

UI and Web Development

Software application and software product :

- A Software that is designed according to client requirements is known as “Application”.
- A Software designed according to market requirements is known as “Software product”.

Example : IRCTC - Application

MS Office - Product

Types Of Applications and Products :

Type	Description
1. Desktop	It is a stand alone application that is installed and used in the business with less user base and no remote access.
2. Web Application	It is an internet based application used for business with large user base and remote access.
3. Distributed	A Distribute technology allows applications running on two different machines to share info between them.
4. Gaming	It is a 2D and 3D graphics application supported to game consoles like X-Box.
5. Mobile	It is a mobile relative application that runs on android, IOS and Windows.

6. AI	It is an Artificial Intelligence application that enables automation and robotics.
7. IOT	It refers to Internet Of Things, which is an embedded technology that allows remote access.

What Is UI ?

- A Software application handles various functionalities according to the business requirements. The Functionalities are designed by using different technologies.
- The user requires special skills in handling an application designs with specific technology. Hence we provide a user friendly interface so that users can easily interact with application. It is often known as “UI”

What Is UX ?

- UX in software application refers to user experience.
- The modern applications require more UX along with the UI.
- The challenges with UX in modern web applications are
 - (a) Fluid UX
 - (b) Unified UX
- The Fluid UX enables an application to load every content on to single page. The new details are updated page. The new details are updated without reloading the page.
- The Unified UX provider same experience for application across any device i.e from a mobile to browser.

What are Progressive Applications ?

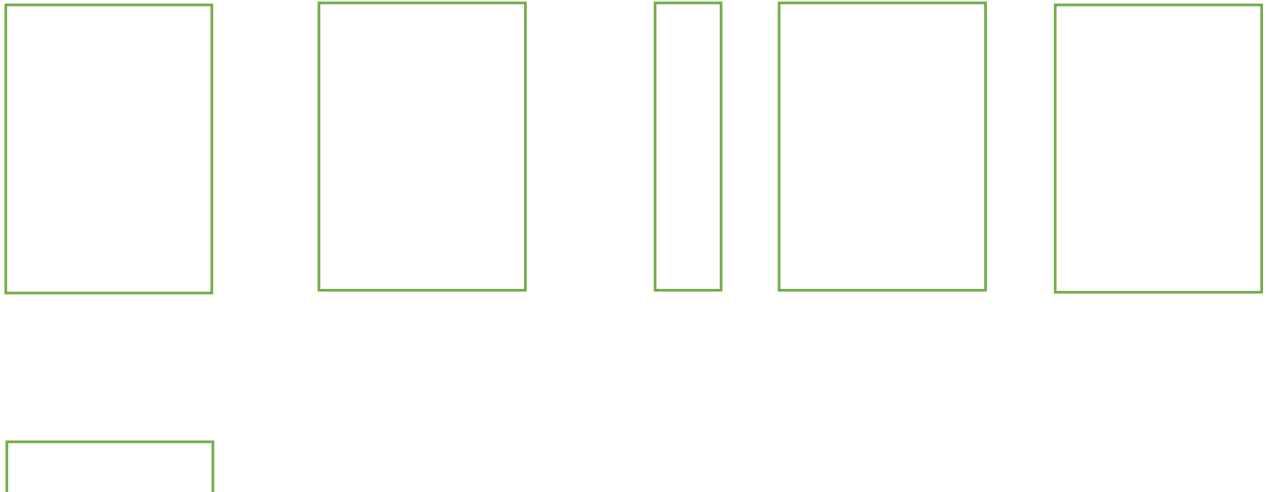
A Progressive application will provide app like experience even on a browser.

WEB DEVELOPMENT

1. Network : A Computer network comprises of group of computers connecting with each other for sharing info and resources.
2. Types of Networks : The Computer networks are categorized into 3 major types -
 - i. LAN (Local Area Network)
 - ii. MAN (Metropolitan Area Network)
 - iii. WAN (Wide Area Network)
3. Internet : It resembles a wide area network that connects computer across the world.
4. Web :
 - Web is a portion of internet with restricted access.
 - Tim Berner's Lee introduced web in 1990.
 - It uses a network mechanism where client sends request and server send response.

- The standards of web are maintained by W3C (World Wide Web Consortium)

Web Architecture



Data base server : Data base server host the data and handles the various responsibilities which includes

- a) Load Balancing
- b) Performance Turning
- c) High availability
- d) Security etc.,

Application Server : It hosts the application and it is responsible for various application level operations, which includes

- a) Load Balancing

b) High Availability

c) Security etc.,

Web Server :

- Web server resembles both hardware and software. It satisfies the client request by sending and receiving the data.
- The popular web server software's are IIS, TOMCAT, JBOSS etc.,

Locating web server on windows :

- Open Windows control panel
- Switch to large icons
- Go to Administrative tools
- Look for Internet Information Services (IIS) manager

Add IIS to Windows PC :

- Go to Control panel
- Open “ Programs and features “
- Click the option “ Turn the windows features on “
- Select “ Internet information services “
- Click OK

Test your Web server :

- Open any browser : Chrome

- Type the following in address bar

<http://localhost/amazon> → Created website name

(or)

<http://127.0.0.1/amazon>

Note : Make sure that your IIS is started. If stopped, then go to IIS and click START

Website : website is a virtual directory on web server that comprises of resources which are given access to client.

Creating a website on local server :

1. Open IIS [for that RUN → inetmgr]
2. Expand [+] local computer
3. Expand [+] “sites” folder
4. Right click on “default websites”
5. Select the option

“Add virtual directory”

Alias (website name) : Amazon

Physical Path : c:\amazon

6. Click OK

Virtual path : <http://localhost/amazon>

Physical path : c:/amazon>

7. Go to “c:\amazon” using file explorer and add following files
 - images - car.jpg

-docs - cssdemo.pdf

8. Access from browser

<http://localhost/amazon/images/car.jpeg>

Web Page :

Web page is a hyper text document that provide an UI for website. It allows the user to view,access and interact with resources in a website.

The web pages are categorized in to 2 types

1. Static Page
2. Dynamic Page

Static Page :

- The term static refers to continuous memory.
- The memory allocated for the first request the same memory will be used across other requests.
- A static page contains same information to display across any no.of requests and clients.
- The static pages are designed by using HTML, CSS, and Client side script.
- A static page have two extensions .html, .htm.

Example : home.html, about.htm

Dynamic Page :

- The term dynamic refers to non static memory.
- It is a discrete memory i.e, memory is newly allocated for every request.
- A dynamic page contains information that change according to client requests, it generates a response customized for every request.
- The dynamic pages are designed by using server side technologies like JSP, PHP, ASP etc.,
- They have the extension like .jsp, .asp, .php, .aspx.

Example : results.jsp, movies.aspx, ticket.php.

Web application :

A Web application provides an UI from where user can interact with the business. It requires server side technologies like JSP, PHP, ASP etc.,

Blog : [web-log]

Blogs are like journals on internet. Usually published by individual users and updated periodically.

Example : blogger.com, wix.com

Micro blog :

A micro blog is similar to blog but allows multiple users to post their personal information on to single page.

Example : twitter

Wiki : [quick-hawain]

The term wiki means quick in hawain language. It allows any anonymous user to edit its content.

Example : wikipedia, google maps

Podcasting :

A podcasting allows to broad cast media, which includes audio and video content. You can cast the content on to other devices like smart tv, bluetooth speaker etc.,

Example : youtube, itunes etc.,

Widgets :

It is a gadget for a website (or) web application. It is an application designed to perform specific functionality in a website.

Example : currency convertor, loan calculator etc.,

Web debugger :

It is a software tool used to track the performance of any page (or) application. Every browser have a debugger which you can invoke by using F12 function key.

Example : fiddler, post man etc.,

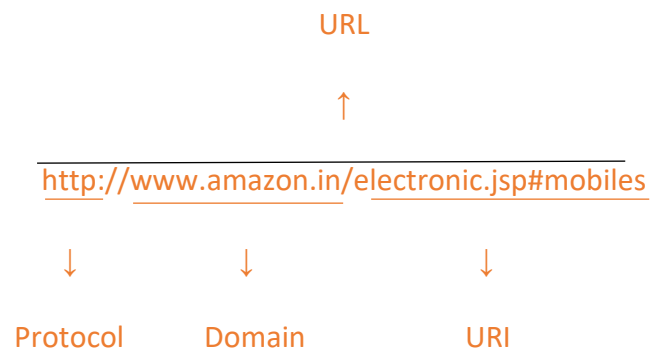
Api's :

Api is a service provided by third party sources which you can integrate in to any application to provide a specific functionality.

Example : google maps api, facebook login api, captcha api etc.,

URL and URI :

- URL is the virtual path generated by a web server and used by clients in order to access the resources from a website.
- It is a uniform resource locator.
- The URI is an identifier used to access any specific location within the resource.



Protocol : A protocol defines a set of rules that are used by computers in network to communicate with each other. The popular protocols are - http, https, ftp, file, ttp/ip, msmq, named pipes, smtp etc.,

http - Hyper text transfer protocol - web

https - secured http - web

ftp/file - file transfer protocol - intranet

Smtp - simple mail transfer protocol - email

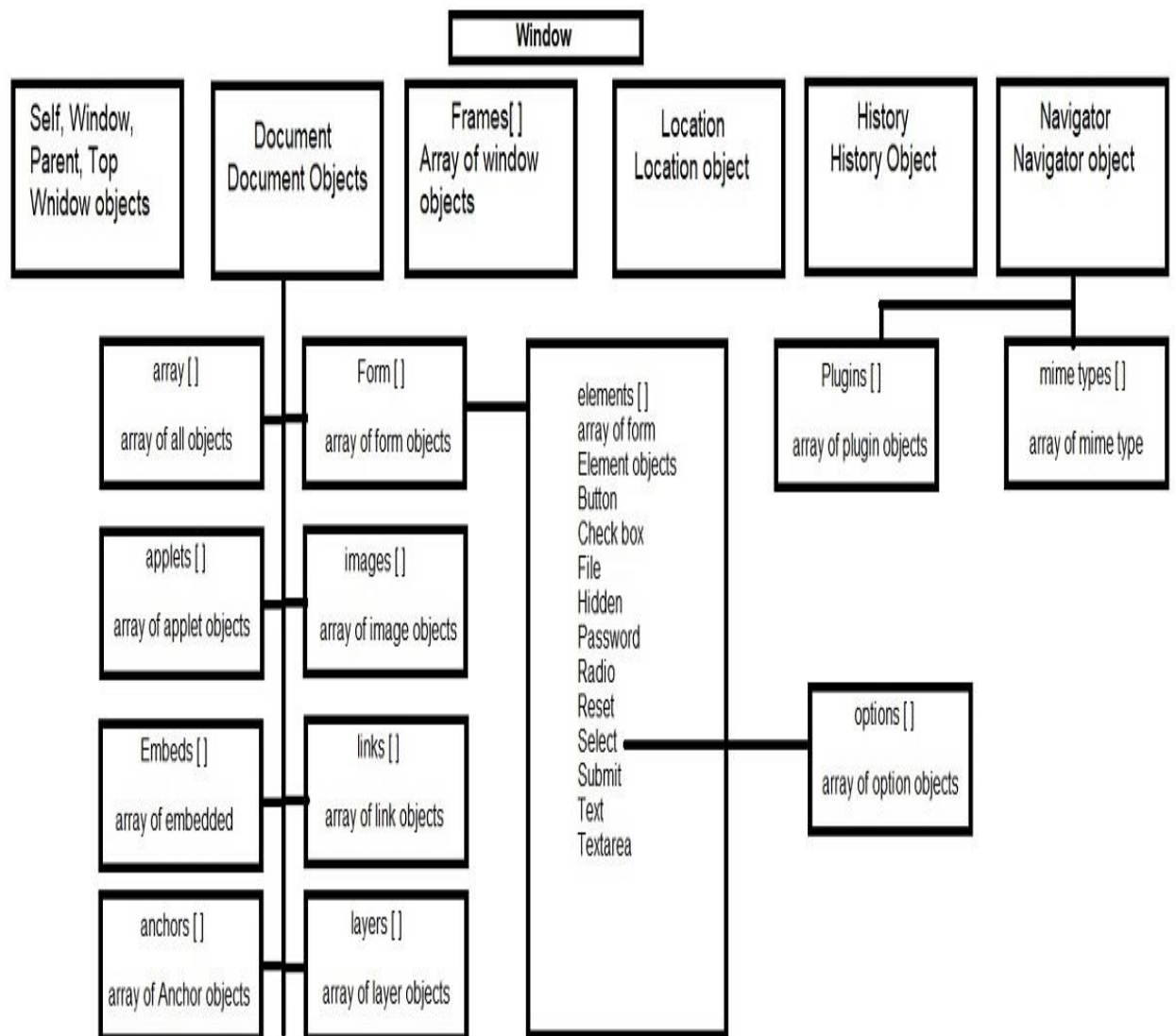
HTML

Hyper Text Markup Language

- The term Markup is used in computing technologies for specifying a presentation.
- The contents are marked up in order to present on screen.
- Markup languages are used on internet for presenting information on a browser.
- GML - Generic Markup Language

&

- SGML - Standard Generic Markup Language are the first markup languages used for internet.
- TIM BERNERS LEE introduced “HTML” as a markup language for web in early 1990’s.
- HTML is a markup language that follows the standard of “http” [protocol].
- HTML is a collection of elements arranged in a hierarchy called DOM [Document Object Model].
- HTML is a collection of elements that are presented by using tags.
- The Elements in HTML are categorized in to five groups -
 - i. Normal Elements
 - ii. Void Elements
 - iii. RC Data Elements
 - iv. Raw Text Elements
 - v. Foreign Elements



(i) Normal Elements :

Normal elements return a presentation directly on call back. They need a start and end tag i.e you have to explicitly end an element.

Example : `html`

(ii) Void Elements :

The void elements will not return any presentation directly on call back. The return void specifies no return type. These elements return a value and stop implicitly. Hence, they doesn't require an end tag.

Example : ``

(iii) RC Data Elements :

The RC Data elements are used to present "Rich Text Content". They will not allow any another element within the context.

Example : `I/P <textarea>`

Your text HTML

</textarea>

O/P Your text HTML

(iv) Raw Text Elements :

The Raw text elements (or) HTML elements which are presented by using raw text characters like &, <, >.

Example : ₹ 4500/- → It convert to rupee

<html>

(v) Foreign Elements :

The foreign elements are HTML elements used in HTML but requires external libraries. You Have to import the library in order to implement the elements.

Example : SVG, Mathml, Canvas etc.,

Structure of Static Page

A static page is designed in html comprises of two major specifications at high level.

(i) Document Declaration

(ii) Document Scope

Document Declaration :

- HTML is available in two major versions i.e HTML 4 and HTML 5, which are used in real world development.

- Every browser have a parsar (translator) that can handle both versions of HTML.
However, Parsar uses the 4 version by default.
- You have to define and specify about HTML 5 version by using the document declaration.

<!DOCTYPE html>

Document scope :

A document scope can display multiple documents simultaneously at a same time.

- A Developer must define the scope of every document by using the element <html>.

Syntax :

<!DOCTYPE html>

<html>

Document-1

</html>

<html>

Document-2

</html>

Structure of Document Scope :

The HTML Document scope is defined by using the tag HTML and at high level it comprises of two sections.

1. Head
2. Body

Head Section :

- The Head section comprises of components that are intended to load in to browser memory when a page is requested by client.
- The contents of head section are use by browser (or) by the workspace.
- The Head section is defined by using the tag.

`<head>`

- HTML head section comprises of following components
 - a. Title
 - b. Link
 - c. Meta
 - d. Script
 - e. Style

a) Title :

It is an element defined in head section and used to display a title for page in the browser title box.

Syntax :

`<head>`

`<title> Amazon | Home < /title>`

`</head>`

b) Link :

It is an element defined in head section and used to link external files like CSS, Shortcut icon etc.,

Syntax :

<head>

<link rel="shortcut icon" href="images/favicon.ico">

</head>

❖ It is an void element.

Creating a Favicon :

1. Open MS Paint
2. Set a page to 16*16 (or) 24*24
3. Draw your icon and save in website physical path by

Name : "favicon"

Type : PNG

Process to change PNG to ICO :

- I. Run cmd
- II. Select the drive E: and press enter
- III. E:\>cd E:\images → picture folder and press enter
- IV. E:\images>rename favicon.png favicon.ico and press enter

c) Meta :

- Meta describes the meta data i.e the information about your page.
- The contents of meta are required to define in a web page to make it more SEO friendly [search engine optimization].
- Meta comprises of attributes that are used to configure your page. So that crawlers can easily find the page.
- The attributes are,

Attributes	Description
1. Charset	It defines the charset used for coding. It follows UTF [unicode transformation format]
2.Key words	It defines the keywords used for identifying your page and show in search results.
3.Description	It defines the summary to display about your website in search results.
4.http-equiv	It defines how the request is handled for page.
5.View point	It makes the page more responsive. So that it fits on to any device.

Note : The meta data options are defined by using “name” attribute and the values are defined by using “content” attribute.

Example : About.html

```
<!DOCTYPE html>

<html lang="en-IN">

<head>

    <title> About us | Naresh IT </title>

    <link rel="shortcut icon" href="images/favicon.ico">

<meta charset="UTF-8">

<meta name="keywords" content="Best software training, Best IT training, in HYD, Chennai">

<meta name="description" content="something about your site..">

<meta http-equiv="refresh" content="4">

<meta name="viewport" content="width=device-width,initial-scale=1.0">

</head>

</html>
```

(d) Script : It is used to embedded client side and server side script in to a web page, which includes Java Script, JQuery, Angular, ASP etc.,

Syntax :

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

</script>

<script runat="server">

</script>
```

(e) Style : It is used to embedded styles in to web page. Which makes html more interactive and responsible.

Syntax :

`<style type="text/css">` →Script, Style these two are present in any section not stick to head section.

`</style>`

Body Section in HTML :

The body section comprises of information i.e displayed in the browser workspace. It is defined by using the tag body.

The body tag comprises of attributes that are used to control the appearance of contents in body section.

Attribute	Description
bg color	It sets a background color.
background	It sets a background image.
text	It sets color for body text.
vlink	It sets color for visited link.
alink	It sets color for active link
Left margin Right margin Top margin Bottom margin	It sets space between the content and margin.

Syntax :

`<body bgcolor="green" text="white">`

`<body background="images/logo.png">`

<body vlink="red" alink="green">

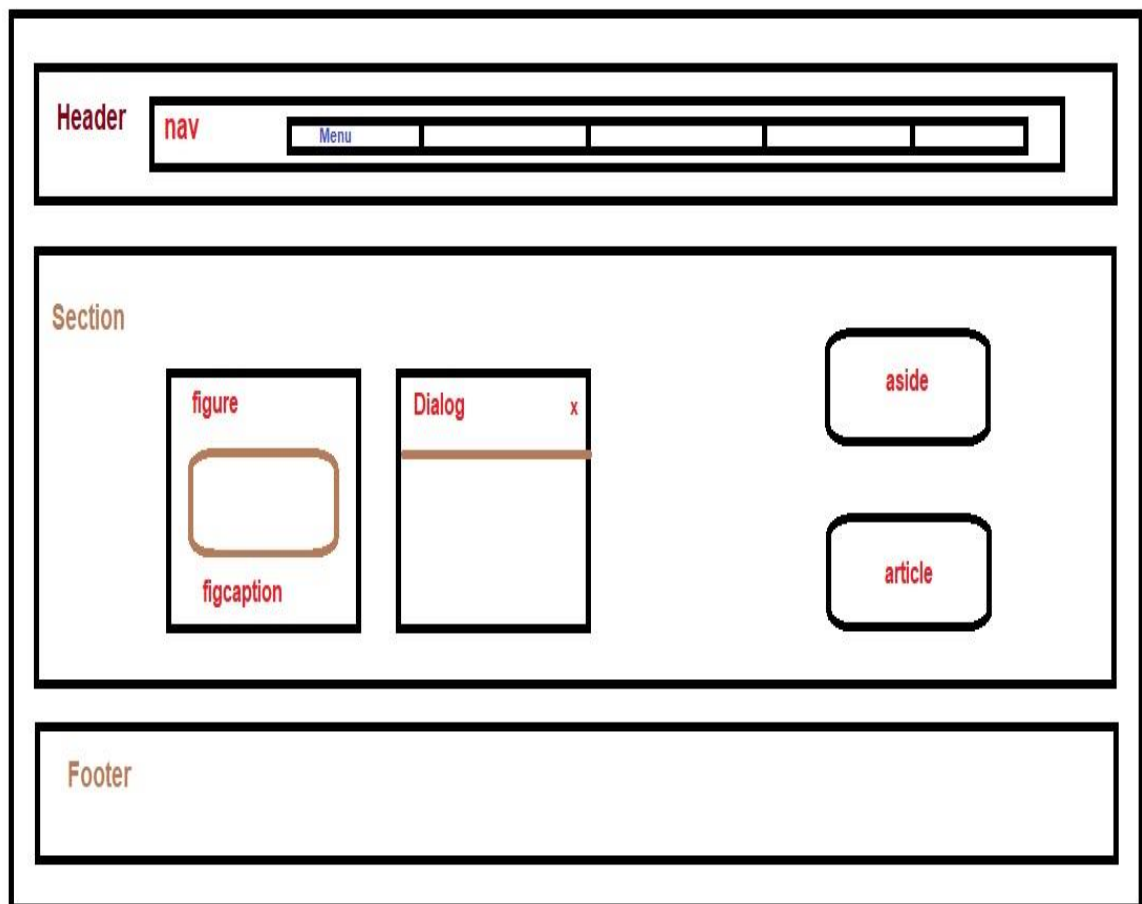
<body leftmargin="50" topmargin="100">

Elements in Body section :

The html-5 body section is classified with several new components, which make the contents more SEO friendly. The elements are,

Element	Description
aside	It is a container used to present information which is not related to the current website.
article	It is used to publish information which is related to the current website.
dialog	It presents an area on page from where user can interact with our website. It can be turned ON (or) OFF.
figure	It encapsulates any picture (or) graphic with a caption.
figcaption	It is used to set a caption for image.
header	It specifies the content to display at the top margin of page.
footer	It specifies the content to display at bottom margin of page.
section	It specifies the content to display between header and footer.
nav	It specifies an area used for navigation.
menu	It defines a collection of items arranged according to functionality in to various categories.
div	It defines a container to display a large block of content and load line by line.

span	It defines a container to present in line block of code to display side by side.
------	--



Setup Environment with IDE : [visual studio code]

- Download and install VS code for your PC

<https://code.visualstudio.com>

- Open VS Code installed on PC

- File Menu → Open → C:/Amazon
- In the left panel choose “Extensions”
- Search online for “Live server” - install
- “open in browser” - install
- “VS.Code.Icons” - install
- Go to “explorer” in left panel.
- ✓ [+] Add File/ Add Folder
- ✓ Name file “home.html”
- ✓ Right click in Design
- ✓ Open with Live server

Literals in HTML Body :

Html cannot identify the formats defined in an editor and presents your content in a single line by removing the formats. Hence you have to use following literals,

literals	Description
 	Inserts a manual line break
 	Inserts a blank space
<pre>	Presents text without removing the formats.[pre formatted]

Example : Line Break

<div>

First line
 Second line

</div>

Blank space

<div>

H T M L

</div>

Pre formatted

<div>

<pre>

#include <stdio.h>

Main()

{

Printf("Welcome to HTML")

}

</pre>

</div>

FAQ What is a difference between
, </br>,
 ?

A) Technically
 is the correct option.

</br> not recommended

 It specifies self ending for void elements to ignore warnings from debugger.

Acronymn : It is used to display screen tips. Which are shown only when mouse pointer is over the content.

Syntax :

<div>

Host your site on <acronymn title="internet Information Services">IIS.</acronymn>

</div>

Headings in HTML :

- Headings in a Web page are defined by using the tag <hn>, Where n refers to level number 1 to 6.
- Headings in a page are used for summarizing the content in SEO.
- Headings have a line break before and after.
- Using lot of headings may spam the page.
- Don't use headings for highlighting the text.
- Heading appearance can be changed by using styles.

Example :

<h1> HTML </h1>

<h2> Web Site </h2>

It is a Virtual Directory.

<h2> Web Server </h2>

It resembles hardware and software.

Attribute :

“Align” is an attribute, which can align heading left, center or right.

