonly known as the United Kingdom (UK) and Britain, is a sovereign state located off the north-western coast of continental Europe. The country includes the island of Great Britain, the north-eastern part of the island of Ireland, and many smaller islands. Northern Ireland is the only part of the UK that shares a land border with another state—the Republic of Ireland. Apart from this land border, the UK is surrounded by the Atlantic Ocean in the west and north, the North Sea in the east, the English Channel in the south and the Irish Sea in the west.</p>
    <h1 id="third">US</h1>
```

```
<p>The United States of America (USA), commonly called the United States (US or U.S.) and America, is a federal constitutional republic consisting of fifty states and a federal district. The country is situated mostly in central North America, where its forty-eight contiguous states and Washington, D.C., the capital district, lie between the Pacific and Atlantic Oceans, bordered by Canada to the north and Mexico to the south. The state of Alaska is in the northwest of the continent, with Canada to the east and Russia to the west across the Bering Strait. The state of Hawaii is an archipelago in the mid-Pacific. The country also possesses several territories in the Pacific and Caribbean. At 3.79 million square miles (9.83 million km2) and with around 315 million people, the United States is the third- or fourth-largest country by total area, and the third-largest by both land area and population. It is one of the world's most ethnically diverse and multicultural nations, the product of large-scale immigration from many countries. The geography and climate of the United States is also extremely diverse and is home to a variety of species.</p>
```

```
</body>
```

```
</html>
```

### Example on <a> with page navigation

#### india.html

```
<html>
  <head>
    <title>India</title>
  </head>
  <body>
    <h1>India</h1>
    <a href="india.html">India</a>
    <a href="uk.html">UK</a>
    <a href="us.html">US</a>
    <p>India, officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, the second-most populous country with over 1.2 billion people, and the most populous democracy in the world. p>
  </body>
</html>
```

#### uk.html

```
<html>
  <head>
    <title>UK</title>
  </head>
  <body>
    <h1>UK</h1>
    <a href="india.html">India</a>
    <a href="uk.html">UK</a>
    <a href="us.html">US</a>
    <p>The United Kingdom of Great Britain and Northern Ireland, commonly known as the United Kingdom (UK) and Britain, is a sovereign state located off the north-western coast of continental Europe. p>
  </body>
</html>
```

---

### us.html

```
<html>
  <head>
    <title>US</title>
  </head>
  <body>
    <h1>US</h1>
    <a href="india.html">India</a>
    <a href="uk.html">UK</a>
    <a href="us.html">US</a>
    <p>The United States of America (USA), commonly called the United States (US or U.S.) and America, is a federal constitutional republic consisting of fifty states and a federal district.</p>
  </body>
</html>
```

## List tags

### Unordered List

- <ul> tag is used to display the list of items with bullets.
- UL stands for “Un-Ordered List”.
- Inside <ul> tag, you should place one or more <li> tags. <li> tag stands for "List Item", which represents a single item in the list.
- Use <ul> tag if you want to display any names such as person names, country names, city names etc.
- It is a paired tag.

#### Syntax:

```
<ul>
  <li>text here</li>
  <li>text here</li>
  ...
</ul>
```

#### Example:

```
<ul>
  <li>India</li>
  <li>UK</li>
  <li>US</li>
</ul>
```

#### Example on <ul>

---

```
<html>
  <head>
    <title>Unordered List</title>
```

```
</head>
<body>
  <h1>Unordered List</h1>
  <ul>
    <li>India</li>
    <li>UK</li>
    <li>US</li>
    <li>China</li>
  </ul>
</body>
</html>
```

### Ordered List

- OL stands for “Ordered List”.
- It is used to display the list of items with numbers.
- Inside
 tag, you should place one or more   - tags. The   - tag represents a single item in the list.
  - Use
 tag if you want to display names with numbers. For example, list of academic subjects (in order).
  - It is a paired tag.

#### Syntax:

```
<ol>
  <li>text here</li>
  <li>text here</li>
  ...
</ol>
```

#### Example:

```
<ol>
  <li>India</li>
  <li>UK</li>
  <li>US</li>
</ol>
```

#### Example on <ol>

```
<html>
  <head>
    <title>Ordered List</title>
  </head>
  <body>
    <h1>Ordered List</h1>
    <ol>
      <li>India</li>
      <li>UK</li>
```

```
<li>US</li>
<li>China</li>
</ol>
</body>
</html>
```

### Definition List

- DL stands for “Definition List”.
- `<dl>` tag is used to display a collection of definitions.
- Inside `<dl>` tag, you should place one or more `<dt>` and `<dd>` tags.
- It is a paired tag.
- `<dt>` and `<dd>` tags are also paired tags. DT stands for "Definition Title".
- `<dt>` is used to specify definition title.
- `<dd>` is used to specify actual definition. DD stands for "Definition Data".

#### Syntax:

```
<dl>
  <dt>title here</dt>
  <dd>definition here</dd>
  <dt>title here</dt>
  <dd>definition here</dd>
  ...
</dl>
```

#### Example:

```
<dl>
  <dt>Computer</dt>
  <dd>Computer definition here</dd>
  <dt>Car</dt>
  <dd>Car definition here</dd>
</dl>
```

#### Example on `<dl>`

```
<html>
  <head>
    <title>Definition List</title>
  </head>
  <body>
    <h1>Definition List</h1>
    <dl>
      <dt>HTML</dt>
      <dd>HTML is used to create elements in the web page.</dd>
      <dt>CSS</dt>
      <dd>CSS is used to apply styles in the web page.</dd>
      <dt>JavaScript</dt>
      <dd>JavaScript is used to create functionality in the web page.</dd>
    </dl>
  </body>
</html>
```

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

## Tables

### Table Tags

- `<table>` tag is used to display table type of data in the web page.
- A table is a collection of rows. Each row is a collection of cells.
- A table is represented as `<table>` tag; A row is represented as `<tr>`; A cell is represented as `<td>`.
- Inside the `<table>` tag, we have to use `<tr>`; Inside the `<tr>` tag, we have to use `<td>`.
- If the cell is representing the column heading, you can use `<th>` tag, instead of `<td>` tag.
- `<caption>` tag is used to specify a title for the table.
- “tr” stands for “Table row”.
- “td” stands for “Table data”.
- “th” stands for “Table header”.
- `<table>`, `<tr>`, `<th>`, `<td>` and `<caption>` tags are paired tags.

### Syntax:

```
<table>
  <tr>
    <td>data here</td>
    <td>data here</td>
    ...
  </tr>
  ...
</table>
```

### Example:

```
<table border="1">
  <tr>
    <td>One</td>
    <td>Two</td>
  </tr>
  <tr>
    <td>Three</td>
    <td>Four</td>
  </tr>
  <tr>
```

```
<td>Five</td>
<td>Six</td>
</tr>
</table>
```

#### Attributes of <table> tag:

##### 1. border

- “0”: No border
- “1”: with border

#### Attributes of <td> or <th> tag:

##### 1. rowspan

- “n”: Specifies the no. of rows to merge. The current cell occupies the space of ‘n’ no. of cells.

##### 2. colspan

- “n”: Specifies the no. of columns to merge. The current cell occupies the space of ‘n’ no. of cells.

#### Example on simple table

```
<html>
  <head>
    <title>Table - Basic Example</title>
  </head>
  <body>
    <h1>Table - Basic Example</h1>
    <table border="1">
      <tr>
        <td>One</td>
        <td>Two</td>
      </tr>
      <tr>
        <td>Three</td>
        <td>Four</td>
      </tr>
      <tr>
        <td>Five</td>
        <td>Six</td>
      </tr>
    </table>
  </body>
</html>
```

#### Realtime Example on table

```
<html>
  <head>
    <title>Table - Students</title>
  </head>
```

```
<body>
  <h1>Table - Students</h1>
  <table border="1">
    <caption>Students</caption>
    <tr>
      <th>Sl. No</th>
      <th>Name</th>
      <th>Marks</th>
    </tr>
    <tr>
      <td>1</td>
      <td>John</td>
      <td>89</td>
    </tr>
    <tr>
      <td>2</td>
      <td>Scott</td>
      <td>45</td>
    </tr>
    <tr>
      <td>3</td>
      <td>Allen</td>
      <td>64</td>
    </tr>
  </table>
</body>
</html>
```

#### Example on colspan attribute

```
<html>
  <head>
    <title>Table - Colspan</title>
  </head>
  <body>
    <h1>Table - Colspan</h1>
    <table border="1">
      <tr>
        <th colspan="2">Phone</th>
      </tr>
      <tr>
        <td>Mobile</td>
        <td>Landline</td>
      </tr>
      <tr>
        <td>9982398231</td>
        <td>22938432</td>
      </tr>
    </table>
  </body>
</html>
```

### Example on rowspan attribute:

```
<html>
  <head>
    <title>Table - Rowspan</title>
  </head>
  <body>
    <h1>Table - Rowspan</h1>
    <table border="1">
      <tr>
        <th rowspan="2">Phone</th>
        <td>Mobile</td>
        <td>9982398231</td>
      </tr>
      <tr>
        <td>Landline</td>
        <td>22938432</td>
      </tr>
    </table>
  </body>
</html>
```

## Miscellaneous Tags

### IFrame

- <iframe> tag is used to display another web page or web site within the current web page.
- Iframe stands for “inline frame”.
- <iframe> is a paired tag.

#### Syntax:

```
<iframe src="web site address here" width="n px" height="n px">
</iframe>
```

#### Example:

```
<iframe src="http://www.airtel.in" width="400px" height="300px">
</iframe>
```

#### Attributes of <iframe> tag:

1. **src**
  - “**web site path**”: Specifies the web site or web page path that is to be displayed in the iframe.
2. **width**
  - “**n px**”: Specifies the horizontal size of the iframe.
3. **height**
  - “**n px**”: Specifies the vertical size of the iframe.
4. **frameborder**
  - “**n px**”: Specifies border of the iframe.

---

### Example on <iframe>

---

```
<html>
  <head>
    <title>Iframe</title>
  </head>
  <body>
    <h1>Iframe</h1>
    <iframe src="http://www.lipsum.com" width="400px" height="300px">
    </iframe>
  </body>
</html>
```

---

### Example on Youtube <iframe>

---

```
<html>
  <head>
    <title>Iframe - Youtube</title>
  </head>
  <body>
    <h1>Iframe - Youtube</h1>
    Enjoy the video:<br>
    <iframe width="560" height="315" src="https://www.youtube.com/embed/EJmlCNGGzdo" frameborder="0" allowfullscreen></iframe>
  </body>
</html>
```

---

### Example on navigation with <iframe>

---

#### Index.html

```
<html>
  <head>
    <title>Index</title>
  </head>
  <body>
    <h1>Index</h1>
    <a href="home.html" target="myiframe">Home</a>
    <a href="about.html" target="myiframe">About</a>
    <a href="contact.html" target="myiframe">Contact</a>
    <br>
    <iframe name="myiframe" width="100%" height="400px" src="home.html"></iframe>
  </body>
</html>
```

#### home.html

```
<html>
  <head>
    <title>Home</title>
  </head>
  <body>
    <h1>Home page</h1>
```

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

### about.html

```
<html>
  <head>
    <title>About</title>
  </head>
  <body>
    <h1>About page</h1>
  </body>
</html>
```

### contact.html

```
<html>
  <head>
    <title>Contact</title>
  </head>
  <body>
    <h1>Contact Page</h1>
  </body>
</html>
```

## HTML Entities

- HTML Entities are pre-defined codes for displaying special symbols within the web page.
- HTML Entities are case sensitive. These must be used in lower case only.

Result	Description	Entity Name	Entity Number
[space]	non-breaking space	&nbsp;	&#160;
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
&	ampersand	&amp;	&#38;
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
€	euro	&euro;	&#8364;
§	section	&sect;	&#167;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;
™	trademark	&trade;	&#8482;

### Example on HTML Entities

```
<html>
  <head>
    <title>Entities</title>
  </head>
  <body>
    <h1>Entities</h1>
    <h1>HTML entities</h1>
    hai      hello<br>
    &nbsp;<br>
    hai&nbsp;&nbsp;&nbsp;hello<br>
    &reg;<br>
    &copy;<br>
    &trade;<br>
    &pound;<br>
    &#8377;<br>
    &cent;<br>
    &lt;<br>
    &gt;<br>
    &amp;<br>
  </body>
</html>
```

### Meta

- <meta> tag is used to specify meta data (additional details) about the web page.
- <meta> tag provides information about the web page. Google like search engines will display your web page in the google search results, whenever one or more keywords are matching. You must upload your web page in the internet server to really use this.
- <meta> tag is an unpaired tag.

#### Example:

```
<meta name="keywords" content="Sony, Television, Price, Hyderabad">
<meta name="description" content="Sony LED BRAVIA Prices">
<meta name="author" content="Harsha">
```

### Example on <meta> tag

```
<html>
  <head>
    <title>Meta</title>
    <meta name="keywords" content="ui technologies, full, free, material, html, css, javascript, jquery, angularjs, bootstrap">
    <meta name="description" content="This web page contains full UI Technologies material">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  </head>
  <body>
    <h1>Meta</h1>
  </body>
</html>
```

### Form

- <form> tag is used to group-up the input elements; so that we can submit the entire form to the server.
- A form is a collection of form elements such as <input>, <textarea> and <select>.
- Forms are used to collect information from the user. For example, registration form collects user details such as username, email, password, mobile number etc.
- <form> is a paired tag.

#### Syntax:

```
<form action="server page address here" method="get">  
    form elements here  
</form>
```

#### Example:

```
<form action="http://localhost/serverpage.aspx" method="get">  
</form>
```

#### Attributes of <form> tag:

1. **action:** Used to specify the server page address to which the form is to be submitted.
2. **method:**
  - **get:**
    - Displays the parameter names and values in the browser's address bar.
    - Useful for searching and retrieving the data from database.
  - **post**
    - Hides the parameter names and values in the browser's address bar and allows you to pass the data in hidden format.
    - Useful for insert, update, delete, registration and login operations.
    - The "get" and "post" can be explained in server side courses like Java, .NET, PHP, NodeJS etc.

#### Example on <form> tag

```
<html>  
    <head>  
        <title>form</title>  
    </head>  
    <body>  
        <h1>form</h1>  
        <form>  
            form elements here  
        </form>  
    </body>  
</html>
```

### Input Tag

- <input> tag is used to create a form control (form element).
- <input> tag can create form elements such as Textbox, Checkbox, Radio button, Browser button, submit button etc.
- <input> is an unpaired tag.

#### Syntax:

```
<input type="option here">
```

#### Example:

```
<input type="text">
```

#### Attributes of <input> tag:

1. **type:** text | password | checkbox | radio | file | reset | submit | image | button | hidden | color | date | time | datetime-local | month | week | week | number | range | email | url
2. maxlength
3. value
4. readonly
5. disabled
6. tabindex
7. name
8. id
9. src
10. width
11. height
12. checked
13. placeholder
14. autofocus
15. required
16. pattern
17. min
18. max
19. step

20. formaction

21. formmethod

22. formtarget

23. form

24. multiple

### TextBox

- Textbox is used to accept a string value from the user.
- Textbox can accept email, mobile etc.
- The user can enter any no. of characters in the textbox.

**Syntax:** <input type="text">

### Example on TextBox

```
<html>
  <head>
    <title>textbox</title>
  </head>
  <body>
    <h1>textbox</h1>
    <form>
      Name <input type="text"><br>
      Mobile <input type="text"><br>
      Email <input type="text">
    </form>
  </body>
</html>
```

### Password TextBox

- Password textbox is used to accept a password from the user.
- Each character in the password textbox will be displayed as "." (dot).

**Syntax:** <input type="password">

### Example on Password TextBox

```
<html>
  <head>
    <title>password</title>
  </head>
  <body>
    <h1>password</h1>
    <form>
      Password <input type="password">
    </form>
  </body>
```

```
</html>
```

### CheckBox

- Checkbox is used to display Yes/No type of option to the user.
- If the checkbox is checked, it means "yes". If the checkbox is unchecked, it means "no".

**Syntax:** `<input type="checkbox">`

### Example on CheckBox

```
<html>
<head>
  <title>checkbox</title>
</head>
<body>
  <h1>checkbox</h1>
  <form>
    <input type="checkbox">I accept license agreement
  </form>
</body>
</html>
```

### Checked Attribute

- The "checked" attribute of checkbox makes the checkbox by default checked, while opening the page. Of course, the user can uncheck it later, if required.
- This attribute has only one value, i.e. "checked".

### Example on CheckBox - Checked

```
<html>
<head>
  <title>checkbox - checked</title>
</head>
<body>
  <h1>checkbox - checked</h1>
  <form>
    <input type="checkbox" checked="checked">I accept license agreement
  </form>
</body>
</html>
```

### Radio Button

- Radio button is used to display two or more options to the user and allow the user to select any one of them. **Ex:** Gender - Male; Female
- The "name" attribute specifies common name of radio buttons. The "name" of radio buttons should be same to group-up them. Within a group of radio buttons, only one radio button can be selected by the user at-a-time.

**Syntax:** `<input type="radio">`

### Example on Radio Button

```
<html>
  <head>
    <title>radio</title>
  </head>
  <body>
    <h1>radio</h1>
    <form>
      Bank account type:
      <input type="radio" name="bankaccount">Savings account
      <input type="radio" name="bankaccount">Current account
      <input type="radio" name="bankaccount">Loan account
    </form>
  </body>
</html>
```

### Example on Radio Button - Checked

```
<html>
  <head>
    <title>radio - checked</title>
  </head>
  <body>
    <h1>radio - checked</h1>
    <form>
      Bank account type:
      <input type="radio" name="bankaccount" checked="checked">Savings account
      <input type="radio" name="bankaccount">Current account
      <input type="radio" name="bankaccount">Loan account
    </form>
  </body>
</html>
```

### File Browse Button

- Browse button is used for “attachment” option.
- For example, you can upload image files / documents in gmail or facebook .

**Syntax:** <input type="file">

### Example on File Browse Button

```
<html>
  <head>
    <title>file browse</title>
  </head>
  <body>
    <h1>file browse</h1>
    <form>
      Attachment <input type="file">
    </form>
  </body>
</html>
```

### Reset Button

- Reset button clears the values of all fields (textboxes and others) within the current form.
- The reset button must be a part of the `<form>` tag; then only it can recognize the elements that are present inside the same form.

**Syntax:** `<input type="reset">`

### Example on reset button

```
<html>
  <head>
    <title>reset</title>
  </head>
  <body>
    <h1>reset</h1>
    <form>
      Name <input type="text"><br>
      Mobile <input type="text"><br>
      Email <input type="text"><br>
      Password <input type="password"><br>
      <input type="checkbox">I accept license agreement<br>
      Gender:
      <input type="radio" name="gender">Male
      <input type="radio" name="gender">Female<br>
      Photo:
      <input type="file"><br>
      <input type="reset" value="Clear">
    </form>
  </body>
</html>
```

### Submit Button

- Submit button is used to submit the form to the server page.
- While submitting the form, all the input parameter names and their values will be sent to the server program as "query string". Ex: `?param1=value&param2=value`.
- The query string shown above contains "names" and "values". The names are specified by the developer by using "name" attribute; "value" will be entered by the user.
- The server program receives the submitted values and do some process such as storing the data into database.
- For every form, it is recommended to create a submit button. But the `<input type="submit">` creates a normal (simple submit button)

**Syntax:** `<input type="submit">`

### Example on Submit button

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

```
<title>Submit</title>
</head>
<body>
    <h1>Submit</h1>
    <form action="http://localhost/someaddress">
        Firstname <input type="text" name="firstname"><br>
        Lastname <input type="text" name="lastname"><br>
        Mobile <input type="text" name="mobile"><br>
        Email <input type="text" name="email"><br>
        Password <input type="password" name="password"><br>
        <input type="checkbox" name="license">I accept license agreement<br>
        Gender:
        <input type="radio" name="gender">Male
        <input type="radio" name="gender">Female<br>
        Photo:
        <input type="file" name="attachment"><br>
        <input type="submit">
    </form>
</body>
</html>
```

### Name Attribute

- Name attribute represents programmatic name of the input element that will be submitted to the server. Based on the “name”, we can get the value of the element in the server side program.

**Syntax:** <input type="..." name="any name">

### Login Form

- Login form contains username and password with a submit button.

### Example on Login Form

```
<html>
    <head>
        <title>login</title>
    </head>
    <body>
        <h1>Login</h1>
        <form action="http://localhost/someaddress">
            Username: <input type="text" name="username"><br>
            Password: <input type="password" name="password"><br>
            <input type="submit" value="Login">
        </form>
    </body>
</html>
```

### Registration Form

- Registration form contains username, password and other fields with a submit button.

### Example on Registration Form

```
<html>
  <head>
    <title>Registration</title>
  </head>
  <body>
    <h1>Registration</h1>
    <form action="http://localhost/someaddress">
      <table>
        <tr>
          <td>Name:</td>
          <td><input type="text" name="personname"></td>
        </tr>
        <tr>
          <td>Password:</td>
          <td><input type="password" name="password"></td>
        </tr>
        <tr>
          <td>News Letters:</td>
          <td><input type="checkbox" name="newsletters" value="yes"></td>
        </tr>
        <tr>
          <td>Gender:</td>
          <td>
            <input type="radio" name="gender" value="m">Male
            <input type="radio" name="gender" value="f">Female
          </td>
        </tr>
        <tr>
          <td>Photo:</td>
          <td><input type="file" name="photo"></td>
        </tr>
        <tr>
          <td><input type="submit" value="Register"></td>
          <td><input type="reset"></td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

### Post Submission

- Form submission is of two types: Get | POST.
- In case of "Get" submission, the parameters are visible in the browser's location bar.
- In case of "Post" submission, the parameters are sent in secret mode; and are not visible in the browser's location bar.

### Example on Post Submit Button

```
<html>
```

```
<head>
  <title>Post</title>
</head>
<body>
  <h1>Post</h1>
  <form action="http://localhost/someaddress" method="post">
    Username:
    <input type="text" name="username"><br>
    <input type="submit">
  </form>
</body>
</html>
```

### Image Submit Button

- "Image Submit button" is used to submit the form to the server page.
- When the user clicks on the image submit button
- It also submits the clicked position (X and Y co-ordinates) will be submitted to the server, along with the values of other form elements.
- The "src" attribute is used to specify the path of the image in case of `<input type="image">`. It by default, refers to the same folder.
- It is recommended to place the image file in the same folder, where the html file is present.
- The "width" attribute is used to specify the width of the image in case of `<input type="image">`. It represents the value in the form of pixels.
- The "height" attribute is used to specify the height of the image in case of `<input type="image">`. It represents the value in the form of pixels.

**Syntax:** `<input type="image" src="filename.extension">`

### Example on Image Submit Button

```
<html>
  <head>
    <title>Image</title>
  </head>
  <body>
    <h1>Image</h1>
    <form action="http://localhost/someaddress">
      Username:
      <input type="text" name="username"><br>
      <input type="image" src="ok.png" width="20px" height="20px">
    </form>
  </body>
</html>
```

**Note:** Copy and paste "ok.png" into "c:\html" folder.

### General Button

- When the user clicks on the general button, nothing happens by default, but you can call “JavaScript Click event” when the user clicks on the button.

**Syntax:** `<input type="button" value="some text">`

### Example on General Button

```
<html>
  <head>
    <title>Button</title>
  </head>
  <body>
    <h1>Button</h1>
    <input type="button" value="OK">
  </body>
</html>
```

### Hidden Field

- Hidden field will not be appear in the web page; but will be submitted to the server.
- Hidden values are useful for the programmer to send browser / client details to server.
- Use hidden field when you want to submit some fixed value to server; that you don't want to accept from the user

**Syntax:** `<input type="hidden" name="some name" value="some value">`

### Example on hidden field

```
<html>
  <head>
    <title>Hidden</title>
  </head>
  <body>
    <h1>Hidden</h1>
    <form action="http://localhost/someaddress">
      <input type="hidden" name="x" value="100">
      <input type="submit">
    </form>
  </body>
</html>
```

### Color

- Used to create a color box, where the user can select a color.
- The selected color can be applied to any element, by using JavaScript.

**Syntax:** `<input type="color">`

### Example on Color

```
<!DOCTYPE html>
```

```
<html>
<head>
  <title>color</title>
</head>
<body>
  <h1>color</h1>
  <form action="http://localhost/someaddress">
    Color: <input type="color" name="x"><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

### Date

- Used to create a date box (date picker / popup calendar), where the user can select a date.
- The browser by default provides a built-in date picker.
- It lacks of customization; for example, you can't block some dates in the calendar etc.

**Syntax:** <input type="date">

### Example on Date

```
<!DOCTYPE html>
<html>
<head>
  <title>date</title>
</head>
<body>
  <h1>date</h1>
  <form action="http://localhost/someaddress">
    Date: <input type="date" name="x"><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

### Time

- Used to create a time box, where the user can enter time (hours, minutes and seconds).

**Syntax:** <input type="time">

### Example on Time

```
<!DOCTYPE html>
<html>
<head>
  <title>time</title>
</head>
<body>
  <h1>time</h1>
  <form action="http://localhost/someaddress">
```

```
Time: <input type="time" name="x"><br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

### Datetime-Local

- Used to create a date-cum-time box, where the user can select a date and time also.

**Syntax:** <input type="datetime-local">

### Example on New Input Types - Datetime-local

```
<!DOCTYPE html>
<html>
<head>
  <title>datetime-local</title>
</head>
<body>
  <h1>datetime-local</h1>
  <form action="http://localhost/someaddress">
    Date and Time: <input type="datetime-local" name="x"><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

### Month

- Used to create a month box, where the user can select a month.

**Syntax:** <input type="month">

### Example on Month

```
<!DOCTYPE html>
<html>
<head>
  <title>Month</title>
</head>
<body>
  <h1>Month</h1>
  <form action="http://localhost/someaddress">
    Month: <input type="month" name="x"><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

### Week

- Used to create a week box, where the user can select a week.

**Syntax:** <input type="week">

### Example on Week

```
<!DOCTYPE html>
<html>
<head>
<title>week</title>
</head>
<body>
<h1>week</h1>
<form action="http://localhost/someaddress">
Week: <input type="week" name="x"><br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

### Search

- Used to create a search box, where the user can enter some search text. It also displays clear button (X) to clear the text inside the search box.

**Syntax:** <input type="search">

### Example on Search

```
<!DOCTYPE html>
<html>
<head>
<title>search</title>
</head>
<body>
<h1>search</h1>
<form action="http://localhost/someaddress">
Search: <input type="search" name="x"><br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

### Number

- Used to create a numerical textbox, where the user can enter some number.
- It either prevents typing alphabets and special symbols, or shows error message when alphabets / special symbols are entered.
- Some browsers also display increase / decrease buttons for the number textbox.

**Syntax:** <input type="number">

### Example on Number

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>number</title>
</head>
<body>
  <h1>number</h1>
  <form action="http://localhost/someaddress">
    Number: <input type="number" name="x"><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

### Range

- Used to create a slider, based on the specific range (minimum to maximum). Ex: price range, volume.

**Syntax:** <input type="range" min="minimum" max="maximum">

### Example on Range

```
<!DOCTYPE html>
<html>
  <head>
    <title>range</title>
  </head>
  <body>
    <h1>range</h1>
    <form action="http://localhost/someaddress">
      Range: <input type="range" name="x" min="0" max="5"><br>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

### Email

- Used to create an email textbox, where the user can enter a valid email address.
- It displays error message automatically, if the given email address is invalid.

**Syntax:** <input type="email">

### Example on Email

```
<!DOCTYPE html>
<html>
  <head>
    <title>email</title>
  </head>
  <body>
    <h1>email</h1>
    <form action="http://localhost/someaddress">
      Email: <input type="email" name="x"> <br>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

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

### Url

- Used to create a url textbox, where the user can enter a valid website url. Ex: <http://www.google.com>

**Syntax:** `<input type="url">`

### Example on Url

```
<!DOCTYPE html>
<html>
  <head>
    <title>url</title>
  </head>
  <body>
    <h1>url</h1>
    <form>
      URL: <input type="url" name="x"><br>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

### Maxlength Attribute

- Specifies the maximum no. of characters that can be typed in the textbox.
- By default, the user can enter unlimited no. of characters in the textbox.
- Use "Maxlength" if you want to limit to specific no. of characters.

**Syntax:** `<input type="text" maxlength="n">`

### Example on Maxlength

```
<html>
  <head>
    <title>Maxlength</title>
  </head>
  <body>
    <h1>Maxlength</h1>
    Person name (30 characters):
    <input type="text" maxlength="30">
  </body>
</html>
```

### Value Attribute

- Represents the current value of the input element.

- In case of checkbox and radio button, the "value" must be set by the developer; in other type of inputs, the user enters the value.

**Syntax:** `<input type="..." value="some value">`

### Example on Value

```
<html>
  <head>
    <title>Value</title>
  </head>
  <body>
    <h1>Value</h1>
    Bill amount:
    <input type="text" value="1000">
  </body>
</html>
```

### Readonly Attribute

- Makes the textbox as readonly; so that the user can see the value but can't type anything in the textbox.
- In the readonly textbox, the user can see, select, copy the text value.
- Cursor can be placed inside the readonly textbox.

**Syntax:** `<input type="text" value="some value" readonly="readonly">`

### Example on readonly

```
<html>
  <head>
    <title> Readonly </title>
  </head>
  <body>
    <h1> Readonly </h1>
    Bill amount (readonly):
    <input type="text" value="1000" readonly="readonly">
  </body>
</html>
```

### Disabled Attribute

- Used to disable the element. If the element (button, textbox, checkbox, radio button) is disabled, the user can't modify or touch the value of the element. The disabled element will be out of TAB sequence. That means cursor will not stop at the disabled element.

**Syntax:** `<input type="..." disabled="disabled">`

### Example on TextBox disabled

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

```
<title>TextBox Disabled</title>
</head>
<body>
    <h1>TextBox Disabled</h1>
    Bill amount (disabled):
    <input type="text" value="1000" disabled="disabled">
</body>
</html>
```

### Example on Button disabled

```
<html>
    <head>
        <title>Disabled</title>
    </head>
    <body>
        <h1>Disabled</h1>
        <form action="http://localhost/someaddress">
            <input type="submit" disabled="disabled">
        </form>
    </body>
</html>
```

### TabIndex Attribute

- Specifies tab order. It defines tab sequence.
- When the user presses TAB key on the keyboard, the cursor jumps to the next form element which is having next higher Tabindex.

**Syntax:** `<input type="text" tabindex="n">`

### Example on Tabindex

```
<html>
    <head>
        <title>TabIndex</title>
    </head>
    <body>
        <h1>TabIndex</h1>
        Name: <input type="text" tabindex="1"><br>
        Landline: <input type="text" tabindex="3">
        Mobile: <input type="text" tabindex="2">
    </body>
</html>
```

### ID Attribute

- Represents identification name of the input element that can be used in html, css, and javascript to get the element programmatically.
- ID can be used to create `<label>` tag in html, ID selector in CSS, `getElementById( )` function JavaScript etc.

**Syntax:** <input type="..." id="your id">

### Placeholder Attribute

- Used to display watermark text in the textbox.
- The watermark text explains what value should be entered in the textbox.
- The watermark text automatically disappears if any value is entered in the textbox.

**Syntax:** <input type="text" placeholder="any text">

### Example on Placeholder

```
<!DOCTYPE html>
<html>
  <head>
    <title>placeholder</title>
  </head>
  <body>
    <h1>placeholder</h1>
    <form action="http://localhost/someaddress">
      <input type="text" placeholder="First Name" name="firstname"><br>
      <input type="text" placeholder="Last Name" name="lastname"><br>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

### Autofocus Attribute

- Used to display the cursor directly in the specific textbox, when the web page opened in the browser.
- You can use the "autofocus" attribute only once in the web page.

**Syntax:** <input type="..." autofocus="autofocus">

### Example on Autofocus

```
<!DOCTYPE html>
<html>
  <head>
    <title>autofocus</title>
  </head>
  <body>
    <h1>autofocus</h1>
    <form action="http://localhost/someaddress">
      First name:<br>
      <input type="text" name="firstname" autofocus="autofocus"><br>
      Last name:<br>
      <input type="text" name="lastname"><br>
      <input type="submit">
    </form>
  </body>
</html>
```

### Required Attribute

- Used to make the field (for example, textbox) as mandatory.
- If the textbox is empty, it shows error message automatically and the form will not be submitted to server.

**Syntax:** `<input type="..." required="required">`

### Example on Required

```
<!DOCTYPE html>
<html>
  <head>
    <title>required</title>
  </head>
  <body>
    <h1>required</h1>
    <form action="http://localhost/someaddress">
      Username:
      <input type="text" name="Username" required="required" title="Please enter your name">
      <input type="submit">
    </form>
  </body>
</html>
```

### Pattern Attribute

- Used to apply a regular expression for the textbox for validation purpose.
- Regular expression represents "pattern" of the value.
- Ex: Alphabets only allowed, numbers only allowed etc.

**Syntax:** `<input type="text" pattern="regular expression here">`

### Example on Pattern

```
<!DOCTYPE html>
<html>
  <head>
    <title>pattern</title>
  </head>
  <body>
    <h1>pattern</h1>
    <form action="http://localhost/someaddress">
      Person Name:<br>
      <input type="text" pattern="^[a-zA-Z ]*$" title="Only alphabets allowed"><br>
      <input type="submit">
    </form>
  </body>
</html>
```

## Min and Max Attributes

### "min" attribute

- Specifies the minimum value that you want to accept.
- Applicable only for `<input type="number">`, `<input type="range">` and `<input type="date">`.

**Syntax:** `<input type="number | range | date" min="minimum value">`

### "max" attribute

- Specifies the maximum value that you want to accept.
- Applicable only for `<input type="number">`, `<input type="range">` and `<input type="date">`.

**Syntax:** `<input type="number | range | date" max="maximum value">`

## Example on Min and Max

```
<!DOCTYPE html>
<html>
  <head>
    <title>min and max</title>
  </head>
  <body>
    <h1>min and max</h1>
    <form action="http://localhost/someaddress">
      Quantity (between 1 and 50):<br>
      <input type="number" name="quantity" min="1" max="50">
      <br><br>
      DOB:<br>
      <input type="date" name="dateofbirth" min="1980-1-1" max="2017-12-31">
      <br><br>
      <input type="submit">
    </form>
  </body>
</html>
```

## Step Attribute

- Specifies increment / decrement value for `<input type="number">`.
- If the user clicks on "up" button in number textbox, the "step" value will be increased.
- If the user clicks on "down" button in number textbox, the "step" value will be decreased.

**Syntax:** `<input type="number" step="value here">`

## Example on Step

```
<!DOCTYPE html>
<html>
  <head>
    <title>step</title>
  </head>
```

```
<body>
  <h1>step</h1>
  <form action="http://localhost/someaddress">
    Amount: <input type="number" name="amount" step="10"><br>
    <input type="submit">
  </form>
</body>
</html>
```

### Novalidate Attribute

- HTML supports built-in validations such as required, min, max, email, date, number etc.
- Disables the built-in HTML 5 validations, such as required, min, max, email, date, number etc.
- If "novalidate" attribute is applied and the user has entered invalid values in the textboxes, and click on "Submit" button, the form will be automatically submitted to the server, without any validation.

**Syntax 1:** <form novalidate="novalidate">

**Syntax 2:** <input type="submit" formnovalidate="novalidate">

### Example on Novalidate

```
<!DOCTYPE html>
<html>
  <head>
    <title>novalidate</title>
  </head>
  <body>
    <h1>novalidate</h1>
    <form action="http://localhost/someaddress" novalidate="novalidate" method="get">
      <label for="txt1">E-mail:</label><br>
      <input type="email" name="email" id="txt1" required="required"><br>
      <label for="txt2">Age:</label><br>
      <input type="number" name="age" id="txt2" min="18" max="70"><br>
      <input type="submit">
    </form>
  </body>
</html>
```

### Multiple Attribute

- By default, the user can select only one file in <input type="file">.
- The "multiple" attribute allows the user select multiple files in the File Browse Button, created using <input type="file">.

**Syntax:** <input type="file" form="multiple">

### Example on Multiple

```
<!DOCTYPE html>
<html>
  <head>
```

```
<title>multiple</title>
</head>
<body>
  <h1>multiple</h1>
  <form action="http://localhost/someaddress">
    Select images:
    <input type="file" name="myfiles" multiple="multiple"><br>
    <input type="submit">
  </form>
</body>
</html>
```

### AutoComplete Attribute

- Browser automatically stores the history of textbox values when the form is submitted; and display the same in the textbox that has same name.
- The "autocomplete" attribute disables this feature, for security purpose.

**Syntax 1:** <input type="text" autocomplete="off">

**Syntax 2:** <form autocomplete="off"> </form>

### Example on Autocomplete

```
<!DOCTYPE html>
<html>
  <head>
    <title>autocomplete</title>
  </head>
  <body>
    <h1>autocomplete</h1>
    <form action="http://localhost/someaddress">
      First name:<br>
      <input type="text" name="firstname"><br>
      Last name:<br>
      <input type="text" name="lastname" autocomplete="off"><br>
      <input type="submit">
    </form>
  </body>
</html>
```

### Button Tag

- <button> tag is used to display a submit button with image and text.
- The <button> tag is a container, inside which you can place any content text or other html tags.
- <button> is a paired tag.

#### Syntax:

```
<button>
  
  Text here
```

```
</button>
```

**Example:**

```
<button>  
    <br>OK  
</button>
```

**Example on <button> tag**

```
<html>  
    <head>  
        <title>Button Tag</title>  
    </head>  
    <body>  
        <h1>Button Tag</h1>  
        <form action="http://localhost/someaddress">  
            Username:  
            <input type="text" name="username"><br>  
            <button><br>OK</button>  
        </form>  
    </body>  
</html>
```

**Note:** Place "tick.png" in the current folder.

**Fieldset**

- <fieldset> tag is used to display a box around a set of fields.
- <fieldset> tag is a paired tag.
- Fieldset is used to group-up the set of form elements. For example, all the personal details of the user such as Firstname, Lastname, Email, Mobile etc., can be placed inside the fieldset.
- Inside a <form> tag, we can place any no. of <fieldset> tags.

**Syntax:**

```
<fieldset>  
    Your textboxes here  
</fieldset>
```

**Example:**

```
<fieldset>  
    <input type="text"><br>  
    <input type="text">  
</fieldset>
```

**Example on Fieldset**

```
<html>  
    <head>  
        <title>Fieldset</title>
```

```
</head>
<body>
  <h1>Fieldset</h1>
  <form action="http://localhost/someaddress">
    <fieldset>
      Username: <input type="text"><br>
      Password: <input type="password"><br>
      <input type="submit" value="Submit">
    </fieldset>
  </form>
</body>
</html>
```

### Legend

- <legend> tag is used to display a title for the fieldset.
- <legend> tag is a paired tag.
- <legend> tag can be used only inside the <fieldset>.

### Syntax:

```
<fieldset>
  <legend>title here</legend>
  Your textboxes here
</fieldset>
```

### Example:

```
<fieldset>
  <legend>User details</legend>
  <input type="text"><br>
  <input type="text">
</fieldset>
```

### Example on Legend

```
<html>
  <head>
    <title>Fieldset and Legend</title>
  </head>
  <body>
    <h1>Fieldset and Legend</h1>
    <form action="http://localhost/someaddress">
      <fieldset>
        <legend>Login</legend>
        Username: <input type="text"><br>
        Password: <input type="password"><br>
        <input type="submit" value="Submit">
      </fieldset>
    </form>
  </body>
```

</html>

### Label

- <label> tag is used to create field heading.
- The label provides description for the textbox, what value should be entered in the textbox.
- When the user clicks on the label, cursor will be appeared in the associated textbox automatically.
- <label> is a paired tag.

#### Syntax:

```
<label for="id of textbox here">label text here</label>
```

#### Example:

```
<label for="txt1">Username</label>
```

#### Attributes of <label> tag:

1. **for:** Used to specify the id of the textbox that is associated with the textbox.

### Example on <label> tag

```
<html>
  <head>
    <title>Label</title>
  </head>
  <body>
    <h1>Label</h1>
    <form action="http://localhost/someaddress">
      <label for="txt1">First Name:</label>
      <input type="text" id="txt1"><br>
      <label for="txt2">Last Name:</label>
      <input type="text" id="txt2"><br>
      <input type="submit">
    </form>
  </body>
</html>
```

### DropdownList

- <select> tag is used to create a dropdownlist or listbox.
- DropDownList is used to display few options to the user and allow the user to select any one of them.
- ListBox is used to display few options to the user and allow the user to select one or more of them.
- Inside <select> tag, you should use <option> tags. Each <option> tag represents an option in the dropdownlist.
- <select>, <option> tags are paired tags.

#### Syntax:

```
<select name="name here">
  <option value="short name here">Full name here</option>
```

```
<option value="short name here">Full name here</option>
```

```
...  
</select>
```

**Example:**

```
<select name="PaymentMode">  
    <option value="Select">Select</option>  
    <option value="DC">Debit Card</option>  
    <option value="CC">Credit Card</option>  
    <option value="NB">Net Banking</option>  
</select>
```

**Attributes of <select> tag:**

1. **multiple="multiple"**: Used to convert the dropdownlist as listbox.

**Example on Dropdownlist**

```
<html>  
    <head>  
        <title>Dropdownlist</title>  
    </head>  
    <body>  
        <h1>Dropdownlist</h1>  
        <form action="http://localhost/someaddress">  
            Country:  
            <select name="country">  
                <option>Please Select</option>  
                <option>India</option>  
                <option>China</option>  
                <option>UK</option>  
                <option>USA</option>  
                <option>Japan</option>  
            </select>  
            <br>  
            <input type="submit">  
        </form>  
    </body>  
</html>
```

**Option Groups**

- <optgroup> tag is used to group-up the <option> tags inside the <select> tag.
- The <select> tag can contain many <optgroup> tags; the <optgroup> tag contains many <option> tags.
- Use <optgroup> tag, if you have too many no. of options in the dropdownlist.

**Example on Optgroup**

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

```
<title>Optgroup</title>
</head>
<body>
    <h1>Optgroup</h1>
    <form action="http://localhost/someaddress">
        Bank:
        <select name="bank">
            <optgroup label="Top Banks">
                <option value="icic">ICICI Bank</option>
                <option value="hdfc">HDFC Bank</option>
                <option value="sbi">State Bank of India</option>
            </optgroup>
            <optgroup label="Other Banks">
                <option value="axis">Axis Bank</option>
                <option value="cb">Canara Bank</option>
                <option value="boi">Bank of India</option>
                <option value="iob">Indian Overseas Bank</option>
            </optgroup>
        </select>
        <br>
        <input type="submit">
    </form>
</body>
</html>
```

### ListBox

- ListBox is also created using `<select>` tag, just like DropDownList.
- But the difference is: DropDownList allows the user to select ONLY ONE element.
- ListBox allows the user select multiple options, by using **Ctrl+Click** or **Shift+Click** on the keyboard.

### Example on ListBox

```
<html>
    <head>
        <title>Listbox</title>
    </head>
    <body>
        <h1>Listbox</h1>
        <form action="http://localhost/someaddress">
            Bank:<br>
            <select name="bank" multiple="multiple">
                <optgroup label="Top Banks">
                    <option value="icic">ICICI Bank</option>
                    <option value="hdfc">HDFC Bank</option>
                    <option value="sbi">State Bank of India</option>
                </optgroup>
                <optgroup label="Other Banks">
                    <option value="axis">Axis Bank</option>
                    <option value="cb">Canara Bank</option>
                    <option value="boi">Bank of India</option>
                </optgroup>
            </select>
        </form>
    </body>
</html>
```

```
<option value="iob">Indian Overseas Bank</option>
</optgroup>
</select>
<br>
<input type="submit">
</form>
</body>
</html>
```

### Selected Attribute

- The "selected" attribute of <option> tag must be set to the <option> tag, which must be by default submitted in the dropdownlist.
- If you don't specify "selected" attribute for any <option> tag, by default, the first <option> tag will be selected in the dropdownlist.
- The "selected" attribute has only one value, i.e. "selected".

### Example on Dropdownlist - Selected

```
<html>
  <head>
    <title>Selected</title>
  </head>
  <body>
    <h1>Selected</h1>
    <form action="http://localhost/someaddress">
      Country:
      <select name="country">
        <option>India</option>
        <option>China</option>
        <option selected="selected">UK</option>
        <option>USA</option>
        <option>Japan</option>
      </select>
      <br>
      <input type="submit">
    </form>
  </body>
</html>
```

### Textarea

- <textarea> tag is used to create a multi-line textbox.
- Ex: Comments, Description etc.
- <textarea> tag is a paired tag.
- The user can resize the textarea, at run time, in the browser.

#### Syntax:

```
<textarea name="name here" rows="no. of rows" cols="no. of columns">
</textarea>
```

**Example:**

```
<textarea name="comments" rows="5" cols="25"></textarea>
```

**Example on Textarea**

```
<html>
  <head>
    <title>Textarea</title>
  </head>
  <body>
    <h1>Textarea</h1>
    <form action="http://localhost/someaddress">
      Comments:<br>
      <textarea name="comments" rows="5" cols="25"></textarea><br>
      <input type="submit">
    </form>
  </body>
</html>
```

**<div> and <span>**

**DIV**

- <div> is a container.
- Inside <div> tag you can place any content like normal text or any other html tags.
- When you want to divide your web page as no. of parts, each part is represented as <div> tag.
- <div> is a paired tag.

**Syntax:**

```
<div>
  Your content here
</div>
```

**Example:**

```
<div>
  Hello
</div>
```

**Span**

- <span> represents a small amount of text, for which you can apply some special formatting.
- <span> tag doesn't provide any style by default, but we can apply style to the span tag, by using CSS.
- <span> is a paired tag.

**Syntax:**

```
<span>Your content here</span>
```

**Example:**

```
<span>Hello</span>
```

## Advanced Tags

### Horizontal Ruler

- `<hr>` tag is used to display a horizontal line (horizontal ruler).
- It acts as separation between two parts of the web page
- It is an unpaired tag.

**Syntax:**

```
<hr>
```

**Example:**

```
<hr>
```

### Example on `<hr>`

```
<html>
  <head>
    <title>Hr</title>
  </head>
  <body>
    <h1>Hr</h1>
    One<hr>Two
  </body>
</html>
```

### Audio

- The `<audio>` tag plays an audio file in the web page.
- We have to maintain the audio file in the following formats, because different browsers support different audio formats:
  - `.mp3` : IE, Chrome, Firefox, Safari, Opera
  - `.ogg` : Chrome, Firefox, Opera
- The `autoplay="autoplay"` attribute starts playing the audio file as soon as web page is loaded in the browser.
- The `controls="controls"` attribute displays audio player skin in the web page.

**Syntax:**

```
<audio autoplay="autoplay" controls="controls">
  <source src="filename.mp3" type="audio/mp3">
  <source src="filename.ogg" type="audio/ogg">
</audio>
```

**Note:** The browsers play the first possible file.

### Example on <audio>

```
<!DOCTYPE html>
<html>
  <head>
    <title>audio</title>
  </head>
  <body>
    <audio autoplay="autoplay" controls="controls">
      <source src="rain.mp3" type="audio/mp3">
      <source src="rain.ogg" type="audio/ogg">
    </audio>
  </body>
</html>
```

**Note:** Copy "rain.mp3" and "rain.ogg" files into current folder.

### Video

- The <video> tag plays a video file in the web page.
- We have to maintain the video file in the following formats, because different browsers support different video formats:
  - .mp4 : IE, Chrome, Firefox, Safari, Opera
  - .webm : Chrome, Firefox, Opera
  - .ogg : Chrome, Firefox, Opera
- The autoplay="autoplay" attribute starts playing the video file as soon as web page is loaded in the browser.
- The controls="controls" attribute displays video player skin in the web page.

#### Syntax:

```
<video autoplay="autoplay" controls="controls" width="pixels" height="pixels">
  <source src="filename.mp4" type="video/mp4">
  <source src="filename.ogg" type="video/ogg">
  <source src="filename.webm" type="video/webm">
</video>
```

**Note:** The browsers play the first possible file.

### Example on <video>

```
<!DOCTYPE html>
<html>
  <head>
    <title>video</title>
```

```
</head>
<body>
  <h1>video</h1>
  <video autoplay="autoplay" controls="controls" width="600px" height="350px">
    <source src="trailer.mp4" type="video/mp4">
    <source src="trailer.ogg" type="video/ogg">
    <source src="trailer.webm" type="video/webm">
  </video>
</body>
</html>
```

**Note:** Copy “trailer.mp4”, “trailer.ogg” and “trailer.webm” files into current folder.

### Details and Summary

- `<details>` and `<summary>` tags are used to allow the user expand / collapse some information, when the user clicks on the heading.

#### Syntax:

```
<details>
  <summary>heading</summary>
  content
</details>
```

### Example on `<details>` and `<summary>`

```
<!DOCTYPE html>
<html>
  <head>
    <title>details and summary</title>
  </head>
  <body>
    <h1>details and summary</h1>
    <details>
      <summary>iPhone 6 Plus</summary>
      <p>The original iPhone introduced the world to Multi-Touch, forever changing the way people experience technology.</p>
    </details>
  </body>
</html>
```

### DataList

- `<datalist>` tag is used to display search suggestions in the textbox.
- When the user types some character in the textbox, the browser automatically searches the matching options from the datalist and display them as a search suggestions below the textbox.

#### Syntax:

```
<datalist id="id here">
```

```
<option value="value 1">text</option>  
<option value="value 1">text</option>  
...  
</datalist>  
<input type="text" list="id here">
```

### Example on <datalist>

---

```
<!DOCTYPE html>  
<html>  
<head>  
<title>Datalist</title>  
</head>  
<body>  
<h1>datalist</h1>  
<form>  
  Search:  
  <input type="text" list="list1">  
  
<datalist id="list1">  
  <option value="Abu Dhabi Commercial Bank"></option>  
  <option value="Allahabad Bank"></option>  
  <option value="Andhra Bank"></option>  
  <option value="Axis Bank"></option>  
  <option value="Bank Of America"></option>  
  <option value="Bank Of Bahrain And Kuwait"></option>  
  <option value="Bank Of Baroda"></option>  
  <option value="Bank of India"></option>  
  <option value="Bank of Maharashtra"></option>  
  <option value="Canara Bank"></option>  
  <option value="Central Bank of India"></option>  
  <option value="City Bank"></option>  
  <option value="City Union Bank"></option>  
  <option value="Corporation Bank"></option>  
  <option value="Dhanlaxmi Bank"></option>  
  <option value="Federal Bank"></option>  
  <option value="HDFC Bank"></option>  
  <option value="HSBC Bank"></option>  
  <option value="ICICI Bank"></option>  
  <option value="IDBI Bank"></option>  
  <option value="Indian Bank"></option>  
  <option value="Indian Overseas Bank"></option>  
  <option value="IndusInd Bank"></option>  
  <option value="ING Vysya Bank"></option>  
  <option value="Jammu and Kashmir Bank"></option>  
  <option value="Karnataka Bank"></option>  
  <option value="Kotak Mahindra Bank"></option>  
  <option value="Lakshmi Vilas Bank"></option>  
  <option value="Nainital Bank Limited"></option>
```

```
<option value="Oman International Bank"></option>
<option value="Oriental Bank of Commerce"></option>
<option value="Punjab National Bank"></option>
<option value="Ratnakar Bank"></option>
<option value="Reserve Bank Of India"></option>
<option value="Royal Bank Of Scotland"></option>
<option value="South Indian Bank"></option>
<option value="Standard Chartered Bank"></option>
<option value="State Bank of India"></option>
<option value="Syndicate Bank"></option>
<option value="Tamilnad Mercantile Bank"></option>
<option value="UCO Bank"></option>
<option value="Union Bank of India"></option>
<option value="United Bank of India"></option>
<option value="Vijaya Bank"></option>
<option value="Yes Bank Ltd"></option>
</datalist>
</form>
</body>
</html>
```

### ProgressBar

- <progress> tag is used to display the progress of a task.
- To move progressbar dynamically, we have to use JavaScript.
- **Ex:** Downloading progress, Uploading progress.

#### Syntax:

```
<progress min="minimum" max="maximum" value="some value">
</progress>
```

### Example on <progress>

```
<!DOCTYPE html>
<html>
  <head>
    <title>progress</title>
  </head>
  <body>
    Progress:
    <progress min="0" max="100" value="80">
    </progress>
  </body>
</html>
```

### Article

- The <article> tag represents “heading” and “one or more paragraphs”.
- It is just used to group-up the headings and paragraphs, if you have too many in the web page.
- It makes no changes in the output.

**Syntax:**

```
<article>  
    Content here  
</article>
```

**Example on <article>**

```
<!DOCTYPE html>  
<html>  
    <head>  
        <title>article</title>  
    </head>  
    <body>  
        <article>  
            <h3>Some heading</h3>  
            <p>Some content here.</p>  
        </article>  
        <article>  
            <h3>another heading</h3>  
            <p>another paragraph.</p>  
        </article>  
    </body>  
</html>
```

## Semantic Tags

### Header

- This tag represents header bar, which may include website logo, search box, main links etc.
- It don't provide any styles by default; we have to apply styles manually, using CSS.

**Syntax:**

```
<header>  
    Content here  
</header>
```

### Nav

- This tag represents navigation bar, which may include top navigation menu.
- It don't provide any styles by default; we have to apply styles manually, using CSS.

**Syntax:**

```
<nav>  
    Content here  
</nav>
```

### Section

- This tag represents specific section of the page (box or container), in which you can place some content.
- It is mainly meant for representation of major parts of the page.
- It doesn't provide any styles by default; we have to apply styles manually, using CSS.

#### Syntax:

```
<section>
  Content here
</section>
```

### Footer

- This tag represents footer part of the web page, which may contain bottom information of the page such as copy rights information, links of other related sites etc.
- It doesn't provide any styles by default; we have to apply styles manually, using CSS.

#### Syntax:

```
<footer>
  Content here
</footer>
```

### Aside

- This tag represents right side bar of the web page, which may contain ads / other promotion information.
- It doesn't provide any styles by default; we have to apply styles manually, using CSS.

#### Syntax:

```
<aside>
  Content here
</aside>
```

### Example on Semantic Tags

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML 5 - Semantic Tags</title>
  </head>
  <body>
    <header>
      this is header area
    </header>
```

```
<nav>
  this is navigation area
</nav>

<section>
  this is content area
</section>

<aside>
  this is side bar area
</aside>

<footer>
  this is footer area
</footer>
</body>
</html>
```

## Storage

### Local Storage

- “Local Storage” is used to store some data in the browser memory permanently, until you manually delete the data (or) clear the browser history.
- Local storage is used to store only string data.
- Local storage can store unlimited data.
- Local storage data is visible in the browser’s developer tools (Inspect Element) option.
- If you clear the browser history, the local storage will be deleted.
- **Ex:** Google login page automatically stores your name, email and photo on successful login.

### Setting data into Local Storage

```
localStorage.property = value;
(or)
localStorage.setItem("property", "value");
```

### Getting data from Local Storage

```
localStorage.property
(or)
localStorage.getItem("property");
```

### Example on Local Storage

```
<!DOCTYPE html>
<html>
```

```
<head>
  <title>local Storage</title>
</head>
<body>
  <h1>Local Storage</h1>
  <form>
    Name:<br>
    <input type="text" id="username"><br>
    Email:<br>
    <input type="text" id="email"><br>
    <input type="button" id="button1" value="Set data into local storage">
    <input type="button" id="button2" value="Get data from local storage">
  </form>

  <script>
    document.getElementById("button1").addEventListener("click", fun1);
    function fun1()
    {
      var username = document.getElementById("username").value;
      var email = document.getElementById("email").value;
      localStorage.username = username;
      localStorage.email = email;
      document.getElementById("username").value = "";
      document.getElementById("email").value = "";
      alert("Saved");
    }

    document.getElementById("button2").addEventListener("click", fun2);
    function fun2()
    {
      var username = localStorage.username;
      var email = localStorage.email;
      document.getElementById("username").value = username;
      document.getElementById("email").value = email;
    }
  </script>
</body>
</html>
```

### Session Storage

- “Session Storage” is used to store some data in the browser memory temporarily, until the browser gets closed.
- Session storage also can store string data only, much like local storage.
- Session storage data is also visible in the browser's developer tools (Inspect Element option).

Ex: Facebook stores your email and password after login.

### Setting data into Session Storage

```
sessionStorage.property = value;
```

(or)

```
sessionStorage.setItem("property", "value");
```

### Getting data from Session Storage

```
sessionStorage.property
```

(or)

```
sessionStorage.getItem("property");
```

### Example on Session Storage

```
<!DOCTYPE html>
<html>
  <head>
    <title>Session Storage</title>
  </head>
  <body>
    <h1>Session Storage</h1>
    <form>
      Name:<br>
      <input type="text" id="username"><br>
      Email:<br>
      <input type="text" id="email"><br>
      <input type="button" id="button1" value="Set data into session storage">
      <input type="button" id="button2" value="Get data from session storage">
    </form>
    <script>
      document.getElementById("button1").addEventListener("click", fun1);
      function fun1()
      {
        var username = document.getElementById("username").value;
        var email = document.getElementById("email").value;
        sessionStorage.username = username;
        sessionStorage.email = email;
        document.getElementById("username").value = "";
        document.getElementById("email").value = "";
        alert("Saved");
      }

      document.getElementById("button2").addEventListener("click", fun2);
      function fun2()
      {
        var username = sessionStorage.username;
        var email = sessionStorage.email;
        document.getElementById("username").value = username;
        document.getElementById("email").value = email;
      }
    </script>
  </body>
</html>
```

## Geo Location

- “Geo Location” concept is used to identify the current location of the client device.
- Geo Location returns latitude, longitude values of the client device location, based on which you can display maps, driving directions, traffic, address etc.

### Steps for development of Geo Location

- **Send request to server to get geo location:**

```
navigator.geolocation.getCurrentPosition(successcallback, failurecallback, { timeout: milliseconds });
```

- **Receive the latitude, longitude values:**

```
function successcallback(event)
{
    event.coords.latitude
    event.coords.longitude
}
```

### Example on Geo Location

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML 5 - Geo Location</title>
    </head>
    <body>
        Latitude: <span id='span1'></span><br>
        Longitude: <span id="span2"></span>

        <script>
            navigator.geolocation.getCurrentPosition(fun1, fun2, { timeout: 6000 } );
            function fun1(event)
            {
                document.getElementById("span1").innerHTML = event.coords.latitude;
                document.getElementById("span2").innerHTML = event.coords.longitude;
            }
            function fun2()
            {
                alert("Error");
            }
        </script>
    </body>
</html>
```

**Example on Geo Location - with area name**

```
<html>
  <head>
    <title>GeoLocation</title>
  </head>
  <body>
    <h1>GeoLocation</h1>
    <div id="div1">div 1</div>
    <div id="div2">div 2</div>
    <div id="div3">div 3</div>

    <script src="https://maps.googleapis.com/maps/api/js">
    </script>

    <script>
      window.navigator.geolocation.getCurrentPosition(onsuccess, onerror, { timeout: 5000 });
      function onsuccess(event)
      {
        document.getElementById("div1").innerHTML = event.coords.latitude;
        document.getElementById("div2").innerHTML = event.coords.longitude;

        var geocoder = new google.maps.Geocoder();
        var latlng = new google.maps.LatLng(event.coords.latitude, event.coords.longitude);
        geocoder.geocode( {"latLng": latlng}, fun3 );
        function fun3(results)
        {
          document.getElementById("div3").innerHTML = results[0].formatted_address;
        }
      }
      function onerror()
      {
      }
    </script>
  </body>
</html>
```

**Web Workers**

- “Web Workers” concept is used to execute a javascript file in background, without effecting the actual performance of the page.
- It is mainly used to do some background process, such as uploading the file, processing some records etc.
- The specified javascript file executes asynchronously; the code present within the javascript file can't access the DOM; but it can use `postMessage()` function to pass value from the javascript to regular script of the web page; then the script present within the web page can receive the value and do some process for it.

**Steps for development of Web Workers**

- **Create a javascript file (for async execution):**

filename.js

Any js code here

postMessage(value);

**Note:** We can't access DOM in the async javascript file. We have to use postMessage() function to pass data from the javascript file to the web page.

- **Create an object for "Worker" class:**

var w = new Worker("filename.js");

The "Worker" class represents an async javascript file.

- **Receive message (data) from async javascript file:**

```
w.onmessage = functionname;  
function functionname(event)  
{  
    event.data  
}
```

**Example on Web Workers**

webworkers.html

```
<!DOCTYPE html>  
<html>  
    <head>  
        <title>Web Workers</title>  
    </head>  
    <body>  
        <div id="div1"></div>  
  
        <script>  
            var worker = new Worker("script.js");  
            worker.addEventListener("message", fun1);  
            function fun1(event)  
            {  
                document.getElementById("div1").innerHTML = event.data;  
            }  
        </script>
```

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

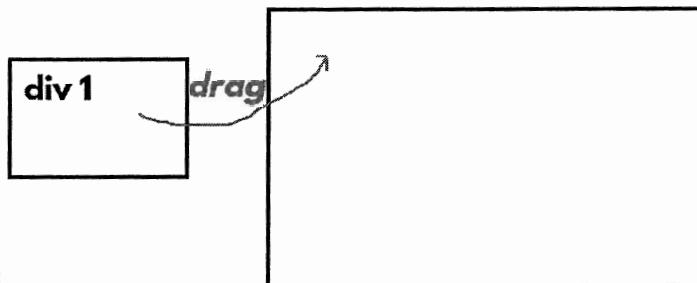
#### script.js

```
var i = 0;  
setInterval(sample, 100);  
  
function sample()  
{  
    i++;  
    postMessage(i);  
}
```

### Drag and Drop

- “Drag and Drop” concept allows the user to drag an element and drop into another element.
- We can drag any element and drop into another element.
- When we start dragging the source element, it executes "dragstart" event.
- When we hover on the droppable element, it executes "dragover" event.
- When we release the mouse pointer on the droppable element, it executes "drop" event.

#### div 2



### Steps for development of Drag and Drop

- **Create a draggable element:**

```
<div id="div1" draggable="true">  
</div>
```

- **Create a droppable element:**

```
<div id="div2">  
</div>
```

- **Handle “dragstart” event of draggable element:**

```
document.getElementById("div1").addEventListener("dragstart", fun1);
function fun1(event)
{
    event.dataTransfer.setData("variablename", event.target.id);
}
```

**Note:** The “dragstart” event executes when the user starts dragging the draggable element.

- **Handle “dragover” event of droppable element:**

```
document.getElementById("div2").addEventListener("dragover", fun2);
function fun2(event)
{
    event.preventDefault();
}
```

**Note:** The “dragover” event executes when the user drags the draggable element and places it on the top of droppable element.

- **Handle “drop” event of droppable element:**

```
document.getElementById("div2").addEventListener("drop", fun3);
function fun3(event)
{
    var id = event.dataTransfer.getData("variablename");
    document.getElementById("div2").appendChild(document.getElementById(id));
}
```

**Note:** The “drop” event executes when the user drops the draggable element on the droppable element.

### Example on Drag and Drop

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Drag and drop</title>
        <style type="text/css">
```

```
#div1
{
    width: 200px;
    height: 200px;
    padding: 10px;
    border: 3px solid red;
    float: left;
    background-color: skyblue;
    cursor: pointer;
}
#div2
{
    width: 400px;
    height: 400px;
    padding: 10px;
    border: 6px solid darkgreen;
    float: left;
    background-color: hotpink;
    position: absolute;
    left: 280px;
    top: 5px;
    cursor: pointer;
}
</style>
</head>
<body>
    <h1>Drag and drop</h1>
    <div id="div1" draggable="true">
        <h1>div 1</h1>
    </div>
    <div id="div2">
        <h1>div 2</h1>
    </div>
    <script>
        document.getElementById("div1").addEventListener("dragstart", fun1);
        function fun1(event)
        {
            event.dataTransfer.setData("myvariable", event.target.id); //we are storing "div1" into
            "myvariable"
        }

        document.getElementById("div2").addEventListener("dragover", fun2);
        function fun2(event)
        {
            event.preventDefault(); //stop the default functionality
        }

        document.getElementById("div2").addEventListener("drop", fun3);
        function fun3(event)
        {
```

```
var id = event.dataTransfer.getData("myvariable"); //we are getting "div1" from
"myvariable"
    document.getElementById("div2").appendChild(document.getElementById(id));
}
//event = browser given information
</script>
</body>
</html>
```

## Canvas

- Canvas concept is used to create graphics in the web page programmatically.
- Canvas graphics can be created using JavaScript.
- Canvas is optional & mostly not used in 95% of regular websites.
- Canvas is rare to use.

