

## **UI FULL STACK WEB DEVELOPMENT**

- What is Full Stack?

Full Stack Web Developer

A full stack web developer is a person who can develop both client and server software.

- In addition to mastering HTML and CSS, he/she also knows how to:

1. Program a browser (like using JavaScript, jQuery, Angular, or Vue)
  2. Program a server (like using PHP, ASP, Python, or Node)
  3. Program a database (like using SQL, SQLite, or Mongo DB)
- 

Client Software (Front End): HTML5, CSS3, BOOTSTRAP5, JAVA SCRIPTS, ADV JAVA SCRIPTS.

- Server Software (Back End) : NODE JS, EXPRESS JS , MANGO DB
- 

### **Front End:**

- HTML4&5
- CSS2&3
- JAVA SCRIPTS
- ADVANCED JAVA SCRIPTS
- JQUERY
- ECMA 6
- BOOTSTRAP 5

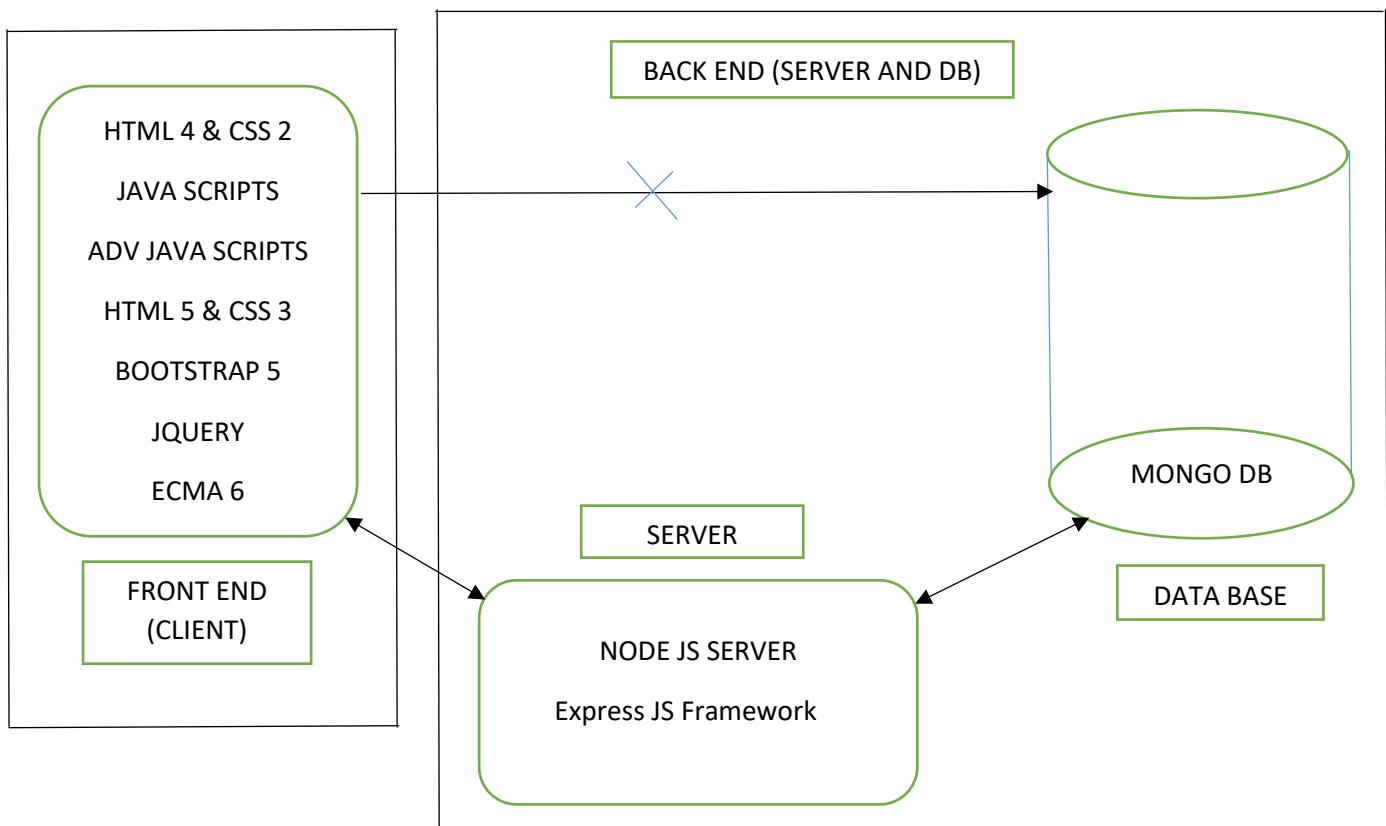
### **BACK END:**

- NODE JS
- EXPRESS JS
- SOCKET IO

### **DATA BASE:**

- MANGO DB.

➤ **Front End & Back End Technologies:**



**Front End:**

It only defines UI which a user or a client will unable to access the data in the database.

➤ **HTML:**

HTML Stands for **Hyper Text Markup Language** which is capable for display the data within the page means the user/client were unable to access the Data Base (DB). It is only to display what it has.

➤ **CSS:**

CSS Stands for **Cascading Style Sheets**. In order to add beautification to the HTML Page.

➤ **Static Web Pages:**

A **static web page** (sometimes called a **flat page** or a **stationary page**) is a web page that is delivered to the user's web browser exactly as stored.

➤ **Dynamic Web Pages:**

A **Dynamic Website** (also referred to as a database-driven site) requires web programming and database design. A **dynamic website** contains

information and content that changes, depending on factors such as the viewer of the site, the time of the day, the time zone, or the native language of the country the viewer).

➤ **Java Scripts:**

**JavaScript** is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. Where **HTML** and **CSS** are languages that give structure and style to web pages, **JavaScript** gives web pages interactive elements that engage a user.

➤ **ECMA 6:**

**ES6** stands for **ECMAScript 6**. **ECMAScript** was created to standardize **JavaScript**, and **ES6** is the **6th** version of **ECMAScript**, it was published in 2015, and is also known as **ECMAScript 2015**.

➤ **BootStrap:**

**Bootstrap** is the most popular CSS Framework for developing **responsive** and mobile-first Websites.

➤ **JQuery:**

**JQuery** is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API (Application Programming Interface) that works across a multitude of browsers.

## **Back End:**

**Back end Development** refers to the server side of **development** where you are primarily focused on how the site works. ... This type of **web development** usually consists of three parts: a server, an application, and a database. Code written by **back end** developers is what communicates the database information to the browser.

➤ **Node.JS:**

**Node. Js** is a JavaScript **run-time** environment built on Chrome's V8 JavaScript engine. It comes with an http module that provides a set of functions and classes for building a **HTTP server**. For this basic **HTTP server**, we will also be using file system, path and URL, all of which are native **Node**.

➤ **Express.JS:**

**Express. Js** is a free and open-source web application framework for **Node. Js**. It is **used** for designing and building web applications quickly and easily.

## **Data Base (DB):**

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). ... The data can then be easily accessed, managed, modified, updated, controlled, and organized.

### **➤ Mongo DB :**

**Mongo DB** is a document database with the scalability and flexibility that you want with the querying and indexing that you need.

## **Software we are going to use:**

- **Browsers:** Edge, Internet Explorer, Google Chrome, Safari, Firefox etc....
- **Editors :**Sublime Text, Visual Studio Code(Microsoft's)
- **Repository:** GitHub
- **Tools:** Node.JS (This tool is mostly used by **programmers** who use JavaScript to write Server-Side scripts.)
- **Data Base:** Mongo DB & Mongo DB Compass.

### **➤ What is Mark-up Language?**

Language that uses tags to represent the content. Tag is in the form of  
<Tag Name >

## ***HTML (Hypertext Mark-up Language)***

➤ **What is HTML:**

HTML Stands for **Hyper Text Markup Language** which is capable for display the data within the page means the user/client were unable to access the Data from the Data Base (DB). It is only to display what it has.

➤ **Structure of HTML :**

```
<!Doctype HTML>
<html>
    <head>
        <title>
        </title>
    </head>
    <body>
    </body>
</html>
```

➤ **Creation of Sample Web Page Using HTML:**

```
<html>
<head>
<title> Sample Web Page </title>
</head>
<body>
<h1>Hello World </h1>
</body>
</html>
```

Save this file as .HTML or with .HTM Extension.

- A predefined language using which we could able to add content and resources to the Web pages. It comes with Predefined set of HTML Tags used to add different types of content accordingly.

**Some Examples of Tags are:**

- **<HTML>**  
It is used to hold complete content of the Web Page.
- **<head>**  
It is used to indicate the head of the Web Page.
- **<title>**  
It holds the title of the Web Page
- **<body>**  
To hold the actual content of the page

- <p>  
Paragraph tag used to hold the multiline text content
- <br>  
Break tag adds single line break
- <div>  
Hold block content
- <span>  
Inline elements to hold content in the same line
- <table>  
To render the content in the row and column way
- <ul><ol>  
To render the content in unordered or ordered way
- <li>  
To display the list of items

➤ **HTML Attributes;**

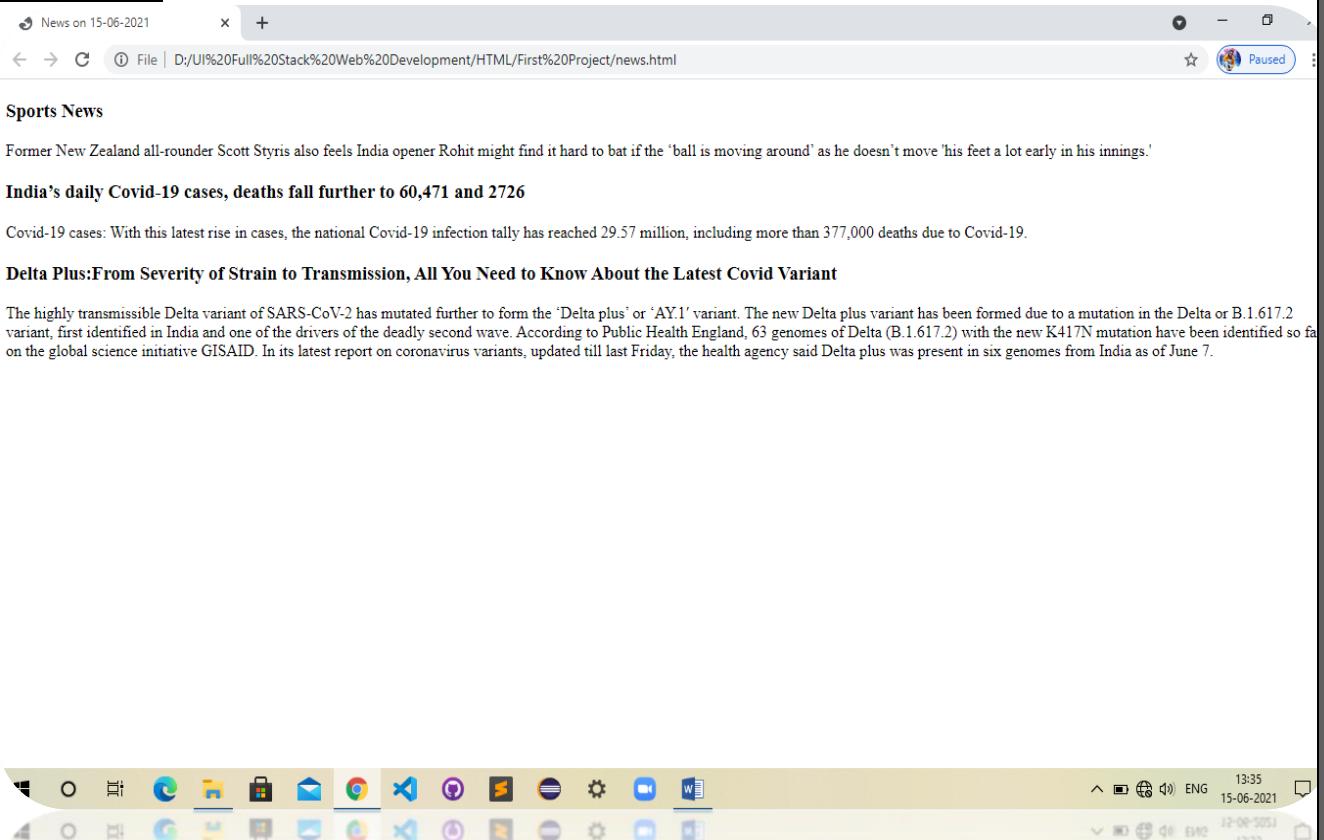
We can add extra information to the HTML elements through HTML Attributes. The attributes can be both predefined and user defined. Following some of the predefined attributes can be added to elements,

- id: using which we can unique reference to elements.
- Name: name value can be added to elements.
- Class: using which we can add single/multiple CSS classes to elements.
- Style: to add single or multiline CSS properties to the elements.
- Alt: using which we can add alternative text content.
- Title: using which we can add title to any HTML elements.

## ➤ Sample Web Page Using HTML Attributes:

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8">
5          <title>News on 15-06-2021</title>
6      </head>
7      <body>
8          <h3> Sports News </h3>
9          <p id="sportsNews" name="cricket" title=" about ICC World Test Champion Ship 2021 india VS newzeland">
Former New Zealand all-rounder Scott Styris also feels India opener Rohit might find it hard to bat if the
'ball is moving around' as he doesn't move 'his feet a lot early in his innings.'</p>
10         <h3>India's daily Covid-19 cases, deaths fall further to 60,471 and 2726 </h3>
11         <p id="Covid-19" name="India'scovidnews" title="Covid-19 cases in india as on 15-06-2021">Covid-19 cases:
With this latest rise in cases, the national Covid-19 infection tally has reached 29.57 million, including
more than 377,000 deaths due to Covid-19.</p>
12         <h3>Delta Plus:From Severity of Strain to Transmission, All You Need to Know About the Latest Covid Variant
</h3>
13         <p id="CoronaVirus" name="covid news" title="Delta Plus variant From Severity of Strain to Transmission,
All You Need to Know About the Latest Covid Variant">The highly transmissible Delta variant of SARS-CoV-2
has mutated further to form the 'Delta plus' or 'AY.1' variant. The new Delta plus variant has been formed
due to a mutation in the Delta or B.1.617.2 variant, first identified in India and one of the drivers of
the deadly second wave. According to Public Health England, 63 genomes of Delta (B.1.617.2) with the new
K417N mutation have been identified so far on the global science initiative GISAID. In its latest report on
coronavirus variants, updated till last Friday, the health agency said Delta plus was present in six
genomes from India as of June 7.</p>
14
15
16  </body>
</html>
```

## OUTPUT:



## **CSS (Cascading Style Sheets)**

### ➤ **Inline CSS:**

An inline CSS is used to apply a unique style to a single HTML Element.

An inline CSS uses the **STYLE** attribute of an HTML Element.

- **STYLE Attribute:**

Setting the style of an HTML element, can be done with the style attribute.

The HTML STYLE attribute has the following Syntax,

```
<TagName style="" ">
```

.....

```
</TagName>
```

Or

```
<style>
```

```
.className{
```

.....

```
}
```

```
</style>
```

**Style** is a predefined tag capable of holding any no.of CSS Classes.

- **Background:**

The CSS Background indicates the background color of a HTML page.

- **Color:**

The CSS Color indicates the Text Color of a HTML Page.

### ➤ **CSS Class:**

A CSS Classes is a group of set of CSS properties and assigning a name to it.

- A concept of grouping all the required CSS Properties as an individual blocks, assigning a user defined name to it is called as a **CSS Class**.
- Every CSS Class name should be started with **.** (**Dot**) operator.
- In a single page we can define any number of CSS classes.
- Class is a predefined attribute using which we could able to inject one or multiple CSS Classes to a single HTML Element.
- Once a CSS Class been defined it can be used any no.of.times to any no.of.elements.

**Syntax:** .className {

.....// Set of CSS Properties

```
}
```

Classes are used by CSS to select and access a specific elements.

➤ **Sample Webpage using Style Attribute:**

**CODE:**

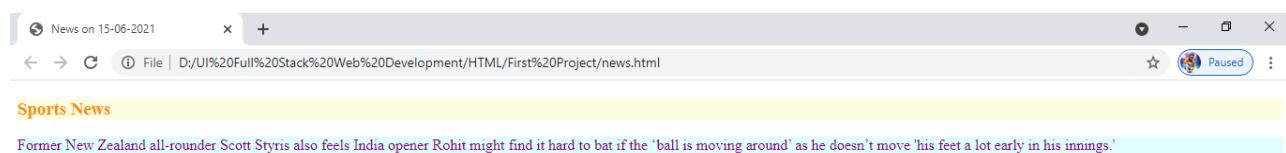
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>News on 15-06-2021</title>
        <style type="text/css">
            .newsHeading {
                background: lightyellow;
                color: darkorange;
            }

            .newsDescription {
                background: lightcyan;
                color: darkmagenta;
            }

        </style>
    </head>
    <body>
        <h3 class="newsHeading"> Sports News </h3>
        <p id="sportsNews" name="cricket" title=" about ICC
World Test Champion Ship 2021 India VS newzeland"
class="newsDescription">Former New Zealand all-
rounder Scott Styris also feels India opener Rohit might find
it hard to bat if the 'ball is moving around' as he doesn't
move 'his feet a lot early in his innings.'</p>

    </body>
</html>
```

**OUTPUT:**



## ➤ CSS Color Values:

In CSS, colors can also be specified using RGB values.

Ex: background-color : rgb ( 255, 99, 71 );

## ➤ Anchor tag:

The <a> tag defines a hyperlink, which is used to link from one page to another.

The most important attribute of the <a> element is the **href** attribute, which indicates the link's destination.

Ex: <a href:<https://www.google.com>>press here</a>

## ➤ Image Tag:

The <img> tag is used to embed an image in an HTML page.

Images are not technically inserted into a web page; images are linked to web pages. The <img> tag creates a holding space for the referenced image.

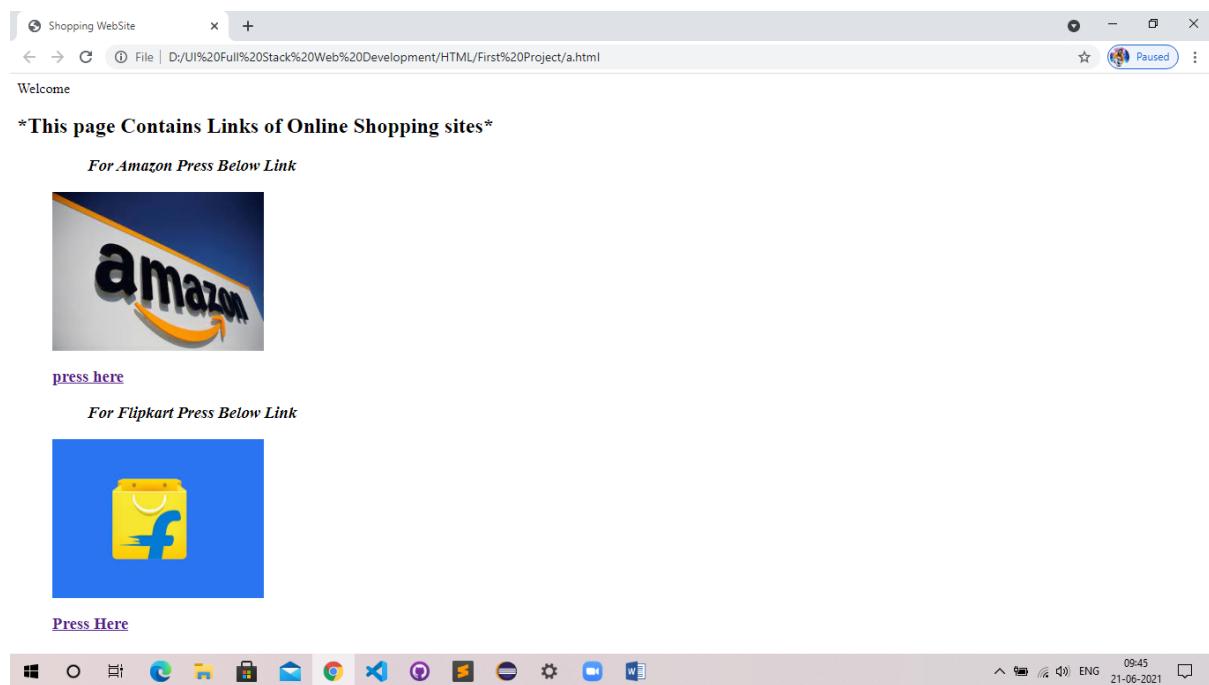
Ex: 

<IMG> tag is a self-closing tag.