```
<h1 align="center"> HTML </h1>
```

Paragraphs :

Paragraphs are used to present text. A paragraph encapsulates set of lines in to a container and aligns left, center, right (or) justify.

Example :

```
<p align="center">  
...your text....  
</p>
```

BlockQuote :

- It similar to paragraphs but can indent left and right margin.
- It can be use in SEO for summarizing the contents.

Example :

```
<blockquote align="justify">  
....your text.....  
</blockquote>
```

Data List :

A Data list is a collection of terms and definitions, which are displayed in a pre defined format. The tags used for data list are,

Tag	Description
<dl>	It specifies the list of terms and definitions.
<dt>	It defines a term in data list.
<dd>	It defines a definition under data term.

Example :

```
<dl>

    <dt>Web Site</dt>

    <dd>It is a Virtual Directory</dd>

    <dt>URL</dt>

    <dd>It is Virtual Path<dd>

</dl>
```

Details and Summary :

HTML provides a set of elements for displaying the contents with a summary which you can expand (or) collapse dynamically.

- The <details> element is used to encapsulate any content and summary is used to define a summary text for the content.

Example :

```
<body>

    <h1>HTML</h1>

    <details open>

        <summary>Web Server</summary>
```

It resembles hardware and software.

</details>

<details>

<summary>Web Site</summary>

It is a Virtual Directory

</details>

</body>

Ordered List:

- It is a collection of items defined with auto numbering style.
- The numbering will update automatically when you add (or) delete an item.
- The ordered list is defined within tag.
- The list items are defined by using tag.

Syntax :

 Item-1

 Item-2

Ordered list can defined by using following attributes,

Attribute	Description
-----------	-------------

type	It specifies the numbering type of list them, which can "1,a,A,l,i "
start	It specifies the level number to start with. It requires a numeric value for level.
reversed	It will display the numbering in reverse order.

Syntax :

```
<ol type="a" start="5">
```

```
<li> item-1 </li>
```

```
<li> item-2 </li>
```

```
</ol>
```

Output : e. item-1

F. Item-2

```
<ol type="1" reversed>
```

```
<li> item-1 </li>
```

```
<li> item-2 </li>
```

```
</ol>
```

Output : 2. item-1

1. Item-2

Example : Nested Ordered List

```
<body>
```

```
<h2>Tutorial</h2>
```

```
<ol>
```

```
<li>HTML
```

```
<ol type="a">
```

```
<li>Normal Elements
```

```
<ol type="I">
```

```
<li>Bold</li>
```

```
<li>Italic</li>
```

```
</li>
```

```
<li>Void Elements
```

```
<ol type="I">
```

```
<li>img</li>
```

```
<li>link</li>
```

```
</ol>
```

```
</li>
```

```
</ol>
```

```
</li>
```

```
<li>Java Script
```

```
<ol type="a">
```

```
<li>Variables
```

```
<ol type="I">
```

```
<li>var</li>
```

```
<li>let</li>
```

```
</ol>
```

```
</li>

<li>Data Types

<ol type="I">

<li>Number</li>

<li>String</li>

</ol>

</li>

</ol>

</li>

</ol>

</body>
```

Nested Data List INPUT

```
<!DOCTYPE html>

<html>

  <body>

    <h2>Tutorial</h2>

    <ol>

      <li>HTML

        <ol type="a">

          <li>Normal Elements

            <ol type="i">

              <li>Bold</li>

              <li>Italic</li>

            </ol>

          </ol>

        </li>

      </ol>

    </ol>
```

```
<li>Void Elements

  <ol type="i">

    <li>Img</li>

    <li>Link</li>

  </ol>

</li>

</li>

</ol>

<li>Java Script

  <ol type="a">

    <li>Variables

      <ol type="i">

        <li>Var</li>

        <li>Let</li>

      </ol>

      <li>Data Types

        <ol type="i">

          <li>Number</li>

          <li>String</li>

        </ol>

      </li>

    </li>

  </ol>

</li>

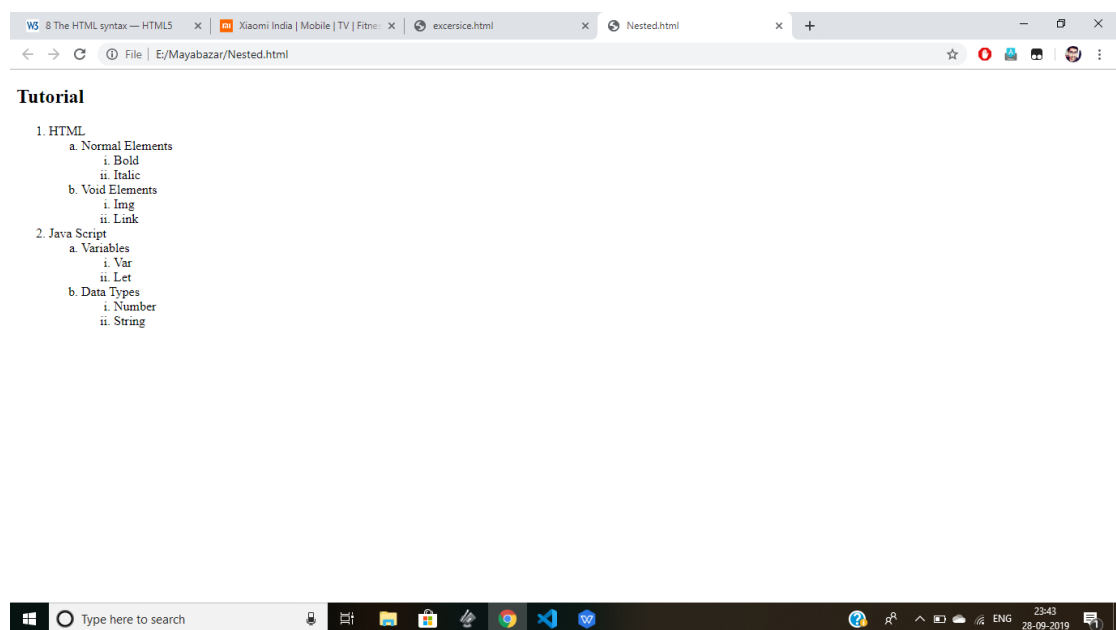
</ol>
```



```
</body>

</html>
```

OUTPUT



Ordered List : INPUT

```
<!DOCTYPE html>

<html>

    <body>

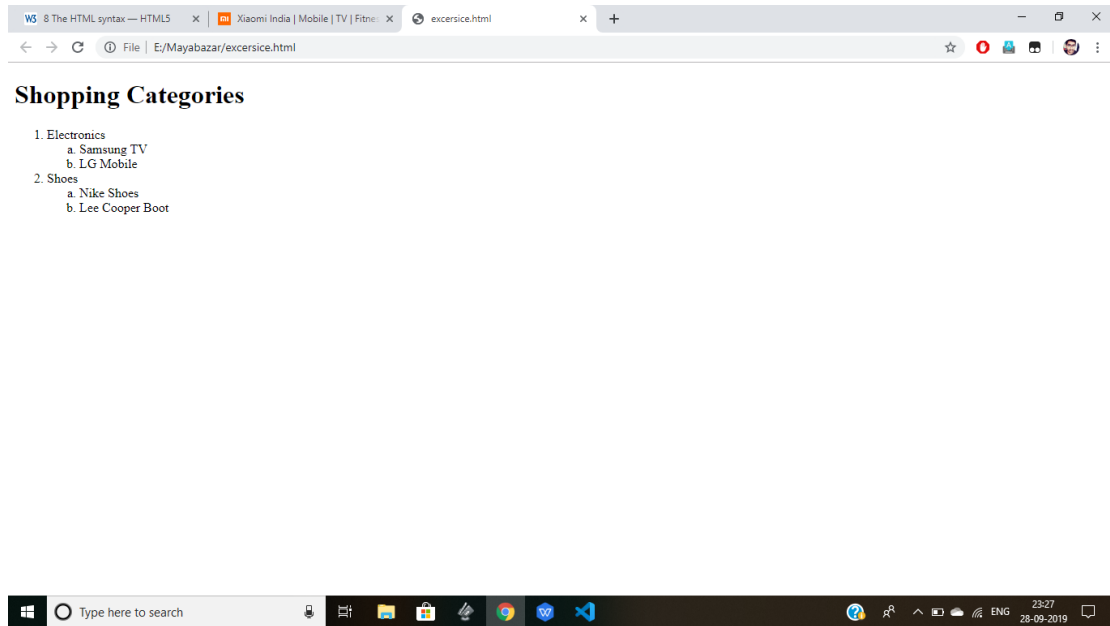
        <h1>Shopping Categories</h1>

        <ol>

            <li>Electronics
```

```
<ol type="a">
    <li>Samsung TV</li>
    <li>LG Mobile</li>
</ol>
<li>Shoes
    <ol type="a">
        <li>Nike Shoes</li>
        <li>Lee Cooper Boot</li>
    </ol>
</li>
</li>
</ol>
```

OUTPUT



Un ordered List :

- It sets a bulleted list in html page.
- It comprises of list items defined in element.
- Bulleted list can define the item type in to disc, circle and square.

Syntax :

```
<ul type="square">
```

```
<li>HTML</li>
```

```
<li>Java Script</li>
```

```
</ul>
```

Example : Nested list with ordered and un ordered items

```
<ol>

<li> HTML

    <ul>

        <li> Normal elements </li>

        <li> Void Element </li>

    </ul>

</li>

<li> Java Script

    <ul>

        <li> Variables </li>

        <li> Data Types </li>

    </ul>

</li>

</ol>
```

Text Formatting in html

The formatting of text includes changing of font, style and effects for text present in a page.

➤ HTML provides several tags that are used for formatting text. They are,

1.

It is used to change the character size, color and face. The attributes are

attribute	description
face	It specifies the font family
Size	It defines the character size in 7 levels. The size increases from 1 to 7.
color	It sets a color for text enclosed in font.

Example :

```
<font size="6" face="Arial" color="blue">
```

Welcome to HTML

```
</font>
```

2. Font Styles

Tag	Description
	Specifies bold letters.
	It is a similar to bold but used in review

	mode.
<i>	Specifies italic letters.
	Emphasized to show italics in review mode.

3. Font effects

Tag	Description
<u>	It underlines the given text.
<ins>	It is used to define inserted text in review mode. Similar to underline.
<strike>	It is used to strike out the given text, which indicates temporary deletion.
	It is used to strike out the text, which specified permanent deletion in review mode.
<sup>	It is used to raise the character position from base line.
<sub>	It is used to lower the character position from base line.

Example :

5th October

H₂O

Images In HTML :

HTML provides options to display and handle images in a Web Page. The major type of images supported on internet are,

Image type	Description
.png	<ul style="list-style-type: none">➤ Portable Network Graphics.➤ High resolution and high definition.➤ Have more pixel depth.➤ Good to zoom and view.➤ Mostly used for download and use.➤ Occupy more space.
.jpg/jpeg	<ul style="list-style-type: none">➤ Joint Photographic Expert Group.➤ High definition but less resolution.➤ Have less pixel depth.➤ Not suitable to zoom and view.➤ It is a compressed image format. Hence occupies less space.
.gif	<ul style="list-style-type: none">➤ Graphic Interchange Format.➤ Less resolution and low definition.➤ It supports only 256 colors.➤ It can be animated.➤ It is mostly used for buttons, icons, logos, bullets etc.,
.svg	<ul style="list-style-type: none">➤ Scalar Vector Graphics.➤ They are vector based and not pixel based.➤ They have high resolution and definition.

	➤ Mostly used for Architecture diagrams.
--	--

Image Tag and its Attributes :

The tag is used to embed image into Web Page. It is a void element. Hence it doesn't require an end tag.

Attribute	Description
alt	It specifies the alternative text to display when image fails to load.
src	It gets the path and file name which is used to display as image source.
Height, width	It specifies the width and height of image in pixels (or) in percent. The image size in percent will fit the image according to browser size.
hspace	It defines the horizontal space between image and text.
vspace	It defines the vertical space between image and text.
border	It sets a border for image by specifies pixels.
align	It sets left and right align for image so that text can wrap beside image.

Syntax :

```

```


Advanced Attributes :

Attribute	Description
crossorigin	<p>It is used to define CORS settings [cross origin resource sharing]. it is an attribute that specifies whether to allow resources from external servers.</p> <p>Values : Anonymous</p> <p>Use-credentials</p>
decoding	<p>It specifies how an image must be decoded. It can use following values,</p> <p>sync</p> <p>async</p> <p>auto</p>
importance	<p>It defines the priority for image so that browser loads images according to priority.</p> <p>Values :</p> <p>low</p> <p>medium</p> <p>high</p>
srcset	<p>It defines the set of images to display when the browser is changing the state i.e image can change according to browser size.</p>

Image size for ads banner :

Google AdSense define standards for displaying images as advertisement banners. The standard size of banners in width*height are shown below,

1. Leader board

728x90 pixels

2. Half Page

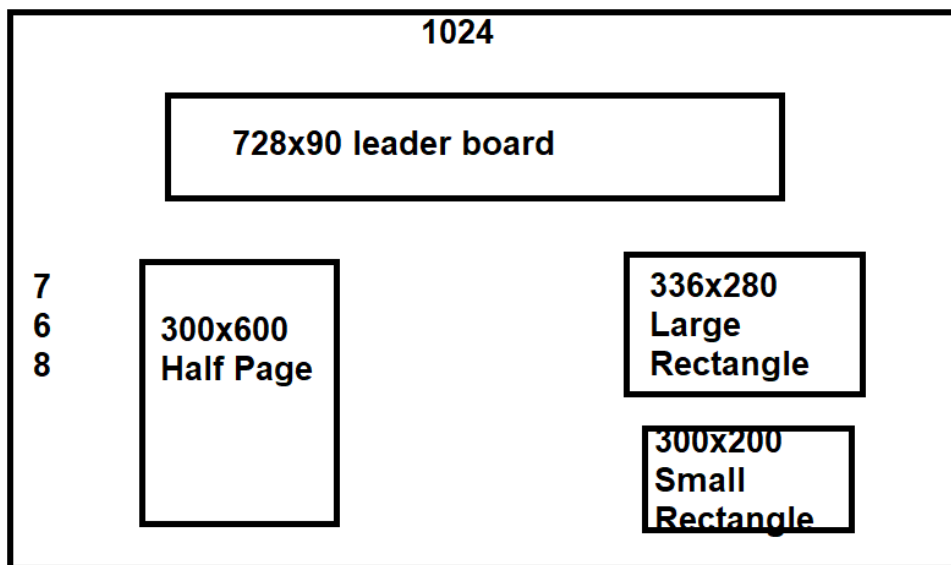
300x600 pixels

3. Large Rectangle

336x280 pixels

4. Medium Rectangle

300x200 pixels



Hyper links in HTML

1. A link is clickable text, picture (or) graphic that navigates the user to any named location when clicked.
2. A hyper link handles navigation by following the standards of http.
3. Hyper links are created in html by using anchor element <a>.

4. Hyper links are classified in to two types,
- a. Intra document links
 - b. Inter document links

Intra document links :

- Intra document link specifies navigation from one location to another within the same page.
- You have to configure a reference id to the elements so that you can access from a hyper link.

Syntax :

`<h2 id="html"> HTML </h2>`

`<figure id="tv">`

`<ol id="list">`

- Anchor <a> element can get and set the id into url by using the attribute href.

Syntax :

` Goto HTML `

` View TV `

` Read List `

Example :

`<body>`

`<h1 id="toc"> Table of Contents </h1>`

```
<ol>

  <li><a href="#html"> HTML </a></li>

  <li><a href="#css"> CSS </a></li>

  <li><a href="#js"> JavaScript </a></li>

</ol>

<h2 id="html">HTML</h2>

<a href="#toc">Back to Top</a>

<p> Your Text..</p>

<h2 id="js">JavaScript</h2>

<a href="#toc">Back to Top</a>

<p> your text... </p>

</body>
```

Inter document links :

- An inter document link defines navigation to any file (or) url.
- The target file opens in a browser only when the relative plug-ins are supported.
- If the plugin is not available then browser will download the target file.

Example :

```
<ol>

  <li><a href="tutorial.html">HTML Tutorial</a></li>

  <li><a href="../images/3.jpg">Range Rower</a></li>

  <li> Email:<a href="mailto:hr@nareshit.in">hr@nareshit.in</a></li>
```

```
<li><a href="http://www.w3.org"> W3C Official Site</a></li>

<li><a href="../Docs/cssdemo.pdf"> View CSS Tutorial</a></li>

<li><a href="../Docs/Collections.doc">Collections Document</a></li>

</ol>
```

FAQ

1. How to open link target in new tab ?

A) By defining "target=_blank"

```
<a href="../images/3.jpg" target="_blank"> Range rowler </a>
```

2. How to open a link target in a new window ?

A) By using java script window.open()

```
<a href="javascript:window.open('../images/3.jpg', 'range rowler', 'width=400
height=300')"> range rowler </a>
```

3. How to open the link target in same window without disturbing existing content ?

A) By using

-Frames

-Iframe

Frames can split the window in to horizontal and vertical panes. Every frame designates a window and display a document.

Frames are defines <frameset> by using <frame> element.

Syntax :

```
<frameset cols="30%70%">  
  
<frame name="" src=""> </frame>  
  
<frame name="" src=""> </frame>  
  
</frameset>
```

1. Add following pages in to project

- shopping.html
- electronics.html
- shoes.html
- leftpanel.html

2. Shopping.html [source]

```
<!DOCTYPE html>  
  
<html>  
  
<head><title>Amazon Shopping</title></head>  
  
    <frameset cols="30%70%" noresize>  
  
        <frame name="frameleft" src="leftpanel.html"></frame>  
  
        <frame name="frameright" src=""></frame>  
  
    </frameset>  
  
</html>
```

3. Leftpanel.html [source]

```
<div>
```

```
<h2>Amazon Shopping</h2>

<ol>

  <li><a href="electronics.html" target="frameright">Electronics</a></li>

  <li><a href="shoes.html" target="frameright">shoes</a></li>

</ol>

</div>
```

4. electronics.html [source]

```
<style>

.product {

  border:2px;

  Padding:10px;

  Margin:10px;

  Background-color:antiquewhite;

}

</style>

<div>

  <h2>Electronic Products</h2>

  <div class="product">

    <figure>

      
```

```
<figcaption>Samsung TV : ₹ 45,000/-</figcaption>

</figure>

</div>

<div class="product">

<figure>

    <figcaption>LG Mobile : ₹ 15,000/-</figcaption>

</figure>

</div>

</div>
```

5. Shoes.html [source similar to electronics]

iframe :

It is a new element introduced in to html 5, which allows to embed the contents of an existing website (or) page in to the current page. So that the target page information can be broadcast-ed directly in to the current page.

Syntax :

```
<iframe name="" src="" width="" height=""></iframe>
```

Example :

Index.html

```
<body>

  <header style="text-align: center">

    <nav>

      <menu>

        <a href="electronics.html" target="framebody">Electronics</a>

        <span>|</span>

        <a href="shoes.html" target="framebody">Shoes</a>

      </menu>

    </nav>

  </header>

  <section>

    <iframe width="800" height="300" name="framebody"
style="border-style:none"></frame>

  </section>

</body>
```

How to Embed Youtube videos ?

1. Open any youtube channel and select a video.
2. Copy the video URL from address bar.
3. Create a new HTML Page with iframe src as video URL changed to "embed"

Example :

```
<iframe src="https://www.youtube.com/embed/some code" width="500"
height="300" style="border-style:none"></iframe>
```

Tables in HTML

Tables are used to organize information in to rows and columns.

HTML provides the following elements to configure and design table.

Element	Description
<table>	It encapsulates a set of rows and columns.
<caption>	It sets a caption for table, which is used in SEO to identify the table.
<thead>	It defines the head section of table.
<tbody>	It defines the body section of table.
<tfoot>	It defines the footer section of table.
<th>	It is used to define a table header i.e the column heading.
<tr>	It specifies table row.
<td>	It specifies table cell content.
<colgroup>	It groups a set of columns inorder to apply effects.

Example :

```
<body>
```

```
<div>
```

```
<table>
```

```
<caption>Products List</caption>

<colgroup span="2" style="background-color:yellow"></colgroup>

<thead>

    <th>Product Id</th>

    <th>Name</th>

    <th>Price</th>

</thead>

<tbody>

    <tr>

        <td>1</td>

        <td>Samsung TV</td>

        <td>45000.66</td>

    </tr>

    <tr>

        <td>2</td>

        <td>Mobile</td>

        <td>12000.44</td>

    </tr>

</tbody>

<tfoot>

<tr>

<td>&nbsp;</td>

<td>&copy; copyright 2019</td>
```

<td> </td>

</tr>

</tfoot>

</table>

</div>

</body>

Table Attributes

1. Border, Frame and Rules :

Attribute	Description
frame	It sets a frame for table.which includes box, rhs, lhs, above, below and void.
rules	It applies rules for row, Cols and groups.
border	It sets a border for cell.

Syntax :

<table frame="box" rules="all">

<table frame="void" rules="groups">

<table frame="void" border="2">

<table frame="box" border="2">

Note :

Rules cannot use groups if table doesn't have groups.

Border can be applied only when rules are not defined.

2. Cell Spacing and Cell Padding :

Attribute	Description
cellspacing	It sets space between cells.
cellpadding	It sets space between the cell border and its content

Note :

Spacing and padding can be observed only when border is defines.

Syntax :

`<table border="2" frame="void" cellspacing="10" cellpadding="5">`

3. Height and Width :

Attribute	Description
height	It specifies the height in pixels. You can define height for table and tr.
width	It specifies width in pixels. You can define width for table, th and td.

Syntax :

```
<table width="400" height="100">  
  
<tr height="200">  
  
<th width="400">  
  
<td width="400">
```

Example :

1. Add a new page

"home.html"

```
<head><title>Home</title></head>  
  
<body>  
  
<table border="1" cellspacing="15" cellpadding="15" width="1024">  
  
<tr height="100">  
  
<td colspan="3">  
  
<h1 align="center">  
  
Online UI Tutorial</h1>  
  
</td>  
  
</tr>  
  
<tr height="300">  
  
<td width="200">  
  
<font size="5">  
  
<ol>
```

```
<li><a href="html.html" target="frameBody">HTML</a></li>

<li><a href="css.html" target="frameBody">CSS</a></li>

<li><a href="js.html" target="frameBody">JavaScript</a></li>

</ol>

</font>

</td>

<td width="700">

  <iframe style="border-style: none;" name="frameBody" width="700"
height="300"></iframe>

</td>

<td width="100">

</td>

</tr>

<tr height="100">

  <td colspan="3">

    <div align="center">

      <i><b>&copy; copyright 2019</b></i>

    </div>

  </td>

</tr>

</table>

</body>
```

4. align and valign :

Attribute	Description
align	It aligns the content horizontally left, center (or) right.
valign	It aligns the content vertically top, center (or) bottom

Syntax :

`<table align="center">`

`<tr align="center">`

`<td align="right">`

`<td align="center" valign="top">`

5. Table Background :

Attribute	Description
bgcolor	It sets a background color for table, group,row (or) cell.
background	It sets a background image for table, group, row (or) cell.

Syntax :

`<table bgcolor="yellow">`

`<thead bgcolor="yellow">`

`<table background="../images/logo.png">`

6. Merging rows and columns :

Attribute	Description
colspan	It merges specified number of columns in to a single cell.
rowspan	It merges specified number of rows in to single cell.

Example :

```
<body>
```

```
  <table border="1" width="400">
```

```
    <thead>
```

```
      <tr>
```

```
        <td rowspan="3" align="center">Head section</td>
```

```
      </tr>
```

```
    <tr>
```

```
      <th colspan="2">Name</th>
```

```
      <th colspan="3">Address</th>
```

```
    </tr>
```

```
  <tr>
```

```
    <td>First Name</td>
```

```
    <td>Last Name</td>
```

```
    <td>City</td>
```

```
    <td>State</td>
```

```
    <td>Postal code</td>
```

```
</tr>

</thead>

<tbody>

<tr>

<td rowspan="3" align="center">Body Section</td>

</tr>

<tr>

<td>Raj</td>

<td>Kumar</td>

<td>Chennai</td>

<td>TN</td>

<td>600076</td>

</tr>

<tr>

<td>Raj</td>

<td>Kumar</td>

<td>Chennai</td>

<td>TN</td>

<td>600076</td>

</tr>

</tbody>

<tfoot>

<tr>
```

```
<td rowspan="2" align="center">Footer Section</td>

</tr>

<tr>

<td colspan="5" align="center">&copy; Copyright 2019</td>

</tr>

</tfoot>

</table>

</body>
```

Example :

Nested Table

```
<body>

  <table width="400" cellspacing="10" cellpadding="10" border="1"
  align="center">

    <caption>Products Catalog</caption>

    <tr align="center">

      <td>

      </td>

      <td>

        <table frame="box" rules="all" width="300" height="150">

          <colgroup span="1" style="background-color:yellow;"></colgroup>

        <tr>
```

```
<td>Name</td>
```

```
<td>Samsung TV</td>
```

```
</tr>
```

```
<tr>
```

```
<td>Price</td>
```

```
<td>56000.66</td>
```

```
</tr>
```

```
</table>
```

```
</td>
```

```
</tr>
```

```
<tr align="center">
```

```
<td>
```

```

```

```
</td>
```

```
<td>
```

```
<table frame="box" rules="all" width="300" height="150">
```

```
<colgroup span="1" style="background-color:yellow;"></colgroup>
```

```
<tr>
```

```
<td>Name</td>
```

```
<td>Nike casuals</td>
```

```
</tr>
```

```
<tr>
```

```
<td>Price</td>
```

```
<td>4200.55</td>
```

```
</tr>
```

```
</table>
```

```
</td>
```

```
</tr>
```

```
</table>
```

```
</body>
```

Forms in HTML

Form provides an UI that allows the user to interact with the data I.e input, view, edit, delete etc.,

- A form comprises of collection of elements which includes buttons, text box, checkbox, drop down list etc.,
- A form is defined in a document by using the <form>.
- The form elements are defined by using the following tags,

```
<input>
```

```
<select>
```

```
<datalist>
```

```
<option>
```

```
<optgroup>
```

```
<meter>
```

<progress>

<textarea>

The <form> Tag :

It is a container that encapsulates the set of elements and submit their to specified server side source.

| Attribute | Description |
|------------|--|
| id | It defines a unique identification for the form in the page. |
| name | It defines a reference name for the form element. |
| class | It defines the css class for form element. |
| method | It specifies the request type to be handle by the form. It can be GET (or) POST. |
| action | It specifies the target location where the form data need to be submitted. |
| novalidate | It is used to disable the default HTML validations in a form. |
| target | It specifies the action target which can submit with in the same window (or) a new window. |

Syntax :

```
<form name="frmRegister" method="POST" action="welcome.aspx" novalidate  
target="_blank">
```

.....form elements.....

