

UI TECHNOLOGIES

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

by Harsha Vardhan (UI Expert)



Contents

UI Technologies	22
Fundamentals	22
Application	22
Client	22
Browser	22
Web application	23
Http	23
HTML 4 & 5	24
HTML Basics	24
Introduction to HTML.....	24
HTML Versions	24
Tag	24
Syntax of HTML Program.....	25
Steps to Prepare First Example in HTML.....	25
1. Installing Visual Studio Code	25
2. Create HTML Program	31
3. Execute the HTML Program	34
Understanding HTML Syntax.....	35
<html>	35
<head>.....	36
<body>.....	36
<title>.....	36
Attributes	36
DOCTYPE.....	37
Basic Tags in HTML	37
Headings	37
Paragraphs.....	39
Line Breaks	39
Text Formatting Tags	40
Bold	40
Italic.....	40
Underline	41
Strikeout	41
Strong.....	42

Emphasis	42
Superscript	43
Subscript	43
Images.....	44
Images	44
Hyperlinks	45
Hyperlinks.....	45
List tags	49
Unordered List	49
Ordered List	50
Definition List	51
Tables.....	52
Table Tags.....	52
Miscellaneous Tags.....	55
IFrame	55
HTML Entities.....	57
Meta	58
Forms	59
Form	59
Input Tag	60
TextBox.....	61
Password TextBox	61
CheckBox	62
Checked Attribute	62
Radio Button	62
File Browse Button	63
Reset Button	64
Submit Button	64
Name Attribute	65
Login Form.....	65
Registration Form.....	66
Post Submission	66
Image Submit Button	67
General Button.....	68
Hidden Field	68
Color	68
Date	69

Time	69
Datetime-Local.....	70
Month.....	70
Week.....	71
Search.....	71
Number	71
Range.....	72
Email.....	72
Url	73
Maxlength Attribute	73
Value Attribute	74
Readonly Attribute	74
Disabled Attribute.....	74
Tabindex Attribute	75
ID Attribute	75
ID Attribute	76
Placeholder Attribute	76
Autofocus Attribute	76
Required Attribute	77
Pattern Attribute.....	77
Min and Max Attributes	78
Step Attribute	78
Novalidate Attribute.....	79
FormAction, FormMethod, FormTarget Attribute	80
Form Attribute	80
Multiple Attribute	81
AutoComplete Attribute	81
Button Tag	82
Fieldset	83
Legend.....	83
Label	84
DropdownList.....	85
Option Groups.....	86
ListBox	87
Selected Attribute.....	87
Textarea	88
<div> and	89
DIV	89

Span	89
Advanced Tags.....	89
Horizontal Ruler.....	89
Pre-formatted Text	90
Abbreviations.....	90
Bi-Directional Override.....	91
Audio	92
Video	92
Details and Summary.....	93
Figure and Figcaption.....	94
DataList.....	94
ProgressBar	96
Meter.....	97
Output.....	97
Article.....	98
Semantic Tags.....	99
Header.....	99
Nav.....	99
Section	99
Footer.....	99
Aside.....	100
Storage	100
Local Storage.....	100
Session Storage.....	102
Geo Location.....	103
Web Workers	105
Drag and Drop	107
Offline Apps.....	110
Server Sent Events.....	111
Canvas.....	114
Creating Canvas Container.....	114
Get context of canvas	114
strokeStyle	114
lineWidth	114
strokeRect	114
fillStyle.....	114
fillRect.....	115

Example on Canvas	115
SVG.....	115
Syntax to create SVG container	115
Example on SVG	115
Deprecated Tags / Attributes	116
Deprecated / Removed Tags in HTML 5.....	116
Deprecated / Removed Attributes in HTML 5	117
XHTML	117
Introduction to XHTML	117
XHTML Rules	118
XHTML – Online Validator	119
XHTML - Example.....	119
CSS 2 & 3	120
Fundamentals of CSS.....	120
Introduction to CSS	120
CSS Basic Selectors.....	120
First Example on CSS.....	121
Example on ID Selector	121
CSS - Properties	121
Colors.....	122
color.....	122
background-color.....	122
Types of colors	123
Font Styles	125
font-family.....	125
font-size	126
font-weight.....	127
font-style	127
Text Styles.....	128
letter-spacing	128
word-spacing	129
line-height	130
text-decoration	131
text-transform	132
text-align	133
text-indent.....	134
.....	136

Background Image	136
background-image.....	136
background-repeat.....	137
background-position	139
background-attachment	140
Lists	141
list-style-type for 	141
list-style-type for 	143
list-style-image	145
DIV.....	146
<div> tag	146
"width" and "height" properties.....	147
float	148
clear.....	149
Box Model	150
Understanding Box Model.....	150
border-style.....	151
border-width	151
border-color	151
border - shortcut.....	152
border - sides.....	153
margin	155
margin - sides	156
margin - shortcut	157
padding	159
padding - sides	160
padding - shortcut	160
Advanced CSS Properties.....	161
"opacity" property.....	161
display	161
visibility.....	162
overflow.....	163
position.....	165
position: static.....	165
position: absolute	166
z-index.....	167
position: relative	168

position: fixed	169
Types of Style Sheets	170
1. Internal Style Sheet / Embedded Style Sheet.....	170
2. External Style Sheet.....	170
3. Inline Style Sheet.....	171
CSS Selectors	172
Tag Selector.....	173
ID Selector	173
Class Selector	174
Compound Selector.....	174
Grouping Selector.....	175
Child Selector	176
Direct Child Selector	176
Adjacent Siblings Selector.....	177
Adjacent One Sibling Selector	178
Attribute Selector	179
Hover Selector	179
Focus Selector.....	180
Universal Selector.....	181
First-child selector.....	181
Last-child selector	182
Nth-child selector	183
Nth-child(even) selector	184
Nth-child(odd) selector.....	185
Before selector	185
After selector	186
Selection selector	187
CSS Style Precedence	187
CSS Realtime Examples	189
Table Styles	189
Hyperlink Styles	190
Menubar.....	191
Header.....	192
Advanced CSS	195
Resize	195
Word Wrap.....	196
RGBA	196
Border Radius.....	197

Understanding the Columns	248
Grid System with Responsive Web Design	252
Jumbotron	257
Jumbotron	257
Images.....	258
Image Shapes	258
Image Alignment	259
Image Fluid	260
Tables.....	261
Basic Table	261
Borderless Table	263
Bordered Table	265
Striped Table	267
Hover Table	269
Table Background Colors.....	271
Table Header Background Colors	273
Table Small	276
Table Responsive	277
Alerts	279
Alerts.....	279
Buttons	281
Button Colors	281
Button Outline	282
Button Sizes.....	284
Button Groups.....	284
Button Vertical Groups	285
Button DropDown	286
Badges	287
Basic Badges.....	287
Badge Colors	288
Pill Badges	289
Progress Bar.....	290
Basic Progress Bar	290
Progress Bar Colors	291
Pagination	293
Basic Pagination.....	293
Pagination Size.....	294

Pagination Alignment	295
Breadcrumbs.....	296
Breadcrumbs	296
List Groups	297
Basic List Groups.....	297
List Group Colors	298
Cards	300
Basic Cards.....	300
Card Colors	301
Card Images	303
Card Groups	304
Tooltip	306
Tooltip.....	306
Popover	307
Popover.....	307
Collapsible	308
Basic Collapsible	308
Link Collapsible	309
Accordion	310
Forms	312
Inline Form.....	312
Stacked Form.....	313
Form Grid	316
Horizontal Form Grid.....	318
Input Groups.....	321
Form Validation	322
Navigation.....	325
Basic Navigation	325
Navigation Alignment	326
Vertical Navigation.....	327
Tabs	329
Pills	330
Tabs with DropDown.....	332
Navigation Bar.....	333
Basic Navigation Bar	333
NavBar Collapsible	336
NavBar DropDown.....	337

NavBar Search	339
NavBar FixedTop.....	341
NavBar Sticky Top.....	342
Scrollspy.....	344
Carousel.....	347
Carousel	347
Modal	350
Modal.....	350
LESS	352
Fundamentals of LESS	352
Introduction to LESS.....	352
Steps to Prepare First Example in LESS.....	352
1. Downloading and Installing WinLess	352
2. Creating LESS File	355
3. Compile LESS File into CSS File	359
4. Import the CSS File into HTML File.....	362
5. Run the HTML File	365
Basic LESS.....	365
Variables	365
Mixins	366
Example on Mixins.....	367
Mixins With Parameters.....	367
Nested Rules	369
Example on Nested Rules	369
Advanced LESS	370
Operators.....	370
Example on Operators	371
Color Functions	371
Example on Operators	372
JavaScript 5, 6 & 7	373
Fundamentals	373
Introduction to JavaScript.....	373
Syntax of JavaScript	373
console.log()	373
Steps to Prepare First Example in JavaScript	374
1. Installing Visual Studio Code	374
2. Create JavaScript Program	379

3. Execute the JavaScript Program	384
Variables.....	385
Operators.....	386
Arithmetical Operators.....	386
Assignment Operators.....	387
Increment and Decrement Operators.....	388
Relational Operators.....	388
Logical Operators	389
Concatenation Operator.....	389
Control Statements	390
if.....	390
Switch-case	395
While	395
Do-While	396
For.....	397
Functions.....	398
Arguments and Return.....	400
Recursion	401
Arrays.....	402
Steps for development of arrays	402
Object Oriented Programming in JavaScript.....	404
Introduction to Objects	404
Types of Object Oriented Programming (OOP) languages.....	405
Creating Objects	405
Object Literals.....	405
Constructor Function	406
Object.Keys	407
JSON.....	408
Stringify	408
Parse.....	409
Object Array Literal.....	410
Object Array	411
Prototype.....	412
Inheritance.....	412
Data Types	413
String	414
typeof.....	415

undefined vs null	415
== and ===.....	416
String Function.....	417
ToString Function	417
Number Function	418
ParseInt Function.....	419
ParseFloat Function	420
+ Unary Operator	421
toFixed() function.....	421
String Functions	422
toUpperCase()	422
toLowerCase()	423
length	423
charAt()	424
charCodeAt()	424
substr()	424
indexOf()	425
replace()	426
split()	427
trim().....	427
concat()	428
Date Functions	428
new Date()	428
toLocaleDateString().....	429
toLocaleTimeString().....	429
getTime()	429
getDay()	430
getDate().....	431
getMonth().....	431
getFullYear().....	431
getHours().....	432
getMinutes()	432
getSeconds()	433
getMilliseconds()	433
Creating a Custom Date:.....	433
Advanced.....	434
NoScript	434
Closures	435

Hoisting.....	436
DOM	436
I) Window	438
1. location.href.....	438
2. navigator.userAgent	438
3. navigator.screenX.....	438
4. navigator.screenY.....	438
5. alert().....	439
6. confirm()	439
7. print()	440
8. setTimeout().....	440
9. setInterval().....	441
10. scrollTo().....	441
11. open()	442
II) Document	443
1. title	443
2. head.....	443
3. body.....	443
4. images	443
5. links	443
6. URL	443
document.write()	444
document.getElementById()	444
document.getElementsByName()	444
document.getElementsByTagName().....	445
document.getElementsByClassName()	445
document.querySelectorAll().....	446
document.querySelector()	446
III) Element.....	447
tagName	447
id	447
innerHTML	448
innerText	448
style	448
parentElement	449
children	449
scrollTop	450
setAttribute()	451

getAttribute()	451
removeAttribute()	451
attributes.....	452
hasAttribute()	452
focus()	453
click()	453
remove().....	454
document.createElement()	454
appendChild().....	454
addEventListener()	455
Introduction to Events	455
List of Events	455
“Click” event.....	456
“Dblclick” event	456
“Mouseover” and “Mouseout” events	457
“Mousemove” event.....	458
“Keyup” event	459
“Keypress” event	460
“Focus” and “Blur” events	462
“Change” event.....	463
“Contextmenu” event	466
“Cut”, “Copy”, “Paste” events.....	467
“this” keyword	468
Login - Example.....	468
Add, Subtract, Multiply, Divide Example	469
Validations	471
Regular Expressions	472
Types of JavaScript	474
1. Internal JavaScript	474
2. External JavaScript.....	474
AJAX	475
Introduction to AJAX	475
Advantages of AJAX	476
Types of AJAX request	476
The “XMLHttpRequest” class.....	476
AJAX – NodeJS – Simple - Example	478
AJAX – NodeJS – Json Object - Example	479
AJAX – NodeJS – Json Array - Example	481

JavaScript - New Features	483
Introduction to JavaScript 6.....	483
Class-based OOP	483
Classes.....	483
Classes.....	485
Constructors.....	486
Methods	487
Inheritance.....	488
Method Overriding	490
Misc. Concepts.....	491
Set and Get Methods.....	491
Default Arguments.....	493
Arrow Functions.....	494
Let.....	494
Const	495
Rest (...) Operator	495
Destructuring	497
Multiline Strings	498
String Interpolation.....	498
Reading Elements from Array	499
jQuery	503
Fundamentals of jQuery	503
Generations of jQuery.....	503
Formats of jQuery Library file.....	503
Downloading jQuery	504
“\$” Function	504
Steps to Prepare First Example in jQuery	504
1. Installing Visual Studio Code	504
2. Create jQuery Program.....	509
3. Execute the HTML Program	513
Manipulating Content	514
Get html().....	514
Get text().....	515
Set html()	516
Set text()	517
before()	517
prepend()	518

append()	519
after()	520
insertBefore()	520
prependTo()	521
appendTo()	522
insertAfter()	523
wrap()	524
wrapAll()	524
empty()	525
remove()	526
replaceWith()	527
Event Handling	527
List of jQuery Functions to handle Events	527
The "click()" function	528
The "dblclick()" function	529
The "mouseover()" function and "mouseout()" function	530
The "hover()" function	531
The "mousemove()" function	532
The "focus()" function	533
The "keyup()" function	535
The "keypress()" function	536
The "change()" function	537
The "on()" function	540
The "off()" function	541
The "contextmenu" event	542
The "cut", "copy", "paste" events	543
The "data()" function	544
Effects	545
toggle()	548
"this" keyword	549
fadeTo()	550
Manipulating Attributes	552
Manipulating CSS	555
Animations	560
document.ready()	565
Selectors	566
Tag Selector	568

ID Selector	569
Class Selector	569
Compound Selector.....	571
Grouping Selector.....	571
Child Selector (or) Descendent Selector	572
Direct Child Selector	574
Adjacent Siblings Selector.....	575
Adjacent One Sibling Selector	576
:first filter	577
:last filter	577
:even filter	578
:odd filter	579
:eq filter	579
:gt filter	580
:lt filter	581
:not filter.....	582
Attribute Selector	583
Attribute Selector - Not.....	584
Attribute Selector – Starts With	585
Attribute Selector – Ends With	586
Attribute Selector – Contains	587
:contains filter	588
:has filter.....	589
:empty filter	590
:first-child filter.....	591
:last-child filter.....	592
:nth-child filter	594
:only-child filter.....	595
parent()	596
next()	597
prev()	598
siblings().....	598
children().....	599
index()	600
Form Filters	601
Table Styles	607
Append – Advanced Examples	609
jQuery – JavaScript Objects - Examples.....	612

CDN	616
jQuery UI	617
Introduction to jQuery UI	617
datepicker()	617
spinner()	618
autocomplete()	619
tooltip()	620
resizable()	620
draggable()	621
droppable()	622
selectable()	623
sortable()	624
accordion()	625
tabs()	626
dialog()	627
buttonset()	628
menu()	629
progressbar()	630
slider()	630
Color Animation	631
Easing Effects	632
addClass() with animation	633
removeClass() with animation	634
Effects	635
jQuery Validations	637
jQuery AJAX	639
jQuery AJAX	639
jQuery AJAX – NodeJS – Simple - Example	639
jQuery AJAX – NodeJS – Get - Example	642
jQuery AJAX – NodeJS – Search - Example	644
jQuery AJAX – NodeJS – Post - Example	647
jQuery AJAX – NodeJS – Put - Example	650
jQuery AJAX – NodeJS – Delete - Example	652
jQuery AJAX – .NET – Simple - Example	655
jQuery AJAX – .NET – Get - Example	657
jQuery AJAX – .NET – Search - Example	660
jQuery AJAX – .NET – Post - Example	663
jQuery AJAX – .NET – Put - Example	666

jQuery AJAX – .NET – Delete - Example	668
jQuery AJAX – .NET – Grid - Example	671

HARSHA

UI Technologies

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

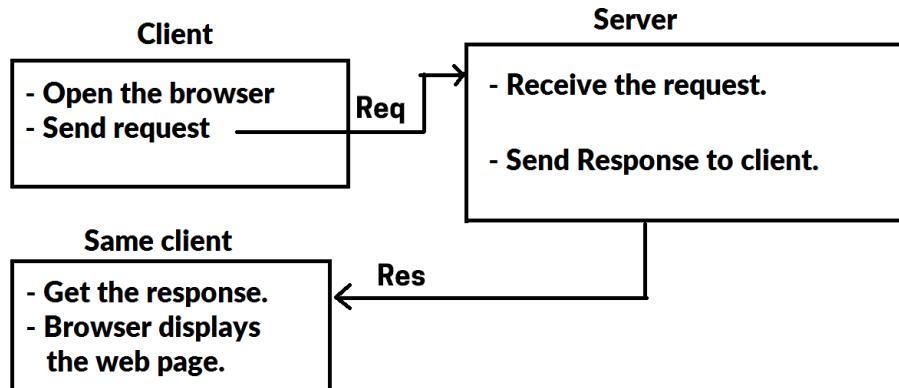
- Opera

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 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:
 - 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>
 - 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".

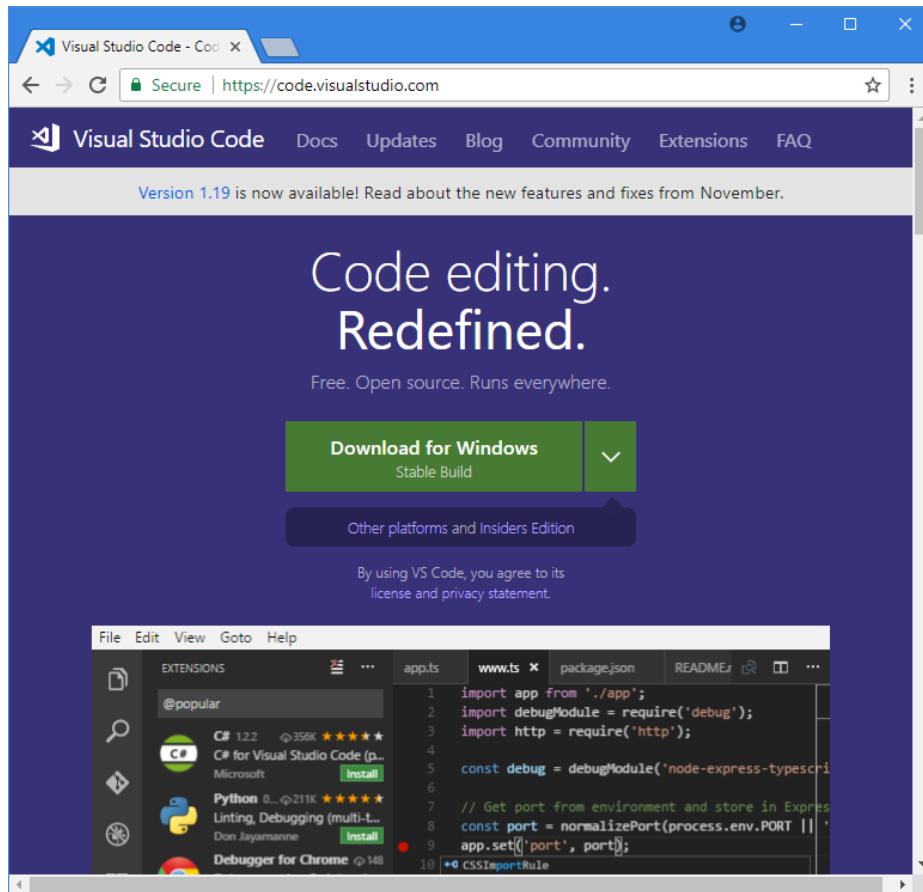
Steps to Prepare First Example in HTML

- Installing Visual Studio Code
- Creating HTML Program
- Executing HTML Program

1. Installing Visual Studio Code

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

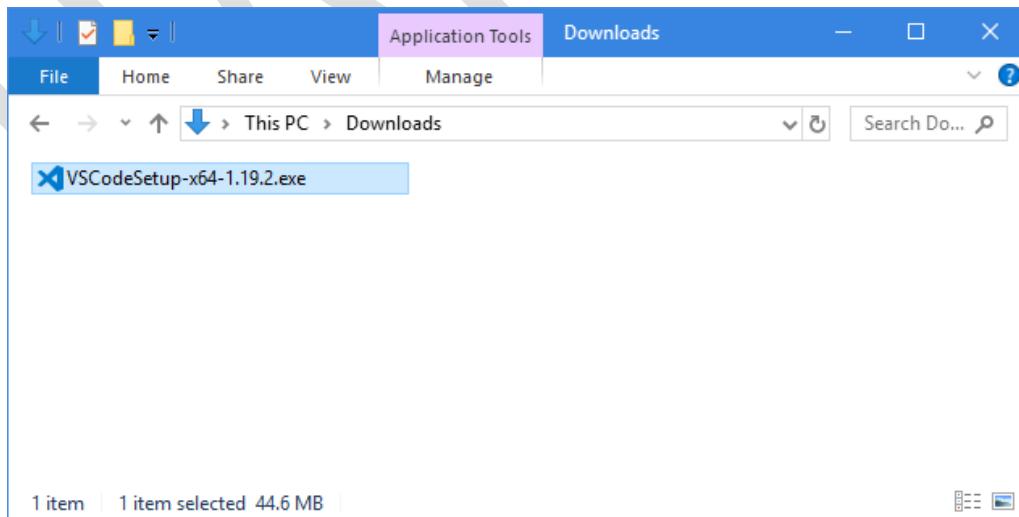
Go to <https://code.visualstudio.com>



Click on “Download for Windows”.

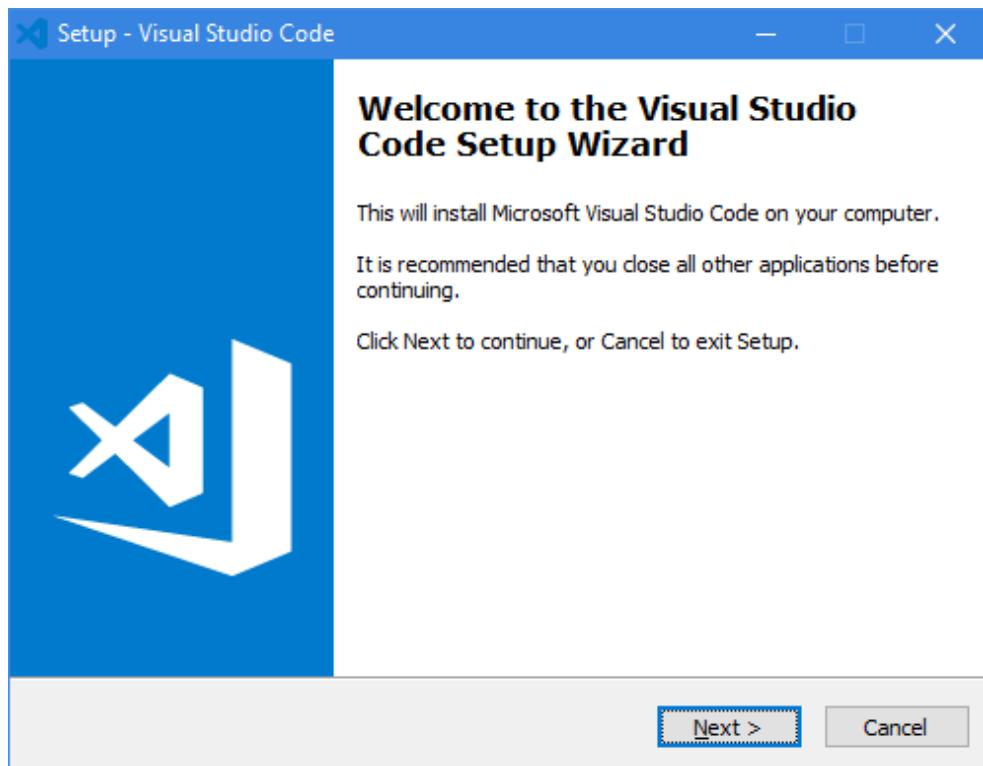
Note: The version number may be different at your practice time.

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

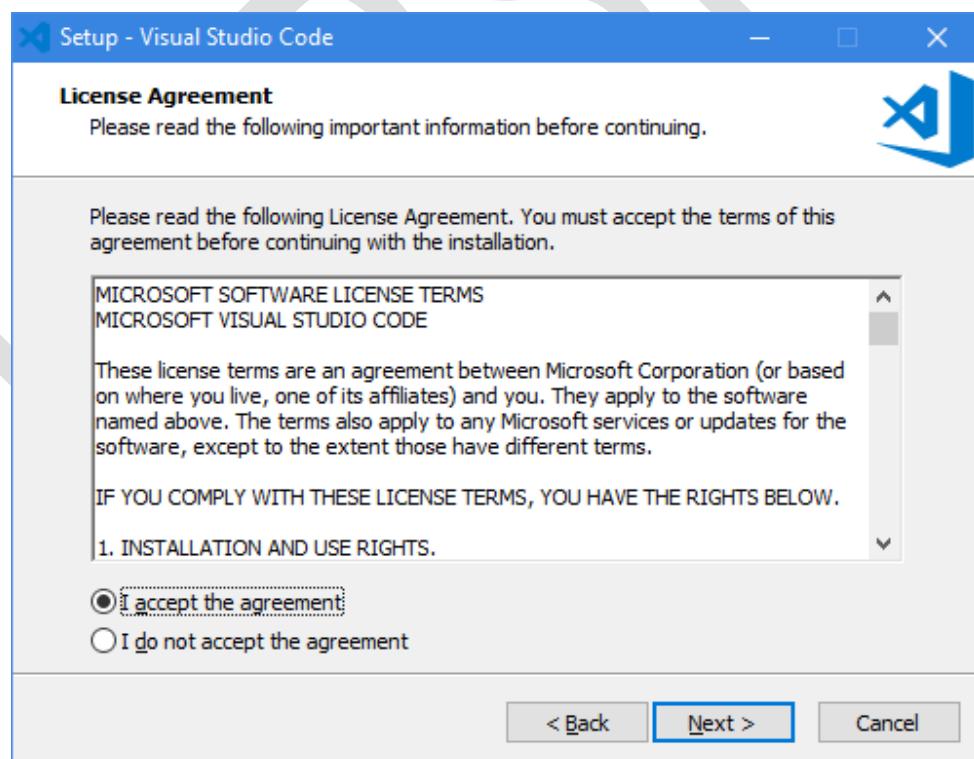


Double click on “VSCodeSetup-x64-1.19.2.exe” file.

Click on “Yes”.

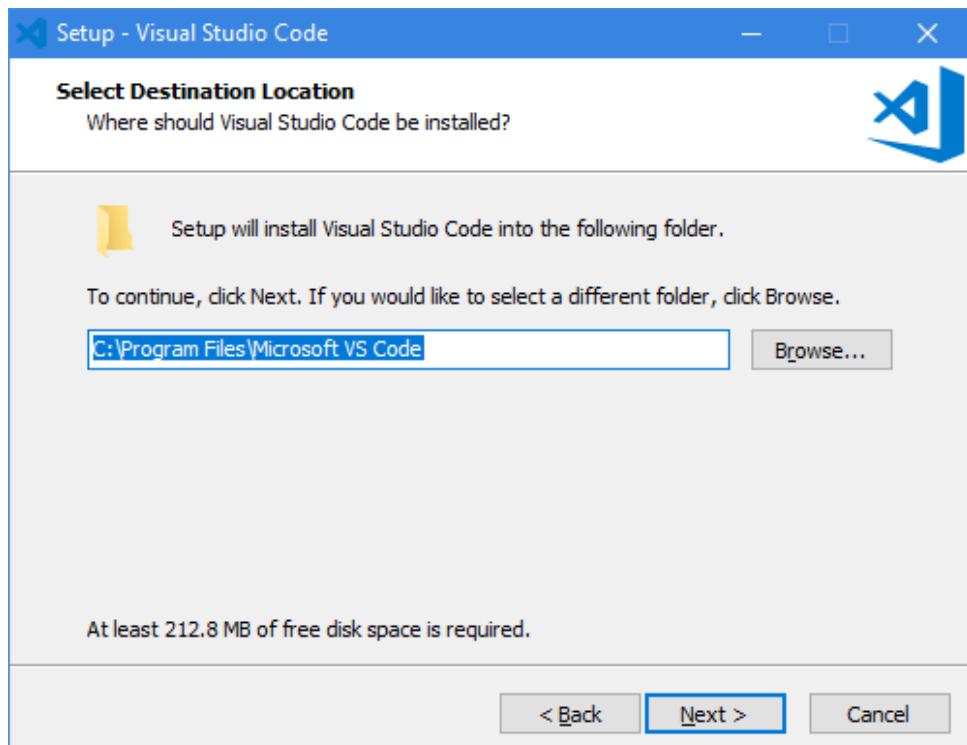


Click on “Next”.

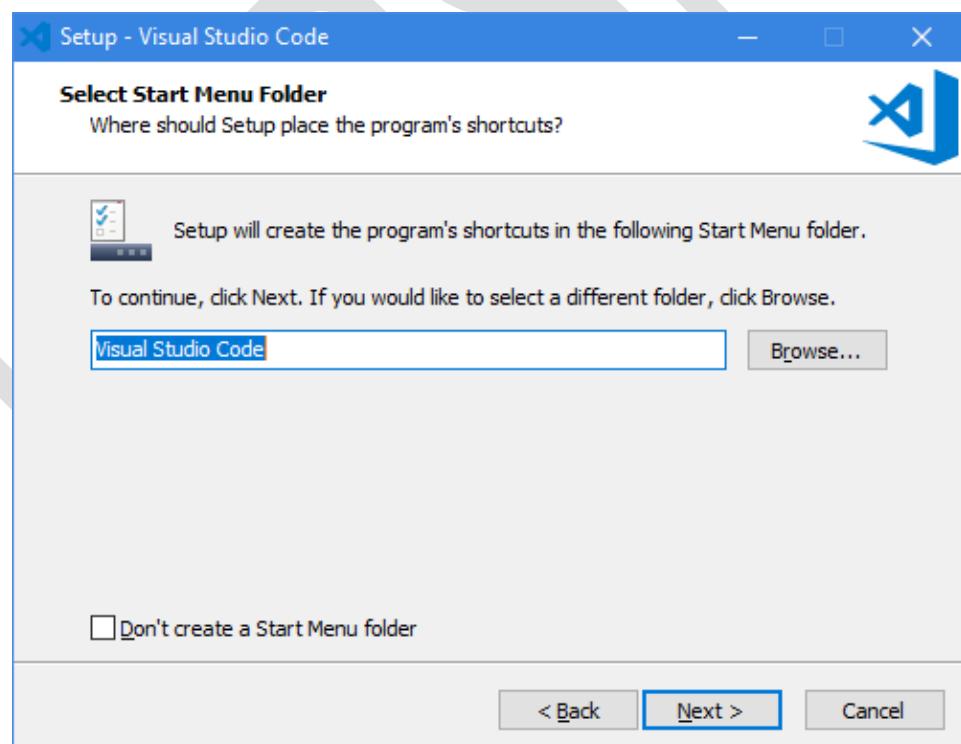


Click on “I accept the agreement”.

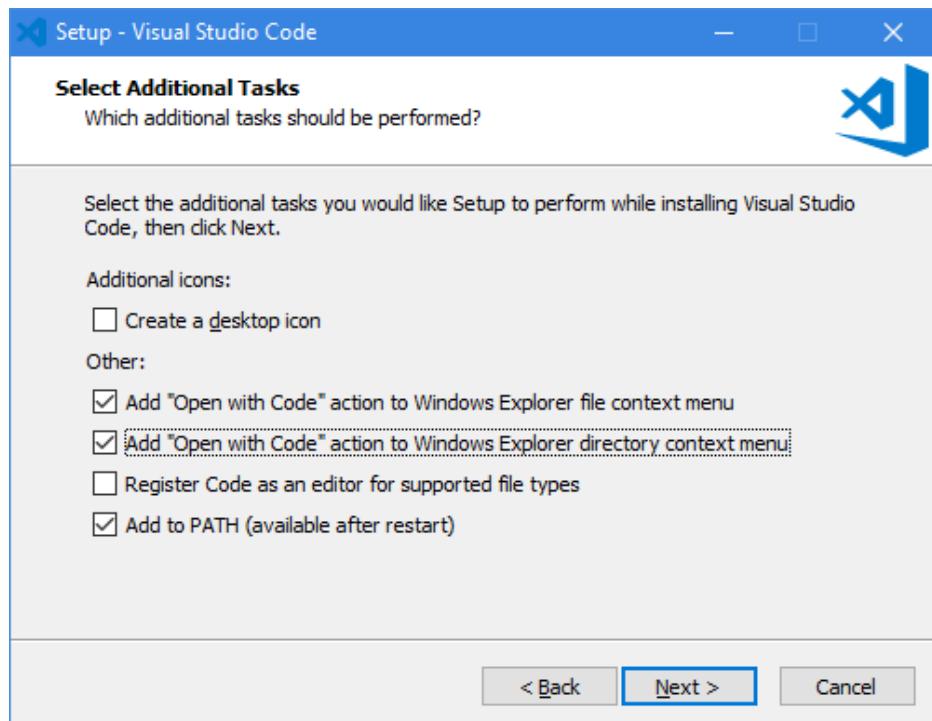
Click on “Next”.



Click on “Next”.



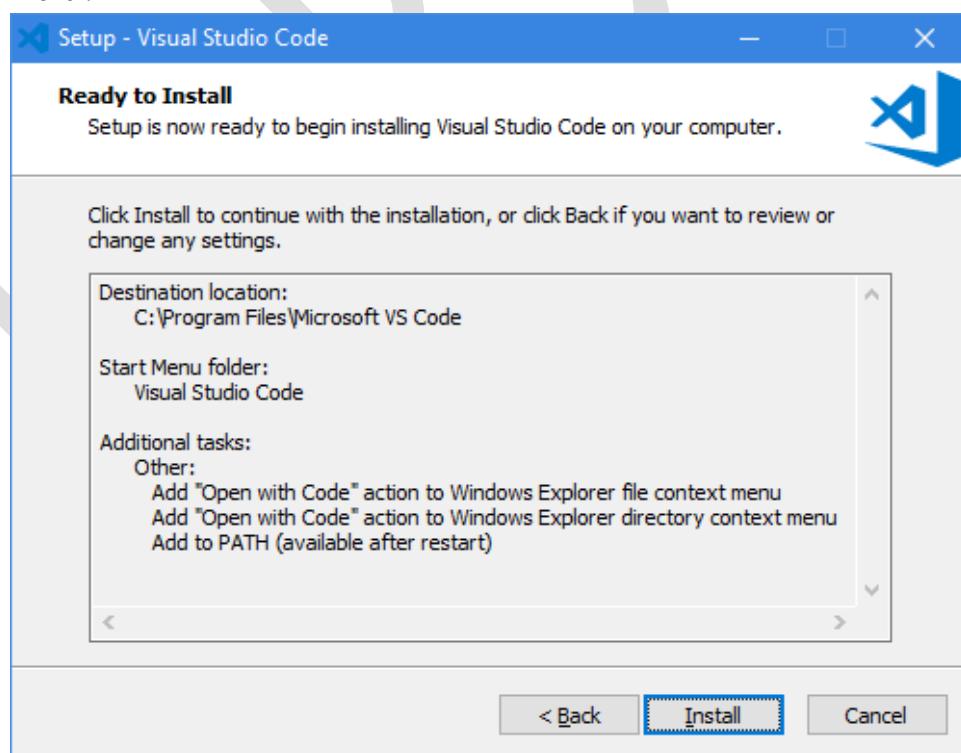
Click on “Next”.



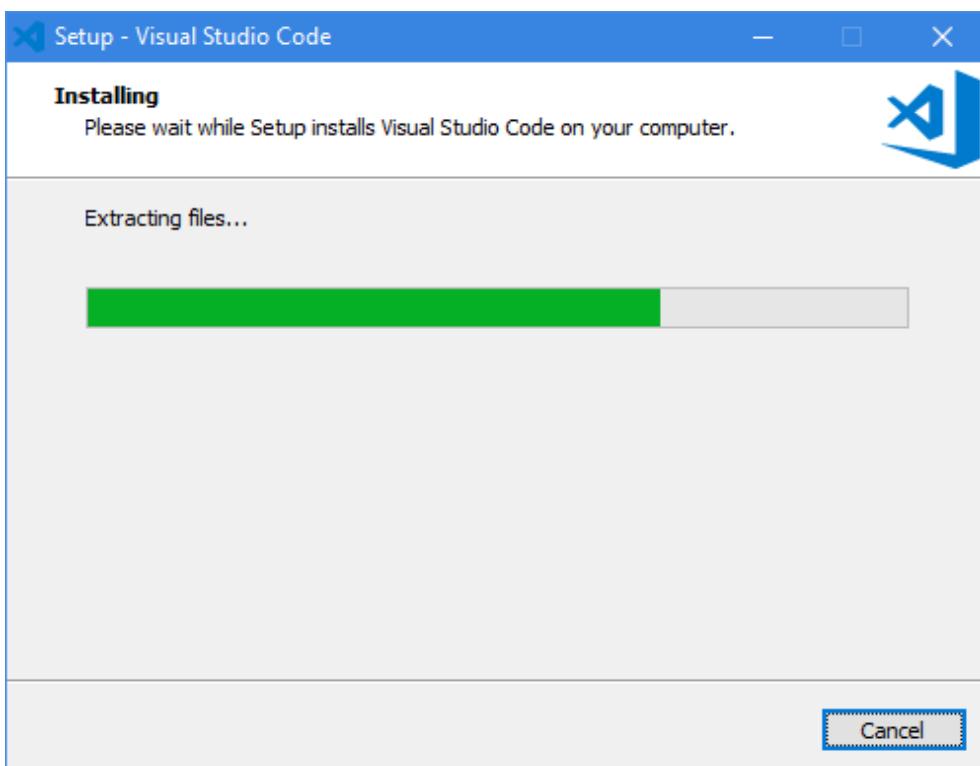
Check the checkbox “Add Open with Code action to Windows Explorer file context menu”.

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

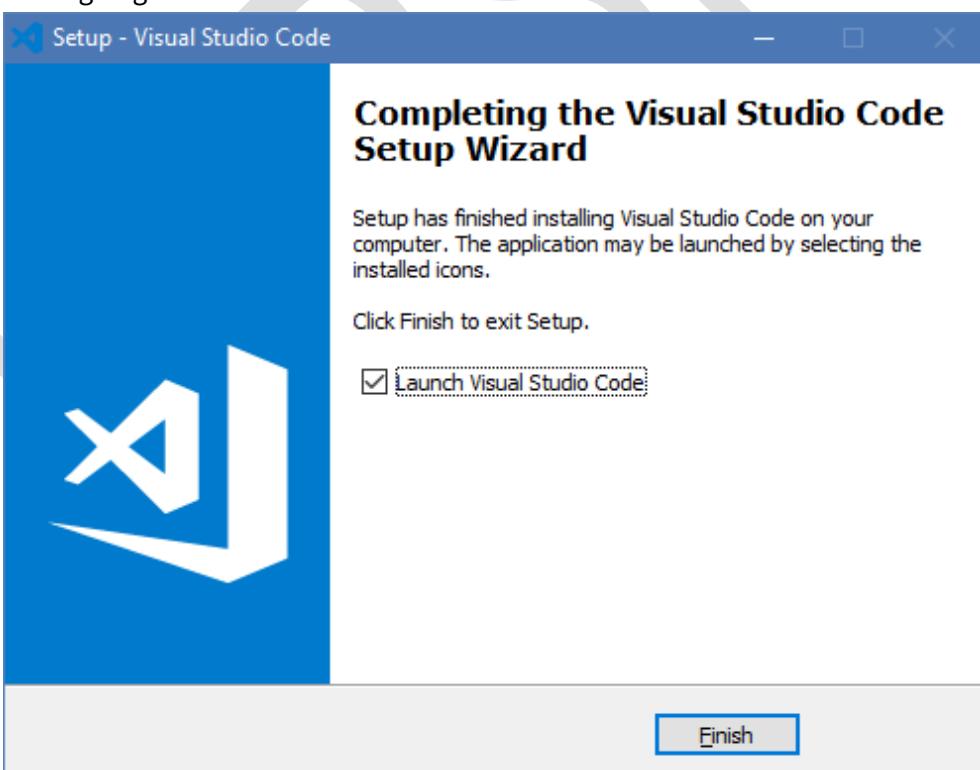
Click on “Next”.



Click on “Install”.



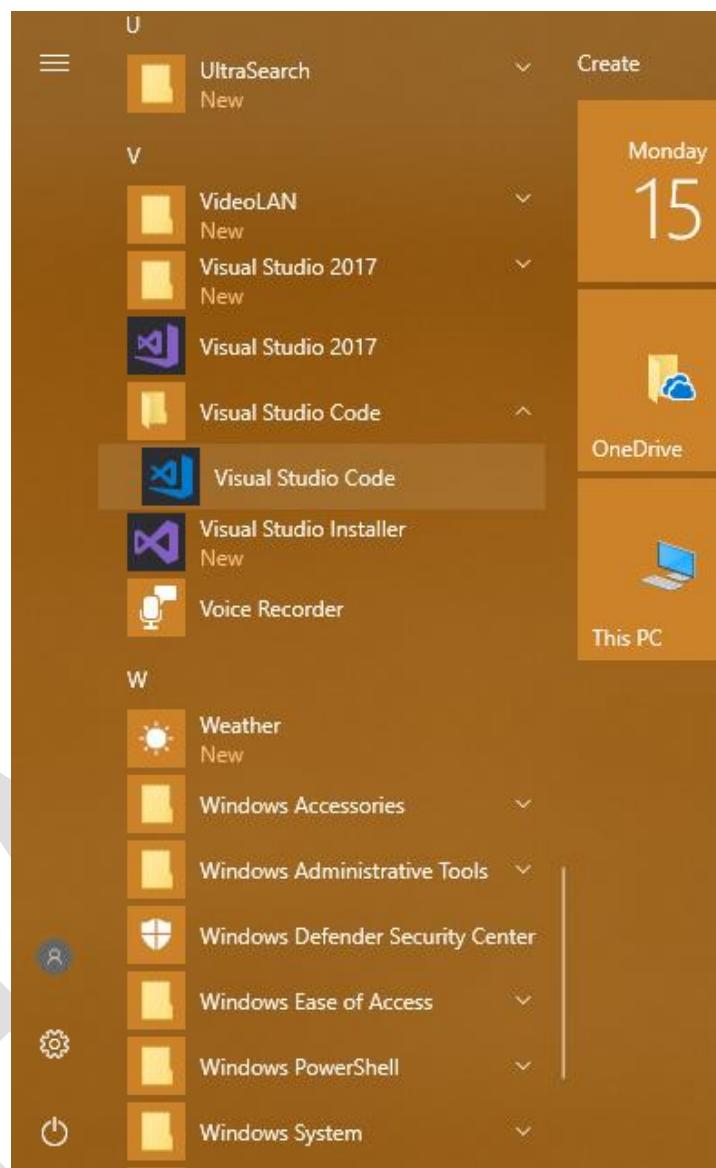
Installation is going on....



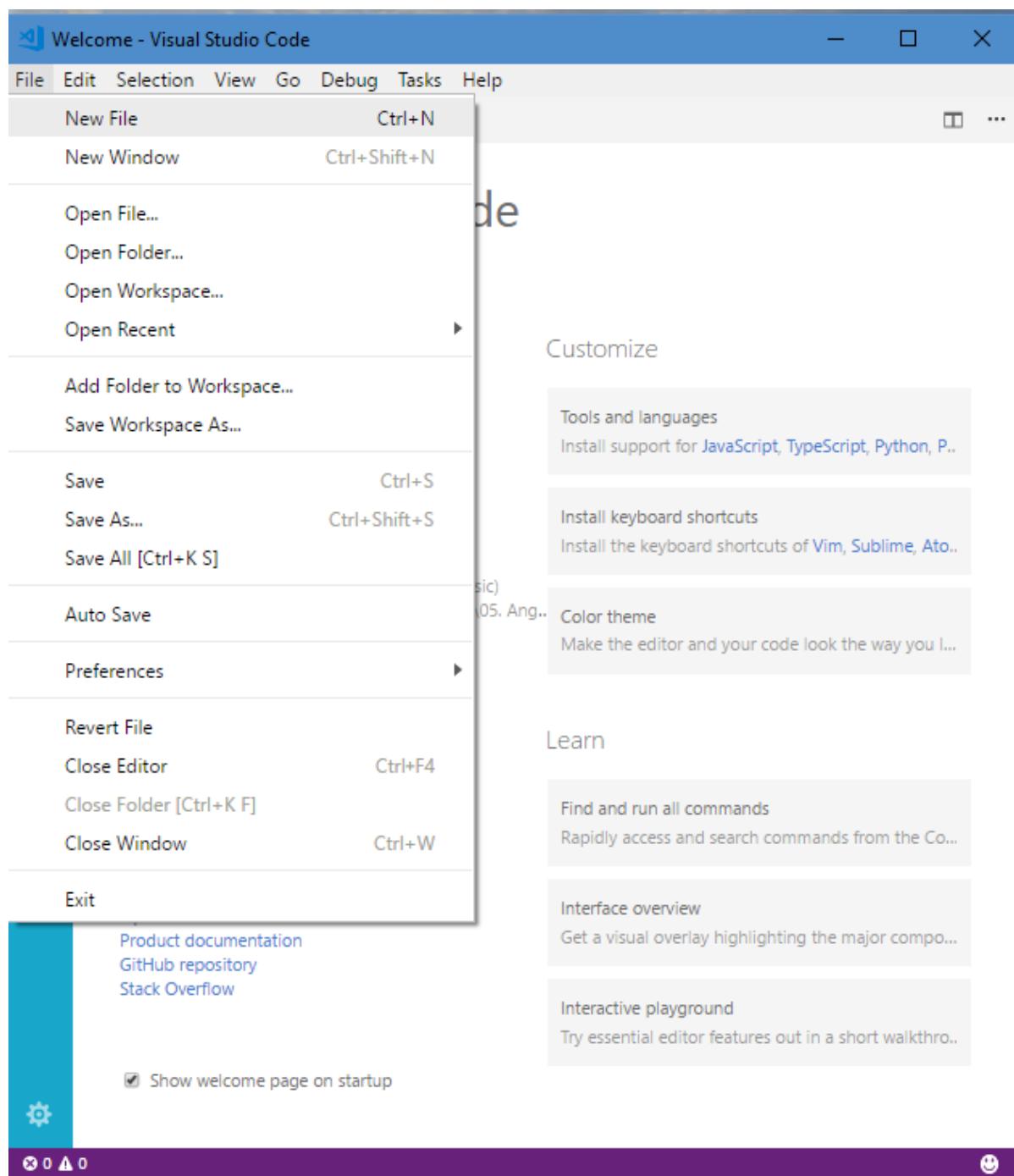
Click on "Finish".

2. Create HTML Program

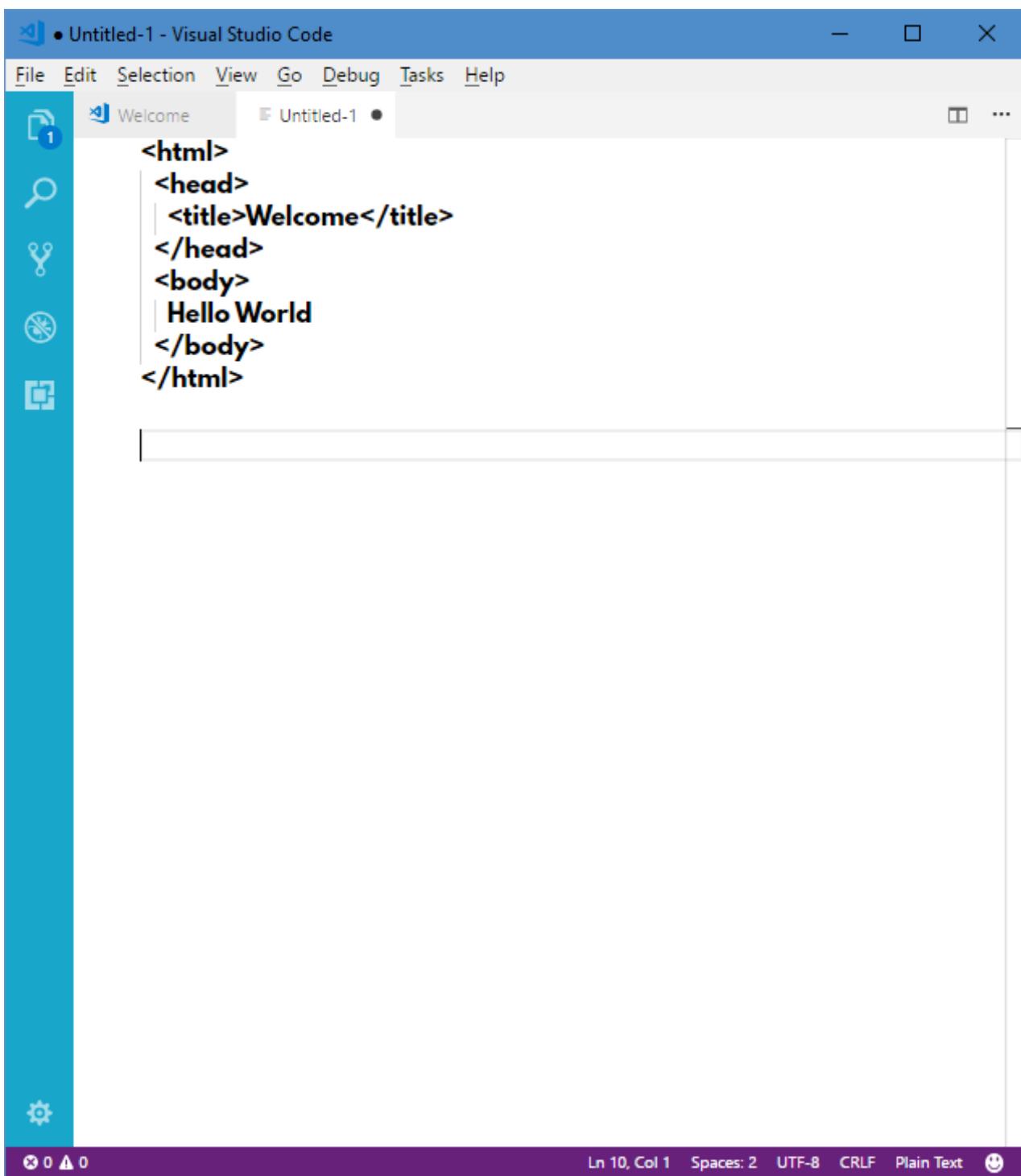
- Open “Visual Studio Code”, by clicking on “Start” – “Visual Studio Code”.



- Visual Studio Code opened.



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

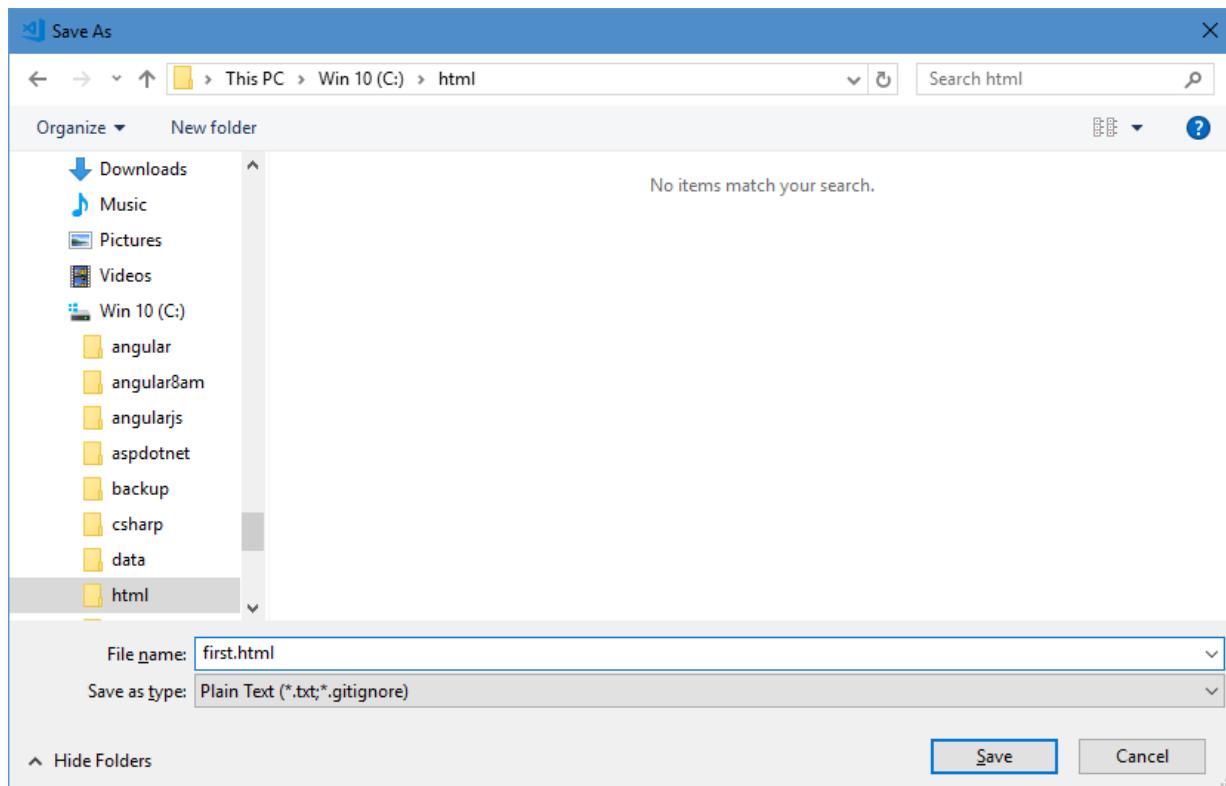


The screenshot shows the Visual Studio Code interface. The title bar reads "• Untitled-1 - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Debug, Tasks, and Help. The left sidebar has icons for File Explorer, Search, Problems, and Terminal, with a gear icon at the bottom. The main editor area displays the following HTML code:

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

The status bar at the bottom shows "Ln 10, Col 1" and "Spaces: 2" and other file details.

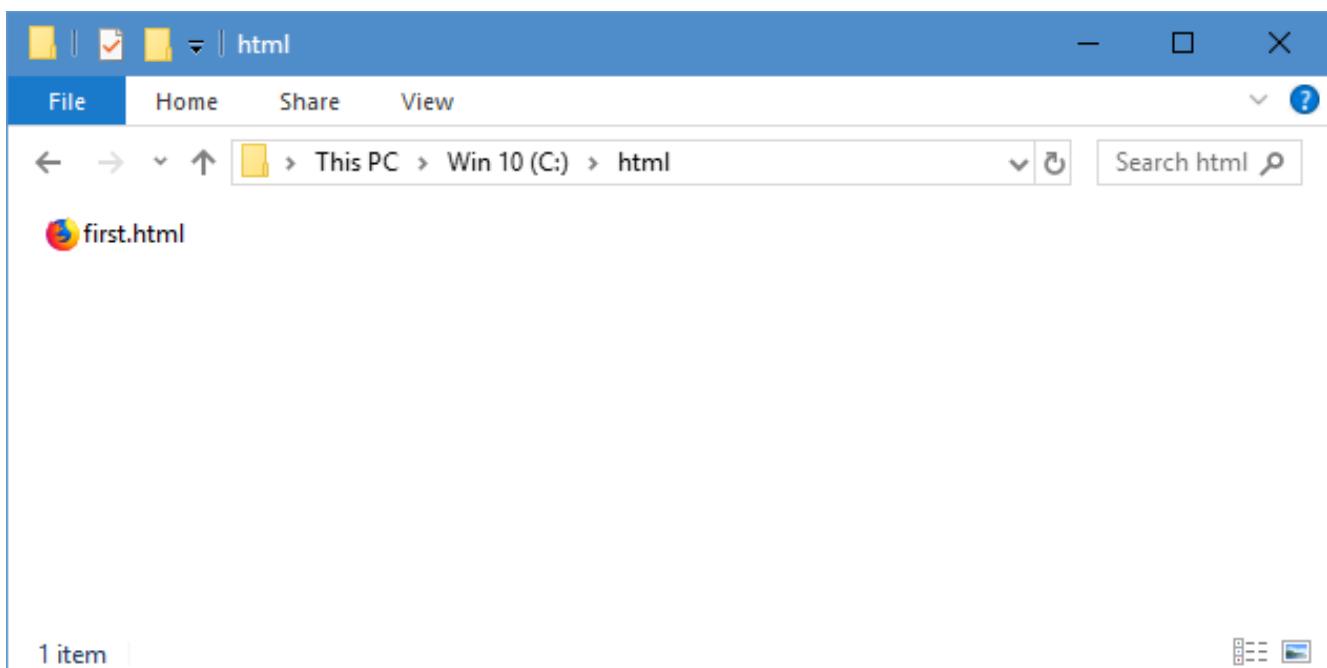
- Go to “File” menu – “Save” (or) Press Ctrl+S.



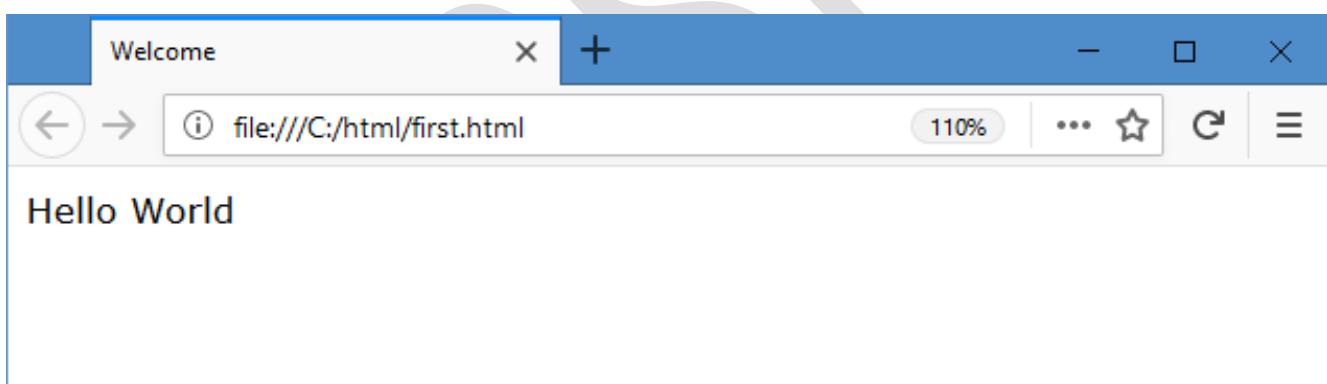
- Go to "c:\\" and click on "New folder". Enter the new folder name as "html".
- Select "c:\html" folder and enter the filename as "first.html".
- Click on "Save".
- Now the html file (c:\html\first.html) is ready.

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

-
 tag is used to create "line breaks".
- It moves the cursor to the next line.

- The content next to the
 will be display in the next line.
- It is an unpaired tag. That means no closing tag for
 tag.

Syntax:

```
<br>
```

Example:

```
<br>
```

Example on


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

Text Formatting Tags

Bold

- tag is used to display the text in bold.
- The text enclosed within the tag will be display as bold.
- Use 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

```
<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 displayed 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 to 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>
```

Strong

- tag is used to display the text in strong.
- and tags are visually same.
- The strong tag content will be pronounced "strongly" (stressfully) by the screen readers for the blind people.
- It is a paired tag.

Syntax:

```
<strong>strong text</strong>
```

Example:

```
<strong>Hello</strong>
```

**Example on **

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

Emphasis

- tag is used to display the text in emphasis (special status).
- and <i> tags visually same.

- The strong tag content will be pronounced stylishly by the screen readers for the blind people.
- It is a paired tag.

Syntax:

```
<em>emphasis text</em>
```

Example:

```
<em>Hello</em>
```

Example on

```
<html>
  <head>
    <title>Emphasis</title>
  </head>
  <body>
    <h1>Emphasis</h1>
    This is normal text.
    <i>This is italic text</i>
    <em>This is emphasis text</em>
  </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:

_{subscript text}

Example:

_{Hello}

Example on <sub>

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

Images

Images

- 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

tag in real-time, because, if the image is not loaded, at least the "alt" text appears to the user.

Example on

```
<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

```
<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

- tag is used to display the list of items with bullets.
- UL stands for “Un-Ordered List”.
- Inside tag, you should place one or more tags. tag stands for "List Item", which represents a single item in the list.
- Use 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 **

```
<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 **

```
<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>
```

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	 	
<	less than	<	<
>	greater than	>	>
&	ampersand	&	&
¢	cent	¢	¢
£	pound	£	£
¥	yen	¥	¥
€	euro	€	€
§	section	§	§
©	copyright	©	©
®	registered trademark	®	®

™	trademark	™	™
---	-----------	---------	---------

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>
```

Forms

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 | 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¶m2=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>
```

```

        </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 . 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 . It represents the value in the form of pixels.
- The "height" attribute is used to specify the height of the image in case of . 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>
```

```
</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 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>
```

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">

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>
```

FormAction, FormMethod, FormTarget Attribute

FormAction Attribute

- "formaction" attribute specifies server url, to which you want to submit the form. If the user clicks on "Submit" button, the form data (parameter names and values) will be submitted to the url specified in "formaction" attribute.

Syntax: <input type="submit" formaction="server url here">

FormMethod Attribute

- "formmethod" attribute specifies type of submission: GET or POST.
- In case of "GET" submission, the parameter names and values are visible in the browser's address bar.
- In case of "POST" submission, the parameter names and values are not visible in the browser's address bar, but will be sent in hidden mode (request body).

Syntax: <input type="submit" formmethod="get | post">

FormTarget Attribute

- "formtarget" attribute opens the server page in separate browser tab, after submission.

Syntax: <input type="submit" formtarget="_blank">

Example on Formaction Formmethod Formtarget

```
<!DOCTYPE html>
<html>
  <head>
    <title>formaction formmethod formtarget</title>
  </head>
  <body>
    <h1>formaction formmethod formtarget</h1>
    <form>
      First name:<br>
      <input type="text" name="FirstName"><br>
      Last name:<br>
      <input type="text" name="LastName"><br>
      <input type="submit" formaction="http://localhost/someaddress" formmethod="get"
            formtarget="_blank" value="Submit">
    </form>
  </body>
</html>
```

Form Attribute

- Used to make the element as a member of the form; so that the element also will be submitted along with other elements, while submitting the form.
- We have to specify ID of the <form> tag in "form" attribute of <input> tag.

Syntax: <input type="..." form="id of form">

Example on Form

```
<!DOCTYPE html>
<html>
  <head>
    <title>form</title>
  </head>
  <body>
    <h1>form</h1>
    <form action="http://localhost/someaddress" id="form1">
      First name:<br>
      <input type="text" name="firstname" id="txt1"><br>
      <input type="submit" value="Submit">
    </form>
    <hr>
    Last name (logically member of form1):<br>
    <input type="text" name="lastname" id="txt2" form="form1">
  </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 if 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="icici">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>
    </form>
  </body>
</html>

```

```

<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="icici">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>

```

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

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

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

```

Pre-formatted Text

- <pre> tag is used to display the text as-it-is, along with the spaces and line breaks.
- It is a paired tag.
- Generally, <pre> tag is meant for displaying computer programs (for example, "for" loop syntax) as-it-is.
- It is rare to use <pre> tag in real-time.

Syntax:

```
<pre>your text here</pre>
```

Example:

```

<pre>
  one  two
  three    four
            five
</pre>

```

Example on <pre>

```

<html>
  <head>
    <title>Pre</title>
  </head>
  <body>
    <h1>Pre</h1>
    one  two
    three    four
    <hr>

    <pre>
      one  two

      three    four
      five
    </pre>
  </body>
</html>

```

Abbreviations

- <abbr> tag is used to display full-form of a short-form when the user places mouse pointer on it.

- HTML / Browser doesn't provide any built-in abbreviations. We have to set both short form and full form in the code itself.
- It is a paired tag.
- The "title" tag represents the tooltip (full form), which appears when the user places mouse pointer on the content. The <title> can be applicable for any html tag (not only for <abbr> tag).

Syntax:

```
<abbr title="full form here">short form here</abbr>
```

Example:

```
<abbr title="as soon as possible">ASAP</abbr>
```

Example on <abbr>

```
<html>
  <head>
    <title>Abbr</title>
  </head>
  <body>
    <h1>Abbr</h1>
    <abbr title="as soon as possible">asap</abbr>
  </body>
</html>
```

Bi-Directional Override

- <bdo> is used to display the text in reverse (right-to-left) order.
- The default is "left-to-right" (LTR).
- The "dir" (direction) attribute decides whether the text should be "left to right" or "right to left".
- The text enclosed within the <bdo> tag will be displayed in the specific direction.
- It is a paired tag.

Syntax:

```
<bdo dir="rtl">your text here</bdo>
```

Example:

```
<bdo dir="rtl">Hai how are you</bdo>
```

Attributes:

1. **dir:**
 - **ltr:** It displays the text in left-to-right.
 - **rtl:** It displays the text in right-to-left.

Example on <bdo>

```
<html>
  <head>
    <title>Bdo</title>
  </head>
  <body>
```

```
<h1>Bdo</h1>
<p>Hai how are you</p>
<p><bdo dir="rtl">Hai how are you</bdo></p>
</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>
```

Figure and Figcaption

- <figure> and <figcaption> tags are used to represent an image along with its description.
- Search engines like google can identify the image based on its description.

Syntax:

```
<figure>
  
  <figcaption>description here</figcaption>
</figure>
```

Example on <figure> and <figcaption>

```
<!DOCTYPE html>
<html>
  <head>
    <title>figure</title>
  </head>
  <body>
    <h1>figure</h1>
    <figure>
      
      <figcaption>Pulpit rock in Norway</figcaption>
    </figure>
  </body>
</html>
```

Note: Copy and paste “pulpit.jpg” into current folder.

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>

```

Meter

- <meter> tag is used to display percentage of utilization.
- **Ex:** Disk usage.

Syntax:

```
<meter min="minimum" max="maximum" value="value here">
</meter>
```

Example on <meter>

```
<!DOCTYPE html>
<html>
  <head>
    <title>meter</title>
  </head>
  <body>
    <h1>meter</h1>
    Disk usage:
    <meter min="0" max="100" value="60">
    </meter>
  </body>
</html>
```

Output

- <output> tag is used to display the result of some calculation.
- The <output> tag is readonly; the user can see the value; but can't modify the value.
- **Ex:** Displaying “tax” based on “amount”.

Syntax:

```
<output id="id here" >
</output>
```

Example on <output>

```
<!DOCTYPE html>
<html>
  <head>
    <title>output</title>
  </head>
  <body>
    <h1>output</h1>
    <form>
      Product Price:<br>
      <input type="text" name="a" value="1000"><br>
      Discount:<br>
      <input type="text" name="b" value="50"><br>
      Net Price:
      <output name="c" for="a b"></output>
    </form>
  </body>
</html>
```

```

</form>

<script>
    document.getElementsByName("a")[0].addEventListener("keyup", fun1);
    document.getElementsByName("b")[0].addEventListener("keyup", fun1);
    function fun1()
    {
        var val1 = document.getElementsByName("a")[0].value;
        var val2 = document.getElementsByName("b")[0].value;
        var val3 = parseInt(val1) - parseInt(val2);
        document.getElementsByName("c")[0].value = val3;
    }
</script>
</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 don'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 don'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 don't provide any styles by default; we have to apply styles manually, using CSS.

Syntax:

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

Example on Template 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);  
    </script>
```

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

```

<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, status)
        {

```

```

    if (status == google.maps.GeocoderStatus.OK)
    {
        if (results[0])
        {
            var areaname= results[0].formatted_address ;
            document.getElementById("div3").innerHTML = areaname;
        }
        else
        {
            alert("address not found");
        }
    }
    else
    {
        alert("Geocoder failed due to: " + status);
    }
}

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>
```

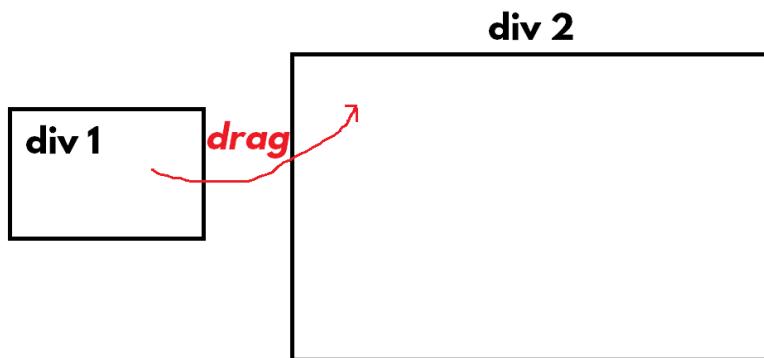
script.js

```
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.



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>

```

Offline Apps

- “Offline Apps” concept enables the browser to load the web page from the server for the first time and store it in the browser memory; but second time onwards, it loads the page from the browser memory directly, instead of loading it from the server.
- Advantage:** The user can view the web page, even without network connection.

Steps for development of Offline Apps

- Create a “manifest” file:

filename.appcache

```
CACHE MANIFEST
#id
filename.jpg
filename.css
filename.js
```

- Set the “manifest” file to the html file:

<html manifest="filename.appcache">

Example on Offline Apps

c:\html\StyleSheet1.css

```
body,input
{
    font-family: Tahoma;
    font-size: 16px;
}
```

c:\html\JavaScript.js

```
function fun1()
{
    alert("hai");
}
```

c:\html\sample.appcache

```
CACHE MANIFEST
# 101
JavaScript.js
StyleSheet1.css
img1.jpg
```

c:\html\index.html

```

<!DOCTYPE html>
<html manifest="sample.appcache">
  <head>
    <title>Title here</title>
    <link href="StyleSheet1.css" rel="stylesheet">
    <script src="JavaScript.js"></script>
  </head>
  <body>
    <h1>Heading</h1>
    <p>para</p>
    <p><input type="button" value="Click me" id="btn1"></p>
    <p></p>

    <script>
      document.getElementById("btn1").addEventListener("click", fun1);
    </script>
  </body>
</html>

```

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

Execution

- Download and install “nodejs” from “<https://nodejs.org>”. Click here to find steps to install NodeJS.
- Open “Command Prompt” and run the following commands:


```

npm install http-server -g
cd c:\html
http-server

```
- Open browser and enter the url: <http://localhost:8080/index.html>

Server Sent Events

- Browser send continuous requests to server, for every "n" seconds. Ex: 5 seconds.
- This concept is used to get continuous updated information (live information) from server.
- **Ex:** Cricket score board, election results, share market etc.
- The "EventSource" constructor function is used to start sending continuous requests to server.
- Every time when we get response (update) from server, the ""message" event executes for "EventSource".
- The "server url" represents the address of server side program, to which you want to send request.
- The "event.data" property represents the actual data that is received from the browser.

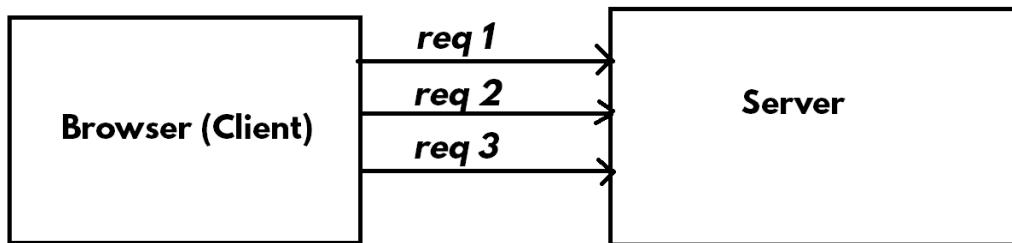
Steps for development of Offline Apps

- **Create a server side program:**

- Set the response content type as “text/event-stream”
- Write data to response object
- Flush the response

- **Create a html page and call the server program:**

```
var variablename = new EventSource("server url");
variablename.addEventListener("message", functionname);
function functionname(event)
{
    //event.data = response from server
}
```



Example on Server Sent Events

c:\html\httpserver.js

```
var http = require("http");
var fs = require("fs");
var server = http.createServer(engine);
server.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}

function engine(request, response)
{
    if (request.url == "/" || request.url == "/index.html")
    {
        fs.readFile("index.html", "utf8", fun1);
```

```

function fun1(error, data)
{
    if (error)
    {
        response.setHeader("content-type", "text/html");
        response.writeHead(404);
        response.write("<h1 style='color:red>Page not found</h1>");
        response.end();
    }
    else
    {
        response.setHeader("content-type", "text/html");
        response.writeHead(200);
        response.write(data);
        response.end();
    }
}
else if (request.url.startsWith("/getdata"))
{
    console.log("request received at " + new Date().toLocaleTimeString());
    response.setHeader("content-type", "text/event-stream");
    response.writeHead(200);
    response.write("data: Response at " + new Date().toLocaleTimeString() + "\n\n");
    response.end();
}
}

```

c:\html\index.html

```

<!DOCTYPE html>
<html>
<head>
    <title>Server sent events</title>
</head>
<body>
    <h1>Server sent events</h1>
    <div id="result"></div>

    <script>
        var source = new EventSource("/getdata");
        source.addEventListener("message", fun1);
        function fun1(event)
        {
            document.getElementById("result").innerHTML += event.data + "<br>";
        };
    </script>
</body>
</html>

```

Execution:

- Download and install “nodejs” from “<https://nodejs.org>”. Click here to find steps to install NodeJS.
- Open “Command Prompt” and run the following commands:

```
cd c:\html  
node httpserver.js
```
- Open browser and enter the url: <http://localhost:8080/index.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 loses 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.

- <acronym>**
 - Use <abbr> tag instead.
- <applet>**
 - Java applets is out-dated concept.
- <basefont>**
 - Use CSS to set the default font.
- <big>**
 - Use CSS to increase text size.
- <center>**
 - Use CSS to set center alignment.
- <dir>**
 - Use or to display list of items.
- **
 - Use CSS to set font.
- <frame>**
 - Use <iframe>.
- <frameset>**
 - Use <iframe>
- <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		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.

XHTML**Introduction to XHTML**

- XHTML is “HTML” + “set of rules to maintain good quality and standards” in the html code.
- XHTML is the strict and cleaner version of HTML, recommended by W3C (World Wide Web Consortium).
- XHTML was released in 2000.

- In realtime, XHTML is recommended.

XHTML Rules

1. <!DOCTYPE> is must.

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

2. <html>, <head>, <title>, <body> tags are must.

3. A <meta> tag with “content-type” is mandatory.