### Creating Canvas Container

```
<canvas id="canvas1" style="border:4px solid black" width="pixels" height="pixels">
</canvas>
```

### Get context of canvas

Context = It is an object, which is used to write graphics on the canvas.

```
var ctx = document.getElementById("id").getContext("2d");
```

### strokeStyle

It specifies color of the stroke (line).

```
ctx.strokeStyle = "color name";
```

### lineWidth

It specifies thickness of the stroke (line).

```
ctx.lineWidth = pixels;
```

### strokeRect

It draws a rectangle, with a line.

```
ctx.strokeRect(x, y, width, height);
```

### fillStyle

It specifies fill color.

```
ctx.fillStyle = "color name";
```

### fillRect

It draws a filled rectangle.

```
ctx.fillRect(x, y, width, height);
```

### Example on Canvas

```
<!DOCTYPE html>
<html>
  <head>
    <title>Canvas - Rectangle</title>
  </head>
  <body>
    <h1>Canvas - Rectangle</h1>
    <canvas id="canvas1" style="border: 5px solid red" width="600px" height="400px">
    </canvas>

    <script>
      var c = document.getElementById("canvas1").getContext("2d");
      c.fillStyle = "pink";
      c.fillRect(50, 50, 100, 100);
      c.strokeStyle = "blue";
      c.lineWidth = 10;
      c.strokeRect(200, 50, 200, 200);
    </script>
  </body>
</html>
```

### SVG

- SVG stands for Scalable Vector Graphics.
- Both Canvas and SVG are used to create graphics in the web page.
- Canvas losses its quality, if zoom is increased; but SVG is not.
- Canvas is “JavaScript-based”; SVG is “CSS-based”.

### Syntax to create SVG container

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1" style="border:4px solid black"
width="pixels" height="pixels">
  Graphic elements here
</svg>
```

### Example on SVG

```
<!DOCTYPE html>
<html>
  <head>
    <title>svg</title>
  </head>
```

```
<body>
  <h1>svg</h1>
  <svg xmlns="http://www.w3.org/2000/svg" version="1.1" style="border:solid 1px red" width="400px"
height="400px">
    <rect x="50" y="50" width="300" height="100" style="fill:#99ff99; stroke-width:5; stroke:blue" />
  </svg>
</body>
</html>
```

## Deprecated Tags / Attributes

### Deprecated / Removed Tags in HTML 5

- The following tags have been removed in HTML 5.
  1. **<acronym>**
    - Use **<abbr>** tag instead.
  2. **<applet>**
    - Java applets is out-dated concept.
  3. **<basefont>**
    - Use CSS to set the default font.
  4. **<big>**
    - Use CSS to increase text size.
  5. **<center>**
    - Use CSS to set center alignment.
  6. **<dir>**
    - Use **<ul>** or **<ol>** to display list of items.
  7. **<font>**
    - Use CSS to set font.
  8. **<frame>**
    - Use **<iframe>**.
  9. **<frameset>**
    - Use **<iframe>**
  10. **<noframes>**
    - Use **<iframe>**
  11. **<strike>**
    - Use CSS to set strikeout.

**12. <tt>**

- Use CSS to change font.

**13. <bgsound>**

- Use <audio> tag to play audio.

**14. <marquee>**

- Use CSS / jQuery for animations.

**15. <u>**

- Use CSS to set underline.

**Deprecated / Removed Attributes in HTML 5**

- The following attributes have been removed in HTML 5.

Sl. No	Tag	Attribute	Description
1	<body>	background, bgcolor, link, alink, vlink, text	Use CSS instead.
2	<h1> to <h6>	align	Use CSS instead.
3	<p>	align	Use CSS instead.
4	<hr>	align, noshade, size, width	Use CSS instead.
5	<input>	align	Use CSS instead.
6	<img>	align, border	Use CSS instead.
7	<table>	align, bgcolor, cellpadding, cellspacing, width	Use CSS instead.
8	<td>, <th>	align, valign, bgcolor, width, height	Use CSS instead.
9	<div>	align	Use CSS instead.

## CSS 2 & 3

### Fundamentals of CSS

#### Introduction to CSS

- CSS stands for "Cascading Style Sheet".
- CSS is a "styling language", which is used to apply styles to the html elements in the web page.
- CSS styles include with backgrounds, colors, margins, borders, paddings, alignments etc.
- A CSS style can overlap other CSS style; that's why it is called as "cascading" style sheets.

#### Syntax of CSS:

```
<style type="text/css">  
    css code here  
</style>
```

#### Syntax of CSS Style Definition:

```
selector  
{  
    property: value;  
    property: value;  
}
```

#### CSS Basic Selectors

- "Select" is a syntax to select the desired elements.
- CSS supports many selectors. The most important css selectors are:
  1. Tag Selector
  2. ID Selector

#### 1. Tag Selector

- The "tag selector" selects all the instances of specific tag.
- Use tag selector to select multiple elements.

Syntax: tagname

Example: p

#### 2. ID Selector

- The "id selector" selects a single tag, based on the "id".
- "ID" is the "identification name"; it must be unique.
- Use ID selector to select a exact single element.

- "#" is the symbol of "ID".

**Syntax:** #id

**Example:** #p1

### First Example on CSS

```
<html>
  <head>
    <title>CSS - First Example</title>
    <style type="text/css">
      p
      {
        color: green;
      }
    </style>
  </head>
  <body>
    <p>This is para</p>
    <p>This is para</p>
    <p>This is para</p>
    <p>This is para</p>
  </body>
</html>
```

### Example on ID Selector

```
<html>
  <head>
    <title>CSS - ID Selector</title>
    <style type="text/css">
      #p3
      {
        color: blue;
      }
    </style>
  </head>
  <body>
    <p>para 1</p>
    <p>para 2</p>
    <p id="p3">para 3</p>
    <p>para 4</p>
    <p>para 5</p>
  </body>
</html>
```

### CSS - Properties

- "Properties" are "details" or "settings" of html tag.
- Every property contains a value.

**Syntax:** property: value;

Example: color: green;

## Colors

### color

- This property specifies text color (font color) of the element.
- You can specify any color of your choice.

Syntax: color: colorname;

Example: color: green;

### Example of "color" property

```
<html>
  <head>
    <title>CSS - Color</title>
    <style type="text/css">
      #p1
      {
        color: red;
      }
      #p2
      {
        color: green;
      }
      #p3
      {
        color: blue;
      }
    </style>
  </head>
  <body>
    <p id="p1">para 1. You can find more information about how Google uses and stores content in
    the privacy policy or additional terms for particular Services.</p>
    <p id="p2">para 2. You can find more information about how Google uses and stores content in
    the privacy policy or additional terms for particular Services.</p>
    <p id="p3">para 3. You can find more information about how Google uses and stores content in
    the privacy policy or additional terms for particular Services.</p>
  </body>
</html>
```

### background-color

- This property specifies background color of the element.
- You can set any color as background color.

Syntax: background-color: colorname;

Example: background-color: green;

---

**Example of "background-color" property**

---

```
<html>
  <head>
    <title>CSS - Background Color</title>
    <style type="text/css">
      #p1
      {
        background-color: red;
      }
      #p2
      {
        background-color: green;
      }
      #p3
      {
        background-color: blue;
      }
    </style>
  </head>
  <body>
    <p id="p1">para 1</p>
    <p id="p2">para 2</p>
    <p id="p3">para 3</p>
  </body>
</html>
```

**Types of colors**

- Colors can be represented in 3 formats.
  1. Named colors
  2. RGB
  3. Hexadecimal number

**1. Named colors:**

- We can write name of the color directly.
- These are limited.
- We can't get exact shade of the color.
- Ex: white, black, red, green, orange, pink etc.
- It is not recommended in real-time.

**2. RGB:**

- RGB formula specifies that the composition of 3 basic colors (red, green, blue), generates 16 million colors.
- **Syntax:** `rgb(red, green, blue)`
- **Red** = 0 to 255

- **Green** = 0 to 255

- **Blue** = 0 to 255

- **Example:**

- `rgb(0, 0, 0)` = black
- `rgb(255, 0, 0)` = red
- `rgb(0, 255, 0)` = green
- `rgb(0, 0, 255)` = blue
- `rgb(255, 255, 0)` = yellow
- `rgb(255, 255, 255)` = white

### 3. Hexadecimal format:

- “Hexadecimal format” is the shortcut for “RGB”.
- Hexadecimal number system supports 16 symbols as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f.
- Hexadecimal number should be 6 symbols long, with “#” prefix. First two symbols represent red; Second two symbols represent green; Third two symbols represent blue.
- **Syntax:** `#redgreenblue`
- **Example:**
  - `#ffffff` = white
  - `#000000` = black
  - `#ff0000` = red
- “Hexadecimal format” is recommended in realtime.

### Named colors - Example

---

```
<html>
  <head>
    <title>CSS - Background Color - Named Colors</title>
    <style type="text/css">
      #p1
      {
        background-color: green;
      }
    </style>
  </head>
  <body>
    <p id="p1">Para 1</p>
  </body>
</html>
```

### RGB - Example

```
<html>
  <head>
    <title>CSS - Background Color - RGB</title>
    <style type="text/css">
      #p1
      {
        background-color: rgb(240, 250, 160);
      }
    </style>
  </head>
  <body>
    <p id="p1">Para 1</p>
  </body>
</html>
```

### Hexadecimal colors - Example

```
<html>
  <head>
    <title>CSS - Background Color - Hex</title>
    <style type="text/css">
      #p1
      {
        background-color: #6fdca3;
      }
    </style>
  </head>
  <body>
    <p id="p1">Para 1</p>
  </body>
</html>
```

## Font Styles

### font-family

- This property specifies name of the font. You refer the font family names from notepad.
- It is recommended to specify multiple font family names; if specified browser tries the subsequent font if the previous font is not found / not supported in the browser.

**Syntax:** font-family: "fontname";

**Example:** font-family: "Arial";

### Example of "font-family" property

```
<html>
  <head>
    <title>CSS - Font-family</title>
    <style type="text/css">
      #p1
```

```
<html>
  <head>
    <style type="text/css">
      #p1
      {
        font-family: 'Times New Roman';
        color: red;
      }
      #p2
      {
        font-family: 'Consolas';
        color: green;
      }
      #p3
      {
        font-family: 'Comic Sans MS';
        color: blue;
      }
    </style>
  </head>
  <body>
    <p id="p1">para 1.</p>
    <p id="p2">para 2</p>
    <p id="p3">para 3</p>
  </body>
</html>
```

### font-size

- This property specifies size of the font.
- You can specify the font size in the form of pixels / percentage / "EM" size.

**Syntax:** font-size: pixels;

**Example:** font-size: 30px;

### Example of "font-size" property

```
<html>
  <head>
    <title>CSS - Font-size</title>
    <style type="text/css">
      #p1
      {
        font-size: 16px;
        color: red;
      }
      #p2
      {
        font-size: 30px;
        color: green;
      }
      #p3
      {
        font-size: 50px;
        color: blue;
      }
    </style>
  </head>
  <body>
    <p id="p1">para 1</p>
    <p id="p2">para 2</p>
    <p id="p3">para 3</p>
  </body>
</html>
```

```
        }
    </style>
</head>
<body>
    <p id="p1">para 1</p>
    <p id="p2">para 2</p>
    <p id="p3">para 3</p>
</body>
</html>
```

### font-weight

- This property applies bold.
- The default value is "normal".
- If the value is "normal", the text appears normally.
- If the value is "bold", the text appears thick (bold).

Syntax:      font-weight: normal | bold;

Example:      font-weight: bold;

### Example of "font-weight" property

```
<html>
    <head>
        <title>CSS - Font-weight</title>
        <style type="text/css">
            #p1
            {
                font-weight: normal;
                color: red;
            }
            #p2
            {
                font-weight: bold;
                color: green;
            }
        </style>
    </head>
    <body>
        <p id="p1">para 1. You can find more information about how Google uses and stores content in the privacy policy.</p>
        <p id="p2">para 2. You can find more information about how Google uses and stores content in the privacy policy.</p>
    </body>
</html>
```

### font-style

- This property applies italic.

- The default value is "normal".
- If the value is "normal", the text appears normally.
- If the value is "italic", the text appears *italic*.

Syntax:      font-style: normal | italic;

Example:      font-style: italic;

#### Example of "font-style" property

```
<html>
  <head>
    <title>CSS - Font-style</title>
    <style type="text/css">
      #p1
      {
        font-style: normal;
        color: red;
      }
      #p2
      {
        font-style: italic;
        color: green;
      }
    </style>
  </head>
  <body>
    <p id="p1">para 1. You can find more information about how Google uses and stores content in
    the privacy policy.</p>
    <p id="p2">para 2. You can find more information about how Google uses and stores content in
    the privacy policy.</p>
  </body>
</html>
```

## Text Styles

### letter-spacing

- This property specifies gap between letters. For example, in "ABC", the letter-spacing specifies gap between "A" and "B"; and also gap between "B" and "C".

Syntax:      letter-spacing: pixels;

Example:      letter-spacing: 10px;

#### Example of "letter-spacing" property

```
<html>
  <head>
    <title>CSS - Letter-spacing</title>
    <style type="text/css">
      #p1
```

```
<html>
  <head>
    <style type="text/css">
      #p1
      {
        letter-spacing: normal;
        color: red;
      }
      #p2
      {
        letter-spacing: -3px;
        color: green;
      }
      #p3
      {
        letter-spacing: 10px;
        color: blue;
      }
    </style>
  </head>
  <body>
    <p id="p1">para 1. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>
    <p id="p2">para 2. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>
    <p id="p3">para 3. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>
  </body>
</html>
```

### word-spacing

- This property specifies gap between words.
- Words are identified with spaces. Space is the separator of words.

**Syntax:** word-spacing: pixels;

**Example:** word-spacing: 10px;

### Example of "word-spacing" property

```
<html>
  <head>
    <title>CSS - Word-spacing</title>
    <style type="text/css">
      #p1
      {
        word-spacing: normal;
        color: red;
      }
      #p2
      {
        word-spacing: -10px;
        color: green;
      }
      #p3
      {
```

```
        word-spacing: 20px;
        color: blue;
    }
</style>
</head>
<body>
<p id="p1">para 1. You can find more information about how Google uses and stores content in
the privacy policy or additional terms for particular Services.</p>
<p id="p2">para 2. You can find more information about how Google uses and stores content in
the privacy policy or additional terms for particular Services.</p>
<p id="p3">para 3. You can find more information about how Google uses and stores content in
the privacy policy or additional terms for particular Services.</p>
</body>
</html>
```

### line-height

- This property specifies height of the line of text.
- You can specify the value in the form of pixels or percentage.

Syntax: line-height: pixels;

Example: line-height: 10px;

### Example of "line-height" property

```
<html>
<head>
    <title>CSS - Line-height</title>
    <style type="text/css">
        #p1
        {
            line-height: normal;
            color: red;
        }
        #p2
        {
            line-height: 12px;
            color: green;
        }
        #p3
        {
            line-height: 50px;
            color: blue;
        }
    </style>
</head>
<body>
    <p id="p1">para 1. You can find more information about how Google uses and stores content in
the privacy policy or additional terms for particular Services. You can find more information about how
Google uses and stores content in the privacy policy or additional terms for particular Services. You can
find more information about how Google uses and stores content in the privacy policy or additional
terms for particular Services.</p>

```

```
<p id="p2">para 2. You can find more information about how Google uses and stores content in  
the privacy policy or additional terms for particular Services. You can find more information about how  
Google uses and stores content in the privacy policy or additional terms for particular Services. You can  
find more information about how Google uses and stores content in the privacy policy or additional  
terms for particular Services.</p>  
<p id="p3">para 3. You can find more information about how Google uses and stores content in  
the privacy policy or additional terms for particular Services. You can find more information about how  
Google uses and stores content in the privacy policy or additional terms for particular Services. You can  
find more information about how Google uses and stores content in the privacy policy or additional  
terms for particular Services.</p>  
</body>  
</html>
```

### text-decoration

- This property specifies underline / overline / strikeout for the text.
- The default value is "none".

**Syntax:** text-decoration: none | underline | overline | line-through;

**Example:** text-decoration: underline;

### Example of "text-decoration" property

```
<html>  
  <head>  
    <title>CSS - Text-decoration</title>  
    <style type="text/css">  
      #p1  
      {  
        text-decoration: none;  
        color: red;  
      }  
      #p2  
      {  
        text-decoration: underline;  
        color: green;  
      }  
      #p3  
      {  
        text-decoration: overline;  
        color: blue;  
      }  
      #p4  
      {  
        text-decoration: line-through;  
        color: hotpink;  
      }  
    </style>  
  </head>  
  <body>
```

<p id="p1">para 1. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>

<p id="p2">para 2. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>

<p id="p3">para 3. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>

<p id="p4">para 4. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>

</body>  
</html>

### **text-transform**

- This property specifies uppercase / lowercase / title case for the text.
- The default value is "none".
- If the value is "none", the text appears normally.
- If the value is "uppercase", the text appears **UPPER CASE**.
- If the value is "lowercase", the text appears lower case.
- If the value is "capitalize", the first letter of each word will be Capital.

**Syntax:** text-transform: none | uppercase | lowercase | capitalize;

**Example:** text-transform: underline;

### **Example of "text-transform" property**

```
<html>
  <head>
    <title>CSS - Text-transform</title>
    <style type="text/css">
      #p1
      {
        text-transform: none;
        color: red;
      }
      #p2
      {
        text-transform: uppercase;
```

```
        color: green;
    }
    #p3
    {
        text-transform: lowercase;
        color: blue;
    }
    #p4
    {
        text-transform: capitalize;
        color: hotpink;
    }

```

</style>

```
</head>
<body>
    <p id="p1">para 1. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>
    <p id="p2">para 2. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>
    <p id="p3">para 3. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>
    <p id="p4">para 4. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>

```

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

### **text-align**

- This property specifies alignment for the text.
- The default value is "left".

**Syntax:** text-align: left | right | center | justify;

**Example:** text-align: left;

### **Example of "text-align" property**

```
<html>
    <head>
        <title>CSS - Text-align</title>
```

```
<style type="text/css">
  #p1
  {
    text-align: left;
    color: red;
  }
  #p2
  {
    text-align: right;
    color: green;
  }
  #p3
  {
    text-align: center;
    color: blue;
  }
  #p4
  {
    text-align: justify;
    color: hotpink;
  }
</style>
</head>
<body>
  <p id="p1">para 1. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>
  <p id="p2">para 2. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>
  <p id="p3">para 3. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>
  <p id="p4">para 4. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services. You can find more information about how Google uses and stores content in the privacy policy or additional terms for particular Services.</p>
</body>
</html>
```

#### text-indent

- This property specifies left margin for the first line of the paragraph.
- It is rare to use in real-time.

- Use text-indent if you want to start first line of the paragraph from right side.

Syntax: text-indent: pixels;

Example: text-indent: 10px;

### Example of "text-indent" property

```
<html>
  <head>
    <title>CSS - Text-indent</title>
    <style type="text/css">
      #p1
      {
        text-indent: 0px;
        color: red;
      }
      #p2
      {
        text-indent: 20px;
        color: green;
      }
      #p3
      {
        text-indent: 40px;
        color: blue;
      }
      #p4
      {
        text-indent: 60px;
        color: hotpink;
      }
    </style>
  </head>
  <body>
    <p id="p1">para 1. You can find more information about how Google uses and stores content in
    the privacy policy or additional terms for particular Services. You can find more information about how
    Google uses and stores content in the privacy policy or additional terms for particular Services. You can
    find more information about how Google uses and stores content in the privacy policy or additional
    terms for particular Services.</p>
    <p id="p2">para 2. You can find more information about how Google uses and stores content in
    the privacy policy or additional terms for particular Services. You can find more information about how
    Google uses and stores content in the privacy policy or additional terms for particular Services. You can
    find more information about how Google uses and stores content in the privacy policy or additional
    terms for particular Services.</p>
    <p id="p3">para 3. You can find more information about how Google uses and stores content in
    the privacy policy or additional terms for particular Services. You can find more information about how
    Google uses and stores content in the privacy policy or additional terms for particular Services. You can
    find more information about how Google uses and stores content in the privacy policy or additional
    terms for particular Services.</p>
    <p id="p4">para 4. You can find more information about how Google uses and stores content in
    the privacy policy or additional terms for particular Services. You can find more information about how
    Google uses and stores content in the privacy policy or additional terms for particular Services. You can
```

```
find more information about how Google uses and stores content in the privacy policy or additional  
terms for particular Services.</p>  
</body>  
</html>
```

## Span

- <span> represents a small amount of text, for which you can apply some special formatting.
- <span> is a paired tag.
- <span> tag doesn't apply any style by default; but we can apply any style for it.

### Syntax:

```
<span>Your content here</span>
```

### Example:

```
<span>Hello</span>
```

### Example of span

```
<html>  
  <head>  
    <title>CSS - Span</title>  
    <style type="text/css">  
      #span1  
      {  
        color: hotpink;  
        font-weight: bold;  
      }  
    </style>  
  </head>  
  <body>  
    <p id="p1">You can find more information about how Google uses and stores content in the  
    <span id="span1">privacy policy</span> or additional terms for particular Services.</p>  
  </body>  
</html>
```

## Background Image

### background-image

- This property specifies background image of the element.
- It is recommended to place the background image file in the current folder.

**Syntax:** background-image: url("filename.extension");

**Example:** background-image: url("sample.png");

### Example of "background-image" property

```
<html>  
  <head>  
    <title>CSS - Background-image</title>
```

```
<style type="text/css">
  #p1
  {
    background-color: skyblue;
    background-image: url('sample.png');
  }
</style>
</head>
<body>
  <p id="p1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. </p>
</body>
</html>
```

Note: Place "sample.png" in the current folder.

### background-attachment

- This property specifies whether the background image should be scrolled along with the text, when the user uses the scrollbars of the web page.
- If the value is "scroll", the background image will be scrolled along with the text.
- If the value is "fixed", the background image will not be scrolled along with the text; Only the text will be scrolled; background image will be constantly appears.

Syntax: background-attachment: scroll | fixed;

Example: background-attachment: scroll;

**Example of "background-attachment" property**

```
<html>
  <head>
    <title>CSS - Background-attachment</title>
    <style type="text/css">
      body
      {
        background-color: skyblue;
        background-image: url("trees.jpg");
        background-attachment: fixed;
      }
    </style>
  </head>
  <body>
    <p>Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. </p>
  </body>
</html>
```

**Note:** Place "trees.jpg" in the current folder.

**Lists**

**list-style-type for <ul>**

- This property specifies type of the bullet for the <ul> tag.
- <ul> tag is used to display a list of names. <li> tag represents an item in the <ul>
- The default value is "disc", in case of <ul> tag.

**Syntax:** list-style-type: none | disc | square | circle;

**Example:** list-style-type: disc;

---

Example of "list-style-type" property for <ul>

---

```
<html>
  <head>
    <title>CSS - List-style-type - UL</title>
    <style type="text/css">
      #list1
      {
        list-style-type: disc;
      }

      #list2
      {
        list-style-type: square;
      }

      #list3
      {
        list-style-type: circle;
      }

      #list4
      {
        list-style-type: none;
      }
    </style>
  </head>
  <body>
    <h2>UL - disc</h2>
    <ul id="list1">
      <li>India</li>
      <li>Japan</li>
      <li>China</li>
      <li>USA</li>
      <li>UK</li>
    </ul>

    <h2>UL - square</h2>
    <ul id="list2">
      <li>India</li>
      <li>Japan</li>
      <li>China</li>
      <li>USA</li>
      <li>UK</li>
    </ul>

    <h2>UL - circle</h2>
    <ul id="list3">
      <li>India</li>
      <li>Japan</li>
      <li>China</li>
      <li>USA</li>
```

```
<li>UK</li>
</ul>

<h2>UL - none</h2>
<ul id="list4">
    <li>India</li>
    <li>Japan</li>
    <li>China</li>
    <li>USA</li>
    <li>UK</li>
</ul>
</body>
</html>
```

### **list-style-type for <ol>**

- This property specifies type of numbering for the <ol> tag.
- If the value is "none", no numbering will be displayed.
- If the value is "decimal", the numbers will be "1, 2, 3 ...".
- If the value is "decimal-leading-zero", the numbers will be "01, 02, 03, ...".
- If the value is "lower-alpha", the numbers will be "a, b, c, ...".
- If the value is "upper-alpha", the numbers will be "A, B, C, ...".
- If the value is "lower-roman", the numbers will be "i, ii, iii, ...".
- If the value is "upper-roman", the numbers will be "I, II, III, ...".
- If the value is "lower-greek", the numbers will be Greek letters.

**Syntax:** list-style-type: none | decimal | decimal-leading-zero |

lower-alpha | upper-alpha | lower-roman | upper-roman | lower-greek;

**Example:** list-style-type: decimal;

### **Example of "list-style-type" property for <ol>:**

```
<html>
  <head>
    <title>CSS - List-style-type - OL</title>
    <style type="text/css">
      #list1
      {
        list-style-type: decimal;
      }

      #list2
      {
        list-style-type: decimal-leading-zero;
      }
    </style>
  </head>
  <body>
    <ol id="list1">
      <li>India</li>
      <li>Japan</li>
      <li>China</li>
      <li>USA</li>
      <li>UK</li>
    </ol>

    <ol id="list2">
      <li>India</li>
      <li>Japan</li>
      <li>China</li>
      <li>USA</li>
      <li>UK</li>
    </ol>
  </body>
</html>
```

```
#list3
{
    list-style-type: lower-alpha;
}

#list4
{
    list-style-type: upper-alpha;
}

#list5
{
    list-style-type: lower-roman;
}

#list6
{
    list-style-type: upper-roman;
}

#list7
{
    list-style-type: lower-greek;
}

#list8
{
    list-style-type: none;
}
</style>
</head>
<body>
<h2>OL - decimal</h2>
<ol id="list1">
<li>India</li>
<li>Japan</li>
<li>China</li>
<li>USA</li>
<li>UK</li>
</ol>

<h2>OL - decimal-leading-zero</h2>
<ol id="list2">
<li>India</li>
<li>Japan</li>
<li>China</li>
<li>USA</li>
<li>UK</li>
</ol>

<h2>OL - lower-alpha</h2>
```

```
<ol id="list3">
  <li>India</li>
  <li>Japan</li>
  <li>China</li>
  <li>USA</li>
  <li>UK</li>
</ol>

<h2>OL - upper-alpha</h2>
<ol id="list4">
  <li>India</li>
  <li>Japan</li>
  <li>China</li>
  <li>USA</li>
  <li>UK</li>
</ol>

<h2>OL - lower-roman</h2>
<ol id="list5">
  <li>India</li>
  <li>Japan</li>
  <li>China</li>
  <li>USA</li>
  <li>UK</li>
</ol>

<h2>OL - upper-roman</h2>
<ol id="list6">
  <li>India</li>
  <li>Japan</li>
  <li>China</li>
  <li>USA</li>
  <li>UK</li>
</ol>

<h2>OL - lower-greek</h2>
<ol id="list7">
  <li>India</li>
  <li>Japan</li>
  <li>China</li>
  <li>USA</li>
  <li>UK</li>
</ol>

<h2>OL - none</h2>
<ol id="list8">
  <li>India</li>
  <li>Japan</li>
  <li>China</li>
  <li>USA</li>
  <li>UK</li>
```

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

### list-style-image

- This property specifies bullet image file path for the list.
- It is recommended to place the image file in the same folder, where the html file is present.

**Syntax:** list-style-image: url("filename.extension");

**Example:** list-style-image: url("tick.gif");

### Example of "list-style-image" property:

```
<html>
  <head>
    <title>CSS - List-style-image</title>
    <style type="text/css">
      #list1
      {
        list-style-image: url("tick.gif");
      }
    </style>
  </head>
  <body>
    Top most browsers:
    <ul id="list1">
      <li>Google Chrome</li>
      <li>Mozilla Firefox</li>
      <li>Internet Explorer</li>
      <li>Safari</li>
      <li>Opera</li>
    </ul>
  </body>
</html>
```

**Note:** Place "tick.gif" in the current folder.

## DIV

### <div> tag

- <div> is the most important tag of html.
- <div> tag represents a division (part) of the page.
- <div> is a container, in which you can place any other elements.
- A web page can have any no. of <div> tags.
- A <div> tag can be placed in another <div> tag also.

- <div> is a paired tag.
- <div> is a block level element. That means it occupies 100% of the width, by default. So the content next to the <div> tag, appears in the next line.
- **Syntax:** <div> any content </div>
- **Example:** <div> Hello </div>

### Example of <div> tag

```
<html>
  <head>
    <title>CSS - Div</title>
    <style type="text/css">
      #div1
      {
        background-color: #00ffcc;
      }
      #div2
      {
        background-color: #ff3366;
      }
      #div3
      {
        background-color: #00ccff;
      }
    </style>
  </head>
  <body>
    <div id="div1">div 1</div>
    <div id="div2">div 2</div>
    <div id="div3">div 3</div>
  </body>
</html>
```

### "width" and "height" properties

#### "width" property

- This property specifies horizontal size of the element.
- You can specify the value in the form of pixels or percentages.

**Syntax:** width: pixels;

**Example:** width: 200px;

#### "height" property

- This property specifies vertical size of the element.
- You can specify the value in the form of pixels or percentages.

**Syntax:** height: pixels;

Example: height: 100px;

#### Example of "width" and "height" properties

```
<html>
  <head>
    <title>CSS - Width and Height</title>
    <style type="text/css">
      #div1
      {
        background-color: #00ffcc;
        width: 300px;
        height: 300px;
      }
      #div2
      {
        background-color: #ff3366;
        width: 300px;
        height: 300px;
      }
      #div3
      {
        background-color: #00ccff;
        width: 300px;
        height: 300px;
      }
    </style>
  </head>
  <body>
    <div id="div1">div 1</div>
    <div id="div2">div 2</div>
    <div id="div3">div 3</div>
  </body>
</html>
```

#### float

- This property displays the elements side-by-side.
- If the value is "left", the elements will be displayed side-by-side, starting from "left", towards "right".
- If the value is "right", the elements will be displayed side-by-side, starting from "right", towards "left".

Syntax: float: left | right;

Example: float: left;

#### Example of "float - left"

```
<html>
  <head>
    <title>CSS - Float=Left</title>
    <style type="text/css">
```

```
#div1
{
    background-color: #00ffcc;
    width: 300px;
    height: 300px;
    float: left;
}
#div2
{
    background-color: #ff3366;
    width: 300px;
    height: 300px;
    float: left;
}
#div3
{
    background-color: #00ccff;
    width: 300px;
    height: 300px;
    float: left;
}
</style>
</head>
<body>
    <div id="div1">div 1</div>
    <div id="div2">div 2</div>
    <div id="div3">div 3</div>
</body>
</html>
```

#### Example of "float = right"

```
<html>
    <head>
        <title>CSS - Float=Right</title>
        <style type="text/css">
            #div1
            {
                background-color: #00ffcc;
                width: 300px;
                height: 300px;
                float: right;
            }
            #div2
            {
                background-color: #ff3366;
                width: 300px;
                height: 300px;
                float: right;
            }
            #div3
            {
```

```
background-color: #00ccff;
width: 300px;
height: 300px;
float: right;
}
</style>
</head>
<body>
<div id="div1">div 1</div>
<div id="div2">div 2</div>
<div id="div3">div 3</div>
</body>
</html>
```

### clear

- This property cancels the effect of “float” and push the current element to next line.
- It stops the sequence of “float”.
- To stop “float:left” sequence, we use “clear:left”.
- To stop “float:right” sequence, we use “clear:right”.

Syntax: clear: left | right;

Example: clear: left;

### Example of “clear - left”

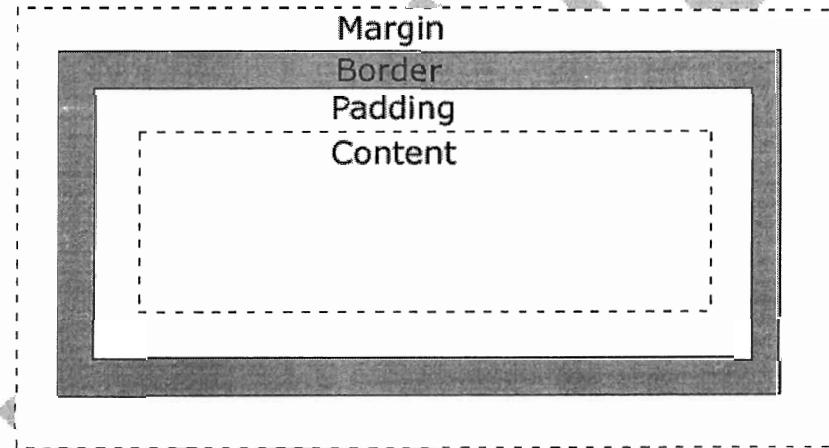
```
<html>
<head>
<title>CSS - Clear=left</title>
<style type="text/css">
#div1
{
background-color: #00ffcc;
width: 300px;
height: 300px;
float: left;
}
#div2
{
background-color: #ff3366;
width: 300px;
height: 300px;
float: left;
}
#div3
{
background-color: #00ccff;
width: 300px;
height: 300px;
clear: left;
}
```

```
</style>
</head>
<body>
  <div id="div1">div 1</div>
  <div id="div2">div 2</div>
  <div id="div3">div 3</div>
</body>
</html>
```

## Box Model

### Understanding Box Model

- Box model is combination of "padding, border and margin".
- All visible elements are displayed based on box model in the web page.



### border-style

- This property specifies type of the border, around the element.
- The default value is "none".
- "Solid" is recommended.
- Depending on the requirement, you can use any of the other borders.

**Syntax:** border-style: none | solid | dotted | dashed | double | inset | outset | ridge | groove;

**Example:** border-style: solid;

### border-width

- This property specifies thickness of the border.
- You can specify the border width in the form of pixels.

**Syntax:** border-width: pixels;

**Example:** border-width: 5px;

### border-color

- This property specifies color of the border.
- You can specify any color.

**Syntax:** border-color: any color;

**Example:** border-color: red;

### Example of "border-style", "border-width", "border-color"

```
<html>
  <head>
    <title>CSS - Border</title>
    <style type="text/css">
      #div1
      {
        background-color: #00ffcc;
        width: 300px;
        height: 300px;
        float: left;
        border-style: solid; /* none | solid | double | dotted | dashed | groove | inset | outset */
        border-width: 5px;
        border-color: red;
      }
      #div2
      {
        background-color: #cc33ff;
        width: 300px;
        height: 300px;
        float: left;
        border-style: dotted;
        border-width: 5px;
        border-color: green;
      }
      #div3
      {
        background-color: #00ccff;
        width: 300px;
        height: 300px;
        float: left;
        border-style: dashed;
        border-width: 5px;
        border-color: blue;
      }
    </style>
  </head>
  <body>
    <div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry.</div>
    <div id="div2">Lorem Ipsum is simply dummy text of the printing and typesetting industry.</div>
```

```
<div id="div3">Lorem Ipsum is simply dummy text of the printing and typesetting industry.</div>
</body>
</html>
```

### border - shortcut

- This property specifies border width, border style and border color, at-a-time, in shortcut way.

**Syntax:** border: borderwidth borderstyle borderColor;

**Example:** border: 5px solid red;

### Example of "border" property

```
<html>
  <head>
    <title>CSS - Border - shortcut</title>
    <style type="text/css">
      #div1
      {
        background-color: #00ffcc;
        width: 300px;
        height: 300px;
        float: left;
        border: 5px solid red;
      }
      #div2
      {
        background-color: #ccccff;
        width: 300px;
        height: 300px;
        float: left;
        border: 5px dotted green;
      }
      #div3
      {
        background-color: #00ccff;
        width: 300px;
        height: 300px;
        float: left;
        border: 5px dashed blue;
      }
    </style>
  </head>
  <body>
    <div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem
    Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took
    a galley of type and scrambled it to make a type specimen book.</div>
    <div id="div2">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem
    Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took
    a galley of type and scrambled it to make a type specimen book.</div>
  </body>
</html>
```

```
<div id="div3">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem  
Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took  
a galley of type and scrambled it to make a type specimen book.</div>  
</body>  
</html>
```

## border - sides

### "border-top" property

- This property specifies border width, border style and border color for top side only.

**Syntax:** border-top: borderwidth borderstyle bordercolor;

**Example:** border-top: 5px solid red;

### "border-right" property

- This property specifies border width, border style and border color for right side only.

**Syntax:** border-right: borderwidth borderstyle bordercolor;

**Example:** border-right: 5px solid red;

### "border-bottom" property

- This property specifies border width, border style and border color for bottom side only.

**Syntax:** border-bottom: borderwidth borderstyle bordercolor;

**Example:** border-bottom: 5px solid red;

### "border-left" property

- This property specifies border width, border style and border color for left side only.

**Syntax:** border-left: borderwidth borderstyle bordercolor;

**Example:** border-left: 5px solid red;

### Example of "border - sides" property

```
<html>  
  <head>  
    <title>border-sides</title>  
    <style type="text/css">  
      #div1  
      {  
        background-color: #0099ff;  
        width: 300px;  
        height: 100px;  
        border-left: 10px solid red;  
        border-top: 10px solid red;  
      }  
      #div2
```

```
<html>
  <head>
    <style type="text/css">
      #div1
      {
        background-color: #ff6699;
        width: 300px;
        height: 100px;
        border-top: 10px solid green;
        border-right: 10px solid green;
      }
      #div3
      {
        background-color: #ccff99;
        width: 300px;
        height: 100px;
        border-top: 10px solid darkblue;
        border-bottom: 10px solid darkblue;
      }
    </style>
  </head>
  <body>
    <div id="div1">div 1</div>
    <div id="div2">div 2</div>
    <div id="div3">div 3</div>
  </body>
</html>
```

### margin

- This property specifies the margin (gap) between element to element, surrounding the element.
- For example, if two `<div>` tags are display side-by-side, the gap between them is called as "margin".
- You can specify the value in the form of pixels.

**Syntax:** margin: pixels;

**Example:** margin: 10px;

### Example of "margin" property

```
<html>
  <head>
    <title>CSS - Margin</title>
    <style type="text/css">
      body
      {
        text-align: justify;
      }
      #div1
      {
        background-color: #00ffcc;
        width: 300px;
        height: 300px;
        float: left;
        border: 5px solid red;
      }
    </style>
  </head>
  <body>
    <div id="div1">Div 1</div>
    <div id="div2">Div 2</div>
  </body>
</html>
```

```
margin: 10px;
}
#div2
{
background-color: #ff3366;
width: 300px;
height: 300px;
float: left;
border: 5px solid green;
margin: 10px;
}
#div3
{
background-color: #00ccff;
width: 300px;
height: 300px;
float: left;
border: 5px solid blue;
margin: 10px;
}
</style>
</head>
<body>
<div id="div1">div 1</div>
<div id="div2">div 2</div>
<div id="div3">div 3</div>
</body>
</html>
```

## margin - sides

### "margin-top" property

- This property specifies the top margin (gap) between element to element.

**Syntax:** margin-top: pixels;

**Example:** margin-top: 10px;

### "margin-right" property

- This property specifies the right margin (gap) between element to element.

**Syntax:** margin-right: pixels;

**Example:** margin-right: 10px;

### "margin-bottom" property

- This property specifies the bottom margin (gap) between element to element.

**Syntax:** margin-bottom: pixels;

**Example:** margin-bottom: 10px;

### **“margin-left” property**

- This property specifies the left margin (gap) between element to element.

**Syntax:** margin-left: pixels;

**Example:** margin-left: 10px;

### **Example of “margin” property - sides**

```
<html>
  <head>
    <title>CSS - Margin - separate</title>
    <style type="text/css">
      body
      {
        text-align: justify;
      }
      #div1
      {
        background-color: #00ffcc;
        width: 300px;
        height: 300px;
        float: left;
        border: 5px solid red;
        margin-top: 10px;
        margin-right: 20px;
        margin-bottom: 20px;
        margin-left: 5px;
      }
      #div2
      {
        background-color: #ff3366;
        width: 300px;
        height: 300px;
        float: left;
        border: 5px solid green;
        margin-top: 10px;
        margin-right: 40px;
        margin-bottom: 20px;
        margin-left: 5px;
      }
      #div3
      {
        background-color: #00ccff;
        width: 300px;
        height: 300px;
        float: left;
        border: 5px solid blue;
        margin-top: 10px;
        margin-right: 20px;
        margin-bottom: 20px;
        margin-left: 5px;
      }
    </style>
  </head>
  <body>
    <div id="div1">Div 1</div>
    <div id="div2">Div 2</div>
    <div id="div3">Div 3</div>
  </body>
</html>
```

```
        }
    </style>
</head>
<body>
    <div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.</div>
    <div id="div2">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.</div>
    <div id="div3">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.</div>
</body>
</html>
```

### margin - shortcut

- This property specifies the margin (gap) between element to element, all sides independently at-a-time.

**Syntax:** margin: topmargin rightmargin bottommargin leftmargin;

**Example:** margin: 10px 5px 15px 30px;

### Example of "margin" property - shortcut

```
<html>
<head>
    <title>CSS - Margin - shortcut</title>
    <style type="text/css">
        body
        {
            text-align: justify;
        }
        #div1
        {
            background-color: #00ffcc;
            width: 300px;
            height: 300px;
            float: left;
            border: 5px solid red;
            margin: 10px 20px 20px 5px;
        }
        #div2
        {
            background-color: #ff3366;
            width: 300px;
            height: 300px;
            float: left;
            border: 5px solid green;
            margin: 10px 40px 20px 5px;
        }
    </style>
</head>
<body>
    <div id="div1"></div>
    <div id="div2"></div>
</body>

```

```
<html>
  <head>
    <style type="text/css">
      #div3
      {
        background-color: #00ccff;
        width: 300px;
        height: 300px;
        float: left;
        border: 5px solid blue;
        margin: 10px 20px 20px 5px;
      }
    </style>
  </head>
  <body>
    <div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.</div>
    <div id="div2">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.</div>
    <div id="div3">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.</div>
  </body>
</html>
```

## padding

- This property specifies the padding (gap) between border and content of the element.
- You can specify the value in the form of pixels.
- Inner margin is called as "padding".
- Outer margin is called as "margin".

Syntax: padding: pixels;

Example: padding: 10px;

## Example of "padding" property

```
<html>
  <head>
    <title>CSS - Padding</title>
    <style type="text/css">
      body
      {
        text-align: justify;
      }
      #div1
      {
        background-color: #00ffcc;
        width: 300px;
        height: 300px;
      }
    </style>
  </head>
  <body>
    <div id="div1">This is a sample text for demonstration purposes. The text is justified to show the effect of padding on the content area.
    </div>
  </body>
</html>
```

```
float: left;
border: 5px solid red;
margin: 10px;
padding: 20px;
}
#div2
{
background-color: #ff3366;
width: 300px;
height: 300px;
float: left;
border: 5px solid green;
margin: 10px;
padding: 20px;
}
#div3
{
background-color: #00ccff;
width: 300px;
height: 300px;
float: left;
border: 5px solid blue;
margin: 10px;
padding: 20px;
}
</style>
</head>
<body>
<div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.</div>
<div id="div2">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.</div>
<div id="div3">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.</div>
</body>
</html>
```

### **padding - sides**

#### **"padding-top" property**

- This property specifies the top padding.

**Syntax:** padding-top: pixels;

**Example:** padding-top: 10px;

### **"padding-right" property**

- This property specifies the right padding.

**Syntax:** padding-right: pixels;

**Example:** padding-right: 10px;

### **"padding-bottom" property**

- This property specifies the bottom padding.

**Syntax:** padding-bottom: pixels;

**Example:** padding-bottom: 10px;

### **"padding-left" property**

- This property specifies the left padding.

**Syntax:** padding-left: pixels;

**Example:** padding-left: 10px;

### **padding - shortcut**

- This property specifies the padding for all sides independently at-a-time.

**Syntax:** padding: top padding right padding bottom padding left padding;

**Example:** padding: 10px 5px 15px 30px;

## **Advanced CSS Properties**

### **"opacity" property**

- This property makes the element transparent (background information is visible through the element).
- You can specify any number between 0 to 1.
- Any middle number between 0 and 1 is recommended. For example, "0.6".

**Syntax:** opacity: 0 to 1;

**Example:** opacity: 0.6;

1 : fully visible

0.9, ..., 0.1 : transparent

0 : fully transparent / invisible

### **Example of "opacity" property:**

```
<html>
  <head>
    <title>CSS - Opacity</title>
    <style type="text/css">
```

```
<html>
  <head>
    <style>
      body
      {
        background-image: url('img1.jpg');
      }
      #div1
      {
        font-size: 40px;
        font-family: 'Tahoma';
        width: 300px;
        height: 200px;
        background-color: #009966;
        opacity: 0.9; /* 0 to 1 */
      }
    </style>
  </head>
  <body>
    <div id="div1">div1</div>
  </body>
</html>
```

Note: Place "img1.jpg" in the current folder.

### display

- This property specifies display mode of the element.

**Syntax:** display: block | inline | none;

**Example:** display: none;

- "display: block" is default for all the block level elements.
- "display: inline" is default for all the inline elements.
- "display: none" hides the element and its space will be reclaimed by other elements automatically.

### Example of "display" property

```
<html>
  <head>
    <title>CSS - display</title>
    <style type="text/css">
      #div1
      {
        background-color: #00ccff;
        display: block;
      }
      #div2
      {
        background-color: #ff6666;
        display: none;
      }
      #div3
```

```
<html>
  <head>
    <style>
      #div1
      {
        background-color: #00cc99;
        display: block;
      }
    </style>
  </head>
  <body>
    <div id="div1">div 1</div>
    <div id="div2">div 2</div>
    <div id="div3">div 3</div>
  </body>
</html>
```

### visibility

- This property specifies whether the element is visible or invisible in the web page.

**Syntax:** visibility: visible | hidden;

**Example:** visibility: hidden;

- “visibility: visible” shows the element.
- “visibility: hidden” hides the element and its space will be reserved as-it-is.

### Example of “visibility” property

```
<html>
  <head>
    <title>CSS - display</title>
    <style type="text/css">
      #div1
      {
        background-color: #00ccff;
        visibility: visible;
      }
      #div2
      {
        background-color: #00ff99;
        visibility: hidden;
      }
      #div3
      {
        background-color: #ff0099;
        visibility: visible;
      }
    </style>
  </head>
  <body>
    <div id="div1">div 1</div>
    <div id="div2">div 2</div>
    <div id="div3">div 3</div>
  </body>
```

```
</html>
```

### overflow

- This property specifies how the text should appear, which is outside the boundaries of the element.

**Syntax:** overflow: visible | hidden | auto;

**Example:** overflow: auto;

- “overflow: visible” shows the additional content outside the element, which doesn’t fit within the element.
- “overflow: hidden” hides the additional content, which doesn’t fit within the element.
- “overflow: auto” shows scrollbars automatically if necessary.

#### Example of “overflow:visible” property

```
<html>
  <head>
    <title>CSS - overflow - visible</title>
    <style type="text/css">
      #div1
      {
        background-color: #00ccff;
        font-size: 25px;
        width: 300px;
        height: 200px;
        text-align: justify;
        overflow: visible;
      }
    </style>
  </head>
  <body>
    <div id="div1">Our Services are very diverse, so sometimes additional terms or product
    requirements (including age requirements) may apply. Additional terms will be available with the
    relevant Services, and those additional terms become part of your agreement with us if you use those
    Services.</div>
  </body>
</html>
```

#### Example of “overflow:hidden” property

```
<html>
  <head>
    <title>CSS - overflow - hidden</title>
    <style type="text/css">
      #div1
      {
        background-color: #00ccff;
        font-size: 25px;
        width: 300px;
      }
    </style>
  </head>
  <body>
    <div id="div1">Our Services are very diverse, so sometimes additional terms or product
    requirements (including age requirements) may apply. Additional terms will be available with the
    relevant Services, and those additional terms become part of your agreement with us if you use those
    Services.</div>
  </body>
</html>
```

```
height: 200px;
text-align: justify;
overflow: hidden;
}
</style>
</head>
<body>
<div id="div1">Our Services are very diverse, so sometimes additional terms or product
requirements (including age requirements) may apply. Additional terms will be available with the
relevant Services, and those additional terms become part of your agreement with us if you use those
Services.</div>
</body>
</html>
```

#### Example of "overflow:auto" property

```
<html>
<head>
<title>CSS - overflow - auto</title>
<style type="text/css">
#div1
{
background-color: #00ccff;
font-size: 25px;
width: 300px;
height: 200px;
text-align: justify;
overflow: auto;
}
</style>
</head>
<body>
<div id="div1">Our Services are very diverse, so sometimes additional terms or product
requirements (including age requirements) may apply. Additional terms will be available with the
relevant Services, and those additional terms become part of your agreement with us if you use those
Services.</div>
</body>
</html>
```

### Types of Style Sheets

- CSS style sheets are 3 types:
  1. Internal Style Sheet / Embedded Style Sheet
  2. External Style Sheet
  3. Inline Style Sheet

#### 1. Internal Style Sheet / Embedded Style Sheet

- The css styles are defined inside the ".html" file itself.

- **Advantage:** Easy-to-understand.
- **Disadvantage:** No re-usability. That means, the styles defined in one html file, can't be accessed in another html file.

**Syntax:**

```
<style type="text/css">  
    css code  
</style>
```

## 2. External Style Sheet

- The css styles are defined in a separate ".css" file and html tags are defined in the ".html" file.
- We can link (connect) the css file with html file.
- **Advantage:** It supports re-usability. That means, the css file can be called in many html files.

**Syntax to import css file:**

```
<link href="filename.css" rel="stylesheet">
```

### Example of External Style Sheet

#### StyleSheet.css

```
h1  
{  
    font-family: 'Tahoma';  
    font-size: 25px;  
    color: green;  
    background-color: darkgreen;  
    color: yellow;  
}
```

#### Page1.html

```
<html>  
    <head>  
        <title>CSS - External stylesheet - page 1</title>  
        <link href="StyleSheet.css" rel="stylesheet">  
    </head>  
    <body>  
        <h1>hello</h1>  
    </body>  
</html>
```

#### Page2.html

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

```
<title>CSS - External stylesheet - page 2</title>
<link href="StyleSheet.css" rel="stylesheet">
</head>
<body>
  <h1>hai</h1>
</body>
</html>
```

### 3. Inline Style Sheet

- The css styles will be defined inside the html tag itself.
- This is NOT recommended in realtime.

#### Drawbacks:

- a) HTML code will be cumbersome (confusing).
- b) No style re-usability.

#### Syntax:

```
<tag style="property:value; property:value; ...">
</tag>
```

#### Example of Inline Style Sheet

```
<html>
  <head>
    <title>CSS - Inline CSS</title>
  </head>
  <body>
    <p style="background-color:cyan; font-size:40px">para 1</p>
    <p style="background-color:cyan; font-size:40px">para 2</p>
  </body>
</html>
```

### CSS Selectors

- First we have to select the element / elements, and then only we can apply some styles to it.
- “Selector” is a “syntax to select”. It is used to select the desired elements in the web page.
- When we use a selector, the browser searches the entire web page for the matching elements and returns the matching elements; and we apply styles only for those matching elements.
- [List of CSS Selectors](#)

1. Tag Selector
2. ID Selector
3. Class Selector
4. Compound Selector
5. Grouping Selector

6. Child Selector
7. Direct Child Selector
8. Attribute Selector
9. Hover Selector
10. Focus Selector
11. Universal Selector

### Tag Selector

- It selects all the instances of the specified tag.
- You can specify any tag name.

**Syntax:** tag

**Example:** p

### Example on Tag Selector

```
<html>
  <head>
    <title>CSS - Tag Selector</title>
    <style type="text/css">
      p
      {
        background-color: skyblue;
      }
    </style>
  </head>
  <body>
    <h1>Tag Selector</h1>
    <p>para 1</p>
    <p>para 2</p>
    <p>para 3</p>
    <p>para 4</p>
    <p>para 5</p>
  </body>
</html>
```

### ID Selector

- It selects a single instance of the tag, based on the id.
- “ID” is “identification name”.
- “ID” should be unique (can’t be duplicate) in the web page.
- # is symbol of ID selector.

**Syntax:** #id

**Example:** #p1

### Example on ID Selector

```
<html>
  <head>
    <title>CSS - ID Selector</title>
    <style type="text/css">
      #p2
      {
        background-color: skyblue;
      }
    </style>
  </head>
  <body>
    <h1>ID Selector</h1>
    <p>para 1</p>
    <p id="p2">para 2</p>
    <p>para 3</p>
    <p>para 4</p>
  </body>
</html>
```

### Class Selector

- It selects one or more elements, based on the class name.
- We use same class for similar elements.
- ". " is the symbol of class selector.

**Syntax:** .class

**Example:** .c1

### Example on Class Selector

```
<html>
  <head>
    <title>CSS - Class Selector</title>
    <style type="text/css">
      .c1
      {
        background-color: skyblue;
      }
    </style>
  </head>
  <body>
    <h1>Class Selector</h1>
    <p>para 1</p>
    <p class="c1">para 2</p>
    <p>para 3</p>
    <p class="c1">para 4</p>
    <p>para 5</p>
    <p class="c1">para 6</p>
    <p class="c1">para 7</p>
  </body>
```

</html>

### Compound Selector

- It selects the instances of specific tag, which have specified class.

**Syntax:** tag.class

**Example:** p.c1

### Example on Compound Selector

```
<html>
  <head>
    <title>CSS - Compound Selector</title>
    <style type="text/css">
      /* selecting only <p> tags that have class="class1" */
      p.class1
      {
        background-color: skyblue;
      }
    </style>
  </head>
  <body>
    <h1>Compound Selector</h1>
    <p>para 1</p>
    <p class="class1">para 2</p>
    <p>para 3</p>
    <p class="class1">para 4</p>
    <p>para 5</p>
    <p class="class1">para 6</p>
    <input type="text">
    <input type="text" class="class1">
    <input type="text">
    <input type="text" class="class1">
  </body>
</html>
```

### Grouping Selector

- It selects the specified group of tags.
- "," is the symbol of grouping selector.

**Syntax:** tag1,tag2,tag3,...

**Example:** div,p,h2

### Example on Grouping Selector

```
<html>
  <head>
    <title>CSS - Grouping Selector</title>
    <style type="text/css">
```

```
/* Selecting all <h1> and all <p> tags */
h1,p,input,span
{
    background-color: skyblue;
}
</style>
</head>
<body>
    <h1>Grouping Selector</h1>
    <p>para 1</p>
    <p class="class1">para 2</p>
    <p>para 3</p>
    <p class="class1">para 4</p>
    <p>para 5</p>
    <p class="class1">para 6</p>
    <input type="text" value="hello">
    <input type="text" value="hello" class="class1">
    <input type="text" value="hello" class="class1">
    <span>span 1</span>
    <span>span 2</span>
</body>
</html>
```

### Child Selector

- It selects all the child tags (including grand children) of the specified parent tag.
- "space" is the symbol of child selector.

**Syntax:** parenttag childtag

**Example:** div p

### Example on Child Selector

```
<html>
    <head>
        <title>CSS - Child Selector</title>
        <style type="text/css">
            /* Selecting <p> which are children of <div> */
            div p
            {
                background-color: skyblue;
            }
        </style>
    </head>
    <body>
        <h1>Child Selector</h1>
        <div id="div1">
            <p>para 1</p>
            <p>para 2</p>
            <p>para 3</p>
            <p>para 4</p>
```

```
</div>
<p>para 5</p>
<p>para 6</p>
</body>
</html>
```

### Direct Child Selector

- It selects only the direct child tags (excluding the grand children) of the specified parent tag.
- ">" is the symbol of direct child selector.
- **Syntax:** parenttag>childtag
- **Example:** div>p

### Example on Direct Child Selector

```
<html>
  <head>
    <title>CSS - Direct Child selector</title>
    <style type="text/css">
      div>p
      {
        background-color: skyblue;
      }
    </style>
  </head>
  <body>
    <h1>Direct Child Selector</h1>
    <div id="div1">
      <p>para 1</p>
      <p>para 2</p>
      <p>para 3</p>
      <p>para 4</p>
      <b>
        <p>para 5</p>
        <p>para 6</p>
      </b>
    </div>
    <p>para 7</p>
  </body>
</html>
```

### Attribute Selector

- It selects all the tags that are having specified attribute.
- "[ ]" are the symbols of attribute selector.
- We can use any attribute of any html tag.

**Syntax:** tag[attribute="value"]

**Example:** img[width="120px"]

### Example on Attribute Selector

```
<html>
  <head>
    <title>CSS - Attribute selector</title>
    <style type="text/css">
      /* selecting element based on the attribute value */
      img[width='120px']
      {
        border: 4px solid red;
      }
    </style>
  </head>
  <body>
    <h1>Attribute Selector</h1>
    
    
    
    
    
    
    
    
    
  </body>
</html>
```

**Note:** Place "img1.jpg", "img2.jpg", "img3.jpg", "img4.jpg", "img5.jpg", "img6.jpg", "img7.jpg", "img8.jpg", "img9.jpg" in the current folder.

### Hover Selector

- It applies the style only when the user places the mouse pointer on the element, at run time.
- It automatically removes the style, if the mouse pointer is moved outside the element.
- It is also called as "pseudo class". Whenever the selector starts with ":", it is called as "pseudo class".
- Pseudo = unrealistic

**Syntax:** tag:hover

**Example:** p:hover

### Example on Hover Selector

```
<html>
  <head>
    <title>CSS - Hover</title>
    <style type="text/css">
      p
      {
        background-color: skyblue;
      }
      p:hover
```

```
<html>
  <head>
    <style>
      h1:hover {
        background-color: hotpink;
      }
    </style>
  </head>
  <body>
    <h1>hover</h1>
    <p>para 1</p>
    <p>para 2</p>
    <p>para 3</p>
    <p>para 4</p>
  </body>
</html>
```

### Focus Selector

- It applies the style only when the focus (cursor) is inside the element.
- It automatically removes the style when the cursor gets out of the element.
- It is also called as “pseudo class”.

**Syntax:** tag:focus

**Example:** input:focus

### Example on Focus Selector

```
<html>
  <head>
    <title>CSS - Focus psuedo class</title>
    <style type="text/css">
      input:focus {
        border: 4px solid red;
      }
    </style>
  </head>
  <body>
    <h1>focus psuedo class</h1>
    <input type="text"><br>
    <input type="text"><br>
  </body>
</html>
```

### Universal Selector

- It selects all the tags in the web page (including <html>, <head>, <body> etc.).

- It is used to apply global styles.

**Syntax:** \*

**Example:** \*

### Example on Universal Selector

```
<html>
  <head>
    <title>CSS - Universal selector</title>
    <style type="text/css">
      /* selects all tags in the web page */
      *
      {
        font-family: 'Segoe UI';
      }
    </style>
  </head>
  <body>
    <h1>Universal Selector</h1>
    <p>para 1</p>
    <input type="text"><br>
    <input type="text"><br>
    <input type="text"><br>
    <input type="text"><br>
  </body>
</html>
```

### CSS Style Precedence

- The css styles are applied in the following order (lowest priority to highest priority).
- The higher priority style overrides the same property's value of the lower priority.
  1. Browser default style
  2. Tag Selector
  3. Direct Child Selector
  4. Child Selector
  5. Class Selector
  6. Attribute Selector
  7. ID Selector

**Note:** “!important” is used to override the “style precedence”.

### Example on Style Precedence

```
<html>
  <head>
    <title>Style Precedence</title>
    <style type="text/css">
      p
```

```
<style type="text/css">
  p
  {
    color: blue;
  }
  div>p
  {
    color: red;
  }
  .c1
  {
    color: green;
  }
  #p1
  {
    color: pink;
  }
</style>
</head>
<body>
<div>
  <p id="p1" class="c1">para 1</p>
</div>
</body>
</html>
```

#### Example on important:

```
<html>
<head>
  <title>important</title>
  <style type="text/css">
    p
    {
      color: blue !important;
    }
    div>p
    {
      color: red;
    }
    .c1
    {
      color: green;
    }
    #p1
    {
      color: pink;
    }
  </style>
</head>
<body>
<div>
  <p id="p1" class="c1">para 1</p>
</div>
```

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

## CSS Realtime Examples

### Table Styles

```
<html>  
  <head>  
    <title>CSS - Table Styles</title>  
    <style type="text/css">  
      #table1  
      {  
        background-color: #003399;  
        font-size: 30px;  
        font-family: Tahoma';  
      }  
      #table1 tr  
      {  
        background-color: #33ccff;  
      }  
      #table1 tr:nth-child(1)  
      {  
        background-color: #ff0099;  
      }  
      #table1 tr td  
      {  
        padding: 5px;  
      }  
      #table1 tr th  
      {  
        padding: 10px;  
      }  
      #table1 tr:hover  
      {  
        background-color: #00ffcc;  
        cursor: pointer;  
      }  
    </style>  
  </head>  
  <body>  
    <table id="table1">  
      <tr>  
        <th>Name</th>  
        <th>Email</th>  
      </tr>  
      <tr>  
        <td>Scott</td>  
        <td>scott@gmail.com</td>  
      </tr>  
      <tr>
```

```
<td>Allen</td>
<td>allen@gmail.com</td>
</tr>
<tr>
<td>Jones</td>
<td>jones@gmail.com</td>
</tr>
</table>
</body>
</html>
```

### Hyperlink Styles

- It is used to apply styles to hyperlinks.

**a:link:** Applies styles to unvisited hyperlinks.  
**a:hover:** Applies styles to hyperlinks, when we place mouse pointer on it.  
**a:active:** Applies styles to hyperlinks, when we click and hold it.  
**a:visited:** Applies styles to visited hyperlinks.

### Example on Hyperlink Styles

```
<html>
<head>
<title>CSS - Links</title>
<style>
/* unvisited link */
a:link
{
    background-color: darkgreen;
    font-size: 20px;
    color: white;
    padding: 2px;
    text-decoration: none;
    font-weight: bold;
}

/* mouse over link */
a:hover
{
    background-color: yellow;
    color: darkgreen;
    text-decoration: underline;
}

/* selected link */
a:active
{
    background-color: darkred;
    color: cyan;
}
```

```
/* visited link */
a:visited
{
    background-color: black;
    color: yellow;
}
</style>
</head>
<body>
    <a href="http://www.google.com">Google</a>
    <a href="http://www.facebook.com">Facebook</a>
    <a href="http://www.twitter.com">Twitter</a>
</body>
</html>
```

### Menubar

- Menubar is a collection of hyperlinks arranged vertically / horizontally.