```
</form>
```

| GET | POST |
|--|--|
| 1. Data will be arranged in HTTP header by appending to the URL as query string. | Data will be arranged in HTTP message body [form body]. |
| 2. Data is in query string so user can view the data. | Not visible to user. |
| 3. Less secured compared to POST method because data is in query string so it will be saved in browser history and webserver logs. | Bit safer than GET method because data is saved in history (or) web server logs. |
| 4. As data is saved in URL so its saves only 2048 chars. | Can be used for any amount of data. |
| 5. Can be bookmarked. | Can't bookmarked. |
| 6. Hacking will be easy. | Hacking is difficult. |
| 7. Only ASCII character data type allowed. | No restriction. Allows binary data also. |
| 8. Caching is possible. | No caching. |
| | |

<fieldset> and <legend> :

A <fieldset> sets a frame for group of elements in a form and a <legend> will set a caption for frame.

Syntax :

<fieldset>

<legend> Personal Details </legend>

.....form elements.....

</fieldset>

The <input> Element :

It is a form element that allows to input any value from the UI. It can handle various types of values for different type of situations. The commonly used attributes are,

1. **Id** : it specifies a unique reference id for input element.

Syntax : `<input id="txtName">`

2. **name** : it specifies a name for input element so that form can submit its value to server. If name not defined then form cannot submit its value.

Syntax : `<input name="txtName">`

3. **class** : it defines a CSS class to input element so that you can customize its appearance.

Syntax : `<input class="cssClassName">`

4. **value** : It defines the default value to be used while submitting the form.

Syntax : `<input name="txtAge" value="22">`

5. **type** : it specifies the data type to allow in the input field. HTML-5 introduced several new data types which can restrict the input value it includes text, password, number, date, year, month, file, range, color, url, email etc.,

Syntax :

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

Password : `<input type="password">`

Age : `<input type="number">`

Email : `<input type="email">`

Photo : `<input type="file">`

Note : The new input types of HTML 5 are browser dependent. Hence will not have the same behaviour across browsers.

6. **Minlength** : it restricts the minimum length of characters in a string input.

7. **Maxlength** : it restricts the maximum length of characters in a string input.

Syntax :

```
<input minlength="4" maxlength="10" type="password" name="txtPwd">
```

8. **Min**

9. **Max** these are the attributes used to define min and max numeric range constraints in a number input.

Syntax :

```
<input min="15" max="30" type="number" name="txtAge" value="22">
```

10. **placeholder** : it defines the water mark text to display for input field when there is no value defined.

Syntax :

```
<input type="password" placeholder="min 4 and max 10 chars" minlength="4"
maxlength="10">
```

11. **autofocus** : it sets a focus to any input element in a form so that user can start input immediately after loading the form.

Syntax :

```
<input type="text" name="txtName" autofocus>
```

12. **size** : it is used to set width for text box the default width is '20'.

Syntax :

<input size="3" type="text" name="txtCvv">

13. **required** : it is used to define mandatory fields in a form so that form will not submit until the field is defined with a value.

Syntax :

<input size="3" type="text" name="txtCvv" required>

14. **pattern** :

- it is an validation attribute used to verify the format of input value by matching it against a regular expression.
- The regular expressions are built by using meta characters and quantifiers.

| Meta character | Description |
|----------------|--|
| ? | Specifies zero (or) one occurrence of given character. |
| + | Specifies one (or) more occurrence of given character. |
| * | Specifies zero (or) more occurrence of given character. |
| \ | Specifies the precedence of any entity. |
| \^ | Starts with |
| \$ | Ends with |
| w | Specifies an word with lowercase, uppercase, number and underscore [_]. word must start with alphabet. |
| W | Specifies non word can start with number (or) underscore. |
| d | Numeric with decimal places. |
| D | Number without decimal places. |
| s | Blank space allowed. |
| S | Blank space not allowed. |

| | |
|----------------------|---|
| [a-z] | Only lower case allowed. |
| [A-Z] | Only upper case allowed. |
| [a-zA-Z] | Both upper and lower case allowed. |
| [a-Z] | Both upper and lower case allowed. |
| [0-9] | Only numeric allowed. |
| [a-zA-Z0-9] | Alpha numeric |
| [a,d,s] | Only specified characters allowed. |
| [^a,d,s] | Excluding specified characters all are allowed. |
| [a-mA-M4-9] | Only characters within specified range. |
| \+ \- \. \. \. \. \. | Only individual special characters. |
| [!@#%&] | All special characters. |
| (?=.*[A-Z]) | Atleast 1 uppercase letter. |
| (?=.*[!@#%&]) | Atleast 1 special character. |

| Quantifier | Description |
|------------|-----------------------------------|
| {n} | Exactly n - number of characters. |
| {n,m} | Minimum-n and maximum-m. |
| {n, } | Minimum-n and maximum-any. |

Syntax :

```
<input type="text" name="txtMobile" placeholder="+91 10digits number"
pattern="\+91[0-9]{10}">
```

15. list : it is a new HTML property used to define a data list of any input element.

So that the options of data list will be displayed during input.

Syntax :

```
<div>

    <input type="text" name="txtSearch" size="40" list="terms">

    <datalist id="terms">

        <option>HTML Tutorial</option>

        <option>Java script Examples</option>

        <option>Java tutorial</option>

        <option>HTML Examples</option>

    </datalist>

</div>
```

Radio Buttons :

- The radio buttons are form elements that allow the user to select one (or) multiple options from a group of choices.
- Radio's once checked cannot be unchecked. Hence radio's must be used with MUTEX mechanism [Mutual Exclusion].
- To define MUTEX all radio's in a group must be defined with same name.
- If MUTEX is defined for radio's then it will allow the user to select any one option from a group of choices.

Attributes -

1. id
2. name

3. class
4. checked
5. value

Note : The radio's submit the value ON when the value is not defined.

Syntax :

```
<input type="radio" name="" value="" checked>
```

Example :

```
<body>
```

```
<form>
```

```
<fieldset>
```

```
<legend>Choose Gender</legend>
```

```
<input type="radio" value="male" name="gender" checked>Male
```

```
<input type="radio" value="female" name="gender">Female
```

```
<br>
```

```
<button>Submit</button>
```

```
</fieldset>
```

```
</form>
```

```
</body>
```

Check Box :

- A check box is same like a radio button which allows to select one (or) multiple options but it can check (or) uncheck the option without MUTEX.

Properties :

1. Is
2. Name
3. Class
4. Value
5. Checked

Syntax :

```
<input type="checkbox" name="" value="" checked>
```

Example :

```
<body>
```

```
<form>
```

```
<fieldset>
```

```
<legend> Your Hobbies</legend>
```

```
<input type="checkbox" name="hobbies" value="Music" checked>Music
```

```
<input type="checkbox" name="hobbies" value="movies" checked>Movies
```

```
<br>
```

```
<button>Submit</button>
```

```
</fieldset>
```

```
</form>
```

```
</body>
```

Drop down List (or) Combo Box :

- A drop down list provides an UI from where it will allow the user to select only one option from a group of choices.
- Drop down's are designed y using following elements,

| Element | Description |
|------------|--|
| <select> | It creates an empty dropdown list. |
| <option> | It defines an option for dropdown list. |
| <optgroup> | It groups a list of options under one category and defines a group name. |

| Option Property | Description |
|-----------------|--|
| text | It specifies the text to display in list. |
| value | It specifies the value to submit when option selected. |
| disabled | It will not allow to select option. |
| selected | It makes the option selected automatically on page load. |

Example :

```
<body>

<form>

<fieldset>

<legend>Select a Product</legend>

<select name="lstproducts">

<option>Select Your Product</option>

<optgroup lable="Electronics">

<option value="1001">Samsung TV</option>

<option value="1002" disabled>Mobile</option>

</optgroup>
```

```
<optgroup lable="shoes">  
  
<option value="2001">Nike Casuals</option>  
  
<option value="Lee Cooper Boot">Lee Cooper Boot</option>  
  
</optgroup>  
  
</select>  
  
<button>Submit</button>  
  
</fieldset>  
  
</form>  
  
</body>
```

List Box :

- A list box is similar to a dropdown in elements used to designed the UI i.e it uses

```
<select>  
  
<option>  
  
<optgroup>
```

- You can transform the behaviour to list box by defining the attributes

- A. size
- B. multiple

Example :

```
<fieldset>  
  
    <legend>Select City</legend>  
  
<select size="2" multiple>
```



```
<option>Delhi</option>  
  
<option>Hyd</option>  
  
<option>Chennai</option>  
  
</select>  
  
</fieldset>
```

Text area :

- It is an RC Data element used to present multi line text.
- It will not allow any formats for text.

Syntax :

```
<textarea rows="4" cols="40">  
  
.....your text is necessary.....  
  
</textarea>
```

Note : You can disabled, and set readonly for text.

Syntax :

```
<textarea rows="3" cols="40" disabled>
```

Terms of Service

```
</textarea>
```

Meter Control :

- It is a new HTML control used to display grade meter with various effects.
- The properties used with meter control are,

| Property/attribute | Description |
|--------------------|--|
| min | It specifies the lower bound value. |
| max | It specifies the upper bound value. |
| value | It specifies the default value. |
| low | It defines a value above min and below high. |
| high | It defines a value above low and below max. |

Syntax :

```
<meter min="1" max="100" value="100" low="0" high="0"></meter>
```

Example :

```
<body>
```

```
<fieldset>
```

```
<legend>Ratings Meter</legend>
```

```
<dl>
```

```
<dt>Normal Meter</dt>
```

```
<dd>
```

```
<meter></meter>
```

```
</dd>
```

```
<dt>Good 100%</dt>
```

```
<dd>
```

```
<meter min="1" max="100" value="100"></meter>
```

```
</dd>
```

```
<dt>Good 40%</dt>
```

```
<dd>

<meter min="1" max="100" value="40"></meter>

</dd>

<dt>Average</dt>

<dd>

<meter min="1" max="100" value="100" low="40" high="80"></meter>

</dd>

<dt>Poor</dt>

<dd>

<meter min="1" max="100" value="100" low="60" high="80"></meter>

</dd>

</dl>

</fieldset>

</body>
```

Progress Control :

- It is a form element used with Ajax calls.
- Ajax defines asynchronous tasks in a page.
- It allows partial post back i.e only a portion of page is posted to server.
- Progress control can show the status of any task performed within the page. It includes downloading, uploading, copying, installing etc.,

Syntax :

```
<progress min="1" max="100" value="100"></progress>
```

Example :

```
<body>

<fieldset>

<legend>Downloading File</legend>

<dl>

<dt>Preparing For Download</dt>

<dd>

<progress></progress>

</dd>

<dt>10% Downloaded</dt>

<dd>

<progress min="1" max="100" value="10"></progress>

</dd>

<dt>80% Downloaded</dt>

<dd>

<progress min="1" max="100" value="80"></progress>

</dd>

<dt>Download Completed</dt>

<dd>

<progress min="1" max="100" value="100"></progress>

</dd>

</dl>

</fieldset>
```

</body>

HTML MultiMedia and Animations

1. Marquee : It is used to define scrolling and sliding contents in a page. You can optionally control the scrolling and sliding animation by using following attributes,

Attribute	Description
scrollamount	It defines the scrolling speed.
direction	It defines the scrolling direction left, right, up (or) down.
width, height	Sets width and height for content in marquee.
bgcolor	Sets a background color.
behaviour	Changes the scrolling behaviour to alternate.
loop	It specifies the number of times marquee content have to display.

Syntax :

```
<marquee scrollamount="15" direction="right" width="400" height="50"
bgcolor="yellow" loop="3">
```

.....your content.....

```
</marquee>
```

```
<marquee behaviour="alternate">
```

.....your content.....

```
</marquee>
```

Flash

License number WPD800-59935-59132-35408

- It is an animation software used for creating 2D and 3D animations.
- It provides publishing features for web so that you can create and publish animations for web.
- It also supports java script to define queries for animation and make it dynamic.

Flash Workspace :

1. Tool Bar

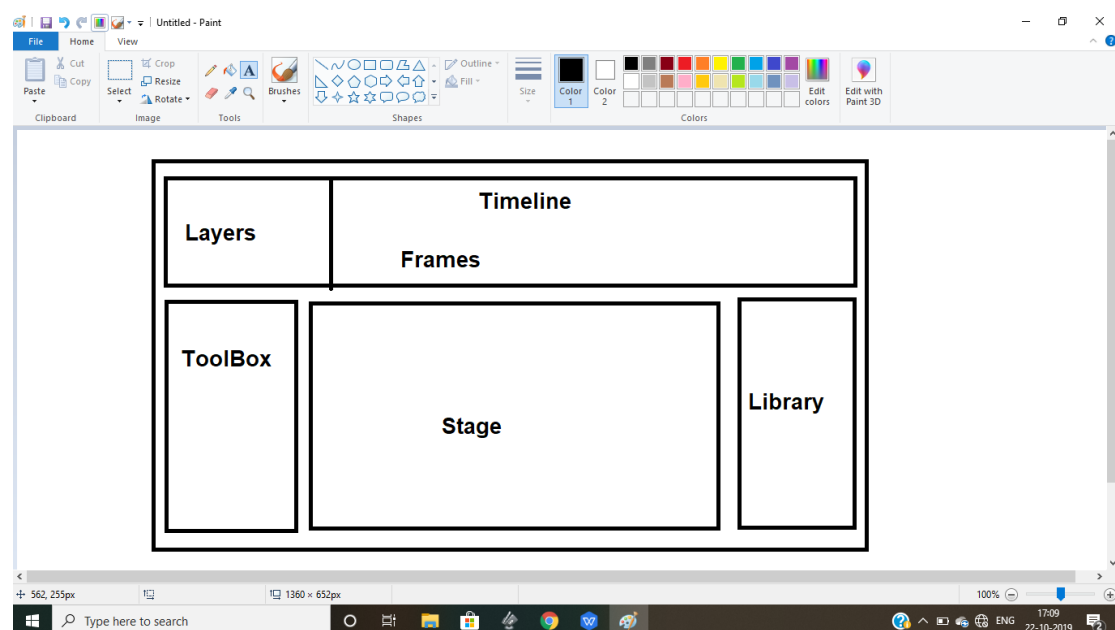
2. Stage

3. Timeline

A) Layers

B) Frames

4. Library



Creating a Flash Document :

1. Open Flash Software
2. File Menu → New → Flash Document
3. Right click on “stage” → select “Properties”

Title : Leaderboard

Description : any..

Width : 728 px

Height : 90 px

Frame/sec : 24

Flash Tools : [Tool Box]

Tool	Description
Select	To select and move object.
Subselection	To select node and change shape.
Transform	To select, resize and rotate.
Transform Gradient	To transform gradient colors.
Line	Straight line.
Lasso	Free form selection.
Pen	Draw by joining end points.
Text	Inserts Text.
Oval	Draws oval

Rectangle	Rectangle
Pencil	Draw thin line.
Brush	Draw thick line with brush shapes.
Fill color	Fills an area with color.
Eye Dropper	Picks a color from picture.
Eraser	Erases a portion of picture.

Animations in Flash

➤ Flash can create two type of animations,

1. Frame by Frame
2. Tweening

Frame by Frame Animation :

1. Draw an object on stage at Frame-1
2. Insert a Keyframe for Frame-1 with “F6” function key.
3. Change the position (or) size of object in Frame-2.
4. Insert Keyframe for Frame-2 [F6].
5. Change position of size of object.
6. Repeat the steps to complete animation.
7. Press “ctrl + Enter” to test animation.

Tweening Animation :

In tweening animation you have to define the key frames at random positions from begin to end.

Flash will create the frames with animation between the start and end points.

The tween animation is categorized in to two types,

1. Motion tween
2. Shape tween

Motion tween :

In motion tween you can change the position, size and orientation of any object.

Example :

1. Create a new flash document.
2. Rename Layer1 as "oval".
3. Draw an Oval at Frame-1.
4. Select Frame-10 and insert keyframe "F6".
5. Change the position of size of oval at Frame-10.
6. Similarly insert keyframe at 20, 30 & 40 and change position (or) size of oval at every key frame.
7. Select all frames from 1 to 40.
8. Right click on frames and choose the option "Create Motion Tween".

Adding a motion guide :

A motion guide allows you to define your own path for moving an object.

Example :

1. Add a new text layer and insert text.
2. Insert key frame at 50 and choose motion tween.[right click create motion tween]
3. Right click on text layer and choose the option
“ Add Motion Guide “
4. This will add a guide layer for text.
5. Add a circle in guide layer, remove fill color, break the circle at any position using eraser.
6. Keep
Text Frame-1 at starting point of guide
Text frame-50 at ending point of guide
7. Select TextObject and open properties [bottom of stage].
8. Select the option “orient to path”.

Mask Effect :

1. Add a new layer by name
“Car”
2. Insert car image
File → import → import to library
Choose image from your computer and add to library.
Drag and drop in to stage from library.
Go to properties and set size and position.
Width : 550
Height : 400

X : 0.0

Y : 0.0

3. Insert a new layer : oval
4. Draw an oval and set motion tween for oval.
5. Up to 50 frame car uses static “F5”
6. Up to 40 frame oval uses key frame “F6”
7. Right click on oval layer and select

“Mask”

Note : Masking Layer must be above the background (or) object.

Flash Symbols :

A symbol is used for creating reusable objects which you can import and use in any scene.

The symbols are categorized in to 3 types

- A) Graphic symbol → images without animations
- B) Movie symbol → image (or) object with animation
- C) Button symbol → for buttons like OK, Cancel, Login etc.,

Example :

1. Goto “insert menu → New Symbol”
2. Select “Movie Symbol”
3. Insert any object and animate with Frame by Frame (or) tweening
4. Go back to “scene” and insert symbol from library.

Scenes in Animation

1. A movie comprises of collection of scenes that are displayed in a sequence and executed one after another.

Example :

1. Create a new flash project.
2. By default you are in Scene-1.
3. To insert a new scene

Insert menu → scene

4. You can change animation in every scene.
5. To change between scenes select the option “edit scene”.

Shape tween :

It is a tweening animation that can transform an object in to another object. You can transform any object only after break apart.

Example :

1. Insert an object in frame-1.
2. Insert a blank key frame at frame-20 [F5].
3. Add a new object at 20 [break apart- if it is a symbol].
4. Select frames from 1 to 20.
5. Go to properties and select tween animation as shape.

Publishing a flash animation :

1. Save your flash document

“diwali fla”

2. Go to file menu → export → export movie

3. Save in your website physical path

“diwali.swf”

Embed flash animation in to web page :

1. Go to your web page

“index.html”

2. Embed flash animation

<div align="center">

<embed src="diwali.swf" width="360" height="280">

</div>

Styles in HTML

Styles are set of attributes defined for HTML elements to make the presentation more interactive and responsive.

Styles can be defined to HTML elements by using following methods,

1. Inline
2. Embed
3. CSS [Cascade Style Sheet]

Inline :

In this method the styles are defined for every element individually by using a style attribute. These effects are not accessible to other elements.

Syntax :

```
<body>

<h2 style="background-color:red; color:white; text-align:center">HTML</h2>

</body>
```

Embed :

In this method the styles are defined in page by using a <style> element, so that the styles are accessible to any element in page.

Syntax :

```
<head>

<style>

h2 {

    Background-color:red;

    Color:white;

    Text-align:center;

}

</style>

</head>

<body>

<h2>HTML</h2>
```

<h2>CSS</h2>

<h2>JavaScript</h2>

</body>

CSS [Cascade Style Sheet] :

In this method the styles are maintained in a separate style sheet as external file, so that you can access the styles and use in any page. However, using an external file will increase the number of requests to a page and the load time.

Syntax :

1. Create a new folder by name “Styles”
2. Add a new file in to a folder by name “effects.css”

Effects.css

```
h2 {  
  
    Background-color:red;  
  
    Color:white;  
  
    Text-align:center;  
  
}
```

3. Link to your HTML page

```
<head>  
  
<ink rel="stylesheet" href="../styles/effects.css">  
  
<body>  
  
<h2>HTML</h2>  
  
<h2>CSS</h2>  
  
<h2>JavaScript</h2>  
  
</body>
```

Minification

Minification is a technique used for CSS and JavaScript files to reduce their size.

Minification will remove the blank spaces, line breaks and will use all possible shortcuts to reduce the file size.

Some of the popular minification tools are

1. Smalify
2. CSS minifier
3. js minifier
4. ajax minifier

Note :

Always use the minified files in production and the uncompressed files in development because minified files cannot track the error location if any exception occurred.

CDN [Content Distribution Network] :

The CSS files are maintained in a separate repository server and can be accessed and used in any project remotely this mechanism is known as CDN.

Example :

```
<link rel="stylesheet" href="http://localhost/cdn/effects.css">
```

Style Syntax

1. Inline Style Syntax : In this method the style attributes and values are applied to any element by using "style" attribute.

```
<div style="attributeName:value; attributeName:value"></div>
```

2. Embedded (or) CSS : In this method the styles are applied to elements based on the selectors.

```
<style>
```

```
    Selector
```

```
{
```

```
    attributeName:value;
```

```
    attributeName:Value;
```

```
}
```

```
</style>
```

The selectors used in basic CSS are categorized in to following types-

1. Type selector
2. ID Selector
3. Class Selector
4. Attribute Selector
5. Decendent/Child Selector

Type Selector : It directly specifies the tag name to which the given effects are applied. In this method whenever same tag is added in to page it is applied with the effects. You cannot ignore for any specific element.

Example :

```
<style>

h2, p {

    Text-align:center;

    Color:red;

}

</style>
```

ID Selector : An ID selector is defined by using “#” reference and it is accessible by using the attribute ID. You can apply effects only for required elements. However every element can be defined with only one ID.

Example :

```
<style>

#effects {

    Text-align:center;

    Color:red;

}

</style>

<body>

<h2 id="effects"> HTML </h2>

<p id="effects"> Some text </p>

</body>
```

Class Selector : It is defined by using “.” and can be accessed by using the attribute class. It allows multiple classes for any single element.

Example :

```
<style>

.backEffects {

    Background-color:red;

}

.textEffects {

    Text-align:center;

    Color:red;

}

</style>

<body>

    <h2 class="backEffects textEffects"> HTML </h2>

    <p class="backEffects"> Some text </p>

</body>
```

Attribute Selector : It defines effects to any element based on specific attribute and its value. So, that effects can be applied to elements that are using the given attribute value.

Example :

```
<style>

Input[type="text"]{

    Text-align:center;

    Background-color:red;

}
```

```
</style>

<body>

<form>

Name: <input type="text">

      <input type="button" value="submit">

</form>

</body>
```

Decendent Selector : It is used to apply effects based on the child element under specific parent element. It is mostly used when same type of child element is available for different HTML elements.

Example :

```
<style>

    ol > li {

        Color:red;

    }

</style>

<body>

<ol>

<li>item-1</li>

<li>item-2</li>

</ol>

</body>
```

Meta Characters in Selectors :

Character	Description
^	It applies effects to the element whose attribute value starts with specified text.
\$	It applies effects to the element whose attribute value ends with specified text.
*	It applies effects when specified value occurs at any position.

Example :

```
<style>
```

```
Input[class^="effects"]{
```

```
    Color:red;
```

```
}
```

```
Input[class$="effects"]{
```

```
Color:green;
```

```
}
```

```
</style>
```

```
<body>
```

```
<p class="Texteffects"> Para-1 </p>
```

```
<p class="effectsText"> Para-2 </p>
```

```
</body>
```

Pseudo Selectors : the pseudo selectors are used to change the appearance of existing elements by using a pseudo name. They are defined by using “:” along with element name (or) class name.

1. `:link {}` - It is used to change the colour of hyper link if it is not visited.
2. `:visited {}` - It is used to change the colour of visited link.

Note : Effects can be background, border, colour (or) any other related to appearance.

Example :

```
<head>

<style>

    a:link {

        Color: black;

    }

    a:visited {

        Color:black;

    }

</style>

</head>

<body>

<a href="../about.html">Home</a>

<span>|</span>

<a href="../demo.html">CSS Page</a>

</body>
```

3. **:hover** - It specifies the actions to perform and effects to apply and then the mouse pointer is over any element.

Syntax :

```
element : hover {  
  
}
```

Example :

```
<head>
```

```
<style>
```

```
img {  
  
    width: 100px;  
  
    height: 100px;  
  
}
```

```
img:hover {  
  
    Width: 300px;  
  
    Height: 300px;  
  
}
```

```
button {  
  
    background-color: lightyellow;  
  
}  
  
button:hover: yellow;  
  
}  
  
</style>  
  
</head>
```

```
<body>

<h2>Mouse over to Zoom</h2>



<h2>Button Hover</h2>

<button>Submit</button>

</body>
```

4. **:focus {}** - It specifies effects to element when it is in focus. It is mostly used for form elements.

5. **:active {}** - It defines effects when user hold down the mouse button on any element.

Example :

```
<head>

<style>

input[type="text"]:focus {

border: solid 2px green;

}

button:focus {

background-color: yellow;

}

img:active {

border: red solid 4px;

}

</style>
```



```
</head>

<body>

<div>



</div>

<div>

<form>

Name:

<input type="text">

<button>Submit</button>

</form>

</div>

</body>
```

6. :enable

7. :disable

8. :readonly

These are the selectors used to verify the state of any element and apply effects. The state indicates enabled, disabled (or) readonly.

Example :

```
<head>

<style>

button:enabled{
```

```
cursor:grab;

}

button:disabled{

cursor: not-allowed;

}

input[type="text"]:disabled{

background-color: darkgrey;

}

input[type="text"]:read-only{

background-color: red;

}

</style>

</head>

<form>

Name:

<input type="text" readonly>

<button disabled>Submit</button>

</form>

</body>
```

9. **:checked** - It is a pseudo selector used to apply effects based on the checked property of any element. It is mostly used for input elements like radio and checkbox.

Example:

```
<head>

<style>

input[type="checkbox"]~span {

color:red;

}

input[type="checkbox"]:checked~span {

color:green;

}

</style>

</head>

<body>

<fieldset>

<legend> Terms Of Service </legend>

<textarea rows="4" cols="40" disabled>

.....conditions.....

