

WAD / UI Technologies

Harsha Vardhan

UI Expert

Contents

Fundamentals of WAD / UI Technologies	8
Introduction to WAD / UI Technologies	9
Fundamentals of WAD / UI Technologies	10
H T M L	14
Introduction to HTML	15
HTML – First Example	21
Headings	24
Paragraphs.....	26
Text Formatting Tags.....	28
	33
<hr>.....	34
<pre>	35
<abbr>	36
<bdo>.....	37
.....	38
<a>	39
List tags.....	46
Table tags	50
<iframe>.....	55
HTML Entities.....	58
<meta>	60
Forms.....	61
<div> and	81
DOCTYPE	82
Deprecated Tags.....	83
X H T M L	84
Introduction to XHTML.....	85
XHTML Rules	86
C S S	89
Introduction to CSS	90
color.....	93
background-color	94

WAD / UI Technologies

font-family	95
font-size	96
font-weight	97
font-style	98
letter-spacing	99
word-spacing	101
Types of colors	103
line-height	106
text-decoration	108
text-transform	110
text-align	112
text-indent	114
background-image	116
background-repeat	118
background-position	121
background-attachment	123
list-style-type for 	125
list-style-type for 	127
list-style-image	130
<div> tag	131
"width" and "height" properties	133
float	135
clear	138
border-style, border-width, border-color	140
border - shortcut	142
border - sides	144
margin	146
margin - sides	148
margin - shortcut	151
padding	153
padding - sides	155
padding - shortcut	156
Box Model.....	157

WAD / UI Technologies

opacity.....	158
position	159
display	167
visibility.....	169
overflow.....	171
Types of Style Sheets.....	174
CSS Selectors.....	177
Tag Selector.....	178
ID Selector	179
Class Selector.....	180
Compound Selector.....	181
Grouping Selector.....	182
Child Selector.....	183
Direct Child Selector	184
Adjacent Siblings Selector.....	185
Adjacent One Sibling Selector	186
Attribute Selector	187
Hover Selector	188
Focus Selector	189
Universal Selector	190
First-child selector.....	191
Last-child selector	193
Nth-child selector	195
Nth-child(even) selector.....	197
Nth-child(odd) selector.....	198
Before selector	199
After selector.....	200
Selection selector.....	201
CSS Style Precedence.....	202
Table Styles	204
Hyperlink Styles.....	206
Menubar	208
Header.....	210

WAD / UI Technologies

Static Page Template	214
Static Page Template	215
Responsive Web Design	224
Responsive Web Design	225
JavaScript	239
Introduction to JavaScript	240
Variables	242
Arrays	244
Operators	246
1. Arithmetical Operators	247
2. Assignment Operators.....	249
3. Increment and Decrement Operators	251
4. Relational Operators	252
5. Logical Operators	254
6. Concatenation Operator	255
Control Statements.....	256
1) if.....	257
2) Switch-case.....	263
3) While	265
4) Do-While	266
5) For	267
Break	269
Continue	270
Noscript.....	271
Popup boxes.....	272
Functions	275
DOM.....	279
Getting Existing Elements from DOM and Changing Content.....	281
Manipulating Attributes of Existing Elements of DOM	288
Manipulating CSS of Existing Elements of DOM.....	291
Event Handling.....	292
“Click” event.....	293
“Dblclick” event.....	294

WAD / UI Technologies

"Mouseover" and "Mouseout" events.....	295
"Mousemove" event.....	298
"Keyup" event.....	300
"Keypress" event.....	301
"Focus" and "Blur" events	305
"Change" event.....	307
"Contextmenu" event.....	313
"Cut", "Copy", "Paste" events	314
>this keyword	316
Adding New Elements to DOM.....	318
Removing Existing Elements from DOM.....	320
Image Gallery	322
Photo Navigation.....	324
Random	326
Conversion Functions.....	328
Login.....	331
String Functions.....	332
Date Functions.....	336
setTimeout().....	341
setInterval().....	342
Validations.....	345
Regular Expressions	348
Window.print()	350
Popup Window.....	351
External JavaScript	353
Object Oriented Programming.....	355
Object Literals.....	357
New Object.....	360
Constructor Function	361
Object.Keys	364
Stringify	366
Parse.....	368
Object Array Literal.....	369

WAD / UI Technologies

Object Array	372
Complex Object Literal.....	373
Prototype.....	375
Inheritance	378
Clousers	380
Hoisting	382

HARSHA

Fundamentals of WAD / UI Technologies

Introduction to WAD / UI Technologies

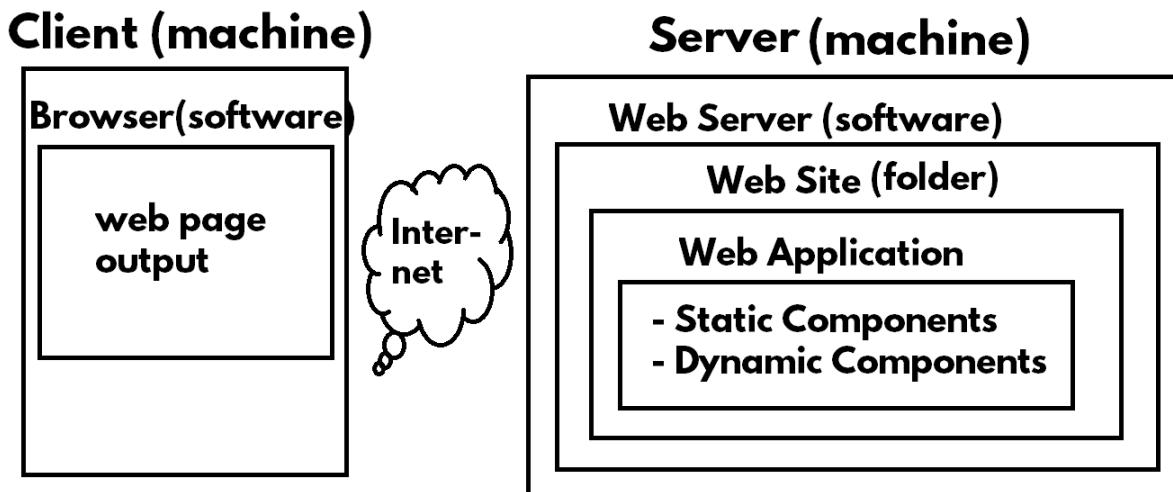
Introduction to WAD / UI Technologies

- WAD course covers all the client side web technologies.
- To create a real-time web sites (that can interact with the database), you should use “WAD + any one server side web technology”.
- **WAD Course:**
 1. HTML 4
 2. XHTML
 3. CSS 2.1
 4. Static Web Design
 5. Responsive Web Design
 6. JavaScript
 7. HTML 5
 8. CSS 3
 9. Bootstrap
 10. jQuery
 11. jQuery UI / Plugins
 12. AJAX
 13. AngularJS

Fundamentals of WAD / UI Technologies

Fundamentals of WAD / UI Technologies

- **Application:** It is a program (compiled program) that runs based on the operating system.
- **Client:** It is a machine or device (desktop, laptop, tablet, phone or Smart TV), which can access the data from server.
- **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.
 - Important browsers:
 - Google Chrome
 - Mozilla Firefox
 - Microsoft Internet Explorer
 - Microsoft Edge
 - Apple Safari
 - Android Browser
 - Opera
- **Server:** It is a machine that serves data to clients.
- **Web Server:** It is a software installed on the server, which serves websites to browser. A server can have any no. of web servers.
- **Web site:** It is a folder in the web server, where the servable data will be stored. A web server can have any no. of web sites.
- **Web application:** It is a collection of web components, which is stored in the web site.
- **Web Component:** It is a file present in the web application. Web components are two types:
 - **Static Web Component:** Provides same output for all requests. Ex: HTML files, CSS files, Image files etc.
 - **Dynamic Web Component:** Provides different output for each request. Ex: ASPX files

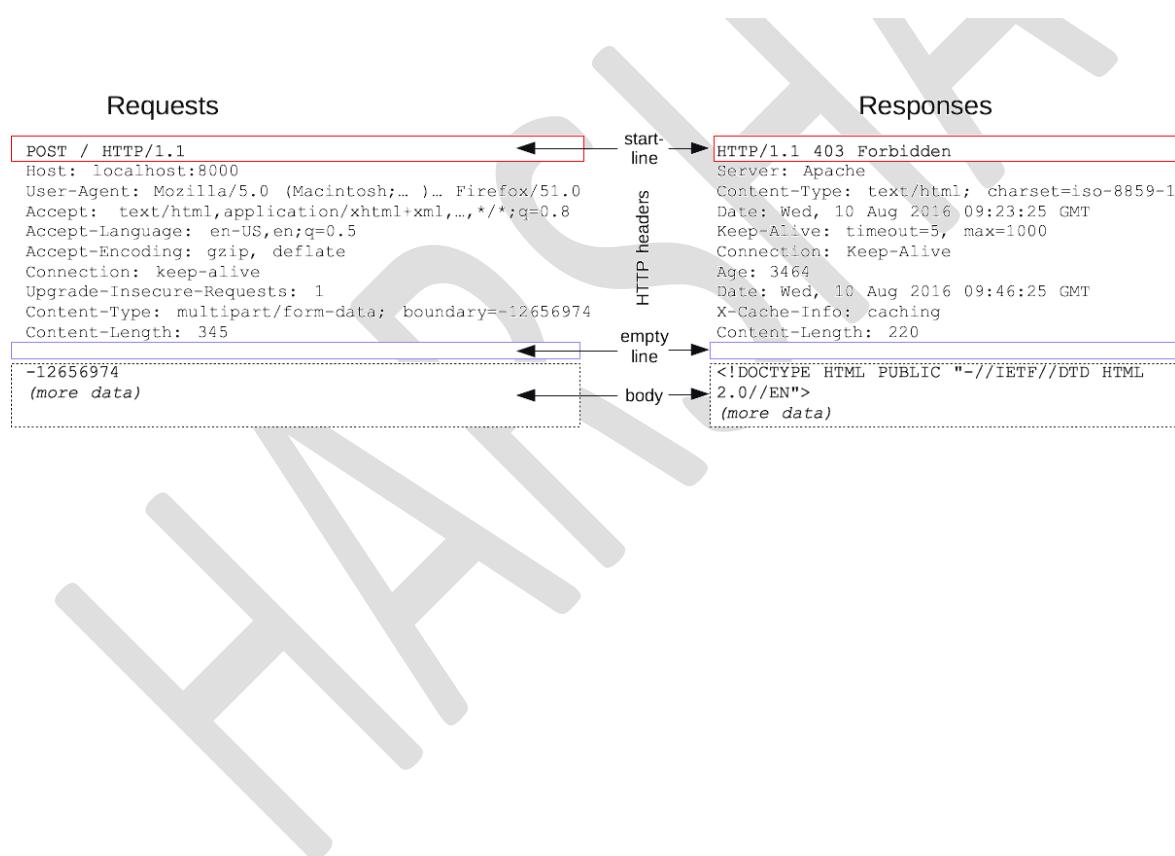
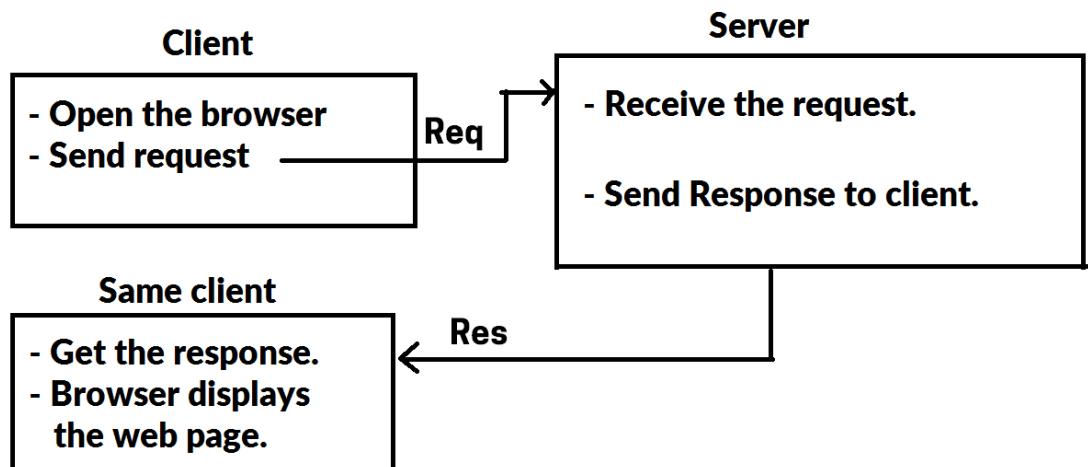


- **Web technologies:** These are programming languages that are used to develop (create) web applications.
- **Types of web technologies:** Web technologies are two types. 1) Client side web technologies. 2) Server side web technologies
- **Client side web technologies:** These are the programming languages or concepts, which code executes on the client (browser). Ex: HTML, CSS, JavaScript, jQuery, AngularJS, Bootstrap, AJAX etc.
- **Server side web technologies:** These are the programming languages or concepts, which code executes on web server. Ex: .NET, Java, PHP, Ruby on Rails etc.
- **MIME Types:** Represents type of data (content).
 - text/plain
 - text/html
 - text/css
 - text/javascript
 - application/json
 - text/xml
 - image/png
 - image/jpeg
 - audio/mp3
 - video/mp4

WAD / UI Technologies

- **Character Encoding Formats:** The characters (alphabets, digits or symbols) are converted into equivalent numbers based on “character encoding formats”.
 - ASCII (American Standard Code for Information Interchange): It contains standard characters only.
 - A-Z: 65 to 90
 - a-z: 97 to 122
 - 0-9: 48 to 57
 - Space: 32
 - Backspace: 8
 - Unicode: It is the extension of ASCII. Unicode includes with all language characters worldwide.
- **Http:** It is a protocol, which provides a set of rules to send request to server and get response from server.
- **HTTP Status Codes:** Represents type of data (content).
 - 100: Continue
 - 200: OK
 - 302: Redirection
 - 304: Not Modified
 - 400: Bad Request
 - 401: Unauthorized
 - 404: Page not found
 - 500: Internal Server Error
- **Http Request:** It is a HTTP message sent from client to server.
- **Http Response:** It is a HTTP message sent from server to client.

WAD / UI Technologies





HTML

Introduction to HTML

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. A markup language is a language that 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 format. 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.

Tag

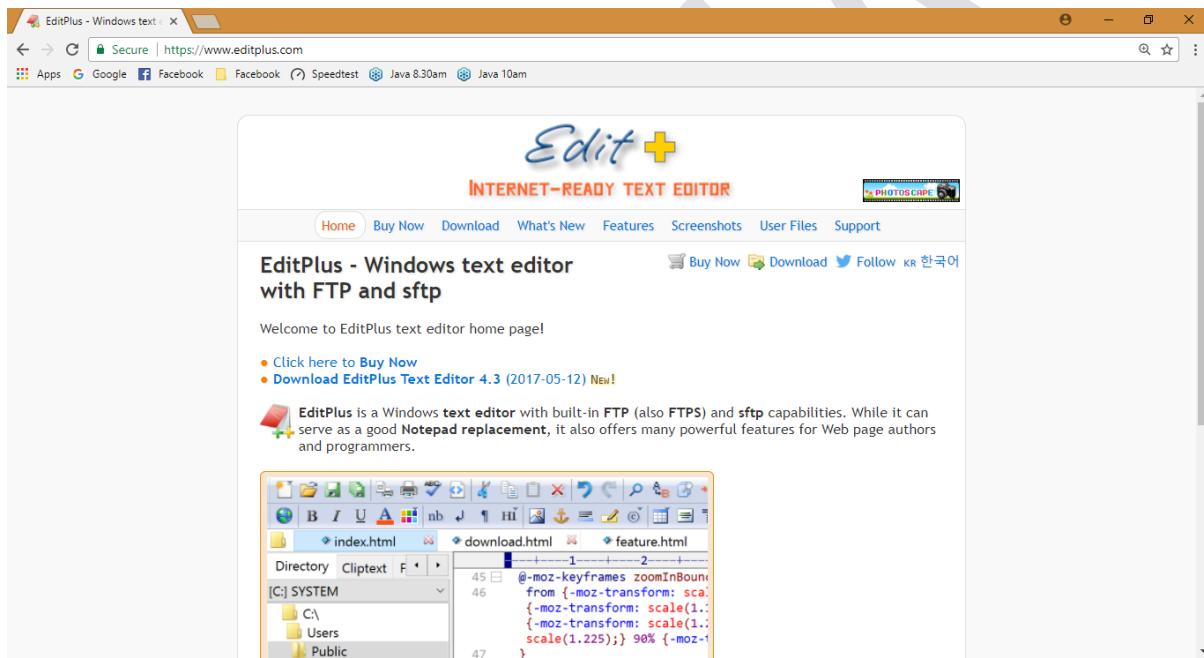
- A tag is a keyword, enclosed within "<" and ">" in HTML language.
- **Syntax:** <tag>

Types of tags

- Tags are two types:
 1. **Paired tags:** Contains starting tag and ending tag. Ex: <h1>hello</h1>
 2. **Unpaired tags:** Contains single tag only (no separate ending tag).
Ex: <hr>

Installing EditPlus

Go to <https://www.editplus.com>



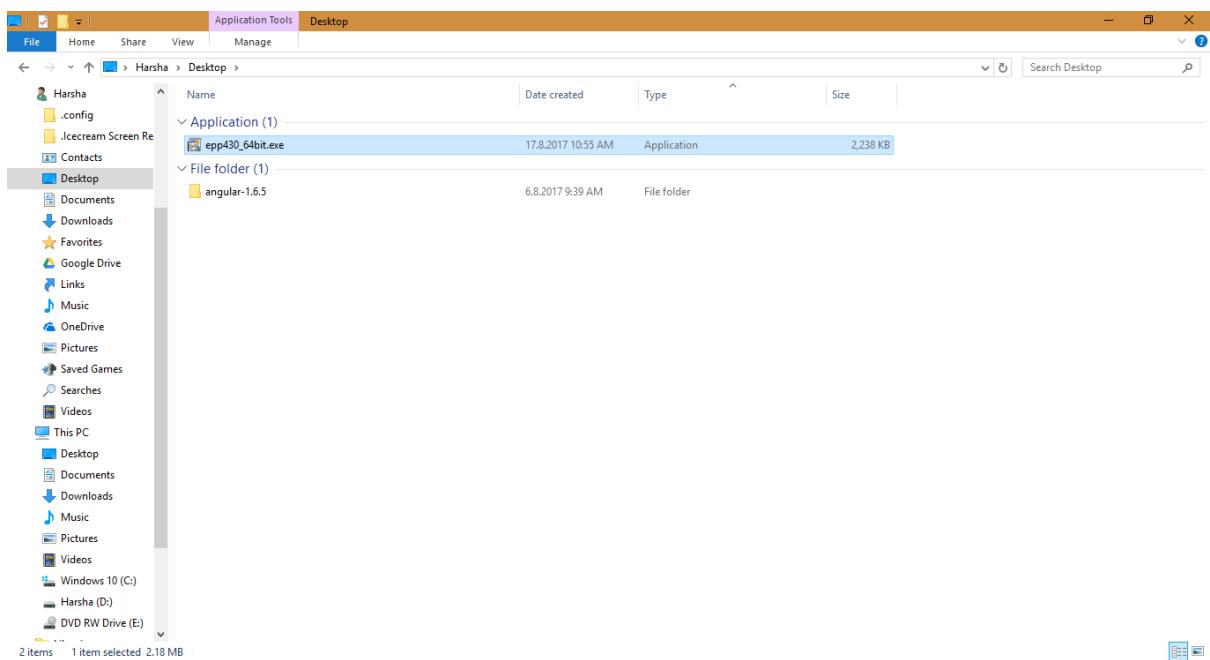
Click on “Download EditPlus Text Editor”

If your machine is 32-bit machine, click on “Download (1.95 MB)”.