## ➤ Sample web page using image tag and anchor tag:

```
D:\UI Full Stack Web Development\HTML\First Project\ a.html
File Edit Selection Find View Goto Tools Project Preferences Help
Student Details.HTML a.html
1 <!DOCTYPE html>
2 <head>
3 <title>Shopping WebSite</title>
4 </head>
5 <body>
6 <p> Welcome </p>
7
8 <h2> *This page Contains Links of Online Shopping sites* </h2>
9 <ul>
10 <h3><ol> <i>For Amazon Press Below Link</i> </ol></h3>
11 
12 <br>
13 <h3><a href="https://www.amazon.in"> press here</a></h3>
14
15
16 <h3><ol> <i>For Flipkart Press Below Link</i> </ol></h3>
17 <br>
18 <h3><a href="https://www.flipkart.com">Press Here</a></h3>
19 </ul>
20 </body>
21 </html>
```

## OUTPUT:



### ➤ Target Attribute:

- The **target** attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.
- Ex:

```
<h3><a href="https://www.amazon.in" target="_blank"> press here</a></h3>
```

### ➤ Different ways to inject CSS to an Element:

1) Adding inline CSS through **<style>** attribute.

Ex :< p style="color: red;" > abcd </p>

2) Creating **Class** and injecting through Class attribute.

Ex : <style type="text/css">

```
.newsHeading {  
    background: lightyellow;  
    color: darkorange;  
}
```

```
</style>
```

```
<body>
```

```
    <h3 class="newsHeading"> aaaaa </h3>
```

```
</body>
```

3) Creating a Class through **ID** of an element.

```
Ex : #idname {  
    background: lightyellow;  
    color: darkorange;  
}  
  
<body>  
    <p id="idname"> Ananth </p>
```

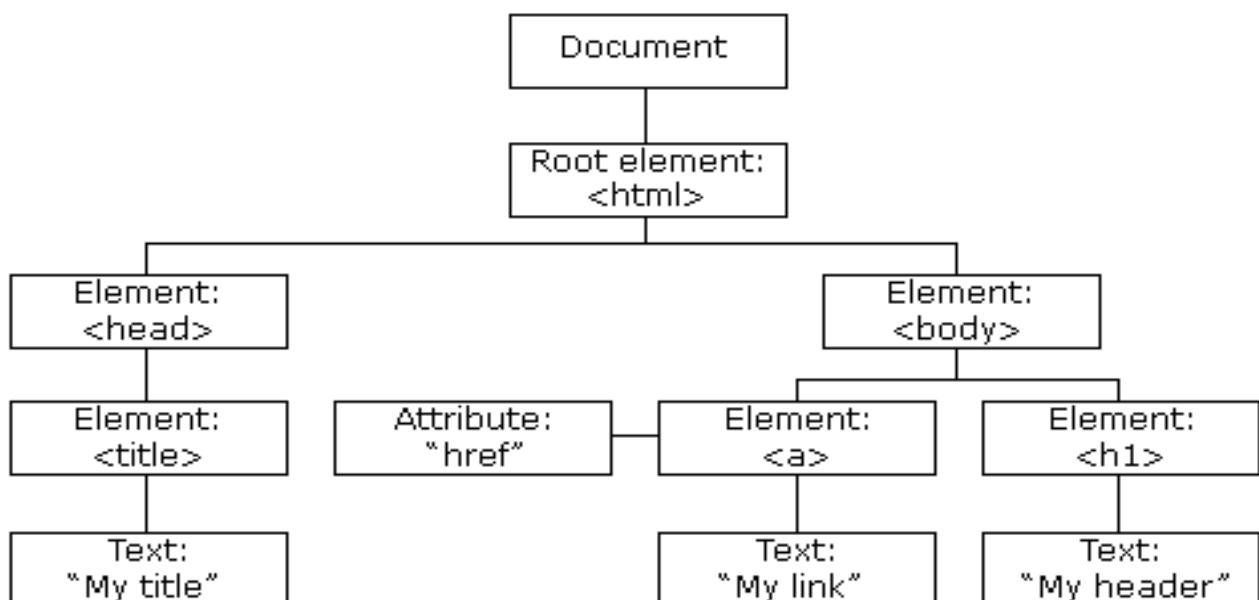
4) Creating a CSS class using the **Tag Name** itself.

```
Ex : h3 {  
    background: lightyellow;  
    color: darkorange;  
}  
  
<body>  
    <h3> Hi Ananth </h3>
```

#### ➤ DOM (Document Object Model) Structure :

For every web page browser dynamically creates a DOM Structure, which indicates the tree structure of the current web pages with all the elements.

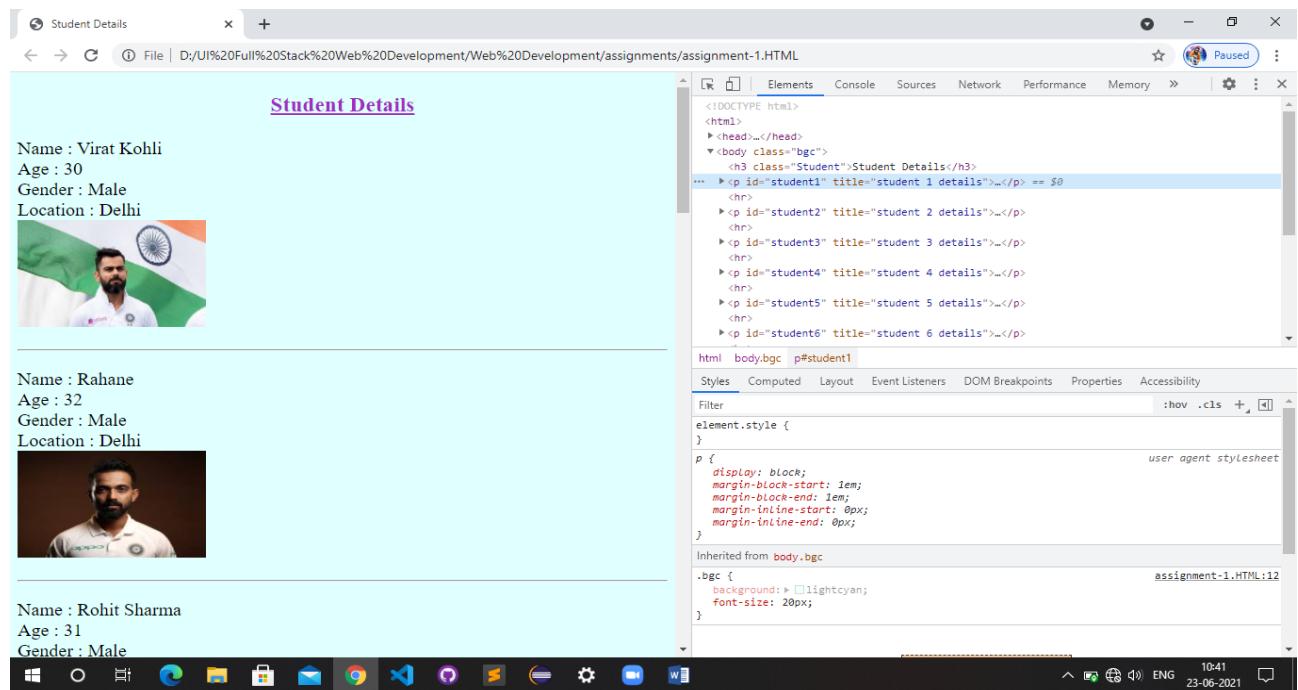
DOM Structure specifies the relationship between the Elements, list of Elements participated within the page, attributes and its corresponding values etc...



The above diagram is example HTML Tree structure.

➤ **Debugger Tool:**

- Every browser by default comes with a default debugger tool using which we could able to debug any web page running under the Browser.
- It shows DOM Structure of the current Webpage.
- It provides an option to create a dynamic HTML Elements while the page is running.
- It provides an option to edit, delete or update any HTML Element and its corresponding attributes while the page is running.
- Any CSS property of an HTML Element can be edited, deleted, add a new property can be done through debugging tool.
- It also provides an option to explore the list of resources been loaded in the current Webpage.
- It also provides an option to explore the list of resources getting loaded through Network or Server (image, JS files, CSS files, XHR request etc...).
- Press **ctrl + shift + I** in a web page in browser to get debugging tool



**Fig.** The above picture shows the debugging tool of the running webpage.

➤ **Priority chain of CSS:**

As we have the four different ways to apply the CSS Properties for the HTML Element, if the same CSS Property been applied to a single element using all the four different ways with different values, by default Browser follows the below priority order,

- Among all the four different ways CSS been applied inline (through **STYLE** attribute) will always takes the higher priority in order.
- CSS been applied through **ID**, which takes the second priority order.
- CSS been applied through **Class**, which takes third priority in order.
- CSS been applied through **TAG**, which takes least priority in order.

Ex: <head>

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

    .newsHeading {
        background: lightyellow;
        color: darkorange;
    }

    .newsDescription {
        background: lightcyan;
        color: darkmagenta;
    }

    #sportsNews {
        background: deeppink;
        color: goldenrod;
    }

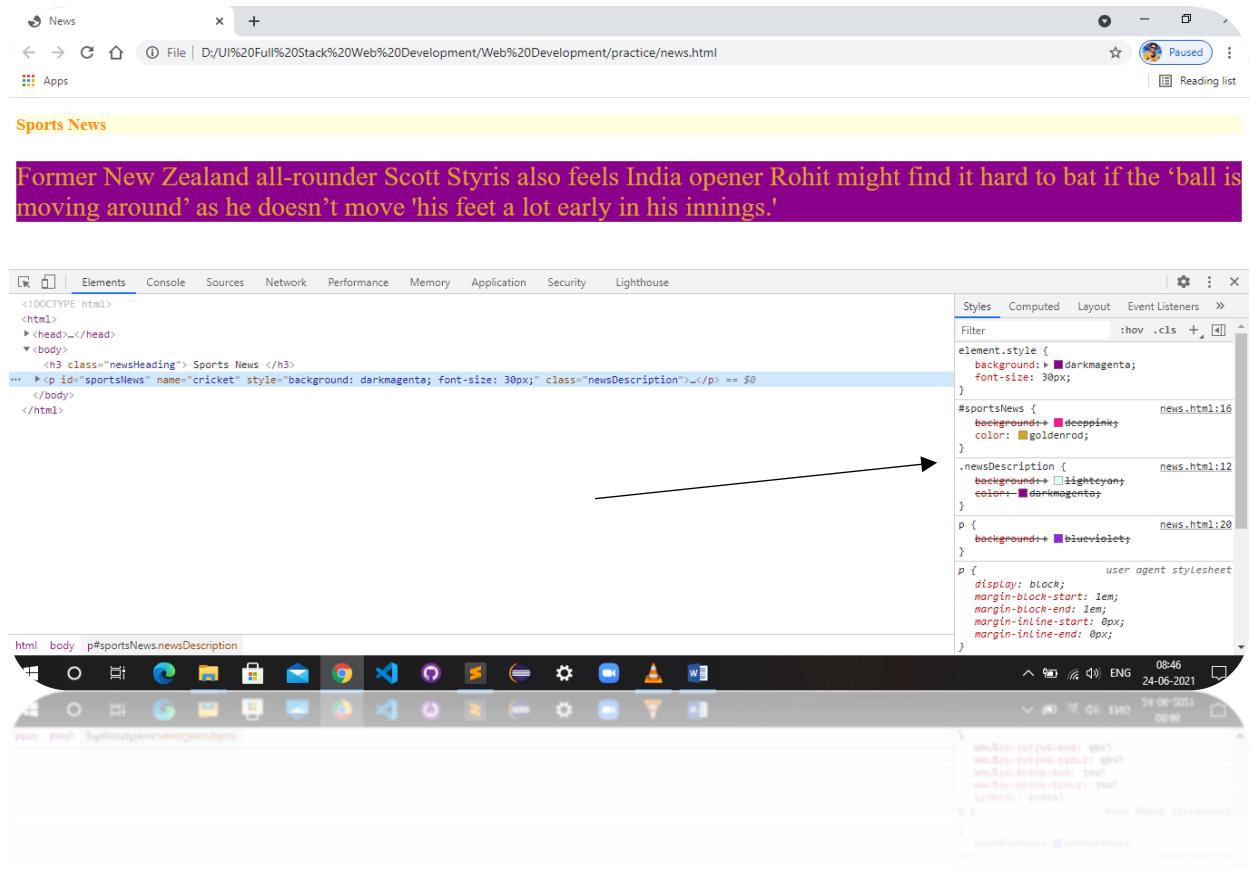
    p {
        background: blueviolet;
    }

</style>

</head>

<body>

    <h3 class="newsHeading"> Sports News </h3>
    <p id="sportsNews" name="cricket" style="background: darkmagenta; font-size: 30px;" class="newsDescription">Former New Zealand all-rounder Scott Styris also feels India opener Rohit might find it hard to bat if the 'ball is moving around' as he doesn't move 'his feet a lot early in his innings.'</p> </body>
```



❖ Note:

Irrespective of the above priority order any CSS property being added with ‘! Important’ will always takes the Higher Priority.

Ex: <head>

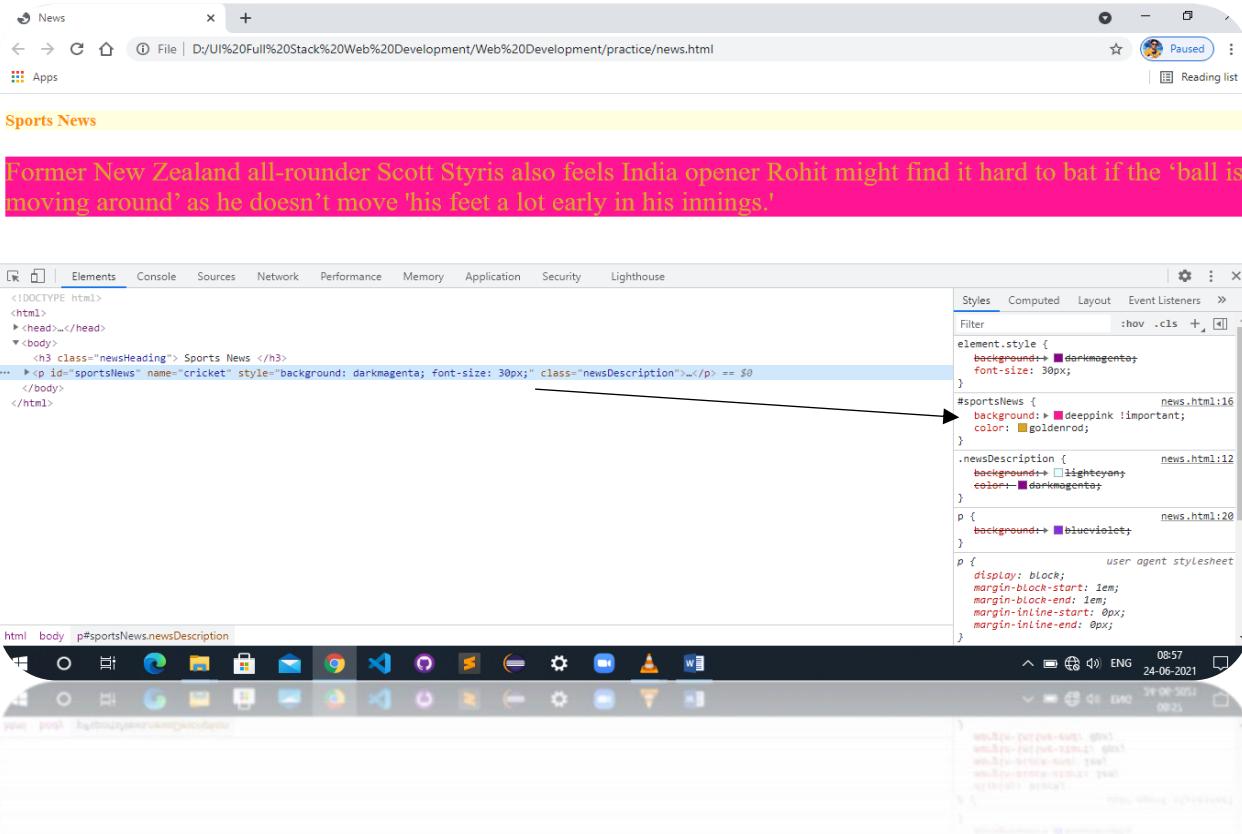
```

<style type="text/css">
    .newsHeading {
        background: lightyellow;
        color: darkorange;
    }
    .newsDescription {
        background: lightcyan;
        color: darkmagenta;
    }
    #sportsNews {
        background: deeppink !important;
        color: goldenrod;
    }
    p {
        background: blueviolet;
    }
</style>
</head>
    