```
<meta http-equiv="Content-type" content="text/html; charset=UTF-8">
```

That means the html file has the content of “text/html” type; and the character encoding format is “UTF” (Unicode Transformation Format), which supports all worldwide language characters.

4. “xmlns” attribute is must for <html> tag.

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

An “xmlns” (XML namespace) contains all the pre-defined html tags.

5. XHTML tags and attributes must be in lower case only.

Ex: <h1>hello</h1>

6. Attribute’s values should be in double quotes.

Ex:

7. All the paired tags must be closed.

Ex: <p>Hello</p>

8. All the unpaired tags must be end with “/”.

Ex:

9. The inner-most tags should be closed first.

Ex: <p><i>Hello</i></p>

10. Attribute minification not allowed. We must write both attribute and value.

Ex: readonly="readonly"

11. The following tags are only allowed in <body> tag directly. All other tags must be used only inside any of the following tags.

- <p>
- <table>

- o <div>
- o
- o
- o <h1> to <h6>

12. “alt” attribute is must for tag.

XHTML – Online Validator

- Open browser: Ex: Mozilla Firefox
- Go to <https://validator.w3.org>.
- Click on “Validate by File Upload”.
- Click on “Browse”.
- Select the html file. Ex: xhtml.html
- Click on “Check”.
- It shows the list of errors or “Passed” message.

XHTML - Example

xhtml.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>this is title</title>
  <meta http-equiv="Content-type" content="text/html;charset=UTF-8" />
  </head>
  <body>
    <h1>hai</h1>
    <p>
      <input type="text" />
      <b> <i>this is bold and italic</i> </b>
      <input type="checkbox" checked="checked" />
    </p>
    <hr />
  </body>
</html>
```

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
```

```

    {
        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>

```

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
      {
        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. 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-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. 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-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.</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>

```


- represents a small amount of text, for which you can apply some special formatting.
- is a paired tag.
- 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">Background Image Example</p>
  </body>
</html>
```

```

</style>
</head>
<body>


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.


</body>
</html>

```

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

background-repeat

- This property specifies repeat mode of the background image.
- The default value is "repeat".
- If the value is "repeat", the background image repeats horizontally and vertically within the image.
- If the value is "no-repeat", the background image will not be repeated; will be displayed only once at top left corner.
- If the value is "repeat-x", the background image will be repeated horizontally.
- If the value is "repeat-y", the background image will be repeated vertically.

Syntax: background-repeat: repeat | no-repeat | repeat-x | repeat-y;

Example: background-repeat: no-repeat;

Example of "background-repeat" – "no-repeat":

```

<html>
<head>

```

```
<title>CSS - No-repeat</title>
<style type="text/css">
    #p1
    {
        background-color: skyblue;
        background-image: url('sample.png');
        background-repeat: no-repeat;
    }
</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.</p>
</body>
</html>
```

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

Example of "background-repeat" – "repeat-x":

```
<html>
    <head>
        <title>CSS - Repeat-x</title>
        <style type="text/css">
            #p1
            {
                background-color: skyblue;
                background-image: url('sample.png');
                background-repeat: repeat-x;
            }
        </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.</p>
    </body>
</html>
```

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

Example of "background-repeat" – "repeat-y":

```
<html>
    <head>
        <title>CSS - Repeat-y</title>
        <style type="text/css">
            #p1
            {
```

```

        background-color: skyblue;
        background-image: url('sample.png');
        background-repeat: repeat-y;
    }

```

</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.</p>

</body>

</html>

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

background-position

- This property specifies position of the background image.
- The default value is "top left".

Syntax: background-position: top left | top center | top right |
 center left | center center | center right | bottom left |
 bottom center | bottom right;

Example: background-position: top center;

- The "background-position" property will be useful only when "background-repeat" is set to "no-repeat".

Example of "background-position" property

```

<html>
  <head>
    <title>CSS - Background-position</title>
    <style type="text/css">
      #p1
      {
        background-color: skyblue;
        background-image: url('sample.png');
        background-repeat: no-repeat;
        background-position: bottom center;
        text-align: justify;
      }
    </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
  </body>
</html>

```

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. </p>

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

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

Lists

list-style-type for

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

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

Example: list-style-type: disc;

Example of "list-style-type" property for

```
<html>  
  <head>  
    <title>CSS - List-style-type - UL</title>  
    <style type="text/css">  
      #list1  
      {  
        list-style-type: disc;  
      }  
  
      #list2  
      {  
        list-style-type: square;  
      }  
    </style>  
  </head>  
  <body>  
    <ul id="list1">  
      <li>Item 1</li>  
      <li>Item 2</li>  
      <li>Item 3</li>  
    </ul>  
  
    <ul id="list2">  
      <li>Item 1</li>  
      <li>Item 2</li>  
      <li>Item 3</li>  
    </ul>  
  </body>  
</html>
```

```
#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 **

- This property specifies type of numbering for the 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 :

```
<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;
      }

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

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

      #list5
      {
        list-style-type: lower-roman;
      }
    </style>
  </head>
  <body>
    <ol id="list1">
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
    </ol>

    <ol id="list2">
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
    </ol>

    <ol id="list3">
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
    </ol>

    <ol id="list4">
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
    </ol>

    <ol id="list5">
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
    </ol>
  </body>
</html>
```

```
#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”.

- Top stop “float:left” sequence, we use “clear:left”.
- Top stop “float:right” sequence, we use “clear:right”.

Syntax: clear: left | right;

Example: clear: left;

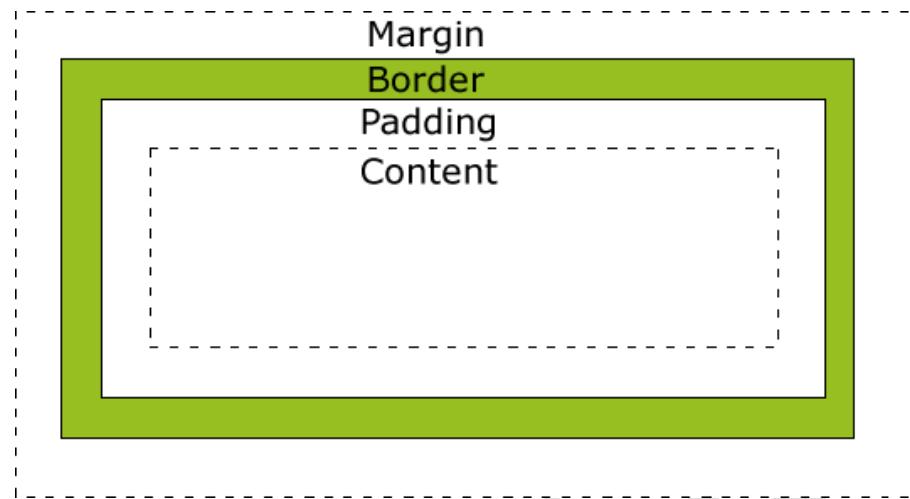
Example of “clear – left”

```
<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>

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

```

```

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">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;
      }
      #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>

```

```
</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;  
        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: toppadding rightpadding bottompadding leftpadding;

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">
      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
      {
        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;
      }
    </style>
  </head>
  <body>
    <div id="div1">Div 1</div>
  </body>
</html>

```

```
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;
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>

```

position

- This property specifies position of the element in the web page.
- It specifies where exactly the element has to be appear.
- By default, all the elements appear in sequence, how they have written in the program.

Syntax: position: static | absolute | relative | fixed;

Example: position: absolute;

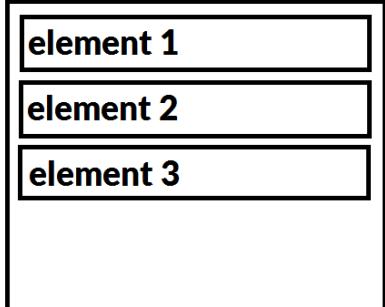
position: static

- The block level elements appear line-by-line; inline elements appear side-by-side.
- It is the default.

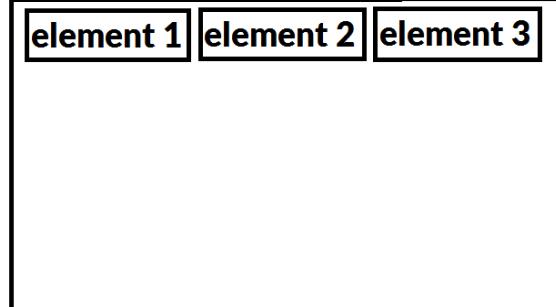
Syntax: position: static;

Example: position: static;

block level elements



inline elements



Example of “position:static”

```

<html>
    <head>
        <title>CSS - position: static</title>
        <style type="text/css">
            #div1
            {
                background-color: #ff66ff;

```

```

        position: static;
    }
    #div2
    {
        background-color: #00cccc;
        position: static;
    }

```

</style>

</head>

<body>

<div id="div1">div1</div>

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

span 1

span 2

</body>

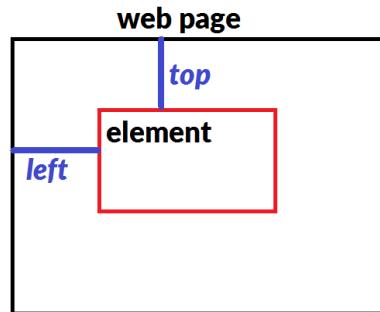
</html>

position: absolute

- It displays the element exactly in the specified position.
- It requires “left” and “top” properties.

Syntax: position: absolute; left: x; top: y;

Example: position: absolute; left:50px; top:50px;



Example of “position:absolute”

```

<html>
  <head>
    <title>CSS - position: absolute</title>
    <style type="text/css">
      #div1
      {
        background-color: #ff66ff;
        width: 100px;
        height: 100px;
        font-size: 25px;
        position: absolute;
        left: 200px;
        top: 250px;
      }

```

```

</style>
</head>
<body>
  <div id="div1">div1</div>
</body>
</html>

```

z-index

- This property specifies the display order (front / back side) of the element. If the z-index value is higher, it appears to the front to the user. If the z-index value is lower, it appears at backside to the user.

Syntax: z-index: n;

Example: z-index: 1;

Example of “z-index”

```

<html>
  <head>
    <title>CSS - z-index</title>
    <style type="text/css">
      #div1
      {
        background-color: #ff66ff;
        width: 100px;
        height: 100px;
        font-size: 25px;
        position: absolute;
        left: 200px;
        top: 250px;
        z-index: 3;
      }
      #div2
      {
        background-color: #00cccc;
        width: 120px;
        height: 120px;
        font-size: 25px;
        position: absolute;
        left: 250px;
        top: 170px;
        z-index: 2;
      }
      #div3
      {
        background-color: #00ff33;
        width: 120px;
        height: 120px;
        font-size: 25px;
        position: absolute;
        left: 280px;
        top: 150px;
        z-index: 1;
      }
    </style>
  </head>
  <body>
    <div id="div1">div1</div>
    <div id="div2">div2</div>
    <div id="div3">div3</div>
  </body>
</html>

```

```

        top: 190px;
        z-index: 1;
    }

```

</style>

</head>

<body>

```

<div id="div1">div1</div>
<div id="div2">div2</div>
<div id="div3">div3</div>
```

</body>

</html>

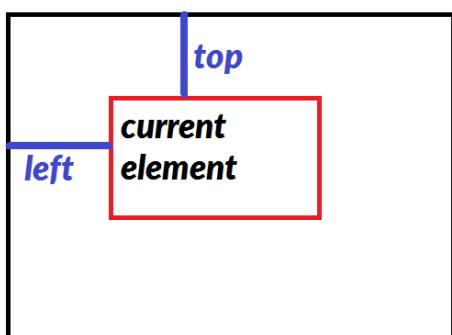
position: relative

- It also displays the element based on “x” and “y” co-ordinates (just like absolute).
- **Absolute:** x and y co-ordinates will be calculated from the browser.
- **Relative:** x and y co-ordinates will be calculated from the previous element.

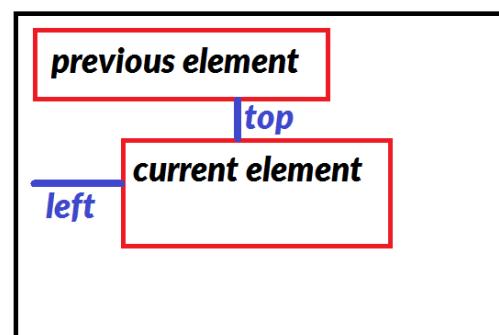
Syntax: position: relative; left: x; top: y;

Example: position: relative; left: 50px; top: 50px;

absolute



relative



Example of “position:relative”

```

<html>
  <head>
    <title>CSS - position: relative</title>
    <style type="text/css">
      #div1
      {
        background-color: #33ff99;
        height: 400px;
        font-size: 25px;
        margin: 50px;
        padding: 30px;
      }
      #div2
      {
        background-color: #00cccc;
      }
    </style>
  </head>
  <body>
    <div id="div1">div1</div>
    <div id="div2">div2</div>
  </body>
</html>

```

```

        width: 220px;
        height: 120px;
        font-size: 25px;
    }
    #div3
    {
        background-color: #ff3333;
        width: 120px;
        height: 120px;
        font-size: 25px;
        position: relative;
        left: 30px;
        top: 30px;
    }

```

</style>

</head>

<body>

<div id="div1">

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

<div id="div3">div3</div>

</div>

</body>

</html>

position: fixed

- It is same as “absolute”. That means the “x” and “y” co-ordinates will be calculated from the browser.
- When we scroll the web page, the fixed element will not be scrolled; appears in the fixed position.

Syntax: position: fixed; left: x; top: y;

Example: position: fixed; left: 50px; top: 50px;

Example of “position:fixed”

```

<html>
    <head>
        <title>CSS - position: fixed</title>
        <style type="text/css">
            #div1
            {
                background-color: #0066ff;
                width: 300px;
                height: 100px;
                font-size: 25px;
                position: fixed;
                left: 300px;
                top: 150px;
            }

```

</style>

</head>

<body>

```

<div id="div1">div1</div>
<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>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>

```

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. Adjacent Siblings Selector
9. Adjacent One Sibling Selector
10. Attribute Selector
11. Hover Selector
12. Focus Selector
13. Universal Selector
14. First-Child Selector
15. Last-Child Selector
16. Nth-Child Selector
17. Nth-Child Even Selector
18. Nth-Child Odd Selector
19. Before Selector
20. After Selector
21. Selection 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>
```

Adjacent Siblings Selector

- It selects all the adjacent (next) sibling tags of current tag.
- Siblings are the elements that have a common parent.
- "~" is the symbol of adjacent siblings selector.

Syntax: parenttag~childtag

Example: #p2~p

Example on Adjacent Siblings Selector

```
<html>
  <head>
    <title>CSS - Adjacent Siblings Selector</title>
    <style type="text/css">
      div~p
      {
        background-color: skyblue;
      }
    </style>
```

```

</head>
<body>
  <h1>Adjacent Siblings 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>
  <p>para 7</p>
  <p>para 8</p>
  <p>para 9</p>
  <p>para 10</p>
</body>
</html>

```

Adjacent One Sibling Selector

- It selects the immediate next sibling tag of current tag.
- "+" is the symbol of adjacent one sibling selector.

Syntax: currenttag+Siblingtag

Example: div+p

Example on Adjacent One Sibling Selector

```

<html>
  <head>
    <title>CSS - Adjacent One Sibling Selector</title>
    <style type="text/css">
      /*Selecting next <p> tag after </div> */
      div+p
      {
        background-color: skyblue;
      }
    </style>
  </head>
  <body>
    <h1>Adjacent One Sibling 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>
    <p>para 7</p>
    <p>para 8</p>
    <p>para 9</p>
  </body>

```

```

<p>para 10</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
      {
        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>
  </body>
</html>
```

```

<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>

```

First-child selector

- It selects the child tag, which is the first child of its parent tag.
- It is also called as “pseudo class”.

Note: Index starts from “1”.

Syntax: childtag:first-child

Example: p:first-child

Example on First-child selector

```

<html>
  <head>
    <title>CSS - First-child</title>
    <style type="text/css">
      /* selects first <p> in its parent */
      p:first-child
      {
        background-color: #33cc99;
      }
    </style>
  </head>
  <body>
    <div>
      <p>para 1</p>
      <p>para 2</p>
      <p>para 3</p>
      <p>para 4</p>
      <p>para 5</p>
    </div>
    <div>
      <p>para 6</p>
      <p>para 7</p>
      <p>para 8</p>
      <p>para 9</p>
      <p>para 10</p>
    </div>
    <div>
      <p>para 11</p>
      <p>para 12</p>
      <p>para 13</p>
      <p>para 14</p>
      <p>para 15</p>
    </div>
  </body>
</html>

```

Last-child selector

- It selects the child tag, which is the last child of its parent tag.
- It is also called as “pseudo class”.

Note: Index starts from “1”.

Syntax: childtag:last-child

Example: p:last-child

Example on Last-child selector

```

<html>
  <head>
    <title>CSS - Last-child</title>

```

```

<style type="text/css">
    /* selects last <p> in <div> */
    p:last-child
    {
        background-color: #33cc99;
    }
</style>
</head>
<body>
    <div>
        <p>para 1</p>
        <p>para 2</p>
        <p>para 3</p>
        <p>para 4</p>
        <p>para 5</p>
    </div>
    <div>
        <p>para 6</p>
        <p>para 7</p>
        <p>para 8</p>
        <p>para 9</p>
        <p>para 10</p>
    </div>
    <div>
        <p>para 11</p>
        <p>para 12</p>
        <p>para 13</p>
        <p>para 14</p>
        <p>para 15</p>
    </div>
</body>
</html>

```

Nth-child selector

- It selects the child tag, which is the n^{th} child tag of its parent tag.

Note: Index starts from “1”.

Syntax: childtag:nth-child(n)

Example: p:nth-child(2)

Example on Nth-child selector

```

<html>
    <head>
        <title>CSS - Nth-child</title>
        <style type="text/css">
            p:nth-child(2)
            {
                background-color: #33cc99;
            }
        </style>

```

```

</head>
<body>
  <div>
    <p>para 1</p>
    <p>para 2</p>
    <p>para 3</p>
    <p>para 4</p>
    <p>para 5</p>
  </div>
  <div>
    <p>para 6</p>
    <p>para 7</p>
    <p>para 8</p>
    <p>para 9</p>
    <p>para 10</p>
  </div>
  <div>
    <p>para 11</p>
    <p>para 12</p>
    <p>para 13</p>
    <p>para 14</p>
    <p>para 15</p>
  </div>
</body>
</html>

```

Nth-child(even) selector

- It selects all the even child tags. Ex: 2, 4, 6, ...

Note: Index starts from “1”.

Syntax: childtag:nth-child(even)

Example: p:nth-child(even)

Example on Nth-child(even) selector

```

<html>
  <head>
    <title>CSS - Even</title>
    <style type="text/css">
      /* selects all even <p>. Means 2, 4, 6 */
      p:nth-child(even)
      {
        background-color: #0099cc;
      }
    </style>
  </head>
  <body>
    <div>
      <p>para 1</p>
      <p>para 2</p>
      <p>para 3</p>
    </div>
  </body>
</html>

```

```

<p>para 4</p>
<p>para 5</p>
<p>para 6</p>
<p>para 7</p>
<p>para 8</p>
<p>para 9</p>
<p>para 10</p>
</div>
</body>
</html>

```

Nth-child(odd) selector

- It selects all the odd child tags. Ex: 1, 3, 5, ...

Note: Index starts from “1”.

Syntax: childtag:nth-child(odd)

Example: p:nth-child(odd)

Example on Nth-child(odd) selector

```

<html>
  <head>
    <title>CSS - Odd</title>
    <style type="text/css">
      /* selects all odd <p>. Means 1, 3, 5 */
      p:nth-child(odd)
      {
        background-color: #0099cc;
      }
    </style>
  </head>
  <body>
    <div>
      <p>para 1</p>
      <p>para 2</p>
      <p>para 3</p>
      <p>para 4</p>
      <p>para 5</p>
      <p>para 6</p>
      <p>para 7</p>
      <p>para 8</p>
      <p>para 9</p>
      <p>para 10</p>
    </div>
  </body>
</html>

```

Before selector

- It inserts an image before (at left side of) the current element.
- It is also called as “pseudo element”.

Syntax: tag::before

Example: h1::before

Example on Before selector

```
<html>
  <head>
    <title>CSS - Before</title>
    <style type="text/css">
      /* :: means psuedo element */
      h1::before
      {
        content: url('home.jpg');
      }
    </style>
  </head>
  <body>
    <h1>heading</h1>
    <h1>heading</h1>
    <h1>heading</h1>
    <h1>heading</h1>
  </body>
</html>
```

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

After selector

- It inserts an image after (at right side of) the current element.
- It is also called as “pseudo element”.

Syntax: tag::after

Example: h1::after

Example on After selector

```
<html>
  <head>
    <title>CSS - After</title>
    <style type="text/css">
      /* :: means psuedo element */
      h1::after
      {
        content: url('home.jpg');
      }
    </style>
  </head>
  <body>
    <h1>heading</h1>
    <h1>heading</h1>
    <h1>heading</h1>
```

```

<h1>heading</h1>
<h1>heading</h1>
</body>
</html>

```

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

Selection selector

- It applies the style for the selected text of the web page.
- It is also called as "pseudo element".
- "moz" stands for "Mozilla Firefox".

Syntax for Mozilla Firefox: ::-moz-selection

Syntax for Other Browsers: ::selection

Example on Selection selector

```

<html>
  <head>
    <title>CSS - Selection</title>
    <style type="text/css">
      /* Code for Mozilla Firefox */
      ::-moz-selection
      {
        color: red;
        background-color: yellow;
      }
      /* for all remaining browsers */
      ::selection
      {
        color: red;
        background: yellow;
      }
    </style>
  </head>
  <body>
    <h1>Select some text on this page:</h1>
    <p>This is a paragraph.</p>
    <div>This is some text in a div element.</div>
  </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. Adjacent Sibling Selector
5. Child Selector
6. Class Selector
7. Attribute Selector
8. 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
      {
        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;
    }
    .menubar ul li:hover
    {
        background-color: #33cc99;
        cursor: pointer;
        text-decoration: underline;
    }

```

</style>

</head>

<body>

<div class="menubar">

- Home
- About
- Contact
- Services

</div>

</body>

</html>

Header

- Header is the top content of the web page, which includes website logo, website name, main options for navigation.

```

<html>
  <head>
    <title>Header</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 25px;
      }
      *
      {
        padding: 0px;
      }

```

```
margin: 0px;
}
#header
{
    background-color: #4867AA;
    height: 200px;
}
#leftdiv
{
    /*background-color: #00ccff;*/
    width: 30%;
    float: left;
    height: 200px;
}
#rightdiv
{
    /*background-color: #00cc99;*/
    width: 70%;
    float: left;
    height: 200px;
    color: white;
}
#logoimage
{
    margin-left: 40px;
    margin-top: 20px;
    width: 200px;
}
#rightdiv1
{
    /*background-color: #66ff66;*/
    width: 30%;
    height: 200px;
    float: left;
}
#rightdiv2
{
    /*background-color: #ff6699;*/
    width: 30%;
    height: 200px;
    float: left;
}
#rightdiv3
{
    /*background-color: #cccccc;*/
    width: 30%;
    height: 200px;
    float: left;
}
#cleardiv
{
    clear: left;
```

```
        }
    .part1,.part2,.part3
    {
        margin-top: 10px;
        margin-bottom: 10px;
    }
    #submitbutton
    {
        background-color: #4867AA;
        border-style: ridge;
        color: white;
        padding: 2px;
    }
    #forgotlink
    {
        color: #8CB4C4;
        text-decoration: none;
    }
</style>
</head>
<body>
    <div id="header">
        <div id="leftdiv">
            
        </div>
        <div id="rightdiv">

            <div id="rightdiv1">
                <div class="part1">
                    Email or phone
                </div>
                <div class="part2">
                    <input type="text">
                </div>
                <div class="part3">
                    &nbsp;
                </div>
            </div>

            <div id="rightdiv2">
                <div class="part1">
                    Password
                </div>
                <div class="part2">
                    <input type="password">
                </div>
                <div class="part3">
                    <a href="#" id="forgotlink">Forgotten account?</a>
                </div>
            </div>

            <div id="rightdiv3">
```

```

<div class="part1">
  &nbsp;
</div>
<div class="part2">
  <input type="submit" value="Login" id="submitbutton">
</div>
<div class="part3">
  &nbsp;
</div>
</div>

<div id="cleardiv">
</div>
</div>

</div>
</body>
</html>

```

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

Advanced CSS

Resize

- It is used to make an html element, resizable in the browser.
- **Syntax:** resize: horizontal | vertical | both;
- **Example:** resize: both;

Example on Resize

```

<!DOCTYPE html>
<html>
  <head>
    <title>CSS 3 - Resize</title>
    <style type="text/css">
      #div1
      {
        border: 2px solid;
        padding: 10px 40px;
        width: 500px;
        resize: both; /*Options: none | both | horizontal | vertical */
        overflow: auto;
      }
    </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>
  </body>
</html>

```

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.

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

Word Wrap

- It is used to split the long words into next line.
- Syntax:** word-wrap: break-word;
- Example:** word-wrap: break-word;

Example on Word Wrap

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS 3 - Word Wrap</title>
    <style type="text/css">
        p.test1
        {
            width: 270px;
            border: 1px solid #000000;
        }
        p.test2
        {
            width: 270px;
            border: 1px solid #000000;
            word-wrap: break-word;
        }
    </style>
</head>
<body>
    <p class="test1">This paragraph contains a very long word: thisisaveryveryveryveryverylongword. The long word will break and wrap to the next line.</p>
    <p class="test2">This paragraph contains a very long word: thisisaveryveryveryveryverylongword. The long word will break and wrap to the next line.</p>
</body>
</html>
```

RGBA

- It is used to apply transparency to the background color only, instead of applying transparency for the content.
- Syntax:** rgba(red, green, blue, alpha)

Red = 0 to 255

Green = 0 to 255
Blue = 0 to 255
Alpha = 0 to 1

Example on RGBA

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS 3 - RGBA</title>
    <style type="text/css">
        body
        {
            background-image: url('sample.png');
        }

        #div1
        {
            border: 2px solid #a1a1a1;
            padding: 10px 40px;
            width: 300px;
            background-color: rgba(50, 180, 110, 0.6); /*Syntax: rgba(red, green, blue, opacity) */
        }
    </style>
</head>
<body>
    <div id="div1">Hello how are you!!</div>
</body>
</html>
```

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

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 #a1a1a1;
            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 #a1a1a1;
```

```

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>
```

Multiple Columns

- It is used to divide the text of the element into multiple columns.

- **Set no. of columns:**

column-count: no. of columns;
-moz-column-count: no. of columns;
-webkit-column-count: no. of columns;

- **Set column gap – distance between columns:**

Column-gap: pixels;
-moz-column-gap: pixels;
-webkit-column-gap: pixels;

- **Set column rule – divider line between the columns:**

column-rule: width style color;
-moz-column-rule: width style color;
-webkit-column-rule: width style color;

-moz- : Mozilla Firefox

-webkit- : Chrome and Safari

Example on Multiple Columns

```
<!DOCTYPE html>
<html>
<head>
<title>CSS 3 - Multiple Columns</title>
<style type="text/css">
.newspaper
{
    text-align: justify;
    border: 1px solid gray;
```

```

    column-count: 4;
    -moz-column-count: 4;
    -webkit-column-count: 4;
    column-gap: 40px;
    -moz-column-gap: 40px;
    -webkit-column-gap: 40px;
}
</style>
</head>
<body>
    <div class="newspaper">Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32. The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English versions from the 1914 translation by H. Rackham.</div>
</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:

font-size



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;  
        }  
    </style>
```

```
        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 specified 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;
        }
    </style>
</head>
<body>
    <div id="div1"></div>
</body>
</html>

```

```

        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>

```

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>
```

Example on Image Gallery

```
<html>
<head>
    <title>CSS 3 - Transformations - Image Gallery</title>
    <style type="text/css">
        .photogallery
        {
            width: 200px;
            height: 190px;
            background-color: #00CCFF;
            margin: 50px;
            float: left;
            cursor: pointer;
            box-shadow: 10px 10px 10px 10px #ff0000;
            transition: transform 0.8s;
        }
        .photogallery:hover
        {
            background-color: #0099FF;
            transform: rotate(360deg) scale(1.2);
        }
        .photogallery img
        {
            width: 100%;
        }
    </style>
</head>
<body>
```

```

<div class="photogallery">
  
  <span>image 1</span>
</div>

<div class="photogallery">
  
  <span>image 2</span>
</div>

<div class="photogallery">
  
  <span>image 3</span>
</div>

<div class="photogallery">
  
  <span>image 4</span>
</div>

<div class="photogallery">
  
  <span>image 5</span>
</div>

<div class="photogallery">
  
  <span>image 6</span>
</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



10px 50px 20px 100px 15px 40px

10s

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>
```

Gradient Colors

- It is used to apply multiple colors to the element.
- Go to <http://www.colorzilla.com/gradient-editor>
- Generate the colors.
- Copy and paste the code into the program.

Example on Gradient Colors

```
<html>
<head>
    <title>CSS 3 - Gradient Colors</title>
    <style type="text/css">
        #div1
        {
            width: 300px;
            height: 300px;
            background: #03bc09;
            background: -moz-linear-gradient(top, #03bc09 0%, #477c02 38%, #073f00 100%);
            background: -webkit-linear-gradient(top, #03bc09 0%,#477c02 38%,#073f00 100%);
            background: linear-gradient(to bottom, #03bc09 0%,#477c02 38%,#073f00 100%);
            filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#03bc09',
endColorstr='#073f00',GradientType=0);
        }
    </style>
</head>
<body>
    <div id="div1">div 1</div>
</body>
</html>
```

Font Face

- It is used to apply third party fonts in the web page.
- **Steps:**
 - Go to "<https://www.fontsquirrel.com>".
 - Click on "Web Font" in "Licenses".
 - Click on the desired font.
 - Click on "Webfont Kit".
 - Select all the checkboxes (TTF, EOT, WOFF, SVG).
 - TTF: Works in most browsers except IE and iPhone
 - EOT: IE only
 - WOFF: Compressed, emerging standard
 - SVG: iPhone/iPad.

- Click on “Download @Fontface Kit”.
- Download the font zip file.
- After downloading, extract the zip file. It creates a folder and extracts the files into it. Go to the extracted folder.
- Copy and paste the following files from the “extracted folder” into “application folder (c:\css)”.
- Copy and paste the following files from the “extracted folder” into “application folder (c:\css)”.

Extracted folder:

- fontname.eot
- fontname.svg
- fontname.ttf
- fontname.woff
- fontname.css

Application folder (c:\css):

- fontname.eot
- fontname.svg
- fontname.ttf
- fontname.woff
- fontname.css
- filename.html

Step 1: Import the css file into the web page:

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

Step 2: Create @font-face rule:

```
@font-face
{
    font-family: "fontname";
    src: url("filename.eot");
    src: url("filename.eot?#iefix") format("embedded-opentype"),
         url("filename.woff") format("woff"),
         url("filename.ttf") format("truetype"),
         url("filename.svg@fontname") format("svg");
```

```
font-weight: normal;  
font-style: normal;  
}
```

Step 3: Set the font-family:

```
selector  
{  
    font-family: "fontname";  
}
```

Example on Font Face

c:\css\Stylesheet.css

```
@font-face  
{  
    font-family: 'amadeus_regulararamadeusRg';  
    src: url('Amadeus-webfont.eot');  
    src: url('Amadeus-webfont.eot?#iefix') format('embedded-opentype'),  
        url('Amadeus-webfont.woff') format('woff'),  
        url('Amadeus-webfont.ttf') format('truetype'),  
        url('Amadeus-webfont.svg#amadeus_regulararamadeusRg') format('svg');  
    font-weight: normal;  
    font-style: normal;  
}
```

c:\css\Fontexample.html

```
<html>  
<head>  
    <title>Font example</title>  
    <link href="stylesheet.css" rel="stylesheet">  
    <style type="text/css">  
        #div1  
        {  
            font-family: amadeus_regulararamadeusRg;  
            font-size: 40px;  
        }  
    </style>  
</head>  
<body>  
    <div id="div1">Hello</div>  
</body>  
</html>
```

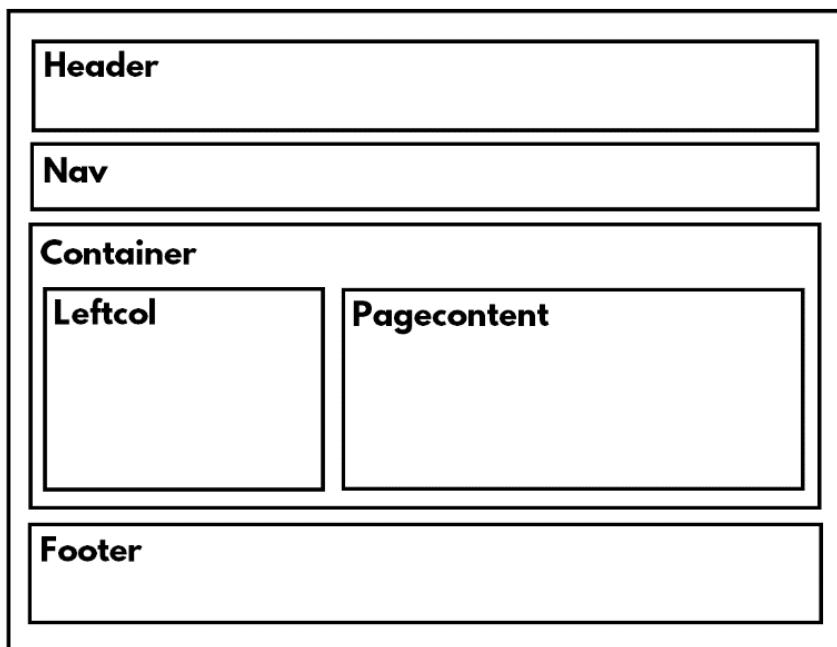
Note: Place “Amadeus-webfont.eot”, “Amadeus-webfont.svg”, “Amadeus-webfont.ttf”, “Amadeus-webfont.woff” in the current folder (c:\css).

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>
            </div>
        </div>
    </body>
</html>

```

```
<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>
```

```
</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.
- We divide the devices into 4 types, based on the browser width:
 1. **Large devices:** 1200px to unlimited
 2. **Medium devices:** 1024px to 1199px
 3. **Small devices:** 768px to 1023px
 4. **Extra Small devices:** 300px to 767px

Media Queries

- We use “Media Queries” to create responsive web design. The media queries apply the styles based on the specified resolution.

1. Large devices (1200px to unlimited):

```
@media (min-width: 1200px){  
}
```

2. Medium devices (1024px to 1199px):

```
@media (min-width: 1024px) and (max-width: 1199px){  
}
```

3. Small devices (68px to 1023px):

```
@media (min-width: 768px) and (max-width: 1023px){  
}
```

4. Extra Small devices (300px to 767px):

@media (min-width: 300px) and (max-width: 767px)

```
{  
}
```

View port meta tag

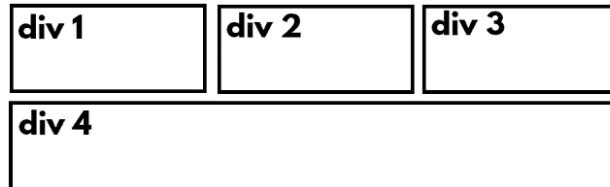
- This tag tells to the mobile browsers that we are using “responsive web design”, and don’t treat it as pc-based web page.
- Without viewport meta tag, the mobile browsers treat the web page as pc-based web page and apply the pc-based media query.
- With viewport meta tag, the mobile browsers apply the appropriate “mobile-based media query” to the web page.

Responsive Web Design - Example

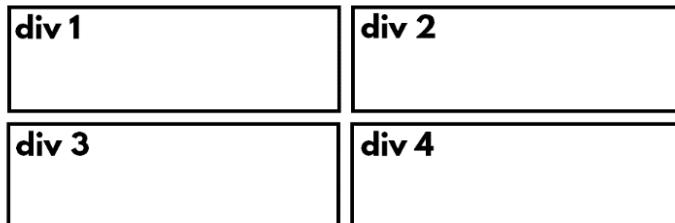
Large Devices



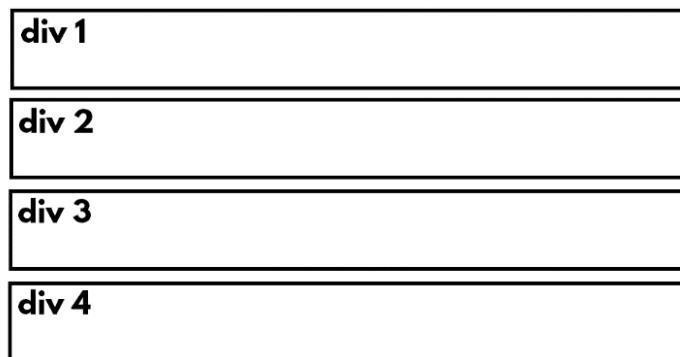
Medium Devices



Small Devices



Extra Small Devices



Example on Responsive Web Design

home.html

```
<html>
  <head>
    <title>Home page</title>
    <link href="StyleSheet.css" rel="stylesheet">
```

```
<meta name="viewport" content="width=device-width">
</head>
<body>

    <!-- outercontainer starts -->
    <div id="outercontainer">

        <!-- header starts -->
        <div id="header">
            header content here
        </div>
        <!-- header ends -->

        <!-- nav starts -->
        <div id="nav">
            nav content here
        </div>
        <!-- nav ends -->

        <!-- container starts -->
        <div id="container">

            <!-- leftcol starts -->
            <div id="leftcol">
                leftcol content here
            </div>
            <!-- leftcol ends -->

            <!-- pagecontent starts -->
            <div id="pagecontent">
                <h2>page content here</h2>
                <div id="div1">div 1</div>
                <div id="div2">div 2</div>
                <div id="div3">div 3</div>
                <div id="div4">div 4</div>
            </div>
            <!-- pagecontent ends -->

        </div>
        <!-- container ends -->

        <!-- footer starts -->
        <div id="footer">
            footer content here
        </div>
        <!-- footer ends -->

    </div>
    <!-- outercontainer ends -->

</body>
</html>
```

StyleSheet.css

```
@media (min-width: 1200px)
{
    body,input
    {
        font-family: Tahoma;
        font-size: 28px;
    }

    *
    {
        margin: 0px;
        padding: 0px;
    }

    #outercontainer
    {
        background-color: #00ffcc;
        width: 98%;
        margin: auto;
    }

    #header
    {
        background-color: #0066cc;
        height: 200px;
    }

    #nav
    {
        background-color: #33ccff;
        height: 120px;
    }

    #container
    {
        background-color: #ccffcc;
    }

    #leftcol
    {
        background-color: #ffcccc;
        height: 700px;
        width: 30%;
        float: left;
    }

    #pagecontent
    {
        background-color: #99ccff;
```

```
height: 700px;
width: 70%;
float: left;
}

#footer
{
background-color: #ff0099;
height: 100px;
clear: left;
}

#div1
{
background-color: #ccffcc;
width: 25%;
height: 200px;
float: left;
}

#div2
{
background-color: #cccc99;
width: 25%;
height: 200px;
float: left;
}

#div3
{
background-color: #00ff33;
width: 25%;
height: 200px;
float: left;
}

#div4
{
background-color: #ffff99;
width: 25%;
height: 200px;
float: left;
}

}

@media (min-width: 1024px) and (max-width: 1199px)
{
body, input
{
font-family: Tahoma;
```

```
    font-size: 28px;  
}  
  
*  
{  
    margin: 0px;  
    padding: 0px;  
}  
  
#outercontainer  
{  
    background-color: #00ffcc;  
    width: 98%;  
    margin: auto;  
}  
  
#header  
{  
    background-color: #0066cc;  
    height: 200px;  
}  
  
#nav  
{  
    background-color: #33ccff;  
    height: 120px;  
}  
  
#container  
{  
    background-color: #ccffcc;  
}  
  
#leftcol  
{  
    background-color: #ffcccc;  
    height: 700px;  
    width: 30%;  
    float: left;  
}  
  
#pagecontent  
{  
    background-color: #99ccff;  
    height: 700px;  
    width: 70%;  
    float: left;  
}  
  
#footer  
{  
    background-color: #ff0099;
```

```
height: 100px;
clear: left;
}

#div1
{
background-color: #ccffcc;
width: 33%;
height: 200px;
float: left;
}

#div2
{
background-color: #cccc99;
width: 33%;
height: 200px;
float: left;
}

#div3
{
background-color: #00ff33;
width: 34%;
height: 200px;
float: left;
}

#div4
{
background-color: #ffff99;
width: 100%;
height: 200px;
clear: left;
}

}

@media (min-width: 768px) and (max-width: 1023px)
{
body, input
{
font-family: Tahoma;
font-size: 28px;
}

*
{
margin: 0px;
padding: 0px;
}
```

```
}

#outercontainer
{
    background-color: #00ffcc;
    width: 98%;
    margin: auto;
}

#header
{
    background-color: #0066cc;
    height: 200px;
}

#nav
{
    background-color: #33ccff;
    height: 120px;
}

#container
{
    background-color: #ccffcc;
}

#leftcol
{
    background-color: #ffcccc;
    height: 150px;
    width: 100%;
}

#pagecontent
{
    background-color: #99ccff;
    height: 500px;
    width: 100%;
}

#footer
{
    background-color: #ff0099;
    height: 100px;
    clear: left;
}

#div1
{
    background-color: #ccffcc;
    width: 50%;
    height: 200px;
```

```
        float: left;
    }

#div2
{
    background-color: #cccc99;
    width: 50%;
    height: 200px;
    float: left;
}

#div3
{
    background-color: #00ff33;
    width: 50%;
    height: 200px;
    float: left;
    clear: left;
}

#div4
{
    background-color: #ffff99;
    width: 50%;
    height: 200px;
    float: left;
}

}

@media (min-width: 300px) and (max-width: 767px)
{
    body, input
    {
        font-family: Tahoma;
        font-size: 28px;
    }

    *
    {
        margin: 0px;
        padding: 0px;
    }

    #outercontainer
    {
        background-color: #00ffcc;
        width: 98%;
        margin: auto;
    }
}
```

```
#header
{
    background-color: #0066cc;
    height: 200px;
}

#nav
{
    background-color: #33ccff;
    height: 120px;
}

#container
{
    background-color: #ccffcc;
}

#leftcol
{
    background-color: #ffcccc;
    height: 150px;
    width: 100%;
}

#pagecontent
{
    background-color: #99ccff;
    height: 850px;
    width: 100%;
}

#footer
{
    background-color: #ff0099;
    height: 100px;
    clear: left;
}

#div1
{
    background-color: #ccffcc;
    width: 100%;
    height: 200px;
}

#div2
{
    background-color: #cccc99;
    width: 100%;
    height: 200px;
}

#div3
{
    background-color: #00ff33;
    width: 100%;
    height: 200px;
```

```
}
```

```
#div4
```

```
{
```

```
background-color: #ffff99;
```

```
width: 100%;
```

```
height: 200px;
```

```
}
```

```
}
```

HARSHA

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>
```

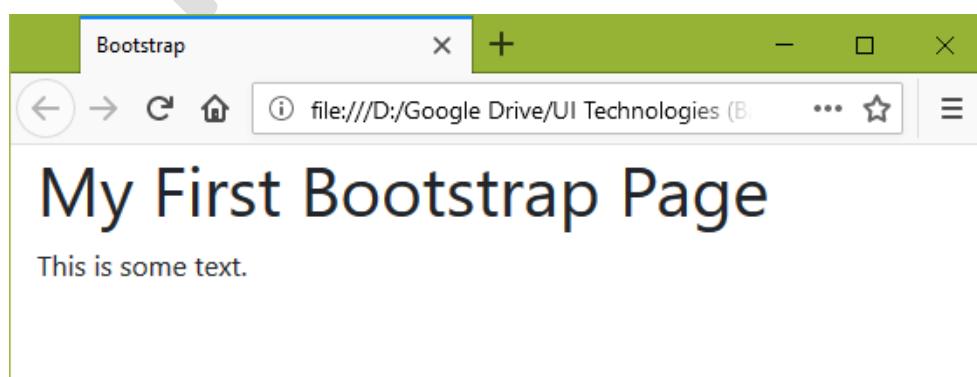
Importing Bootstrap

```
<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

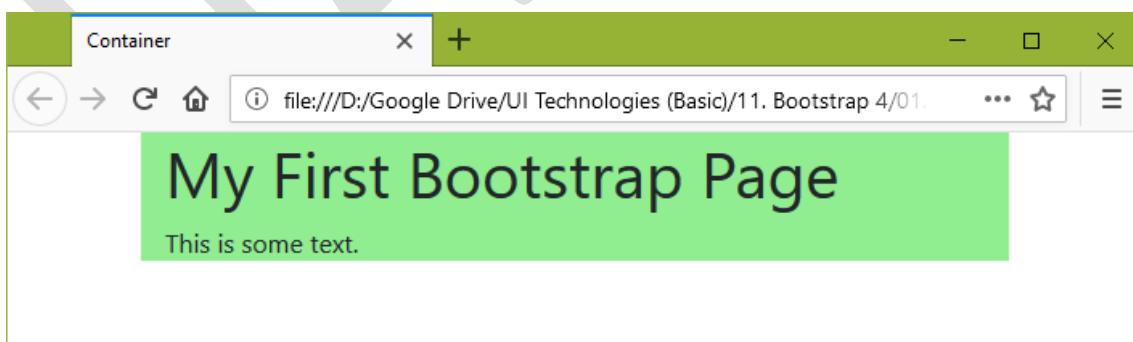
- It acts as “outer container” for the entire page.
- It makes the web page responsive.
- The entire content of the page should be inside the “container”.
- It provides margin left and margin right for the page.

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”.

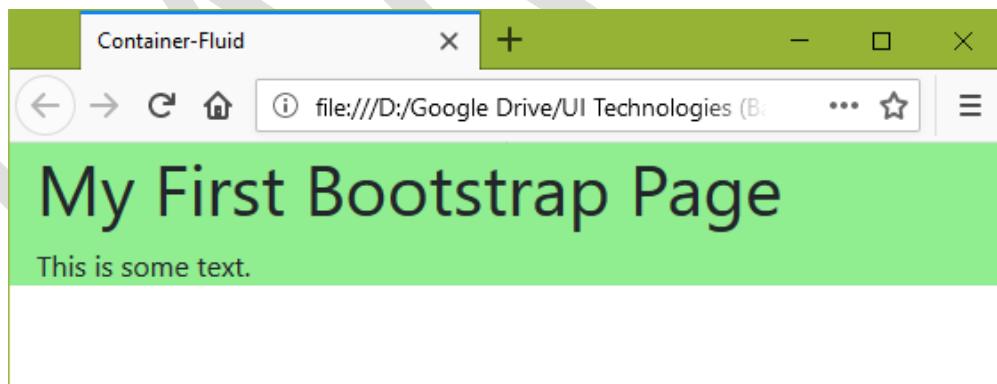
- It removes margin left and margin right for the page. It makes the content occupy the full available width of the web page.

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

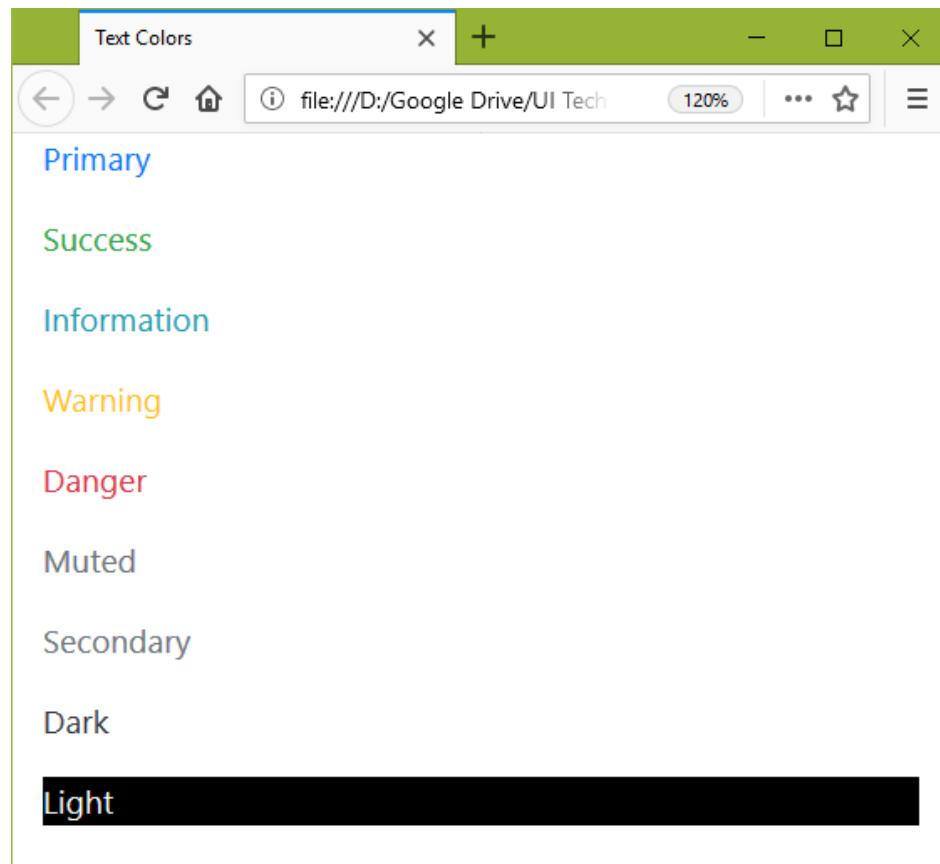
- It is used to set colors of the text.
- Based on the requirement, the developer can use any of the available colors.
- If you want other color, you can use CSS.

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.

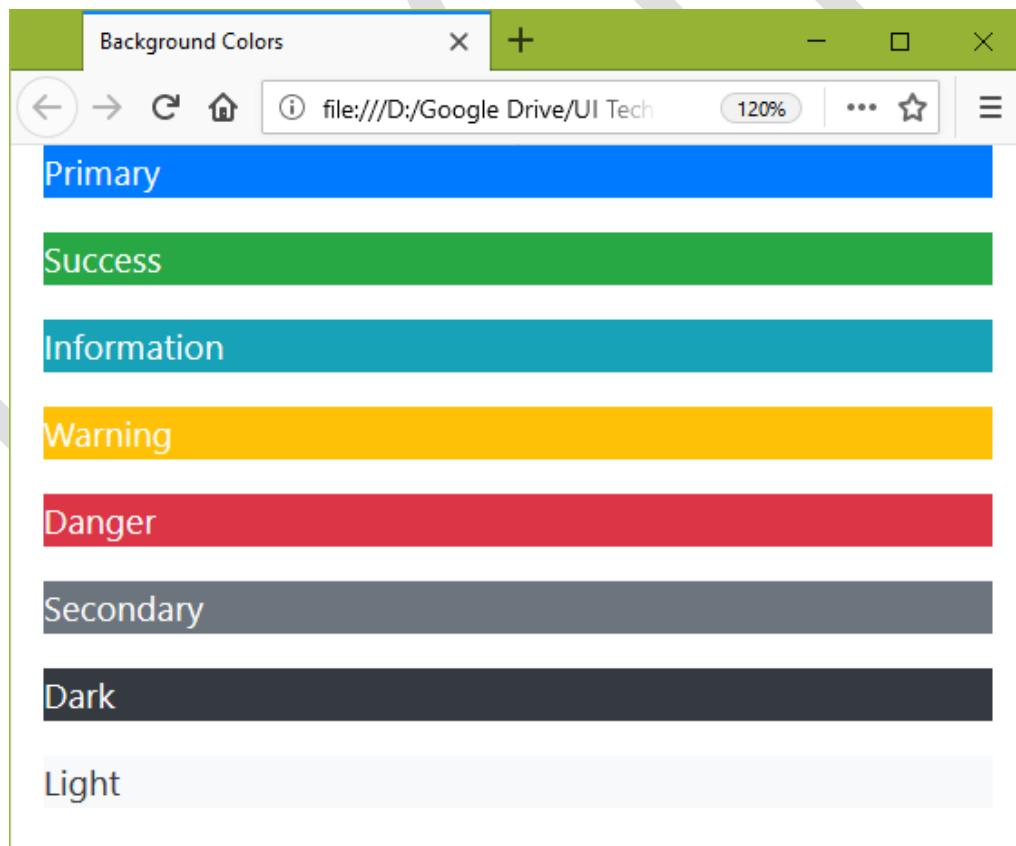
List of Classes

- 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

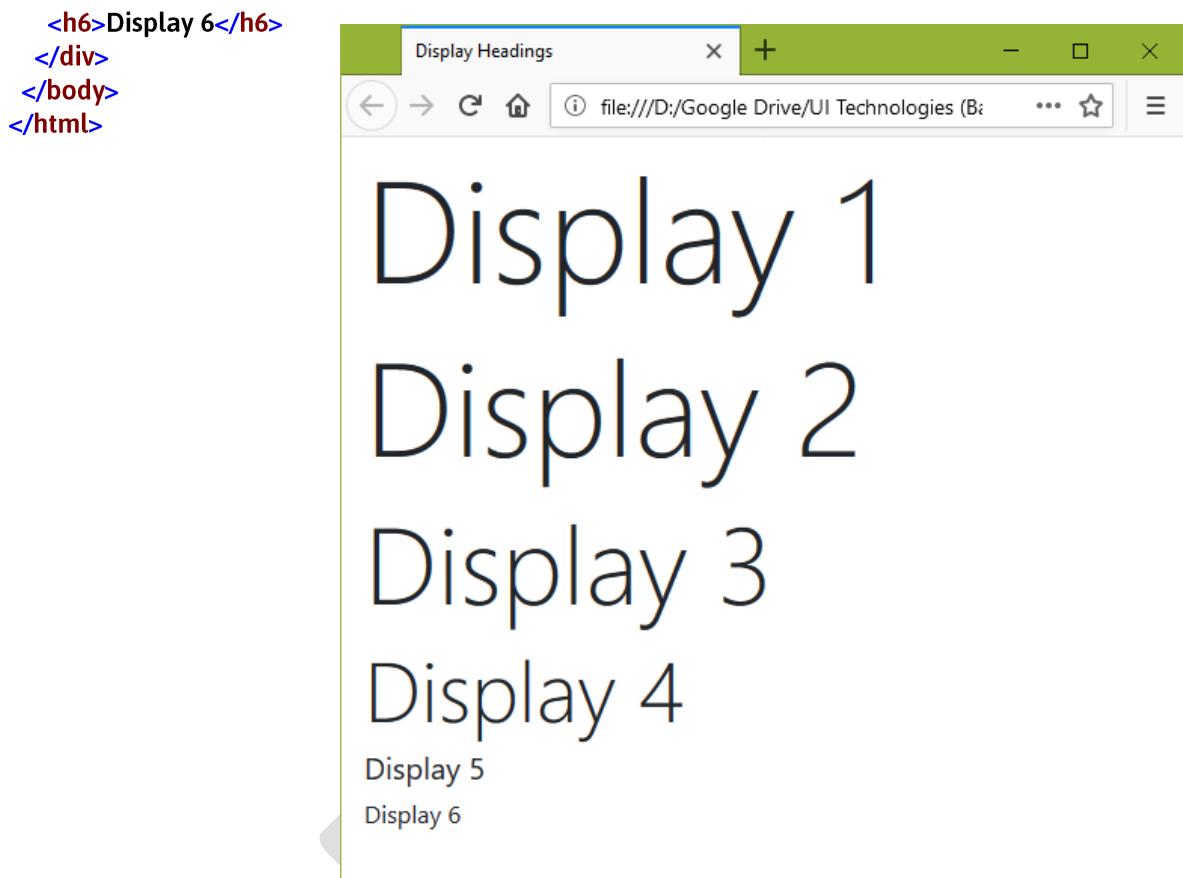
- It is used to display headings with thin text and larger font size.

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>
```



Text Alignment

- We can apply text alignment, by using the following bootstrap css classes.
- Default is "left alignment".

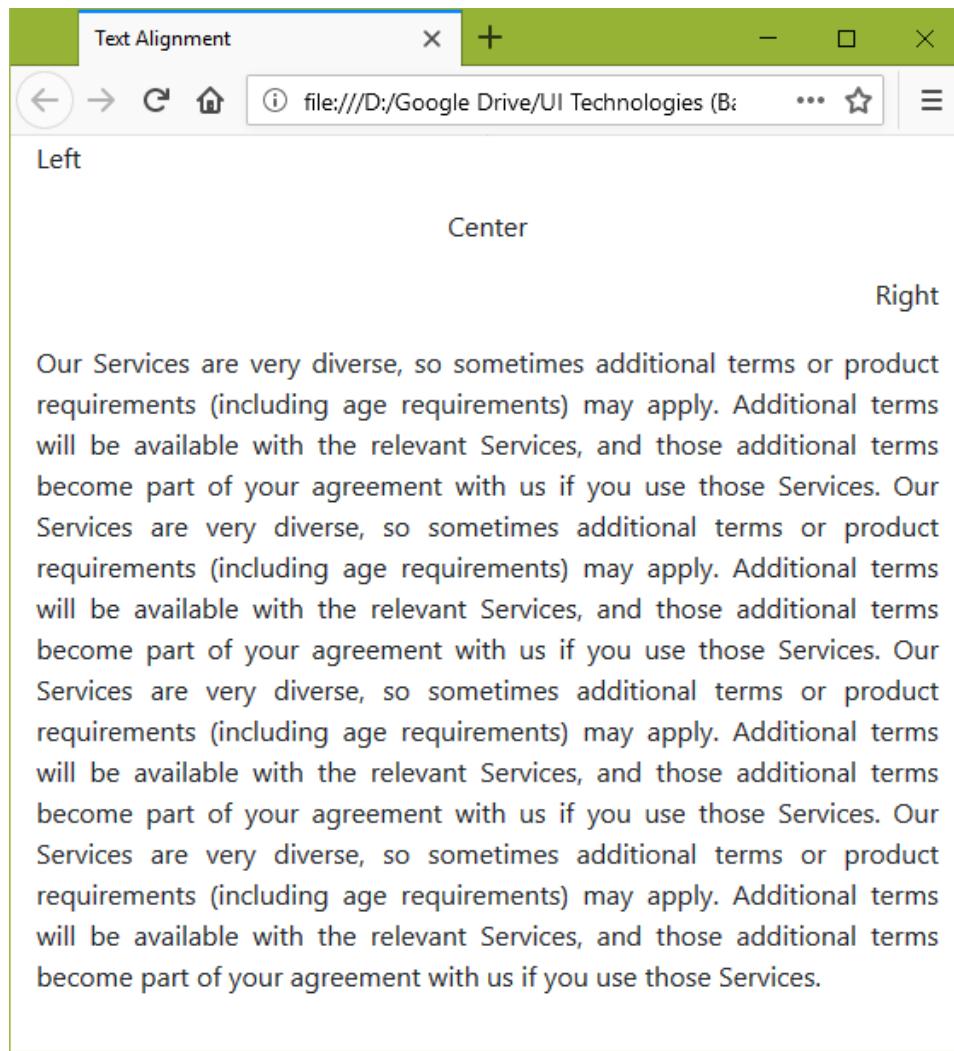
List of Classes

- 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 : Capitalize

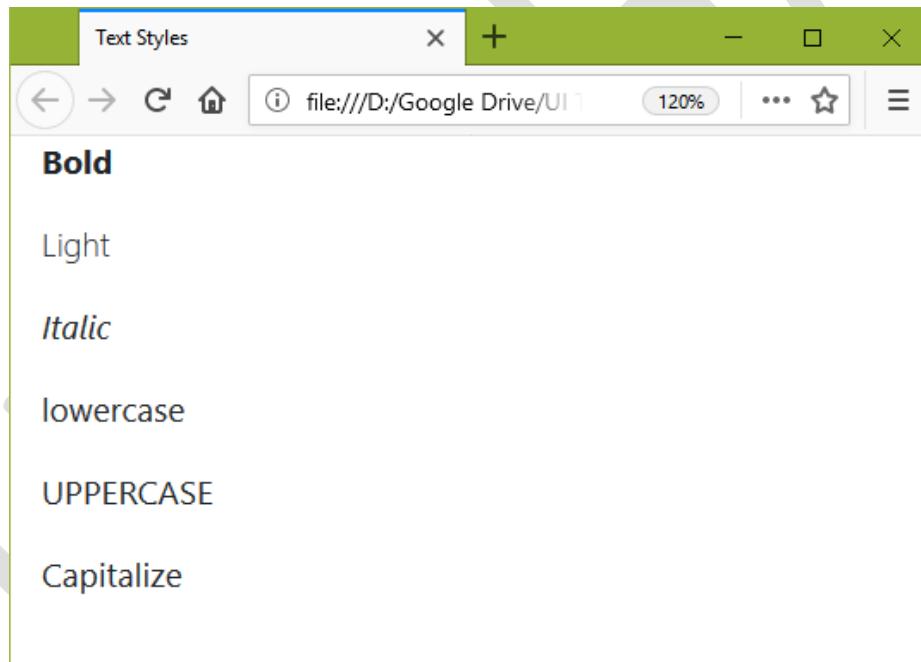
Example on Text Styles

```
<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

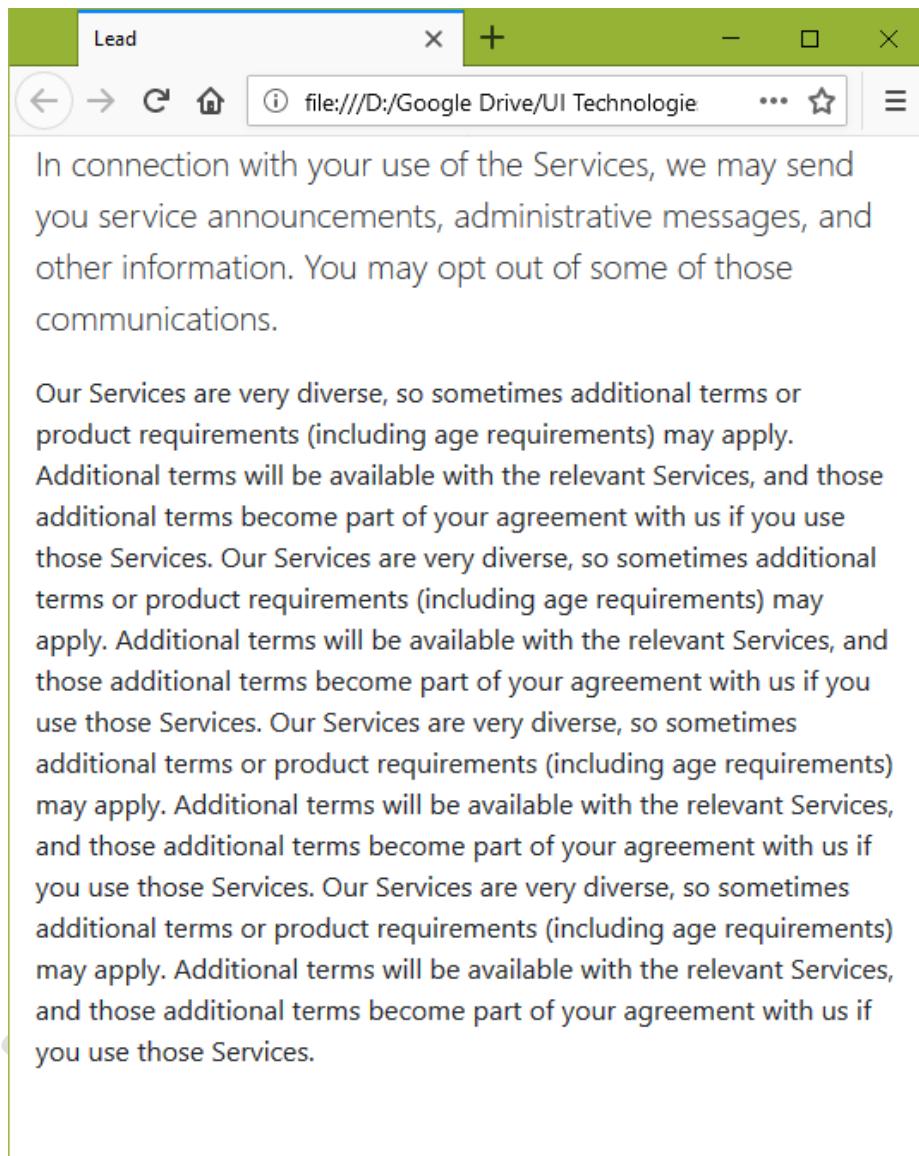
Example on Lead

```

<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. 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>
```



Visibility

- It is used to show / hide the element in the web page.

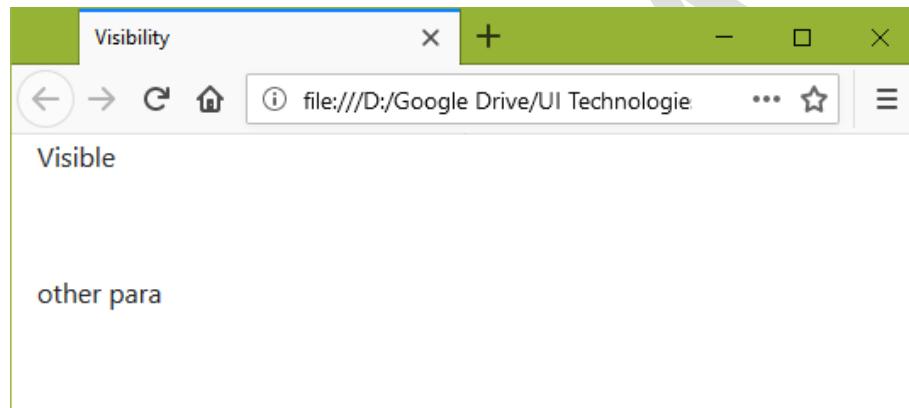
List of Classes

- 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

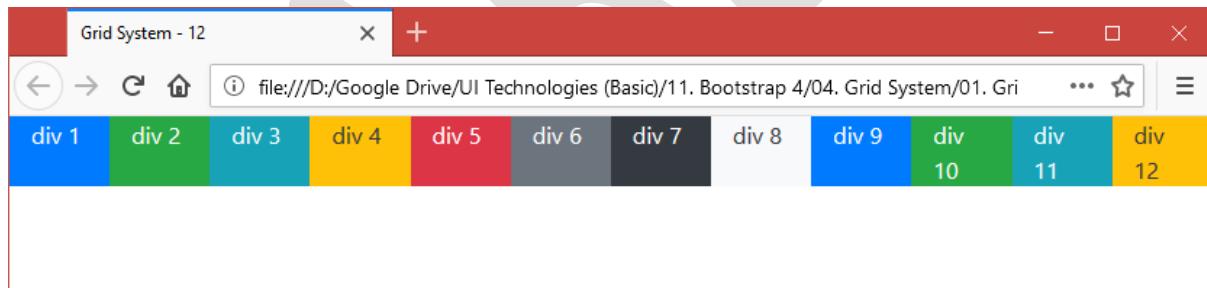
Example on Grid System - 12

```
<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>

```



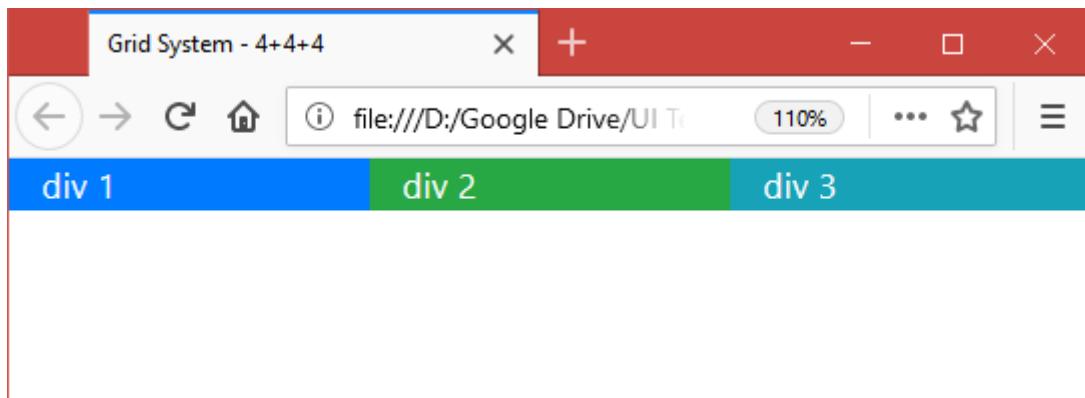
Example on Grid System - 4+4+4

```

<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">

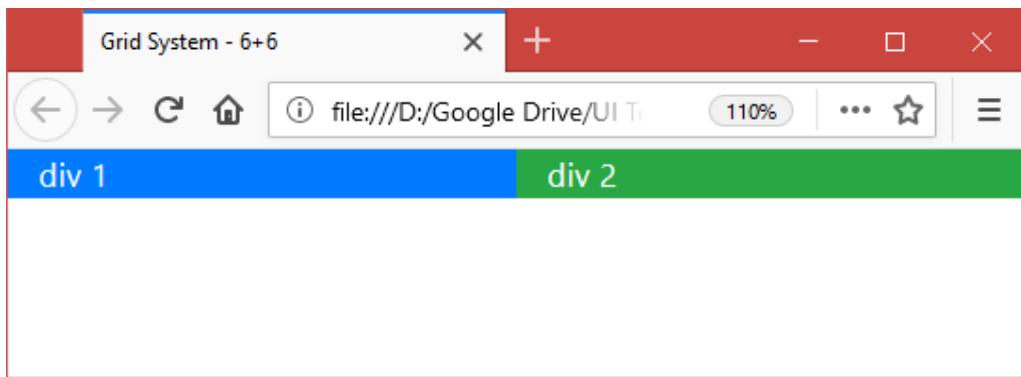
```

```
<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>
```



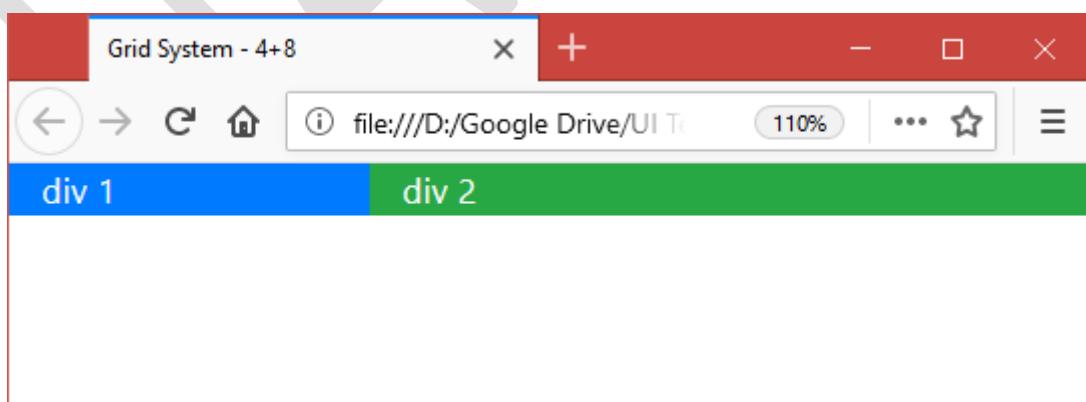
Example on Grid System - 6+6

```
<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>
```



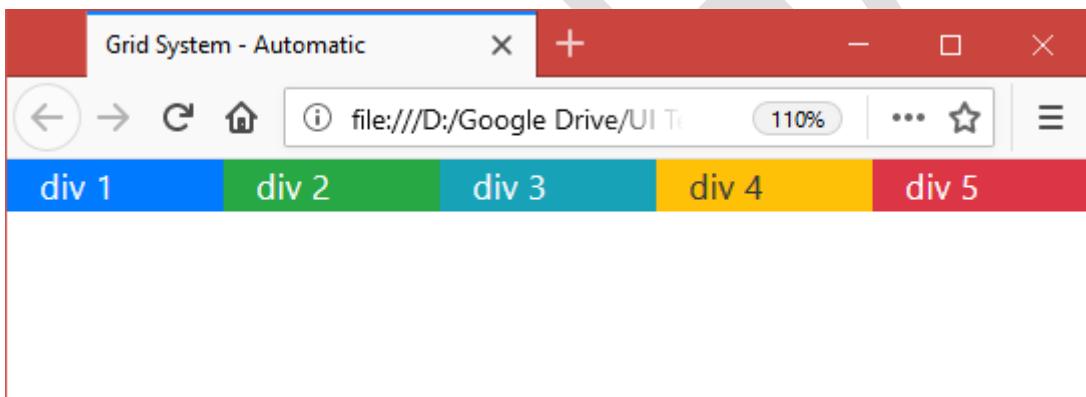
Example on Grid System - Automatic

```
<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 / 1334px 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)

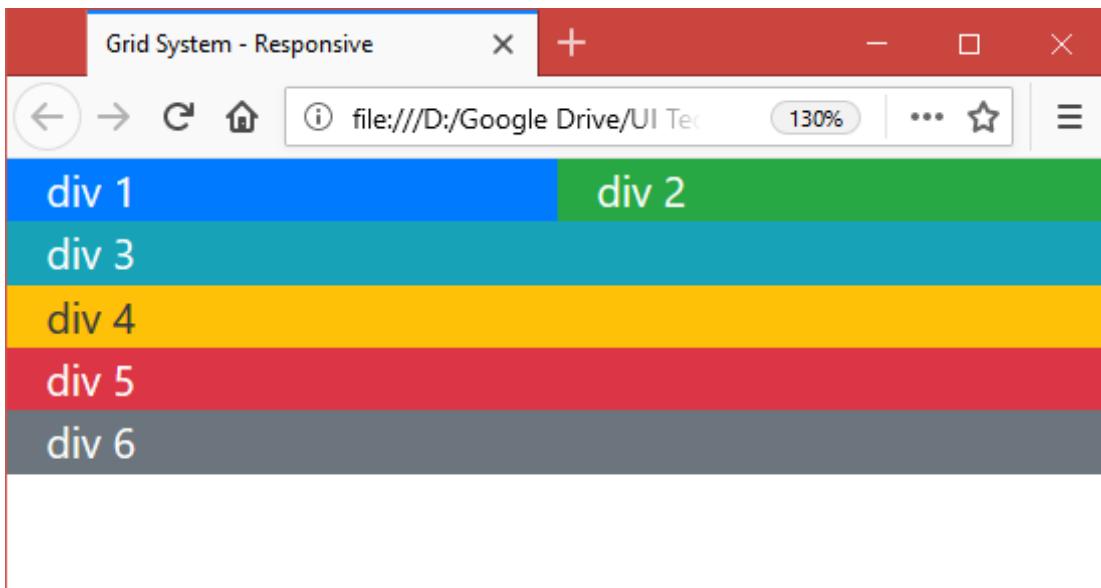
```
<div class="col-lg-n">...</div>
```

```
<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>
</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">div 1</div>
      <div class="col-xl-2 col-lg-3 col-md-4 col-sm-6 col-6 bg-success text-white">div 2</div>
      <div class="col-xl-2 col-lg-3 col-md-4 col-sm-6 col-12 bg-info text-white">div 3</div>
      <div class="col-xl-2 col-lg-3 col-md-4 col-sm-6 col-12 bg-warning text-dark">div 4</div>
      <div class="col-xl-2 col-lg-6 col-md-4 col-sm-6 col-12 bg-danger text-white">div 5</div>
      <div class="col-xl-2 col-lg-6 col-md-4 col-sm-6 col-12 bg-secondary text-white">div 6</div>
    </div>
  </div>
</body>
</html>
```



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>
```



Jumbotron

Jumbotron

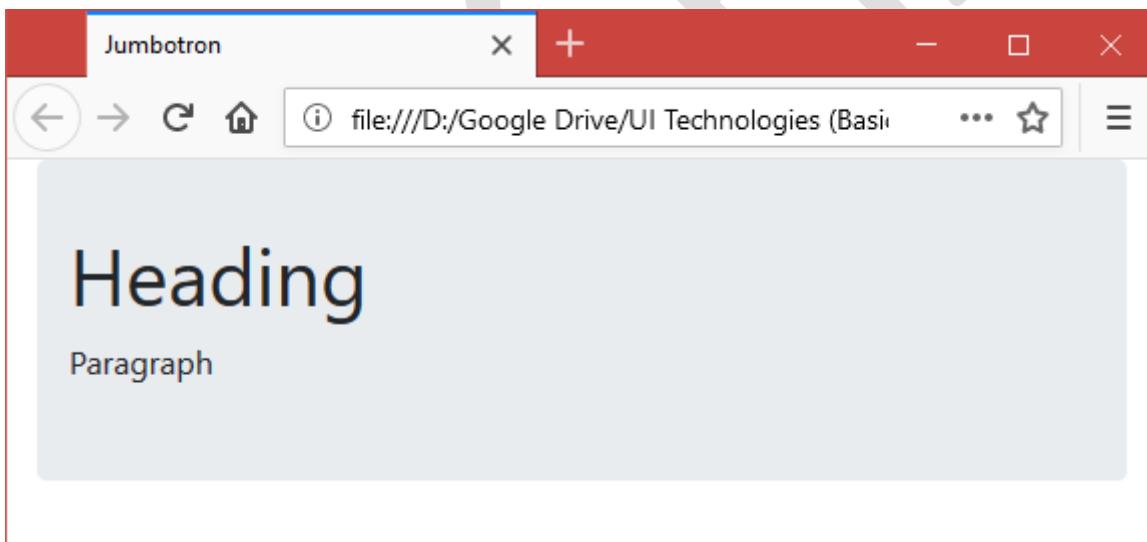
- It is used to display heading and paragraph in large size with a special box to highlight its content.