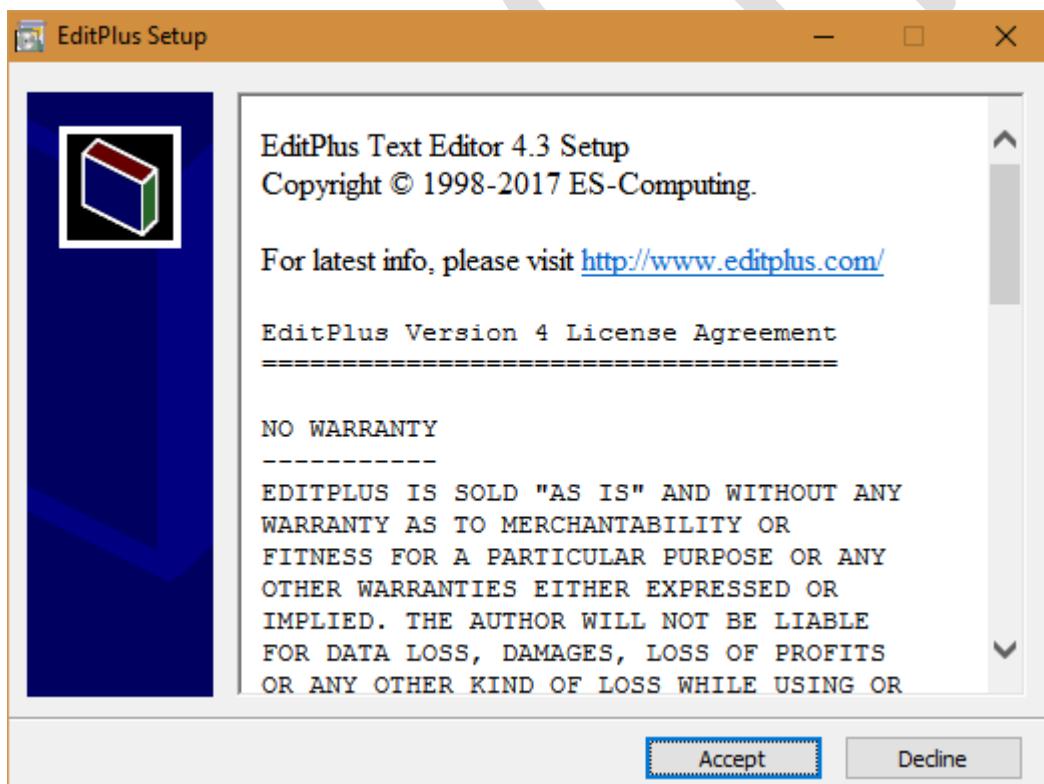
If your machine is 64-bit machine, click on “Download 64-bit (2.18 MB)”.

If you click on “Download 64-bit” option, you will get a file “epp430_64bit.exe”.

WAD / UI Technologies



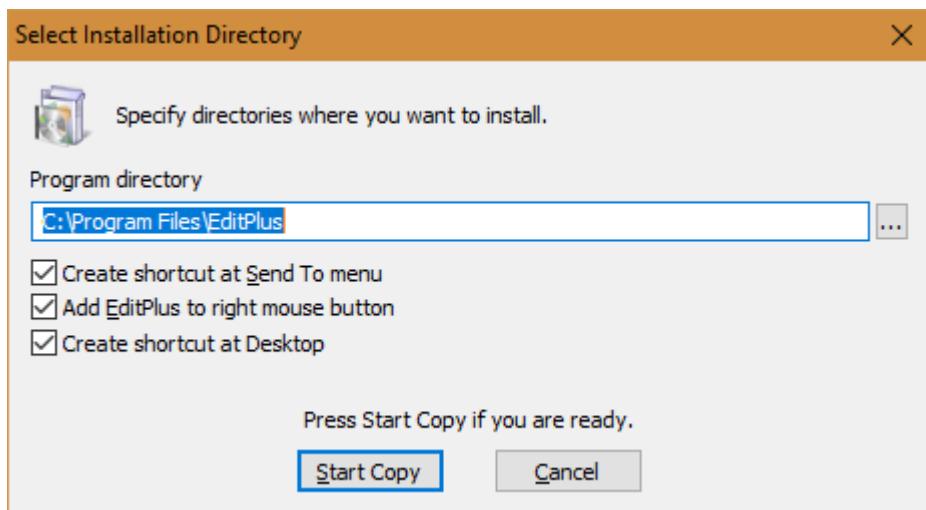
Double click on "epp430_64bit.exe" file.



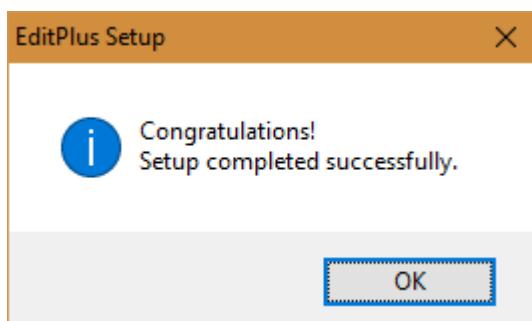
Click on "Accept".

Click on "Yes" to run as administrator.

WAD / UI Technologies



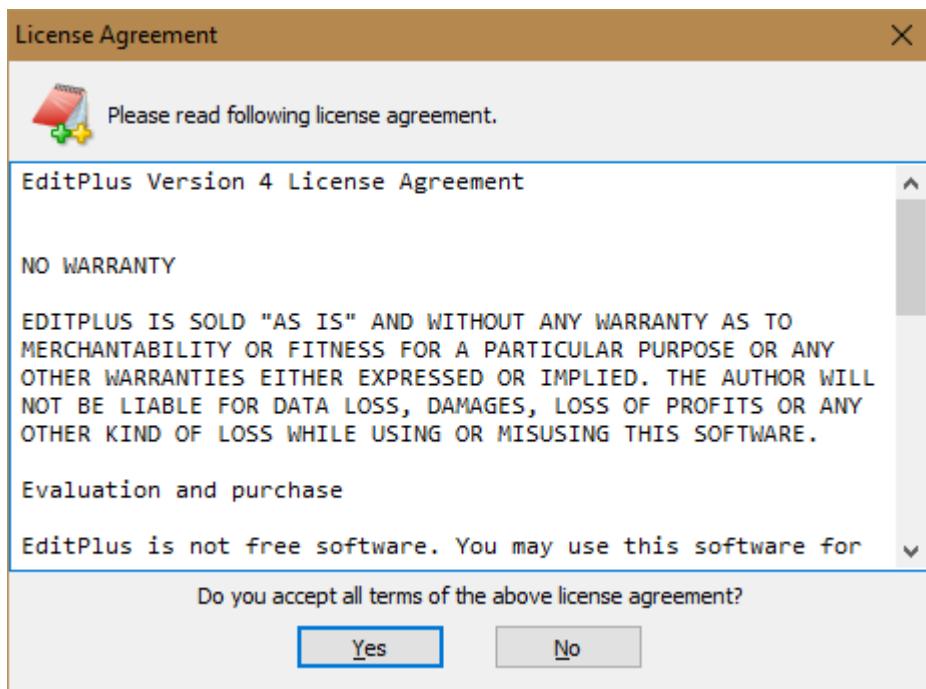
Click on "Start Copy".



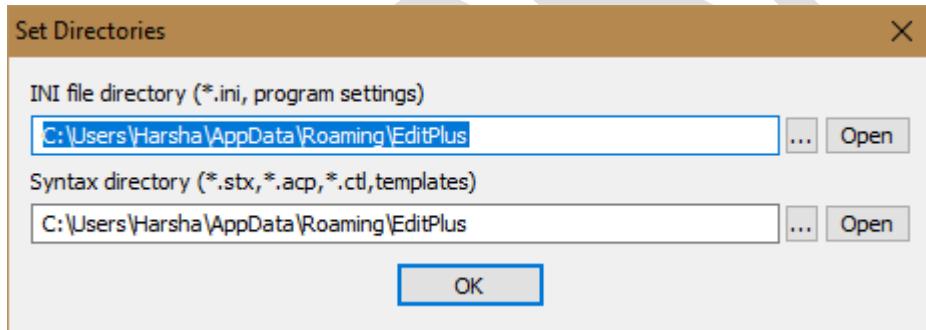
Click on OK.

EditPlus Installation is finished.

Open "Start" – "EditPlus".



Click on "Yes".



Click on OK.

Click on "Enter Registration Code".

Enter the following registration code:

Name: Harsha

Serial: 5ACBF-6AA58-14Z84-6AW10-ADT3D

WAD / UI Technologies

Enter Registration Code X

If you have paid for the license, please enter BOTH 'username' and 'regcode' EXACTLY as shown in the license e-mail that you received.

Username:

Regcode:

Register Cancel Help

Click on "Register".

HTML – First Example

Syntax of HTML Program

- Every html program should have the following syntax:

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

First Example in HTML:

- Open Notepad / Edit Plus.
- Type the following program:

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

- Go to “File” menu and click on “Save”.
- Type the file name as “first.html”.
- Select the folder as “c:\html”.
- Click on “Save”.
- Go to “c:\html” folder and double click on “Page1.html”.
- It opens “first.html” in browser (Google Chrome, Mozilla Firefox, Internet Explorer etc.).

- You will get the output as follows:



<html>

- <html> tag represents starting and ending point of the html program.
- **Syntax:**
<html> </html>
- **Example:**
<html> </html>

<head>

- <head> tag represents non content information of the page.
- **Syntax:**
<head> </head>
- **Example:**
<head> </head>

<body>

- <body> tag represents content information of the page.
- **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.
- <head> tag is used to specify non-content information of the page.
- <title> is a paired tag.
- **Syntax:**
`<title>Title here</title>`
- **Example:**
`<title>My title</title>`

Headings

<h1>

- It is used to create first level heading (main heading).
- 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 fourth 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>

- It is used to create a paragraph.

- 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>

HARSHA

Text Formatting Tags

- It is used to display the text in bold.
- It is a paired tag.
- **Syntax:**
`bold text`
- **Example:**
`Hello`

Example of :

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

<i>

- It is used to display the text in 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>
```

<u>

- It is used to display the text in underline.
- 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>
```

<strike>

- It is used to display the text in strikeout.
- 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>
```


- It is used to display the text in strong.
- The strong tag content will be pronounced strongly in screen readers for the blind people.
- It is a paired tag.
- Syntax:
`strong text`
- Example:
`Hello`

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>
```


- It is used to display the text in emphasis (special status).
- The strong tag content will be pronounced stylishly in screen readers for the blind people.
- It is a paired tag.
- **Syntax:**
`emphasis text`
- **Example:**
`Hello`

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>
```

<sup>

- It is used to display the text in superscript (The text appears a bit upper side of normal line).
- It is a paired tag.
- **Syntax:**
`^{superscript text}`
- **Example:**
`^{Hello}`

Example on <sup>:

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

<sub>

- It 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>
```


- It is used to moves the cursor to the next line.
- It is an unpaired tag.
- **Syntax:**

- **Example:**

Example on
:

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

<hr>

<hr>

- It is used to display a horizontal line (horizontal ruler).

- 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>

<pre>

- It is used to display the text as-it-is, along with the spaces and line breaks.
- It is a paired tag.
- **Syntax:**

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

- **Example:**

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

Example on <pre>:

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

<abbr>

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>

<abbr>

- It is used to display full-form of a short-form when the user places mouse pointer on it.
- It is a paired 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>
```

<bdo>

<bdo>

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


- It is used to display an image in the web page.

- 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.

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).

Example on :

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

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

<a>

<a>

- It is used to create a hyperlink. When the user clicks on the hyperlink, the specified web page or web site will be opened.
- It is a paired tag.
- **Syntax:**
`link text here`
- **Example:**
`Google`
- **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.
 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:

Countries.html

```
<html>  
  <head>  
    <title>Hyperlinks</title>  
  </head>  
  <body>  
    <h1>Hyperlinks</h1>  
    <a href="india.html">India</a>  
    <a href="uk.html">UK</a>  
    <a href="us.html">US</a>  
  </body>  
</html>
```

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. 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>

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

List tags

- UL stands for “Unordered List”.
- It is used to display the list of items with bullets.
- Inside tag, you should place one or more tags.
- 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>
    <li>Canada</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>
```


- 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.
- 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>
```

<dl>

- DL stands for “Definition List”.
- It 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> is used to specify definition title.
- <dd> is used to specify definition data.
- **Syntax:**

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

- **Example:**

```
<dl>
    <dt>HTML</dt>
    <dd>HTML is used to design web pages.</dd>
    <dt>CSS</dt>
    <dd>CSS is used to apply styles.</dd>
    <dt>JavaScript</dt>
    <dd>JavaScript is used to create functionality.</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>
        </dl>
    </body>
</html>
```

```
<dt>JavaScript</dt>
<dd>JavaScript is used to create functionality in the web page.</dd>
</dl>
</body>
</html>
```

HARSHA

Table tags

<table>

- <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 px": Specifies the no. of pixels distance between cell border and cell content.

Example on simple use of <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>
```

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>
        <td colspan="3".NET</td>
        <td colspan="3">Java</td>
      </tr>
      <tr>
        <td>C#.NET</td>
        <td>ASP.NET</td>
        <td>ADO.NET</td>
        <td>Core Java</td>
        <td>Adv Java</td>
        <td>J2EE</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>
        <td rowspan="3".NET</td>
        <td>C#.NET</td>
      </tr>
      <tr>
        <td>ASP.NET</td>
      </tr>
      <tr>
        <td>J2EE</td>
      </tr>
    </table>
  </body>
</html>
```

```
<tr>
    <td>ADO.NET</td>
</tr>
<tr>
    <td rowspan="3">Java</td>
    <td>Core Java</td>
</tr>
<tr>
    <td>Adv Java</td>
</tr>
<tr>
    <td>J2EE</td>
</tr>
</table>
</body>
</html>
```

<iframe>

<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 video with <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

- 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
	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>
```

HARSHA

<meta>

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

Forms

<input>

- <input> tag is used to create a form control (form element).
- 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**”: Creates textbox. Textbox is used to accept a string value from the user.
- “**password**”: Creates password textbox. Password textbox is used to accept a password from the user.
- “**submit**”: Creates a submit button. Submit button is used to submit the form to the server page.
- “**image**”: Creates a “image submit” button. Submit button is used to submit the form to the server page.
- “**button**”: Creates a normal button. You can call “JavaScript Click event” when the user clicks on the button.
- “**checkbox**”: Creates a check box. Checkbox is used to display Yes/No type of option to the user.
- “**radio**”: Creates a radio button. Radiobutton is used to display two or more options to the user and allow the user to select any one of them. The “name” of radio buttons should be same to group-up them.
- “**hidden**”: Creates a hidden field. Hidden field will not be appear in the web page; but will be submitted to the server.
- “**file**”: Creates a file browse button. Browse button is used for “attachment” option.

- “**reset**”: Creates a reset button. Reset button clears all fields (textboxes and others) within the current form.

2. `maxlength`

- Specifies the maximum no. of characters that can be typed in the textbox.

3. `readonly="readonly"`

- Makes the textbox as readonly; so that the user can see the value but can't type anything in the textbox.

4. `tabindex`

- Specifies tab order.

5. `value`

- Represents the current value of the input element.

6. `name`

- Represents programmatic name of the input element that will be submitted to the server.

7. `id`

- Represents identification name of the input element that can be used in html, css, and javascript to get the element programmatically.