```

```

<body>
    <h3 class="newsHeading"> Sports News </h3>
    <p id="sportsNews" name="cricket" style="background: darkmagenta; font-size: 30px;" class="newsDescription">Former New Zealand all-rounder Scott Styris also feels India opener Rohit might find it hard to bat if the 'ball is moving around' as he doesn't move 'his feet a lot early in his innings.'</p>
</body>

```



**Fig:** Example with !important

### ➤ Inline and Block level Elements:

All the HTML Elements are been categorized into two types

- Block Level Element
- Inline Element.

### ❖ Block level Elements:

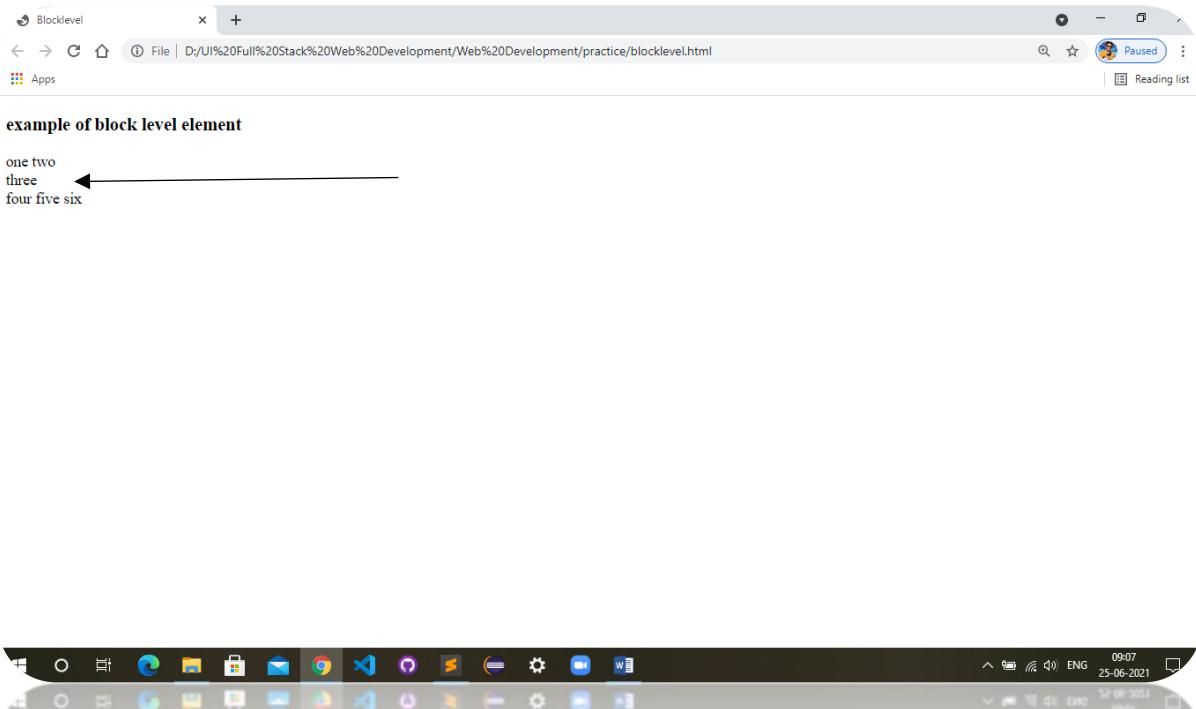
Any HTML Element which comes under the block level category holds the following properties,

- It occupies 100% width of its container by default.
- Even though it occupies 100% width we can still control the dimensions of the block level element through CSS width and height properties.

- While getting rendered within the page by default it comes to new line and gets rendered.
- The element which is following a block level element also automatically comes to a new line and gets rendered.
- Block level elements are mainly used to hold the group of relative items as like an individual block.
- ‘Div’ tag is a best example for a block level element.

**Ex: <body>**

```
<h3> example of block level element </h3>
one
two
<div>three</div>
four
five
six
</body>
```



❖ **Inline Element:**

Any HTML Element comes under inline category holds following properties,

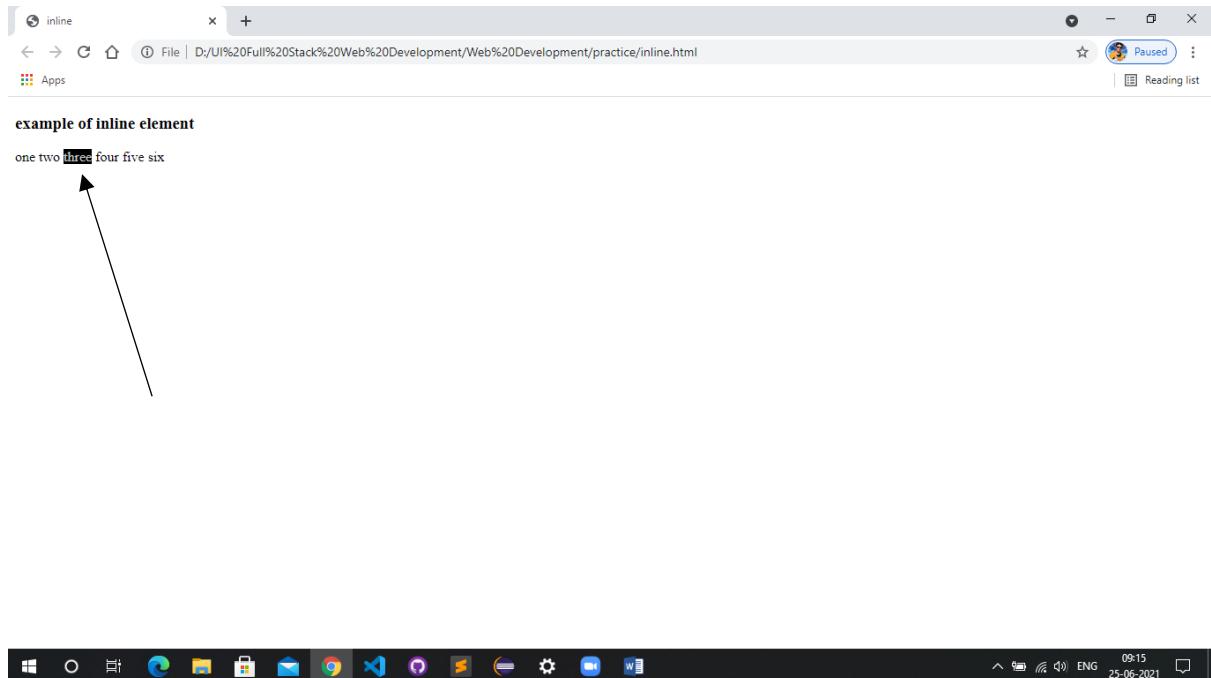
- While rendering on the page all the inline elements tries to render within the same line.
- Inline elements always occupies the width within the container based on the content it is holding.
- We cannot control the dimensions of an inline element.
- ‘Span’ tag is the best example for an inline element.

Ex: **<head>**

```
<meta charset="utf-8">
<title>inline</title>

<style type="text/css">
    .clr {
        color: #ccc;
        font-size: 16px;
        background: black;
    }
</style>

</head>
<body>
    <h3> example of inline element </h3>
    one
    two
    <span class="clr">three</span>
    four
    five
    six
</body>
```



- **Span Tag:**

The <span> tag is an inline container used to mark up a part of a text, or a part of a document.

- **Div Tag:**

The <div> tag defines a division or a section in an HTML document.

## ➤ **CSS Overflow:**

The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

The overflow property has the following values:

- Visible - Default. The overflow is not clipped. The content renders outside the element's box
- hidden - The overflow is clipped, and the rest of the content will be invisible
- scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
- auto - Similar to scroll, but it adds scrollbars only when necessary

➤ **CSS ‘Padding’ and ‘margin’ properties:**

By default while the elements getting rendered on the page they try to occupy very next line of the previous element without maintaining any space.

In order to add space or gap between or within the elements we make use the CSS Properties ‘**Padding**’ and ‘**Margin**’.

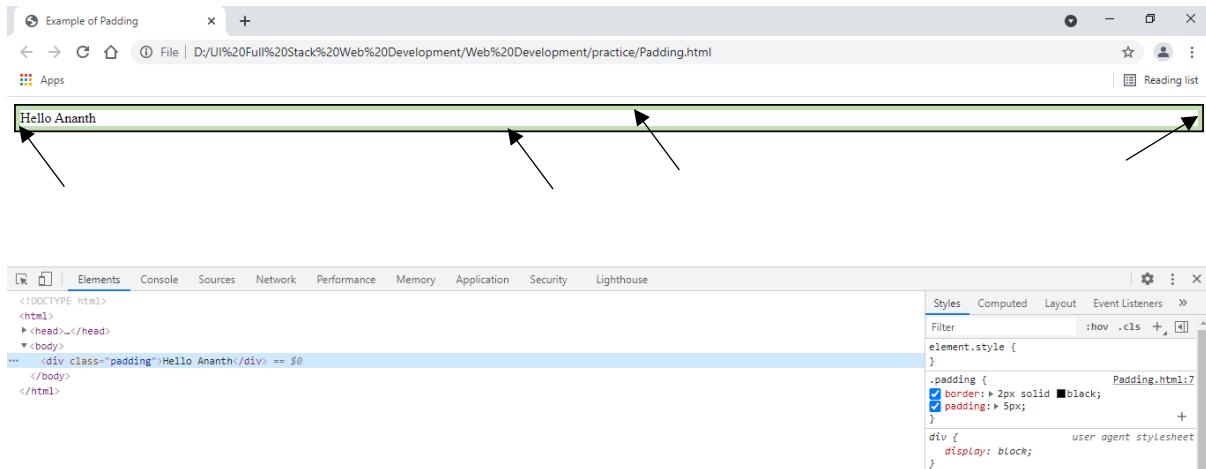
➤ **CSS ‘Padding’ property:**

Through ‘**padding**’ property we could able to add space or gap between the border and the content of the container.

**Example:**

- Padding : 5px;  
**{Adds Padding of 5px to all the Direction (Top, Left, Right & Bottom)}**
- Padding : 5px 10px; **{Adds Padding of 5px to Top and Bottom and 10px to Left and Right}**
- Padding 5px 10px 15px 20px; **{Adds 5px to Top, 10px to Right, 15px to Bottom, 20px to Left}**
- Padding-top : 10px; **{Adds 10px to Top}**
- Padding-left : 10px; **{Adds 10px to Left}**
- Padding-right : 10px; **{Adds 10px to Right}**
- Padding-bottom : 10px; **{Adds 10px to Bottom}**

Ex: <head>  
    <meta charset="utf-8">  
    <title>Example of Padding</title>  
    <style type="text/css">  
        .padding {  
            border: 2px solid black;  
            padding: 5px;  
        }  
    </style>  
</head>  
<body>  
    <div class="padding">Hello Ananth</div>  
</body>



In the above picture we can see the green color border space inside the main container which is indicated by arrows occurred by **padding** property.

## ➤ CSS 'Margin' Property:

Through which we could able to add space or gap above the border of the container.

### Example:

- Margin : 5px; **{Adds Margin of 5px to all the Direction (Top, Left, Right & Bottom)}**
- Margin : 5px 10px; **{Adds Margin of 5px to Top and Bottom and 10px to Left and Right}**
- Margin 5px 10px 15px 20px; **{Adds 5px to Top, 10px to Right, 15px to Bottom, 20px to Left}**
- Margin-top : 10px; **{Adds 10px to Top}**
- Margin-left : 10px; **{Adds 10px to Left}**
- Margin-right : 10px; **{Adds 10px to Right}**
- Margin-bottom : 10px; **{Adds 10px to Bottom}**

Ex: <head>

```

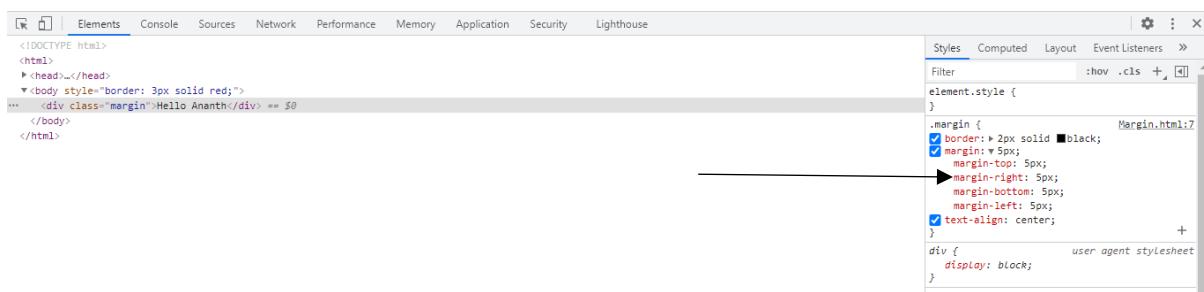
<meta charset="utf-8">
<title>Example of Margin</title>
<style type="text/css">
    .margin {
        border: 2px solid black;
        margin: 5px;
        text-align: center;
    }
</style>

```

```

</head>
<body style="border: 3px solid red;">
    <div class="margin">Hello Ananth</div>
</body>

```



In the above picture we can see the orange color space between two borders which is done by using **Margin** property.

## ➤ CSS Box Model:

When an element gets rendered on the page, browser calculates the actual dimensions of the existing elements on the page, accordingly it renders the new DOM Element. While calculating the dimensions of an element, it follows Box Model feature.

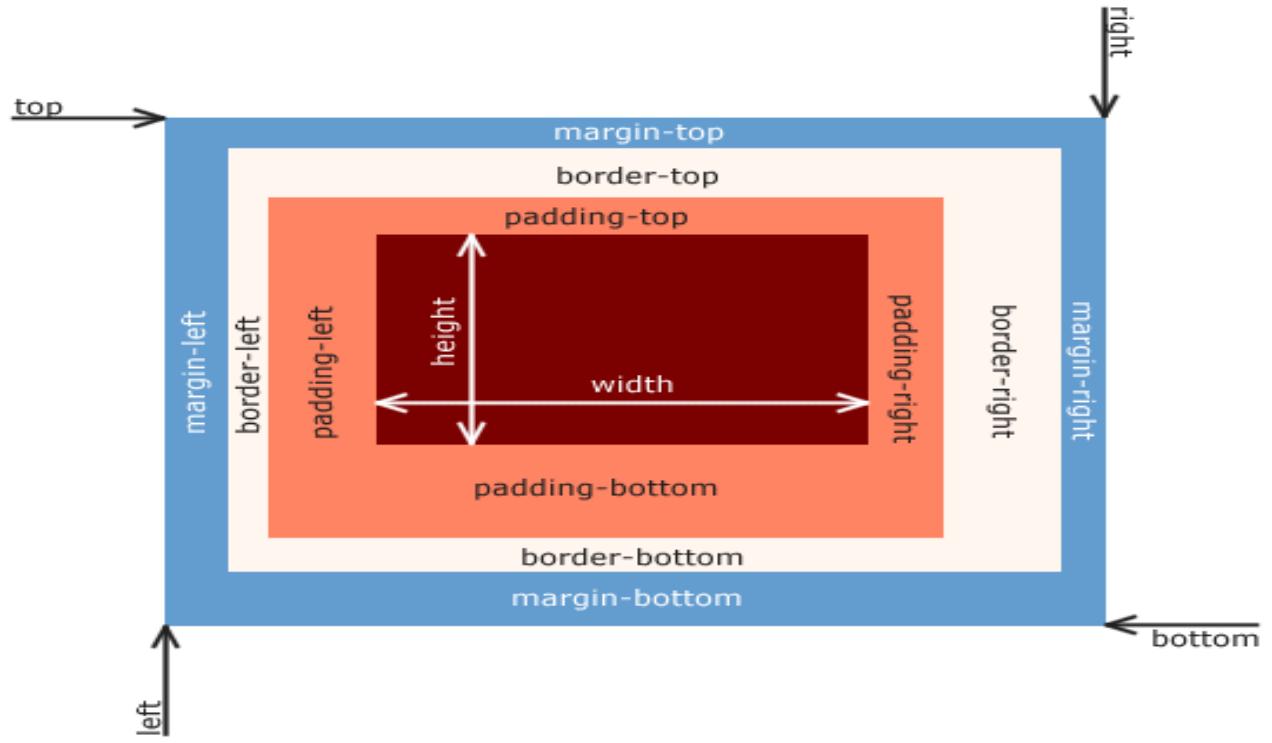
In CSS Box Model property while calculating the actual dimensions of an element, it considers the following CSS Properties,

- The total Border space been Occupied.
- The total Margin space been Occupied.
- Total Padding space been Occupied.
- Actual width & height of the DOM Element.

Explanation of the different parts:

- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements.

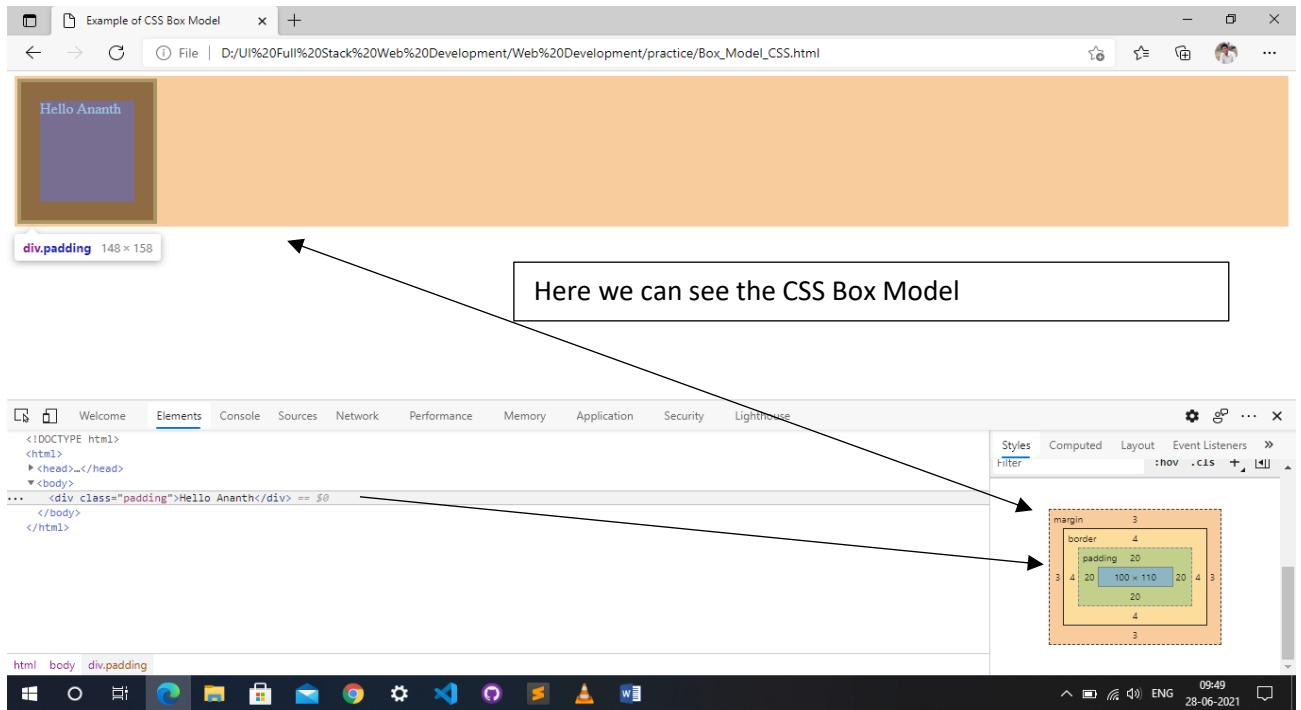


Ex: <head>

```

<meta charset="utf-8">
<title>Example of CSS Box Model</title>
<style type="text/css">
.padding {
    border: 4px solid black;
    padding: 20px;
    margin: 3px;
    height: 110px;
    width: 100px;
    background: darkred;
    color: whitesmoke;
}
</style>
</head>
<body>
    <div class="padding">Hello Ananth</div>
</body>

```



## ➤ HTML UL and OL Tags:

The two predefined HTML tags using which we could able to group relative set of items as an individual block.

**'UL'** and **'OL'** tags renders the elements almost same where the only difference is **'UL'** (**Unordered list**) tag renders the element with default bullet symbols, whereas **'OL'** (**Ordered List**) tag renders the element with default serial number.

- Syntax:      <ul>

```

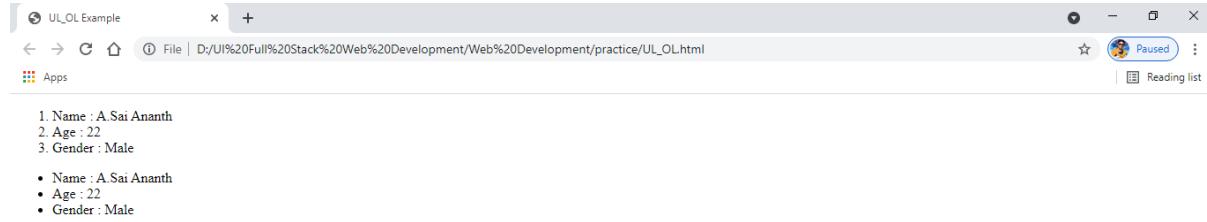
<li> item-1</li>
<li> item-2</li>
</ul>
<ol>
<li> item-1</li>
<li> item-2</li>
</ol>

```

```

Ex: <body>
      <div>
        <ol>
          <li>Name : A.Sai Ananth</li>
          <li>Age : 22</li>
          <li>Gender : Male</li>
        </ol>
      </div>
      <div>
        <ul>
          <li>Name : A.Sai Ananth</li>
          <li>Age : 22</li>
          <li>Gender : Male</li>
        </ul>
      </div>

```



- ‘list-style’ is the CSS Property through which we can control the type of List style getting rendered for both ‘UL’ and ‘OL’ tags.
- For More Visit [https://www.w3schools.com/cssref/pr\\_list-style-type.asp](https://www.w3schools.com/cssref/pr_list-style-type.asp)

```

Ex: <head>
      <meta charset="utf-8">
      <title>UL_Ol Example</title>
      <style type="text/css">
        .list {
          list-style: decimal;
        }
      </style>
    </head>
    <body>
      <div>
        <ol>
          <li>Name : A.Sai Ananth</li>
          <li>Age : 22</li>
          <li>Gender : Male</li>
        </ol>
      </div>

```

```

<div>
    <ul class="list">
        <li>Name : A.Sai Ananth</li>
        <li>Age : 22</li>
        <li>Gender : Male</li>
    </ul>
</div>
</body>

```



## ➤ Adding Background image for Containers:

A background image is specified for almost any HTML Element. To add background image on an HTML Element, use the HTML STYLE attribute and the CSS **background-image** property.

**Syntax:** background-image : url('Enter the url');

**Ex :** <head>

```

<title>Example of CSS Box Model</title>
<style type="text/css">
    .padding {
        border: 4px solid black;
        padding: 20px;
        margin: 3px auto;
        height: 230px;
        width: 410px;
        background-image:
            url('https://th.bing.com/th/id/OIP.fknHCHfIA3LC
            nTkqKeA2sQHaEK?pid=ImgDet&rs=1');
        background-repeat: no-repeat;
        color: darkred;
    }
</style>
</head>
<body>
    <div class="padding">Hello Ananth</div>
</body>

```



- **Background-repeat:** no-repeat; is used for the image not to repeat.

#### ➤ **Downloading and Setting Up Node.js Server:**

- Open the command prompt and check whether Node.js Server is already installed or not by using command **node -v**.
- If not download it from [www.nodejs.org/en/](http://www.nodejs.org/en/)
- After Successful download install the package.
- After Successful installation once again open the command prompt and check the weather the setup is successfully installed or not.
- After type **npm install -g express**. Here npm means Node Package Manager.
- After successful download type the command **npm install -g express-generator**.

A screenshot of a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe". The window shows the following terminal session:

```
C:\> C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1081]
(c) Microsoft Corporation. All rights reserved.