</textarea>

<div>

<input type="checkbox">

<span>| Accept </span>

</div>

</fieldset>

</body>
```

10. :required

11. :valid

12. :invalid

These are the selectors used to define the effects for HTML elements based on the various validation states like required, valid and invalid.

Example :

```
<head>
```

```
<style>
```

```
input[type="email"]:required{
```

```
border: yellow 2px solid;
```

```
}
```

```
input[type="text"]:valid{
```

```
border:green 2px solid;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<form>
```

Name:

```
<input type="text" palceholder="Name required" required minlength="4"  
pattern="[A-Z]{4,10}">
```

```
<br>
```

Email:

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

```
</form>
```

```
</body>
```

13. **:optional** - It is used to define effects for optional fields. Any field is identified as optional if required not defined.

Example :

```
<head>
```

```
<style>
```

```
    input[type="email"]:optional{
```

```
        border:yellow 2px solid;
```

```
    }
```

```
    input[type="email"]:required{
```

```
        border:red 2px solid;
```

```
    }
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<form>
```

```
Email: <input type="email" required>
```

```
</form>
```

```
</body>
```

Note : Remove required for email, it becomes optional.

14. :in-range

15. :out-of-range

These are the selectors used to verify the numeric range of input value and change the effects based on value in-range (or) out-of-range.

Example :

```
<head>

<style>

input[type="number"]:in-range {

border:green 2px solid;

}

input[type="number"]:out-of-range {

border:red 2px solid;

}

</style>

</head>

<body>

<form>

Age :

<input type="number" min="15" max="30">

</form>

</body>
```

16. `:target` - It is a selector used for applying effects for any element that becomes a target in the page. The element must be defined with an "id" and the id's are referred from URL using "#".

Example :

```
<head>

<style>

.section {

border:solid 1ps black;

margin:10px;

padding: 10px;

}

.section:target {

background-color: green;

color:white;

}

</style>

</head>

<body>

<header>

<a href="#html">HTML</a>

<span>|</span>

<a href="#css">CSS</a>

<span>|</span>

<a href="#js">JavaScript</a>
```

```
</header>

<section style="margin-top: 30px;">

<div id="html" class="section">

HTML Tutorial

</div>

<div id="css" class="section">

CSS Tutorial

</div>

<div id="js" class="section">

JavaScript Tutorial

</div>

</section>

</body>
```

Level Based Selectors in CSS

CSS provides a set of selectors that are used to apply effects to the elements based on their occurrence i.e the level number in a hierarchy. They are,

1. :first-of-type
2. :last-of-type
3. :only-of-type

4. :only-child
5. :nth-child()
6. :nth-last-child()
7. :nth-of-type()
8. :nth-last-of-type()
9. :last-child
10. :root
11. :empty

:first-of-type]

:last-of-type]

These are the selectors used to apply effects for any element when it is at the first and last occurrence in a list.

Example :

```
<head>
```

```
<style>
```

```
tr:first-of-type {
```

```
background-color: darkmagenta;
```

```
color:white;
```

```
}
```

```
tr:last-of-type {
```

```
background-color: teal;
```

```
color: white;
```

```
}  
  
</style>  
  
</head>  
  
<body>  
  
<div>  
  
<table width="400" border="1">  
  
<tr>  
  
<td>Name</td>  
  
<td>Prcie</td>  
  
</tr>  
  
<tr>  
  
<td>Samsung Tv</td>  
  
<td>120000</td>  
  
</tr>  
  
<tr>  
  
<td>Mobile</td>  
  
<td>14000</td>  
  
</tr>  
  
<td colspan="2" align="center"> &copy; Copyright</td>  
  
</tr>  
  
</table>  
  
</div>  
  
</body>
```

CSS Effects

CSS provides a set of effects that are used to change and customize the appearance of any content. It includes backgrounds, borders, text, margins etc.,

background Effects :

1. background-color
2. background-image
3. background-position
4. background-size
5. background-repeat
6. background-origin
7. background-clip
8. background-attachment

Effect	Options
background-color	Sets background color.
background-image	Sets a background image using "url()".
background-position	Sets position of image. - top left, center, right - center left, center, right - bottom left, center, right - x:pixels y:pixels
background-size	Sets height and width by % (or) pixels.

background-repeat	Repeats background - repeat - no-repeat - repeat-x - repeat-y
background-origin	Specifies the origin from margin. - border-box - content-box
background-clip	Specifies the origin from padding. - border-box - content-box - padding-box
background-attachment	Makes the background scroll (or) fixed.

Example :

```
<head>
```

```
<style>
```

```
.effects{
```

```
background-image: url ("../images/back.png");
```

```
background-repeat: no-repeat;
```

```
background-clip: content-box;
```

```
border: 3px red dotted;
```

```
padding: 20px;
```

```
}  
  
</style>  
  
<p class="effects"> Some Text </p>
```

Example :

```
<style>  
  
body {  
  
    background-image: url ("images/logo.png");  
  
    background-position:center center;  
  
    background-repeat:no-repeat;  
  
    background-attachment: fixed;  
  
    background-size: 30%;  
  
}  
  
</style>
```

CSS Borders

1. border-style
2. border-color
3. border-width
4. border-left
5. border-right
6. border-top
7. border-bottom

Example :

```
<head>

<style>

.effects {

border-top-color: red;

border-left-color: red;

border-left-style: double;

border-top-style: double;

border-top-width: 5px;

border-left-width: 5px;

border-right-width: 8px;

border-right-color: green;

border-bottom-color: green;

border-right-style: dotted;

border-bottom-style: dotted;

}

</style>

<p class="effects"> Some text </p>
```

Example : ***Border in Shorthand style***

```
<style>
P {
    border: 2px solid red;
}
```

CSS Margins

1. margin
2. margin-left
3. margin-top
4. margin-bottom

Example : ***Shorthand***

```
<style>
p {
    Margin: 100px
}
</style>
```

Example : *individual Margins*

```
<style>
```

```
P {
```

```
    margin-left: 100px;
```

```
    margin-top: 20px;
```

```
}
```

```
</style>
```

CSS Padding

Padding defines the distance between border and the content. You can configure in pixels (or) in percent. They are,

padding

padding-left

padding-right

padding-top

padding-bottom

Example :

```
<style>
```

```
P {
```

```
    border: 2px solid black;
```

```
    padding-left: 20px;
```

```
    padding-right: 20px;
```



```
padding-top: 30px;  
  
padding-bottom: 30px;  
  
}  
  
</style>  
  
<body>  
  
<p> Some Text..</p>  
  
</body>
```

CSS Height and Width

CSS provides a set of properties define the dimensions for adjusting height and width of any element. The properties are,

height
max-height
max-width
min-height
min-width
width

The attributes min and max are used to define that the content width and height will not be responsive and they remain with fixed height and width.

Example :

```
<style>

p{

    border: solid 2px black;

    width: 400px;

    height: 100px;

    min-width:300px;

    min-height: 75px;

}

</style>

<body>

<p> Some text </p>

</body>
```

CSS Text Effects

CSS provides a set of attributes that are used to control the behaviour of text in a page. It includes the following,

color

direction

letter-spacing

line-height

text-align

text-decoration

text-indent
text-shadow
text-transform
text-overflow
unicode-bidi
vertical-align
white-space
word-spacing

Example :

```
<body>  
  
<dl>  
  
<dt>Color</dt>  
  
<dd style="color:red">Text Color</dd>  
  
<dt>Direction</dt>  
  
<dd style="direction:rtl;">Text Direction</dd>  
  
<dt>Letter/Word Spacing</dt>  
  
<dd style="letter-spacing: 5px; word-spacing: 5px;">Space Between Letters</dd>  
  
<dt>Text Decoration</dt>  
  
<dd><a href="../home.html" style="text-decoration: none;">Home</a></dd>  
  
<dt>Text Shadow</dt>  
  
<dd><h1 style="text-shadow: gray 2px 3px;">Welcome</h1></dd>  
  
<dt>Text Transform</dt>  
  
<dd style="text-transform: capitalize;">Welcome</dd>
```

```
<dt>Text Indent</dt>
```

```
<dd><p style="text-indent: 100px;">Some Text....</p></dd>
```

```
<dt> Over Flow </dt>
```

```
<dd>
```

```
<div style="border: 2px solid red; white-space:nowrap; overflow:hidden; text-overflow:ellipses; width:100px">
```

```
Your text
```

```
</div>
```

```
</dl>
```

CSS Fonts

CSS provides several font options that are used to change the character appearance with font face, size, style etc., The attributes are,

font

font-family

font-size

font-style

font-variant

font-weight

Note : Font size is defined in pixels (or) in em where "1 em= 16px".

Example :

```
<style>
```

```
p {
```

```
font-family: 'courier new', courier, monospace;

font-size: 3em;

font-style: italic;

font-weight: bold;

font-variant: small-caps;

}

</style>

<body>

<p> Some text </p>

</body>
```

CSS List

CSS provides a set of attributes that are used to control the ordered and unordered list by using the following attributes,

list-style

list-style-image

list-style-position

list-style-type

Example :

```
<style>

ul {

    list-style-image: url('../images/star.png');

    list-style-position: outside;
```

}

</style>

CSS Position

CSS provides a set of positions that are used to change the placement of an object and control the behaviour. The position properties are,

Property	Description
position:bottom	It sets the bottom margin for specified content. It will position at the bottom of the box.
position:top	It sets the margin top for specified content.
position:left	It sets the margin left.
position:right	It set the margin right.
position:relative	It defines a relative position for an element with regard to the parent element. Its placement will not be effected with the properties like top, left, right etc.,
position:absolute	It defines the absolute position with parent element which effects the properties like left, right, top and bottom
position:fixed	It sets the position fixed based on left right margins. It can allow the scrolling of the parent content but will fix at initial position.
position:sticky	It keeps the initial position and the final position different. It becomes sticky when it reaches the final position.

Example :

<head>

```
<style>

article {

    bottom: 0px;

    right: 0px;

    width: 300px;

    height: 50px;

    background-color: yellow;

    border: 2px solid;

    position: fixed;

}

.toolbar {

    background-color:green;

    color: white;

    text-align: center;

    position: sticky;

    top: 0px;

    margin-top: 100px;

}

</style>

</head>

<body>

<article>

Results Released..
```

```
</article>

<div class="toolbar">

<a>Home</a>

<span>|</span>

<a>About</a>

<span>|</span>

<a>Contact</a>

</div>

<div>

<p> ....some paragraphs...</p>

</div>

</body>
```

BootStrap

Booststarp is one of the worlds largest repository of HTML, CSS and JavaScript.

It provides a set of HTML templates, CSS classes and JavaScript functions.

It comprises of an open source library which allows you to customize according to requirements.

Installing BoostStrap :

1. Install node JS for NPM [Package Manager]

- Visit the following

<https://nodejs.org/en/download/>

- Download and install “.msi” file for windows.

2. Test from your command prompt

```
C:\> node -v
```

```
C:\> npm -v
```

3. Open your project in VS Code

4. Open Terminal [CTRL + `]

5. Install bootstrap by using the following command

```
npm install bootstrap
```

6. This will install bootstrap library.

```
Node_modules
```

```
-bootstrap
```

```
-dist
```

```
-CSS
```

```
-bootstrap.css
```

```
Bootstrap.min.css
```

Note : you can also install bootstrap in package managers like Yarn, Composer, Ruby Gems, NuGet etc.,

[www.npmjs.org] : search for packages

Install plugin for VS code :

- Go to Extensions

- Search for

“intellisense for CSS class names in HTML”

Publisher : Zignd

- install for VS Code

Configure BootStrap for HTML Page :

1. Create a new HTML page

“demo.html”

2. Link BootStrap CSS

```
<head>

<link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">

</head>
```

3. Implement CSS class for HTML elements

```
<div class="container">

<div class="alert alert-success">

    Welcome to Bootstrap

</div>

</div>
```

BootStrap Alerts

BootStrap provides a set of classes that are used to display alerts in a message box on the screen by using various effects. The commonly used bootstrap alert classes are,

Class Name

alert

alert-primary]

alert-secondary]

alert-success]	
alert-warning]	alert colors
alert-danger]	
alert-info]	
alert-dark]	
alert-light]	
alert-heading] alert heading text	
alert-link] alert link text	

Example :

```
<div class="alert alert-info">
```

```
<h2 class="alert-heading">info</h2>
```

Welcome to Bootstrap alerts

```
<div>
```

```
<a class="alert-link" href="demo.html">
```

Visit Our Site

```
</a>
```

```
</div>
```

```
</div>
```

BootStrap Badge

It is a CSS effect used for highlighting any specific character (or) symbol within the context of existing content. The classes are,

ClassName

badge - used to define a badge

badge-pill - Defines a oval shaped

badge-primary]

badge-secondary]

badge-Info]

badge-dark] badge colors

badge-light]

badge-success]

Example :

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

```
    <h2 class="badge badge-pill badge-danger">Badge</h2>
```

```
    <div class="alert alert-success">
```

```
        <span class="badge badge-danger">4</span> Likes
```

```
    </div>
```

```
    <div>
```

```
        <span class="badge badge-primary">1</span>
```

```
        HTML Examples
```

```
    </div>
```

```
    <div>
```

```
        <span class="badge badge-primary">2</span>
```

```
        JavaScript Tutorial
```

</div>

<div>

3

CSS Templates

</div>

</div>

</body>

BootStrap Buttons

BootStrap provides a set of classes that are used to define appearance for buttons and control their behaviour to make responsive. The classes are,

btn - base class

btn-sm] button si

btn-lg] small and large

btn-block] responsive

btn-link] button as h

btn-group] groups a set of buttons

btn-group-sm] small and large

btn-group-lg] button group

btn-group-vertical]

btn-group-vertical-sm] groups vertically

btn-group-vertical-lg]

btn-group-horizontal]

btn-group-horizontal-sm] groups horizontal

btn-group-horizontal-lg]

btn-primary]

btn-secondary] contextual buttons

btn-success] with colors

btn-danger]

btn-outline-primary] contextual with outline

btn-outline-success] color and effects

btn-toolbar] tool bar with a set of groups

Example :

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

```
<dl>
```

```
<dt>Button Colors</dt>
```

```
<dd>
```

```
<button class="btn btn-primary btn-sm">Submit</button>
```

```
<button class="btn btn-danger btn-lg">Cancel</button>
```

```
<button class="btn btn-link">SignOut</button>
```

```
</dd>
```

```
<dt>Button Responsive</dt>
```

<dd>

<button class="btn btn-primary btn-block">Register</button>

</dd>

<dt>Button Outline</dt>

<dd>

<button class="btn btn-outline-success">Insert</button>

<button class="btn btn-outline-danger">Danger</button>

</dd>

<dt>Button Group Horizontal</dt>

<dd>

<div class="btn-group">

<button class="btn btn-primary">Home</button>

<button class="btn btn-secondary">About</button>

</div>

</dd>

<dt>Button Group Vertical</dt>

<dd>

<div class="btn-group-vertical"><button class="btn btn-primary">HOMe</button>

<button class="btn btn-secondary">About</button>

</div>

</dd>

<dt>Button Tool Bar</dt>

<dd>

```
<div class="btn-toolbar bg-primary justify-content-between"><div  
class="btn-group"><a href="../home.html" class="btn btn-primary">Home</a>  
  
<a href="../about.html" class="btn btn-primary">About</a>  
  
</div>  
  
<div class="btn-group">  
  
<button class="btn btn-primary">Search</button>  
  
</div>  
  
</div>  
  
</dd>  
  
</dl>  
  
</div>
```

BootStrap Card

Example :

```
<head>  
  
<!-- link bootstrap.css -->  
  
<style>  
  
img {  
  
Height:200px;  
  
Width:150px;
```



```
}

</style>

</head>

<body>

<div class="container">

<h1 class="text-primary text-center">Amazon Shopping</h1>

<h2 class="text-danger">offered products 50% sale</h2>

<div class="card-deck">

<div class="card">



<div class="card-header">

<h2 class="card-title">Bose Speakers</h2>

</div>

<div class="card body">

<p class="card-text">....some text....</p>

<a href="#" class="card-link">more</a>

</div>

<div class="card-footer">

<h2>₹ 4,600/-</h2>

<button class="btn btn-primary">Add to Cart</button>

</div>

</div>

<!--similar for other card>

</div>
```

Bootstrap Dropdown

Bootstrap provides a set of classes that are used to design a dropdown menu.

It is a static dropdown menu used to display a list of items in the format of links, buttons and images.

The classes that are used to design bootstrap dropdown menu are,

Class	Description
dropdown	It defines a dropdown menu.
dropdown-menu	It defines collection of menu items.
dropdown-item	It defines items in menu
dropdown-item-text	It defines text in menu item.
dropdown-divider	It splits the items in dropdown.

Example :

- install jquery for project

```
npm install jquery
```

- Add a new HTML page

```
"home.html"
```

```
<head>
```

```
<link....bootstrap.css>
```

```
<script src="../node_modules/jquery/dist/jquery.js"></script>
```

```
<script src="../node_modules/bootstrap/dist/js/bootstrap.bundle.js">
```

```
</script>
```

```
</head>
```

```
<body>

<div class="container">

<h2>BootStrap Dropdown</h2>

<div class="dropdown">

<button class="btn btn-primary dropdown-toggle"
data-toggle="dropdown">NareshIT</button>

<div class="dropdown-menu">

<div class="dropdown-item">

<a href="#">

>span class="dropdown-item-text">Home</span></a></div>

<div class="dropdown-item"><a href="#"><span
class="dropdown-item-text">About</span></a></div>

<div class="dropdown-divider"></div>

</div>

<div class="dropdown-item"><a href="#"><span
class="dropdwon-item-text">Faculty</span></a></div>

</div>

</div>

</div>

</body>
```

BootStrap Forms

BootStrap provides set of classes that are used to appearance of forms and its elements. The classes are,

Class	Description
form-group	It is used for a container that contains set of controls, such as a category in form.
form-control	It is used to define controls in form i.e textbox, checkbox, dropdown etc.,
form-text	It is used for text in form.
form-inline	It arranges elements side by side.
form-row	It organizes form elements horizontally in a row. [it must be used for inline].

1. Visit the website

www.getBootstrap.com

2. Go to examples category

3. Select “signin” template

4. Right click and “view page source”

5. Right click on “signin.css” --- save link as

6. Save in your project bootstrap/dist/css

7. Create a new HTML page

“login.html”

<head>

<!.... link bootstrap.css, signin.css..>

</head>

<body>

<div class=“container”>

```
<div class="btn-toolbar bg-primary text-white justify-content-between">

<div class="btn-group">

<a class="btn btn-primary">Home</a>

<a class="btn btn-primary">About</a>

</div>

<div class="btn-group">

<div class="form-row">

<form class="form-inline">

<input placeholder="user name" type="text" class="form-control">

<input placeholder="password" type="password" class="form-control">

<button class="btn btn-primary">login</button>

</form>

</div>

</div>

</div>

</div>

<form class="form-signin form">

<h2 class="form-text text-primary">user login</h2>

<div class="form-group">

<label for="txtName">user name</label>

<div>

<input name="txtName" type="text" class="form-control">

<span class="text-danger">Name Required</span>

</div>

</div>
```

```
</div>

<div class="form-group">

<label for="txtPwd">Password</label>

<div>

<input type="password" name="txtPwd" class="form-control">

<span class="text-danger">password 4 to 15 chars only</span>

</div>

</div>

<div class="form-group">

<label for="lstCities">Your City</label>

<div>

<select class="form-control" name="lstCities">

<option>Delhi</option>

<option>Hyd</option>

</select>

</div>

</div>

<div class="btn btn-primary btn-block">Login</button>

</div>

</form>

</div>

</body>
```

Input group :

The Bootstrap input group provides a set of classes that are used to arrange contents in line with prefix and suffix. The classes are,

Class	Description
input-group	It defines line group.
input-group-sm	Size small.
input-group-lg	Size large.
input-group-text	Highlights the text used for prepend (or) append.
input-group-prepend	Content as prefix.
input-group-append	Content as suffix.

Example :

```
<head>

<!--link bootstrap.css-->

</head>

<body>

<div class="container">

<h2>Input group</h2>

<dl>

<dt> Input Append</dt>

<dd>

<div class="input-group input-group-sm">

<input type="text" class="form-control">
```

```
<div class="input-group-append">  
  
<span class="input-group-text">@gmail.com</span>  
  
</div>  
  
</div>  
  
</dd>  
  
<dt>Input Prepend</dt>  
  
<dd>  
  
<div class="input-group input-group-sm">  
  
<div class="input-group-prepend">  
  
<span class="input-group-text">&#8377;</span>  
  
</div>  
  
<input type="text" class="form-control">  
  
<div class="input-group-append">  
  
<span class="input-group-text">.00</span>  
  
</div>  
  
</div>  
  
</dd>  
  
</dl>  
  
</div>  
  
</body>
```

BootStrap List group

A list group comprises of list of items which can be organized horizontally and vertically as a tool bar. The classes are,

className	
list-group	
list-group-horizontal	
list-group-flush	Without vertical line.
list-group-item	
list-group-item-primary, success, warning	
list-group-item-action	Used for links.

Example :

```
<h2>List Group Links</h2>
```

```
<ul class="list-group list-group-horizontal">
```

```
<li class="list-group-item">
```

```
<a href="#" class="list-group-item-action">Home</a>
```

```
</li>
```

```
<li class="list-group-item">
```

```
<a href="#" class="list-group-item-action">About</a>
```

```
</li>
```

```
<li class="list-group-item">
```

```
<a href="#" class="list-group-item-action">Contact</a></li>
```

```
</ul>
```

Note : you can add "active, disabled" options for "list-group-item" to display active (or) disabled state.

Syntax :

```
<li class="list-group-item active"></li>
```

BootStrap Nav Bars

The Nav Bars are used for designing navigation area in a web page. It comprises of a navigation bar with collection of items arranged in horizontal (or) vertical orientation. The classes are,

ClassName
.nav
.nav-pills
.nav-item
.nav-link
.nav-tabs

Example :

```
<nav>

<ul class="nav nav-tabs">

<li class="nav-item active">

<a href="#" class="nav-link">Home</a></li>

<li class="nav-item disabled">

<a href="#" class="nav-link">About</a></li>

<li class="nav-item">

<a href="#" class="nav-link">Contact</a>

</li>
```


</nav>

CSS 3 and 4 Transitions

1. CSS 3 and 4 versions introduced several transition effects that are used to handle animations in web based applications.
2. The transitions have browser compatibility issues. Hence, we have to write for various browsers the plugins used for different browsers are shown below.

- -ms-transform:rotate(30deg);/* IE 9 */
- -moz-transform:rotate(30deg);/* FireFox*/
- -webkit-transform:rotate(30deg);/* Safari */
- -o-transform:rotate(30deg);/* Opera*/

CSS Transform : The transform provides a set of methods that are used to move, scale, spin, stretch and turn elements. The transforms can be defined by using 3 attributes,

1. transform
2. transform-origin
3. transform-style

transform : Transform is used to apply methods which can handle the following functionalities,

- translate()
- rotate()
- scale()

- skew()
- matrix()

transform() : It is a transform method is used to move any object along x and y axis.
You can define in 3 ways,

- a. translate()
- b. translateX()
- c. translateY()

Syntax :

transform:translate(x.y);

transform:translateX(pixels);

transform:translateY(pixels);

Example :

<head>

<style>

img {

width:100px;

height:100px;

transition:1s;

}

img:hover {

transform:transform(100px,100px);

transition:2s;

```
    }  
  
</style>  
  
</head>  
  
<body>  
  
  
  
</body>
```

rotate() : It is a method used to rotate an object by specified angle. You can define by using the units "deg".

Syntax :

```
transform:rotate(360deg);
```

scale() : It is used to change the width and height of any image along x and y axis. You can define the following,

A. scale(x,y)

B. scaleX(2)

C. scaleY(2) //1=100%

Syntax :

```
transform:scale(2,2)
```

```
transform:scaleX(2)
```

skew() : It is a method used for tilting the image by specified degrees along X and Y axis. The methods are,

a) skew()

b) skewX()

c) skewY()

Syntax :

transform:skew(20deg, 30deg);

transform:skew(20deg);

Transform using script :

<head>

<style>

img {

width:100px;

height:100px;

transition:1s;

}

img:hover {

transform:scale(2,2);

transition:2s;

}

</style>

</head>

<body>

<div>


```
<marqueeonmouseover="this.stop()" onmouseout="this.start()" scrollamount="10">  
  
  
  
  
  
</marquee>  
  
</div>  
  
</body>
```

JavaScript

HTML - W3.org

JavaScript - MDN

JavaScript is a light weight, interpreted, just-in-Time compiled programming language.

It is an interpreted language as an interpreter is responsible for translating JavaScript line by line.

It is a Just-In-Time compiled programming language as it is responsible for translating the JavaScript programs in to cross platform programs. It makes JavaScript native to every browser and device.

JavaScript is a language which is supported with object oriented, imperative, functional, dynamic and many other programming approaches.

JavaScript is used with various technologies which includes the following,

Technology	Role of JavaScript
HTML	➤ Making static DOM in to Dynamic DOM.

	<ul style="list-style-type: none">➤ Client side interactions.➤ Client side validations.
CSS	<ul style="list-style-type: none">➤ Dynamic Control of effects.➤ Accuracy in Animations.
NodeJS	<ul style="list-style-type: none">➤ Server Side Programming.➤ Handling request and response.➤ Handling static files.➤ Parsing.
MongoDb	<ul style="list-style-type: none">➤ Database queries.➤ Data manipulations.
Flash	<ul style="list-style-type: none">➤ Creating and Controlling 2D and 3D animations.

Evolution of JavaScript

- ECMA Script is the first scripting language used for internet.
- Later, “Brendan Eich” developed a script called MOCHA, which was later changed to “Live Script” and then “JavaScript”.
- JavaScript was designed for Netscape browser with HTML.
- JavaScript have compatibility issues across various browsers.
- JQuery is a library designed to reduce compatibility issues.
- However a library cannot control the application flow. Better use a Framework.

Web application

<https://github.com/karrasankar158>

(Jquery- Library)

Client request from browser

Page loaded in to browser

Page loads the static DOM

Loads library in to browser

Library is Passive

Browser uses Events to make library active

Application converts the static DOM in to Dynamic DOM

WEB APPLICATION

(Framework-AngularJS)

Client Requests from Browser

Page is loaded in to Browser

Page Loads the Static DOM

Framework Loads in to Browser

Framework is Active

Application converts the Static DOM in to Dynamic DOM

“ A Framework is an software architectural patterns, which is responsible for both build the application and controlling its flow.”

JavaScript with HTML

- JavaScript is used with HTML as a Client Side Script.
- It reduces the burden on server by handling various activities Client Side.
- JavaScript is used with HTML to make the static DOM in to dynamic DOM.
- It is used to handle client side validations and interactions.
- It is not safe as it can broadcast client information to server and it can be blocked by browser.
- The latest version of JavaScript is ES8 - 2018.

Integrating JavaScript in to HTML :

JavaScript functions can be integrated in to a HTML page by using following techniques,

1. Inline
2. Embed
3. External file

Inline : In this technique the JavaScript functions are defined within an element on specific event. These functions are not accessible to other elements.

Example :

```
<body>

<h2>Click Print Button to Print page </h2>

<input type="button" value="print" onclick="window.print()">

</body>
```

Embed : In this technique JavaScript functions are defined within the page by using script element. So, that they are accessible to any element in page.

Syntax :

```
<script>

function PrintPage() {

window.print();

}
```

```
</script>
```

```
<input type="button" value="PrintPage" onclick="PrintPage()">
```

Location of Script :

The script element in a page can be defined at <head> section (or) <body>section.

Script's in head section are recommended when you want to execute on specific event and explicit calling.

Script's in <body> section are recommended when we want to execute the functions automatically on page load.

Syntax :

```
<head>
```

```
    <script>
```

```
    </script>
```

```
</head>
```

```
    <body>
```

```
        <script>
```

```
        </script>
```

```
    </body>
```

MIME Type of JavaScript :

1. Script can execute both client side (or) server side.
2. MIME type specifies the script type whether client side (or) server side.

3. You can define the type attribute (or) the language attribute for specifying MIME type.

Syntax :

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

```
</script>
```

(or)

```
<script language="javascript">
```

```
</script>
```

Strict Mode of JavaScript :

1. JavaScript functions can be written without strict mode, which allows the developers to implement all possible shortcuts.

2. If script is not in strict mode it leads to coding standard errors. Hence it is recommended to write JavaScript in strict mode.

3. You can turn on the strict mode by using

“use strict”

Example :

```
<script>
```

```
“use strict”;
```

```
function f1(){
```

```
x=10;
```

```
// invalid - x not defined
```

```
document.write(“x=” + x);
```

```
}
```

```
f1();
```

</script>

Note : If strict mode is removed, then above program executes without errors.

JavaScript for Legacy Browsers :

1. JavaScript functions can run on legacy browsers [old version] when defined in HTML comments.

Syntax :

<script>

<!--

alert("Welcome");

-->

</script>

External File :

JavaScript functions can be maintained in a separate script file that have the extension.js, so that the functions are accessible to many pages. However, it will increase the number of requests and page load time.

Example :

1. Add a new folder in your project by name

“Scripts”

2. Add a new file in to “Scripts” folder

“Print.js”

```
"use strict";

function PrintPage() {

    window.print();

}
```

3. Link the file to your Home.html

```
<head>

<script src="Scripts/print.js"></script>

</head>
```

Reference Techniques Used by JavaScript to Access HTML elements :

1. Refer by using DOM hierarchy.

In this technique JavaScript follows the DOM hierarchy for accessing elements of HTML. It uses the memory index reference for elements presented as collection. Hence, the change of position for any element will effect the logic. We have to update the index reference every time.

Example :

```
<head>

<script>

function bodyload() {

    window.document.images[0].src="../images/tv.jpg";

    window.document.forms[0].elements[0].value="Login";

    window.document.forms[1].elements[1].value="Register";

}
```

```
</script>

</head>

<body onload="bodyload()">

<div>

<img width="50" height="50" border="1" id="pic" name="pic">

</div>

<div>

<form name="frmLogin" id="frmLogin">

<h2>Login</h2>

<input type="button" name="btnLogin" id="btnLogin">

</form>

</div>

<form name="frmRegister" id="frmRegister">

<h2>Register</h2>

Email :

<input type="email" name="txtEmail" id="txtEmail">

<input type="button" name="btnRegister" id="btnRegister">

</form>

</div>

</body>
```

2. Refer by using Name :

Every element is defined with reference name by using “name” attribute, which can be common for several elements. You can access by referring to

name. However, you cannot access any child element directly. You have to follow the parent - child hierarchy.

Example :

```
<head>

<script>

function bodyload(){

pic.src="../images/shoes.jpg";

frmLogin.btnLogin.value="Login";

frmRegister.btnRegister.value="Register";

}

</script>

</head>
```

3. Refer by using ID :

Every element in HTML can be defined with a unique reference ID. You can access the element directly by using ID without any parent-child hierarchy.

However, ID can be used by CSS and Server Side Scripting. Hence, you cannot use ID for referring multiple elements.

The function used by JavaScript is

`“document.getElementById()”`

Example :

```
<script>

function bodyload() {

document.getElementById("pic").src="../images/tv.jpg";
```

```
document.getElementById("btnLogin").value="Login";  
  
document.getElementById("btnRegister").value="Register";  
  
}  
  
</script>
```

4. Refer by using Tag Name :

HTML elements are presented by using a specific tag. You can access all elements that are designed by using the given tag. JavaScript provides the function,

`“document.getElementsByTagName()”`

Example :

```
<script>  
  
function bodyload(){  
  
var divs = document.getElementsByTagName("div");  
  
alert("Total No of Divs are : "+ divs.length);  
  
}  
  
</script>
```

5. Refer by Class Name :

HTML elements can use class attributes for reference. Every element can be defined with multiple references by giving the class attribute. You can access the elements by using

`“document.getElementsByClassName()”`

Example :

```
<head>
```

```
<style>

.frm {

    border: 2px solid black;

    height: 200px;

    width:400px;

}

</style>

<script>

function bodyload() {

    var classes= document.getElementsByClassName("frm");

    alert("Total No of Elements using Frm class are: "+ classes.length);

}

</script>

</head>

<body onload="bodyload()">

<form class="frm">

.....