8. `src`

- Used to specify the path of the image in case of `<input type="image">`

9. `width`

- Used to specify the width of the image in case of `<input type="image">`

10. `height`

- Used to specify the height of the image in case of `<input type="image">`

11. checked="checked"

- Used to check the checkbox or radio button, in case of <input type="checkbox"> or <input type="radio">

12. disabled="disabled"

- Used to disable the button.

Example on Textboxes:

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

Example on Password TextBox:

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

Example on CheckBox:

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

Example on CheckBox - Checked:

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

Example on Radio Buttons:

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

Example on Radio Button - Checked:

```
<html>
  <head>
    <title>radio - checked</title>
  </head>
  <body>
    <h1>radio - checked</h1>
    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
  </body>
</html>
```

Example on File Browse Button:

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

<form>

- <form> tag is used to group-up the input elements; so that we can submit the entire form to the server.
- A form is a collection of form elements such as <input>, <textarea> and <select>.
- <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">
    <input type="text" name="Username">
        <input type="submit">
</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.

Example on **<form>** tag:

```
<html>
    <head>
        <title>form</title>
    </head>
    <body>
        <h1>form</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">
```

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

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>
```

Example on Submit button:

```
<html>
  <head>
    <title>Submit</title>
  </head>
  <body>
    <h1>Submit</h1>
    <form action="http://localhost/someaddress">
      First Name:
      <input type="text" name="firstname"><br>
```

Last Name:

```
<input type="text" name="lastname"><br>
<input type="submit">
</form>
</body>
</html>
```

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>
```

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

```

</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>

```

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>

```

Example on Image Submit Button:

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

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

<button>

- <button> tag is used to display a submit button with image and text.
- <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.

Example on general button:

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

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

```
<input type="submit">  
</form>  
</body>  
</html>
```

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>
```

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>
```

Example on Maxlength:

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

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>
```

<label>

- <label> tag is used to create field heading.
- 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="TextBox1">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>
```

<select>

- <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 drop downlist.
- Optionally, <optgroup> tag can be used to group-up the <option> tags.
- <select>, <option> and <optgroup> 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>
```

Example on OptGroup:

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

Example on ListBox:

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

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

```
<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.
- <textarea> tag is a paired tag.
- 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>
```

<fieldset>

- <fieldset> tag is used to display a box around a set of fields.
- <fieldset> tag is a paired tag.
- **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.
- 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>
```

<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>
```


- represents a small amount of text, for which you can apply some special formatting.
- is a paired tag.
- **Syntax:**

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

- **Example:**

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

DOCTYPE

<!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>
```

Deprecated Tags

HTML 4 – Deprecated Tags

- The following tags are deprecated (not recommended) in html 4.

1. <marquee>
2.
3. <bgsound>
4. <frameset>
5. <frame>
6. <basefont>
7. <applet>
8. <tt>
9. <center>
10. <u>
11. <acronym>
12. <big>
13. <dir>
14. <strike>

XHTML



Introduction to 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.

HARSHA

XHTML Rules

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 world wide 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"`

Additional XHTML Rules

1. The following tags are only allowed in <body> tag directly. All other tags must be used only inside any of the following tags.
 - o <p>
 - o <table>
 - o <div>
 - o
 - o
 - o <h1> to <h6>
2. “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

Introduction to 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.
- Syntax of CSS:

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

- Syntax of CSS Style Definition:

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

CSS 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.
- 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.
- 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;

color

“color” property

- This property specifies text color (font color) of the element.
- 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

“background-color” property

- This property specifies background color of the element.
- 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>
```

font-family

“font-family” property

- This property specifies name of the font.
- 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

“font-size” property

- This property specifies size of the font.
- Syntax: font-size: pixels;
- Example: font-size: 30px;

Example of “font-size” property

```
<html>
<head>
    <title>CSS - Font-size</title>
    <style type="text/css">
        body
        {
            font-family: 'Tahoma';
        }
        #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

"font-weight" property

- This property applies 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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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

“font-style” property

- This property applies 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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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>
```

letter-spacing

“letter-spacing” property

- This property specifies gap between letters.
- Syntax: letter-spacing: pixels;
- Example: letter-spacing: 10px;

Example of “letter-spacing” property

```
<html>
  <head>
    <title>CSS - Letter-spacing</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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>**

Harsha

word-spacing

“word-spacing” property

- This property specifies gap between 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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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>**

Harsha

Types of colors

Types of colors

- Colors can be represented in 3 formats.
 1. Named colors
 2. RGB
 3. Hex format
- **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 realtime.
- **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
- **Hexa decimal format:**
 - "Hexa decimal format" is the shortcut for "RGB".
 - **Syntax:** `#redgreenblue`

- Example:
 - #ffffff = white
 - #000000 = black
 - #ff0000 = red
- “Hexa decimal 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>
```

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>
```

Hexa decimal 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>
```

line-height

“line-height” property

- This property specifies height of the line of text.
- Syntax: line-height: pixels;
- Example: line-height: 10px;

Example of “line-height” property

```
<html>
  <head>
    <title>CSS - Line-height</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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

“text-decoration” property

- This property specifies underline / overline / strikeout for the text.
- 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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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>


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.



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.



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.



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.


</body>
</html>
```

text-transform

“text-transform” property

- This property specifies uppercase / lowercase / titlecase for the text.
- 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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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>


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.



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.



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.



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.


</body>
</html>
```

text-align

“text-align” property

- This property specifies alignment for the text.
- 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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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>


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.



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.



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.



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.


</body>
</html>
```

text-indent

“text-indent” property

- This property specifies left margin for the first line of the paragraph.
- Syntax: text-indent: pixels;
- Example: text-indent: 10px;

Example of “text-indent” property

```
<html>
  <head>
    <title>CSS - Text-indent</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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>


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.



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.



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.



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.


</body>
</html>
```

background-image

“background-image” property

- This property specifies background image of the element.
- 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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #p1
      {
        background-color: skyblue;
        background-image: url('sample.png');
      }
    </style>
  </head>
  <body>
    <p id="p1">Lorem Ipsum is simply dummy text of the printing and
    typesetting industry. Lorem Ipsum has been the industry's standard
    dummy text ever since the 1500s, when an unknown printer took a galley
    of type and scrambled it to make a type specimen book. It has survived not
    only five centuries, but also the leap into electronic typesetting, remaining
    essentially unchanged. It was popularised in the 1960s with the release of
    Letraset sheets containing Lorem Ipsum passages, and more recently with
    desktop publishing software like Aldus PageMaker including versions of
    Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and
    typesetting industry. Lorem Ipsum has been the industry's standard
    dummy text ever since the 1500s, when an unknown printer took a galley
    of type and scrambled it to make a type specimen book. It has survived not
    only five centuries, but also the leap into electronic typesetting, remaining
  </body>

```

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-repeat

“background-repeat” property

- This property specifies repeat mode of the background image.
- 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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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

“background-position” property

- This property specifies position of the background image.
- 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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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
  </body>
</html>

```

Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. </p>

</body>
</html>

Note: Place “sample.png” in the current folder.

background-attachment

“background-attachment” property

- This property specifies whether the background image should be scrolled along with the text, when the user uses the scrollbars of the web page.
- Syntax: background-attachment: scroll | fixed;
- Example: background-attachment: scroll;
- scroll: “background image” will be scrolled along with the text.
- fixed: “background image” will not be scrolled along with the text.

Example of “background-attachment” property

```

<html>
  <head>
    <title>CSS - Background-attachment</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      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.

list-style-type for

"list-style-type" property for

- This property specifies type of the bullet for the 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">
body
{
    font-family: 'Tahoma';
    font-size: 30px;
}
#list1
{
    list-style-type: disc;
}
#list2
{
    list-style-type: square;
}
#list3
{
    list-style-type: circle;
}
#list4
{
    list-style-type: none;
}
</style>
</head>
<body>

<h2>UL - disc</h2>
```

```
<ul id="list1">
    <li>Google Chrome</li>
    <li>Mozilla Firefox</li>
    <li>Internet Explorer</li>
    <li>Safari</li>
    <li>Opera</li>
</ul>

<h2>UL - square</h2>
<ul id="list2">
    <li>Google Chrome</li>
    <li>Mozilla Firefox</li>
    <li>Internet Explorer</li>
    <li>Safari</li>
    <li>Opera</li>
</ul>

<h2>UL - circle</h2>
<ul id="list3">
    <li>Google Chrome</li>
    <li>Mozilla Firefox</li>
    <li>Internet Explorer</li>
    <li>Safari</li>
    <li>Opera</li>
</ul>

<h2>UL - none</h2>
<ul id="list4">
    <li>Google Chrome</li>
    <li>Mozilla Firefox</li>
    <li>Internet Explorer</li>
    <li>Safari</li>
    <li>Opera</li>
</ul>

</body>
</html>
```

list-style-type for

"list-style-type" property for

- This property specifies type of numbering for the tag.
- 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">
body
{
    font-family: 'Tahoma';
}
#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;
}
```

```
#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>Google Chrome</li>
        <li>Mozilla Firefox</li>
        <li>Internet Explorer</li>
        <li>Safari</li>
        <li>Opera</li>
    </ol>
    <h2>OL - decimal-leading-zero</h2>
    <ol id="list2">
        <li>Google Chrome</li>
        <li>Mozilla Firefox</li>
        <li>Internet Explorer</li>
        <li>Safari</li>
        <li>Opera</li>
    </ol>
    <h2>OL - lower-alpha</h2>
    <ol id="list3">
        <li>Google Chrome</li>
        <li>Mozilla Firefox</li>
        <li>Internet Explorer</li>
        <li>Safari</li>
        <li>Opera</li>
    </ol>
    <h2>OL - upper-alpha</h2>
    <ol id="list4">
        <li>Google Chrome</li>
```

```
<li>Mozilla Firefox</li>
<li>Internet Explorer</li>
<li>Safari</li>
<li>Opera</li>
</ol>
<h2>OL - lower-roman</h2>
<ol id="list5">
<li>Google Chrome</li>
<li>Mozilla Firefox</li>
<li>Internet Explorer</li>
<li>Safari</li>
<li>Opera</li>
</ol>
<h2>OL - upper-roman</h2>
<ol id="list6">
<li>Google Chrome</li>
<li>Mozilla Firefox</li>
<li>Internet Explorer</li>
<li>Safari</li>
<li>Opera</li>
</ol>
<h2>OL - lower-greek</h2>
<ol id="list7">
<li>Google Chrome</li>
<li>Mozilla Firefox</li>
<li>Internet Explorer</li>
<li>Safari</li>
<li>Opera</li>
</ol>
<h2>OL - none</h2>
<ol id="list8">
<li>Google Chrome</li>
<li>Mozilla Firefox</li>
<li>Internet Explorer</li>
<li>Safari</li>
<li>Opera</li>
</ol>
</body>
</html>
```

list-style-image

“list-style-image” property

- This property specifies bullet image file path for the list.
- 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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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> tag

<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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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>
```

HARSHA

“width” and “height” properties

“width” property

- This property specifies horizontal size of the element.
- Syntax: width: *pixels*;
- Example: width: 200px;

“height” property

- This property specifies vertical size of the element.
- Syntax: height: *pixels*;
- Example: height: 100px;

Example of “width” and “height” properties:

```
<html>
<head>
    <title>CSS - Width and Height</title>
    <style type="text/css">
        body
        {
            font-family: 'Tahoma';
            font-size: 24px;
        }
        #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>
```

HARSHA

Float

"float" property

- This property displays the elements side-by-side.
- Syntax: float: left | right;
- Example: float: left;

Example of "float – left":

```
<html>
  <head>
    <title>CSS - Float=Left</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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>
    <div id="div2"></div>
    <div id="div3"></div>
  </body>
</html>
```

```
</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">
    body
    {
      font-family: 'Tahoma';
      font-size: 24px;
    }
    #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>
```

HARSHA

clear

“clear” property

- 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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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>
```

HARSHA

border-style, border-width, border-color

“border-style” property

- This property specifies type of the border, around the element.
- Syntax: border-style: none | solid | dotted | dashed | double | inset | outset | ridge | groove;
- Example: border-style: solid;

“border-width” property

- This property specifies thickness of the border.
- Syntax: border-width: pixels;
- Example: border-width: 5px;

“border-color” property

- This property specifies color of the border.
- Syntax: border-color: any color;
- Example: border-style: red;

Example of “border-style”, “border-width”, “border-color”:

```
<html>
  <head>
    <title>CSS - Border</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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

“border” property

- 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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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">
    div
    {
        font-size: 35px;
        font-family: Tahoma;
    }
    #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

“margin” property

- This property specifies the margin (gap) between element to element, surrounding the element.
- Syntax: margin: pixels;
- Example: margin: 10px;

Example of “margin” property:

```
<html>
  <head>
    <title>CSS - Margin</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
        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
      {
        font-family: 'Tahoma';
        font-size: 24px;
        text-align: justify;
      }
      #div1
      {
        background-color: #00ffcc;
        width: 300px;
        height: 300px;
        float: left;
        border: 5px solid red;
        margin-top: 10px;
        margin-right: 20px;
        margin-bottom: 20px;
        margin-left: 5px;
      }
      #div2
      {
        background-color: #ff3366;
        width: 300px;
        height: 300px;
        float: left;
        border: 5px solid green;
        margin-top: 10px;
        margin-right: 40px;
        margin-bottom: 20px;
        margin-left: 5px;
      }
      #div3
      {
        background-color: #00ccff;
        width: 300px;
        height: 300px;
        float: left;
```

```
border: 5px solid blue;
margin-top: 10px;
margin-right: 20px;
margin-bottom: 20px;
margin-left: 5px;
}

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

“margin” property - 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
      {
        font-family: 'Tahoma';
        font-size: 24px;
        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>
```

padding

“padding” property

- This property specifies the padding (gap) between border and content of the element.
- Syntax: padding: pixels;
- Example: padding: 10px;

Example of “padding” property:

```
<html>
  <head>
    <title>CSS - Padding</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
        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;
      }
    </style>
  </head>
  <body>
    <div id="div1"></div>
    <div id="div2"></div>
  </body>
</html>
```

```
        }
    #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

“padding” property - 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;

HARSHA

Box Model

Box Model

- Every element has border, margin and padding as follows:



Harshvardhan

opacity

“opacity” property

- This property makes the element transparent (background information is visible through the element).
- 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.

position

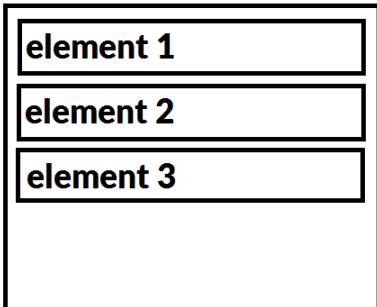
“position” property

- This property specifies position of the element in the web page.
- Syntax: position: static | absolute | relative | fixed;
- Example: position: absolute;

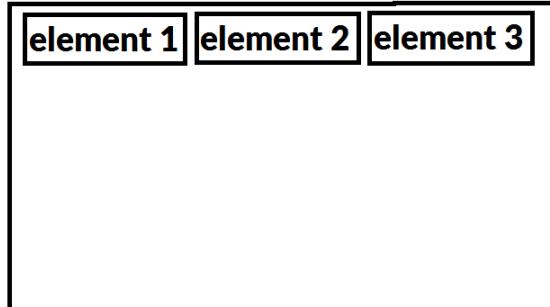
position: static

- 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



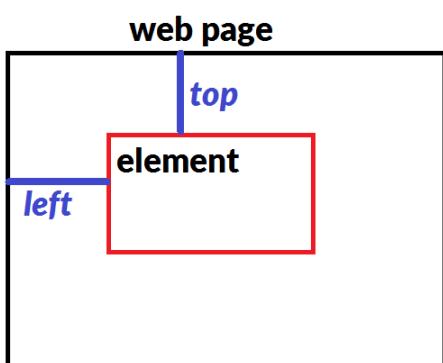
Example of “position:static”:

```
<html>
  <head>
    <title>CSS - position: static</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
    </style>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

```
#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>span 1</span>
    <span>span 2</span>
</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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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.
- Syntax: z-index: n;
- Example: z-index: 1;

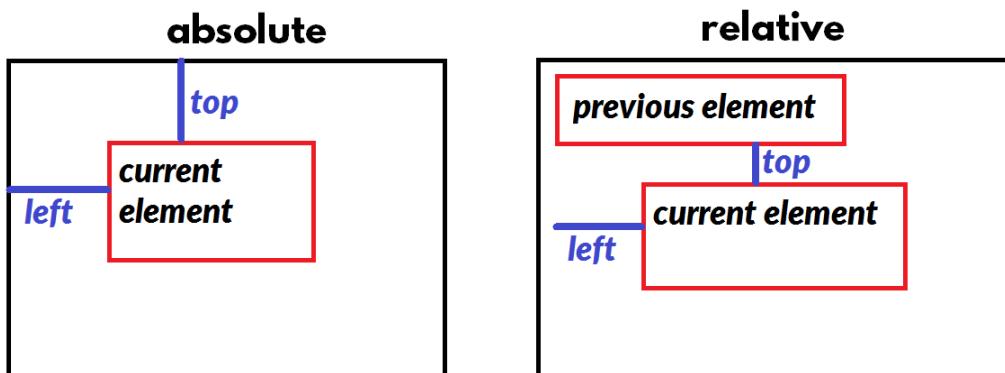
Example of “z-index”:

```
<html>
  <head>
    <title>CSS - z-index</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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: 190px;
      }
    </style>
  </head>
  <body>
    <div id="div1">Div 1</div>
    <div id="div2">Div 2</div>
    <div id="div3">Div 3</div>
  </body>
</html>
```

```
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;



Example of “position:relative”:

```
<html>
  <head>
    <title>CSS - position: relative</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #div1
      {
        background-color: #33ff99;
        height: 400px;
        font-size: 25px;
        margin: 50px;
        padding: 30px;
      }
      #div2
      {
        background-color: #00cccc;
        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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      #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>

display

“display” property

- 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">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
      #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>
```