C:\Users\anant>node -v
v14.17.1

C:\Users\anant>npm install -g express
+ express@4.17.1
added 50 packages from 37 contributors in 21.861s

C:\Users\anant>npm install -g express-generator
npm WARN deprecated mkdirp@0.5.1: Legacy versions of mkdirp are no longer supported. Please update to mkdirp 1.x. (Note that the API surface has changed to use Promises in 1.x.)
C:\Users\anant\AppData\Roaming\npm\express -> C:\Users\anant\AppData\Roaming\npm\node_modules\express-generator\bin\express-cli.js
+ express-generator@4.16.1
added 10 packages from 13 contributors in 6.325s

C:\Users\anant>
```

The taskbar at the bottom of the screen shows the time as 01:10:15.

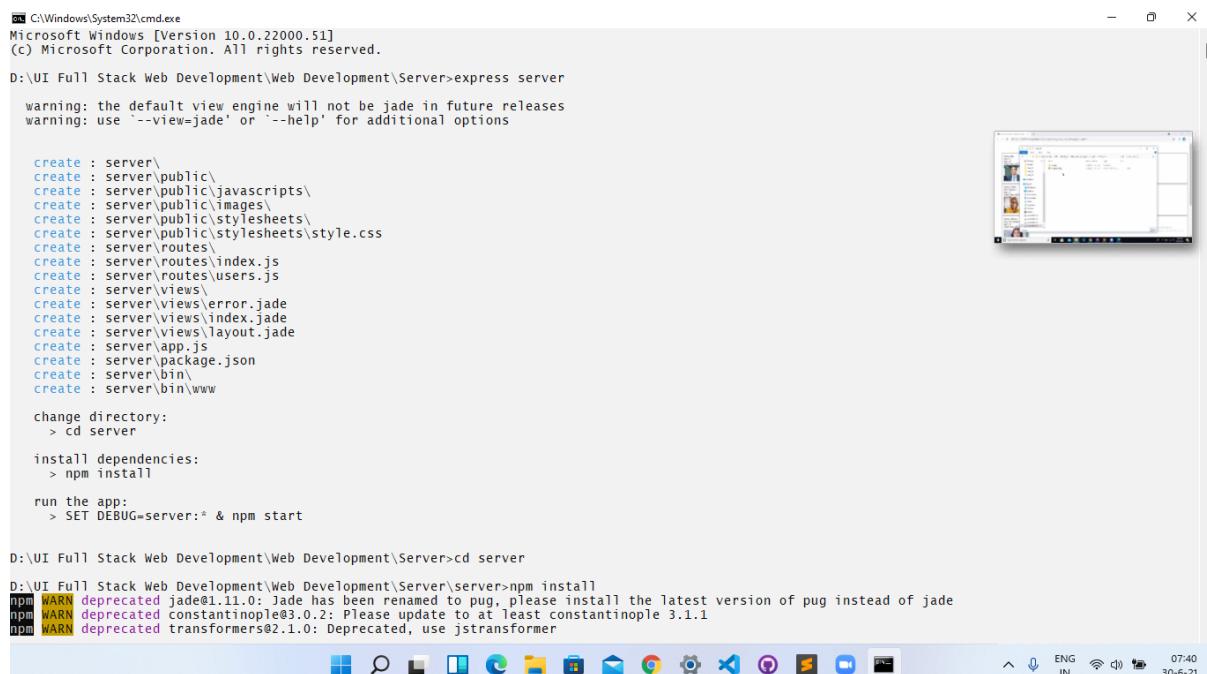
#### ➤ **Setting up Node.js Server:**

- Open Command Prompt.
- Open the path in the command prompt where you want to setup the server  
**Ex: D:\UI Full Stack Web Development\Web Development\Server**
- Type **express folder\_name** in the next line.

- Go to the folder which you set up the server.
- After type **npm install**.
- After open the app(js) file in editor and go to the line 24 and add

```
app.listen(8081, function() {
  console.log("Server is listing at 8081");
});
```

 // where console.log is used to print the data which is given by us.
- After that check whether it is running or not by using command **npm start**.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22000.51]
(c) Microsoft Corporation. All rights reserved.

D:\UI Full Stack Web Development\Web Development\Server>express server
warning: the default view engine will not be jade in future releases
warning: use '--view=jade' or '--help' for additional options

create : server\
create : server\public\
create : server\public\javascripts\
create : server\public\images\
create : server\public\stylesheets\
create : server\public\stylesheets\style.css
create : server\routes\
create : server\routes\index.js
create : server\routes\users.js
create : server\views\
create : server\views\error.jade
create : server\views\index.jade
create : server\views\layout.jade
create : server\app.js
create : server\package.json
create : server\bin\
create : server\bin\www

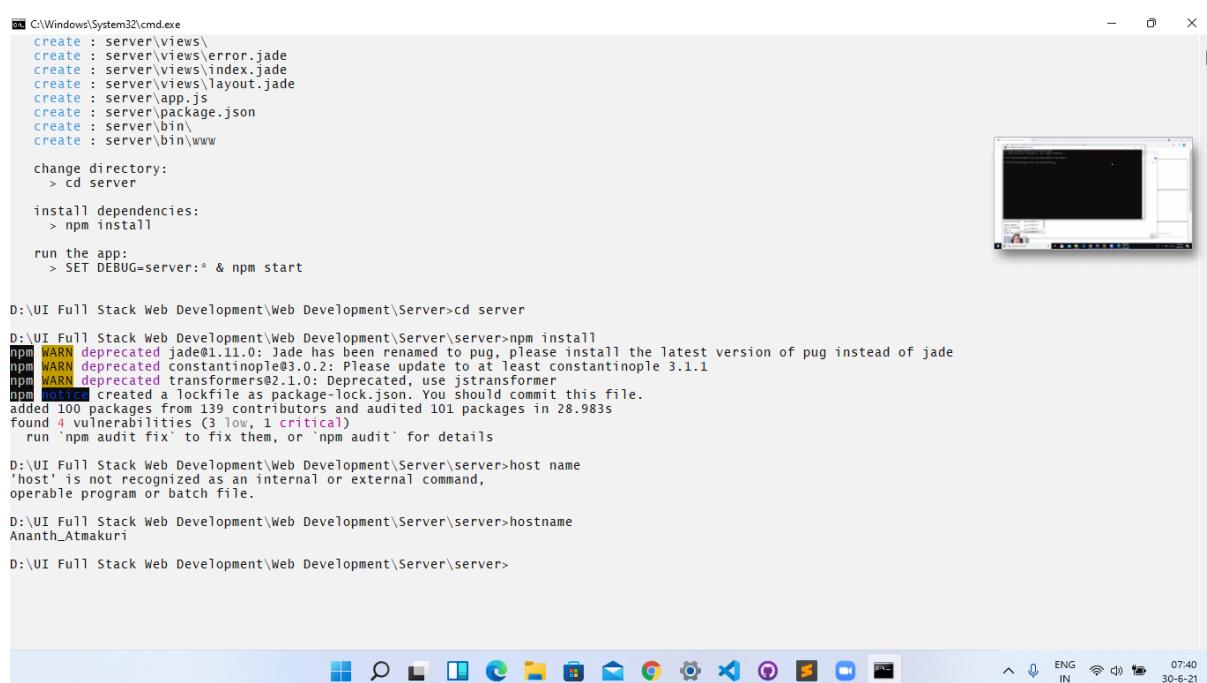
change directory:
> cd server

install dependencies:
> npm install

run the app:
> SET DEBUG=server:* & npm start

D:\UI Full Stack Web Development\Web Development\Server>cd server
D:\UI Full Stack Web Development\Web Development\Server>npm install
npm WARN deprecated jade@1.11.0: Jade has been renamed to pug, please install the latest version of pug instead of jade
npm WARN deprecated constantinople@3.0.2: Please update to at least constantinople 3.1.1
npm WARN deprecated transformers@2.1.0: Deprecated, use jstransformer

```



```
C:\Windows\System32\cmd.exe
create : server\views\
create : server\views\error.jade
create : server\views\index.jade
create : server\views\layout.jade
create : server\app.js
create : server\package.json
create : server\bin\
create : server\bin\www

change directory:
> cd server

install dependencies:
> npm install

run the app:
> SET DEBUG=server:* & npm start

D:\UI Full Stack Web Development\Web Development\Server>cd server
D:\UI Full Stack Web Development\Web Development\Server>npm install
npm WARN deprecated jade@1.11.0: Jade has been renamed to pug, please install the latest version of pug instead of jade
npm WARN deprecated constantinople@3.0.2: Please update to at least constantinople 3.1.1
npm WARN deprecated transformers@2.1.0: Deprecated, use jstransformer
npm notice created a lockfile as package-lock.json. You should commit this file.
added 100 packages from 139 contributors and audited 101 packages in 28.983s
found 4 vulnerabilities (3 low, 1 critical)
  run 'npm audit fix' to fix them, or 'npm audit' for details

D:\UI Full Stack Web Development\Web Development\Server>host name
'host' is not recognized as an internal or external command,
operable program or batch file.

D:\UI Full Stack Web Development\Web Development\Server>hostname
Ananth_Atmakuri

D:\UI Full Stack Web Development\Web Development\Server>
```

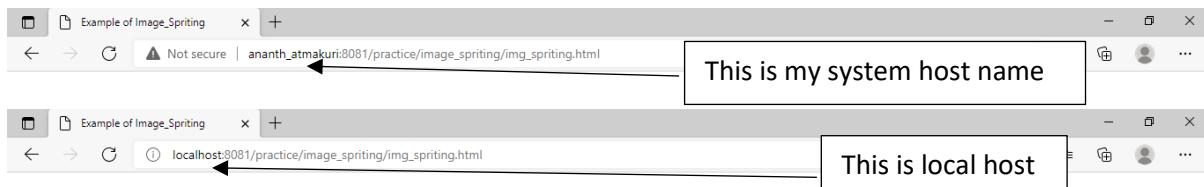
**In the above picture you can see the server setup.**

➤ **Opening the files from the server:**

- Copy the file into the public folder which is present where you created the server.
- Start the server by using command **npm start**.
- Open any file in the public folder where you created.

Ex: [http://localhost:8081/Assignment-2\\_StudenDetails\\_Updated.HTML](http://localhost:8081/Assignment-2_StudenDetails_Updated.HTML)

- We can open the file by using host name and local host



➤ **Image Spriting:**

The concept of merging all the static images to form a single image, through background position property, we show only required image is called as image spriteing.

Through image spriteing we increase the performance of the page by loading all the static images in a single column.

➤ **Note:**

Image spriteing is only recommended for static web pages but not for dynamic web pages.

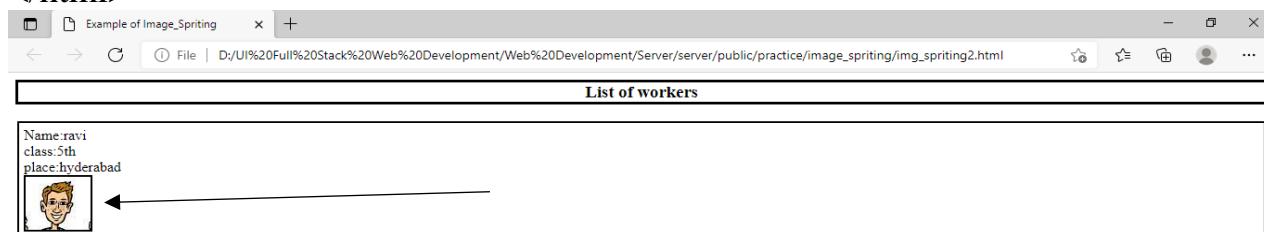
**Ex:** <html>

```
<head>
    <meta charset="utf-8">
    <title>Example of Image_Spriting</title>
    <style type="text/css">
        .mainBorders {
            border: 2px solid;
            padding: 5px;
            margin: 2px;
        }
        .allImages {
            border: 2px solid;
            width: 70px;
            height: 60px;
            background-image: url("images/fullimages.jpg");
        }
        .worker1 {
```

```

        background-position: 222px -9px;
    }
</style>
</head>
<body>
<h3 style="border: 3px solid; text-align: center;">List of workers</h3>
<div class="mainBorders">
    <div>Name:ravi</div>
    <div>class:5th</div>
    <div>place:hyderabad</div>
    <div class="allImages worker1"></div>
</div>
</body>
</html>

```



## ➤ CSS Positions:

Till now in order to move the DOM Elements to a specified position we are using Padding and Margin properties. While using these Padding or Margin properties to move the DOM Element, it is not actually moving the DOM Element but it is increasing the dimensions.

In order to actually move the DOM Elements to a required position without increasing its dimensions we use the following CSS Properties (**Top, Left, Right and Bottom**).

Not every DOM Element is capable of considering the above four CSS Properties but the DOM Element which are position can only consider above properties.

**CSS Position** is the property through which we could able to control the position of any DOM Element.

Following the possible value a position attribute takes,

1. Static
2. Relative
3. Absolute
4. Fixed
5. Sticky

➤ **Element with position Static:**

Every DOM Element by default holds the static position which indicates the DOM Element cannot be moved to any position from its default position. (**It will not consider the Top, Left, Right and Bottom properties**).

**Syntax:**      **position : static;**

**Ex:**

➤ **Element with position Relative:**

Any DOM Element with position relative holds the following properties,

- It is capable of moving to any required position within the page. (**It considers the Top, Right, Left and Bottom properties**).
- While moving to a new position it never loses space been occupied on load of the page.
- While moving to a new position it always moves relevant to its default position.

**Syntax:**      **position : relative;**

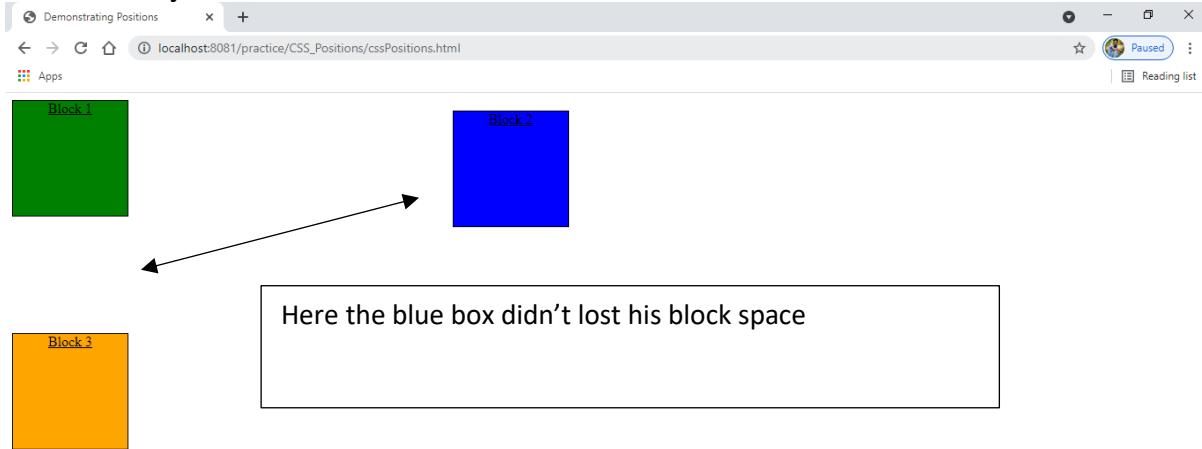
**Ex:** <head>

```
<title>
    Demonstrating Positions
</title>
<style type="text/css">
    .block {
        border: 1px solid;
        text-decoration: underline;
        text-align: center;
        width: 130px;
        height: 130px;
    }
    .block1 {
        background: green;
    }
    .block2 {
        background: blue;
        left: 500px;
        top: -120px;
        position: relative;
    }
    .block3 {
        background: orange;
    }
</style>
</head>
<body>
    <div class="block block1">
        Block 1
    </div>
    <div class="block block2">
        Block 2
    </div>
    <div class="block block3">
        Block 3
    </div>
</body>
</html>
```

```

    </div>
    <div class="block block2">
        Block 2
    </div>
    <div class="block block3">
        Block 3
    </div>
</body>

```



## ➤ Element with position absolute:

Any DOM Element with position absolute holds the following properties,

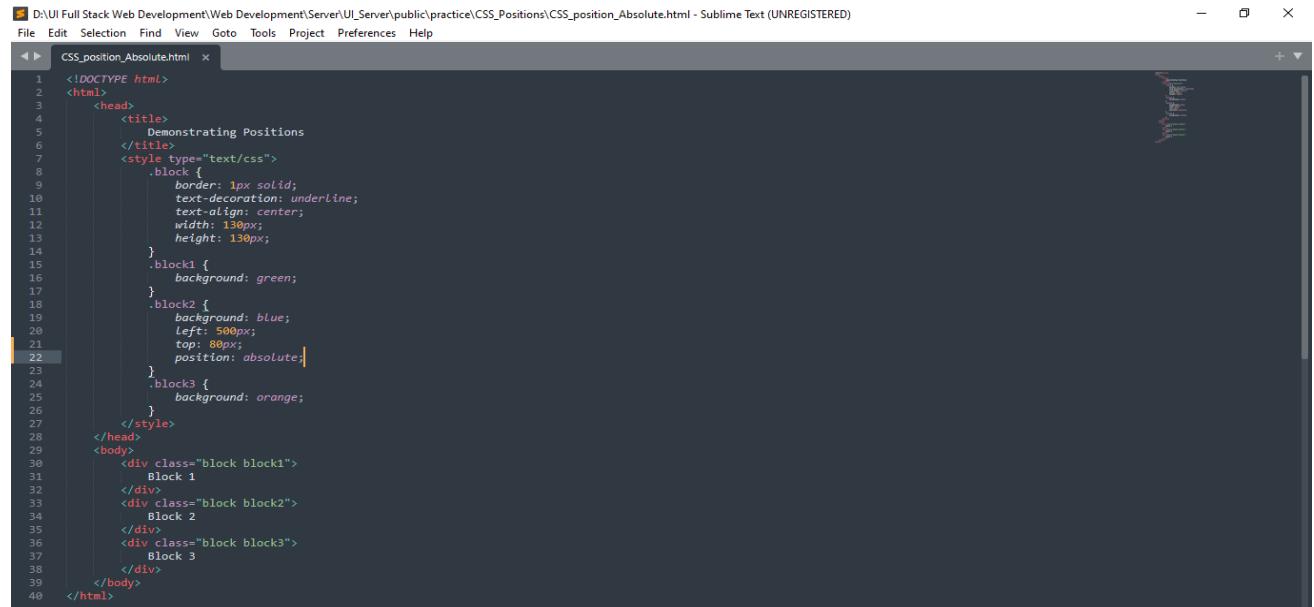
- It is capable of moving to any required position.
- It considers Top, Left, Right and Bottom properties.
- Element with position absolute will automatically loses its default space it occupies in the page.
- While moving to a new position it always moves relevant to its parent position.
- While depending on the parent position it only depends on the parent whose position value is non-static.
- If its intermediate parent doesn't hold position on static, it traverses to its grandparents until it finds an element or a parent with position non-static.