```
<html>
    <head>
        <title>CSS - Menu Bar</title>
        <style type="text/css">
            .menubar
            {
                background-color: #00ffcc;
                height: 40px;
                font-size: 24px;
                font-family: Tahoma;
                width: 800px;
                margin: auto;
            }
            .menubar ul
            {
                list-style-type: none;
                padding: 0px;
            }
            .menubar ul li
            {
                display: inline;
                width: 200px;
                float: left;
                height: 40px;
                text-align: center;
            }
            .menubar ul li a
            {
                line-height: 40px;
                text-decoration: none;
            }
        </style>
    </head>
    <body>
        <div class="menubar">
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">About</a></li>
                <li><a href="#">Services</a></li>
                <li><a href="#">Contact</a></li>
            </ul>
        </div>
    </body>
</html>
```

```
.menubar ul li:hover
{
    background-color: #33cc99;
    cursor: pointer;
    text-decoration: underline;
}
</style>
</head>
<body>
<div class="menubar">
<ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Contact</a></li>
    <li><a href="#">Services</a></li>
</ul>
</div>
</body>
</html>
```

### Login Form

- Login form contains username and password.

```
<!DOCTYPE html>
<html>
<head>
<title>Login</title>
<style>
.login
{
    border: 1px solid blue;
    font-size: 30px;
    border-radius: 10px;
    width: 800px;
    margin: auto;
    box-shadow: 3px 3px 3px 2px black;
    margin-top: 200px;
}
.login input
{
    font-size: 30px;
}
.login button
{
    font-size: 30px;
}
.c1
{
    margin-bottom: 10px;
    margin-left: 3px;
}
```

```
margin-right: 3px;
margin-top: 3px;
height: 45px;
}
.c3
{
width: 250px;
float: left;
height: 45px;
text-align: right;
}
.c4
{
width: 500px;
float: left;
height: 45px;
}
.c2
{
width: 500px;
margin-left: 250px;
margin-bottom: 10px;
}
</style>
</head>
<body>
<div class="login">
<form>

<div class="c1">
<div class="c3">
<label for="txtEmail">Email:</label>
</div>
<div class="c4">
<input type="text" id="txtEmail" name="email">
</div>
</div>

<div class="c1">
<div class="c3">
<label for="txtPassword">Password:</label>
</div>
<div class="c4">
<input type="password" name="password" id="txtPassword">
</div>
</div>

<div class="c2">
<input type="checkbox" id="chkRememberMe" value="yes">
<label for="chkRememberMe">Remember Me</label>
</div>
```

```
<div class="c2">
  <button>Login</button>
</div>
</form>
</div>
</body>
</html>
```

## Advanced CSS

### Border Radius

- It is used to apply rounded corners for any html element.
- Syntax: border-radius: pixels;
- Example: border-radius: 10px;

### Border Radius - Cornerwise

- Syntax: border-radius: topleft topright bottomright bottomleft;
- Example: border-radius: 10px 20px 15px 5px;

### Example on Border Radius

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS 3 - Border Radius</title>
  <style type="text/css">
    #div1
    {
      border: 2px solid #a1a1a1;
      padding: 10px 40px;
      background-color: pink;
      width: 300px;
      border-radius: 40px;
    }
  </style>
</head>
<body>
  <div id="div1">The border-radius property allows you to add rounded corners to elements.</div>
</body>
</html>
```

### Example 2 on Border Radius

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS 3 - Border Radius</title>
  <style>
```

```
input
{
    border: 2px solid #alala1;
    padding: 5px;
    background-color: green;
    color: white;
    border-radius: 25px;
}
</style>
</head>
<body>
<input type="text">
<input type="text">
</body>
</html>
```

### Example 3 on Border Radius

```
<!DOCTYPE html>
<html>
<head>
<title>CSS 3 - Border Radius</title>
<style type="text/css">
body
{
    font-family: 'Tahoma';
    font-size: 30px;
}
#p1
{
    border: 2px solid #alala1;
    padding-left: 25px;
    background-color: green;
    color: white;
    width: 300px;
    /* Syntax: border-radius: topLeft topRight bottomRight bottomLeft */
    border-radius: 5px 45px 5px 45px;
}
</style>
</head>
<body>
<p id="p1">
    The border-radius property allows you to add rounded corners to elements.
</p>
</body>
</html>
```

## Shadow

### Box-Shadow

- It is used to apply shadow for the element.

- **Syntax:** box-shadow: HorizontalPosition VerticalPosition BlurRadius Spread ShadowColor;
- **Example:** box-shadow: 5px 5px 5px 2px red;

### Text-Shadow

- It is used to apply shadow for the text of the element.
- **Syntax:** text-shadow: HorizontalPosition VerticalPosition BlurRadius ShadowColor;
- **Example:** text-shadow: 2px 2px 5px 1px red;

### Example on Shadow

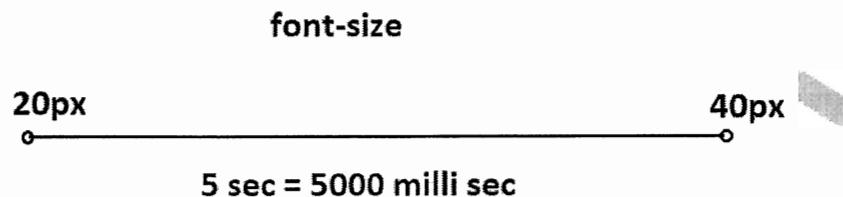
```
<!DOCTYPE html>
<html>
<head>
<title>CSS 3 - Shadow</title>
<style type="text/css">
#div1
{
    width: 300px;
    height: 100px;
    background-color: yellow;
    box-shadow: 15px 15px 10px 5px darkgreen;
    text-shadow: 5px 5px 5px red;
}
#txt1
{
    box-shadow: 15px 15px 10px 5px darkgreen;
    text-shadow: 5px 5px 5px red;
    font-size: 30px;
}
</style>
</head>
<body>
    <div id="div1">CSS 3 shadow example</div><br><br>
    <input type="text" id="txt1">
</body>
</html>
```

### Transitions

- Transition is a process of changing a CSS property's value gradually, based on the specified no. of seconds.
- Transitions support only pixels based color based properties. Ex: width, height, opacity, font-size, border-width, background-color, color, border-color etc.
- **Syntax:**

```
selector
{
    property: startvalue;
    transition: property seconds;
}
selector:hover
{
    property: endvalue;
}
```

Example:



#### Example on Transitions - Width

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS 3 - Transitions - Width</title>
    <style type="text/css">
        #div1
        {
            width: 300px;
            height: 100px;
            background-color: darkred;
            color: yellow;
            transition: width 2s;
        }
        #div1:hover
        {
            width: 600px;
        }
    </style>
</head>
<body>
    <div id="div1">hover me</div>
</body>
</html>
```

#### Example on Transitions - Height

```
<!DOCTYPE html>
```

```
<html>
<head>
  <title>CSS 3 - Transitions - Height</title>
  <style type="text/css">
    #div1
    {
      width: 300px;
      height: 100px;
      background-color: darkred;
      color: yellow;
      transition: height 2s;
    }
    #div1:hover
    {
      height: 300px;
    }
  </style>
</head>
<body>
  <div id="div1">hover me</div>
</body>
</html>
```

### Example on Transitions - Font-size

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS 3 - Transitions - Font-Size</title>
  <style type="text/css">
    #div1
    {
      width: 200px;
      height: 200px;
      background-color: darkgreen;
      position: relative;
      color: yellow;
      font-size: 30px;
      transition: font-size 3s;
    }
    #div1:hover
    {
      font-size: 60px;
    }
  </style>
</head>
<body>
  <div id="div1">hover me</div>
</body>
</html>
```

---

### Example on Transitions - Multiple Properties

---

```
<!DOCTYPE html>
<html>
<head>
<title>CSS 3 - Transitions - Multiple Properties</title>
<style type="text/css">
#div1
{
    width: 100px;
    height: 100px;
    background-color: darkgreen;
    position: relative;
    color: yellow;
    font-size: 25px;
    transition: background-color 2s, font-size 1s, width 2s;
}
#div1:hover
{
    background-color: blue;
    font-size: 50px;
    width: 400px;
}
</style>
</head>
<body>
<div id="div1">hover me</div>
</body>
</html>
```

---

### Example on Transitions - Opacity

---

```
<!DOCTYPE html>
<html>
<head>
<title>CSS 3 - Transitions - Opacity</title>
<style type="text/css">
#div1
{
    width: 300px;
    height: 300px;
    background-color: darkgreen;
    position: relative;
    color: yellow;
    opacity: 1;
    transition: opacity 1.5s;
}
#div1:hover
{
    opacity: 0;
}
</style>
</head>
```

```
<body>
  <div id="div1">hover me</div>
</body>
</html>
```

### Example on Transitions - Position

```
<!DOCTYPE html>
<html>
<head>
  <title>CSS 3 - Transitions - Position</title>
  <style type="text/css">
    #div1
    {
      width: 200px;
      height: 200px;
      background-color: darkgreen;
      color: yellow;
      position: relative;
      left: 0px;
      top: 0px;
      transition: left 1.5s, top 1.5s;
    }
    #div1:hover
    {
      left: 50px;
      top: 50px;
    }
  </style>
</head>
<body>
  <div id="div1">hover me</div>
</body>
</html>
```

### Transformations

- Transformations are used to display the element in a different visual dimension.
- Types of transformations:
  1. Rotate Transformation
  2. Skew Transformation
  3. Scale Transformation
  4. Translate Transformation

#### 1. Rotate Transformation

- It is used to rotate the element, based on the specifies no. of degrees. Ex: 45deg.

**Syntax:** `transform: rotate(degrees);`

**Example:** `transform: rotate(45deg);`

### Example on Rotate Transformation

```
<html>
<head>
    <title>CSS 3 - Transformations - Rotate Transformation</title>
    <style type="text/css">
        #div1
        {
            width: 200px;
            height: 200px;
            background-color: #00CCFF;
            margin-left: 150px;
            cursor: pointer;
            box-shadow: 10px 10px 10px 10px #ff0000;
        }
        #div1:hover
        {
            transform: rotate(45deg); /* 0 to 360 deg */
        }
    </style>
</head>
<body>
    <h3>CSS 3 Transformations - Rotate Transformation</h3>
    <div id="div1">Hello</div>
</body>
</html>
```

### 2. Skew Transformation

- It warps the element.

Syntax: transform: skew(degrees);

Example: transform: skew(30deg);

### Example on Skew Transformation

```
<html>
<head>
    <title>CSS 3 - Transformations - Skew Transformation</title>
    <style type="text/css">
        #div1
        {
            width: 200px;
            height: 200px;
            background-color: #00CCFF;
            margin-left: 150px;
            cursor: pointer;
            box-shadow: 10px 10px 10px 10px #ff0000;
        }
        #div1:hover
        {
            background-color: #0099FF;
            transform: skew(30deg); /* 0 to 360 deg */
        }
    </style>
</head>
<body>
    <h3>CSS 3 Transformations - Skew Transformation</h3>
    <div id="div1">Hello</div>
</body>
</html>
```

```
        }
    </style>
</head>
<body>
    <h3>CSS 3 Transformations - Skew Transformation</h3>
    <div id="div1">Hello</div>
</body>
</html>
```

### 3. Scale Transformation

- It shows the element in large size / small size, visually.

Syntax: transform: scale(number);

Example: transform: scale(2);

1 = 100% size

2 = 200% size

0.5 = 50% size

1.5 = 150% size

...

#### Example on Scale Transformation

```
<html>
<head>
    <title>CSS 3 - Transformations - Scale Transformation</title>
    <style type="text/css">
        #div1
        {
            width: 200px;
            height: 200px;
            background-color: #00CCFF;
            margin-left: 150px;
            cursor: pointer;
            box-shadow: 10px 10px 10px 10px #ff0000;
        }
        #div1:hover
        {
            background-color: #0099FF;
            transform: scale(1.5); /* 0 to n */
        }
    </style>
</head>
<body>
    <h3>CSS 3 Transformations - Scale Transformation</h3>
    <div id="div1">Hello</div>
</body>
```

</html>

#### 4. Translate Transformation

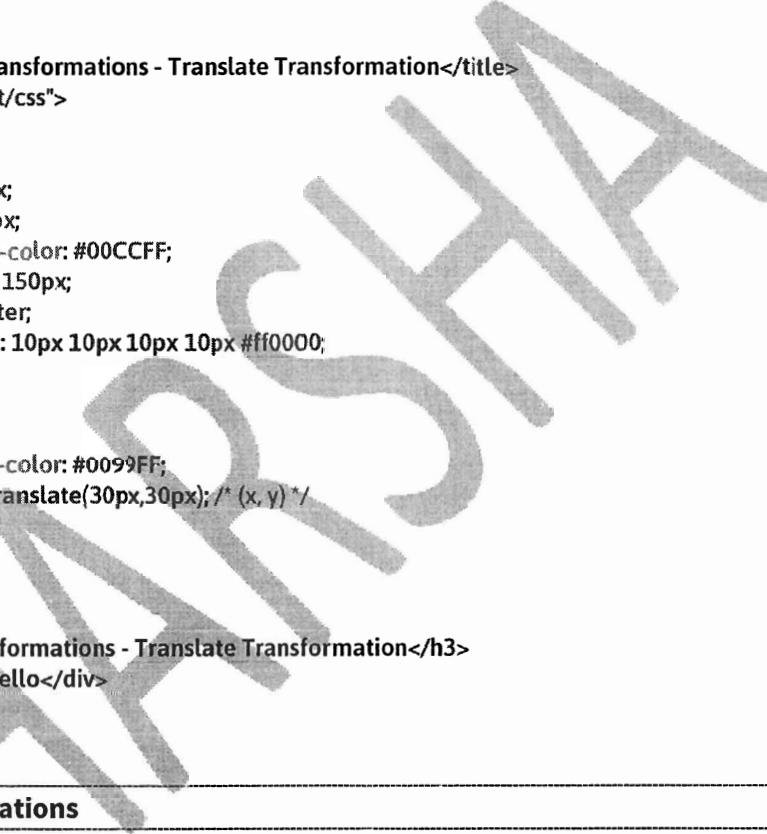
- It changes the visual position of the element.

**Syntax:** transform: translate(x, y);

**Example:** transform: translate(30px, 30px);

#### Example on Translate Transformation

```
<html>
<head>
  <title>CSS 3 - Transformations - Translate Transformation</title>
  <style type="text/css">
    #div1
    {
      width: 200px;
      height: 200px;
      background-color: #00CCFF;
      margin-left: 150px;
      cursor: pointer;
      box-shadow: 10px 10px 10px 10px #ff0000;
    }
    #div1:hover
    {
      background-color: #0099FF;
      transform: translate(30px,30px); /* (x, y) */
    }
  </style>
</head>
<body>
  <h3>CSS 3 Transformations - Translate Transformation</h3>
  <div id="div1">Hello</div>
</body>
</html>
```



#### Multiple Transformations

- We can apply more than one transformation at-a-time.

**Syntax:** transform: rotate(degrees) scale(n) translate(x, y) skew(degrees);

**Example:** transform: rotate(40deg) scale(1.5) translate(30px, 30px) skew(20deg);

#### Example on Multiple Transformations

```
<html>
<head>
  <title>CSS 3 - Transformations - Multiple Transformations</title>
  <style type="text/css">
    #div1
    {
      width: 200px;
```

```
height: 200px;
background-color: #00CCFF;
margin-left: 150px;
margin-top: 100px;
cursor: pointer;
box-shadow: 10px 10px 10px 10px #ff0000;
}
#div1:hover
{
background-color: #0099FF;
transform: rotate(30deg) scale(1.5);
}
</style>
</head>
<body>
<h3>CSS 3 Transformations - Multiple Transformations</h3>
<div id="div1">Hello</div>
</body>
</html>
```

#### transform-origin

- This property specifies the fixed point for rotate transformation and scale transformation etc.
- **Syntax:** transform-origin: top left | top center | top right | center left | center center | center right | bottom left | bottom center | bottom right;
- **Example:** transform-origin: top left;

#### Example on Transform Origin

```
<html>
<head>
<title>CSS 3 - Transformations - Transform Origin</title>
<style type="text/css">
#div1
{
width: 200px;
height: 200px;
background-color: #00CCFF;
margin-left: 150px;
margin-top: 100px;
cursor: pointer;
box-shadow: 10px 10px 10px 10px #ff0000;
}
#div1:hover
{
background-color: #0099FF;
transform: rotate(30deg);
transform-origin: top left;
}
</style>
```

```
</head>
<body>
  <h3>CSS 3 Transformations - Transform Origin</h3>
  <div id="div1">Hello</div>
</body>
</html>
```

### Example on Transformation with Transition

```
<html>
<head>
  <title>CSS 3 - Transformations - with Transition</title>
  <style type="text/css">
    #div1
    {
      width: 200px;
      height: 200px;
      background-color: #00CCFF;
      margin-left: 150px;
      cursor: pointer;
      box-shadow: 10px 10px 10px 10px #ff0000;
      transition: transform 0.8s;
    }
    #div1:hover
    {
      background-color: #0099FF;
      transform: scale(1.5) skew(30deg);
    }
  </style>
</head>
<body>
  <h3>CSS 3 Transformations - Transformation with Transition</h3>
  <div id="div1">Hello</div>
</body>
</html>
```

### Animations

- Animations are “group of transitions”, which will be performed one after another.
- Transition contains two points only (starting point + ending point).
- Animation contains multiple points of milestones.

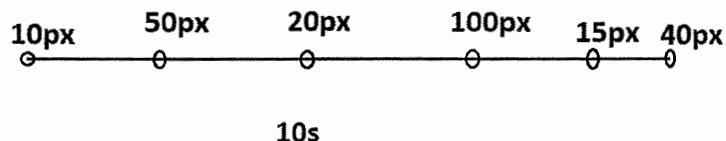
#### Syntax:

```
selector
{
  animation: animationname seconds;
}
@keyframes animationname
{
```

```
0% { property:value; property:value; ... },  
25% { property:value; property:value; ... },  
50% { property:value; property:value; ... },  
75% { property:value; property:value; ... },  
100% { property:value; property:value; ... }  
}
```

Example:

font-size



### Example on Animations

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Animations</title>  
    <style>  
      #div1  
      {  
        background-color: #2eda59;  
        font-size: 35px;  
        width: 300px;  
        height: 220px;  
        margin: 60px;  
      }  
      #div1:hover  
      {  
        cursor: pointer;  
        animation: myanimation 10s;  
      }  
      @keyframes myanimation  
      {  
        0% { transform: translate(0px, 0px); }  
        25% { transform: translate(50px, 0px); }  
        50% { transform: translate(50px, 100px); }  
        75% { transform: translate(0px, 100px); }  
        100% { transform: translate(0px, 0px); }  
      }  
    </style>  
  </head>  
  <body>
```

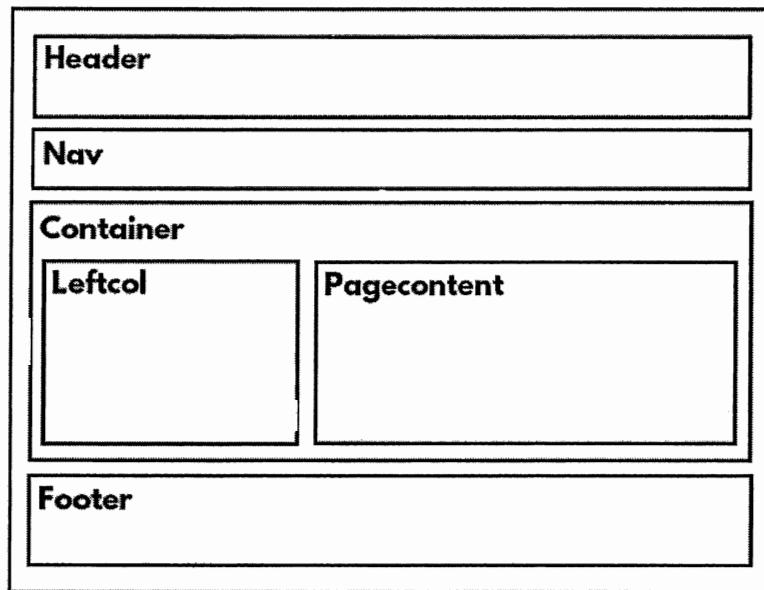
```
<div id="div1">
  
  Hyderabad
</div>
</body>
</html>
```

## Static Page Template

- It is used to create page template (layout).
- **Header:** Contains company logo, website name, login, logout, search etc.
- **Nav:** Contains menubar.
- **Container:** Contains left column and page content.
- **Leftcol:** Contains left side menu.
- **PageContent:** Contains actual content of the page.
- **Footer:** Contains bottom information and links for other related sites.

## Static Page Template

### OuterContainer



### Example on Static Page Template

#### home.html

```
<html>
```

```
<head>
  <title>Home Page</title>
  <style>
    body
    {
      font-size: 20px;
    }

    *
    {
      margin: 0px;
      padding: 0px;
    }

    #outercontainer
    {
      background-color: #ffccff;
      width: 100%;
      margin: auto;
    }

    #header
    {
      background-color: #00ff99;
      height: 200px;
      width: 100%;
    }

    #nav
    {
      background-color: #ccccff;
      height: 80px;
      width: 100%;
    }

    #container
    {
      height: 500px;
      width: 100%;
    }

    #leftcol
    {
      height: 500px;
      background-color: #66ccff;
      width: 20%;
      float: left;
    }

    #pagecontent
    {
```

```
height: 500px;
background-color: #ccffcc;
width: 80%;
float: left;
}

#footer
{
height: 50px;
background-color: #ff0099;
width: 100%;
clear: left;
}
</style>
</head>
<body>
<div id="outercontainer">
<div id="header">
Header
</div>

<div id="nav">
Nav
</div>

<div id="container">
<div id="leftcol">
Leftcol
</div>
<div id="pagecontent">
Page content
</div>
</div>

<div id="footer">
Footer
</div>
</div>
</body>
</html>
```

#### Example on Page Navigation using Static Page Template

##### home.html

```
<html>
<head>
<title>Home</title>
<link href="StyleSheet.css" rel="stylesheet">
</head>
<body>
```

```
<div id="outercontainer">
  <div id="header">
    Header
  </div>

  <div id="nav">
    <a href="home.html">Home</a>
    <a href="about.html">About</a>
    <a href="contact.html">Contact</a>
  </div>

  <div id="container">
    <div id="leftcol">
      Leftcol
    </div>
    <div id="pagecontent">
      Home Page content
    </div>
  </div>

  <div id="footer">
    Footer
  </div>
</div>
</body>
</html>
```

### [about.html](#)

```
<html>
  <head>
    <title>About</title>
    <link href="StyleSheet.css" rel="stylesheet">
  </head>
  <body>
    <div id="outercontainer">
      <div id="header">
        Header
      </div>

      <div id="nav">
        <a href="home.html">Home</a>
        <a href="about.html">About</a>
        <a href="contact.html">Contact</a>
      </div>

      <div id="container">
        <div id="leftcol">
          Leftcol
        </div>
```

```
<div id="pagecontent">
    About Page content
</div>
</div>

<div id="footer">
    Footer
</div>
</div>

</body>
</html>
```

### contact.html

```
<html>
    <head>
        <title>Contact</title>
        <link href="StyleSheet.css" rel="stylesheet">
    </head>
    <body>
        <div id="outercontainer">
            <div id="header">
                Header
            </div>

            <div id="nav">
                <a href="home.html">Home</a>
                <a href="about.html">About</a>
                <a href="contact.html">Contact</a>
            </div>

            <div id="container">
                <div id="leftcol">
                    Leftcol
                </div>
                <div id="pagecontent">
                    Contact Page content
                </div>
            </div>

            <div id="footer">
                Footer
            </div>
        </div>

        </body>
    </html>
```

StyleSheet.css

```
body
{
    font: 20px 'Tahoma';
}

*
{
    margin: 0px;
    padding: 0px;
}

#outercontainer
{
    background-color: #ffccff;
    width: 100%;
    margin: auto;
}

#header
{
    background-color: #00ff99;
    height: 200px;
    width: 100%;
}

#nav
{
    background-color: #ccccff;
    height: 80px;
    width: 100%;
}

#container
{
    height: 500px;
    width: 100%;
}

#leftcol
{
    height: 500px;
    background-color: #66ccff;
    width: 20%;
    float: left;
}

#pagecontent
{
    height: 500px;
```

```
background-color: #ccffcc;
width: 80%;
float: left;
}

#footer
{
height: 50px;
background-color: #ff0099;
width: 100%;
clear: left;
}
```

## Responsive Web Design

### What is Responsive Web Design

- “Responsive Web Design” (RWD) is used to make the web page automatically fit based on the current device resolution.
- It is used to display the content differently on different devices, based on the screen resolution.
- We divide the devices into 4 types, based on the browser width:

Sl. No	Type of Device	Screen width (pixels)	Examples
1	Extra Small Devices	1px to 575px	iPhone 2G, 3G (320px / 480px height)
2	Small Devices	576px to 767px	iPhone 4, 4S (640px width / 960px height) iPhone 5, 5C, 5C, SE (640px width / 1136px) iPhone 6, 6S, 7, 8 (750px width / 1134px height)
3	Medium Devices	768px to 991px	Samsung Note 1 (800px width / 1280px height)
4	Large Devices	992px to 1199px	iPhone 6 Plus (1080px width / 1920px height) iPhone 10 (1125px width / 2436px height)
5	Extra Large Devices	1200px to unlimited	iPhone 7+, 8+ (1242px width / 2208px height) Most-used laptops

### Media Queries

- We use “Media Queries” to create responsive web design.
- The media queries apply the styles only when the current device / browser's width meet the given condition / requirements.

**1. Extra Small devices (1px to 575px):**

```
@media (min-width: 1px) and (max-width: 575px){  
}
```

**2. Small devices (576px to 767px):**

```
@media (min-width: 576pxpx) and (max-width: 767px){  
}
```

**3. Medium devices (767px to 991px):**

```
@media (min-width: 767pxpx) and (max-width: 991px){  
}
```

**4. Large devices (992px to 1199px):**

```
@media (min-width: 992px) and (max-width: 1199px){  
}
```

**5. Extra Large devices (1200px to unlimited):**

```
@media (min-width: 1200px){  
}
```

**View port meta tag**

- This tag tells to the mobile browsers that we are using “responsive web design”, and don’t treat it as pc-based web page.
- Without viewport meta tag, the mobile browsers treat the web page as pc-based web page and apply the pc-based media query.

- With viewport meta tag, the mobile browsers apply the appropriate “mobile-based media query” to the web page.

### Example on Responsive Web Design

#### home.html

```
<html>
  <head>
    <title>Responsive Web Design</title>
    <meta name="viewport" content="width=device-width">
    <style>
      body
      {
        font-family: Verdana;
        font-size: 20px;
      }
      *
      {
        margin: 0px;
        padding: 0px;
      }
      .div1
      {
        background-color: #2ef314;
        height: 50px;
      }
      .div2
      {
        background-color: #14abbe;
        height: 50px;
      }
      .div3
      {
        background-color: #0cb35f;
        height: 50px;
      }
      .div4
      {
        background-color: #149af3;
        height: 50px;
      }
      .div5
      {
        background-color: #f3b814;
        height: 50px;
      }
      .div6
      {
        background-color: #eb3948;
      }
    </style>
  </head>
  <body>
    <div class="div1">Div 1</div>
    <div class="div2">Div 2</div>
    <div class="div3">Div 3</div>
    <div class="div4">Div 4</div>
    <div class="div5">Div 5</div>
    <div class="div6">Div 6</div>
  </body>
</html>
```

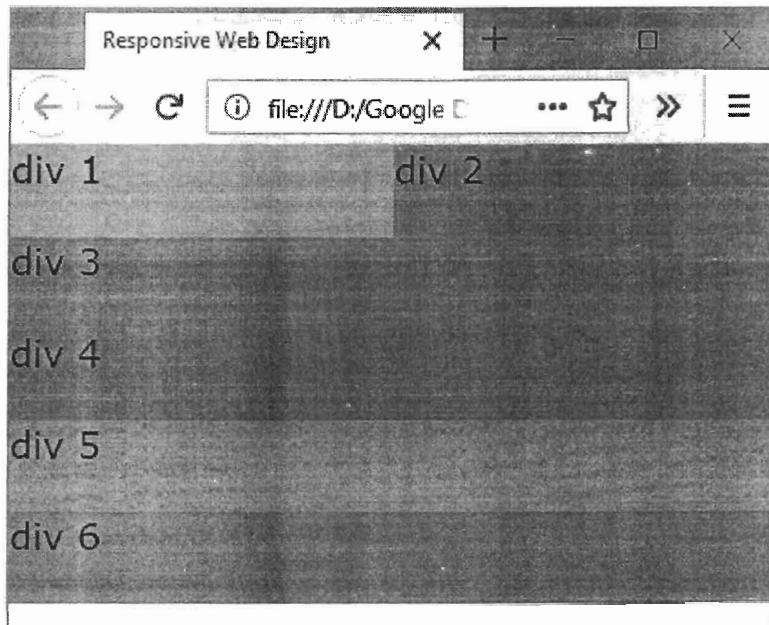
```
height: 50px;
}
@media (min-width:100px) and (max-width:575px)
{
.div1
{
width: 50%;
float: left;
}
.div2
{
width: 50%;
float: left;
}
.div3
{
width: 100%;
clear: left;
}
.div4
{
width: 100%;
}
.div5
{
width: 100%;
}
.div6
{
width: 100%;
}
}
@media (min-width:576px) and (max-width:767px)
{
.div1
{
width: 50%;
float: left;
}
.div2
{
width: 50%;
float: left;
}
.div3
{
width: 50%;
clear: left;
float: left;
}
.div4
```

```
{  
  width: 50%;  
  float: left;  
}  
.div5  
{  
  width: 50%;  
  clear: left;  
  float: left;  
}  
.div6  
{  
  width: 50%;  
  float: left;  
}  
}  
@media (min-width:768px) and (max-width:991px)  
{  
.div1  
{  
  width: 33%;  
  float: left;  
}  
.div2  
{  
  width: 33%;  
  float: left;  
}  
.div3  
{  
  width: 34%;  
  float: left;  
}  
.div4  
{  
  width: 33%;  
  clear: left;  
  float: left;  
}  
.div5  
{  
  width: 33%;  
  float: left;  
}  
.div6  
{  
  width: 34%;  
  float: left;  
}  
}  
@media (min-width:992px) and (max-width:1199px)
```

```
{  
  .div1  
  {  
    width: 25%;  
    float: left;  
  }  
  .div2  
  {  
    width: 25%;  
    float: left;  
  }  
  .div3  
  {  
    width: 25%;  
    float: left;  
  }  
  .div4  
  {  
    width: 25%;  
    float: left;  
  }  
  .div5  
  {  
    width: 50%;  
    clear: left;  
    float: left;  
  }  
  .div6  
  {  
    width: 50%;  
    float: left;  
  }  
}  
@media (min-width:1200px)  
{  
  .div1  
  {  
    width: 17%;  
    float: left;  
  }  
  .div2  
  {  
    width: 17%;  
    float: left;  
  }  
  .div3  
  {  
    width: 17%;  
    float: left;  
  }  
  .div4  
  {  
    width: 17%;  
    float: left;  
  }  
}
```

```
        {
            width: 17%;
            float: left;
        }
        .div5
        {
            width: 16%;
            float: left;
        }
        .div6
        {
            width: 16%;
            float: left;
        }
    }
</style>
</head>
<body>
<div class="container">
<div class="div1">
    div 1
</div>
<div class="div2">
    div 2
</div>
<div class="div3">
    div 3
</div>
<div class="div4">
    div 4
</div>
<div class="div5">
    div 5
</div>
<div class="div6">
    div 6
</div>
</body>
</html>
```

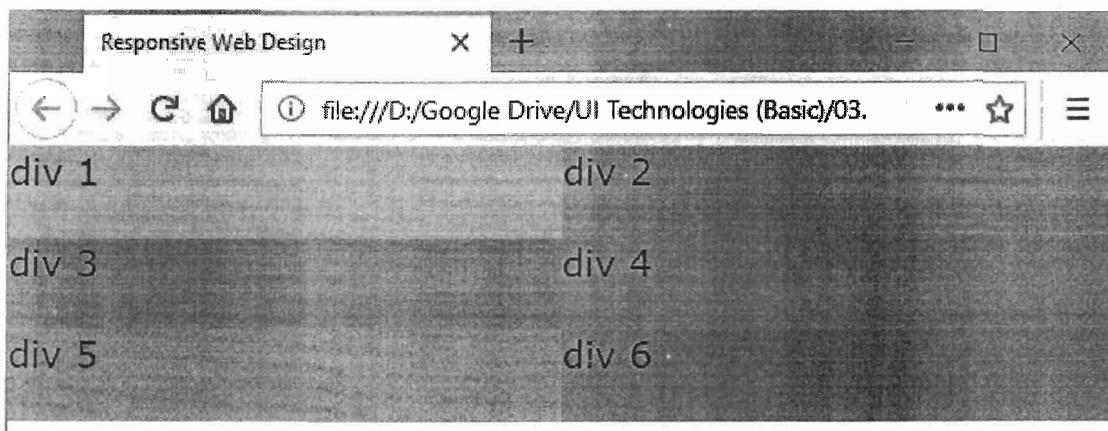
Extra Small Devices:



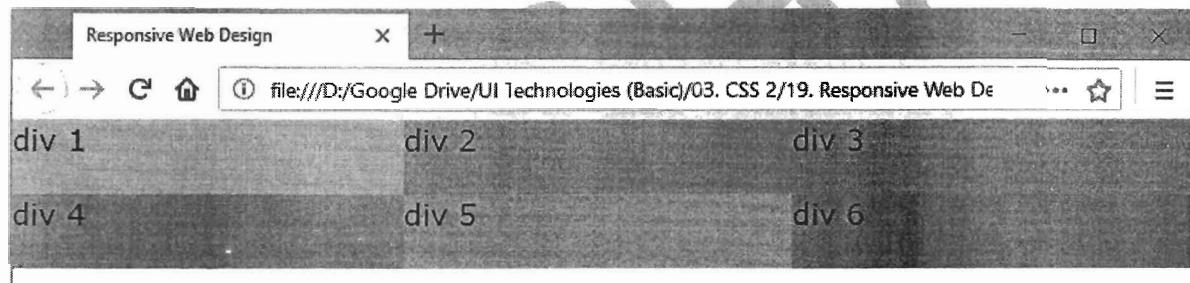
Harsha Vardhan (UI Expert) at Durgasoft

Udemy courses at 650 INR - <https://www.udemy.com/courses/search/?q=harsha>

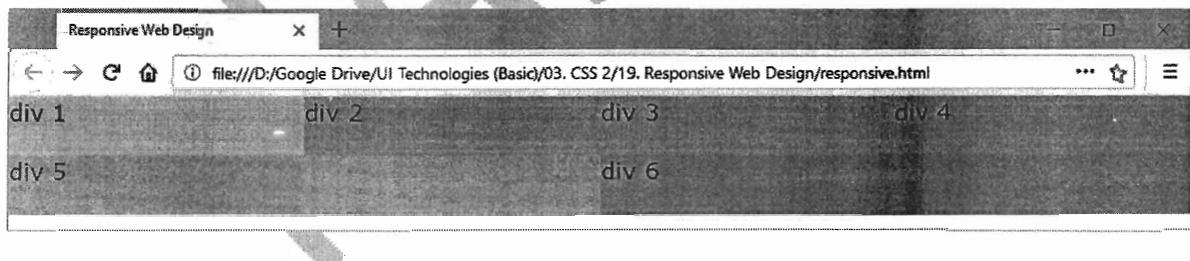
Small Devices:



Medium Devices:



Large Devices:



Extra Large Devices:



# BOOTSTRAP

## Fundamentals of Bootstrap

### Introduction to Bootstrap

- “Bootstrap” is a “CSS framework”, developed by “Twitter” company, which is a collection of ready-made “css classes”, which can be used in web pages to apply styles to the elements easily.
- Bootstrap can be used without css knowledge also. To customize the bootstrap styles, CSS knowledge is required.
- Bootstrap provides responsive web design by default.
- Bootstrap was developed based on “jQuery” and “Popper”.
- jQuery is a JavaScript library, which provides a set of functions to perform DOM manipulations easily.
- Popper is a JavaScript library, which provides a set of functions to display popup messages.

### Preparing Environment for Bootstrap

#### Downloading Bootstrap

- Go to “<http://getbootstrap.com>”.
- Select the latest version in the dropdownlist (Ex: v4.1). It by default, selects the latest version.
- Click on “Download”.
- Click on “Download” again under “Compiled CSS and JS”.
- Download the file “bootstrap-4.1.1.zip”.
- Go to the downloaded location, right click on the “bootstrap-4.1.1.zip” file and click on “Extract All”.
- After extracting, you will get extracted folder.
- Copy the following files from “extracted folder (bootstrap-4.1.1)”, into “application folder (c:\bootstrap)”.
  - css\bootstrap.css
  - js\bootstrap.js

#### Downloading jQuery

- Go to “<http://jquery.com>”.
- Click on “Download jQuery 3.3.1” (version number may vary).
- Click on “Download the uncompressed, development jQuery 3.3.1”.
- If required, press “Ctrl+S” to save the file. You will get a file “jquery-3.3.1.js”.
- Copy and paste “jquery-3.3.1.js” file from Downloaded location into “c:\bootstrap” folder.

### Downloading Popper

---

- Go to "<https://unpkg.com/popper.js>".
- If required, press "Ctrl+S" to save the file. You will get a file "popper.js".
- Copy and paste "popper.js" file from Downloaded location into "c:\bootstrap" folder.

### Importing Bootstrap

---

```
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
```

### Importing Bootstrap

---

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

- The "viewport meta tag" is used to set the actual width of the device as width of the web page.
- It is must to make the web page responsive.
- It also sets the initial zoom to "1" (actual size).

### First Example on Bootstrap (c:\bootstrap\first.html)

---

```
<html>
  <head>
    <title>Bootstrap</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container">
      <h1>My First Bootstrap Page</h1>
      <p>This is some text.</p>
    </div>
  </body>
</html>
```



### **“container” class**

- It acts as “outer container” for the entire page.
- It makes the web page responsive.
- The entire content of the page should be inside the “container”.
- It provides margin left and margin right for the page.

#### **Syntax:**

```
<div class="container">  
    any content  
</div>
```

#### **Example on Container**

```
<html>  
  <head>  
    <title>Container</title>  
    <meta charset="utf-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <link rel="stylesheet" href="bootstrap.css">  
    <script src="jquery-3.3.1.js"></script>  
    <script src="popper.js"></script>  
    <script src="bootstrap.js"></script>  
  </head>  
  <body>  
    <div class="container" style="background-color:lightgreen">  
      <h1>My First Bootstrap Page</h1>  
      <p>This is some text.</p>  
    </div>  
  </body>  
</html>
```



### **"container-fluid" class**

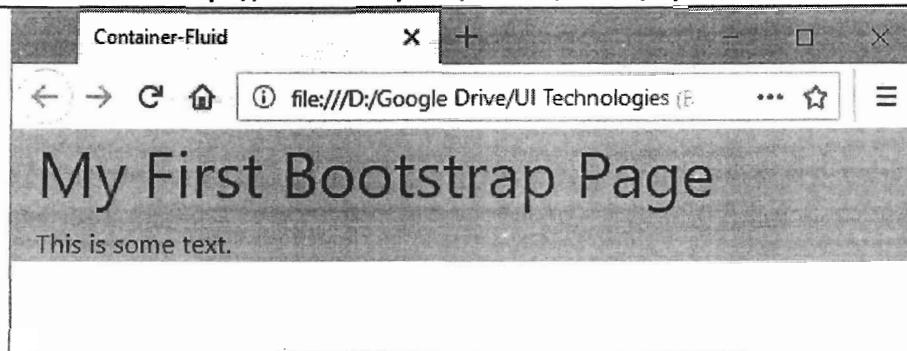
- It also acts as "outer container" for the entire page.
- It also makes the web page responsive.
- The entire content of the page should be inside the "container-fluid".
- It removes margin left and margin right for the page. It makes the content occupy the full available width of the web page.

#### **Syntax:**

```
<div class="container">  
    any content  
</div>
```

#### **Example on Container-Fluid**

```
<html>  
  <head>  
    <title>Container-Fluid</title>  
    <meta charset="utf-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <link rel="stylesheet" href="bootstrap.css">  
    <script src="jquery-3.3.1.js"></script>  
    <script src="popper.js"></script>  
    <script src="bootstrap.js"></script>  
  </head>  
  <body>  
    <div class="container-fluid" style="background-color:lightgreen">  
      <h1>My First Bootstrap Page</h1>  
      <p>This is some text.</p>  
    </div>  
  </body>  
</html>
```



## Colors

### Text Colors

- It is used to set colors of the text.
- Based on the requirement, the developer can use any of the available colors.
- If you want other color, you can use CSS.

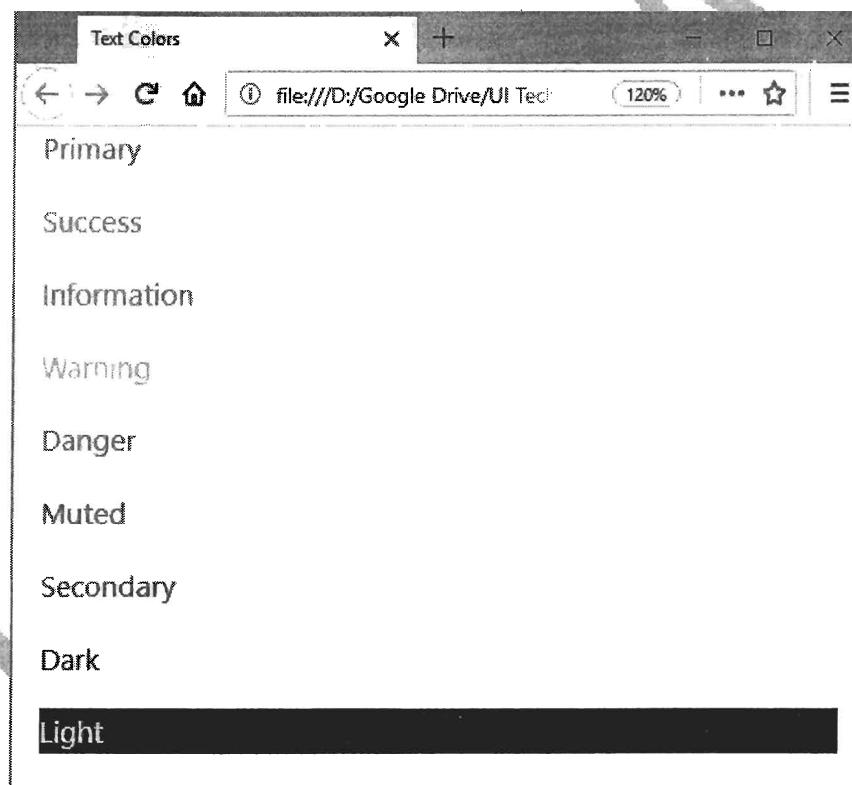
### List of Classes

- text-primary : Blue text color
- text-success : Green text color
- text-info : Light blue text color
- text-warning : Orange text color
- text-danger : Red text color
- text-muted : Grey color
- text-secondary : Darker grey text color
- text-dark : Dark grey text color
- text-light : Light grey text color

### Example on Text Colors

```
<html>
  <head>
    <title>Text Colors</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
```

```
<div class="container">
  <p class="text-primary">Primary</p>
  <p class="text-success">Success</p>
  <p class="text-info">Information</p>
  <p class="text-warning">Warning</p>
  <p class="text-danger">Danger</p>
  <p class="text-muted">Muted</p>
  <p class="text-secondary">Secondary</p>
  <p class="text-dark">Dark</p>
  <p class="text-light" style="background-color:black">Light</p>
</div>
</body>
</html>
```



### Background Colors

- It is used to set background colors of the element.
- Based on the requirement, the developer can use any of the available colors.
- If you want other color, you can use CSS.

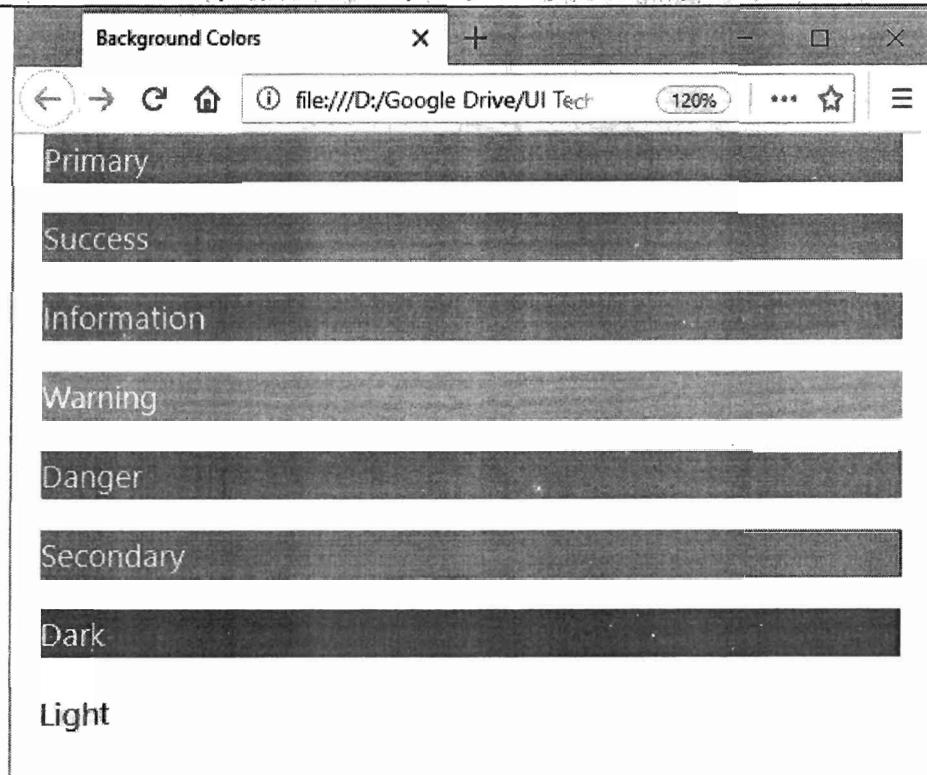
### List of Classes

- bg-primary : Blue background color
- bg-success : Green background color

- bg-info : Light blue background color
- bg-warning : Orange background color
- bg-danger : Red background color
- bg-secondary : Grey background color
- bg-dark : Dark grey background color
- bg-light : Light grey background color

### Example on Background Colors

```
<html>
  <head>
    <title>Background Colors</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container">
      <p class="bg-primary text-white">Primary</p>
      <p class="bg-success text-white">Success</p>
      <p class="bg-info text-white">Information</p>
      <p class="bg-warning text-white">Warning</p>
      <p class="bg-danger text-white">Danger</p>
      <p class="bg-secondary text-white">Secondary</p>
      <p class="bg-dark text-white">Dark</p>
      <p class="bg-light text-dark">Light</p>
    </div>
  </body>
</html>
```



## Text

### Display Headings

- It is used to display headings with thin text and larger font size.

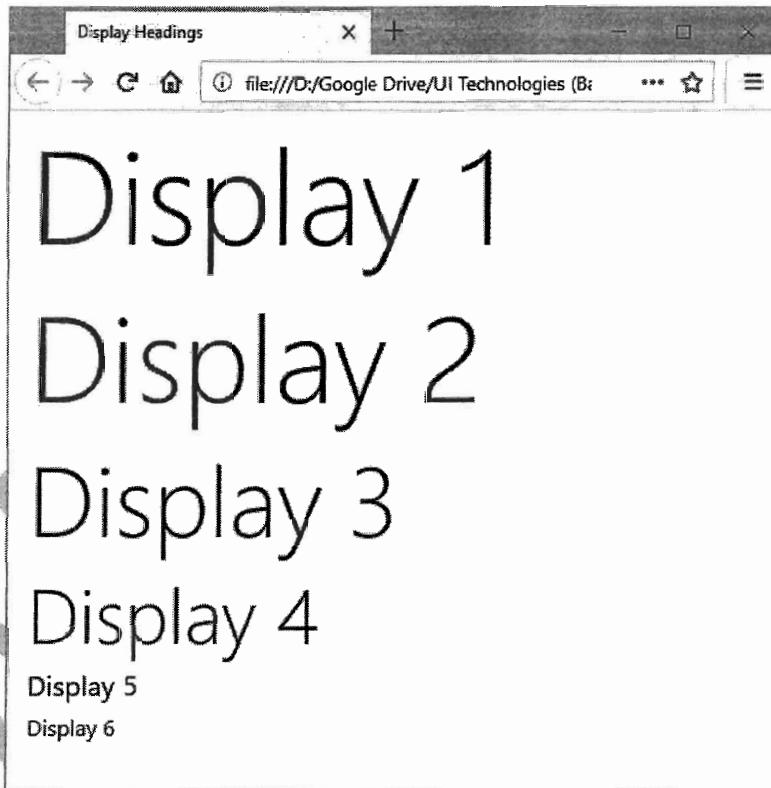
#### List of Classes

- display-1 : Display Heading 1
- display-2 : Display Heading 2
- display-3 : Display Heading 3
- display-4 : Display Heading 4

#### Example on Display Headings

```
<html>
  <head>
    <title>Display Headings</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
```

```
</head>
<body>
<div class="container">
<h1 class="display-1">Display 1</h1>
<h2 class="display-2">Display 2</h2>
<h3 class="display-3">Display 3</h3>
<h4 class="display-4">Display 4</h4>
<h5>Display 5</h5>
<h6>Display 6</h6>
</div>
</body>
</html>
```



### Text Alignment

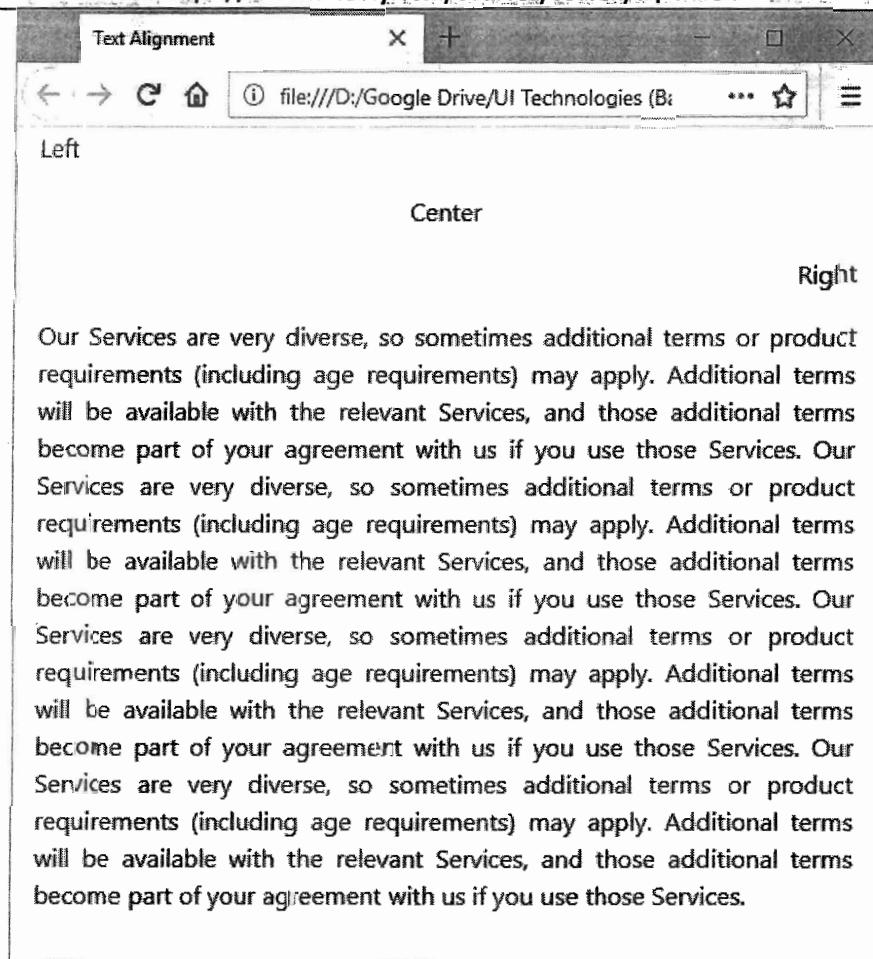
- We can apply text alignment, by using the following bootstrap css classes.
- Default is "left alignment".

### List of Classes

- text-left : Left alignment
- text-center : Center alignment
- text-right : Right alignment
- text-justify : Justify alignment

## Example on Text Alignment

```
<html>
<head>
<title>Text Alignment</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container">
<p class="text-left">Left</p>
<p class="text-center">Center</p>
<p class="text-right">Right</p>
<p class="text-justify">Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services. Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services. Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services. Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
</div>
</body>
</html>
```



### Text Styles

- The following set of bootstrap classes are used to set text styles such as bold, italic, uppercase etc.

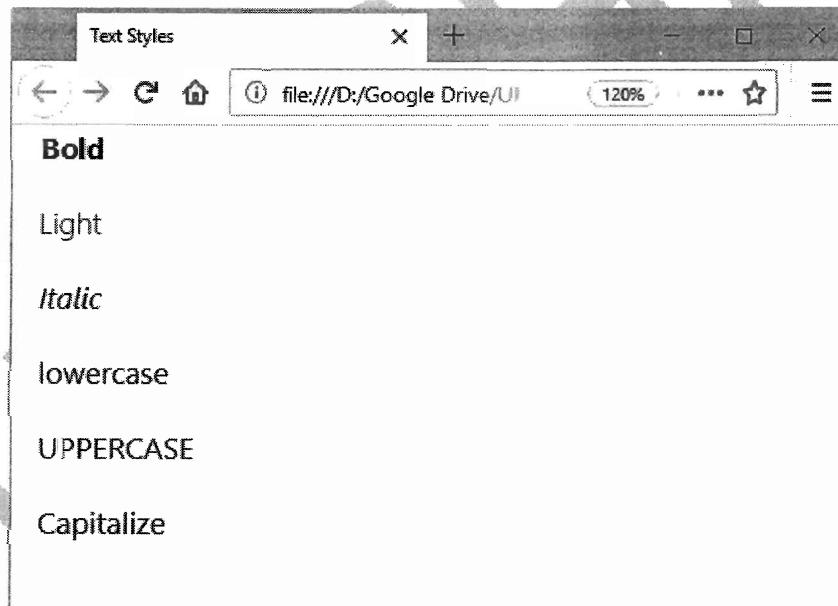
#### List of Classes

- `font-weight-bold` : Bold text
- `font-weight-light` : Light weight text
- `font-italic` : Italic text
- `text-lowercase` : Lowercase
- `text-uppercase` : Uppercase
- `text-capitalize` : Every word's first letter is capital

#### Example on Text Styles

```
<html>
  <head>
    <title>Text Styles</title>
```

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container">
<p class="font-weight-bold">Bold</p>
<p class="font-weight-light">Light</p>
<p class="font-italic">Italic</p>
<p class="text-lowercase">Lowercase</p>
<p class="text-uppercase">Uppercase</p>
<p class="text-capitalize">Capitalize</p>
</div>
</body>
</html>
```



### Lead

- It is used to display a leading paragraph (in larger font size and more line height).

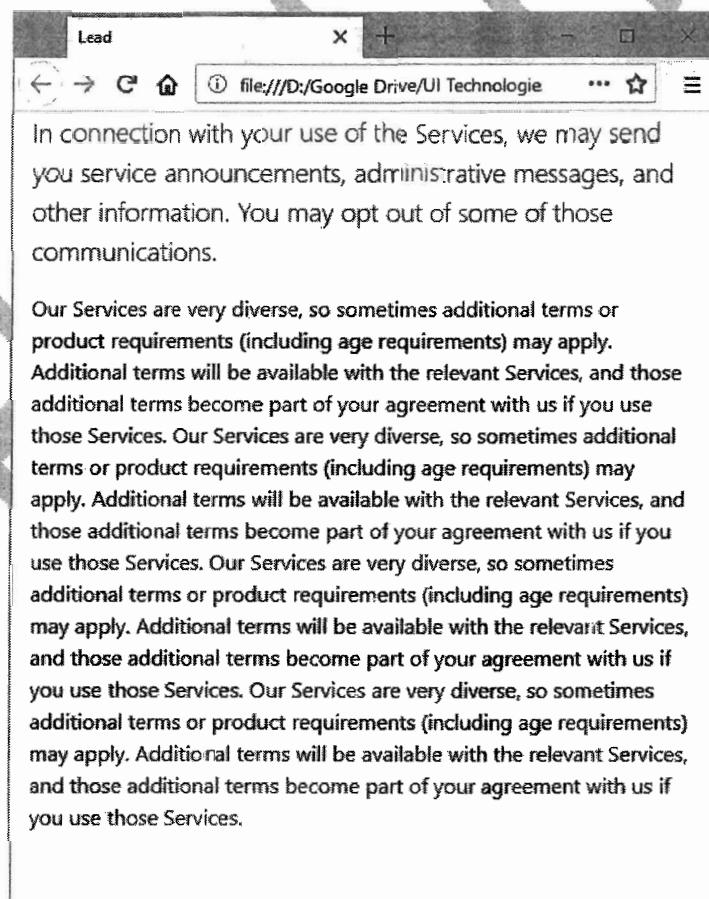
### List of Classes

- lead : Paragraph with larger font size and larger line height

### Example on Lead

```
<html>
<head>
<title>Lead</title>
```

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container">
<p class="lead">In connection with your use of the Services, we may send you service announcements, administrative messages, and other information. You may opt out of some of those communications.</p>
<p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services. Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services....</p>
</div>
</body>
</html>
```



## Visibility

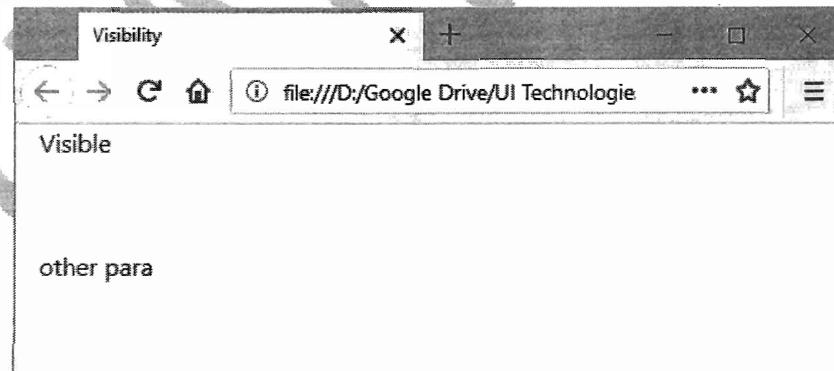
- It is used to show / hide the element in the web page.

### List of Classes

- visible : Element is visible
- invisible : Element is invisible

### Example on Visibility

```
<html>
<head>
<title>Visibility</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container">
<p class="visible">Visible</p>
<p class="invisible">Invisible</p>
<p>other para</p>
</div>
</body>
</html>
```



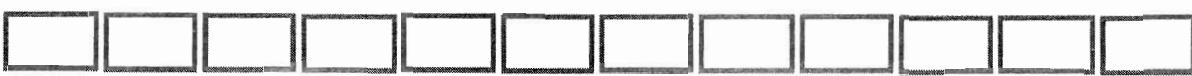
## Grid System

### Understanding the Columns

- It is used to divide the web page as rows.
- Each row contains 12 equal blocks / columns.
- A <div> tag can occupy one or more blocks.

- A row can have maximum 12 <div> tags.