HARSHA

visibility

“visibility” property

- 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: visible” 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">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
      #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>
```

HARSHA

overflow

“overflow” property

- This property specifies how the text should appear, which is outside the boundaries of the element.
- Syntax: overflow: visible | hidden | auto;
- Example: overflow: auto;
- “overflow: visible” shows the additional content outside the element, which doesn’t fit within the element.
- “overflow: hidden” hides the additional content, which doesn’t fit within the element.
- “overflow: auto” shows scrollbars automatically if necessary.

Example of “overflow:visible” property:

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

Example of “overflow:hidden” property:

```
<html>
  <head>
    <title>CSS - overflow - hidden</title>
    <style type="text/css">
      #div1
      {
        background-color: #00ccff;
        font-size: 25px;
        width: 300px;
        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>
```

```
        }
    </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>
```

HARSHA

Types of Style Sheets

Types of Style Sheets / Types of CSS

- 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">
```

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 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>
```

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

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

Tag Selector

- It selects all the instances of the specified tag.
- 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

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.
- 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

Class Selector

- It selects one or more elements, based on the class name.
- We use same class for similar elements.
- 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

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

Grouping Selector

- It selects the specified group of tags.
- 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

Child Selector

- It selects all the child tags (including grand children) of the specified parent tag.
- 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

Direct Child Selector

- It selects only the direct child tags (excluding grand children) of the specified parent tag.
- 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

Adjacent Siblings Selector

- It selects all the adjacent (next) sibling tags of current tag.
- 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

Adjacent One Sibling Selector

- It selects the immediate next sibling tag of current tag.
- 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>
    <p>para 10</p>
</body>
</html>
```

Attribute Selector

Attribute Selector

- It selects all the tags that are having specified attribute.
- 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

Hover Selector

- It applies the style only when the user places the mouse pointer on the element, at run time.
- 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

Focus Selector

- It applies the style only when the focus (cursor) is inside the element.
- It is also called as “pseudo class”.
- Syntax: tag:focus
- Example: input:focus

Example on Focus Selector:

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

Universal Selector

Universal Selector

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

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">
        body
        {
            font-family: 'Tahoma';
            font-size: 24px;
        }
        /* 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>
```

HARSHA

Last-child selector

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">
        body
        {
            font-family: 'Tahoma';
            font-size: 24px;
        }
        /* 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>
```

HARSHA

Nth-child selector

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">
      body
      {
        font-family: 'Tahoma';
        font-size: 24px;
      }
      /* selects 2nd <p> in <div> */
      /* index starts from 1 */
      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>
```

HARSHA

Nth-child(even) selector

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>
      <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

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

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">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      /* :: 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

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">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      /* :: 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

Selection Selector

- It applies the style for the selected text of the web page.
- It is also called as “pseudo element”.
- 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

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>
```

Table Styles

Example on 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

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;
        }
    </style>
</head>
<body>
    <a href="#">Unvisited Link</a>
    <a href="#">Visited Link</a>
    <a href="#">Active Link</a>
    <a href="#">Hovered Link</a>
</body>
</html>
```

```
}

/* 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

Example on Menubar:

```
<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">
    <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#">About</a></li>
      <li><a href="#">Contact</a></li>
      <li><a href="#">Services</a></li>
    </ul>
  </div>
</body>
</html>
```

HARSHA

Header

Example on Header:

```
<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;
      }
    </style>
  </head>
  <body>
    <div id="header"></div>
    <div id="leftdiv"></div>
    <div id="rightdiv"></div>
    
  </body>
</html>
```

```
    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>
```

```
<div class="part3">
  &nbsp;
</div>
</div>

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

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

Note: Place “fb.png” in the current folder.

Static Page Template

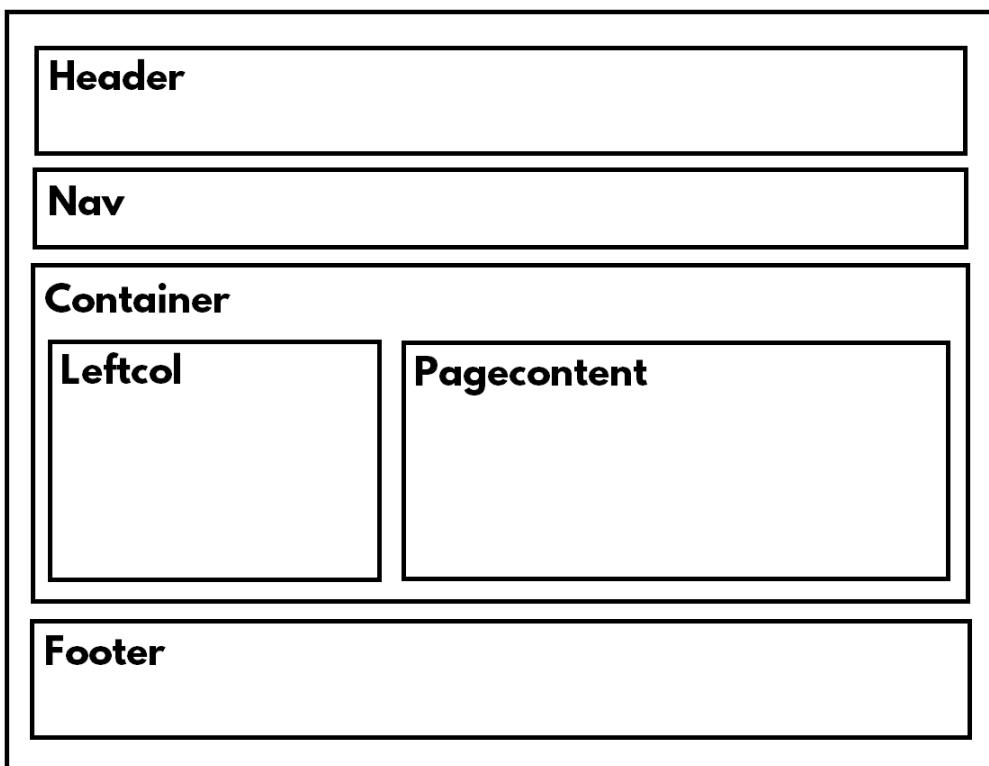
Static Page Template

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>
    <link href="StyleSheet.css" rel="stylesheet">
  </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>
```

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;  
}
```

Example on Page Navigation using Static Page Template:

home.html

```
<html>
  <head>
    <title>Home</title>
    <link href="StyleSheet.css" rel="stylesheet">
  </head>
  <body>
    <div id="outercontainer">
      <div id="header">
        Header
      </div>

      <div id="nav">
        <a href="home.html">Home</a>
        <a href="about.html">About</a>
        <a href="contact.html">Contact</a>
      </div>

      <div id="container">
        <div id="leftcol">
          Leftcol
        </div>
        <div id="pagecontent">
          Home Page content
        </div>
      </div>

      <div id="footer">
        Footer
      </div>
    </div>

    </body>
  </html>
```

about.html

```
<html>
  <head>
    <title>About</title>
    <link href="StyleSheet.css" rel="stylesheet">
  </head>
  <body>
    <div id="outercontainer">
      <div id="header">
        Header
      </div>

      <div id="nav">
        <a href="home.html">Home</a>
        <a href="about.html">About</a>
        <a href="contact.html">Contact</a>
      </div>

      <div id="container">
        <div id="leftcol">
          Leftcol
        </div>
        <div id="pagecontent">
          About Page content
        </div>
      </div>

      <div id="footer">
        Footer
      </div>
    </div>

    </body>
  </html>
```

contact.html

```
<html>
  <head>
    <title>Contact</title>
    <link href="StyleSheet.css" rel="stylesheet">
  </head>
  <body>
    <div id="outercontainer">
      <div id="header">
        Header
      </div>

      <div id="nav">
        <a href="home.html">Home</a>
        <a href="about.html">About</a>
        <a href="contact.html">Contact</a>
      </div>

      <div id="container">
        <div id="leftcol">
          Leftcol
        </div>
        <div id="pagecontent">
          Contact Page content
        </div>
      </div>

      <div id="footer">
        Footer
      </div>
    </div>

    </body>
  </html>
```

StyleSheet.css

```
body
{
    font: 20px 'Tahoma';
}

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

#outercontainer
{
    background-color: #ffccff;
    width: 100%;
    margin: auto;
}

#header
{
    background-color: #00ff99;
    height: 200px;
    width: 100%;
}

#nav
{
    background-color: #ccccff;
    height: 80px;
    width: 100%;
}

#container
{
    height: 500px;
    width: 100%;
}

#leftcol
{
```

```
height: 500px;  
background-color: #66ccff;  
width: 20%;  
float: left;  
}  
  
#pagecontent  
{  
height: 500px;  
background-color: #ccffcc;  
width: 80%;  
float: left;  
}  
  
#footer  
{  
height: 50px;  
background-color: #ff0099;  
width: 100%;  
clear: left;  
}
```

Responsive Web Design

Responsive Web Design

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

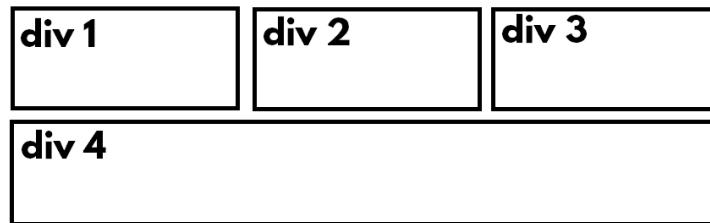
- This tag tells to the mobile browsers that we are using “responsive web design”, and don’t treat it as pc-based web page.
- Without viewport meta tag, the mobile browsers treat the web page as pc-based web page and apply the pc-based media query.
- With viewport meta tag, the mobile browsers apply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

Responsive Web Design - Example

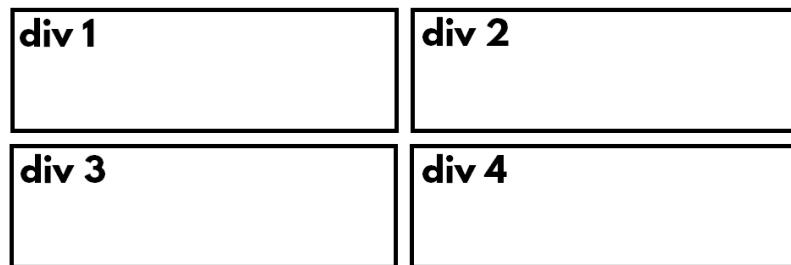
**Large
Devices**



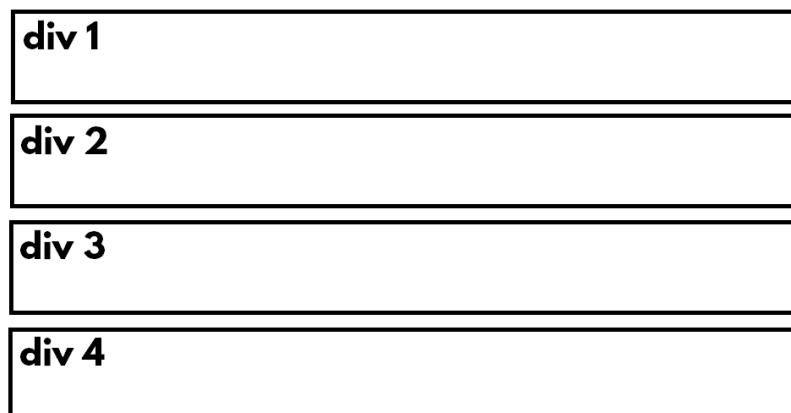
**Medium
Devices**



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

```
<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;
}
```

JavaScript

Introduction to JavaScript

Introduction to JavaScript

- JavaScript is a programming language, which is used to create functionality in the web page.
- JavaScript is client-side (browser-side) language. That means it executes on the browser.
- 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.

Syntax of JavaScript

```
<script>  
    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.

document and document.write()

- “`document`” is a keyword, which represents “current web page”.
- “`write`” is a method (function) that displays a value in the web page.
- Syntax: `document.write(value);`

JavaScript – First Example

```
<html>
  <head>
    <title>JavaScript - First Example</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>Message from html</h1>
    <script>
      document.write("<h1>Message from javascript</h1>");
    </script>
    <h1>Message from html</h1>
  </body>
</html>
```

Variables

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

Data Types

- “Data type” specifies “type of the value”.
- JavaScript supports 6 data types:
 1. Number Ex: 10, 10.82
 2. String Ex: ‘abc’, “abc”
 3. Boolean Ex: true, false
 4. Undefined Ex: undefined
 5. Function Ex: function() {}
 6. Object Ex: JSON objects {}

Example on Variables and Data Types

```
<html>
  <head>
    <title>JavaScript - variables</title>
  </head>
  <body>
    <h1>JavaScript - variables</h1>

    <script>
      var a = 10; //integer type (or) numeric
      var b = 67.7876; //floating-point
      var c = "Hyderabad"; //string
      var d = 'Hyderabad'; //string
      var e = true; //boolean
      var f = false; //boolean
      var g; //undefined
      var h = null; //null (empty)

      document.write(a); //10
      document.write("<br>");
      document.write(b); //67.7876
      document.write("<br>");
      document.write(c); //Hyderabad
      document.write("<br>");
      document.write(d); //Hyderabad
      document.write("<br>");
      document.write(e); //true
      document.write("<br>");
      document.write(f); //false
      document.write("<br>");
      document.write(g); //undefined
      document.write("<br>");
      document.write(h); //null

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

Arrays

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.

array	
0	element 1
1	element 2
2	element 3
3	element 4

Steps for development of arrays

- Create an array: var variablename = [value1, value2, ...];
- Set value into the array element: variablename[index] = value;
- Get value from the array element: variablename[index]
- Get size of the array: variablename.length
- Add new element into the array: variablename.push(element);
- Remove an existing element: variablename.splice(index, no. of elements to remove);

Example on Arrays

```
<html>
<head>
  <title>JavaScript - arrays</title>
  <style type="text/css">
    body
    {
      font-family: 'Tahoma';
      font-size: 30px;
```

```
        }
    </style>
</head>
<body>
<h1>JavaScript - arrays</h1>
<script>
    var n = [ 10, 20, 50, 150, 220 ];
    document.write(n[0]);
    document.write("<br>");
    document.write(n[1]);
    document.write("<br>");
    document.write(n[2]);
    document.write("<br>");
    document.write(n[3]);
    document.write("<br>");
    document.write(n[4]);
</script>
</body>
</html>
```

Operators

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

1. Arithmetical Operators

Arithmetical Operators

- + Addition
- Subtraction
- * Multiplication
- / Division
- % Remainder

Example on Arithmetical Operators

```
<html>
  <head>
    <title>JavaScript - arithmetical operators</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - 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
      document.write(a); //10
      document.write("<br>");
      document.write(b); //3
      document.write("<br>");
      document.write(c); //13
      document.write("<br>");
      document.write(d); //7
    </script>
  </body>
</html>
```

```
document.write("<br>");  
document.write(e); //30  
document.write("<br>");  
document.write(f); //3.333333333333335  
document.write("<br>");  
document.write(g); //1  
</script>  
</body>  
</html>
```

HARSHA

2. Assignment Operators

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>JavaScript - assignment operators</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - assignment operators</h1>
    <script>
      var a;
      a = 100; //assignment operator
      document.write(a); //100
      document.write("<br>");

      var b;
      b = a; //assignment operator
      document.write(b); //100
      document.write("<br>");

      a += 10;
      document.write(a); //110
    </script>
  </body>
</html>
```

```
document.write("<br>");

a -= 10;
document.write(a); //100
document.write("<br>");

a *= 10;
document.write(a); //1000
document.write("<br>");

a /= 10;
document.write(a); //100
document.write("<br>");

a %= 30;
document.write(a); //10
</script>
</body>
</html>
```

3. Increment and Decrement Operators

Increment and Decrement Operators

++ Increment ($+=1$)
-- Decrement ($-=1$)

Example on Increment and Decrement Operators

```
<html>
  <head>
    <title>JavaScript - increment and decrement operators</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - increment and decrement operators</h1>
    <script>
      var a = 10;
      document.write(a); //10
      document.write("<br>");

      a++; //increment operator
      document.write(a); //11
      document.write("<br>");

      a--; //decrement operator
      document.write(a); //10
    </script>
  </body>
</html>
```

4. Relational Operators

Relational Operators

<code>==</code>	Equal to
<code>!=</code>	Not Equal to
<code><</code>	Less than
<code>></code>	Greater than
<code><=</code>	Less than or equal to
<code>>=</code>	Greater than or equal to

Example on Relational Operators

```
<html>
  <head>
    <title>JavaScript - relational operators</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - relational operators</h1>

    <script>
      var x = 100;
      var y = 200;
      var temp1, temp2, temp3, temp4, temp5, temp6;

      temp1 = (x == y);
      document.write(temp1); //false
      document.write("<br>");

      temp2 = (x != y);
      document.write(temp2); //true
      document.write("<br>");
```

```
temp3 = (x < y);
document.write(temp3); //true
document.write("<br>");

temp4 = (x <= y);
document.write(temp4); //true
document.write("<br>");

temp5 = (x > y);
document.write(temp5); //false
document.write("<br>");

temp6 = (x >= y);
document.write(temp6); //false
</script>
</body>
</html>
```

5. Logical Operators

Logical Operators

- && And (both conditions should be true)
- || Or (Atleast any one condition should be true)
- ! Not (given condition will be reverse)

Example on Logical Operators

```
<html>
  <head>
    <title>JavaScript - logical operators</title>
  </head>
  <body>
    <h1>JavaScript - logical operators</h1>
    <script>
      var x = 100;
      var y = 200;
      var z = 50;

      var temp1 = ( (x<y) && (x>z) );
      document.write(temp1); //true
      document.write("<br>");

      var temp2 = ( (x<y) || (x<z) );
      document.write(temp2); //true
      document.write("<br>");

      var temp3 = ( !(x<y) );
      document.write(temp3); //false
    </script>
  </body>
</html>
```

6. Concatenation Operator

Concatenation Operator

+ Attaches two strings and returns a single string.