</form>

</body>
```

6. Refer Elements by using Common Name :

HTML supports a Common Name for a set of elements. You can access all of them by using the function

“document.getElementsByName()”

Example :

```
<head>

<script>

function bodyload(){

    var payments= document.getElementsByName("pay");

    alert("Total No pf Payment Modes :" + payments.length);

}

</script>

</head>

<body onload="bodyload()">

<fieldset>

<legend>Payment Mode</legend>

<input type="radio" name="Pay">Cash

<input type="radio" name="pay">Credit Card

</fieldset>

</body>
```

JavaScript Output Techniques

JavaScript is a Programming Language that handles interaction between the User and Hardware services.

It is responsible for taking Input from user, processing the input and generating an output. The various output functions and properties used in JavaScript are,

1. alert()
2. confirm()

3. console.log()

console.info()

console.debug()

console.error()

console.warn()

4. document.write()

5. innerText

6. innerHTML

alert() : It pops up a message box with any specifies message and seeks confirmation from the user. It will not allow to CANCEL (or) perform any another task until the alert is confirmed by clicking OK.

Syntax :

```
alert("Message");
```

confirm() : It is similar to an alert but allows to CANCEL. It is a boolean function that returns true on OK and false on CANCEL.

Syntax :

```
confirm("Message");
```

Console Functions :

A Browser debugger comprises of console where you can log the messages to track the activities performed within the page.

The log messages can be displayed in various styles by using the methods

console.info(), console.debug(), console.error(), console.warn().

Syntax :

```
console.log("message/expression");
```

Example :

```
<head>
```

```
<script>
```

```
    console.log("Delete Not Yet USed");
```

```
    function DeleteClick() {
```

```
        console.warn("Delete Clicked");
```

```
        var status=confirm("Record Will be Deleted Are You Sure?");
```

```
if(status==true) {
```

```
    alert("Record Deleted");
```

```
    console.error("Delete Clicked and Confirmed");
```

```
} else {
```

```
    alert("Canceled");
```

```
    console.info("Delete Clicked but Cancelled")
```

```
}
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<input type="button" value="Delete" onclick="DeleteClick()">
```

</body>

Note : Press F12 function key to open debugger console.

`document.write()` : It is an Output function that can print the output in the same page but on a new screen. It can automatically erase when page is refresh.

Syntax :

```
Document.write("message");
```

`innerHTML` : It is an output property used to display output within the page in any specified element that can present text. It allows the formats for text.

Syntax :

```
p.innerHTML ="<b>Welcome</b>";
```

`innerText` : It is similar to `innerHTML` but it not allow any format for text. It is an RC Data content.

Syntax :

```
p.innerText ="Welcome";
```

Example :

```
<head>
```

```
<!-- link bootstrap.css-->
```

```
<script>
```

```
function DeleteClick() {
```

```
var msg = document.getElementById("msg");

var status = confirm("Record will be Deleted Are You Sure?");

if(status==true) {

    msg.innerHTML = "<font color='green'>Record Deleted..</font>";

} else {

    msg.innerText="canceled";

}

}

</script>

</head>

<body>

<div class="container">

<div align="center">

<input type="button" value="Delete" onclick="DeleteClick()" class="btn
btn-outline-danger">

</div>

<h2 align="center" id="msg">

</h2>

</div>

</body>
```

JavaScript Input Techniques

JavaScript with HTML will allow input from the user by using,

1. prompt()

2. HTML form input elements

`prompt()` : It is an input method that popsup an input box from were user can input value.

`prompt ()` is used when input is occasional.

`prompt()` returns the input value when it click to OK and it returns null when it canceled.

Syntax :

`prompt("Your message", "Default value")`

Example :

```
<head>
```

```
<!--link bootstrap.css-->
```

```
<script>
```

```
function CreateClick() {
```

```
    var folderName=prompt("Folder Name", "New_Folder");
```

```
    var msg=document.getElementById("msg");
```

```
    if(folderName==null) {
```

```
        alert("You Canceled");
```

```
    } else {
```

```
        msg.innerHTML+="Folder Created By Name:"+folderNAME + "<br>";
```

```
    }
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>

<div class="container">

<fieldset>

<legend>Add New Folder</legend>

<input type="button" value="Create" class="btn btn-success"
onclick="CreateClick()">

<div id="msg">

</div>

</fieldset>

</div>

</body>
```

Input using form Input Elements :

HTML provides several form elements that are used to input a value dynamically. It includes textbox, checkbox, dropdown list, listbox and radio button etc.,

You can access the elements by using id (or) name.

You can access the element values by using the property value.

Syntax :

```
document.getElementById("txtName").value;
```

Example :

```
<!DOCTYPE html>
```

```
<html>
```

```
    <head>
```

```
<link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">

<style>

    #dialog {

        width: 300px;

        height: 300px;

        position: fixed;

        top:10px;

        left:200px;

    }

</style>

<script>

    function RegisterClick(){

        document.getElementById("dialog").style.display="block";

        document.getElementById("fieldRegister").style.opacity="0.3";


        document.getElementById("lblName").innerHTML =
document.getElementById("txtName").value;

        document.getElementById("lblMfd").innerHTML =
document.getElementById("txtMfd").value;

        document.getElementById("lblStock").innerHTML =
frmRegister.stock.value;

        document.getElementById("lblShipped").innerHTML =
document.getElementById("lstCity").value;

    }
```

```
function OkClick() {  
  
    document.getElementById("dialog").style.display="none";  
  
    document.getElementById("fieldRegister").style.opacity="1.0";  
  
}  
  
</script>  
  
</head>  
  
<body>  
  
    <div class="container">  
  
        <form name="frmRegister">  
  
            <fieldset id="fieldRegister">  
  
                <legend>Register Product</legend>  
  
                <dl>  
  
                    <dt>Name</dt>  
  
                    <dd><input                type="text"                id="txtName"  
class="form-control"></dd>  
  
                    <dt>Manufactured</dt>  
  
                    <dd><input                type="date"                id="txtMfd"  
class="form-control"></dd>  
  
                    <dt>Is In Stock</dt>  
  
                    <dd>  
  
                        <input type="radio" name="stock" value="Available">  
Yes  
  
                        <input type="radio" name="stock" value="Out of  
Stock"> No
```

```
</dd>

<dt>Shipped To</dt>

<dd>

    <select class="form-control" id="lstCity">

        <option>Delhi</option>

        <option>Hyderabad</option>

    </select>

</dd>

</dl>

<input type="button" value="Register" class="btn btn-success"
onclick="RegisterClick()">

</fieldset>

<div id="dialog" style="display: none;" class="card">

    <div class="card-header">

        <h3>Product Details</h3>

    </div>

    <div class="card-body">

        <table class="table table-hover">

            <tr>

                <td>Name</td>

                <td id="lblName"></td>

            </tr>

            <tr>

                <td>Manufactured</td>
```

```
<td id="lblMfd"></td>

</tr>

<tr>

<td>Is In Stock</td>

<td id="lblStock"></td>

</tr>

<tr>

<td>Shipped To</td>

<td id="lblShipped"></td>

</tr>

</table>

</div>

<div class="card-body">

<input type="button" value="OK" class="btn btn-danger"

onclick="OkClick()">

</div>

</div>

</form>

</div>

</body>

</html>
```

1. Variables are storage locations in memory where you store a value and use it as a part of any expression.

2. Configuring variables includes 3 phases,

A. Declaration

`var x;`

B. Rendering

`X=10;`

C. Initializing

3. Declaring variables in javascript is mandatory only when it is defined in strict mode.

Syntax :

`<script>`

`function f1() {`

`x=10; //valid`

`document.write("x=" +x);`

`}`

`f1();`

`</script>`

Note : If you defined "use strict"; then `x=10; //invalid`, it needs declaration.

4. Variables can be declared by using following key words,

Keyword	Description
---------	-------------

var	<p>It defines a function scope for variable.</p> <p>It allows declaration and rendering.</p> <p>It can also be initialized.</p> <p>It supports shadowing.</p>
let	<p>It defines a block scope for a variable.</p> <p>It also allows declaration, rendering and initialization.</p> <p>It will not support shadowing.</p>
const	<p>It defines a block scope for a variable.</p> <p>It will not allow declaration and rendering.</p> <p>It must have initialization.</p> <p>It will not support shadowing.</p>

Shadowing is a process of declaring same name variable within the given scope.

Syntax :

```
<script>
```

```
"use strict";
```

```
function f1() {
```

```
var x;
```

```
x=10;
```

```
if(x==10) {
```

```
var x;                                //valid - can shadow
```



```
x=40;

let y;

y=20;

let y;                //invalid - can't shadow

y=50;

const z=30;

document.write("x=" +x + "<br>" + "y=" +y + "<br>" + "z=" +z);

}

}

f1();

</script>
```

➤ JavaScript variables are hoisted. It is a technique in computer programming where compiler will execute the declaration part before the rendering without considering the sequential order.

Example :

```
<script>

"use strict";

function f1() {

x=10;

document.write("x=" +x);

var x;    //hoisted

}

f1();
```

</script>

- A variable name must start with an alphabet (or) underscore.
- Name can be alpha numeric but cannot start with a number.
- Name of variable must speak what it is i.e the purpose and reference it is used for.
- Variable name can be maximum 255 characters long.
- Always use camel case for naming variables.

Data Types

A Data type determines the type of value that can be stored in a memory reference.

JavaScript is implicitly typed language i.e the data type for memory is determined according to the value assigned.

The JavaScript data types are categorized in to two groups,

1. Primitive types

2. Non - Primitive types

Primitive Types :

- Primitive types are stores in a memory stack.
- Stack uses the mechanism last-in-first-out.
- The range for values is fixed in memory.
- Values can be directly accessed by using the name of memory allocated for it. Hence, they are also known as “Value Types”.
- The JavaScript value types are

- i. Number
- ii. Boolean
- iii. String
- iv. Null
- v. Undefined

Number Type :

JavaScript number type is used to handle numeric values, which can be any one of the following,

- 1) Signed integer - var x = -10
- 2) Unsigned integer - var x = 10
- 3) Floating point - var x = 4.5
- 4) Double - var x = 567.789
- 5) Decimal - var x = 4.6784516548465165 [29 places]
- 6) Exponent - var x = 2e3

Example :

```
<script>

"use strict"

function f1() {

var decimal = 5.412165546161;

var exp = 2e2;

document.write("Decimal=" + decimal + "<br>" + "Exponent=" + exp);

}

f1();
```

</script>

Boolean Type :

A Boolean value is used in decision making.

JavaScript boolean types can be store the boolean values true (or) false.

JavaScript boolean values are designated with numeric type

true = 1

false = 0

JavaScript boolean conditions can be handled directly by using zero (or) one.

It is always recommended to define boolean conditions by using true (or) false instead of zero and one.

Example :

<script>

"use strict"

function f1() {

var name="Samsung TV";

var price=46555.64;

var IsInStock = false;

document.write("Name=" + name + "
" + "Price=" + price + "
" + "Is In
Stock=" + ((IsInStock==true)?"Available":"Out Of Stock"));

}

f1();

</script>

String Type :

String is a literal enclosed with ASCII values, which contain a combination of characters, numbers and special characters.

JavaScript string literals are enclosed by using the following,

Name	Entity	Description
double quote	" "	Used for outer string
single quote	' '	Used for inner string
back tick	` `	Used for string embed with expression

Syntax :

```
var link="<a href='home.html'> Home </a>";
```

Note : the expression to embed into a string is defines by using “\${}”

Syntax :

```
var string= `Total amount ${qty * price} you have to pay for TV`.
```

Escape Sequence Character :

JavaScript string literal cannot print several special characters as they escape printing during compilation. To print the non printable characters you have to use the “ Escape Sequence Character “\” “.

Example :

```
<script>
```

```
function f1() {
```

```
var path = "\"D:\\images\\car.jpg\"";
```

```
document.write("Path=" + path);  
  
}  
  
f1();  
  
</script>
```

String Manipulation :

Function	Description
charAt()	It returns the character at specified index.
indexOf()	It returns the first occurrence index number of given character in a string.
lastIndexOf()	It returns the last occurrence index number of given character in a string.
substring()	It returns a string defined between the given index number range.
length	It returns the total count of characters in a given string.

Syntax :

```
var str = "Welcome";  
  
str.charAt(0);           -W  
  
str.indexOf("e");        - 1  
  
str.lastIndexOf("e");     - 6  
  
str.substring(2)          - lcome [ 2 to end ]  
  
str.length                - 7
```

Example :

```
<head>

<script>

function VerifyClick()

{

var email = document.getElementById("txtEmail").value;

var msg = document.getElementById("msg");

var atpos = email.indexOf("@");

var dotpos = email.lastIndexOf(".");

if(atpos<2 && (dotpos-atpos)<1) {

msg.innerHTML = "Invalid Email";

} else {

document.write("Email Verified");

}

}

</script>

</head>

<body>

Your Email:

<input type="text" id="txtEmail">

<input type="button" value="Verify" onclick="VerifyClick()">

<br>

<h3 id="msg"></h3>
```

</body>

For defining Id and Domain -

var id=

```
email.substring(0,email.indexOf("@"));
```

var domain=

```
email.substring(email.indexOf("@")+1);
```

String Formatting Functions :

Function	Description
big()	Sets font size to large.
blink()	Blinks the given text (not supported on every browser).
bold()	Sets Bold .
fixed()	Sets Normal Font size.
fontcolor('green')	Sets a font color.
fontsize('4')	Sets a font size.
italics()	Applies <i> emphasized.
small()	Sets font level small.
strike()	Sets <strike> effect.
sub()	Sets subscript <sub>.
sup()	Sets superscript <sup>.
toLowerCase()	Converts all letters to Lower case.

toUpperCase()	Converts to Uppercase.
---------------	------------------------

Example :

```
<script>
```

```
function ShowTip() {
```

```
document.getElementById("msg").innerHTML = "Please Enter Name in Block Letters";
```

```
}
```

```
function HideTip() {
```

```
document.getElementById("msg").innerHTML="";
```

```
var uname = document.getElementById("txtName").value;
```

```
document.getElementById("txtName").value = uname.toUpperCase();
```

```
document.getElementById("msg").innerHTML="Hello!"
```

```
uname.bold().italics().fontcolor('green').fontsize(5).toUpperCase();
```

```
}
```

```
</script>
```

```
<body>
```

Your Name:

```
<input type="text" placeholder="Name in Block Letters" id="txtName"
onfocus="ShowTip()" onblur="Hide Tip()">
```

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

```
</body>
```

Example : Write logic to print any given sentence in "Sentence Case" i.e First letter of sentence must be caps ?

```
<script>

function f1() {

var msg = "welCOME to JavaScript";

var firstChar=msg.charAt(0);

var restChars=msg.substring(1);

var newMsg=firstChar.toUpperCase() + restChars.toLowerCase();

document.write(newMsg);

}

f1();

</script>
```

Example : Write a program to print the given sentence in "Title Case" i.e every word first letter must be capital ? ----- Task

String split() : It is a string function that splits the given string at specified character and stores in an array.

Syntax :

```
var msg="Welcome to javascript"

var str=msg.split(' ');
```

```
str[0] // Welcome
```

```
str[1] // to
```

```
str[2] // javascript
```

Undefined Type :

- Undefined is a Data type specified for variables when the value is not defined.
- JavaScript variables are implicitly typed. Hence, if a value is not specified then they are assigned with “undefined” key word.

Example :

```
<script>

function f1() {

var name="Samsung TV";

var price;

if(price==undefined) {

document.write("Name=" + name);

}else{

document.write("Name=" + name + "<br>" + "Price=" + price);

}

}

f1();

</script>
```

Null Type :

- Null is assigned to reference if a value is not passed dynamically during run time.

- The null types are verified by using the keyword “null”.

Example :

```
<script>

function f1() {

var name= prompt("Enter Name:");

if(name=="") {

document.write("Name can't be Empty");

} else if(name==null){

document.write("You Canceled");

} else {

document.write("Hello ! " + name);

}

}

f1();

</script>
```

Note :

Undefined is a type returned when variable is defined but value not defined.

“Not Defined” is an error returned when variable is not defined.

Summary

Data Type	Description
-----------	-------------

String	Represents sequence of characters e.g “hello”.
Number	Represents numeric values e.g 100.
Boolean	Represents boolean value either true (or) false.
Undefined	Represents undefined value.
Null	Represents null i.e no reference at all.

Non-Primitive Types

- The Non - Primitive types are stored in a memory heap.
- They don't have any fixed range for values.
- They are stored in the references of main memory and accessed using the reference memory. Hence, they are also known as reference types.
- The JavaScript reference types are,
 - a) Array
 - b) Object
 - c) Regular Expression

Initializing Array :

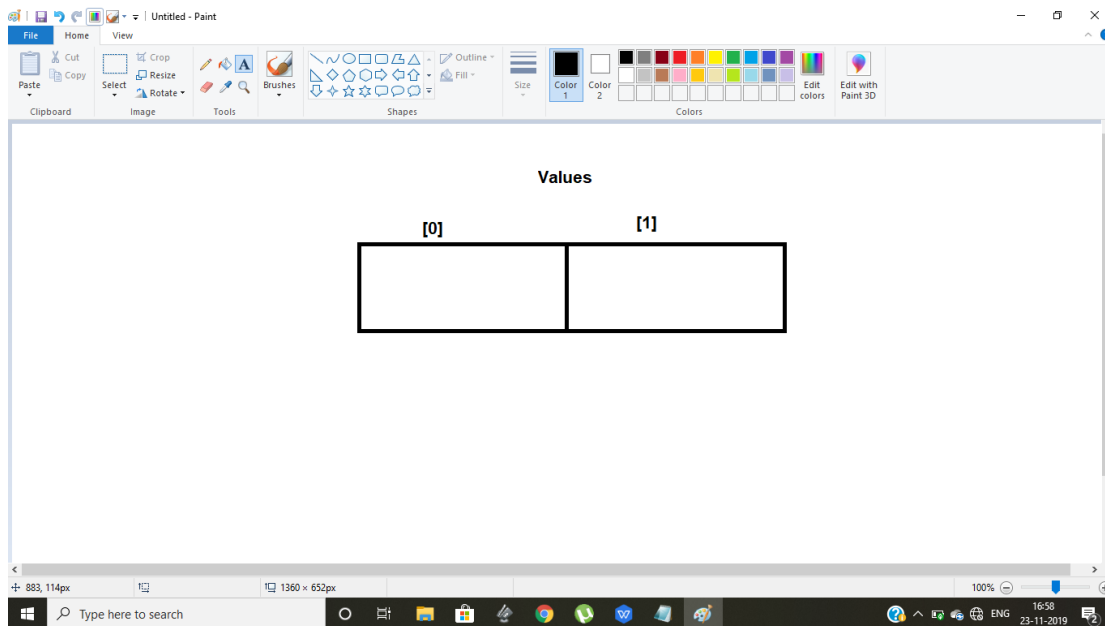
- Array cannot be declared in JavaScript. It must be initialized with memory.
- You can initialize memory for array by using,
 - a) Array Meta Character []
 - b) Array Constructor “Array()”
- The meta characters can initialize memory but cannot initialize the size of memory.

- The Array constructor can initialize the memory and also can allocate the memory dynamically. It can restrict the array size.

Syntax:

```
var values = [];           //length = 0
```

```
var values = new Array(2); //length = 2
```



Rendering values in an array :

Values can be rendered in to an array while initializing the memory (or) after initializing the memory.

Syntax :

```
var values = new Array(10, "A", true);
```

(or)

```
var values = new Array(3);
```

```
values[0] = 10;
```

```
values[1] = "A";
```

```
values[2] = true;
```

Note : Array constructor is over loaded with following functionalities,

a) `Array(...values);` `//Multiple values`

b) `Array(size);` `//Number`

If values are not rendered in to array and memory size is defined then all values are rendered with undefined.

Array allows to render values in random order.

Syntax :

```
var values = new array(2);
```

```
values[1] = 10;
```

```
values[0] = 20;
```

prints : 20, 10

Reading Values from an Array :

- You can read values from an array by using the properties of array.
- The index references are defined as properties and they are of Type String.
- The default properties for array are initialized from "0".

Syntax :

```
var products = ["TV","Mobile"];
```

```
products[0];      //TV
```

```
products["1"]      //Mobile
```

JavaScript provides a set of array functions that are used to read values from an array. They are,

Function	Description
to String()	It reads and returns the values in a string format separated with “,”.
join()	It is similar to to String() but you can use custom delimiter (separator).
slice()	It is used to extract values from an array by using their index number.
find()	It can search for value in an array and return only the first value that matches given condition.
filter()	It is similar to “find()” but can return all values that match the condition.

Syntax :

```
var values = [];  
  
values.toString();  
  
values.join("<br>");  
  
values.slice(startIndex,endIndex);  
  
values.find(function(){});  
  
values.filter(function(){});
```

Example :

```
<script>  
  
function f1(){
```



```
var products = ["TV","Mobile","Shoe"];

var price = [13500, 56000, 46000, 5240];

document.write("ToString=" + products.toString() + "<br>");

document.write("join=" + products.join("<br>") + "<br>");

document.write("Slice=" + products.slice(1,2) + "<br>");

document.write("price > 40000 (Filter) : " + price.filter(function(price){return
price>=40000}) + "<br>");

document.write("Price > 40000 (Find) :" + price.find(function(price){return
price>=40000})))

}

f1();

</script>
```

Adding Values in to Array :

Function	Description
push()	It is used to add one (or) more elements in to array towards the end of array i.e as last item.
unshift()	It adds new elements in to array as first item.
splice()	It adds new elements in to array at specified index.

Syntax :

```
var values = [];

values.push("item1", "item2",....);

values.unshift("item1", "item2",...);
```

```
values.splice(InsertAt, DeleteCount, "NewItem,....");
```

Example :

```
<script>

function f1(){

var products = ["TV"."Mobile"];

products.push("shoe");

products.unshift("watch");

products.splice(2,0,"Sunglasees");

document.write(products.join("<br>"));

}

f1();

</script>
```

Remove Values from Array :

Function	Description
pop()	It removes and returns the last item.
shift()	It removes and returns the first item.
splice()	It removes any specified item based on its index number.

Example :

```
<script>

function f1(){
```

```
var products = ["TV","Mobile","Shoe"];

document.write(products.join("<br>" + "<br>"));

var product = products.splice(1,1);

document.write(product + "-Removed <br>");

document.write(products.toString());

}

f1();

</script>
```

Search for values in an Array :

Function	Description
indexOf()	It can search for any value in an array and return its index number. If the specified element not found then it returns “-1”.
lastIndexOf()	It returns the last occurrence index number of given value.

Example :

```
<script>

function f1(){

var products = ["TV","mobilr","Shoe","Mobile"];

if(products.lastIndexOf("Mobile")==-1) {

document.write("Not Found");

} else{

document.write("Mobile Found at " + products.lastIndexOf("Mobile"));
```

```
}  
  
}  
  
f1();  
  
</script>
```

Concat Arrays :

The function “concat()” is used to combine values from multiple arrays and store in a single array.

Example :

```
<script>  
  
function f1(){  
  
var shoes = ["Nike Casuals","Lee Cooper Boot"];  
  
var electronics = ["Samsung TV","MI Mobile"];  
  
var products = shoes.concat(electronics);  
  
document.write(products.join("<br>") + "<br>");  
  
}  
  
f1();  
  
</script>
```

Object Type

- Object in computer programming represents data and logic.
- The data is stored in the form of properties and logic is defined in the form of methods.
- An object in javascript encapsulates all the members in “{ }”.

- Object provides re-usability for properties and methods.

Syntax :

```
var object = {  
  
property : value,  
  
method : function() {}  
  
};  
  
object.property;  
  
object.method();
```

Example :

```
<script>  
  
function f1(){  
  
var product = {  
  
Name : "Samsung TV",  
  
Price : 45000.55,  
  
Qty : 2,  
  
IsInStock : true,  
  
Shipped To : ['Delhi','Hyd'],  
  
Total : function(){  
  
return product.Qty*product.Price;  
  
},  
  
print: function(){  
  
document.write("Name=" + product.Name + "<br>" + "Price=" + product.Price + "<br>  
+      "Qty="      +      product.Qty      +      "<br>"      +      "InStock="      +
```

```
((Product.IsInStock==true)?"Available":"Out of Stock") + "<br>" + "Shipped To=" +  
product.shippedTo.toString() + "<br>" + "Totla Amount=" + product.Total() + ",br>");  
  
}  
  
}  
  
product.print();  
  
document.write("<hr>");  
  
product.Name : "Samsung TV";  
  
product.Price : 45000.55;  
  
product.Qty : 2;  
  
product.IsInStock : false;  
  
product.Shipped To :['Delhi','Hyd'];  
  
product.Print();  
  
}  
  
f1();  
  
</script>
```

Example : JSON Type Data

- JSON is a format used to present data.
- It is JavaScript Object Notation.
- JSON is cross platform and can store data office.
- It is an array of objects.

<!DOCTYPE html>

<html>

```
<!--link bootstrap.css-->

<style>

Img {

width:200px;

height:200px;

}

</style>

<script>

var products = [

{Name:"Samsung TV", Price : 45000.55, Photo:"../images/tv.jpg"},

.....