## row

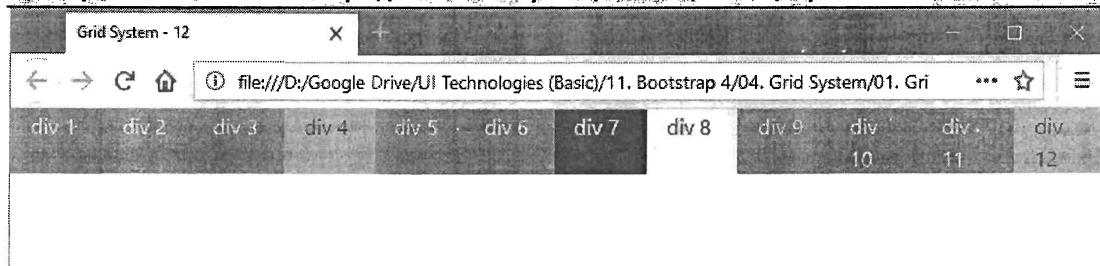


### List of Classes

- .col-n : Specifies how many columns are occupied by the <div> tag

### Example on Grid System - 12

```
<html>
  <head>
    <title>Grid System - 12</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="row">
        <div class="col-1 bg-primary text-white">div 1</div>
        <div class="col-1 bg-success text-white">div 2</div>
        <div class="col-1 bg-info text-white">div 3</div>
        <div class="col-1 bg-warning text-dark">div 4</div>
        <div class="col-1 bg-danger text-white">div 5</div>
        <div class="col-1 bg-secondary text-white">div 6</div>
        <div class="col-1 bg-dark text-white">div 7</div>
        <div class="col-1 bg-light text-dark">div 8</div>
        <div class="col-1 bg-primary text-white">div 9</div>
        <div class="col-1 bg-success text-white">div 10</div>
        <div class="col-1 bg-info text-white">div 11</div>
        <div class="col-1 bg-warning text-dark">div 12</div>
      </div>
    </div>
  </body>
</html>
```



### Example on Grid System - 4+4+4

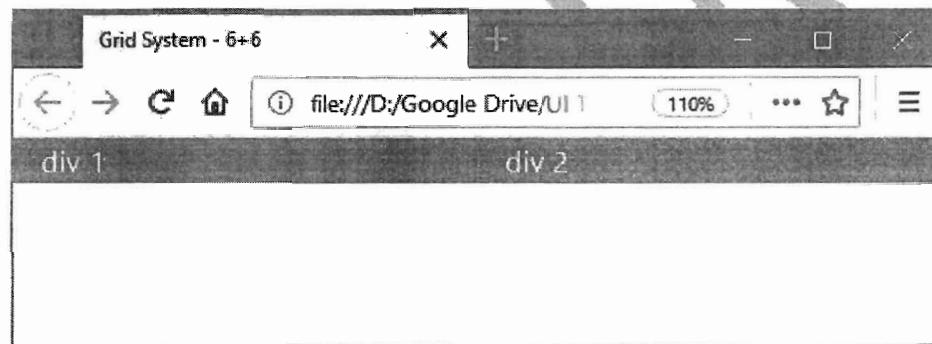
```
<html>
  <head>
    <title>Grid System - 4+4+4</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="row">
        <div class="col-4 bg-primary text-white">div 1</div>
        <div class="col-4 bg-success text-white">div 2</div>
        <div class="col-4 bg-info text-white">div 3</div>
      </div>
    </div>
  </body>
</html>
```



### Example on Grid System - 6+6

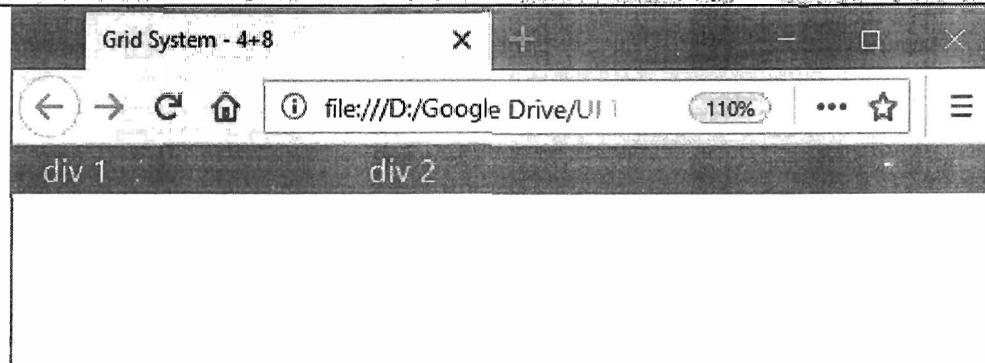
```
<html>
  <head>
    <title>Grid System - 6+6</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">
<div class="row">
<div class="col-6 bg-primary text-white">div 1</div>
<div class="col-6 bg-success text-white">div 2</div>
</div>
</div>
</body>
</html>
```



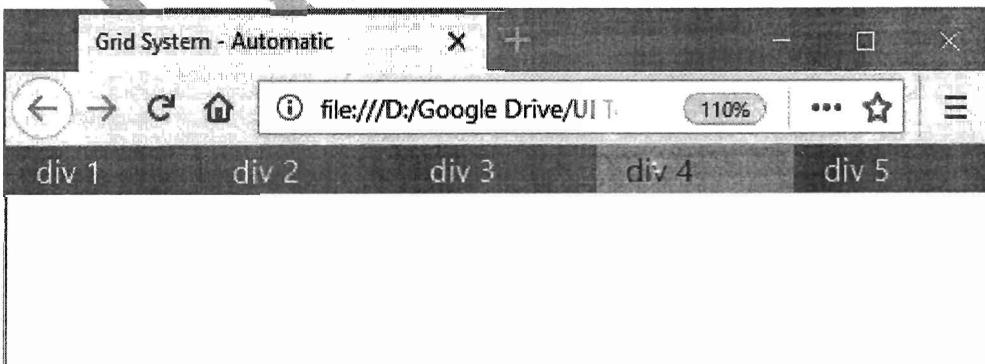
### Example on Grid System - 4+8

```
<html>
<head>
<title>Grid System - 4+8</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">
<div class="row">
<div class="col-4 bg-primary text-white">div 1</div>
<div class="col-8 bg-success text-white">div 2</div>
</div>
</div>
</body>
</html>
```



### Example on Grid System - Automatic

```
<html>
  <head>
    <title>Grid System - Automatic</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="row">
        <div class="col bg-primary text-white">div 1</div>
        <div class="col bg-success text-white">div 2</div>
        <div class="col bg-info text-white">div 3</div>
        <div class="col bg-warning text-dark">div 4</div>
        <div class="col bg-danger text-white">div 5</div>
      </div>
    </div>
  </body>
</html>
```



### Grid System with Responsive Web Design

- It is used to display the content differently on different devices, based on the screen resolution.
- It makes the web page fit for the current resolution automatically.
- We divide the devices into 5 types.

Sl. No	Type of Device	Screen width (pixels)	Examples
1	Extra Small Devices	1px to 575px	iPhone 2G, 3G (320px / 480px height)
2	Small Devices	576px to 767px	iPhone 4, 4S (640px width / 960px height) iPhone 5, 5C, 5C, SE (640px width / 1136px) iPhone 6, 6S, 7, 8 (750px width / 1134px height)
3	Medium Devices	768px to 991px	Samsung Note 1 (800px width / 1280px height)
4	Large Devices	992px to 1199px	iPhone 6 Plus (1080px width / 1920px height) iPhone 10 (1125px width / 2436px height)
5	Extra Large Devices	1200px to unlimited	iPhone 7+, 8+ (1242px width / 2208px height) Most-used laptops

#### Extra Small Devices:

- Very-Low-range Phones: Screens between 1px to 575px width.
- Ex: iPhone 2G, 3G (320px width / 480px height)

```
<div class="col-n">...</div>
<div class="col-n">...</div>
```

...

#### Small Devices:

- Low-range phones & Tablets: Screens between 576px to 767px width.
- Ex: iPhone 4, 4s (640px width / 960px height)
- Ex: iPhone 5, 5s, 5C, SE (640px width / 1136px height)
- Ex: iPhone 6, 6s, 7, 8 (750px width / 1134px height)

```
<div class="col-sm-n">...</div>
```

```
<div class="col-sm-n">...</div>
```

...

#### Medium Devices:

- Lower-mid range phones & Small laptops: Screens between 768px to 991px width.
- Ex: Samsung Note 1 (800px width / 1280px height)

```
<div class="col-md-n">...</div>
```

```
<div class="col-md-n">...</div>
```

...

### Large Devices:

- Mid-range phones & Small laptops: Screens between 992px to 1199px width.
- Ex: iPhone 6 Plus (1080px width / 1920px height)
- Ex: iPhone 10 (1125px width / 2436px height)

```
<div class="col-lg-n">...</div>
```

```
<div class="col-lg-n">...</div>
```

...

### Extra Large Devices:

- High-end phones & Most used laptops: Screens between 1200px to unlimited width.
- Ex: iPhone 7+, 8+ (1242px width / 2208px height)
- Ex: iPad (all types)

```
<div class="col-lg-n">...</div>
```

```
<div class="col-lg-n">...</div>
```

...

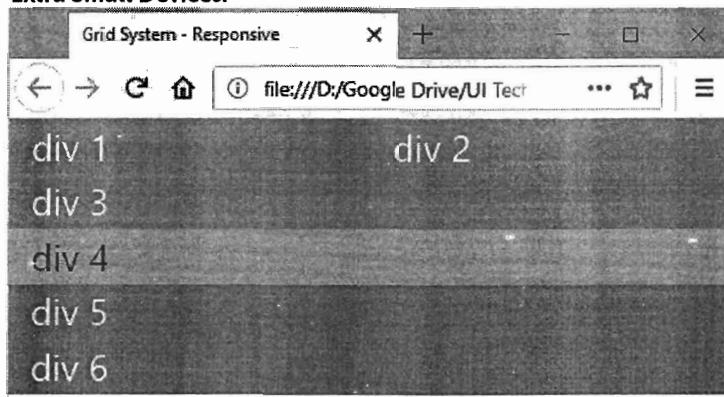
### Example on Grid System - Responsive

---

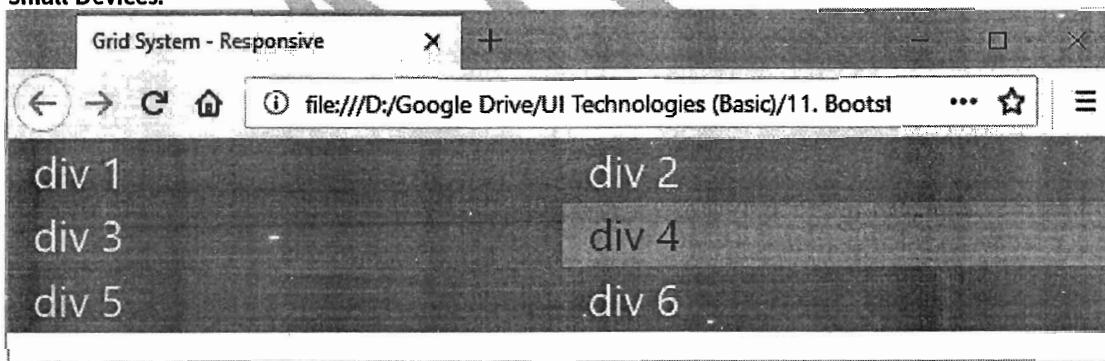
```
<html>
  <head>
    <title>Grid System - Responsive</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
    <style>
      body
      {
        font-size: 25px;
      }
    </style>
  </head>
  <body>
```

```
<div class="container-fluid">
<div class="row">
  <div class="col-6 col-sm-6 col-md-4 col-lg-3 col-xl-2 bg-primary text-white">div 1</div>
  <div class="col-6 col-sm-6 col-md-4 col-lg-3 col-xl-2 bg-success text-white">div 2</div>
  <div class="col-12 col-sm-6 col-md-4 col-lg-3 col-xl-2 bg-info text-white">div 3</div>
  <div class="col-12 col-sm-6 col-md-4 col-lg-3 col-xl-2 bg-warning text-dark">div 4</div>
  <div class="col-12 col-sm-6 col-md-4 col-lg-6 col-xl-2 bg-danger text-white">div 5</div>
  <div class="col-12 col-sm-6 col-md-4 col-lg-6 col-xl-2 bg-secondary text-white">div 6</div>
</div>
</div>
</body>
</html>
```

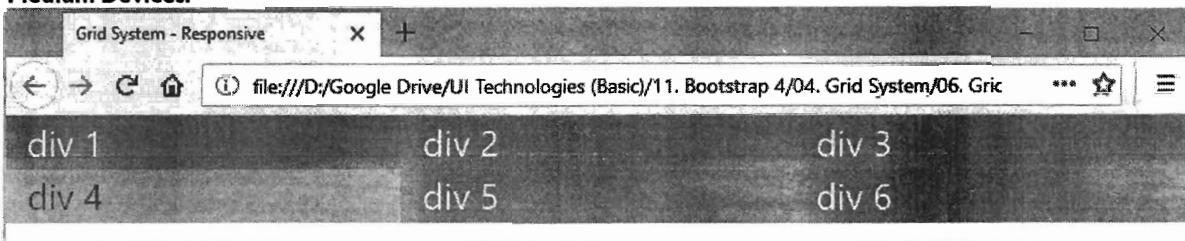
**Extra Small Devices:**



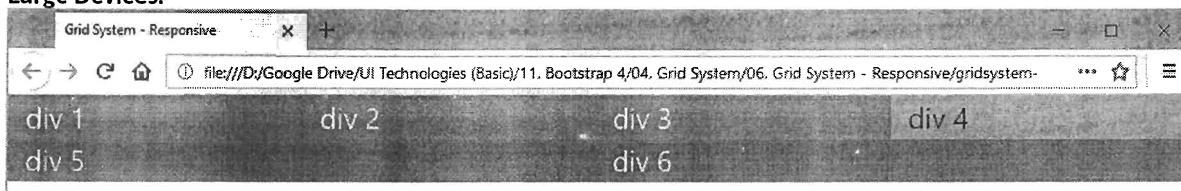
**Small Devices:**



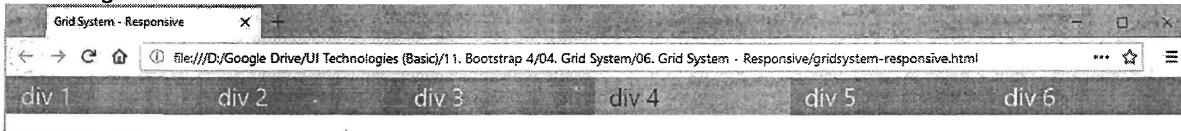
**Medium Devices:**



Large Devices:



Extra Large Devices:

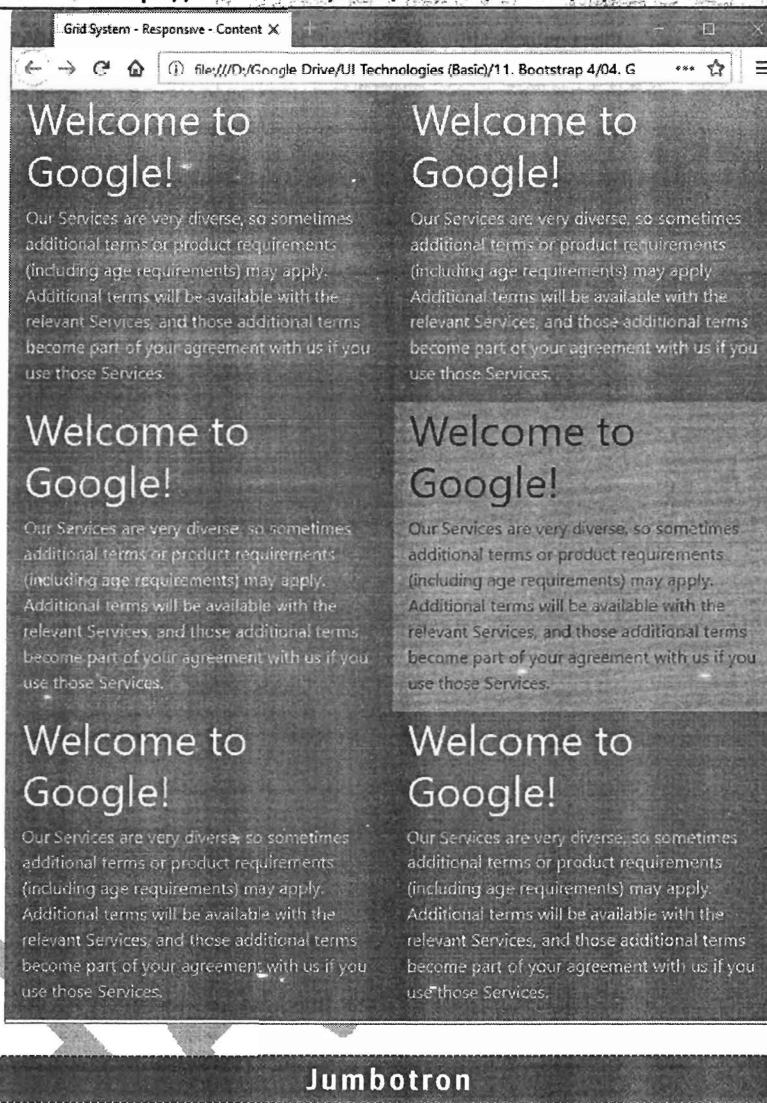


**Example on Grid System - Responsive - With Content**

```
<html>
  <head>
    <title>Grid System - Responsive - Content</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="row">
        <div class="col-xl-2 col-lg-3 col-md-4 col-sm-6 col-6 bg-primary text-white">
          <h1>Welcome to Google!</h1>
          <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
        </div>
        <div class="col-xl-2 col-lg-3 col-md-4 col-sm-6 col-6 bg-success text-white">
          <h1>Welcome to Google!</h1>
          <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
        </div>
        <div class="col-xl-2 col-lg-3 col-md-4 col-sm-6 col-12 bg-info text-white">
          <h1>Welcome to Google!</h1>
          <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
        </div>
        <div class="col-xl-2 col-lg-3 col-md-4 col-sm-6 col-12 bg-warning text-dark">
          <h1>Welcome to Google!</h1>
```

```
<p>Our Services are very diverse, so sometimes additional terms or product requirements  
(including age requirements) may apply. Additional terms will be available with the relevant Services,  
and those additional terms become part of your agreement with us if you use those Services.</p>  
</div>  
<div class="col-xl-2 col-lg-6 col-md-4 col-sm-6 col-12 bg-danger text-white">  
<h1>Welcome to Google!</h1>  
<p>Our Services are very diverse, so sometimes additional terms or product requirements  
(including age requirements) may apply. Additional terms will be available with the relevant Services,  
and those additional terms become part of your agreement with us if you use those Services.</p>  
</div>  
<div class="col-xl-2 col-lg-6 col-md-4 col-sm-6 col-12 bg-secondary text-white">  
<h1>Welcome to Google!</h1>  
<p>Our Services are very diverse, so sometimes additional terms or product requirements  
(including age requirements) may apply. Additional terms will be available with the relevant Services,  
and those additional terms become part of your agreement with us if you use those Services.</p>  
</div>  
</div>  
</div>  
</body>  
</html>
```

HARSH



### Jumbotron

- It is used to display heading and paragraph in large size with a special box to highlight its content.

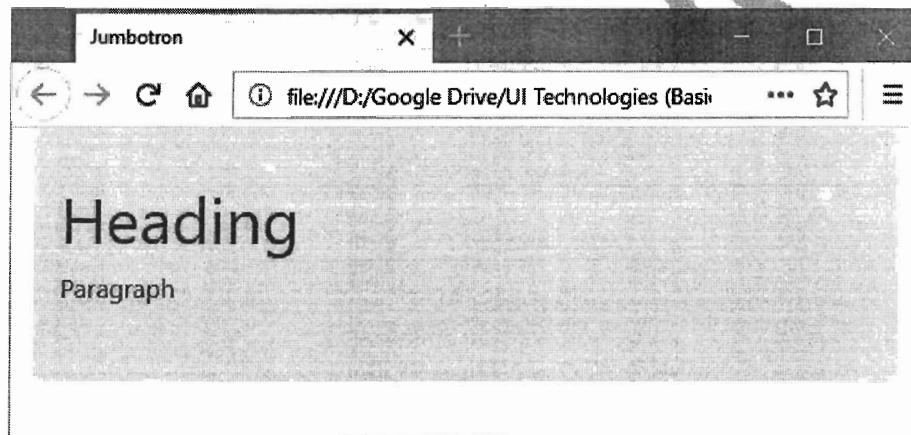
### List of Classes

- `jumbotron` : Large heading and paragraph.

### Example on Jumbotron

```
<html>
  <head>
    <title>Jumbotron</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
```

```
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">
<div class="jumbotron">
<h1>Heading</h1>
<p>Paragraph</p>
</div>
</div>
</body>
</html>
```



## Images

### Image Shapes

- It is used to display images with rounded corners.

### List of Classes

- rounded : Rounded corners
- rounded-circle : Circle-shaped image
- img-thumbnail : Bordered image

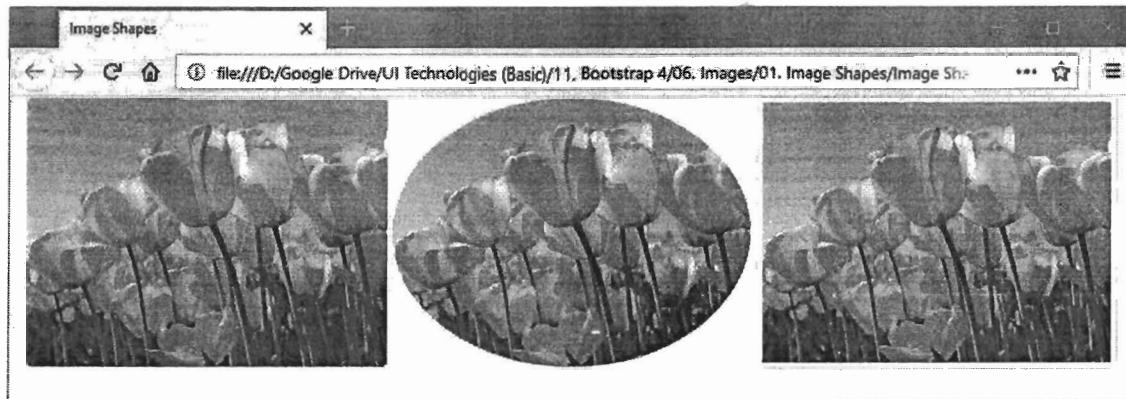
### Example on Image Shapes

```
<html>
<head>
<title>Image Shapes</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
```

```
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">



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



## Image Alignment

- It is used to specify horizontal alignment of the image.

### List of Classes

- float-left : Left alignment
- float-right : Right alignment
- mx-auto : Center alignment
- d-block : Display the image as a block; this is needed to apply "mx-auto".

### Example on Image Alignment

```
<html>
<head>
<title>Image Alignment</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">

```

```


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



### Image Fluid

- It is used to display reduce the size of the image automatically, if the page width is reduced.

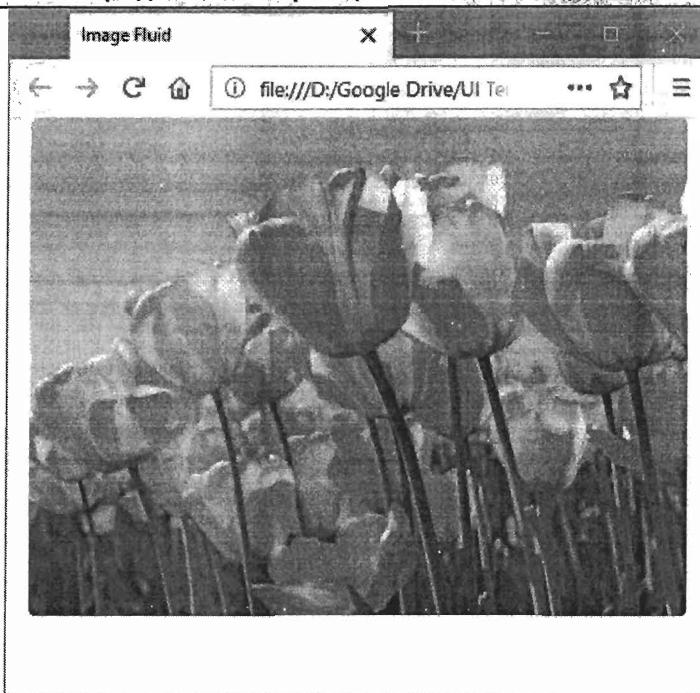
### List of Classes

- img-fluid : Image size automatically gets reduced if the screen width is reduced.

### Example on Image Fluid

```
<html>
<head>
<title>Image Fluid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">

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



## Tables

### Basic Table

- It is used to display table with expand width and padding, horizontal borders.

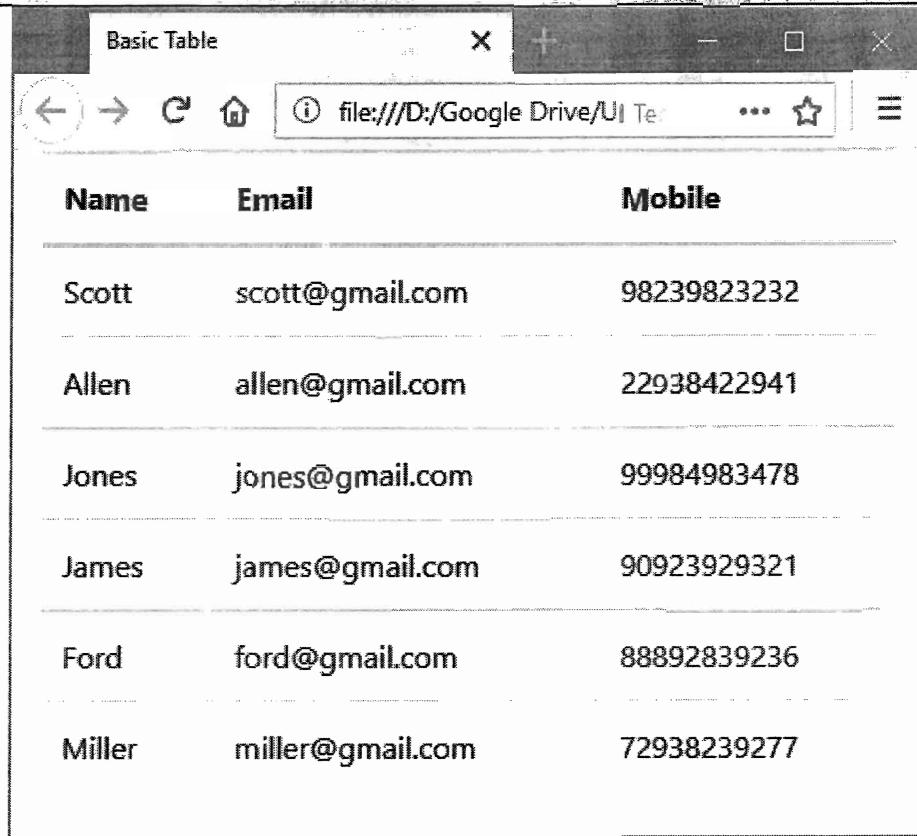
### List of Classes

- `table` : bootstrap table style.

### Example on Basic Table

```
<html>
  <head>
    <title>Basic Table</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container">
      <table class="table">
        <thead>
          <tr>
            <th>Name</th>
```

```
<th>Email</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scott</td>
<td>scott@gmail.com</td>
<td>9823982323</td>
</tr>
<tr>
<td>Allen</td>
<td>allen@gmail.com</td>
<td>22938422941</td>
</tr>
<tr>
<td>Jones</td>
<td>jones@gmail.com</td>
<td>99984983478</td>
</tr>
<tr>
<td>James</td>
<td>james@gmail.com</td>
<td>90923929321</td>
</tr>
<tr>
<td>Ford</td>
<td>ford@gmail.com</td>
<td>88892839236</td>
</tr>
<tr>
<td>Miller</td>
<td>miller@gmail.com</td>
<td>72938239277</td>
</tr>
</tbody>
</table>
</div>
</body>
</html>
```



Name	Email	Mobile
Scott	scott@gmail.com	98239823232
Allen	allen@gmail.com	22938422941
Jones	jones@gmail.com	99984983478
James	james@gmail.com	90923929321
Ford	ford@gmail.com	88892839236
Miller	miller@gmail.com	72938239277

### Borderless Table

- It is used to display table without borders.

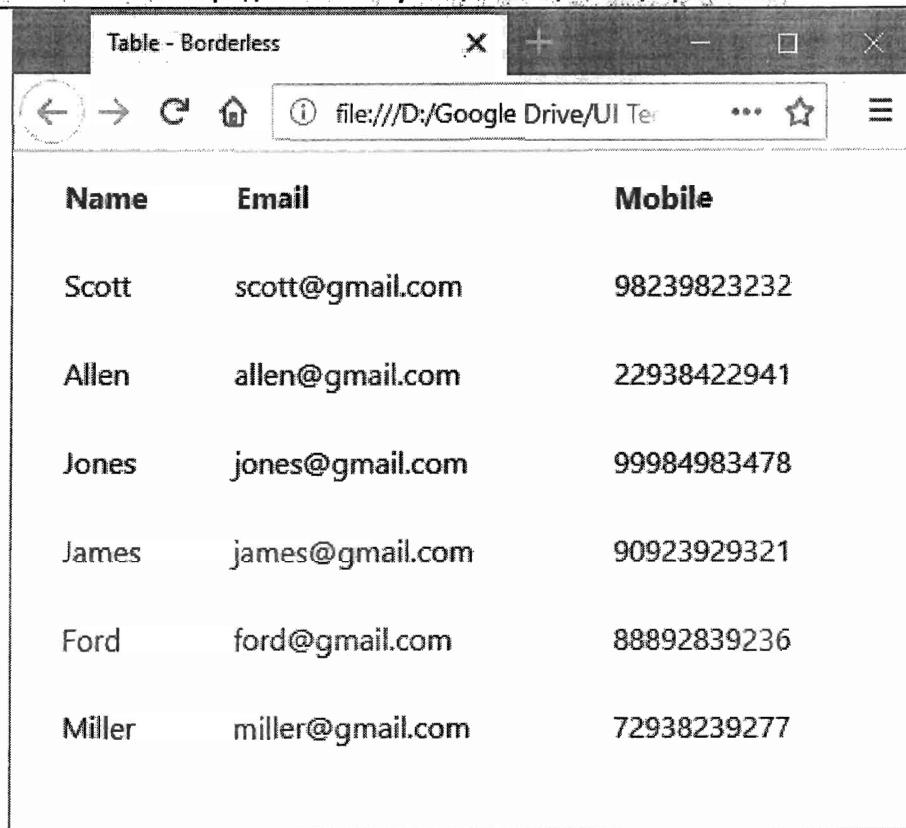
### List of Classes

- `table-borderless` : Table without border

### Example on Borderless Table

```
<html>
  <head>
    <title>Table - Borderless</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container">
      <table class="table table-borderless">
        <thead>
```

```
<tr>
<th>Name</th>
<th>Email</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scott</td>
<td>scott@gmail.com</td>
<td>98239823232</td>
</tr>
<tr>
<td>Allen</td>
<td>allen@gmail.com</td>
<td>22938422941</td>
</tr>
<tr>
<td>Jones</td>
<td>jones@gmail.com</td>
<td>99984983478</td>
</tr>
<tr>
<td>James</td>
<td>james@gmail.com</td>
<td>90923929321</td>
</tr>
<tr>
<td>Ford</td>
<td>ford@gmail.com</td>
<td>88892839236</td>
</tr>
<tr>
<td>Miller</td>
<td>miller@gmail.com</td>
<td>72938239277</td>
</tr>
</tbody>
</table>
</div>
</body>
</html>
```



Name	Email	Mobile
Scott	scott@gmail.com	98239823232
Allen	allen@gmail.com	22938422941
Jones	jones@gmail.com	99984983478
James	james@gmail.com	90923929321
Ford	ford@gmail.com	88892839236
Miller	miller@gmail.com	72938239277

### Bordered Table

- It is used to display table with both horizontal & vertical borders.

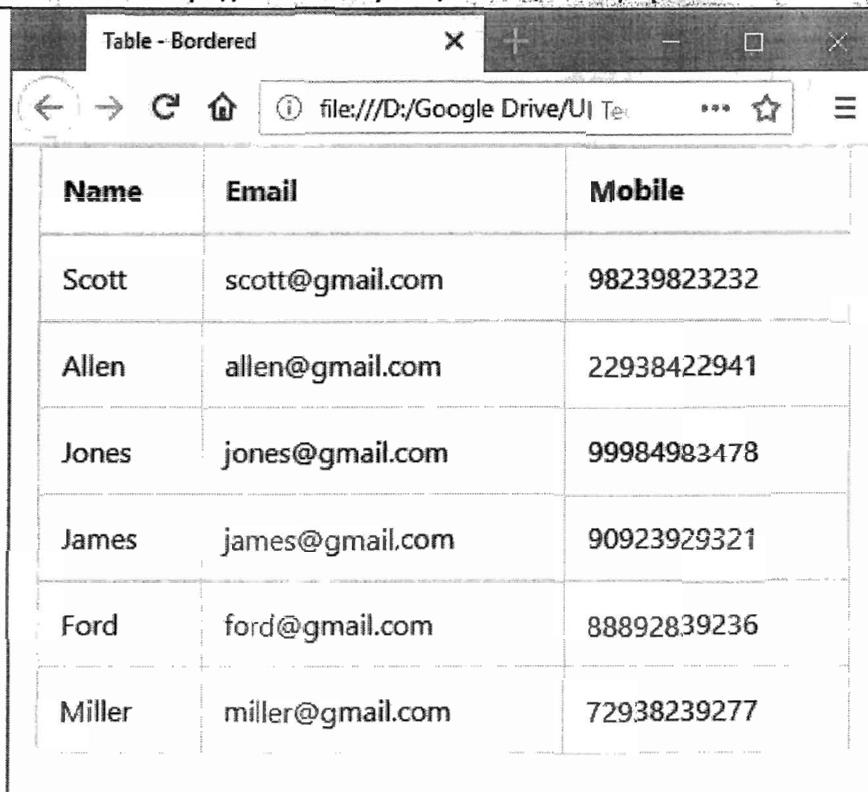
### List of Classes

- `table-bordered` : Table with border

### Example on Bordered Table

```
<html>
  <head>
    <title>Table - Bordered</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container">
      <table class="table table-bordered">
        <thead>
          <tr>
```

```
<th>Name</th>
<th>Email</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scott</td>
<td>scott@gmail.com</td>
<td>98239823232</td>
</tr>
<tr>
<td>Allen</td>
<td>allen@gmail.com</td>
<td>22938422941</td>
</tr>
<tr>
<td>Jones</td>
<td>jones@gmail.com</td>
<td>99984983478</td>
</tr>
<tr>
<td>James</td>
<td>james@gmail.com</td>
<td>90923929321</td>
</tr>
<tr>
<td>Ford</td>
<td>ford@gmail.com</td>
<td>88892839236</td>
</tr>
<tr>
<td>Miller</td>
<td>miller@gmail.com</td>
<td>72938239277</td>
</tr>
</tbody>
</table>
</div>
</body>
</html>
```



Name	Email	Mobile
Scott	scott@gmail.com	98239823232
Allen	allen@gmail.com	22938422941
Jones	jones@gmail.com	99984983478
James	james@gmail.com	90923929321
Ford	ford@gmail.com	88892839236
Miller	miller@gmail.com	72938239277

### Striped Table

- It is used to display table with alternate row background

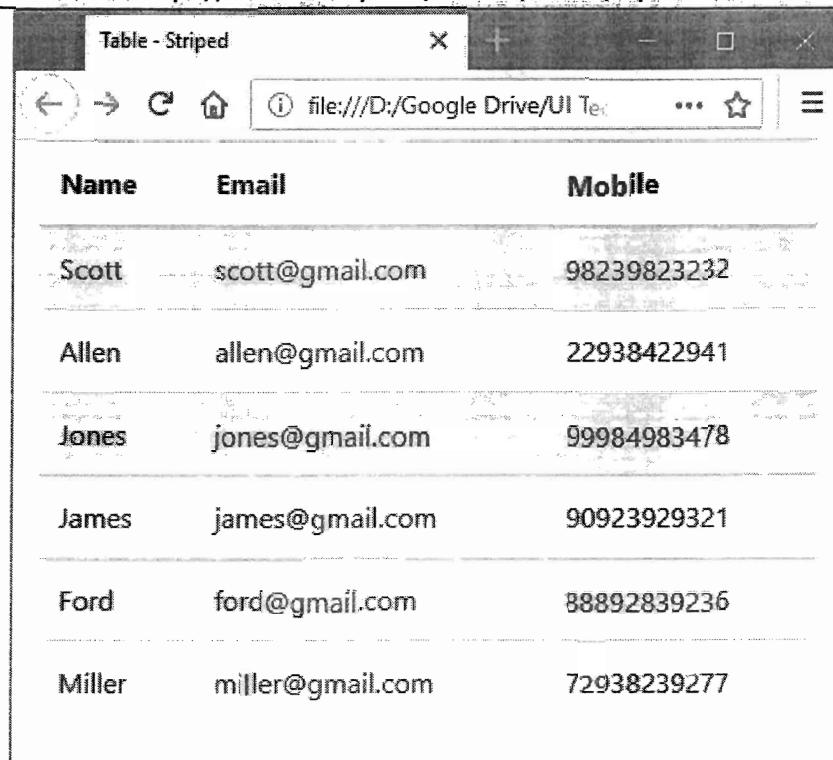
### List of Classes

- `table-striped` : Table with alternate row background.

### Example on Striped Table

```
<html>
  <head>
    <title>Table - Striped</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container">
      <table class="table table-striped">
        <thead>
          <tr>
```

```
<th>Name</th>
<th>Email</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scott</td>
<td>scott@gmail.com</td>
<td>98239823232</td>
</tr>
<tr>
<td>Allen</td>
<td>allen@gmail.com</td>
<td>22938422941</td>
</tr>
<tr>
<td>Jones</td>
<td>jones@gmail.com</td>
<td>99984983478</td>
</tr>
<tr>
<td>James</td>
<td>james@gmail.com</td>
<td>90923929321</td>
</tr>
<tr>
<td>Ford</td>
<td>ford@gmail.com</td>
<td>88892839236</td>
</tr>
<tr>
<td>Miller</td>
<td>miller@gmail.com</td>
<td>72938239277</td>
</tr>
</tbody>
</table>
</div>
</body>
</html>
```



The screenshot shows a web browser window with the title 'Table - Striped'. The address bar indicates the file is located on Google Drive. The table has three columns: 'Name', 'Email', and 'Mobile'. The rows are styled with alternating background colors. The first row is labeled 'Table - Striped'. The last row is labeled 'Table - Hover'.

Name	Email	Mobile
Scott	scott@gmail.com	98239823232
Allen	allen@gmail.com	22938422941
Jones	jones@gmail.com	99984983478
James	james@gmail.com	90923929321
Ford	ford@gmail.com	88892839236
Miller	Miller@gmail.com	72938239277

### Hover Table

- It is used to display table with background color change on row hover.

#### List of Classes

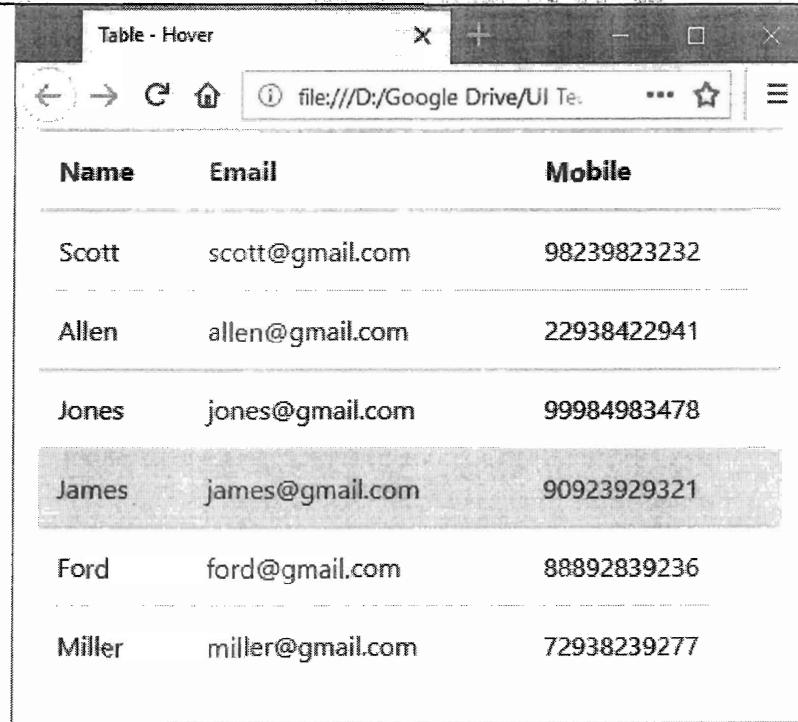
- table-hover : Table row background color gets changed on hover

#### Example on Hover Table

```
<html>
  <head>
    <title>Table - Hover</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container">
      <table class="table table-hover">
        <thead>
          <tr>
            <th>Name</th>
            <th>Email</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>Scott</td>
            <td>scott@gmail.com</td>
            <td>98239823232</td>
          </tr>
          <tr>
            <td>Allen</td>
            <td>allen@gmail.com</td>
            <td>22938422941</td>
          </tr>
          <tr>
            <td>Jones</td>
            <td>jones@gmail.com</td>
            <td>99984983478</td>
          </tr>
          <tr>
            <td>James</td>
            <td>james@gmail.com</td>
            <td>90923929321</td>
          </tr>
          <tr>
            <td>Ford</td>
            <td>ford@gmail.com</td>
            <td>88892839236</td>
          </tr>
          <tr>
            <td>Miller</td>
            <td>Miller@gmail.com</td>
            <td>72938239277</td>
          </tr>
        </tbody>
      </table>
    </div>
  </body>

```

```
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scott</td>
<td>scott@gmail.com</td>
<td>98239823232</td>
</tr>
<tr>
<td>Allen</td>
<td>allen@gmail.com</td>
<td>22938422941</td>
</tr>
<tr>
<td>Jones</td>
<td>jones@gmail.com</td>
<td>99984983478</td>
</tr>
<tr>
<td>James</td>
<td>james@gmail.com</td>
<td>90923929321</td>
</tr>
<tr>
<td>Ford</td>
<td>ford@gmail.com</td>
<td>88892839236</td>
</tr>
<tr>
<td>Miller</td>
<td>miller@gmail.com</td>
<td>72938239277</td>
</tr>
</tbody>
</table>
</div>
</body>
</html>
```



Name	Email	Mobile
Scott	scott@gmail.com	98239823232
Allen	allen@gmail.com	22938422941
Jones	jones@gmail.com	99984983478
James	james@gmail.com	90923929321
Ford	ford@gmail.com	88892839236
Miller	miller@gmail.com	72938239277

### Table Background Colors

- It is used to display table / rows with different background colors.
- These classes can be applicable for `<table>`, `<tr>` or `<td>` tags.

### List of Classes

- `table-primary` : Blue color
- `table-success` : Green color
- `table-danger` : Red color
- `table-info` : Light blue color
- `table-warning` : Orange color
- `table-active` : Very light Grey color
- `table-secondary` : Grey color
- `table-light` : Light grey color
- `table-dark` : Dark grey color

### Example on Table Background Colors

```
<html>
<head>
<title>Table - Background Colors</title>
<meta charset="utf-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container">
<table class="table">
<thead>
<tr>
<th>Name</th>
<th>Email</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr class="table-primary">
<td>Scott</td>
<td>scott@gmail.com</td>
<td>98239823232</td>
</tr>
<tr class="table-success">
<td>Allen</td>
<td>allen@gmail.com</td>
<td>22938422941</td>
</tr>
<tr class="table-danger">
<td>Jones</td>
<td>jones@gmail.com</td>
<td>99984983478</td>
</tr>
<tr class="table-info">
<td>James</td>
<td>james@gmail.com</td>
<td>90923929321</td>
</tr>
<tr class="table-warning">
<td>Ford</td>
<td>ford@gmail.com</td>
<td>88892839236</td>
</tr>
<tr class="table-active">
<td>Miller</td>
<td>miller@gmail.com</td>
<td>72938239277</td>
</tr>
<tr class="table-secondary">
<td>Miller</td>
<td>miller@gmail.com</td>
<td>88829389231</td>

```

```
</tr>
<tr class="table-light">
<td>Alice</td>
<td>alice@gmail.com</td>
<td>77772983898</td>
</tr>
<tr class="table-dark">
<td>Bob</td>
<td>bob@gmail.com</td>
<td>33982930110</td>
</tr>
</tbody>
</table>
</div>
</body>
</html>
```



Table - Background Colors

file:///D:/Google Drive/UI Te

Name	Email	Mobile
Scott	scott@gmail.com	98239823232
Allen	allen@gmail.com	22938422941
Jones	jones@gmail.com	99984983478
James	james@gmail.com	90923929321
Ford	ford@gmail.com	88892839236
Miller	miller@gmail.com	72938239277
Miller	miller@gmail.com	88829389231
Alice	alice@gmail.com	77772983898
Bob	bob@gmail.com	33982930110

### Table Header Background Colors

- It is used to display table header row with dark / light colors.
- These classes can be applicable only for `<thead>` tag.

#### List of Classes

- `thead-dark` : Black color
- `thead-light` : Light Grey color

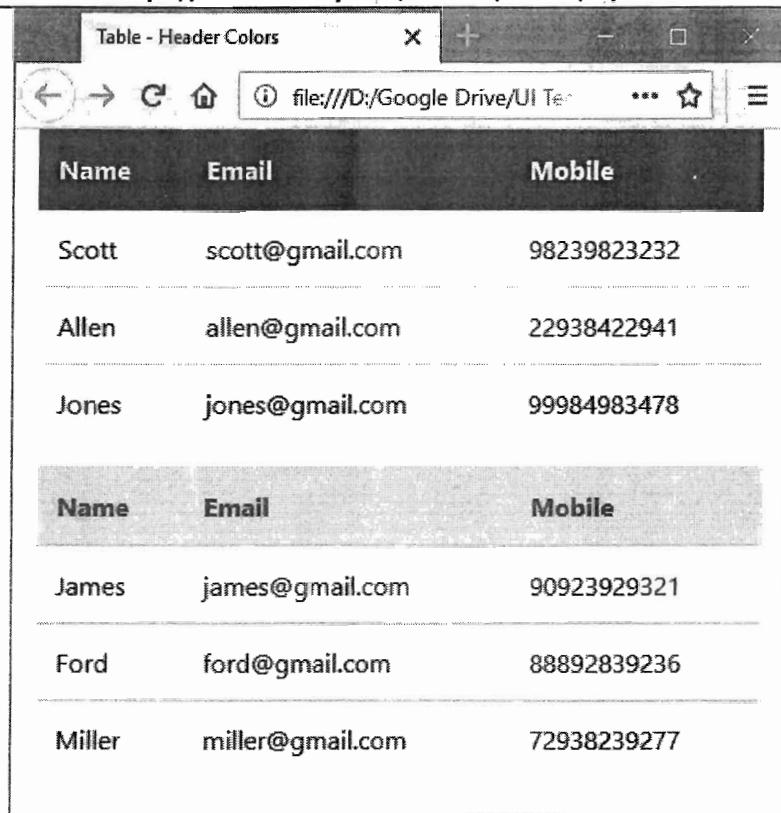
#### Example on Table Header Background Colors

```
<html>
  <head>
    <title>Table - Header Colors</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container">
      <table class="table">
        <thead class="thead-dark">
          <tr>
            <th>Name</th>
            <th>Email</th>
            <th>Mobile</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>Scott</td>
            <td>scott@gmail.com</td>
            <td>9823982323</td>
          </tr>
          <tr>
            <td>Allen</td>
            <td>allen@gmail.com</td>
            <td>22938422941</td>
          </tr>
          <tr>
            <td>Jones</td>
            <td>jones@gmail.com</td>
            <td>99984983478</td>
          </tr>
        </tbody>
      </table>
    </div>
  </body>

```

```
<table class="table">
  <thead class="thead-light">
    <tr>
      <th>Name</th>
      <th>Email</th>
      <th>Mobile</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>James</td>
      <td>james@gmail.com</td>
      <td>90923929321</td>
    </tr>
    <tr>
      <td>Ford</td>
      <td>ford@gmail.com</td>
      <td>88892839236</td>
    </tr>
    <tr>
      <td>Miller</td>
      <td>miller@gmail.com</td>
      <td>72938239277</td>
    </tr>
  </tbody>
</table>
</div>
</body>
</html>
```

HA  
R  
SH  
A



Name	Email	Mobile
Scott	scott@gmail.com	98239823232
Allen	allen@gmail.com	22938422941
Jones	jones@gmail.com	99984983478

Name	Email	Mobile
James	james@gmail.com	90923929321
Ford	ford@gmail.com	88892839236
Miller	miller@gmail.com	72938239277

### Table Small

- It is used to display table with small size (less padding).

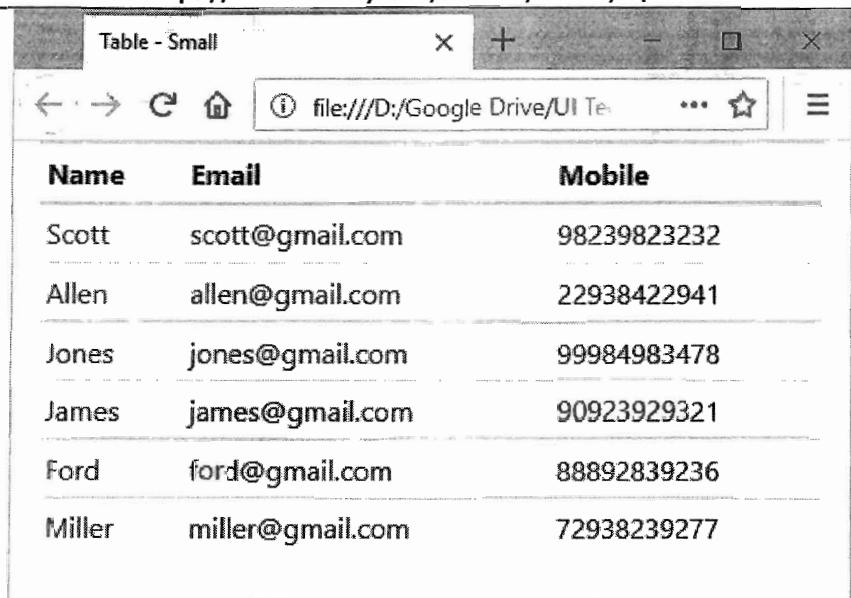
### List of Classes

- `table-sm` : Table with Less padding

### Example on Table Small

```
<html>
  <head>
    <title>Table - Small</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container">
      <table class="table table-sm">
        <thead>
          <tr>
```

```
<th>Name</th>
<th>Email</th>
<th>Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scott</td>
<td>scott@gmail.com</td>
<td>98239823232</td>
</tr>
<tr>
<td>Allen</td>
<td>allen@gmail.com</td>
<td>22938422941</td>
</tr>
<tr>
<td>Jones</td>
<td>jones@gmail.com</td>
<td>99984983478</td>
</tr>
<tr>
<td>James</td>
<td>james@gmail.com</td>
<td>90923929321</td>
</tr>
<tr>
<td>Ford</td>
<td>ford@gmail.com</td>
<td>88892839236</td>
</tr>
<tr>
<td>Miller</td>
<td>miller@gmail.com</td>
<td>72938239277</td>
</tr>
</tbody>
</table>
</div>
</body>
</html>
```



Name	Email	Mobile
Scott	scott@gmail.com	98239823232
Allen	allen@gmail.com	22938422941
Jones	jones@gmail.com	99984983478
James	james@gmail.com	90923929321
Ford	ford@gmail.com	88892839236
Miller	miller@gmail.com	72938239277

### Table Responsive

- It is used to display scrollbar for the table automatically, when the web page is resized.
- This class can be applicable only for `<div>` tag that contains `<table>` tag.

### List of Classes

- `table-responsive` : Table with scrollbar

### Example on Table Responsive

```

<html>
  <head>
    <title>Table - Responsive</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container">
      <div class="table-responsive">
        <table class="table">
          <thead>
            <tr>
              <th>Name</th>
              <th>Email</th>
              <th>Mobile</th>
            </tr>
          </thead>

```

```
<tbody>
<tr>
<td>Scott</td>
<td>scott@gmail.com</td>
<td>98239823232</td>
</tr>
<tr>
<td>Allen</td>
<td>allen@gmail.com</td>
<td>22938422941</td>
</tr>
<tr>
<td>Jones</td>
<td>jones@gmail.com</td>
<td>99984983478</td>
</tr>
<tr>
<td>James</td>
<td>james@gmail.com</td>
<td>90923929321</td>
</tr>
<tr>
<td>Ford</td>
<td>ford@gmail.com</td>
<td>88892839236</td>
</tr>
<tr>
<td>Miller</td>
<td>miller@gmail.com</td>
<td>72938239277</td>
</tr>
</tbody>
</table>
</div>
</div>
</body>
</html>
```

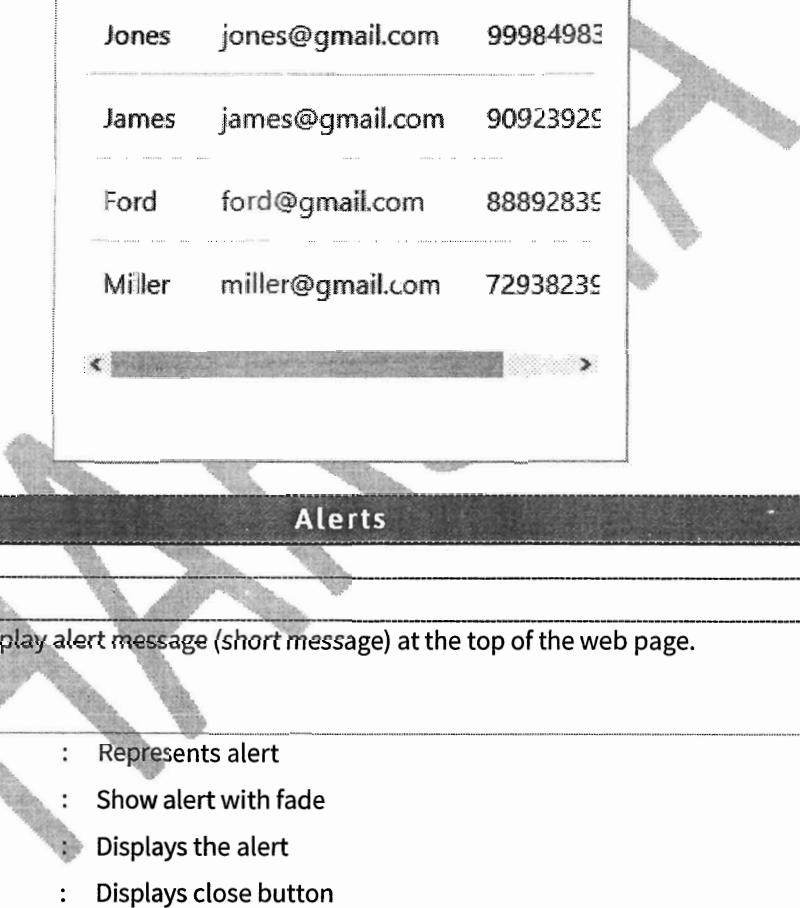


Table - Respon + - □ ×

file:///D:/C/

Name	Email	Mobile
Scott	scott@gmail.com	98239823
Allen	allen@gmail.com	22938422
Jones	jones@gmail.com	99984983
James	james@gmail.com	90923929
Ford	ford@gmail.com	88892839
Miller	miller@gmail.com	72938239

## Alerts

### Alerts

- It is used to display alert message (short message) at the top of the web page.

### List of Classes

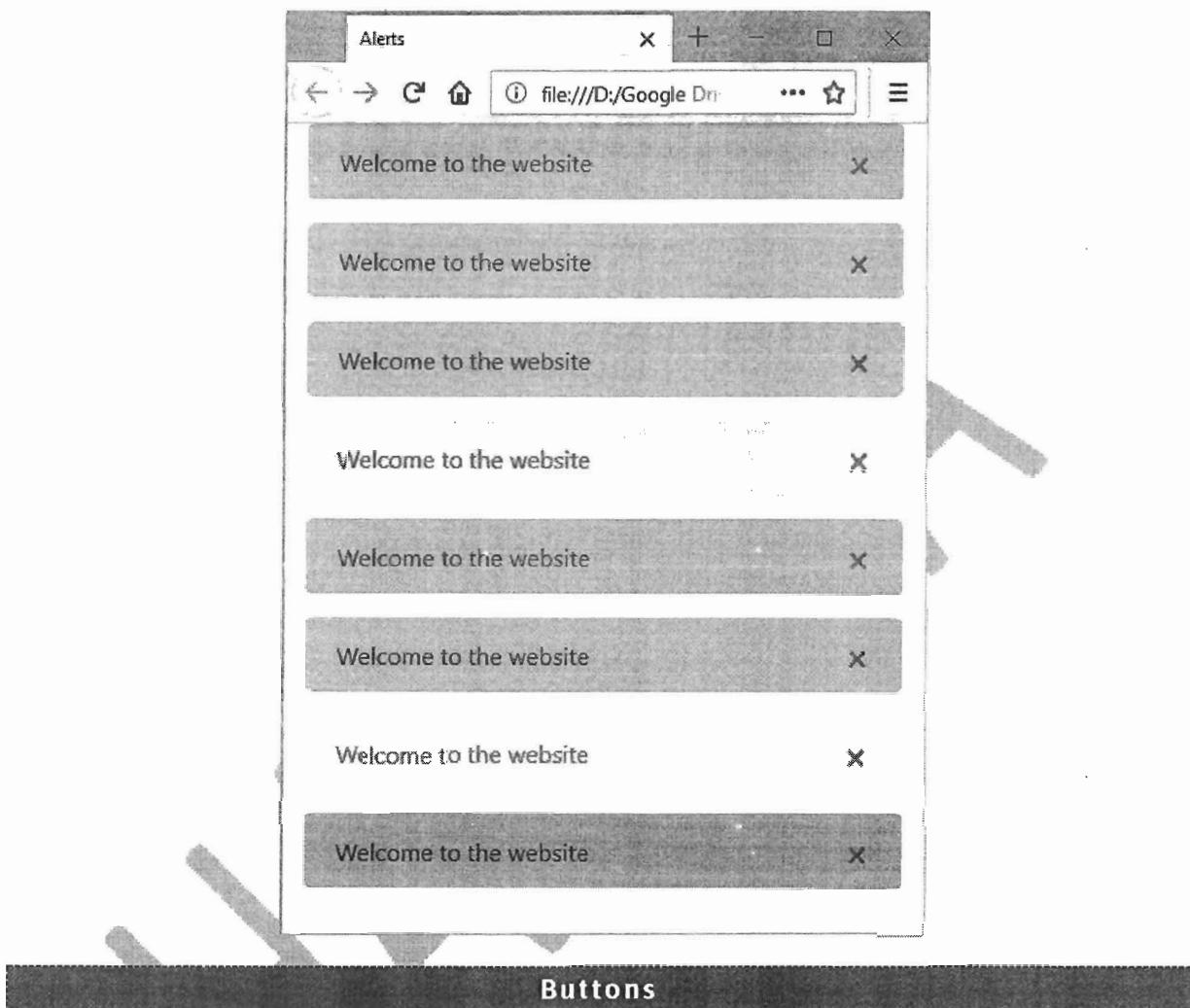
- alert : Represents alert
- fade : Show alert with fade
- show : Displays the alert
- close : Displays close button
- alert-primary : Blue color
- alert-success : Green color
- alert-info : Light blue color
- alert-warning : Orange color
- alert-danger : Red color
- alert-secondary : Grey color

- alert-light : Light grey color
- alert-dark : Black color

### Example on Alerts

```
<html>
  <head>
    <title>Alerts</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="alert alert-primary fade show">
        <button class="close" data-dismiss="alert">&times;</button>
        Welcome to the website
      </div>
      <div class="alert alert-success fade show">
        <button class="close" data-dismiss="alert">&times;</button>
        Welcome to the website
      </div>
      <div class="alert alert-info fade show">
        <button class="close" data-dismiss="alert">&times;</button>
        Welcome to the website
      </div>
      <div class="alert alert-warning fade show">
        <button class="close" data-dismiss="alert">&times;</button>
        Welcome to the website
      </div>
      <div class="alert alert-danger fade show">
        <button class="close" data-dismiss="alert">&times;</button>
        Welcome to the website
      </div>
      <div class="alert alert-secondary fade show">
        <button class="close" data-dismiss="alert">&times;</button>
        Welcome to the website
      </div>
      <div class="alert alert-light fade show">
        <button class="close" data-dismiss="alert">&times;</button>
        Welcome to the website
      </div>
      <div class="alert alert-dark fade show">
        <button class="close" data-dismiss="alert">&times;</button>
        Welcome to the website
      </div>
    </div>
  </body>
```

</html>



### Button Colors

- It is used to display buttons with different colors.

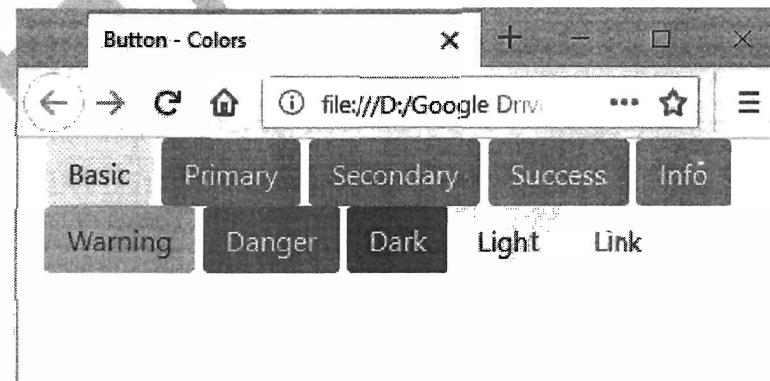
### List of Classes

- btn : Bootstrap button style
- btn-primary : Blue color
- btn-secondary : Grey color
- btn-success : Green color
- btn-info : Light blue color
- btn-warning : Orange color

- btn-danger : Red color
- btn-dark : Black color
- btn-light : Light grey color
- btn-link : Link style

### Example on Button Colors

```
<html>
  <head>
    <title>Button - Colors</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <button class="btn">Basic</button>
      <button class="btn btn-primary">Primary</button>
      <button class="btn btn-secondary">Secondary</button>
      <button class="btn btn-success">Success</button>
      <button class="btn btn-info">Info</button>
      <button class="btn btn-warning">Warning</button>
      <button class="btn btn-danger">Danger</button>
      <button class="btn btn-dark">Dark</button>
      <button class="btn btn-light">Light</button>
      <button class="btn btn-link">Link</button>
    </div>
  </body>
</html>
```



### Button Outline

- It is used to display buttons with only border (white background).

---

### List of Classes

---

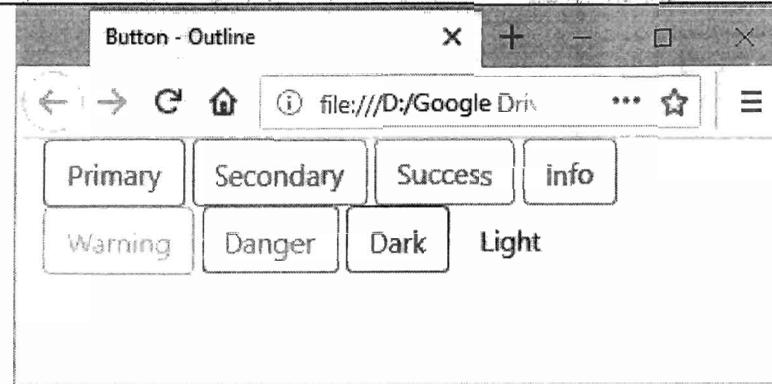
- btn-outline-primary : Blue color
- btn-outline-secondary : Grey color
- btn-outline-success : Green color
- btn-outline-info : Light blue color
- btn-outline-warning : Orange color
- btn-outline-danger : Red color
- btn-outline-dark : Black color
- btn-outline-light : Light grey color

---

### Example on Button Outline

---

```
<html>
  <head>
    <title>Button - Outline</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <button class="btn btn-outline-primary">Primary</button>
      <button class="btn btn-outline-secondary">Secondary</button>
      <button class="btn btn-outline-success">Success</button>
      <button class="btn btn-outline-info">Info</button>
      <button class="btn btn-outline-warning">Warning</button>
      <button class="btn btn-outline-danger">Danger</button>
      <button class="btn btn-outline-dark">Dark</button>
      <button class="btn btn-outline-light text-dark">Light</button>
    </div>
  </body>
</html>
```



### Button Sizes

- It is used to display large / small buttons.

### List of Classes

- `btn-block` : Full-width button
- `btn-lg` : Large button
- `btn-sm` : Small button

### Example on Button Sizes

```
<html>
  <head>
    <title>Button - Sizes</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <button class="btn btn-primary btn-block">Block</button>
      <button class="btn btn-primary btn-lg">Large</button>
      <button class="btn btn-primary">Default</button>
      <button class="btn btn-primary btn-sm">Small</button>
    </div>
  </body>
</html>
```



### Button Groups

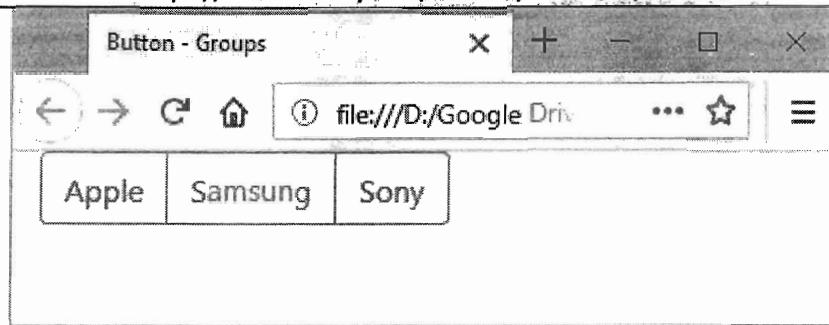
- It is used to display grouped buttons.
- It reduces margin between buttons.

### List of Classes

- `btn-group` : Group of buttons

### Example on Button Groups

```
<html>
  <head>
    <title>Button - Groups</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="btn-group">
        <button class="btn btn-outline-primary">Apple</button>
        <button class="btn btn-outline-primary">Samsung</button>
        <button class="btn btn-outline-primary">Sony</button>
      </div>
    </div>
  </body>
</html>
```



### Button Vertical Groups

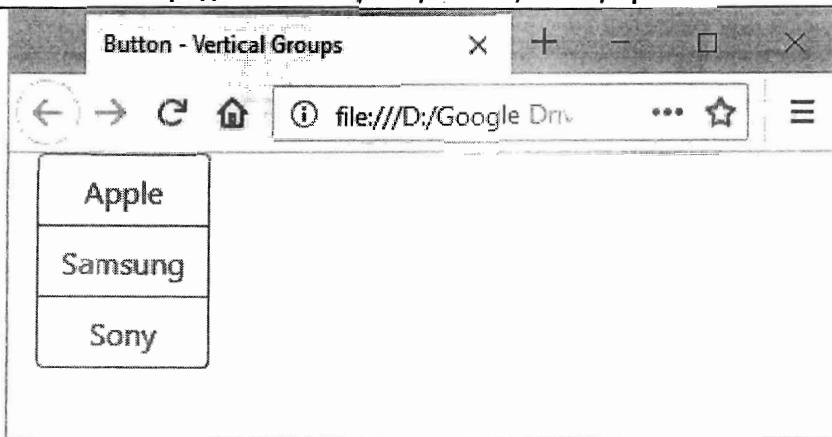
- It is used to display vertically grouped buttons.
- It reduces margin between buttons.

### List of Classes

- `btn-group-vertical` : Vertical group of buttons

### Example on Button Vertical Groups

```
<html>
  <head>
    <title>Button - Vertical Groups</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="btn-group-vertical">
        <button class="btn btn-outline-primary">Apple</button>
        <button class="btn btn-outline-primary">Samsung</button>
        <button class="btn btn-outline-primary">Sony</button>
      </div>
    </div>
  </body>
</html>
```



### Button DropDown

- It is used to display dropdownlist for the button.

#### List of Classes

- dropdown-toggle : Dropdownbutton
- dropdown-toggle-split : Split button
- caret : Caret symbol
- dropdown-menu : Represents the menu
- dropdown-item : Represents item in the menu

#### Example on Button DropDown

```
<html>
  <head>
    <title>Button - DropDown</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="btn-group">
        <button type="button" class="btn btn-primary">Sony</button>
        <button type="button" class="btn btn-primary dropdown-toggle dropdown-toggle-split" data-toggle="dropdown">
          <span class="caret"></span>
        </button>
        <div class="dropdown-menu">
          <a class="dropdown-item" href="#">Tablet</a>
          <a class="dropdown-item" href="#">Smartphone</a>
        </div>
      </div>
    </div>
  </body>
</html>
```

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



## Badges

### Basic Badges

- It is used to indication near heading / button.

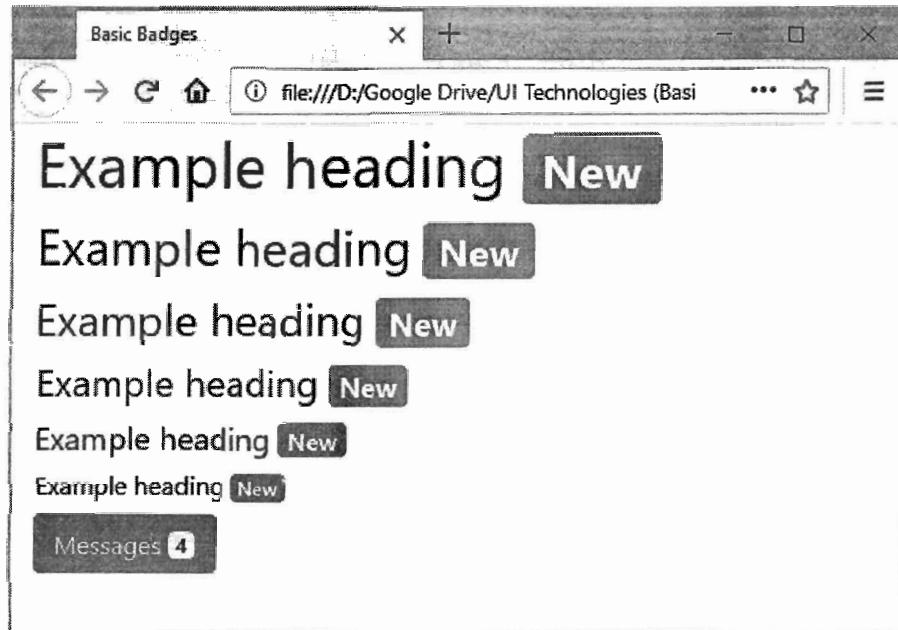
### List of Classes

- badge : Represents Badge

### Example on Badges

```
<html>
  <head>
    <title>Basic Badges</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <h1>Example heading <span class="badge badge-secondary">New</span></h1>
      <h2>Example heading <span class="badge badge-secondary">New</span></h2>
      <h3>Example heading <span class="badge badge-secondary">New</span></h3>
      <h4>Example heading <span class="badge badge-secondary">New</span></h4>
      <h5>Example heading <span class="badge badge-secondary">New</span></h5>
      <h6>Example heading <span class="badge badge-secondary">New</span></h6>
    </div>
  </body>
</html>
```

```
<button type="button" class="btn btn-primary">  
  Messages <span class="badge badge-light">4</span>  
</button>  
</div>  
</body>  
</html>
```



### Badge Colors

- It is used to set background colors for the badges

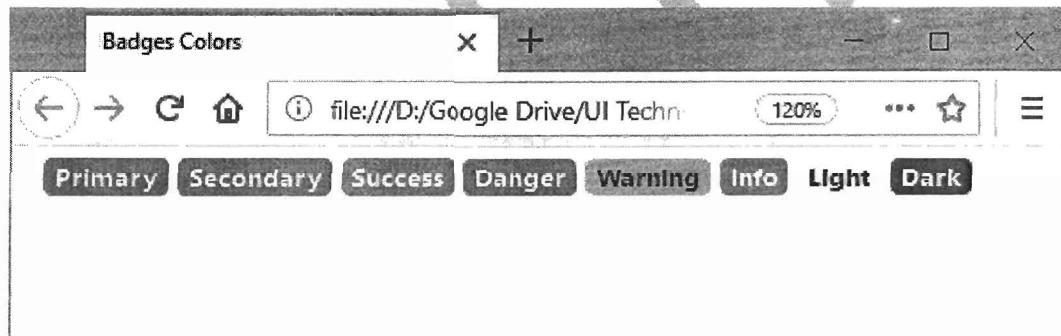
### List of Classes

- badge-primary : Blue color
- badge-secondary : Grey color
- badge-success : Green color
- badge-info : Light blue color
- badge-warning : Orange color
- badge-danger : Red color
- badge-dark : Black color
- badge-light : Light grey color

### Example on Badge Colors

```
<html>  
<head>  
<title>Basic Badges</title>
```

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">
  <h1>Example heading <span class="badge badge-secondary">New</span></h1>
  <h2>Example heading <span class="badge badge-secondary">New</span></h2>
  <h3>Example heading <span class="badge badge-secondary">New</span></h3>
  <h4>Example heading <span class="badge badge-secondary">New</span></h4>
  <h5>Example heading <span class="badge badge-secondary">New</span></h5>
  <h6>Example heading <span class="badge badge-secondary">New</span></h6>
  <button type="button" class="btn btn-primary">
    Messages <span class="badge badge-light">4</span>
  </button>
</div>
</body>
</html>
```



### Pill Badges

- It is used to display more rounded badges.