Ex: "peers" + "tech" = "peerstech"

Number + Number = addition

String + String = concatenation

String + Number = concatenation

Number + String = concatenation

Example on Concatenation Operator

```
<html>
  <head>
    <title>JavaScript - concatenation operator</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - concatenation operator</h1>
    <script>
      var s1 = "peers";
      var s2 = "tech";
      var s3;
      s3 = s1 + s2; //string + string
      document.write(s3); //peerstech
    </script>
  </body>
</html>
```

Control Statements

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

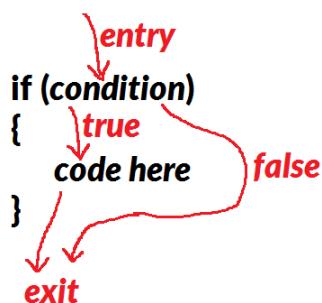
1) if

if

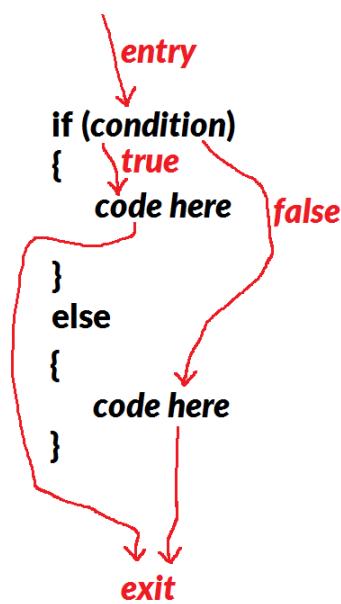
- "If" statement is used to check a condition, and execute the code only if the condition is TRUE.
- Types of "if"
 1. If
 2. If-else
 3. Else-if
 4. Nested if



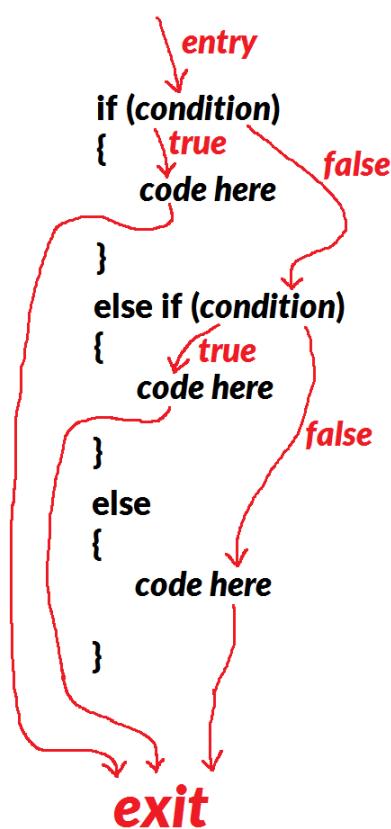
A) if



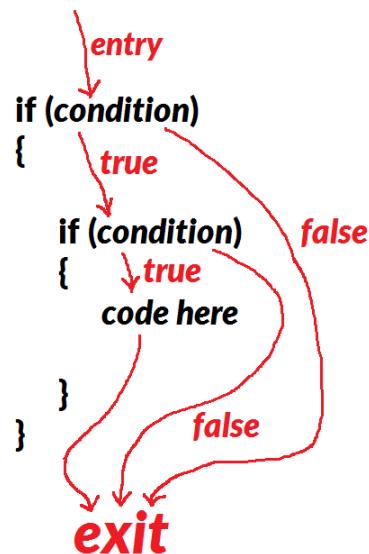
B) if-else



C) else-if



D) Nested if



Example of "if":

```

<html>
  <head>
    <title>JavaScript - if</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - if</h1>
    <script>
      var n = 10;
      if (n == 10)
      {
        document.write("<h3>n is equal to 10</h3>");
      }
      if (n != 10)
    </script>
  </body>
</html>
  
```

```
{  
    document.write("<h3>n is not equal to 10</h3>");  
}  
</script>  
</body>  
</html>
```

Example of "if-else":

```
<html>  
  <head>  
    <title>JavaScript - if-else</title>  
    <style type="text/css">  
      body  
      {  
        font-family: 'Tahoma';  
        font-size: 30px;  
      }  
    </style>  
  </head>  
  <body>  
    <h1>JavaScript - if-else</h1>  
    <script>  
      var n = 10;  
      if (n == 10)  
      {  
        document.write("<h3>n is equal to 10</h3>");  
      }  
      else  
      {  
        document.write("<h3>n is not equal to 10</h3>");  
      }  
    </script>  
  </body>  
</html>
```

Example of "else-if":

```
<html>
  <head>
    <title>JavaScript - else-if</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - else-if</h1>
    <script>
      var a = 10, b = 20;
      if (a < b)
      {
        document.write("<h3>a is less than b</h3>");
      }
      else if (a > b)
      {
        document.write("<h3>a is greater than b</h3>");
      }
      else
      {
        document.write("<h3>a and b are equal</h3>");
      }
    </script>
  </body>
</html>
```

Example of “nested if”:

```
<html>
  <head>
    <title>JavaScript - Nested if</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - Nested if</h1>
    <script>
      var a = 10, b = 20;
      if (a != b)
      {
        if (a > b)
        {
          document.write("<h3>a is greater than b</h3>");
        }
        else
        {
          document.write("<h3>a is less than b</h3>");
        }
      }
      else
      {
        document.write("<h3>a and b are equal</h3>");
      }
    </script>
  </body>
</html>
```

2) Switch-case

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 value1: code here; break;
    ...
    case value1: code here; break;
}
```

Example on Switch-case:

```
<html>
<head>
    <title>JavaScript - switch-case</title>
    <style type="text/css">
        body
        {
            font-family: 'Tahoma';
            font-size: 30px;
        }
    </style>
</head>
<body>
    <h1>JavaScript - 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;
        }
    </script>

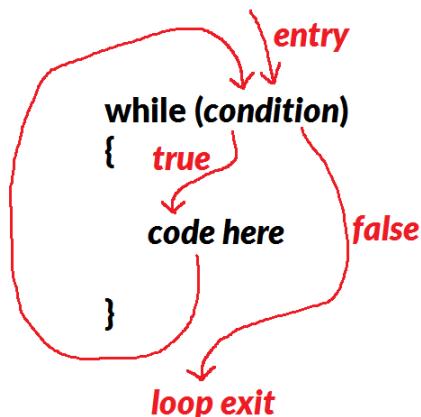
```

```
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;
}
document.write(monthname);
</script>
</body>
</html>
```

3) While

While

- “While” statement is used to execute the code repeatedly, while the condition is TRUE.



Example on While:

```

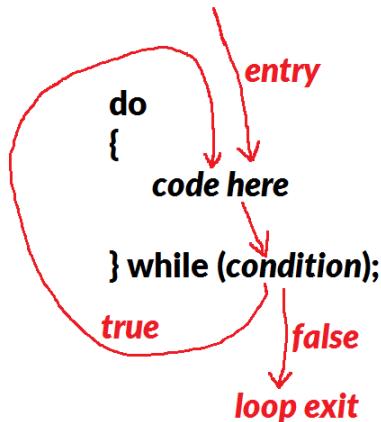
<html>
  <head>
    <title>JavaScipt - while</title>
  </head>
  <body>
    <h1>JavaScipt - while</h1>
    <script>
      var i = 1;
      while (i <= 10)
      {
        document.write(i);
        document.write("<br>");
        i++;
      }
    </script>
  </body>
</html>

```

4) Do-While

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:

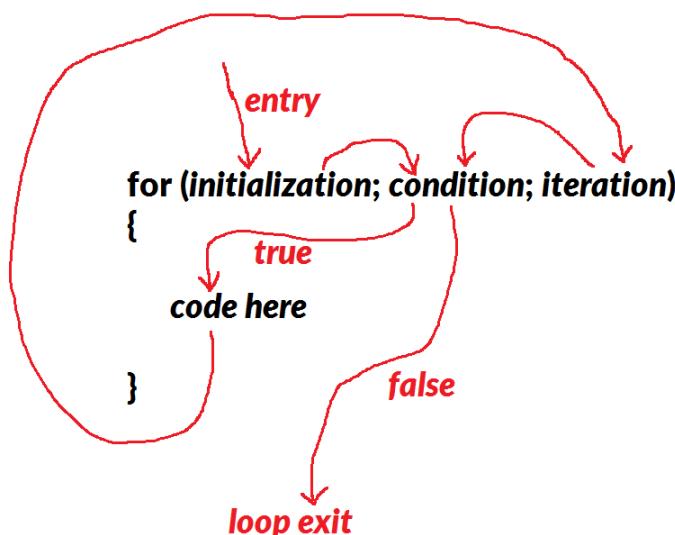
```

<html>
  <head>
    <title>JavaScript - do-while</title>
  </head>
  <body>
    <h1>JavaScript - do-while</h1>
    <script>
      var i = 1;
      do
      {
        document.write(i);
        document.write("<br>");
        i++;
      }while(i <= 10);
    </script>
  </body>
</html>
    
```

5) For

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>JavaScript - for</title>
  </head>
  <body>
    <h1>JavaScript - for</h1>
    <script>
      var i;
      for (i=1; i<=10; i++)
      {
        document.write(i);
        document.write("<br>");
      }
    </script>
  </body>
</html>
  
```

Example 2 on For:

```
<html>
  <head>
    <title>JavaScript - for</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - for</h1>

    <script>
      var i;
      document.write("<h3>loop starts</h3>");
      for (i=1; i <= 100; i++)
      {
        if (i % 2 == 0)
        {
          document.write("<span style='color:green'>");
          document.write(i);
          document.write(" </span>");
        }
        else
        {
          document.write("<span style='color:red'>");
          document.write(i);
          document.write(" </span>");
        }
      }
      document.write("<h3>loop finished</h3>");
    </script>
  </body>
</html>
```

Break

Break

- It is used to stop (terminate) the loop.

```
for (initialization; condition; iteration)
{
    break;
}

}
```

exit from the loop

Example on Break:

```
<html>
  <head>
    <title>JavaScript - break</title>
  </head>
  <body>
    <h1>JavaScript - break</h1>

    <script>
      var i;
      for(i=1; i<=10; i++)
      {
        document.write(i + "<br>");

        if (i == 7)
        {
          break; //stop loop
        }
      }
    </script>
  </body>
</html>
```

Continue

Continue

- It is used to “skip the current iteration” and go to the “next iteration”.

```
for (initialization; condition; iteration)  
{
```

```
    continue;
```

```
}
```

Example on Continue:

```
<html>  
  <head>  
    <title>JavaScript - continue</title>  
  </head>  
  <body>  
    <h1>JavaScript - continue</h1>  
    <script>  
      var i;  
      for(i=1; i<=10; i++)  
      {  
        if (i == 7)  
        {  
          continue; //skip the number and goto next number  
        }  
        document.write(i + "<br>");  
      }  
    </script>  
  </body>  
</html>
```

Noscript

Noscript

- It is used to display a message to the user, when the user has disabled JavaScript in the browser.
- Syntax: <noscript> *any message* </noscript>
- Example: <noscript> Please enable JavaScript </noscript>

Example on Noscript:

```
<html>
  <head>
    <title>JavaScript - noscript</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <script>
      document.write("<h1>Message from javascript</h1>");
    </script>

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

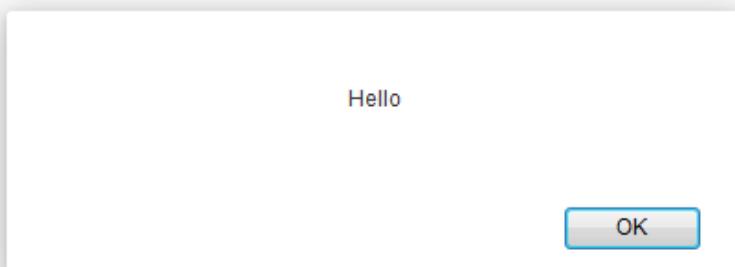
Popup boxes

Popup boxes

- Popup box / dialog box is a box in the browser, that shows a small message to the user.
- JavaScript supports two types of popup boxes:
 1. `alert()`
 2. `confirm()`

1. `alert()`

- This function is used to display an information message to the user.
- It contains OK button only.
- Syntax: `alert("message");`
- Example: `alert("Hello");`

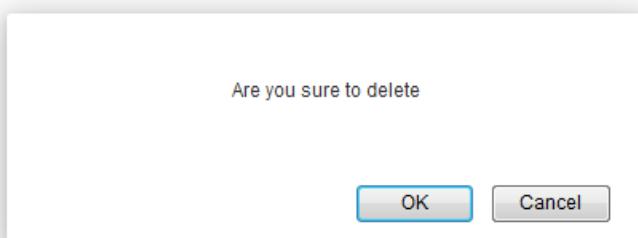


Example on alert:

```
<html>
  <head>
    <title>JavaScript - alert</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>alert</h1>
    <script>
      alert("Hello");
    </script>
  </body>
</html>
```

2. confirm()

- This function is used to display a confirmation dialogbox to the user with OK and Cancel buttons.
- The confirm() function returns “true”, if the user clicks on “OK”.
- The confirm() function returns “false”, if the user clicks on “Cancel”.
- Syntax: confirm(“message”);
- Example: confirm(“Are you sure to delete”);



Example on confirm:

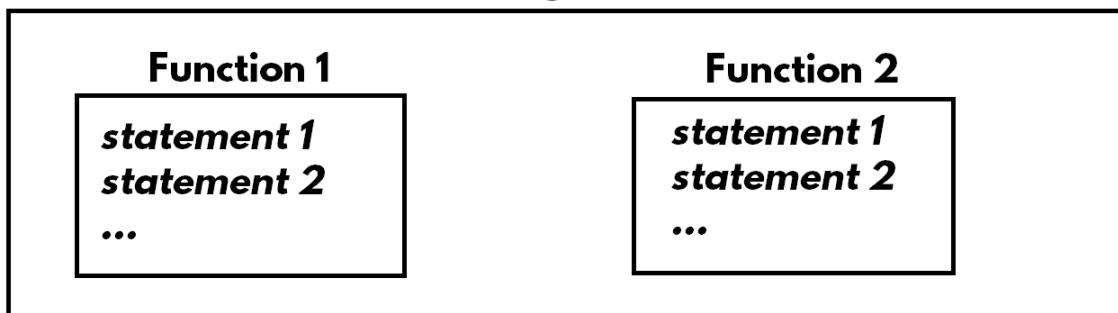
```
<html>
  <head>
    <title>JavaScript - confirm</title>
    <style type="text/css">
      body
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>confirm</h1>
    <script>
      var result = confirm("Are you sure to delete");
      document.write(result);
    </script>
  </body>
</html>
```

Functions

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. Everytme 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(arguments)
{
    code here
}
```

Arguments: 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, ...);
```

Example 1 on Functions:

```
<html>
<head>
    <title>JavaScript - functions</title>
    <style type="text/css">
        body
        {
            font-family: 'Tahoma';
            font-size: 30px;
        }
    </style>
</head>
<body>
    <h1>JavaScript - functions</h1>
    <script>
        function country()
        {
            document.write("<h3>India</h3>");
        }
        country();
        country();
        country();
    </script>
</body>
</html>
```

Example 2 on Functions:

```
<html>
<head>
    <title>JavaScript - functions</title>
    <style type="text/css">
        body
        {
            font-family: 'Tahoma';
            font-size: 30px;
        }
    </style>
</head>
<body>
```

```
        }
    </style>
</head>
<body>
    <h1>JavaScript - functions</h1>
    <script>
        function country()
        {
            document.write("<h2>India</h2>");
        }
        function city()
        {
            document.write("<h2>Hyderabad</h2>");
            country();
        }
        country();
        city();
    </script>
</body>
</html>
```

Example 3 on Functions:

```
<html>
    <head>
        <title>JavaScript - functions</title>
        <style type="text/css">
            body
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
        </style>
    </head>
    <body>
        <h1>JavaScript - functions</h1>
        <script>
            function add(a,b)
            {
                var c; //local variable
                c = a+b;
```

```
        return(c);
    }
var result;
result = add(10, 20);
document.write(result); //30
document.write("<br>");
var x = 100;
var y = 250;
var result2 = add(x, y);
document.write(result2); //350
</script>
</body>
</html>
```

HARSHA

DOM

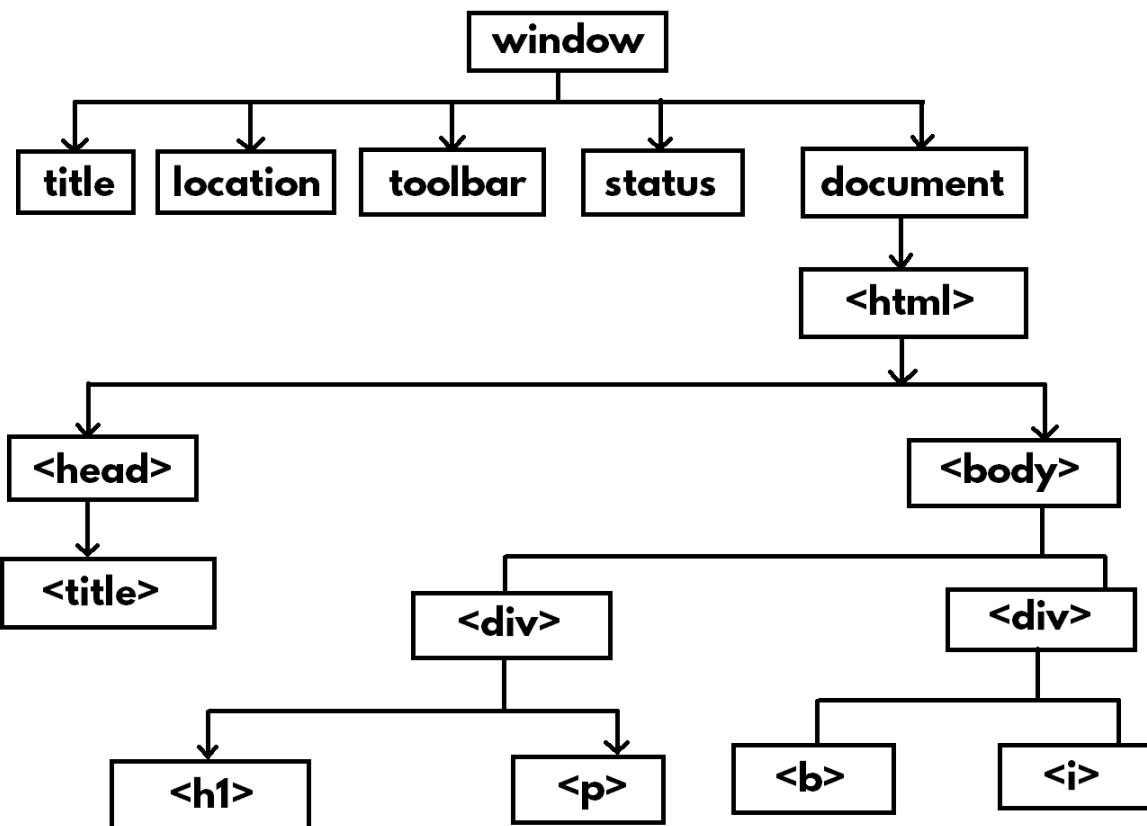
DOM

- 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.

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>
```

Example:



Types of DOM manipulations

- I. Modifying existing elements in DOM
- II. Adding new elements to DOM
- III. Removing existing elements from DOM

I) Modifying Existing Elements in DOM

1. Getting Existing Elements from DOM
2. Changing Content of Existing Elements of DOM
3. Changing Attributes of Existing Elements of DOM
4. Changing CSS Styles of existing elements of DOM
5. Adding Events to Existing Elements of DOM.