### Note:

Elements with position absolute automatically jumps from default X-Y Axis to Z-Axis.

Syntax:      **position : absolute;**

## Ex:



The screenshot shows the Sublime Text editor with an open file named "CSS\_position\_Absolute.html". The code demonstrates absolute positioning with three blocks: Block1 (green), Block2 (blue), and Block3 (orange). Block2 is positioned at left: 500px; top: 80px; position: absolute;. A browser preview window below shows the result: Block1 is at the top left, Block3 is at the bottom left, and Block2 is positioned at the specified coordinates, overlapping the other blocks.



Demonstrating Positions

localhost:8081/practice/CSS\_Positions/CSS\_position\_Absolute.html

Block 1

Block 2

Block 3

Here the blue block lost its position

### ➤ Element with position fixed:

Any DOM Element with position fixed is almost like an element with position absolute. Where the only difference is once the element gets its fixed position it doesn't move from its original position even when we scroll.

Syntax:            **position : fixed;**

Ex:                **.block2 {**

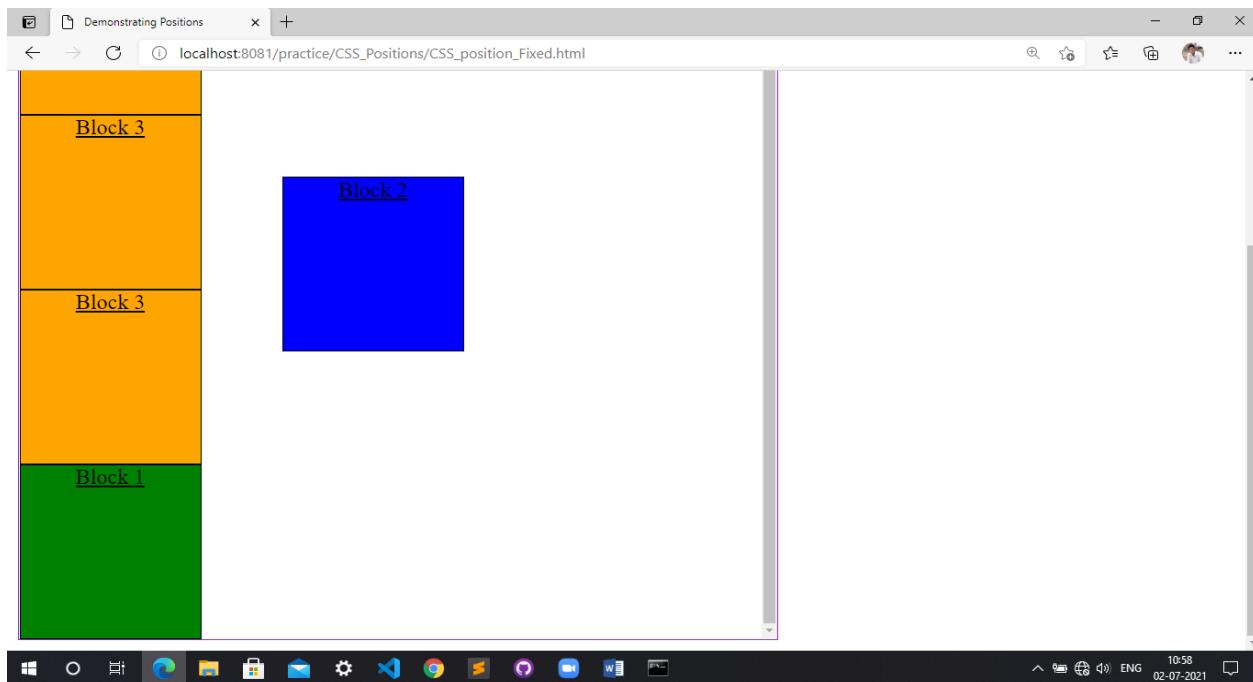
**background: blue;**

**left: 200px;**

**top: 80px;**

**position: fixed;**

**}**



## ➤ Element with CSS Position Sticky:

Any DOM Element with position sticky is almost like an element with position relative, where the only difference is once its position values been given (Top, Left, Right and bottom properties), if we try to scroll the element out of its view port, it automatically turns to fixed position and doesn't get scrolled.

**Syntax:**           **position : sticky;**

**Ex:**               **.block2 {**

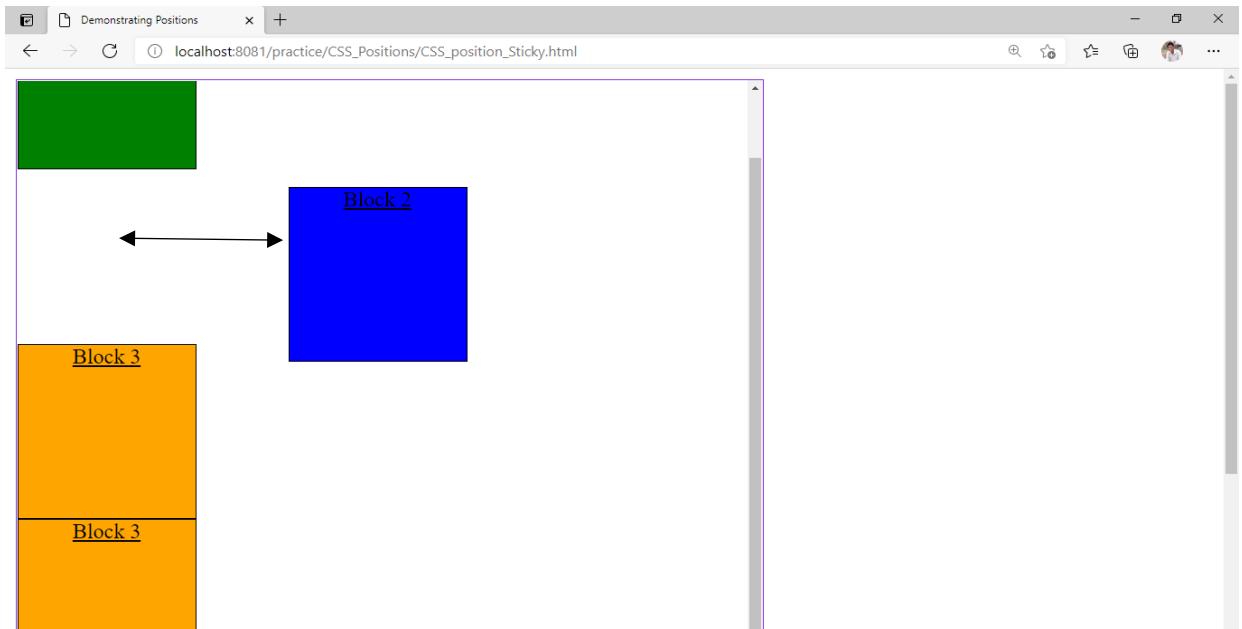
**background: blue;**

**left: 200px;**

**top: 80px;**

**position: sticky;**

**}**



## ➤ CSS Z-Index Property:

The elements which are falling under Z-Axis there is a chance of multiple elements override each other while rendering on the page.

While the Elements are overriding we can control the rendering order through CSS Z-Index Property.

Z-Index is a CSS Property takes a positive number as a value through which we can specify the priority order, the with the higher priority order always renders on the top.

**Syntax: z-index: integer\_value;**

- Note:

Z-Index property can only be applied to elements which fall under Z-Axis (Elements with position Non-Static).

**Ex:** .block1 {  
    background: green;  
    position: relative;  
    **z-index: 3;**  
}



## ➤ CSS Float Property:

By default while the elements getting rendered on the page they try to render from Left-Top Position of the container, the block level element fall under new line, Inline elements are gets rendered within the same line.

Using CSS Float Property we can make the elements to the right side of the container or to the left side of the container. Following are the possible values of a float property takes,

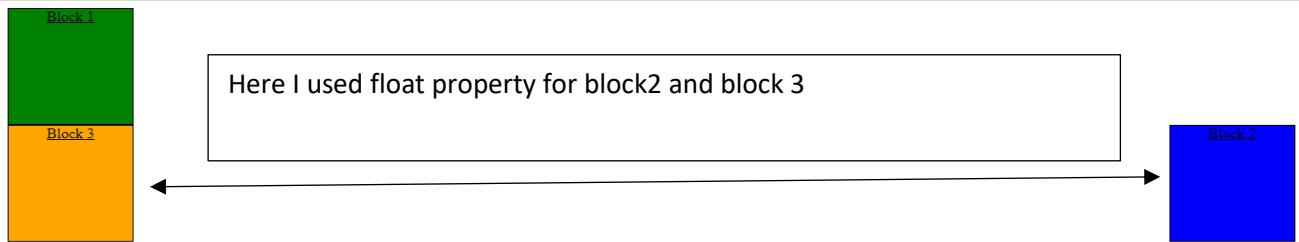
**Syntax: float : left/right;**

- Element with **float left** property will render to complete left side of the container, element with **float right** property will render to complete right side of the container.
- Multiple continuous **DOM Elements** applied with **CSS Float** property will always to render with in the same line irrelevant of block or inline type of the Element.
- If an element changes its default direction through float property, the element which is following it will also try to render in the same line. To make the element not to follow previous element direction we make use of the CSS Property **clear**.

**Syntax: clear:left/right/both;**

**Ex:** .block1 {  
    background: green;  
    position: relative;  
}  
.block2 {  
    background: blue;  
**float: right;**  
}  
.block3 {  
    background: orange;  
**float: left;**

}



➤ **HTML Tables:**

Following are the predefined tags supported in HTML, using which we could able to render any data in the form of row wise and column wise.

- **<table>** Holds the complete table content.
- **<thead>** Holds the table header.
- **<tbody>** Holds the body of the table.
- **<th>** Holds the table header cell.
- **<tr>** Holds the row of the table.
- **<td>** Holds the table data cell.
- **<tfoot>** Holds footer of the table.

➤ **Structure of Basic table in HTML:**

```
<table>
  <thead>
    <tr>
      <th> head 1 </th>
      <th> head 2 </th>
      ...
    </tr>
  </thead>
  <tbody>
    <tr>
      <td> data </td>
      ...
    </tr>
    ...
  </tbody>
  <tfoot>
```

```

<tr>
    <td> ... </td>
</tr>
</tfoot>
</table>

```

### Ex:

```

12 <body>
13     <table border="2px solid">
14         <thead>
15             <tr>
16                 <th>Sno</th>
17                 <th>Date</th>
18                 <th>Time</th>
19                 <th>Present Location</th>
20                 <th>Past Location</th>
21             </tr>
22         </thead>
23         <tbody>
24             <tr>
25                 <td>1</td>
26                 <td rowspan="2">03-07-2021</td>
27                 <td>08:11 am</td>
28                 <td colspan="2">Vizag</td>
29             </tr>
30             <tr>
31                 <td>2</td>
32                 <td>09:11 am</td>
33                 <td>Hyderabad</td>
34                 <td>delhi</td>
35             </tr>
36         </tbody>
37     </table>

```

Sno	Date	Time	Present Location	Past Location
1	03-07-2021	08:11 am	Vizag	
2		09:11 am	Hyderabad	delhi

- We can use border inside table attribute without style attribute.
- **Row Span and Col Span:**

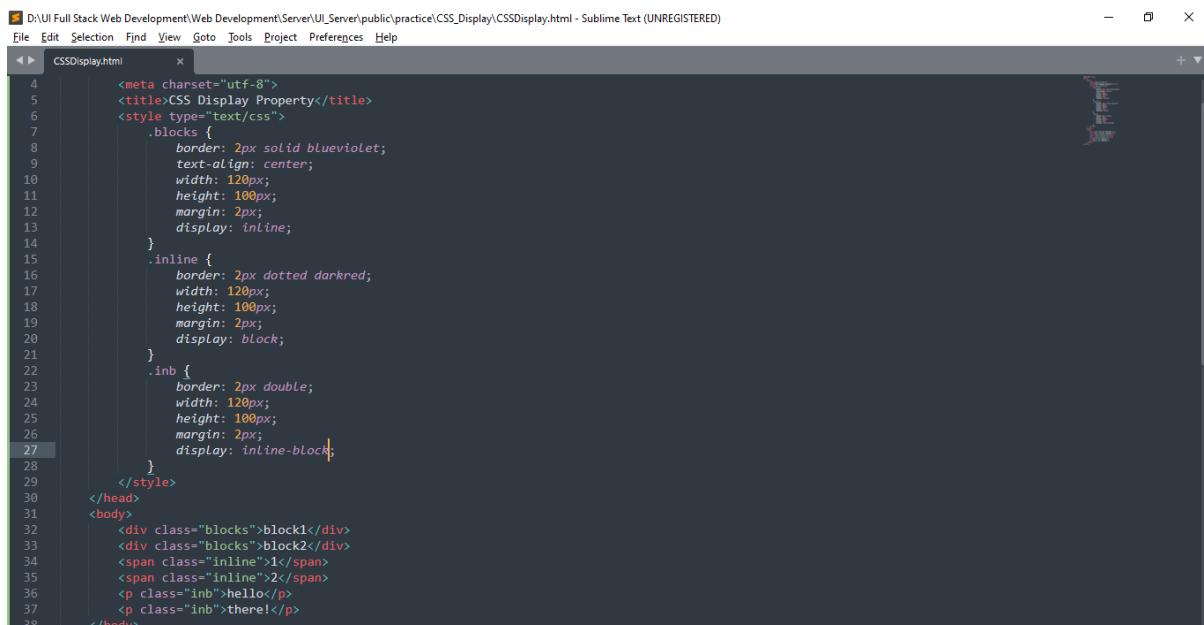
The rowspan and colspan are `<td>` tag attributes. These are used to specify the number of rows or columns a cell should span. The rowspan attribute is for rows as well as the colspan attribute is for columns.

These attributes have numeric values, for example, `colspan=3` will span three columns. As I mentioned in above Example.

## ➤ CSS Display Property:

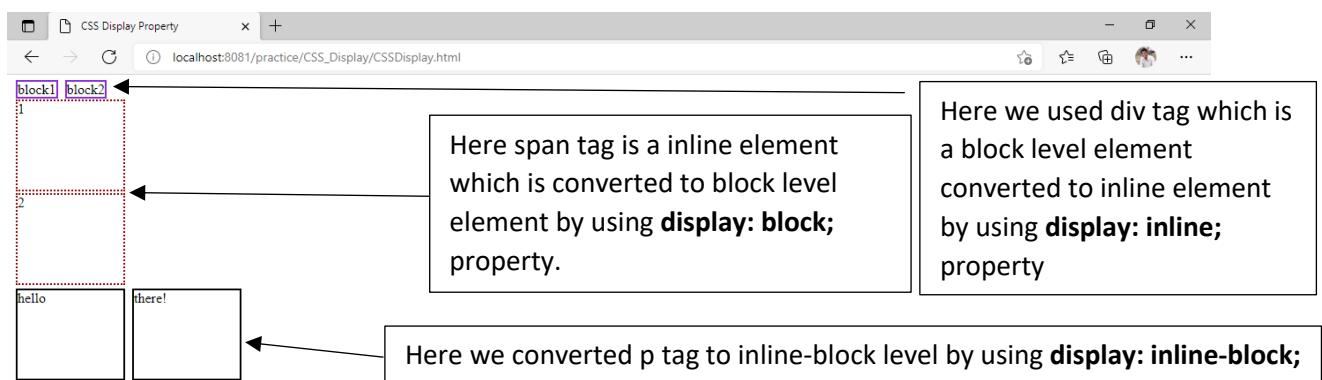
‘Display’ is a CSS Property through which we could able to change the default rendering type of any DOM Element. Following that the possible values it takes,

- **Display: Block;**  
Makes the DOM Element to render like a block level element.
- **Display: inline;**  
Makes any DOM Element to render like an Inline Element.
- **Display: inline-block;**  
Makes any DOM Element to render like a Block Level Element, but occupies in the same line as like in Inline Element.



The screenshot shows a Sublime Text editor window titled "CSSDisplay.html". The code defines three CSS classes: ".blocks", ".inline", and ".inb". The ".blocks" class sets a width of 120px and height of 100px, and uses "display: inline;". The ".inline" class uses "display: block;". The ".inb" class uses "display: inline-block;". The corresponding HTML body contains two divs with class "blocks", two spans with class "inline", and two p tags with class "inb".

```
4     <meta charset="utf-8">
5     <title>CSS Display Property</title>
6     <style type="text/css">
7         .blocks {
8             border: 2px solid blueviolet;
9             text-align: center;
10            width: 120px;
11            height: 100px;
12            margin: 2px;
13            display: inline;
14        }
15        .inline {
16            border: 2px dotted darkred;
17            width: 120px;
18            height: 100px;
19            margin: 2px;
20            display: block;
21        }
22        .inb {
23            border: 2px double;
24            width: 120px;
25            height: 100px;
26            margin: 2px;
27            display: inline-block;
28        }
29    </style>
30 </head>
31 <body>
32     <div class="blocks">block1</div>
33     <div class="blocks">block2</div>
34     <span class="inline">1</span>
35     <span class="inline">2</span>
36     <p class="inb">Hello</p>
37     <p class="inb">there!</p>
38 </body>
```



- **Display: none;**  
Makes element to not to be shown on the page. It still exist in DOM Structure).

- Display: Flex;  
A CSS3 Property to render flexible items on page.