];

function GetDetails(index){

document.getElementById("lblName").innerHTML = products[index].Name;

document.getElementById("lblPrice").innerHTML = "&8377;" +

products[index].Price;

document.getElementById("imgProduct").src=products[index].Photo;

}

function bodyload(){

GetDetails(0);

}

var count=0;

function NextClick(){

count++;
```

```
if(products.length==count){  
  
    alert("Last Product");  
  
}  
  
GetDetails(count);  
  
}  
  
function PreviousClick(){  
  
    count--;  
  
    GetDetails(count);  
  
    if(count==0){  
  
        alert("No More Back...");  
  
    }  
  
}  
  
</script>  
  
</head>  
  
<body onload="bodyload()">  
  
<div class="contaioner text-center">  
  
<h2 class="text-primary">Amazon Shopping</h2>  
  
<div class="card bg-light text-danger">  
  
<div class="card-header">  
  
<h3 id="lblName"></h3>  
  
</div>  
  
<div class="card body">  
  
<input type="button" value="<" class="btn btn-danger" onclick="PreviousClick()">
```



```
<img id="imgProduct" width="200" height="200" class="img-thumbnail">  
  
<input type="button" value=">" class="btn btn-danger" onclick="NextClick()">  
  
</div>  
  
<div class="card-footer">  
  
<h3 id="lblPrice"></h3>  
  
</div>  
  
</div>  
  
</div>  
  
</body>  
  
</html>
```

Regular Expression Type

A regular expression is used to verify the format of input value.

It is a literal used to compare the format by using meta characters and quantifiers.

Regular expressions are enclosed in “/ /”.

The values are verified with regular expression by using the function “match()”.

Syntax :

```
var regExp = /\+91[0-9]{10} /;
```

Example :

```
<!DOCTYPE html>
```

```
<html>

<head>

<!--link bootstrap.css-->

<script>

var regExp;

var tip;

function CountryChanged(){

var CountryName = document.getElementById("lstCountries").value;

var flagImage;

switch(countryName)

{

case "India":

tip = "India CallingCode +91 and 10 digits";

flagImage="../images/india.png";

regExp=/\+91[0-9]{10}/;

break;

case "US" :

tip = "US CallingCode +001 and 6 Digits";

flagImage="../images/us.png";

regExp=/\+001[0-9]{6}/;

break;

case "UK":

tip = "UK Calling Code +4 and 5 Digits";
```

```
flagImage=/\+4[0-9]{5}/;

break;

}

document.getElementById("txtMobile").placeholder=tip;

document.getElementById("flag").src=flagImage;

}

function VerifyMobile() {

var mobile = document.getElementById("txtMobile").value;

var msg = document.getElementById("msg");

if(mobile.match(regExp)) {

document.write("Mobile Registered...Confirm by SMS");

} else {

msg.style.color="red";

msg.innerHTML="invalid Mobile <br> " + tip;

}

}

</script>

</head>

<body>

<div class="container">

<fieldset>

<legend> Register Your Mobile

<img id="flag" width="30" height="30" class="img-thumbnail">
```

```
</legend>

<dl>

<dt>Your Country</dt>

<dd>

<select id="lstCountries" class="form-control" onchange="CountryChanged()">

<option value="-1">Select a Country</option>

<option value="India">India</option>

<option value="USA">USA</option>

<option value="UK">UK</option>

</select>

</dd>

<dt>Mobile Number</dt>

<dd>

<input id="txtMobile" type="text" class="form-control">

</dd>

</dl>

<input type="button" value="Submit Mobile" class="btn btn-primary"
onclick="VerifyMobile()">

</fieldset>

<h3 align="center" id="msg"></h3>

</div>

</body>

</html>
```

Date Type :

JavaScript can handle the date values by using a "Date()" provided by date object.

The date object allocates memory for storing date values.

Syntax :

```
var now = new Date();
```

```
var mfd = new Date("2019/02/22");
```

You can read the date values and use them in date manipulations by using a set of date functions.

Function	Description
getHours()	Returns hour number in 24 hr format.
getMinutes()	Returns minutes number.
getSeconds()	Returns seconds number.
getDay()	Returns week day number i.e Sunday=0.
getDate()	Returns the Date value.
getMonth()	Returns month number i.e january=0.
getFullYear()	Returns year number in Y2K i.e 2019=119.
getFullYear()	Returns full year :2019

Example :

```
<script>
```

```
function f1() {
```

```
var product = {
```

```
  Name : "Samsung Tv",
```

```
Price : 45000.53;

Mfd:new Date("2019/03/23")

};

var months = ["January","February","March"];

vr days = ["Sunday","Monday","Tuesday","Wed","Thu","Fri","Saturday"];

document.write(`Name:${product.Name} <br>

Price:${product.Price}  <br>  Manufactured   :   ${days[product.Mfd.getDay()]}

${product.Mfd.getDate()}, ${months[product.Mfd.getFullYear()]}`

}

f1();

</script>
```

Math Type Functions :

JavaScript Math Object :

Math is JavaScript Built-in object that provides a set of mathematical functions which can evaluate and return a value. The Built-in math functions are,

Function	Returned Value
Math.abs(-6.5)	6.5
Math.acos(.5)	1.0
Math.asin(1)	1.5
Math.atan(5)	0.4
Math.ceil(7.6)	8

Math.cos(.4)	0.9
Math.exp(8)	2980.95
Math.floor(8.9)	8
Math.log(5)	1.6
Math.max(1,700)	700
Math.min(1,700)	1
Math.pow(6,2)	36
Math.random()	.7877896
Math.round(.567)	1
Math.sin(Math.PI)	0
Math.sqrt(9801)	99
Math.tan(1.5 * Math.PI)	INF(Infinity)

Example :

```
<style>
```

```
dt{
```

```
font-weight:bold;
```

```
}
```

```
</style>
```

```
<script>
```

```
function Captcha() {
```

```
var code;
```

```
var a= Math.random() * 10;
```

```
var b= Math.random() * 10;

var c= Math.random() * 10;

var d= Math.random() * 10;

var e= Math.random() * 10;

var f= Math.random() * 10;

code = Math.round(a) + " " + Math.round(b) + " " + Math.round(c) + " " +
Math.round(d) + " " + Math.round(e) + " " + Math.round(f);

return code;

}

function bodyload() {

document.getElementById("lblCode").innerHTML = Captcha();

}

</script>

<body onload="bodyload()">

<fieldset>

<legend>User Login</legend>

<dl>

<dt>User Name</dt>

<dd><input type="text"></dd>

<dt>Password</dt>

<dd><input type="password"></dd>

<dt>Verify Code</dt>

<dd id="lblCode"></dd>

<input type="button" value="Refresh" onclick="location.reload()">
```


</dl>

<input type="button" value="Login">

</fieldset>

</body>

Operators in JavaScript

- Operator is technically an object in computer programming that returns a value evaluated by using “operands”.
- The operators are categorized in to 3 groups based on no.of operands they can handle.

a) Unary Operator

- one operand

Ex: x++

b) Binary Operator

- two operands

Ex: x + y

c) Ternary operator

-three operands

Ex : (condition)?true:false

The operators are classified in to several types based on the type of value they return,

1. Arithmetic Operators
2. Comparison (Relational) Operators

3. Bitwise Operators

4. Logical Operators

5. Assignment Operators

6. Special Operators

Operator	Description	Example
+	Addition	10+20=30
-	Subtraction	20-10=10
*	Multiplication	10*20=200
/	Division	20/10=2
%	Modulus (Remainder)	20%10=0
++	Increment	Var a=10; a++; Now a =11
--	Decrement	Var a=10; a--; Now a=9
**	Exponent	2**3=8
-X	Unary Negation	Var x=10, Var y=-x, X=10, Y=-10

Addition :

Number + Number = Number

Number + String = String

Number + Boolean = Number

String + String = String

String + Number = String

String + Boolean = String

Boolean + Number = Number

Boolean + string = String

Boolean + Boolean = Number

Number + Null = Number

Number + Undefined = NaN

String + Null = String

Boolean + Null = Number

Substraction :

Number - Number = Number

Number - String = Number if string is numeric type value. ("10" - 4)

Number - String = NaN

Number - Boolean = Number

Boolean - Boolean = Number

Anything with string is NaN

Multiplication and Division : Similar to Substraction.

Increment and Decrement :

- Increment adds the current value with one and assigns to the current value.

Syntax :

```
var x=10
```

```
x++;
```

```
x=x+1 = 11
```

Increment can be configure as post and pre increment

Post Increment :

It assigns the current value to left operand and then increments.

Syntax :

```
var x = 10;
```

```
var y = x++;
```

O/P :

```
x=11;
```

```
y=10;
```

Pre Increment :

It increments the current value and then assigns to left operand.

Syntax :

```
var x=10;
```

```
var y=++x;
```

O/P:

x=11

y=11

Decrement :

It decrements the current value by one and assigns to the current reference. It can also be defined as,

1. Post decrement :

```
var x=10;
```

```
var y=x--;
```

O/P:

x=10

y=9

2. Pre decrement :

```
var x=10;
```

```
var y=--x;
```

O/P:

x=9

y=9

Exponent Operator :

It is a new operator in ES6.

It raises the base value to the power of given value.

It is represented with “ * * “.

JS older versions can understand as

“Math.pow()”

Example :

```
var x=2;
```

```
var y=3;
```

```
x**y;           //8
```

```
Math.pow(x,y);  //8
```

Comparison Operators :

Operator	Description	Example
==	Is equal to	10==20=false
===	identical (equal and of same type)	10===20=false
!=	Not equal to	10!=20=true
!==	Not identical	20!==20=false
>	Greater than	20>10=true
>=	Greater than or equal to	20>=10=true
<	Less than	20<10=false
<=	Less than or equal to	20<=10=false

Note : Identical equal can compare only the values if they are of same type.

Syntax :

```
var x=10;
```

```
var y="10";
```

`x==y; //true`

`x===y; //false`

Bitwise operators :

Operator	Description	Example
<code>&</code>	Bitwise AND	<code>(10==20 & 20==33)=false</code>
<code> </code>	Bitwise OR	<code>(10==20 20==33)=false</code>
<code>^</code>	Bitwise XOR	<code>(10==20 ^ 20==33)=false</code>
<code>~</code>	Bitwise NOT	<code>(~10) = -11</code>
<code><<</code>	Bitwise Left Shift	<code>(10<<2) = 40</code>
<code>>></code>	Bitwise Right Shift	<code>(10>>2)=2</code>
<code>>>></code>	Bitwise Right Shift with Zero	<code>(10>>>2)=2</code>

Logical Operators :

Operator	Description	Example
<code>&&</code>	Logical AND	<code>(10==20=&&20==33)=false</code>
<code> </code>	Logical OR	<code>(10==20=&&20==33)=false</code>
<code>!</code>	Logical NOT	<code>!(10==20)=true</code>

Assignment Operators :

Operator	Description	Example
=	Assign	10+10=20
+=	Add and Assign	Var a=10; a+=20; Now a=30
-=	Subtract and assign	Var a=20; a-=10; Now a=10
=	Multiply and assign	Ar a=10; a=20; Now a=200
/=	Divide and assign	Var a=10; a/=2; Now a=5
%=	Modulus and assign	Var a=10; a%=2; Now a=0

Special Operators :

1. Ternary Operator :

It is a special operator that handles the three operands, which includes a condition, statement if true and statement if false.

Syntax :

(condition)?statement_true: statement_false

2. delete :

It is a special operator used to delete any property from an object you cannot delete the properties of built in objects like,

Math, Array, Location, Navigator etc..

Example :

```
<script>

function f1() {

var product = {

Name: "Tv",

Price: 45000.64

};

delete product.Price;

document.write("Name=" + product.Name + "<br>" + "Price=" + product.Price);

}

f1();

</script>
```

3. typeof :

It is a special operator that can verify and return the type of value stored in any reference.

Syntax :

```
var x=10;

typeof x;           //number
```

Example :

```
<script>
```

```
function f1() {  
  
    var product = {  
  
        Name: "Tv",  
  
        Price: 45000.64,  
  
        Stock: true  
  
    };  
  
    document.wrtie(`Name is ${typeof product.name} <br> Price is ${typeof  
product.Price} <br> Stock is${typeof product.Stock}`)  
  
}  
  
f1();  
  
</script>
```

4. instanceof :

It is a boolean operator that verifies the given object with specified class and return a boolean true (or) false.

Example :

```
<script>  
  
class Product {  
  
}  
  
function f1() {  
  
    var product=new Product();  
  
    var pic=new Image();  
  
    document.write(pic instanceof Image);  
  
}
```

```
document.write(product instanceof Product);  
  
}  
  
f1();  
  
</script>
```

in Operator :

It verifies whether the given property is available in specified object and returns boolean true (or) false.

Syntax :

“Property” in object

Example :

```
<script>  
  
function f1() {  
  
var product = {  
  
Name: "TV",  
  
Price:56000.66  
  
};  
  
document.wrtie(`Is ID Avail in Product? ${"Id" in product}`);  
  
}  
  
f1();  
  
</script>
```

of Operator :

It is used to verify and read all values in a collection. It is used over iterations.

Example :

```
<script>

function f1() {

var products = ["TV","Mobile"];

for(var value of products) {

document.write(value + "<br>");

}

}

f1();

</script>
```

void Operator :

It is a special character used to define that the given function (or) object doesn't return any value i.e without any return type.

Example :

```
<body>

<a href="javascript:void()">Home</a>

</body>
```

new Operator :

It is a dynamic memory allocating operator that allocates memory for any specified object and loads its members in to memory.

Example :

```
<script>
```

```
function bodyload() {  
  
var pic=new Image();  
  
pic.width="100";  
  
pic.height="100";  
  
pic.src="../images/tv.jpg";  
  
document.getElementById("conatiner").appendChild(pic);  
  
}  
  
</script>  
  
<body onload="bodyload()">  
  
<h3>Image Below</h3>  
  
<div id="container"></div>  
  
</body>
```

Statements in JavaScript

- The statements are program instructions used to control the execution flow.
- Statements are categorized in to following types based on how they control the flow.

a) Selection Statements

if, else, switch, case

b) Iteration Statements

for, while, do while

c) Jump Statements

break, continue, return

d) Exception Handling Statements

try, catch, throw, finally

Selection Statements

- Selection statements are decision making statements in computer programming.
- A program executes in sequential order you can change the execution order by using the selection statement.
- The decision making is again categorized in to following,

1. forward Jump
2. simple Decisions
3. multiple Decisions
4. multi Level

forward jump :

In this technique the statement execution will continue only when the given condition is satisfied. There will be no alternative process.

//////////////////////////////////fig//

```
if(condition/booleanExpression)
{
    statement if true;
}
```

Example : forwardjump.html [in GITHUB]

```
<!DOCTYPE html>

<html>

  <head>

    <link                                rel="stylesheet"
href="../../node_modules/bootstrap/dist/css/bootstrap.css">

    <script>

      var userDetails = {

        CardNo: "5555666688887777",

        Year:"2019",

        Cvv:"3344"

      };

      function VerifyCard() {

        var cardnumber = document.getElementById("txtCard").value;

        if(userDetails.CardNo==cardnumber) {

          document.getElementById("lstYear").disabled=false;

        }

      }

      function VerifyYear(){

        var year = document.getElementById("lstYear").value;

        if(userDetails.Year==year) {

          document.getElementById("txtCvv").disabled=false;

        }

      }

      function VerifyCvv(){
```

```
var cvv = document.getElementById("txtCvv").value;

if(userDetails.Cvv==cvv) {

    document.getElementById("btnPay").disabled=false;

}

}

</script>

</head>

<body>

    <div class="container">

        <fieldset>

            <legend class="text-primary">Payment Details</legend>

            <dl>

                <dt>Card Number</dt>

                <dd><input type="text" id="txtCard" class="form-control"
onkeyup="VerifyCard()"></dd>

                <dt>Expiry Year</dt>

                <dd>

                    <select disabled id="lstYear" class="form-control"
onchange="VerifyYear()">

                        <option>2019</option>

                        <option>2020</option>

                        <option>2021</option>

                    </select>

                </dd>

            </dl>

        </fieldset>

    </div>

</body>

</html>
```



```
<dt>CVV</dt>

<dd>

    <input      disabled      type="text"      id="txtCvv"
class="form-control" onkeyup="VerifyCvv()">

</dd>

</dl>

<button      id="btnPay"      disabled      class="btn
btn-primary">Pay</button>

</fieldset>

</div>

</body>

</html>
```

Exercise :

Hotel Registration Form

Hotel Registration Form

Customer Details

Customer Name :
Check in Date :
Total No of Days :
Total No of Persons :

Select Room

image-1

☐ Delux Room [2000/day]

image-2

☐ Suite Room [4400/day]

Select Amenities

image-1

☐ Locker [500/day]

image-2

☐ A/C [1000/day]

Advance Amount

Pay Advance

Register Room

Booking Summary

Customer Name	
Check in Date	
Total No of Days	
Total No of Persons	
Selected Room Type	
Selected Aminities	
Room Amount	
Advance Paid	
Balance Amount	

Close

Example : KFC

```
<!DOCTYPE html>

<html>

  <head>

    <link                                rel="stylesheet"
href=" ../node_modules/bootstrap/dist/css/bootstrap.css">

    <style>

      .setMargin {

        margin-top: 20px;

        background-color: maroon;

        color:white;

        width:900px;

        font-size: 2em;

        text-align: center;

      }

    </style>

    <script>

      function OrderClick() {

        document.getElementById("frmRegister").style.display="none";

        document.getElementById("frmDetails").style.display="block";
```

```
var customerName = document.getElementById("txtName").value;
```

```
var mealName;
```

```
var mealCost;
```

```
var adonName="";
```

```
var adonCost;
```

```
var optBurger = document.getElementById("optBurger");
```

```
var optRoller = document.getElementById("optRoller");
```

```
var optWings = document.getElementById("optWings");
```

```
var optKrusher = document.getElementById("optKrusher");
```

```
if(optBurger.checked) {
```

```
    mealName=optBurger.value;
```

```
    mealCost=130;
```

```
}
```

```
if(optRoller.checked) {
```

```
    mealName=optRoller.value;
```

```
    mealCost=100;
```

```
}
```

```
if(optWings.checked) {
```

```
    adonName+=optWings.value + "<br>";
```

```
    adonCost=80;
```

```
        mealCost=mealCost + adonCost;
    }

    if(optKrusher.checked) {

        adonName+=optKrusher.value + "<br>";

        adonCost=40;

        mealCost=mealCost + adonCost;

    }

    document.getElementById("lblName").innerHTML=customerName;

    document.getElementById("lblMeal").innerHTML=mealName;

    document.getElementById("lblAdon").innerHTML = adonName;

    document.getElementById("lblAmount").innerHTML = "&#8377;" +
mealCost;

    }

    function CloseClick() {

        document.getElementById("frmRegister").style.display="block";

        document.getElementById("frmDetails").style.display="none";

    }

</script>

</head>

<body>

    <div class="container">

        <div id="frmRegister">

            <div class="setMargin">

            </div>

        </div>

    </div>

</body>

</html>
```

```
</div>
```

```
<div class="setMargin">
```

```
    Customer Details
```

```
</div>
```

```
<div class="setMargin form-inline">
```

```
    <label>Customer Name</label>
```

```
    <input type="text" id="txtName" class="form-control">
```

```
</div>
```

```
<div class="setMargin">
```

```
    Select a Meal
```

```
    <div class="card-deck">
```

```
        <div class="card">
```

```
            <div class="card-body">
```

```
                
```

```
            </div>
```

```
            <div class="card-footer text-primary">
```

```
                <input        class="form-control"        type="radio"
id="optBurger" name="Meal" value="OMG Burger"> OMG Burger
```

```
                &#8377; 130/-
```

```
            </div>
```

```
    </div>
```

```
    <div class="card">
```

```
<div class="card-body">

</div>

<div class="card-footer text-primary">

    <input    class="form-control"    type="radio"
id="optRoller" name="Meal" value="OMG Roller"> OMG Roller

    &#8377; 100/-

</div>

</div>

</div>

</div>

<div class="setMargin">

    Select AD-ON

    <div class="card-deck">

        <div class="card">

            <div class="card-body">

            </div>

            <div class="card-footer text-primary">

                <input    class="form-control"    type="checkbox"
id="optWings" value="Hot Wings"> Hot Wings
```


₹ 80/-

</div>

</div>

<div class="card">

<div class="card-body">

<img
class="img-thumbnail">

</div>

<div class="card-footer text-primary">

<input class="form-control" type="checkbox"
id="optKrusher" value="Krusher Brownie"> Krusher Brownie

₹ 40/-

</div>

</div>

</div>

</div>

<div style="margin-top: 20px;" class="text-center form-group">

<button class="btn btn-primary btn-lg"
onclick="OrderClick()">Place Order</button>

</div>

</div>

<div id="frmDetails" style="display: none;">

<h2>Order Details</h2>

```
<table class="table table-hover table-danger" style="font-size:
2em;">

    <tr>

        <td>Customer Name</td>

        <td id="lblName"></td>

    </tr>

    <tr>

        <td>Selected Meal</td>

        <td id="lblMeal"></td>

    </tr>

    <tr>

        <td>Selected Ad-On's</td>

        <td id="lblAdon"></td>

    </tr>

    <tr>

        <td>Total Amount</td>

        <td id="lblAmount"></td>

    </tr>

    <tr>

        <td colspan="2" align="center">

            <button class="btn btn-danger"
onclick="CloseClick()">Close</button>

        </td>

    </tr>
```

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

Simple Decision :

In this approach the conditions are defined with an else clause, so that the statements are defined for both condition true (or) false.

Syntax :

```
if (booleanExpression)
```

```
{
```

```
statements if true;
```

```
}
```

```
else
```

```
{
```

```
statements if false;
```

```
}
```

Example : Login.html [Git Hub]

Multiple Decisions :

It is a technique where multiple alternatives are provided to handle any specified task.

It requires an “else if” clause

Syntax :

```
if (condition1)
{
    statements if condition 1 true;
}

else if (condition2)
{
    statements if condition 2 true;
}

else
{statements if all conditions false;
}
```

Example : Multiple conditions.html [GIT Hub]

Multi Level Conditions :

It is a programming technique where conditions are nested in a multi level hierarchy.

It allows to check the next conditions only when the given first condition is satisfied.

It will have an alternative for every condition.

Syntax :

```
if(condition-1) {  
  
if(condition-2) {  
  
if(contion-3){  
  
statement if all conditions true;  
  
} else {  
  
statement if condition-3 false;  
  
}  
  
} else {  
  
statement if condition-2 false;  
  
}  
  
}  
  
else {  
  
statement if condition-1 false;  
  
}
```

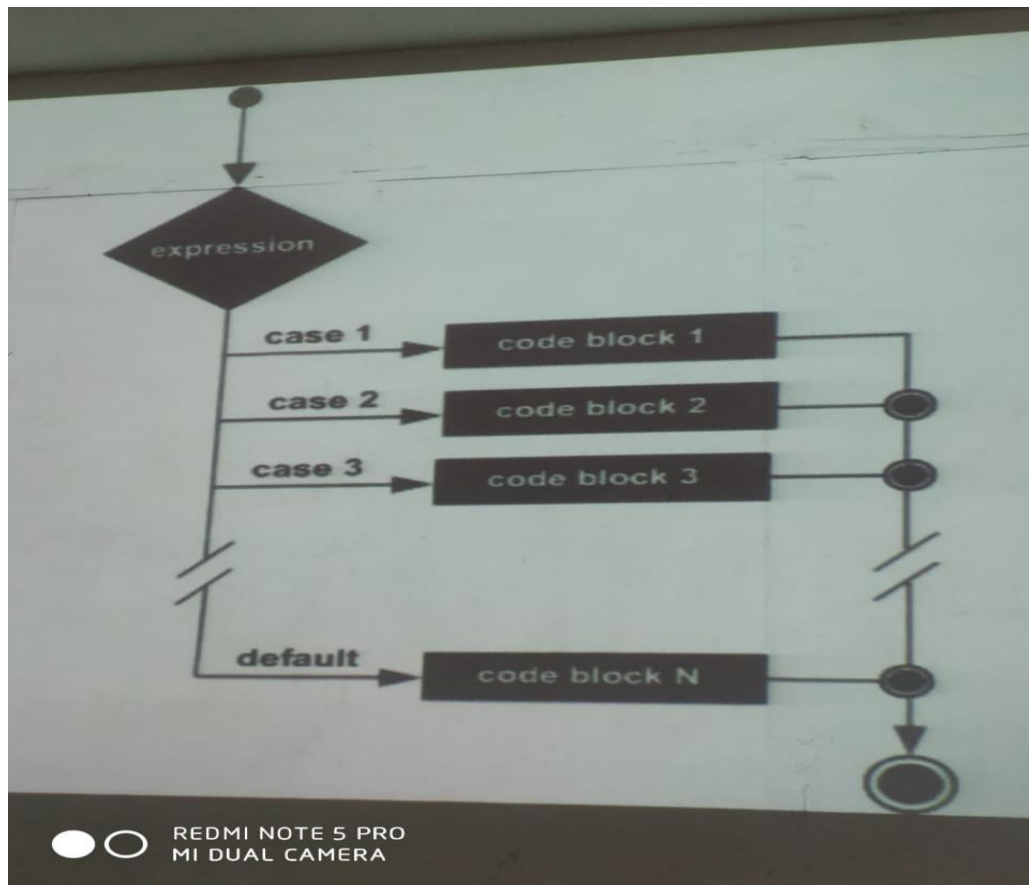
Example : MultiLevelCondition.html [GITHub]

Switch Statement :

It is a selection statement used to execute only the specified block based on given condition.

Switch in electronics is used to interrupt the flow of electrons.

A selector switch will execute only the specified without verifying the other statements defined in the context.



Syntax :

switch (expression)

{

case constant-expression:

statement

jump-statement

[default:

statement

jump-statement]

}

Example :

```
<script>

function GetClick() {

var n = document.getElementById("txtNo").value;

var msg = document.getElementById("msg");

switch(n)

{

case "1":

msg.innerHTML = "ONE";

break;

case "2":

msg.innerHTML="TWO";

break;

case "3":

msg.innerHTML="THREE";

break;

default:

msg.innerHTML="Please enter 1 to 3";

break;

}

}

</script>

<body>

Number:
```

```
<input type="text" id="txtNo">  
  
<button onclick="GetClick()">Get</button>  
  
<br>  
  
<div id="msg"></div>  
  
</body>
```

FAQ's :

1. Can you define a case with numeric values ?

A. Yes

2. Are case values "Case Sensitive" ?

A. Yes

3. How to handle multiple Case for various text-variants ?

A. Write multiple cases for single set of statements.

Example :

```
<script>  
  
function f1() {  
  
var x = "N";  
  
switch(x){  
  
case "y":  
  
case "Y":  
  
document.write("Selected Yes");
```



```
break;

case "n":

case "N":

document.write("Selcted No");

break;

default:

document.write("Choose y or n");