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>
```

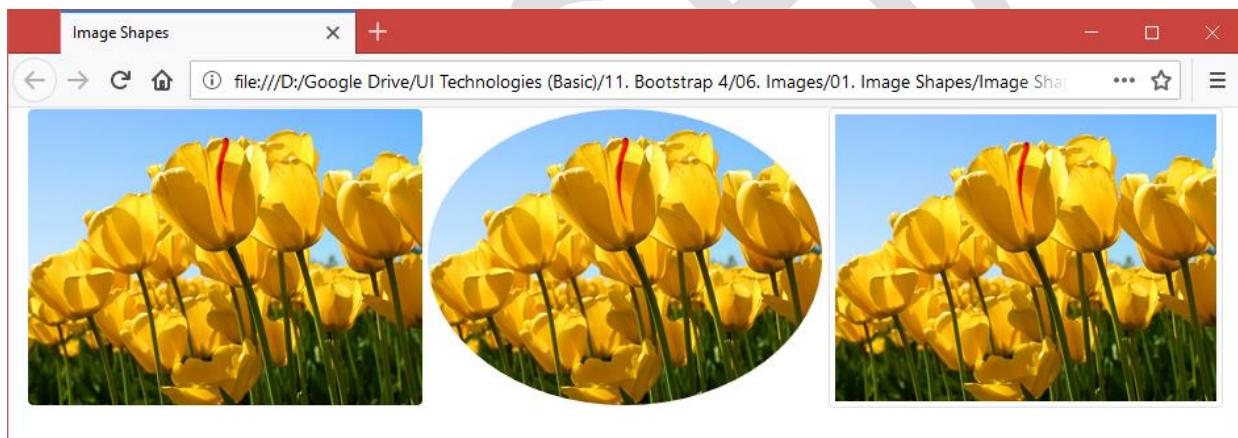


Image Alignment

- 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.

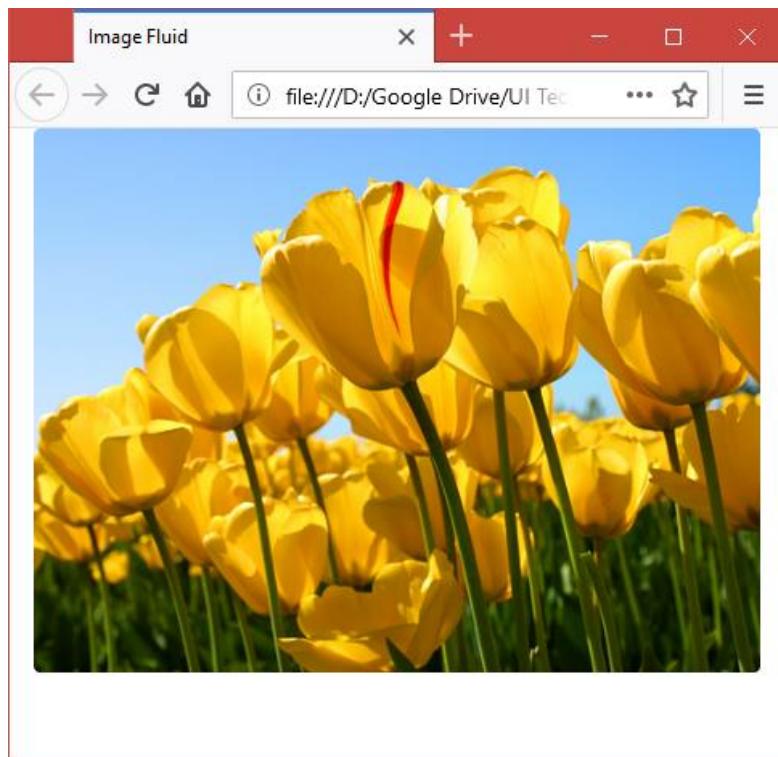
List of Classes

- 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

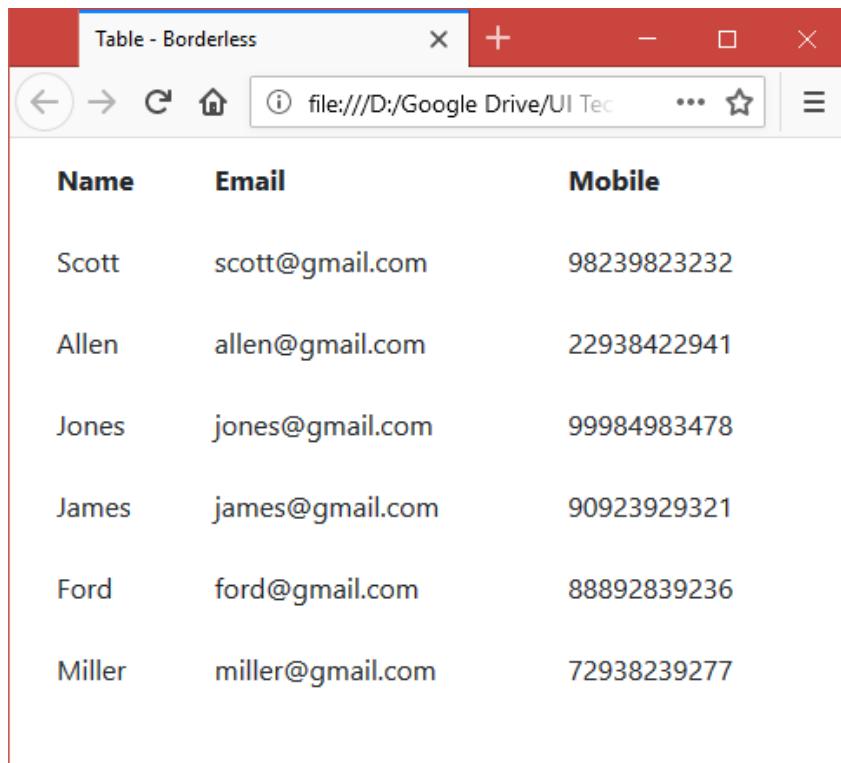
Example on Borderless Table

```

<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>

```

```
</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>
```



The screenshot shows a web browser window titled "Table - Borderless". The address bar indicates the file is located at "file:///D:/Google Drive/UI Tec". The table has a red border around the entire structure. It contains six rows of data with three columns each: "Name", "Email", and "Mobile".

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

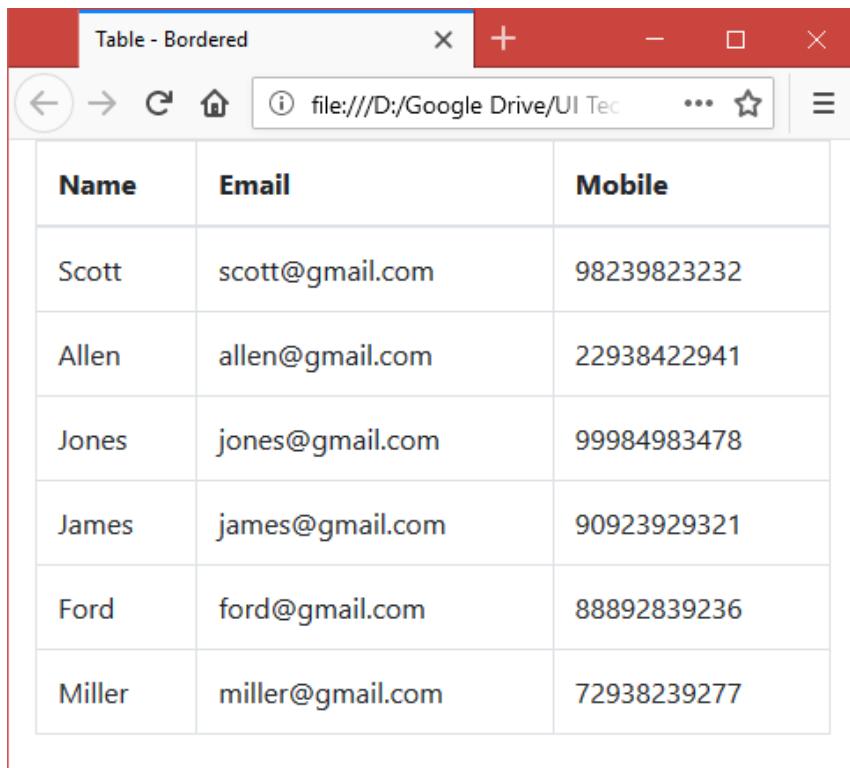
Example on Bordered Table

```

<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>

```

```
</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

Striped Table

- It is used to display table with alternate row background

List of Classes

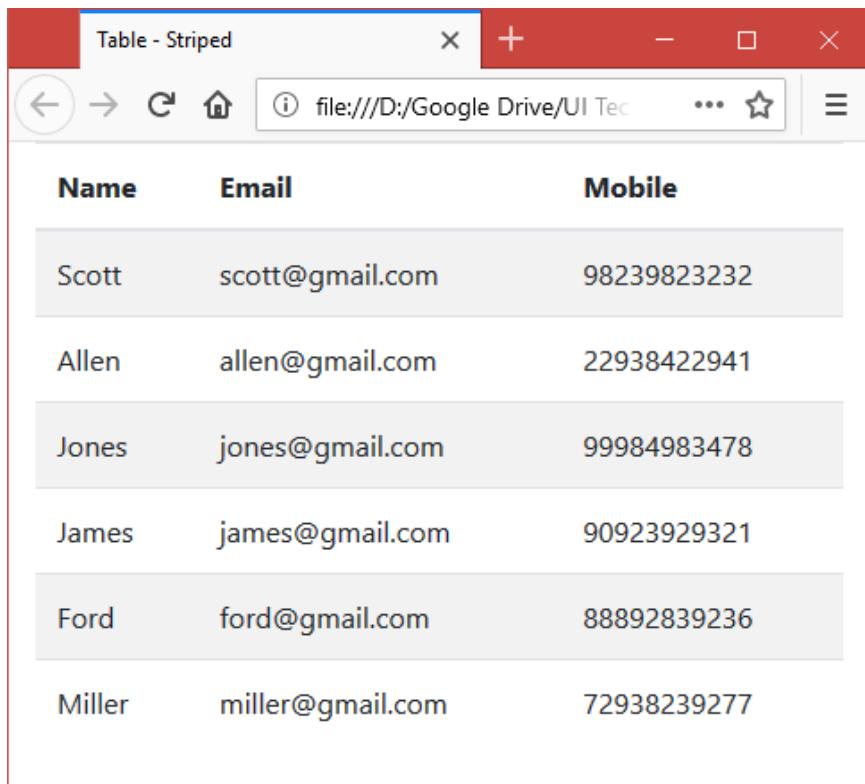
- table-striped : Table with alternate row background.

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 table titled "Table - Striped" in a browser window. The table has three columns: "Name", "Email", and "Mobile". The rows alternate in background color, with the first row being white and the subsequent rows having a light gray background. The data in the table is as follows:

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.

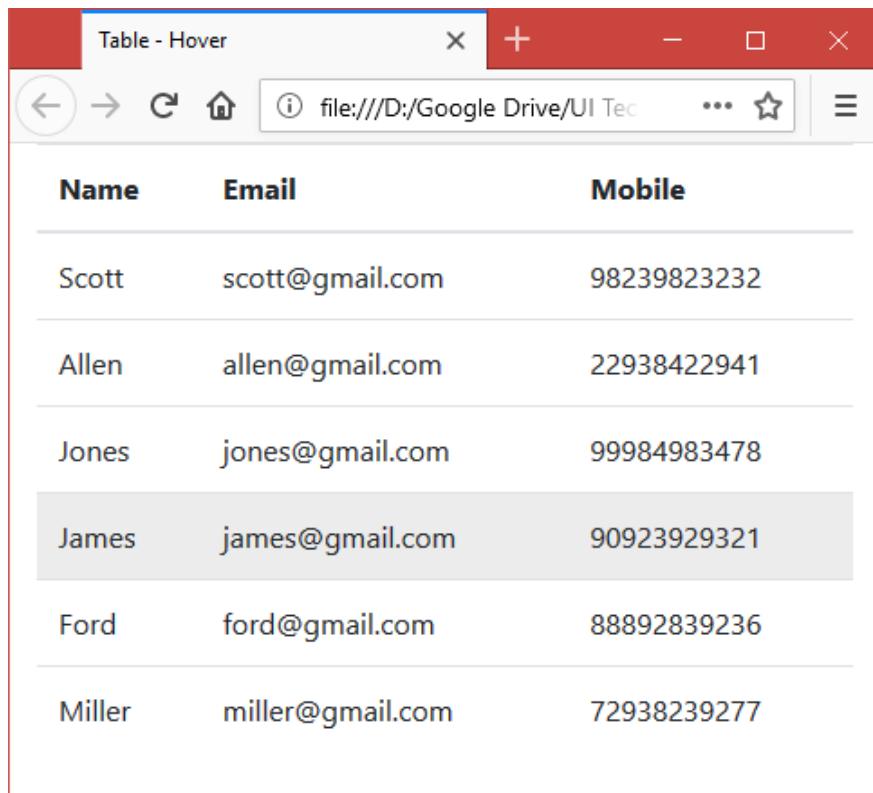
List of Classes

- 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>
<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>
```



The screenshot shows a table with a red header row and a light grey footer row. The main body rows have white backgrounds. The table has three columns: Name, Email, and Mobile. The data is as follows:

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 : Grey color
- table-secondary : Different 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">

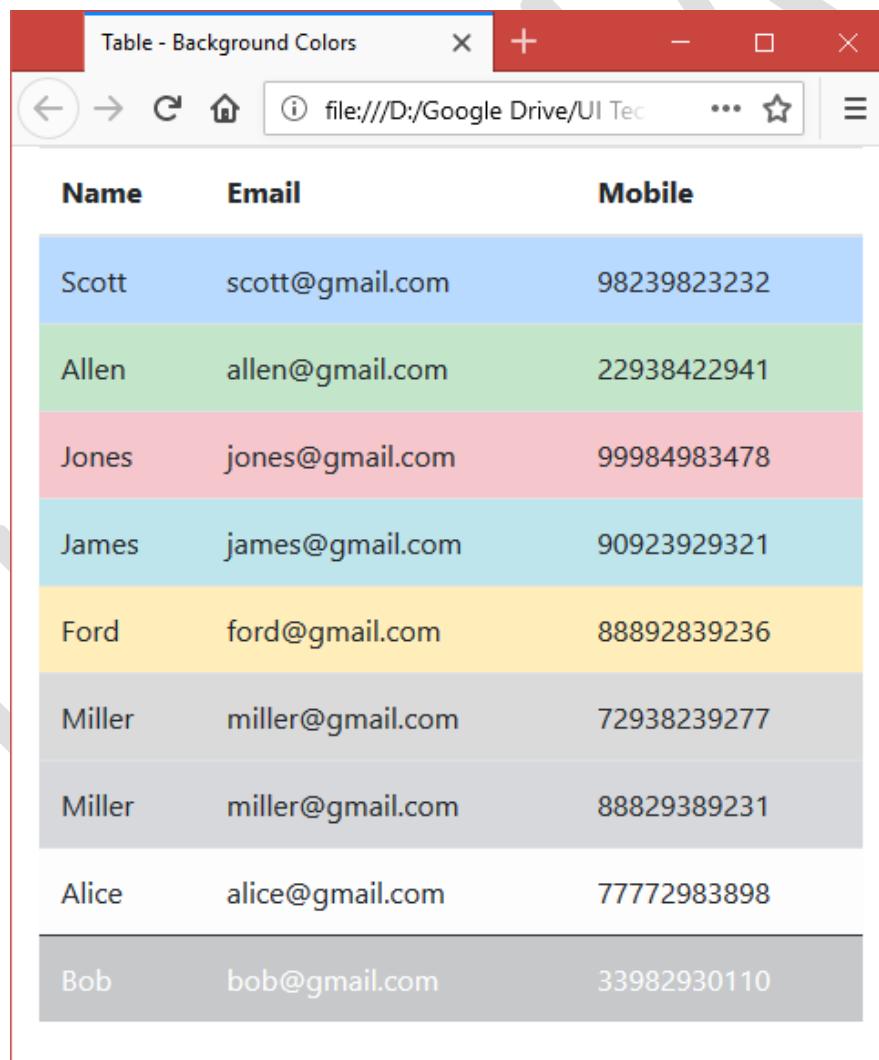
```

```
<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>

```



The screenshot shows a table with 9 rows. The rows alternate in background color: light blue, light green, light red, light teal, yellow, grey, grey, light grey, and dark grey. The columns are labeled "Name", "Email", and "Mobile". The data is as follows:

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>
  <table class="table">
    <thead class="thead-light">
      <tr>
        <th>Name</th>
        <th>Email</th>
        <th>Mobile</th>
      </tr>
```

```
</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>
```

The screenshot shows a web browser window titled "Table - Header Colors". The address bar indicates the file is located at "file:///D:/Google Drive/UI Tec". The browser interface includes standard controls like back, forward, and search.

The page displays two tables side-by-side. Both tables have three columns: "Name", "Email", and "Mobile".

Top Table (Dark Header):

Name	Email	Mobile
Scott	scott@gmail.com	98239823232
Allen	allen@gmail.com	22938422941
Jones	jones@gmail.com	99984983478

Bottom Table (Light Header):

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

```

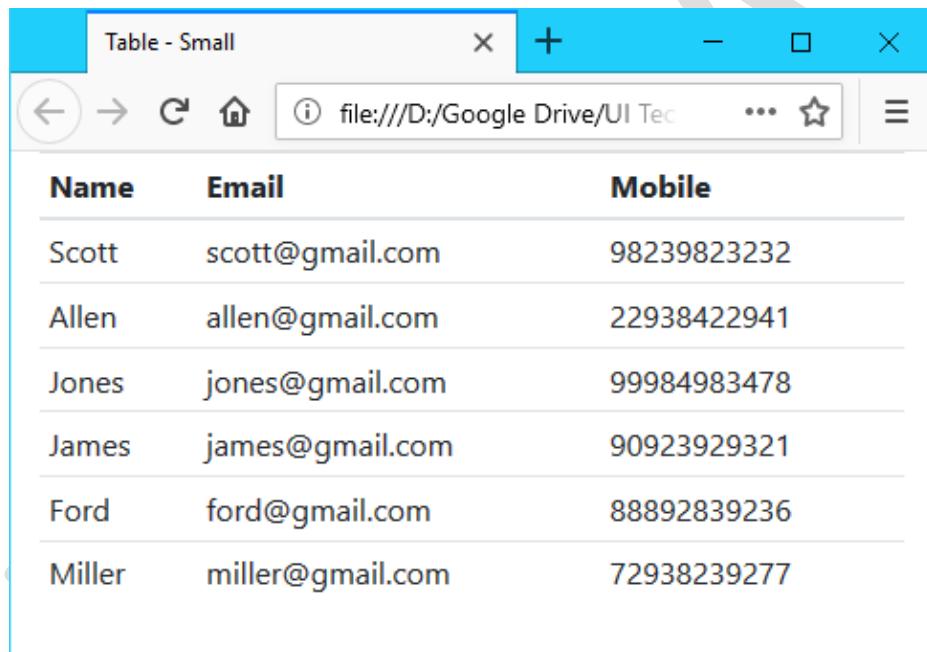
<html>
  <head>
    <title>Table - Small</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></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>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>
          </tr>
        </tbody>
      </table>
    </div>
  </body>
</html>

```

```

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

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.

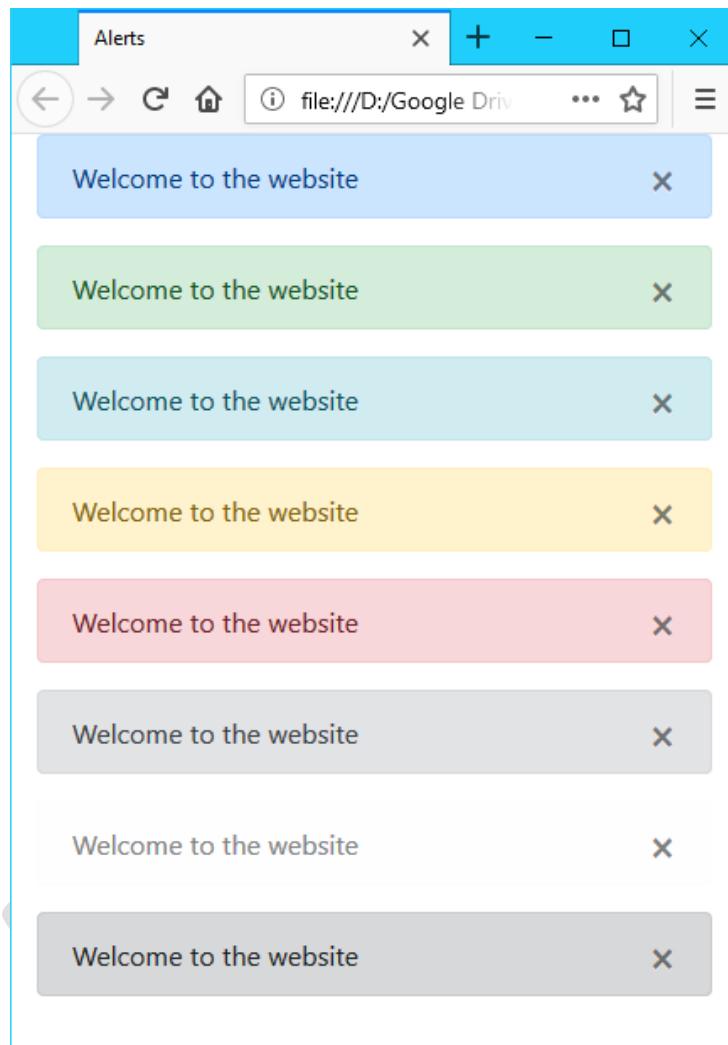
List of Classes

- 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>
```



Buttons

Button Colors

- It is used to display buttons with different colors.

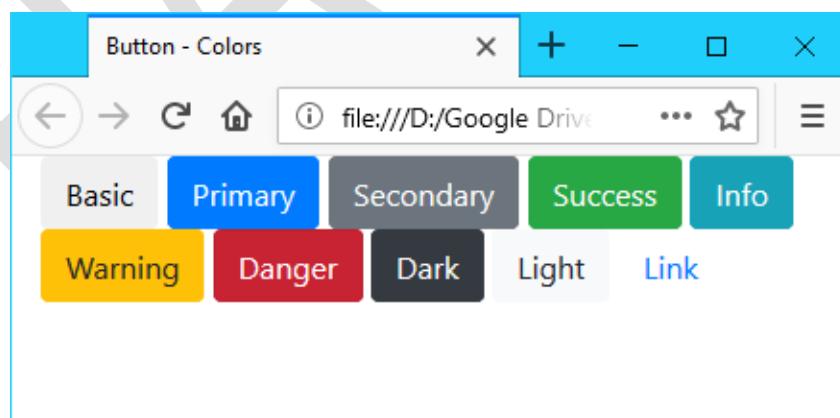
List of Classes

- 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

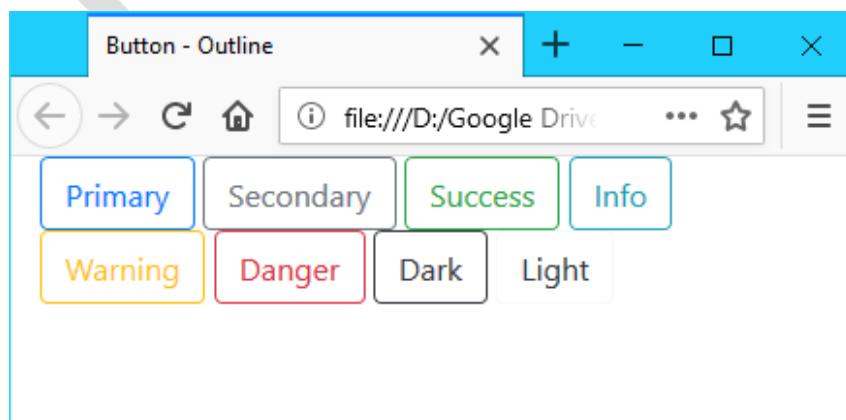
- It is used to display buttons with only border (white background).

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

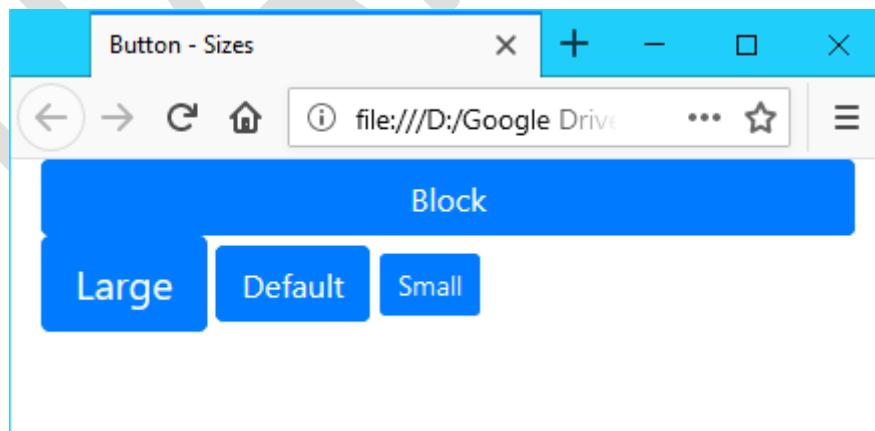
- It is used to display large / small buttons.

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

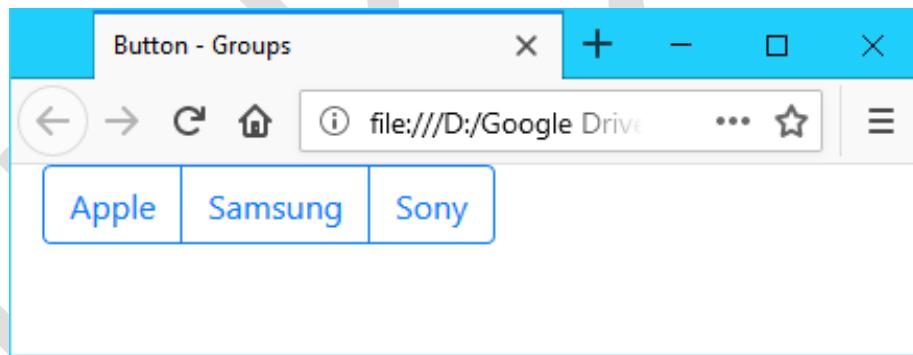
- It is used to display grouped buttons.
- It reduces margin between buttons.

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

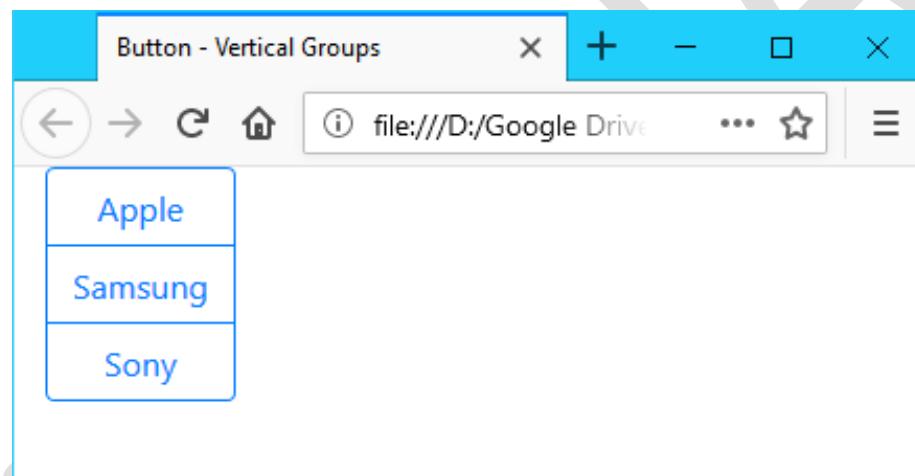
Example on Button Vertical Groups

```
<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 : Dropdown button
- dropdown-toggle-split : Split button
- caret : Caret symbol
- dropdown-menu : Represents the menu
- dropdown-item : Represents item in the menu

Example on Button DropDown

```

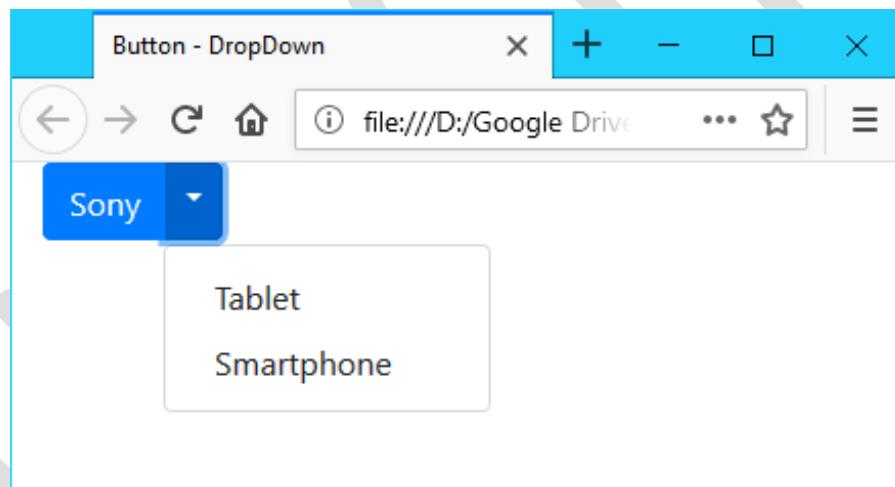
<html>
<head>
<title>Button - DropDown</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">

```

```

<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>

```



Badges

Basic Badges

- It is used to indication near heading / button.

List of Classes

- badge : Represents Badge

Example on Badges

```

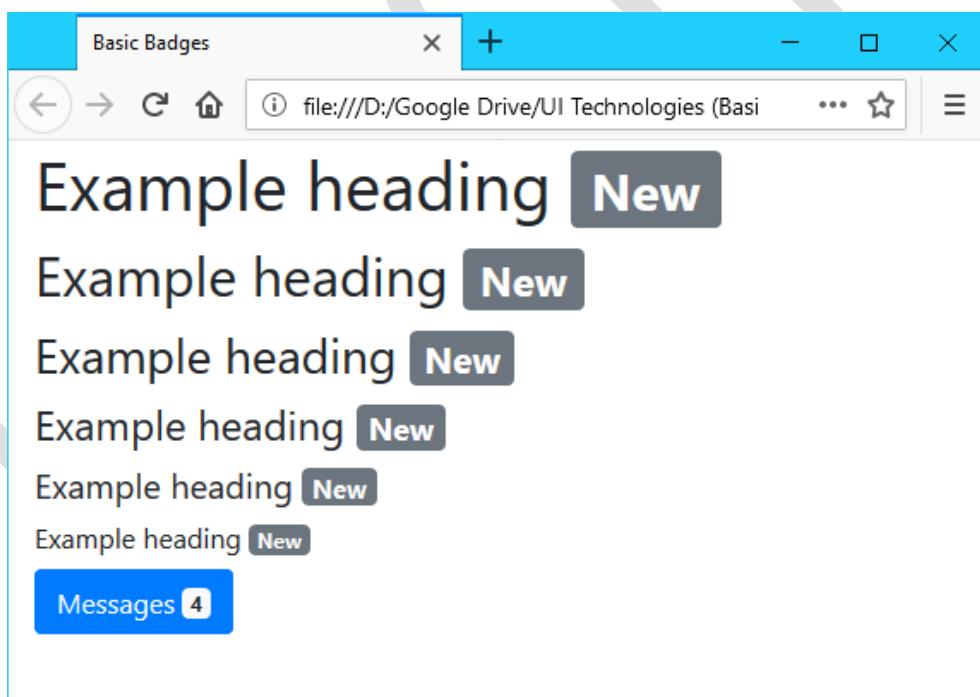
<html>
<head>
<title>Basic Badges</title>

```

```

<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>
<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

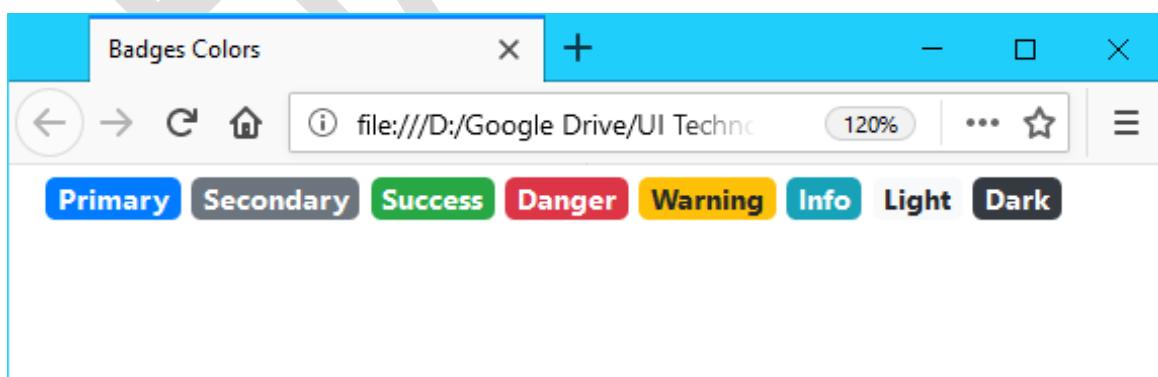
List of Classes

- 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

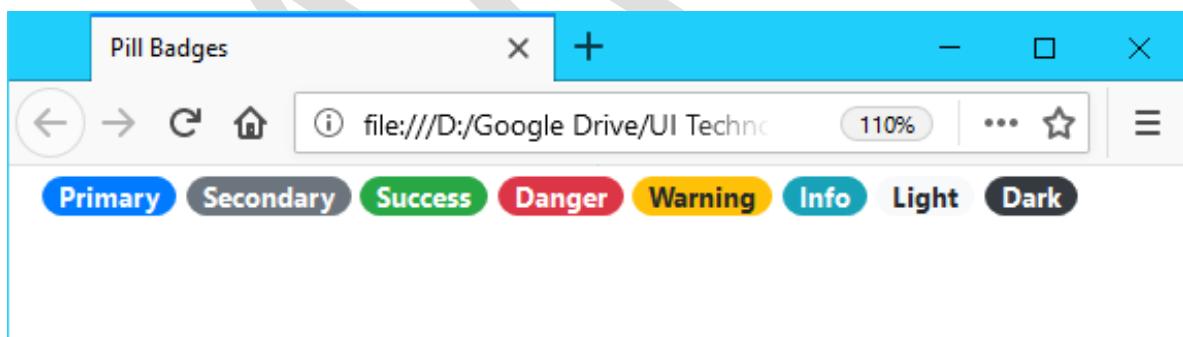
- It is used to display more rounded badges.

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

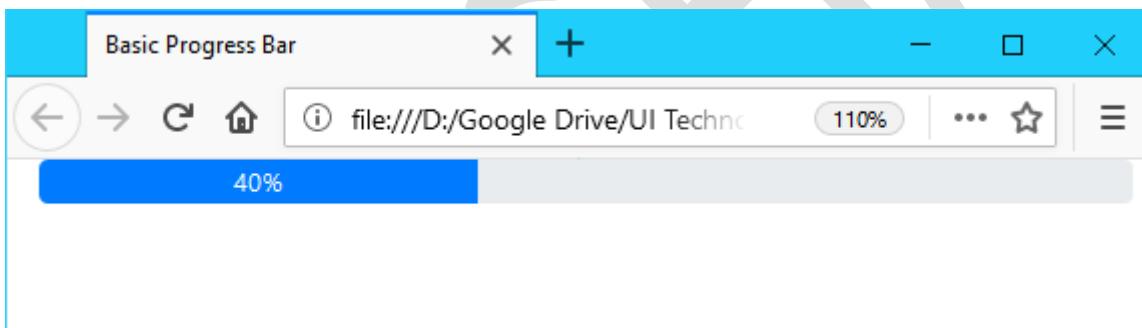
- It is used to display progress bar, based on the given width and height.

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>

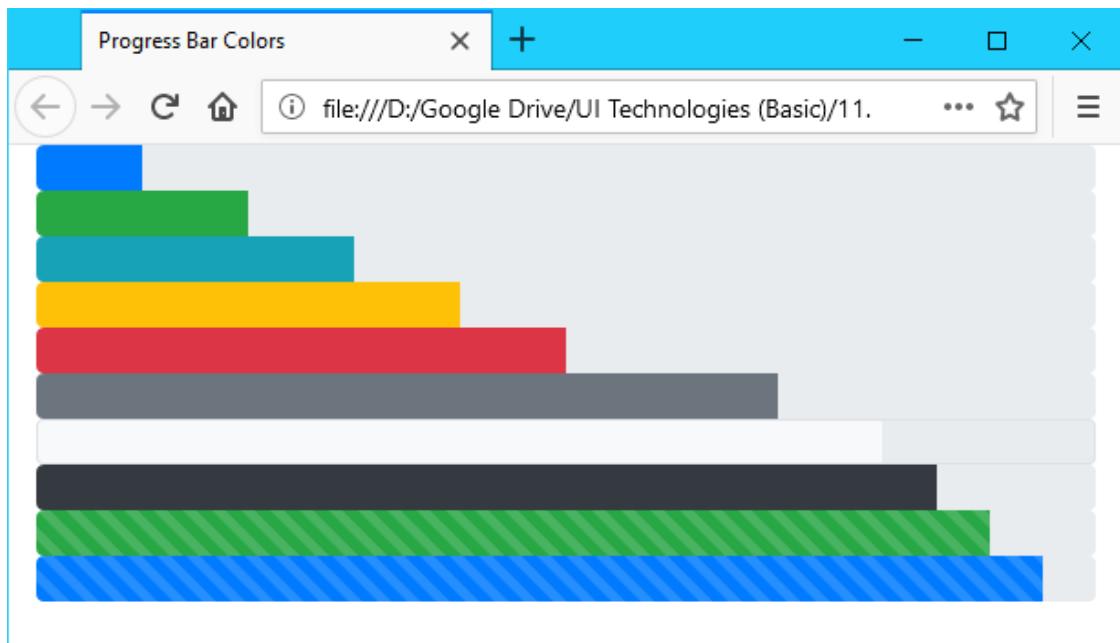
        <div class="progress" style="height:25px">
            <div class="progress-bar bg-dark" style="width:85%"></div>
        </div>

        <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>

```

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



Pagination

Basic Pagination

- It is used to display page numbers.

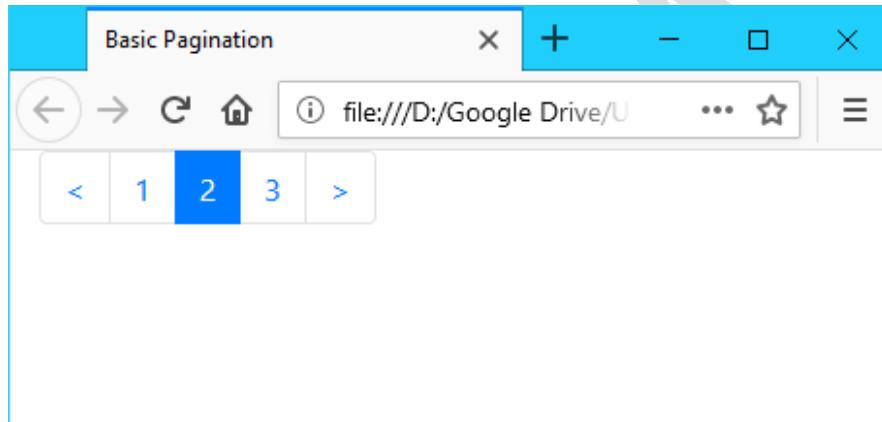
List of Classes

- 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

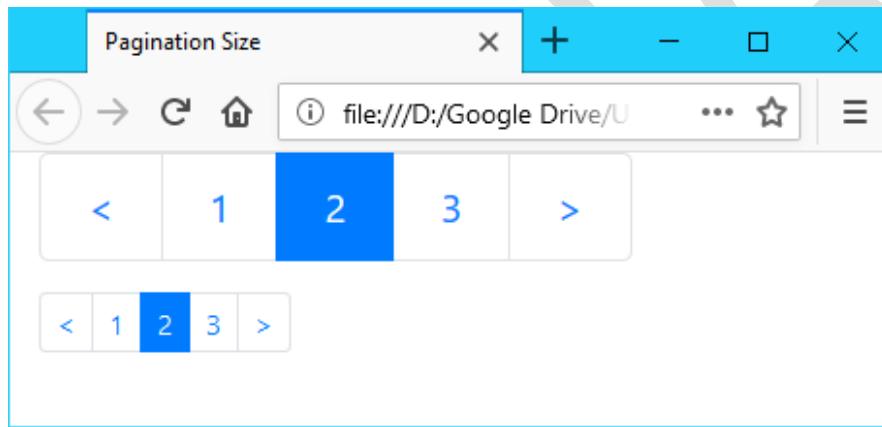
```
<html>
<head>
<title>Pagination Size</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="bootstrap.css">
<script src="jquery-3.3.1.js"></script>
<script src="popper.js"></script>
<script src="bootstrap.js"></script>
</head>
<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>

<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">

```

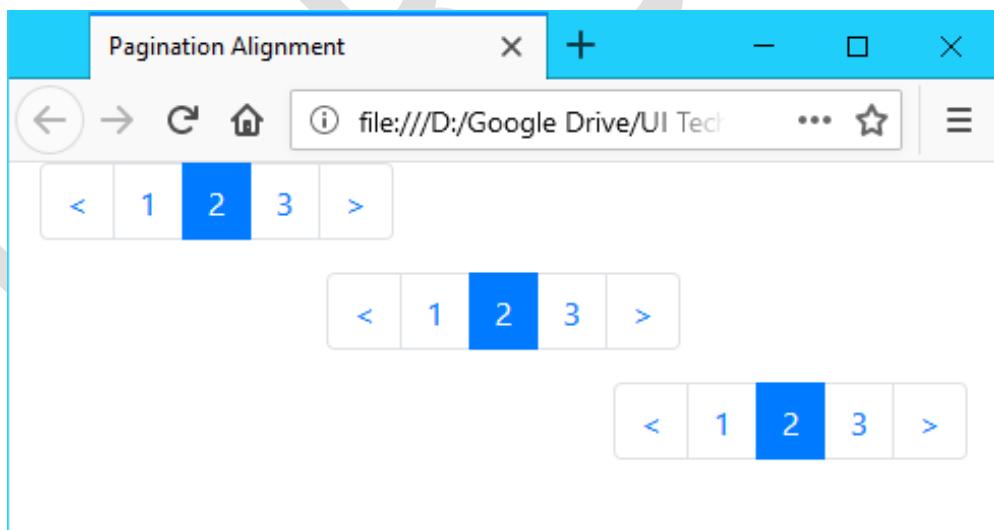
```

<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

Breadcrumbs

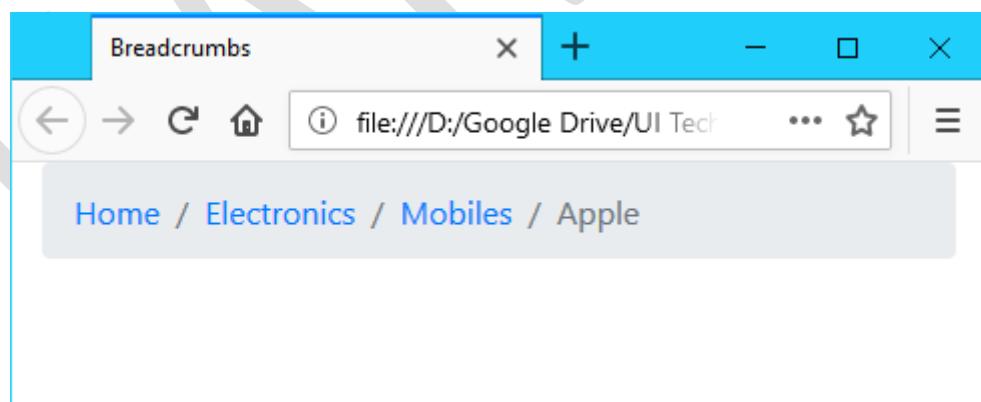
- It is used to display location (navigation path) of the current web page in the website.

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.

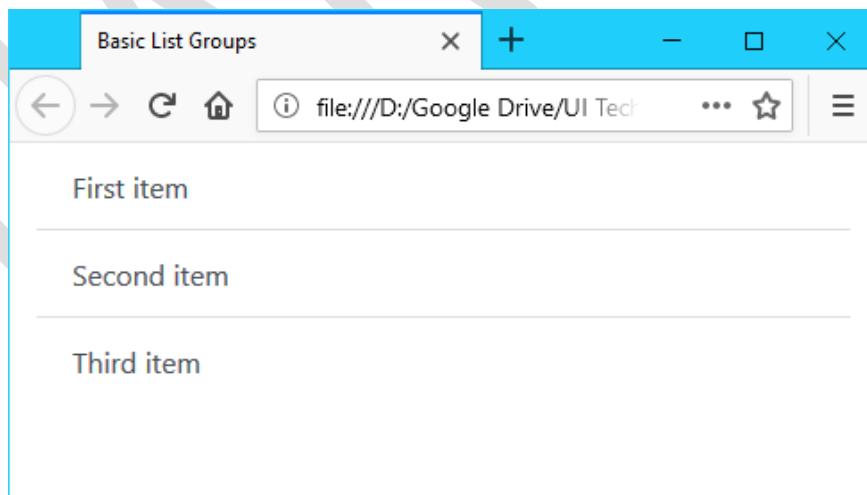
List of Classes

- 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.

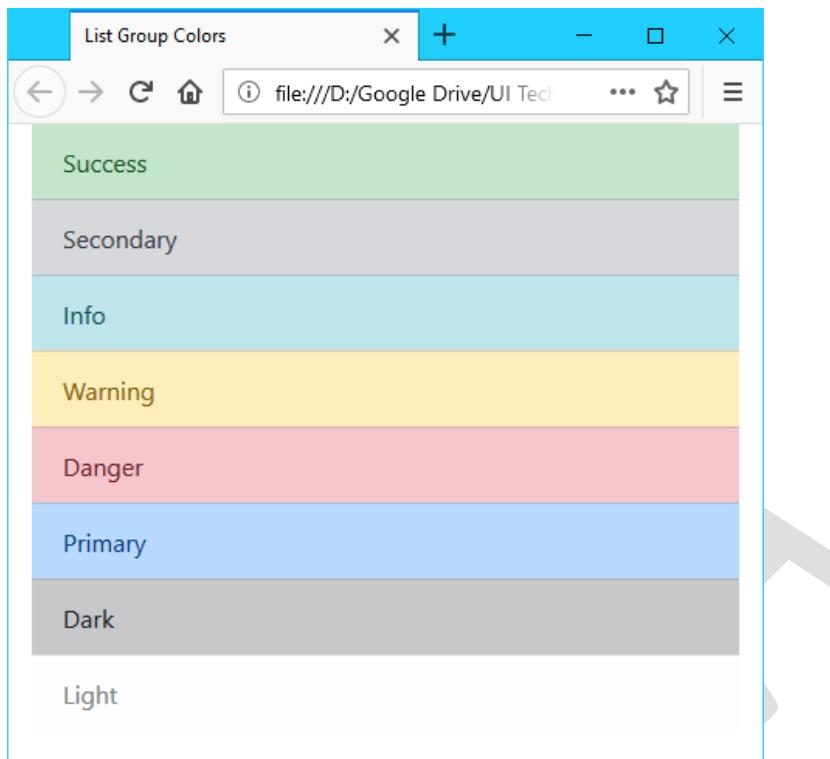
List of Classes

- 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.

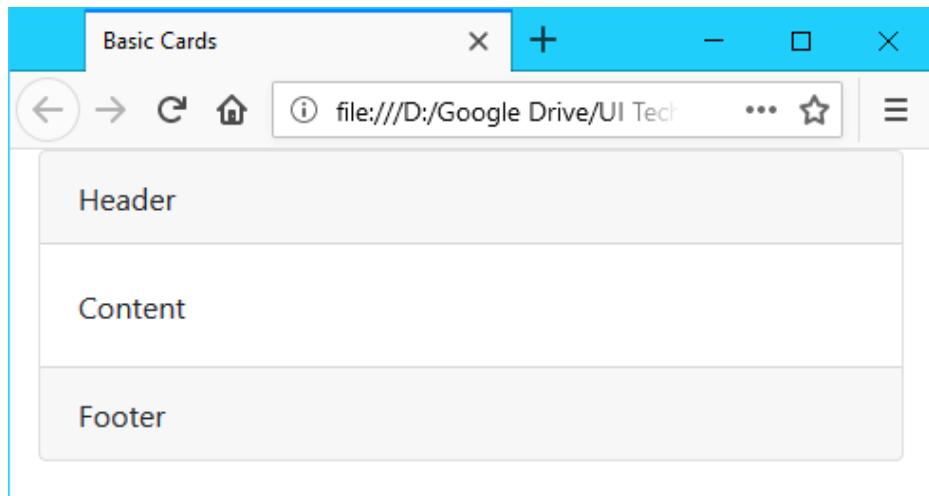
List of Classes

- 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.

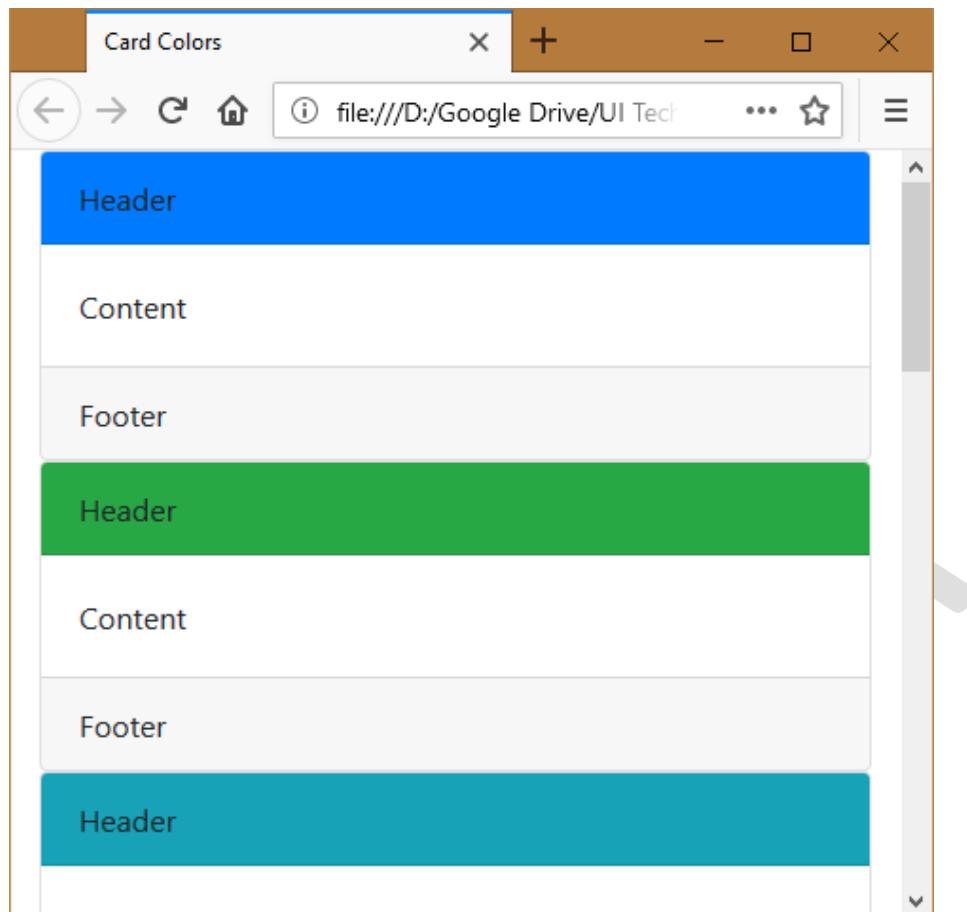
List of Classes

- 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

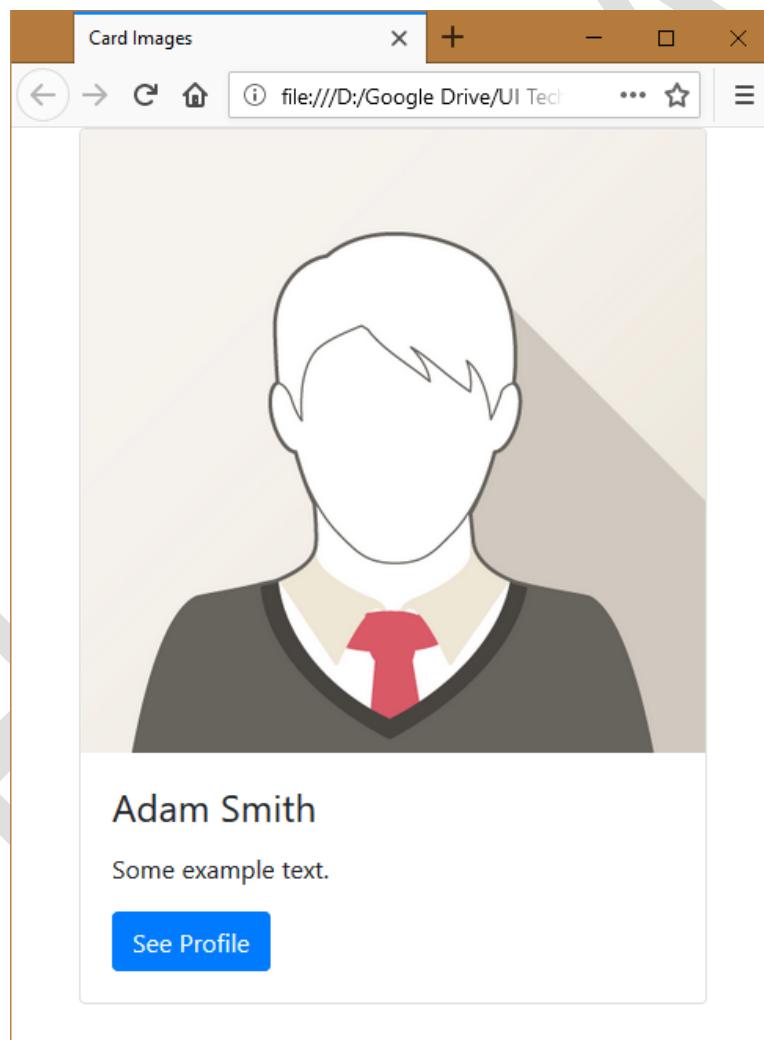
- card-img-top : Represents card image at top
- card-img-bottom : Represents card image at bottom
- card-title : Title of the card
- card-text : Normal text of the card

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

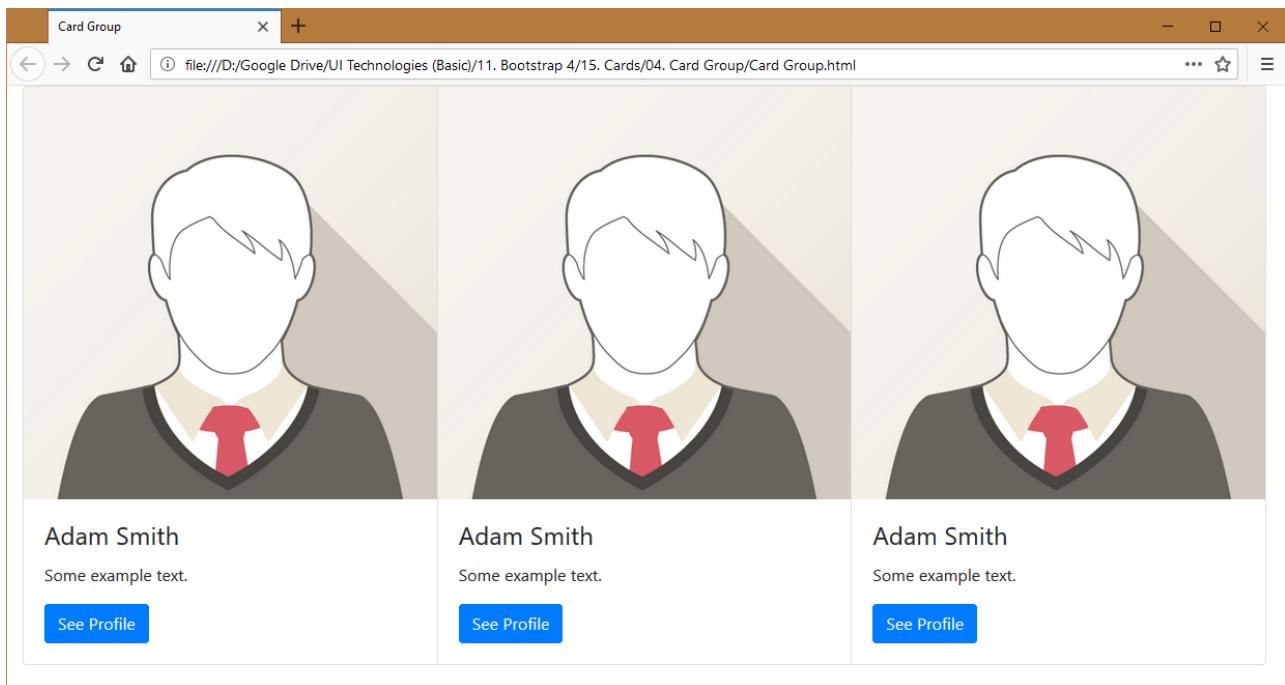
- It is used to display group of cards side by side.

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 |

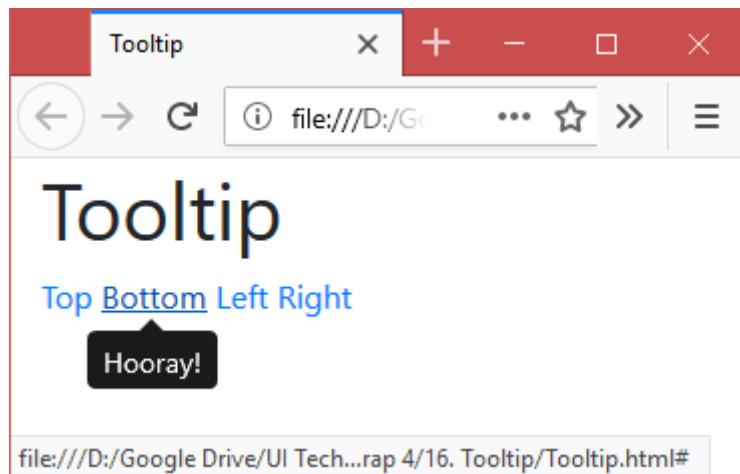
Example on Tooltip

```
<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>
<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

```

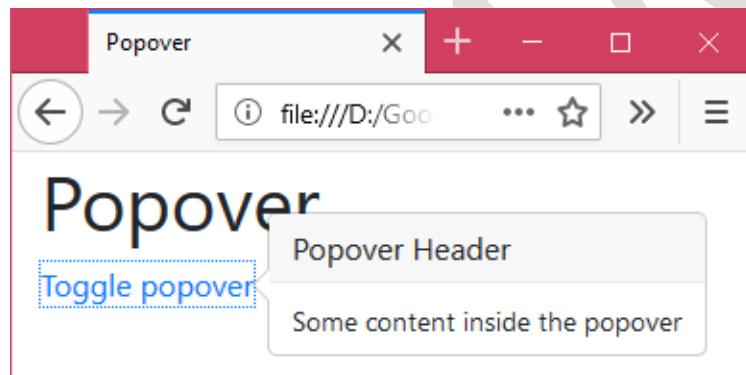
<html>
<head>
  <title>Popover</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="bootstrap.css">
  <script src="jquery-3.3.1.js"></script>

```

```

<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>
    $('[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>

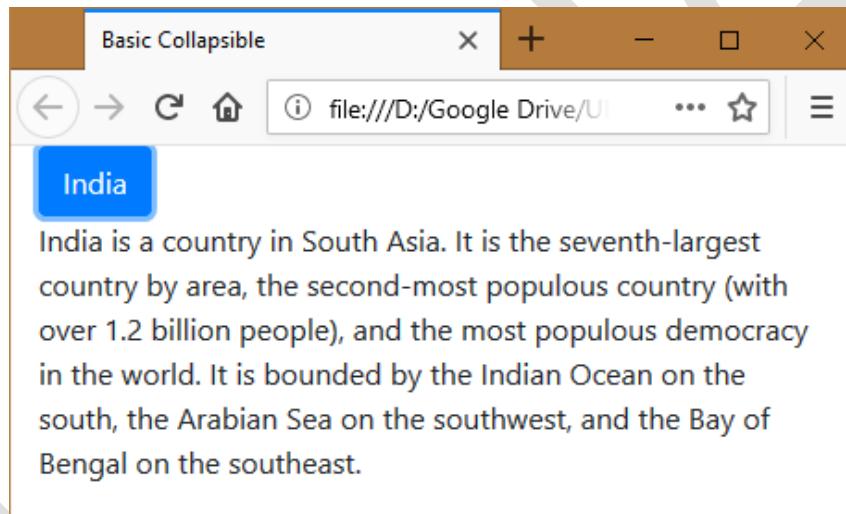
```

```

<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>

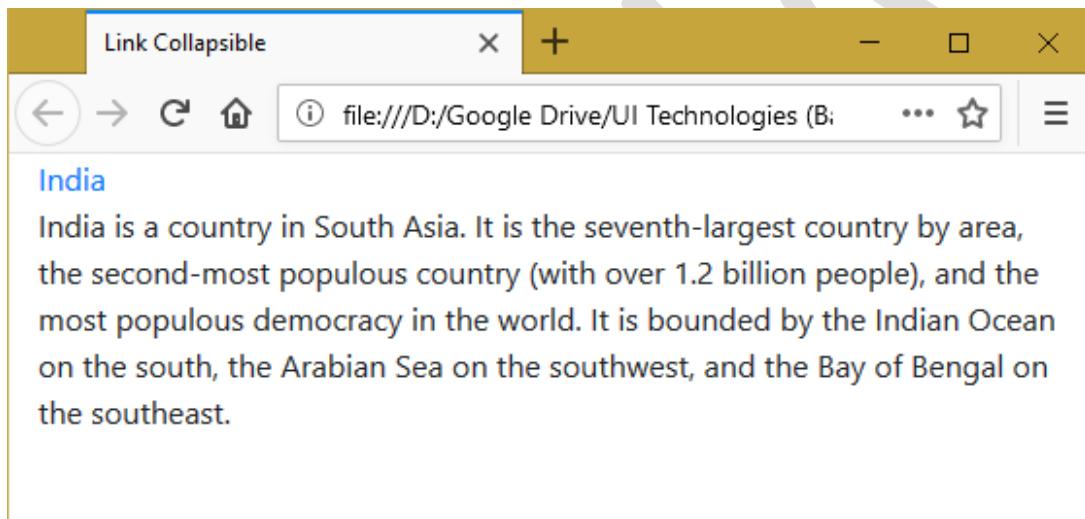
```

```

</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>

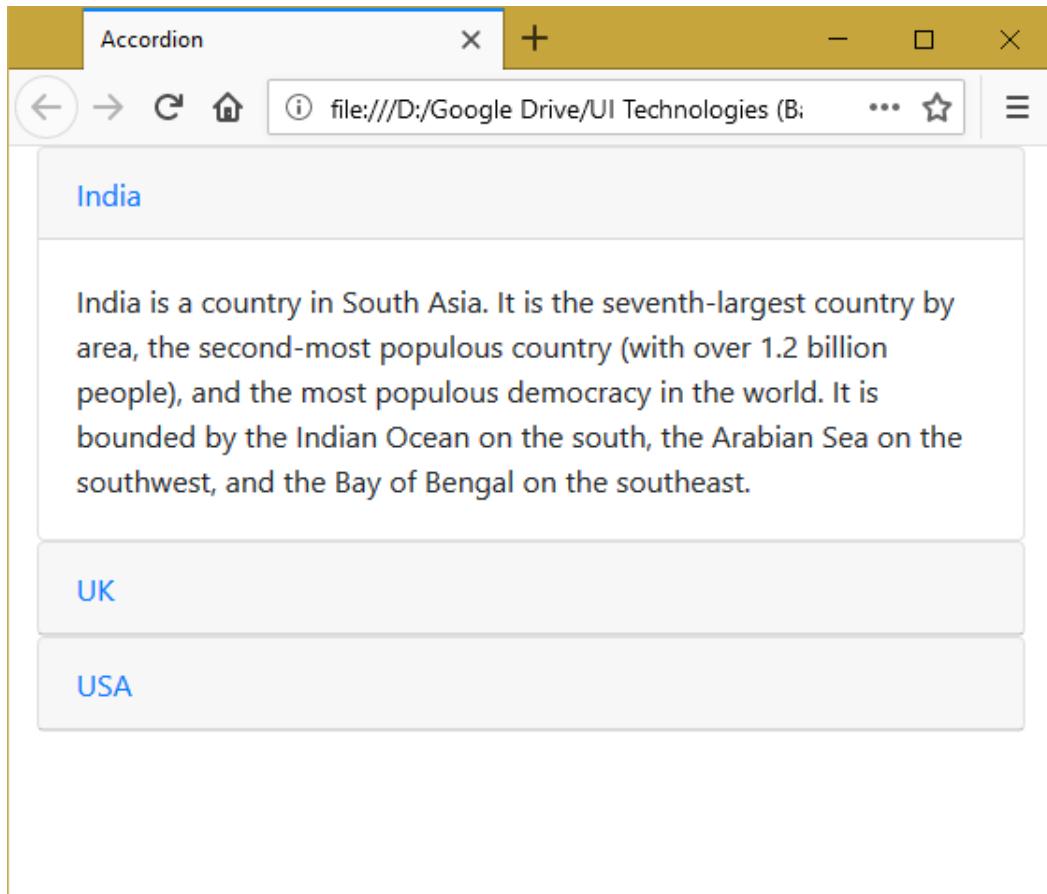
```

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



Forms

Inline Form

- It is used to create a simple form (with limited no. of elements), where the elements appear side-by-side.