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>CSS Display Property</title>
6     <style type="text/css">
7       .blocks {
8         border: 2px solid blueviolet;
9         text-align: center;
10        width: 120px;
11        height: 100px;
12        margin: 2px;
13      }
14    </style>
15  </head>
16  <body>
17    <div class="blocks">block1</div>
18    <div class="blocks" style="display: none;">block2</div>
19    <div class="blocks">block3</div>
20  </body>
21 </html>

```

## ➤ Difference between display: none and visibility hidden properties:

- Both CSS properties are used to make the DOM element to be not visible on the page where the only difference is display ‘none’ makes the element to be not visible on the page, it doesn’t even occupy its own space.
- Visibility ‘hidden’ also makes the element to be not shown on the page but it still occupies its own space with in the page.

### Note:

Both visibility hidden and display ‘none’ properties makes element to be not visible on the page but the element still exists within the DOM structure.

➤ **CSS Pseudo Classes:**

Following are the predefined pseudo classes been supported using which we could able to apply the CSS on elements not on load of the page but based on current state of the element.

- **:hover :**  
Applies set of CSS when there is a hover
- **:empty :**  
Applied to any element which doesn't have child element
- **:disabled :**  
Applied to element with disabled state
- **:enabled :**  
Applied to element with enabled state
- **:active :**  
Selects only active link
- **:checked :**  
Selects checkbox element with checked state
- **:focus :**  
Selects the element where it is in focus state
- **:first-child :**  
Element which is in first child state
- **:last-child :**  
Element which is in last child state
- **Div:first-of-type :**  
To select every div element which is the first div of its parent
- **a:link :**  
Selects unvisited links
- **a:visited :**  
Selects visited links
- **n<sup>th</sup> – child(2) :**  
Selects element in second position.
- **P:only-child :**  
Selects p tag which is only child of its element.

## ➤ Pseudo Elements:

Following are the predefined Pseudo Elements been supported using which we could able to apply CSS for not to the Complete Element but partially to the content of the element.

- **::after :**

Add any element after the selected element.

- **::before :**

Add any element before the selected element.

- **::first-letter :**

Only applies to the first letter of the container.

- **::first-line :**

Only applies to the first line of the content.

- **::marker :**

Only applies to set CSS for markers of the list items.

- **::selection :**

Select the part/portion of the element content which is selected.

```
5      <title>pseudo elements</title>
6      <style type="text/css">
7          p::first-letter {
8              font-size: 50px;
9          }
10         p::first-line {
11             color: red;
12         }
13         ::marker {
14             color: green;
15             font-weight: bolder;
16         }
17         p::after {
18             content: 'and how are you?';
19             color: black;
20         }
21         p::before {
22             content: 'Hello! ';
23             color: violet;
24         }
25         p::selection {
26             background: ghostwhite;
27             color: darkviolet;
28             font-weight: bolder;
29         }
30     </style>
31 </head>
32 <body>
33     <p>this is Ananth </p>
34     <ol>
35         <li>list item1</li>
36         <li>list item2</li>
37         <li>list item3</li>
38     </ol>
39 </body>
```

Hello! this is Ananth and how are you?

1. list item1
2. list item2
3. list item3

## ➤ CSS Opacity Property:

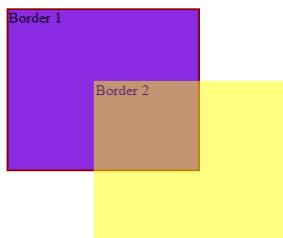
While elements get override each other we could able to control the transparency level of the elements through CSS Opacity property.

It takes a value between 0 to 1

Ex: opacity: 0.2;



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>CSS_OPACITY</title>
6     <style type="text/css">
7       .Border1 {
8         border: 2px solid darkred;
9         background-color: blueviolet;
10        height: 180px;
11        width: 200px;
12      }
13      .Border2 {
14        border: 2px solid yellow;
15        background-color: yellow;
16        height: 180px;
17        width: 200px;
18        position: absolute;
19        left: 100px;
20        top: 90px;
21        opacity: 0.5;
22      }
23    </style>
24  </head>
25  <body>
26    <div class="Border1">Border 1</div>
27    <div class="Border2">Border 2</div>
28  </body>
29 </html>
```



## ➤ HTML Input Elements:

Following are the Pre-Defined HTML Elements supported using which we could able to read different types of data from the user.

- **<input type="text">**  
To read text type of content.
- **<input type="password">**  
To read sensitive data.
- **<input type="date">**
- **<input type="radio">**  
Creates radio button.
- **<input type="checkbox">**  
Creates check box.

- **<input type="button">**  
Creates a button.
- **<input type="submit">**
- **textarea**  
Creates a multiline text container.
- **<select>**

```

<option>option1</option>
<option>option2</option>

```

**</select>**  
Creates a dropdown with multiple options.

**Ex:** `<form action="http://www.serverlocation.in.com" method="Post">`

```

<ul>
<li>
<label for="username"><b>Enter the User Name :</b></label>
<input type="text" name="username" placeholder="maximum
length is 15" maxlength="15">
</li>
<li>
<label for="psw"><b>Enter the password :</b></label>
<input type="password" name="psw" placeholder="maximum
length is 10" maxlength="10" title="password should contain
1upper case and lower case and special symbol and number">
</li>
<li>
<label for="date"><b>Enter the Date Of Birth</b></label>
<input type="date" name="date">
</li>
<li>
<label for="gender">Select Gender</label>
<input type="radio" name="gender">Male
<input type="radio" name="gender">Female
<input type="radio" name="gender">Others
</li>
<li>
<label>Select Country</label>
<select title="click to get country name">
<option></option>
<option>Australia</option>
<option>China</option>
<option>India</option>

```

```

<option>Sri Lanka</option>
<option>England</option>
<option>Pakistan</option>
<option>UAE</option>
<option>Canada</option>
</select>
</li>
<li>
<label for="text">Write about your Self?</label><br>
<textarea name="text" rows="4" cols="50"></textarea>
</li>
<div style="text-align: center;"><input type="submit" name="submit" value="Submit Registration"></div>
</ul>
</form>

```



### ➤ HTML Form Tag:

A predefined tag using which we could able to send user input data to the server.

It takes the following mandatories i.e. method and action.

**METHOD** attribute used to which we could specify the type of the communication while sending or receiving the data.

**ACTION** attribute used for through which we specify the path of the server to which communication should happen.

Ex: `<form action="server url" method="get/post">`

.....

.....

`</form>`

### ➤ Types of Communication:

While communicating with a server, it could be either secured or non-secured type of communication.

- **Non-Secured type of Communication (GET):**

In this type of communication the data will be sent to the server by appending to the URL as query parameters.

Ex: <http://www.abc.com/data/user/info?uname=test&age=20....>

- **Secured Communication(POST):**

In this type of communication the data will be sent to the server by adding it to the request header which is not exposed to the end user.

Any time we send sensitive data to the server we use **POST** type of communication.

## **Java Scripts**

### ➤ **What is Java Script?**

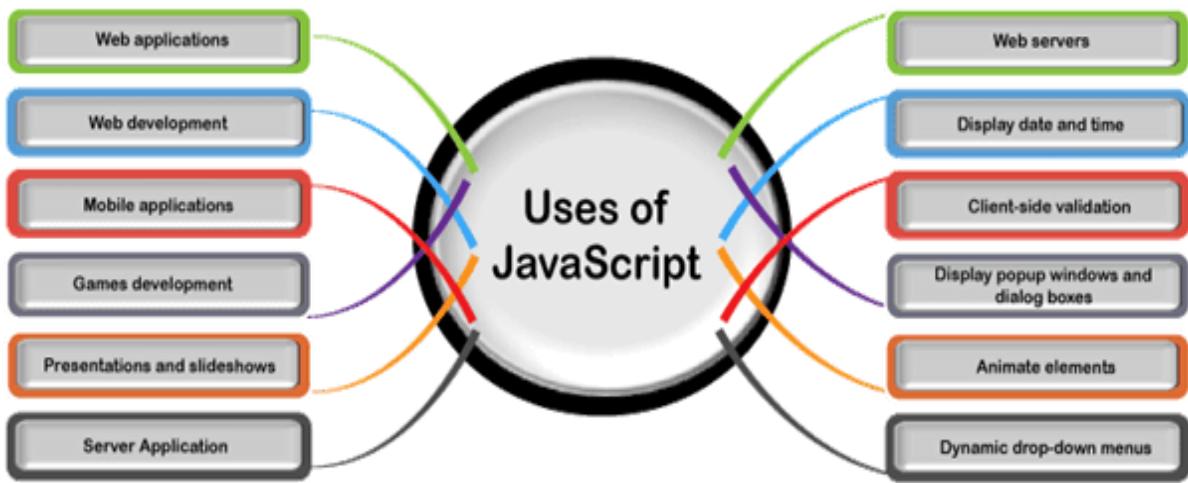
A predefined scripting language where we can implement logical ability, arithmetic operations, DOM Operations in a Web Page.

- It is a client side scripting language capable of executing within the client side itself.
- It is the only programming language that can be understood by the browser.
- By default every browser comes with JavaScript Engine through which JavaScript instructions get compiled and executed within the client side.
- It comes with predefined objects and methods through which DOM Operations can be performed on the page.
- It is a super heroic Programming Language capable of creating, updating or deleting any HTML Element or its corresponding CSS Properties.
- It comes with predefined objects and methods through which we could implement AJAX calls within the Page.
- JavaScript follows top to bottom execution process, it doesn't hold the concept of having predefined static points (like init or main method).

### ➤ **What are the uses of JavaScript?**

- JavaScript is a light-weight object-oriented programming language that is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language. JavaScript enables dynamic interactivity on websites when it is applied to an HTML document.
- JavaScript helps the users to build modern web applications to interact directly without reloading the page every time. JavaScript is commonly used to dynamically modify HTML and CSS to update a user interface by the DOM API. It is mainly used in web applications.

Let's discuss the uses of JavaScript. Some of the uses of JavaScript are represented in the following image.



#### ➤ **Web Applications:**

As day-by-day there is a continuous improvement in the browsers, so JavaScript gained popularity for making robust web applications. We can understand it by taking the example of **Google Maps**. In Maps user just requires to click and drag the mouse; the details are visible just by a click. There is a use of JavaScript behind these concepts.

#### ➤ **Web Development:**

JavaScript is commonly used for creating web pages. It allows us to add dynamic behaviour to the webpage and add special effects to the webpage. On websites, it is mainly used for validation purposes. JavaScript helps us to execute complex actions and also enables the interaction of websites with visitors. Using JavaScript, it is also possible to load the content in a document without reloading the webpage.

#### ➤ **Mobile Applications:**

Now a day's mobile devices are broadly used for accessing the internet. Using JavaScript, we can also build an application for non-web contexts. The features and uses of JavaScript make it a powerful tool for creating mobile applications. The **React Native** is the widely used JavaScript framework for creating mobile applications. Using **React Native**, we can build mobile applications for different operating systems. We do not require writing different codes for the iOS and Android operating systems. We only need to write it once and run it on different platforms.

➤ **Server Applications:**

A large number of web applications have a server-side to them. JavaScript is used to generate content and handle HTTP requests. JavaScript can also run on servers through **Node.js**. The **Node.js** provides an environment containing the necessary tools required for JavaScript to run on servers.

➤ **Web Servers:**

A web server can be created by using **Node.js**. Node.js is event-driven and not waits for the response of the previous call. The servers created using Node.js are fast and don't use buffering and transfer chunks of data. The HTTP module can be used to create the server by using the **createServer()** method. This method executes when someone tries to access the port 8080. As a response, the HTTP server should display HTML and should be included in the HTTP header.

In this article, we discussed various JavaScript applications. JavaScript has various other uses that help us to improve the performance of webpages. The other uses of JavaScript are listed as follows:

- Client-side validation.
- Displaying date and time.
- To validate the user input before submission of the form.
- Open and close new windows.
- To display dialog boxes and pop-up windows.
- To change the appearance of HTML documents.
- To create the forms that respond to user input without accessing the server.

➤ **How to inject JavaScript in HTML:**

- Using inline JavaScript under the **<script>** tag,

**Syntax:** `<script type = “text/javascript”>`

`//JavaScript code`

`</script>`

- In a single page any number of script tags can be placed.
- Inside the script tag we can use ‘n’ number of lines of JavaScript code.

- Script tag can be placed anywhere within the page.

➤ **Data types in JavaScript:**

Data types specifies the type of data that a variable can hold. In JavaScript while declaring a variable it is not required to specify the data type while declaring the variable.

‘**var**’ is a predefined keyword through which we could declare variable in JavaScript.

Following are the differ types of data types supported by JavaScript,

- |           |            |              |              |
|-----------|------------|--------------|--------------|
| 1. Number | 2. String  | 3. Object    | 4. Undefined |
| 5. Null   | 6. Boolean | 7. Function. |              |

**Ex:**

The screenshot shows a code editor with the following code:

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Java Scripts Injection to HTML</title>
6   </head>
7   <body>
8     <h3>java script inject</h3>
9     <script type="text/javascript">
10       var firstnum = 10;
11       var secondnum = 493;
12       var sum = firstnum + secondnum;
13       alert("The Result is : " + sum);
14     </script>
15   </body>
16 </html>

```

Below the code editor is a browser window titled "Java Scripts Injection to HTML". The address bar shows "localhost:8081/practice/JavaScripts/JSDemo/JS.html". The browser displays a message box with the text "localhost:8081 says" and "The Result is : 503", with an "OK" button at the bottom right.

➤ **Write a program for employee details like ename, eage, basicsal, totalsal, hra, pf and department calculate total salary?**

```

<script type="text/javascript">
  var ename ="ananth";
  var eage = 23;
  var gender = "Male";
  var dept = 'IT';
  var basicsal = 15000;

  var totalsal, hra, pf;
  pf = 13/100 * basicsal;
  hra = 25/100 * basicsal;
  totalsal = hra + basicsal + pf;

```

```

        console.log("pf is" + pf);
        console.log("hra is " + hra);
        console.log("totalsal is " + totalsal);
    </script>

```

pf is1950  
hra is 3750  
totalsal is 20700

Emodet.html:20  
Emodet.html:21  
Emodet.html:22

## ➤ Type of Method:

A predefined method being supported in JavaScript using which we could able to get type of data a variable is holding.

It takes a variable as a parameter and returns, the type of data the variable is holding.

**Syntax: typeof(Variable\_Name);**

**Ex:** var num = 10;

```

var name = 'ananth';

console.log(typeof(age));
console.log(typeof(name));

```

number  
string

Typeof.html:11  
Typeof.html:12

- If any variable is not initialized it will be defined as undefined.

## ➤ Type Casting:

Converting a data type into another is known as type casting. Sometimes there is a need to convert the data type of one value to another. Under some circumstances JavaScript will perform automatic type conversion.

```

Ex: <script type="text/javascript">
    var num = "10";
    var name = 'ananth';
    console.log(typeof(num));
    num = parseInt(num);
    console.log(typeof(num));
</script>

```

```

string
number
>

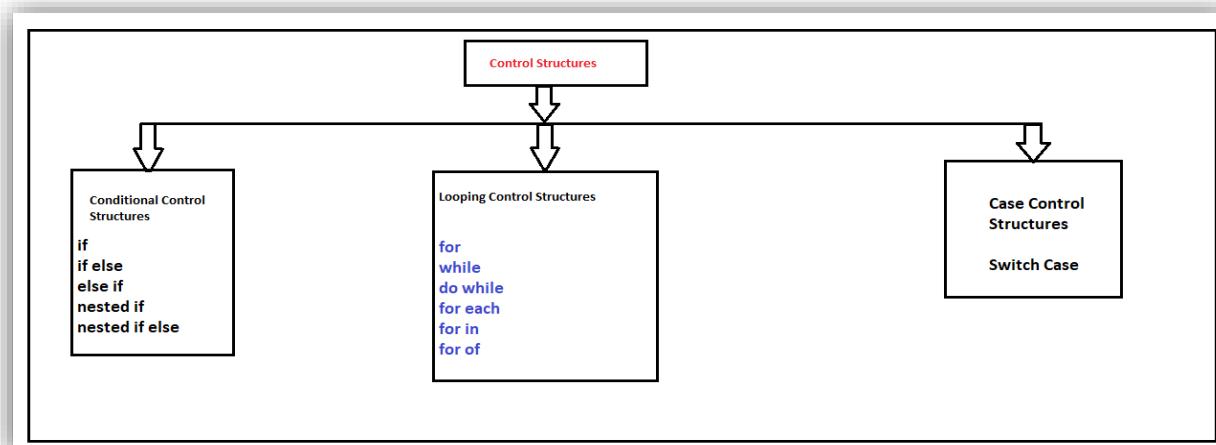
```

TypeCasting.html:11  
TypeCasting.html:13

## ➤ Control Structures:

Following are the set of Control Structures being supported through which we could able to control the sequence of execution flow in a application.

1. Conditional Control Structures.
2. Looping Control Structures.
3. Case Control Structures.



## ➤ IF Condition:

Use the if statement to specify a block of JavaScript code to be executed if a condition is true.

**Syntax:** `if (condition) {  
 // block of code to be executed if the condition is true  
}`

## ➤ Else :

Use the else statement to specify a block of code to be executed if the condition is false.

**Syntax:** `if (condition) {  
 // block of code to be executed if the condition is true  
}  
else {  
 // block of code to be executed if the condition is false  
}`

➤ **Else if Statement :**

Use the else if statement to specify a new condition if the first condition is false.

**Syntax:** **if (condition1) {**

**// block of code to be executed if the condition1 is true.**

**} else if (condition2) {**

**// block of code to be executed if the condition1 is false and condition 2 is true.**

**} else {**

**// block of code to be executed if the condition1 is false and condition 2 is False.**

**}**

➤ **Nested if:**

A nested if is an if statement that is the target of another if or else. Nested if statements means an if statement inside an if statement. Yes, JavaScript allows us to nest if statements within if statements. i.e., we can place an if statement inside another if statement.

**Syntax:** **if (condition1) {**

**// block of code to be executed if the condition1 is true.**

**} if (condition2) {**

**// block of code to be executed if the condition 2 is true.**

**}**

**Ex:**

```
9   <script type="text/javascript">
10    var sname;
11    var sage;
12    var tel=90, hin=80, eng=51, maths=55, science=40, social=40;
13    var avg;
14    var total;
15    total = tel + hin + eng + maths + science + social;
16    avg = (total / 600)*100;
17    console.log("total" + total);
18    console.log("average" + avg);
19    if (avg>70) {
20      console.log("passed in distinction");
21    } if (avg>60 && avg<70) {
22      console.log ("passed in first class");
23    } else {
24      console.log ("second class");
25    }
26  </script>
```

```

total356
average59.333333333333336
second class

```

## ➤ Looping Control Structures:

Looping Control Structures been supported in Java Script takes the set of instructions and repeat them for multiple times.

- For Loop:

Loops through a block of code a number of times.

**Syntax:** `for (initialization, <condition>, increment/decrement) {  
 // set of lines to be Executed  
}`

**Ex:**

```

<script type="text/javascript">
    var i;
    var j=10;
    for (i=1; i<=j; i++){
        console.log(i);
    }
</script>

```

```

1
2
3
4
5
6
7
8
9
10

```

## ➤ Write a program to print even numbers from 1 to 10?

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8">
5          <title>Even Numbers</title>
6      </head>
7      <body>
8          <script type="text/javascript">
9              var i;
10             var j=10;
11             for (i=1; i<=j; i++){
12                 if (i % 2 == 0) {
13                     console.log(i);
14                 }
15             }
16         </script>
17     </body>
18 </html>

```

```
2
4
6
8
10
```

Even Numbers.html:13  
Even Numbers.html:13  
Even Numbers.html:13  
Even Numbers.html:13  
Even Numbers.html:13

### ➤ Difference between “=”, “==” and “====”:

- ‘=’ it is an assignment operator using which we could able to assign a value of right side to the variable of left side.