Getting Existing Elements from DOM and Changing Content

1. Getting Existing Elements from DOM

- We can get existing elements from DOM and we can manipulate them, using JavaScript.

Types of Getting Existing Elements from DOM

- A. Getting Elements Based on ID
- B. Getting Elements Based on Name
- C. Getting Elements Based on Tag
- D. Getting Elements Based on Class

A) Getting Elements Based on ID

- Syntax: `document.getElementById("id here")`
- Example: `document.getElementById("div1")`
- This function returns the single matching element.
- ID can't be duplicate.

B) Getting Elements Based on Name

- Syntax: `document.getElementByName("name")`
- Example: `document.getElementByName("abc")`
- This function returns the elements that have the specified name, as an array.
- "Name" can be duplicate.

C) Getting Elements Based on Tag Name

- Syntax: `document.getElementByTagName("tag name")`
- Example: `document.getElementByTagName("p")`
- This function returns all instances of the specified tag, as an array.

D) Getting Elements Based on Class Name

- Syntax: `document.getElementByClassName("class name")`
- Example: `document.getElementByClassName("c1")`
- This function returns all instances that have specified class, as an array.

2. Changing Content of Existing Elements of DOM

- Syntax: `innerHTML`
- Example: `document.getElementById("p1").innerHTML`
- The "innerHTML" property represents the content of the tag, which is present between opening tag and closing tag. We can set / get the value into the inner html.
`<tag> inner html </tag>`

Example on Getting Elements Based on ID:

```
<html>
  <head>
    <title>JavaScript - Getting element by id</title>
    <style type="text/css">
      body
      {
        font: 30px tahoma;
      }
      #div1
      {
        background-color: green;
      }
    </style>
  </head>
  <body>
    <div id="div1">Hello</div>
    <script>
      document.getElementById("div1").innerHTML = "Hai";
    </script>
  </body>
</html>
```

Example 1 on Getting Elements Based on Name:

```
<html>
  <head>
    <title>JavaScript - getElementsByName</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - getElementsByName</h1>
    <div name="div1">Hai</div>
    <div name="div1">Hai</div>
```

```
<div name="div1">Hai</div>
<div name="div1">Hai</div>

<script>
    document.getElementsByName("div1")[1].innerHTML = "Hello";
</script>
</body>
</html>
```

Example 2 on Getting Elements Based on Name:

```
<html>
    <head>
        <title>JavaScript - getElementsByName 2</title>
        <style type="text/css">
            body,input
            {
                font-family: Tahoma;
                font-size: 30px;
            }
        </style>
    </head>
    <body>
        <h1>JavaScript - getElementsByName 2</h1>
        <div name="sample">Hai</div>
        <div name="sample">Hai</div>
        <div name="sample">Hai</div>
        <div name="sample">Hai</div>
        <div name="sample">Hai</div>

        <script>
            var d = document.getElementsByName("sample");
            for (var i = 0; i < d.length; i++)
            {
                d[i].innerHTML = "Hello";
            }
        </script>
    </body>
</html>
```

Example 1 on Getting Elements Based on Tag Name:

```
<html>
  <head>
    <title>JavaScript - getElementsByTagName</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - getElementsByTagName</h1>
    <div>Hello</div>
    <div>Hello</div>
    <script>
      document.getElementsByTagName("div")[1].innerHTML = "Hai";
    </script>
  </body>
</html>
```

Example 2 on Getting Elements Based on Tag Name:

```
<html>
  <head>
    <title>JavaScript - Getting elements by tag name</title>
    <style type="text/css">
      body
      {
        font: 30px tahoma;
      }
    </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>

<script>
  var data = document.getElementsByTagName("p");
  for (var i=0; i < 6; i++)
  {
    data[i].innerHTML = "Hai";
  }
</script>
</body>
</html>
```

Example 1 on Getting Elements Based on Class Name:

```
<html>
  <head>
    <title>JavaScript - getElementsByClassName</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - getElementsByClassName</h1>
    <div class="c1">Hello</div>
    <div class="c1">Hello</div>
    <script>
      document.getElementsByClassName("c1")[1].innerHTML = "Hai";
    </script>
  </body>
</html>
```

Example 2 on Getting Elements Based on Class Name:

```
<html>
  <head>
    <title>JavaScript - Getting elements by class name</title>
    <style type="text/css">
      body
      {
        font: 30px tahoma;
      }
    </style>
  </head>
  <body>
    <p>para 1</p>
    <p class="c1">para 2</p>
    <p class="c1">para 3</p>
    <p>para 4</p>
    <p class="c1">para 5</p>
    <p>para 6</p>

    <script>
      var data = document.getElementsByClassName("c1");
      for (var i=0; i < data.length; i++)
      {
        data[i].innerHTML = "Hai";
      }
    </script>
  </body>
</html>
```

Manipulating Attributes of Existing Elements of DOM

3. Manipulating Attributes of Existing Elements of DOM

- Attributes are details about the tag.
- By using the following JavaScript functions, we can manipulate the attributes dynamically at run time.
 - A. Setting an attribute value
 - B. Getting the value of an existing attribute
 - C. Removing an existing attribute

A. Setting an attribute value:

- Syntax: `setAttribute("attribute name", "value here");`
- Example: `setAttribute("src", "img2.jpg");`
- This function sets (updates / changes) an attribute of an existing element of DOM.

B. Getting the value of an existing attribute:

- Syntax: `getAttribute("attribute name");`
- Example: `getAttribute("src");`
- This function gets (retrieves) the value of an attribute of an existing element of DOM.

C. Removing an existing attribute:

- Syntax: `removeAttribute("attribute name");`
- Example: `removeAttribute("src");`
- This function removes (deletes) an attribute of an existing element of DOM.

Example on SetAttribute:

```
<html>
  <head>
    <title>JavaScript - setAttribute</title>
  </head>
  <body>
    

    <script>
      document.getElementById("myimage").setAttribute("src",
"img2.jpg");
    </script>
  </body>
</html>
```

Note: Place "img1.jpg", "img2.jpg" in the current folder.

Example on GetAttribute:

```
<html>
  <head>
    <title>JavaScript - getAttribute</title>
  </head>
  <body>
    

    <script>
      var a = document.getElementById("myimage").getAttribute("src");
      alert(a);
    </script>
  </body>
</html>
```

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

Example on RemoveAttribute:

```
<html>
  <head>
    <title>JavaScript - removeAttribute</title>
    <style type="text/css">
      body,input
      {
        font: 30px tahoma;
      }
    </style>
  </head>
  <body>
    
    <script>
      document.getElementById("myimage").removeAttribute("title");
    </script>
  </body>
</html>
```

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

Manipulating CSS of Existing Elements of DOM

4. Manipulating CSS of Existing Elements of DOM

- Syntax: style.propertyname
- Example: style.backgroundColor
- The “style” property is used to set / get the value of css property.

Example on Manipulating CSS of Existing Elements of DOM

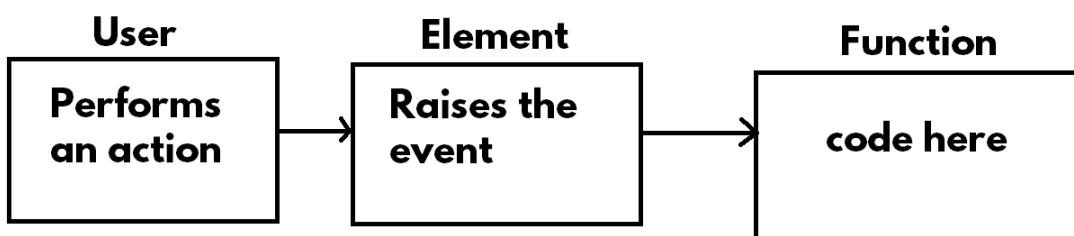
```
<html>
  <head>
    <title>JavaScript - Manipulating CSS</title>
  </head>
  <body>
    <div id="div1">div 1</div>

    <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";
      alert(document.getElementById("div1").style.fontFamily);
    </script>
  </body>
</html>
```

Event Handling

Event Handling

- “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

“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>JavaScript - click</title>  
    <style type="text/css">  
      #div1  
      {  
        background-color: hotpink;  
        width: 200px;  
        height: 200px;  
        border: 4px solid red;  
      }  
    </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

“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>JavaScript - dblclick</title>
    <style type="text/css">
        #div1
        {
            background-color: hotpink;
            width: 300px;
            height: 200px;
            border: 4px solid red;
        }
    </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>JavaScript - mouseover, mouseout</title>
    <style type="text/css">
        body
        {
            font-family: 'Tahoma';
            font-size: 30px;
        }
        #div1
        {
            width: 400px;
            height: 200px;
            background-color: lightgreen;
        }
    </style>

```

```
</style>
</head>
<body>
    <h1>JavaScript Events - 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>
```

Example on Image with “Mouseover” and “Mouseout” events:

```
<html>
    <head>
        <title>JavaScript - Image with hover</title>
        <style type="text/css">
            body
            {
                font-family: 'Tahoma';
                font-size: 30px;
            }
            #myimage
            {
                cursor: pointer;
            }
        </style>
    </head>
    <body>
        
    </body>
</html>
```

```
</style>
</head>
<body>
    <h1>Image with hover</h1>
    

    <script>

        document.getElementById("myimage").addEventListener("mouseover",
fun1);
        function fun1()
        {
            document.getElementById("myimage").setAttribute("src",
"tick2.jpg");
        }

        document.getElementById("myimage").addEventListener("mouseout",
fun2);
        function fun2()
        {
            document.getElementById("myimage").setAttribute("src",
"tick1.jpg");
        }
    </script>
</body>
</html>
```

Note: Place "tick1.jpg", "tick2.jpg" in the current folder.

“Mousemove” event

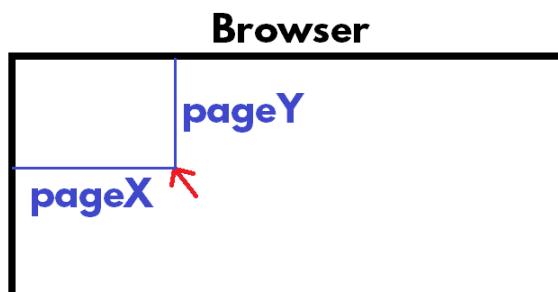
“Mousemove” event

- The “mousemove” event executes when the user moves the mouse pointer across the element.

- Syntax:

```
addEventListener("mousemove", functionname);
function functionname()
{
}
```

- **event:** Represents the information, given by the browser. Whenever the user performs an action, the browser collects some information, and it passes the same information to the function automatically. This is called “event”.
- **event.pageX:** Represents “X” co-ordinate of mouse pointer position.
- **event.pageY:** Represents “Y” co-ordinate of mouse pointer position.



Example on “Mousemove” event:

```
<html>
<head>
  <title>JavaScript - mousemove</title>
  <style type="text/css">
    body
    {
      font-family: 'Tahoma';
      font-size: 26px;
```

```
        }
    #div1
    {
        width: 400px;
        height: 200px;
        background-color: lightgreen;
    }
</style>
</head>
<body>
    <h1>JavaScript Events - 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;
            document.getElementById("div1").innerHTML = x + ", " + y;
        }
    </script>
</body>
</html>
```

“Keyup” event

“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>JavaScript - Keyup</title>
  </head>
  <body>
    <h1>Keyup</h1>
    Source Text:
    <input type="text" id="txt1">
    <br>
    Destination Text:
    <input type="text" id="txt2">
    <script>
      document.getElementById("txt1").addEventListener("keyup",
fun1);
      function fun1()
      {
        document.getElementById("txt2").value =
document.getElementById("txt1").value;
      }
    </script>
  </body>
</html>
```

“Keypress” event

“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 = Americal 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>JavaScript - Keypress</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>Keypress</h1>
    Source Text:
    <input type="text" id="txt1">
    <br>
    Destination Text:
    <input type="text" id="txt2">

    <script>
      document.getElementById("txt1").addEventListener("keypress",
fun1);
      function fun1()
      {
        document.getElementById("txt2").value =
document.getElementById("txt1").value;
      }
    </script>
  </body>
</html>
```

Example on “Keypress” event – Alphabets only:

```

<html>
  <head>
    <title>JavaScript - Alphabets only</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>Alphabets only</h1>
    <input type="text" id="txt1">

    <script>
      document.getElementById("txt1").addEventListener("keypress",
fun1);
      function fun1(event)
      {
        //event means "browser given details"

        //ascii value of currently pressed character
        var ch = event.which;
        //alert(ch);

        if ( !((ch >=65 && ch <=90) ||(ch >=97 && ch <=122) || (ch == 32) || (ch
== 8) || (ch == 0)))
        {
          event.preventDefault(); //cancel the currently pressed
          character
        }
      }
    </script>
  </body>
</html>

```

Example on “Keypress” event – Numbers only:

```

<html>
  <head>
    <title>JavaScript - Numbers only</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>Numbers only</h1>
    <input type="text" id="txt1">
    <script>
      document.getElementById("txt1").addEventListener("keypress",
fun1);
      function fun1(event)
      {
        //event means "browser given details"
        //ascii value of currently pressed character
        var ch = event.which;
        //alert(ch);

        if ( !((ch >=48 && ch <=57) || (ch == 8) || (ch == 0)))
        {
          event.preventDefault(); //cancel the currently pressed
character
        }
      }
    </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>JavaScript - focus and blur</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </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

“Change” event

- The “change” event executes when the value of the element has been changed.
- That means it executes in following cases:
 1. When the user modifies the value of textbox and presses TAB key.
 2. When the user checks / unchecks the checkbox.
 3. When the user selects the radio button.
 4. When the user selects an item in the dropdownlist.