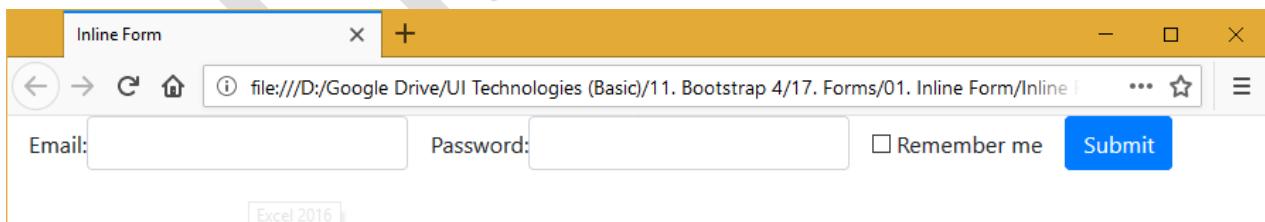
List of Classes

- form-inline
- form-group
- form-control
- mr-n
- form-check
- form-check-label

- form-check-input

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

- form-control-file

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>
                        <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>
                </form>
            </div>
        </div>
    </div>
</body>
```

```
</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>
```

The screenshot shows a registration form titled "Registration" within a browser window titled "Stacked Form". The form includes fields for Email, Password, Country (a dropdown menu), Comments (a text area), Attachment (a file selection button), Gender (radio buttons for Male and Female), and a Remember Me checkbox. A "Register" button is at the bottom. The browser interface includes standard navigation buttons and a status bar indicating the file path.

Email
Enter email
We'll never share your email with anyone else.

Password
Password

Country
Please Select

Comments

Attachment
Browse... No file selected.

Gender
 Male
 Female

Remember Me

Register

Form Grid

- It is used to create a form, where the desired form elements appear side-by-side.

List of Classes

- form-row

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>
        <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"), Password, Street (with placeholder "Street"), City (with placeholder "City"), Country (with placeholder "Please Select" and a dropdown arrow), and a large Comments area (with a text area placeholder). A blue "Register" button is located at the bottom left of the form.

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
- col-sm-n
- col-form-label
- offset-sm-n

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>
            </div>
          </form>
        </div>
      </div>
    </div>
  </body>

```

```
</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". The form includes fields for "Email" (with placeholder "Enter Email"), "Password" (with placeholder "Password"), "Country" (with placeholder "Please Select"), "Gender" (with radio buttons for "Male" and "Female"), and a "Remember me" checkbox. A blue "Sign in" button is at the bottom.

Input Groups

- It is used to create a form element, with some text inside it.

List of Classes

- input-group
- input-group-prepend
- input-group-append
- input-group-text

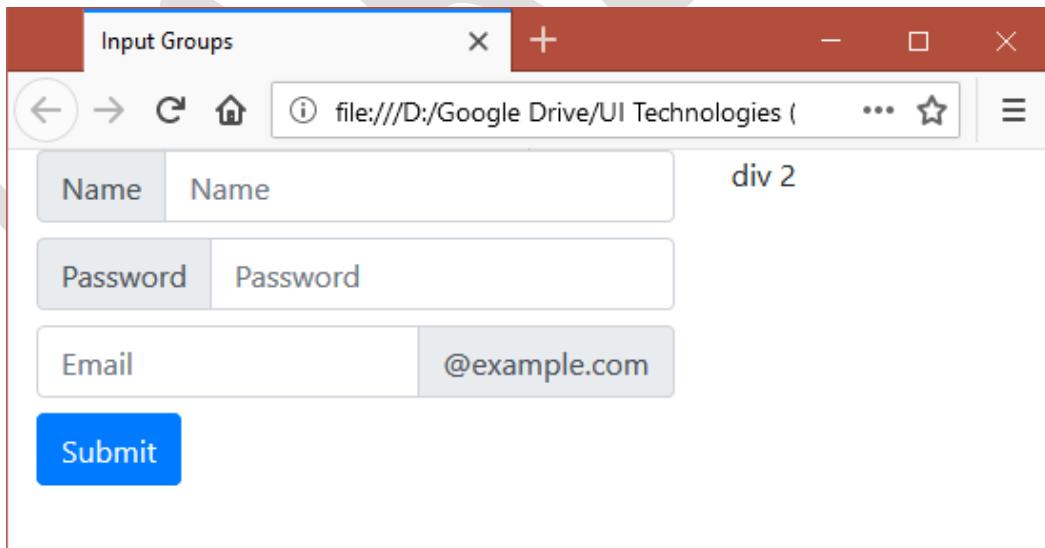
Example on Input Groups

```
<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>
</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

- novalidate="novalidate"
- valid-feedback
- invalid-feedback
- valid-tooltip
- invalid-tooltip
- required="required"
- pattern="reg exp"

Example on Form Validation

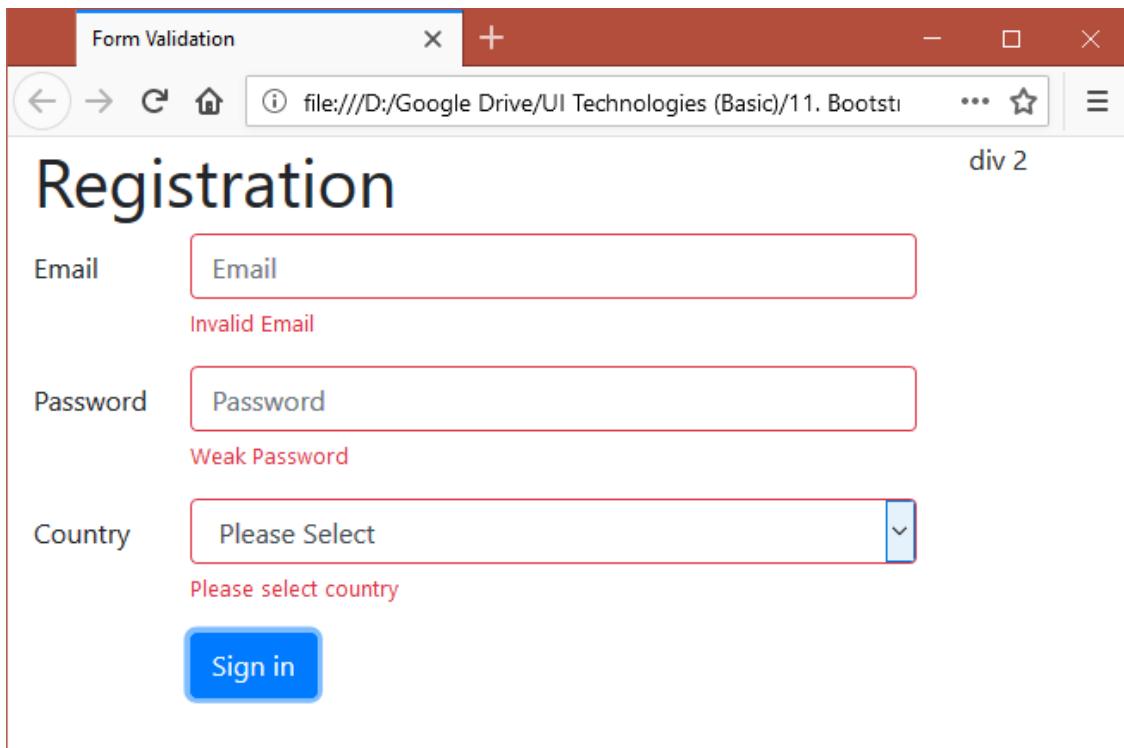
```
<html>
  <head>
    <title>Form Validation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></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 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>
```



Navigation

Basic Navigation

- 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
- nav-item
- nav-link

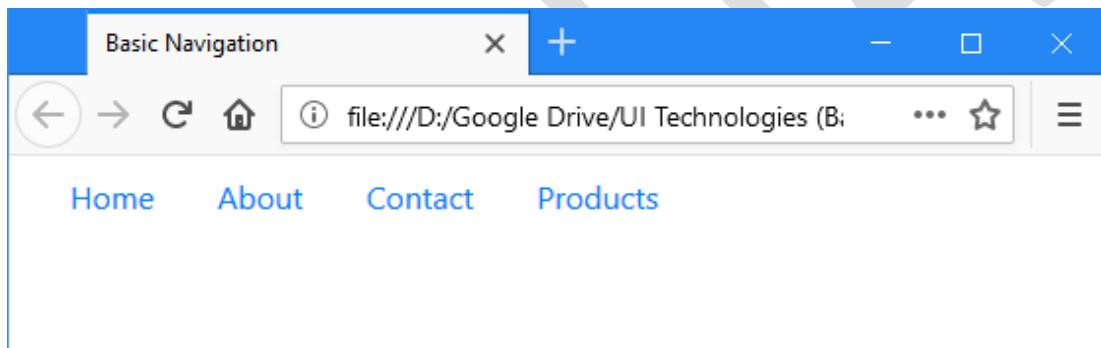
Example on Basic Navigation

```
<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 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
- justify-content-end

Example on Navigation 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>

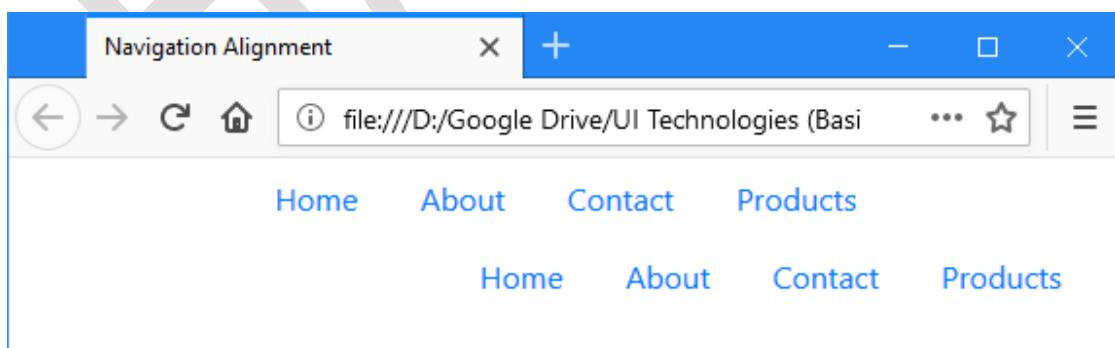
```

```

</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

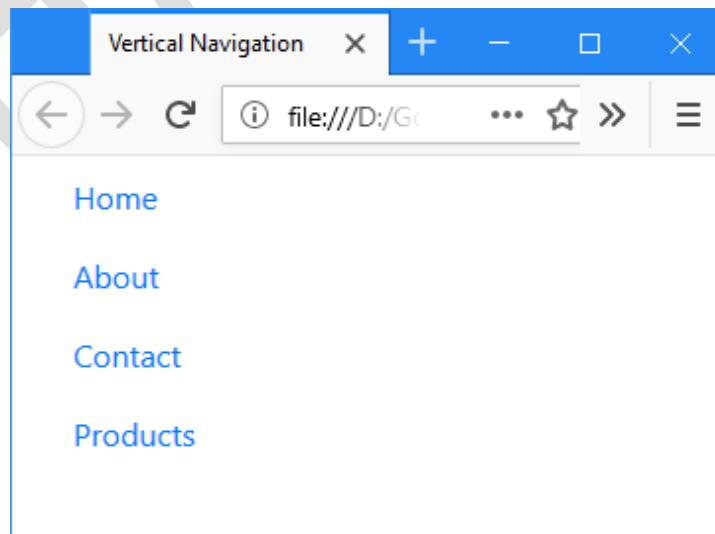
- It is used to display the simple navigation bar in vertical mode.

List of Classes

- flex-column

Example on Vertical Navigation

```
<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
- data-toggle="tab"
- tab-content
- tab-pane
- container
- active
- fade

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>

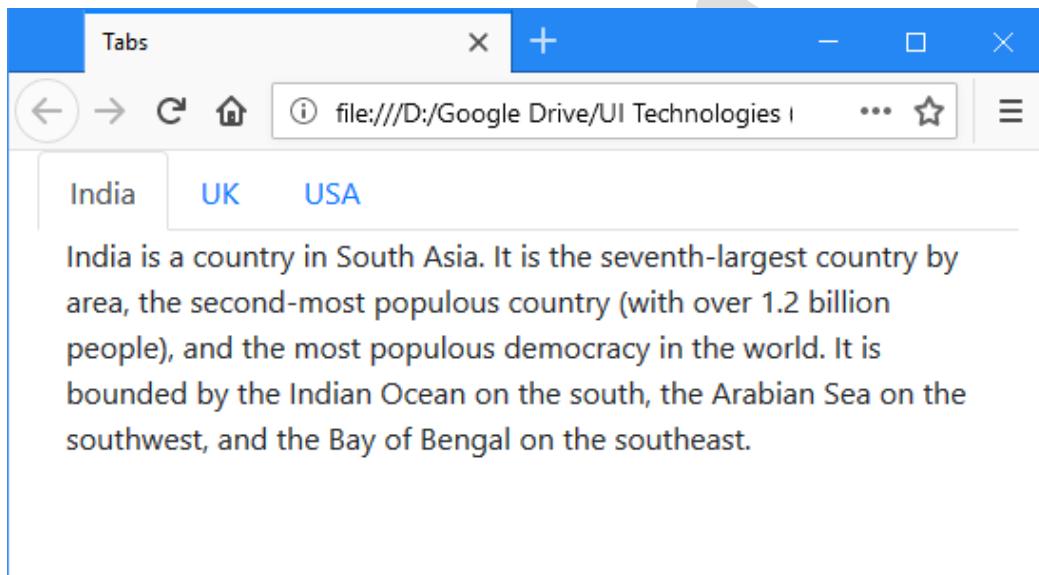
      <div class="tab-content">
        <div class="tab-pane container active" id="menu1">India is a country in South Asia. It is the seventh-largest country by area, the second-most populous country (with over 1.2 billion people), and the most populous democracy in the world. It is bounded by the Indian Ocean 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>
  </body>

```

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 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>
```



Pills

- It is used to display a set of tabs with more rounded corners and background color.

List of Classes

- nav-pills
- data-toggle="pill"

Example on Pills

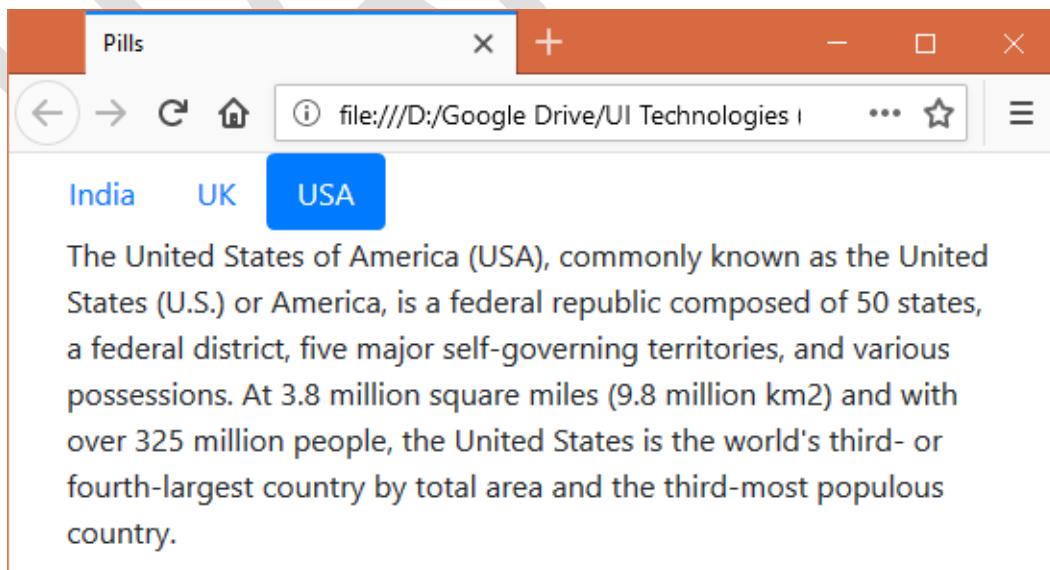
```
<html>
<head>
  <title>Pills</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="bootstrap.css">
  <script src="jquery-3.3.1.js"></script>
  <script src="popper.js"></script>
  <script src="bootstrap.js"></script>
</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 (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>
</body>
</html>

```



Tabs with DropDown

- It is used to display dropdown menu in the tabs.

List of Classes

- dropdown
- dropdown-toggle
- data-toggle="dropdown"
- dropdown-menu
- dropdown-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>

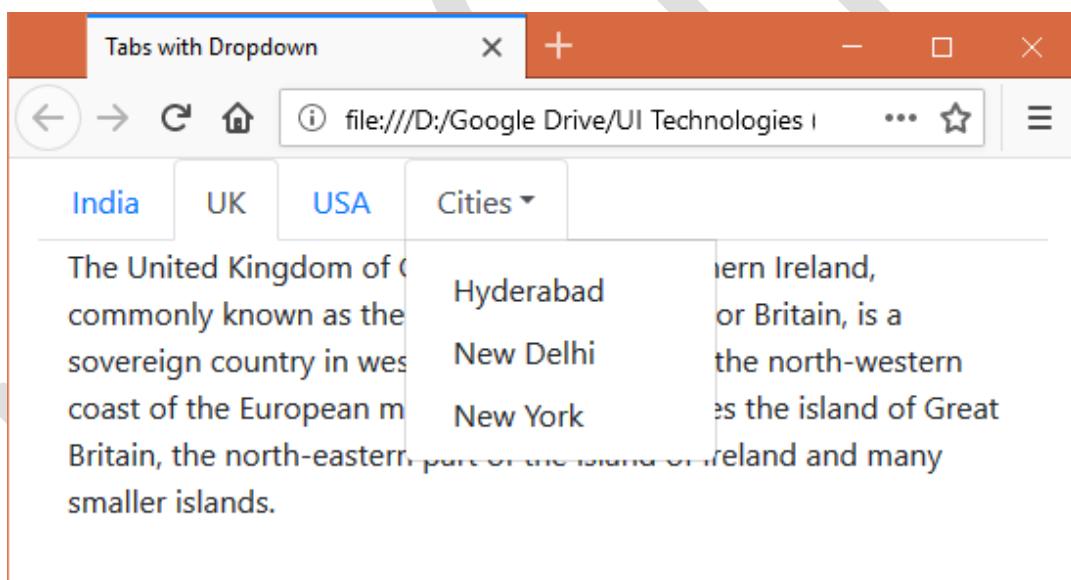
      <div class="tab-content">
        <div class="tab-pane container active" id="menu1">India is a country in South Asia. It is the seventh-largest country by area, the second-most populous country (with over 1.2 billion people), and the most populous democracy in the world. It is bounded by the Indian Ocean on the south, the Arabian Sea on the southwest, and the Bay of Bengal on the southeast.</div>
      </div>
    </div>
  </body>
</html>

```

```

<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 (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 class="tab-pane container fade" id="menu4">Hyderabad is the capital of the Indian state of
Telangana and de jure capital of Andhra Pradesh.</div>
<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 km2), 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
- navbar-expand-sm
- navbar-dark
- navbar-nav
- nav-item
- nav-link
- active

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>

```

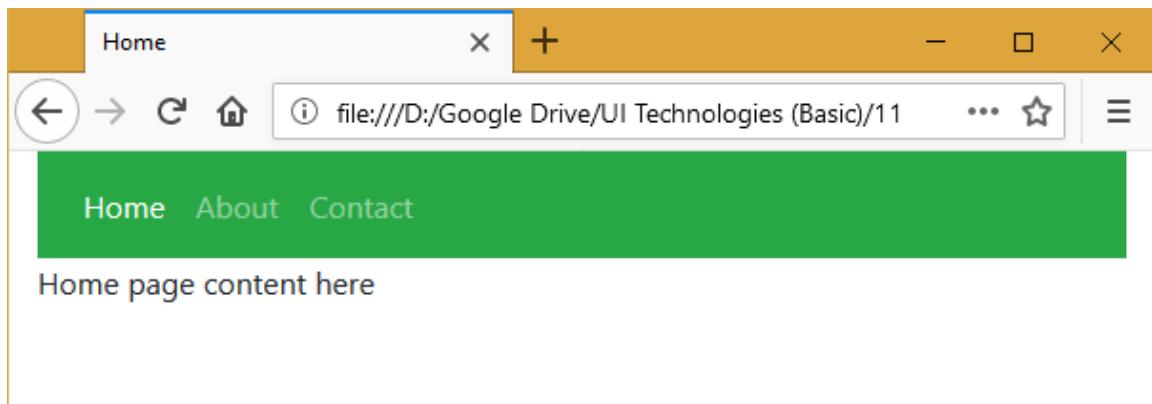
contact.html

```

<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>
<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-expand-sm
- navbar-brand
- navbar-toggler
- data-toggle="collapse"
- data-target="#id"
- collapse
- navbar-collapse

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>  

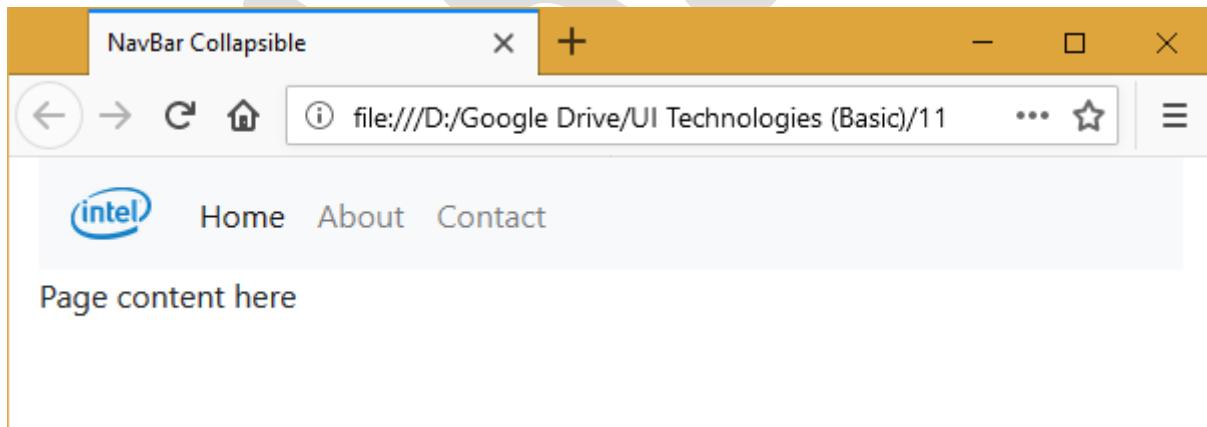
  Page content here  

</div>  

</body>  

</html>

```



NavBar DropDown

- It is used to display a dropdown menu for the navigation bar item.

List of Classes

- dropdown
- dropdown-toggle
- data-toggle="dropdown"

- dropdown-menu
- dropdown-item

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>

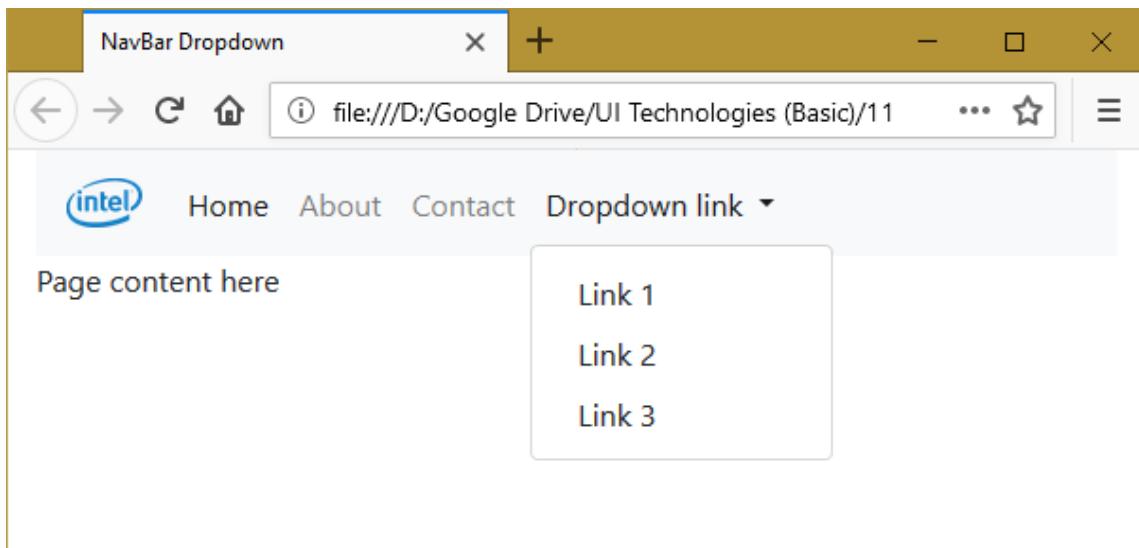
        <div class="collapse navbar-collapse" id="collapsibleNavbar">
          <ul class="navbar-nav">
            <li class="nav-item">
              <a class="nav-link active" href="#">Home</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">About</a>
            </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>
    </div>
  </body>
</html>

```

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
- form-control
- mr-sm-n

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>
      </nav>
    </div>
  </body>
</html>
```

```

<button class="navbar-toggler" data-toggle="collapse" data-target="#collapsibleNavbar">
  <span class="navbar-toggler-icon"></span>
</button>

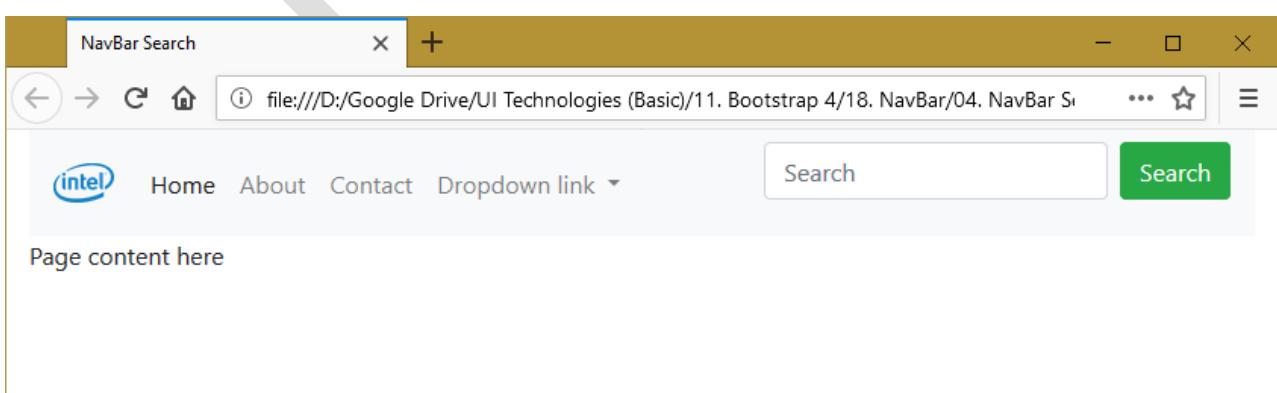
<div class="collapse navbar-collapse" id="collapsibleNavbar">
  <ul class="navbar-nav">
    <li class="nav-item">
      <a class="nav-link active" href="#">Home</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">About</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" 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

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 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>
    </div>
  </body>
</html>

```

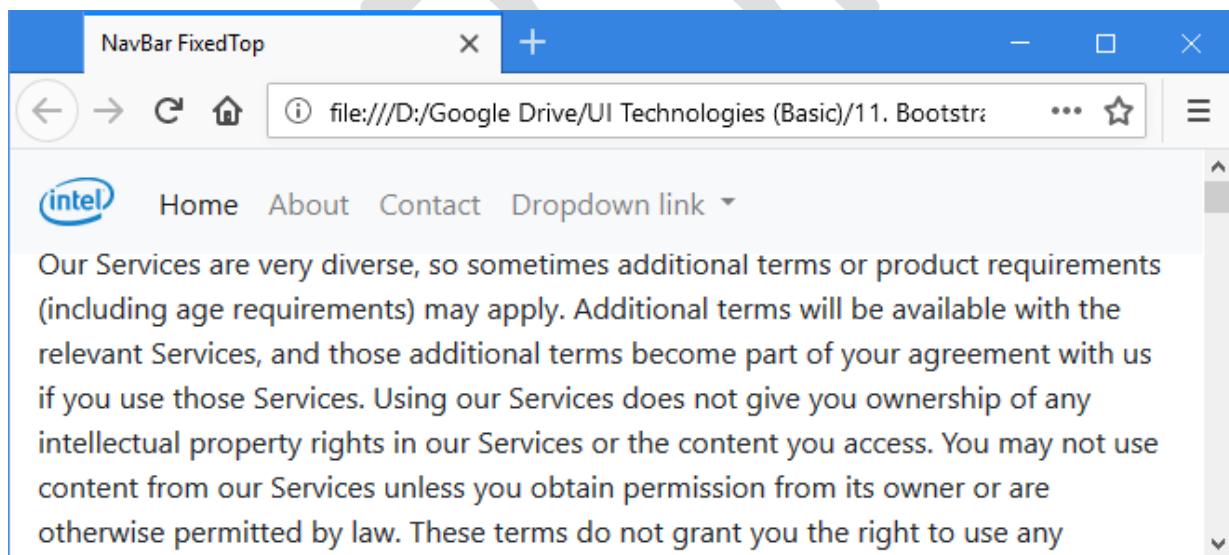
```

</div>
</li>
</ul>
</div>
</nav>


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.


</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

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>

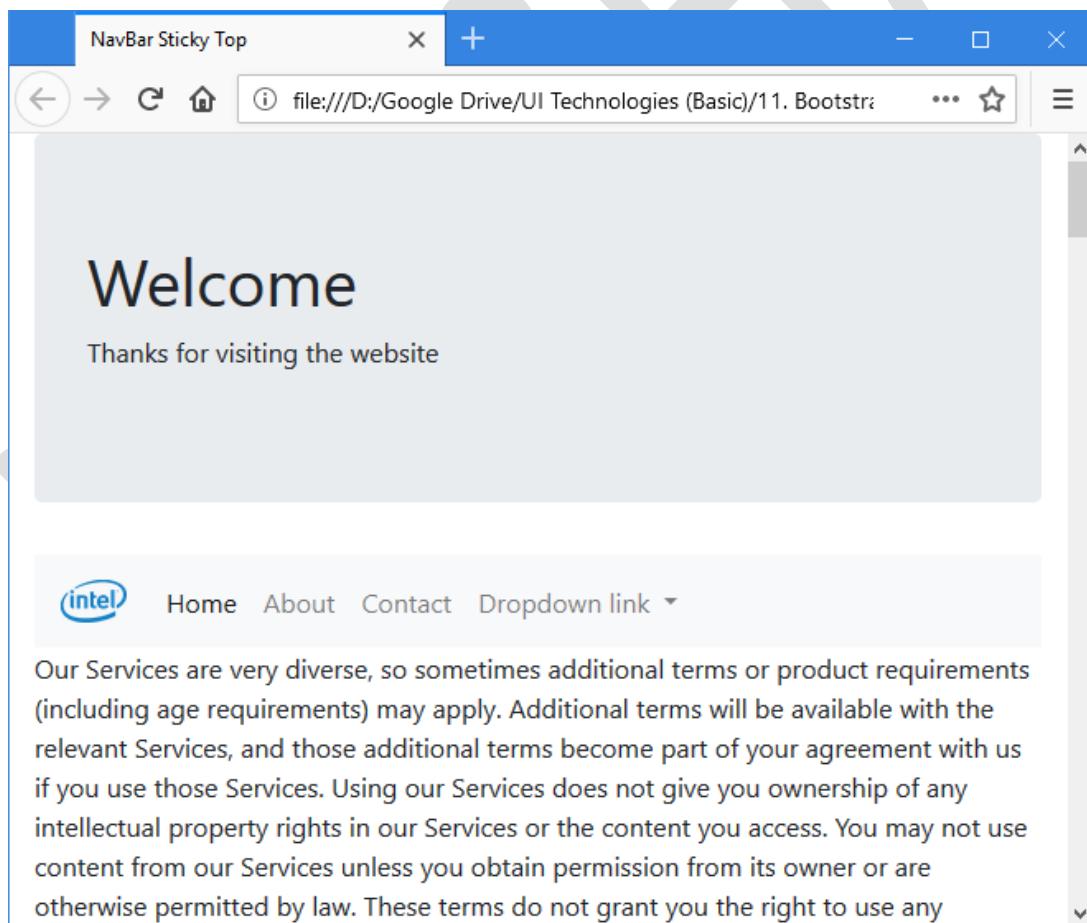
        <div class="collapse navbar-collapse" id="collapsibleNavbar">
          <ul class="navbar-nav">
            <li class="nav-item">
              <a class="nav-link active" href="#">Home</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">About</a>
            </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>
    </div>
  </body>

```

<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"
- data-target=".navbar"
- data-offset="pixels"

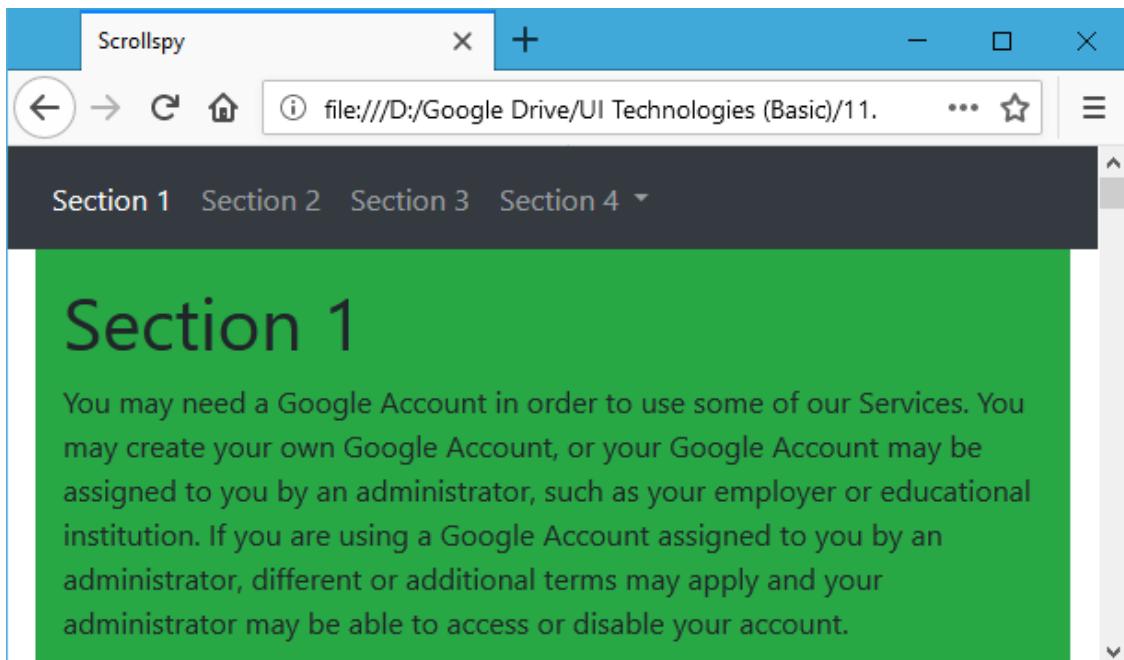
Example on ScrollSpy

```
<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>

```

```
<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 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
- slide
- data-ride="carousel"
- data-interval="milli seconds"
- carousel-indicators
- data-target="#id"
- data-slide-to="n"
- active
- carousel-inner
- carousel-item
- carousel-caption
- carousel-control-prev
- carousel-control-prev-icon
- data-slide="prev"
- data-slide="next"

Example on Carousel

```

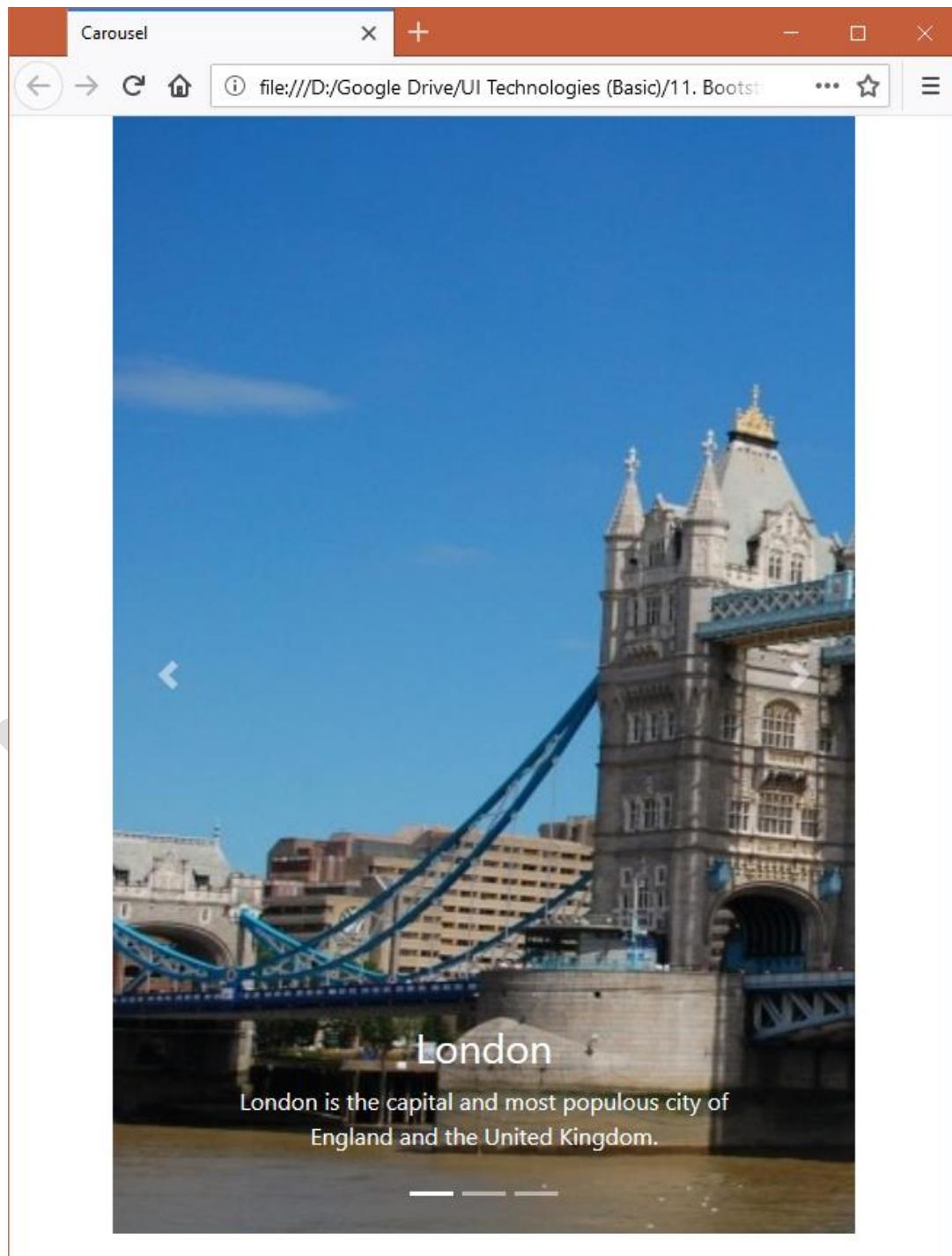
<html>
  <head>
    <title>Carousel</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap.css">
    <script src="jquery-3.3.1.js"></script>
    <script src="popper.js"></script>
    <script src="bootstrap.js"></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>
      </div>
    </div>
  </body>
</html>

```

```
</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-toggle="modal"
- data-target="#id"
- modal
- fade
- modal-dialog
- modal-lg
- modal-dialog-centered
- modal-content
- modal-header
- modal-body
- modal-footer
- close
- data-dismiss="modal"

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>

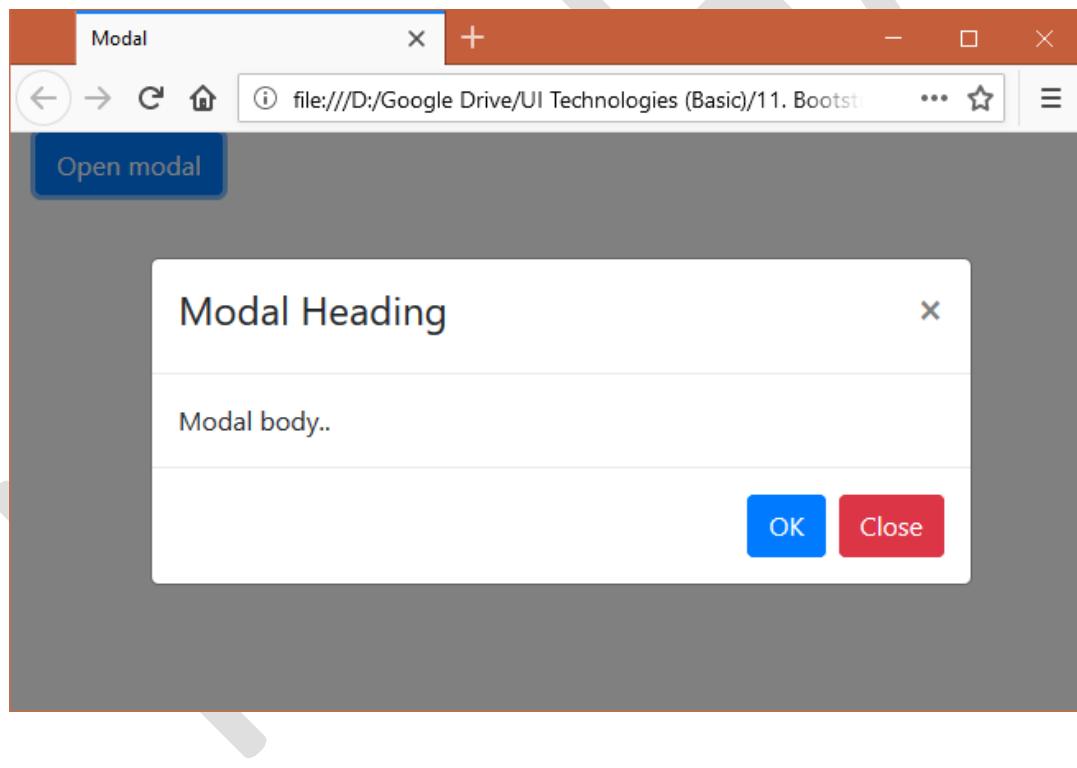
      <div class="modal fade" id="myModal">
        <div class="modal-dialog modal-lg modal-dialog-centered">
          <div class="modal-content">

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



LESS

Fundamentals of LESS

Introduction to LESS

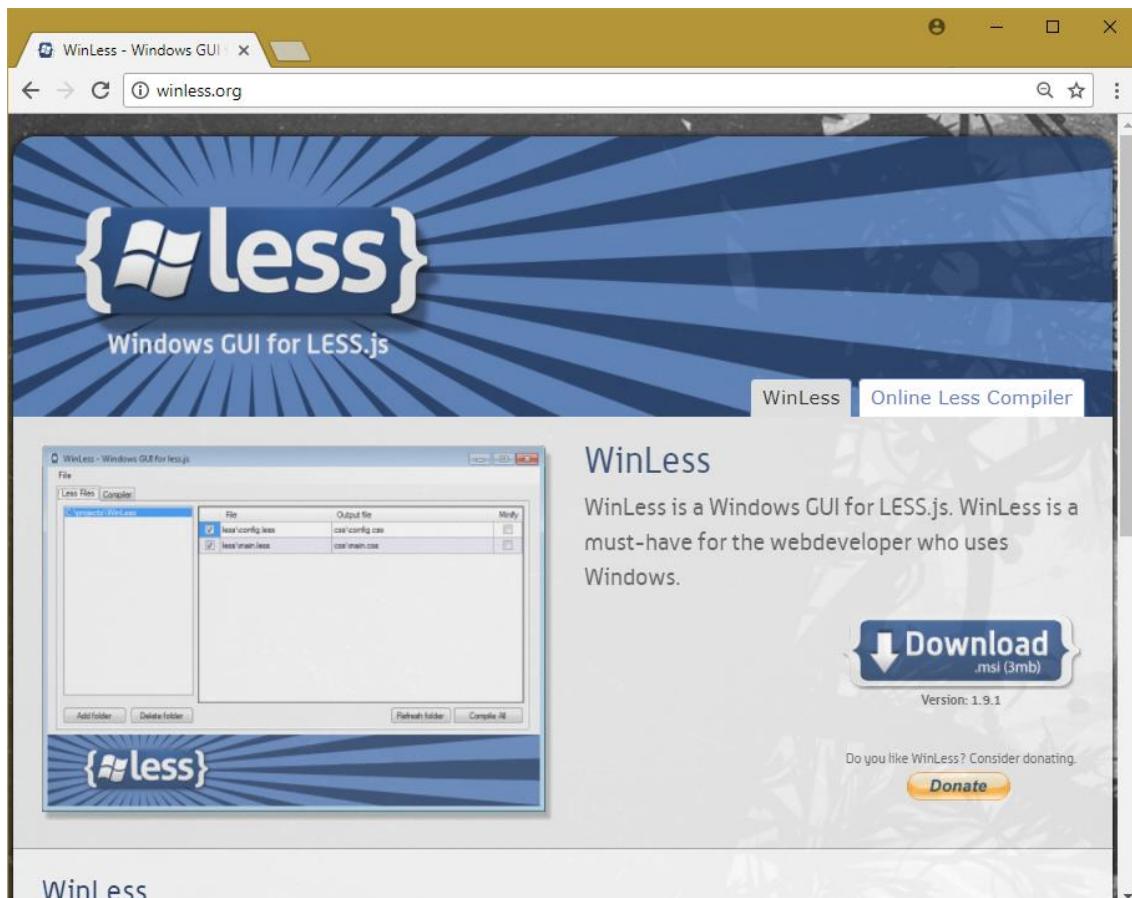
- LESS stands for "LEAN CSS".
- "LESS" makes it easy to modify the CSS file. If you make the change in one place, the same will be automatically effected in multiple places.
- The LESS code is written in ".less" files and will be converted into "css" files, as the browser supports only CSS, but not LESS.
- If you modify the LESS file and compile it into CSS file, the change will be effected in multiple places in CSS.
- Additionally, LESS supports some additional features such as variables, operators, mixins, color functions etc., which are not supported by CSS. LESS is the superset of CSS, which supports all concepts of CSS along with the specified additional features.

Steps to Prepare First Example in LESS

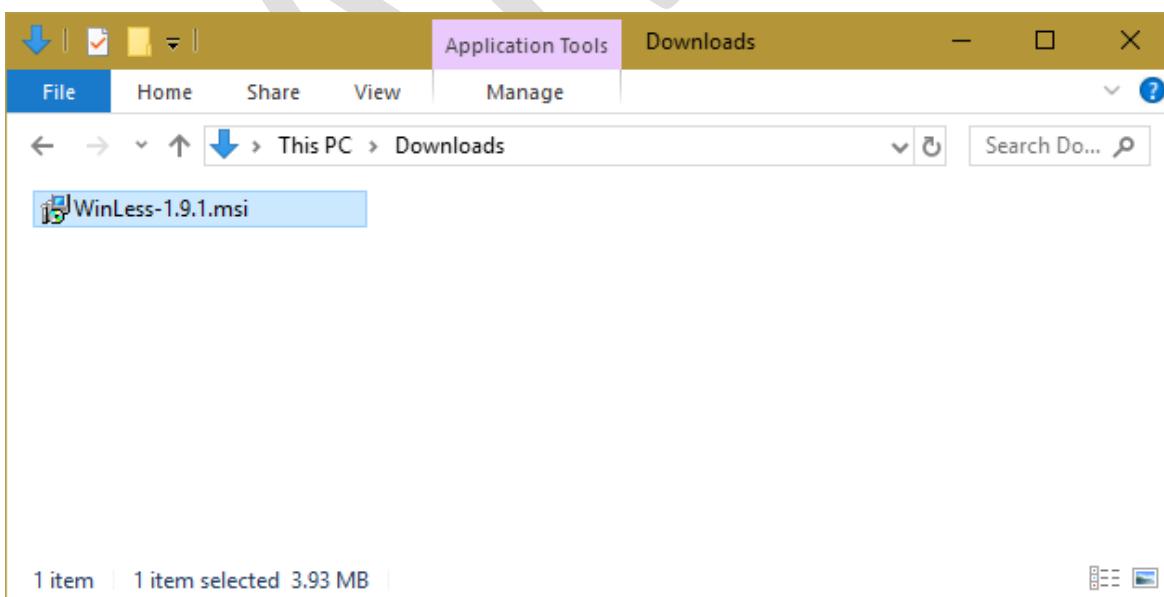
1. Download and Install WinLess
2. Create LESS file
3. Compile LESS file to CSS file
4. Import CSS file into HTML file
5. Run the HTML file

1. Downloading and Installing WinLess

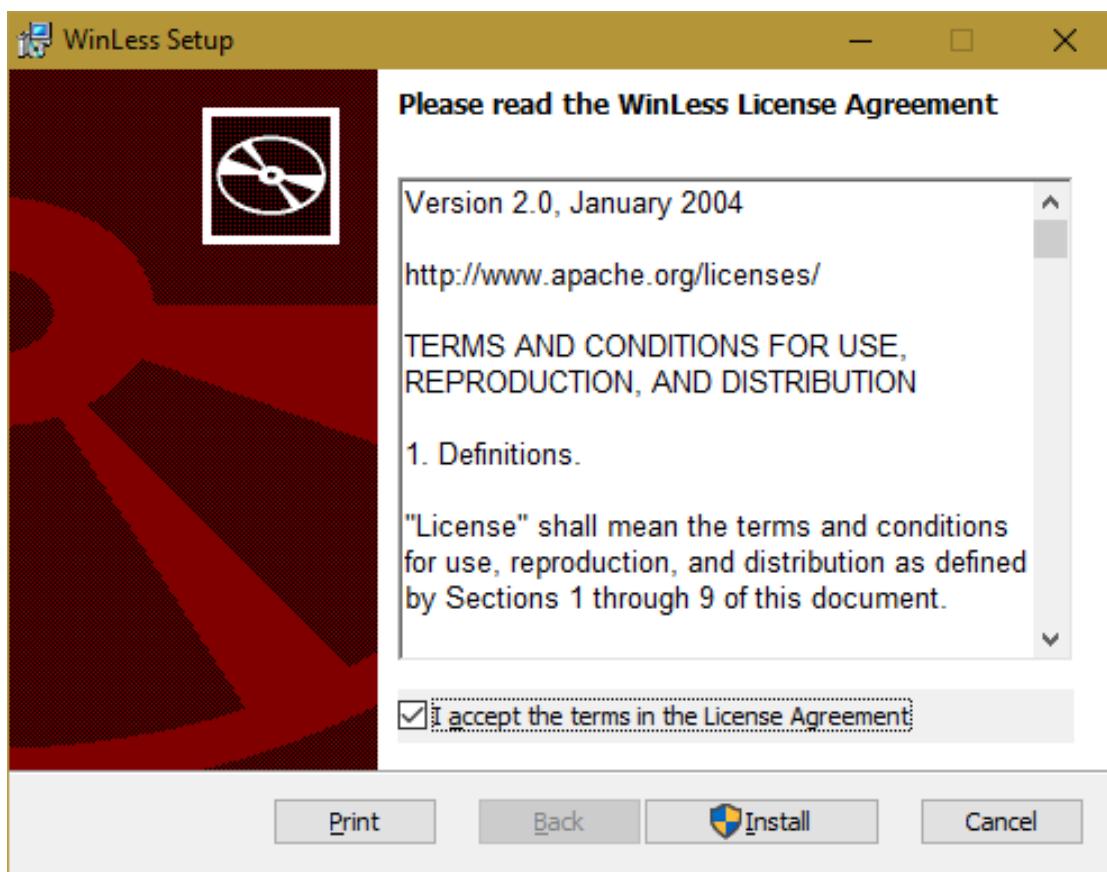
- Go to "<http://winless.org>".



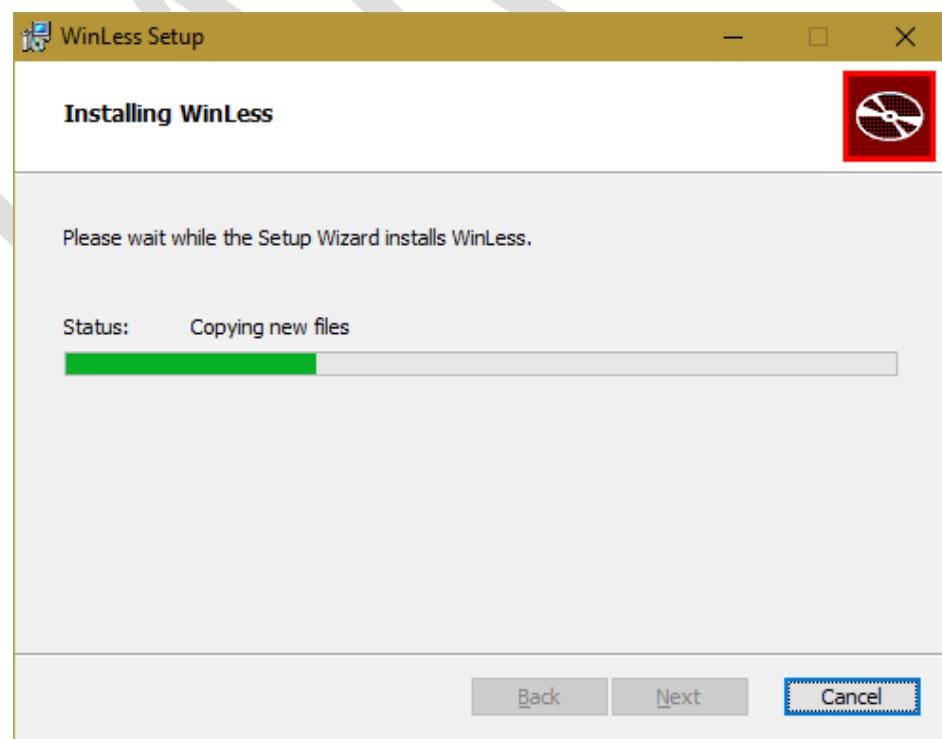
- Click on "Download".
- You will get a file called "WinLess-1.9.1.msi".



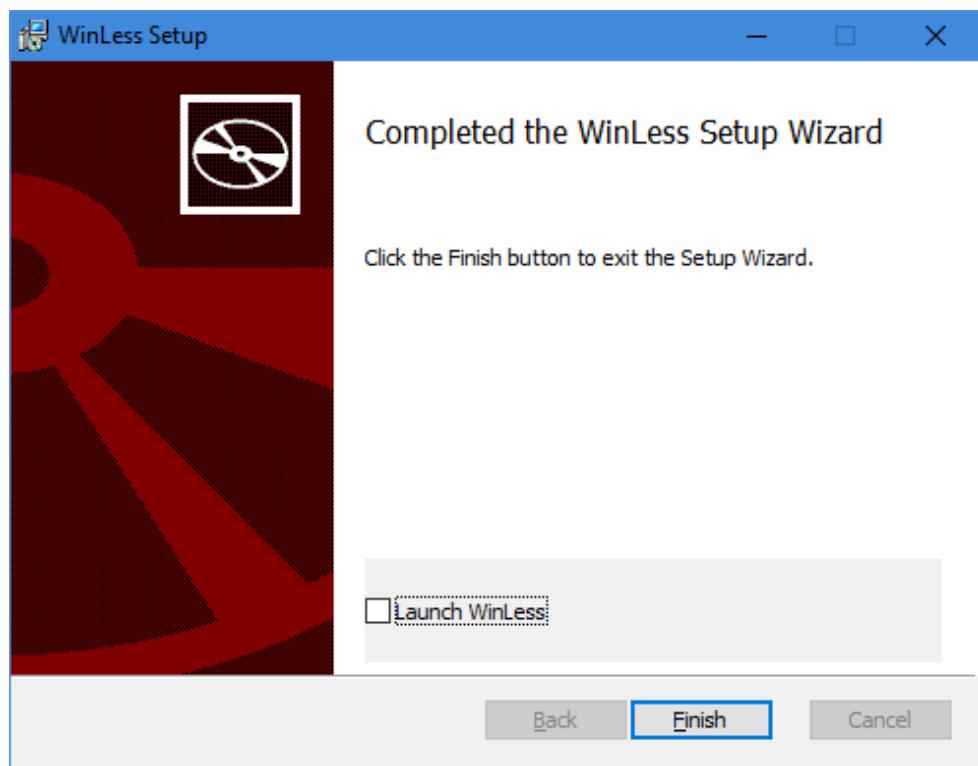
- Double click on "WinLess-1.9.1.msi".



- Check the checkbox "I accept the terms in the License Agreement".
- Click on "Install".



- Click on "Finish".



2. Creating LESS File

- Open Visual Studio Code.
- Go to "File" > "New File".
- Write the following program:



A screenshot of Visual Studio Code showing a CSS file named "Untitled-1". The code defines three CSS rules: h1, p, and span, each using variables @mycolor and @x. The code is as follows:

```
@mycolor: darkgreen;
@x: bold;

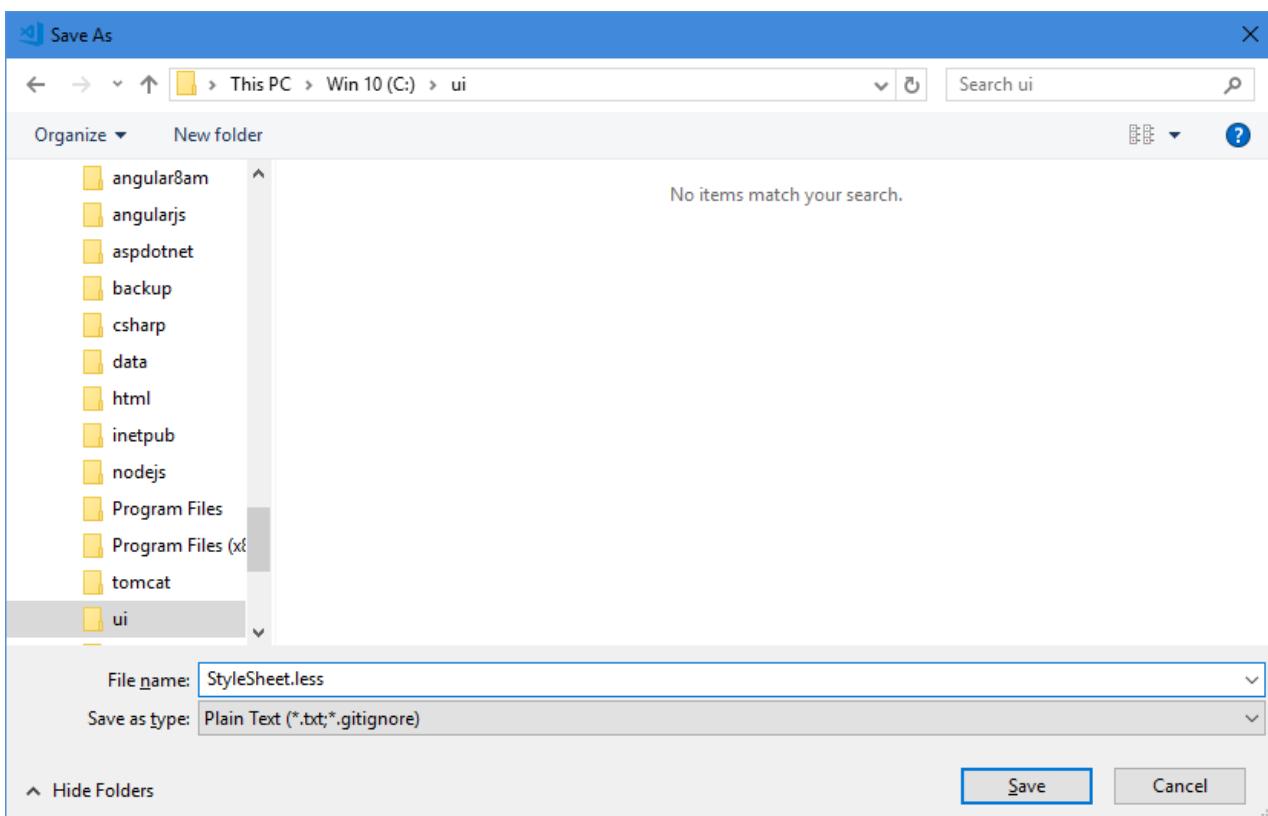
h1
{
    background-color: @mycolor;
}

p
{
    color: @mycolor;
    font-weight: @x;
}

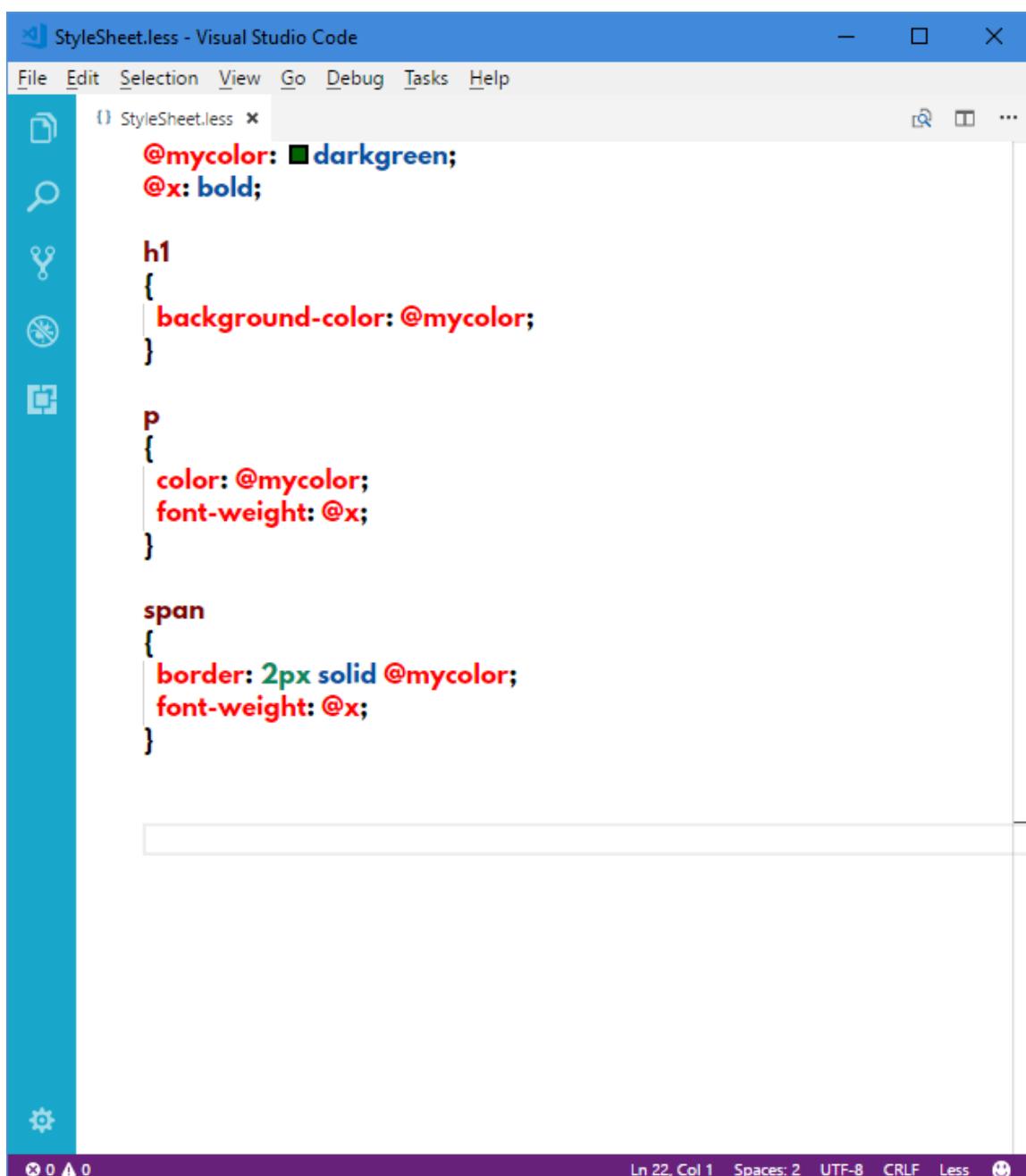
span
{
    border: 2px solid @mycolor;
    font-weight: @x;
}
```

The status bar at the bottom shows: Ln 22, Col 1 | Spaces: 2 | UTF-8 | CRLF | Plain Text | 😊

- Go to "File" menu – "Save".



- Enter filename as "StyleSheet.less".
- Select the folder "c:\ui".
- Click on "Save".



The screenshot shows a Visual Studio Code window with the title "StyleSheet.less - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Debug, Tasks, and Help. A tab bar shows "StyleSheet.less x". The code editor contains the following LESS code:

```
@mycolor: darkgreen;
@x: bold;

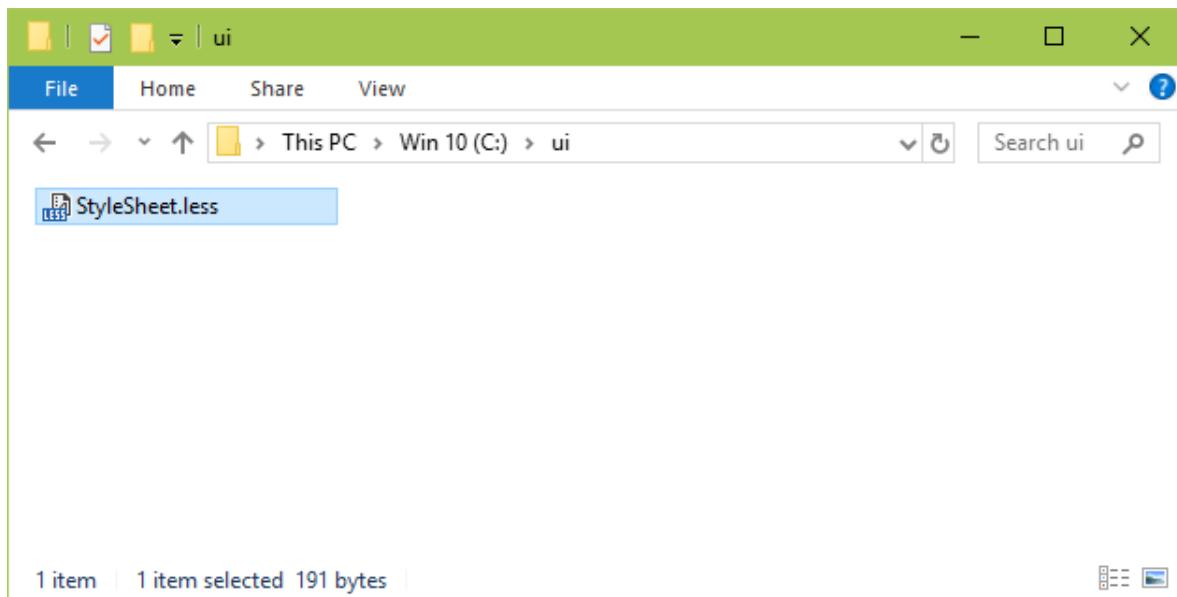
h1
{
    background-color: @mycolor;
}

p
{
    color: @mycolor;
    font-weight: @x;
}

span
{
    border: 2px solid @mycolor;
    font-weight: @x;
}
```

The LESS variables and properties are highlighted in red, while the CSS output is in black. The status bar at the bottom shows "Ln 22, Col 1" and other file details.

- Now the "c:\ui\StyleSheet.less" file is ready.

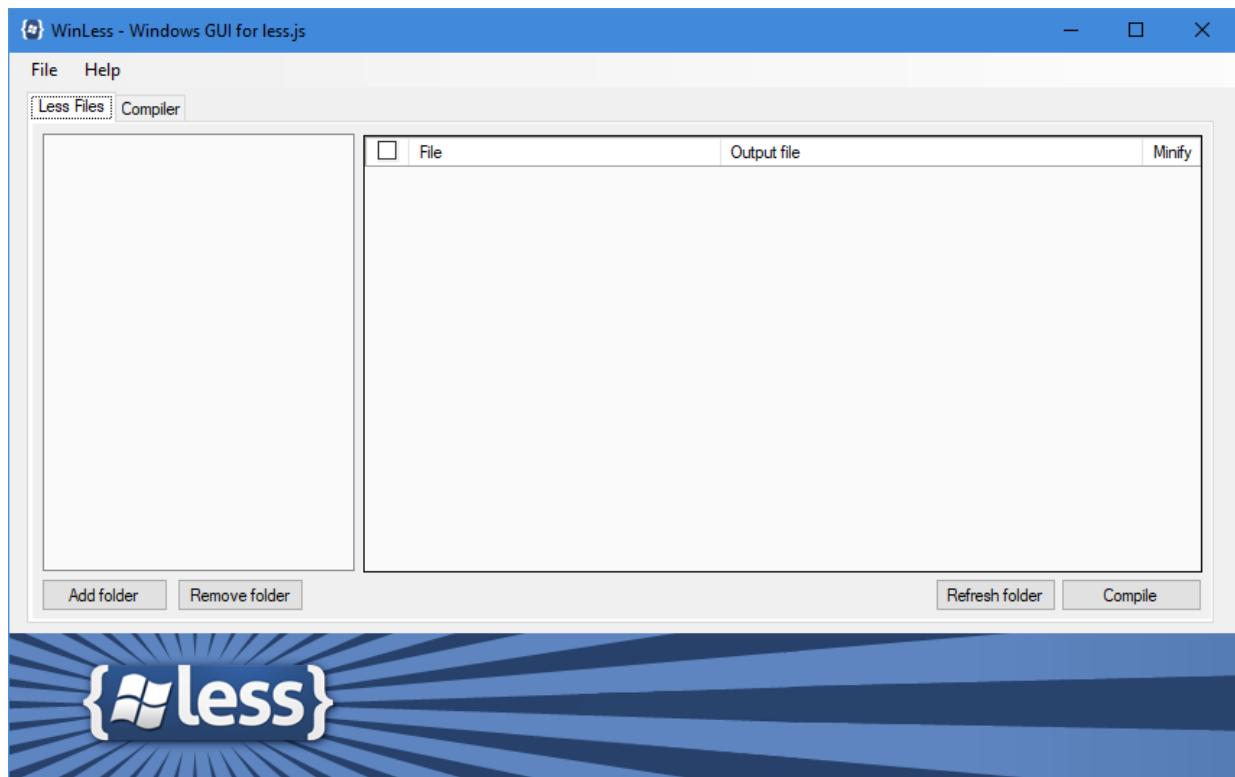


3. Compile LESS File into CSS File

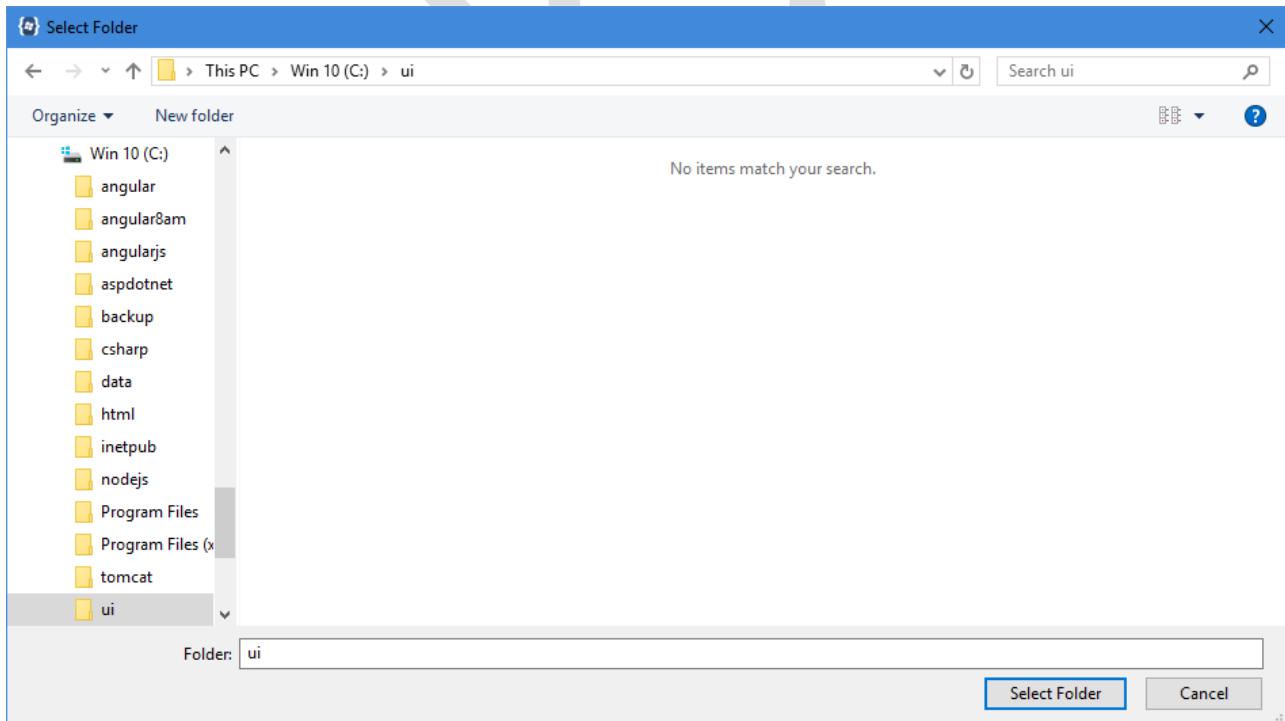
- Go to "Start" > "WinLess" > "WinLess".



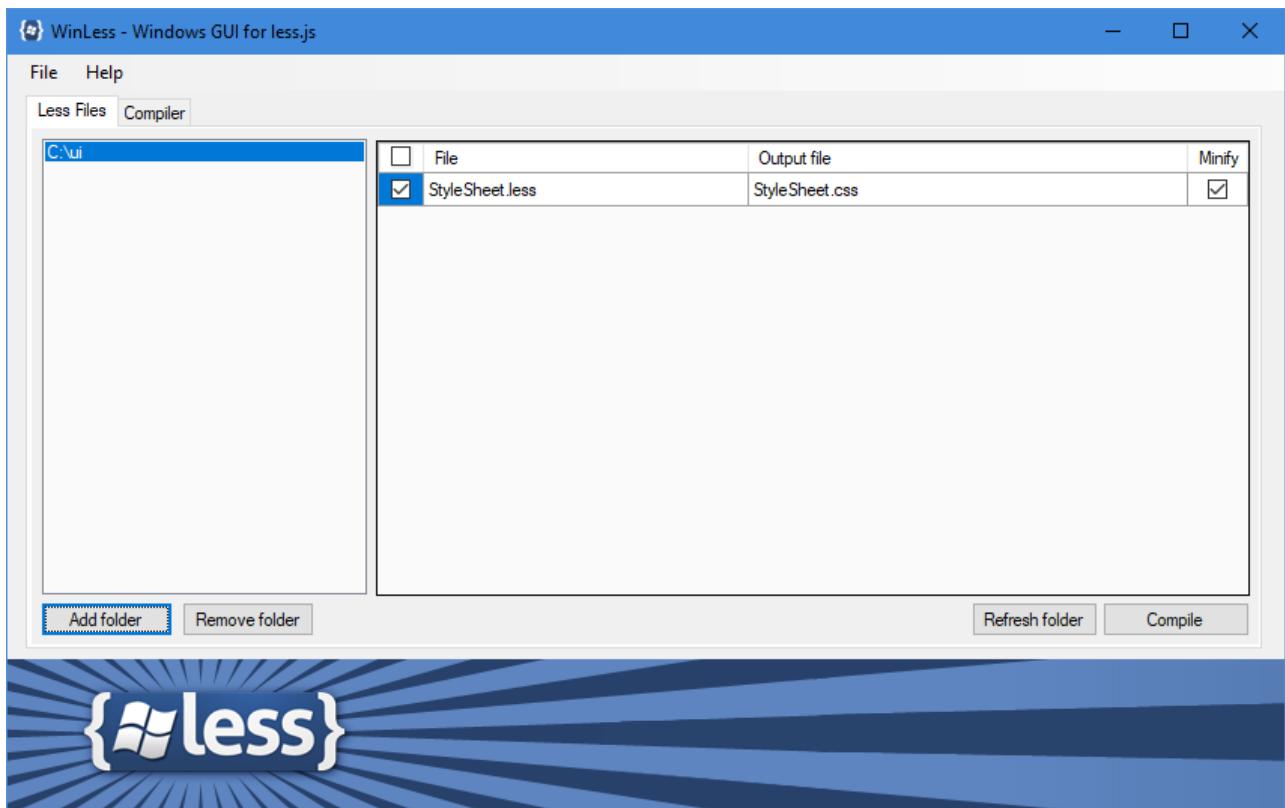
- WinLess opened.



- Click on "Add folder".

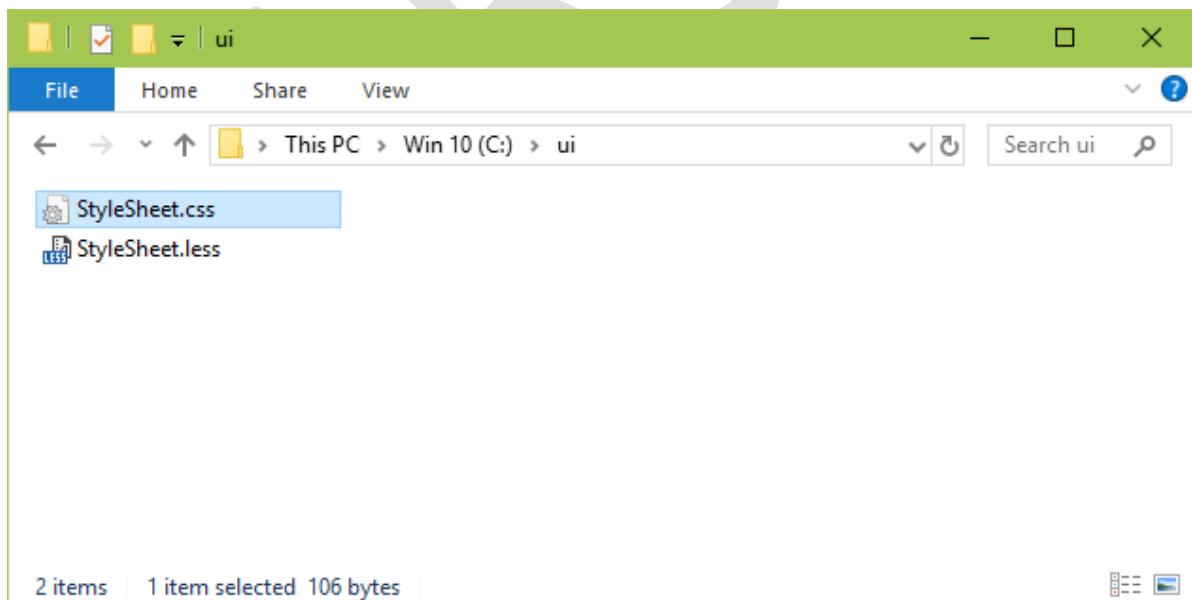


- Select "c:\ui". Click on "Select Folder".



{less}

- It shows the list of LESS files automatically. It also generates CSS filename automatically.
- Click on "Compile".



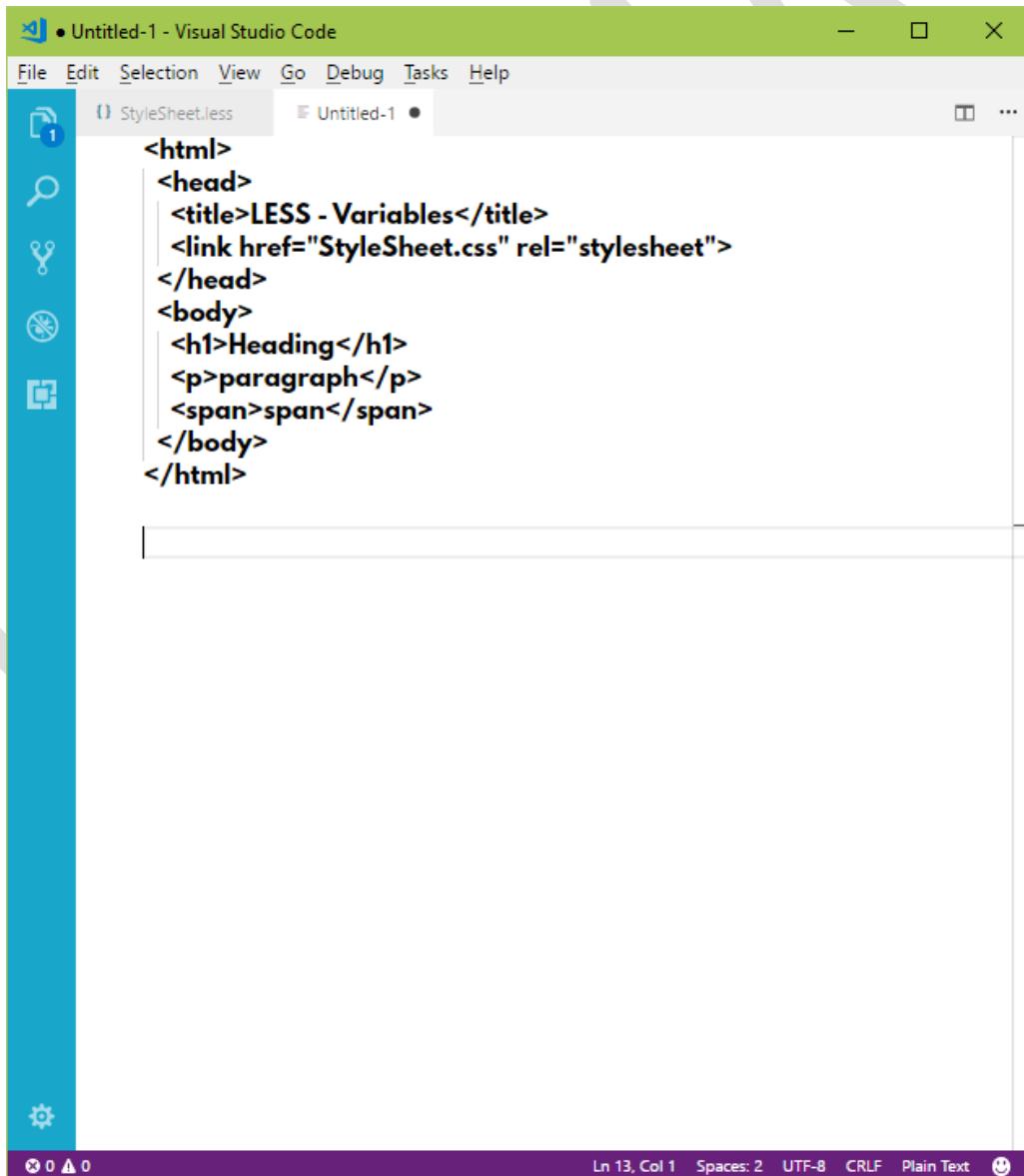
- It compiles the LESS file and generates CSS file automatically.
- The compiled CSS code (automatically generated):