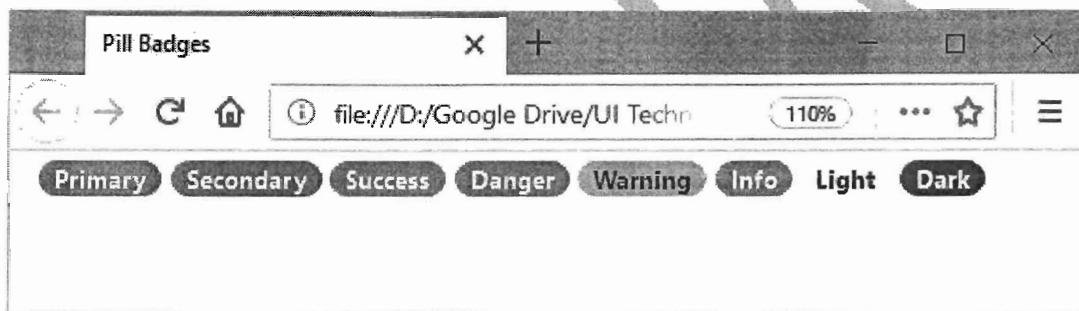
### List of Classes

- badge-pill : More rounded badge

### Example on Pill Badges

```
<html>
<head>
<title>Pill Badges</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
```

```
</head>
<body>
<div class="container-fluid">
  <span class="badge badge-pill badge-primary">Primary</span>
  <span class="badge badge-pill badge-secondary">Secondary</span>
  <span class="badge badge-pill badge-success">Success</span>
  <span class="badge badge-pill badge-danger">Danger</span>
  <span class="badge badge-pill badge-warning">Warning</span>
  <span class="badge badge-pill badge-info">Info</span>
  <span class="badge badge-pill badge-light">Light</span>
  <span class="badge badge-pill badge-dark">Dark</span>
</div>
</body>
</html>
```



## Progress Bar

### Basic Progress Bar

- It is used to display progress bar, based on the given width and height.

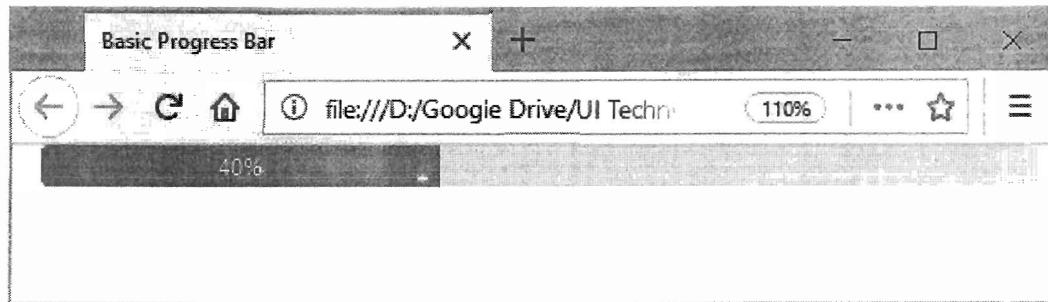
### List of Classes

- progress : Progress bar container
- progress-bar : Progress
- mx-auto : Center alignment

### Example on Basic Progress Bar

```
<html>
<head>
  <title>Basic Progress Bar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="bootstrap.css">
  <script src="jquery-3.3.1.js"></script>
  <script src="popper.js"></script>
  <script src="bootstrap.js"></script>
</head>
<body>
```

```
<div class="container-fluid">
  <div class="progress mx-auto" style="width:500px; height:20px">
    <div class="progress-bar" style="width:40%;height:20px">40%</div>
  </div>
</div>
</body>
</html>
```



### Progress Bar Colors

- It is used to display progress bar with different colors..

#### List of Classes

- bg-success : Green color
- bg-info : Light blue color
- bg-warning : Orange color
- bg-danger : Red color
- bg-secondary : Grey color
- bg-light : Light grey color
- bg-dark : Black color
- progress-bar-striped : Striped progress bar
- progress-bar-animated : Animated progress bar

#### Example on Progress Bar Colors

```
<html>
  <head>
    <title>Progress Bar Colors</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
```

```
<div class="progress" style="height:25px">
  <div class="progress-bar" style="width:10%"></div>
</div>

<div class="progress" style="height:25px">
  <div class="progress-bar bg-success" style="width:20%"></div>
</div>

<div class="progress" style="height:25px">
  <div class="progress-bar bg-info" style="width:30%"></div>
</div>

<div class="progress" style="height:25px">
  <div class="progress-bar bg-warning" style="width:40%"></div>
</div>

<div class="progress" style="height:25px">
  <div class="progress-bar bg-danger" style="width:50%"></div>
</div>

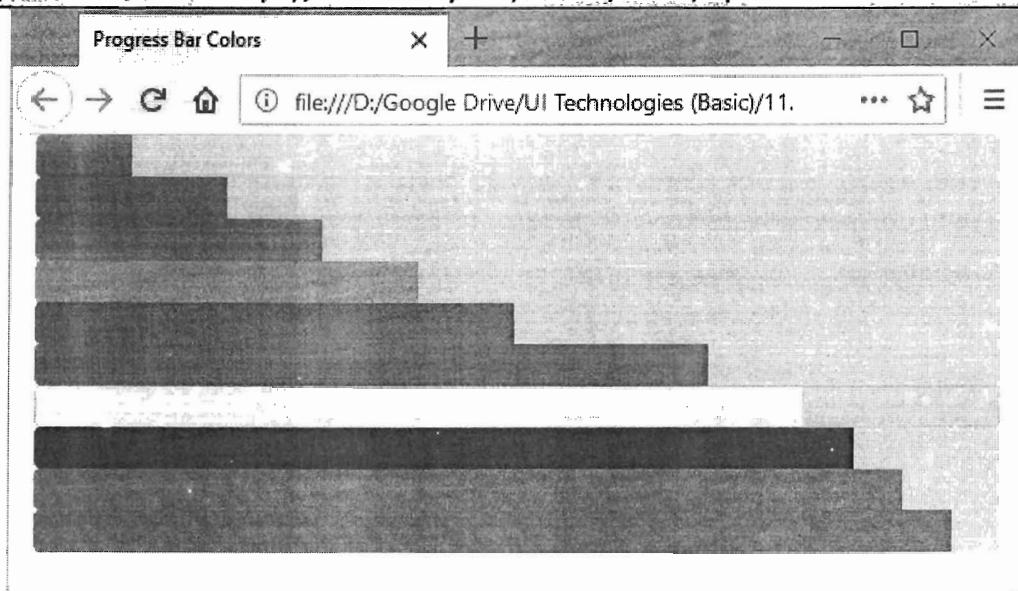
<div class="progress" style="height:25px">
  <div class="progress-bar bg-secondary" style="width:70%"></div>
</div>

<div class="progress border" style="height:25px">
  <div class="progress-bar bg-light" style="width:80%"></div>
</div>

<div class="progress" style="height:25px">
  <div class="progress-bar bg-dark" style="width:85%"></div>
</div>

<div class="progress" style="height:25px">
  <div class="progress-bar progress-bar-striped bg-success" style="width:90%"></div>
</div>

<div class="progress" style="height:25px">
  <div class="progress-bar progress-bar-striped progress-bar-animated bg-primary" style="width:95%"></div>
</div>
</div>
</body>
</html>
```



## Pagination

### Basic Pagination

- It is used to display page numbers.

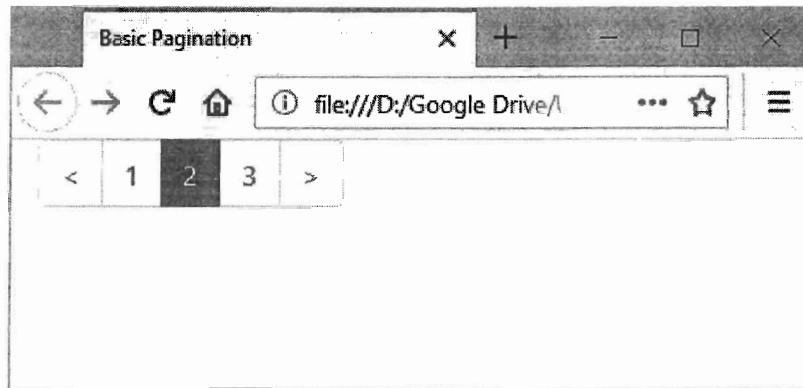
### List of Classes

- `pagination` : Enable pagination style
- `page-item` : Page number
- `page-link` : Page number link
- `active` : Current page

### Example on Basic Pagination

```
<html>
  <head>
    <title>Basic Pagination</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <ul class="pagination">
        <li class="page-item"><a class="page-link" href="#">&lt;</a></li>
        <li class="page-item"><a class="page-link" href="#">1</a></li>
```

```
<li class="page-item active"><a class="page-link" href="#">2</a></li>
<li class="page-item"><a class="page-link" href="#">3</a></li>
<li class="page-item"><a class="page-link" href="#">&gt;</a></li>
</ul>
</div>
</body>
</html>
```



### Pagination Size

- It is used to display pagination in large / small size.

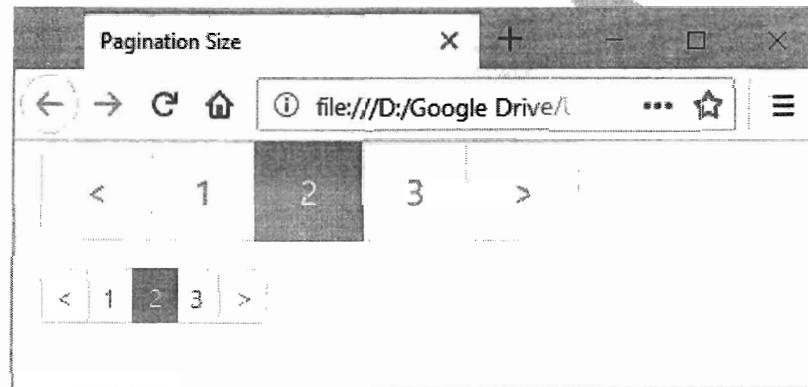
### List of Classes

- pagination-lg : Large size progress bar
- pagination-sm : Small size progress bar

### Example on Pagination Size

```
<html>
<head>
<title>Pagination Size</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">
<ul class="pagination pagination-lg">
<li class="page-item"><a class="page-link" href="#">&lt;</a></li>
<li class="page-item"><a class="page-link" href="#">1</a></li>
<li class="page-item active"><a class="page-link" href="#">2</a></li>
<li class="page-item"><a class="page-link" href="#">3</a></li>
<li class="page-item"><a class="page-link" href="#">&gt;</a></li>
</ul>
</div>
</body>
</html>
```

```
<ul class="pagination pagination-sm">
<li class="page-item"><a class="page-link" href="#">&lt;</a></li>
<li class="page-item"><a class="page-link" href="#">1</a></li>
<li class="page-item active"><a class="page-link" href="#">2</a></li>
<li class="page-item"><a class="page-link" href="#">3</a></li>
<li class="page-item"><a class="page-link" href="#">&gt;</a></li>
</ul>
</div>
</body>
</html>
```



### Pagination Alignment

- It is used to display pagination left / center / right side of the page.

### List of Classes

- justify-content-center : Center alignment for pagination
- justify-content-end : Right alignment for pagination

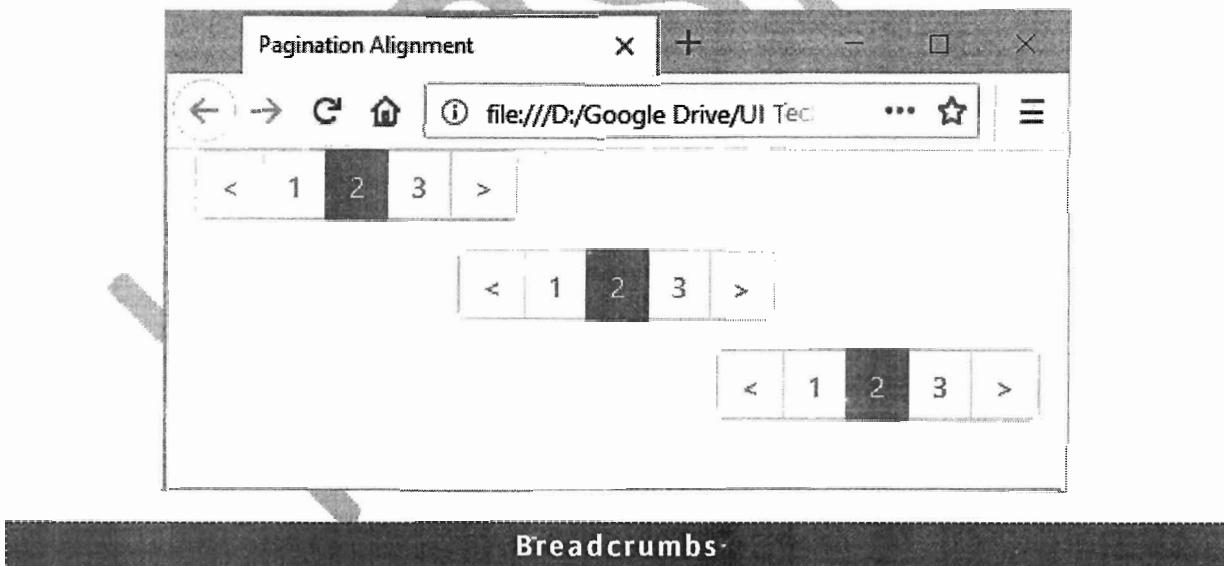
### Example on Pagination Alignment

```
<html>
<head>
<title>Pagination Alignment</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">
<ul class="pagination">
<li class="page-item"><a class="page-link" href="#">&lt;</a></li>
<li class="page-item"><a class="page-link" href="#">1</a></li>
```

```
<li class="page-item active"><a class="page-link" href="#">2</a></li>
<li class="page-item"><a class="page-link" href="#">3</a></li>
<li class="page-item"><a class="page-link" href="#">&gt;</a></li>
</ul>

<ul class="pagination justify-content-center">
<li class="page-item"><a class="page-link" href="#">&lt;</a></li>
<li class="page-item"><a class="page-link" href="#">1</a></li>
<li class="page-item active"><a class="page-link" href="#">2</a></li>
<li class="page-item"><a class="page-link" href="#">3</a></li>
<li class="page-item"><a class="page-link" href="#">&gt;</a></li>
</ul>

<ul class="pagination justify-content-end">
<li class="page-item"><a class="page-link" href="#">&lt;</a></li>
<li class="page-item"><a class="page-link" href="#">1</a></li>
<li class="page-item active"><a class="page-link" href="#">2</a></li>
<li class="page-item"><a class="page-link" href="#">3</a></li>
<li class="page-item"><a class="page-link" href="#">&gt;</a></li>
</ul>
</div>
</body>
</html>
```



## Breadcrumbs

- It is used to display location (navigation path) of the current web page in the website.

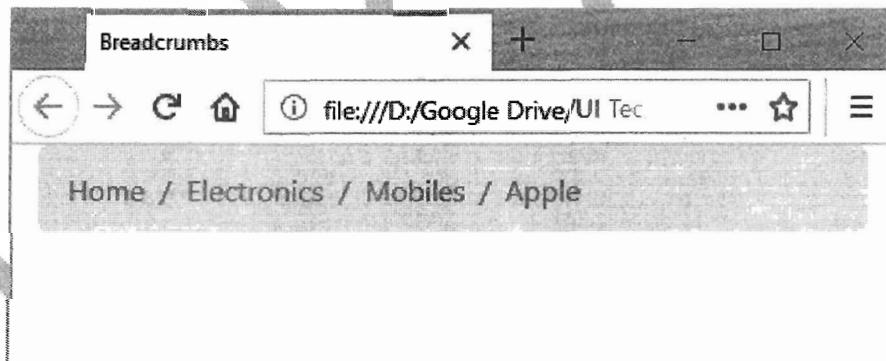
### List of Classes

- breadcrumb : Represents entire breadcrumb
- breadcrumb-item : Represents single item in the breadcrumb

- active : Represents current item

### Example on Breadcrumbs

```
<html>
  <head>
    <title>Breadcrumbs</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <ul class="breadcrumb">
        <li class="breadcrumb-item"><a href="#">Home</a></li>
        <li class="breadcrumb-item"><a href="#">Electronics</a></li>
        <li class="breadcrumb-item"><a href="#">Mobiles</a></li>
        <li class="breadcrumb-item active">Apple</li>
      </ul>
    </div>
  </body>
</html>
```



## List Groups

### Basic List Groups

- It is used to display items (text / hyperlinks) as a list.

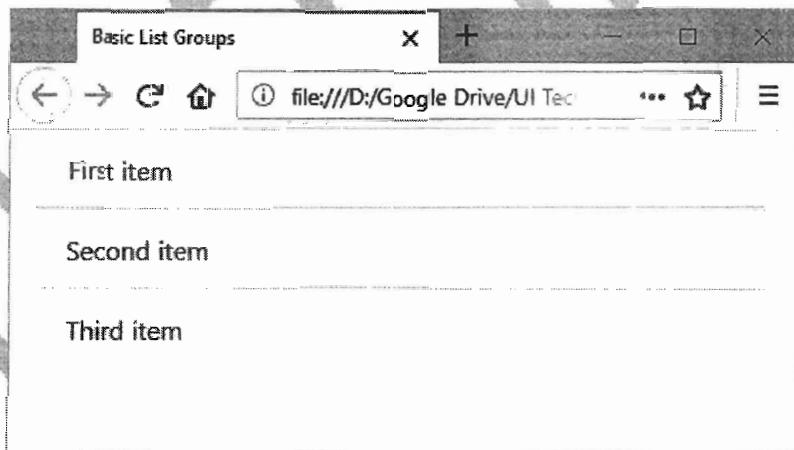
### List of Classes

- list-group : Represents the list
- list-group-flush : Remove borders for the list group
- list-group-item : Represents single item

- list-group-item-action : Hover style for the item

### Example on Basic List Groups

```
<html>
<head>
<title>Basic List Groups</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">
<div class="list-group list-group-flush">
<a href="#" class="list-group-item list-group-item-action">First item</a>
<a href="#" class="list-group-item list-group-item-action">Second item</a>
<a href="#" class="list-group-item list-group-item-action">Third item</a>
</div>
</div>
</body>
</html>
```



### List Group Colors

- It is used to display list group with different background colors.

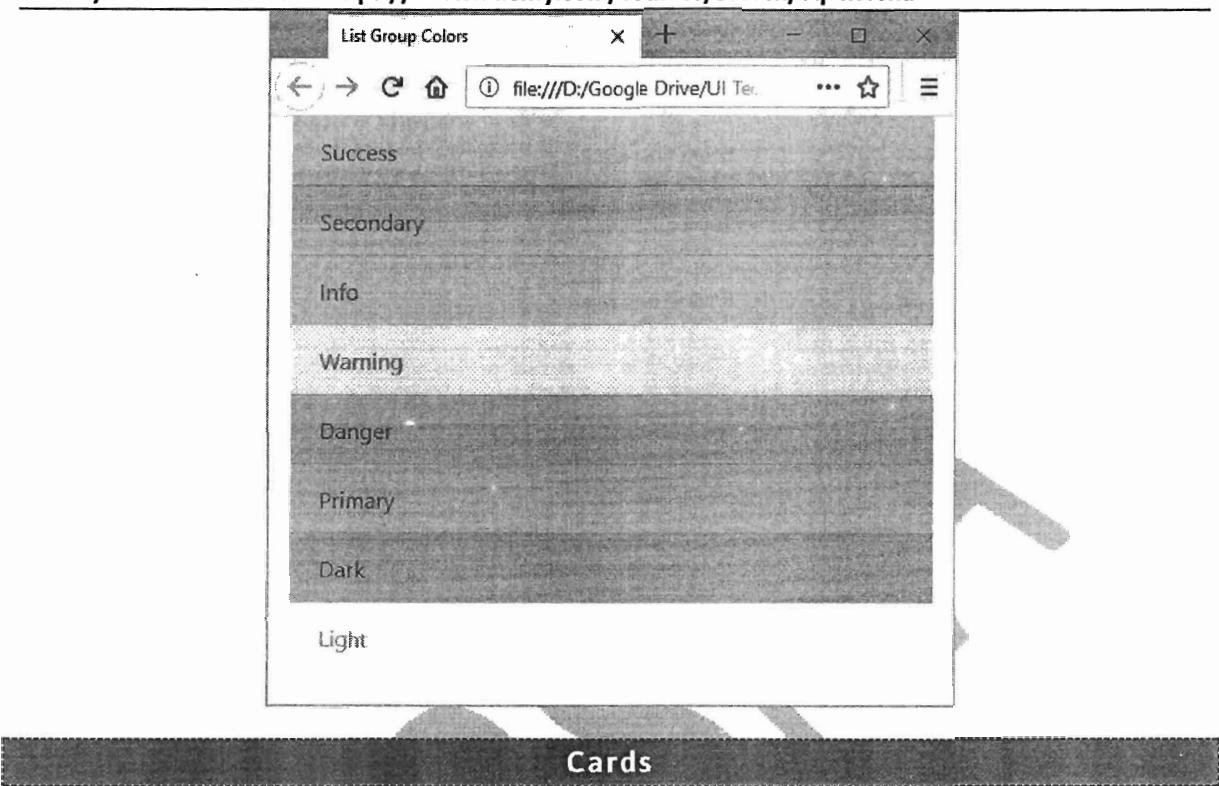
### List of Classes

- list-group-item-success : Green color
- list-group-item-secondary : Grey color
- list-group-item-info : Light blue color
- list-group-item-warning : Orange color

- list-group-item-danger : Red color
- list-group-item-dark : Black color
- list-group-item-light

### Example on List Group Colors

```
<html>
  <head>
    <title>List Group Colors</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="list-group list-group-flush">
        <a href="#" class="list-group-item list-group-item-action list-group-item-success">Success</a>
        <a href="#" class="list-group-item list-group-item-action list-group-item-secondary">Secondary</a>
        <a href="#" class="list-group-item list-group-item-action list-group-item-info">Info</a>
        <a href="#" class="list-group-item list-group-item-action list-group-item-warning">Warning</a>
        <a href="#" class="list-group-item list-group-item-action list-group-item-danger">Danger</a>
        <a href="#" class="list-group-item list-group-item-action list-group-item-primary">Primary</a>
        <a href="#" class="list-group-item list-group-item-action list-group-item-dark">Dark</a>
        <a href="#" class="list-group-item list-group-item-action list-group-item-light">Light</a>
      </div>
    </div>
  </body>
</html>
```



## Cards

### Basic Cards

- It is used to display some content (box) with header and footer.

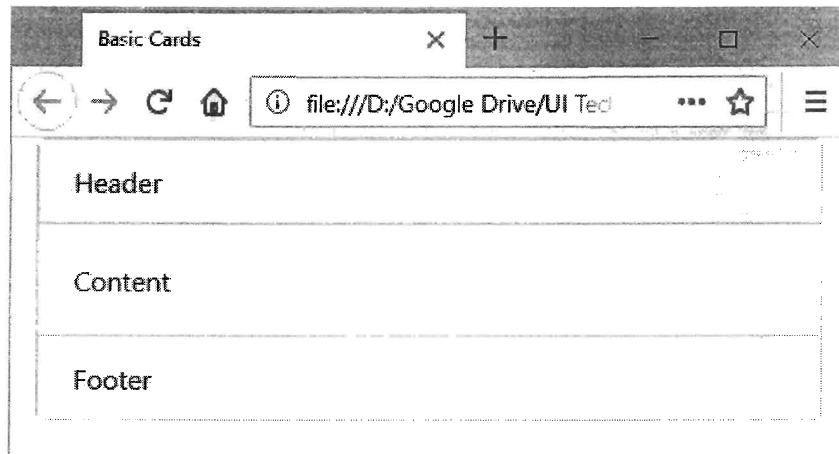
### List of Classes

- card : Represents entire card
- card-header : Header of card
- card-body : Content of card
- card-footer : Footer of card

### Example on Basic Cards

```
<html>
  <head>
    <title>Basic Cards</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
```

```
<div class="card">
  <div class="card-header">Header</div>
  <div class="card-body">Content</div>
  <div class="card-footer">Footer</div>
</div>
</div>
</body>
</html>
```



### Card Colors

- It is used to display some content (box) with header and footer.

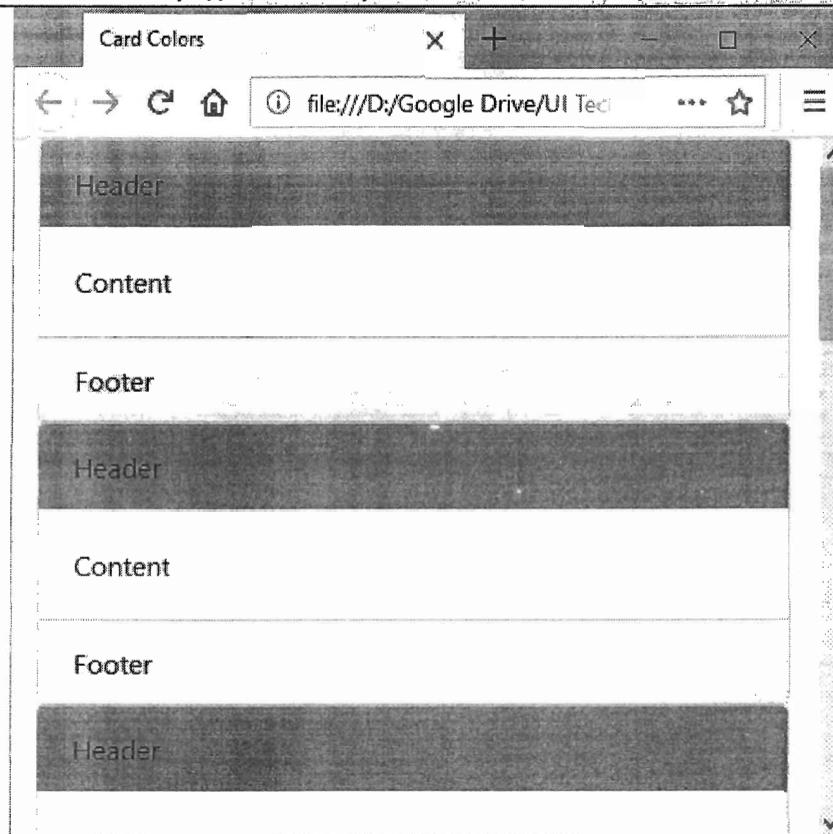
### List of Classes

- `bg-primary` : Blue color
- `bg-success` : Green color
- `bg-info` : Light blue color
- `bg-warning` : Orange color
- `bg-danger` : Red color
- `bg-secondary` : Grey color
- `bg-dark` : Black color

### Example on Card Colors

```
<html>
  <head>
    <title>Card Colors</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
```

```
</head>
<body>
<div class="container-fluid">
  <div class="card">
    <div class="card-header bg-primary">Header</div>
    <div class="card-body">Content</div>
    <div class="card-footer">Footer</div>
  </div>
  <div class="card">
    <div class="card-header bg-success">Header</div>
    <div class="card-body">Content</div>
    <div class="card-footer">Footer</div>
  </div>
  <div class="card">
    <div class="card-header bg-info">Header</div>
    <div class="card-body">Content</div>
    <div class="card-footer">Footer</div>
  </div>
  <div class="card">
    <div class="card-header bg-warning">Header</div>
    <div class="card-body">Content</div>
    <div class="card-footer">Footer</div>
  </div>
  <div class="card">
    <div class="card-header bg-danger">Header</div>
    <div class="card-body">Content</div>
    <div class="card-footer">Footer</div>
  </div>
  <div class="card">
    <div class="card-header bg-secondary">Header</div>
    <div class="card-body">Content</div>
    <div class="card-footer">Footer</div>
  </div>
  <div class="card">
    <div class="card-header bg-dark text-light">Header</div>
    <div class="card-body">Content</div>
    <div class="card-footer">Footer</div>
  </div>
  <div class="card">
    <div class="card-header bg-danger">Header</div>
    <div class="card-body">Content</div>
    <div class="card-footer">Footer</div>
  </div>
</div>
</body>
</html>
```



### Card Images

- It is used to display cards with images.

### List of Classes

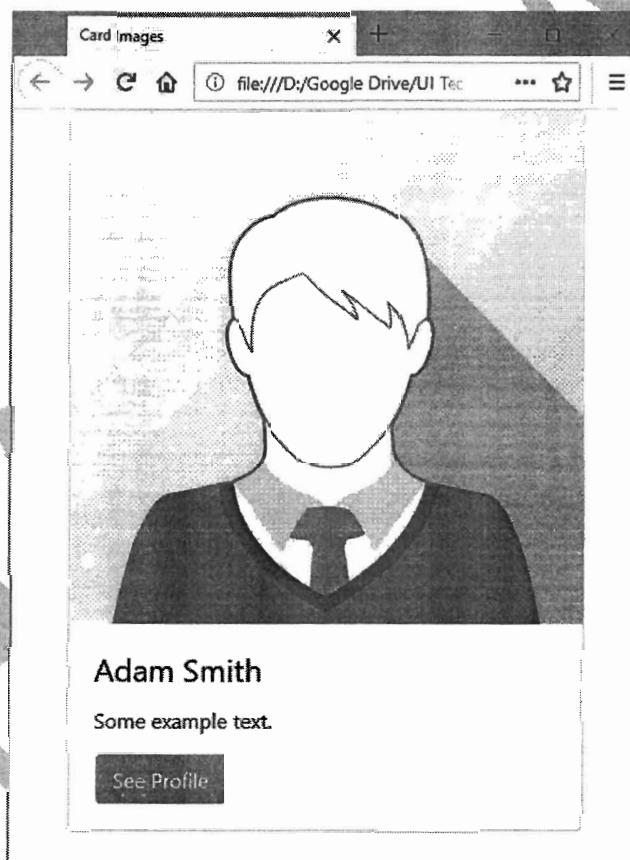
- card-img-top : Represents card image at top
- card-img-bottom : Represents card image at bottom
- card-title : Title of the card
- card-text : Normal text of the card

### Example on Card Images

```
<html>
  <head>
    <title>Card Images</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
```

```
</head>
<body>
<div class="container-fluid">
<div class="card mx-auto" style="width:400px">

<div class="card-body">
<h4 class="card-title">Adam Smith</h4>
<p class="card-text">Some example text.</p>
<a href="#" class="btn btn-primary">See Profile</a>
</div>
</div>
</div>
</body>
</html>
```



### Card Groups

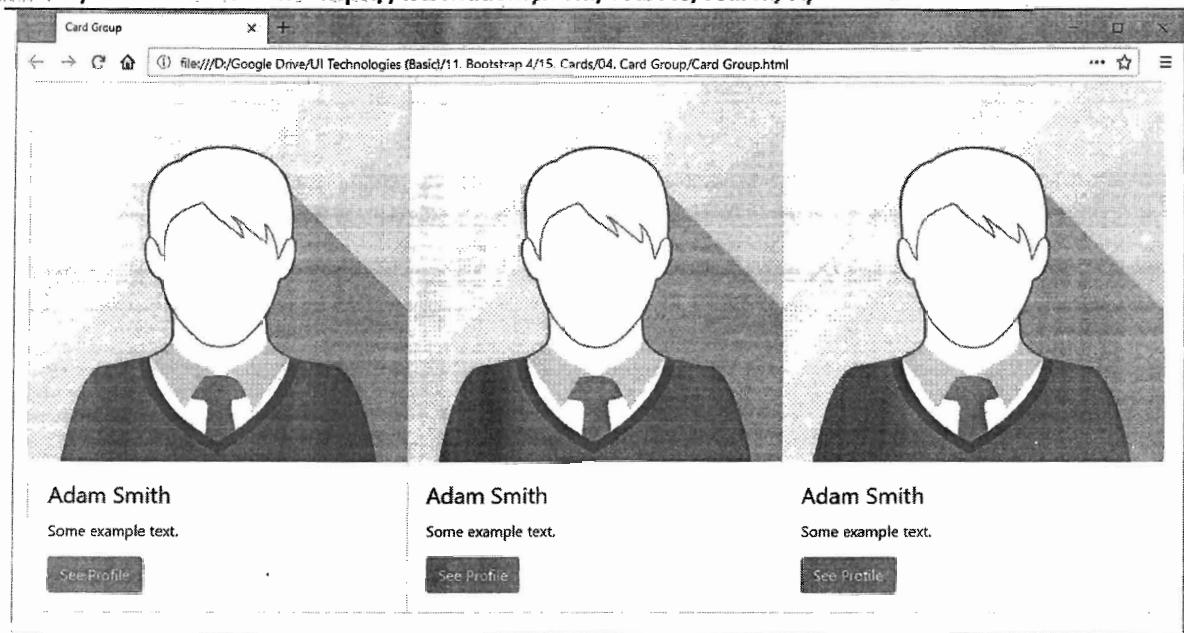
- It is used to display group of cards side by side.

### List of Classes

- card-group : Group of cards

**Example on Card Groups**

```
<html>
  <head>
    <title>Card Group</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="card-group" style="width:1200px">
        <div class="card" style="width:400px">
          
          <div class="card-body">
            <h4 class="card-title">Adam Smith</h4>
            <p class="card-text">Some example text.</p>
            <a href="#" class="btn btn-primary">See Profile</a>
          </div>
        </div>
        <div class="card" style="width:400px">
          
          <div class="card-body">
            <h4 class="card-title">Adam Smith</h4>
            <p class="card-text">Some example text.</p>
            <a href="#" class="btn btn-primary">See Profile</a>
          </div>
        </div>
        <div class="card" style="width:400px">
          
          <div class="card-body">
            <h4 class="card-title">Adam Smith</h4>
            <p class="card-text">Some example text.</p>
            <a href="#" class="btn btn-primary">See Profile</a>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```



## Tooltip

### Tooltip

- It is used to display tooltip message when the user places mouse pointer on it.

### List of Classes

- `data-toggle="tooltip"` : Enables tooltip for the element
- `data-placement="top | bottom | left | right"` : Specifies position of tooltip
- `title` : Tooltip text

### Example on Tooltip

```
<html>
  <head>
    <title>Tooltip</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <h1>Tooltip</h1>
      <a href="#" data-toggle="tooltip" data-placement="top" title="Hooray!">Top</a>
    </div>
  </body>
</html>
```

```
<a href="#" data-toggle="tooltip" data-placement="bottom" title="Hooray!">Bottom</a>
<a href="#" data-toggle="tooltip" data-placement="left" title="Hooray!">Left</a>
<a href="#" data-toggle="tooltip" data-placement="right" title="Hooray!">Right</a>
</div>
<script>
$( "[data-toggle='tooltip']" ).tooltip();
</script>
</body>
</html>
```



## Popover

### Popover

- It is used to display message (some text) when the user clicks an element.
- The popover can show when the cursor focuses the element.
- The popover can have title and content also.

### List of Classes

- |  |                         |
|--|-------------------------|
| • data-toggle="popover"                        | : Enables popover       |
| • title  | : Title of popover      |
| • data-content=" <i>some text</i> "            | : Text of popover       |
| • data-placement="top   bottom   left   right" | : Position of popover   |
| • data-trigger="focus"                         | : Show popover on focus |

### Example on Popover

```
<html>
<head>
<title>Popover</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
```

```
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">
<h1>Popover</h1>
<a href="#" data-toggle="popover" title="Popover Header" data-content="Some content inside the
popover" data-placement="right" data-trigger="focus">Toggle popover</a>
</div>
<script>
$(document).ready(function() {
  $('[data-toggle="popover"]').popover();
});
</script>
</body>
</html>
```



## Collapsible

### Basic Collapsible

- It is used to display expandable / collapsible content.
- The content will be shown when the user clicks on the button / link.

#### List of Classes

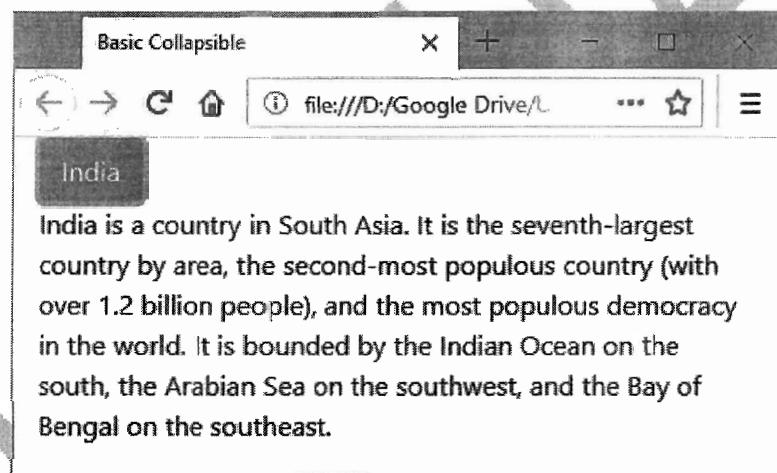
- collapse : Represents collapsible container.
- data-toggle="collapse" : Enables collapsible content, when button is clicked
- data-target="#id" : Connects the button with collapsible container.

#### Example on Basic Collapsible

```
<html>
<head>
<title>Basic Collapsible</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
```

```
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">
<button class="btn btn-primary" data-toggle="collapse" data-target="#div1">India</button>

<div id="div1" class="collapse">
  India is a country in South Asia. It is the seventh-largest country by area, the second-most populous country (with over 1.2 billion people), and the most populous democracy in the world. It is bounded by the Indian Ocean on the south, the Arabian Sea on the southwest, and the Bay of Bengal on the southeast.
</div>
</div>
</body>
</html>
```



### Link Collapsible

- It is used to display expandable / collapsible content, based on the link.
- The content will be shown when the user clicks on the link.

### List of Classes

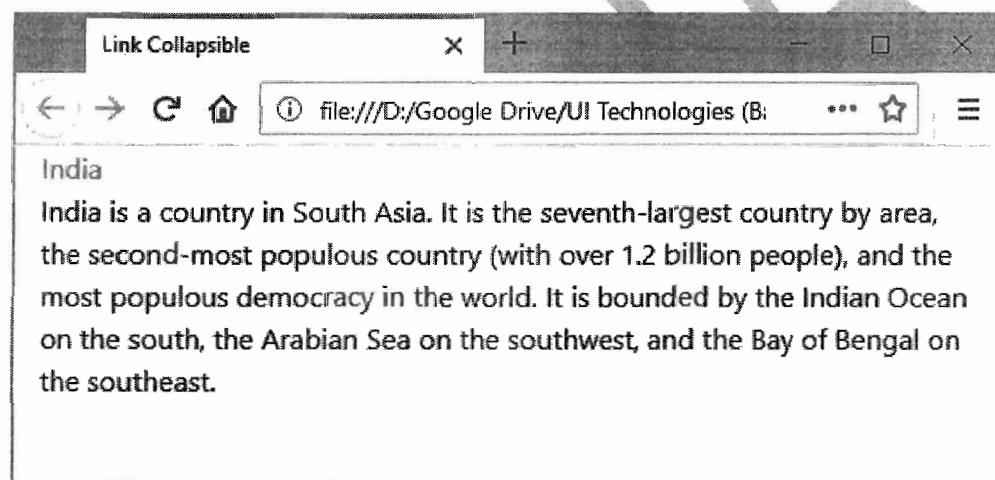
- show : Shows the collapsible content by default, while page is loading.

### Example on Link Collapsible

```
<html>
<head>
<title>Link Collapsible</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
```

```
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">
<a data-toggle="collapse" href="#div1">India</a>

<div id="div1" class="collapse show">
  India is a country in South Asia. It is the seventh-largest country by area, the second-most populous country (with over 1.2 billion people), and the most populous democracy in the world. It is bounded by the Indian Ocean on the south, the Arabian Sea on the southwest, and the Bay of Bengal on the southeast.
</div>
</div>
</body>
</html>
```



### Accordion

- It is used to display a group of collapsible items and show any one of them, when the user clicks it.
- Out of few collapsible items, only one is visible to the user.

### List of Classes

- `show` : Shows the card
- `data-parent="id"` : Specifies reference to the parent div

### Example on Accordion

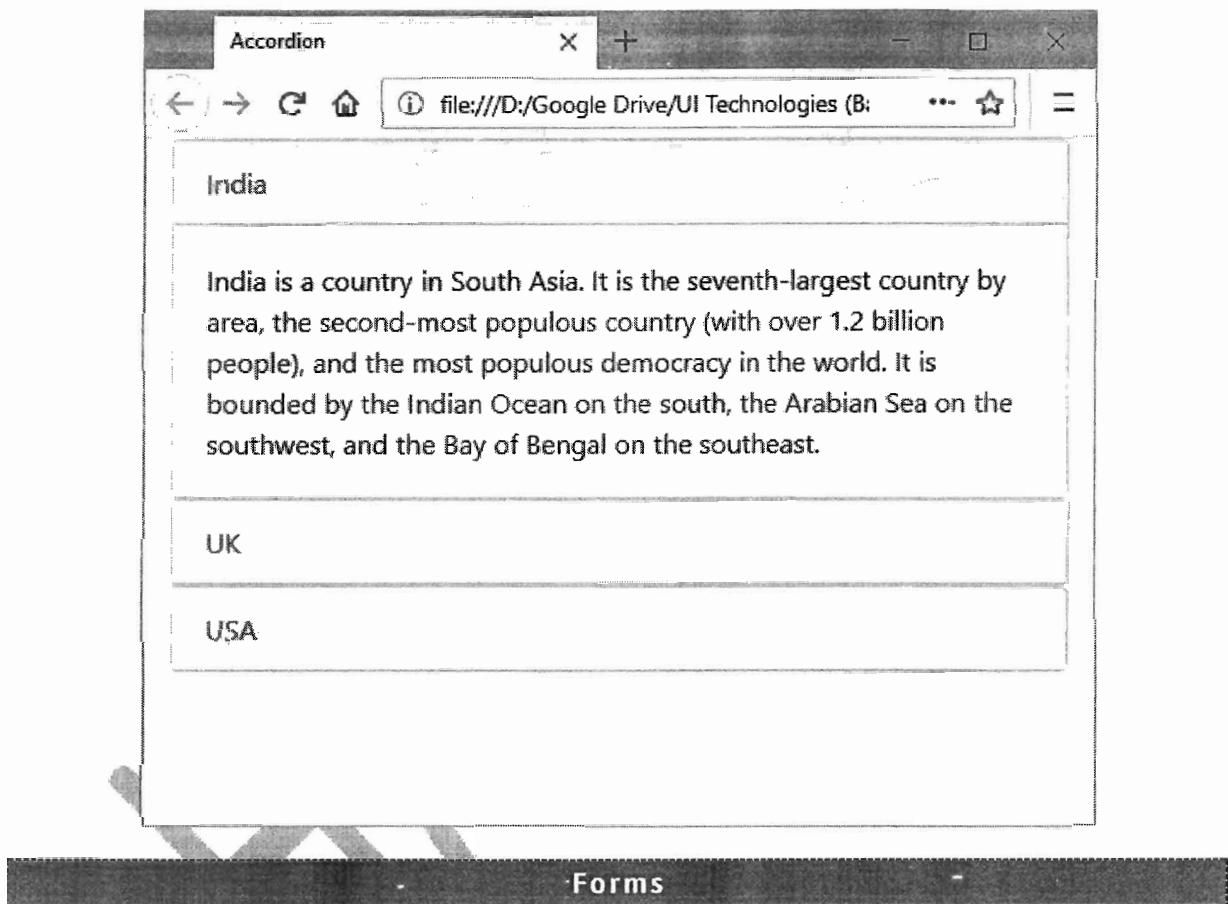
```
<html>
<head>
<title>Accordion</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
```

```
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">
<div id="accordion">
<div class="card">
<div class="card-header">
<a class="card-link" data-toggle="collapse" href="#collapseOne">
    India
</a>
</div>
<div id="collapseOne" class="collapse show" data-parent="#accordion">
<div class="card-body">
    India is a country in South Asia. It is the seventh-largest country by area, the second-most populous country (with over 1.2 billion people), and the most populous democracy in the world. It is bounded by the Indian Ocean on the south, the Arabian Sea on the southwest, and the Bay of Bengal on the southeast.
</div>
</div>
</div>

<div class="card">
<div class="card-header">
<a class="collapsed card-link" data-toggle="collapse" href="#collapseTwo">
    UK
</a>
</div>
<div id="collapseTwo" class="collapse" data-parent="#accordion">
<div class="card-body">
    The United Kingdom of Great Britain and Northern Ireland, commonly known as the United Kingdom (UK) or Britain, is a sovereign country in western Europe. Lying off the north-western coast of the European mainland, the UK includes the island of Great Britain, the north-eastern part of the island of Ireland and many smaller islands.
</div>
</div>
</div>

<div class="card">
<div class="card-header">
<a class="collapsed card-link" data-toggle="collapse" href="#collapseThree">
    USA
</a>
</div>
<div id="collapseThree" class="collapse" data-parent="#accordion">
<div class="card-body">
    The United States of America (USA), commonly known as the United States (U.S.) or America, is a federal republic composed of 50 states, a federal district, five major self-governing territories, and various possessions. At 3.8 million square miles (9.8 million km2) and with over 325 million people, the United States is the world's third- or fourth-largest country by total area and the third-most populous country.
</div>
</div>
</div>
```

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



#### Inline Form

- It is used to create a simple form (with limited no. of elements), where the elements appear side-by-side.

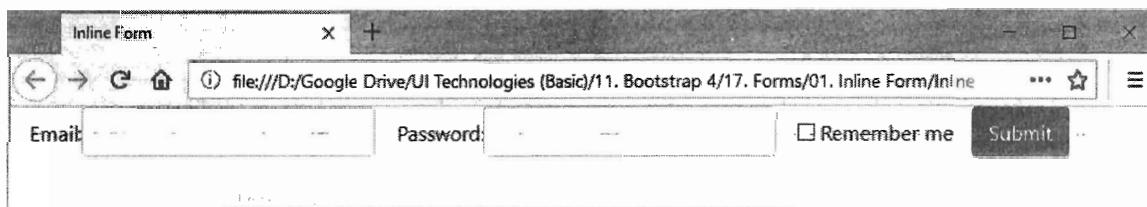
#### List of Classes

- form-inline : Creates inline form
- form-group : Wrapper for label and form element
- form-control : Represents form element
- mr-n : Margin right

- mb-n : Margin bottom
- mt-n : Margin top
- ml-n : Margin left
- form-check : Wrapper for checkbox / radio button
- form-check-label : Label for checkbox / radio button
- form-check-input : Checkbox / Radio buttn

### Example on Inline Form

```
<html>
  <head>
    <title>Inline Form</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <form class="form-inline">
        <div class="form-group mr-3">
          <label for="email">Email:</label>
          <input type="email" class="form-control" id="email">
        </div>
        <div class="form-group mr-3">
          <label for="pwd">Password:</label>
          <input type="password" class="form-control" id="pwd">
        </div>
        <div class="form-group form-check mr-3">
          <label class="form-check-label">
            <input class="form-check-input" type="checkbox">Remember me
          </label>
        </div>
        <button class="btn btn-primary">Submit</button>
      </form>
    </div>
  </body>
</html>
```



### Stacked Form

- It is used to create a form, where the form elements appear line-by-line.

#### List of Classes

- `form-text` : Plain text in the form
- `form-control-file` : Represents file browse button

#### Example on Stacked Form

```
<html>
  <head>
    <title>Stacked Form</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="row">
        <div class="col-10">
          <h1>Registration</h1>
          <form>
            <div class="form-group">
              <label for="txtEmail">Email</label>
              <input type="email" class="form-control" id="txtEmail" placeholder="Enter email">
              <small class="form-text text-muted">We'll never share your email with anyone else.</small>
            </div>

            <div class="form-group">
              <label for="txtPassword">Password</label>
              <input type="password" class="form-control" id="txtPassword" placeholder="Password">
            </div>

            <div class="form-group">
              <label for="drpCountry">Country</label>
              <select class="form-control" id="drpCountry">
                <option>Please Select</option>
                <option>India</option>
                <option>UK</option>
                <option>USA</option>
                <option>Japan</option>
                <option>China</option>
              </select>
            </div>

            <div class="form-group">
              <label for="txtComments">Comments</label>
            </div>
          </form>
        </div>
      </div>
    </div>
  </body>

```

```
<textarea class="form-control" id="txtComments" rows="3"></textarea>
</div>

<div class="form-group">
<label for="txtFile">Attachment</label>
<input type="file" class="form-control-file" id="txtFile">
</div>

<div class="form-group">
<label>Gender</label>
<div class="form-check">
<label class="form-check-label">
<input type="radio" class="form-check-input" name="rbGender" value="male">Male
</label>
</div>
<div class="form-check">
<label class="form-check-label">
<input type="radio" class="form-check-input" name="rbGender" value="female">
Female
</label>
</div>
</div>

<div class="form-group form-check">
<label class="form-check-label">
<input type="checkbox" class="form-check-input" checked="checked">
Remember Me
</label>
</div>

<button type='submit' class="btn btn-primary">Register</button>
</form>
</div>
<div class="col-2">div 2</div>
</div>
</div>
</body>
</html>
```

Stacked Form

div 2

# Registration

Email

We'll never share your email with anyone else.

Password

Country

Comments

Attachment

No file selected.

Gender

Male

Female

Remember Me

### Form Grid

- It is used to create a form, where the desired form elements appear side-by-side.

### List of Classes

- `form-row` : Creates a grid in the form

### Example on Form Grid

---

```
<html>
  <head>
    <title>Form Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <h1>Registration</h1>
      <form>
        <div class="form-row">
          <div class="form-group col-sm-6">
            <label for="txtEmail">Email</label>
            <input type="email" class="form-control" id="txtEmail" placeholder="Enter email">
          </div>
          <div class="form-group col-sm-6">
            <label for="txtPassword">Password</label>
            <input type="password" class="form-control" id="txtPassword" placeholder="Password">
          </div>
        </div>
        <div class="form-row">
          <div class="form-group col-sm-4">
            <label for="txtStreet">Street</label>
            <input type="text" class="form-control" id="txtStreet" placeholder="Street">
          </div>
          <div class="form-group col-sm-4">
            <label for="txtCity">City</label>
            <input type="text" class="form-control" id="txtCity" placeholder="City">
          </div>
          <div class="form-group col-sm-4">
            <label for="drpCountry">Country</label>
            <select class="form-control" id="drpCountry">
              <option>Please Select</option>
              <option>India</option>
              <option>UK</option>
              <option>USA</option>
              <option>Japan</option>
              <option>China</option>
            </select>
          </div>
        </div>
      </form>
    </div>
  </body>
</html>
```

```
</div>

<div class="form-group">
  <label for="txtComments">Comments</label>
  <textarea class="form-control" id="txtComments" rows="3"></textarea>
</div>

<button type="submit" class="btn btn-primary">Register</button>
</form>
</div>
</body>
</html>
```

The screenshot shows a registration form titled "Registration". The form is structured using a horizontal grid layout. It includes fields for Email (with placeholder "Enter email") and Password. Below these, there are three input fields: Street, City, and Country, arranged horizontally. The Street and City fields are simple text inputs, while the Country field is a dropdown menu with "Please Select" as the default value. Below these fields is a large text area labeled "Comments" with a placeholder. At the bottom of the form is a "Register" button.

### Horizontal Form Grid

- It is used to create a form, where the labels and form elements appear side-by-side.

#### List of Classes

- form-row : Creates a grid in the form
- col-sm-n : Specifies no. of columns in all devices above extra small devices
- col-form-label : Makes vertical alignment center
- offset-sm-n : Specifies no. of columns to leave empty

---

### Example on Horizontal Form Grid

---

```
<html>
  <head>
    <title>Horizontal Form Grid</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="row">
        <div class="col-10">
          <h1>Registration</h1>
          <form>
            <div class="form-group form-row">
              <label for="txtEmail" class="col-sm-2 col-form-label">Email</label>
              <div class="col-sm-10">
                <input type="email" class="form-control" id="txtEmail" placeholder="Enter Email">
              </div>
            </div>

            <div class="form-group form-row">
              <label for="txtPassword" class="col-sm-2 col-form-label">Password</label>
              <div class="col-sm-10">
                <input type="password" class="form-control" id="txtPassword" placeholder="Password">
              </div>
            </div>

            <div class="form-group form-row">
              <label for="drpCountry" class="col-sm-2 col-form-label">Country</label>
              <div class="col-sm-10">
                <select class="form-control" id="drpCountry">
                  <option>Please Select</option>
                  <option>India</option>
                  <option>UK</option>
                  <option>USA</option>
                  <option>Japan</option>
                  <option>China</option>
                </select>
              </div>
            </div>

            <div class="form-group form-row">
              <label class="form-label col-sm-2">Gender</label>
              <div class="col-sm-10">
                <div class="form-check form-check-inline">
                  <label class="form-check-label">
                    <input class="form-check-input" type="radio" name="gender" value="male">

```

```
        Male
        </label>
    </div>
    <div class="form-check form-check-inline">
        <label class="form-check-label">
            <input class="form-check-input" type="radio" name="gender" value="female">
        Female
        </label>
    </div>
</div>
</div>

<div class="form-group form-row">
    <label class="col-sm-2"></label>
    <div class="col-sm-10">
        <div class="form-check">
            <label class="form-check-label">
                <input class="form-check-input" type="checkbox">
            Remember me
            </label>
        </div>
    </div>
</div>

<div class="form-group form-row">
    <div class="offset-sm-2 col-sm-10">
        <button type="submit" class="btn btn-primary">Sign in</button>
    </div>
</div>