- Syntax:

```
addEventListener("change", functionname);
function functionname()
{
}
```

Example on “Change” event with TextBox:

```
<html>
<head>
    <title>JavaScript - Change - TextBox</title>
    <style type="text/css">
        body,input
        {
            font-family: 'Tahoma';
            font-size: 30px;
        }
    </style>
</head>
<body>
    <h1>Change</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>JavaScript - Change - CheckBox</title>
    <style type="text/css">
        body,input
        {
            font-family: 'Tahoma';
            font-size: 30px;
        }
    </style>
</head>
<body>
    <h1>JavaScript - 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 = "";
            }
        }
    </script>
</body>
</html>
```

```
        else
        {
            document.getElementById("btn1").disabled = "disabled";
        }
    }
</script>
</body>
</html>
```

Example on “Change” event with RadioButton:

```
<html>
<head>
    <title>JavaScript - Change - RadioButton</title>
    <style type="text/css">
        body,input
        {
            font-family: 'Tahoma';
            font-size: 30px;
        }
        #div1
        {
            color: darkblue;
        }
    </style>
</head>
<body>
    <h1>JavaScript -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>
</body>
```

```

<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>JavaScript - Change - DropDownList</title>
    <style type="text/css">
        body,input,select
        {
            font-family: 'Tahoma';
            font-size: 30px;
        }
    </style>
</head>
<body>
    <h1>JavaScript - 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>
```

Example 2 on “Change” event with DropDownList:

```
<html>
<head>
    <title>JavaScript - Change - Dropdownlist - background color</title>
    <style type="text/css">
        body,input,select
        {
            font-family: 'Tahoma';
            font-size: 30px;
        }
        #div1
        {
            width: 300px;
            height: 300px;
            border: 4px solid orange;
        }
    </style>
</head>
<body>
    <h1>Choose background color</h1>
```

```
<select id="drp1">
    <option value="select">Select</option>
    <option value="red">Red</option>
    <option value="green">Green</option>
    <option value="blue">Blue</option>
</select>
<br>
<div id="div1">
    div1
</div>

<script>
    document.getElementById("drp1").addEventListener("change",
fun1);
    function fun1()
    {
        var selectedcolor = document.getElementById("drp1").value;
        if (selectedcolor == "red")
            document.getElementById("div1").style.backgroundColor =
"red";
        else if (selectedcolor == "green")
            document.getElementById("div1").style.backgroundColor =
"green";
        else if (selectedcolor == "blue")
            document.getElementById("div1").style.backgroundColor =
"blue";
        else
            document.getElementById("div1").style.backgroundColor =
"white";
    }
</script>
</body>
</html>
```

“Contextmenu” event

“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>JavaScript - 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");
      }
      //event = browser given information
    </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>JavaScript - Disabling cut, copy, paste</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </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(); //default functionality will be stopped
        alert("cut copy paste not allowed");
      }
      //event = browser given information
    </script>
  </body>
</html>
```

“this” keyword

“this” keyword

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

Example on “this” keyword:

```
<html>
  <head>
    <title>JavaScript - This</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </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">
    <input type="button" id="button6" value="click me">
    <input type="button" id="button7" value="click me">
    <input type="button" id="button8" value="click me">
    <input type="button" id="button9" value="click me">
    <input type="button" id="button10" 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);
document.getElementById("button6").addEventListener("click",
fun1);
document.getElementById("button7").addEventListener("click",
fun1);
document.getElementById("button8").addEventListener("click",
fun1);
document.getElementById("button9").addEventListener("click",
fun1);
document.getElementById("button10").addEventListener("click",
fun1);
function fun1()
{
  this.setAttribute("value", "thanx");
  this.style.backgroundColor = "skyblue";
}
</script>
</body>
</html>
```

Adding New Elements to DOM

II) Adding New Elements to DOM

- To add elements to DOM, follow the steps:
- Create the element:

Syntax: var *variablename* = document.createElement("tag");
Example: var x = document.createElement("div");

- Add the new element to existing container element:

Syntax: appendChild(*variablename*);
Example: appendChild(x);

Example on Adding Elements:

```
<html>
<head>
    <title>JavaScript - Adding elements</title>
    <style type="text/css">
        body,input
        {
            font: 30px tahoma;
        }
        div
        {
            background-color: #00cc99;
            padding: 20px;
            margin: 20px;
        }
        div p
        {
            background-color: #ff0099;
            padding: 10px;
            margin: 10px;
        }
    </style>
</head>
<body>
    <div id="div1">
        <p>para 1</p>
```

```
<p>para 2</p>
<p>para 3</p>
</div>
<input type="button" id="btn1" value="add para to div">

<script>
  document.getElementById("btn1").addEventListener("click", fun1);
  var n = 4;
  function fun1()
  {
    var mypara = document.createElement("p");
    mypara.innerHTML = "para " + n;
    mypara.setAttribute("id", "p" + n);
    document.getElementById("div1").appendChild(mypara);
    n++;
  }
</script>
</body>
</html>
```

Removing Existing Elements from DOM

III) Removing Existing Elements from DOM

- To remove elements from DOM, use the following method:

Syntax: remove();

Example: document.getElementById("div1").remove();

Example on Removing Existing Elements from DOM:

```

<html>
  <head>
    <title>JavaScript - Removing elements</title>
    <style type="text/css">
      body,input
      {
        font: 30px tahoma;
      }
      div
      {
        background-color: #00cc99;
        padding: 20px;
        margin: 20px;
      }
      div p
      {
        background-color: #ff0099;
        padding: 10px;
        margin: 10px;
      }
    </style>
  </head>
  <body>
    <div>
      <p>para 1</p>
      <p id="p2">para 2</p>
      <p>para 3</p>
    </div>
    <input type="button" id="btn1" value="remove para 2">
  </body>
</html>

```

```
<script>
    document.getElementById("btn1").addEventListener("click", fun1);
    function fun1()
    {
        document.getElementById("p2").remove();
    }
</script>
</body>
</html>
```

HARSHA

Image Gallery

Example on Image Gallery

```

<html>
  <head>
    <title>JavaScript - Image gallery</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      .class1
      {
        cursor: pointer;
      }
    </style>
  </head>
  <body>
    <h1>Image gallery</h1>
    
    <br><br>
    
    
    
    
    
    
    
    
    
    

    <script>
      var allimages = document.getElementsByClassName("class1");
      for (i=0; i<allimages.length; i++)
      {
        allimages[i].addEventListener("click", fun1);
      }
      function fun1()
      {
    
```

```
var currentsrc = this.getAttribute("src");
document.getElementById("myimage").setAttribute("src",
currentsrc);
}
</script>
</body>
</html>
```

Note: Place "img1.jpg" , "img2.jpg" , "img3.jpg" , "img4.jpg" , "img5.jpg" , "img6.jpg" , "img7.jpg" , "img8.jpg" , "img9.jpg" , "img10.jpg" in the current folder.

HARSHA

Photo Navigation

Example on Photo Navigation

```

<html>
  <head>
    <title>JavaScript - Photo navigation</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>Photo navigation</h1>
    <br>
    <input type="button" value="Previous" id="button1">
    <input type="button" value="Next" id="button2">

    <script>
      var allimages = [ "img1.jpg", "img2.jpg", "img3.jpg", "img4.jpg",
"img5.jpg", "img6.jpg", "img7.jpg", "img8.jpg", "img9.jpg", "img10.jpg" ];
      var c = 0;

      document.getElementById("button1").addEventListener("click",
fun1);
      function fun1()
{
  c--;
  if (c == -1)
  {
    c = 0;//reset c
  }
  var i = document.getElementById("myimage");
  i.setAttribute("src", allimages[c]);
}

      document.getElementById("button2").addEventListener("click",
fun2);
      function fun2()

```

```
{  
    c++;  
    if (c == allimages.length)  
    {  
        c = allimages.length - 1; //reset c  
    }  
    var i = document.getElementById("myimage");  
  
    i.setAttribute("src", allimages[c]);  
}  
</script>  
</body>  
</html>
```

Note: Place "img1.jpg" , "img2.jpg" , "img3.jpg" , "img4.jpg" , "img5.jpg" , "img6.jpg" , "img7.jpg" , "img8.jpg" , "img9.jpg" , "img10.jpg" in the current folder.

Random

Random

- This is a pre-defined method, which generates a random (non-sequential) number, between 0 to 1.
- Syntax: Math.random();
- Example: Math.random();

Example on Random:

```
<html>
  <head>
    <title>JavaScript - Random</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>random</h1>
    <script>
      var n = Math.random();
      document.write(n);
    </script>
  </body>
</html>
```

Example on Random Image:

```
<html>
  <head>
    <title>JavaScript - Random</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';

```

```
        font-size: 30px;  
    }  
  </style>  
</head>  
<body>  
  <h1>Random image</h1>  
  <img src="" alt="Image here" width="200px" id="randomimage">  
  
  <script>  
    var n = Math.floor(Math.random() * 10);  
    var myimages = [ "img1.jpg", "img2.jpg", "img3.jpg", "img4.jpg",  
    "img5.jpg", "img6.jpg", "img7.jpg", "img8.jpg", "img9.jpg", "img10.jpg" ];  
    document.getElementById("randomimage").setAttribute("src",  
    myimages[n]);  
  </script>  
  </body>  
</html>
```

Note: Place “img1.jpg”, “img2.jpg”, “img3.jpg”, “img4.jpg”, “img5.jpg”,
“img6.jpg”, “img7.jpg”, “img8.jpg”, “img9.jpg”, “img10.jpg” in the current
folder.

Conversion Functions

Conversion Functions

- JavaScript provides the following two functions to convert a value from “string” data type to “number” data type.
 1. parseInt()
 2. parseFloat()

1. parseInt()

- This function converts a value from “string” data type to “int” data type (number without decimal places).
- Syntax: parseInt(“string value”)
- Example:parseInt(“10”)

2. parseFloat()

- This function converts a value from “string” data type to “float” data type (number with decimal places).
- Syntax: parseFloat(“string value”)
- Example:parseFloat(“10.824”)

Example on Conversion Functions:

```
<html>
  <head>
    <title>JavaScript - Add Subtract Multiply Divide</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
      #txt3
      {
        background-color: lightgray;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - Add Subtract Multiply Divide</h1>
    <input type="text" id="txt1" value="100" />
    <input type="text" id="txt2" value="200" />
    <input type="button" value="Add" onclick="add()" />
    <input type="button" value="Subtract" onclick="subtract()" />
    <input type="button" value="Multiply" onclick="multiply()" />
    <input type="button" value="Divide" onclick="divide()" />
    <input type="text" id="txt3" value="Result" />
  </body>
</html>
```

```

#button1,#button2,#button3,#button4
{
    background-color: #ffcc99;
    border: 1px ridge red;
}
</style>
</head>
<body>
    <h1>Math</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;
        }
    </script>

```

```
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>
```

Login

Example on Login:

```
<html>
  <head>
    <title>JavaScript - 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>
```

String Functions

String Functions

- “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.
- JavaScript string literals should be in either single quotes or double quotes.
Ex: 'hello123'
"hello123"

List of String Functions

1. toUpperCase()

- This function converts the string value into uppercase and returns it.
- Syntax: string.toUpperCase()
- Example: "hello".toUpperCase() → "HELLO"

2. toLowerCase()

- This function converts the string value into lowercase and returns it.
- Syntax: string.toLowerCase()
- Example: "HELLO".toLowerCase() → "hello"

3. length

- This property returns the no. of characters in the string.
- Syntax: string.length
- Example: "hello".length → 5

4. 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

5. charCodeAt()

- This function returns the ASCII value of the single character present at the specified index.
- Syntax: string.charCodeAt()
- Example: "hello".charCodeAt(1) → 101

6. slice()

- This function returns a part of the string, starting from specified index, upto end of the string.
- Syntax: string.slice(index)
- Example: "hyderabad".slice(2) → derabad

7. 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.
- Syntax: string.indexOf("character")
- Example: "hyderabad".indexOf("a") → 5
- Example: "hyderabad".indexOf("era") → 3
- Example: "hyderabad".indexOf("z") → -1
- This function always considers the first occurrence only.

8. replace()

- It replaces a “word” with “another word”.
- Syntax: string.replace(“old word”, “new word”)
- Example: “MS Office”.replace(“Office”, “Windows”) → MS Windows

9. 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”]

10. 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”

11. concat()

- This function attaches two strings and make them as a single string.
- Syntax: string.concat(“another string”)
- Example: “peers”.concat(“tech”) → “peerstech”

Example on String Functions:

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

```
</style>
</head>
<body>
    <h1>JavaScript - String functions</h1>

    <script>
        document.write("hello" + "<br>");
        document.write("hello".toUpperCase() + "<br>");
        document.write("HELLO".toLowerCase() + "<br>");
        document.write("hello 123".length + "<br>");
        document.write("hello 123".charAt(1) + "<br>");
        document.write("hello 123".charCodeAt(1) + "<br>");
        document.write("hyderabad".slice(4) + "<br>");
        document.write("hyderabad".indexOf("a") + "<br>");
        document.write("hyderabad".indexOf("a",
("hyderabad".indexOf("a")+1)) + "<br>");
        document.write("Microsoft Office".replace("Office", "Windows") +
"<br>");
        document.write("how are you".split(" ")[2] + "<br>");
        document.write(" abc def ".trim().length + "<br>");
        document.write("peers".concat("tech") + "<br>");
    </script>
</body>
</html>
```

Date Functions

Date Functions

- Date functions are used to manipulate date and time.

List of Date Functions

1. **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)

2. **toLocaleDateString()**

- This function returns the date in the following format: M/d/yyyy
- Syntax: new Date().toLocaleDateString()
- Example: new Date().toLocaleDateString() → 8/10/2017

3. **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

4. **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

5. 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

6. getDate()

- This function returns only the date.
- Syntax: new Date().getDate()
- Example: new Date().getDate() → 10

7. getMonth()

- This function returns only the month (0 to 11).
- Syntax: new Date().getMonth()
- Example: new Date().getMonth() → 7

8. getFullYear()

- This function returns only the year.
- Syntax: new Date().getFullYear()
- Example: new Date().getFullYear() → 2017

9. getHours()

- This function returns only the hours (in 24 hours format).
- Syntax: new Date().getHours()

- Example: new Date().getHours() → 11

10. getMinutes()

- This function returns only the minutes.
- Syntax: new Date().getMinutes()
- Example: new Date().getMinutes() → 4

11. getSeconds()

- This function returns only the seconds.
- Syntax: new Date().getSeconds()
- Example: new Date().getSeconds() → 51

12. getMilliseconds()

- This function returns only the milli seconds.
- Syntax: new Date().getMiliseconds()
- Example: new Date().getMillisconds() → 481

13. 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 Date Functions:

```
<html>
  <head>
    <title>JavaScript - Date functions</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - Date functions</h1>
    <script>
      document.write(new Date() + "<br>");
      document.write(new Date().toLocaleDateString() + "<br>");
      document.write(new Date().toLocaleTimeString() + "<br>");
      document.write(new Date().getTime() + "<br>");
      document.write(new Date().getDay() + "<br>");
      document.write(new Date().getDate() + "<br>");
      document.write(new Date().getMonth() + "<br>");
      document.write(new Date().getFullYear() + "<br>");
      document.write(new Date().getHours() + "<br>");
      document.write(new Date().getMinutes() + "<br>");
      document.write(new Date().getSeconds() + "<br>");
      document.write(new Date().getMilliseconds() + "<br>");
    </script>
  </body>
</html>
```

Example on Custom Date:

```
<html>
  <head>
    <title>JavaScript - Custom date</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>Custom Date</h1>
    <script>
      var d = new Date();
      d.setFullYear(2016);
      d.setMonth(11); //0 to 11
      d.setDate(31);
      document.write(d.toLocaleDateString());
    </script>
  </body>
</html>
```

setTimeout()

setTimeout()

- It calls a function automatically after the specific no. of milli seconds completed.
- Syntax: `setTimeout(function, milli seconds);`
- Example: `setTimeout(fun1, 1000);`

```
function fun1()  
{  
}  
}
```

Example on setTimeout()

```
<html>  
  <head>  
    <title>JavaScript - setTimeout</title>  
  </head>  
  <body>  
    <h1>setTimeout</h1>  
    <input type="button" id="button1" value="Show Message after 3  
seconds">  
  
    <script>  
      document.getElementById("button1").addEventListener("click",  
fun1);  
      function fun1()  
      {  
        setTimeout(fun2, 3000);  
      }  
      function fun2()  
      {  
        alert("Hello");  
      }  
    </script>  
  </body>  
</html>
```

setInterval()

setInterval()

- It calls a function repeatedly, for every completion of the given interval time (milli seconds).
- Syntax: `setInterval(function, milli seconds);`
- Example: `setInterval(fun1, 1000);`

```

function fun1()
{
}
```
- We use `clearInterval()` function to stop the execution of `fun1()` repeatedly.

Example on setInterval()

```

<html>
  <head>
    <title>JavaScript - setInterval</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>setInterval</h1>
    <input type="button" id="button1" value="Show Message for every 3
seconds">

    <script>
      document.getElementById("button1").addEventListener("click",
fun1);
      function fun1()
      {
        setInterval(fun2, 3000);
      }
    </script>
  </body>
</html>
```

```
function fun2()
{
    alert("Hello");
}
</script>
</body>
</html>
```

Example 2 on setInterval()

```
<html>
<head>
    <title>JavaScript - setInterval - System Time</title>
    <style type="text/css">
        body,input
        {
            font-family: 'Tahoma';
            font-size: 30px;
        }
    </style>
</head>
<body>
    <h1>setInterval - System Time</h1>
    <div id="div1"></div>
    <script>
        setInterval(fun1, 1000); //for every 1000 milli sec (1 sec) it calls fun1.
        function fun1()
        {
            var d = new Date();
            var t = d.toLocaleTimeString();
            document.getElementById("div1").innerHTML = t;
        }
    </script>
</body>
</html>
```

Example 3 on setInterval()

```

<html>
  <head>
    <title>JavaScript - Automatic Photo Slide Show</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>Automatic Photo Slide Show (2 seconds)</h1>
    

    <script>
      var allimages = [ "img1.jpg", "img2.jpg", "img3.jpg", "img4.jpg",
"img5.jpg", "img6.jpg", "img7.jpg", "img8.jpg", "img9.jpg", "img10.jpg" ];
      var c = 0;
      var repeat = setInterval(fun1, 2000); //1000 milli sec = 1 sec

      function fun1()
      {
        c++;
        if (c == allimages.length)
        {
          clearInterval(repeat);
          alert("Done");
        }
        var i = document.getElementById("myimage");
        i.setAttribute("src", allimages[c]);
      }
    </script>
  </body>
</html>

```

Note: Place “img1.jpg” , “img2.jpg” , “img3.jpg” , “img4.jpg” , “img5.jpg” , “img6.jpg” , “img7.jpg” , “img8.jpg” , “img9.jpg” , “img10.jpg” in the current folder.

Validations

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)
    {
        //hide error message
    }
    else
    {
        //show error message
        event.preventDefault();
    }
}
```

Example on Validations:

```

<html>
  <head>
    <title>JavaScript - Validations</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
      .error
      {
        color: red;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - Validations</h1>
    <form id="f1">
      Username:
      <input type="text" id="txtusername" required> *
      <span id="spanusername" class="error"></span><br>
      Password:
      <input type="password" id="txtpassword" required> *
      <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
      }
    </script>
  </body>
</html>

```

```
{  
    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 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.

Syntax for Regular Expressions

/regular expression/.test(value)

Example on Regular Expressions:

```
<html>
  <head>
    <title>JavaScript - Regular Expressions</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
      .error
      {
        color: red;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - 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>
  </body>
</html>
```