```
h1{
  background-color: darkgreen;
}
```

```
p {  
    color: darkgreen;  
    font-weight: bold;  
}  
span {  
    border: 2px solid darkgreen;  
    font-weight: bold;  
}
```

4. Import the CSS File into HTML File

- Open Visual Studio Code.
- Go to "File" > "New File".
- Write the following program:

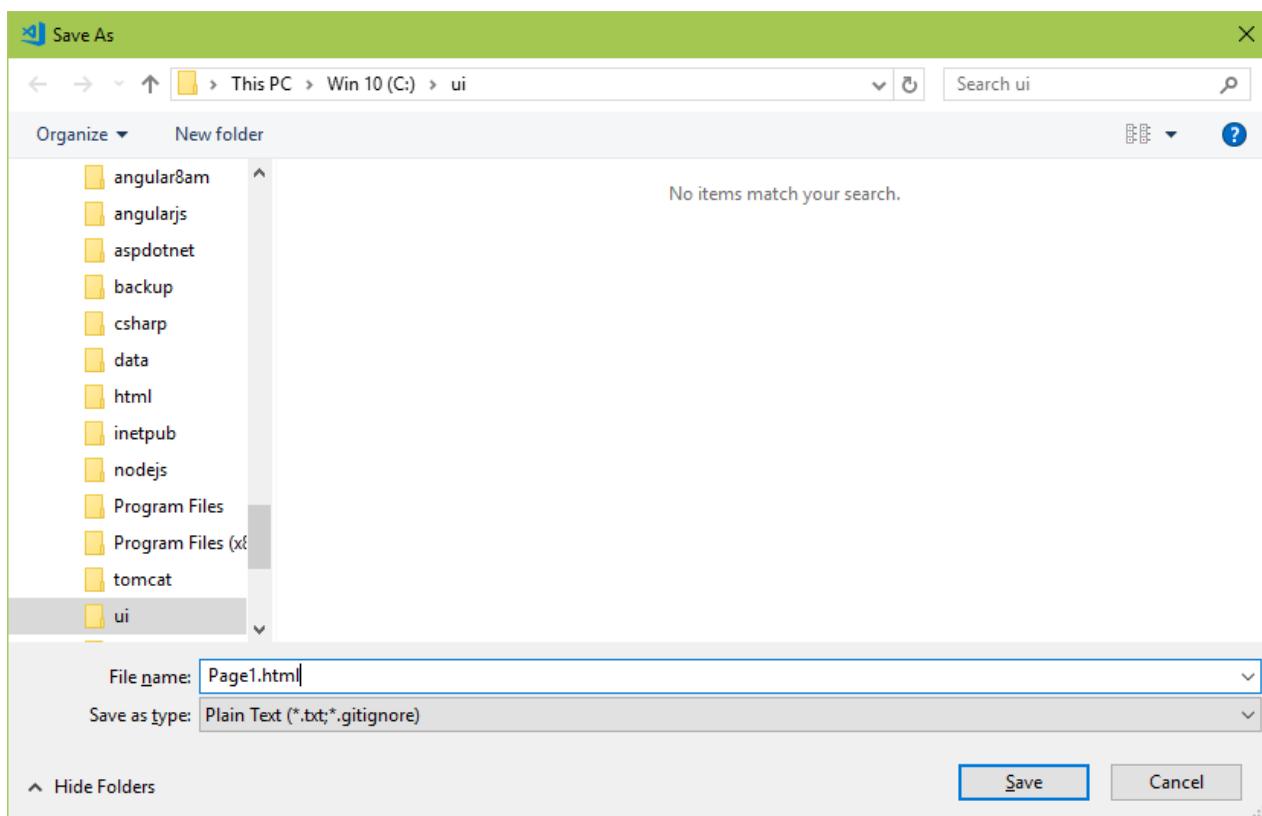


The screenshot shows the Visual Studio Code interface. The title bar says "Untitled-1 - Visual Studio Code". The left sidebar has icons for file operations. The main editor area contains the following HTML code:

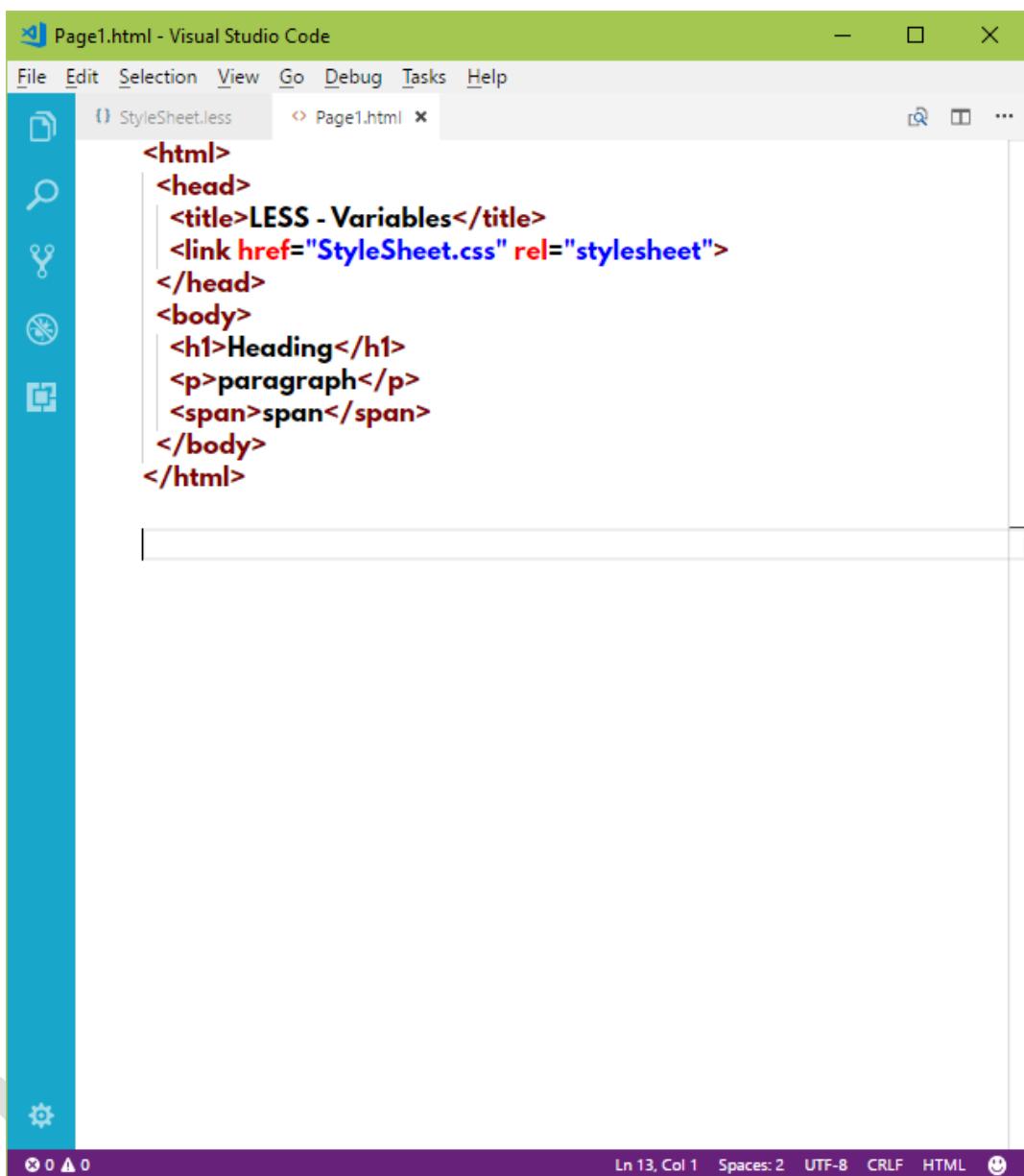
```
<html>  
  <head>  
    <title>LESS - Variables</title>  
    <link href="StyleSheet.css" rel="stylesheet">  
  </head>  
  <body>  
    <h1>Heading</h1>  
    <p>paragraph</p>  
    <span>span</span>  
  </body>  
</html>
```

The status bar at the bottom shows "Ln 13, Col 1" and other settings like "Spaces: 2", "UTF-8", "CRLF", "Plain Text".

- Go to "File" menu – "Save".



- Enter filename as "Page1.html".
- Select the folder "c:\ui".
- Click on "Save".

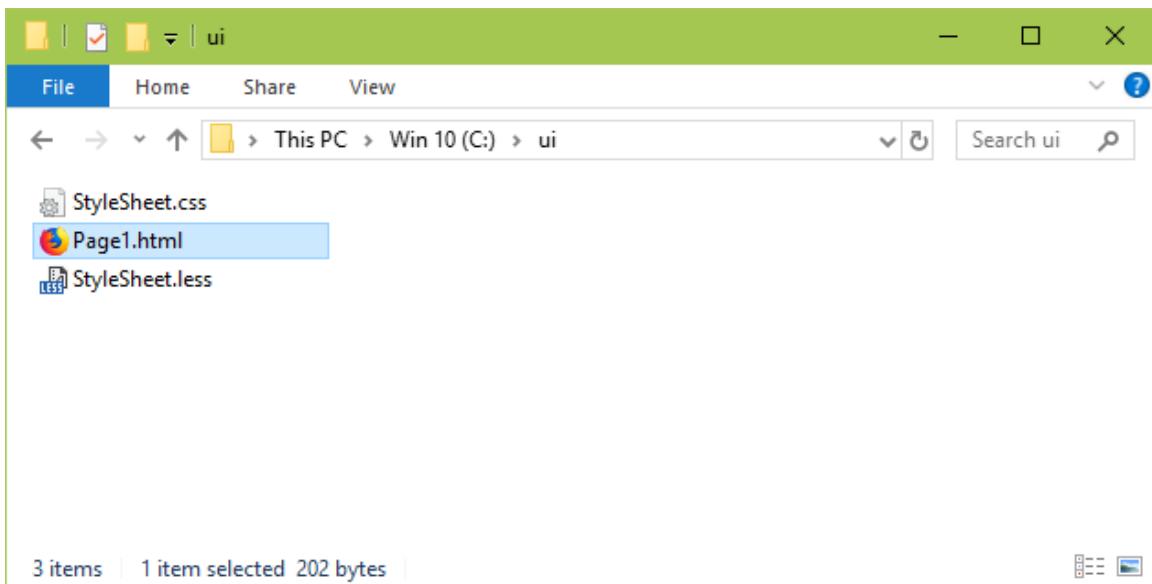


The screenshot shows the Visual Studio Code interface with the title bar "Page1.html - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Debug, Tasks, and Help. The left sidebar has icons for File, Find, Save, and others. There are two tabs open: "StyleSheet.less" and "Page1.html". The "Page1.html" tab contains the following HTML code:

```
<html>
<head>
<title>LESS - Variables</title>
<link href="StyleSheet.css" rel="stylesheet">
</head>
<body>
<h1>Heading</h1>
<p>paragraph</p>
<span>span</span>
</body>
</html>
```

The status bar at the bottom shows "Ln 13, Col 1" and "Spaces: 2" and "UTF-8" and "CRLF" and "HTML".

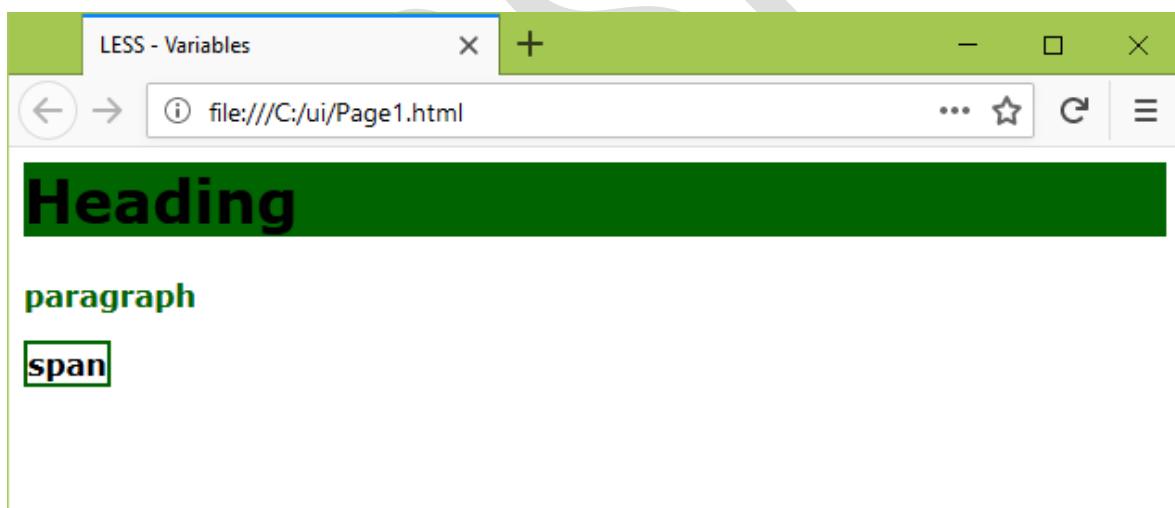
- Now the "c:\ui\Page1.html" file is ready.



5. Run the HTML File

- Go to "c:\ui" folder and double click on "Page1.html".

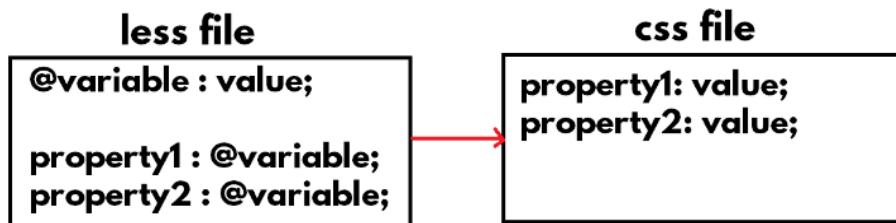
Output:



Basic LESS

Variables

- In LESS, variables are used to store a value.
- The variable create once, can be used many times within the same less file. While converting into CSS, the variables will be replaced with actual value.
- Variable names must be started with "@".



Mixins

- Mixins are used to mix a CSS class with another style.

Syntax (LESS File):

```

.classname
{
    property1: value;
    property2: value;
}
selector
{
    .classname;
    property3: value;
    property3: value;
}

```

Result (CSS File):

```

.classname
{
    property1: value;
    property2: value;
}
selector
{
    property1: value;
    property2: value;
    property3: value;
    property3: value;
}

```

```
}
```

Example on Mixins

c:\ui\StyleSheet.less

```
.class1
{
    background-color: #006633;
    border: 3px solid #ff3333;
    padding: 10px;
    font-family: 'Segoe UI';
}

h1
{
    .class1;
    color: #0066ff;
}

p
{
    .class1;
    color: #ffff99;
}

span
{
    .class1;
    color: #ccffcc;
}
```

c:\ui\Page1.html

```
<html>
    <head>
        <title>LESS - Mixins</title>
        <link href="StyleSheet.css" rel="stylesheet">
    </head>
    <body>
        <h1>Heading</h1>
        <p>paragraph</p>
        <span>span</span>
    </body>
</html>
```

Mixins With Parameters

- Mixins can receive one or more parameters (arguments), and utilize those values into the styles. Everytime when you call the mixin, you can pass different values to the mixin.

Syntax (LESS File):

```
.classname(@parameter1, @parameter2, ...)
{
  property1: value;
  property2: value;
}
```

Example on Mixins with Parameters**c:\ui\StyleSheet.less**

```
.class1(@borderwidth:3px)
{
  background-color: #006633;
  border: @borderwidth solid #ff3333;
  padding: 10px;
  font-family: 'Segoe UI';
}

h1
{
  .class1(5px);
  color: #0066ff;
}

p
{
  .class1(10px);
  color: #ffff99;
}

span
{
  .class1;
  color: #ccffcc;
}
```

c:\ui\Page1.html

```
<html>
  <head>
    <title>LESS - Mixins</title>
    <link href="StyleSheet.css" rel="stylesheet">
  </head>
  <body>
    <h1>Heading</h1>
    <p>paragraph</p>
    <span>span</span>
  </body>
```

</html>

Nested Rules

- These are used to apply CSS styles for the child elements of a specific parent tag.

Syntax (LESS File):

```
parenttag  
{  
    child1  
    {  
        styles here  
    }  
    child2  
    {  
        styles here  
    }  
}
```

Result (CSS File):

```
parenttag child1  
{  
}  
  
parenttag child2  
{  
}
```

Example on Nested Rules

```
c:\ui\StyleSheet.less  
#div1  
{  
  
    h1  
    {  
        background-color: #0099ff;  
    }  
  
    p
```

```

{
    background-color: #99ff99;
}

span
{
    background-color: #ff3399;
}

}

```

c:\ui\Page1.html

```

<html>
    <head>
        <title>LESS - Nested Rules</title>
        <link href="StyleSheet.css" rel="stylesheet">
    </head>
    <body>
        <div id="div1">
            <h1>Heading</h1>
            <p>paragraph</p>
            <span>span</span>
        </div>
    </body>
</html>

```

Advanced LESS**Operators**

- In LESS, we can use all the arithmetical operators such as +, -, *, / etc.
- The LESS compiler calculates the operators and generates the result.

Syntax (LESS File):

```

selector
{
    property: a+b;
}

```

Result (CSS File):

```

selector
{
    property: sum;
}

```

```
}
```

Example on Operators

c:\ui\StyleSheet.less

```
@defaultFontSize: 40px;

h1
{
    background-color: #0099ff;
    font-size: @defaultFontSize * 2;
}

p
{
    background-color: #99ff99;
    font-size: @defaultFontSize;
}

span
{
    background-color: #ff3399;
    font-size: @defaultFontSize + 2;
}
```

c:\ui\Page1.html

```
<html>
  <head>
    <title>LESS - Operators</title>
    <link href="StyleSheet.css" rel="stylesheet">
  </head>
  <body>
    <div>
      <h1>Heading</h1>
      <p>paragraph</p>
      <span>span</span>
    </div>
  </body>
</html>
```

Color Functions

- These are used to make the color darker / lighter.

Syntax (LESS File):

darken(color, percentage);

lighten(color, percentage);

Result (CSS File):

```
#ffffff  
#000000
```

Example on Operators

c:\ui\StyleSheet.less

```
@baseColor: #0033cc;  
  
h1  
{  
    background-color: darken(@baseColor,10%);  
}  
  
p  
{  
    background-color: @baseColor;  
}  
  
span  
{  
    background-color: lighten(@baseColor,10%);  
}
```

c:\ui\Page1.html

```
<html>  
  <head>  
    <title>LESS - Color Functions</title>  
    <link href="StyleSheet.css" rel="stylesheet">  
  </head>  
  <body>  
    <div>  
      <h1>Heading</h1>  
      <p>paragraph</p>  
      <span>span</span>  
    </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.
- Its 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);

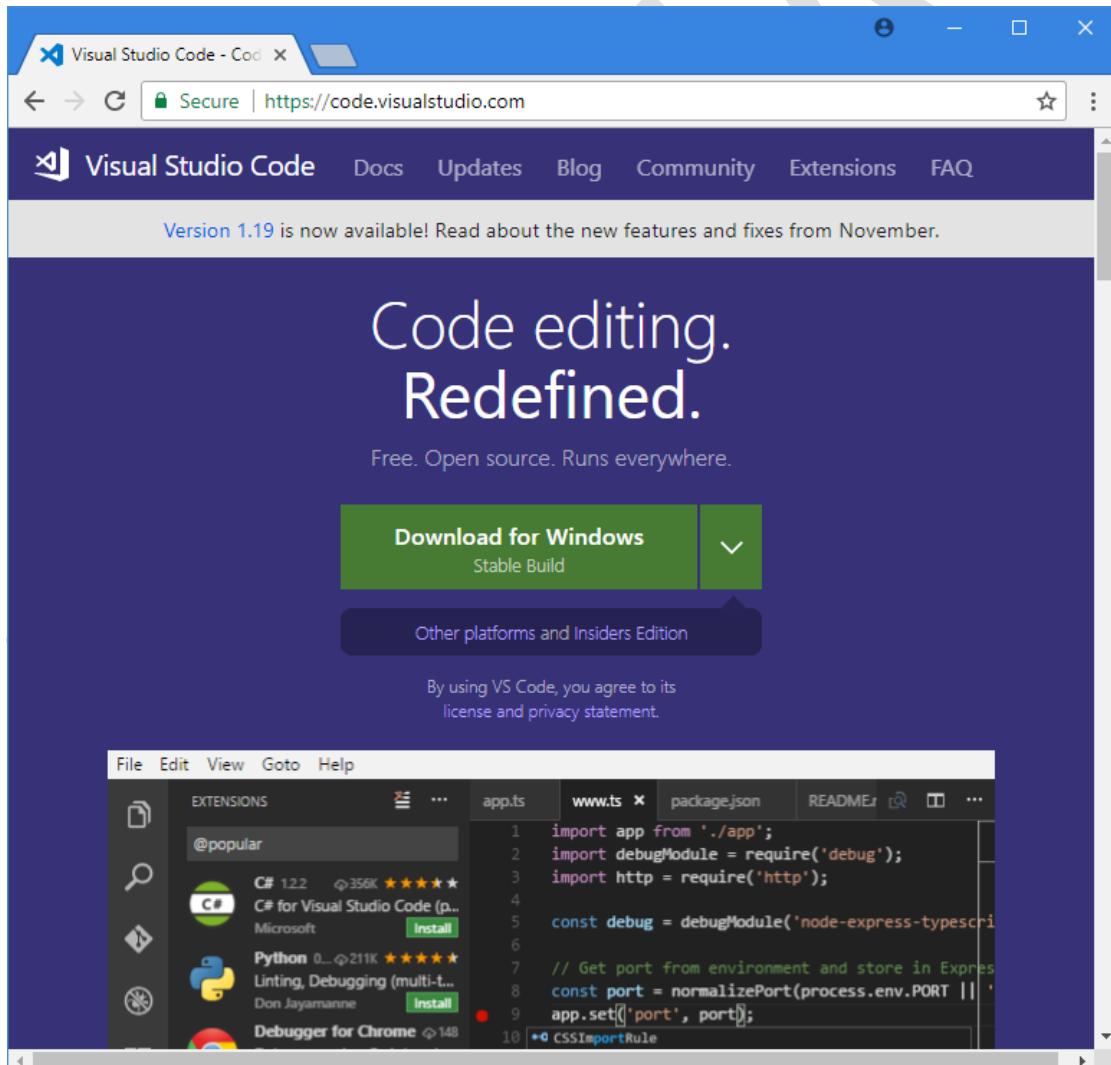
Steps to Prepare First Example in JavaScript

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

1. Installing Visual Studio Code

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

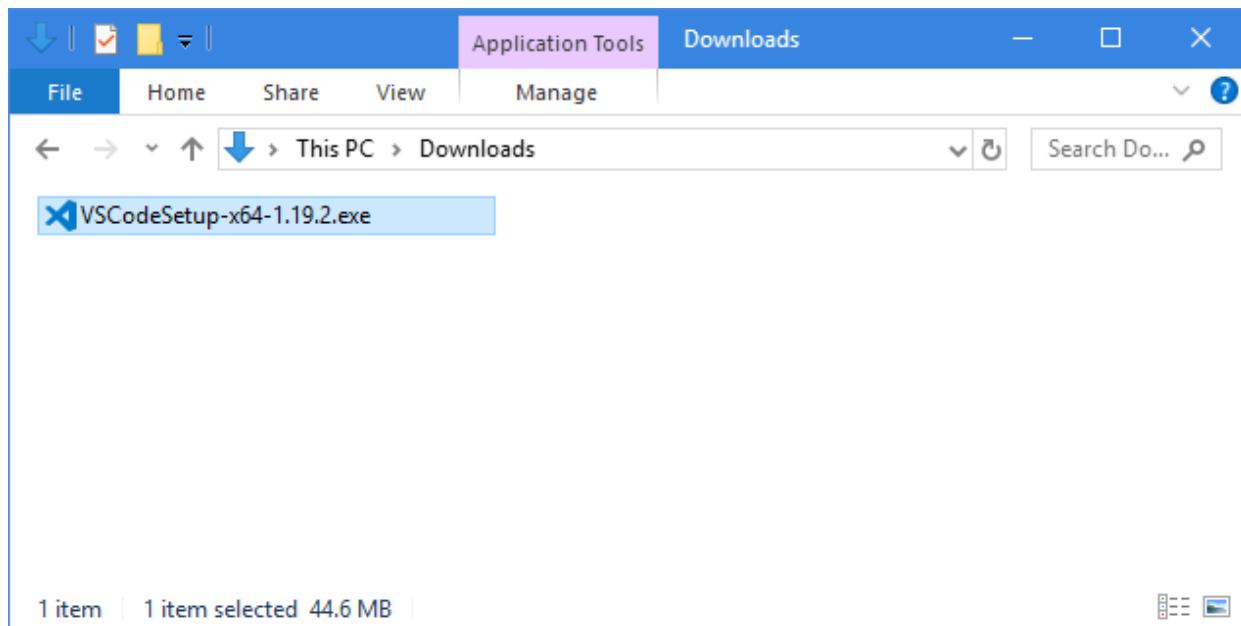
Go to <https://code.visualstudio.com>



Click on “Download for Windows”.

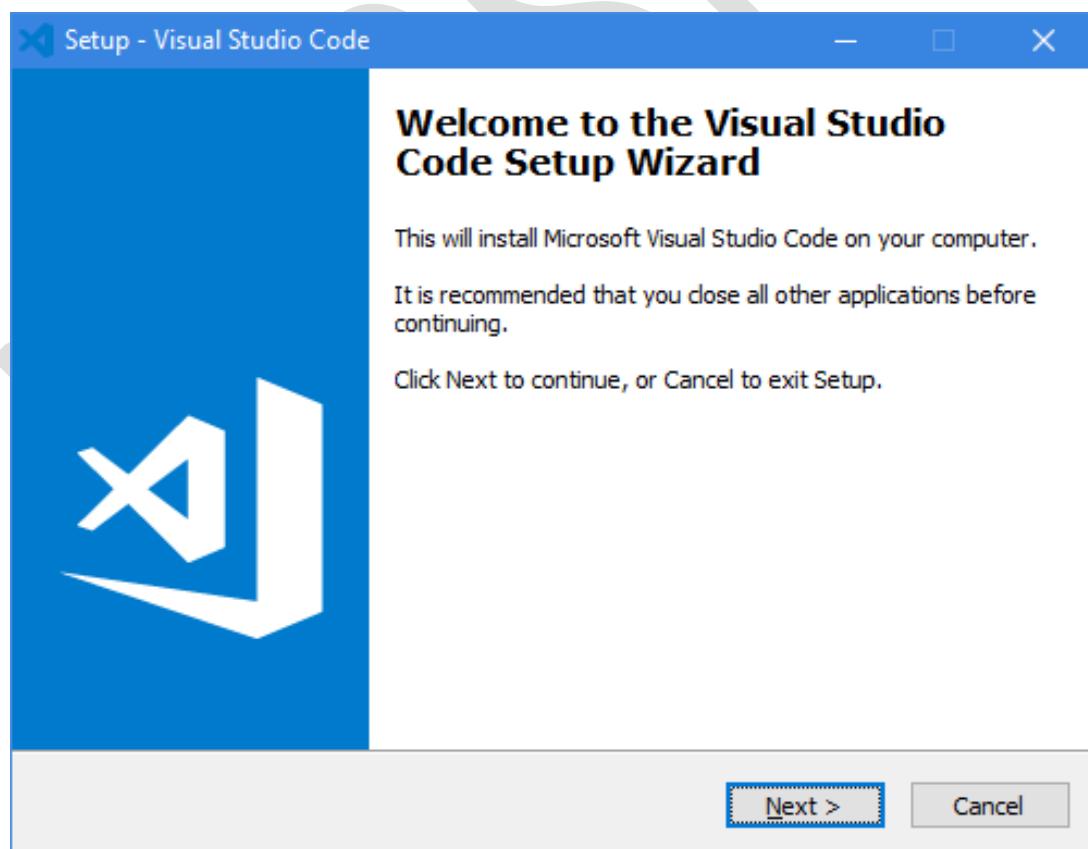
Note: The version number may be different at your practice time.

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

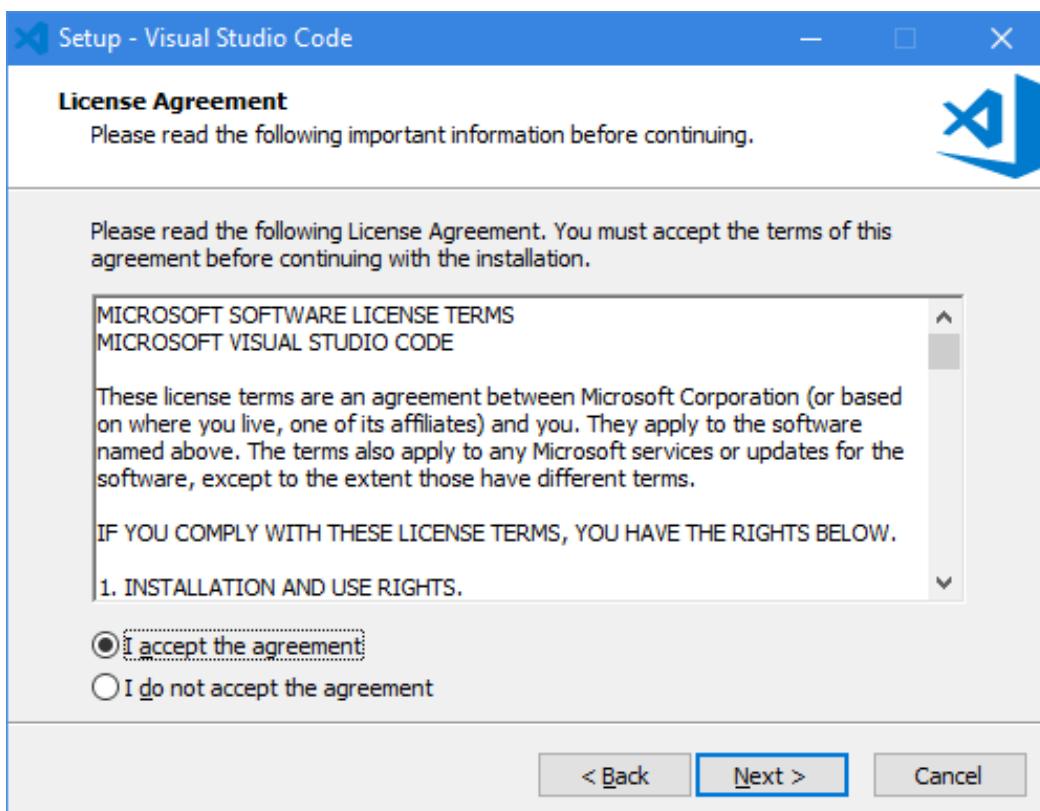


Double click on “VSCodeSetup-x64-1.19.2.exe” file.

Click on “Yes”.

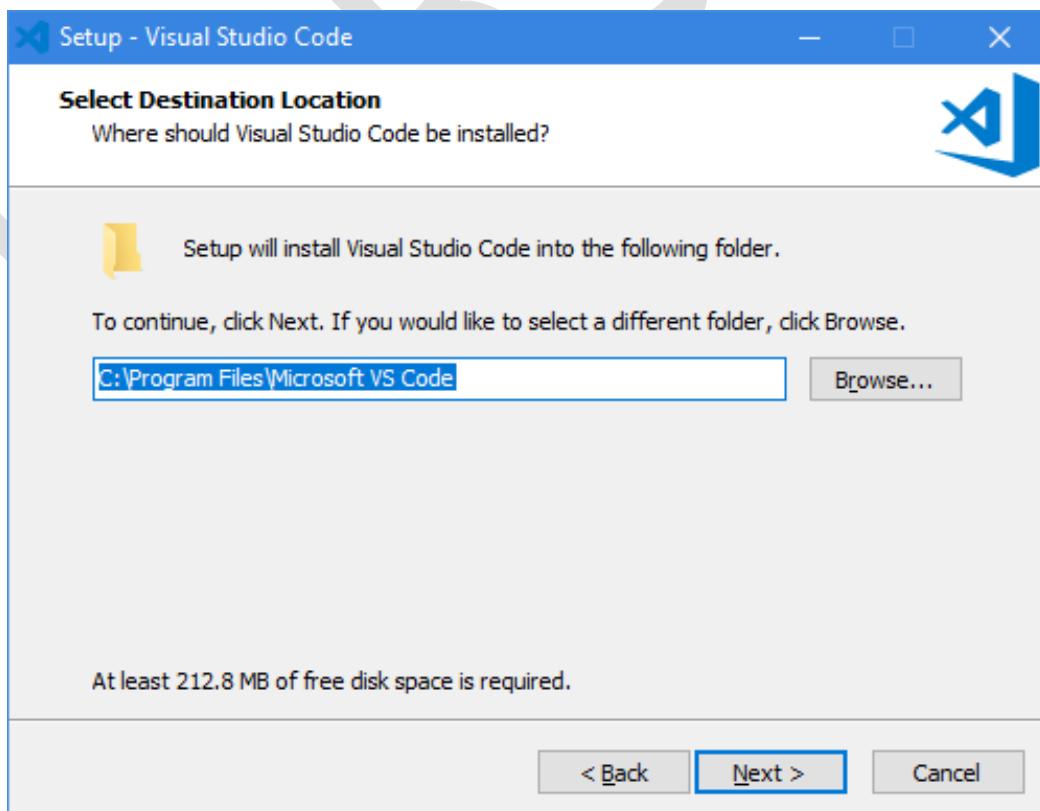


Click on “Next”.

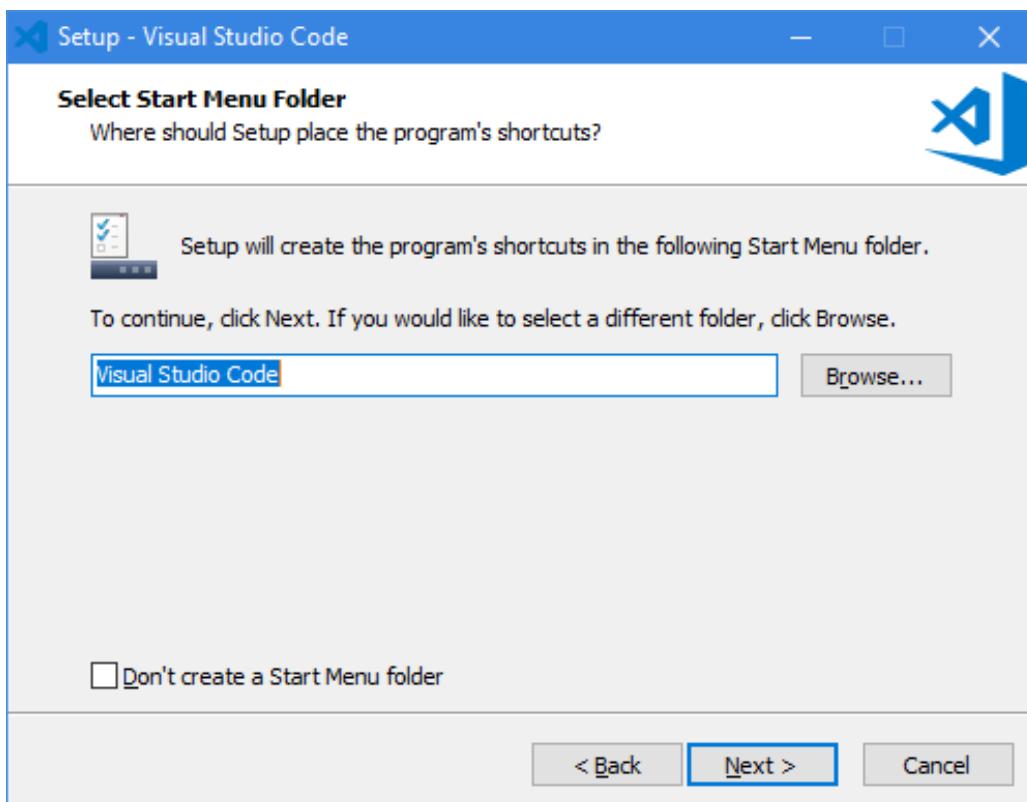


Click on "I accept the agreement".

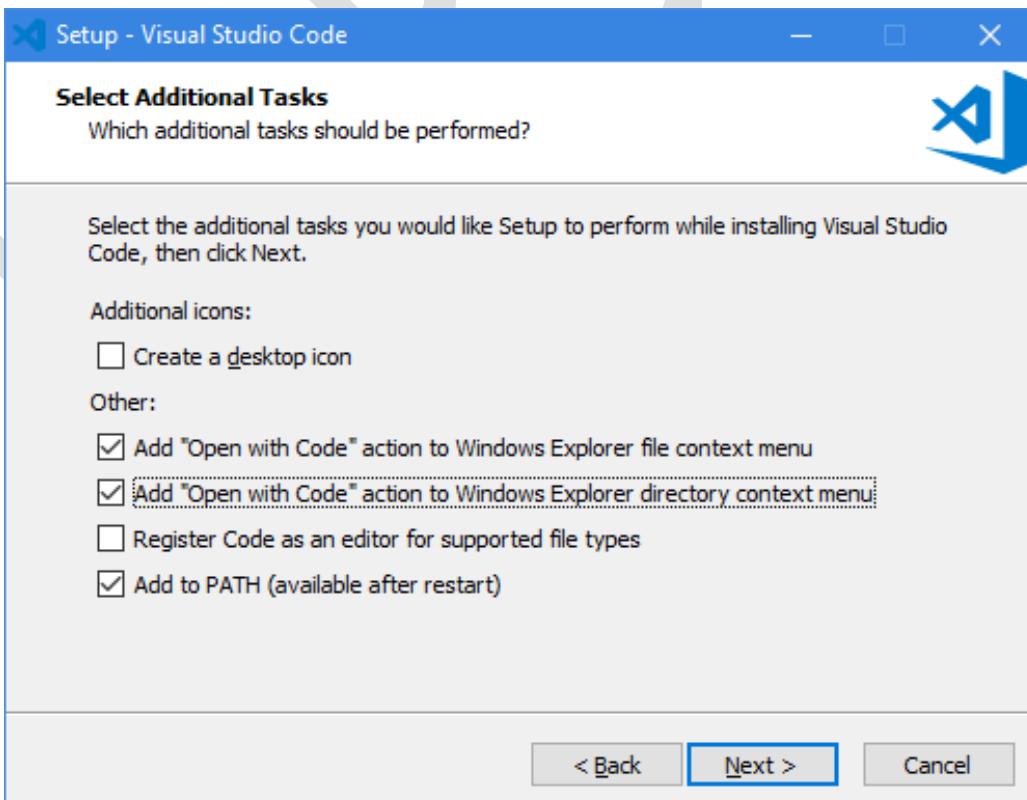
Click on "Next".



Click on “Next”.



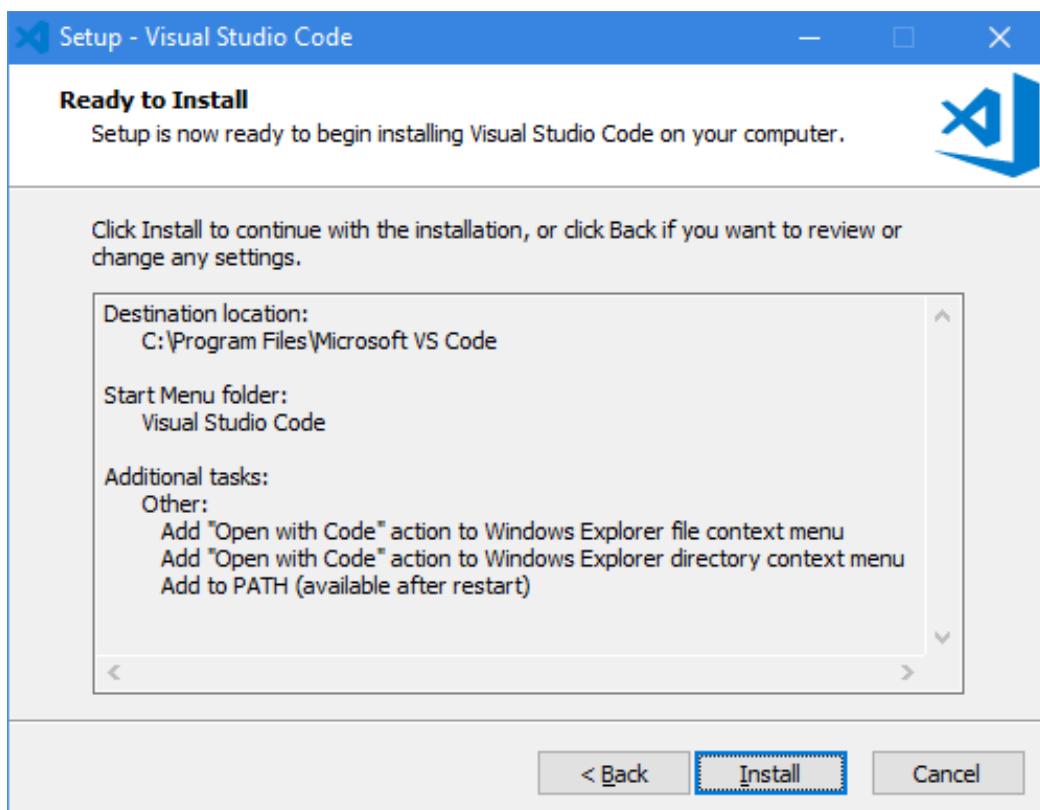
Click on “Next”.



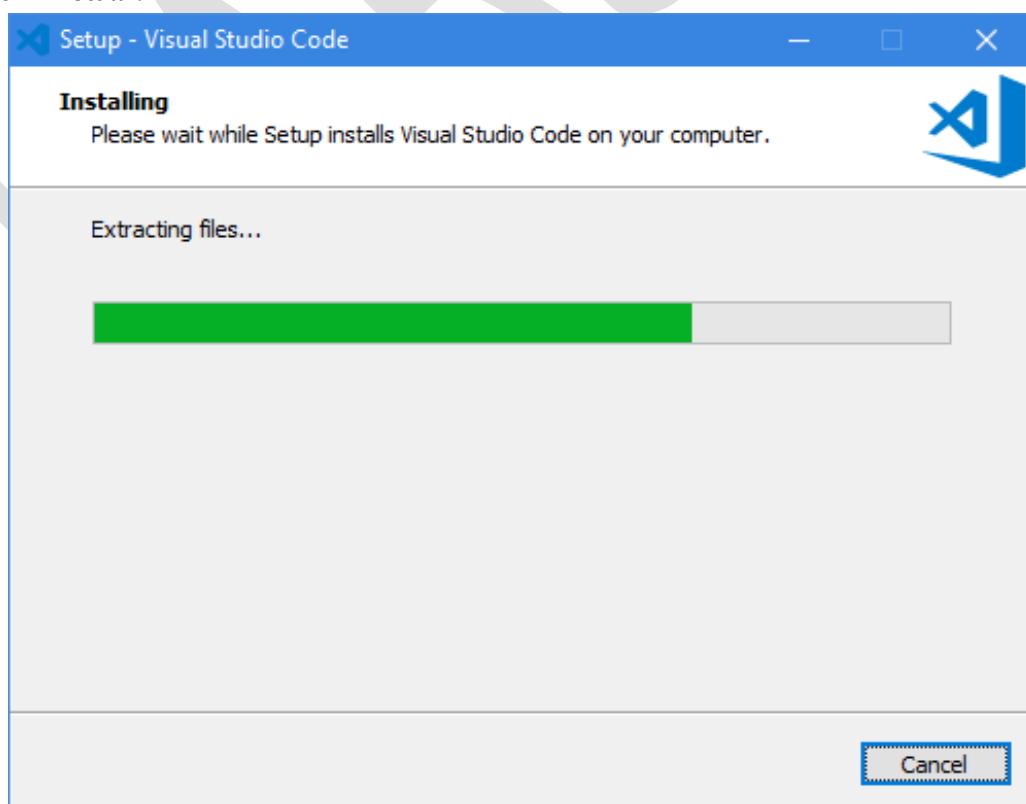
Check the checkbox “Add Open with Code action to Windows Explorer file context menu”.

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

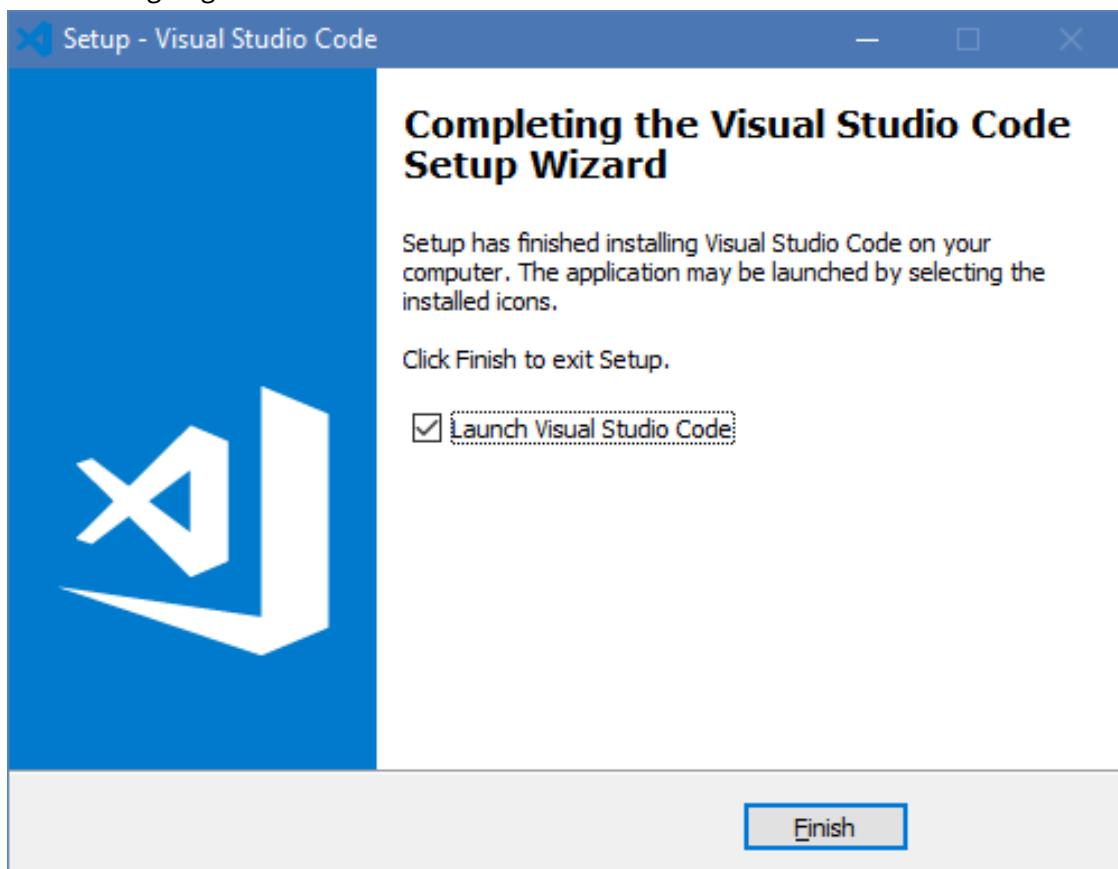
Click on “Next”.



Click on “Install”.



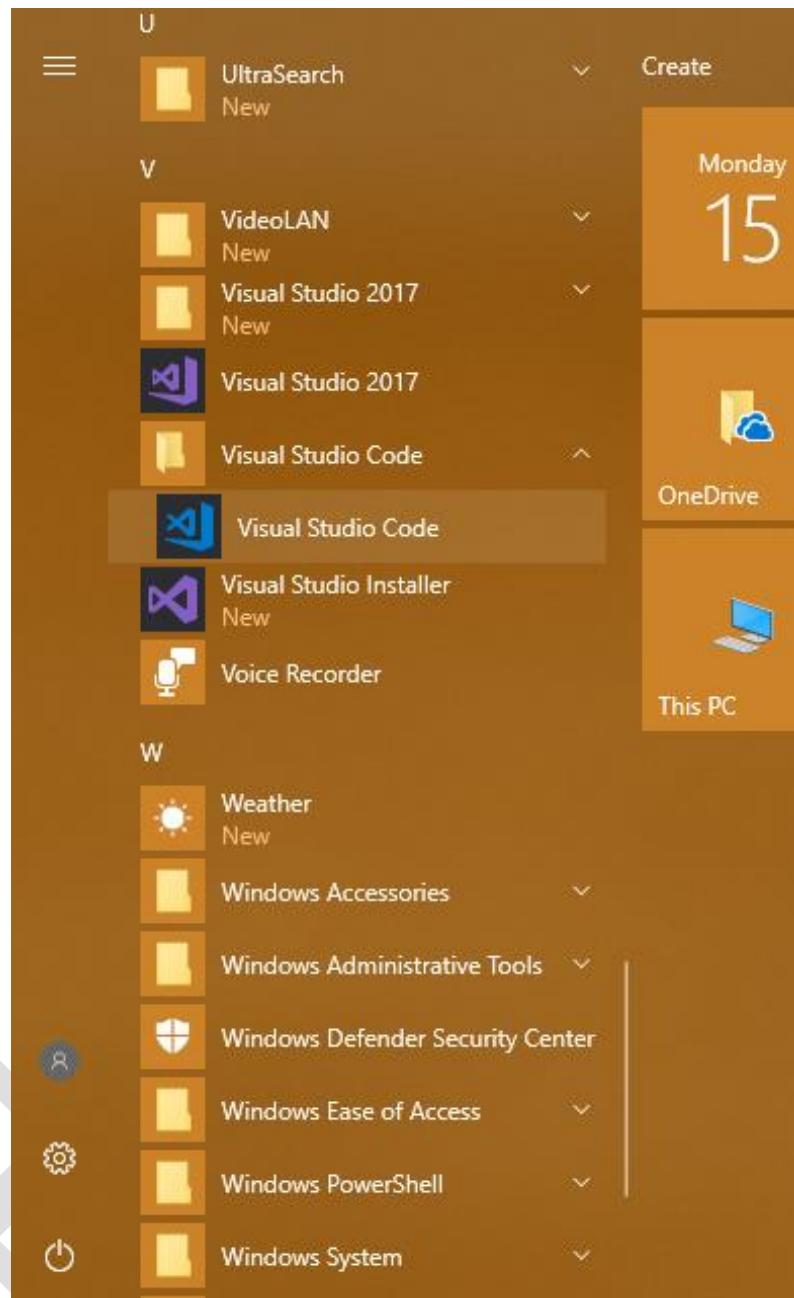
Installation is going on....

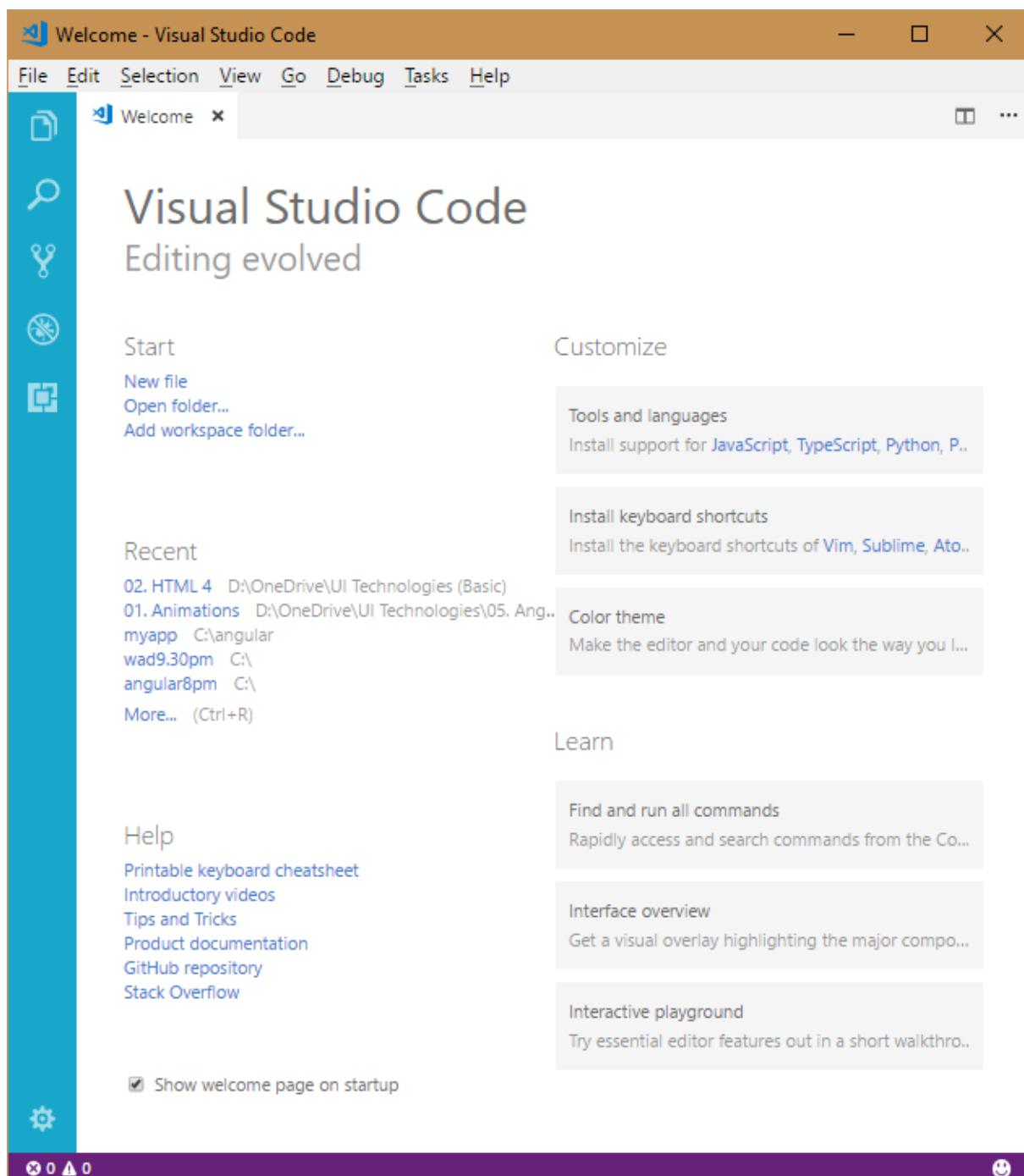


Click on “Finish”.

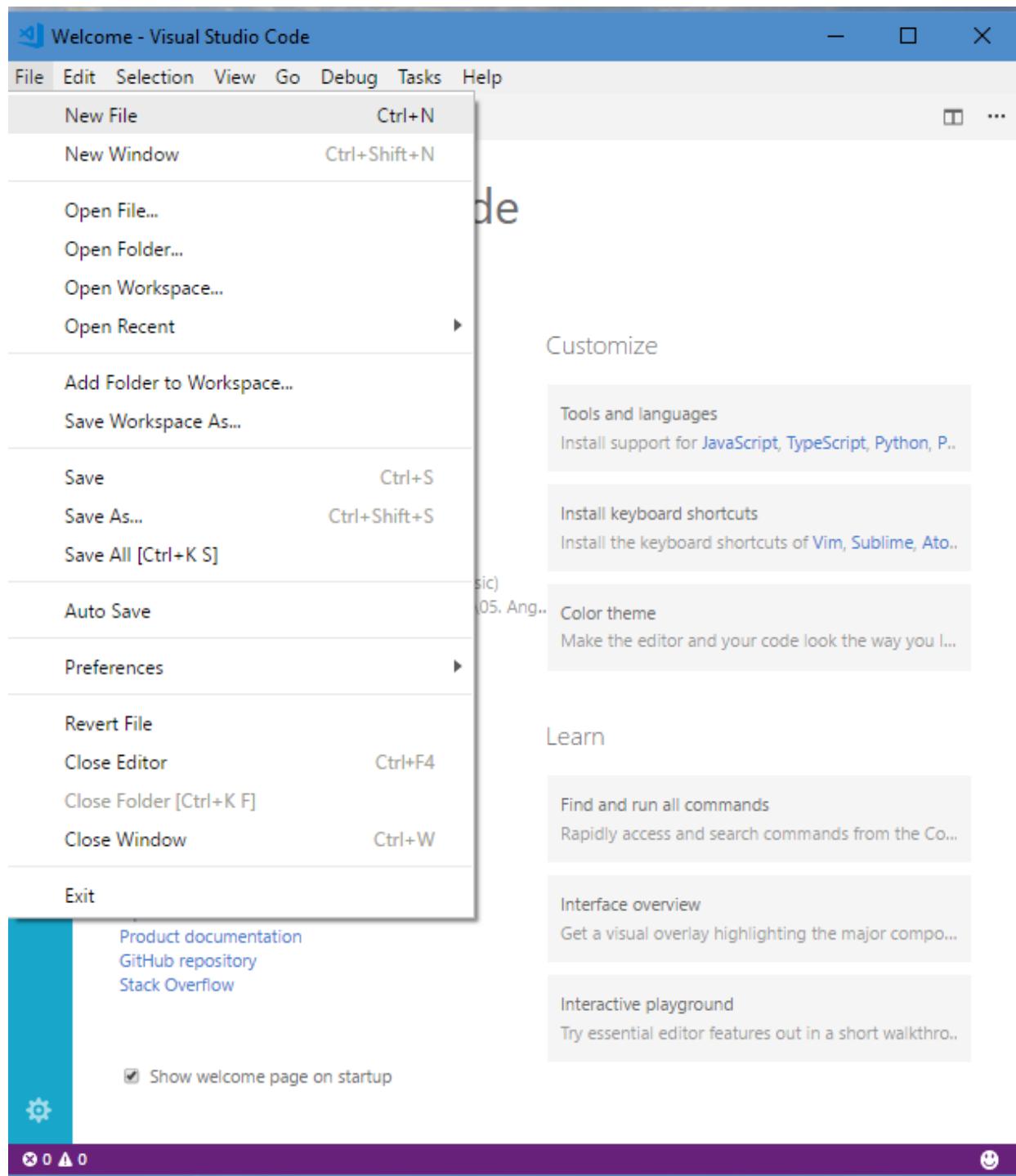
2. Create JavaScript Program

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

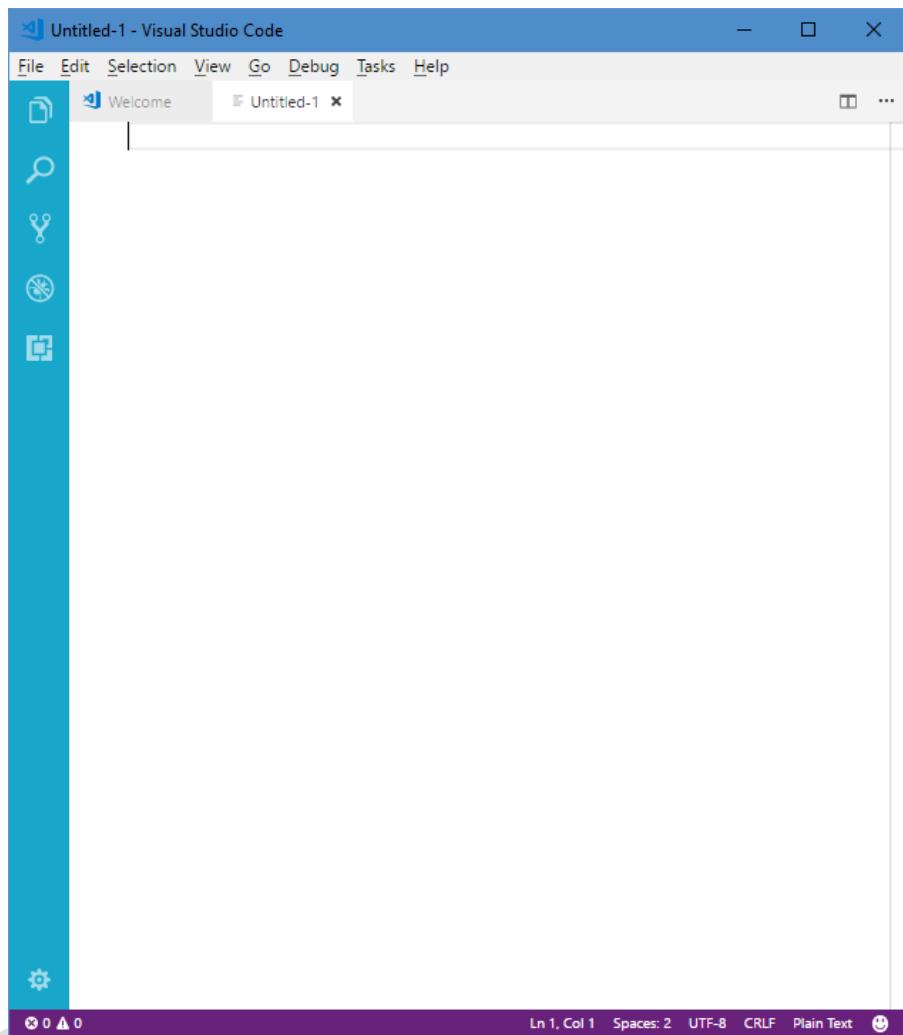




- Visual Studio Code opened.



- Go to “File” – “New File”.



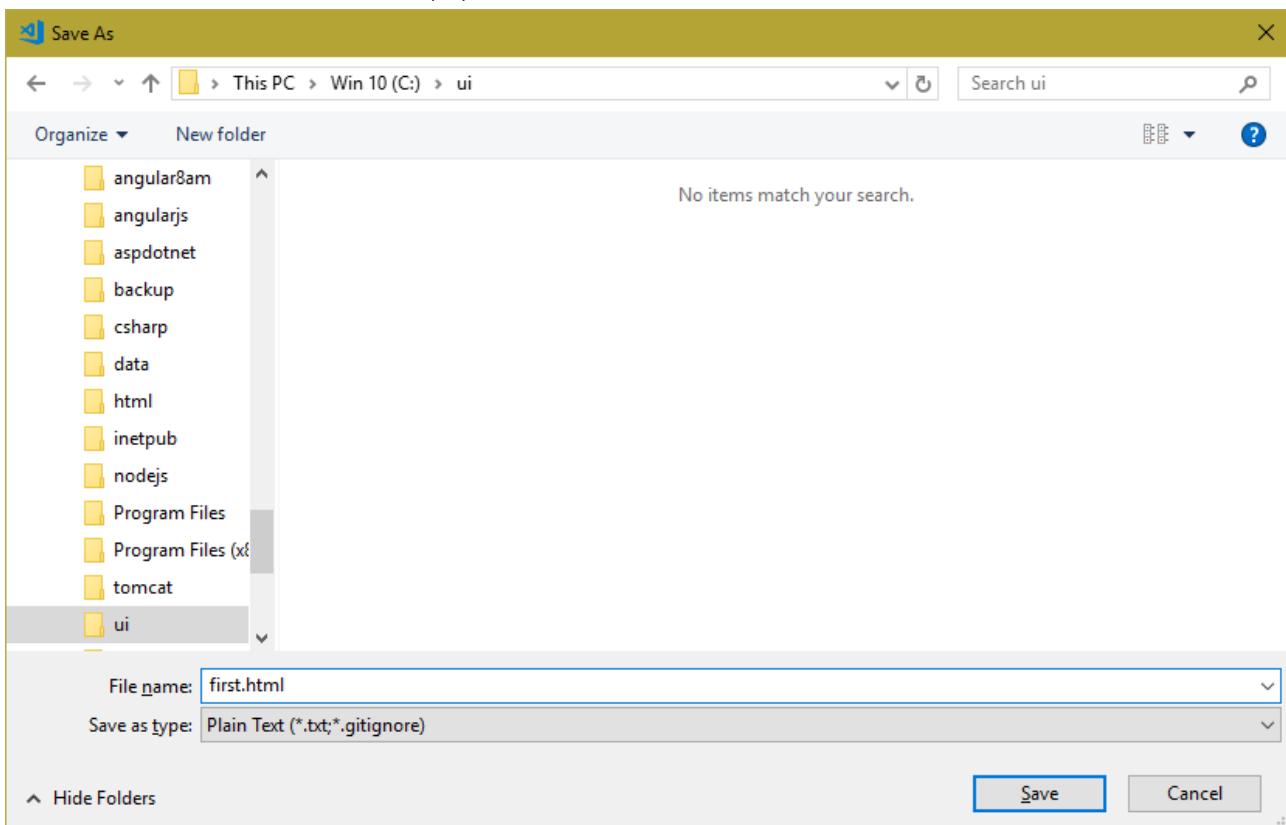
Type the program as follows:

A screenshot of the Visual Studio Code interface showing a completed HTML script. The title bar reads "Untitled-1 - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Debug, Tasks, and Help. The left sidebar has icons for file operations like Open, Save, Find, and Refresh. A tab bar shows "Welcome" and "Untitled-1". The main editor area contains the following HTML code:

```
<html>
  <head>
    <title>JavaScript</title>
  </head>
  <body>
    <h1>First Example</h1>
    <script>
      console.log("Hello World");
    </script>
  </body>
</html>
```

The status bar at the bottom shows "Ln 13, Col 1" and "Plain Text".

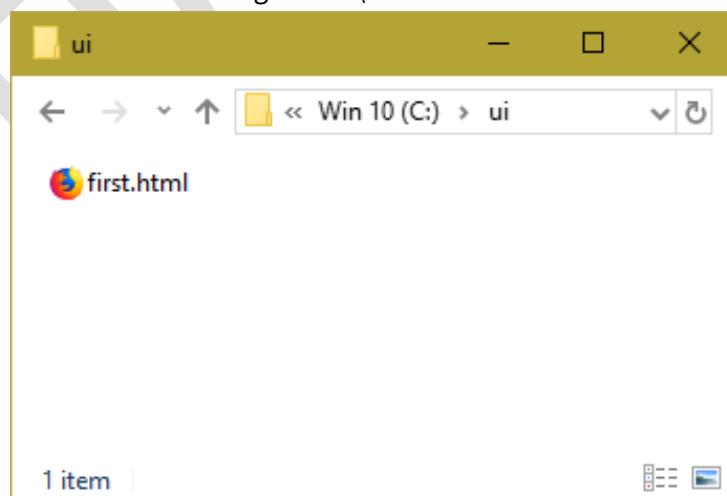
- Go to “File” menu – “Save” (or) Press Ctrl+S.



- Go to "c:\\" and click on "New folder". Enter the new folder name as "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.

```
first.html:8:13
```

The character encoding of the HTML document was not declared. The document will render with garbled text in some browser configurations if the document contains characters from outside the US-ASCII range. The character encoding of the page must be declared in the document or in the transfer protocol.

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
    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>
  </body>
</html>

```

```

<script>
  var x = 100;
  var y = 200;
  var temp1, temp2, temp3, temp4, temp5, temp6;
  temp1 = (x == y);
  console.log(temp1); //false
  temp2 = (x != y);
  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:
 1. Conditional Control Statements: Used to jump forward.
 2. Looping Control Statements: Used to jump backward.

1. Conditional Control Statements

- If
- Switch-case

2. Looping Control Statements

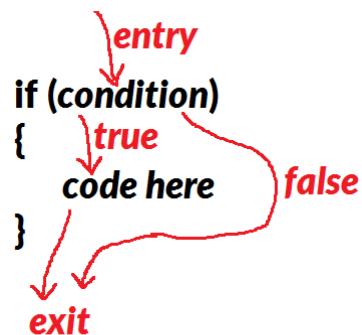
- While
- Do-while
- For

if

- “If” statement is used to check a condition, and execute the code only if the condition is TRUE.

- Types of “if”
 1. If
 2. If-else
 3. Else-if
 4. Nested if

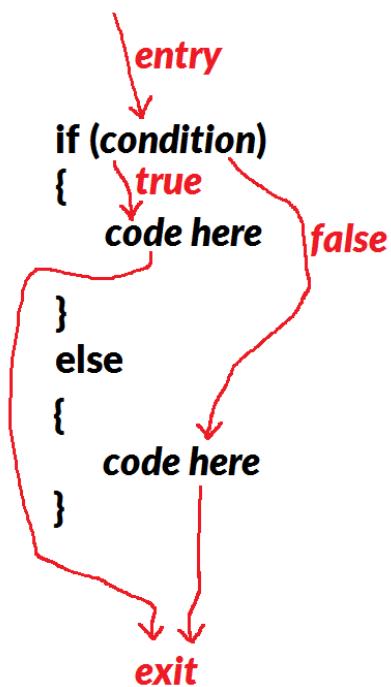
Simple If



Example of “Simple if”

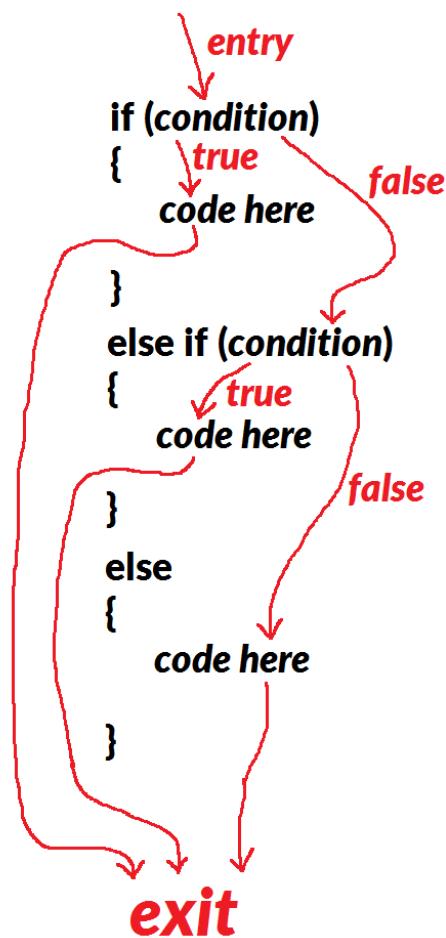
```
<html>
  <head>
    <title>If</title>
  </head>
  <body>
    <h1>If</h1>
    <script>
      var n = 10;
      if (n == 10)
      {
        console.log("n is equal to 10");
      }
      if (n != 10)
      {
        console.log("n is not equal to 10");
      }
    </script>
  </body>
</html>
```

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>
```

Else if**Example of "else-if"**

```

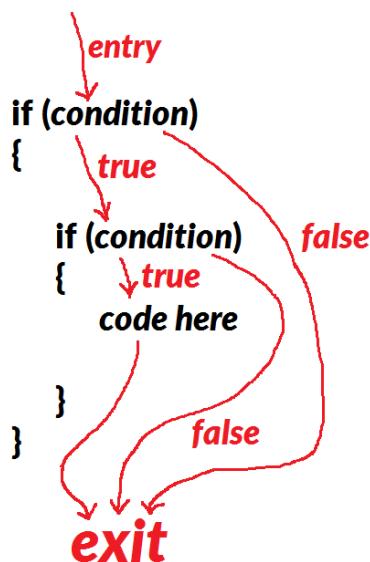
<html>
  <head>
    <title>else-if</title>
  </head>
  <body>
    <h1>else-if</h1>
    <script>
      var a = 10, b = 20;
      if(a < b)
      {
        console.log("a is less than b");
      }
      else if (a > b)
      {
        console.log("a is greater than b");
      }
      else
      {
        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:

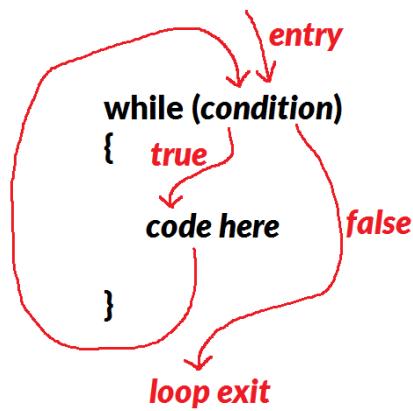
```
switch (variable)
{
    case value1: code here ; break;
    case value2: code here ; break;
    ...
    default: code here ; break;
}
```

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.