</form>
</div>
<div class="col-2">div 2</div>
</div>
</div>
</body>
</html>
```

The screenshot shows a registration form titled "Registration" within a browser window titled "Horizontal Form Grid". The form is contained within a "div 2" container. It includes fields for "Email" (with placeholder "Enter Email"), "Password" (with placeholder "Password"), "Country" (with placeholder "Please Select" and a dropdown arrow), "Gender" (with radio buttons for "Male" and "Female"), a "Remember me" checkbox, and a "Sign in" button.

### Input Groups

- It is used to create a form element, with some text inside it.

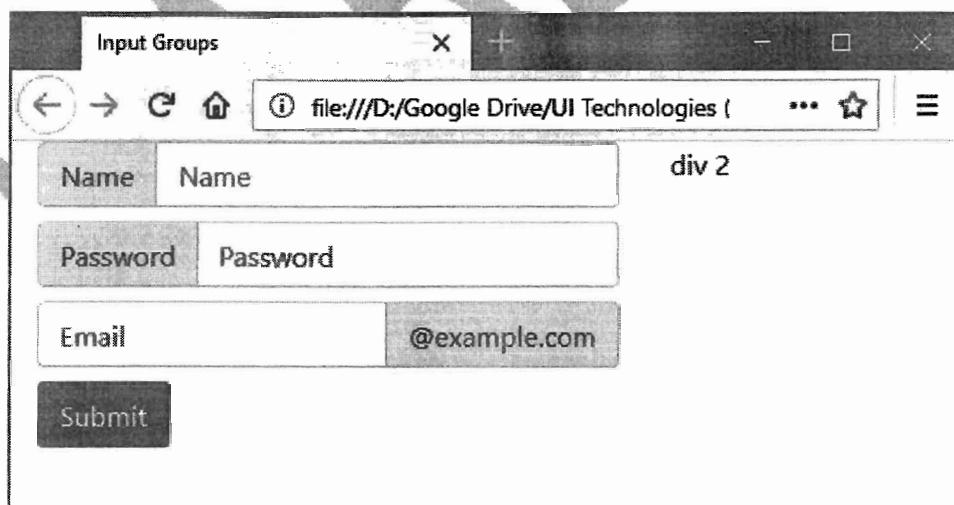
### List of Classes

- `input-group` : Container for input group (icon + form element)
- `input-group-prepend` : Displays the icon at left side of form element
- `input-group-append` : Displays the icon at right side of form element
- `input-group-text` : Displays the text as icon

### Example on Input Groups

```
<html>
  <head>
    <title>Input Groups</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="row">
        <div class="col-8">
```

```
<form>
  <div class="input-group mb-2">
    <div class="input-group-prepend">
      <span class="input-group-text">Name</span>
    </div>
    <input type="text" class="form-control" placeholder="Name">
  </div>
  <div class="input-group mb-2">
    <div class="input-group-prepend">
      <span class="input-group-text">Password</span>
    </div>
    <input type="password" class="form-control" placeholder="Password">
  </div>
  <div class="input-group mb-2">
    <input type="text" class="form-control" placeholder="Email">
    <div class="input-group-append">
      <span class="input-group-text">@example.com</span>
    </div>
  </div>
  <button class="btn btn-primary">Submit</button>
</form>
</div>
<div class="col-4">div 2</div>
</div>
</body>
</html>
```



### Form Validation

- It is used to display success message / error message, based on the value that is entered by the user.

### List of Classes

- needs-validation : Enables bootstrap validation

- novalidate="novalidate" : Disables HTML 5 built-in validation
- valid-feedback : Shows message if the value is valid
- invalid-feedback : Shows message if the value is invalid
- valid-tooltip : Shows tooltip message if the value is valid
- invalid-tooltip : Shows tooltip message if the value is invalid
- required="required" : Makes the field mandatory
- pattern="reg exp" : Specifies regular expression for validation

### Example on Form Validation

```
<html>
  <head>
    <title>Form Validation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="row">
        <div class="col-10">
          <h1>Registration</h1>
          <form class="needs-validation" novalidate="novalidate">

            <div class="form-group row">
              <label for="txtEmail" class="col-sm-2 col-form-label">Email</label>
              <div class="col-sm-10">
                <input type="email" class="form-control" id="txtEmail" placeholder="Email" required="required">
                <div class="valid-feedback">
                  Looks good!
                </div>
                <div class="invalid-feedback">
                  Invalid Email
                </div>
              </div>
            </div>

            <div class="form-group row">
              <label for="txtPassword" class="col-sm-2 col-form-label">Password</label>
              <div class="col-sm-10">
                <input type="password" class="form-control" id="txtPassword" placeholder="Password" pattern="((?=.\d)(?=.[a-z])(?=.[A-Z]).{6,15})" required="required">
                <div class="valid-feedback">
                  Great Password!
                </div>
              </div>
            </div>
          </form>
        </div>
      </div>
    </div>
  </body>

```

```
</div>
<div class="invalid-feedback">
  Weak Password
</div>
</div>
</div>

<div class="form-group row">
  <label for="drpCountry" class="col-sm-2 col-form-label">Country</label>
  <div class="col-sm-10">
    <select class="form-control" id="drpCountry" required="required">
      <option value="">Please Select</option>
      <option>India</option>
      <option>UK</option>
      <option>USA</option>
      <option>Japan</option>
      <option>China</option>
    </select>
    <div class="invalid-feedback">
      Please select country
    </div>
  </div>
</div>

<div class="form-group row">
  <div class="offset-sm-2 col-sm-10">
    <button type="submit" class="btn btn-primary">Sign in</button>
  </div>
</div>

</form>
</div>
<div class="col-2">div 2</div>
</div>
</div>

<script>
  document.getElementsByClassName("needs-validation")[0].addEventListener("submit",
    function(event)
  {
    if (event.target.checkValidity() == false)
    {
      event.preventDefault();
      event.target.classList.add("was-validated");
    }
  });
</script>
</body>
</html>
```

The screenshot shows a registration form titled "Form Validation". The form includes fields for Email, Password, and Country, each with a validation message below it. The "Email" field has an error message "Invalid Email". The "Password" field has an error message "Weak Password". The "Country" field has an error message "Please select country". A "Sign in" button is at the bottom.

Email	<input type="text" value="Email"/>
	Invalid Email
Password	<input type="text" value="Password"/>
	Weak Password
Country	<input type="text" value="Please Select"/>
	Please select country

Sign in

## Navigation Menu

### Basic Navigation Menu

- It is used to display a simple navigation bar with few hyperlinks
- When the user clicks on any hyperlink, the corresponding web page (html file) will be opened.

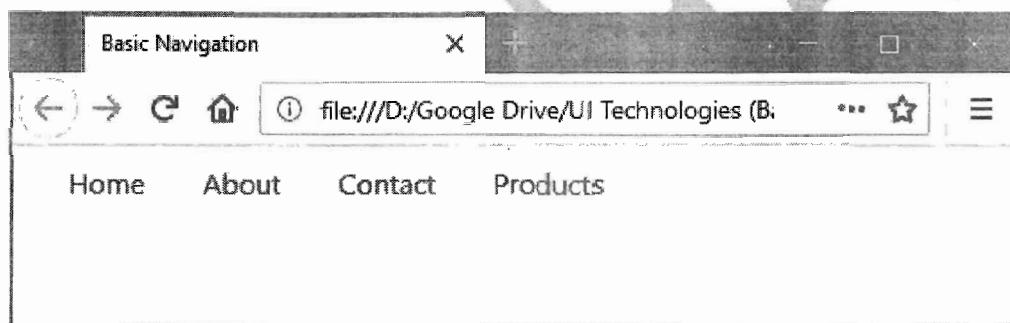
### List of Classes

- `nav` : Navigation menu
- `nav-item` : Item in the navigation
- `nav-link` : Hyperlink in the nav item

### Example on Basic Navigation Menu

```
<html>
<head>
  <title>Basic Navigation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="bootstrap.css">
  <script src="jquery-3.3.1.js"></script>
  <script src="popper.js"></script>
  <script src="bootstrap.js"></script>
</head>
<body>
  <div class="container-fluid">
```

```
<ul class="nav">
  <li class="nav-item">
    <a class="nav-link" href="#">Home</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">About</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Contact</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Products</a>
  </li>
</ul>
</div>
</body>
</html>
```



### Navigation Menu Alignment

- It is used to set alignment for the simple navigation bar.
- Default is left alignment.
- You can set center / right alignment.

### List of Classes

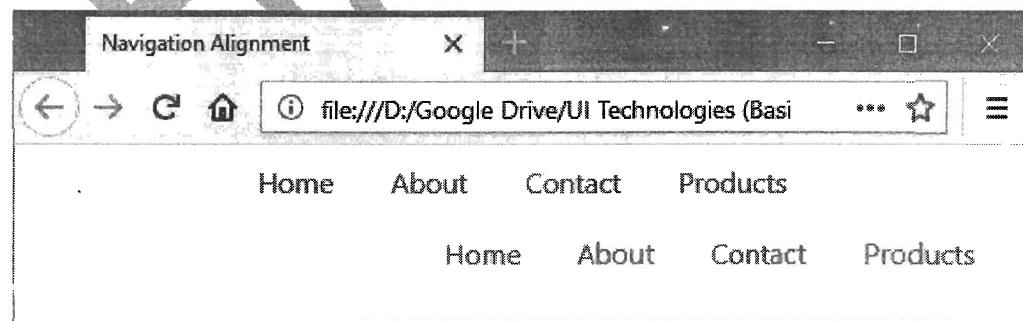
- justify-content-center : Displays navigation menu with center alignment
- justify-content-end : Displays navigation menu with right alignment

### Example on Navigation Menu Alignment

```
<html>
  <head>
    <title>Navigation Alignment</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
```

```
</head>
<body>
  <div class="container-fluid">
    <ul class="nav justify-content-center">
      <li class="nav-item">
        <a class="nav-link" href="#">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">About</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Contact</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Products</a>
      </li>
    </ul>

    <ul class="nav justify-content-end">
      <li class="nav-item">
        <a class="nav-link" href="#">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">About</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Contact</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Products</a>
      </li>
    </ul>
  </div>
</body>
</html>
```



### Vertical Navigation Menu

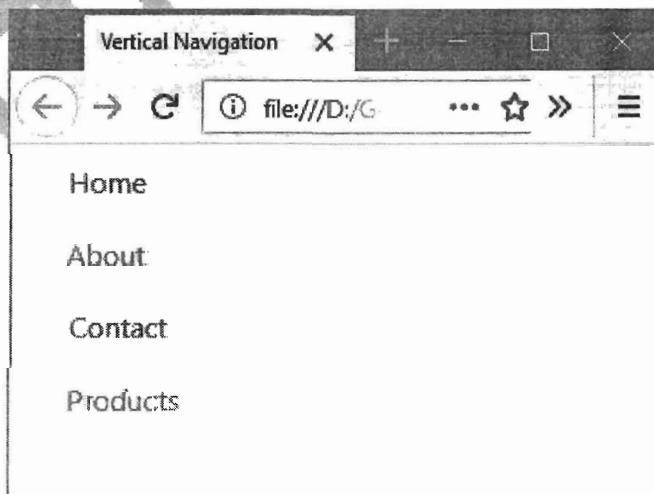
- It is used to display the simple navigation bar in vertical mode.

### List of Classes

- flex-column : Vertical navigation menu

### Example on Vertical Navigation Menu

```
<html>
  <head>
    <title>Vertical Navigation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <ul class="nav flex-column">
        <li class="nav-item">
          <a class="nav-link" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">About</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Contact</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Products</a>
        </li>
      </ul>
    </div>
  </body>
</html>
```



### Tabs

- It is used to display a set of tabs and show the corresponding content below, when the user clicks on any one tab.

#### List of Classes

- nav-tabs : Enables tabs
- data-toggle="tab" : Represents link for tab
- tab-content : Container for tabs
- tab-pane : Represents a single tab
- container : Represents a container tab
- active : Represents hyperlink of current tab
- fade : Shows the tab content with fade effect

#### Example on Tabs

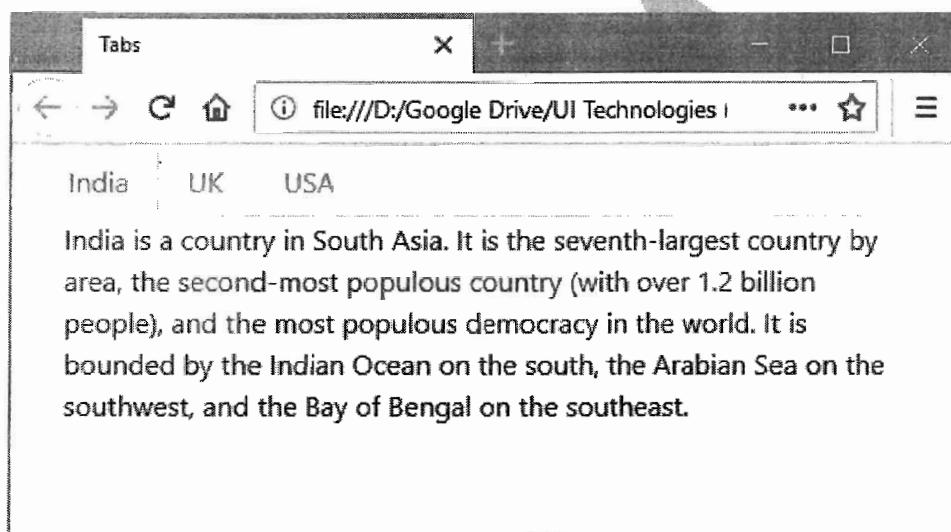
```
<html>
  <head>
    <title>Tabs</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <ul class="nav nav-tabs">
        <li class="nav-item">
          <a class="nav-link active" data-toggle="tab" href="#menu1">India</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" data-toggle="tab" href="#menu2">UK</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" data-toggle="tab" href="#menu3">USA</a>
        </li>
      </ul>

      <div class="tab-content">
        <div class="tab-pane container active" id="menu1">India is a country in South Asia. It is the seventh-largest country by area, the second-most populous country (with over 1.2 billion people), and the most populous democracy in the world. It is bounded by the Indian Ocean on the south, the Arabian Sea on the southwest, and the Bay of Bengal on the southeast.</div>
        <div class="tab-pane container fade" id="menu2">The United Kingdom of Great Britain and Northern Ireland, commonly known as the United Kingdom (UK) or Britain, is a sovereign country in</div>
      </div>
    </div>
  </body>
</html>
```

western Europe. Lying off the north-western coast of the European mainland, the UK includes the island of Great Britain, the north-eastern part of the island of Ireland and many smaller islands.</div>

<div class="tab-pane container fade" id="menu3">The United States of America (USA), commonly known as the United States (U.S.) or America, is a federal republic composed of 50 states, a federal district, five major self-governing territories, and various possessions. At 3.8 million square miles (9.8 million km<sup>2</sup>) and with over 325 million people, the United States is the world's third- or fourth-largest country by total area and the third-most populous country.</div>

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



## Pills

- It is used to display a set of tabs with more rounded corners and background color.

### List of Classes

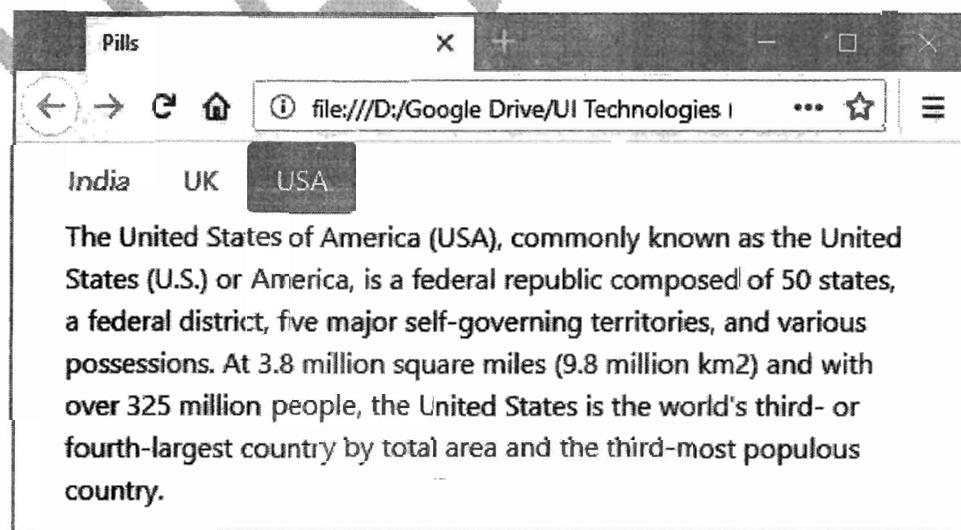
- nav-pills : Enables pills
- data-toggle="pill" : Represents hyperlink for the pill

### Example on Pills

```
<html>
<head>
<title>Pills</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
```

```
<div class="container-fluid">
  <ul class="nav nav-pills">
    <li class="nav-item">
      <a class="nav-link active" data-toggle="pill" href="#menu1">India</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" data-toggle="pill" href="#menu2">UK</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" data-toggle="pill" href="#menu3">USA</a>
    </li>
  </ul>

  <div class="tab-content">
    <div class="tab-pane container active" id="menu1">India is a country in South Asia. It is the seventh-largest country by area, the second-most populous country (with over 1.2 billion people), and the most populous democracy in the world. It is bounded by the Indian Ocean on the south, the Arabian Sea on the southwest, and the Bay of Bengal on the southeast.</div>
    <div class="tab-pane container fade" id="menu2">The United Kingdom of Great Britain and Northern Ireland, commonly known as the United Kingdom (UK) or Britain, is a sovereign country in western Europe. Lying off the north-western coast of the European mainland, the UK includes the island of Great Britain, the north-eastern part of the island of Ireland and many smaller islands.</div>
    <div class="tab-pane container fade" id="menu3">The United States of America (USA), commonly known as the United States (US.) or America, is a federal republic composed of 50 states, a federal district, five major self-governing territories, and various possessions. At 3.8 million square miles (9.8 million km2) and with over 325 million people, the United States is the world's third- or fourth-largest country by total area and the third-most populous country.</div>
  </div>
</div>
</body>
</html>
```



### Tabs with DropDown

- It is used to display dropdown menu in the tabs.

#### List of Classes

- dropdown : Enables Dropdown
- dropdown-toggle : Shows Caret symbol
- data-toggle="dropdown" : Shows dropdown, when the link is clicked
- dropdown-menu : Represents the dropdown menu to show
- dropdown-item : Represents dropdown menu item

#### Example on DropDown

```
<html>
  <head>
    <title>Tabs with Dropdown</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <ul class="nav nav-tabs">
        <li class="nav-item">
          <a class="nav-link active" data-toggle="tab" href="#menu1">India</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" data-toggle="tab" href="#menu2">UK</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" data-toggle="tab" href="#menu3">USA</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#">Cities</a>
          <div class="dropdown-menu">
            <a class="dropdown-item" data-toggle="tab" href="#menu4">Hyderabad</a>
            <a class="dropdown-item" data-toggle="tab" href="#menu5">New Delhi</a>
            <a class="dropdown-item" data-toggle="tab" href="#menu6">New York</a>
          </div>
        </li>
      </ul>

      <div class="tab-content">
        <div class="tab-pane container active" id="menu1">India is a country in South Asia. It is the seventh-largest country by area, the second-most populous country (with over 1.2 billion people), and the most
      </div>
    </div>
  </body>
</html>
```

populous democracy in the world. It is bounded by the Indian Ocean on the south, the Arabian Sea on the southwest, and the Bay of Bengal on the southeast.

<div class="tab-pane container fade" id="menu2">The United Kingdom of Great Britain and Northern Ireland, commonly known as the United Kingdom (UK) or Britain, is a sovereign country in western Europe. Lying off the north-western coast of the European mainland, the UK includes the island of Great Britain, the north-eastern part of the island of Ireland and many smaller islands.</div>

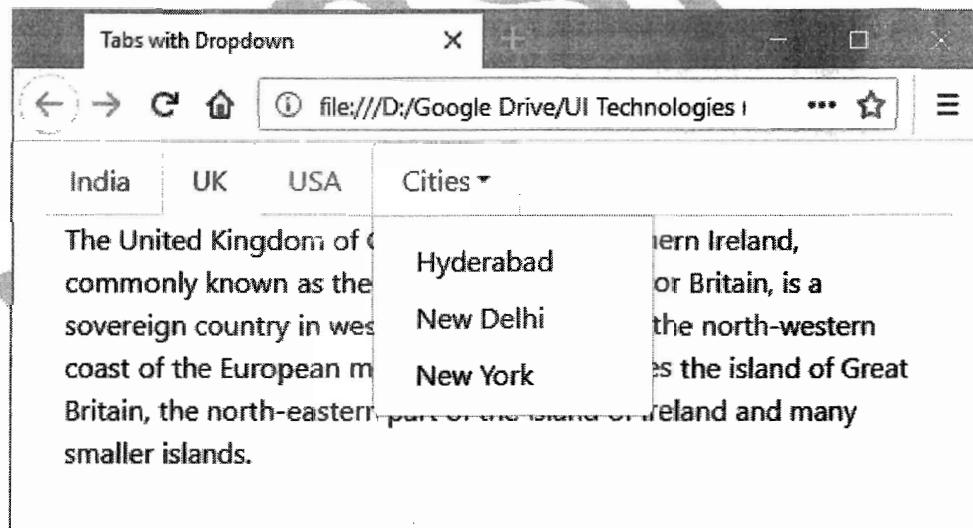
The United States of America (USA), commonly known as the United States (U.S.) or America, is a federal republic composed of 50 states, a federal district, five major self-governing territories, and various possessions. At 3.8 million square miles (9.8 million km<sup>2</sup>) and with over 325 million people, the United States is the world's third- or fourth-largest country by total area and the third-most populous country.

Hyderabad is the capital of the Indian state of Telangana and de jure capital of Andhra Pradesh.

<div class="tab-pane container fade" id="menu5">New Delhi is an urban district of Delhi which serves as the capital of India and seat of all three branches of Government of India.</div>

<div class="tab-pane container fade" id="menu6">The City of New York, often called New York City (NYC) or simply New York, is the most populous city in the United States. With an estimated 2017 population of 8,622,698 distributed over a land area of about 302.6 square miles (784 km<sup>2</sup>), New York City is also the most densely populated major city in the United States.</div>

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



## Navigation Bar

## Basic Navigation Bar

- It is used to display a complex navigation bar (NavBar) with few hyperlinks.
  - When the user clicks on any hyperlink, the corresponding web page (html file) will be opened.
  - NavBar can shrink when we reduce the width of the browser / in mobile devices, automatically.

- NavBar supports to display website logo (text / image).
- NavBar supports to display form elements such as textboxes etc.

### List of Classes

- navbar : Creates a navbar
- navbar-expand-sm : Expands the navigation bar
- navbar-dark : Black color for navigation bar
- navbar-nav : Navigation menu inside the navigation bar
- nav-item : Nav menu item
- nav-link : Link in the Nav menu item
- active : Represents current Nav menu item

### Example on Basic Navigation Bar

#### home.html

```
<html>
  <head>
    <title>Home</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <nav class="navbar navbar-expand-sm bg-success navbar-dark">
        <ul class="navbar-nav">
          <li class="nav-item">
            <a class="nav-link active" href="home.html">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="about.html">About</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="contact.html">Contact</a>
          </li>
        </ul>
      </nav>
      Home page content here
    </div>
  </body>
</html>
```

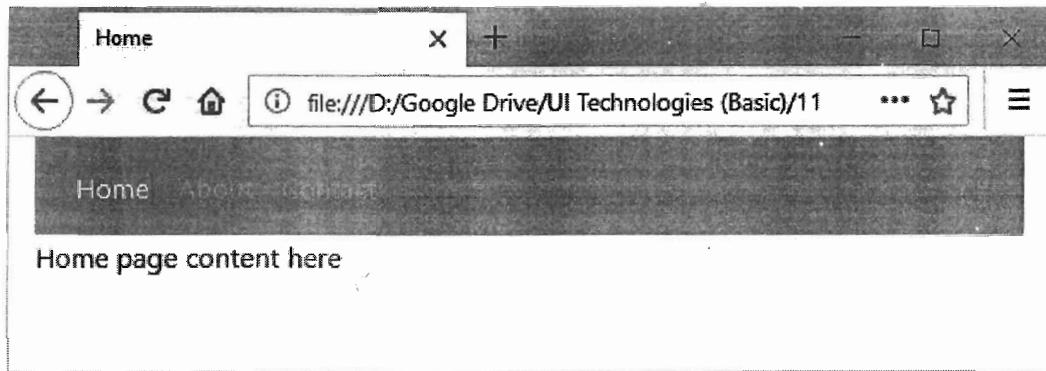
[about.html](#)

```
<html>
  <head>
    <title>About</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <nav class="navbar navbar-expand-sm bg-success navbar-dark">
        <ul class="navbar-nav">
          <li class="nav-item">
            <a class="nav-link" href="home.html">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link active" href="about.html">About</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="contact.html">Contact</a>
          </li>
        </ul>
      </nav>
      About page content here
    </div>
  </body>
</html>
```

[contact.html](#)

```
<html>
  <head>
    <title>Contact</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <nav class="navbar navbar-expand-sm bg-success navbar-dark">
        <ul class="navbar-nav">
          <li class="nav-item">
            <a class="nav-link" href="home.html">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="about.html">About</a>
          </li>
        </ul>
      </nav>
    </div>
  </body>
</html>
```

```
</li>
<li class="nav-item">
  <a class="nav-link active" href="contact.html">Contact</a>
</li>
</ul>
</nav>
Contact page content here
</div>
</body>
</html>
```



### NavBar Collapsible

- It is used to display a collapsible navbar. That means when the web page has been opened in small devices, the menu will be automatically converted into "=" icon. When the user clicks on this icon, the menu gets opened.

### List of Classes

- navbar-brand : Represents logo of navbar
- navbar-toggler : Displays icon for Toggle button for navbar
- data-toggle="collapse" : Toggle functionality for
- data-target="#id" : Creates connection with toggle button and nav menu
- collapse : Collapses the menu, by default
- navbar-collapse : Avoids the position problem of collapse button
- navbar-toggler-icon : Style for display icon

### Example on NavBar Collapsible

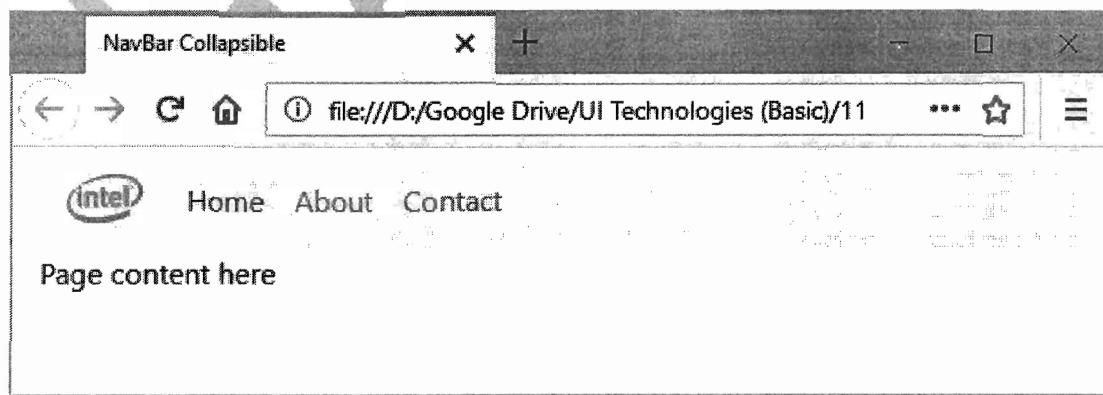
```
<html>
<head>
  <title>NavBar Collapsible</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="bootstrap.css">
  <script src="jquery-3.3.1.js"></script>
```

```
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">
<nav class="navbar navbar-expand-sm bg-light navbar-light">
<a class="navbar-brand" href="#">

</a>

<button class="navbar-toggler" data-toggle="collapse" data-target="#collapsibleNavbar">
<span class="navbar-toggler-icon"></span>
</button>

<div class="collapse navbar-collapse" id="collapsibleNavbar">
<ul class="navbar-nav">
<li class="nav-item">
<a class="nav-link active" href="#">Home</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">About</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">Contact</a>
</li>
</ul>
</div>
</nav>
Page content here
</div>
</body>
</html>
```



#### NavBar DropDown

- It is used to display a dropdown menu for the navigation bar item.

---

### List of Classes

---

- dropdown : Menu item with dropdown
- dropdown-toggle : Caret symbol
- data-toggle="dropdown" : Shows dropdown menu when clicked
- dropdown-menu : Container for dropdown menu
- dropdown-item : Item in the dropdown menu

### Example on NavBar DropDown

---

```
<html>
<head>
<title>NavBar Dropdown</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">
<nav class="navbar navbar-expand-sm bg-light navbar-light">
<a class="navbar-brand" href="#">

</a>

<button class="navbar-toggler" data-toggle="collapse" data-target="#collapsibleNavbar">
<span class="navbar-toggler-icon"></span>
</button>

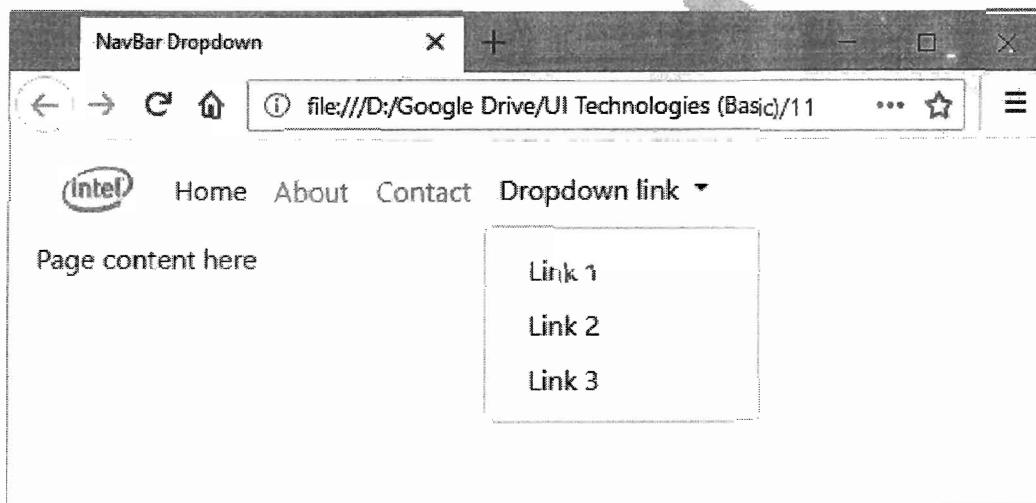
<div class="collapse navbar-collapse" id="collapsibleNavbar">
<ul class="navbar-nav">
<li class="nav-item">
<a class="nav-link active" href="#">Home</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">About</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">Contact</a>
</li>

<li class="nav-item dropdown">
<a class="nav-link dropdown-toggle" href="#" data-toggle="dropdown">
Dropdown link
</a>
<div class="dropdown-menu">
<a class="dropdown-item" href="#">Link 1</a>
<a class="dropdown-item" href="#">Link 2</a>

```

```
<a class="dropdown-item" href="#">Link 3</a>
</div>
</li>
</ul>
</div>

</nav>
Page content here
</div>
</body>
</html>
```



### NavBar Search

- It is used to display a search box for the navigation bar.

### List of Classes

- form-inline : Inline form
- form-control : Form textbox
- mr-sm-n : Margin right

### Example on NavBar Search

```
<html>
<head>
<title>NavBar Search</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
```

```
<body>
<div class="container-fluid">
<nav class="navbar navbar-expand-sm bg-light navbar-light">
<a class="navbar-brand" href="#">

</a>

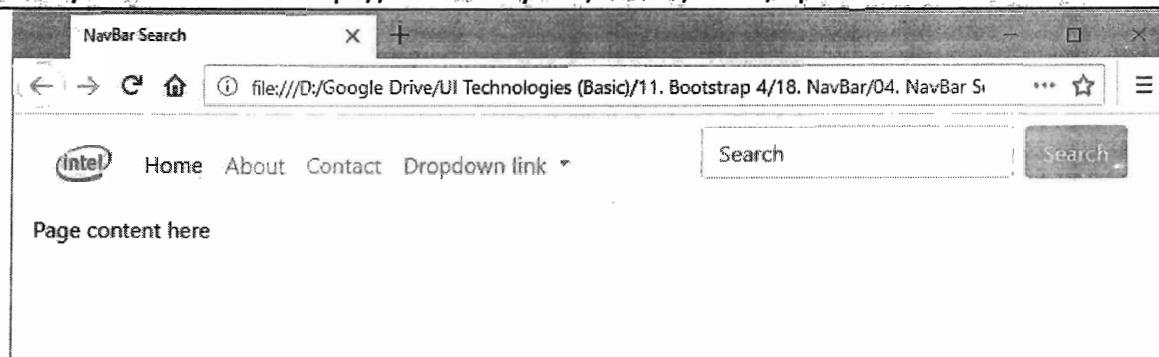
<button class="navbar-toggler" data-toggle="collapse" data-target="#collapsibleNavbar">
<span class="navbar-toggler-icon"></span>
</button>

<div class="collapse navbar-collapse" id="collapsibleNavbar">
<ul class="navbar-nav">
<li class="nav-item">
<a class="nav-link active" href="#">Home</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">About</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">Contact</a>
</li>

<li class="nav-item dropdown">
<a class="nav-link dropdown-toggle" href="#" data-toggle="dropdown">
Dropdown link
</a>
<div class="dropdown-menu">
<a class="dropdown-item" href="#">Link 1</a>
<a class="dropdown-item" href="#">Link 2</a>
<a class="dropdown-item" href="#">Link 3</a>
</div>
</li>
</ul>
</div>

<form class="form-inline">
<input type="text" class="form-control mr-sm-2" placeholder="Search">
<button class="btn btn-success">Search</button>
</form>

</nav>
Page content here
</div>
</body>
</html>
```



### NavBar FixedTop

- It is used to display a navbar always at the top of the page, even though the user has scrolled the page up / down.

#### List of Classes

- `fixed-top` : Fixes the navbar at top
- `fixed-bottom` : Fixes the navbar at bottom

#### Example on NavBar FixedTop

```
<html>
  <head>
    <title>NavBar FixedTop</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <nav class="navbar navbar-expand-sm bg-light navbar-light fixed-top">
        <a class="navbar-brand" href="#">
          
        </a>

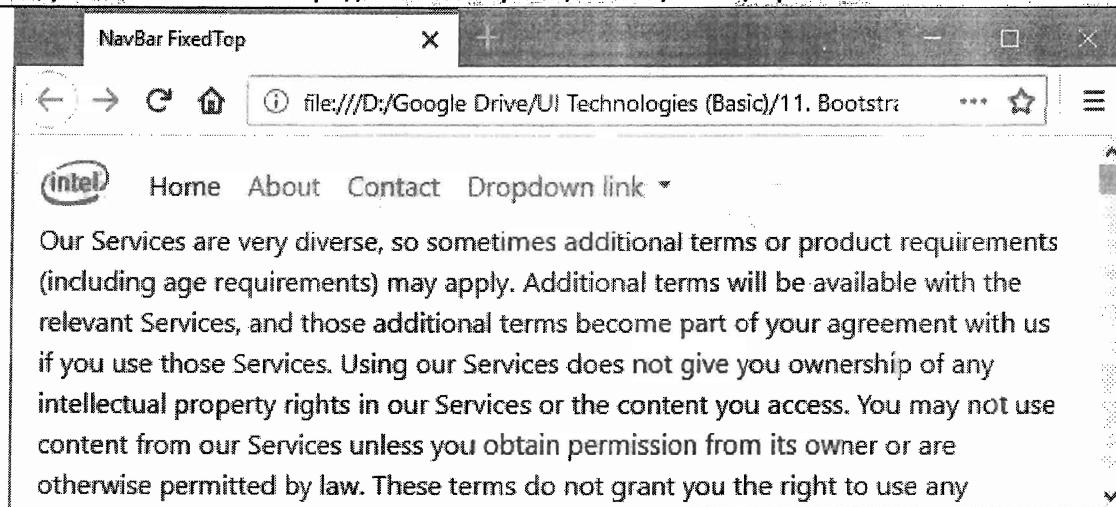
        <button class="navbar-toggler" data-toggle="collapse" data-target="#collapsibleNavbar">
          <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="collapsibleNavbar">
          <ul class="navbar-nav">
            <li class="nav-item">
              <a class="nav-link active" href="#">Home</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">About</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Contact</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Dropdown link</a>
            </li>
          </ul>
        </div>
      </nav>
    </div>
  </body>
</html>
```

```
<a class="nav-link" href="#">About</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="#">Contact</a>
</li>

<li class="nav-item dropdown">
  <a class="nav-link dropdown-toggle" href="#" data-toggle="dropdown">
    Dropdown link
  </a>
  <div class="dropdown-menu">
    <a class="dropdown-item" href="#">Link 1</a>
    <a class="dropdown-item" href="#">Link 2</a>
    <a class="dropdown-item" href="#">Link 3</a>
  </div>
</li>
</ul>
</div>
</nav>

<p class="mt-5">Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services. Using our Services does not give you ownership of any intellectual property rights in our Services or the content you access. You may not use content from our Services unless you obtain permission from its owner or are otherwise permitted by law. These terms do not grant you the right to use any branding or logos used in our Services. Don't remove, obscure, or alter any legal notices displayed in or along with our Services. Using our Services does not give you ownership of any intellectual property rights in our Services or the content you access. You may not use content from our Services unless you obtain permission from its owner or are otherwise permitted by law. These terms do not grant you the right to use any branding or logos used in our Services. Don't remove, obscure, or alter any legal notices displayed in or along with our Services. Using our Services does not give you ownership of any intellectual property rights in our Services or the content you access. You may not use content from our Services unless you obtain permission from its owner or are otherwise permitted by law. These terms do not grant you the right to use any branding or logos used in our Services. Don't remove, obscure, or alter any legal notices displayed in or along with our Services.</p>
</div>
</body>
</html>
```



### NavBar Sticky Top

- By default, the navbar appears in the middle of the page; and the navbar will be displayed always at the top of the page, only when the user has scrolled the page up / down.

#### List of Classes

- sticky-top : Fixes the navbar at top when subsequent content is scrolled

#### Example on NavBar Sticky-Top

```
<html>
  <head>
    <title>NavBar Sticky Top</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container-fluid">
      <div class="jumbotron">
        <h1>Welcome</h1>
        <p>Thanks for visiting the website</p>
      </div>

      <nav class="navbar navbar-expand-sm bg-light navbar-light sticky-top">
        <a class="navbar-brand" href="#">
          
        </a>

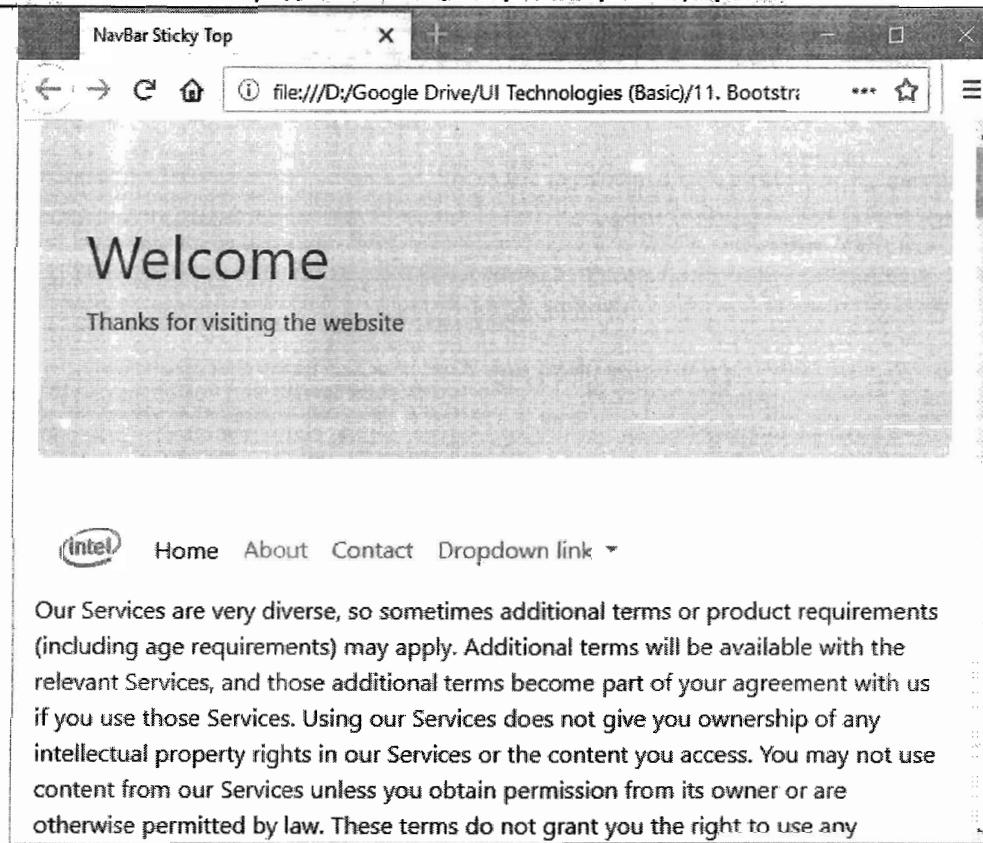
        <button class="navbar-toggler" data-toggle="collapse" data-target="#collapsibleNavbar">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="collapsibleNavbar">
          <ul class="navbar-nav">
            <li class="nav-item active">
              <a class="nav-link" href="#">Home
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">About
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Contact
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Dropdown link
            </li>
          </ul>
        </div>
      </nav>
    </div>
  </body>
</html>
```

```
</button>

<div class="collapse navbar-collapse" id="collapsibleNavbar">
  <ul class="navbar-nav">
    <li class="nav-item">
      <a class="nav-link active" href="#">Home</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">About</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Contact</a>
    </li>

    <li class="nav-item dropdown">
      <a class="nav-link dropdown-toggle" href="#" data-toggle="dropdown">
        Dropdown link
      </a>
      <div class="dropdown-menu">
        <a class="dropdown-item" href="#">Link 1</a>
        <a class="dropdown-item" href="#">Link 2</a>
        <a class="dropdown-item" href="#">Link 3</a>
      </div>
    </li>
  </ul>
</div>
</nav>

<p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services. Using our Services does not give you ownership of any intellectual property rights in our Services or the content you access. You may not use content from our Services unless you obtain permission from its owner or are otherwise permitted by law. These terms do not grant you the right to use any branding or logos used in our Services. Don't remove, obscure, or alter any legal notices displayed in or along with our Services. Using our Services does not give you ownership of any intellectual property rights in our Services or the content you access. You may not use content from our Services unless you obtain permission from its owner or are otherwise permitted by law. These terms do not grant you the right to use any branding or logos used in our Services. Don't remove, obscure, or alter any legal notices displayed in or along with our Services. Using our Services does not give you ownership of any intellectual property rights in our Services or the content you access. You may not use content from our Services unless you obtain permission from its owner or are otherwise permitted by law. These terms do not grant you the right to use any branding or logos used in our Services. Don't remove, obscure, or alter any legal notices displayed in or along with our Services.</p>
</div>
</body>
</html>
```



### Scrollspy

- It is used to change the "active" class to the current menu item, when the web page is scrolled vertically.

### List of Classes

- **data-spy="scroll"** : Sets "active" class for the menu item when the content is scrolled.
- **data-target=".navbar"** : Creates connection with navigation bar.
- **data-offset="pixels"** : Ignores the specified pixels at top.

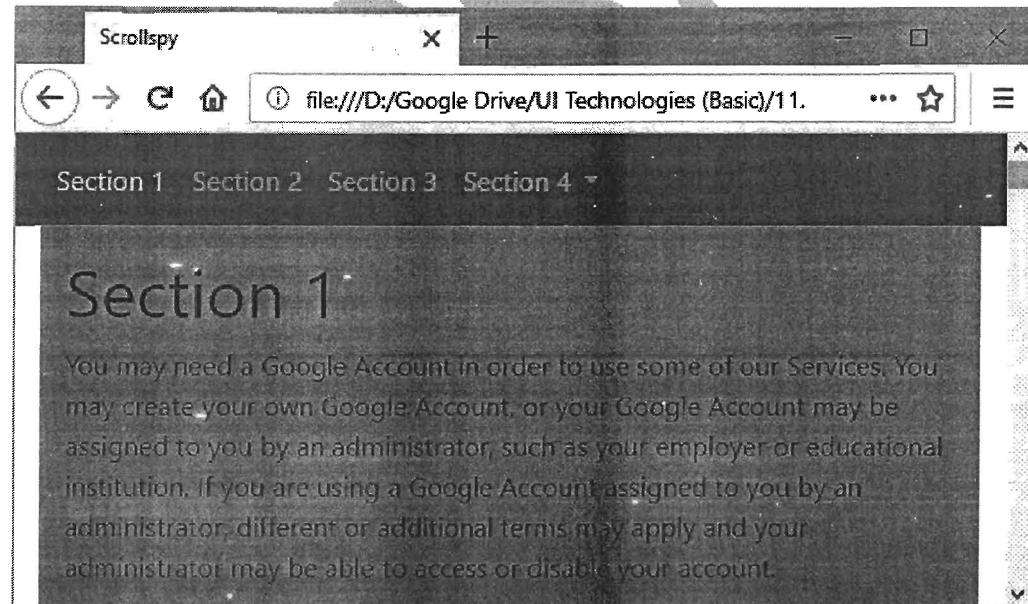
### Example on ScrollSpy

```
<html>
  <head>
    <title>Scrollspy</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
```

```
<body data-spy="scroll" data-target=".navbar" data-offset="50">
<div class="container-fluid">
<nav class="navbar navbar-expand-sm bg-dark navbar-dark fixed-top">
<ul class="navbar-nav">
<li class="nav-item">
<a class="nav-link" href="#section1">Section 1</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#section2">Section 2</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#section3">Section 3</a>
</li>
<li class="nav-item dropdown">
<a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-toggle="dropdown">
Section 4
</a>
<div class="dropdown-menu">
<a class="dropdown-item" href="#section41">Link 1</a>
<a class="dropdown-item" href="#section42">Link 2</a>
</div>
</li>
</ul>
</nav>

<div id="section1" class="container-fluid bg-success" style="padding-top:70px;padding-bottom:70px">
<h1>Section 1</h1>
<p>You may need a Google Account in order to use some of our Services. You may create your own Google Account, or your Google Account may be assigned to you by an administrator, such as your employer or educational institution. If you are using a Google Account assigned to you by an administrator, different or additional terms may apply and your administrator may be able to access or disable your account.</p>
</div>
<div id="section2" class="container-fluid bg-warning" style="padding-top:70px;padding-bottom:70px">
<h1>Section 2</h1>
<p>You may need a Google Account in order to use some of our Services. You may create your own Google Account, or your Google Account may be assigned to you by an administrator, such as your employer or educational institution. If you are using a Google Account assigned to you by an administrator, different or additional terms may apply and your administrator may be able to access or disable your account.</p>
</div>
<div id="section3" class="container-fluid bg-secondary" style="padding-top:70px;padding-bottom:70px">
<h1>Section 3</h1>
<p>You may need a Google Account in order to use some of our Services. You may create your own Google Account, or your Google Account may be assigned to you by an administrator, such as your employer or educational institution. If you are using a Google Account assigned to you by an administrator, different or additional terms may apply and your administrator may be able to access or disable your account.</p>
</div>
```

```
</div>
<div id="section41" class="container-fluid bg-danger" style="padding-top:70px;padding-bottom:70px">
  <h1>Section 4 Submenu 1</h1>
  <p>You may need a Google Account in order to use some of our Services. You may create your own Google Account, or your Google Account may be assigned to you by an administrator, such as your employer or educational institution. If you are using a Google Account assigned to you by an administrator, different or additional terms may apply and your administrator may be able to access or disable your account.</p>
</div>
<div id="section42" class="container-fluid bg-info" style="padding-top:70px;padding-bottom:70px">
  <h1>Section 4 Submenu 2</h1>
  <p>You may need a Google Account in order to use some of our Services. You may create your own Google Account, or your Google Account may be assigned to you by an administrator, such as your employer or educational institution. If you are using a Google Account assigned to you by an administrator, different or additional terms may apply and your administrator may be able to access or disable your account.</p>
</div>
</div>
</body>
</html>
```



## Carousel

### Carousel

- It is used to display image slide show with / without text.

---

### List of Classes

---

- **carousel** : Creates Carousel
- **slide** : Slide effect
- **data-ride="carousel"** : Starts slideshow as soon as web page is loaded
- **data-interval="milli seconds"** : Specifies slide time in the form of milli seconds
- **carousel-indicators** : Indicators at bottom of carousel
- **data-target="#id"** : Creates connection between indicator and carousel
- **data-slide-to="n"** : Specifies slide number for the indicator
- **active** : Represents current indicator
- **carousel-inner** : Container for carousel slides
- **carousel-item** : Carousel slide
- **data-slide="prev"** : Previous button
- **data-slide="next"** : Next button
- **carousel-control-prev** : Link for previous button
- **carousel-control-prev-icon** : Icon for previous button
- **carousel-caption** : Carousel slide text

---

### Example on Carousel

---

```
<html>
  <head>
    <title>Carousel</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></script>
  </head>
  <body>
    <div class="container">
      <div id="demo" class="carousel slide" data-ride="carousel" data-interval="3000">

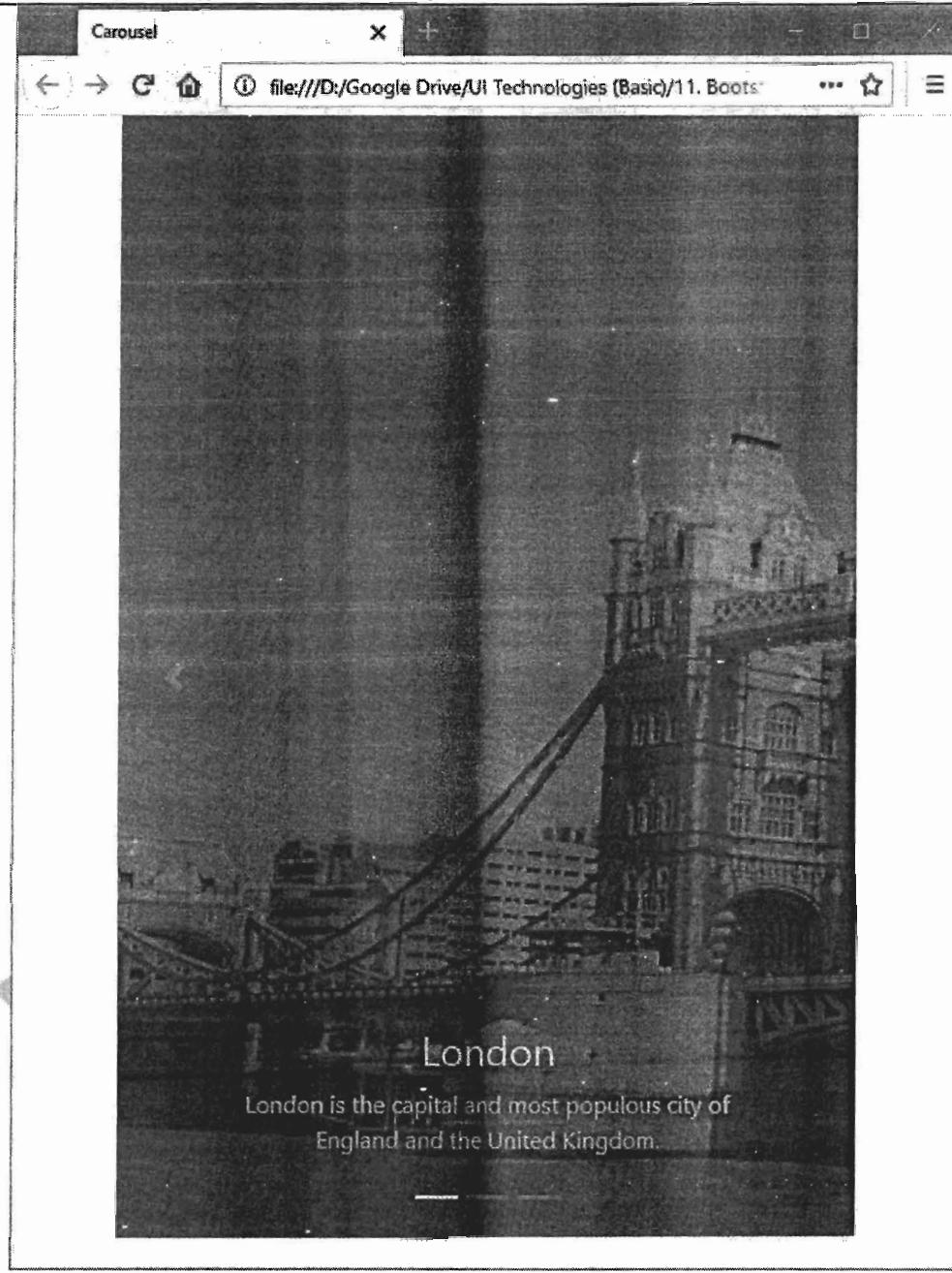
        <ul class="carousel-indicators">
          <li data-target="#demo" data-slide-to="0" class="active"></li>
          <li data-target="#demo" data-slide-to="1"></li>
          <li data-target="#demo" data-slide-to="2"></li>
        </ul>

        <div class="carousel-inner">
          <div class="carousel-item active">
            
            <div class="carousel-caption">
```

```
<h3>London</h3>
<p>London is the capital and most populous city of England and the United Kingdom.</p>
</div>
</div>
<div class="carousel-item">

<div class="carousel-caption">
<h3>New York</h3>
<p>The City of New York, often called New York City (NYC) or simply New York, is the most
populous city in the United States.[</p>
</div>
</div>
<div class="carousel-item">

<div class="carousel-caption">
<h3>Singapore</h3>
<p>Singapore, officially the Republic of Singapore, is a sovereign city-state and island country
in Southeast Asia.</p>
</div>
</div>
</div>
<a class="carousel-control-prev" href="#demo" data-slide="prev">
<span class="carousel-control-prev-icon"></span>
</a>
<a class="carousel-control-next" href="#demo" data-slide="next">
<span class="carousel-control-next-icon"></span>
</a>
</div>
</div>
</body>
</html>
```



## Modal

### Modal

- It is used to display modal popup box with desired content.

### List of Classes

- data-target="#id" : Creates connection between modal popup and button
- modal : Modal container
- data-toggle="modal" : Show Modal popup when button is clicked
- fade : Fade effect for modal popup
- modal-dialog : Dialog modal popup
- modal-lg : Large modal popup
- modal-sm : Small modal popup
- modal-dialog-centered : Shows dialog vertically centered
- modal-content : Container for content of modal
- modal-header : Header of modal
- modal-body : Body of modal
- modal-footer : Footer of modal
- close : Style for close button
- data-dismiss="modal" : Closes the modal popup when the user clicks

### Example on Modal

```
<html>
<head>
<title>Modal</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<body>
<div class="container-fluid">
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#myModal">
  Open modal
</button>

<div class="modal fade" id="myModal">
<div class="modal-dialog modal-lg modal-dialog-centered">
<div class="modal-content">

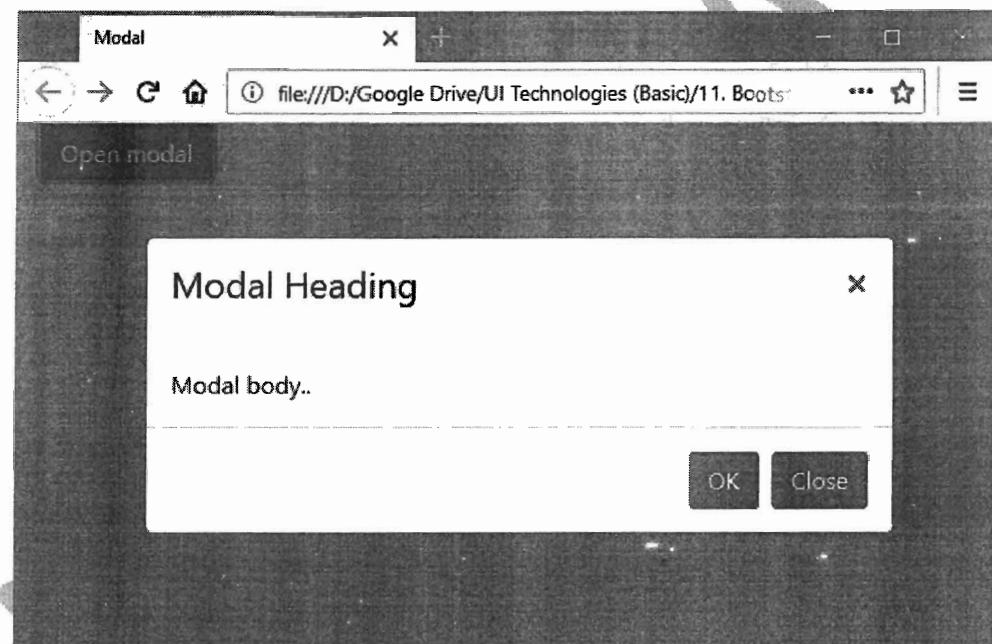
  <div class="modal-header">
    <h4>Modal Heading</h4>
    <button type="button" class="close" data-dismiss="modal">&times;</button>
  </div>

  <div class="modal-body">
    Modal body..
</div>
</div>
</div>
</div>
```

```
</div>

<div class="modal-footer">
  <button type="button" class="btn btn-primary" data-dismiss="modal">OK</button>
  <button type="button" class="btn btn-danger" data-dismiss="modal">Close</button>
</div>

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



# JAVASCRIPT 5, 6 & 7

## Fundamentals

### Introduction to JavaScript

- JavaScript is a programming language, which is used to create functionality in the web page. Functionality means, "receiving inputs from the user and providing output to the user".
- It can perform tasks such as calculations, decision making, repetitive tasks, dynamically displaying the output, reading inputs from the user dynamically, updating content on the web page based on the inputs, interacting with server, validations etc.
- It's operators and control statements are similar to "C" language.
- JavaScript is client-side (browser-side) language. That means it executes on the browser. It can also be used in server by using NodeJS.
- JavaScript is a case sensitive language.
- JavaScript is "interpreter-based" language. That means the code will be converted into machine language, line-by-line.
- JavaScript was developed by "Netscape Corporation" in 1996.
- JavaScript is the implementation of "EcmaScript". "EcmaScript" is the specification of "JavaScript".
- "EcmaScript" is designed by "Ecma International".

### Syntax of JavaScript

```
<script type="text/javascript">  
    Javascript code here  
</script>  
  
-- or --  
  
<script>  
    Javascript code here  
</script>
```

- **Note:** You can write the `<script>` tag either in `<head>` tag or `<body>` tag also; however, writing `<script>` tag at the end of `<body>` tag is a best practice.
- The `type="text/javascript"` attribute specifies that you are using javascript language; It is optional.

### console.log()

- The `console.log()` statement is used to display value in the browser console.
- To see console, first run the program in the browser and press "F12" or right click and choose "Inspect Element" – "Console" option in the browser.

**Syntax:** console.log(value);

### Steps to Prepare First Example in JavaScript

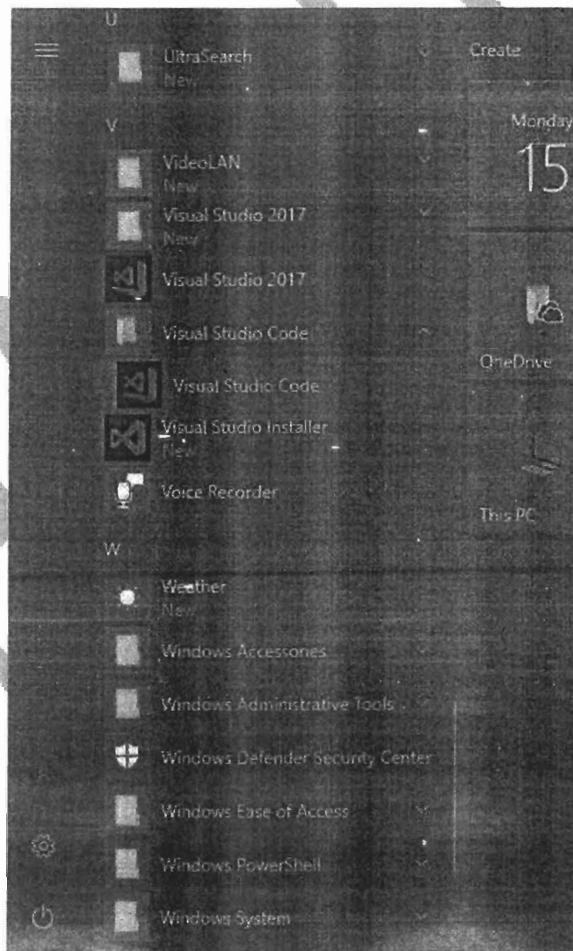
1. Installing Visual Studio Code
2. Creating JavaScript Program
3. Executing JavaScript Program

#### 1. Installing Visual Studio Code

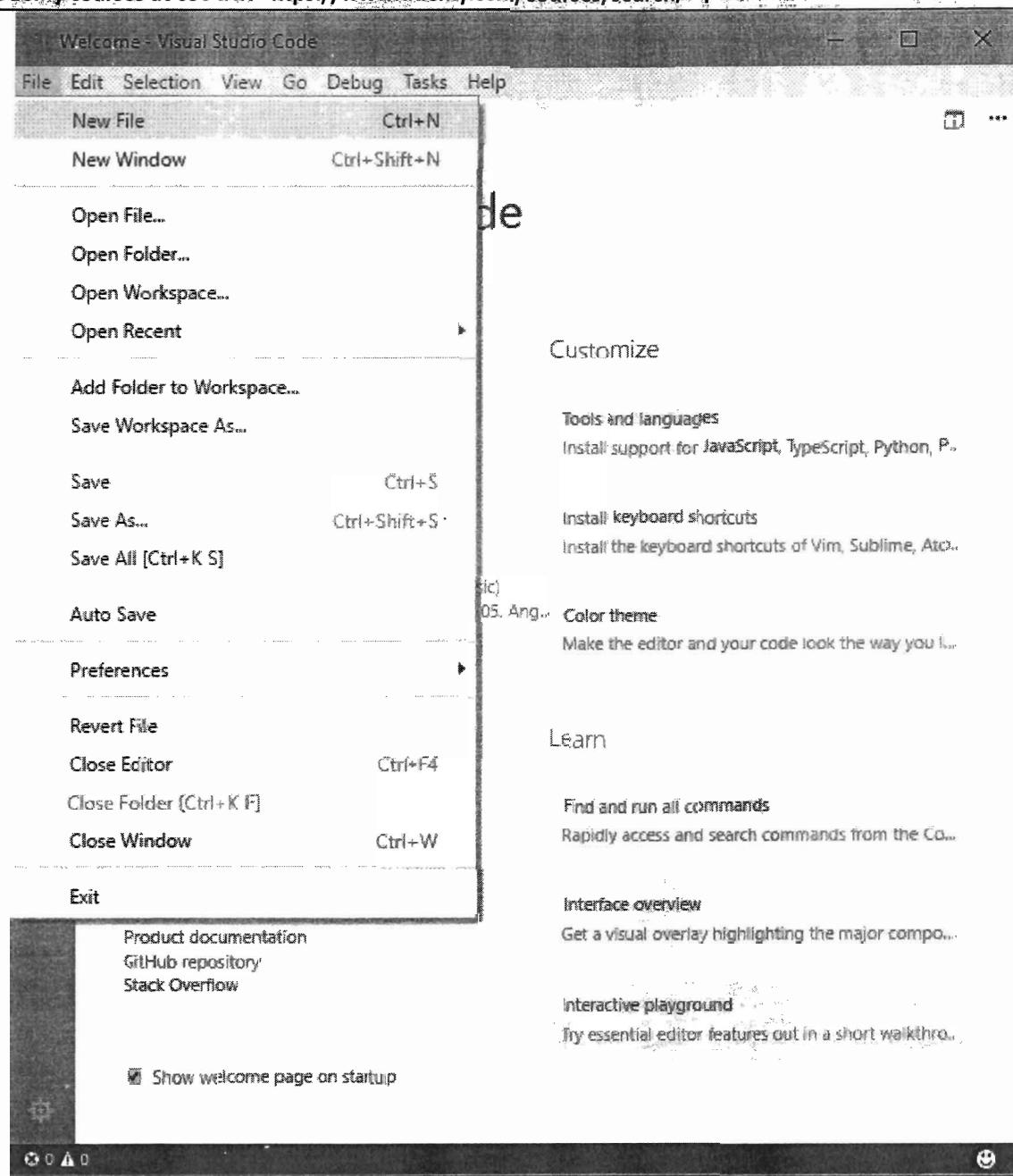
- [as shown in the previous chapters]

#### 2. Create JavaScript Program

- Open “Visual Studio Code”, by clicking on “Start” – “Visual Studio Code”.