}

}

f1();

</script>
```

4. How to handle a case if it is a word with various text-variants ?

A. Convert the input string in to any one specified CASE i.e upper or lower.

Example :

```
var x = "YES";

switch(x.toLowerCase()) {

case "yes":

.....

break;

}
```

5. Can you define Case without break ?

A. Yes. It will stop when next break occurs.

6. Can you define switch without default ?

A. Yes.

7. Can you define default between or above the cases ?

A. Yes.

8. How you will define a case for range of values ?

A . By passing a boolean value in to switch and defining case with boolean condition.

Example :

```
<script>

function f1() {

var x = 7;

switch(true){

case (x>=1 && x<=5):

document.write("Number Between 1 to 5");

break;

case (x>=6 && x<=10):

document.write("Number Between 6 to 10");

break;

default:

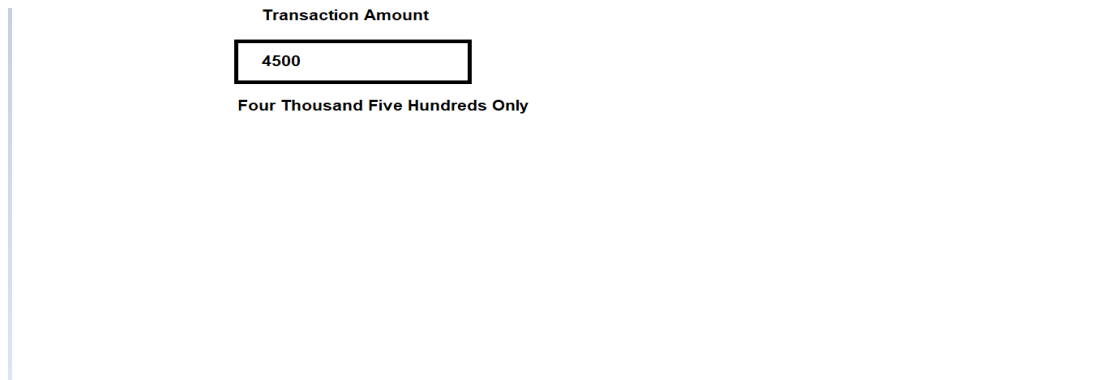
document.write("Enter number 1 to 10");

break;
```

```
}  
  
}  
  
f1();  
  
</script>
```

Task :

Write a JavaScript program to convert number into words ?



The screenshot shows a web form with a title "Transaction Amount". Below the title is a text input field containing the number "4500". Below the input field, the text "Four Thousand Five Hundreds Only" is displayed, representing the number in words.

Looping Control and Iteration Statements :

- Looping is a process of executing a set of statements repeatedly until the given condition satisfied.
- It requires an initialization, condition and counter.

//////////////////fig////////////////////////////////////

Syntax :

for(initialization; condition; counter)

```
{  
  
statement;  
  
}
```

Example :

```
for(var i=1; i<=5; i++)  
  
{  
  
document.write(i + "<br>");  
  
}
```

Counter decrement in loop :

Example :

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

Counter Step value in loop :

Example :

```
for(var i=2; i<=20; i=i+2) {  
  
document.write(i + "<br>");  
  
}
```

Skip Counter in loop :

Terminate Loop :

The looping control statements can be terminated by using any condition and jump statement i.e break (or) return.

Example :

```
<head>

<script>

var users =[

{UserName:"john123"},

{UserName:"john"},

{UserName:"john_ui"},

{UserName:"david"}

];

function VerifyUser(){

var username = document.getElementById("txtName").value;
```

```
var msg = document.getElementById("msg");

for(var i=0; i<users.length; i++) {

if(username==users[i].UserName){

msg.style.color="red";

msg.innerHTML="User Name Taken - Try Another";

break;

} else {

msg.style.color="green";

msg.innerHTML="User Name Available";

}

}

}

</script>

</head>

<body>

<fieldset>

<legend>Register User </legend>

<input type="text" id="txtName" onkeyup="VerifyUser()">

<div id="msg"></div>

</body>
```

Skip Loop Counter :

Example :

```
<script>

var products = [

{Category:"Electronics", Items: ["TV", "MOBILE"]},

{Category:"Shoes", Items:["Nike", "Lee Cooper"]}

];

function bodyload() {

var list = document.getElementById("list");

for(var i=0; i<products.length; i++) {

var li = document.createElement("li");

li.innerHTML=products[i].Category;

list.appendChild(li);

for(var j=0; j<products[i].Items.length; j++) {

var nestedLi = document.createElement("li");

nestedLi.innerHTML=products[i].Items[j];

ul=document.createElement("ul");

ul.appendChild(nestedLi);

li.appendChild(ul);

}

}

}

</script>

<body onload="bodyload()">
```

```
<ol id="list">
```

```
</body>
```

Class Example :

Source Code -

```
<head>
```

```
<!-- Link Bootstrap.css-->
```

```
<style>
```

```
#imgProduct {
```

```
Width:200px;
```

```
height: 200px;
```

```
}
```

```
#status {
```

```
width: 50;
```

```
height: 60;
```

```
position: fixed;
```

```
right: 10;
```

```
top: 10;
```

```
border: solid darkcyan 1px;
```

```
text-align: center;
```

```
cursor:grab;
```

```
}
```



```
#count{

font-weight: bold;

color: red;

font-size: 1em;

text-align: center;

}

</style>

<script>

var categories = [

"Select a Category",

"Electronics",

"shoes"

];

function bodyload(){

var lstCategories = document.getElementById("lstCategories");

for(var i=0; i<categories.length; i++) {

var option = document.createElement("option");

option.text = categories[i];

option.value=categories[i];

lstCategories.appendChild(option);

}

}

var electronics = [
```

```
"Select Electronic Product",

"Samsung Tv",

"MI Mobile"

];

var shoes = [

"Select Shoe Product",

"Nike Casuals",

"Lee Cooper Boot"

];

function AddProducts(collection){

var lstProducts=document.getElementById("lstProducts");

lstProducts.innerHTML="";

for(var i=0; i<collection.length; i++) {

var option = document.createElement("option");

option.text=collection[i];

option.value=collection[i];

lstProducts.appendChild(option);

}

}

function CategoryChanged(){

var category = document.getElementById("lstCategories").value;

switch(category){

case "Electornics":
```

```
AddProducts(electronics);

break;

case "Shoes" :

AddProducts(shoes);

break;

default:

document.getElementById("lstProducts").innerHTML="";

}

}

var productsData = [

{Name:"Samsung TV", Price:56000.66, Photo:"../images/tv.jpg"}

{Name:"MI Mobile", Price: 4000.66, Photo:"../images/mobile.jpg"}

{Name::Nike Casuals", Price:4000.66, Photo:"../images/shoe.jpg"}

{Name:"Lee Cooper Boot", Price"6000.66, Photo:"../images/shoe1.jpg"}

]

function ProductChange(){

var ProductName = document.getElementById("lstProducts").value;

window.product=productsData.filter(x=>.Name==prodyctName);

document.getElementById("lblName").innerHTML=product[0].Name;

document.getElementById("lblPrice").innerHTML="₹" + product[0].Price;

document.getElementById("imgProduct").src=product[0].Photo;

}

var cartItems=[];
```

```
function AddToCartClick(){  
  
    cartItems.push(product[0]);  
  
    var totalAmount=0;  
  
    var tbody=document.getElementById("tbody");  
  
    for(var i=0; i<cartItems.length; i++) {  
  
        var tr=document.createElement("tr");  
  
        var td1=document.createElement("td");  
  
        var td2=document.createElement("td");  
  
        var td3=document.createElement("td");  
  
        td1.innerHTML=cartItems[i].Name;  
  
        td2.innerHTML=cartItems[i].Price;  
  
  
  
        var img=new Image();  
  
        img.src=cartItems[i].Photo;  
  
        img.height=50;  
  
        img.width=50;  
  
  
        td3.appendChild(img);  
  
        tr.appendChild(td1);  
  
        tr.appendChild(td2);  
  
        tr.appendChild(td3);  
  
    }  
  
    tbody.appendChild(tr);
```

```
document.getElementById("count").innerHTML=cartItems.length;

document.getElementById("lblAmount").innerHTML=totalAmount;

}

function ShowCart(){

document.getElementById("cart").style.display="block";

}

</script>

</head>
```

while :

The while loop is used when the exact number of iterations are not known and when iteration counter may change dynamically. It will start the loop only when the condition evaluates to true.

////////////////////fig////////////////////

Syntax :

```
while (booleanExpression)

{

    counter;
```

```
statements;  
}
```

Example :

```
var i=0;  
while(i<10) {  
    i++;  
    document.write(i+ "<br>");  
}
```

do while :

It is similar to while loop but gurantee that the statements will execute atleast once even when the condition is false.

////////////////////////fig////////////////////////////////

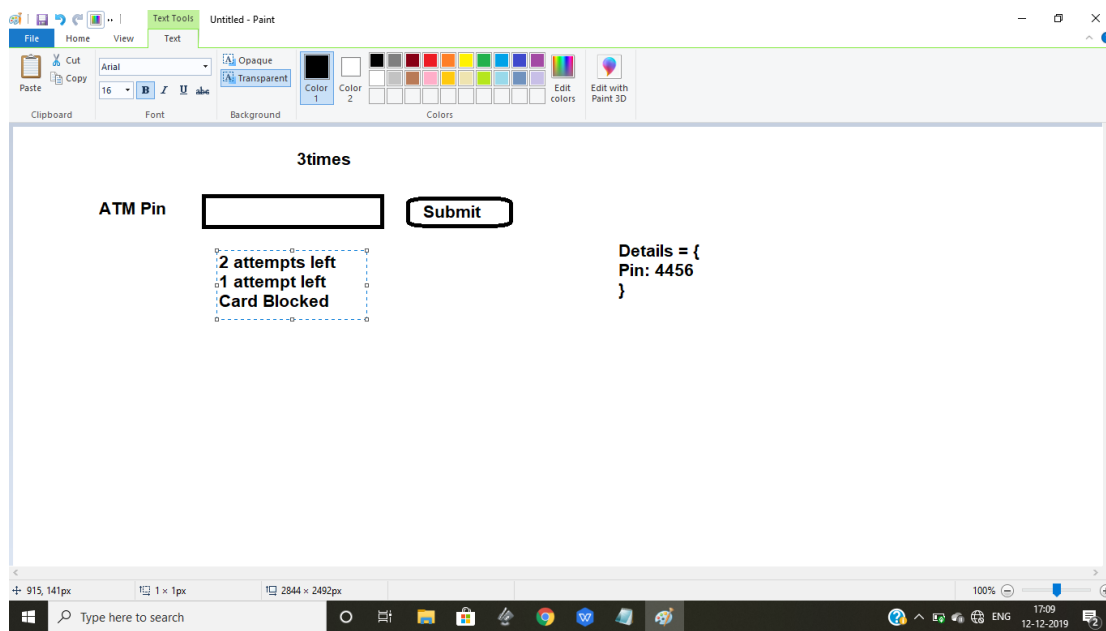
Syntax :

```
do  
{  
    counter;  
    statements;  
} while(booleanExpression);
```

Example :

```
<script>.  
  
function f1() {  
  
var i=11;  
  
do{  
  
if(i>10) {  
  
document.write("Can't Loop");  
  
} else {  
  
i++;  
  
document.write(i + "<br>");  
  
}  
  
while(i<10);  
  
}  
  
f1();  
  
</script>
```

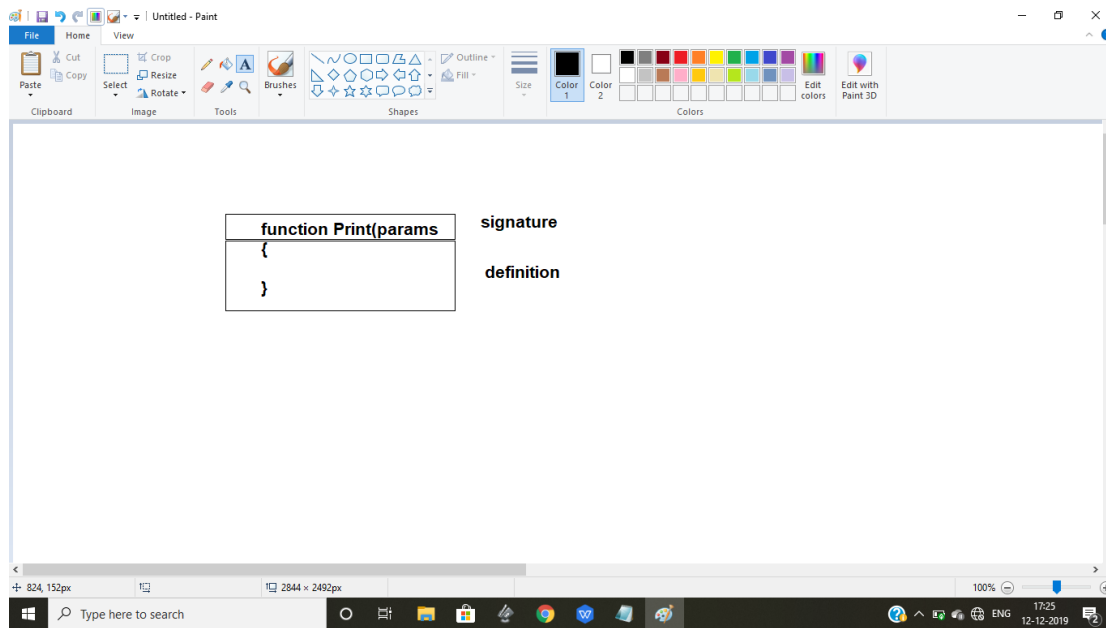
Note : Task



Try this using do and while loops and for loops

Functions in JavaScript

- Function is a refactor which encapsulates a set of statements so that you can re use the statements across the application.
- A function in JavaScript is always intended to return a value.
- The functions are defined by using the key word “function”.
- Every function have a signature and definition.



Anonymous Function :

- The functions that are defined without name are known as Anonymous Functions.
- They are mostly used in call back mechanism where functions execute automatically according to specified condition.
- Anonymous functions are defined and accessed by using the "()".

Example :

<script>

(function(){

document.write("Hello !
");

document.write("Welcome");

})();

</script>

Named Functions :

A function defined with specified name so that it can be accessed from any location by using the name.

Example :

```
<script>

function Hello(){

    document.write("Hello ! <br>");

    document.write("Welcome");

}

Hello();

</script>
```

Referred Functions :

These are the functions not defined with any name and loaded in to memory.

You can assign a memory reference for function and access it by using the reference name [not function name].

Example :

```
<script>

var hello=function() {

    document.write("Hello ! <br>");

}
```

```
hello();  
  
</script>
```

Multiple Functions with one Reference :

- You can configure a single reference in memory that can access multiple functions.
- It requires a reference type memory i.e array (or) object.

Example :

```
<script>  
  
var math = {  
  
  Add: function(){  
  
    document.write("Addition=" + (10+20));  
  
  },  
  
  Multiply: function(){  
  
    document.write("Multiply=" + (5*6));  
  
  }  
  
}  
  
math.Multiply();  
  
math.Add();  
  
</script>
```

Function Parameters :

- A parameter less function is used to perform the same functionality for every situation.
- A parameterized function can modify the functionality according to situation.
- A parameter is defined in function signature is known as “Formal Parameter”.
- The parameters passed while calling the function are known as “Actual Parameters”.

Example :

```
<script>

function PrintNumbers(howMany){

for(var i=1; i<howMany; i++) {

document.write(i + "<br>" );

}

}

function PrintMessage(userName) {

document.write(`Hello ! ${userName} <br>`);

}

PrintNumbers(5);

PrintMessage("John");

</script>
```

Multiple Parameters :

- A Function can be defined with more than one parameter.

- Every parameter is responsible for modifying specific value in a function.
- Every formal parameter is mandatory and it have order dependency.
- The parameters type depends on the value specified.

Example :

```
<style>
```

```
div {
```

```
    border:2px darkcyan solid;
```

```
    margin-top: 10px;
```

```
}
```

```
</style>
```

```
<script>
```

```
function PrintProduct(name, price, mfd){
```

```
    document.write("<div>Name=" + name + "<br>" + "Price" + price + "<br>" +  
    "Manufactured=" +mfd.toLocateDateString() + "</div>");
```

```
}
```

```
PrintProduct("Samsung TV",40000.53, new Date("2019/02/10"));
```

```
PrintProduct("Nike Casuals",4300.56,new Date("2019/10/10"));
```

```
</script>
```

Optional Parameters :

JavaScript doesn't support optional parameters, you have to configure the parameters and verify them by using the undefined type.

Example :

```
<script>

function PrintProduct(name, price){

if(price==undefined) {

document.write("<div>Name=" + name + "</div>");

} else {

document.write("<div>Name=" + name + "<br>" + "Price=" + price + "</div>");

}

}

PrintProduct("Samsung TV");

PrintProduct("Nike Casuals", 5600.66);

</script>
```

Array Parameters :

An array parameter is reference type parameter which can handle a list of values in any function.

The actual values in to parameter can be passed by using array type meta character “[]” (or) “Array()”.

Example :

```
<script>

function PrintList(yourList){

for(var item of yourList) {

document.write(item + "<br>");

}

}
```

```
}  
  
var electronics=["TV","Mobile"];  
  
PrintList(electronics);  
  
PrintList(["Watc", "Sunglasses"]);  
  
PrintList(new Array("Nike","Lee Cooper"));  
  
</script>
```

Rest Parameters :

- A rest Parameter allows a function to define one formal parameter which can access multiple values.
- ES-5 supports rest parameters define by using “...”

Example :

```
<script>  
  
function PrintList(...yourList){  
  
for(var item of yourList){  
  
document.write(item + "<br>");  
  
}  
  
}  
  
PrintList("TV", "Mobile", "Nike");  
  
</script>
```

Note :

1. Every function can be defined with only one rest parameter.
2. Rest parameter must be the last parameter in formal list.

Function with Return Value :

The JavaScript functions can be defined with a return value so that they can execute the given statements, evaluate a value and store the return value.

Syntax :

```
function Name(params) {  
  
    return value;  
  
}
```

FAQ :

1. Can a function have multiple return values ?

A. Yes.

Ex :

```
<script>  
  
function Login(pwd) {  
  
    if(pwd=="admin") {  
  
        return "success..";  
  
    } else {  
  
        return "invalid";  
  
    }  
}
```



```
}  
  
}  
  
document.write(Login("admin"));  
  
</script>
```

Function Recursion :

Recursion is a technique where a function is defined with relative access within the scope i.e a function called within the same function.

Example :

```
<script>  
  
function fact(n) {  
  
if(n<=0) {  
  
return 1;  
  
} else {  
  
return n * fact(n-1);  
  
}  
  
}  
  
document.write(fact(6));  
  
</script>
```

OOP in JavaScript

- Object in computer programming comprises of Logic and Data.
- The Data is stored in properties and the Logic in functions.
- The concept of object was introduced in early 1950's by Alan Kay.

- The re-usability of objects was designed in early 1960's by Johan Olay and Kristien Nyaggard with a language called SIMULA.
- The code separation with re-usability was designed in early 1970's by Trygve. He introduced a framework called MVC [Model View Controller].

Features of OOP :

- ◆ Code Re-usability
- ◆ Code Security
- ◆ Code Separation
- ◆ Code Extensibility
- ◆ Code Maintainability and
- ◆ Code Testability

Drawbacks of OOP :

- ✓ Uses more memory
- ✓ Very complex in configurations
- ✓ Slow in responding
- ✓ Heavy applications
- ✓ Cant directly interact with hardware

Characteristics of OOP :

- Inheritance
- Polymorphism
- Encapsulation

- Abstraction

Class in JavaScript :

- Class is an structural entity with data and logic.
- It is an logical entity that provides a set of members that are used to store data and handle the logic.
- Techniqually a class is blue print for creating of objects.
- It comprises of three members

A. Constructor

B. Properties

C. Methods

- Properties configure the data.
- Methods configure the functionality.
- Constructor defines initialization.
- A class is created by using the key word “class”.
- The members in a class are accessible within the class by using the keyword “this” and outside the class they are accessible by using an object.

Syntax :

```
class className {  
  
  constructor() {}  
  
  property;  
  
  method() {};  
  
}  
  
var obj = new className();
```

Example :

```
<style>
```

```
dl {
```

```
width:200px;
```

```
border:2px darcyan solid;
```

```
border-radius:10px;
```

```
margin:10px;
```

```
float:left;
```

```
}
```

```
dt{
```

```
font-weight:bold;
```

```
background-color:light cyan;
```

```
}
```

```
</style>
```

```
<script>
```

```
class Product {
```

```
    Name="";
```

```
    Price=0.0;
```

```
    Qty=0;
```

```
    Total(){
```

```
        return this.Qty * this.Price;
```

```
};
```

```
Print(){  
  
    document.write(`  
  
        <dl>  
  
        <dt>Name</dt>  
  
        <dd>${this.Name}</dd>  
  
        <dt>Price</dt>  
  
        <dd>${this.Price}</dd>  
  
        <dt>Quantity</dt>  
  
        <dd>${this.Qty}</dd>  
  
        <dt>Total</dt>  
  
        <dd>${this.Total()}</dd>  
  
    </dl>  
  
    `);  
  
    }  
  
    }  
  
    var tv = new Product;  
  
    tv.Name="Samsung TV";  
  
    tv.Price=35400.55;  
  
    tv.Qty=2;  
  
    tv.Print();  
  
    var shoe = new Product;  
  
    shoe.Name="Nike Casuals";  
  
    shoe.Price=4200.44;
```

```
shoe.Qty=3;
```

```
shoe.Print();
```

```
</script>
```

Constructor :

- ✓ Constructor is a special type of method that executes automatically for every object.
- ✓ Every class has an implicit constructor.
- ✓ You can define an explicit constructor by using the keyword constructor.
- ✓ The constructor methods are anonymous i.e. they cannot contain a name; the class name is given to the constructor.
- ✓ Every class can have only one explicit constructor.

Example :

```
<script>
```

```
class Database{
```

```
constructor(){
```

```
document.write("Connected to Database....<br>");
```

```
}
```

```
Insert(){
```

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

```
}
```

```
Delete(){
```

```
document.write("Record Deleted<br>");  
  
}  
  
}  
  
function InsertClick(){  
  
var obj = new Database;  
  
obj.Insert();  
  
}  
  
function DeleteClick(){  
  
var obj = new Database;  
  
obj.Delete();  
  
}  
  
</script>  
  
<body>  
  
<button onclick="InsertClick()">Insert</button>  
  
<button onclick="DeleteClick()">Delete</button>  
  
</body>
```

Parameterized Constructor :

A Constructor is by default parameter less.

You can configure a parameterized constructor to modify the its functionality.

The parameters in to constructor are passed while allocating memory for class.

Example :

```
<script>

class Database {

constructor(dbName){

document.write(`Connected to ${dbName} Database...<br>`);

}

Insert(){

document.write("Record Inserted<br>");

}

Delete(){

document.wrtie("Record Deleted<br>");

}

}

function InsertClick(){

var db=document.getElementById("lstDb").value;

var obj=new Database(db);

obj.Insert();

}

function DeleteClick(){

var db=document.getElementById("lstDb").value;

var obj = new Database(db);

obj.Delete();

}

</script>
```



```
<body>

<div>

<fieldset>

<legend>Select Database</legend>

<select id="lstDb">

<option>Oracle</option>

<option>Mysql</option>

<option>MongoDB</option>

</select>

<button onclick="InsertClick()">Insert</button>

<button onclick="DeleteClick()">Delete</button>

</fieldset>

</div>

</body>
```

Code Re Usability :

- It is one of the key feature of Object Oriented Programming.
- It is a technique that allows to re-use the code without re-writing the code.
- It can be achieved by using following techniques -
 1. Aggregation
 2. Inheritance

Aggregation :

It is a technique that allows to access the members of one class in another class without creating any relationship between classes. It uses “Object-to-Object” communication and is known as “Has-A-Relation”.

Example :

```
<script>
```

```
class Camera {
```

```
    Pixels;
```

```
    Print(){
```

```
        document.write(`Camera - ${this.Pixels}<br>`);
```

```
    }
```

```
}
```

```
class Memory {
```

```
    RAM;
```

```
    Print(){
```

```
        document.write(`Memory - ${this.RAM}<br>`);
```

```
}  
  
}  
  
class Mobile {  
  
Print(){  
  
var camera = new Camera();  
  
var memory = new Memory();  
  
camera.Pixels="20px";  
  
memory.RAM="4GB";  
  
document.write(`Mobile - Samsung j7 <br>`);  
  
camera.Print();  
  
memory.Print();  
  
}  
  
}  
  
var mobile = new Mobile();  
  
mobile.Print();  
  
</script>
```

Inheritance :

It is a technique that allows to re-use the code without re-writing the code by configuring relationship between classes, which is known as “Is-A-Relation”.

A class can extended another class by using the keyword “extends”.

The newly extended class is known as “Derived Class” and existing class is known as “Super Class”.

The members of super class are accessible in derived class by using the keyword “super”.

Every class can extended only one super class, it will support multilevel inheritance but not multiple inheritance.

Syntax :

```
class Base {  
  
}  
  
class Derived extend Base {  
  
}
```

Note :

Multiple inheritance is not supported for classes because of Dead Lock's in constructor.

Dead Lock is a situation that allows only one super class constructor in to process, which is not possible when there are multiple super classes.

Example :

```
<script>  
  
class Camera {
```

```
Pixels = "20px";

Print(){

document.write(`Camera - ${this.Pixels}<br>`);

}

}

class Memory extends Camera {

RAM = "4GB";

Print(){

super.Print();

document.write(`Memory - ${this.RAM}<br>`);

}

}

class Mobile extends Memory {

Print(){

document.write(`Mobile - Samsung j7 <br>`);

super.Print();

}

}

var mobile = new Mobile();

mobile.Print();

</script>
```

Note : A super class constructor is shared to derived class hence avoid using a constructor in derived class.

Polymorphism

- The term polymorphism refers to many forms.

Poly = Many

Morphos = forms

- In computer programming polymorphism is a technique where single base class object is designed to use the memory of multiple derived classes.
- Polymorphism allows a single base class object to handle multiple functionalities.

Example :

```
<script>
```

```
class Employee {
```

```
  firstName="";
```

```
  lastName="";
```

```
  Print(){
```

```
    document.write(`${this.firstName} ${this.lastName} <br>`);
```

```
  }
```

```
}
```

```
class Developer extends Employee {
```

```
  firstName="Raj";
```

```
  lastName="Kumar";
```

```
  Print(){
```

```
    document.write(`${this.firstName} ${this.lastName} - Developer <br>`);
```

```
}  
  
}  
  
class admin extends Employee {  
  
    firstName="Tom";  
  
    lastName="Hanks";  
  
    Print(){  
  
        document.write(`${this.firstName} ${this.lastName} - Admin <br>`);  
  