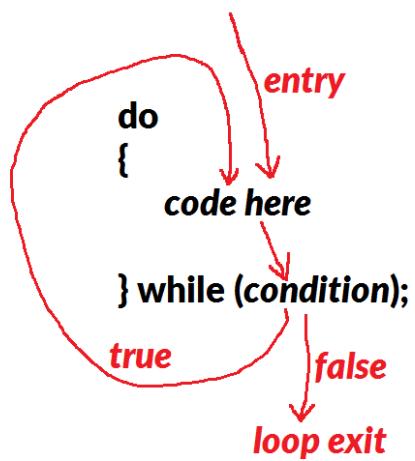
Example on While

```

<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.



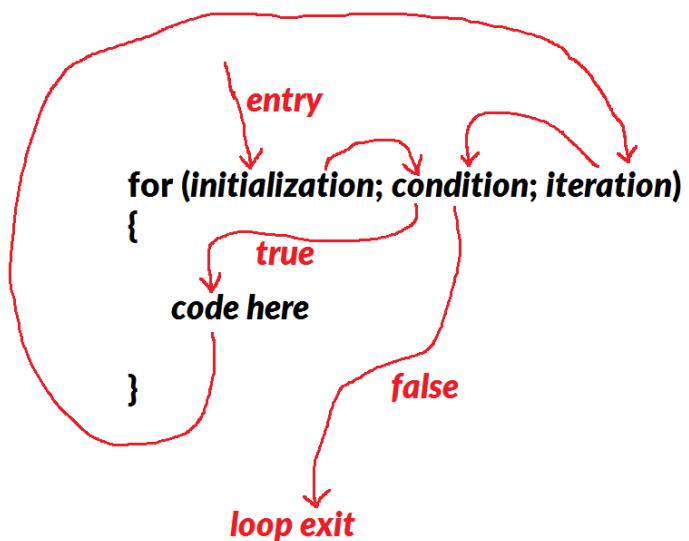
Example on Do-While

```

<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

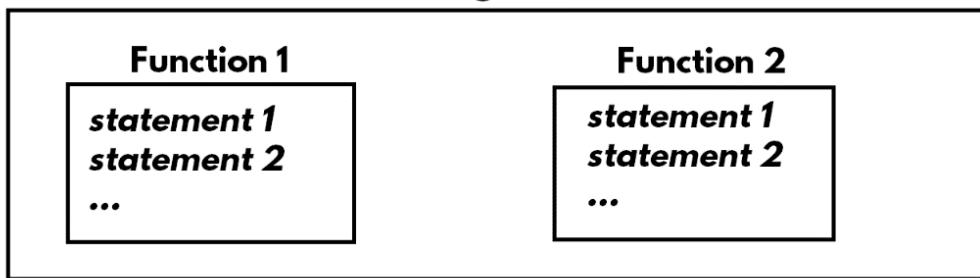
```

<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");
      }
    </script>
  </body>
</html>
  
```

```
    }
    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)

```

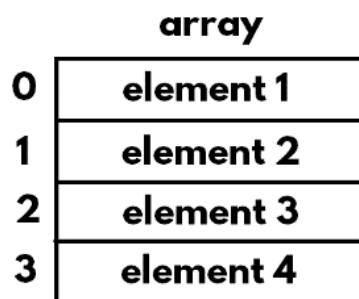
```

{
  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.



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:
 - Class-based Object Oriented Programming Language
 - Prototype-based Object Oriented Programming Language

Creating Objects

- We can create object in 2 ways:
 - Object Literals
 - 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 }';
    </script>
  </body>
</html>
```

```

        console.log(s);
        var stu = JSON.parse(s);
        console.log(stu);
        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 to 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: numervalue.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); // [empty]
      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);
    </script>
  </body>
</html>
```

```

        console.log(typeof k);
        console.log(typeof l);
    </script>
</body>
</html>

```

Parselnt 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); //empty
        console.log(j); //NaN
        console.log(k); //space
        console.log(l); //NaN
        console.log(typeof a);
    </script>
</body>
</html>

```

```

        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
        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>
```

```
</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;
    </script>
  </body>
</html>
```

```

        console.log("Before trim: " + n1); //55
        console.log("After trim: " + n2); //40
    </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();
    </script>
  </body>
</html>

```

```

        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

NoScript

- It is used to display a message to the user, when the user has disabled JavaScript in the browser.

Syntax:

```
<noscript>message</noscript>
```

Examples on NoScript

```
<html>
  <head>
    <title>noscript</title>
  </head>
  <body>
    <script>
      console.log("<h1>Message from javascript</h1>");
    </script>

    <noscript>
      <h1>Please enable javascript</h1>
    </noscript>
  </body>
</html>
```

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 Clousers

```
<html>
<head>
    <title>Clousers</title>
</head>
<body>
    <h1>Clousers</h1>
    <script>
        var Sample = function()
        {
            var x = 0;
            return {
                increment: function()
                {
                    x++;
                },
                decrement: function()
                {
                    x--;
                }
            };
        };
    </script>
</body>
</html>
```

```

    {
        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>

```

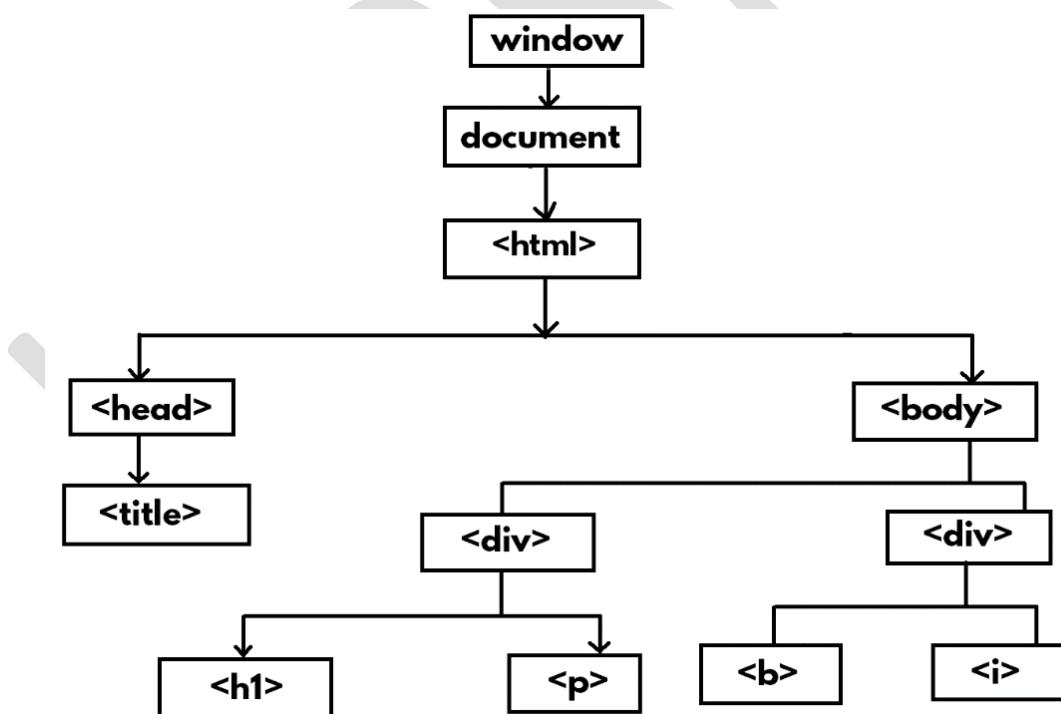
DOM

- 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

- Window
- Document
- 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 an confirmation dialogbox 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>
```

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

7. print()

- This method displays print dialogbox, 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>
```

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>

<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>getElementsByTagName</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.getElementsByTagName("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>

```

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

```

```

<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>
    <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>
    </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;
    </script>
  </body>
</html>
```

```

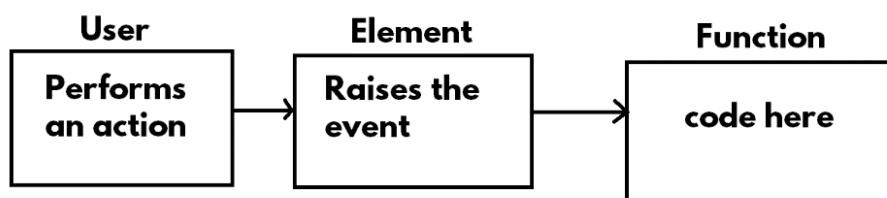
mympara.setAttribute("id", "p" + n);
document.getElementById("div1").appendChild(mympara);
</script>
</body>
</html>

```

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";
      }

      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( )
{
}

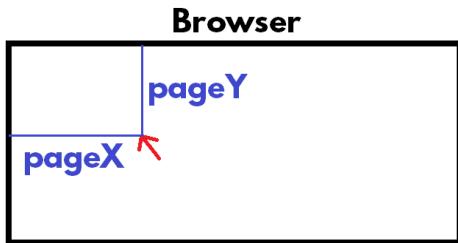
```

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.pageX: Represents “Y” co-ordinate of mouse pointer position.



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>
</html>
```

“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:

- When the user modifies the value of textbox and presses TAB key.
- When the user checks / unchecks the checkbox.
- When the user selects the radio button.
- 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()  
      {  
        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>
```

```

    </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>

```

“this” keyword

- The “this” keyword represents the “current element”, which has raised the event.

Example on “this” keyword

```

<html>
  <head>
    <title>this</title>
  </head>
  <body>
    <h1>this</h1>
    <input type="button" id="button1" value="click me">
    <input type="button" id="button2" value="click me">
    <input type="button" id="button3" value="click me">
    <input type="button" id="button4" value="click me">
    <input type="button" id="button5" value="click me">
    <script>
      document.getElementById("button1").addEventListener("click", fun1);
      document.getElementById("button2").addEventListener("click", fun1);
      document.getElementById("button3").addEventListener("click", fun1);
      document.getElementById("button4").addEventListener("click", fun1);
      document.getElementById("button5").addEventListener("click", fun1);
      function fun1()
      {
        console.log(this);
        this.setAttribute("value", "thanx");
        this.style.backgroundColor = "skyblue";
      }
    </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>
</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"> *
<span id="spanusername" class="error"></span><br>
Password:
<input type="password" id="txtpassword"> *
<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();
          document.getElementById("spanemail").innerHTML = "Invalid email";
        }
        else
        {
          document.getElementById("spanemail").innerHTML = "";
        }
      }
    </script>
  </body>
</html>

```

```

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

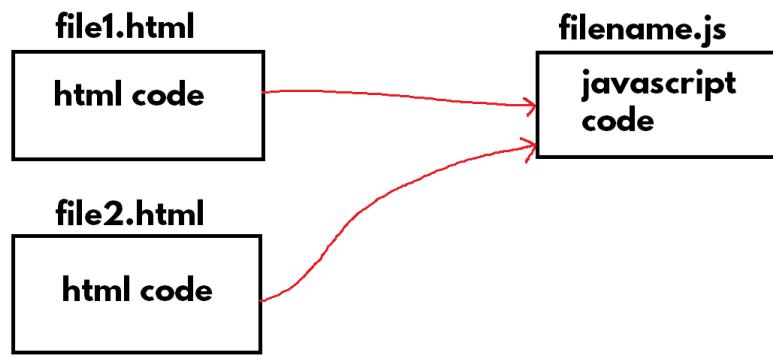
1. Internal JavaScript

- 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”.



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>
```

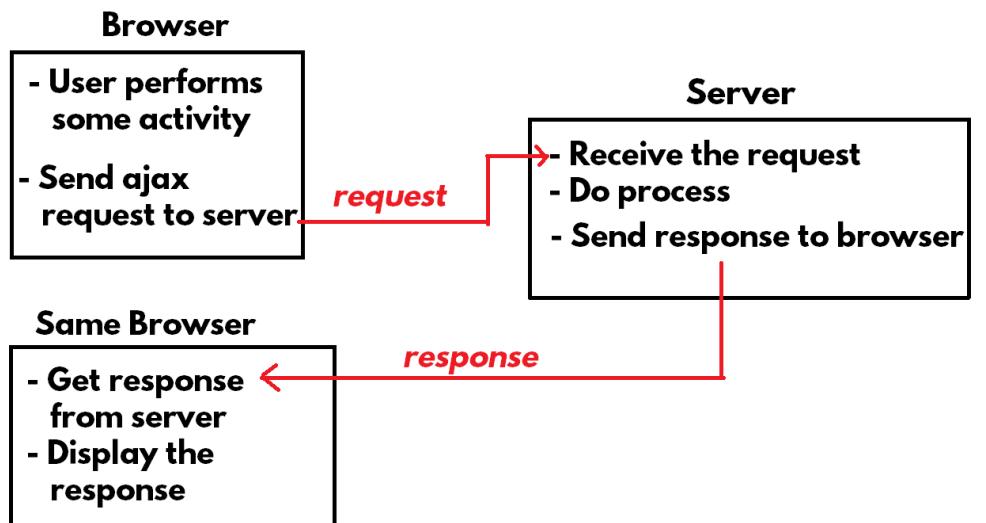
AJAX

Introduction to AJAX

- AJAX is not a language, but it is a “concept”, which is used to “send a background request from browser to server” and also “get background response from server to browser”, without refreshing the web page in the browser.

- AJAX allows us to interact with the server and get some data from server, without refreshing the full web page.
- Ex: Google search, IRCTC search trains.

Execution Flow of AJAX



Advantages of AJAX

- Executes faster.
- Less burden on browser (client) and server.
- User experience is better.

Types of AJAX request

- In AJAX, the browser can send the following four types of requests:
 1. GET : Used to retrieve / search data from server.
 2. POST : Used to insert data to server.
 3. PUT : Used to update data on server.
 4. DELETE : Used to delete data from server.

The “XMLHttpRequest” class

- We can send ajax request to server and get ajax response from server by using a pre-defined class called “XMLHttpRequest”.
- This class is supported by all the modern browsers (except IE 6, 7 and 8).

Members of XMLHttpRequest

- XMLHttpRequest class
 - 1. “readyState” property
 - 2. “onreadystatechange” property
 - 3. “responseText” property
 - 4. “open()” method
 - 5. “send()” method

open() method:

- This method is used to specify the request details such as server url (address), type of the request (GET / POST).

Server url: The address (url) of server program, to which you want to send request.

Type: Represents type of the request.

Options: GET | POST | PUT | DELETE

GET : Retrieving data from server

POST : Inserting data

PUT : Updating data

DELETE: Deleting data

Syntax:

open(“type of request”, “url”)

Example:

open(“get”, “http://localhost:7070”)

send() method

- This method sends an asynchronous request (background request) to server.

Syntax: send()

Example: send()

readyState

- It represents a number, that represents current status of the request.

Syntax: readyState

Example: readyState

0 : Request not initialized.

1 : Request opened (open() method is called).

2 : Request sent to server (send() method called).

- 3 : Server started processing the request.
- 4 : Response received from server.

onreadystatechange

- o This property stores a callback function, which executes automatically, when the “readyState” property gets changed.

Syntax: onreadystatechange = functionname;

Example: onreadystatechange = fun1;

responseText

- o This property stores the actual response sent from the server to the browser.
- o It represents the data in string format.

Syntax: responseText

Example: responseText

AJAX – NodeJS – Simple - Example

Creating the application

- Create “c:\ajax” folder.
- Create “index.html” and “index.html” files in the “c:\ajax” folder.

c:\ajax

- message.txt
- index.html

c:\ajax\message.txt

Hello World

c:\ajax\index.html

```
<html>
  <head>
    <title>JavaScript AJAX - Simple</title>
  </head>
  <body>
    <h1>JavaScript AJAX - Simple</h1>
    <input type="button" id="button1" value="GET data from server">
    <div id="div1"></div>
    <script>
      var xhr;
```

```
document.getElementById("button1").addEventListener("click", fun1);
function fun1()
{
    xhr = new XMLHttpRequest();
    xhr.open("GET", "/message.txt");
    xhr.onreadystatechange = fun2;
    xhr.send();
}

function fun2()
{
    if (xhr.readyState == 4)
    {
        document.getElementById("div1").innerHTML += xhr.responseText + "<br>";
    }
}
</script>
</body>
</html>
```

Execution

- Download and install “nodejs” from “<https://nodejs.org>”
- Open “Command Prompt” and run the following commands:
cd c:\ajax
npm install http-server -c-0
http-server
- Open browser and enter the url: <http://localhost:8080/index.html>

AJAX – NodeJS – Json Object - Example

Creating the application

- Create “c:\ajax” folder.
- Create “employee.json” and “index.html” files in the “c:\ajax” folder.

```
c:\ajax
- employee.json
- index.html
```

c:\ajax\employee.json

```
{ "empid": 1, "empname": "Scott", "salary": 4000 }
```

c:\ajax\index.html

```

<html>
  <head>
    <title>JavaScript AJAX - Json Object</title>
  </head>
  <body>
    <h1>JavaScript AJAX - Json Object</h1>
    <input type="button" id="button1" value="Get data from server"><br>
    Emp ID: <input type="text" id="txt1"><br>
    Emp Name: <input type="text" id="txt2"><br>
    Salary: <input type="text" id="txt3"><br>

    <script>
      var xhr;
      document.getElementById("button1").addEventListener("click", fun1);
      function fun1()
      {
        xhr = new XMLHttpRequest();
        xhr.open("GET", "/employee.json");
        xhr.onreadystatechange = fun2;
        xhr.send();
      }

      function fun2()
      {
        if (xhr.readyState == 4)
        {
          var emp = JSON.parse(xhr.responseText);
          document.getElementById("txt1").value = emp.empid;
          document.getElementById("txt2").value = emp.empname;
          document.getElementById("txt3").value = emp.salary;
        }
      }
    </script>
  </body>
</html>

```

Execution

- Download and install “nodejs” from “<https://nodejs.org>”
- Open “Command Prompt” and run the following commands:
 cd c:\ajax
 npm install http-server -c-0
 http-server
- Open browser and enter the url: <http://localhost:8080/index.html>

AJAX – NodeJS – Json Array - Example**Creating the application**

- Create “c:\ajax” folder.
- Create “employees.json” and “index.html” files in the “c:\ajax” folder.

c:\ajax

- employees.json
- index.html

c:\ajax\employees.json

```
[  
  { "empid": 1, "empname": "Scott", "salary": 4000 },  
  { "empid": 2, "empname": "Allen", "salary": 7500 },  
  { "empid": 3, "empname": "Jones", "salary": 9200 },  
  { "empid": 4, "empname": "James", "salary": 8400 },  
  { "empid": 5, "empname": "Smith", "salary": 5600 }  
]
```

c:\ajax\index.html

```
<html>  
  <head>  
    <title>JavaScript AJAX - Json Array</title>  
  </head>  
  <body>  
    <h1>JavaScript AJAX - Json Array</h1>  
    <input type="button" id="button1" value="Get data from server">  
    <table id="table1" border="1"></table>  
  
    <script>  
      var xhr;  
      document.getElementById("button1").addEventListener("click", fun1);  
      function fun1()  
      {  
        xhr = new XMLHttpRequest();  
        xhr.open("GET", "/employees.json");  
        xhr.onreadystatechange = fun2;  
        xhr.send();  
      }  
  
      function fun2()  
      {  
        if (xhr.readyState == 4)  
        {  
          var emps = JSON.parse(xhr.responseText);  
        }  
      }  
    </script>  
  </body>  
</html>
```

```
document.getElementById("table1").innerHTML = "<tr> <th>Emp ID</th> <th>Emp  
Name</th> <th>Salary</th> </tr>";  
for (var i = 0; i < emps.length; i++)  
{  
    document.getElementById("table1").innerHTML += "<tr> <td>" + emps[i].empid + "</td>  
<td>" + emps[i].empname + "</td> <td>" + emps[i].salary + "</td> </tr>";  
}  
}  
}  
</script>  
</body>  
</html>
```

Execution

- Download and install “nodejs” from “<https://nodejs.org>”
- Open “Command Prompt” and run the following commands:
cd c:\ajax
npm install http-server -c-0
http-server
- Open browser and enter the url: <http://localhost:8080/index.html>

JavaScript - New Features

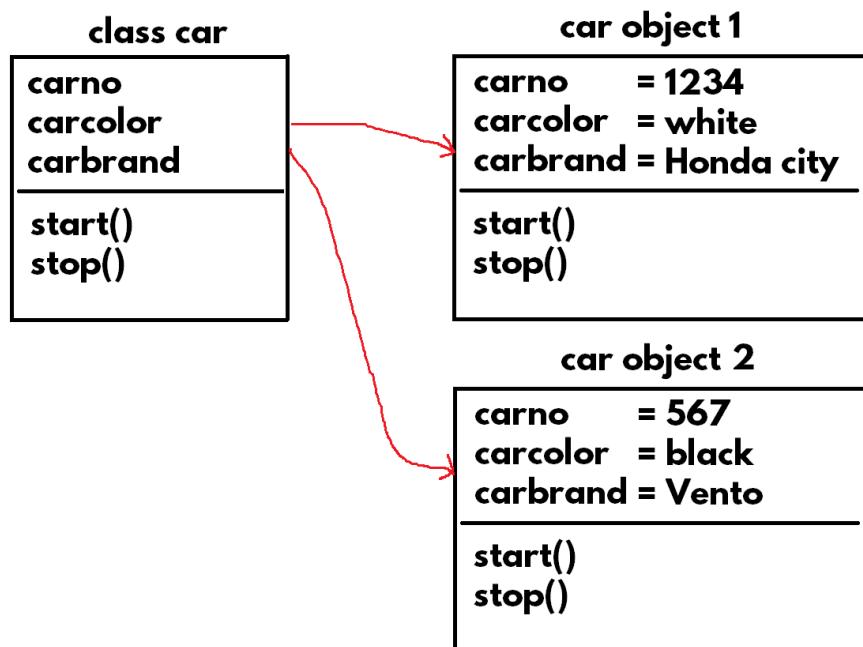
Introduction to JavaScript 6

- "JavaScript 6" version was 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

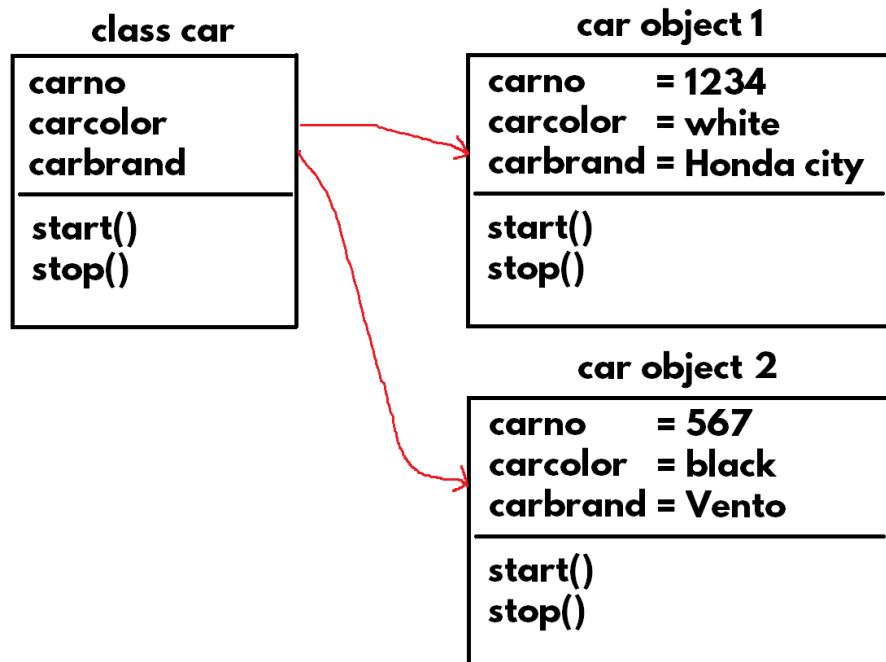
```
new classname();
```

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();
    </script>
  </body>
</html>
```

```

        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 and 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);
    </script>

```

```

        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;
    }
}

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 accessorname(argumentname)
{
    this.property = argumentname;
}

```

Create Get Method

```
get accessorname(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";
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>
```

HARSHA

JQUERY

Fundamentals of jQuery

- jQuery is a JavaScript library, which is a collection of pre-defined functions that are written in JavaScript, which are used to perform DOM manipulations easily.
- jQuery is the shortcut syntax for JavaScript.
- jQuery internally uses JavaScript
- jQuery supports DOM manipulations and AJAX
- jQuery is case sensitive
- jQuery is open source. That means its source code is available for public online for free
- jQuery is cross browser compatible. That means jQuery supports all the browsers such as Google Chrome, Mozilla Firefox, Safari, Microsoft Internet Explorer, Microsoft Edge etc.

Generations of jQuery

1. Generation 1.x

- It supports all the browsers including IE 6, 7, 8.

2. Generation 2.x

- It supports all the browsers except IE 6, 7, 8.

3. Generation 3.x

- It supports all the browsers except IE 6, 7, 8.

Formats of jQuery Library file

- “jQuery library file” is available in two formats:

1. Uncompressed
2. Compressed (or) Minified

1. Uncompressed

- It contains the code with comments, line breaks and spaces.
- It is understandable.
- The file size is: 260 kb.
- It is recommended in development time.

2. Compressed

- It contains the code with no comments, no line breaks and no spaces.

- It is not understandable.
- The file size is: 88 kb.
- It is recommended, while uploading the project into the server.

Downloading jQuery

- Go to “<http://jquery.com>”.
- Click on “Download jQuery”.
- Click on “Download the uncompressed, development jQuery 3.2.1”.
- You will get “jquery-3.2.1.js” file.
- Press Ctrl+S to save (download) the file.
- After downloading the file, go to the downloaded location and copy-paste it into the “application folder” (c:\jquery).
- You have to create the html file in the same folder.

“\$” function

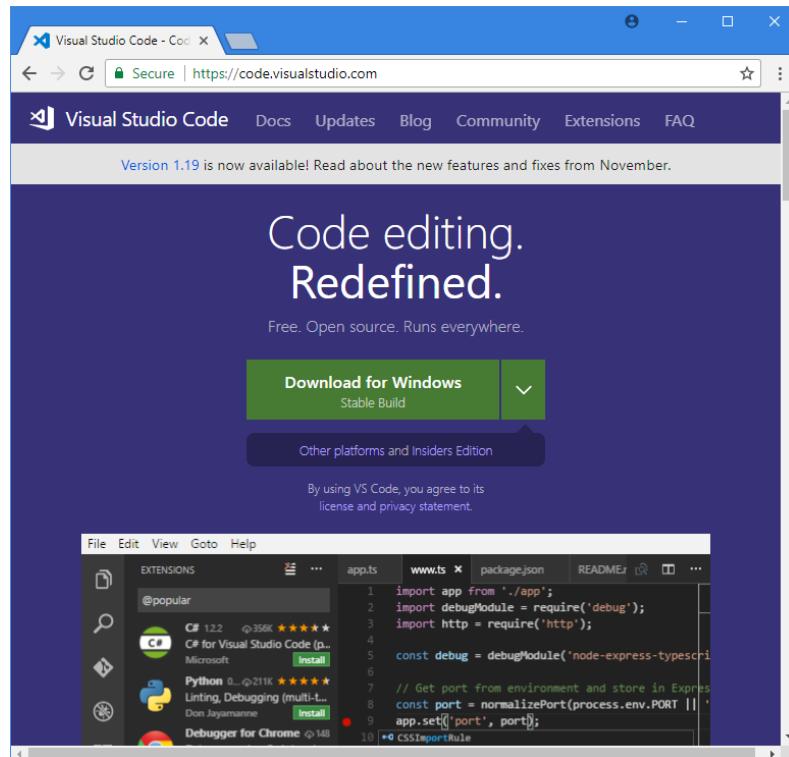
- “\$” is a pre-defined function in jQuery, which is used to select the elements.
- It receives a “selector” asw argument, searches the DOM for the matching elements and returns the matching elements as an array.
- **Syntax:** `$(“selector”)`

Steps to Prepare First Example in jQuery

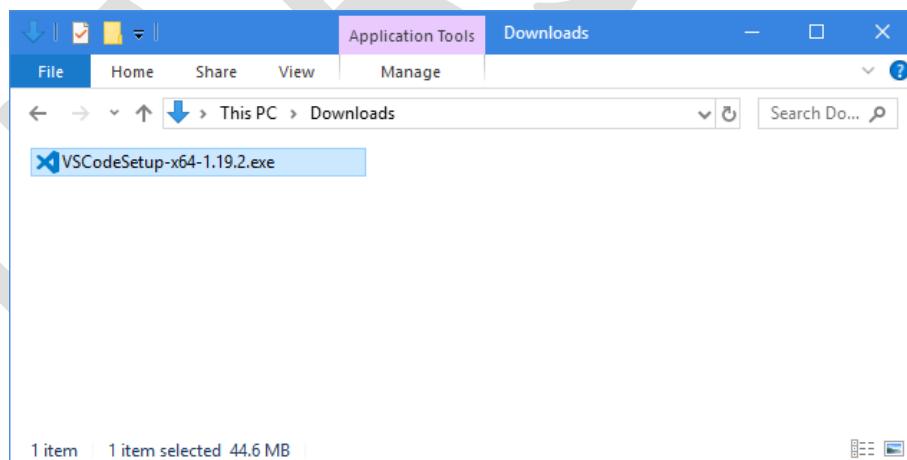
1. Installing Visual Studio Code
2. Creating jQuery Program
3. Executing jQuery Program

1. Installing Visual Studio Code

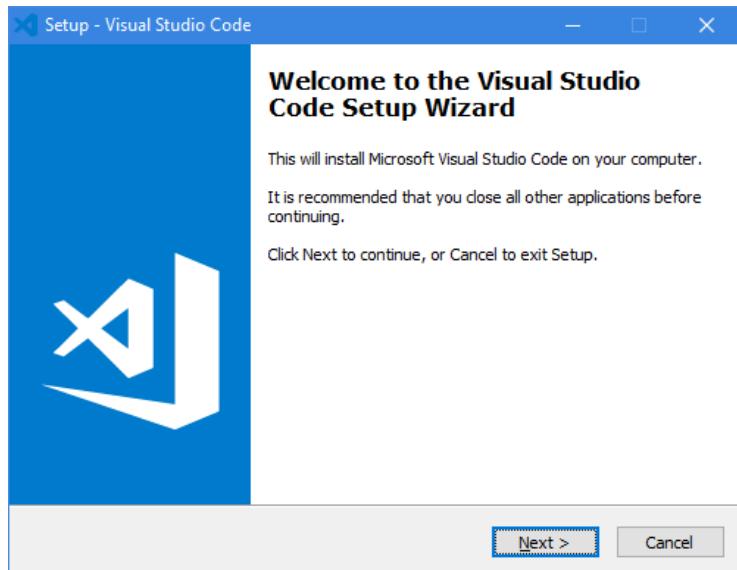
- “Visual Studio Code” is the recommended editor for html, css, javascript, jquery etc.
- Go to <https://code.visualstudio.com>



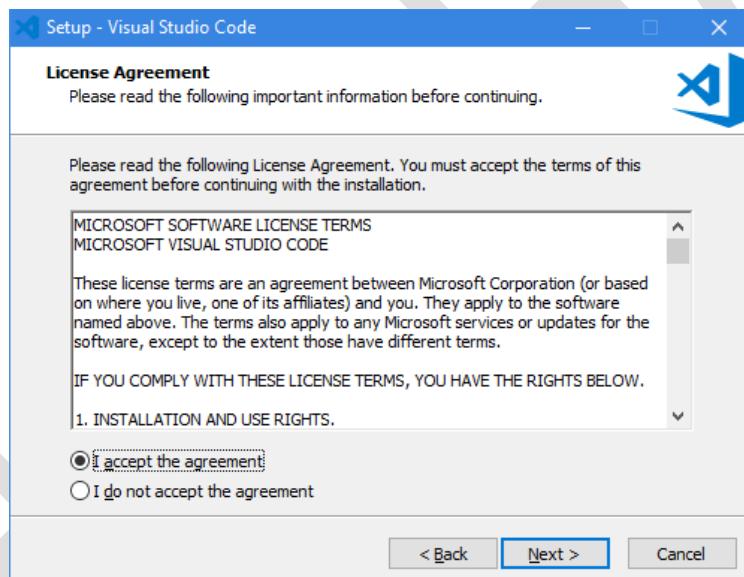
- Click on “Download for Windows”.
- **Note:** The version number may be different at your practice time.
- Go to “Downloads” folder; you can find “VSCodeSetup-x64-1.19.2.exe” file.



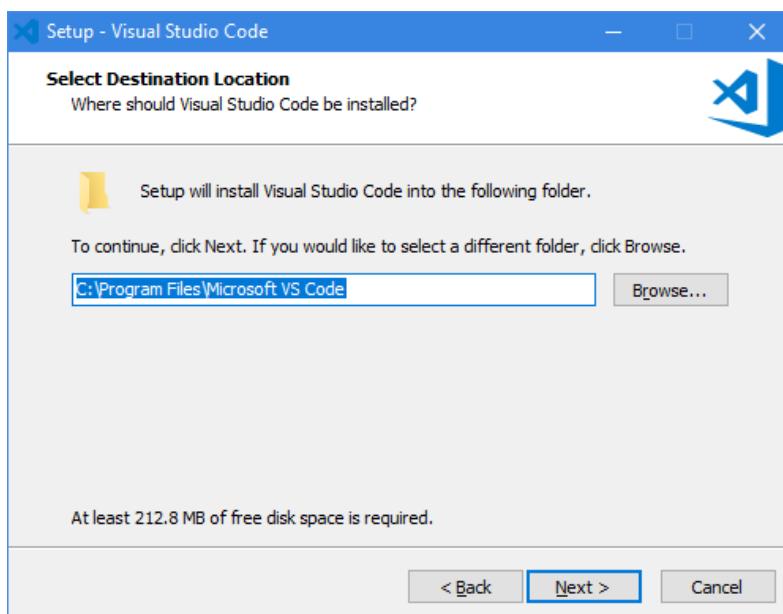
- Double click on “VSCodeSetup-x64-1.19.2.exe” file.
- Click on “Yes”.



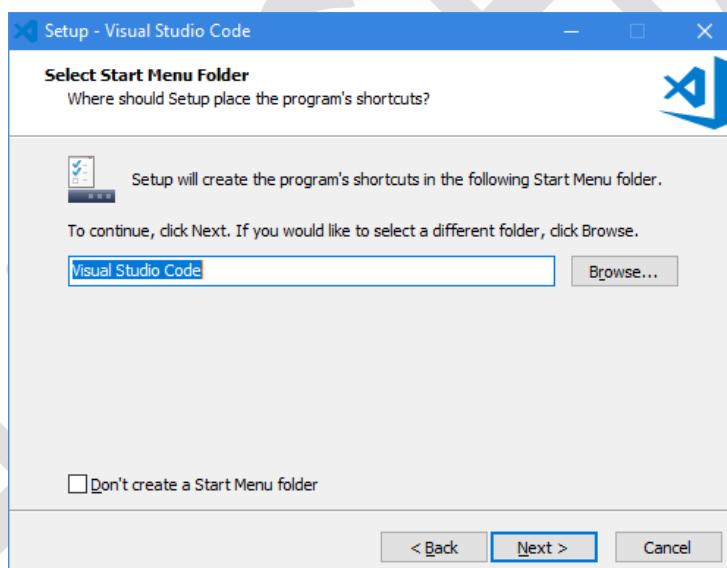
- Click on “Next”.



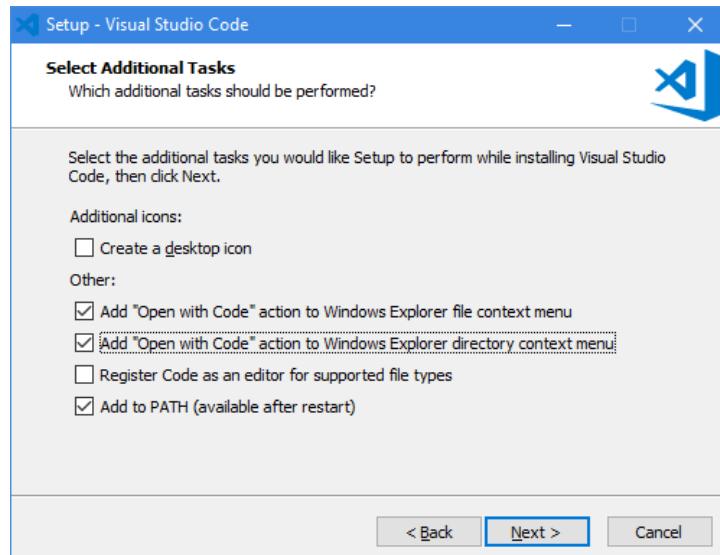
- Click on “I accept the agreement”.
- Click on “Next”.



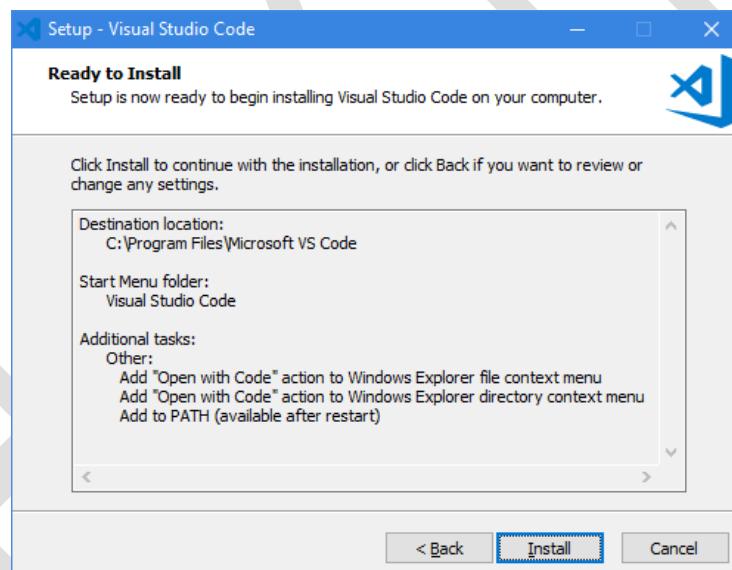
- Click on “Next”.



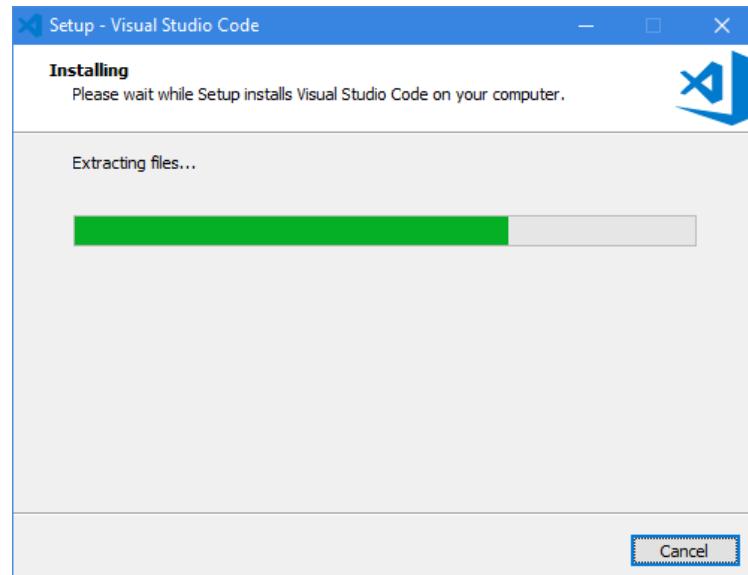
- Click on “Next”.



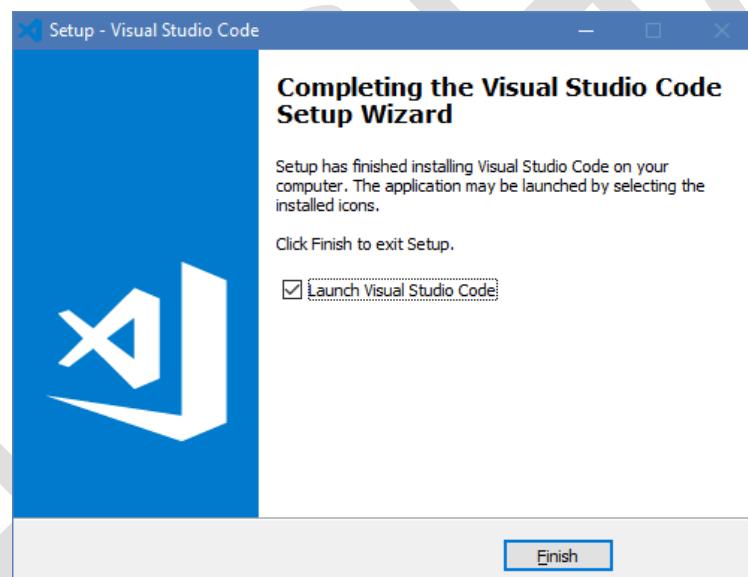
- Check the checkbox “Add Open with Code action to Windows Explorer file context menu”.
- Check the checkbox “Add Open with Code action to Windows Explorer directory context menu”.
- Click on “Next”.



- Click on “Install”.



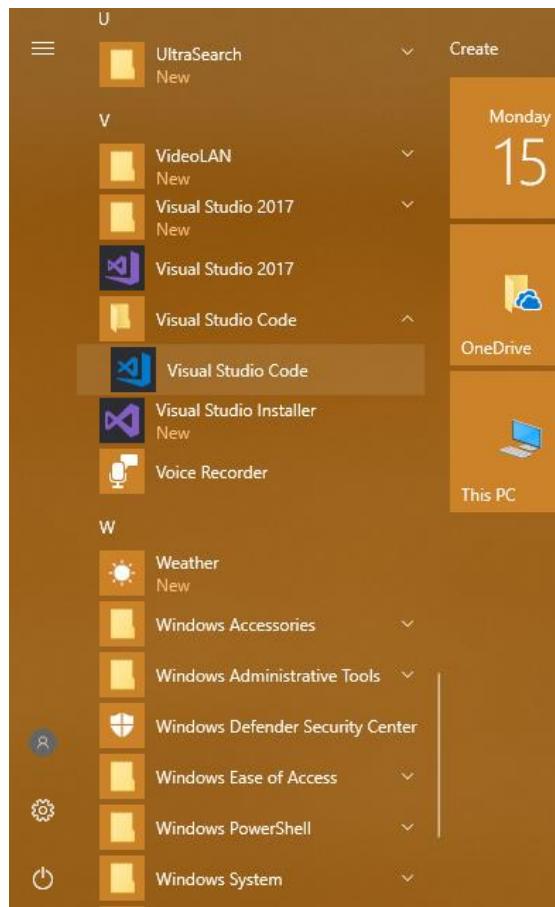
- Installation is going on....

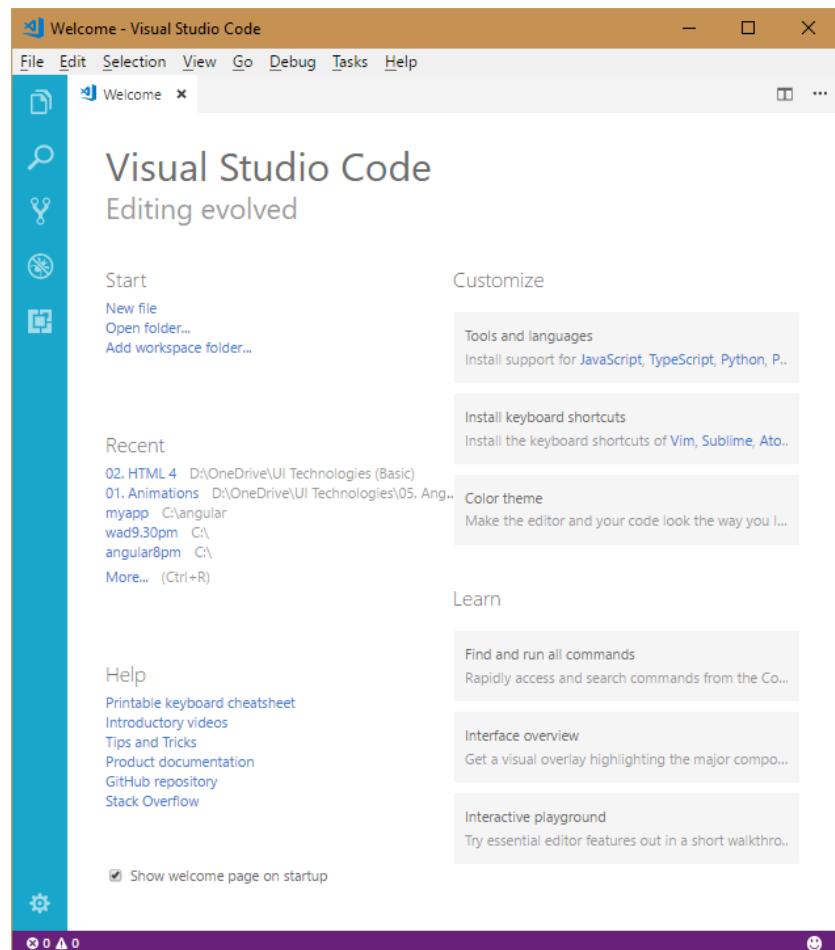


- Click on "Finish".

2. Create jQuery Program

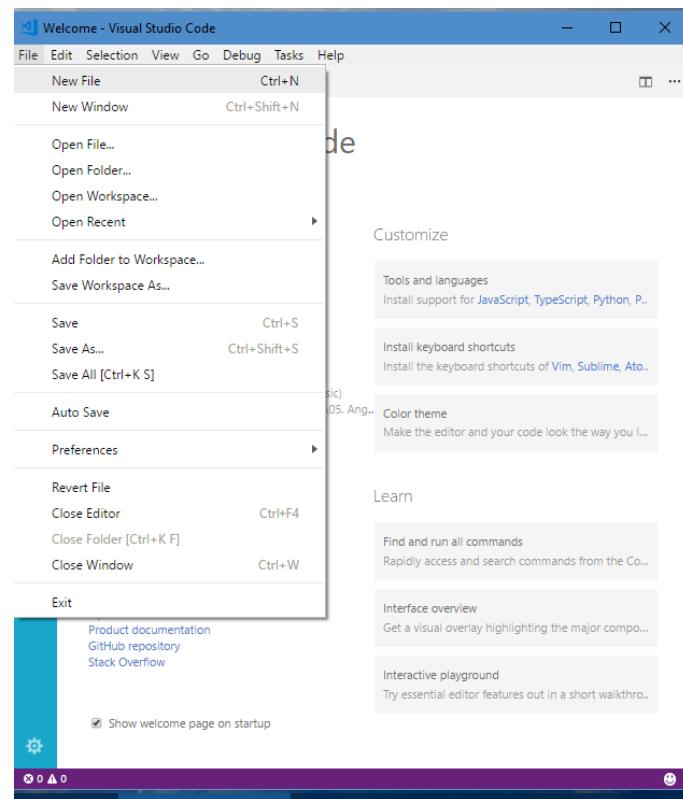
- Open "Visual Studio Code", by clicking on "Start" – "Visual Studio Code".



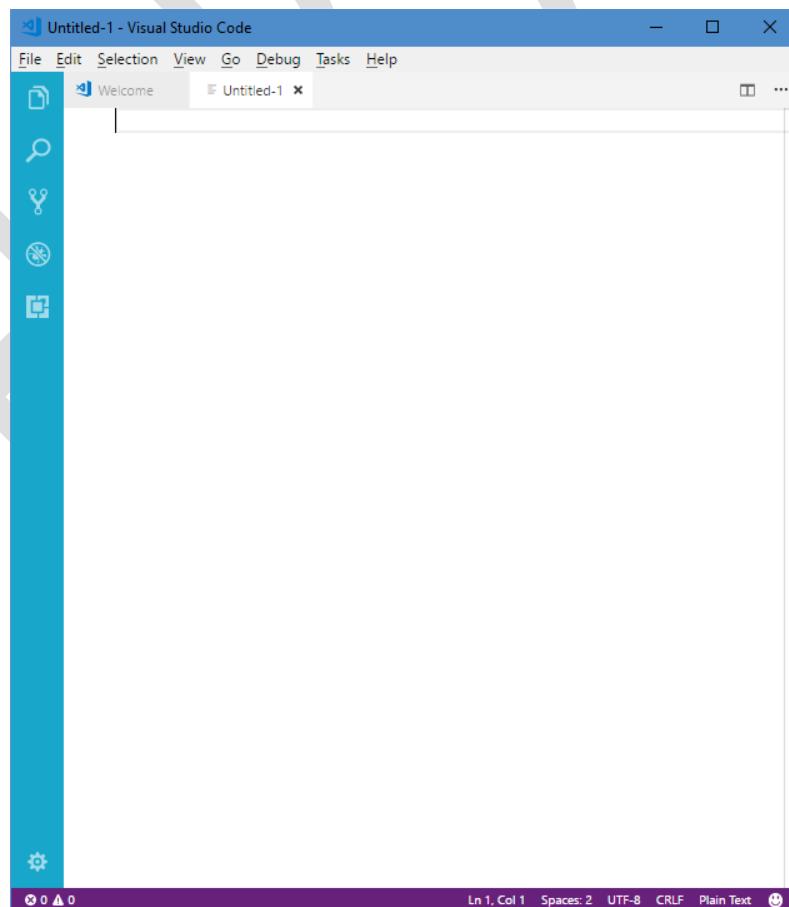


- Visual Studio Code opened.

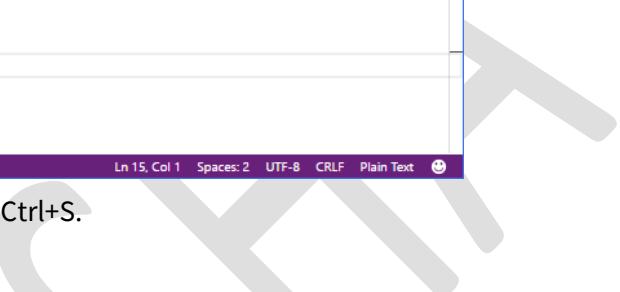
UI Technologies



- Go to “File” – “New File”.



- Type the program as follows:

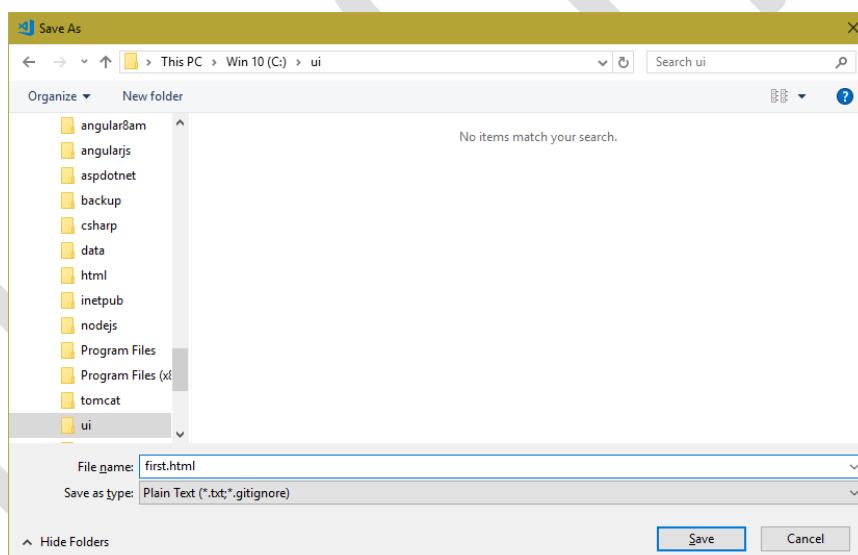


A screenshot of Visual Studio Code showing an untitled file named "Untitled-1". The code editor contains the following HTML and JavaScript code:

```
<html>
<head>
<title>jQuery</title>
</head>
<body>
<p id="p1">para 1</p>
<script src="jquery-3.2.1.js"></script>
<script>
    $('#p1').html("Hello");
</script>
</body>
</html>
```

The status bar at the bottom shows: Ln 15, Col 1, Spaces: 2, UTF-8, CRLF, Plain Text.

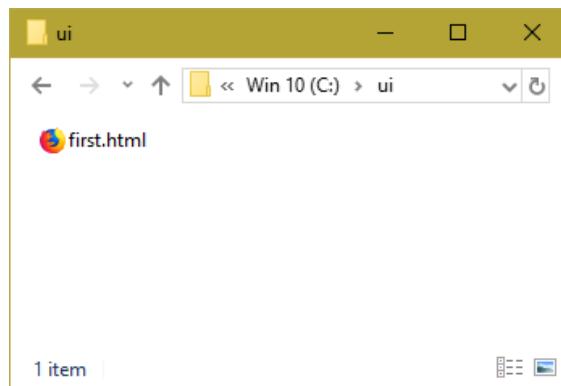
- Go to “File” menu – “Save” (or) Press Ctrl+S.



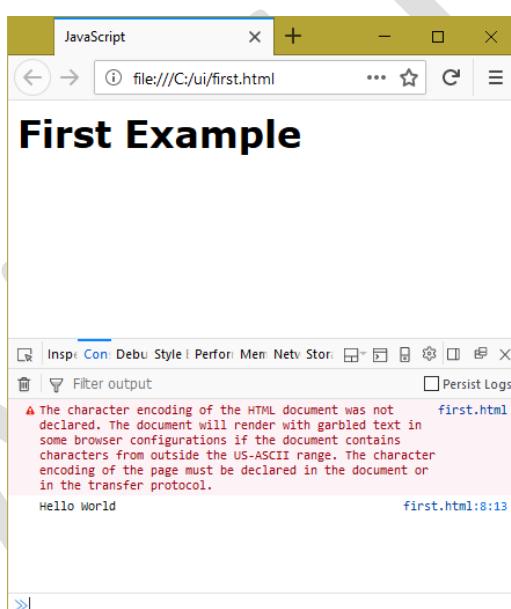
- Go to "c:\\" and click on "New folder". Enter the new folder name as "ui".
- Select "c:\ui" folder and enter the filename as "first.html".
- Click on "Save".
- Now the typescript file (c:\ui\first.html) is ready.

3. Execute the 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" – "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

Manipulating Content

Get html()

- The `html()` gets the current inner html of the tag (including child tags).

Syntax: `html()`

Example: `$("#p1").html()`

Example on get html()

```
<html>
  <head>
    <title>jQuery - Get Html</title>
    <style type="text/css">
      #div1
      {
        border: 1px solid red;
      }
    </style>
  </head>
  <body>
    <div id="div1">
      Hello, <b>how</b> are you
    </div>
    <input type="button" id="button1" value="get html">
    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        //get html (including child tags)
        var s=$("#div1").html();
        alert(s);
      }
    </script>
  </body>
</html>
```

Get text()

- The text() function gets the current inner html of the tag (only plain text, excluding child tags).

Syntax: text()

Example: \$("#p1").text()

Example on get text()

```
<html>
  <head>
    <title>jQuery - Get Text</title>
    <style type="text/css">
      #div1
      {
        border: 1px solid red;
      }
    </style>
  </head>
  <body>
    <div id="div1">
```

```

Hello, <b>how</b> are you
</div>
<br>
<input type="button" id="button1" value="get text">
<script src="jquery-3.2.1.js"></script>

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    //get text (excluding child tags)
    var s=$("#div1").text();
    alert(s);
  }
</script>
</body>
</html>

```

Set html()

- The html() function is used to set (overwrite) the inner html of the tag (including child tags).

Syntax: `html("new inner html")`

Example: `$("#div1").html("<p>Hello</p>")`

Example on set html()

```

<html>
  <head>
    <title>jQuery - Set Html</title>
    <style type="text/css">
      #div1
      {
        border: 1px solid red;
      }
    </style>
  </head>
  <body>
    <div id="div1">
      Hello, <b>how</b> are you
    </div>
    <br>
    <input type="button" id="button1" value="set html">
    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        //set html
        $("#div1").html("I am <b>fine</b>");
      }
    </script>

```

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

Set text()

- The text() function is used to set (overwrite) the inner html of the tag (only plain text).

Syntax: text("new inner html")

Example: \$("#div1").text("Hello")

Example on set text()

```
<html>
  <head>
    <title>jQuery - Set Text</title>
    <style type="text/css">
      #div1
      {
        border: 1px solid red;
      }
    </style>
  </head>
  <body>
    <div id="div1">
      Hello, <b>how</b> are you
    </div>
    <br>
    <input type="button" id="button1" value="set text">
    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        //set text
        $("#div1").text("I am fine");
      }
    </script>
  </body>
</html>
```

before()

- This function adds new content before the start tag of the element.

Syntax: before("new inner html")

Example: \$("#div1").before("<p>new para</p>")

Example on before()

```

<html>
  <head>
    <title>jQuery - Before</title>
    <style type="text/css">
      #div1
      {
        border: 1px solid red;
      }
    </style>
  </head>
  <body>
    <div id="div1">
      <p>para 1</p>
      <p>para 2</p>
      <p>para 3</p>
    </div>
    <br>
    <input type="button" id="button1" value="before">
    <script src="jquery-3.2.1.js"></script>
    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("#div1").before("<p>new para</p>");
      }
    </script>
  </body>
</html>

```

prepend()

- This function adds new content after the start tag of the element.

Syntax: `prepend("new content")`

Example: `$("#div1").prepend("<p>new para</p>")`

Example on prepend()

```

<html>
  <head>
    <title>jQuery - Prepend</title>
    <style type="text/css">
      #div1
      {
        border: 1px solid red;
      }
    </style>
  </head>
  <body>

```

```

<div id="div1">
  <p>para 1</p>
  <p>para 2</p>
  <p>para 3</p>
</div>
<br>
<input type="button" id="button1" value="prepend">
<script src="jquery-3.2.1.js"></script>
<script>
  $("#button1").click(fun1);
  function fun1()
  {
    $("#div1").prepend("<p>new para</p>");
  }
</script>
</body>
</html>

```

append()

- This function adds new content before the end tag of the element.

Syntax: append("new content")

Example: \$("#div1").append("<p>new para</p>")

Example on append()

```

<html>
  <head>
    <title>jQuery - Append</title>
    <style type="text/css">
      #div1
      {
        border: 1px solid red;
      }
    </style>
  </head>
  <body>
    <div id="div1">
      <p>para 1</p>
      <p>para 2</p>
      <p>para 3</p>
    </div>
    <br>
    <input type="button" id="button1" value="append">
    <script src="jquery-3.2.1.js"></script>
    <script>
      $("#button1").click(fun1);
      function fun1()
      {

```

```

        $("#div1").append("<p>new para</p>");
    }
</script>
</body>
</html>

```

after()

- This function adds new content after the closing tag of the element.

Syntax: `after("new content")`

Example: `$("#div1").after("<p>new para</p>")`

Example on after()

```

<html>
  <head>
    <title>jQuery - After</title>
    <style type="text/css">
      #div1
      {
        border: 1px solid red;
      }
    </style>
  </head>
  <body>
    <div id="div1">
      <p>para 1</p>
      <p>para 2</p>
      <p>para 3</p>
    </div>
    <br>
    <input type="button" id="button1" value="after">
    <script src="jquery-3.2.1.js"></script>
    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("#div1").after("<p>new para</p>");
      }
    </script>
  </body>
</html>

```

insertBefore()

- This function cuts an existing element and pastes the same before the start tag of destination element.

Syntax: insertBefore("destination element")

Example: \$("#p1").insertBefore("#div1")

Example on insertBefore()

```
<html>
  <head>
    <title>jQuery - InsertBefore</title>
    <style type="text/css">
      #div1
      {
        border: 1px solid red;
      }
    </style>
  </head>
  <body>
    <div id="div1">
      <p>para 1</p>
      <p>para 2</p>
      <p>para 3</p>
    </div>
    <span id="span1">span1</span><br>
    <input type="button" id="button1" value="insert before">
    <script src="jquery-3.2.1.js"></script>
    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("#span1").insertBefore("#div1");
      }
    </script>
  </body>
</html>
```

prependTo()

- This function cuts an existing element and pastes the same after the start tag of destination element.

Syntax: prependTo("destination element")

Example: \$("#p1").prependTo("#div1")

Example on prependTo()

```
<html>
  <head>
    <title>jQuery - PrependTo</title>
    <style type="text/css">
      #div1
      {
```

```

        border: 1px solid red;
    }
</style>
</head>
<body>
<span id="span1">span1</span>
<div id="div1">
    <p>para 1</p>
    <p>para 2</p>
    <p>para 3</p>
</div><br>
<input type="button" id="button1" value="prepend to">
<script src="jquery-3.2.1.js"></script>
<script>
    $("#button1").click(fun1);
    function fun1()
    {
        $("#span1").prependTo("#div1");
        //$("#span1").clone().prependTo("#div1");
    }
</script>
</body>
</html>

```

appendTo()

- This function cuts an existing element and pastes the same before the end tag of destination element.

Syntax: appendTo("destination element")

Example: \$("#p1").appendTo("#div1")

Example on appendTo()

```

<html>
    <head>
        <title>jQuery - AppendTo</title>
        <style type="text/css">
            #div1
            {
                border: 1px solid red;
            }
        </style>
    </head>
    <body>
        <span id="span1">span1</span>
        <div id="div1">
            <p>para 1</p>
            <p>para 2</p>
            <p>para 3</p>
        </div>
        <br>
        <input type="button" id="button1" value="append to">

```

```

<script src="jquery-3.2.1.js"></script>
<script>
    $("#button1").click(fun1);
    function fun1()
    {
        //$("#span1").appendTo("#div1");
        $("#span1").clone().appendTo("#div1");
    }
</script>
</body>
</html>

```

insertAfter()

- This function cuts an existing element and pastes the same after end tag of destination element.

Syntax: insertAfter("destination element")

Example: \$("#p1").insertAfter("#div1")

Example on insertAfter()

```

<html>
    <head>
        <title>jQuery - InsertAfter</title>
        <style type="text/css">
            #div1
            {
                border: 1px solid red;
            }
        </style>
    </head>
    <body>
        <span id="span1">span1</span>
        <div id="div1">
            <p>para 1</p>
            <p>para 2</p>
            <p>para 3</p>
        </div>
        <br><input type="button" id="button1" value="insert after">
        <script src="jquery-3.2.1.js"></script>
        <script>
            $("#button1").click(fun1);
            function fun1()
            {
                //insert after ending tag
                $("#span1").insertAfter("#div1");
            }
        </script>
    </body>
</html>

```

wrap()

- This function adds a parent tag for each selected element.

Syntax: wrap("<parenttag></parenttag>")

Example: \$("p").wrap("<div></div>")

Example on wrap()

```
<html>
  <head>
    <title>jQuery - Wrap</title>
    <style type="text/css">
      div
      {
        border: 1px solid red;
      }
    </style>
  </head>
  <body>
    <p>para 1</p>
    <p>para 2</p>
    <p>para 3</p>
    <br>
    <input type="button" id="button1" value="wrap">

    <script src="jquery-3.2.1.js"></script>
    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("p").wrap("<div></div>");
      }
    </script>
  </body>
</html>
```

wrapAll()

- This function adds a common parent tag for all the selected elements.

Syntax: wrapAll("<parenttag></parenttag>")

Example: \$("p").wrapAll("<div></div>")

Example on wrapAll()

```
<html>
```

```

<head>
    <title>jQuery - WrapAll</title>
    <style type="text/css">
        div
        {
            border: 1px solid red;
        }
    </style>
</head>
<body>
    <p>para 1</p>
    <p>para 2</p>
    <p>para 3</p>
    <br>
    <input type="button" id="button1" value="wrap all">
    <script src="jquery-3.2.1.js"></script>
    <script>
        $("#button1").click(fun1);
        function fun1()
        {
            $("p").wrapAll("<div></div>");
        }
    </script>
</body>
</html>

```

empty()

- This function is used to delete the inner html of the element.

Syntax: empty()

Example: \$("p").empty()

Example on empty()

```

<html>
    <head>
        <title>jQuery - Empty</title>
        <style type="text/css">
            body, input
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
            #div1
            {
                border: 1px solid red;
            }
        </style>
    </head>
    <body>

```

```

<div id="div1">Hello</div>
<br>
<input type="button" id="button1" value="empty">
<script src="jquery-3.2.1.js"></script>
<script>
    $("#button1").click(fun1);
    function fun1()
    {
        $("#div1").empty();
    }
</script>
</body>
</html>

```

remove()

- This function removes an existing element.

Syntax: remove()

Example: \$("p").remove()

Example on remove()

```

<html>
    <head>
        <title>jQuery - Remove</title>
        <style type="text/css">
            #div1
            {
                border: 1px solid red;
            }
        </style>
    </head>
    <body>
        <div id="div1">Hello</div><br>
        <input type="button" id="button1" value="remove">
        <script src="jquery-3.2.1.js"></script>

        <script>
            $("#button1").click(fun1);
            function fun1()
            {
                $("#div1").remove();
            }
        </script>
    </body>
</html>

```

replaceWith()

- This function replaces (overwrites) an existing element with another element.

Syntax: replaceWith("new content")

Example: \$("div").replaceWith("<p>new para</p>")

Example on replaceWith()

```
<html>
  <head>
    <title>jQuery - ReplaceWith</title>
    <style type="text/css">
      #div1
      {
        border: 1px solid red;
      }
    </style>
  </head>
  <body>
    <div id="div1">Hello</div>
    <br>
    <input type="button" id="button1" value="replace with">
    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("#div1").replaceWith("<p>new para</p>");
      }
    </script>
  </body>
</html>
```

Event Handling

- Event:** A keyboard/ mouse action, performed by the user, at run time.
- jQuery provides a new way to handle the events more easily.
- jQuery supports all the javascript events.

List of jQuery Functions to handle Events

1. click()
2. dblclick()
3. mouseover()

4. mouseout()
5. hover()
6. mousemove()
7. focus()
8. blur()
9. keyup()
10. keypress()
11. change()
12. on()
13. off()

The “click()” function

- The “click()” function handles “click” event.
- The “click” event executes when the user clicks on an element.

Syntax:

```
$(“selector”).click(functionname);  
function functionname()  
{  
    Code here  
}
```

Example on “Click”

```
<html>  
  <head>  
    <title>jQuery - Click</title>  
    <style type="text/css">  
      body,input  
      {  
        font-family: Tahoma;  
        font-size: 30px;  
      }  
      #div1  
      {  
        width: 300px;  
        height: 200px;  
        background-color: #00cccc;  
      }  
    </style>
```

```

</head>
<body>
  <div id="div1">div 1</div>
  <script src="jquery-3.2.1.js"></script>
  <script>
    $("#div1").click(fun1);
    function fun1()
    {
      $("#div1").html("Thanx");
    }
  </script>
</body>
</html>

```

The “dblclick()” function

- The “dblclick()” function handles “dblclick” event.
- The “dblclick” event executes when the user double clicks on an element.

Syntax:

```

$(“selector”).dblclick(functionname);

function functionname()
{
  Code here
}

```

Example on “Dblclick”

```

<html>
  <head>
    <title>jQuery - Dblclick</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
      #div1
      {
        width: 300px;
        height: 200px;
        background-color: #00cccc;
      }
    </style>
  </head>
  <body>
    <div id="div1">double click me</div>
  
```

```
<script src="jquery-3.2.1.js"></script>

<script>
  $("#div1").dblclick(fun1);
  function fun1()
  {
    $("#div1").html("Thanx");
  }
</script>
</body>
</html>
```

The “mouseover()” function and “mouseout()” function

The mouseover() function

- The “mouseover()” function handles “mosueover” event.
- The “mouseover” event executes when the user places the mouse pointer on the element.

Syntax:

```
$(“selector”).mouseover(functionname);

function functionname()
{
  Code here
}
```

The mouseout() function

- The “mouseout()” function handles “mouseout” event.
- The “mouseout” event executes when the user moves the mouse pointer from inside to outside the element.

Syntax:

```
$(“selector”).mouseout(functionname);

function functionname()
{
  Code here
}
```

Example on “Mouseover” and “mouseout”

```

<html>
  <head>
    <title>jQuery - Mouseover and Mouseout</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
      #div1
      {
        width: 300px;
        height: 200px;
        background-color: #00cccc;
      }
    </style>
  </head>
  <body>
    <div id="div1">hover me</div>

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#div1").mouseover(fun1);
      $("#div1").mouseout(fun2);
      function fun1()
      {
        $("#div1").html("Thanx");
      }
      function fun2()
      {
        $("#div1").html("hover me");
      }
    </script>
  </body>
</html>

```

The “hover()” function

- The “hover()” function is a shortcut to handle both “mouseover” and “mouseout” events at-a-time.

Syntax:

```

$(“selector”).hover(functionname1, functionname2);

function functionname1()
{
  Code here
}

function functionname2()

```

```
{
    Code here
}
```

Example on “hover”

```
<html>
    <head>
        <title>jQuery - Hover</title>
        <style type="text/css">
            body,input
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
            #div1
            {
                width: 200px;
                height: 200px;
                background-color: #00ff99;
            }
        </style>
    </head>
    <body>
        <div id="div1">
            hover me
        </div>

        <script src="jquery-3.2.1.js"></script>

        <script>
            $("#div1").hover(fun1, fun2);
            function fun1()
            {
                $(this).html("Thanx");
            }
            function fun2()
            {
                $(this).html("hover me");
            }
        </script>
    </body>
</html>
```

The “mousemove()” function

- The “mousemove()” function handles “mousemove” event.
- The “mousemove” event executes when the user moves the pointer from one place to another place within the same element (for every pixel change).

Syntax:

```
$(“selector”).mousemove(functionname);
function functionname(event)
{
    event.pageX = X position of mouse pointer
    event.pageY = Y position of mouse pointer
}
```

Example on “mousemove”

```
<html>
  <head>
    <title>jQuery - Mousemove</title>
    <style type="text/css">
      body,input
      {
        font-family:Tahoma;
        font-size:30px;
      }
      #div1
      {
        width:300px;
        height:200px;
        background-color:#00cccc;
      }
    </style>
  </head>
  <body>
    <div id="div1">mouse move me</div>

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#div1").mousemove(fun1);
      function fun1(event)
      {
        $("#div1").html(event.pageX + ", " + event.pageY);
      }
    </script>
  </body>
</html>
```

The “focus()” function

- The “focus()” function handles “focus” event.
- The “focus” event executes when the cursor enters into the element.

Syntax:

```
$(“selector”).focus(functionname);
```

```
function functionname()
{
    Code here
}
```

The blur() function

- The “blur()” function handles “blur” event.
- The “blur” event executes when the cursor goes out of the element.

Syntax:

```
$(“selector”).blur(functionname);

function functionname()
{
    Code here
}
```

Example on “focus” and “blur”

```
<html>
<head>
    <title>jQuery - Focus and Blur</title>
    <style type="text/css">
        body,input
        {
            font-family: Tahoma;
            font-size: 30px;
        }
        #div1
        {
            width: 300px;
            height: 200px;
            background-color: #00cccc;
        }
        #span1
        {
            color: green;
            font-weight: bold;
            display: none;
        }
    </style>
</head>
<body>
    Email: <input type="text" id="txt1">
    <span id='span1'>Use gmail only.</span>

```

```

<script src="jquery-3.2.1.js"></script>

<script>
  $("#txt1").focus(fun1);
  $("#txt1").blur(fun2);
  function fun1()
  {
    $("#span1").show();
  }
  function fun2()
  {
    $("#span1").hide();
  }
</script>
</body>
</html>

```

The “keyup()” function

- The “keyup()” function handles “keyup” event.
- The “keyup” event executes when the user presses any key on the keyboard, while the cursor is inside the textbox.

Syntax:

```

$(“selector”).keyup(functionname);

function functionname()
{
  Code here
}

```

Example on “keyup”

```

<html>
  <head>
    <title>jQuery - Keyup</title>
  </head>
  <body>
    Source text: <input type="text" id="txt1"><br>
    Destination text: <input type="text" id="txt2"><br>

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#txt1").keyup(fun1);
      function fun1()
      {
        $("#txt2").val($("#txt1").val());
      }
    </script>

```

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

The “keypress()” function

- The “keypress()” function handles “keypress” event.
- The “keypress” event executes when the user presses any key on the keyboard, while the cursor is inside the textbox.



Keypress (vs) Keyup

Keypress:

- Executes before placing the character into the textbox.
- We can accept / reject the currently pressed character.

Keyup:

- Executes after placing the character into the textbox.
- We can't reject the currently pressed character, because it is already accepted before calling the “keyup” event.

Syntax:

```
$("selector").keypress(functionname);
function functionname(event)
{
    event.which = ASCII value of currently pressed character.
    event.preventDefault() = Cancells the currently pressed character.
}
```

Example on “Keypress”

```
<html>
  <head>
    <title>jQuery - Keypress</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
```

```

Source text: <input type="text" id="txt1"><br>
Destination text: <input type="text" id="txt2"><br>
<script src="jquery-3.2.1.js"></script>

<script>
  $("#txt1").keypress(fun1);
  function fun1()
  {
    $("#txt2").val($("#txt1").val());
  }
</script>
</body>
</html>

```

The “change()” function

- The “change()” function handles “change” event.
- The “change” event executes when the user do following:
 1. Modify the value of a textbox and press TAB key
 2. Check / uncheck the checkbox
 3. Select the radio button.
 4. Select an item in the dropdownlist.

Syntax:

```

$(“selector”).change(functionname);

function functionname()
{
}

```

Example on “Change” with TextBox

```

<html>
  <head>
    <title>jQuery - Change - TextBox</title>
    <style type="text/css">
      body,input,select
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    Source text: <input type="text" id="txt1"><br>
    Destination text: <input type="text" id="txt2"><br>
  
```

```

<script src="jquery-3.2.1.js"></script>

<script>
  $("#txt1").change(fun1);
  function fun1()
  {
    var s=$("#txt1").val();
    $("#txt2").val(s);
  }
</script>
</body>
</html>

```

Example on “Change” with CheckBox

```

<html>
  <head>
    <title>jQuery - Change - Checkbox</title>
    <style type="text/css">
      body,input,select
      {
        font-family: Tahoma;
        font-size: 30px;
      }
      #p1
      {
        font-weight: bold;
        font-size: 40px;
        color: darkblue;
      }
    </style>
  </head>
  <body>
    <input type="checkbox" id="checkbox1">
    <label for="checkbox1">I accept license agreement</label>
    <p id="p1"></p>

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#checkbox1").change(fun1);
      function fun1()
      {
        if ($("#checkbox1")[0].checked == true)
        {
          $("#p1").html("Accepted");
        }
        else
        {
          $("#p1").html("Not accepted");
        }
      }
    </script>
  </body>

```

```
</html>
```

Example on “Change” with RadioButton

```
<html>
  <head>
    <title>jQuery - Change - RadioButton</title>
    <style type="text/css">
      body,input,select
      {
        font-family: Tahoma;
        font-size: 30px;
      }
      #p1
      {
        font-weight: bold;
        font-size: 40px;
        color: darkblue;
      }
    </style>
  </head>
  <body>
    Gender:
    <input type="radio" id="rb1" name="gender">
    <label for="rb1">Male</label>
    <input type="radio" id="rb2" name="gender">
    <label for="rb2">Female</label>
    <p id="p1"></p>

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#rb1").change(fun1);
      $("#rb2").change(fun1);
      function fun1()
      {
        if ($("#rb1")[0].checked == true)
        {
          $("#p1").html("Male selected");
        }
        else
        {
          $("#p1").html("Female selected");
        }
      }
    </script>
  </body>
</html>
```

Example on “Change” with DropDownList

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

```

<title>jQuery - Change - Dropdownlist</title>
<style type="text/css">
body,input,select
{
    font-family: Tahoma;
    font-size: 30px;
}
#p1
{
    font-weight: bold;
    font-size: 40px;
    color: darkblue;
}
</style>
</head>
<body>
Country:
<select id="dropdownlist1">
    <option>Please select</option>
    <option>India</option>
    <option>China</option>
    <option>UK</option>
    <option>USA</option>
</select>
<p id="p1">para 1</p>

<script src="jquery-3.2.1.js"></script>

<script>
$("#dropdownlist1").change(fun1);
function fun1()
{
    if($("#dropdownlist1").val() == "Please select")
    {
        $("#p1").html("You selected none");
    }
    else
    {
        $("#p1").html($("#dropdownlist1").val());
    }
}
</script>
</body>
</html>

```

The “on()” function

- The “on()” function handles any event. It attaches a function with an event; so that whenever the event occurs, the function will be called automatically.
- It supports all the events: click, dblclick, mouseover, mouseout, mousemove, focus, blur, keyup, keypress, change, contextmenu, cut, copy, paste etc.