- Visual Studio Code opened.



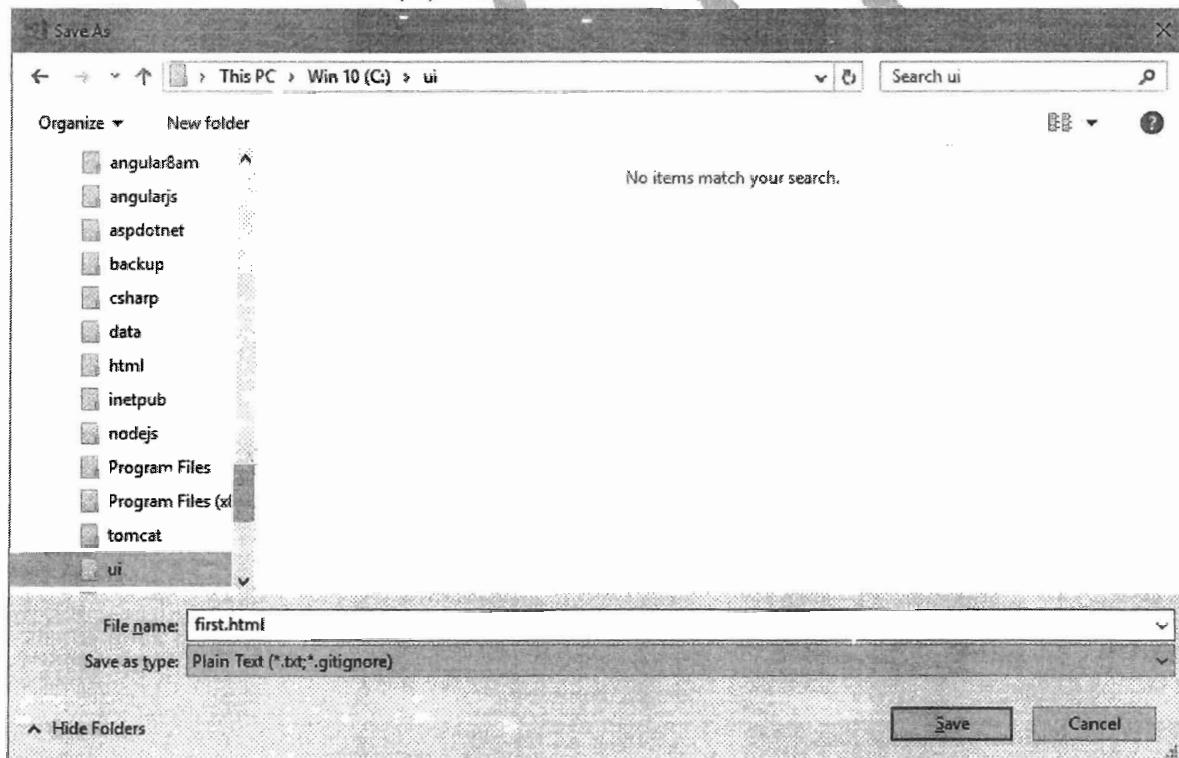
Go to "File" – "New File".

Type the program as follows:



```
<html>
  <head>
    <title>JavaScript</title>
  </head>
  <body>
    <h1>First Example</h1>
    <script>
      console.log("Hello World");
    </script>
  </body>
</html>
```

- Go to “File” menu – “Save” (or) Press Ctrl+S.

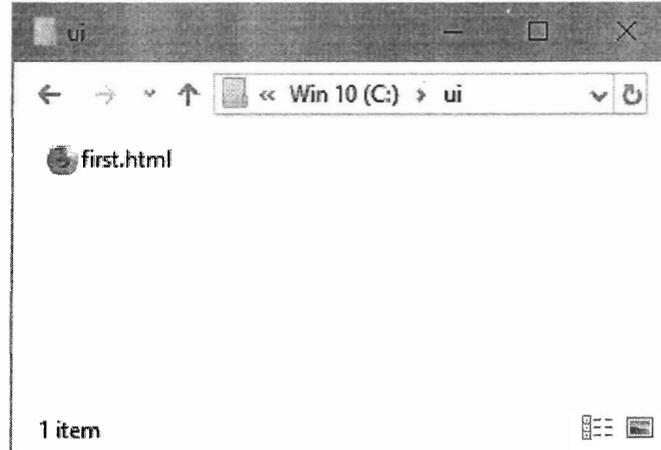


- Go to "c:\\" and click on "New folder". Enter the new folder name as "ui".
- Select "c:\ui" folder and enter the filename as "first.html".
- Click on "Save".

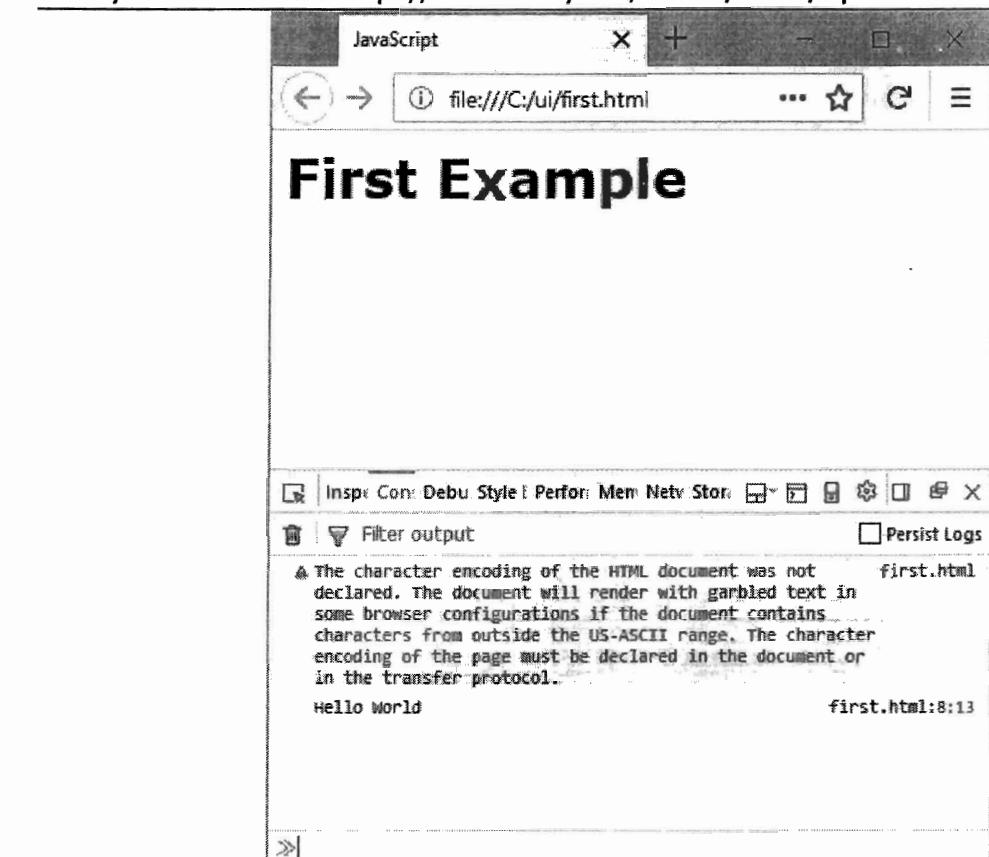
- Now the typescript file (c:\ui\first.html) is ready.

### 3. Execute the JavaScript Program

- Go to "Computer" or "This PC" and go to "c:\ui" folder.



- Double click on "first.html" (or) Right click on "first.html" and click on "Open With" – "Mozilla Firefox" / "Open With" – "Google Chrome".
- In the browser, right click in the web page and click on "Inspect Element" (or) press "F12" function key to open console.



**Output:** Hello World

## Variables

- Variable is a “named memory location” in RAM, to store a value temporarily, while executing the program.
- In JavaScript, the variables will be persisted (stored), while the web page is running in the browser.
- The value of variable can be changed any no. of times during the web page execution.
- The data type of the variable can be changed any no. of times during the web page execution, in JavaScript.

### Steps for development of variables

- **Declare (create) the variable - optional:** `var variablename;`
- **Set value into the variable:** `variablename = value;`
- **Get value from the variable:** `variablename`

### Example on Variables

```
<html>
  <head>
    <title>Variables</title>
  </head>
  <body>
    <h1>Variables</h1>
    <script>
      var a = 10;
      var b = "Hello";
      console.log(a);
      console.log(b);
    </script>
  </body>
</html>
```

## Operators

- Operator is a symbol, which represents an operation.
- JavaScript supports the following types of operators.
  1. Arithmetical Operators
  2. Assignment Operators
  3. Increment and Decrement Operators
  4. Relational Operators
  5. Logical Operators
  6. Concatenation Operator

### Arithmetical Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder

### Example on Arithmetical Operators

```
<html>
  <head>
    <title>Arithmetical operators</title>
  </head>
  <body>
    <h1>Arithmetical operators</h1>
    <script>
      var a = 10, b = 3;
      var c = a + b; //addition
      var d = a - b; //subtraction
    </script>
  </body>
</html>
```

```
var e = a * b; //multiplication
var f = a / b; //division
var g = a % b; //remainder
console.log(a); //10
console.log(b); //3
console.log(c); //13
console.log(d); //7
console.log(e); //30
console.log(f); //3.3333333333333335
console.log(g); //1
</script>
</body>
</html>
```

### Assignment Operators

- = Assigns to
- += Add and assigns to
- = Subtract and assigns to
- \*= Multiply and assigns to
- /= Divide and assigns to
- %= Remainder and assigns to

### Example on Assignment Operators

```
<html>
  <head>
    <title>Assignment operators</title>
  </head>
  <body>
    <h1>Assignment operators</h1>
    <script>
      var a;
      a = 100; //assignment operator
      console.log(a); //100
      var b;
      b = a; //assignment operator
      console.log(b); //100
      a += 10;
      console.log(a); //110
      a -= 10;
      console.log(a); //100
      a *= 10;
      console.log(a); //1000
      a /= 10;
      console.log(a); //100
      a %= 30;
      console.log(a); //10
    </script>
  </body>
```

```
</html>
```

### Increment and Decrement Operators

`++` Increment (`+1`)

`--` Decrement (`-1`)

### Example on Increment and Decrement Operators

```
<html>
  <head>
    <title>Increment and decrement operators</title>
  </head>
  <body>
    <h1>Increment and decrement operators</h1>
    <script>
      var a = 10;
      console.log(a); //10
      a++; //increment operator
      console.log(a); //11
      a--; //decrement operator
      console.log(a); //10
    </script>
  </body>
</html>
```

### Relational Operators

`==` Equal to

`!=` Not Equal to

`<` Less than

`>` Greater than

`<=` Less than or equal to

`>=` Greater than or equal to

### Example on Relational Operators

```
<html>
  <head>
    <title>Relational operators</title>
  </head>
  <body>
    <h1>Relational operators</h1>
    <script>
      var x = 100;
      var y = 200;
      var temp1, temp2, temp3, temp4, temp5, temp6;
      temp1 = (x == y);
      console.log(temp1); //false
      temp2 = (x != y);
    </script>
  </body>
</html>
```

```
        console.log(temp2); //true
        temp3 = (x < y);
        console.log(temp3); //true
        temp4 = (x <= y);
        console.log(temp4); //true
        temp5 = (x > y);
        console.log(temp5); //false
        temp6 = (x >= y);
        console.log(temp6); //false
    </script>
</body>
</html>
```

### Logical Operators

- && And (both conditions should be true)
- || Or (At least any one condition should be true)
- ! Not (given condition will be reverse)

### Example on Logical Operators

```
<html>
<head>
    <title>Logical operators</title>
</head>
<body>
    <h1>Logical operators</h1>
    <script>
        var x = 100;
        var y = 200;
        var z = 50;
        var temp1 = ((x < y) && (x > z));
        console.log(temp1); //true
        var temp2 = ((x < y) || (x < z));
        console.log(temp2); //true
        var temp3 = !(x < y);
        console.log(temp3); //false
    </script>
</body>
</html>
```

### Concatenation Operator

- + Attaches two strings and returns a single string.

Ex: "new" + "delhi" = "newdelhi"

Number + Number = addition

String + String = concatenation

String + Number = concatenation

Number + String = concatenation

### Example on Concatenation Operator

```
<html>
  <head>
    <title>Concatenation operator</title>
  </head>
  <body>
    <h1>Concatenation operator</h1>
    <script>
      var s1 = "peers";
      var s2 = "tech";
      var s3;
      s3 = s1 + s2; //string + string
      console.log(s3); //peerstech
    </script>
  </body>
</html>
```

## Control Statements

- Control statements are used to control (change) the program execution flow.
- These are used to make the execution flow jump forward / jump backward.
- JavaScript supports two types of control statements:
  - Conditional Control Statements: Used to jump forward.
  - Looping Control Statements: Used to jump backward.

### 1. Conditional Control Statements

- If
- Switch-case

### 2. Looping Control Statements

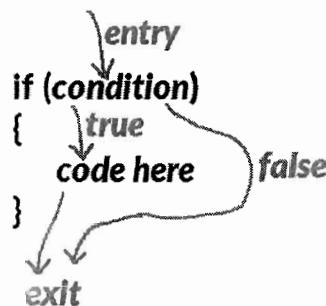
- While
- Do-while
- For

#### if

- “If” statement is used to check a condition, and execute the code only if the condition is TRUE.
- Types of “if”
  - If
  - If-else
  - Else-if

4. Nested if

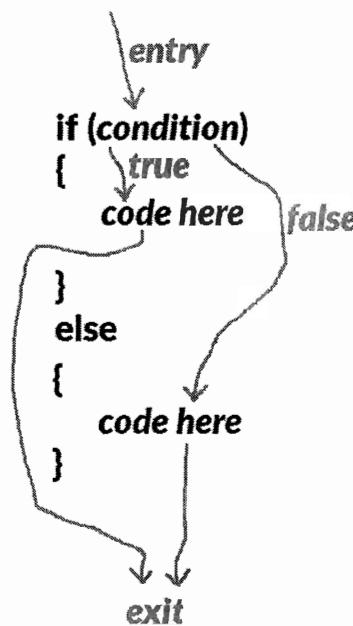
### Simple If



### Example of "Simple if"

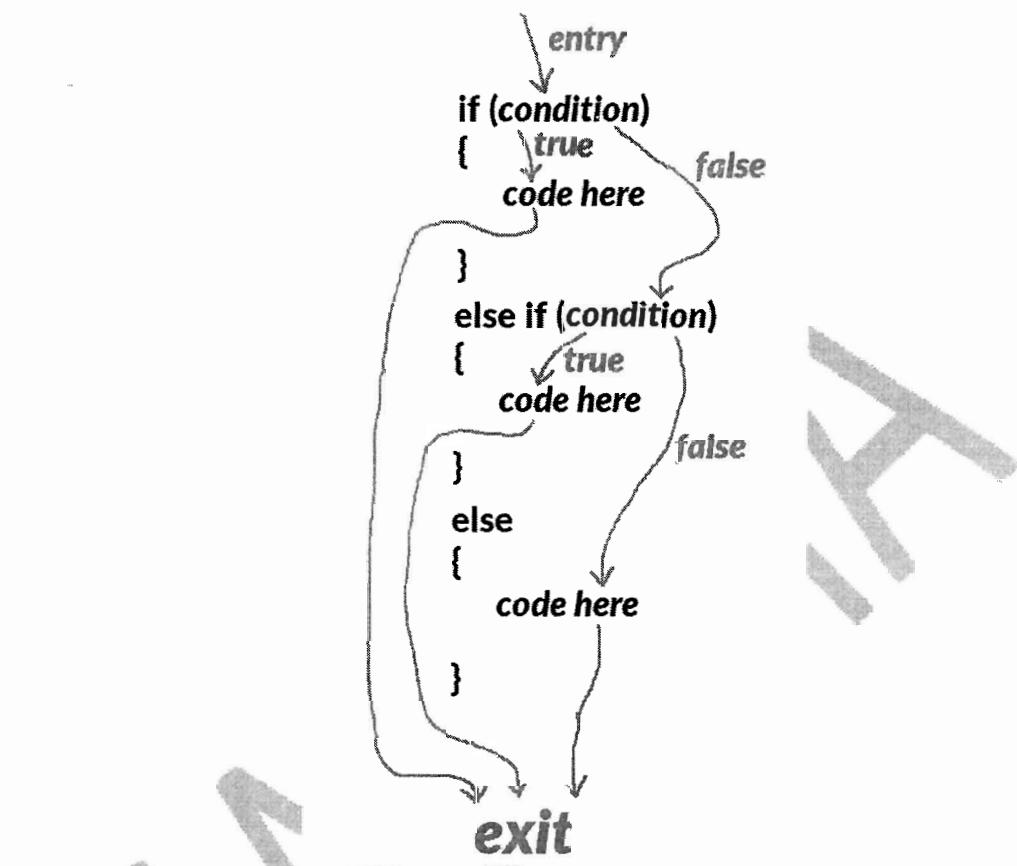
```
<html>
  <head>
    <title>If</title>
  </head>
  <body>
    <h1>If</h1>
    <script>
      var n = 10;
      if (n == 10)
      {
        console.log("n is equal to 10");
      }
      if (n != 10)
      {
        console.log("n is not equal to 10");
      }
    </script>
  </body>
</html>
```

## If Else



### Example of "if-else"

```
<html>
  <head>
    <title>If-else</title>
  </head>
  <body>
    <h1>If-else</h1>
    <script>
      var n = 10;
      if (n == 10)
      {
        console.log("n is equal to 10");
      }
      else
      {
        console.log("n is not equal to 10");
      }
    </script>
  </body>
</html>
```

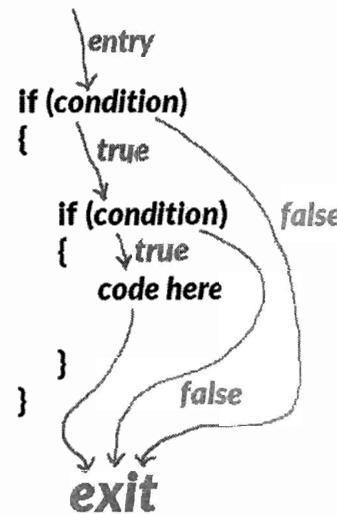


Example of "else-if"

```
<html>
  <head>
    <title>else-if</title>
  </head>
  <body>
    <h1>else-if</h1>
    <script>
      var a = 10, b = 20;
      if (a < b)
      {
        console.log("a is less than b");
      }
      else if (a > b)
      {
        console.log("a is greater than b");
      }
      else
      {
        console.log("a and b are equal");
      }
    </script>
  </body>
</html>
```

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

## Nested If



### Example of "nested if"

```
<html>
  <head>
    <title>Nested if</title>
  </head>
  <body>
    <h1>Nested if</h1>
    <script>
      var a = 10, b = 20;
      if (a != b)
      {
        if (a > b)
        {
          console.log("a is greater than b");
        }
        else
        {
          console.log("a is less than b");
        }
      }
      else
      {
        console.log("a and b are equal");
      }
    </script>
  </body>
</html>
```

### Switch-case

- It is used to check a variable's value, whether it matches with any one of the set of cases, and execute the code of the matched case.

#### Syntax:

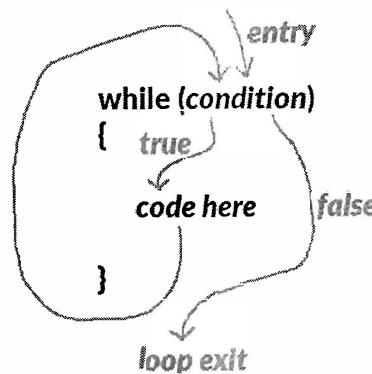
```
switch (variable )  
{  
    case value1: code here ; break;  
    case value2: code here ; break;  
    ...  
    default: code here ; break;  
}
```

### Example on Switch-case

```
<html>  
  <head>  
    <title>switch-case</title>  
  </head>  
  <body>  
    <h1>switch-case</h1>  
    <script>  
      var n = 7;  
      var monthname;  
      switch (n)  
      {  
        case 1: monthname = "Jan"; break;  
        case 2: monthname = "Feb"; break;  
        case 3: monthname = "Mar"; break;  
        case 4: monthname = "Apr"; break;  
        case 5: monthname = "May"; break;  
        case 6: monthname = "Jun"; break;  
        case 7: monthname = "Jul"; break;  
        case 8: monthname = "Aug"; break;  
        case 9: monthname = "Sep"; break;  
        case 10: monthname = "Oct"; break;  
        case 11: monthname = "Nov"; break;  
        case 12: monthname = "Dec"; break;  
        default monthname = "Unknown"; break;  
      }  
      console.log(monthname);  
    </script>  
  </body>  
</html>
```

### While

- “While” statement is used to execute the code repeatedly, while the condition is TRUE.

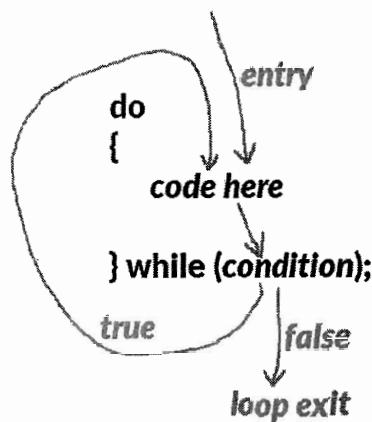


### Example on While

```
<html>
  <head>
    <title>While</title>
  </head>
  <body>
    <h1>While</h1>
    <script>
      var i = 1;
      while (i <= 10)
      {
        console.log(i);
        i++;
      }
    </script>
  </body>
</html>
```

### Do-While

- “Do-while” loop is mostly same as “while” loop.
- The difference is: “while” loop checks the condition for the first time also; but “do-while” loop doesn’t check the condition for the first time.

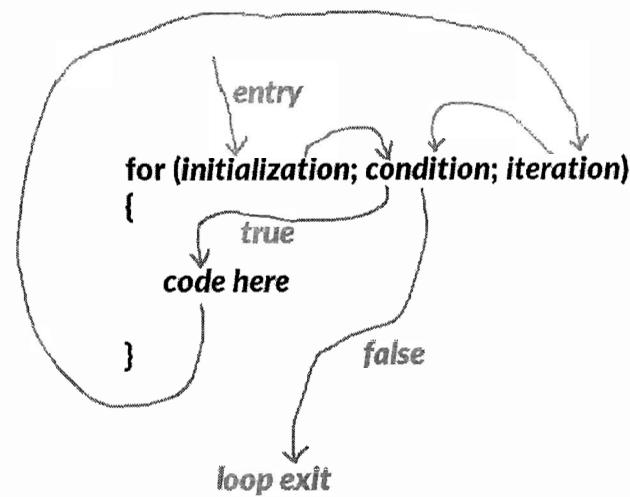


### Example on Do-While

```
<html>
  <head>
    <title>do-while</title>
  </head>
  <body>
    <h1>do-while</h1>
    <script>
      var i = 1;
      do
      {
        console.log(i);
        i++;
      } while (i <= 10);
    </script>
  </body>
</html>
```

### For

- “For” loop is mostly same as “while” loop.
- The difference is: “While” loop has the initialization, condition and iteration in three different places, so that it will be difficult to understand, if the code increases. But “for” loop has the initialization, condition and iteration in the same line, so that it will be easy to understand.



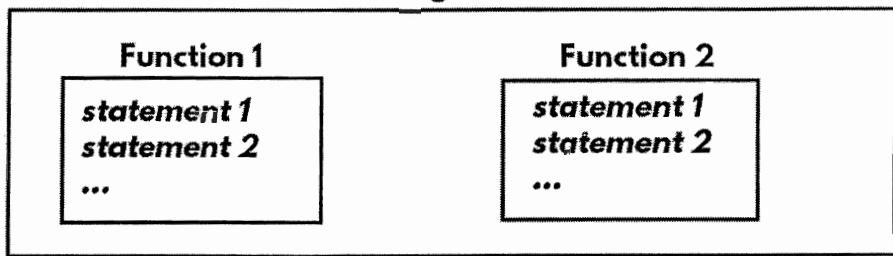
### Example on For

```
<html>
  <head>
    <title>for</title>
  </head>
  <body>
    <h1>for</h1>
    <script>
      var i;
      for (i = 1; i <= 10; i++)
      {
        console.log(i);
      }
    </script>
  </body>
</html>
```

## Functions

- Function is a re-usable “block” of the program, which is a set of statements with a name.
- The large program can be divided into many parts; each individual part is called as “function”.
- Functions are re-usable. That means functions can be called anywhere and any no. of times within the program. Every time when we call the function, the execution flow jumps to the function definition; executes the function and comes back to the calling portion.

## Program



### Steps for development of functions

- **Create the function:**

```
function functionname(parameter1, parameter2, ...)  
{  
    code here  
}
```

**Parameters:** The values that are passed from "calling portion" to the "function definition" are called as "arguments" and "parameters".

**Return:** The value that is passed from "function definition" to the "calling portion" is called as "return".

- **Call the function:**

```
functionName(value1, value2, ...);
```

- **Access the list of arguments**

arguments

**Note:** Every function has a property called "arguments", which represents the list of arguments that are passed to the function. arguments = { 0: argument1, 1: argument1, ... }

### Simple Example on Functions

```
<html>  
  <head>  
    <title>functions</title>  
  </head>  
  <body>  
    <h1>functions</h1>  
    <script>  
      function country()  
      {  
        console.log("India");
```

```
        }
        country();
        country();
        country();
    </script>
</body>
</html>
```

### Calling Function in Another Function - Example

```
<html>
<head>
    <title>functions</title>
</head>
<body>
    <h1> functions</h1>
    <script>
        function country()
        {
            console.log("India");
        }
        function city()
        {
            console.log("Hyderabad");
            country();
        }
        country();
        city();
    </script>
</body>
</html>
```

### Arguments and Return

```
<html>
<head>
    <title>functions</title>
</head>
<body>
    <h1>functions</h1>
    <script>
        function add(a, b)
        {
            var c; //local variable
            c = a + b;
            return (c);
        }
        var result;
        result = add(10, 20);
        console.log(result); //30
        var x = 100;
        var y = 250;
```

```
var result2 = add(x, y);
  console.log(result2); //350
</script>
</body>
</html>
```

### Arguments - Example

```
<html>
<head>
  <title>Arguments</title>
</head>
<body>
  <h1>Arguments</h1>
  <script>
    function fun1(a, b)
    {
      console.log(arguments);
    }
    fun1(10, 20);
    fun1(10, 20, 30);
  </script>
</body>
</html>
```

### Recursion

- Recursion is a technique of calling a function inside itself.
- Whenever a function calls itself, it is said to be "recursion".
- We should check the condition inside the function and call the same function, only if the condition is TRUE.

**Example:** Factorial of number =  $n * n-1 * n-2 * n-3 \dots * 0$

Factorial of 5 =  $5 * 4 * 3 * 2 * 1 = 120$

### Syntax:

```
function functionname()
{
  samefunctionname();
}
```

### Recursion - Example

```
<html>
<head>
  <title>Recursion</title>
</head>
<body>
  <h1>Recursion</h1>
  <script>
```

```
function factorial(n)
{
    if (n == 0)
    {
        return 1;
    }
    else
    {
        return n * factorial(n - 1);
    }
}
console.log(factorial(6));
</script>
</body>
</html>
```

## Arrays

- Array is a collection of multiple values.
- The no. of elements of the array is called as "array size" or "array length".
- Index (starts from 0) will be maintained for each element automatically.
- For example, you can store list of friends inside an array.
- In JavaScript, there is no rule of maintain "same type" of values in the array. JavaScript array can store the value of ANY data type. For example, you can store numbers, strings, objects in the same array.

array	
0	element 1
1	element 2
2	element 3
3	element 4

### Steps for development of arrays

- **Create an array:** var variablename = [ value1, value2, ... ];
- **Set value into the array element:** variablename[index] = value;
- **Get value from the array element:** variablename[index]
- **Get size of the array:** variablename.length
- **Add new element into the array:** variablename.push(newvalue);
- **Remove an existing element:** variablename.splice(index, no. of elements to remove);
- **Insert new element in the middle:** variablename.splice(index, 0, newvalue);

---

### Example on Arrays

---

```
<html>
  <head>
    <title>arrays</title>
  </head>
  <body>
    <h1>arrays</h1>
    <script>
      var n = [10, 20, 50, 150, 220];
      console.log(n.length);
      console.log(n[0]);
      console.log(n[1]);
      console.log(n[2]);
      console.log(n[3]);
      console.log(n[4]);
    </script>
  </body>
</html>
```

---

### Example on Push

---

```
<html>
  <head>
    <title>Push</title>
  </head>
  <body>
    <h1>Push</h1>
    <script>
      var n = [10, 20, 50, 150, 220];
      console.log(n);
      n.push(500);
      console.log(n);
    </script>
  </body>
</html>
```

---

### Example on Splice

---

```
<html>
  <head>
    <title>Splice</title>
  </head>
  <body>
    <h1>Splice</h1>
    <script>
      var n = [10, 20, 50, 150, 220];
      console.log(n);
      n.splice(3, 1);
      console.log(n);
    </script>
  </body>
</html>
```

### Example on Insert

```
<html>
  <head>
    <title>Splice</title>
  </head>
  <body>
    <h1>Splice</h1>
    <script>
      var n = [10, 20, 50, 150, 220];
      console.log(n);
      n.splice(3, 0, 800);
      console.log(n);
    </script>
  </body>
</html>
```

## Object Oriented Programming in JavaScript

### Introduction to Objects

- Object Oriented Programming (OOP) is a programming paradigm (programming style), which is based on the concept of "objects".
- Object represents a physical item / entity.
- Object is a collection of two types of members:
  1. Properties / Fields
  2. Methods
- **Properties / Fields:** Details about the object. Properties are the variables stored inside the object. Properties are used to store data about specific person, product or thing.
- **Methods:** Manipulations on the properties. Methods are the functions stored inside the object. Functions read values from properties and/or write values into properties.

#### Example:

"Car" object

- Properties
  - carModel: Honda City
  - carColor: Black
  - colorNo: 1234
- Methods
  - start()
  - changeGear()
  - stop()
- In the above example, the "Car" object has three properties called "carModel", "carColor", "colorNo", which have respective values. The

### Types of Object Oriented Programming (OOP) languages

- We have two types of OOP languages:
  1. Class-based Object Oriented Programming Language
  2. Prototype-based Object Oriented Programming Language

### Creating Objects

- We can create object in 2 ways:
  1. Object Literals
  2. Constructor Function

### Object Literals

- Object literals are represented as curly braces {}, which can include properties and methods.
- The property and value are separated with : symbol.
- **Syntax:** { "property": value, "method": function() { ... } }

#### Example 1 on Object Literals

```
<html>
  <head>
    <title>Object Literals</title>
  </head>
  <body>
    <h1>Object Literals</h1>
    <script>
      var stu = { studentid:1, studentname: "scott", marks: 80 };
      console.log(stu);
      console.log(stu.studentid);
      console.log(stu.studentname);
      console.log(stu.marks);
    </script>
  </body>
</html>
```

#### Example 2 on Object Literals

```
<html>
  <head>
    <title>Object Literals - Methods</title>
  </head>
  <body>
    <h1>Object Literals - Methods</h1>
    <script>
      var stu = {
        studentid: 1, studentname: "scott", marks: 80, "result": function ()
      {
        if (this.marks >= 35)
      {
```

```
        return "Pass";
    }
    else
    {
        return "Fail";
    }
}
);
console.log(stu);
console.log(stu.studentid);
console.log(stu.studentname);
console.log(stu.marks);
console.log(stu.result());
</script>
</body>
</html>
```

### Constructor Function

- Constructor function is a function that receives an empty (new) object, initializes properties and methods to the object.
- The "this" keyword inside the constructor function represents the current working object. For example, if it is called for the first time, the "this" keyword represents the first object; if it is called second time, the "this" keyword represents the second object.
- The constructor function can receive one or more parameters and initializes the same values into the respective properties.
- The "reference variable" stores the reference (address) of the object and used to access its members (properties and methods), outside the object literal / constructor function.

### Syntax of Constructor Function

```
function functionname(arguments)
{
    this.property = value;
    this.method = function() { ... };
}
var variablename = new functionname();
```

### Example on Constructor Function

```
<html>
<head>
    <title>Constructor Function</title>
</head>
<body>
    <h1>Constructor Function</h1>
```

```

<script>
    function Student(a, b, c)
    {
        this.studentid = a;
        this.studentname = b;
        this.marks = c;
        this.result = function ()
        {
            if (this.marks >= 35)
            {
                return "Pass";
            }
            else
            {
                return "Fail";
            }
        };
    }
    var stu = new Student(1, "Scott", 80);
    console.log(stu);
    console.log(stu.studentid);
    console.log(stu.studentname);
    console.log(stu.marks);
    console.log(stu.result());
</script>
</body>
</html>

```

### Object.Keys

- The “Object.keys” method is used to retrieve the list of properties of an object as an array.
- Sometimes, you will get data from server / browser storage. Then you don't know what properties / methods are present inside the object. Then you have to use Object.keys() are used to properties / methods of the object and also read its values programmatically.

**Syntax:** Object.keys(reference variable);

**Example:** Object.keys(stu);

- This is useful if you don't know what properties are exist in the object.

### Example on Object.Keys

```

<html>
    <head>
        <title>Keys</title>
    </head>
    <body>
        <h1>Keys</h1>
        <script>
            function Student(a, b, c)
            {

```

```
        this.studentid = a;
        this.studentname = b;
        this.marks = c;
        this.result = function ()
        {
            if (this.marks >= 35)
            {
                return "Pass";
            }
            else
            {
                return "Fail";
            }
        };
    }
    var stu = new Student(1, "Scott", 80);
    console.log(stu);
    var keys = Object.keys(stu);
    console.log(keys);
    for (var i = 0; i < keys.length; i++)
    {
        console.log(stu[keys[i]]);
    }
</script>
</body>
</html>
```

## JSON

- "JSON" stands for "JavaScript Object Notation".
- JSON is similar to "Object Literal", but having the following differences.
  - In "Object Literal", double quotes are optional for the properties; In "JSON", double quotes are must for properties.
  - In "Object Literal", methods are allowed; In "JSON", methods are not allowed.
- JSON is mainly used as data exchange format; it can be transferred from browser to server; and vice versa; and also it is can be stored in the local storage and session storage.

### Syntax:

```
{ "property" : value, "property" : value, ... }
```

## Stringify

- The "JSON.stringify" is used to convert "Object Literal" to "JSON" format. JSON is a text format which follows "JavaScript Object Literal" syntax. JSON is mainly used for store or exchange data between browser and server.

Syntax:      `JSON.stringify(reference variable)`

Example:    `JSON.stringify(stu)`

---

### Example on JSON.stringify

---

```
<html>
  <head>
    <title>Stringify</title>
  </head>
  <body>
    <h1>Stringify</h1>
    <script>
      function Student(a, b, c)
      {
        this.studentid = a;
        this.studentname = b;
        this.marks = c;
        this.result = function ()
        {
          if (this.marks >= 35)
          {
            return "Pass";
          }
          else
          {
            return "Fail";
          }
        };
      }
      var stu = new Student(1, "Scott", 80);
      var str = JSON.stringify(stu);
      console.log(str);
    </script>
  </body>
</html>
```

### Parse

- The “JSON.parse” is used to convert “JSON” to “Object Literal” format.

**Syntax:** JSON.parse(json data)

**Example:** JSON.parse(' { "a":10, "b":20 } ')

---

### Example on JSON.parse

---

```
<html>
  <head>
    <title>Parse</title>
  </head>
  <body>
    <h1>Parse</h1>
    <script>
      var s = ' { "studentid":1, "studentname": "scott", "marks": 80 } ';
      console.log(s);
      var stu = JSON.parse(s);
      console.log(stu);
    </script>
  </body>
</html>
```

```
        console.log(stu.studentid);
        console.log(stu.studentname);
        console.log(stu.marks);
    </script>
</body>
</html>
```

### Object Array Literal

- “Object Array Literal” is a collection of “object literals”, stored as an array.
- It is used to represent group of records, for example list of students.
- Each object inside the object array represents one single record; for example “one student”.

#### Syntax:

```
[  
    { property : value, property : value, ... },  
    { property : value, property : value, ... },  
    { property : value, property : value, ... },  
    ...  
)
```

### Example on Object Array Literal

```
<html>
<head>
    <title>Object Array Literal</title>
</head>
<body>
    <h1>Object Array Literal</h1>
    <script>
        var employees =
        [
            { "empid": 101, "empname": "Scott", "salary": 4000 },
            { "empid": 102, "empname": "Smith", "salary": 5690 },
            { "empid": 103, "empname": "Allen", "salary": 6723 },
            { "empid": 104, "empname": "John", "salary": 8729 }
        ];
        console.log(employees);

        for (var i = 0; i < employees.length; i++)
        {
            console.log(employees[i].empid);
            console.log(employees[i].empname);
            console.log(employees[i].salary);
        }
    </script>
</body>
</html>
```

### Object Array

- “Object Array Literal” is a collection of “objects created using constructor function”, stored as an array.

#### Syntax:

[

```
  new constructorfunction(argument1, argument2, ...),  
  new constructorfunction(argument1, argument2, ...),  
  new constructorfunction(argument1, argument2, ...),
```

```
  ...
```

]

### Example on Object Array

```
<html>  
  <head>  
    <title>Object Array</title>  
  </head>  
  <body>  
    <h1>Object Array</h1>  
    <script>  
      function Employee(a, b, c)  
      {  
        this.empid = a;  
        this.empname = b;  
        this.salary = c;  
      }  
  
      var employees =  
      [  
        new Employee(101, "Scott", 4000),  
        new Employee(102, "Smith", 5690),  
        new Employee(103, "Allen", 9500),  
        new Employee(104, "John", 7400)  
      ];  
      console.log(employees);  
      for (var i = 0; i < employees.length; i++)  
      {  
        console.log(employees[i]);  
      }  
    </script>  
  </body>  
</html>
```

### Prototype

- The "prototype" generally represents model of the object, which contains list of properties and methods of the object.

- Every Constructor Function has a property called "prototype".
- Any properties or methods added to "prototype", will be automatically added to every object that is created based on the same constructor function.

### Example on Prototype

```
<html>
  <head>
    <title>Prototype</title>
  </head>
  <body>
    <h1>Prototype</h1>
    <script>
      function Student(a, b)
      {
        this.studentid = a;
        this.studentname = b;
      }
      Student.prototype.marks = 70;
      Student.prototype.result = function ()
      {
        if (this.marks >= 35)
          return "Pass";
        else
          return "Fail";
      };
      var s = new Student(101, "scott");
      console.log(s);
      console.log(s.studentid);
      console.log(s.studentname);
      console.log(s.marks);
      console.log(s.result());
    </script>
  </body>
</html>
```

### Inheritance

- The process of creating an object based on another object is called as "inheritance".
- So all the properties and methods of the parent object is inherited into the child object.

#### Syntax:

```
function parentconstructorfunction(arguments)
{
  ...
}
```

```
function childconstructorfunction(arguments)
```

```
{  
    parentconstructorfunction.call(this, arguments);  
  
    ...  
}
```

### Example on Inheritance

```
<html>  
  <head>  
    <title>Inheritance</title>  
  </head>  
  <body>  
    <h1>Inheritance</h1>  
    <script>  
      function Person(a, b)  
      {  
        this.name = a;  
        this.email = b;  
      }  
      function Student(a, b, c)  
      {  
        Person.call(this, a, b);  
        this.marks = c;  
      }  
      var stu = new Student("scott", "scott@gmail.com", 70);  
      console.log(stu);  
    </script>  
  </body>  
</html>
```

### Data Types

- "Data type" specifies type of data that you want to store in the variable or property.
- JavaScript supports the following data types:
  1. number : Any numbers
  2. string : Collection of characters
  3. Boolean : true, false
  4. undefined : undefined
  5. object : {}, new
  6. function : function() {}

### Example on Data Types

```
<html>  
  <head>  
    <title>Data Types</title>  
  </head>
```

```
<body>
  <h1>Data Types</h1>
  <script>
    var a = 10;
    var b = 20.3248;
    var c = "hello";
    var d = 'hello'
    var e = true;
    var f = false;
    var g;
    var h = {};
    var i = new fun1();
    var j = function () {};
    console.log(a);
    console.log(b);
    console.log(c);
    console.log(d);
    console.log(e);
    console.log(f);
    console.log(g);
    console.log(h);
    console.log(i);
    console.log(j);

    function fun1()
    {
    }
  </script>
</body>
</html>
```

### String

- "String" is a collection of characters. The characters include with the following:
  1. Uppercase alphabets : A-Z
  2. Lowercase alphabets : a-z
  3. Digits : 0-9
  4. Symbols : \$ # @ & \* etc.
  5. Spaces
- JavaScript string literals should be in either single quotes or double quotes.  
Ex: 'hello123'  
"hello123"

### typeof

- The "typeof" keyword is used to get the data type of given value.

**Syntax:**      `typeof value`

Example:    `typeof 10`

### Example on `typeof`

```
<html>
  <head>
    <title>typeof</title>
  </head>
  <body>
    <h1>typeof</h1>
    <script>
      var a = 10;
      var b = 20.3248;
      var c = "hello";
      var d = 'hello'
      var e = true;
      var f = false;
      var g;
      var h = {};
      var i = new fun1();
      var j = function () {};
      console.log(typeof a);
      console.log(typeof b);
      console.log(typeof c);
      console.log(typeof d);
      console.log(typeof e);
      console.log(typeof f);
      console.log(typeof g);
      console.log(typeof h);
      console.log(typeof i);
      console.log(typeof j);

      function fun1()
      {
      }
    </script>
  </body>
</html>
```

### **undefined vs null**

- "undefined" represents "empty value", which is by default assigned to every uninitialized variables. The developer is not supposed to assign "undefined" manually.
- "null" represents "empty value", which can be assigned by the developer.
- The datatype of "undefined" is "undefined".
- The datatype of "null" is "object".

Syntax of undefined:    `undefined`

Syntax of null:    `null`

### Example on undefined vs null

```
<html>
  <head>
    <title>undefined vs null</title>
  </head>
  <body>
    <h1>undefined vs null</h1>
    <script>
      var a;
      var b = null;
      console.log(a);
      console.log(b);
      console.log(typeof a);
      console.log(typeof b)
    </script>
  </body>
</html>
```

#### == and ===

- == operator checks only value; === operator checks value and data type also.
- == operator internally first converts the right side value in the data type of left side and checks the value. === operator will not perform any automatic conversion and directly checks the value.
- Use == operator to check only value. Use === operator check value and data type

Syntax of ==      value1 == value2

Syntax of ===      value1 === value2

#### Example on == vs ===

```
<html>
  <head>
    <title>== and ===</title>
  </head>
  <body>
    <h1>== and ===</h1>
    <script>
      var a = 10;
      var b = '10';
      console.log(a == b); //true
      console.log(a === b); //false
    </script>
  </body>
</html>
```

#### String Function

- The "String()" is a pre-defined function, which is used to convert a number into string.

Syntax:      String(number value)

### Example on String Function

```
<html>
  <head>
    <title>String</title>
  </head>
  <body>
    <h1>String</h1>
    <script>
      var a = 10;
      var b = String(a);
      console.log(a);
      console.log(b);
      console.log(typeof a);
      console.log(typeof b);
    </script>
  </body>
</html>
```

### ToString Function

- The "toString()" is a pre-defined function, which is used to convert a number into string.
- The difference between String() and toString() function is:
- The String() function is a global function; The toString() function is available inside number data type.

Syntax:    numbervalue.toString()

### Example on ToString Function

```
<html>
  <head>
    <title>toString</title>
  </head>
  <body>
    <h1>toString</h1>
    <script>
      var a = 10;
      var b = a.toString();
      console.log(a);
      console.log(b);
      console.log(typeof a);
      console.log(typeof b);
    </script>
  </body>
</html>
```

### Number Function

- The "Number()" is a pre-defined function, which is used to convert string to number.
- It returns "NaN", if the string is alphanumerical / alphabetic. "NaN" stands for "Not A Number", which indicates the value is not a number. The datatype of "NaN" is "number".

- It returns "0", if the string is empty / space.

**Syntax:** Number(string value)

### Example on Number Function

```
<html>
  <head>
    <title>Number</title>
  </head>
  <body>
    <h1>Number</h1>
    <script>
      var a = "10";
      var b = Number(a);
      var c = "10ab";
      var d = Number(c);
      var e = "ab10";
      var f = Number(e);
      var g = "ab";
      var h = Number(g);
      var i = "";
      var j = Number(i);
      var k = " ";
      var l = Number(k);
      console.log(a); //10
      console.log(b); //10
      console.log(c); //10ab
      console.log(d); //NaN
      console.log(e); //ab10
      console.log(f); //NaN
      console.log(g); //ab
      console.log(h); //NaN
      console.log(i); //[]
      console.log(j); //0
      console.log(k); //space
      console.log(l); //10
      console.log(typeof a);
      console.log(typeof b);
      console.log(typeof c);
      console.log(typeof d);
      console.log(typeof e);
      console.log(typeof f);
      console.log(typeof g);
      console.log(typeof h);
      console.log(typeof i);
      console.log(typeof j);
      console.log(typeof k);
      console.log(typeof l);
    </script>
  </body>
</html>
```

### ParseInt Function

- The "parseInt()" is a pre-defined function, which is used to convert string to number.
- parseInt() doesn't supports decimal places.
- It returns number, if the string is alphanumerical that starts with number.
- It returns "NaN", if the string is alphanumerical that starts with alphabet.
- It returns "NaN", if the string is alphabetic.
- It returns "NaN", if the string is empty / space.

**Syntax:** parseInt(string value)

### Example on parseInt Function

```
<html>
  <head>
    <title>ParseInt</title>
  </head>
  <body>
    <h1>ParseInt</h1>
    <script>
      var a = "10.72";
      var b = parseInt(a);
      var c = "10ab";
      var d = parseInt(c);
      var e = "ab10";
      var f = parseInt(e);
      var g = "ab";
      var h = parseInt(g);
      var i = "";
      var j = parseInt(i);
      var k = " ";
      var l = parseInt(k);
      console.log(a); //10.72
      console.log(b); //10
      console.log(c); //10ab
      console.log(d); //10
      console.log(e); //ab10
      console.log(f); //NaN
      console.log(g); //ab
      console.log(h); //10
      console.log(i); //[]
      console.log(j); //NaN
      console.log(k); //[]
      console.log(l); //NaN
      console.log(typeof a);
      console.log(typeof b);
      console.log(typeof c);
      console.log(typeof d);
      console.log(typeof e);
      console.log(typeof f);
```

```
        console.log(typeof g);
        console.log(typeof h);
        console.log(typeof i);
        console.log(typeof j);
        console.log(typeof k);
        console.log(typeof l);
    </script>
</body>
</html>
```

### ParseFloat Function

- The "parseFloat()" is a pre-defined function, which is used to convert string to number.
- It is same as parseInt(); but it accepts decimal places.
- It returns number, if the string is alphanumerical that starts with number.
- It returns "NaN", if the string is alphanumerical that starts with alphabet.
- It returns "NaN", if the string is alphabetic.
- It returns "NaN", if the string is empty / space.

**Syntax:** parseFloat(string value)

### Example on parseFloat Function

```
<html>
<head>
    <title>ParseFloat</title>
</head>
<body>
    <h1>ParseFloat</h1>
    <script>
        var a = "10.72";
        var b = parseFloat(a);
        var c = "10ab";
        var d = parseFloat(c);
        var e = "ab10";
        var f = parseFloat(e);
        var g = "ab";
        var h = parseFloat(g);
        var i = "";
        var j = parseFloat(i);
        var k = " ";
        var l = parseFloat(k);
        console.log(a); //10.72
        console.log(b); //10.72
        console.log(c); //10ab
        console.log(d); //10
        console.log(e); //ab10
        console.log(f); //NaN
        console.log(g); //ab
        console.log(h); //10
    </script>
</body>
</html>
```

```
console.log(i); // [empty]
console.log(j); // Nan
console.log(k); // [space]
console.log(l); // Nan
console.log(typeof a);
console.log(typeof b);
console.log(typeof c);
console.log(typeof d);
console.log(typeof e);
console.log(typeof f);
console.log(typeof g);
console.log(typeof h);
console.log(typeof i);
console.log(typeof j);
console.log(typeof k);
console.log(typeof l);
</script>
</body>
</html>
```

### + Unary Operator

- It is same as "Number()" function, but it supports only numbers.

Syntax: +

### Example on + Unary Operator

```
<html>
  <head>
    <title>+</title>
  </head>
  <body>
    <h1>+</h1>
    <script>
      var a = "10.92";
      var b = +a;
      console.log(a);
      console.log(b);
      console.log(typeof a);
      console.log(typeof b);
    </script>
  </body>
</html>
```

### toFixed() function

- It converts the number into string and adds the specified no. of decimal places.
- It also rounds the number.

Syntax: numbertovalue.toFixed(no. of decimals)

### Example on toFixed()

```
<html>
  <head>
    <title>toFixed</title>
  </head>
  <body>
    <h1>toFixed</h1>
    <script>
      var a = 10.86231;
      var b = a.toFixed(0);
      var c = a.toFixed(1);
      var d = a.toFixed(2);
      var e = a.toFixed(3);
      var f = a.toFixed(7);
      console.log(a);
      console.log(b);
      console.log(c);
      console.log(d);
      console.log(e);
      console.log(f);
      console.log(typeof a);
      console.log(typeof b);
      console.log(typeof c);
      console.log(typeof d);
      console.log(typeof e);
      console.log(typeof f);
    </script>
  </body>
</html>
```

## String Functions

- String functions are used to perform manipulations on strings

Ex: Converting into uppercase.

### toUpperCase()

- This function converts the string value into uppercase and returns it.

Syntax: string.toUpperCase()

Example: "hello".toUpperCase() → "HELLO"

### Example toUpperCase()

```
<html>
  <head>
    <title>toUpperCase</title>
  </head>
  <body>
    <h1>toUpperCase</h1>
    <script>
      var s1 = "Hyderabad";
```

```
var s2 = s1.toUpperCase();
  console.log(s2);
</script>
</body>
</html>
```

### toLowerCase( )

- This function converts the string value into lowercase and returns it.

**Syntax:** string.toLowerCase()

**Example:** "HELLO".toLowerCase() → "hello"

### Example on toLowerCase()

```
<html>
  <head>
    <title>toLowerCase</title>
  </head>
  <body>
    <h1>toLowerCase</h1>
    <script>
      var s1 = "Hyderabad";
      var s2 = s1.toLowerCase();
      console.log(s2);
    </script>
  </body>
</html>
```

### length

- This property returns the no. of characters in the string.

**Syntax:** string.length

**Example:** "hello".length → 5

### Example on length

```
<html>
  <head>
    <title>length</title>
  </head>
  <body>
    <h1>length</h1>
    <script>
      var s1 = "Hyderabad";
      var n = s1.length;
      console.log(n);
    </script>
  </body>
</html>
```

### charAt( )

- This function returns the single character present at the specified index.
- Index starts from 0 (zero).

**Syntax:** string.charAt( )

**Example:** "hello".charAt(1) → e

### Example on charAt()

```
<html>
  <head>
    <title>charAt</title>
  </head>
  <body>
    <h1>charAt</h1>
    <script>
      var s1 = "Hyderabad";
      var ch = s1.charAt(5);
      console.log(ch);
    </script>
  </body>
</html>
```

### charCodeAt( )

- This function returns the ASCII value of the single character present at the specified index.

**Syntax:** string.charCodeAt( )

**Example:** "hello".charCodeAt(1) → 101

### Example on charCodeAt()

```
<html>
  <head>
    <title>charCodeAt</title>
  </head>
  <body>
    <h1>charCodeAt</h1>
    <script>
      var s1 = "Hyderabad";
      var n = s1.charCodeAt(5);
      console.log(n);
    </script>
  </body>
</html>
```

### substr( )

- This function returns a part of the string, starting from specified index, upto the specified length of the string.

**Syntax:** string.substr(index, length)

Example: "hyderabad".substr(2, 4) → dera

### Example on substr()

```
<html>
  <head>
    <title>Substr</title>
  </head>
  <body>
    <h1>Substr</h1>
    <script>
      var s1 = "Hyderabad";
      var n1 = 2;
      var n2 = 4;
      var s2 = s1.substr(n1, n2);
      console.log(s2); //dera
    </script>
  </body>
</html>
```

### indexOf()

- This function searches for the given sub string in the string, and returns the index of the first character in the given string if it is found; it returns -1, if the character is not found.
- In case if you specify the start index, searching starts from the specified startindex.

Syntax: string.indexOf("character", startindex)

Example: "hyderabad".indexOf("a") → 5

Example: "hyderabad".indexOf("era") → 3

Example: "hyderabad".indexOf("z") → -1

- This function always considers the first occurrence only.

### Example on indexOf()

```
<html>
  <head>
    <title>indexOf</title>
  </head>
  <body>
    <h1>indexOf</h1>
    <script>
      var s1 = "Hyderabad";
      var n = s1.indexOf("r");
      console.log(n);
    </script>
  </body>
</html>
```

### Example on indexOf() - second occurrence

```
<html>
```

```
<head>
  <title>indexOf - second occurrence</title>
</head>
<body>
  <h1>indexOf - second occurrence</h1>
  <script>
    var s1 = "Hyderabad";
    var n = s1.indexOf("a");
    var n2 = s1.indexOf("a", n + 1);
    console.log(n2);
  </script>
</body>
</html>
```

### Example indexOf() with -1

```
<html>
  <head>
    <title>indexOf with -1</title>
  </head>
  <body>
    <h1>indexOf - with -1</h1>
    <script>
      var s1 = "Hyderabad";
      var n = s1.indexOf("q");
      console.log(n);
    </script>
  </body>
</html>
```

### replace()

- It replaces a “word” with “another word”.

Syntax: string.replace("old word", "new word")

Example: "MS Office".replace("Office", "Windows") → MS Windows

### Example on replace()

```
<html>
  <head>
    <title>Replace</title>
  </head>
  <body>
    <h1>Replace</h1>
    <script>
      var s1 = "Hyderabad is one of the cities in India";
      var s2 = "Hyderabad";
      var s3 = "Chennai";
      var s4 = s1.replace(s2, s3);
      console.log(s4); //Chennai is one of the cities in India
    </script>
  </body>
```

```
</html>
```

### split()

- This function converts a string into an array of many small strings and returns the array, based on the separator character.

**Syntax:** string.split("separator character")

**Example:** "how are you".split("") → [ "how", "are", "you" ]

### Example on split()

```
<html>
  <head>
    <title>Split</title>
  </head>
  <body>
    <h1>Split</h1>
    <script>
      var s1 = "How are you";
      var a = s1.split(' ');
      for (var i = 0; i < a.length; i++)
      {
        console.log(a[i]);
      }
    </script>
  </body>
</html>
```

### trim()

- This function removes the unnecessary spaces at left side and right side of the string.

**Syntax:** string.trim()

**Example:** " abc def ".trim() → "abc def"

### Example on trim()

```
<html>
  <head>
    <title>Trim</title>
  </head>
  <body>
    <h1>Trim</h1>
    <script>
      var s1 = "      Hyderabad is one of the cities in India.      ";
      var s2 = s1.trim();
      console.log(s1);
      console.log(s2);
      var n1 = s1.length;
      var n2 = s2.length;
      console.log("Before trim: " + n1); //55
      console.log("After trim: " + n2); //40
    </script>
  </body>
</html>
```

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

### concat()

- This function attaches two strings and make them as a single string.

**Syntax:** string.concat("another string")

**Example:** "united".concat("states") → "unitedstates"

### Example on concat()

```
<html>
  <head>
    <title>Concat</title>
  </head>
  <body>
    <h1>Concat</h1>
    <script>
      var s1 = "hydera";
      var s2 = "bad";
      var s3 = s1.concat(s2);
      console.log(s3); //hyderabad
    </script>
  </body>
</html>
```

## Date Functions

- Date functions are used to manipulate date and time.

### new Date()

- This is used to get the current system date and time.

**Syntax:** new Date()

**Example:** new Date() → Thu Aug 10 2017 11:04:51 GMT+0530 (India Standard Time)

### Example on new Date()

```
<html>
  <head>
    <title>Date</title>
  </head>
  <body>
    <h1>Date</h1>
    <script>
      var d = new Date();
      console.log(d);
    </script>
  </body>
</html>
```

### toLocaleDateString()

- This function returns the date in the following format: M/d/yyyy

Syntax: new Date().toLocaleDateString()

Example: new Date().toLocaleDateString() → 8/10/2017

#### Example on toLocaleDateString()

```
<html>
  <head>
    <title>toLocaleDateString</title>
  </head>
  <body>
    <h1>toLocaleDateString</h1>
    <script>
      var d = new Date();
      var s = d.toLocaleDateString();
      console.log(s);
    </script>
  </body>
</html>
```

### toLocaleTimeString()

- This function returns the time in the following format: hh:mi:ss am/pm

Syntax: new Date().toLocaleTimeString()

Example: new Date().toLocaleTimeString() → 11:04:51 AM

#### Example on toLocaleTimeString()

```
<html>
  <head>
    <title>toLocaleTimeString</title>
  </head>
  <body>
    <h1>toLocaleTimeString</h1>
    <script>
      var d = new Date();
      var s = d.toLocaleTimeString();
      console.log(s);
    </script>
  </body>
</html>
```

### getTime()

- This function returns the no. of milli seconds since "1/1/1970 12:00:00 AM".

Syntax: new Date().getTime()

Example: new Date().getTime() → 1502343291481

### Example on getTime()

```
<html>
  <head>
    <title>getTime</title>
  </head>
  <body>
    <h1>getTime</h1>
    <script>
      var d = new Date();
      var n = d.getTime(); //milliseconds since 01.01.1970 12:00:00 AM
      console.log(n);
    </script>
  </body>
</html>
```

### getDay()

- This function returns the no. of the day of the week.

0 = Sunday  
1 = Monday  
2 = Tuesday  
3 = Wednesday  
4 = Thursday  
5 = Friday  
6 = Saturday

Syntax:

`new Date().getDay()`

Example:

`new Date().getDay()` → 4

### Example on getDay()

```
<html>
  <head>
    <title>getDay</title>
  </head>
  <body>
    <h1>getDay</h1>
    <script>
      var d = new Date();
      n = d.getDay();
      console.log(n);
    </script>
  </body>
</html>
```

### getDate()

- This function returns only the date.

Syntax: new Date().getDate()

Example: new Date().getDate() → 10

### Example on getDate()

```
<html>
  <head>
    <title>getDate</title>
  </head>
  <body>
    <script>
      var d = new Date();
      var s = d.getDate();
      console.log(s);
    </script>
  </body>
</html>
```

### getMonth( )

- This function returns only the month (0 to 11).

Syntax: new Date().getMonth()

Example: new Date().getMonth() → 7

### Example on getMonth()

```
<html>
  <head>
    <title>getMonth</title>
  </head>
  <body>
    <script>
      var d = new Date();
      var s = d.getMonth();
      console.log(s);
    </script>
  </body>
</html>
```

### getFullYear( )

- This function returns only the year.

Syntax: new Date().getFullYear()

Example: new Date().getFullYear() → 2017

### Example on getFullYear()

```
<html>
  <head>
    <title>getFullYear</title>
  </head>
```

```
<body>
  <h1>getFullYear</h1>
  <script>
    var d = new Date();
    var s = d.getFullYear();
    console.log(s);
  </script>
</body>
</html>
```

### getHours( )

- This function returns only the hours (in 24 hours format).

**Syntax:** new Date().getHours()

**Example:** new Date().getHours() → 11

### Example on getHours()

```
<html>
  <head>
    <title>getHours</title>
  </head>
  <body>
    <script>
      var d = new Date();
      var s = d.getHours();
      console.log(s);
    </script>
  </body>
</html>
```

### getMinutes( )

- This function returns only the minutes.

**Syntax:** new Date().getMinutes()

**Example:** new Date().getMinutes() → 4

### Example on getMinutes()

```
<html>
  <head>
    <title>getMinutes</title>
  </head>
  <body>
    <script>
      var d = new Date();
      var s = d.getMinutes();
      console.log(s);
    </script>
  </body>
</html>
```

### getSeconds()

- This function returns only the seconds.

**Syntax:** new Date().getSeconds()

**Example:** new Date().getSeconds() → 51

#### Example on getSeconds()

```
<html>
  <head>
    <title>getSeconds</title>
  </head>
  <body>
    <script>
      var d = new Date();
      var s = d.getSeconds();
      console.log(s);
    </script>
  </body>
</html>
```

### getMilliseconds()

- This function returns only the milli seconds.

**Syntax:** new Date().getMilliseconds()

**Example:** new Date().getMilliseconds() → 481

#### Example on getMilliseconds()

```
<html>
  <head>
    <title>getMilliSeconds</title>
  </head>
  <body>
    <script>
      var d = new Date();
      var s = d.getMilliseconds();
      console.log(s);
    </script>
  </body>
</html>
```

### Creating a Custom Date:

- We can create custom (user-defined) date using the following steps:
- **Create a date object and variable:** var variable = new Date();
- **Set year:** variable.setFullYear(year);
- **Set month:** variable.setMonth(month); **Note:** month is 0 to 11
- **Set date:** variable.setDate(date);

### Example on Custom Date

```
<html>
  <head>
    <title>Custom date</title>
  </head>
  <body>
    <h1>Custom Date</h1>
    <script>
      var d = new Date();
      d.setFullYear(2018);
      d.setMonth(11); //0 to 11
      d.setDate(31);
      console.log(d.toLocaleDateString());
    </script>
  </body>
</html>
```

## Advanced

### Clousers

- “Clousers” are used to create private variables that are accessible to only a set of methods.
- Create a function; Create private variables in the same function with "var" keyword; Return an object from this function; If you call the function, the private variables of this function are not accessible outside the function.

#### Syntax:

```
var functionname = function()
{
  var variablename = value;
  return
  {
    method: function()
    {
      code
    },
    method: function()
    {
      code
    }
  };
  var x = new functionname();
```

//Private variables are not accessible using "x".