    }  
  
}  
  
class Manager extends Employees{  
  
    firstName="Rakesh";  
  
    lastName="Kumar";  
  
    Print(){  
  
        doucment.write(`${this.firstName} ${this.lastName} - Manager <br>`);  
  
    }  
  
}  
  
var employees = new Array();  
  
employees[0] = new Developer();  
  
employees[1] = new Admin();  
  
employees[2] = new Manager();  
  
  
for(var i=0; i<employees.length; i++) {  
  
    employees[i].Print();  
  
}
```

```
}
```

```
</script>
```

Event Handling in JavaScript

Object Oriented Programming have three specifications

- a. Properties
- b. Methods
- c. Events
 - Properties contain data.
 - Method defines a functionality.
 - Event specifies when the functionality need to be performed.

Event is a message sent by sender to its subscriber in order to notify the change. It follows a software design pattern called “Observer”.

Events in JavaScript are handled by a browser object.

Syntax :

```
function InsertClick()
```

```
{
```

```
}
```



```
<button onclick="InsertClick">
```

Event handler is function pointer that uses Delegate mechanism i.e pointing towards the same signature function.

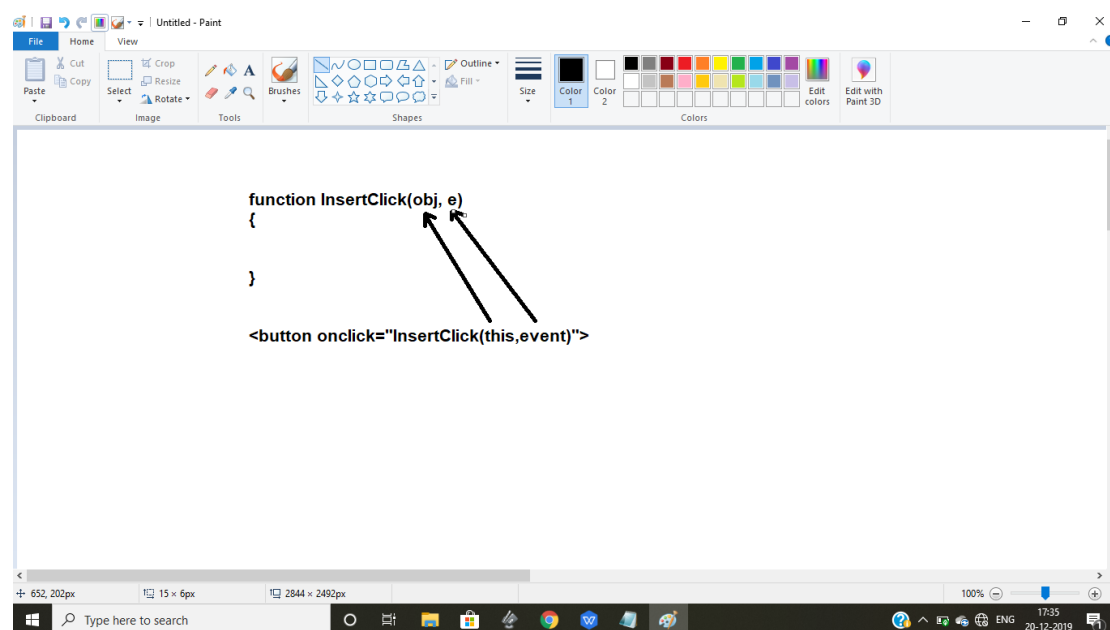
Every event handler function can be defined with two parameters

- this
- event

The parameter “this” can give access to all properties and methods of current object.

The “event” parameter can give access to all event related properties and methods.

Syntax:



The object Properties are name, value, id, className, style etc.,

The event properties are clientX, clientY, shiftKey, altKey etc.,

Example :

```
<script>

function btnClick(obj){

qwitch(obj.value) {

case "Insert" :

document.write("Record Inserted");

bresk;

case "Delete":

document.write("Record Deleted");

}

}

</script>

<body>

<button value="Insert" onclick="btnClick(this)">Insert</button>

<button value="Delete" onclick="btnClick(this)">Delete</button>

</body>
```

/////////////////////////////////INCOMPLETE////////////////////////////////

oncut

oncopy

onpaste Specifies actions to perform when text is copied, cut
 or pasted in to element.

Example :

```
<head>
```

```
<script>
```

```
function Copy() {
```

```
document.getElementById("status").innerHTML="Mobile No Copied";
```

```
}
```

```
function Cut() {
```

```
document.getElementById("status").innerHTML="Mobile No is in Memory";
```

```
}
```

```
function Paste() {
```

```
document.getElementById("status").innerHTML="Mobile No Inserted";
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<div>
```

Mobile:

```
<input oncopy="Copy()" oncut="Cut()" onpaste="Paste()" value="9995578483"  
type="text" id="txtMobile">
```

```
<span id="status"></span>
```

```
</div>
```

```
</body>
```

FAQ :

Q. How to disable right click and selection in a page ?

A. By configuring the following events to return false.

B. Oncontextmenu - for right click

Onselectstart - for drag and select

On select -- for double click and select

Example :

```
<head>
```

```
<script>
```

```
  document.oncontextmenu = function(){
```

```
    return false;
```

```
  }
```

```
</script>
```

```
</head>
```

```
<body oncontextmenu="return false" onselectstart="return false" onselect="return false" oncopy="return false">
```

```
<h1>You can't right click or copy our content </h1>
```

```
</body>
```

onsubmit

onreset these are the events associated with form element, which are used to specify actions to perform when user clicks submit (or) reset buttons.

Note : A form can be submitted only by using generic buttons like

```
<input type="submit">
```

```
<button></button>
```

A form can be reset only by using generic buttons like

```
<input type="reset">
```

Example :

```
<head>
```

```
<script>
```

```
function SubmitClick(){
```

```
  alert("Form Data will be submitted..");
```

```
}
```

```
function ResetClick(){
```

```
  alert("Form will reset");
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>

<form onsubmit="SubmitClick()" onreset="ResetClick()">

Name :

<input type="text" name="txtName">

<input type="submit" value="Submit">

<input type="reset" value="Reset">

</form>

</body>
```

Timer Events :

1. setTimeout() :

It is a javascript timer event used for synchronization task for memory.

It is used to load any task in to memory and lock the task for a specific duration. It will release the task in to process after the specified time interval.

Syntax :

```
setTimeout(function(){}, timeInterval);
```

2. clearTimeout() :

It is a timer event used to kill the task in memory. It can remove the specified task from memory before execution.

Syntax :

```
clearTimeout(functionreference);
```

Example :

```
<head>

<script>

function msg1() {

document.getElementById("msg").innerHTML="Hello !";

}

function msg2() {

document.getElementById("msg").innerHTML="How are you?";

}

function msg3() {

document.getElementById("msg").innerHTML="Welcome to JavaScript";

}

function bodyload(){

window.m1=setTimeout(msg1,2000);

window.m2=setTimeout(msg2,4000);

window.m3=setTimeout(msg3,6000);

}

function CancelClick() {

clearTimeout(m2);

}

</script>

</head>

<body onload="bodyload()">
```



```
<h2 align="center" id="msg"></h2>

<button onclick="CancelClick()">cancel msg2 </button>

</body>
```

setInterval :

It is a timer function which loads the given task in to memory and executes at regular time intervals until the task is erased from memory.

You can erase the task from memory by using clear interval.

Syntax :

```
setInterval(function(){}, timeInterval);
```

```
clearInterval(functionReference);
```

Example :

```
<head>

<script>

function Clock(){

var now=new Date();

document.getElementById("msg").innerHTML=now.toLocaleTimeString();

}

function bodyload(){

window.time=setInterval(Clock,1000);

}

function StopClick(){
```

```
clearInterval(time);

}

function StartClick(){

window.time=setInterval(Clock,1000);

}

</script>

</head>

<body onload="bodyload()">

<h2 align="center" id="msg"></h2>

<button onclick="StopClick()">Stop</button>

<button onclick="StartClick()">Start</button>

</body>
```

Example :

Interval and Clear Interval

```
<head>

<!.... link bootstrap ...>

<script>

var products = [

{Name : "Samsung TV", Price:56000.66, Photo:"../images/tv.jpg"}

.....
```

```
];

function ShowImage(index){

document.getElementById("lblName").innerHTML=products[index].Name;

document.getElementById("lblPrice").innerHTML=products[index].Price;

document.getElementById("imgProduct").innerHTML=products[index].Photo;

}

var i=0;

function SlideShow(){

i++;

ShowImage(i);

if(i==products.length) {

clearInterval(show);

document.getElementById("status").innerHTML="Slide Show - End";

}

}

function PlayClick(){

window.show=setInterval(SlideShow,5000);

document.getElementById("status").innerHTML="Slide Show - Running";

}

function PauseClick(){

clearInterval(show);

document.getElementById("status").innerHTML="Slide Show - Paused";

}
```

```
</script>

</head>

<body onload="ShowImage(0)">

<div class="container">

<h2 class="text-center text-primary">

Amazon Products

<br>

(<span id="status">Manual</span>)

</h2>

<div class="card text-center">

<div class="card-header">

<h2 id="lblName"></h2>

</div>

<div class="card-body">

<img width="300" height="200" id="imgProduct">

</div>

<div class="card-footer">

<h3 id="lblPrice"></h3>

<button onclick="PayClick()" class="btn btn-primary"><font
face="webdings">4</font></button>

<button onclick="PauseClick()" class="btn btn-danger"><font
face="webdings">;</button>

</div>

</div>
```

</div>

</body>

Example :

Progress using timer event

<head>

<style>

progress {

width:400px;

}

</style>

<script>

var i=0;

function Download(){

i = i+2;

document.getElementById("lblStatus").innerHTML=i + "% Completed";

}

function StartDownload() {

window.x=setInterval(Download,1000);

}

function PauseDownload() {

clearInterval(x);

document.getElementById("lblStatus").innerHTML= i + "% Paused";

```
}  
  
</script>  
  
</head>  
  
<body>  
  
<fieldset>  
  
<legend>  
  
Download Software  
  
<br>  
  
<div id="lblStatus"></div>  
  
</legend>  
  
<progress id="download" min="1" max="100" value="0"></progress>  
  
<br><br>  
  
<button onclick="StartDownload()">Start Download</button>  
  
<button onl=click="PauseDownload()">Pause Download</button>  
  
</fieldset>  
  
</body>
```

JavaScript Browser Objects

1. window :

It provides a set of properties and methods that are used to control the browser window. The methods are,

Method	Description
--------	-------------

open()	It opens the specified document (or) URL in a new window.
close()	It closes the browser window.
print()	It invokes the printer properties in order to print the current page.

Syntax :

```
window.open("path/url" "title", "width=100 height=100");
```

```
window.close();
```

```
window.print();
```

Example :

```
<head>
```

```
<script>
```

```
function OpenPic() {
```

```
window.open("../images/tv.jpg", "Samsung", "width=400 height=400");
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```

```

```
<br>
```

```
<p>Double Click on Image to View Large </p>
```

```
<button onclick="window.close()">Close Browser</button>
```

```
<button onclick="window.print()">Print Page</button>
```

```
</body>
```

2. Location Object :

The javascript location object provides a set of properties and methods that are used to access the client location details which includes the following,

Member	Description
host or hostname	It returns the server name (or) IP address.
pathname	It returns the page path.
href	It returns complete url of current requested page.
hash	It returns the current hash location, which is “#” name.
search	It returns the query string.
protocol	It returns the current protocol i.e file, http, https etc.,
port	It returns the port number. [if any]
reload	It reloads the current page.

Syntax :

```
location.host;
```

```
location.href;
```

Example :

```
<head>
```

```
<style>
```



```
dt{

font-weight: bold;

background-color: lightyellow;

}

</style>

<script>

function GetDetails(){

document.getElementById("server").innerHTML=location.host;

var protocol = location.protocol;

document.getElementById("protocol").innerHTML=`Your Protocol is ${protocol} <br>
${(protocol=="file:")?'You are accessing from File Server':'You are accessing from
Web Server'}`;

document.getElementById("url").innerHTML=location.href;

document.getElementById("path").innerHTML=location.pathname;

}

</script>

</head>

<body>

<button onclick="GetDetails()">Get Client Location Details</button>

<button onclick="location.reload()">Reload</button>

<dl>

<dt>Server/IP Address</dt>

<dd id="server"></dd>

<dt>Protocol</dt>
```

```
<dd id="protocol"></dd>
```

```
<dt>URL</dt>
```

```
<dd id="url"></dd>
```

```
<dt>Page Path</dt>
```

```
<dd id="path"></dd>
```

```
</dl>
```

```
</body>
```

Example : Location search to access Query String

```
<head>
```

```
<style>
```

```
div {
```

```
margin-top: 20px;
```

```
}
```

```
</style>
```

```
<script>
```

```
var tutorial = [
```

```
{
```

```
Topic : "JavaScript",
```

```
Description : "JavaScript is a light weight language."
```

```
},
```

```
{
```

```
Topic : "Web Technologies"
```

```
Description : "it comprised of various tools and languages to build progressive web applications"
```

```
}  
  
];  
  
function GetQueryString(){  
  
var querystring=location.search;  
  
var word=querystring.substring(querystring.indexOf("=") + 1);  
  
var results = [];  
  
results = tutorial.filter(x=>x.Topic==word);  
  
document.getElementById("result").innerHTML=`Searched For : ${word} <br>  
${results[0].Description}`;  
  
}  
  
</script>  
  
</head>  
  
<body>  
  
<div align="center">  
  
<form>  
  
<div>  
  
<font size="6" face="Arial">Google</font>  
  
</div>  
  
<div>  
  
<input type="text" name="search" size="40" id="txtSearch">  
  
</div>  
  
<div>  
  
<button>Search Web </button>  
  
<input type="button" value="Get Details" onclick="GetQueryString()">
```

```
</div>

<div id="result">

</div>

</form>

</div>

</body>
```

Location Hash :

It is a location property is used to access current hash location, which refers to any named location in page.

Example :

Goto “Intradocument links example “

Add a button control in toolbar

```
<button class="btn btn-primary" onclick="GetLocation()">Recently Viewed
</button>
```

Add following in head section

```
<script>

function GetLocation() {

var loc=location.hash;

switch(loc) {

case "#html":

alert("You Viewed HTML Tutorial");

break;
```

```
case "#css":  
  
    alert("You recently viewed CSS Tutorial");  
  
    break;  
  
case "#js":  
  
    alert("Your last topic was JavaScript");  
  
    break;  
  
}  
  
}  
  
</script>
```

Navigator Object

The JavaScript navigator object provides a set of properties and methods that are used to access the client browser details. It includes the following,

Member	Description
appName	Browser family. [netscape]
appVersion	Browser version.
userAgent	Supported on various environments.
mimeTypes[]	File types supported.
plugins[]	List of plugins supported.
language	Current browser language.
onLine	Status online (or) offline
systemLanguage	Language used on local computer.

cookieEnabled	Cookie status, blocked (or) Enabled.
---------------	--------------------------------------

Syntax :

navigator.cookieEnabled;

navigator.language;

Example :

<head>

<script>

function bodyload(){

document.getElementById("msg").innerHTML=`Cookies Status :

\${(navigator.cookieEnabled)?"Cookie Enabled":"Blocked - Please Enable
Cookies"}

your Browser Family : \${navigator.appName}

your Browser Language : \${navigator.language}

Your Browser is supported on : \${navigator.userAgent}`;

}

</script>

</head>

<body onload="bodyload()">

<div id="msg">

</div>

</body>

Example : Program to find all plugins

```
function bodyload() {  
  
for(var item of navigator.plugins) {  
  
alert(item.name);  
  
}  
  
}  
  
<body onload="bodyload()"></body>
```

Example : To Verify Plugin

```
<script>  
  
function bodyload(){  
  
if(navigator.plugins["Chrome PDF Viewer"]==undefined) {  
  
alert("PDF Plugin not found - Please Install");  
  
location.href="http://www.adobe.co/download";  
  
} else {  
  
alert("You can view PDF Document");  
  
}  
  
}  
  
</script>  
  
<body onload="bodyload()">  
  
<a href="../Docs/cssdemo.pdf">Tutorial</a>  
  
</body>
```

Note : To Get MIME types

```
navigator.mimeTypes[0].type
```

History Object

It is a browser object used to access the client browsing history details. It includes the following members,

Member	Description
length	Total count of pages in history of browser.
back()	Moves on level back.
forward()	Moves one level forward.
goto()	Moves to specific page in history. goto(2) - forward 2 goto(-2) - back 2 goto('home.html')

Syntax :

```
<button onclick="history.back()">Back </button>
```


JQuery

- JQuery is an JavaScript library that provides predefined functionalities to handle client side interactions in web applications.
- JQuery was introduced in 2016 by John Resig.
- JQuery provides functions to manipulate CSS and HTML. It's intention is write less and do more.
- JQuery can handle browser compatibility issues implicitly.
- JQuery provides cross browser functionality so that its functions are executable from any location [Browser] .

Installing JQuery :

1. Open your project location in command prompt
2. C:\Amazon>
3. Type the command
4. > npm install jquery

Node_modules

Jquery

Dist

Jquery.js

First JQuery Program :

1. Create a new HTML page.
2. Add JQuery Library using

<script>

3. Write JQuery functionality in another

<script>

4. `$()` to invoke the JQuery library.

5. `ready()` to identify that page loaded.

6. `function()` a callback to handle any functionality.

Example :

<head>

<script src="../../node_modules/jquery/dist/jquery.js"></script>

<script>

`$(document).ready(function() {`

`document.write("Welcome to JQuery");`

`})`

</script>

</head>

<body>

</body>

JQuery Selectors :

A selector is used to refer HTML elements dynamically. The commonly used selectors are,

a) Type selector `$("h2")`

b) Id selector `$("#id")`

c) Class selector `$(".class")`

Example :

```
<head>

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

<script>

$(document).ready(function(){

$("h2").text("Heading accessed by using Type Selector");

$("#para").text("Paragraph using ID");

$(".divClass").text("Div Using Class Selector");

})

</script>

</head>

<body>

<h2></h2>

<p id="para"></p>

<div class="divClass"></div>

</body>
```

Note :

JQuery cannot access attribute of HTML element directly. It requires the function `"attr()"`.

Syntax :

```
$("img").attr("src", "../images/tv.jpg");
```

```
$("td").attr("colspan",2);
```

Example : Accessing the HTML elements and attributes.

```
<head>

<link bootstrap.css>

<script scr="..jquery.js"></script.

<script>

var product = {

Name : "Samsung TV",

Price : 45000,

Photo : "../images/tv.jpg"

};

$(document).ready(function(){

$("#prodName").text(product.Name);

$("#prodPrice").text(product.Price);

$("#prodImg").attr("src",product.Photo);

})

</script>

</head>

<body>

<div class="container">

<div class="card">

<div class="card-header">

<h2 id="prodName"></h2>
```

```
</div>

<div class="card-body">

<img id="prodImg" width="200" height="200">

</div>

<div class="card-footer">

<h2 id="prodPrice"></h2>

</div>

</div>

</div>

</div>

</body>
```

JQuery HTML and CSS Functions

1. html() :

It is a jquery function used to append text in to any html container with formats [innerHTML].

Syntax :

```
$("p").html("<b> Bold </b>");
```

2. text() :

It is similar to html function but will not support format for texts. It is used for presenting RC Data elements.

Syntax :

```
$("p").text("<b> Bold </b>");
```

3. css() :

It is a jquery function used to apply any CSS effect to html element.

Syntax :

```
$("#p").css("Property", "value");
```

```
$("#p").css({property:value, ...});
```

Example :

```
<head>
```

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

```
<script>
```

```
$(document).ready(function(){
```

```
$("#p1").text("<b> This is Inner Text </b>");
```

```
$("#p2").html("<b> This is Inner HTML </b>");
```

```
$("#p3").html("Jquery with Single css Effect").css("color", "red");
```

```
$("#p4").html("Jquery with multiple CSS").css({color:"red",  
'background-color':"yellow"});
```

```
})
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<p id=p1"></p>
```

```
<p id="p2"></p>
```

```
<p id="p3"></p>
```

```
<p id="p4"></p>
```

</body>

4. val() :

It is a jquery property used to access the value of any html input element. It can also set a value in to the element.

Syntax :

```
$("#txtName").val("john");
```

```
var name=$("#txtName").val();
```

5. attr() :

It is used to access any attribute of html element, which is not available as a property.

Syntax :

```
$("#img").attr("src")="images/pic.jpg";
```

Example :

```
<head>
```

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

```
<script>
```

```
$(document).ready(function(){
```

```
$("#btnSubmit").click(function(){
```

```
var name=$("#txtName").val();
```

```
var city=$("#lstCity").val();
```

```
$("#details").html(`

Name



Stock



Shipped To



Delhi



HYD



Details


```


</body>

Iterations in JQuery :

JQuery can use all JavaScript iterators but it is in-built provided with the function “.each()” which can read values and keys from a collection.

Syntax :

```
$.each(collection, function(key, value){//print key - value })
```

append() :

It is used to append a new element (or) value to an existing element without overriding the existing content.

Syntax :

```
$("p").append("some text/element")
```

Example :

```
<head>
```

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

```
<script>
```

```
$(document).ready(function(){
```

```
var products=["TV","Mobile","Shoe"];
```

```
$.each(products, function(key, value){
```

```
$("#list").append(`[${key}] ${value}<br>`);
```

```
})
```

```
})  
  
</script>  
  
</head>  
  
<body>  
  
<div id="list">  
  
</div>  
  
</body>
```

prepend() :

It is a jquery function used to add any new content in to existing content as prefix.

remove() :

It is used to remove any element dynamically from the DOM hierarchy.

appendTo() :

It adds a new element in to existing element by using parent and child hierarchy.

Example :

```
<head>  
  
<script src=..jquery.js></script>  
  
<script>  
  
$(document).ready(function(){  
  
$("#btnAdd").click(fucntion){
```

```
$("#<h2>Details Added to Page</h2>").appendTo("#details");

})

$("#btnRemove").click(function(){

$("#h2").remove();

})

})

</script>

</head>

<body>

<div>

<button id="btnRemove">Remove</button>

<button id="btnAdd">Add</button>

</div>

<div id="details">

</div>

</body>
```

Example :

Write a program to add collection as List Item in to

```
var products = ["TV", "Shoe", "Mobile"]
```

```
<ol id="list"> </ol>
```

```
$(document).ready(function()  
  
})
```

Answer :

```
<script>  
  
var products = ["TV", "Shoe", "Mobile"];  
  
$(document).ready(function(){  
  
$.each(products, function(key, value);  
  
$(' <li>${value}</li> ').appendTo("#list");  
  
})  
  
})  
  
</script>  
  
<body>  
  
<ol id="list"></ol>  
  
</body>
```

Example :

Program to Add only Names in to List

```
var products = [  
  
{ Name : "TV", Price : 45000.55},  
  
{ Name : "Mobile", Price : 12000.44}  
  
];  
  
<ol id="list"></ol>
```

Answer :

```
<script>

var products = [

{Name: "TV", Price : 45000.55},

{Name: "Mobile", Price : 12000.44}

];

$(document).ready(function(){

$.each(products,function(key,value);

$('<li>${value.name}</li>').appendTo("#list");

})

})

</script>

<body>

<ol id="list">

</ol>
```

Example :

```
var data = [

{

Category : "Electronics",

Products : ["TV", "Mobile"]

},

{
```

Category : "Shoes",

Products : ["Nike", "Lee Cooper"]

}

]

Answer :

```
<script>
```

```
var data = [
```

```
{
```

```
Category : "Electronics",
```

```
Products : ["TV", "Mobile"]
```

```
},
```

```
{
```

```
Category : "Shoes",
```

```
Products : ["Nike", "Lee Cooper"]
```

```
}
```

```
]
```

```
$(document).ready(function(){
```

```
$.each(data, function(key, value){
```

```
  $('<li>${value.category}</li>').appendTo("#list");
```

```
  $.each(value.Products,function(key, value){
```

```
    $('<ul><li>${value}</li></ul>').appendTo("#list");
```

```
  })
```

```
})
```

```
})
```

```
</script>
```

```
<body>
```

```
<ol id="list"></ol>
```

Exercise :

```
var products = [
```

```
{ Name: "TV", Price: 34000.55},
```

```
{Name: "Mobile", Price: 45000.44},
```

```
{Name: "Nike", Price: 3200.44}
```

```
];
```

Print the Given data in Table

/////////////////////////////////INCOMPLETE////////////////////////////////

JQuery UI

1. JQuery provides a set of functions that are used to handle UI which includes a set of effects, widgets and themes build on the top of JQuery JavaScript library.
2. The UI comprises of following categories,
 - A. Interactions
 - B. Widgets
 - C. Effects
 - D. Utilities

Install JQuery UI for project :

1. Visit the site

<https://jqueryui.com/>

2. Download “Stable” version
3. Unzip the downloaded file
4. Open JQuery folder “jquery-ui-1.12.1”
5. Copy all files
6. Goto your project location and create new folder in “node_modules” by name “jquery_ui”
7. Paste all copied files
8. Your JQuery UI Path will be

Node_modules

Jquery_ui

Jquery-ui.js

JQuery interactions :

jQuery provides a set of functions used for interactions which includes the following,

- A. Draggable
- B. Droppable
- C. Resizable
- D. Selectable
- E. Sortable

Example :

```
<head>

<style>

img {

cursor :move;

}

</style>

<link rel="stylesheet" href="../node_modules/jquery_ui/jquery-ui.css">

<script src="../node_modules/jquery/dist/jquery.js"></script>

<script src="../node_modules/jquery_ui/jquery-ui.js"></script>

<script>

$(document).ready(function(){

$("#pic").draggable();

$("#pic").resizable();

})
```

</script>

<body>

</body>

JQuery Widgets :

1. A widget is a gadget for website it comprises of a presentation in HTML, Effects in CSS and logic in JQuery. You can input these widgets and customize according to your requirements. The commonly used widgets are,

- i. Accordion
- ii. Button
- iii. Controlgroup
- iv. Dialog
- v. Progerssbar
- vi. Slider
- vii. Tabs
- viii. Autocomplete
- ix. Checkboxradio
- x. Datepicker
- xi. Menu
- xii. Selectmenu
- xiii. Spinner
- xiv. Tooltip

Example :

```
<head>

<!.... link jquery and jquery-ui scripts...>

<!... link jqierly-ui.css...>

<script>

$(document).ready(function(){

$("#txtdate").datepicker();

})

</script>

</head>

<body>

<fieldset>

<legend>Select Travle Date</legend>

<input title="Date between 10 jan to 20 jan only" type="text" id="txtdate">

</fieldset>

</body>
```

JQuery Ajax

- Ajax is Asynchronous JavaScript and XML.
- Asynchronous enables partial Post Back.
- Ajax allows to submit only a specific portion of page without reloading the complete page.
- Ajax makes the HTML page more responsive and interactive.
- JQuery handles Ajax requests by using the JavaScript object XMLHttpRequest.

<https://github.com/karrasankar158>