• Advantages:

- You can use “off()” to unhandle the event.
- You can handle rare events such as contextmenu, cut, copy, paste etc.

Syntax:

```
$(document).on("event", "selector", functionname);  
function functionname()  
{  
}  
}
```

Example on “On”

```
<html>  
  <head>  
    <title>jQuery - On</title>  
    <style type="text/css">  
      body, input  
      {  
        font-family: Tahoma;  
        font-size: 30px;  
      }  
      #div1  
      {  
        width: 300px;  
        height: 200px;  
        background-color: #00cccc;  
      }  
    </style>  
  </head>  
  <body>  
    <div id="div1">div 1</div>  
    <script src="jquery-3.2.1.js"></script>  
    <script>  
      $(document).on("click", "#div1", fun1);  
      function fun1()  
      {  
        $("#div1").html("Thanx");  
      }  
    </script>  
  </body>  
</html>
```

The “off()” function

- The “off()” function unhandles any event. It detaches a function from an event; so that whenever the event occurs, the function will not be called.

Syntax:

```
$(document).off("event", "selector");
```

Example on “Off”

```
<html>
  <head>
    <title>jQuery - Off</title>
    <style type="text/css">
      #div1
      {
        width: 300px;
        height: 200px;
        background-color: #00cccc;
      }
    </style>
  </head>
  <body>
    <div id="div1">div 1</div>

    <script src="jquery-3.2.1.js"></script>

    <script>
      $(document).on("click", "#div1", fun1);
      function fun1()
      {
        $("#div1").html("Thanx at " + new Date().toLocaleTimeString());
        $(document).off("click", "#div1");
      }
    </script>
  </body>
</html>
```

The “contextmenu” event

- The “contextmenu” event executes when the user right clicks on an element.

Syntax:

```
$(document).on("contextmenu", document, functionname);

function functionname()
{
}
```

Example on “Contextmenu”

```
<html>
  <head>
    <title>jQuery - Contextmenu</title>
  </head>
```

```
<body>
    <input type="text" id="txt1">
    <input type="text" id="txt2">
    <input type="text" id="txt3">

    <script src="jquery-3.2.1.js"></script>

    <script>
        $(document).on("contextmenu", document, fun1);
        function fun1(event)
        {
            alert("right click not allowed");
            event.preventDefault();
        }
        //event = browser given information
    </script>
</body>
</html>
```

The “cut”, “copy”, “paste” events

- The “cut” event executes when the user selects “cut” option with keyboard / mouse.
- The “copy” event executes when the user selects “copy” option with keyboard / mouse.
- The “paste” event executes when the user selects “paste” option with keyboard / mouse.

Syntax:

```
$(document).on("cut", document, functionname);

function functionname( )
{
}
```

```
$(document).on("copy", document, functionname);

function functionname( )
{
}
```

```
$(document).on("paste", document, functionname);

function functionname( )
{
}
```

Example on “Cut, copy, paste”

```

<html>
  <head>
    <title>jQuery - Disabling cut, copy, paste</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <input type="text" id="txt1">
    <input type="text" id="txt2">
    <input type="text" id="txt3">
    <script src="jquery-3.2.1.js"></script>
    <script>
      $(document).on("cut copy paste", "document", fun1);
      function fun1(event)
      {
        alert("cut copy paste not allowed");
        event.preventDefault(); //default functionality will be stopped
      }
      //event = browser given information
    </script>
  </body>
</html>

```

The “data()” function

- The “data()” function is used to store / retrieve data in the browser memory, temporarily (while the web page is running the browser).

Syntax to set data:

`$(“selector”).data(“key”, “value”);`

Syntax to get data:

`$(“selector”).data(“key”);`

Example on “data()”

```

<html>
  <head>
    <title>jQuery - Data</title>
    <style type="text/css">
      body,input
      {

```

```

        font-family: Tahoma;
        font-size: 30px;
    }

```

```

</style>
</head>
<body>
    Username: <input type="text" id="txt1"><br>
    <input type="button" value="Set data into memory" id="button1">
    <input type="button" value="Get data from memory" id="button2">

    <script src="jquery-3.2.1.js"></script>

    <script>
        $("#button1").click(fun1);
        $("#button2").click(fun2);
        function fun1()
        {
            $("#txt1").data("x",$("#txt1").val());
            $("#txt1").val("");
            alert("Saved");
        }
        function fun2()
        {
            $("#txt1").val($("#txt1").data("x"));
        }
    </script>
</body>
</html>

```

Effects

- jQuery effects are used to hide / show the elements smoothly.
- It is used for showing menus, validation messages etc.
- jQuery supports 3 types of effects:
 1. fade effect : fade out / fade in
 2. slide effect : slide up / slide down
 3. hide / show effect : hide / show

1. Fade effect:

```

fadeOut(milli seconds);
fadeIn(milli seconds);

```

2. slide effect:

```

slideUp(milli seconds);
slideDown(milli seconds);

```

3. hide / show effect:

```
hide(milli seconds);
show(milli seconds);
```

Note: 1000 milli seconds = 1 second

Example on Fade

```
<html>
  <head>
    <title>jQuery - Fade</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      #div1
      {
        width: 400px;
        height: 300px;
        background-color: #ff6699;
        text-align: justify;
      }
    </style>
  </head>
  <body>
    <div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum is
simply dummy text of the printing and typesetting industry.</div>
    <input type="button" id="button1" value="fade out">
    <input type="button" id="button2" value="fade in">

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      $("#button2").click(fun2);
      function fun1()
      {
        $("#div1").fadeOut(1000); //1000 milli sec = 1 sec
      }
      function fun2()
      {
        $("#div1").fadeIn(1000);
      }
    </script>
  </body>
</html>
```

Example on Slide

```

<html>
  <head>
    <title>jQuery - Slide</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      #div1
      {
        width: 400px;
        height: 300px;
        background-color: #ff6699;
        text-align: justify;
      }
    </style>
  </head>
  <body>
    <div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum is  
simply dummy text of the printing and typesetting industry.</div>
    <input type="button" id="button1" value="slide up">
    <input type="button" id="button2" value="slide down">
    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      $("#button2").click(fun2);
      function fun1()
      {
        $("#div1").slideUp(1000);
      }
      function fun2()
      {
        $("#div1").slideDown(1000);
      }
    </script>
  </body>
</html>

```

Example on Hide / Show

```

<html>
  <head>
    <title>jQuery - Hide or Show</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      #div1
    </style>
  </head>
  <body>

```

```

{
    width:400px;
    height:300px;
    background-color: #ff6699;
    text-align:justify;
}
</style>
</head>
<body>
    <div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum is  

simply dummy text of the printing and typesetting industry.</div>
    <input type="button" id="button1" value="hide">
    <input type="button" id="button2" value="show">
    <script src="jquery-3.2.1.js"></script>
    <script>
        $("#button1").click(fun1);
        $("#button2").click(fun2);
        function fun1()
        {
            $("#div1").hide(1000);
        }
        function fun2()
        {
            $("#div1").show(1000);
        }
    </script>
</body>
</html>

```

toggle()

- toggle() function hides / shows the element.
- It hides the element if it is already visible.
- It shows the element if it is already invisible.

fadeToggle(milli seconds)	: fade out / fade in
slideToggle(milli seconds)	: slide up / slide down
toggle(milli seconds)	: hide / show

Example on toggle()

```

<html>
<head>
    <title>jQuery - Toggle</title>
    <style type="text/css">
        body,input
        {
            font-family: 'Tahoma';
            font-size: 30px;
        }
    </style>

```

```

        }
    #div1
    {
        width: 400px;
        height: 300px;
        background-color: #ff6699;
        text-align: justify;
    }

```

</style>

</head>

<body>

<div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum is
simply dummy text of the printing and typesetting industry.</div>

<input type="button" id="button1" value="fade toggle">

<input type="button" id="button2" value="slide toggle">

<input type="button" id="button3" value="show/hide toggle">

<script src="jquery-3.2.1.js"></script>

<script>

```

        $("#button1").click(fun1);
        function fun1()
        {
            $("#div1").fadeToggle(1000); //fadeOut or fadeIn
        }

        $("#button2").click(fun2);
        function fun2()
        {
            $("#div1").slideToggle(1000); //slideUp or slideDown
        }

        $("#button3").click(fun3);
        function fun3()
        {
            $("#div1").toggle(1000); //hide or show
        }
    </script>

```

</body>

</html>

“this” keyword

- “this” keyword represents the “current element”, which has raised the current event.
- **Syntax:** \$(this)

Example on “this” keyword

```

<html>
    <head>
        <title>jQuery - Fadeout with Click</title>
        <style type="text/css">

```

```

body,input
{
    font-family: 'Tahoma';
    font-size: 30px;
}
div.class1
{
    width: 140px;
    height: 100px;
    background-color: #00cc99;
    margin: 10px;
    float: left;
    color: #003399;
    font-size: 40px;
    padding: 5px;
    font-weight: bold;
}
</style>
</head>
<body>
    <div id="div1" class="class1">1</div>
    <div id="div2" class="class1">2</div>
    <div id="div3" class="class1">3</div>
    <div id="div4" class="class1">4</div>
    <div id="div5" class="class1">5</div>
    <script src="jquery-3.2.1.js"></script>
    <script>
        $("div").click(fun1);
        function fun1()
        {
            $(this).fadeOut(700);
            //this = current element (div), which is currently clicked
        }
    </script>
</body>
</html>

```

fadeTo()

- fadeTo() function is used to change the “opacity” property gradually, based on the given no. of milliseconds.
- **Syntax:** fadeTo(milli seconds, opacity);
- **Example:** fadeTo(2000, 0.6);

Example on fadeTo()

```

<html>
    <head>
        <title>jQuery - fadeTo</title>
        <style type="text/css">
            body,input

```

```

    {
        font-family: 'Tahoma';
        font-size: 30px;
    }
    div.class1
    {
        width: 150px;
        height: 130px;
        background-color: #00cc99;
        margin: 10px;
        float: left;
        color: #003399;
        font-size: 40px;
        padding: 5px;
        font-weight: bold;
    }

```

</style>

</head>

<body>

```

        <div id="div1" class="class1">1</div>
        <div id="div2" class="class1">2</div>
        <div id="div3" class="class1">3</div>
        <div id="div4" class="class1">4</div>
        <div id="div5" class="class1">5</div>
        <script src="jquery-3.2.1.js"></script>
        <script>
            $("div").click(fun1);
            function fun1()
            {
                $(this).fadeTo(700, 0.4); //Syntax: fadeTo(milli sec, opacity)
            }
        </script>

```

</body>

</html>

Example 2 on fadeTo()

```

<html>
    <head>
        <title>jQuery - FadeTo and FadeTo</title>
        <style type="text/css">
            body, input
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
            div.class1
            {
                width: 150px;
                height: 130px;
                background-color: #00cc99;
                margin: 10px;
                float: left;
                color: #003399;
            }

```

```

        font-size: 40px;
        padding: 5px;
        font-weight: bold;
    }
</style>
</head>
<body>
    <div id="div1" class="class1">1</div>
    <div id="div2" class="class1">2</div>
    <div id="div3" class="class1">3</div>
    <div id="div4" class="class1">4</div>
    <div id="div5" class="class1">5</div>

    <script src="jquery-3.2.1.js"></script>

    <script>
        $("div").click(fun1);
        $("div").data("flag", "0");
        function fun1()
        {
            var n = $(this).data("flag");
            if(n == "0")
            {
                $(this).fadeTo(1000, 0.4); //milli sec, opacity
                $(this).data("flag", "1");
            }
            else
            {
                $(this).fadeTo(1000, 1); //milli sec, opacity
                $(this).data("flag", "0");
            }
        }
    </script>
</body>
</html>

```

Manipulating Attributes

- We can manipulate attributes by using the following functions in jQuery.

1. Set an attribute:

- attr("attribute name", "value")

2. Get an attribute:

- attr("attribute name")

3. Removing an attribute:

- removeAttr("attribute name")

4. Setting multiple attributes:

- attr({ "attribute": "value", "attribute": "value", ... })

Example on Set Attribute

```
<html>
  <head>
    <title>jQuery - Set Attribute</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <br>
    <input type="button" id="button1" value="set attribute">
    <script src="jquery-3.2.1.js"></script>
    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("#myImage").attr("src", "img2.jpg");
      }
    </script>
  </body>
</html>
```

Note: Place "img1.jpg" and "img2.jpg" in "c:\jquery" folder.

Example on Set Multiple Attributes

```
<html>
  <head>
    <title>jQuery - Set Multiple Attributes</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <br>
    <input type="button" id="button1" value="set multiple attributes">

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
```

```

        $("#" + myImage).attr({ "src": "img3.jpg", "title": "this is tooltip", "alt": "this is alternate text" });
    }
</script>
</body>
</html>

```

Note: Place "img1.jpg" and "img3.jpg" in "c:\jquery" folder.

Example on Remove Attribute

```

<html>
<head>
<title>jQuery - Remove Attr</title>
<style type="text/css">
    body,input
    {
        font-family: 'Tahoma';
        font-size: 30px;
    }
</style>
</head>
<body>
    <br>
    <input type="button" id="button1" value="remove attribute">
    <script src="jquery-3.2.1.js"></script>
    <script>
        $("#button1").click(fun1);
        function fun1()
        {
            $("#myImage").removeAttr("title");
        }
    </script>
</body>
</html>

```

Example on Get Attribute

```

<html>
<head>
<title>jQuery - Get Attribute</title>
<style type="text/css">
    body,input
    {
        font-family: 'Tahoma';
        font-size: 30px;
    }
</style>
</head>
<body>
    <br>
    <input type="button" id="button1" value="get attribute">

    <script src="jquery-3.2.1.js"></script>

```

```

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    var s=$("#myImage").attr("src");
    alert(s);
  }
</script>
</body>
</html>

```

Note: Place "img1.jpg" in "c:\jquery" folder.

Manipulating CSS

- We can manipulate css styles dynamically at run time by using the following jquery functions.

1.Adding CSS class to the element:

- addClass("css class name")

2.Removing CSS class to the element:

- removeClass("css class name")

3.Toggle (add / remove) CSS class:

- toggleClass("css class name")

4.Check whether the element has specific css class or not:

- hasClass("css class name")

5.Setting individual css properties using jquery:

- css("property", "value")

6.Getting the individual css properties using jquery:

- css("property")

7.Setting multiple individual css properties:

- css({ "property": "value", "property": "value", ... })

Example on Add Class

```

<html>
  <head>
    <title>jQuery - AddClass</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      .class1
    </style>
  </head>
  <body>
    <input type="button" value="Click Me" class="class1" />
  </body>
</html>

```

```

    {
        background-color: darkred;
        color: cyan;
        font-size: 50px;
    }

```

```

</style>
</head>
<body>
    <div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum is
simply dummy text of the printing and typesetting industry.</div>
    <input type="button" id="button1" value="add class">
    <script src="jquery-3.2.1.js"></script>

```

```

<script>
    $("#button1").click(fun1);
    function fun1()
    {
        $("#div1").addClass("class1");
    }
</script>
</body>
</html>

```

Example on Remove Class

```

<html>
    <head>
        <title>jQuery - RemoveClass</title>
        <style type="text/css">
            body, input
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
            .class1
            {
                background-color: darkred;
                color: cyan;
            }
        </style>
    </head>
    <body>
        <div id="div1" class="class1">Lorem Ipsum is simply dummy text of the printing and typesetting industry.
        Lorem Ipsum is simply dummy text of the printing and typesetting industry.</div>
        <input type="button" id="button1" value="remove class">
        <script src="jquery-3.2.1.js"></script>
        <script>
            $("#button1").click(fun1);
            function fun1()
            {
                $("#div1").removeClass("class1");
            }
        </script>
    </body>

```

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

Example on Toggle Class

```
<html>
  <head>
    <title>jQuery - ToggleClass</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      .class1
      {
        background-color: darkred;
        color: cyan;
      }
    </style>
  </head>
  <body>
    <div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum is
simply dummy text of the printing and typesetting industry.</div>
    <input type="button" id="button1" value="toggle class">

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("#div1").toggleClass("class1");
      }
    </script>
  </body>
</html>
```

Example on Has Class

```
<html>
  <head>
    <title>jQuery - HasClass</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      .class1
      {
        background-color: darkred;
      }
    </style>
  </head>
  <body>
```

```

        color: cyan;
    }

```

```

</style>
</head>
<body>
    <div id="div1" class="class1">Lorem Ipsum is simply dummy text of the printing and typesetting industry.
    Lorem Ipsum is simply dummy text of the printing and typesetting industry.</div>
    <input type="button" id="button1" value="has class">

    <script src="jquery-3.2.1.js"></script>

    <script>
        $("#button1").click(fun1);
        function fun1()
        {
            var b=$("#div1").hasClass("class1");
            alert(b);
        }
    </script>
</body>
</html>

```

Example on Get CSS

```

<html>
    <head>
        <title>jQuery - Get CSS</title>
        <style type="text/css">
            body,input
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
            .class1
            {
                background-color: darkred;
                color: cyan;
            }
        </style>
    </head>
    <body>
        <div id="div1" class="class1">Lorem Ipsum is simply dummy text of the printing and typesetting industry.
        Lorem Ipsum is simply dummy text of the printing and typesetting industry.</div>
        <input type="button" id="button1" value="get css property">

        <script src="jquery-3.2.1.js"></script>

        <script>
            $("#button1").click(fun1);
            function fun1()
            {
                var item=$("#div1").css("font-size");
                alert(item);
            }
        </script>
    </body>

```

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

Example on Set CSS

```
<html>
  <head>
    <title>jQuery - Set CSS</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum is
simply dummy text of the printing and typesetting industry.</div>
    <input type="button" id="button1" value="set a css property">

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("#div1").css("color", "#ff0099");
      }
    </script>
  </body>
</html>
```

Example on Set Multiple CSS

```
<html>
  <head>
    <title>jQuery - Set CSS Multiple Properties</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum is
simply dummy text of the printing and typesetting industry.</div>
    <input type="button" id="button1" value="set a css property">

    <script src="jquery-3.2.1.js"></script>
```

```

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    $("#div1").css({ "color": "yellow", "background-color": "darkblue", "font-size": "30px", "font-weight": "bold", "font-family": 'Tahoma' });
  }
</script>
</body>
</html>

```

Animations

- jQuery Animations are used to change a css property's value gradually, based on the specified milliseconds.
- We can animate any pixels-based and color-based properties.

Syntax:

```
animate( { "property": "value", "property": "value", ... }, milli seconds)
```

Example on Animations - Width

```

<html>
  <head>
    <title>jQuery - Animations</title>
    <style type="text/css">
      body, input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      #div1
      {
        width: 100px;
        height: 100px;
        background-color: #00cc66;
        border: 2px solid red;
        position: relative;
      }
    </style>
  </head>
  <body>
    <div id="div1">Hello World</div><br>
    <input type="button" id="button1" value="animate width">

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
    
```

```
function fun1()
{
    $("#div1").animate({"width":"200px"},1000);
}
</script>
</body>
</html>
```

Example on Animations - Height

```
<html>
<head>
    <title>jQuery - Animations - Height</title>
    <style type="text/css">
        body,input
        {
            font-family: 'Tahoma';
            font-size: 30px;
        }
        #div1
        {
            width:100px;
            height:100px;
            background-color: #00cc66;
            border: 2px solid red;
            position: relative;
        }
    </style>
</head>
<body>
    <div id="div1">Hello World</div><br>
    <input type="button" id="button1" value="animate height">

    <script src="jquery-3.2.1.js"></script>

    <script>
        $("#button1").click(fun1);
        function fun1()
        {
            $("#div1").animate({"height":"200px"},1000);
        }
    </script>
</body>
</html>
```

Example on Animations – Width and Height

```
<html>
<head>
    <title>jQuery - Animations - Width and Height</title>
    <style type="text/css">
        body,input
        {
```

```

        font-family: 'Tahoma';
        font-size: 30px;
    }
    #div1
    {
        width: 100px;
        height: 100px;
        background-color: #00cc66;
        border: 2px solid red;
        position: relative;
    }

```

</style>

</head>

<body>

<div id="div1">Hello World</div>

<input type="button" id="button1" value="animate width and height">

<script src="jquery-3.2.1.js"></script>

<script>

\$("#button1").click(fun1);
 function fun1()
 {
 \$("#div1").animate({"width": "300px", "height": "300px"}, 1000);
 }

</script>

</body>

</html>

Example on Animations – Width and Then Height

```

<html>
    <head>
        <title>jQuery - Animations</title>
        <style type="text/css">
            body, input
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
            #div1
            {
                width: 100px;
                height: 100px;
                background-color: #00cc66;
                border: 2px solid red;
                position: relative;
            }

```

</style>

</head>

<body>

<div id="div1">Hello World</div>

<input type="button" id="button1" value="animate width and then height">

```

<script src="jquery-3.2.1.js"></script>

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    //jquery chaining (calling jquery functions in a sequence)
    $("#div1").animate({"width": "300px"}, 1000).animate({"height":
    "300px"}, 1000).delay(1000).fadeOut(3000);
  }
</script>
</body>
</html>

```

Example on Animations – Multiple Properties

```

<html>
  <head>
    <title>jQuery - Animations - Multiple Properties</title>
    <style type="text/css">
      body, input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      #div1
      {
        width: 100px;
        height: 100px;
        background-color: #00cc66;
        border: 2px solid red;
        position: relative;
        left: 0px;
        top: 0px;
      }
    </style>
  </head>
  <body>
    <div id="div1">Hello World</div>
    <br>
    <input type="button" id="button1" value="animate multiple properties">
    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("#div1").animate({ "font-size": "50px", "border-width": "10px", "left": "30px", "top": "50px", "width": "400px", "height": "400px" }, 1000);
      }
    </script>
  </body>
</html>

```

Example on Animations – scrollTop

```

<html>
  <head>
    <title>jQuery - Animations - scrollTop</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <a href="#" id="link1">Go to Heading 7</a>

    <h1 id="heading1">Heading 1</h1>
    <p>If you have a Google Account, we may display your Profile name, Profile photo, and actions you take on Google or on third-party applications connected to your Google Account (such as +1's, reviews you write and comments you post) in our Services, including displaying in ads and other commercial contexts. We will respect the choices you make to limit sharing or visibility settings in your Google Account. For example, you can choose your settings so your name and photo do not appear in an ad.</p>
    <h1 id="heading2">Heading 2</h1>
    <p>If you have a Google Account, we may display your Profile name, Profile photo, and actions you take on Google or on third-party applications connected to your Google Account (such as +1's, reviews you write and comments you post) in our Services, including displaying in ads and other commercial contexts. We will respect the choices you make to limit sharing or visibility settings in your Google Account. For example, you can choose your settings so your name and photo do not appear in an ad.</p>
    <h1 id="heading3">Heading 3</h1>
    <p>If you have a Google Account, we may display your Profile name, Profile photo, and actions you take on Google or on third-party applications connected to your Google Account (such as +1's, reviews you write and comments you post) in our Services, including displaying in ads and other commercial contexts. We will respect the choices you make to limit sharing or visibility settings in your Google Account. For example, you can choose your settings so your name and photo do not appear in an ad.</p>
    <h1 id="heading4">Heading 4</h1>
    <p>If you have a Google Account, we may display your Profile name, Profile photo, and actions you take on Google or on third-party applications connected to your Google Account (such as +1's, reviews you write and comments you post) in our Services, including displaying in ads and other commercial contexts. We will respect the choices you make to limit sharing or visibility settings in your Google Account. For example, you can choose your settings so your name and photo do not appear in an ad.</p>
    <h1 id="heading5">Heading 5</h1>
    <p>If you have a Google Account, we may display your Profile name, Profile photo, and actions you take on Google or on third-party applications connected to your Google Account (such as +1's, reviews you write and comments you post) in our Services, including displaying in ads and other commercial contexts. We will respect the choices you make to limit sharing or visibility settings in your Google Account. For example, you can choose your settings so your name and photo do not appear in an ad.</p>
    <h1 id="heading6">Heading 6</h1>
    <p>If you have a Google Account, we may display your Profile name, Profile photo, and actions you take on Google or on third-party applications connected to your Google Account (such as +1's, reviews you write and comments you post) in our Services, including displaying in ads and other commercial contexts. We will respect the choices you make to limit sharing or visibility settings in your Google Account. For example, you can choose your settings so your name and photo do not appear in an ad.</p>
    <h1 id="heading7">Heading 7</h1>
    <p>If you have a Google Account, we may display your Profile name, Profile photo, and actions you take on Google or on third-party applications connected to your Google Account (such as +1's, reviews you write and
  
```

comments you post) in our Services, including displaying in ads and other commercial contexts. We will respect the choices you make to limit sharing or visibility settings in your Google Account. For example, you can choose your settings so your name and photo do not appear in an ad.</p>

<h1 id="heading8">Heading 8</h1>

<p>If you have a Google Account, we may display your Profile name, Profile photo, and actions you take on Google or on third-party applications connected to your Google Account (such as +1's, reviews you write and comments you post) in our Services, including displaying in ads and other commercial contexts. We will respect the choices you make to limit sharing or visibility settings in your Google Account. For example, you can choose your settings so your name and photo do not appear in an ad.</p>

<h1 id="heading9">Heading 9</h1>

<p>If you have a Google Account, we may display your Profile name, Profile photo, and actions you take on Google or on third-party applications connected to your Google Account (such as +1's, reviews you write and comments you post) in our Services, including displaying in ads and other commercial contexts. We will respect the choices you make to limit sharing or visibility settings in your Google Account. For example, you can choose your settings so your name and photo do not appear in an ad.</p>

<h1 id="heading10">Heading 10</h1>

<p>If you have a Google Account, we may display your Profile name, Profile photo, and actions you take on Google or on third-party applications connected to your Google Account (such as +1's, reviews you write and comments you post) in our Services, including displaying in ads and other commercial contexts. We will respect the choices you make to limit sharing or visibility settings in your Google Account. For example, you can choose your settings so your name and photo do not appear in an ad.</p>

```
<script src="jquery-3.2.1.js"></script>
```

```
<script>
  $("#link1").click(fun1);
  function fun1()
  {
    $("body").animate({ "scrollTop": $("#heading7").offset().top }, 1000);
  }
</script>
</body>
</html>
```

document.ready()

- “document.ready()” function is used to call a function automatically after the web page loading completed in the browser.

Syntax 1:

```
$(document).ready(functionname);
```

Syntax 2:

```
$(functionname);
```

Syntax 3:

```
jQuery(functionname);
```

Example on Document.Ready()

```

<html>
  <head>
    <title>jQuery - Document.ready</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>

    <script src="jquery-3.2.1.js"></script>

    <script>
      //$(document).ready(fun1);
      $fun1;
      //jQuery(fun1);
      function fun1()
      {
        alert("Page loaded");
      }
    </script>
  </head>
  <body>
    <h1>Page</h1>
    <div id="div1">div 1</div>
  </body>
</html>

```

Selectors

- “Selector” is a syntax to select the desired elements in the web page.
- We have to select the elements, and then we can manipulate them.
- We have to pass the selector as argument to the \$() function. The \$() function receives the selector, searches the web page and returns the matching elements in the web page.

Syntax: `$(“selector”)`

Example: `$(“p”)`

List of jQuery Selectors

1. Tag Selector
2. ID Selector
3. Class Selector

4. Compound Selector
5. Grouping Selector
6. Child Selector
7. Direct Child Selector
8. Adjacent Siblings Selector
9. Adjacent One Sibling Selector
10. :first filter
11. :last filter
12. :even filter
13. :odd filter
14. :eq filter
15. :gt filter
16. :lt filter
17. :not filter
18. Attribute Selector
19. Attribute Selector – Not
20. Attribute Selector – Starts With
21. Attribute Selector – Ends With
22. Attribute Selector – Contains
23. Contains
24. Has
25. Empty
26. :first-child filter
27. :last-child filter
28. :nth-child filter
29. :only-child filter
30. parent()
31. next()
32. prev()
33. siblings()
34. children()
35. index()
36. :input filter
37. :text filter

38. :password filter
39. :radio filter
40. :checkbox filter
41. :image filter
42. :file filter
43. :submit filter
44. :reset filter
45. :button filter
46. :text:disabled filter
47. :radio:checked filter
48. :checkbox:checked filter

Tag Selector

- It selects all the instances of the specified tag.

Syntax: `$(“tag”)`

Example: `$(“p”)`

Example on Tag Selector

```
<html>
  <head>
    <title>jQuery - Tag Selector</title>
  </head>
  <body>
    <p>para 1</p>
    <p>para 2</p>
    <p>para 3</p>
    <p>para 4</p>
    <p>para 5</p>
    <input type="button" value="Select" id="button1">

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("p").css("background-color", "lightgreen"); //tag selector
        //It selects all <p> tags
        //Selector = Syntax to select
      }
    </script>
  </body>
```

</html>

ID Selector

- It selects a single element based on the ID.

Syntax: `$("#id")`

Example: `$("#p1")`

Example on ID Selector

```

<html>
  <head>
    <title>jQuery - ID Selector</title>
  </head>
  <body>
    <p>para 1</p>
    <p>para 2</p>
    <p id="p3">para 3</p>
    <p>para 4</p>
    <p>para 5</p>
    <input type="button" value="Select" id="button1">

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("#p3").css("background-color", "lightgreen"); //id selector
        //It selects the <p> tag that has id="p3"
      }
    </script>
  </body>
</html>

```

Class Selector

- It selects all the instances based on the class name.

Syntax: `$(".class")`

Example: `(".c1")`

Example on Class Selector

```

<html>
  <head>
    <title>jQuery - Class Selector</title>
  </head>
  <body>

```

```

<p class="c1">para 1</p>
<p>para 2</p>
<p class="c1">para 3</p>
<p class="c1">para 4</p>
<p>para 5</p>
<input type="button" value="Select" id="button1">

<script src="jquery-3.2.1.js"></script>

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    $(".c1").css("background-color", "lightgreen"); //class selector
    //It selects the tags that have class="class1"
  }
</script>
</body>
</html>

```

Example 2 on Class Selector

```

<html>
  <head>
    <title>jQuery - Class Selector 2</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <p class="c1">para 1</p>
    <p>para 2</p>
    <p class="c1">para 3</p>
    <p class="c1">para 4</p>
    <p>para 5</p>
    <div class="c1">div1</div>
    <div>div2</div>
    <div class="c1">div3</div>
    <div>div4</div>
    <input type="button" value="Select" id="button1">
    <script src="jquery-3.2.1.js"></script>
    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $(".c1").css("background-color", "lightgreen"); //class selector
        //It selects the <p> tags and <div> tags that have class="c1"
      }
    </script>
  </body>

```

```
</html>
```

Compound Selector

- It selects the element that has specified tag and specified class name.

Syntax: `$(“tag.class”)`

Example: `$(“p.c1”)`

Example on Compound Selector

```
<html>
  <head>
    <title>jQuery - Compound Selector</title>
  </head>
  <body>
    <p class="c1">para 1</p>
    <p>para 2</p>
    <p class="c1">para 3</p>
    <p class="c1">para 4</p>
    <p>para 5</p>
    <div class="c1">div1</div>
    <div>div2</div>
    <div class="c1">div3</div>
    <div>div4</div>
    <input type="button" value="Select" id="button1">
    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("p.c1").css("background-color", "lightgreen");
      }
    </script>
  </body>
</html>
```

Grouping Selector

- It selects all the specified group of tags.

Syntax: `$(“tag1,tag2,tag3,...”)`

Example: `$(“div,p,h2,span”)`

Example on Grouping Selector

```
<html>
  <head>
    <title>jQuery - Grouping Selector</title>
```

```

<style type="text/css">
  body, input
  {
    font-family: 'Tahoma';
    font-size: 30px;
  }
  div
  {
    margin-bottom: 5px;
  }
</style>
</head>
<body>
  <p>para 1</p>
  <p>para 2</p>
  <div>div1</div>
  <div>div2</div>
  <div>div3</div>
  <div>div4</div>
  <span>span</span>
  <span>span</span>
  <span>span</span>
  <br>
  <input type="button" value="Select" id="button1">

<script src="jquery-3.2.1.js"></script>

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    $("p,div,span").css("background-color", "lightgreen"); //grouping selector
    //It selects all <div> tags, <p> tags and all <span> tags
  }
</script>
</body>
</html>

```

Child Selector (or) Descendent Selector

- It selects all the child tags of the specified parent tag.
- It selects all the child tags including grand children.

Syntax: `$(“parent child”)`

Example: `$(“div p”)`

Example on Child Selector / Descendent Selector

```

<html>
  <head>
    <title>jQuery - Descendent Selector</title>
    <style type="text/css">

```

```

body,input
{
    font-family: 'Tahoma';
    font-size: 30px;
}
div
{
    margin-bottom: 5px;
}
</style>
</head>
<body>
<div>
<p>para 1</p>
<p>para 2</p>
<p>para 3</p>
</div>
<p>para 4</p>
<p>para 5</p>
<p>para 6</p>
<input type="button" value="Select" id="button1">

<script src="jquery-3.2.1.js"></script>

<script>
$("#button1").click(fun1);
function fun1()
{
    $("div p").css("background-color", "lightgreen"); //Descendent Selector or Child Selector
    //It selects all the <p> tags that are children of <div>
}
</script>
</body>
</html>

```

Example 2 on Child Selector / Descendent Selector

```

<html>
<head>
    <title>jQuery - Descendent Selector 2</title>
    <style type="text/css">
        body,input
        {
            font-family: 'Tahoma';
            font-size: 30px;
        }
        div
        {
            margin-bottom: 5px;
        }
    </style>
</head>
<body>
<div>

```

```

<p>para 1</p>
<p>para 2</p>
<b>
    <p>para 3</p>
    <p>para 4</p>
</b>
</div>
<p>para 5</p>
<p>para 6</p>
<p>para 7</p>
<input type="button" value="Select" id="button1">
<script src="jquery-3.2.1.js"></script>
<script>
    $("#button1").click(fun1);
    function fun1()
    {
        $("div p").css("background-color", "lightgreen"); //Descendent Selector or Child Selector (including grand children)
        //It selects all the <p> tags that are direct children or grand children of <div>
    }
</script>
</body>
</html>

```

Direct Child Selector

- It selects all the child tags that are direct children of the specified parent tag.
- It selects the child tags excluding grand children.

Syntax: `$(“parent>child”)`

Example: `$(“div>p”)`

Example on Direct Child Selector

```

<html>
    <head>
        <title>jQuery - Direct Child Selector</title>
        <style type="text/css">
            body,input
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
            div
            {
                margin-bottom: 5px;
            }
        </style>
    </head>
    <body>

```

```

<div>
  <p>para 1</p>
  <p>para 2</p>
  <p>para 3</p>
  <b>
    <p>para 4</p>
    <p>para 5</p>
  </b>
</div>
<p>para 6</p>
<p>para 7</p>
<p>para 7</p>
<input type="button" value="Select" id="button1">
<script src="jquery-3.2.1.js"></script>
<script>
  $("#button1").click(fun1);
  function fun1()
  {
    $("div>p").css("background-color", "lightgreen"); //Direct Child Selector (excluding grand children)
    //It selects all the <p> tags that are direct children of <div>
  }
</script>
</body>
</html>

```

Adjacent Siblings Selector

- It selects all the sibling tags, which are adjacent to the current tag.

Syntax: `$(“currenttag~siblingtag”)`

Example: `$("#p2~p")`

Example on Adjacent Siblings Selector

```

<html>
  <head>
    <title>jQuery - Adjacent Siblings Selector</title>
  </head>
  <body>
    <div id="div1">
      <p>para 1</p>
      <p id="p2">para 2</p>
      <p>para 3</p>
      <p>para 4</p>
      <p>para 5</p>
      <p>para 6</p>
    </div>
    <input type="button" value="Select" id="button1">

    <script src="jquery-3.2.1.js"></script>

    <script>

```

```

$( "#button1" ).click(fun1);
function fun1()
{
    $("#p2~p").css("border", "10px solid lightgreen");
}
</script>
</body>
</html>

```

Note: Place “img4.jpg”, “img5.jpg”, “img6.jpg” in the current folder (c:\jquery).

Adjacent One Sibling Selector

- It selects the only one sibling element, which is immediate adjacent to the current element.

Syntax: \$(“currenttag+Siblingtag”)

Example: \$(“#p2+p”)

Example on Adjacent One Sibling Selector

```

<html>
    <head>
        <title>jQuery - Adjacent One Sibling Selector</title>
    </head>
    <body>
        <div id="div1">
            <p>para 1</p>
            <p id="p2">para 2</p>
            <p>para 3</p>
            <p>para 4</p>
            <p>para 5</p>
            <p>para 6</p>
        </div>
        <input type="button" value="Select" id="button1">
        <script src="jquery-3.2.1.js"></script>

        <script>
            $('#button1').click(fun1);
            function fun1()
            {
                $("#p2+p").css("border", "10px solid lightgreen");
            }
        </script>
    </body>
</html>

```

Note: Place “img4.jpg”, “img5.jpg”, “img6.jpg” in the current folder (c:\jquery).

:first filter

- It selects the first element.

Syntax: `$(“tag:first”)`

Example: `$(“p:first”)`

Example on :first filter

```
<html>
  <head>
    <title>jQuery - First</title>
  </head>
  <body>
    <p>para 1</p>
    <p>para 2</p>
    <p>para 3</p>
    <p>para 4</p>
    <p>para 5</p>
    <p>para 6</p>
    <p>para 7</p>
    <p>para 8</p>
    <p>para 9</p>
    <p>para 10</p>
    <input type="button" value="Select" id="button1">
    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("p:first").css("background-color", "lightgreen");
      }
    </script>
  </body>
</html>
```

:last filter

- It selects the last element.

Syntax: `$(“tag:last”)`

Example: `$(“p:last”)`

Example on :last filter

```
<html>
  <head>
    <title>jQuery - Last</title>
  </head>
  <body>
```

```

<p>para 1</p>
<p>para 2</p>
<p>para 3</p>
<p>para 4</p>
<p>para 5</p>
<p>para 6</p>
<p>para 7</p>
<p>para 8</p>
<p>para 9</p>
<p>para 10</p>
<input type="button" value="Select" id="button1">
<script src="jquery-3.2.1.js"></script>

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    $("p:last").css("background-color", "lightgreen");
  }
</script>
</body>
</html>

```

:even filter

- It selects all the even elements (0, 2, 4, 6, ...) etc.

Syntax: `$(“tag:even”)`

Example: `$(“p:even”)`

Example on :even filter

```

<html>
  <head>
    <title>jQuery - Even</title>
  </head>
  <body>
    <p>para 1</p>
    <p>para 2</p>
    <p>para 3</p>
    <p>para 4</p>
    <p>para 5</p>
    <p>para 6</p>
    <p>para 7</p>
    <p>para 8</p>
    <p>para 9</p>
    <p>para 10</p>
    <input type="button" value="Select" id="button1">
    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);

```

```

function fun1()
{
    $("p:even").css("background-color", "lightgreen");
}
</script>
</body>
</html>

```

:odd filter

- It selects all the odd elements (1, 3, 5, 7, ...) etc.

Syntax: `$(“tag:odd”)`

Example: `$(“p:odd”)`

Example on :odd filter

```

<html>
    <head>
        <title>jQuery - Odd</title>
    </head>
    <body>
        <p>para 1</p>
        <p>para 2</p>
        <p>para 3</p>
        <p>para 4</p>
        <p>para 5</p>
        <p>para 6</p>
        <p>para 7</p>
        <p>para 8</p>
        <p>para 9</p>
        <p>para 10</p>
        <input type="button" value="Select" id="button1">

        <script src="jquery-3.2.1.js"></script>

        <script>
            $("#button1").click(fun1);
            function fun1()
            {
                $("p:odd").css("background-color", "lightgreen");
            }
        </script>
    </body>
</html>

```

:eq filter

- It selects the single element, based on the specified index.

Syntax: `$(“tag:eq(n)”)`

Example: `$(“p:eq(n)”)`

Example on :eq filter

```

<html>
  <head>
    <title>jQuery - Eq</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <p>para 1</p>
    <p>para 2</p>
    <p>para 3</p>
    <p>para 4</p>
    <p>para 5</p>
    <p>para 6</p>
    <p>para 7</p>
    <p>para 8</p>
    <p>para 9</p>
    <p>para 10</p>
    <input type="button" value="Select" id="button1">
    <script src="jquery-3.2.1.js"></script>
    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("p:eq(3)").css("background-color", "lightgreen"); //equal - filter
        // $("p").eq(3).css("background-color", "lightgreen"); //equal - function
        // It selects the <p> which index position is "3". Index starts from zero (0).
      }
    </script>
  </body>
</html>

```

:gt filter

- It selects the elements where index is greater than specified index.

Syntax: `$(“tag:gt(n)”)`

Example: `$(“p:gt(n)”)`

Example on :gt filter

```

<html>
  <head>
    <title>jQuery - Gt</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <p>para 1</p>
    <p>para 2</p>
    <p>para 3</p>
    <p>para 4</p>
    <p>para 5</p>
    <p>para 6</p>
    <p>para 7</p>
    <p>para 8</p>
    <p>para 9</p>
    <p>para 10</p>
    <input type="button" value="Select" id="button1">

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("p:gt(3)").css("background-color", "lightgreen"); //greater than - filter
        //It selects the <p> which index position is greater than "3". Index starts from zero (0).
      }
    </script>
  </body>
</html>

```

:lt filter

- It selects the elements where index is less than specified index.

Syntax: `$(“tag:lt(n)”)`

Example: `$(“p:lt(n)”)`

Example on :lt filter

```

<html>
  <head>
    <title>jQuery - Lt</title>
    <style type="text/css">
      body,input

```

```

    {
        font-family: 'Tahoma';
        font-size: 30px;
    }

```

</style>

</head>

<body>

<p>para 1</p>

<p>para 2</p>

<p>para 3</p>

<p>para 4</p>

<p>para 5</p>

<p>para 6</p>

<p>para 7</p>

<p>para 8</p>

<p>para 9</p>

<p>para 10</p>

<input type="button" value="Select" id="button1">

<script src="jquery-3.2.1.js"></script>

<script>

\$("#button1").click(fun1);

function fun1()

{

\$("p:lt(3)").css("background-color", "lightgreen"); //less than - filter

//It selects the <p> which index position is less than "3". Index starts from zero (0).

}

</script>

</body>

</html>

:not filter

- It selects all the tags, except the specified tags.

Syntax: `$(“selector:not(another selector)”)`

Example: `$(“p:not(#p3)”)`

Example on :not filter

```

<html>
    <head>
        <title>jQuery - Not</title>
        <style type="text/css">
            body,input
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
        </style>
    </head>

```

```

<body>
  <p>para 1</p>
  <p>para 2</p>
  <p>para 3</p>
  <p>para 4</p>
  <p>para 5</p>
  <p>para 6</p>
  <p>para 7</p>
  <p>para 8</p>
  <p>para 9</p>
  <p>para 10</p>
  <input type="button" value="Select" id="button1">
  <script src="jquery-3.2.1.js"></script>
  <script>
    $("#button1").click(fun1);
    function fun1()
    {
      $("p:not(p:eq(3))").css("background-color", "lightgreen"); //not - filter
      //It selects the <p> which index position is not equal to "3".
      //$("p: eq(3),p: eq(5)").css("background-color", "lightgreen");
      //It selects the <p> which index position is equal to "3" and "5".
      //$("p: not(p: eq(3),p: eq(5))").css("background-color", "lightgreen"); //not - filter
      //It selects the <p> which index position is not equal to "3" and "5".
    }
  </script>
</body>
</html>

```

Attribute Selector

- It selects all the elements that have specified attribute.

Syntax: `$(“tag[attribute='value']”)`

Example: `$(“img[src='img1.jpg']”)`

Example on Attribute Selector

```

<html>
  <head>
    <title>jQuery - Attribute Selector</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    
  
```

```






<br>
<input type="button" value="Select" id="button1">

<script src="jquery-3.2.1.js"></script>

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    $("img[width='100px']").css("border", "4px solid red"); //attribute selector
    //It selects the <img> tag that have an attribute called width="100px"
  }
</script>
</body>
</html>

```

Note: Place “img1.jpg”, “img2.jpg”, “img3.jpg”, “img4.jpg”, “img5.jpg”, “img6.jpg” in the current folder (c:\jquery).

Attribute Selector - Not

- It selects all the elements that are having the specified attribute, where the value is not equal to the specified value.

Syntax: `$(“tag[attribute!=’value’]”)`

Example: `$(“img[src!=’img1.jpg’]”)`

Example on Attribute Selector - Not

```

<html>
  <head>
    <title>jQuery - Attribute Selector - Not</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    
    
    
    
  
```

```



<br>
<input type="button" value="Select" id="button1">

<script src="jquery-3.2.1.js"></script>

<script>
$("#button1").click(fun1);
function fun1()
{
    $("img[width!='100px']").css("border", "4px solid red"); //attribute selector - not
    //$("img:not(img[width='100px'])").css("border", "4px solid red"); //attribute selector - not
    //It selects the <img> tag that width is not equal to "100px"
}
</script>
</body>
</html>

```

Note: Place “img1.jpg”, “img2.jpg”, “img3.jpg”, “img4.jpg”, “img5.jpg”, “img6.jpg” in the current folder (c:\jquery).

Attribute Selector – Starts With

- It selects all the elements that have specified attribute, where the value starts with specified value.

Syntax: `$(“tag[attribute^=’value’]”)`

Example: `$(“img[src^=’i’]”)`

Example on Attribute Selector – Starts With

```

<html>
  <head>
    <title>jQuery - Attribute Selector - Starts With</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    
    
    
    
    
    
    
    
    <br>

```

```

<input type="button" value="Select" id="button1">

<script src="jquery-3.2.1.js"></script>

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    $("img[src^='wa']").css("border", "4px solid red"); //attribute selector - starts with
    //It selects <img> tag that has "src" attribute that starts with "wa".
  }
</script>
</body>
</html>

```

Note: Place “img1.jpg”, “img2.jpg”, “img3.jpg”, “img4.jpg”, “img5.jpg”, “img6.jpg”, “Spring.jpg”, “Water.jpg” in the current folder (c:\jquery).

Attribute Selector – Ends With

- It selects all the elements that have specified attribute, where the value ends with specified value.

Syntax: \$(“tag[attribute\$=’value’]”)

Example: \$(“img[src\$=’jpg’]”)

Example on Attribute Selector – Ends With

```

<html>
  <head>
    <title>jQuery - Attribute Selector - Ends With</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    
    
    
    
    
    
    
    
    
    
    <br>
    <input type="button" value="Select" id="button1">
  </body>

```

```

<script src="jquery-3.2.1.js"></script>

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    $("img[src$='jpg']").css("border", "4px solid red"); //attribute selector - ends with
    //It selects <img> tag that has "src" attribute that ends with "jpg".
  }
</script>
</body>
</html>

```

Note: Place “img1.jpg”, “img2.jpg”, “img3.jpg”, “img4.jpg”, “img5.jpg”, “img6.jpg”, “Spring.jpg”, “Water.jpg”, “apple.gif”, “earth.gif” in the current folder (c:\jquery).

Attribute Selector - Contains

- It selects all the elements that have specified attribute, where the value contains specified value.

Syntax: `$(“tag[attribute*=’value’]”)`

Example: `$(“img[src*=j]”)`

Example on Attribute Selector – Contains

```

<html>
  <head>
    <title>jQuery - Attribute Selector - Contains</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    
    
    
    
    
    
    
    
    
    
    <br>
    <input type="button" value="Select" id="button1">

  <script src="jquery-3.2.1.js"></script>

```

```

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    $("img[src*='5']").css("border", "4px solid red"); //attribute selector - contains
    //It selects <img> tag that has "src" attribute that contains "5".

    //$("#img[src*='5'],img[src*='2']").css("border", "4px solid red"); //attribute selector - contains
    //It selects <img> tag that has "src" attribute that contains "5" or "2".
  }
</script>
</body>
</html>

```

Note: Place “img1.jpg”, “img2.jpg”, “img3.jpg”, “img4.jpg”, “img5.jpg”, “img6.jpg”, “Spring.jpg”, “Water.jpg”, “apple.gif”, “earth.gif” in the current folder (c:\jquery).

:contains filter

- It selects the elements that have inner html that contains specified value.

Syntax: `$(“tag:contains(‘value’)”)`

Example: `$(“p:contains(‘Services’)”)`

Example on Contains Filter

```

<html>
  <head>
    <title>jQuery - Contains</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <p>Lorem Ipsum is simply dummy text of the printing.</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.</p>
    <p>Contrary to popular belief, Lorem Ipsum is not simply random text.</p>
    <p>It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old.</p>
    <p>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.</p>
    <p>The generated Lorem Ipsum is therefore always free from repetition, injected humour, or non-
characteristic words etc.</p>
    <p>Various versions have over the years, sometimes by accident, sometimes on purpose.</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.</p>
  </body>
</html>

```

```

<p>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.</p>
<p>All the Lorem Ipsum generators on the Internet tend to repeat predefined chunks as necessary, making
this the first true generator on the Internet.</p>
<input type="button" value="Select" id="button1">

<script src="jquery-3.2.1.js"></script>

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    $("p:contains('you')").css("background-color", "lightgreen"); //contains
    //It selects <p> that has content "you"
  }
</script>
</body>
</html>

```

:has filter

- It selects the elements that have child elements that matches with specified selector.

Syntax: `$(“tag:has(‘selector’)”)`

Example: `$(“p:has(‘span’)”)`

Example on Has Filter

```

<html>
  <head>
    <title>jQuery - Has</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <p>Lorem Ipsum is simply dummy text of the printing.</p>
    <p>It is a long established fact that a reader <span>will be distracted</span> by the readable content of a
page when looking at its layout.</p>
    <p>Contrary to popular belief, Lorem Ipsum is not simply random text.</p>
    <p>It has roots in a piece of classical Latin literature from <span>45 BC</span>, making it over 2000 years
old.</p>
    <p>If you are going to use a passage of <span>Lorem Ipsum</span>, you need to be sure there isn't anything
embarrassing hidden in the middle of text.</p>
    <p>The generated Lorem Ipsum is therefore always free from repetition, injected humour, or non-
characteristic words etc.</p>
    <p>Various versions have over the years, sometimes by accident, sometimes on purpose.</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.</p>

<p>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.</p>

<p>All the Lorem Ipsum generators on the Internet tend to repeat predefined chunks as necessary, making this the first true generator on the Internet.</p>

```
<input type="button" value="Select" id="button1">

<script src="jquery-3.2.1.js"></script>

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    $("p:has('span')").css("background-color", "lightgreen"); //has
    //It selects the <p> tags that have a child called <span>
  }
</script>
</body>
</html>
```

:empty filter

- It selects the elements where inner html is empty.

Syntax: \$("tag:empty")

Example: \$("p:empty")

Example on Empty Filter

```
<html>
  <head>
    <title>jQuery - Empty</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      p
      {
        padding: 5px;
      }
    </style>
  </head>
  <body>
    <p>Lorem Ipsum is simply dummy text of the printing.</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.</p>
    <p></p>
    <p>It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old.</p>
  </body>
</html>
```

```

<p>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.</p>
<p></p>
<p>Various versions have over the years, sometimes by accident, sometimes on purpose.</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.</p>
<p></p>
<p>All the Lorem Ipsum generators on the Internet tend to repeat predefined chunks as necessary, making this the first true generator on the Internet.</p>
<input type="button" value="Select" id="button1">

<script src="jquery-3.2.1.js"></script>

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    $("p:empty").css("border", "3px solid red"); //empty - filter
    //It selects the empty <p> tags
  }
</script>
</body>
</html>

```

:first-child filter

- It selects all the elements that are first child of its parent.

Syntax: `$(“tag:first-child”)`

Example: `$(“p:first-child”)`

Example on :first-child Filter

```

<html>
  <head>
    <title>jQuery - First-Child</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      #div1
      {
        background-color: #ffccff;
      }
      #div2
      {
        background-color: #ff9966;
      }
      #div3
      {

```

```

        background-color: #99ffcc;
    }

```

```

</style>
</head>
<body>
    <div id="div1">
        <p>para 1</p>
        <p>para 2</p>
        <p>para 3</p>
        <p>para 4</p>
        <p>para 5</p>
    </div>
    <div id="div2">
        <p>para 6</p>
        <p>para 7</p>
        <p>para 8</p>
        <p>para 9</p>
        <p>para 10</p>
    </div>
    <div id="div3">
        <p>para 11</p>
        <p>para 12</p>
        <p>para 13</p>
        <p>para 14</p>
        <p>para 15</p>
    </div>
    <input type="button" value="Select" id="button1">
<script src="jquery-3.2.1.js"></script>
<script>
    $("#button1").click(fun1);
    function fun1()
    {
        $("p:first-child").css("background-color", "lightgreen"); //first-child
        //It selects the <p> tag, which is the first child or its parent.
    }
</script>
</body>
</html>

```

:last-child filter

- It selects all the elements that are last child of its parent.

Syntax: `$(“tag:last-child”)`

Example: `$(“p:last-child”)`

Example on :last-child Filter

```

<html>
    <head>
        <title>jQuery - Last-Child</title>
        <style type="text/css">

```

```

body,input
{
    font-family: 'Tahoma';
    font-size: 30px;
}
#div1
{
    background-color: #ffccff;
}
#div2
{
    background-color: #ff9966;
}
#div3
{
    background-color: #99ffcc;
}
</style>
</head>
<body>
    <div id="div1">
        <p>para 1</p>
        <p>para 2</p>
        <p>para 3</p>
        <p>para 4</p>
        <p>para 5</p>
    </div>
    <div id="div2">
        <p>para 6</p>
        <p>para 7</p>
        <p>para 8</p>
        <p>para 9</p>
        <p>para 10</p>
    </div>
    <div id="div3">
        <p>para 11</p>
        <p>para 12</p>
        <p>para 13</p>
        <p>para 14</p>
        <p>para 15</p>
    </div>
    <input type="button" value="Select" id="button1">

<script src="jquery-3.2.1.js"></script>

<script>
    $("#button1").click(fun1);
    function fun1()
    {
        $("p:last-child").css("background-color", "lightgreen"); //last-child
        //It selects the <p> tag, which is the last child or its parent.
    }
</script>
</body>
</html>

```

:nth-child filter

- It selects all the elements that are n^{th} child of its parent.

Syntax: `$(“tag:nth-child(n)”)`

Example: `$(“p:nth-child(2)”)`

Note: Index starts from ‘1’.

Example on :nth-child Filter

```
<html>
  <head>
    <title>jQuery - Nth-Child</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      #div1
      {
        background-color: #ffccff;
      }
      #div2
      {
        background-color: #ff9966;
      }
      #div3
      {
        background-color: #99ffcc;
      }
    </style>
  </head>
  <body>
    <div id="div1">
      <p>para 1</p>
      <p>para 2</p>
      <p>para 3</p>
      <p>para 4</p>
      <p>para 5</p>
    </div>
    <div id="div2">
      <p>para 6</p>
      <p>para 7</p>
      <p>para 8</p>
      <p>para 9</p>
      <p>para 10</p>
    </div>
    <div id="div3">
```

```

<p>para 11</p>
<p>para 12</p>
<p>para 13</p>
<p>para 14</p>
<p>para 15</p>
</div>
<input type="button" value="Select" id="button1">

<script src="jquery-3.2.1.js"></script>

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    $("p:nth-child(2)").css("background-color", "lightgreen"); //nth-child. Index starts from '1'.
    //It selects the <p> tag, which index position is "2" in its parent.
  }
</script>
</body>
</html>

```

:only-child filter

- It selects the child element, which is only one child of its parent.

Syntax: `$(“tag:only-child”)`

Example: `$(“p:only-child”)`

Example on :only-child Filter

```

<html>
  <head>
    <title>jQuery - Only-child</title>
    <style type="text/css">
      #div1
      {
        background-color: #ffccff;
      }
      #div2
      {
        background-color: #ff9966;
      }
      #div3
      {
        background-color: #99ffcc;
      }
      #div4
      {
        background-color: #3399cc;
      }
      #div5
      {

```

```

        background-color: #ffff99;
    }

```

```

</style>
</head>
<body>
    <div id="div1">
        <p>para 1</p>
        <p>para 2</p>
        <p>para 3</p>
        <p>para 4</p>
    </div>
    <div id="div2">
        <p>para 5</p>
    </div>
    <div id="div3">
        <p>para 6</p>
        <p>para 7</p>
    </div>
    <div id="div4">
        <p>para 8</p>
        <p>para 9</p>
    </div>
    <div id="div5">
        <p>para 10</p>
    </div>
    <div id="div6">
        <p>para 11</p>
        <p>para 12</p>
        <p>para 13</p>
        <p>para 14</p>
        <p>para 15</p>
    </div>
    <input type="button" value="Select" id="button1">
<script src="jquery-3.2.1.js"></script>
<script>
    $("#button1").click(fun1);
    function fun1()
    {
        $("p:only-child").css("background-color", "lightgreen"); //only-child
        //It selects the <p> tag that is only one child of its parent.
    }
</script>
</body>
</html>

```

parent()

- It selects the parent element of the current element.

Syntax: parent()

Example: \$("#p1").parent()

Example on parent()

```

<html>
  <head>
    <title>jQuery - Parent</title>
  </head>
  <body>
    <div id="div1">
      <p>para 1</p>
      <p id="p2">para 2</p>
      <p>para 3</p>
      <p>para 4</p>
    </div>
    <input type="button" value="Select" id="button1">

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("#p2").parent().css("background-color", "lightgreen");
        //The "parent()" function returns the parent tag of "#p1"
      }
    </script>
  </body>
</html>

```

next()

- It selects the next element of the current element.

Syntax: `next()`

Example: `$("#p1").next()`

Example on next()

```

<html>
  <head>
    <title>jQuery - Next</title>
  </head>
  <body>
    <div id="div1">
      <p>para 1</p>
      <p id="p2">para 2</p>
      <p>para 3</p>
      <p>para 4</p>
    </div>
    <input type="button" value="Select" id="button1">

    <script src="jquery-3.2.1.js"></script>

    <script>

```

```

$( "#button1" ).click( fun1 );
function fun1()
{
    $( "#p2" ).next().css("background-color", "lightgreen");
    //The "next()" function returns next tag, which is present after "#p2".
}
</script>
</body>
</html>

```

prev()

- It selects the previous element of the current element.

Syntax: prev()

Example: \$(“#p2”).prev()

Example on prev()

```

<html>
    <head>
        <title>jQuery - Prev</title>
    </head>
    <body>
        <div id="div1">
            <p>para 1</p>
            <p>para 2</p>
            <p id="p3">para 3</p>
            <p>para 4</p>
            <p>para 5</p>
        </div>
        <input type="button" value="Select" id="button1">
        <script src="jquery-3.2.1.js"></script>
        <script>
            $("#button1").click(fun1);
            function fun1()
            {
                $("#p3").prev().css("background-color", "lightgreen");
                //The "prev()" function returns the previous tag, which is present before "#p3".
            }
        </script>
    </body>
</html>

```

siblings()

- It selects all the sibling elements of the current element.

Syntax: `siblings()`

Example: `$("#p2").siblings()`

Example on `siblings()`

```
<html>
  <head>
    <title>jQuery - Siblings</title>
  </head>
  <body>
    <div id="div1">
      <p>para 1</p>
      <p>para 2</p>
      <p id="p3">para 3</p>
      <p>para 4</p>
      <p>para 5</p>
    </div>
    <p>para 6</p>
    <p>para 7</p>
    <p>para 8</p>
    <input type="button" value="Select" id="button1">

    <script src="jquery-3.2.1.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("#p3").siblings().css("background-color", "lightgreen");
        //It selects all tags that are siblings (brothers) of "#p1".
      }
    </script>
  </body>
</html>
```

`children()`

- It selects all the child tags of the specified parent tag.

Syntax: `children()`

Example: `$("#div1").children()`

Example on `children()`

```
<html>
  <head>
    <title>jQuery - Children</title>
  </head>
```

```

<body>
  <div id="div1">
    <p>para 1</p>
    <p>para 2</p>
    <p>para 3</p>
    <p>para 4</p>
    <p>para 5</p>
  </div>
  <p>para 6</p>
  <p>para 7</p>
  <p>para 8</p>
  <input type="button" value="Select" id="button1">
  <script src="jquery-3.2.1.js"></script>
  <script>
    $("#button1").click(fun1);
    function fun1()
    {
      $("#div1").children().css("background-color", "lightgreen");
      //It selects all tags that are children of "#div1".
    }
  </script>
</body>
</html>

```

index()

- It returns the index of the current element.

Syntax: index()

Example: \$("#p2").index()

Example on index()

```

<html>
  <head>
    <title>jQuery - Index</title>
  </head>
  <body>
    <p>para 1</p>
    <p>para 2</p>
    <p>para 3</p>
    <p id="p4">para 4</p>
    <p>para 5</p>
    <p>para 6</p>
    <p>para 7</p>
    <p>para 8</p>
    <input type="button" value="Index" id="button1">

    <script src="jquery-3.2.1.js"></script>

```

```

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    var n=$("#p4").index();
    alert(n);
  }
</script>
</body>
</html>

```

Form Filters

- Form filters are used to select the various form elements, such as textboxes, checkbox, radio buttons, dropdownlists etc.

• :input	It selects all <input>, <textarea> and <select> tags.
• :text	It selects all <input type="text">
• :password	It selects all <input type="password">
• :submit	It selects all <input type="submit">
• :reset	It selects all <input type="reset">
• :checkbox	It selects all <input type="checkbox">
• :radio	It selects all <input type="radio">
• :file	It selects all <input type="file">
• :image	It selects all <input type="image">
• :hidden	It selects all <input type="hidden">
• :button	It selects all <input type="button">
• :checkbox:checked	It selects all <input type="checkbox"> that are currently checked
• :radio:checked	It selects all <input type="radio"> that are currently checked
• :text:enabled	It selects all <input type="text"> that are currently enabled
• :text:disabled	It selects all <input type="text"> that are currently disabled

Example on Form Filters

```

<html>
  <head>
    <title>jQuery - Form Filters</title>
    <style type="text/css">
      body,input,textarea,select,table
      {
        font-family: 'Tahoma';
        font-size: 25px;
      }

```

```
</style>
</head>
<body>
    <h2>Form Filters</h2>
    <form action="server.aspx" method="post" name="frm">
        <table>
            <tr>
                <td>first name</td>
                <td>:</td>
                <td colspan="4"><input type="text" name="first name" class="color" size="35"></td>
            </tr>
            <tr>
                <td>middle name</td>
                <td>:</td>
                <td colspan="4"><input type="text" name="middle name" class="color" size="35"></td>
            </tr>
            <tr>
                <td>last name</td>
                <td>:</td>
                <td colspan="4"><input type="text" name="last name" class="color" size="35" disabled="disabled"></td>
            </tr>
            <tr>
                <td>password</td>
                <td>:</td>
                <td colspan="3"><input type="password" name="password" class="color" size="35"></td>
            </tr>
            <tr>
                <td>date of birth</td>
                <td>:</td>
                <td>
                    <select class="color" name="month">
                        <option value="na">Month</option>
                        <option value="1">January</option>
                        <option value="2">February</option>
                        <option value="3">March</option>
                        <option value="4">April</option>
                        <option value="5">May</option>
                        <option value="6">June</option>
                        <option value="7">July</option>
                        <option value="8">August</option>
                        <option value="9">September</option>
                        <option value="10">October</option>
                        <option value="11">November</option>
                        <option value="12">December</option>
                    </select>
                </td>
                <td>
                    <select name="day" id="day" class="color">
                        <option value="na">Day</option>
                    </select>
                </td>
                <td>
                    <select name="year" class="color">
                        <option value="na">Year</option>
                    </select>
                </td>
            </tr>
        </table>
    </form>
</body>
```

```
<option value="2015">2018</option>
<option value="2015">2017</option>
<option value="2015">2016</option>
<option value="2015">2015</option>
<option value="2014">2014</option>
<option value="2013">2013</option>
<option value="2012">2012</option>
<option value="2011">2011</option>
<option value="2010">2010</option>
<option value="2009">2009</option>
<option value="2008">2008</option>
<option value="2007">2007</option>
<option value="2006">2006</option>
<option value="2005">2005</option>
<option value="2004">2004</option>
<option value="2003">2003</option>
<option value="2002">2002</option>
<option value="2001">2001</option>
<option value="2000">2000</option>
<option value="1999">1999</option>
<option value="1998">1998</option>
<option value="1997">1997</option>
<option value="1996">1996</option>
<option value="1995">1995</option>
<option value="1994">1994</option>
<option value="1993">1993</option>
<option value="1992">1992</option>
<option value="1991">1991</option>
<option value="1990">1990</option>
<option value="1989">1989</option>
<option value="1988">1988</option>
<option value="1987">1987</option>
<option value="1986">1986</option>
<option value="1985">1985</option>
<option value="1984">1984</option>
<option value="1983">1983</option>
<option value="1982">1982</option>
<option value="1981">1981</option>
<option value="1980">1980</option>
<option value="1979">1979</option>
<option value="1978">1978</option>
<option value="1977">1977</option>
<option value="1976">1976</option>
<option value="1975">1975</option>
<option value="1974">1974</option>
<option value="1973">1973</option>
<option value="1972">1972</option>
<option value="1971">1971</option>
<option value="1970">1970</option>
<option value="1969">1969</option>
<option value="1968">1968</option>
<option value="1967">1967</option>
<option value="1966">1966</option>
<option value="1965">1965</option>
<option value="1964">1964</option>
```

```
<option value="1963">1963</option>
<option value="1962">1962</option>
<option value="1961">1961</option>
<option value="1960">1960</option>
<option value="1959">1959</option>
<option value="1958">1958</option>
<option value="1957">1957</option>
<option value="1956">1956</option>
<option value="1955">1955</option>
<option value="1954">1954</option>
<option value="1953">1953</option>
<option value="1952">1952</option>
<option value="1951">1951</option>
<option value="1950">1950</option>
<option value="1949">1949</option>
<option value="1948">1948</option>
<option value="1947">1947</option>
<option value="1946">1946</option>
<option value="1945">1945</option>
<option value="1944">1944</option>
<option value="1943">1943</option>
<option value="1942">1942</option>
<option value="1941">1941</option>
<option value="1940">1940</option>
<option value="1939">1939</option>
<option value="1938">1938</option>
<option value="1937">1937</option>
<option value="1936">1936</option>
<option value="1935">1935</option>
<option value="1934">1934</option>
<option value="1933">1933</option>
<option value="1932">1932</option>
<option value="1931">1931</option>
<option value="1930">1930</option>
<option value="1929">1929</option>
<option value="1928">1928</option>
<option value="1927">1927</option>
<option value="1926">1926</option>
<option value="1925">1925</option>
<option value="1924">1924</option>
<option value="1923">1923</option>
<option value="1922">1922</option>
<option value="1921">1921</option>
<option value="1920">1920</option>
<option value="1919">1919</option>
<option value="1918">1918</option>
<option value="1917">1917</option>
<option value="1916">1916</option>
<option value="1915">1915</option>
<option value="1914">1914</option>
<option value="1913">1913</option>
<option value="1912">1912</option>
<option value="1911">1911</option>
<option value="1910">1910</option>
<option value="1909">1909</option>
```

```

        </select>
    </td>
</tr>
<tr>
    <td>hidden</td>
    <td>:</td>
    <td colspan="4"><input type="hidden" name="hidden" class="color"></td>
</tr>
<tr>
    <td>gender</td>
    <td>:</td>
    <td colspan="4">
        <input type="radio" name="gender" value="male" class="color" checked="checked">
        male
        <input type="radio" name="gender" value="female" class="color">
        female
    </td>
</tr>
<tr>
    <td>address</td>
    <td>:</td>
    <td colspan="4"><textarea></textarea></td>

</tr>
<tr>
    <td>hobbies</td>
    <td>:</td>
    <td>
        <input type="checkbox" name="hobbies" value="movies" checked="checked" class="color">
        movies<br>
        <input type="checkbox" name="hobbies" value="games" class="color">
        games
    </td>
    <td>
        <p>
            <input type="checkbox" name="hobbies" value="sports" class="color">
            sports <br>
            <input type="checkbox" name="hobbies" value="books" class="color">
            books
        </p>
    </td>
</tr>
<tr>
    <td colspan="4">
        <center>
            <input name="image" type="image" src="submit.png" class="color" value="image">
        </center>
    </td>
</tr>
<tr>
    <td colspan="4">
        <center>
            <input type="file" value="file" class="color" name="file">
        </center>
    </td>

```

```

</tr>
<tr>
    <td colspan="4">
        <center>
            <input type="submit" value="submit" class="color" name="submit">
        </center>
    </td>
</tr>
<tr>
    <td colspan="4">
        <center>
            <input type="reset" value="reset" class="color" name="reset">
        </center>
    </td>
</tr>
<tr>
    <td colspan="4">
        <center>
            <input type="button" value="button" class="color" name="button">
            <br>
            <button>button</button>
        </center>
    </td>
</tr>
</table>
</form>
<input type="button" value="Select" id="button1">

<script src="jquery-3.2.1.js"></script>

<script>
    $("#button1").click(fun1);
    function fun1()
    {
        $("input").css("background-color", "green"); //input, select, textarea
        //$("#text").css("background-color", "green");
        //$("#password").css("background-color", "green");
        //$("#radio").css("background-color", "green")
        //$("#checkbox").css("background-color", "green")
        //$("#image").css("background-color", "green")
        //$("#file").css("background-color", "green")
        //$("#submit").css("background-color", "green")
        //$("#reset").css("background-color", "green")
        //$("#button").css("background-color", "green");
        //$("#text:disabled").css("background-color", "green").val("this is disabled");
        //$("#text:enabled").css("background-color", "green");
        //$("#radio:checked").css("background-color", "green")
        //$("#checkbox:checked").css("background-color", "green")
    }
</script>
</body>
</html>

```

Table Styles

- It is used to apply styles to the tables.