### Examples on Closures

```
<html>
  <head>
    <title>Closures</title>
  </head>
  <body>
    <h1>Closures</h1>
    <script>
      var Sample = function ()
      {
        var x = 0;
        return {
          increment: function ()
          {
            x++;
          },
          decrement: function ()
          {
            x--;
          },
          getValue: function ()
          {
            return x;
          }
        };
      };
      var s = new Sample();
      console.log(s.getValue()); //Output: 0
      s.increment();
      s.increment();
      console.log(s.getValue()); //Output: 2
      s.decrement();
      console.log(s.getValue()); //Output: 1
    </script>
  </body>
</html>
```

### Hoisting

- JavaScript automatically lifts-up the variable declarations (that are created using "var" keyword) to top of the program or top of the function. This is called as "Hoisting".

**Note:** Variable declarations cum initializations are not lifted-up.

### Example on Hoisting

```
<html>
  <head>
    <title>Hoisting</title>
  </head>
```

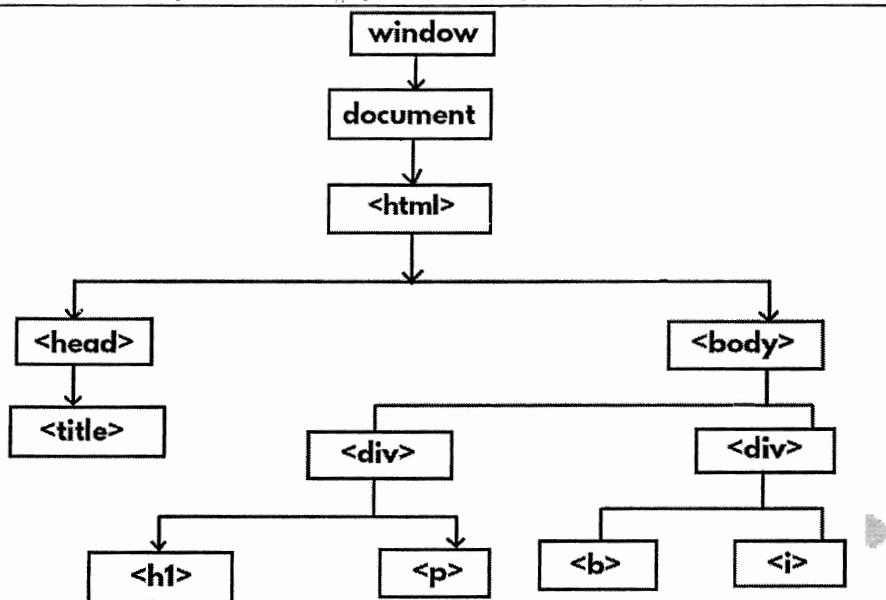
```
<body>
  <h1>Hoisting</h1>
  <script>
    x = 5;
    console.log(x);
    var x; //lifted-up
  </script>
</body>
</html>
```

## DOM

- DOM (Document Object Model) is the tree structure of html elements (tags) that are present within the web page.
- When the web page has been opened in the browser, DOM will be automatically created by the browser.
- The changes made to DOM are called as "DOM manipulations". DOM manipulations are performed using JavaScript.
- The entire browser window is called as "window". The webpage running on the browser is called as "document". It has only one main element called <html>. It has two children <head> and <body>. There are many children for both <head> and <body>.

### Example:

```
<html>
  <head>
    <title>DOM (Document Object Model)</title>
  </head>
  <body>
    <div>
      <h1>Heading</h1>
      <p>Paragraph</p>
    </div>
    <div>
      <b>bold</b>
      <i>italic</i>
    </div>
  </body>
</html>
```



### Objects inside DOM

- I. Window
- II. Document
- III. Element

#### I) Window

- "window" object represents the entire browser window.
- It has the following properties and methods:
  1. location.href
  2. navigator.userAgent
  3. screen
  4. screen
  5. alert()
  6. confirm()
  7. print()
  8. setTimeout()
  9. setInterval()
  10. scrollTo
  11. open()

### 1. location.href

- This property represents url of the current web page running in the browser window.
- **Syntax:** window.location.href

### 2. navigator.userAgent

- This property represents the name of current browser.
- **Syntax:** window.navigator.userAgent

### 3. navigator.screenX

- This property represents X value of current browser position on the screen.
- **Syntax:** window.navigator.screenX

### 4. navigator.screenY

- This property represents Y value of current browser position on the screen.
- **Syntax:** window.navigator.screenY

## Example on Window Properties

```
<html>
  <head>
    <title>window</title>
  </head>
  <body>
    <h1>window</h1>
    <script>
      console.log(window.location.href);
      console.log(window.navigator.userAgent);
      console.log(window.screenX);
      console.log(window.screenY);
    </script>
  </body>
</html>
```

### 5. alert()

- This method displays an information dialogbox (message dialogbox) to the user.
- It contains only OK button.

**Syntax:** window.alert("message")

**Example:** window.alert("Hello");

## Example on alert()

```
<html>
  <head>
    <title>alert</title>
  </head>
```

```
<body>
  <h1>alert</h1>
  <script>
    window.alert("Hello");
  </script>
</body>
</html>
```

## 6. confirm()

- This method displays a confirmation dialog box to the user.
- It contains OK and Cancel buttons.
- It returns "true", if the user clicks on "OK" button; It returns "false", if the user clicks on "Cancel" button.

**Syntax:** `window.confirm("message")`

### Example on confirm()

```
<html>
  <head>
    <title>confirm</title>
  </head>
  <body>
    <h1>confirm</h1>
    <script>
      var result = window.confirm("Are you sure to delete");
      console.log(result);
    </script>
  </body>
</html>
```

## 7. print()

- This method displays print dialog box, which is used to print the current webpage through selected printer.
- In Google Chrome, it shows "Print Preview" also.

**Syntax:** `window.print()`

### Example on print()

```
<html>
  <head>
    <title>Print</title>
  </head>
  <body>
    <h1>Print</h1>
    <p>Paragraph</p>
    <script>
      window.print();
    </script>
  </body>
</html>
```

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

### 8. setTimeout()

- This method calls the specified function, after completion of specified no. of milli seconds.
  - **Note:** 1000 milli seconds = 1 second
- Syntax:** window.setTimeout(function, milli seconds)

#### Example on setTimeout()

```
<html>
  <head>
    <title>setTimeout</title>
  </head>
  <body>
    <h1>setTimeout</h1>
    <script>
      window.setTimeout(fun1, 5000);
      function fun1()
      {
        console.log("Hello");
      }
    </script>
  </body>
</html>
```

### 9. setInterval()

- This method calls the specified function repeatedly, for every completion of specified no. of milli seconds.
  - **Note:** 1000 milli seconds = 1 second
- Syntax:** window.setInterval(function, milli seconds)

#### Example on setInterval()

```
<html>
  <head>
    <title>setInterval</title>
  </head>
  <body>
    <h1>setInterval</h1>
    <script>
      window.setInterval(fun1, 1000);
      function fun1()
      {
        console.log("Hello");
      }
    </script>
  </body>
</html>
```

## 10. scrollTo()

- This method scrolls the web page horizontally / vertically to the specified X and Y co-ordinates.
- The X and Y co-ordinates are calculated in the form of pixels.

Syntax: window.scrollTo(x, y)

### Example on scrollTo()

```
<html>
  <head>
    <title>scrollTo</title>
  </head>
  <body>
    <h1>scrollTo</h1>
    <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
    <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
    <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
    <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
    <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
    <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
    <script>
      window.scrollTo(0, 500);
    </script>
  </body>
</html>
```

## 11. open()

- This method opens a browser child window / popup window.
- It is mainly useful for opening ads.

Syntax: window.open("url", "logical name", "width=pixels, height=pixels");

### Example on open()

### open.html

```
<html>
  <head>
    <title>Open</title>
  </head>
  <body>
    <h1>Open</h1>
    <p>Click "Options" - "Allow Popups"</p>
    <script>
      window.open("mypage.html", "popup1", "width=300, height=300");
    </script>
  </body>
</html>
```

### mypage.html

```
<html>
  <head>
    <title>Popup</title>
  </head>
  <body>
    Hello, World
  </body>
</html>
```

## II) Document

- The "document" object represents the current working web page.
- It has properties and methods to manipulate web page.

### 1. title

- This property represents title of the web page.

**Syntax:** document.title

### 2. head

- This property represents <head> tag of the web page.

**Syntax:** document.head

### 3. body

- This property represents <body> of the web page.

**Syntax:** document.body

### 4. images

- This property represents all images of the web page, as an array.

**Syntax:** document.images

## 5. links

- This property represents all hyperlinks (<a> tags) of the web page, as an array.

**Syntax:** document.links

## 6. URL

- This property represents url of the web page.

**Syntax:** document.URL

### Example on Document Properties

```
<html>
  <head>
    <title>Hello</title>
  </head>
  <body>
    <h1>document</h1>
    
    
    <br>
    <a href="http://www.google.com">Google</a>
    <a href="http://www.facebook.com">Facebook</a>
    <script>
      console.log(document.title);
      console.log(document.head);
      console.log(document.body);
      console.log(document.images);
      console.log(document.links);
      console.log(document.URL);
    </script>
  </body>
</html>
```

### document.write()

- This method displays the given message in the web page.

**Syntax:** document.write("message")

### Example on document.write()

```
<html>
  <head>
    <title>document</title>
  </head>
  <body>
    <script>
      document.write("Hello World");
    </script>
  </body>
</html>
```

### document.getElementById()

- This method retrieves the single element that has specified ID.

**Syntax:** `document.getElementById("id")`

#### Example on document.getElementById()

```
<html>
  <head>
    <title>Getting element by ID</title>
  </head>
  <body>
    <p id="abc">Hello</p>
    <script>
      console.log(document.getElementById("abc"));
    </script>
  </body>
</html>
```

### document.getElementsByName()

- This method retrieves the array of elements that have specified name.

**Syntax:** `document.getElementsByName("name")`

#### Example on document.getElementsByName()

```
<html>
  <head>
    <title>getElementsByName</title>
  </head>
  <body>
    <h1>getElementsByName</h1>
    <p name="abc">Hello 1</p>
    <p name="abc">Hello 2</p>
    <p>Hai</p>
    <p name="abc">Hello 3</p>
    <script>
      console.log(document.getElementsByName("abc"));
    </script>
  </body>
</html>
```

### document.getElementsByTagName()

- This method retrieves the array of elements that have specified tag name.

**Syntax:** `document.getElementsByTagName("tag name")`

#### Example on document.getElementsByTagName()

```
<html>
  <head>
    <title>getElementsByTagName</title>
```

```
</head>
<body>
  <h1>getElementsByTagName</h1>
  <p>Hello 1</p>
  <p>Hello 2</p>
  <p>Hello 3</p>
  <p>Hello 4</p>
  <script>
    console.log(document.getElementsByTagName("p"));
  </script>
</body>
</html>
```

### **document.getElementsByClassName()**

- This method retrieves the array of elements that have specified class name.

**Syntax:** `document.getElementsByClassName("class name")`

### **Example on document.getElementsByClassName()**

```
<html>
  <head>
    <title>getElementsByClassName</title>
  </head>
  <body>
    <h1>getElementsByClassName</h1>
    <p class="abc">Hello 1</p>
    <p class="abc">Hello 2</p>
    <p>Hello</p>
    <p class="abc">Hello 2</p>
    <script>
      console.log(document.getElementsByClassName("abc"));
    </script>
  </body>
</html>
```

### **document.querySelectorAll()**

- This method retrieves the array of elements that are matching with specified selector.
- You can use any CSS selectors:

1. Tag Selector : tag
2. ID Selector : #id
3. Class Selector : .classname
4. Grouping Selector : tag1,tag2,...
5. Child Selector : parent child

**Syntax:** `document.querySelectorAll("selector")`

### Example on document.querySelectorAll()

```
<html>
  <head>
    <title>querySelectorAll</title>
  </head>
  <body>
    <p class="abc">Hello 1</p>
    <p class="abc">Hello 2</p>
    <p class="abc">Hello 3</p>
    <script>
      console.log(document.querySelectorAll(".abc"));
    </script>
  </body>
</html>
```

### document.querySelector()

- This method retrieves the first element that matches with specified selector.

**Syntax:** `document.querySelector("selector")`

### Example on document.querySelector()

```
<html>
  <head>
    <title>querySelector</title>
  </head>
  <body>
    <p id="abc">Hello</p>
    <script>
      console.log(document.querySelector("#abc"));
    </script>
  </body>
</html>
```

## III) Element

- The "element" object represents a single tag.
- It has properties and methods to manipulate the element.

### tagName

- This property represents name of the tag.
- **Syntax:** `document.getElementById("id").tagName`

### Example on tagName

```
<html>
  <head>
    <title>TagName</title>
  </head>
  <body>
```

```
<div id="div1">Hello</div>
<script>
  console.log(document.getElementById("div1").tagName);
</script>
</body>
</html>
```

### id

- This property represents id of the tag.

**Syntax:** `document.getElementById("id").id`

### Example on id

```
<html>
<head>
<title>ID</title>
</head>
<body>
<div id="div1">Hello</div>
<script>
  console.log(document.getElementById("div1").id);
</script>
</body>
</html>
```

### innerHTML

- This property represents content of the tag.

**Syntax:** `document.getElementById("id").innerHTML`

### Example on innerHTML

```
<html>
<head>
<title>innerHTML</title>
</head>
<body>
<div id="div1">Hello</div>
<script>
  console.log(document.getElementById("div1").innerHTML);
  document.getElementById("div1").innerHTML = "Hai";
</script>
</body>
</html>
```

### innerText

- This property represents content of the tag, without tags.

**Syntax:** `document.getElementById("id").innerText`

#### Example on innerText

```
<html>
  <head>
    <title>innerText</title>
  </head>
  <body>
    <div id="div1">Hello <b>World</b></div>
    <script>
      console.log(document.getElementById("div1").innerHTML);
      console.log(document.getElementById("div1").innerText);
    </script>
  </body>
</html>
```

#### style

- This property represents css style of the tag.

**Syntax:** `document.getElementById("id").style.property`

#### Example on style

```
<html>
  <head>
    <title>Style</title>
  </head>
  <body>
    <div id="div1">div 1</div>
    <script>
      document.getElementById("div1").style.fontFamily = "Comic Sans MS";
      document.getElementById("div1").style.fontSize = "50px";
      document.getElementById("div1").style.fontWeight = "bold";
      document.getElementById("div1").style.fontStyle = "italic";
      document.getElementById("div1").style.width = "500px";
      document.getElementById("div1").style.height = "200px";
      document.getElementById("div1").style.backgroundColor = "#006633";
      document.getElementById("div1").style.color = "#ccffff";
      document.getElementById("div1").style.border = "5px double red";
      document.getElementById("div1").style.padding = "20px";
      console.log(document.getElementById("div1").style.fontFamily);
    </script>
  </body>
</html>
```

#### parentElement

- This property represents parent element of the tag.
- **Syntax:** `document.getElementById("id").parentElement`

#### Example on parentElement

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

```
<title>ParentElement</title>
</head>
<body>
<div>
  <p id="p1">Hello</p>
</div>
<script>
  console.log(document.getElementById("p1").parentElement);
</script>
</body>
</html>
```

### children

- This property represents child elements of the tag.

**Syntax:** `document.getElementById("id").children`

### Example on children

```
<html>
<head>
  <title>Children</title>
</head>
<body>
  <div id="div1">
    <p id="p1">Para 1</p>
    <p id="p2">Para 2</p>
    <p id="p3">Para 3</p>
  </div>
  <script>
    console.log(document.getElementById("div1").children);
  </script>
</body>
</html>
```

### scrollTop

- This property moves vertical scrollbar, based on the specified no. of pixels.

**Syntax:** `document.getElementById("id").scrollTop`

### Example on scrollTop

```
<html>
<head>
  <title>scrollTop</title>
  <style>
    #div1
    {
      background-color: skyblue;
      width: 400px;
      height: 400px;
      overflow: auto;
    }
  </style>
</head>
<body>
</body>
</html>
```

```
        }
    </style>
</head>
<body>
    <h1>scrollTop</h1>
    <div id="div1">
        <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
        <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
        <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
        <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
        <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
        <p>Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.</p>
    </div>
    <script>
        document.getElementById("div1").scrollTop = 300;
    </script>
</body>
</html>
```

### setAttribute()

- This method sets an attribute to the tag.

**Syntax:** `document.getElementById("id").setAttribute("attribute name", "value")`

### Example on setAttribute()

```
<html>
    <head>
        <title>setAttribute</title>
    </head>
    <body>
        
        <script>
            document.getElementById("myimage").setAttribute("src", "img2.jpg");
        </script>
    </body>
</html>
```

**Note:** Place "img1.jpg" and "img2.jpg" in the current folder.

### getAttribute()

- This method gets the value of specified attribute of the tag.

- **Syntax:** `document.getElementById("id").getAttribute("attribute name")`

### Example on `getAttribute()`

```
<html>
  <head>
    <title>getAttribute</title>
  </head>
  <body>
    
    <script>
      var a = document.getElementById("myimage").getAttribute("src");
      console.log(a);
    </script>
  </body>
</html>
```

**Note:** Place "img1.jpg" in the current folder.

### `removeAttribute()`

- This method removes the specified attribute of the tag.

**Syntax:** `document.getElementById("id").removeAttribute("attribute name")`

### Example on `removeAttribute()`

```
<html>
  <head>
    <title>removeAttribute</title>
  </head>
  <body>
    
    <script>
      document.getElementById("myimage").removeAttribute("title");
    </script>
  </body>
</html>
```

**Note:** Place "img1.jpg" in the current folder.

### `attributes`

- This property retrieves all the attributes of the tag, along with values.

**Syntax:** `document.getElementById("id").attributes`

### Example on `attributes`

```
<html>
  <head>
    <title>Attributes</title>
  </head>
  <body>
```

```

<script>
  var a = document.getElementById("myimage").attributes;
  console.log(a);
  console.log(a[0]);
  console.log(a[1]);
  console.log(a[2]);
  console.log(a[0].name);
  console.log(a[1].name);
  console.log(a[2].name);
  console.log(a[0].value);
  console.log(a[1].value);
  console.log(a[2].value);
</script>
</body>
</html>
```

**Note:** Place "img1.jpg" in the current folder.

#### **hasAttribute()**

- This method checks whether the element has specified attribute or not; returns "true" if it contains; returns "false" if it doesn't contain.

**Syntax:** `document.getElementById("id").hasAttribute("attribute name");`

#### **Example on hasAttribute()**

```
<html>
  <head>
    <title>hasAttribute</title>
  </head>
  <body>
    
    <script>
      console.log(document.getElementById("myimage").hasAttribute("src"));
    </script>
  </body>
</html>
```

**Note:** Place 'img1.jpg' in the current folder.

#### **focus()**

- This method places the cursor inside the element.

**Syntax:** `document.getElementById("id").focus()`

#### **Example on focus()**

```
<html>
  <head>
    <title>Focus</title>
  </head>
```

```
<body>
  Firstname: <input type="text" id="txt1"><br>
  Lastname: <input type="text" id="txt2"><br>
  <script>
    document.getElementById("txt1").focus();
  </script>
</body>
</html>
```

### click()

- This method clicks the specified element (equal to manual mouse click).
- **Syntax:** `document.getElementById("id").click()`

### Example on click()

```
<html>
  <head>
    <title>Click</title>
  </head>
  <body>
    <form action="http://localhost:8080">
      Firstname: <input type="text" name="firstname" value="abc"><br>
      Lastname: <input type="text" name="lastname" value="xyz"><br>
      <input type="submit" value="Submit" id="button1">
    </form>
    <script>
      document.getElementById("button1").click();
    </script>
  </body>
</html>
```

### remove()

- This method removes the current element.

**Syntax:** `document.getElementById("id").remove()`

### Example on remove()

```
<html>
  <head>
    <title>Remove</title>
  </head>
  <body>
    <p>para 1</p>
    <p id="p2">para 2</p>
    <p>para 3</p>
    <script>
      document.getElementById("p2").remove();
    </script>
  </body>
</html>
```

### document.createElement()

- This method creates a new element for the specified tag.

**Syntax:** document.createElement("tag name")

### appendChild()

- This method adds new child element to the current element.

**Syntax:** document.appendChild(newelement)

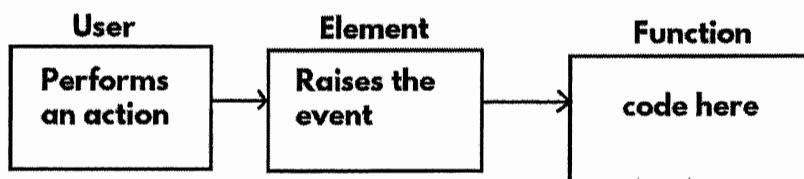
### Example on createElement() and appendChild()

```
<html>
  <head>
    <title>AppendChild</title>
  </head>
  <body>
    <div id="div1">
      <p>para 1</p>
      <p>para 2</p>
      <p>para 3</p>
    </div>
    <script>
      var mypara = document.createElement("p");
      mypara.innerHTML = "para " + n;
      mypara.setAttribute("id", "p" + n);
      document.getElementById("div1").appendChild(mypara);
    </script>
  </body>
</html>
```

### addEventListener()

#### Introduction to Events

- “Event” is a keyboard / mouse action, that is performed by the user, at run time.
- “Event Handling” is a concept of attaching the event with a function.
- Whenever the user performs an action, automatically the element raises the event; then we can call a function.



- **Syntax:** addEventListener("event name", functionname)

- **Example:** addEventListener("click", fun1)

### List of Events

1. click
2. dblclick
3. mouseover
4. mouseout
5. mousemove
6. keyup
7. keypress
8. focus
9. blur
10. change
11. contextmenu
12. cut
13. copy
14. paste

### “Click” event

- The “click” event executes when the user clicks on the element.

#### Syntax:

```
addEventListener("click", functionname);  
function functionname()  
{  
}
```

### Example on “Click” event

```
<html>  
  <head>  
    <title>click</title>  
    <style>  
      #div1  
      {  
        background-color: skyblue;  
        width: 200px;  
        height: 200px;  
      }  
    </style>  
  </head>  
  <body>  
    <h1>click</h1>  
    <div id="div1">click me</div>
```

```
<script>
  document.getElementById("div1").addEventListener("click", fun1);
  function fun1()
  {
    document.getElementById("div1").innerHTML = "thanx";
  }
</script>
</body>
</html>
```

### “Dblclick” event

- The “dblclick” event executes when the user double clicks on the element.

#### Syntax:

```
addEventListener("dblclick", functionname);
function functionname()
{
}
```

### Example on “Dblclick” event

```
<html>
<head>
  <title>dblclick</title>
  <style>
    #div1
    {
      background-color: skyblue;
      width: 200px;
      height: 200px;
    }
  </style>
</head>
<body>
  <h1>dblclick</h1>
  <div id="div1">double click me</div>
  <script>
    document.getElementById("div1").addEventListener("dblclick", fun1);
    function fun1()
    {
      document.getElementById("div1").innerHTML = "thanx";
    }
  </script>
</body>
</html>
```

## “Mouseover” and “Mouseout” events

### “Mouseover” event

- The “mouseover” event executes when the user moves the mouse pointer from outside to inside the element.

#### Syntax:

```
addEventListener("mouseover", functionname);  
function functionname()  
{  
}  
}
```

### “Mouseout” event

- The “mouseout” event executes when the user moves the mouse pointer from inside to outside the element.

#### Syntax:

```
addEventListener("mouseout", functionname);  
function functionname()  
{  
}  
}
```

### Example on “Mouseover” and “Mouseout” events

```
<html>  
  <head>  
    <title>mouseover, mouseout</title>  
    <style>  
      #div1  
      {  
        width: 400px;  
        height: 200px;  
        background-color: lightgreen;  
      }  
    </style>  
  </head>  
  <body>  
    <h1>mouseover, mouseout</h1>  
    <div id="div1">  
      mouseover me  
    </div>  
    <script>  
      document.getElementById("div1").addEventListener("mouseover", fun1);  
      function fun1()  
      {  
        document.getElementById("div1").innerHTML = "thanx";  
      }  
    </script>  
  </body>  
</html>
```

```
document.getElementById("div1").addEventListener("mouseout", fun2);
function fun2()
{
    document.getElementById("div1").innerHTML = "mouseover me";
}
</script>
</body>
</html>
```

### “Mousemove” event

- The “mousemove” event executes when the user moves the mouse pointer across the element.

#### Syntax:

```
addEventListener("mousemove", functionname);
function functionname()
{}
```

#### **event**

Represents the information, given by the browser. Whenever the user performs an action, the browser collects some information, and it passes the same information to the function automatically. This is called “event”.

**event.pageX:** Represents “X” co-ordinate of mouse pointer position.

**event.pageY:** Represents “Y” co-ordinate of mouse pointer position.

#### Browser

pageY

pageX

### Example on “Mousemove” event

```
<html>
<head>
<title>mousemove</title>
<style>
#div1
{
    width: 400px;
    height: 200px;
    background-color: lightgreen;
```

```
        }
    </style>
</head>
<body>
    <h1>mousemove</h1>
    <div id="div1">
        mouseover me
    </div>

    <script>
        document.getElementById("div1").addEventListener("mousemove", fun1);
        function fun1(event)
        {
            //event = browser given information
            var x = event.pageX;
            var y = event.pageY;
            console.log(x + ", " + y);
        }
    </script>
</body>
</html>
```

#### “Keyup” event

- The “keyup” event executes when the user presses any key on the keyboard, while the cursor is inside the element.

#### Syntax:

```
addEventListener("keyup", functionname);
function functionname()
{
}
```

#### Example on “Keyup” event

```
<html>
    <head>
        <title>Keyup</title>
    </head>
    <body>
        <h1>Keyup</h1>
        <input type="text" id="txt1">
        <script>
            document.getElementById("txt1").addEventListener("keyup", fun1);
            function fun1()
            {
                console.log(document.getElementById("txt1").value);
            }
        </script>
    </body>
```

### “Keypress” event

- The “keypress” event executes when the user presses any key on the keyboard, while the cursor is inside the element.
- “Keypress” event is very similar to “keyup” event.
- When you press any key on the keyboard, the following process happens:
  1. “Keypress” event executes.
  2. The character will be added in the textbox.
  3. “Keyup” event executes.
- “Keypress” event executes before accepting the currently pressed character into the element. “Keyup” event executes after accepting the currently pressed character into the element.
- In “keypress” event, we can accept / reject the currently pressed character, because it executes “before accepting” the character.
- In “keyup” event, we can’t reject the currently pressed character, because it executes “after accepting” the character.

#### Syntax:

```
addEventListener("keypress", functionname);  
function functionname()  
{  
}  
}
```

- **event.which:** Represents the ASCII value of currently pressed character. That means when the user presses a character, the browser automatically identifies its ASCII value and passes the same to the function, as “event.which”.
  - ASCII = American Standard Code for Information Interchange
  - As per ASCII, every character has a number.
    - 65 to 90 : A-Z
    - 97 to 122 : a-z
    - 48 to 57 : 0-9
    - 32 : Space
    - 8 : Backspace
    - 9 : TAB
    - 13 : Enter

- **event.preventDefault( )**: It stops the default functionality. That means it rejects the currently pressed character. If this method is not used, by default it accepts the currently pressed character.

### Example on "Keypress" event

```
<html>
  <head>
    <title>Keypress</title>
  </head>
  <body>
    <h1>Keypress</h1>
    <input type="text" id="txt1">
    <script>
      document.getElementById("txt1").addEventListener("keypress", fun1);
      function fun1()
      {
        console.log(document.getElementById("txt1").value);
      }
    </script>
  </body>
</html>
```

### Example on "Keypress" event - Alphabets only

```
<html>
  <head>
    <title>Alphabets only</title>
  </head>
  <body>
    <h1>Alphabets only</h1>
    <input type="text" id="txt1">
    <script>
      document.getElementById("txt1").addEventListener("keypress", fun1);
      function fun1(event)
      {
        var ch = event.which;
        console.log(ch);
        if (!(ch >= 65 && ch <= 90) || (ch >= 97 && ch <= 122) || (ch == 32) || (ch == 8) || (ch == 0))
        {
          event.preventDefault();
        }
      }
    </script>
  </body>
</html>
```

### Example on "Keypress" event - Numbers only

```
<html>
  <head>
    <title>Numbers only</title>
  </head>
  <body>
```

```
<h1>Numbers only</h1>
<input type="text" id="txt1">
<script>
  document.getElementById("txt1").addEventListener("keypress", fun1);
  function fun1(event)
  {
    var ch = event.which;
    console.log(ch);
    if (!(ch >= 48 && ch <= 57) || (ch == 8) || (ch == 0))
    {
      event.preventDefault();
    }
  }
</script>
</body>
</html>
```

### “Focus” and “Blur” events

#### “Focus” event

- The “focus” event executes when the cursor enters into the element.

#### Syntax:

```
addEventListener("focus", functionname);
function functionname()
{
}
```

#### “Blur” event

- The “blur” event executes when the cursor goes out of the element.

#### Syntax:

```
addEventListener("blur", functionname);
function functionname()
{
}
```

#### Example on “Focus” and “Blur” events

```
<html>
  <head>
    <title>focus and blur</title>
  </head>
  <body>
    <h1>focus and blur</h1>
    Email:
    <input type="text" id="txt1">
```

```
<span id="span1" style="color:gray; display:none">Use gmail only</span>
<script>
  document.getElementById("txt1").addEventListener("focus", fun1);
  function fun1()
  {
    document.getElementById("span1").style.display = "inline";
  }

  document.getElementById("txt1").addEventListener("blur", fun2);
  function fun2()
  {
    document.getElementById("span1").style.display = "none";
  }
</script>
</body>
</html>
```

### “Change” event

- The “change” event executes when the value of the element has been changed.
- That means it executes in following cases:
  1. When the user modifies the value of textbox and presses TAB key.
  2. When the user checks / unchecks the checkbox.
  3. When the user selects the radio button.
  4. When the user selects an item in the dropdownlist.

#### Syntax:

```
addEventListener("change", functionname);
function functionname()
{
}
```

### Example on “Change” event with TextBox

```
<html>
  <head>
    <title>Change - TextBox</title>
  </head>
  <body>
    <h1>Change - TextBox</h1>
    Source Text:
    <input type="text" id="txt1">
    <br>
    Destination Text:
    <input type="text" id="txt2">
    <script>
      document.getElementById("txt1").addEventListener("change", fun1);
      function fun1()
```

```
<script>
  document.getElementById("txt2").value = document.getElementById("txt1").value;
</script>
</body>
</html>
```

### Example on "Change" event with CheckBox

```
<html>
  <head>
    <title>Change - CheckBox</title>
  </head>
  <body>
    <h1>Change - CheckBox</h1>
    <input type="checkbox" id="chk1">
    <label for="chk1">I accept license agreement</label><br>
    <input type="submit" id="btn1" disabled="disabled">

    <script>
      document.getElementById("chk1").addEventListener("change", fun1);
      function fun1()
      {
        var b = document.getElementById("chk1").checked;
        if (b == true)
        {
          document.getElementById("btn1").disabled = "";
        }
        else
        {
          document.getElementById("btn1").disabled = "disabled";
        }
      }
    </script>
  </body>
</html>
```

### Example on "Change" event with RadioButton

```
<html>
  <head>
    <title>Change - RadioButton</title>
    <style>
      #div1
      {
        color: darkblue;
      }
    </style>
  </head>
  <body>
    <h1>Change - RadioButton</h1>
    <form>
```

```
<input type="radio" id="rb1" name="group1" checked="checked">
<label for="rb1">Small</label>
<input type="radio" id="rb2" name="group1">
<label for="rb2">Medium</label>
<input type="radio" id="rb3" name="group1">
<label for="rb3">Large</label>
<br>
<div id="div1" style="font-size:20px;">
    Hello, World
</div>
</form>

<script>
    document.getElementById("rb1").addEventListener("change", fun1);
    document.getElementById("rb2").addEventListener("change", fun1);
    document.getElementById("rb3").addEventListener("change", fun1);
    function fun1()
    {
        if (document.getElementById("rb1").checked == true)
            document.getElementById("div1").style.fontSize = "20px"; //small
        else if (document.getElementById("rb2").checked == true)
            document.getElementById("div1").style.fontSize = "35px"; //medium
        else if (document.getElementById("rb3").checked == true)
            document.getElementById("div1").style.fontSize = "50px"; //large
    }
</script>
</body>
</html>
```

#### Example on "Change" event with DropDownList

```
<html>
<head>
    <title>Change - DropDownList</title>
</head>
<body>
    <h1>Change - DropDownList</h1>
    <select id="drp1">
        <option>Choose Country</option>
        <option>India</option>
        <option>UK</option>
        <option>US</option>
    </select>
    <br><br>
    You selected: <span id="span1"></span>
    <script>
        document.getElementById("drp1").addEventListener("change", fun1);
        function fun1()
        {
            document.getElementById("span1").innerHTML = document.getElementById("drp1").value;
        }
    </script>
```

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

### “Contextmenu” event

- The “contextmenu” event executes when the user right clicks on an element.

#### Syntax:

```
addEventListener("contextmenu", functionname);

function functionname( )
{
}
```

**event.preventDefault():** It disables the right click menu (context menu).

### Example on “Contextmenu” event

```
<html>
  <head>
    <title>Disabling right click</title>
  </head>
  <body>
    <h1>Right click disabled</h1>
    <script>
      window.addEventListener("contextmenu", fun1);
      function fun1(event)
      {
        event.preventDefault();
        alert("right click not allowed");
      }
    </script>
  </body>
</html>
```

### “Cut”, “Copy”, “Paste” events

#### “Cut” event

- The “cut” event executes when the user selects “cut” option with keyboard / mouse.

#### Syntax:

```
addEventListener("cut", functionname);

function functionname( )
{
}
```

- **event.preventDefault( ):** It disables the cut operation.

### **"Copy" event**

- The “copy” event executes when the user selects “copy” option with keyboard / mouse.

#### **Syntax:**

```
addEventListener("copy", functionname);  
function functionname( )  
{  
}  
}
```

- **event.preventDefault( ):** It disables the copy operation.

### **"Paste" event**

- The “paste” event executes when the user selects “paste” option with keyboard / mouse.

#### **Syntax:**

```
addEventListener("paste", functionname);  
function functionname( )  
{  
}  
}
```

- **event.preventDefault( ):** It disables the paste operation.

### **Example on “Cut”, “Copy”, “Paste” events**

```
<html>  
  <head>  
    <title>Cut, copy, paste</title>  
  </head>  
  <body>  
    <h1>Cut, copy, paste disabled</h1>  
    <input type="text">  
    <input type="text">  
    <input type="text">  
    <script>  
      window.addEventListener("cut", fun1);  
      window.addEventListener("copy", fun1);  
      window.addEventListener("paste", fun1);  
      function fun1(event)  
      {  
        event.preventDefault();  
        alert("cut copy paste not allowed");  
      }  
    </script>  
  </body>  
</html>
```

### Login - Example

```
<html>
  <head>
    <title>Login</title>
  </head>
  <body>
    <h1>Login</h1>
    <form>
      Username: <input type="text" id="txt1"><br>
      Password: <input type="password" id="txt2"><br>
      <input type="submit" id="button1" value="Login"><br>
      <span id="span1"></span>
    </form>
    <script>
      document.getElementById("button1").addEventListener("click", fun1);
      function fun1(event)
      {
        event.preventDefault();
        var username = document.getElementById("txt1").value;
        var password = document.getElementById("txt2").value;
        if (username == "admin" && password == "manager")
        {
          document.getElementById("span1").innerHTML = "Successful login";
        }
        else
        {
          document.getElementById("span1").innerHTML = "Invalid login";
        }
      }
    </script>
  </body>
</html>
```

### Add, Subtract, Multiply, Divide Example

```
<html>
  <head>
    <title>Add Subtract Multiply Divide</title>
    <style>
      #txt3
      {
        background-color: lightgray;
      }

      #button1, #button2, #button3, #button4
      {
        background-color: #ffcc99;
        border: 1px ridge red;
      }
    </style>
  </head>
```

```
<body>
  <h1>Add Subtract Multiply Divide</h1>
  <form>
    Enter value for a:
    <input type="text" id="txt1"><br>
    Enter value for b:
    <input type="text" id="txt2"><br>
    <input type="button" id="button1" value="Add">
    <input type="button" id="button2" value="Subtract">
    <input type="button" id="button3" value="Multiply">
    <input type="button" id="button4" value="Divide"><br>
    Result:
    <input type="text" id="txt3" readonly="readonly">
  </form>

  <script>
    document.getElementById("button1").addEventListener("click", fun1);
    function fun1()
    {
      var a = parseInt(document.getElementById("txt1").value);
      var b = parseInt(document.getElementById("txt2").value);
      var c = a + b;
      document.getElementById("txt3").value = c;
    }

    document.getElementById("button2").addEventListener("click", fun2);
    function fun2()
    {
      var a = parseInt(document.getElementById("txt1").value);
      var b = parseInt(document.getElementById("txt2").value);
      var c = a - b;
      document.getElementById("txt3").value = c;
    }

    document.getElementById("button3").addEventListener("click", fun3);
    function fun3()
    {
      var a = parseInt(document.getElementById("txt1").value);
      var b = parseInt(document.getElementById("txt2").value);
      var c = a * b;
      document.getElementById("txt3").value = c;
    }

    document.getElementById("button4").addEventListener("click", fun4);
    function fun4()
    {
      var a = parseInt(document.getElementById("txt1").value);
      var b = parseInt(document.getElementById("txt2").value);
      var c = a / b;
      document.getElementById("txt3").value = c;
    }
  </script>
```

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

## Validations

- Validation is a process of checking the form input values, whether those are correct or not.
- If all the form input values are correct, then we will allow the form to be submitted to the server.
- If any one of the form input values are incorrect, then we will stop submitting the form and display appropriate error message to the user.
- Validations are done using JavaScript.

### Common Examples of Validations:

1. The value can't be blank.
2. The value should be in the proper format. Ex: phone number, email etc.
3. The value should be within the given range (minimum and maximum).

**Ex:** Amount should be in between 1000 and 10000 etc.

### Syntax for Validations

```
document.getElementById("form id").addEventListener("submit", functionname);
function functionname(event)
{
    if (condition)
    {
        //show error message
        event.preventDefault();
    }
    else
    {
        //hide error message
    }
}
```

### Example on Validations

```
<html>
<head>
    <title>Validations</title>
    <style>
        .error
        {
            color: red;
        }
    </style>
```

```
</head>
<body>
    <h1>Validations</h1>
    <form id="f1">
        Username:
        <input type="text" id="txtusername" value="Harsha" /> *
        <span id="spanusername" class="error"></span><br>
        Password:
        <input type="password" id="txtpassword" value="123456" /> *
        <span id="spanpassword" class="error"></span><br>
        <input type="submit" value="Login">
    </form>

    <script>
        document.getElementById("f1").addEventListener("submit", fun1);
        function fun1(event)
        {
            //event = browser given information
            var s1 = document.getElementById("txtusername").value;
            if (s1 == "")
            {
                event.preventDefault();
                document.getElementById("spanusername").innerHTML = "Username can't be blank";
            }
            else
            {
                document.getElementById("spanusername").innerHTML = "";
            }

            var s2 = document.getElementById("txtpassword").value;
            if (s2 == "")
            {
                event.preventDefault();
                document.getElementById("spanpassword").innerHTML = "Password can't be blank";
            }
            else
            {
                document.getElementById("spanpassword").innerHTML = "";
            }
        }
    </script>
</body>
</html>
```

### Regular Expressions

- Regular expression represents “pattern” of the value.
- Regular expressions are useful in validations, to check whether it is matching with the specified pattern or not.

**List of Important Regular Expressions:**

Sl. No	Description	Regular Expression
1	Digits only	^[0-9]*\$
2	Alphabets only	^[a-zA-Z]*\$
3	Indian Mobile Number	^[789]\d{9}\$
4	Email	\w+([.-]\w+)*@\w+([.-]\w+)*\.\w+([.-]\w+)*
5	Usernames: Alphabets, Digits and Hyphens only	([A-Za-z0-9-]+)
6	Passwords: 6 to 15 characters; at least one upper case letter, one lower case letter and one digit	((?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,15})

**Syntax to check Regular Expressions in JavaScript:**

/regular expression/.test(value)

**Example on Regular Expressions**

```
<html>
  <head>
    <title>Regular Expressions</title>
    <style>
      .error
      {
        color: red;
      }
    </style>
  </head>
  <body>
    <h1>Regular Expressions</h1>
    <form id="f1" action="http://localhost/serverpage.aspx">
      Email:
      <input type="text" id="txtemail">
      <span id="spanemail" class="error"></span><br>
      Mobile:
      <input type="text" id="txtmobile">
      <span id="spanmobile" class="error"></span><br>
      <input type="submit" value="Login">
    </form>

    <script>
      document.getElementById("f1").addEventListener("submit", fun1);
      function fun1(event)
      {
        //event = browser given information
        var s1 = document.getElementById("txtemail").value;
        var regexpemail = /\w+([.-]\w+)*@\w+([.-]\w+)*\.\w+([.-]\w+)*/;
        if (regexpemail.test(s1) == false)
        {
          event.preventDefault();
        }
      }
    </script>
  </body>
</html>
```

```
        document.getElementById("spanemail").innerHTML = "Invalid email";
    }
    else
    {
        document.getElementById("spanemail").innerHTML = "";
    }

    var s2 = document.getElementById("txtmobile").value;
    var regexpmobile = /^[789]\d{9}$/;
    if (regexpmobile.test(s2) == false)
    {
        event.preventDefault();
        document.getElementById("spanmobile").innerHTML = "Invalid mobile";
    }
    else
    {
        document.getElementById("spanmobile").innerHTML = "";
    }
}
</script>
</body>
</html>
```

## Types of JavaScript

- JavaScript program can be created in two ways:
  1. Internal JavaScript / Embedded JavaScript
  2. External JavaScript

### 1. Internal JavaScript

- Both “html code” and “javascript code” will be written in the same “.html” file.
- **Advantage:** Easy to understand.
- **Disadvantage:** The javascript code that is written in “.html” file can’t be used in another html file.
- All the previous programs are the examples of “Internal JavaScript”.

**filename.html**

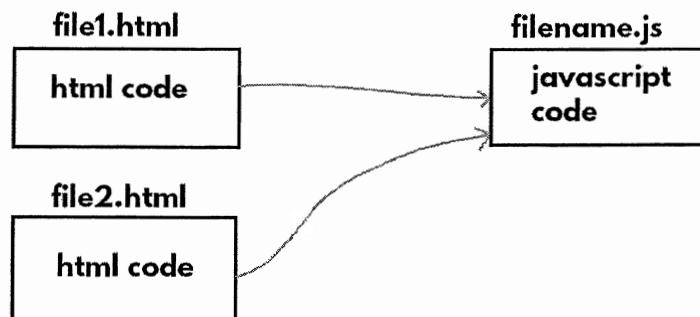
**html code**

**javascript code**

### 2. External JavaScript

- The javascript code will be written in “.js” file and will be called in one or more “.html” files.
- **Advantage:** We can call the same “.js” file from many html files (re-usability).

- In realtime, external JavaScript is recommended.



### Example on External JavaScript

#### JavaScript.js

```
document.write("<h1>Message from javascript</h1>");
```

#### Page1.html

```
<html>
<head>
  <title>External JavaScript - Page 1</title>
</head>
<body>
  <h1>External JavaScript - Page 1</h1>
  <script src="JavaScript.js"></script>
</body>
</html>
```

#### Page2.html

```
<html>
<head>
  <title>External JavaScript - Page 2</title>
</head>
<body>
  <h1>External JavaScript - Page 2</h1>
  <script src="JavaScript.js"></script>
</body>
</html>
```

## JavaScript - New Features

### Introduction to JavaScript 6

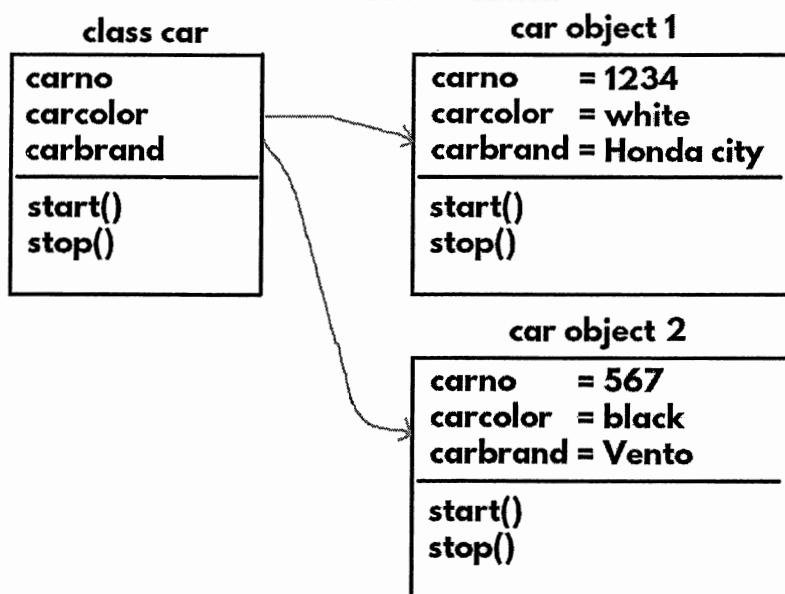
- "JavaScript 6" version released in 2015, which is also called as "EcmaScript 6 or ES 6".
- JavaScript 6 is the superset of JavaScript 5, which contains the following new features:
  1. Classes

2. Constructors
3. Inheritance
4. Method Overriding
5. Set and Get Methods
6. Default Arguments
7. Arrow Functions
8. Let
9. Const
10. Rest
11. Destructuring
12. Multiline Strings
13. String Interpolation
14. Reading Elements from Array

## Class-based OOP

### Classes

- "Object" is a "physical item", which is a collection of properties (details) and methods (manipulations).
- "Class" is a "model" of objects, which defines the list of properties and methods of the objects.



---

### Steps for working with Classes and Objects

---

#### Create a class

```
class classname  
{  
}
```

#### Create an object

```
new classname();
```

---

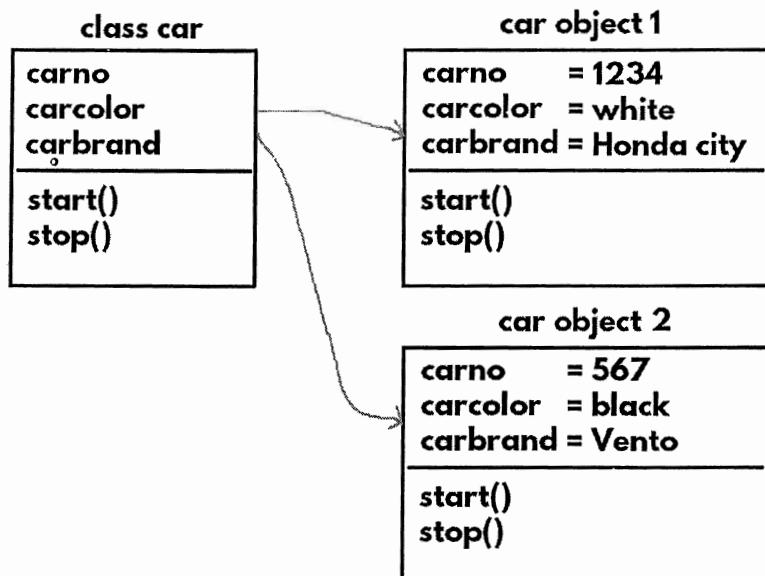
### Example on Classes and Objects

---

```
<html>  
  <head>  
    <title>Classes</title>  
  </head>  
  <body>  
    <h1>Classes</h1>  
    <script>  
      class Student  
      {  
      }  
      var s = new Student();  
      console.log(s);  
    </script>  
  </body>  
</html>
```

#### **Classes**

- "Object" is a "physical item", which is a collection of properties (details) and methods (manipulations).
- "Class" is a "model" of objects, which defines the list of properties and methods of the objects.



### Steps for working with Classes and Objects

#### Create a class

```
class classname  
{  
}
```

#### Create an object

```
new classname();
```

### Example on Classes and Objects

```
<html>  
  <head>  
    <title>Classes</title>  
  </head>  
  <body>  
    <h1>Classes</h1>  
    <script>  
      class Student  
      {  
      }  
      var s = new Student();  
      console.log(s);  
    </script>  
  </body>  
</html>
```

### Constructors

- Constructor is a special method in the class, that executes every time when we created an object for the class.
- Constructor's name should be "constructor".
- Constructor can receive arguments; but can't return any value.
- It is not possible to call the constructor without creating an object.
- Constructors are mainly used to create an initialize properties in the class.
- Parameterized Constructor is a constructor that receives one or more arguments (parameters) and initializes the same to the respective properties.

### Syntax of Constructor

```
class classname
{
    constructor(arguments)
    {
    }
}
```

### Example on Constructors

```
<html>
  <head>
    <title>Constructors</title>
  </head>
  <body>
    <h1>Constructors</h1>
    <script>
      class Student
      {
        constructor()
        {
          this.studentid = 101;
          this.studentname = "Scott";
          this.marks = 70;
        }
      }
      var s = new Student();
      console.log(s.studentid);
      console.log(s.studentname);
      console.log(s.marks);
    </script>
  </body>
</html>
```

---

### Example on Parameterized Constructors

---

```
<html>
  <head>
    <title>Parameterized Constructor</title>
  </head>
  <body>
    <h1>Parameterized Constructor</h1>
    <script>
      class Student
      {
        constructor(a, b, c)
        {
          this.studentid = a;
          this.studentname = b;
          this.marks = c;
        }
      }
      var s = new Student(201, "Smith", 80);
      console.log(s.studentid);
      console.log(s.studentname);
      console.log(s.marks);
      console.log(s.result());
    </script>
  </body>
</html>
```

### Methods

- Method is a function that manipulates data of the object.

### Syntax of Method

```
class classname
{
  methodname(arguments)
  {
  }
}
```

---

### Example on Methods

---

```
<html>
  <head>
    <title>Methods</title>
  </head>
  <body>
    <h1>Methods</h1>
```

```
<script>
  class Student
  {
    constructor(a, b, c)
    {
      this.studentid = a;
      this.studentname = b;
      this.marks = c;
    }
    result()
    {
      if (this.marks >= 35)
      {
        return "Pass";
      }
      else
      {
        return "Fail";
      }
    }
  }
  var s = new Student(101, "Scott", 70);
  console.log(s.studentid);
  console.log(s.studentname);
  console.log(s.marks);
  console.log(s.result());
</script>
</body>
</html>
```

### Inheritance

- Inheritance is a concept of extending the parent class, by creating the child class.
- When you create an object of child class, all the members of parent class will be automatically added to the child class's object.

### Syntax of Inheritance

```
class parentclassname
{
  constructor(arguments)
  {
  }
}

class childclassname extends parentclassname
{
```

```
constructor(arguments)
{
    super(arguments);
}
```

### Example on Inheritance

```
<html>
  <head>
    <title>Inheritance</title>
  </head>
  <body>
    <h1>Inheritance</h1>
    <script>
      class Person
      {
        constructor(name, age)
        {
          this.name = name;
          this.age = age;
        }
      }
      class Student extends Person
      {
        constructor(name, age, studentid, marks)
        {
          super(name, age);
          this.studentid = studentid;
          this.marks = marks;
        }
      }
      var s = new Student("scott", 20, 101, 78);
      console.log(s.name);
      console.log(s.age);
      console.log(s.studentid);
      console.log(s.marks);
    </script>
  </body>
</html>
```

### Method Overriding

- Method Overriding is a concept of extending the "parent class method", by creating similar method in the "child class", with same signature (name and arguments).

### Syntax of Method Overriding

```
class parentclassname
{
```

```
method(arguments)
{
}
}

class childclassname extends parentclassname
{
method(arguments)
{
    super.method(arguments);
}
}
```

### Example on Method Overriding

```
<html>
<head>
    <title>Method Overriding</title>
</head>
<body>
    <h1>Method Overriding</h1>
    <script>
        class Person
        {
            constructor(name, age)
            {
                this.name = name;
                this.age = age;
            }
            display()
            {
                return "name is " + this.name + ", age is " + this.age;
            }
        }
        class Student extends Person
        {
            constructor(name, age, studentid, marks)
            {
                super(name, age);
                this.studentid = studentid;
                this.marks = marks;
            }
            display()
            {
                return super.display() + ", studentid is " + this.studentid + ", marks is " + this.marks;
            }
        }
    </script>
</body>
</html>
```

```
class Employee extends Person
{
    constructor(name, age, employeeid, salary)
    {
        super(name, age);
        this.employeeid = employeeid;
        this.salary = salary;
    }
    display()
    {
        return super.display() + ", employeeid is " + this.employeeid + ", salary is " + this.salary;
    }
}
var s = new Student("scott", 20, 101, 78);
console.log(s.display());
var e = new Employee("smith", 25, 201, 9500);
console.log(e.display());
</script>
</body>
</html>
```

## Misc. Concepts

### Set and Get Methods

- The "set" method executes automatically when we assign a value into the accessor. It performs validation and assigns the value into the property, if it is valid.
- The "get" method executes automatically when we retrieve the value from the accessor. It returns the current value of the property.

### Steps for working with Set Method and Get Method

#### Create Set Method

```
set accessornname(argumentname)
{
    this.property = argumentname;
}
```

#### Create Get Method

```
get accessornname(argumentname)
{
    return this.property;
}
```

**Call Set Method**

accessorname = actualvalue;

**Create Get Method**

accessorname

**Example on Set and Get Methods**

```
<html>
  <head>
    <title>Set and Get Methods</title>
  </head>
  <body>
    <h1>Set and Get Methods</h1>
    <script>
      class Person
      {
        constructor()
        {
          this.name = null;
          this.age = 0;
        }
        set Name(value)
        {
          if (value.length <= 20)
          {
            this.name = value;
          }
        }
        get Name()
        {
          return this.name;
        }
        set Age(value)
        {
          if (value >= 18 && value <= 70)
          {
            this.age = value;
          }
        }
        get Age()
        {
          return this.age;
        }
      }
      var p = new Person();
      p.Name = "abcdefghijklmnopqrstuvwxyz";
      p.Age = 800;
      console.log(p.Name);
      console.log(p.Age);
      p.Name = "Scott";
    </script>
  </body>
</html>
```

```
p.Age = 30;  
console.log(p.Name);  
console.log(p.Age);  
</script>  
</body>  
</html>
```

### Default Arguments

- Default Arguments are used to provide a default value for the method of a parameter.
- When we call the method and don't supply a value for the parameter, the specified default value will be assigned to the parameter automatically.

### Steps for working with Default Arguments

```
method(argument = defaultvalue)  
{  
}
```

### Example on Default Arguments

```
<html>  
  <head>  
    <title>Default Arguments</title>  
  </head>  
  <body>  
    <h1>Default Arguments</h1>  
    <script>  
      class sample  
      {  
        method1(x = 100)  
        {  
          console.log(x);  
        }  
      }  
      var s = new sample();  
      s.method1(200);  
      s.method1();  
    </script>  
  </body>  
</html>
```

### Arrow Functions

- "Arrow Functions" are the functions created using " $=>$ " (arrow) operator.
- Arrow Function's "this" keyword reflects the current object; it will not be changed by the caller.

### Steps for working with Arrow Functions

```
this.method = (arguments) =>
{
}
```

### Example on Arrow Functions

```
<html>
  <head>
    <title>Arrow Functions</title>
  </head>
  <body>
    <h1>Arrow Functions</h1>
    <input type="button" value="Click me" id="button1">
    <script>
      class sample
      {
        constructor()
        {
          this.x = 100;
          this.method1 = () =>
          {
            console.log(this);
          }
        }
        var s = new sample();
        document.getElementById("button1").addEventListener("click", s.method1);
      </script>
    </body>
  </html>
```

### Let

- The "let" keyword can be used as alternative for "var" keyword, to create variables.
- The "var" keyword always creates "local variables" or "block level variables"; but "let" keyword creates "block level variables".

### Steps for working with Let

```
let variablename = value;
```

### Example on Let

```
<html>
  <head>
    <title>Let</title>
  </head>
  <body>
    <h1>Let</h1>
    <script>
```

```
for (var i = 1; i <= 10; i++)  
{  
}  
  console.log(i);  
  
  for (let j = 1; j <= 10; j++)  
{  
}  
  console.log(j);  
</script>  
</body>  
</html>
```

### Const

- The "const" keyword is used to create constants.
- We can't change the value of the constant.
- If you try to change the value of constant, it shows error.

### Steps for working with Const

```
const constantname = value;
```

### Example on Const

```
<html>  
  <head>  
    <title>Const</title>  
  </head>  
  <body>  
    <h1>Const</h1>  
    <script>  
      const n = 10;  
      console.log(n);  
      n = 20;  
      console.log(n);  
    </script>  
  </body>  
</html>
```

### Rest (...) Operator

- The rest (...) operator (should three dots) represents all remaining values.
- It can be used with methods / arrays.

### Steps for working with Rest Operator with Methods

```
method(...argumentname)
```

```
{  
}
```

---

### Steps for working with Rest Operator with Arrays

---

```
var arrayvariable = [...array1, ...array2];
```

### Example on Rest Operator With Methods

---

```
<html>
  <head>
    <title>Rest - Methods</title>
  </head>
  <body>
    <h1>Rest - Methods</h1>
    <script>
      class sample
      {
        method1(arg1, arg2, ...rest)
        {
          console.log(rest);
        }
      }
      var s = new sample();
      s.method1(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
    </script>
  </body>
</html>
```

### Example on Rest Operator With Arrays

---

```
<html>
  <head>
    <title>Rest - Arrays</title>
  </head>
  <body>
    <h1>Rest - Arrays</h1>
    <script>
      var south = ["Hyderabad", "Chennai", "Bangalore"]
      var north = ["Pune", "Mumbai", "New Delhi"];
      var cities = [...south, ...north];
      console.log(cities);
    </script>
  </body>
</html>
```

### Destructuring

- Destructuring is used to retrieve each value of an array into respective variables.

---

### Steps for working with Destructuring

---

```
var [ variable1, variable2, ... ] = arrayname;
```

### Example 1 on Destructuring

---

```
<html>
```

```
<head>
  <title>Destructuring</title>
</head>
<body>
  <h1>Destructuring</h1>
  <script>
    var cities = ["Hyderabad", "Chennai", "Bangalore"];
    var [city1, city2, city3] = cities;
    console.log(city1);
    console.log(city2);
    console.log(city3);
  </script>
</body>
</html>
```

### Example 2 on Destructuring

```
<html>
  <head>
    <title>Destructuring - Skip</title>
  </head>
  <body>
    <h1>Destructuring - Skip</h1>
    <script>
      var cities = ["Hyderabad", "Chennai", "Bangalore", "Pune", "Mumbai"];
      var [city1, city2, , city4] = cities;
      console.log(city1);
      console.log(city2);
      console.log(city4);
    </script>
  </body>
</html>
```

### Example 3 on Destructuring

```
<html>
  <head>
    <title>Destructuring - Rest</title>
  </head>
  <body>
    <h1>Destructuring - Rest</h1>
    <script>
      var cities = ["Hyderabad", "Chennai", "Bangalore", "Pune", "Mumbai"];
      var [city1, city2, ...rest] = cities;
      console.log(city1);
      console.log(city2);
      console.log(rest);
    </script>
  </body>
</html>
```

### Multiline Strings

- This concept is used to create a string with multiple lines of text.
- The multiline string should be enclosed within backticks ( `` ).

### Steps for working with Multiline Strings

line 1

line 2

line 3

...

### Example on Multiline Strings

```
<html>
  <head>
    <title>Multiline Strings</title>
  </head>
  <body>
    <h1>Multiline Strings</h1>
    <script>
      var s = `hello
      how
      are
      you`;
      console.log(s);
    </script>
  </body>
</html>
```

### String Interpolation

- "String Interpolation" replaces the expressions in the string with actual values of the respective variables.
- These strings must be enclosed within backticks ( `` ), instead of single quotes ( ' ) or double quotes ( " ).

### Steps for working with String Interpolation

`` text here \${variable} text here ``

### Example on String Interpolation

```
<html>
  <head>
    <title>String Interpolation</title>
  </head>
```

```
<body>
  <h1>String Interpolation</h1>
  <script>
    var name = "Scott";
    var result = `Hello, my name is ${name}`;
    console.log(result);
  </script>
</body>
</html>
```

### Reading Elements from Array

- The most common task of the developer is reading data from an array and displaying the same.
- This can be done in several ways:
  1. For Loop
  2. ForEach Loop
  3. ForOf Loop
  4. ForIn Loop

#### Steps for working with For Loop

```
for (i = 0; i < array.length; i++)
{
  array[i]
}
```

- For loop is index based.
- Forward / backward iterations are possible.
- We can start the loop from the middle also.

#### Steps for working with ForEach Loop

```
array.forEach( function(variable)
{
  variable
});
```

- For loop is value based. The variable contains "value" directly; but not index.
- Forward iterations are only possible. Backward iterations are not possible.
- We can't start the loop from the middle.

#### Steps for working with ForOf Loop

```
for (variable of array)
```

```
{  
  variable  
}
```

- For loop is value based. The variable contains "value" directly; but not index.
- Forward iterations are only possible. Backward iterations are not possible.
- We can't start the loop from the middle.

### Steps for working with ForIn Loop

```
for (variable in array)  
{  
  array[variable]  
}
```

- For loop is index based. The variable contains "index".
- Forward iterations are only possible. Backward iterations are not possible.
- We can't start the loop from the middle.

### Example on For Loop

```
<html>  
  <head>  
    <title>for</title>  
  </head>  
  <body>  
    <h1>for</h1>  
    <script>  
      var cities = ["Hyderabad", "Chennai", "Bangalore", "Pune"];  
      for (var i = 0; i < cities.length; i++)  
      {  
        console.log(cities[i]);  
      }  
    </script>  
  </body>  
</html>
```

### Example on ForEach Loop

```
<html>  
  <head>  
    <title>Foreach</title>  
  </head>  
  <body>  
    <h1>Foreach</h1>  
    <script>  
      var cities = ["Hyderabad", "Chennai", "Bangalore", "Pune"];  
      cities.forEach(function (city)  
      {
```

```
        console.log(city);
    });
</script>
</body>
</html>
```

### Example on ForOf Loop

```
<html>
<head>
    <title>ForOf</title>
</head>
<body>
    <h1>ForOf</h1>
    <script>
        var cities = ["Hyderabad", "Chennai", "Bangalore", "Pune"];
        for (var city of cities)
        {
            console.log(city);
        }
    </script>
</body>
</html>
```

### Example on ForIn Loop

```
<html>
<head>
    <title>ForIn</title>
</head>
<body>
    <h1>ForIn</h1>
    <script>
        var cities = ["Hyderabad", "Chennai", "Bangalore", "Pune"];
        for (var i in cities)
        {
            console.log(i + ", " + cities[i]);
        }
    </script>
</body>
</html>
```