**Ex:** var a=10;

- ‘==’ it is an comparison operator used to compare value of right side and returns a Boolean value (true/false) based on the equality of the values.

**Ex:** var a = 10;

Var b = 11;

a==b; //returns false

**Note:** ‘==’ operator, while comparing values it only checks for value equality but not data type equality.

- ‘====’ it is almost like an equality ‘==’ operator used to compare value equality were the only difference is while comparing values it will not just checks for value equality but it also checks for data type equality.

**Ex:** var a = “10”;

var b = 10;

a====b; // returns false.

### ➤ While Loop Control Structure:

It is almost like a for loop control structure issued to iterate instructions more than once were the only difference is while loop takes only the condition were the increment and decrement can be placed anywhere within the code.

**Syntax: while (condition) {**

- - - - -

}

**Ex: WAP to print the lucky number of a given number?**

```
8     <script type="text/javascript">
9         var number = 12345;
10        var sum = 0, a;
11        while (number > 0) {
12            a = number % 10;
13            sum += a;
14            number = parseInt(number/10);
15
16            if (number == 0 && sum >= 10) {
17                number = sum;
18                sum = 0;
19            }
20        }
21        console.log(sum);
```

- **WAP to find the reverse of a number?**
- **WAP to check weather given number is palindrome or not?**
- **WAP to list out all palindrome numbers from 300 to 21?**
- **WAP to list out reverse numbers between 32 to 950?**
- **WAP to check Weather the given number is Armstrong or not?**

➤ **Reading values through prompt:**

The prompt () method displays a dialog box that prompts the visitor for input.

A prompt box is often used if you want the user to input a value before entering a page.

By default it takes user input as string.

**Note:** When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value. Do not overuse this method, as it prevents the user from accessing other parts of the page until the box is closed.

**Syntax:** var variable\_Name = prompt ("text");

➤ **Do-while:**

It is almost like a while loop control structure used to repeat a set of instructions for more than once were the only difference is while loop will execute the instructions only when the provided condition is true, were as do-while executes the set of instructions at least for once irrelevant of whether provided condition is true or false.

**Syntax:** do {

**//set of instructions is executed for atleast once irrelevant to the provided condition is true or false**

} While (condition);

➤ **Case Control Structure:**

It takes multiple cases within it and invokes corresponding/particular case based on the value been passed.

**Syntax:** switch (<value>) {

Case <value> :

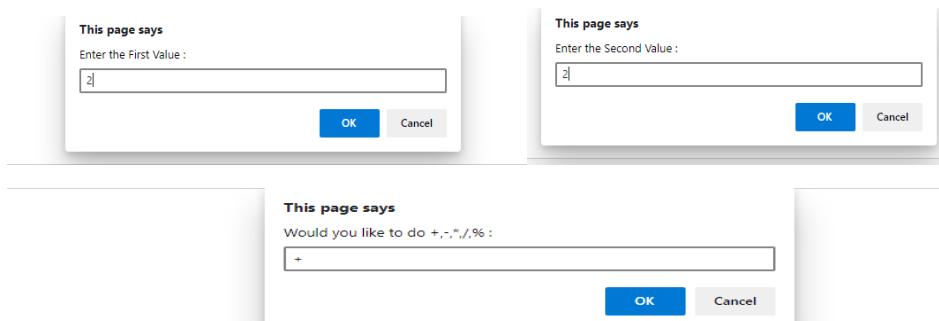
**//stmts**

//stmts

**Break;**

}

```
8     <script type="text/javascript">
9         var firstValue = parseInt(prompt("Enter the First Value : "));
10        var secValue = parseInt(prompt("Enter the Second Value : "));
11        var result;
12        var uinput = prompt ("Would you like to do +,-,*,/,% : ");
13        switch (uinput) {
14            case '+':
15                result = firstValue + secValue;
16                console.log("Addition is : " + result);
17                break;
18            case '-':
19                result = firstValue - secValue;
20                console.log("Subtraction is : " - result);
21                break;
22            case '*':
23                result = firstValue * secValue;
24                console.log("Multiplication is : " + result);
25                break;
26            case '/':
27                result = firstValue / secValue;
28                console.log("Division is : " + result);
29                break;
30            case '%':
31                result = firstValue % secValue;
32                console.log("Modulous is : " + result);
33                break;
34            default:
35                console.log("Wrong option selected..... Please try again.");
36        }
37    </script>
```



## ➤ Variable Hosting:

Most of the programming languages forces to declare a variable only at the starting of the program so that it can allocate the memory and then works on variable.

- In JavaScript, it is not mandatory to declare the variables only at the starting of the program but can be declared anywhere we wanted.
- While executing the application JavaScript performs variable hosting process in which it identifies all the variable declarations within the application and moves all the declarations to starting of the applications.
- The process of moving all the variable declarations to the starting of the application is called variable hosting.

### **Note:**

While moving the declarations it only moves declarations to the starting but not the initialized values.

### ➤ **Data Structures:**

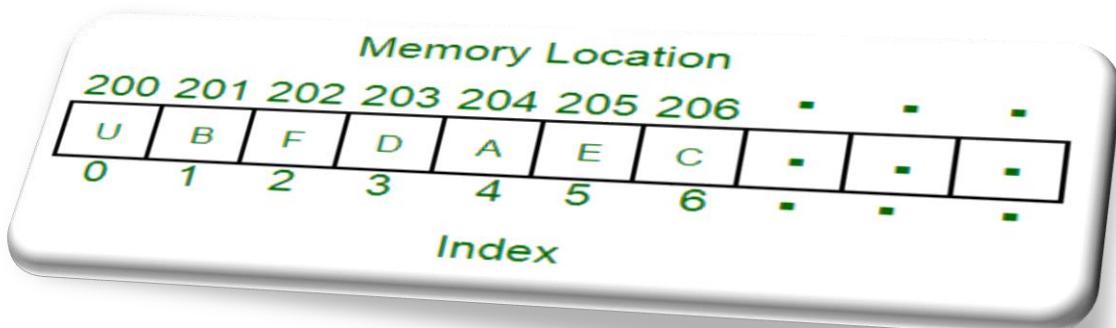
Following are the different data structures being supported in the JavaScript through which different type of data can be stored and retrieved.

- Arrays
- Json
- Map
- Set
- Tables

### ➤ **Array Data Structure:**

A predefined data structure through which we could able to hold group of multiple values within it.

- In JavaScript, arrays are the **heterogeneous data structures** capable of holding different types of data with in it.
- Array is capable of holding different and multiple types of data, every other value will be identified with single variable name.
- All the values within the array will be automatically assigned with a numeric **index** value.
- The **index** value is always starts with “Zero”.
- ‘[ ]’ are always used to represent the arrays in any language.
- Within the memory, arrays always occupy continuous memory allocation for all the values within it.
- We make use combination of array name and corresponding index position while working with an array.



- Following are the two different ways we can create an array in JavaScript,

**var a = new array() //dynamic allocation**

**(OR)**

**var a = [ ]; // empty array**

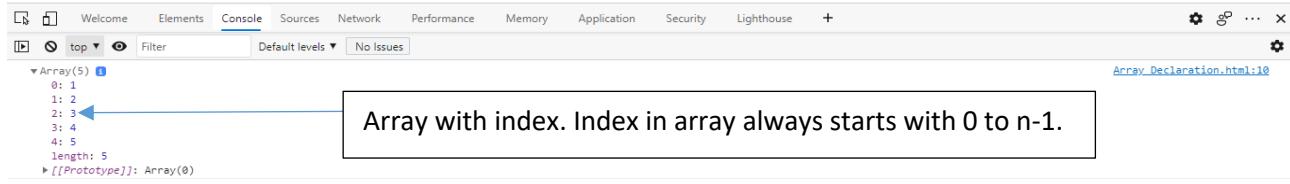
**var a = [1, 2, 3, 4]; //declaring an array with values.**

**Ex:** `<script type="text/javascript">`

**var a = [1, 2, 3, 4, 5];**

**console.log(a);**

**</script>**



### ➤ Pre-Defined methods supports in array:

Following are the some predefined methods supported by array in JavaScript,

- Length:**

It is a property which returns the total number of values in an array.

**Syntax:** `arrayName.length;`

- Push(<values>):**

Used to insert single or multiple values to an array from right direction.

**Syntax:** `arrayName.push(<value>);`

- Pop():**

Used to delete single value from an array from right side.

**Syntax:** `arrayName.pop();`

- Shift():**

Used to delete single value from an array from left direction.

**Syntax:** `arrayName.shift();`

- **Unshift(<values>):**

Used to insert single/multiple values to an array from left direction.

**Syntax:** arrayName.unshift(<values>);

- **Splice(starting position, <no.of.values to be deleted>,<optional values to be inserted>):**

Used to insert or delete values.

**Syntax:** arrayName.splice(starting position, <no.of.values to be deleted>,<optional values to be inserted>);

- **Join():**

Used to merge all the values of array to a single value with/without separator.

➤ **Predefined methods can be applied on string:**

Every string being created in JavaScript internally it gets created in the form of an array. Following are the predefined methods can be applied on string in JavaScript,

- **Length:**

It returns the total number of characters within a string, including spaces.

- **charAt(<index>):**

Returns the character present at the provided index position.

- **charCodeAt(<index>):**

Returns the ASCII value of a character present at provided index position.

- **subStr(index,length of substring):**

Returns the substring from the given string.

- **replace("what","with"):**

Used to replace the string parts with required strings.

- **Split("optional separator"):**

Used to split the main string in the form of an array.

- **Includes(“<string>”):**  
Checks whether the provided string is part of main string and returns true or false.
- **Indexof(“string”):**  
Returns at what index provided substring exists in main string.
- **tolowercase():**  
Converts the provided string to lower case format.
- **touppercase():**  
Converts the provided string to upper case format.
- **Slice(start index, end index):**  
Returns the substring from one index to other.

✓ **Note:**

Any string being defined in JavaScript it occupies the memory internally as like an array where each character occupies individual block with the corresponding index value.

## ➤ **Working with JavaScript “Date” object:**

“Date” is a predefined class supported in JavaScript using which we could able to work with the system's current date or user defined custom date.

### **Syntax:**

**var date = new Date();** //creates a reference of system's current date.

**var date = new Date(<user specific date>);** // user specific date

**Following are the predefined methods which can be applied on date object,**

- **get Date():**  
Returns the current date value( 1<sup>st</sup> -31<sup>st</sup> ).
- **get Month():**  
Returns the current month value(0<sup>th</sup>-11<sup>th</sup>).
- **getFullYear():**  
Returns the full year

- **get Hours():**

Returns the hours value.

- **get Minutes():** Returns 0-59
- **get Seconds():** Returns 0-59
- **get Milliseconds():** Returns 0-999
- **get Day():** Returns 0-6.

➤ **JavaScript functions:**

The set of instructions binded as a module with in “{ }” assigned with an user defined custom name which can be reused anywhere within the page is called a function.

- In JavaScript ‘**function**’ is a predefined keyword using which we can define a function.**[until ECMA6]**.
- In a single page we can define any number of functions in JavaScript.
- Once the function ben defined it can be invoked any number of times through the function name.
- While defining the function we name it through **function** keyword, while calling we simply use function name.

**Ex:**

```
funDemo.html x
practice > JavaScripts > Functions > funDemo.html > html > body > script > secondval
3   <head>
4     <title>JS Functions Demo</title>
5   </head>
6   <body>
7     <script type="text/javascript">
8       var firstval=10;
9       function firstvalue() {
10         var firstval = 11;
11         console.log("inside the first val function : " + firstval)
12       }
13       function secondval() {
14         var secval = 12;
15         console.log("inside the second function : " + secval);
16         firstvalue();
17       }
18       console.log("global variable : " + firstval);
19       firstvalue();
20       secondval();
21     </script>
22   </body>
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
global variable : 10
inside the first val function : 11
inside the second function : 12
inside the first val function : 11
Filter (e.g. text, !exclude)
practice/JavaScripts/Functions/funDemo.html:18
practice/JavaScripts/Functions/funDemo.html:11
practice/JavaScripts/Functions/funDemo.html:15
practice/JavaScripts/Functions/funDemo.html:11
```

## ➤ Local variables and global variables:

Any variable which is declared outside of a module will become global variable can be accessible anywhere within the application.

Global variable can be accessible and modifiable anywhere within the page.

Local variable can't be accessed outside the module it can be accessed within the module.

**Ex: var a = 100 //global variable**

**Function test(){**

**var a = 101; //local variable**

**}**

## ➤ Accessing private data of a function outside of it:

- Through passing parameters while calling a function.
- Returning value from called function to calling function.
- Both passing parameters and returning values.

## ➤ Passing parameters while calling a function:

While calling a function within another function we could able to pass local-variables of calling function to called function.

- While calling a function we can pass any number of parameters.
- While defining a function we could able to catch the values being passed while calling a function, using temporary variables.
- The parameters being passed while calling a function are called as actual parameters whereas the temporary parameter declared at function definition are called formal parameters.

**Ex:**

The screenshot shows a code editor with a dark theme. On the left is a file tree with a single file named 'passinParam.html'. The code is as follows:

```
11
12     function num(){
13         var firstnum = 5;
14         var seconnum = 3;
15
16         var result;
17         add(firstnum,seconnum); //using the values of firstnum,seconnum variable
18         mul(firstnum,seconnum); //using the values of firstnum,seconnum variable
19
20     }
21
22     //performing addition operation and the values taken from local variables which is in "num()" function
23     function add(a,b){
24         result = a + b; //a,b are temporary data
25         console.log("addition is" + result);
26     }
27
28     //performing multiplication operation and the values taken from local variables which is in "num()" function
29     function mul(p,q){
30         result = p * q; //p,q are the temporary data
31         console.log("multiplicaton is : " + result)
32     }
33
34     num();
```

At the bottom of the code editor, there are tabs for PROBLEMS, OUTPUT, TERMINAL, and DEBUG CONSOLE. The DEBUG CONSOLE tab is selected, showing the output:

```
addition is8
multiplicaton is : 15
```

In the bottom right corner of the code editor, there is a terminal window showing the command line interface:

```
Filter (e.g. text, exclude)
-ts/Functions/passingParameters/passinParam.html:24
-ts/Functions/passingParameters/passinParam.html:29
```

- While passing formal parameters at function definition we don't need to specify or declare those parameters we simply specify the variable names.

➤ **Returning a value from the called function to calling function:**

```

6   <script type = "text/JavaScript">
7     function num() {
8       var first = 2;
9       var second = 4;
10      var temp = add(first,second);
11      console.log("addition is " + temp);
12      var temp1 = mul(first,second);
13      console.log("multiplication is " + temp1);
14
15    }
16    function add(a,b){
17      var result;
18      var result = a + b;
19      return result;
20    }
21    function mul(p,q){
22      var result;
23      result = p * q;
24      return result;
25    }
26    num();
27  </script>

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Filter (e.g. text, !exclude) Open UsingReturn.html ▾ ... ^ ×

addition is 6  
multiplication is 8  
...ts/Functions/passingParameters/UsingReturn.html:11  
...ts/Functions/passingParameters/UsingReturn.html:13

➤ **“Arguments” Keyword:**

A predefined keyword supported in JavaScript and by default available in every function which holds all the parameter values been passed to that function in the form of an array.

- In JavaScript it is mandatory to catch the parameters in function definition while matching function call with function definition it only matches function name but never looks for type of parameter or number of parameters being passed.
- Irrelevant of whether the parameters been catched or not, arguments keyword gets created in every function and holds all the parameters being passed to it in the form of an array.

```

argumentKeyword.html ×
Scripts > Functions > argumentKeyword.html > html > body > script > addValues
<!DOCTYPE html>
<html>
  <head>
    <title>JS Functions Demo</title>
  </head>
  <body>
    <script type="text/javascript">
      function test() {
        console.log(arguments);
      }

      function addValues() {
        var result = 0;
        for (var i = 0; i < arguments.length; i++) {
          result += arguments[i];
        }
        // result = a + b + c;
        console.log("The sum is " + result);
      }

      addValues(34, 78, 23, 33, 55, 12);
      test(23, 44);
    </script>

```

### ➤ JSON Data Structure:

**JSON** is derived as **JavaScript Object Notation**.

- It is **predefined data structure** being supported in JavaScript by default.
  - It is used to not just hold the data but, holds the data with the extra information of particular data.
  - The JSON Data gets stored in the format of **key and value** based.

**Ex: key: value,**

**Key2:** value2;

- All the keys within the JSON Object should be only **string** data type, whereas the **corresponding value** could be of any **JavaScript supported Data Type**.
  - While accessing data from object we make use of the combination of object name and corresponding key.

**Ex:**

```
ne wrap
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8">
5          <title></title>
6      </head>
7      <body>
8          <script type="text/javascript">
9              var empdetails = {
10                  'name' : 'raju',
11                  'dept' : 'it',
12                  'age' : 20,
13                  'basicosal' : 25000,
14              }
15              empdetails.hra = 15/100 * empdetails.basicosal;
16              empdetails.pf = 25/100 * empdetails.basicosal;
17              empdetails.total = empdetails.basicosal + empdetails.hra + empdetails.pf;
18              if (empdetails.total > 20000) {
19                  empdetails.tax = empdetails.total / 15;
20              } else {
21                  empdetails.tax = 0;
22              }
23              console.log("name" + empdetails.name);
24              console.log("dept" + empdetails.dept);
25              console.log("age" + empdetails.age);
26              console.log("basic salary" + empdetails.basicosal);
27              console.log("hra is " + empdetails.hra);
28              console.log("pf is " + empdetails.pf);
29              console.log("total is" + empdetails.total);
30              console.log("tax is :" + empdetails.tax);
31          </script>
32      </body>
33  </html>
34
35  <\n\w\j>
36  <\n\w\j>
37  <\n\w\j>
38  <\n\w\j>
39  <\n\w\j>
40  <\n\w\j>
41  <\n\w\j>
42  <\n\w\j>
43  <\n\w\j>
44  <\n\w\j>
45  <\n\w\j>
46  <\n\w\j>
47  <\n\w\j>
```

#### ➤ Memory allocation of JSON object:

Internally JSON object gets memory allocated same as like an array where the only difference is, the key of the data is assigned as index value by default.

**Ex:**

```

7     <body>
8         <script type="text/javascript">
9             var empdetails = {
10                 'name' : 'raju',
11                 'dept' : 'it',
12                 'age' : 20,
13                 'basicSal' : 25000,
14             }
15             console.log(empdetails)
16             var a = [1, 2, 3, 4, 5];
17             console.log(a);
18         </script>

```

```

Welcome Elements Console Sources Network Performance Memory Application Security Lighthouse +
jsonDemo.html:15
top Filter Default levels ▾ 5
jsonDemo.html:15
{
  name: "raju",
  dept: "it",
  age: 20,
  basicSal: 25000
}
{
  age: 20
  basicSal: 25000
  dept: "it"
  name: "raju"
  [[Prototype]]: Object
}
[1, 2, 3, 4, 5]
0: 1
1: 2
2: 3
3: 4
4: 5
length: 5
[[Prototype]]: Array(0)

```

Here the data in the JSON is allocated as same as an array but the only difference is array uses index value but in case of json it uses key.

## ➤ Anonymous function:

Any function which is defined without a name is called an anonymous function. In case we want to define a function it will be invoked only once within the function do not waste the memory by giving a name to it, we can simply create anonymous function.

**Syntax: function (){**

**}**

## ➤ thisKeyword:

A predefined Key word been supported in JavaScript which refers the current object data.

In order to access objects data within its corresponding methods, we can access either by using object name or by using this keyword.

### Ex:

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <title></title>
6      </head>
7      <body>
8          <script type="text/javascript">
9              var empdetails = {
10                  name: "raju",
11                  dept: "it",
12                  age: 20,
13                  basicSal: 25000,
14                  empsal: function () {
15                      this.hra = (15 / 100) * this.basicSal;
16                      console.log("HRA is :" + this.hra);
17                  },
18              };
19              empdetails.empsal();
20          </script>

```

## ➤ for in loop Control Structure:

A Predefined looping Control Structure through which we could able to iterate over a **JSON** object.