```
<html>
  <head>
    <title>jQuery - Table Styles</title>
    <style type="text/css">
      body,table
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      .class1
      {
        background-color: #ff3399;
      }
      .class2
      {
        background-color: #99ffff;
      }
      .class3
      {
        background-color: #33ccff;
      }
      .class4
      {
        padding: 5px;
      }
      .class5
      {
        background-color: #ffff33;
      }
      .class6
      {
        cursor: pointer;
        background-color: #0099ff;
      }
    </style>
  </head>
  <body>
    <table id="table1">
      <tr>
        <th>Item</th>
        <th>Price</th>
      </tr>
      <tr>
        <td>Milk</td>
        <td>1.99</td>
      </tr>
      <tr>
        <td>Eggs</td>
        <td>2.29</td>
      </tr>
```

```
<tr>
<td>Butter</td>
<td>3.49</td>
</tr>
<tr>
<td>Bread</td>
<td>0.99</td>
</tr>
<tr>
<td>Pasta</td>
<td>1.19</td>
</tr>
<tr>
<td>Honey</td>
<td>4.39</td>
</tr>
<tr>
<td>Cookies</td>
<td>2.99</td>
</tr>
<tr>
<td>Apples</td>
<td>0.59</td>
</tr>
<tr>
<td>Sugar</td>
<td>1.78</td>
</tr>
<tr>
<td>Pepper</td>
<td>1.56</td>
</tr>
</table>

<script src="jquery-3.2.1.js"></script>

<script>
$("#table1").addClass("class1");
$("#table1 tr:even").addClass("class2");
$("#table1 tr:odd").addClass("class3");
$("#table1 tr td, #table1 tr th").addClass("class4");
$("#table1 tr:first").addClass("class5");
$("#table1 tr:gt(0)").hover(fun1, fun2);
function fun1()
{
    $(this).addClass("class6");
}
function fun2()
{
    $(this).removeClass("class6");
}
</script>
</body>
</html>
```

Append – Advanced Examples

Append – Advanced Example 1

```
<html>
  <head>
    <title>jQuery - Append 1</title>
    <style type="text/css">
      body,select
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    Years:
    <select id="dropdownlist1">

    </select>

    <script src="jquery-3.2.1.js"></script>

    <script>
      for(var i=2017; i>=2000; i--)
      {
        var s = "<option>" + i + "</option>";
        $("#dropdownlist1").append(s);
      }
    </script>
  </body>
</html>
```

Append – Advanced Example 2

```
<html>
  <head>
    <title>jQuery - Append 2</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      #div1 p
      {
        font-family: Tahoma;
        font-size: 35px;
        background-color: orange;
        margin: 2px;
      }
    </style>
```

```

</head>
<body>
  <h3>Friends</h3>
  <div id="div1">
  </div>
  <input type="text" id="txt1">
  <input type="button" id="button1" value="+" title="Add">

<script src="jquery-3.2.1.js"></script>

<script>
  $("#button1").click(fun1);
  function fun1()
  {
    var s = "<p>" + $("#txt1").val() + "</p>";
    $("#div1").append(s);
    $("#txt1").val("").focus();
  }
</script>
</body>
</html>

```

Append – Advanced Example 3

```

<html>
  <head>
    <title>jQuery - Append 3</title>
    <style>
      #div1 input[type=text]
      {
        font-family: Calibri;
        font-size: 24px;
        background-color: lightgreen;
        margin: 2px;
      }
    </style>
  </head>
  <body>
    <h3>Type some numbers</h3>
    <div id="div1">
    </div>
    <input type="button" id="button1" value="Add">
    <span id="span1">Result here</span>

    <script src="jquery-3.2.1.js"></script>

    <script>
      for(var i=1; i <= 10; i++)
      {
        var s = "<input type='text' id='txt" + i + "'><br>";
        $("#div1").append(s);
      }
    </script>

```

```

var sum = 0;
$("#button1").click(fun1);
function fun1()
{
    var alltextboxes = $("#div1 input[type=text]");
    for (var i = 0; i < alltextboxes.length; i++)
    {
        sum += parseInt($("#alltextboxes[i]").val()); // 'this' means current text box
    }
    $("#span1").html(sum);
}
</script>
</body>
</html>

```

Append – Advanced Example 4

```

<html>
    <head>
        <title>jQuery - Append 4</title>
    </head>
    <body>
        <h3>Enter Employee Details</h3>
        <div id="div1">
            Emp ID:<br><input type="text" id="textbox1"><br>
            Emp Name:<br><input type="text" id="textbox2"><br>
            Salary:<br><input type="text" id="textbox3">
        </div>
        <input type="button" id="button1" value="Add">
        <table id="table1" border="1" cellpadding="5px">
            <tr style="background-color:gold">
                <th>Emp ID</th>
                <th>Emp Name</th>
                <th>Salary</th>
            </tr>
        </table>
        <script src="jquery-3.2.1.js"></script>
        <script>
            $("#button1").click(fun1);
            function fun1()
            {
                var s = "<tr><td>" + $("#textbox1").val() + "</td><td>" + $("#textbox2").val() + "</td><td>" +
                $("#textbox3").val() + "</td></tr>";
                $("#table1").append(s);
                $("#div1 input[type=text]").val("");
                $("#textbox1").focus();
            };
        </script>
    </body>
</html>

```

jQuery – JavaScript Objects - Examples

```

<html>
  <head>
    <title>jQuery - Object</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      span
      {
        font-weight: bold;
        color: green;
      }
    </style>
  </head>
  <body>
    <p>
      Emp ID: <span id="span1"></span><br>
      Emp Name: <span id="span2"></span><br>
      Salary: <span id="span3"></span><br>
    </p>

    <script src="jquery-3.2.1.js"></script>

    <script>
      var emp = { "empid":1, "empname": "Scott", "salary": 5000 };
      $("#span1").html(emp.empid);
      $("#span2").html(emp.empname);
      $("#span3").html(emp.salary);
    </script>
  </body>
</html>

```

jQuery – JavaScript Objects – Example 2

```

<html>
  <head>
    <title>jQuery - Object Array - UL</title>
    <style type="text/css">
      body,table
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <ul id="list1">
    </ul>

    <script src="jquery-3.2.1.js"></script>

```

```

<script>
    var employees =
    [
        {"empid":1, "empname": "Scott", "salary":5000},
        {"empid":2, "empname": "Allen", "salary":6000},
        {"empid":3, "empname": "John", "salary":7000},
        {"empid":4, "empname": "Jones", "salary":8000},
        {"empid":5, "empname": "Smith", "salary":9000}
    ];
    for(i=0; i < employees.length; i++)
    {
        var employee = employees[i];
        var temp = "<li>" + employee.empname + "</li>";
        $("#list1").append(temp);
    }
</script>
</body>
</html>

```

jQuery – JavaScript Objects – Example 3

```

<html>
    <head>
        <title>jQuery - Object Array - Table</title>
        <style type="text/css">
            body,table
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
        </style>
    </head>
    <body>
        <table id="table1" border="1">
            <tr style="background-color: gold;">
                <th>Emp ID</th>
                <th>Emp Name</th>
                <th>Salary</th>
            </tr>
        </table>

        <script src="jquery-3.2.1.js"></script>

        <script>
            var employees =
            [
                {"empid":1, "empname": "Scott", "salary":5000},
                {"empid":2, "empname": "Allen", "salary":6000},
                {"empid":3, "empname": "John", "salary":7000},
                {"empid":4, "empname": "Jones", "salary":8000},
                {"empid":5, "empname": "Smith", "salary":9000}
            ];
            for(i=0; i < employees.length; i++)

```

```

{
    var employee = employees[i];
    var temp = "<tr><td>" + employee.empid + "</td><td>" + employee.empname + "</td><td>" +
employee.salary + "</td></tr>";
    $("#table1").append(temp);
}

$("#table1").on("mouseover", "tr", fun1);
$("#table1").on("mouseout", "tr", fun2);
function fun1()
{
    $(this).css("background-color", "darkgray");
}
function fun2()
{
    $(this).css("background-color", "white");
}
</script>
</body>
</html>

```

jQuery – JavaScript Objects – Example 4

```

<html>
    <head>
        <title>jQuery - Object - User Data</title>
        <style type="text/css">
            body,input
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
        </style>
    </head>
    <body>
        <h3>Type Person Details</h3>
        <div id="div1">
            First Name: <input type="text" id="textbox1"><br>
            Last Name: <input type="text" id="textbox2"><br>
            Age: <input type="text" id="textbox3"><br>
        </div>
        <input type="button" id="button1" value="Add">

        <script src="jquery-3.2.1.js"></script>

        <script>
            var Persons = [];

            $("#button1").click(fun1);
            function fun1()
            {
                var p = { "FirstName": $("#textbox1").val(), "LastName": $("#textbox2").val(), "Age": $("#textbox3").val() };
                Persons.push(p);
                alert(JSON.stringify(Persons)); //convert json to string.
            }
        </script>
    </body>
</html>

```

```

        $("#div1 input[type=text]").val("");
        $("#textbox1").focus();
    };
</script>
</body>
</html>

```

jQuery – JavaScript Objects – Example 5

```

<html>
  <head>
    <title>jQuery - Complex Object Example</title>
    <style type="text/css">
      body,input,table
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>Complex json example</h1>
    <div id="div1"></div>

    <script src="jquery-3.2.1.js"></script>

    <script>
      var Departments =
      [
        {DeptNo: 10, DeptName: "Accounting", Loc: "New York",
         Employees:
         [
           {EmpID: 1, EmpName: "ab"}, {EmpID: 2, EmpName: "cd"}, {EmpID: 3, EmpName: "ef"}
         ]
       },
        {DeptNo: 20, DeptName: "Sales", Loc: "New Delhi",
         Employees:
         [
           {EmpID: 4, EmpName: "gh"}, {EmpID: 5, EmpName: "ij"}
         ]
       },
        {DeptNo: 30, DeptName: "R&D", Loc: "New Mumbai",
         Employees:
         [
           {EmpID: 6, EmpName: "kl"}, {EmpID: 7, EmpName: "mn"}, {EmpID: 8, EmpName: "op"}, {EmpID: 9, EmpName: "qr"}
         ]
       }
     ];
    </script>
  </body>
</html>

```

```
        }

    ];

    var temp = "";
    $(fun1);
    function fun1()
    {
        for (i=0; i<Departments.length; i++)
        {
            temp += "<p>" + Departments[i].DeptNo + ", " + Departments[i].DeptName + ", " + Departments[i].Loc
            + "</p>";
            temp += "<table border='1' cellpadding='4px'>";
            for (j=0; j<Departments[i].Employees.length; j++)
            {
                temp += "<tr>";
                temp += "<td>" + Departments[i].Employees[j].EmpID + "</td>";
                temp += "<td>" + Departments[i].Employees[j].EmpName + "</td>";
                temp += "</tr>";
            }
            temp += "</table>";
        }
        $("#div1").html(temp);
    }
    </script>
</body>
</html>
```

CDN

- CDN stands for “Content Delivery Network”.
- The big companies such as Google, Microsoft etc., maintain all versions of jQuery files on their servers, which you can access directly from the html page, without manually downloading the jquery file and placing it in the current folder.

Google CDN

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.js">
</script>
```

Microsoft CDN

```
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.2.1.js">
</script>
```

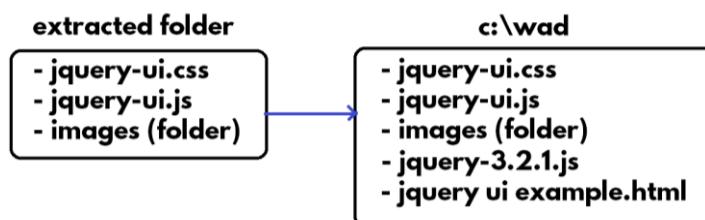
jQuery UI

Introduction to jQuery UI

- “jQuery UI” is a set of “plugins”, developed based on “jQuery”.
- “Plugin” is a “ready made component”.
 - Ex: date picker, auto complete, progress bar.

How to download jQuery UI

- Go to “<https://jqueryui.com>”.
- Click on “Download”.
- Select the version. Ex: 1.12.1
- Select the necessary components (or) select all the components (checkboxes).
- Select the theme. Ex: base
- Click on “Download” button.
- Download the file “jquery-ui-1.12.1.custom.zip”.
- Go to the downloaded folder.
- Right click on “jquery-ui-1.12.1.custom.zip” and click on “Extract All”.
- Go to the extracted folder “jquery-ui-1.12.1.custom”.
- Copy and paste the following files from “extracted folder” into “c:\wad” folder.



Import jQuery UI:

```

<link href="jquery-ui.css" rel="stylesheet">
<script src="jquery-3.2.1.js"></script>
<script src="jquery-ui.js"></script>
  
```

datepicker()

- This function is used to display a popup calendar for the textbox.
- **Syntax:** datepicker()
- **Example:** datepicker()

Example on datepicker()

```
<html>
<head>
<title>jQuery UI - DatePicker</title>
<style type="text/css">
body, input
{
    font-family: 'Tahoma';
    font-size: 25px;
}

#ui-datepicker-div
{
    font-size: 70%;
}
</style>
</head>
<body>
Date: <input type="text" id="txt1">

<script src="jquery-3.2.1.js"></script>
<script src="jquery-ui.js"></script>
<link href="jquery-ui.css" rel="stylesheet">

<script>
$("#txt1").datepicker();
</script>
</body>
</html>
```

spinner()

- This function is used to display “increment” and “decrement” buttons for a number in the textbox.
- Syntax: spinner()
- Example: spinner()

Example on spinner()

```
<html>
<head>
<title>jQuery UI - Spinner</title>
<style type="text/css">
body, input
{
    font-family: 'Tahoma';
    font-size: 20px;
}
```

```

</style>
</head>
<body>
    Amount:<input type="text" id="txt1">

    <script src="jquery-3.1.0.js"></script>
    <script src="jquery-ui.js"></script>
    <link href="jquery-ui.css" rel="stylesheet">

    <script>
        $("#txt1").spinner();
    </script>
</body>
</html>

```

autocomplete()

- This function is used to display suggestions while typing some text in the textbox.
- **Syntax:** autocomplete({ source: arrayname })
- **Example:** autocomplete({ source: myarray })

Example on autocomplete()

```

<html>
    <head>
        <title>jQuery UI - AutoComplete</title>
        <style type="text/css">
            body, input
            {
                font-family: 'Tahoma';
                font-size: 25px;
            }
        </style>
    </head>
    <body>
        <label for="txt1">Search:</label>
        <input type="text" id="txt1">

        <script src="jquery-3.1.0.js"></script>
        <script src="jquery-ui.js"></script>
        <link href="jquery-ui.css" rel="stylesheet">

        <script>
            var myarray = ["ActionScript", "AppleScript", "Asp", "BASIC", "C", "C++", "Clojure", "COBOL", "ColdFusion",
            "Erlang", "Fortran", "Groovy", "Haskell", "Java", "JavaScript", "Lisp", "Perl", "PHP", "Python", "Ruby", "Scala", "Scheme"];
            $("#txt1").autocomplete({ source: myarray });
        </script>
    </body>
</html>

```

tooltip()

- This function displays the tooltip attractively.
- Syntax: tooltip()
- Example: tooltip()

Example on tooltip()

```
<html>
  <head>
    <title>jQuery UI - Tooltip</title>
    <style type="text/css">
      #div1
      {
        background-color: #33ccff;
      }
      #p1
      {
        background-color: #00cc99;
      }
    </style>
  </head>
  <body>
    <div id="div1" title="this is div1">div1</div>
    <p id="p1" title="this is para 1">para 1</p>
    Name: <input type="text" id="txt1" title="Alphabets only">
    <script src="jquery-3.1.0.js"></script>
    <script src="jquery-ui.js"></script>
    <link href="jquery-ui.css" rel="stylesheet">
    <script>
      $(document).tooltip();
    </script>
  </body>
</html>
```

resizable()

- This function is used to make the <div> tag as resizable.
- Syntax: resizable()
- Example: resizable()

Example on resizable()

```
<html>
  <head>
    <title>jQuery UI - Resizable</title>
    <style type="text/css">
      body, input
    </style>
  </head>
  <body>
```

```

{
    font-family: 'Segoe UI';
    font-size: 25px;
}
#div1
{
    width: 200px;
    height: 200px;
    background-color: #99ffcc;
    cursor: pointer;
}
</style>
</head>
<body>
<div id="div1">div 1</div>

<script src="jquery-3.1.0.js"></script>
<script src="jquery-ui.js"></script>
<link href="jquery-ui.css" rel="stylesheet">
<script>
    $("#div1").resizable();
</script>
</body>
</html>

```

draggable()

- This function is used to make the <div> tag draggable across the page.
- Syntax: draggable()
- Example: draggable()

Example on draggable()

```

<html>
    <head>
        <title>jQuery UI - Draggable</title>
        <style type="text/css">
            #div1
            {
                width: 200px;
                height: 150px;
                background-color: #ccffff;
                cursor: pointer;
            }
        </style>
    </head>
    <body>
        <div id="div1">
            <p>Drag me around</p>
        </div>
        <script src="jquery-3.1.0.js"></script>
    
```

```

<script src="jquery-ui.js"></script>
<link href="jquery-ui.css" rel="stylesheet">

<script>
    $("#div1").draggable();
</script>
</body>
</html>

```

droppable()

- This function is used to make an element allow other draggable elements, to be dropped inside it.
- **Syntax:** droppable({ drop: functionname })
- **Example:** droppable({ drop: fun1 })
- When the user drops some element into the droppable element, it calls the “fun1” automatically.

Example on droppable()

```

<html>
    <head>
        <title>jQuery UI - Droppable</title>
        <style type="text/css">
            body, input
            {
                font-family: 'Tahoma';
                font-size: 25px;
            }
            #div1
            {
                width: 200px;
                height: 150px;
                background-color: #ccffff;
                cursor: pointer;
                float: left;
                margin-right: 10px;
            }
            #div2
            {
                width: 400px;
                height: 350px;
                background-color: #99ffcc;
                cursor: pointer;
                float: left;
            }
        </style>
    </head>
    <body>
        <div id="div1">div1</div>
        <div id="div2">div2</div>
    </body>

```

```

<script src="jquery-3.1.0.js"></script>
<script src="jquery-ui.js"></script>
<link href="jquery-ui.css" rel="stylesheet">

<script>
  $("#div1").draggable();
  $("#div2").droppable({drop: fun1});
  function fun1()
  {
    $(this).html("Dropped!").css("background-color", "#009999");
    //this = div2
  }
</script>
</body>
</html>

```

selectable()

- This function is used to make tags selectable.
- **Syntax:** selectable()
- **Example:** selectable()

Example on selectable()

```

<html>
  <head>
    <title>jQuery UI - Selectable</title>
    <style type="text/css">
      body, input
      {
        font-family: 'Segoe UI';
        font-size: 25px;
      }
      .class1 { background-color: #0099ff; }
      #list1 { list-style-type: none; padding: 0; }
      #list1 li { margin: 3px; }
      #list1 .ui-selecting { background-color: #0099cc; }
      #list1 .ui-selected { background-color: darkblue; color: white; }
    </style>
  </head>
  <body>
    <ol id="list1">
      <li class="class1">Item 1</li>
      <li class="class1">Item 2</li>
      <li class="class1">Item 3</li>
      <li class="class1">Item 4</li>
      <li class="class1">Item 5</li>
      <li class="class1">Item 6</li>
      <li class="class1">Item 7</li>
    </ol>
  </body>

```

```

<script src="jquery-3.1.0.js"></script>
<script src="jquery-ui.js"></script>
<link href="jquery-ui.css" rel="stylesheet">

<script>
    $("#list1").selectable();
</script>
</body>
</html>

```

sortable()

- This function is used to allow the user to sort (re-order) the tags of tag.
- **Syntax:** sortable()
- **Example:** sortable()

Example on sortable()

```

<html>
    <head>
        <title>jQuery UI - Sortable</title>
        <style type="text/css">
            body, input
            {
                font-family: 'Tahoma';
                font-size: 25px;
            }
            .class1 { background-color: #0099ff; }
            #list1 { list-style-type: none; padding: 0; }
            #list1 li { margin: 5px; }
        </style>
    </head>
    <body>
        <ul id="list1">
            <li class="class1">Item 1</li>
            <li class="class1">Item 2</li>
            <li class="class1">Item 3</li>
            <li class="class1">Item 4</li>
            <li class="class1">Item 5</li>
            <li class="class1">Item 6</li>
            <li class="class1">Item 7</li>
        </ul>

        <script src="jquery-3.1.0.js"></script>
        <script src="jquery-ui.js"></script>
        <link href="jquery-ui.css" rel="stylesheet">

        <script>
            $("#list1").sortable();
        </script>
    </body>

```

```
</html>
```

accordion()

- This function is used to allow the user to view any one content among few.
- Syntax:** accordion()
- Example:** accordion()

Example on accordion()

```
<html>
  <head>
    <title>jQuery UI - Accordion</title>
    <style type="text/css">
      body, input
      {
        font-family: 'Tahoma';
        font-size: 25px;
      }
    </style>
  </head>
  <body>
    <div id="div1">
      <h3>Section 1</h3>
      <div>Mauris mauris ante, blandit et, ultrices a, suscipit eget, quam. Integer ut neque. Vivamus nisi metus, molestie vel, gravida in, condimentum sit amet, nunc. Nam a nibh. Donec suscipit eros. Nam mi. Proin viverra leo ut odio. Curabitur malesuada. Vestibulum a velit eu ante scelerisque vulputate.</div>
      <h3>Section 2</h3>
      <div>Sed non urna. Donec et ante. Phasellus eu ligula. Vestibulum sit amet purus. Vivamus hendrerit, dolor at aliquet laoreet, mauris turpis porttitor velit, faucibus interdum tellus libero ac justo. Vivamus non quam. In suscipit faucibus urna.</div>
      <h3>Section 3</h3>
      <div>Nam enim risus, molestie et, porta ac, aliquam ac, risus. Quisque lobortis. Phasellus pellentesque purus in massa. Aenean in pede. Phasellus ac libero ac tellus pellentesque semper. Sed ac felis. Sed commodo, magna quis lacinia ornare, quam ante aliquam nisi, eu iaculis leo purus venenatis dui.</div>
      <h3>Section 4</h3>
      <div>Cras dictum. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean lacinia mauris vel est.</div>
    </div>

    <script src="jquery-3.1.0.js"></script>
    <script src="jquery-ui.js"></script>
    <link href="jquery-ui.css" rel="stylesheet">

    <script>
      $("#div1").accordion();
    </script>
  </body>
</html>
```

tabs()

- This function is used to display tabs in the web page.
- Syntax:** tabs()
- Example:** tabs()

Example on tabs()

```

<html>
  <head>
    <title>jQuery UI - Tabs</title>
    <style type="text/css">
      body, input
      {
        font-family: 'Tahoma';
        font-size: 20px;
      }
      .ui-tabs, .ui-tabs-anchor
      {
        font-size: 75%;
      }
    </style>
  </head>
  <body>
    <div id="div1">
      <ul>
        <li><a href="#tab1">Home</a></li>
        <li><a href="#tab2">About</a></li>
        <li><a href="#tab3">Contact</a></li>
      </ul>
      <div id="tab1">Proin elit arcu, rutrum commodo, vehicula tempus, commodo a, risus. Curabitur nec arcu. Donec sollicitudin mi sit amet mauris. Nam elementum quam ullamcorper ante. Etiam aliquet massa et lorem. Mauris dapibus lacus auctor risus. Aenean tempor ullamcorper leo. Vivamus sed magna quis ligula eleifend adipiscing. Duis orci. Aliquam sodales tortor vitae ipsum. Aliquam nulla. Duis aliquam molestie erat. Ut et mauris vel pede varius sollicitudin. Sed ut dolor nec orci tincidunt interdum. Phasellus ipsum. Nunc tristique tempus lectus.</div>
      <div id="tab2">Morbi tincidunt, dui sit amet facilisis feugiat, odio metus gravida ante, ut pharetra massa metus id nunc. Duis scelerisque molestie turpis. Sed fringilla, massa eget luctus malesuada, metus eros molestie lectus, ut tempus eros massa ut dolor. Aenean aliquet fringilla sem. Suspendisse sed ligula in ligula suscipit aliquam. Praesent in eros vestibulum mi adipiscing adipiscing. Morbi facilisis. Curabitur ornare consequat nunc. Aenean vel metus. Ut posuere viverra nulla. Aliquam erat volutpat. Pellentesque convallis. Maecenas feugiat, tellus pellentesque pretium posuere, felis lorem euismod felis, eu ornare leo nisi vel felis. Mauris consectetur tortor et purus.</div>
      <div id="tab3">Mauris eleifend est et turpis. Duis id erat. Suspendisse potenti. Aliquam vulputate, pede vel vehicula accumsan, mi neque rutrum erat, eu congue orci lorem eget lorem. Vestibulum non ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Fusce sodales. Quisque eu urna vel enim commodo pellentesque. Praesent eu risus hendrerit ligula tempus pretium. Curabitur lorem enim, pretium nec, feugiat nec, luctus a, lacus.</div>
    </div>
    <script src="jquery-3.1.0.js"></script>
    <script src="jquery-ui.js"></script>
    <link href="jquery-ui.css" rel="stylesheet">

```

```

<script>
  $("#div1").tabs();
</script>
</body>
</html>

```

dialog()

- This function is used to display modal dialog box (popup box) in the web page.
- Syntax:** dialog({ modal: true, width: pixels, height: pixels, buttons: { "button1": functionname, "button2": functionname } })

Example on dialog()

```

<html>
  <head>
    <title>jQuery UI - Dialog</title>
    <style type="text/css">
      body, input
      {
        font-family: 'Tahoma';
        font-size: 20px;
      }
      #span1
      {
        font-size: 30px;
      }
      .ui-dialog, .ui-dialog-title, .ui-button-text, .ui-dialog p
      {
        font-size: 70%;
      }
    </style>
  </head>
  <body>
    <input type="button" id="button1" value="Show dialog">
    <br>
    <span id="span1">Output here</span>
    <div id="div1" title="Title here" style="display:none">
      <p>Hello, how are you!</p>
    </div>

    <script src="jquery-2.2.0.js"></script>
    <script src="jquery-ui.js"></script>
    <link href="jquery-ui.css" rel="stylesheet">

    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $("#div1").dialog({ modal:true, width:300, height: 200, buttons: { "OK": fun2, "Cancel": fun3 } });
      }
    </script>
  </body>

```

```

function fun2()
{
    $("#spam1").html("OK clicked");
    $(this).dialog( "close" );
}
function fun3()
{
    $("#spam1").html("Cancel clicked");
    $(this).dialog( "close" );
}
</script>
</body>
</html>

```

buttonset()

- It is used to display “radio buttons” as “buttons”.
- Syntax: buttonset()

Example on buttonset()

```

<html>
    <head>
        <title>jQuery UI - ButtonSet</title>
        <style type="text/css">
            body, input
            {
                font-family: 'Tahoma';
                font-size: 25px;
            }
        </style>
    </head>
    <body>
        <div id="div1">
            <input type="radio" id="radio1" name="radio">
            <label for="radio1">Debit Card</label>
            <input type="radio" id="radio2" name="radio">
            <label for="radio2">Credit Card</label>
            <input type="radio" id="radio3" name="radio">
            <label for="radio3">Net Banking</label>
        </div>

        <script src="jquery-3.1.0.js"></script>
        <script src="jquery-ui.js"></script>
        <link href="jquery-ui.css" rel="stylesheet">

        <script>
            $("#div1").buttonset();
        </script>
    </body>
</html>

```

menu()

- This function is used to display a menu (along with sub menus) in the web page.
- **Syntax:** menu()

Example on menu()

```
<html>
  <head>
    <title>jQuery UI - Menu</title>
    <style type="text/css">
      body, input
      {
        font-family: 'Tahoma';
        font-size: 20px;
      }
      .ui-menu
      {
        width: 300px;
      }
    </style>
  </head>
  <body>
    <ul id="list1">
      <li>Home</li>
      <li>About</li>
      <li>Contact</li>
      <li>Download</li>
      <li>Express Download</li>
      <li>Product Info
        <ul>
          <li>Order</li>
          <li>Features</li>
          <li>Installation</li>
          <li>Supported Browsers
            <ul>
              <li>Google Chrome</li>
              <li>Mozilla Firefox</li>
              <li>Internet Explorer</li>
              <li>Safari</li>
              <li>Opera</li>
            </ul>
          </li>
        </ul>
      </li>
      <li>Purchase</li>
      <li>Demo / Installation</li>
    </ul>
    <script src="jquery-3.1.0.js"></script>
```

```

<script src="jquery-ui.js"></script>
<link href="jquery-ui.css" rel="stylesheet">

<script>
  $("#list1").menu();
</script>

</body>
</html>

```

progressbar()

- This function is used to display progress bar within the web page.
- **Syntax:** progressbar({ value: n })
- **Example:** progressbar({ value: 60 })

Example on progressbar()

```

<html>
  <head>
    <title>jQuery UI - Progressbar</title>
    <style type="text/css">
      body, input
      {
        font-family: 'Tahoma';
        font-size: 20px;
      }
      #div1
      {
        height: 20px;
      }
    </style>
  </head>
  <body>
    <div id="div1"></div>

    <script src="jquery-3.1.0.js"></script>
    <script src="jquery-ui.js"></script>
    <link href="jquery-ui.css" rel="stylesheet">

    <script>
      $("#div1").progressbar({ value: 60 });
    </script>
  </body>
</html>

```

slider()

- This function is used to display a slider in the web page. Ex: price range

- **Syntax:** slider({ range: true, min: minimum, max: maximum, values: [value1, value2], slide: functionname });

Example on slider()

```
<html>
  <head>
    <title>jQuery UI - Slider</title>
  </head>
  <body>
    <p>
      Price range:
      <span id="span1" style="color:#f6931f;font-weight:bold;"></span>
    </p>
    <div id="div1"></div>

    <script src="jquery-3.1.0.js"></script>
    <script src="jquery-ui.js"></script>
    <link href="jquery-ui.css" rel="stylesheet">

    <script>
      $("#div1").slider( { range: true, min: 0, max: 500, values: [ 200, 300 ], slide: fun1 });
      function fun1(event, ui)
      {
        $("#span1").html( "Rs. " + ui.values[0] + " - Rs. " + ui.values[1] );
      }
      $("#span1").html( "Rs. " + $("#div1").slider( "values", 0 ) + " - Rs. " + $("#div1").slider( "values", 1 ) );
    </script>
  </body>
</html>
```

Color Animation

- “jQuery UI” supports “color animation”, which can animate (change) the color based properties such as “background-color”, “color”, “border-color” etc.
- **Syntax:** animate({ “property”:”value” }, milli seconds);

Example on Color Animation

```
<html>
  <head>
    <title>jQuery UI - Color Animation</title>
    <style type="text/css">
      body, input
      {
        font-family: 'Tahoma';
        font-size: 20px;
      }
      #div1
```

```

    {
        background-color: darkred;
        color: cyan;
        font-size: 30px;
        font-family: 'Tahoma';
    }
</style>
</head>
<body>
<div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum is
simply dummy text of the printing and typesetting industry.</div>
<input type="button" id="button1" value="color animation">

<script src="jquery-3.1.0.js"></script>
<script src="jquery-ui.js"></script>
<link href="jquery-ui.css" rel="stylesheet">

<script>
    $("#button1").click(fun1);
    function fun1()
    {
        $("#div1").animate({ "background-color": "green", "color": "yellow" }, 2000);
    }
</script>
</body>
</html>

```

Easing Effects

- “Easing Effects” are used to make the animation more attractively.
- Syntax:** `animate({ “property”:”value” }, milli seconds, “easing effect”);`
- List of easing effects:** linear, swing, easeInQuad, easeOutQuad, easeInOutQuad, easeInCubic, easeOutCubic, easeInOutCubic, easeInQuart, easeOutQuart, easeInOutQuart, easeInQuint, easeOutQuint, easeInOutQuint, easeInExpo, easeOutExpo, easeInOutExpo, easeInSine, easeOutSine, easeInOutSine, easeInCirc, easeOutCirc, easeInOutCirc, easeInElastic, easeOutElastic, easeInOutElastic, easeInBack, easeOutBack, easeInOutBack, easeInBounce, easeOutBounce, easeInOutBounce,

Example on Easing Effects

```

<html>
    <head>
        <title>jQuery UI - Easing Effects</title>
        <style type="text/css">
            body, input
            {
                font-family: 'Tahoma';
                font-size: 20px;
            }

```

```

#div1
{
    background-color: darkred;
    color: cyan;
    font-size: 30px;
    position: relative;
    width: 300px;
    padding: 5px;
    border: 2px solid green;
}
</style>
</head>
<body>
<input type="button" id="button1" value="easing effect animation">
<div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum is
simply dummy text of the printing and typesetting industry.</div>

<script src="jquery-3.1.0.js"></script>
<script src="jquery-ui.js"></script>
<link href="jquery-ui.css" rel="stylesheet">

<script>
    $("#button1").click(fun1);
    function fun1()
    {
        $("#div1").animate({ "font-size": "40px", "border-width": "10px", "left": "30px", "width": "700px",
"padding": "15px" }, 2000, "easeOutQuart");
    }
</script>
</body>
</html>

```

addClass() with animation

- addClass() function supports time duration (milli seconds), in jQuery UI.
- **Syntax:** addClass(“class name”, milli seconds);
- **Syntax:** addClass(“class1”, 1000);

Example on addClass() with animation

```

<html>
    <head>
        <title>jQuery UI - addClass</title>
        <style type="text/css">
            body, input
            {
                font-family: 'Tahoma';
                font-size: 20px;
            }
            .class1
            {

```

```

background-color: darkred;
color: cyan;
font-size: 30px;
font-family: 'Tahoma';
width: 800px;
padding: 15px;
}
</style>
</head>
<body>
<div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum is
simply dummy text of the printing and typesetting industry.</div>
<input type="button" id="button1" value="add class">

<script src="jquery-3.1.0.js"></script>
<script src="jquery-ui.js"></script>
<link href="jquery-ui.css" rel="stylesheet">

<script>
$("#button1").click(fun1);
function fun1()
{
    $("#div1").addClass("class1", 1000);
}
</script>
</body>
</html>

```

removeClass() with animation

- removeClass() function supports time duration (milli seconds), in jQuery UI.
- **Syntax:** removeClass(“class name”, milli seconds);
- **Syntax:** removeClass(“class1”, 1000);

Example on removeClass() with animation

```

<html>
<head>
<title>jQuery UI - removeClass</title>
<style type="text/css">
body, input
{
    font-family: 'Tahoma';
    font-size: 20px;
}
.class1
{
    background-color: darkred;
    color: cyan;
    font-size: 30px;
    font-family: 'Tahoma';
}

```

```

        width: 800px;
        padding: 15px;
    }

```

```

</style>
</head>
<body>
    <div id="div1" class="class1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem
    Ipsum is simply dummy text of the printing and typesetting industry.</div>
    <input type="button" id="button1" value="remove class">
    <script src="jquery-3.1.0.js"></script>
    <script src="jquery-ui.js"></script>
    <link href="jquery-ui.css" rel="stylesheet">

    <script>
        $("#button1").click(fun1);
        function fun1()
        {
            $("#div1").removeClass("class1", 1000);
        }
    </script>
</body>
</html>

```

Effects

- jQuery UI supports the following advanced effects to hide / show the content attractively.
- List of effects:** blind | bounce | clip | drop | explode | fold | highlight | puff | pulsate | scale | shake | size | slide
- Show:** show("effect", { options }, milli seconds, callback);
- Hide:** show("effect", { options }, milli seconds, callback);
- Note:** The callback function will be called automatically, after completion of animation. It is optional.

Example on Effects

```

<html>
    <head>
        <title>jQuery UI - Effects</title>
        <style type="text/css">
            body, input
            {
                font-family: 'Tahoma';
                font-size: 20px;
            }
            #div1
            {
                background-color: darkred;
                color: cyan;
                font-size: 30px;
            }
        </style>
    </head>
    <body>
        <div id="div1">Hello World</div>
        <input type="button" value="Show Effect" onclick="showEffect()">
    </body>
</html>

```

```
width: 800px;
padding: 15px;
}
</style>
</head>
<body>
<input type="button" id="button1" value="Hide">
<input type="button" id="button2" value="Show">
<div id="div1">Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum is
simply dummy text of the printing and typesetting industry.</div>

<script src="jquery-3.1.0.js"></script>
<script src="jquery-ui.js"></script>
<link href="jquery-ui.css" rel="stylesheet">

<script>
$("#button1").click(fun1);
function fun1()
{
    $("#div1").hide("bounce", {}, 1000, callback);
}
function callback()
{
    alert("Done");
}
$("#button2").click(fun2);
function fun2()
{
    $("#div1").show("bounce", {}, 1000, callback);
}
</script>
</body>
</html>
```

jQuery Validations

- “jQuery Validation Plugin” is used to create validations in jQuery.
- Download “jquery-3.2.1.js” file from “<https://jquery.com>” and place it in “c:\jquery” folder.
- Download the jquery validation plugin file and place it in “c:\jquery” folder:
<http://ajax.aspnetcdn.com/ajax/jquery.validate/1.15.0/jquery.validate.js>
- Create “validation.html” file in “c:\jquery” folder.

List of files

c:\jquery

- jquery-3.2.1.js
- jquery.validate.js
- validation.html

Code for “Validation.html”

```

<html>
  <head>
    <title>jQuery Validation Plugin</title>
  </head>
  <body>
    <h1>Registration Form</h1>
    <form action="http://localhost/someaddress" id="form1">

      <table>
        <tr>
          <td><label for="t1">Username:</label></td>
          <td><input type="text" id="t1" name="txt1"></td>
        </tr>

        <tr>
          <td><label for="t2">Password:</label></td>
          <td><input type="password" id="t2" name="txt2"></td>
        </tr>

        <tr>
          <td><label for="t3">E-Mail:</label></td>
          <td><input type="text" id="t3" name="txt3"></td>
        </tr>

        <tr>
          <td><label for="t4">Re-type E-Mail:</label></td>
          <td><input type="text" id="t4" name="txt4"></td>
        </tr>

      <tr>
    
```

```

<td><label for="t5">Phone:</label></td>
<td><input type="text" id="t5" name="txt5"></td>
</tr>

<tr>
    <td><label for="t6">Age:</label></td>
    <td><input type="text" id="56" name="txt6" maxlength="2"></td>
</tr>

<tr>
    <td colspan="2"><input type="submit" value="Register"></td>
</tr>

</table>
</form>

<script src="jquery-3.2.1.js"></script>
<script src="jquery.validate.js"></script>

<script>
    $("#form1").validate({
        rules: {
            txt1: { required: true },
            txt2: { required: true, minlength: 6 },
            txt3: { required: true, regexp: /^[a-zA-Z0-9._-]+@[a-zA-Z0-9-]+\.[a-zA-Z.]{2,5}$/i },
            txt4: { required: true, equalTo: '#t3' },
            txt5: { required: true },
            txt6: { required: true, range: [18,70] }
        },
        messages: {
            txt1: { required: "Username can't be blank" },
            txt2: { required: "Password can't be blank", minlength: "Minimum length is 6" },
            txt3: { required: "Email can't be blank", regexp: "Enter a valid E-Mail" },
            txt4: { required: "Re-type email can't be blank", equalTo: "Email and re-type email must be same" },
            txt5: { required: "Phone can't be blank" },
            txt6: { required: "Age can't be blank", range: "Age must be in between 18 and 70" }
        }
    });

    //for regular expressions
    $.validator.addMethod("regexp", function(value, element, param) {
        return this.optional(element) || param.test(value); // Compare with regular expression
    });

</script>

<style type="text/css">
    body, input, select, textarea, table { font-family: 'Tahoma'; font-size: 28px; }
    label.error { color: red; } /*style for validation error messages */

```

```
input.error { border: 1px solid #foo; background-color: #fee; }/* style for validation error text boxes */  
</style>  
  
</body>  
</html>
```

jQuery AJAX

jQuery AJAX

- “jQuery” provides a method called “\$.ajax()”, to send ajax request to server and get ajax response from server, more easily.

Syntax of \$.ajax()

```
$.ajax( { type: "GET | POST | PUT | DELETE", url: "address", success: callbackfunction, error: callbackfunction } );
```

url: Represents the server address, to which you want to send request.

type: Represents type of the request: GET | POST | PUT | DELETE

success: Represents the callback function, which will be called automatically after successfully receiving the response from server.

error: Represents the callback function, which will be called automatically if the server returns an error (exception) as response to the browser.

jQuery AJAX – NodeJS – Simple - Example

Creating the application

- Create “c:\ajax” folder.
- Place “jquery-3.2.1.js” file in “c:\ajax” folder.
- Create “httpserver.js” and “index.html” files in the “c:\ajax” folder.

c:\ajax

- httpserver.js
- index.html

- jquery-3.2.1.js

c:\ajax\httpserver.js

```
var http = require("http");
var fs = require("fs");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
    console.log("Server started at port 8080");
}

function engine(request, response)
{
    if (request.url == "/" || request.url == "/index.html")
    {
        fs.readFile("index.html", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url == "/jquery-3.2.1.js")
    {
        fs.readFile("jquery-3.2.1.js", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/javascript");
                response.writeHead(200);
            }
        }
    }
}
```

```
        response.write(data);
        response.end();
    }
}
else if(request.url.startsWith("/getdata"))
{
    console.log("request received at " + new Date().toLocaleTimeString());
    response.setHeader("content-type", "text/html");
    response.writeHead(200);
    response.write("Response from server at " + new Date().toLocaleTimeString());
    response.end();
}
}
```

c:\ajax\index.html

```
<html>
  <head>
    <title>jQuery AJAX - Simple</title>
  </head>
  <body>
    <h1>jQuery AJAX - Simple</h1>
    <input type="button" id="button1" value="GET data from server">
    <div id="div1"></div>

    <script src="jquery-3.2.1.js"></script>
    <script>
      $("#button1").click(fun1);
      function fun1()
      {
        $.ajax({ url:"GET", url:"/getdata", success: fun2 });
      }
      function fun2(response)
      {
        $("#div1").html(response);
      }
    </script>
  </body>
</html>
```

Execution:

- Download and install “nodejs” from “<https://nodejs.org>”.
- Open “Command Prompt” and run the following commands:
cd c:\ajax
node httpserver.js
- Open browser and enter the url: <http://localhost:8080/index.html>

jQuery AJAX – NodeJS – Get - Example

Creating the application

- Create “c:\ajax” folder.
- Place “jquery-3.2.1.js” file in “c:\ajax” folder.
- Create “httpserver.js” and “index.html” files in the “c:\ajax” folder.

c:\ajax

- httpserver.js
- index.html
- jquery-3.2.1.js

c:\ajax\httpserver.js

```
var http = require("http");
var fs = require("fs");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
    console.log("Server started at port 8080");
}

function engine(request, response)
{
    if (request.url == "/" || request.url == "/index.html")
    {
        fs.readFile("index.html", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url == "/jquery-3.2.1.js")
```

```

{
  fs.readFile("jquery-3.2.1.js", "utf8", fun1);
  function fun1(error, data)
  {
    if(error)
    {
      response.setHeader("content-type", "text/html");
      response.writeHead(404);
      response.write("<h1 style='color:red'>Page not found</h1>");
      response.end();
    }
    else
    {
      response.setHeader("content-type", "text/javascript");
      response.writeHead(200);
      response.write(data);
      response.end();
    }
  }
}
else if(request.url.startsWith("/getemployees"))
{
  console.log("request received at " + new Date().toLocaleTimeString());
  response.setHeader("content-type", "application/json");
  response.writeHead(200);
  response.write('['
    {"empid":1, "empname": "Scott", "salary": 4000},
    {"empid": 2, "empname": "Allen", "salary": 7500},
    {"empid": 3, "empname": "Jones", "salary": 9200},
    {"empid": 4, "empname": "James", "salary": 8400},
    {"empid": 5, "empname": "Smith", "salary": 5600}
  ']);
  response.end();
}
}

```

c:\ajax\index.html

```

<!DOCTYPE html>
<html>
<head>
  <title>NodeJS - jQuery AJAX - GET</title>
</head>
<body>
  <h1>NodeJS - jQuery AJAX - GET</h1>
  <form>
    <input type="button" id="btn1" value="Get Data">
    <table id="table1" border="1">
      <tr><th>Emp ID</th><th>Emp Name</th><th>Salary</th></tr>
    </table>
  </form>
  <script src="/jquery-3.2.1.js"></script>

```

```

<script>
  $("#btn1").click(fun1);
  function fun1(event)
  {
    event.preventDefault();
    $.ajax({ type: "GET", url: "/getemployees", success: fun2, error: fun3 });
  }

  function fun2(response)
  {
    $("#table1 tr:gt(0)").remove();
    for (var i=0; i < response.length; i++)
    {
      $("#table1").append("<tr> <td>" + response[i].empid + "</td> <td>" + response[i].empname + "</td> <td>" +
response[i].salary + "</td> </tr>");
    }
  }

  function fun3(error)
  {
    alert(error);
  }
</script>
</body>
</html>

```

Execution:

- Download and install “nodejs” from “<https://nodejs.org>”.
- Open “Command Prompt” and run the following commands:
 cd c:\ajax
 node httpserver.js
- Open browser and enter the url: <http://localhost:8080/index.html>

jQuery AJAX – NodeJS – Search - Example

Creating the application

- Create “c:\ajax” folder.
- Place “jquery-3.2.1.js” file in “c:\ajax” folder.
- Create “httpserver.js” and “index.html” files in the “c:\ajax” folder.

c:\ajax

- httpserver.js
- index.html

- jquery-3.2.1.js

c:\ajax\httpserver.js

```
var http = require("http");
var fs = require("fs");
var querystring = require("querystring");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
    console.log("Server started at port 8080");
}

function engine(request, response)
{
    if(request.url == "/" || request.url == "/index.html")
    {
        fs.readFile("index.html", "utf8", fun1);
        function fun1(error, data)
        {
            if(error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url == "/jquery-3.2.1.js")
    {
        fs.readFile("jquery-3.2.1.js", "utf8", fun1);
        function fun1(error, data)
        {
            if(error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/javascript");
            }
        }
    }
}
```

```

        response.writeHead(200);
        response.write(data);
        response.end();
    }
}
else if (request.url.startsWith("/searchemployees"))
{
    console.log("request received at " + new Date().toLocaleTimeString());
    response.setHeader("content-type", "application/json");
    response.writeHead(200);
    var employees = [
        { "empid": 1, "empname": "Scott", "salary": 4000 },
        { "empid": 2, "empname": "Allen", "salary": 7500 },
        { "empid": 3, "empname": "Jones", "salary": 9200 },
        { "empid": 4, "empname": "James", "salary": 8400 },
        { "empid": 5, "empname": "Smith", "salary": 5600 }
    ];
    var q = querystring.parse((request.url.split('?')[1]));
    var employees2 = [];
    for (var i = 0; i < employees.length; i++)
    {
        if (employees[i].empname.indexOf(q.searchstr) >= 0)
        {
            employees2.push(employees[i]);
        }
    }
    response.write(JSON.stringify(employees2));
    response.end();
}
}

```

c:\ajax\index.html

```

<!DOCTYPE html>
<html>
<head>
    <title>NodeJS - jQuery AJAX - Search</title>
</head>
<body>
    <h1>NodeJS - jQuery AJAX - Search</h1>
    <form>
        Search employees: <input type="text" id="txt1">
        <input type="submit" id="button1" value="Search">
        <table id="table1" border="1">
            <tr><th>Emp ID</th><th>Emp Name</th><th>Salary</th></tr>
            </table>
    </form>

    <script src="/jquery-3.2.1.js"></script>

    <script>
        $("#button1").click(fun1);
    </script>

```

```
function fun1(event)
{
    event.preventDefault();
    $.ajax({ type: "GET", url: "/searchemployees?searchstr=" + $("#txt1").val(), success: fun2, error: fun3 });
}

function fun2(response)
{
    $("#table1 tr:gt(0)").remove();
    for (var i = 0; i < response.length; i++)
    {
        $("#table1").append("<tr><td>" + response[i].empid + "</td><td>" + response[i].empname + "</td><td>" +
response[i].salary + "</td></tr>");
    }
}

function fun3(error)
{
    alert(error);
}
</script>
</body>
</html>
```

Execution:

- Download and install “nodejs” from “<https://nodejs.org>”.
- Open “Command Prompt” and run the following commands:
cd c:\ajax
node httpserver.js
- Open browser and enter the url: <http://localhost:8080/index.html>

jQuery AJAX – NodeJS – Post - Example

Creating the application

- Create “c:\ajax” folder.
- Place “jquery-3.2.1.js” file in “c:\ajax” folder.
- Create “httpserver.js” and “index.html” files in the “c:\ajax” folder.

c:\ajax

- httpserver.js
- index.html
- jquery-3.2.1.js

c:\ajax\httpserver.js

```

var http = require("http");
var fs = require("fs");
var querystring = require("querystring");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
    console.log("Server started at port 8080");
}

function engine(request, response)
{
    if (request.url == "/" || request.url == "/index.html")
    {
        fs.readFile("index.html", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url == "/jquery-3.2.1.js")
    {
        fs.readFile("jquery-3.2.1.js", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/javascript");
                response.writeHead(200);
                response.write(data);
            }
        }
    }
}

```

```

        response.end();
    }
}
else if(request.url.startsWith("/insertemployee"))
{
    console.log("request received at " + new Date().toLocaleTimeString());
    response.setHeader("content-type", "text/html");
    response.write("Successfully Inserted");
    response.end();
}
}

```

c:\ajax\index.html

```

<!DOCTYPE html>
<html>
<head>
<title>NodeJS - jQuery AJAX - Post</title>
<style>
body, input
{
    font-family: Tahoma;
    font-size: 24px;
}
</style>
</head>
<body>
<h1>NodeJS - jQuery AJAX - Post</h1>
<form>
    Emp ID: <input type="text" id="txt1"><br>
    Emp Name: <input type="text" id="txt2"><br>
    Salary: <input type="text" id="txt3"><br>
    <input type="submit" id="btn1" value="Insert">
    <div id="div1"></div>
</form>

<script src="jquery-3.2.1.js"></script>

<script>
$("#btn1").click(fun1);
function fun1(event)
{
    event.preventDefault();
    var mydata = { "empid": $("#txt1").val(), "empname": $("#txt2").val(), "salary": $("#txt3").val() };
    $.ajax({ type: "POST", url: "/insertemployee", success: fun2, data: mydata });
}

function fun2(response)
{
    $("#div1").html(response);
}
</script>
</body>

```

</html>

Execution:

- Download and install “nodejs” from “<https://nodejs.org>”.
- Open “Command Prompt” and run the following commands:
cd c:\ajax
node httpserver.js
- Open browser and enter the url: <http://localhost:8080/index.html>

jQuery AJAX – NodeJS – Put - Example

Creating the application

- Create “c:\ajax” folder.
- Place “jquery-3.2.1.js” file in “c:\ajax” folder.
- Create “httpserver.js” and “index.html” files in the “c:\ajax” folder.

c:\ajax

- httpserver.js
- index.html
- jquery-3.2.1.js

c:\ajax\httpserver.js

```
var http = require("http");
var fs = require("fs");
var querystring = require("querystring");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
    console.log("Server started at port 8080");
}

function engine(request, response)
{
    if (request.url == "/" || request.url == "/index.html")
    {
        fs.readFile("index.html", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
```

```

    {
        response.setHeader("content-type", "text/html");
        response.writeHead(404);
        response.write("<h1 style='color:red'>Page not found</h1>");
        response.end();
    }
    else
    {
        response.setHeader("content-type", "text/html");
        response.writeHead(200);
        response.write(data);
        response.end();
    }
}
else if (request.url == "/jquery-3.2.1.js")
{
    fs.readFile("jquery-3.2.1.js", "utf8", fun1);
    function fun1(error, data)
    {
        if (error)
        {
            response.setHeader("content-type", "text/html");
            response.writeHead(404);
            response.write("<h1 style='color:red'>Page not found</h1>");
            response.end();
        }
        else
        {
            response.setHeader("content-type", "text/javascript");
            response.writeHead(200);
            response.write(data);
            response.end();
        }
    }
}
else if (request.url.startsWith("/updateemployee"))
{
    console.log("request received at " + new Date().toLocaleTimeString());
    response.setHeader("content-type", "text/html");
    response.write("Successfully Updated");
    response.end();
}
}

```

c:\ajax\index.html

```

<!DOCTYPE html>
<html>
<head>
<title>NodeJS - jQuery AJAX - Put</title>
<style>
body, input

```

```

{
  font-family: Tahoma;
  font-size: 24px;
}
</style>
</head>
<body>
<h1>NodeJS - jQuery AJAX - Put</h1>
<form>
  Existing Emp ID: <input type="text" id="txt1"><br>
  Emp Name: <input type="text" id="txt2"><br>
  Salary: <input type="text" id="txt3"><br>
  <input type="submit" id="btn1" value="Update">
  <div id="div1"></div>
</form>

<script src="jquery-3.2.1.js"></script>

<script>
  $("#btn1").click(fun1);
  function fun1(event)
  {
    event.preventDefault();
    var mydata = { "empid": $("#txt1").val(), "empname": $("#txt2").val(), "salary": $("#txt3").val() };
    $.ajax({ type: "PUT", url: "/updateemployee", data: mydata, success: fun2 });
  }

  function fun2(response)
  {
    $("#div1").html(response);
  }
</script>
</body>
</html>

```

Execution:

- Download and install “nodejs” from “<https://nodejs.org>”.
- Open “Command Prompt” and run the following commands:
 cd c:\ajax
 node httpserver.js
- Open browser and enter the url: <http://localhost:8080/index.html>

jQuery AJAX – NodeJS – Delete - Example**Creating the application**

- Create “c:\ajax” folder.

- Place “jquery-3.2.1.js” file in “c:\ajax” folder.
- Create “httpserver.js” and “index.html” files in the “c:\ajax” folder.

c:\ajax

- httpserver.js
- index.html
- jquery-3.2.1.js

c:\ajax\httpserver.js

```
var http = require("http");
var fs = require("fs");
var querystring = require("querystring");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
  console.log("Server started at port 8080");
}

function engine(request, response)
{
  if(request.url == "/" || request.url == "/index.html")
  {
    fs.readFile("index.html", "utf8", fun1);
    function fun1(error, data)
    {
      if(error)
      {
        response.setHeader("content-type", "text/html");
        response.writeHead(404);
        response.write("<h1 style='color:red'>Page not found</h1>");
        response.end();
      }
      else
      {
        response.setHeader("content-type", "text/html");
        response.writeHead(200);
        response.write(data);
        response.end();
      }
    }
  }
  else if(request.url == "/jquery-3.2.1.js")
  {
    fs.readFile("jquery-3.2.1.js", "utf8", fun1);
    function fun1(error, data)
    {
      if(error)
```

```

    {
        response.setHeader("content-type", "text/html");
        response.writeHead(404);
        response.write("<h1 style='color:red'>Page not found</h1>");
        response.end();
    }
    else
    {
        response.setHeader("content-type", "text/javascript");
        response.writeHead(200);
        response.write(data);
        response.end();
    }
}
else if(request.url.startsWith("/deleteemployee"))
{
    console.log("request received at " + new Date().toLocaleTimeString());
    response.setHeader("content-type", "text/html");
    response.write("Successfully Deleted");
    response.end();
}
}

```

c:\ajax\index.html

```

<!DOCTYPE html>
<html>
<head>
<title>NodeJS - jQuery AJAX - Delete</title>
<style>
body, input
{
    font-family: Tahoma;
    font-size: 24px;
}
</style>
</head>
<body>
<h1>NodeJS - jQuery AJAX -Delete</h1>
<form>
Existing Emp ID: <input type="text" id="txt1"><br>
<input type="submit" id="btn1" value="Delete">
<div id="div1"></div>
</form>

<script src="jquery-3.2.1.js"></script>

<script>
$("#btn1").click(fun1);
function fun1(event)
{
    event.preventDefault();
}

```

```
$ajax({ type: "DELETE", url: "/deleteemployee?empid=" + $("#txt1").val(), success: fun2 });

}

function fun2(response)
{
    $("#div1").html(response);
}
</script>
</body>
</html>
```

Execution:

- Download and install “nodejs” from “<https://nodejs.org>”.
- Open “Command Prompt” and run the following commands:
cd c:\ajax
node httpserver.js
- Open browser and enter the url: <http://localhost:8080/index.html>

jQuery AJAX - .NET - Simple - Example

Creating the application

- Create “c:\ajax” folder.
- Open Visual Studio 2017.
- Click on “File” menu – “New” – “Project”.
- Select “.NET Framework 4.6”. Select “Visual C#”.
- Select “ASP.NET Web Application (.NET Framework)”.
- Name: AjaxSimple
- Location: c:\ajax
- Solution name: AjaxSimple
- Click on OK.
- Click on “Empty”. Check the check boxes “MVC” and “Web API”.
- Click on OK.
- Open Solution Explorer.
- Place “jquery-3.2.1.js” file in the project.
- Right click on the “Controllers” folder and click on “Add” – “Controller”. Select “MVC 5 Controller – Empty”. Click on “Add”. Enter the controller name as “HomeController”. Click on “Add”.

- Right click on “Views\Home” folder and click on “Add” – “View”. Enter the view name as “Index”. Select the template as “Empty (without model)”. Uncheck the checkbox “Use a layout page”. Click on “Add”.

AjaxSimple

- jquery-3.2.1.js
- Controllers
 - HomeController.cs
- Views
 - Home
 - Index.cshtml

Controllers\HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace AjaxSimple.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult GetData()
        {
            return Content("Hello from Server at " + DateTime.Now);
        }
    }
}
```

Views\Home\Index.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery AJAX - Simple</title>
    <style type="text/css">
        body, input
        {
            font-family: Tahoma;
            font-size: 24px;
        }
    </style>
```

```

</head>
<body>
<h1>jQuery AJAX - Simple</h1>
<input type="button" id="btn1" value="Get data from server">
<div id="div1"></div>

<script src="~/jquery-3.2.1.js"></script>

<script>
$("#btn1").click(fun1);
function fun1(event)
{
    event.preventDefault();
    $.ajax({ type: "GET", url: "/home/getdata", success: fun2 });
}
function fun2(response)
{
    $("#div1").html(response);
}
</script>
</body>
</html>

```

Execution:

- Go to “Debug” menu – “Start Debugging” in Visual Studio.

jQuery AJAX – .NET – Get - Example

Creating the application

- Create “c:\ajax” folder.
- Open Visual Studio 2017.
- Click on “File” menu – “New” – “Project”.
- Select “.NET Framework 4.6”.
- Select “Visual C#”.
- Select “ASP.NET Web Application (.NET Framework)”.
- Name: AjaxGet
- Location: c:\ajax
- Solution name: AjaxGet
- Click on OK.
- Click on “Empty”.
- Check the check boxes “MVC” and “Web API”.
- Click on OK.

- Open Solution Explorer.
- Place “jquery-3.2.1.js” file in the project.
- Right click on “Models” and click on “Add” – “New Item”. Click on “Visual C#” – “Class”. Enter the name as “Employee.cs”. Click on “Add”.
- Right click on the “Controllers” folder and click on “Add” – “Controller”. Select “MVC 5 Controller – Empty”. Click on “Add”. Enter the controller name as “HomeController”. Click on “Add”.
- Right click on “Views\Home” folder and click on “Add” – “View”. Enter the view name as “Index”. Select the template as “Empty (without model)”. Uncheck the checkbox “Use a layout page”. Click on “Add”.

AjaxGet

- jquery-3.2.1.js
 - Models
 - Employee.cs
 - Controllers
 - HomeController.cs
 - Views
 - Home
 - Index.cshtml

Models\Employee.cs

```
using System;

namespace AjaxGet.Models
{
    public class Employee
    {
        public int empid { get; set; }
        public string empname { get; set; }
        public double salary { get; set; }
    }
}
```

Controllers\HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using AjaxGet.Models;
```

```

namespace AjaxGet.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
        public ActionResult GetData()
        {
            List<Employee> emps = new List<Employee>()
            {
                new Employee() {empid = 1, empname = "Scott", salary = 4000},
                new Employee() {empid = 2, empname = "Allen", salary = 7500},
                new Employee() {empid = 3, empname = "Jones", salary = 9200},
                new Employee() {empid = 4, empname = "James", salary = 8400},
                new Employee() {empid = 5, empname = "Smith", salary = 5600}
            };
            return Json(emps, JsonRequestBehavior.AllowGet);
        }
    }
}

```

Views\Home\Index.cshtml

```

<!DOCTYPE html>
<html>
<head>
    <title>jQuery AJAX - Get</title>
    <style>
        body, input, table
        {
            font-family: Tahoma;
            font-size: 24px;
        }
    </style>
</head>
<body>
    <h1>jQuery AJAX - Get</h1>
    <form>
        <input type="submit" id="button1" value="Get Data from Database">
        <table id="table1" border="1">
            <tr><th>Emp ID</th><th>Emp Name</th><th>Salary</th></tr>
        </table>
    </form>

    <script src="~/jquery-3.2.1.js"></script>

    <script>
        $("#button1").click(fun1);
        function fun1(event)
        {
            event.preventDefault();

```

```
$ajax({ type: "GET", url: "/home/getdata", success: fun2 });

}

function fun2(response)
{
    $("#table1 tr:gt(0)").remove();
    for (var i=0; i < response.length; i++)
    {
        $("#table1").append("<tr><td>" + response[i].empid + "</td><td>" + response[i].empname + "</td><td>" +
response[i].salary + "</td></tr>");
    }
}
</script>
</body>
</html>
```

Execution:

- Go to “Debug” menu – “Start Debugging” in Visual Studio.

jQuery AJAX – .NET – Search - Example

Creating the application

- Create “c:\ajax” folder.
- Open Visual Studio 2017.
- Click on “File” menu – “New” – “Project”.
- Select “.NET Framework 4.6”.
- Select “Visual C#”.
- Select “ASP.NET Web Application (.NET Framework)”.
- Name: AjaxSearch
- Location: c:\ajax
- Solution name: AjaxSearch
- Click on OK.
- Click on “Empty”.
- Check the check boxes “MVC” and “Web API”.
- Click on OK.
- Open Solution Explorer.
- Place “jquery-3.2.1.js” file in the project.
- Right click on “Models” and click on “Add” – “New Item”. Click on “Visual C#” – “Class”. Enter the name as “Employee.cs”. Click on “Add”.

- Right click on the “Controllers” folder and click on “Add” – “Controller”. Select “MVC 5 Controller – Empty”. Click on “Add”. Enter the controller name as “HomeController”. Click on “Add”.
- Right click on “Views\Home” folder and click on “Add” – “View”. Enter the view name as “Index”. Select the template as “Empty (without model)”. Uncheck the checkbox “Use a layout page”. Click on “Add”.

AjaxSearch

- jquery-3.2.1.js
 - o Models
 - Employee.cs
 - o Controllers
 - HomeController.cs
 - o Views
 - Home
 - Index.cshtml

Models\Employee.cs

```
using System;

namespace AjaxSearch.Models
{
    public class Employee
    {
        public int empid { get; set; }
        public string empname { get; set; }
        public double salary { get; set; }
    }
}
```

Controllers\HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using AjaxSearch.Models;

namespace AjaxSearch.Controllers
{
    public class HomeController : Controller
    {
        //GET: Home/Index
        public ActionResult Index()
        {

```

```

    return View();
}

//GET: Home/searchemployees
public ActionResult searchemployees(string searchstr)
{
    List<Employee> emps = new List<Employee>()
    {
        new Employee() {empid = 1, empname = "Scott", salary = 4000},
        new Employee() {empid = 2, empname = "Allen", salary = 7500},
        new Employee() {empid = 3, empname = "Jones", salary = 9200},
        new Employee() {empid = 4, empname = "James", salary = 8400},
        new Employee() {empid = 5, empname = "Smith", salary = 5600}
    };

    List<Employee> emps2 = new List<Employee>();
    for (int i = 0; i < emps.Count; i++)
    {
        if (emps[i].empname.Contains(searchstr))
        {
            emps2.Add(emps[i]);
        }
    }
    return Json(emps2, JsonRequestBehavior.AllowGet);
}
}
}

```

Views\Home\Index.cshtml

```

<!DOCTYPE html>
<html>
<head>
    <title>jQuery AJAX - Search</title>
    <style>
        body, input, table
        {
            font-family: Tahoma;
            font-size: 24px;
        }
    </style>
</head>
<body>
    <h1>jQuery AJAX - Search</h1>
    <form>
        Search employees: <input type="text" id="txt1">
        <input type="submit" id="btn1" value="Search">
        <table id="table1" border="1">
            <tr><th>Emp ID</th><th>Emp Name</th><th>Salary</th></tr>
            </table>
    </form>

```

```

<script src="~/jquery-3.2.1.js"></script>

<script>
$("#btn1").click(fun1);
function fun1(event)
{
    event.preventDefault();
    $.ajax({ type: "GET", url: "/home/searchemployees?searchstr=" + $("#txt1").val(), success: fun2 });
}

function fun2(response)
{
    $("#table1 tr:gt(0)").remove();
    for (var i = 0; i < response.length; i++)
    {
        $("#table1").append("<tr><td>" + response[i].empid + "</td><td>" + response[i].empname + "</td><td>" +
response[i].salary + "</td></tr>");
    }
}
</script>
</body>
</html>

```

Execution:

- Go to “Debug” menu – “Start Debugging” in Visual Studio.

jQuery AJAX – .NET – Post - Example**Creating the application**

- Create “c:\ajax” folder.
- Open Visual Studio 2017.
- Click on “File” menu – “New” – “Project”.
- Select “.NET Framework 4.6”.
- Select “Visual C#”.
- Select “ASP.NET Web Application (.NET Framework)”.
- Name: AjaxPost
- Location: c:\ajax
- Solution name: AjaxPost
- Click on OK.
- Click on “Empty”.
- Check the check boxes “MVC” and “Web API”.
- Click on OK.

- Open Solution Explorer.
- Place “jquery-3.2.1.js” file in the project.
- Right click on “Models” and click on “Add” – “New Item”. Click on “Visual C#” – “Class”. Enter the name as “Employee.cs”. Click on “Add”.
- Right click on the “Controllers” folder and click on “Add” – “Controller”. Select “MVC 5 Controller – Empty”. Click on “Add”. Enter the controller name as “HomeController”. Click on “Add”.
- Right click on “Views\Home” folder and click on “Add” – “View”. Enter the view name as “Index”. Select the template as “Empty (without model)”. Uncheck the checkbox “Use a layout page”. Click on “Add”.

AjaxPost

- jquery-3.2.1.js
 - Models
 - Employee.cs
 - Controllers
 - HomeController.cs
 - Views
 - Home
 - Index.cshtml

Models\Employee.cs

```
using System;

namespace AjaxPost.Models
{
    public class Employee
    {
        public int empid { get; set; }
        public string empname { get; set; }
        public double salary { get; set; }
    }
}
```

Controllers\HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using AjaxPost.Models;

namespace AjaxPost.Controllers
{
```

```

public class HomeController : Controller
{
    //GET: Home/Index
    public ActionResult Index()
    {
        return View();
    }

    //GET: Home/InsertEmployee
    public ActionResult InsertEmployee(Employee emp)
    {
        return Content("Successfully Inserted");
    }
}

```

Views\Home\Index.cshtml

```

<!DOCTYPE html>
<html>
<head>
    <title>jQuery AJAX - Post</title>
    <style>
        body, input
        {
            font-family: Tahoma;
            font-size: 24px;
        }
    </style>
</head>
<body>
    <h1>jQuery AJAX - Post</h1>
    <form>
        Emp ID: <input type="text" id="txt1"><br>
        Emp Name: <input type="text" id="txt2"><br>
        Salary: <input type="text" id="txt3"><br>
        <input type="submit" id="button1" value="Insert">
        <div id="div1"></div>
    </form>

    <script src="~/jquery-3.2.1.js"></script>

    <script>
        $("#button1").click(fun1);
        function fun1(event)
        {
            event.preventDefault();
            var mydata = { "empid": $("#txt1").val(), "empname": $("#txt2").val(), "salary": $("#txt3").val() };
            $.ajax({ type: "POST", url: "/home/insertemployee", success: fun2, data: mydata });
        }

        function fun2(response)
        {
    
```

```
$("#div1").html(response);
}
</script>
</body>
</html>
```

Execution:

- Go to “Debug” menu – “Start Debugging” in Visual Studio.

jQuery AJAX – .NET – Put - Example

Creating the application

- Create “c:\ajax” folder.
- Open Visual Studio 2017.
- Click on “File” menu – “New” – “Project”.
- Select “.NET Framework 4.6”.
- Select “Visual C#”.
- Select “ASP.NET Web Application (.NET Framework)”.
- Name: AjaxPut
- Location: c:\ajax
- Solution name: AjaxPut
- Click on OK.
- Click on “Empty”.
- Check the check boxes “MVC” and “Web API”.
- Click on OK.
- Open Solution Explorer.
- Place “jquery-3.2.1.js” file in the project.
- Right click on “Models” and click on “Add” – “New Item”. Click on “Visual C#” – “Class”. Enter the name as “Employee.cs”. Click on “Add”.
- Right click on the “Controllers” folder and click on “Add” – “Controller”. Select “MVC 5 Controller – Empty”. Click on “Add”. Enter the controller name as “HomeController”. Click on “Add”.
- Right click on “Views\Home” folder and click on “Add” – “View”. Enter the view name as “Index”. Select the template as “Empty (without model)”. Uncheck the checkbox “Use a layout page”. Click on “Add”.

AjaxPut

- jquery-3.2.1.js
 - o Models
 - Employee.cs
 - o Controllers
 - HomeController.cs
 - o Views
 - Home
 - Index.cshtml

Models\Employee.cs

```
using System;

namespace AjaxPut.Models
{
    public class Employee
    {
        public int empid { get; set; }
        public string empname { get; set; }
        public double salary { get; set; }
    }
}
```

Controllers\HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using AjaxPut.Models;
namespace AjaxPut.Controllers
{
    public class HomeController : Controller
    {
        //GET: Home/Index
        public ActionResult Index()
        {
            return View();
        }

        //GET: Home/UpdateEmployee
        public ActionResult UpdateEmployee(Employee emp)
        {
            return Content("Successfully Updated");
        }
    }
}
```

Views\Home\Index.cshtml

```

<!DOCTYPE html>
<html>
<head>
<title>jQuery AJAX - Put</title>
<style>
body, input
{
    font-family: Tahoma;
    font-size: 24px;
}
</style>
</head>
<body>
<h1>jQuery AJAX - Put</h1>
<form>
Existing Emp ID: <input type="text" id="txt1"><br>
Emp Name: <input type="text" id="txt2"><br>
Salary: <input type="text" id="txt3"><br>
<input type="submit" id="button1" value="Update">
<div id="div1"></div>
</form>

<script src="~/jquery-3.2.1.js"></script>
<script>
$("#button1").click(fun1);
function fun1(event)
{
    event.preventDefault();
    var mydata = { "empid": $("#txt1").val(), "empname": $("#txt2").val(), "salary": $("#txt3").val() };
    $.ajax({ type: "PUT", url: "/home/updateemployee", data: mydata, success: fun2 });
}

function fun2(response)
{
    $("#div1").html(response);
}
</script>
</body>
</html>

```

Execution:

- Go to “Debug” menu – “Start Debugging” in Visual Studio.

jQuery AJAX – .NET – Delete - Example**Creating the application**

- Create “c:\ajax” folder.

- Open Visual Studio 2017.
- Click on “File” menu – “New” – “Project”.
- Select “.NET Framework 4.6”.
- Select “Visual C#”.
- Select “ASP.NET Web Application (.NET Framework)”.
- Name: AjaxDelete
- Location: c:\ajax
- Solution name: AjaxDelete
- Click on OK.
- Click on “Empty”.
- Check the check boxes “MVC” and “Web API”.
- Click on OK.
- Open Solution Explorer.
- Place “jquery-3.2.1.js” file in the project.
- Right click on the “Controllers” folder and click on “Add” – “Controller”. Select “MVC 5 Controller – Empty”. Click on “Add”. Enter the controller name as “HomeController”. Click on “Add”.
- Right click on “Views\Home” folder and click on “Add” – “View”. Enter the view name as “Index”. Select the template as “Empty (without model)”. Uncheck the checkbox “Use a layout page”. Click on “Add”.

AjaxDelete

- jquery-3.2.1.js
 - o Controllers
 - HomeController.cs
 - o Views
 - Home
 - Index.cshtml

Controllers\HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace AjaxDelete.Controllers
{
    public class HomeController : Controller
    {
```

```
//GET: Home/Index
public ActionResult Index()
{
    return View();
}

//GET: Home/DeleteEmployee
public ActionResult DeleteEmployee(int empid)
{
    return Content("Successfully Deleted");
}
}
```

Views\Home\Index.cshtml

```
<!DOCTYPE html>
<html>
<head>
<title>jQuery AJAX - Delete</title>
<style type="text/css">
body, input
{
    font-family: Tahoma;
    font-size: 24px;
}
</style>
</head>
<body>
<h1>jQuery AJAX - Delete</h1>
<form>
Existing Emp ID: <input type="text" id="txt1"><br>
<input type="button" id="button1" value="Delete">
<div id="div1"></div>
</form>

<script src="~/jquery-3.2.1.js"></script>

<script>
$("#button1").click(fun1);
function fun1()
{
    $.ajax({ type: "DELETE", url: "/home/deleteemployee?empid=" + $("#txt1").val(), success: fun2 });
}
function fun2(response)
{
    $("#div1").html(response);
}
</script>
</body>
</html>
```

Execution:

- Go to “Debug” menu – “Start Debugging” in Visual Studio.

jQuery AJAX – .NET – Grid - Example

Creating the application

- Create “c:\ajax” folder.
- Open Visual Studio 2017.
- Click on “File” menu – “New” – “Project”.
- Select “.NET Framework 4.6”.
- Select “Visual C#”.
- Select “ASP.NET Web Application (.NET Framework)”.
- Name: AjaxGrid
- Location: c:\ajax
- Solution name: AjaxGrid
- Click on OK.
- Click on “Empty”.
- Check the check boxes “MVC” and “Web API”.
- Click on OK.
- Open Solution Explorer.
- Place “jquery-3.2.1.js” file in the project.
- Right click on “Models” and click on “Add” – “New Item”. Click on “Visual C#” – “Class”. Enter the name as “Employee.cs”. Click on “Add”.
- Right click on the “Controllers” folder and click on “Add” – “Controller”. Select “MVC 5 Controller – Empty”. Click on “Add”. Enter the controller name as “HomeController”. Click on “Add”.
- Right click on “Views\Home” folder and click on “Add” – “View”. Enter the view name as “Index”. Select the template as “Empty (without model)”. Uncheck the checkbox “Use a layout page”. Click on “Add”.

AjaxGrid

- jquery-3.2.1.js
 - o Models
 - Employee.cs
 - o Controllers
 - HomeController.cs
 - o Views

- Home
 - Index.cshtml

Models\Employee.cs

```
using System;

namespace AjaxGrid.Models
{
    public class Employee
    {
        public int empid { get; set; }
        public string empname { get; set; }
        public double salary { get; set; }
    }
}
```

Controllers\HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using AjaxGrid.Models;

namespace AjaxGrid.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult GetData()
        {
            List<Employee> emps = new List<Employee>()
            {
                new Employee() {empid = 1, empname = "Scott", salary = 4000},
                new Employee() {empid = 2, empname = "Allen", salary = 7500},
                new Employee() {empid = 3, empname = "Jones", salary = 9200},
                new Employee() {empid = 4, empname = "James", salary = 8400},
                new Employee() {empid = 5, empname = "Smith", salary = 5600}
            };
            return Json(emps, JsonRequestBehavior.AllowGet);
        }

        public ActionResult InsertEmployee(Employee emp)
        {
            return Content("Successfully Inserted");
        }
    }
}
```

```

public ActionResult UpdateEmployee(Employee emp)
{
    return Content("Successfully Updated");
}

public ActionResult DeleteEmployee(int empid)
{
    return Content("Successfully Deleted");
}
}

```

Views\Home\Index.cshtml

```

<!DOCTYPE html>
<html>
<head>
<title>jQuery AJAX - Grid</title>
<style>
body, input, table
{
    font-family: Tahoma;
    font-size: 24px;
}
#table1 td
{
    padding: 5px;
}
</style>
</head>
<body>
<h1>jQuery AJAX - Grid</h1>
<form>
<table id="table1" border="1">
<tr><th>Emp ID</th><th>Emp Name</th><th>Salary</th></tr>
</table>
</form>

<script src="~/jquery-3.2.1.js"></script>

<script>
//get
fun1();
function fun1()
{
    $.ajax({ type: "GET", url: "/home/getdata", success: fun2 });
}
function fun2(response)
{
    $("#table1 tr:gt(0)").remove();
    for (var i = 0; i < response.length; i++)
    {

```

```

        $("#table1").append("<tr><td>" + response[i].empid + "</td><td>" + response[i].empname + "</td><td>" +
        response[i].salary + "</td><td><input type='button' value='Edit' class='class1'><input type='button' value='Delete'
        class='class2'></td></tr>");
    }
    $("#table1").append("<tr><td><input type='text' class='class3' placeholder='Emp ID'></td><td><input type='text'
        class='class3' placeholder='Emp Name'></td><td><input type='text' class='class3' placeholder='Salary'></td>
        <td><input type='button' value='Insert' class='class4'></td></tr>");
    }

//post
$(document).on("click", ".class4", fun3);
function fun3()
{
    var d = { "empid": $(".class3:eq(0)").val(), "empname": $(".class3:eq(1)").val(), "salary": $(".class3:eq(2)").val() };
    $.ajax({ type: "POST", url: "/home/insertemployee", data: d, success: fun4 });
}
function fun4(response)
{
    $("#table1 tr:last").before("<tr><td>" + $(".class3:eq(0)").val() + "</td><td>" + $(".class3:eq(1)").val() + "</td><td>" +
    + $(".class3:eq(2)").val() + "</td><td><input type='button' value='Edit' class='class1'><input type='button'
    value='Delete' class='class2'></td></tr>");
    $(".class3").val("");
}

//put
var currentrow = null;
$(document).on("click", ".class1", fun5);
function fun5()
{
    currentrow = $(this).parent().parent();
    currentrow.html("<td><input type='text' class='class5' placeholder='Emp ID' value=''" +
    currentrow.find("td:eq(0)").text() + "'></td><td><input type='text' class='class5' placeholder='Emp Name' value=''" +
    currentrow.find("td:eq(1)").text() + "'></td><td><input type='text' class='class5' placeholder='Salary' value=''" +
    currentrow.find("td:eq(2)").text() + "'></td><td><input type='button' value='Update' class='class6'></td>");
    currentrow.find(".class5:eq(0)").focus();
}
$(document).on("click", ".class6", fun6);
function fun6()
{
    var d = { "empid": $(".class5:eq(0)").val(), "empname": $(".class5:eq(1)").val(), "salary": $(".class5:eq(2)").val() };
    $.ajax({ type: "PUT", url: "/home/updateemployee", data: d, success: fun7 });
}
function fun7(response)
{
    currentrow.html("<td>" + $(".class5:eq(0)").val() + "</td><td>" + $(".class5:eq(1)").val() + "</td><td>" +
    + $(".class5:eq(2)").val() + "</td><td><input type='button' value='Edit' class='class1'><input type='button' value='Delete'
    class='class2'></td>");
}

//delete
var currentrow2 = null;
$(document).on("click", ".class2", fun8);
function fun8()
{
    currentrow2 = $(this).parent().parent();
}

```

```
var empid = currentrow2.find("td:eq(0)").text();
var empname = currentrow2.find("td:eq(1)").text();
if(confirm("Are you sure to delete " + empname + "?"))
{
    $.ajax({ type: "DELETE", url: "/home/deleteemployee?empid=" + empid, success: fun9 });
}
}
function fun9(response)
{
    currentrow2.remove();
}
</script>
</body>
</html>
```

Execution:

- Go to “Debug” menu – “Start Debugging” in Visual Studio.