```
</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 = "";
        }

        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>
```

Window.print()

Window.print()

- This function is used to print the current web page, through the printer.
- Syntax: window.print()

Example on window.print():

```

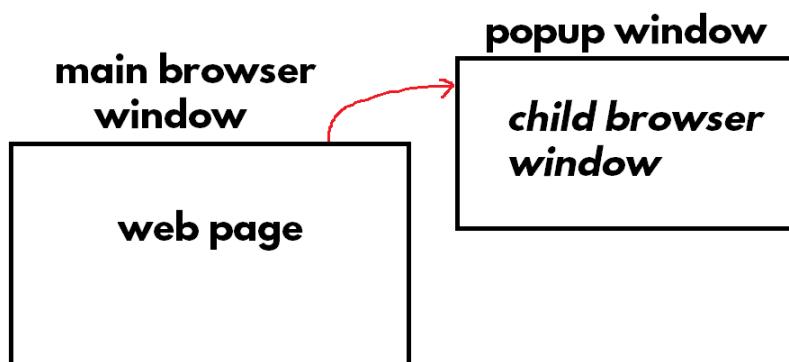
<html>
  <head>
    <title>JavaScript - Print</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>Print</h1>
    <p>Lorem Ipsum is simply dummy text of the printing and typesetting industry. 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.</p>
    <input type="button" value="Print" id="button1">
    <script>
      document.getElementById("button1").addEventListener("click",
fun1);
      function fun1()
      {
        window.print();
      }
    </script>
  </body>
</html>

```

Popup Window

Popup Window

- Popup window is an independent browser window.
- Popup windows are generally used to display ads.
- Syntax: window.open("filename.html", "nick name", "options");
- Example: window.open("mypage.html", "popup1", "width=300, height=300");



Example on Popup window:

mypage.html

```
<html>
  <head>
    <title>Popup</title>
  </head>
  <body>
    Hello, World
  </body>
</html>
```

Popup.html

```
<html>
<head>
    <title>JavaScript - Popup window example</title>
    <style type="text/css">
        body,input
        {
            font-family: 'Tahoma';
            font-size: 30px;
        }
    </style>
    <script>
        window.open("mypage.html", "popup1", "width=300, height=300");
        //syntax: 'url to open', "logical name", options
    </script>
</head>
<body>
    <h1>This page automatically opens a popup-window</h1>
    <p>Choose "Allow Popups" in the options.</p>
</body>
</html>
```

External JavaScript

Types of JavaScript

- JavaScript program can be created in two ways:
 1. Internal JavaScript / Embedded JavaScript
 2. External JavaScript

1. Internal JavaScript

- Both “html code” and “javascript code” will be written in the same “.html” file.
- **Advantage:** Easy to understand.
- **Disadvantage:** The javascript code that is written in “.html” file can’t be used in another html file.
- All the previous programs are the examples of “Internal JavaScript”.

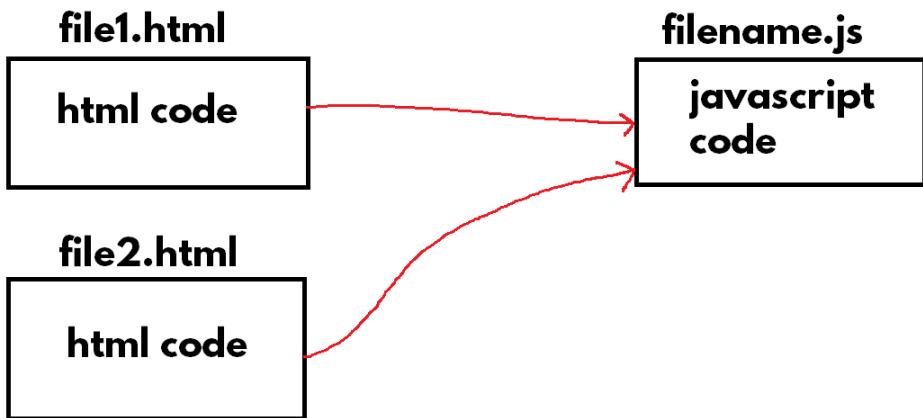
filename.html

html code

javascript code

2. External JavaScript

- The javascript code will be written in “.js” file and will be called in one or more “.html” files.
- **Advantage:** We can call the same “.js” file from many html files (re-usability).
- In relative, 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>
```

Object Oriented Programming

Object Oriented Programming

- 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.
- **Methods:** Manipulations on the properties. Methods are the functions stored inside the object.
- Example:
 - ◆ "Car" object
 - Properties
 - ◆ CarModel: Honda City
 - ◆ CarColor: Black
 - ◆ ColorNo: 1234
 - Methods
 - ◆ Start()
 - ◆ ChangeGear()
 - ◆ Stop()

Types of Creating Object:

- We can create object in 3 ways:
 1. Object Literals
 2. New Object()
 3. Constructor Function

- Object Literal

```
var variable = { "property": value, "method": function() { ... } }
```

- New Object()

```
var variablename = new Object();  
variable.property = value;  
variable.method = function() { ... };
```

- Constructor Function

```
function functionname(arguments)  
{  
    this.property = value;  
    this.method = function() { ... };  
}  
var variablename = new functionname();
```

Object Literals

Example on Object Literals

```
<html>
  <head>
    <title>JavaScript - Object Literals</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - 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>JavaScript - Object Literals - Methods</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
```

```
<body>
  <h1>JavaScript - 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.studentid);
    console.log(stu.studentname);
    console.log(stu.marks);
    console.log(stu.result());
  </script>
</body>
</html>
```

Example 3 on Object Literals

```
<html>
  <head>
    <title>JavaScript - Object Literals - TextBoxes</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - Object Literals - TextBoxes</h1>
    Student ID: <input type="text" id="txt1"><br>
    Student Name: <input type="text" id="txt2"><br>
    Marks: <input type="text" id="txt3"><br>
```

Result: <input type="text" id="txt4">


```
<script>
    var stu = { "studentid":1, "studentname": "scott", "marks": 80,
"result":function(){
    if(this.marks >= 35)
    {
        return "Pass";
    }
    else
    {
        return "Fail";
    }
};
document.getElementById("txt1").value = stu.studentid;
document.getElementById("txt2").value = stu.studentname;
document.getElementById("txt3").value = stu.marks;
document.getElementById("txt4").value = stu.result();
</script>
</body>
</html>
```

New Object

Example on New Object

```
<html>
  <head>
    <title>JavaScript - New Object</title>
  </head>
  <body>
    <h1>JavaScript - New Object</h1>

    <script>
      var stu = new Object();
      stu.studentid = 1;
      stu.studentname = "scott";
      stu.marks = 80;
      stu.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

Example on Constructor Function

```
<html>
  <head>
    <title>JavaScript - Constructor Function</title>
  </head>
  <body>
    <h1>JavaScript - 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>
```

Example on User Data

```
<html>
  <head>
    <title>JavaScript - User Data</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - User Data</h1>
    Student ID: <input type="text" id="txt1"><br>
    Student Name: <input type="text" id="txt2"><br>
    Marks: <input type="text" id="txt3"><br>
    <input type="button" value="OK" id="button1">

    <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";
          }
        };
      }

      document.getElementById("button1").addEventListener("click",
fun1);
      function fun1()
```

```
{  
    var x = document.getElementById("txt1").value;  
    var y = document.getElementById("txt2").value;  
    var z = document.getElementById("txt3").value;  
  
    var stu = new Student(x, y, z);  
    console.log(stu);  
    console.log(stu.studentid);  
    console.log(stu.studentname);  
    console.log(stu.marks);  
    console.log(stu.result());  
}  
</script>  
</body>  
</html>
```

Object.Keys

Object.Keys

- The “Object.keys” method is used to retrieve the list of properties of an object as an array.
- 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>JavaScript - Keys</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - 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);
var keys = Object.keys(stu);
console.log(stu);
console.log(keys);
for (var i = 0; i < keys.length; i++)
{
    console.log(stu[keys[i]]);
}
</script>
</body>
</html>
```

Stringify

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(object literal or reference variable)
- Example: JSON.stringify(stu)

Example on JSON.stringify

```

<html>
  <head>
    <title>JavaScript - Stringify</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - Stringify</h1>
    <div id="div1">div 1</div>

    <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
        }
      }
    </script>
  </body>
</html>

```

```
{  
    return "Fail";  
}  
};  
}  
var stu = new Student(1, "Scott", 80);  
document.getElementById("div1").innerHTML = JSON.stringify(stu);  
</script>  
</body>  
</html>
```

HARSHA

Parse

Parse

- The “JSON.parse” is used to convert “JSON” to “Object Literal” format.
- Syntax: JSON.parse(json data)
- Example: JSON.stringify(' { "a":10, "b": 20 } ')

Example on JSON.parse

```
<html>
  <head>
    <title>JavaScript - Parse</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - Parse</h1>

    <script>
      var s = '{ "studentid":1, "studentname": "scott", "marks": 80 }';
      var stu = JSON.parse(s);
      console.log(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

- “Object Array Literal” is a collection of “object literals”, stored as an array.

Example on Object Array Literal:

```
<html>
  <head>
    <title>JavaScript - Object Array Literal</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
      #div1
      {
        background-color: #00cccc;
        padding: 10px;
        margin:10px;
      }
      #div1 p
      {
        background-color: #009966;
        padding: 10px;
        margin:10px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - Object Array Literal</h1>
    <div id="div1">
    </div>

    <script>
      var employees =
      [
        { "empid": 101, "empname": "Scott", "salary": 4000 },
        { "empid": 102, "empname": "Smith", "salary": 5690 },
      ]
    </script>
  </body>
</html>
```

```

    { "empid": 103, "empname": "Allen", "salary": 6723 },
    { "empid": 104, "empname": "John", "salary": 8729 }
];
for (var i = 0; i < employees.length; i++)
{
    var s = "<p>" + employees[i].empid + ", " + employees[i].empname +
", " + employees[i].salary + "</p>";
    document.getElementById("div1").innerHTML += s;
}
</script>
</body>
</html>

```

Example on Object Array Literal - UL:

```

<html>
  <head>
    <title>JavaScript - Object Array Literal - UL</title>
  </head>
  <body>
    <h1>JavaScript - Object Array Literal - UL</h1>
    <ul id="list1">
    </ul>
    <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 }
];
for (var i = 0; i < employees.length; i++)
{
    var s = "<li>" + employees[i].empid + ", " + employees[i].empname +
", " + employees[i].salary + "</li>";
    document.getElementById("list1").innerHTML += s;
}
</script>
</body>
</html>

```

Example on Object Array Literal - Table:

```
<html>
  <head>
    <title>JavaScript - Object Array Literal - Table</title>
  </head>
  <body>
    <h1>Object Array Literal - Table</h1>
    <table id="table1" border="1">
      <tr>
        <th>Emp ID</th>
        <th>Emp Name</th>
        <th>Salary</th>
      </tr>
    </table>

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

Object Array

Example on Object Array:

```
<html>
  <head>
    <title>JavaScript - Object Array</title>
    <style type="text/css">
      body,input
      {
        font-family: Tahoma;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>JavaScript - 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>
```

Complex Object Literal

Example on Complex Object Literal:

```

<html>
  <head>
    <title>Complex Object Literal</title>
    <style type="text/css">
      body,input,table
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>Complex Object Literal</h1>
    <div id="div1"></div>

    <script>
      var departments =
      [
        { deptno: 10, deptname: "Accounting", loc: "New York",
employees: [
          { empid: 1, empname: "Scott" },
          { empid: 2, empname: "Allen" },
          { empid: 3, empname: "Smith" }
        ] },

        { deptno: 20, deptname: "Sales", loc: "New Delhi", employees: [
          { empid: 4, empname: "Jones" },
          { empid: 5, empname: "John" }
        ] },

        { deptno: 30, deptname: "R&D", loc: "New Mumbai", employees: [
          { empid: 6, empname: "James" },
          { empid: 7, empname: "Mark" },
          { empid: 8, empname: "Potter" },
          { empid: 9, empname: "Jord" }
        ] }
      ];
    </script>
  </body>
</html>

```

```
var temp = "";
for (i=0; i < departments.length; i++)
{
    var dept = departments[i];
    temp += "<p>" + dept.deptno + "," + dept.deptname + "," + dept.loc
+ "</p>";
    temp += "<table border='1' cellpadding='4px'>";
    for (j=0; j < dept.employees.length; j++)
    {
        var emp = dept.employees[j];
        temp += "<tr>";
        temp += "<td>" + emp.empid + "</td>";
        temp += "<td>" + emp.empname + "</td>";
        temp += "</tr>";
    }
    temp += "</table>";
}
document.getElementById("div1").innerHTML = temp;
</script>
</body>
</html>
```

Prototype

Prototype

- Every JavaScript Object has a prototype.
- The “prototype” also is an object that represents the list of properties and methods of its objects.
- We can add properties or methods to the object’s prototype; so that those properties and methods will be automatically added to the corresponding object.

Example on Prototype:

```
<html>
  <head>
    <title>JavaScript - Prototype</title>
  </head>
  <body>
    <h1>JavaScript - Prototype</h1>

    <script>
      /*creating class*/
      function Student(a, b, c, d, e)
      {
        this.studentid = a;
        this.studentname = b;
        this.marks1 = c;
        this.marks2 = d;
        this.marks3 = e;
        this.totalmarks = 0;
        this.averagemarks = 0;
        this.grade = "";
      }

      Student.prototype.calculatetotalmarks = function()
      {
        this.totalmarks = this.marks1 + this.marks2 + this.marks3;
      };

      Student.prototype.calculateaveragemarks = function()
      {
        this.averagemarks = this.totalmarks / 3;
      };
    </script>
  </body>
</html>
```

```

    {
        this.averagemarks = this.totalmarks / 3;
   };

Student.prototype.calculategrade = function()
{
    if(this.marks1 >= 35 && this.marks2 >= 35 && this.marks3 >= 35)
    {
        if(this.averagemarks >= 80 && this.averagemarks <= 100)
        {
            this.grade = "A grade";
        }
        else if(this.averagemarks >= 60 && this.averagemarks < 80)
        {
            this.grade = "B grade";
        }
        else if(this.averagemarks >= 50 && this.averagemarks < 60)
        {
            this.grade = "C grade";
        }
        else if(this.averagemarks >= 35 && this.averagemarks < 50)
        {
            this.grade = "D grade";
        }
    }
    else
    {
        this.grade = "Fail";
    }
};

/*creating objects*/
var student1 = new Student(101, "scott", 70, 85, 90);
var student2 = new Student(102, "allen", 80, 45, 77);

/* call methods */
student1.calculatetotalmarks();
student1.calculateaveragemarks();
student1.calculategrade();
student2.calculatetotalmarks();
student2.calculateaveragemarks();
student2.calculategrade();

```

```
/* display output */
document.write("Student ID: " + student1.studentid);
document.write("<br>");
document.write("Student Name: " + student1.studentname);
document.write("<br>");
document.write("Marks 1: " + student1.marks1);
document.write("<br>");
document.write("Marks 2: " + student1.marks2);
document.write("<br>");
document.write("Marks 3: " + student1.marks3);
document.write("<br>");
document.write("Total Marks: " + student1.totalmarks);
document.write("<br>");
document.write("Average Marks: " + student1.averagemarks);
document.write("<br>");
document.write("Grade: " + student1.grade);
document.write("<hr>");

document.write("Student ID: " + student2.studentid);
document.write("<br>");
document.write("Student Name: " + student2.studentname);
document.write("<br>");
document.write("Marks 1: " + student2.marks1);
document.write("<br>");
document.write("Marks 2: " + student2.marks2);
document.write("<br>");
document.write("Marks 3: " + student2.marks3);
document.write("<br>");
document.write("Total Marks: " + student2.totalmarks);
document.write("<br>");
document.write("Average Marks: " + student2.averagemarks);
document.write("<br>");
document.write("Grade: " + student2.grade);
</script>
</body>
</html>
```

Inheritance

Inheritance

- The process of creating a prototype based on another prototype is called as "inheritance".
- So all the properties and methods of the parent prototype is inherited into the child prototype.
- Syntax:

```
function parent(arguments)
{
    ...
}
```

```
function child(arguments)
{
    parent.call(this, arguments);
    ...
}
```

Example on Inheritance:

```
<html>
  <head>
    <title>Inheritance</title>
    <style type="text/css">
      body,input
      {
        font-family: 'Tahoma';
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>Inheritance</h1>
```

```
<script>
/*parent class*/
function Person(a, b)
{
    this.name = a;
    this.email = b;
}
/*child class 1*/
function Student(a, b, c)
{
    Person.call(this, a, b);
    this.marks = c;
}
/*child class 2*/
function Employee(a, b, c)
{
    Person.call(this, a, b);
    this.salary = c;
}

/*creating objects*/
var stu = new Student("scott", "scott@gmail.com", 70);
var emp = new Employee("allen", "allen@gmail.com", 8000);
/* display output */
document.write("<h3>Student</h3>");
document.write("Name: " + stu.name);
document.write("<br>");
document.write("Email: " + stu.email);
document.write("<br>");
document.write("Marks: " + stu.marks);
document.write("<hr>");
document.write("<h3>Employee</h3>");
document.write("Name: " + emp.name);
document.write("<br>");
document.write("Email: " + emp.email);
document.write("<br>");
document.write("Salary: " + emp.salary);
</script>
</body>
</html>
```

Clousers

Clousers

- “Clousers” are used to create private variables that are accessible to only a set of methods.
- Syntax:

```
var functionname = function()  
{  
    var variablename = value;  
    return {  
        method: function()  
        {  
            code  
        },  
        method: function()  
        {  
            code  
        }  
    };  
};  
  
new functionname();
```

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--;  
        },  
        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

Hoisting

- JavaScript automatically lifts-up the variable declarations 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>
```