**Ex:**

The screenshot shows a code editor window for a file named 'forinLoop.html'. The code defines a variable 'empDetails' with properties name, age, gender, and marks. A 'for' loop iterates over the keys of 'empDetails', logging each key and its corresponding value to the console. The output panel shows the logged values: 'name', 'ananth', 'age', '23', 'gender', 'male', 'marks', and '[10, 20, 30]'. The terminal panel shows multiple log entries for each key-value pair.

```
<html>
  <head>
    <title>ForInLoop Demo</title>
  </head>
  <body>
    <script>
      var empDetails = {
        name: "ananth",
        age: 23,
        gender: "male",
        marks: [10, 20, 30],
      };
      for (var tempData in empDetails) {
        console.log(tempData);
        console.log(empDetails[tempData]);
      }
    </script>
  </body>
</html>
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Filter (e.g. text, exclude)

```
name
ananth
age
23
gender
male
marks
> (3) [10, 20, 30]
```

```
practice/JavaScripts/for in loop/forinLoop.html:14
practice/JavaScripts/for in loop/forinloop.html:15
```

## ➤ JavaScript ‘setTimeout’ and ‘setInterval’ methods:

A two predefined methods been supported in JavaScript using which we could able to invoke set of instructions not just when the controller reaches to it but, executes the set of instructions only after the provided interval time.

**Syntax: setTimeout(<callbackmethod>,<delayvalue>);**

The screenshot shows a code editor window for a file named 'setTimeout\_setInterval.html'. It demonstrates the use of 'setTimeout' and 'setInterval' to log 'hello' and '1' once, and '2' every 3 seconds. The terminal panel shows the output: 'hello', '1', and '2' printed at regular intervals.

```
<html>
  <head>
    <title>setTimeout, setInterval Demo</title>
  </head>
  <head>
    <script>
      console.log("hello");
      function time() {
        console.log("1");
      }
      function interval() {
        console.log("2");
      }
      setTimeout(time, 2000);
      setInterval(interval, 3000);
    </script>
  </head>
</html>
```

- setInterval and setTimeout always works the same where the only difference is setTimeout invokes the function only once after the provided interval time whereas setInterval method invokes the call-back method repeatedly with a gap of provided interval time.

**Note:**

clearInterval () is a predefined method supported through which we could able to stop a running interval job.

**Ex:**

```
16 |     var count = 0;
17 |     var myInterval = setInterval(function () {
18 |         count++;
19 |         console.log("hello");
20 |         if (count == 4) [
21 |             clearInterval(myInterval);
22 |         ]
23 |     }, 3000);
24 |     console.log("done");
25 |     </script>
26 |   </head>
27 | </html>
28 |
```

## **DHTML – Dynamic HTML**

- It is a powerful language comes with the predefined objects using which we could able to perform any type of **CRUD** operations on any DOM Elements on its corresponding CSS Property.
- Using JavaScript we could able to generate the complete HTML page with all **dynamic data** as and when required.
- It provides predefined methods and objects using which we could able to handle any type of events being generated on the page.
- It provides predefined methods and objects using which we could able to read, write or form on the HTML input elements.

### **Note:**

‘Document’ and ‘window’ are two predefined object by default available in every web page is actually holds the complete DOM structure on which we can perform the CRUD operations.

#### ➤ **‘Document’ and ‘window’:**

The two predefined objects been supported in JavaScript using through which we could perform any type of operation on any HTML Element of its corresponding CSS Properties.

Following are the different ways we could refer a DOM Elements on a page,

#### \* **document.getElementById():**

Using which we could refer to a DOM Element on page using its unique ID.

#### \* **document.getElementByClassName():**

Using which we could refer to a DOM Element on page using its name.

#### \* **document.getElementByName():**

Using which we could refer to a DOM Element on page using its tag name.

#### \* **document.querySelector():**

A method been supported in HTML5 using which we can refer to any DOM Element dynamically using JavaScript, may be through class, id or tag name.

**Syntax: document.querySelector(#id / .class / tagName);**

Following are the methods we can be applied on element reference,

### \* **element.setAttribute("attributeName",value):**

Using which we could able to set any attribute dynamically to a DOM Element.

**Ex:** `spanTag.setAttribute('class', 'content');`

`spanTag.setAttribute("id", uid);`

### \* **element.getAttribute("attributeName"):**

Using which we can get corresponding attribute value of any DOM Element.

### \* **Element.style:**

Using which dynamically any CSS Property of any DOM Element can be retrieved or updated.

**Ex:** `spanTag.style.border = '1px solid green';`

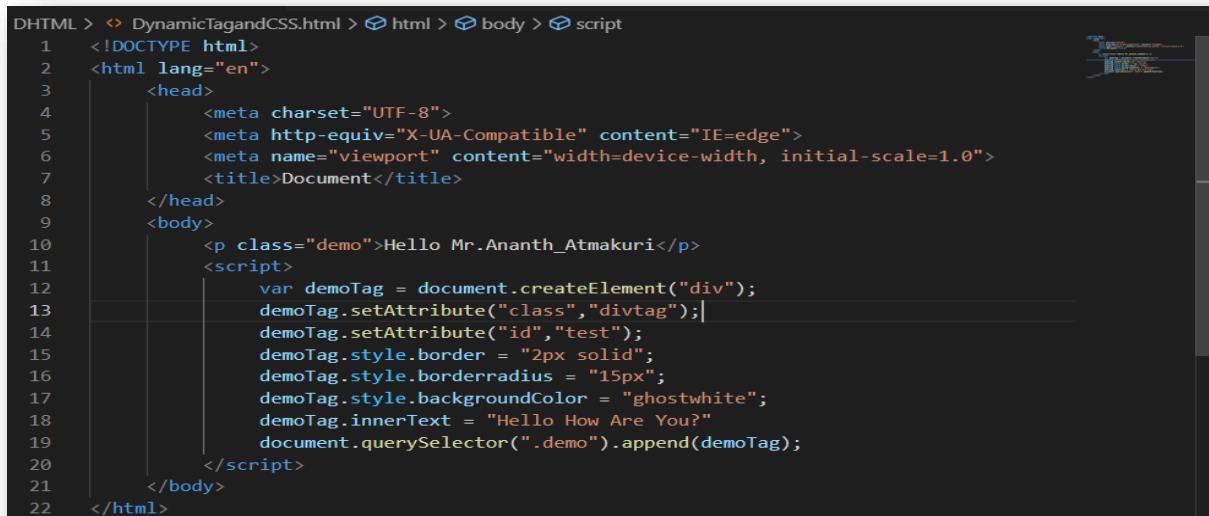
`spanTag.style.backgroundColor = bgColor;`

### \* **Element.appendChild(<element>):**

Using which we could able to append an element to an existing DOM Element page.

### \* **document.createElement(<tag name>):**

Using which we could able to create any DOM Element using JavaScript.



```
DHTML > DynamicTagandCSS.html > html > body > script
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="UTF-8">
5          <meta http-equiv="X-UA-Compatible" content="IE=edge">
6          <meta name="viewport" content="width=device-width, initial-scale=1.0">
7          <title>Document</title>
8      </head>
9      <body>
10         <p class="demo">Hello Mr.Ananth_Atmakuri</p>
11         <script>
12             var demoTag = document.createElement("div");
13             demoTag.setAttribute("class","divtag");
14             demoTag.setAttribute("id","test");
15             demoTag.style.border = "2px solid";
16             demoTag.style.borderradius = "15px";
17             demoTag.style.backgroundColor = "ghostwhite";
18             demoTag.innerText = "Hello How Are You?";
19             document.querySelector(".demo").append(demoTag);
20         </script>
21     </body>
22 </html>
```

## ➤ Reading values from Input Elements:

- JavaScript provides predefined methods and object through which we could able to read data and value from input elements.
- **element.value** which returns the values from input elements and text area.
- **element.options** which returns an array of options as object from a dropdown element.
- **element.selectedIndex** which returns the current selected index of dropdown element.
- **element.checked** which returns the true / false of current selected or non-selected state of checkbox.

### Ex: HTML Code

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PaySlip Details</title>
    <link rel="stylesheet" href="emp.css">
  </head>
  <body>
    <div class="header">Employee Salary Details</div>
    <ul>
      <li>
        <label for="empname">Enter the Employee Name : &nbsp;</label>
        <input type="text" name="empname" id="ename" placeholder="Enter Here">
      </li>
      <li>
        <label for="age">Enter the Employee Age : &nbsp;</label>
        <input type="text" name="age" id="eage" placeholder="Enter Here">
      </li>
      <li>
        <label for="gender">Select the Gender : &nbsp;</label>
        <input type="radio" name="gender" id="egender" value="male">Male
        <input type="radio" name="gender" id="agender" value="Female">Female
      </li>
      <li>
        <label for="empdept">select Department</label>
        <select id="edepth" name="empdept">
          <option></option>
          <option>IT</option>
          <option>Networking</option>
          <option>Sales</option>
          <option>HR</option>
        </select>
      </li>
    </ul>
  </body>
</html>
```

```
<ul>
  <li>
    <label for="rname">Employee Name :</label>
    <input type="text" name="rname" id="rname" value="Naresh i@technologies">
  </li>
  <li>
    <label for="rage">Employee Age :</label>
    <input type="text" name="rage" id="rage" value="28">
  </li>
  <li>
    <label for="rgender">Employee Gender :</label>
    <input type="radio" name="rgender" id="rgender" checked="" value="Male">Male
    <input type="radio" name="rgender" id="rgender" value="Female">Female
  </li>
  <li>
    <label for="rdept">Employee Department :</label>
    <input type="text" name="rdept" id="rdept" value="IT">
  </li>
  <li>
    <label for="rbsal">Employee Basic Salary :</label>
    <input type="text" name="rbsal" id="rbsal" value="50000">
  </li>
  <li>
    <label for="rpF">Employee PF :</label>
    <input type="text" name="rpF" id="rpF" value="5000">
  </li>
  <li>
    <label for="rhra">Employee HRA :</label>
    <input type="text" name="rhra" id="rhra" value="15000">
  </li>

```

```

63         </li>
64     <li class="inner">
65         <b>Employee Total Salary is : </b><span id="rtotalsal"></span>
66     </li>
67     <li class="inner">
68         <b>Employee Tax is : </b><span id="etax"></span>
69     </li>
70     <li class="inner">
71         <b>Employee Salary after Tax Deduction : </b><span id="tsal"></span>
72     </li>
73 </ul>
74 <script src="empDetails.js"></script>
75
76 </body>
77 </html>

```

## • JS Code

File Edit Selection View Go Run Terminal Help empDetails.js - public - Visual Studio Code

practice > DHTML > Employee Details > **JS** empDetails.js > **getEmployeeDetails**

```

1 var employeeDetails = {};
2 employeeDetails.getEmployeeDetails = function () {
3     this.ename = document.querySelector("#ename").value;
4     this.eage = document.querySelector("#eage").value;
5     //this.egender = document.querySelector("input[name=gender]:checked").value;
6     this.edept = document.querySelector("#edept").value;
7     this.basicsal = document.querySelector("#basicsal").value;
8     this.basicsal = parseInt(employeeDetails.basicsal);
9     this.getTotalsal();
10 }
11
12 employeeDetails.getTotalSal = function () {
13     this.hra = 0.13 * this.basicsal;
14     this.pf = 0.15 * this.basicsal;
15     this.totalsal = this.hra + this.pf + this.basicsal;
16     employeeDetails.getTax () => {
17         if (this.egender == "male"){
18             if (this.totalsal > 400000){
19                 this.etax = 0.15 * this.totalsal;
20             } else if (this.totalsal > 200000){
21                 this.etax = 0.10 * this.totalsal;
22             } else if (this.totalsal > 100000){
23                 this.etax = 0.05 * this.totalsal;
24             } else {
25                 this.etax = 0 * this.totalsal;
26             }
27         }
28         if (this.egender == "Female"){
29             if (this.totalsal > 400000){
30                 this.etax = 0.10 * this.totalsal;
31             } else if (this.totalsal > 200000){
32                 this.etax = 0.05 * this.totalsal;
33             } else if (this.totalsal > 100000){
34                 this.etax = 0.02 * this.totalsal;
35             } else {
36                 this.etax = 0 * this.totalsal;
37             }
38         }
39     }
40     this.getTax();
41     employeeDetails.tsal = this.totalsal - this.etax;
42     employeeDetails.displaytotalsal();
43 }
44 employeeDetails.displaytotalsal = function(){
45     document.querySelector(".paylip").style.display = 'block';
46     document.querySelector("#rname").innerHTML = this.ename;
47     document.querySelector("#rage").innerHTML = this.eage;
48     document.querySelector("#rgender").innerHTML = this.egender;
49     document.querySelector("#rdept").innerHTML = this.edept;
50     document.querySelector("#rbasic").innerHTML = this.basicsal;
51     document.querySelector("#rhra").innerHTML = this.hra;
52     document.querySelector("#rpf").innerHTML = this.pf;
53     document.querySelector("#rtotalsal").innerHTML = this.totalsal;
54     document.querySelector("#etax").innerHTML = this.etax;
55     document.querySelector("#tsal").innerHTML = this.tsal;
56 }
57 function generatePaySlip() {
58     employeeDetails.getEmployeeDetails();
59 }

```

File Edit Selection View Go Run Terminal Help empDetails.js - public - Visual Studio Code

practice > DHTML > Employee Details > **JS** empDetails.js > **generatePaySlip**

```

26     if (this.egender == "female"){
27         if (this.totalsal > 400000){
28             this.etax = 0.10 * this.totalsal;
29         } else if (this.totalsal > 200000){
30             this.etax = 0.05 * this.totalsal;
31         } else if (this.totalsal > 100000){
32             this.etax = 0.02 * this.totalsal;
33         } else {
34             this.etax = 0 * this.totalsal;
35         }
36     }
37     this.getTax();
38     employeeDetails.tsal = this.totalsal - this.etax;
39     employeeDetails.displaytotalsal();
40 }
41 employeeDetails.displaytotalsal = function(){
42     document.querySelector(".paylip").style.display = 'block';
43     document.querySelector("#rname").innerHTML = this.ename;
44     document.querySelector("#rage").innerHTML = this.eage;
45     document.querySelector("#rgender").innerHTML = this.egender;
46     document.querySelector("#rdept").innerHTML = this.edept;
47     document.querySelector("#rbasic").innerHTML = this.basicsal;
48     document.querySelector("#rhra").innerHTML = this.hra;
49     document.querySelector("#rpf").innerHTML = this.pf;
50     document.querySelector("#rtotalsal").innerHTML = this.totalsal;
51     document.querySelector("#etax").innerHTML = this.etax;
52     document.querySelector("#tsal").innerHTML = this.tsal;
53 }
54 function generatePaySlip() {
55     employeeDetails.getEmployeeDetails();
56 }

```

Ln 59, Col 2 Spaces: 5 UTF-8 CRLF JavaScript 12:29 10-08-2021

## ➤ **Event Handling:**

### \* **Event:**

Any action been performed on a page is called an event.

- \* The process of performing a particular job based on a particular action been generated on a page is called an Event Handling.
- \* Following are the types of action might get generated on a page,

- Click - dbClick - mouseover
- Drag - drop - keypress
- Keydown - focus - blur

- \* Following are the two ways we could able to implement event handling,

- Static Event Handling
- Dynamic Event Handling

### \* **Static Event Handling:**

In this way, using HTML predefined static event attributes, we statistically specify the particular JavaScript function need to be invoked when a particular action happens on a page.

Following are the pre-defined HTML Attributes,

- onclick	-onmouseover	-onfocus	-onchange	-onblur
-onkeyup	-onkeydown	-onmouseout	-ondbclick	

**Ex: <div onclick = “displaydata();”></div>**

```
23 <div>
24   <ul>
25     <li>
26       <label for="fVal">Enter the First value : &nbsp; </label>
27       <input type = "text" name = "fVal" id = "fVal" required = "fVal">
28     </li>
29     <li>
30       <label for="sVal">Enter the Second value : &nbsp; </label>
31       <input type = "text" name = "sVal" id = "sVal" required = "sVal">
32     </li>
33     <li>
34       <input type="button" value = "Add" onclick="addition();">
35     </li>
36     <span id = "res"></span>
37   </ul>
38   <script>
39     function addition() {
40       var firstVal = document.querySelector("#fVal").value;
41       firstVal = parseInt(firstVal);
42       var secondVal = document.querySelector("#sVal").value;
43       secondVal = parseInt(secondVal);
44       var result = firstVal + secondVal;
45       result = document.querySelector("#res").innerHTML = "The result is : " + result;
46     }
47   </script>
48 </div>
```

The screenshot shows a simple HTML form. It has two text input fields labeled "Enter the First value:" and "Enter the Second value:". Below these is a button labeled "Add". Underneath the button is a span element labeled "The result is : 4".

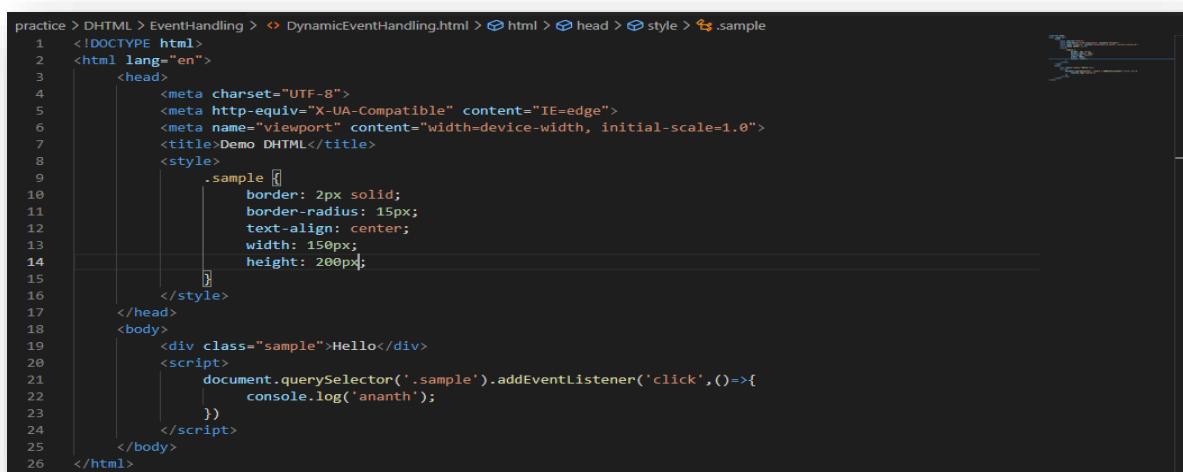
## \* **Dynamic Event Handling:**

- JavaScript provides a feature through which we can dynamically specify what method need to be invoked when a particular action happens on a DOM Element.
- “**addEventListener**” It is a predefined method using which we can handle action been predefined on a page dynamically.

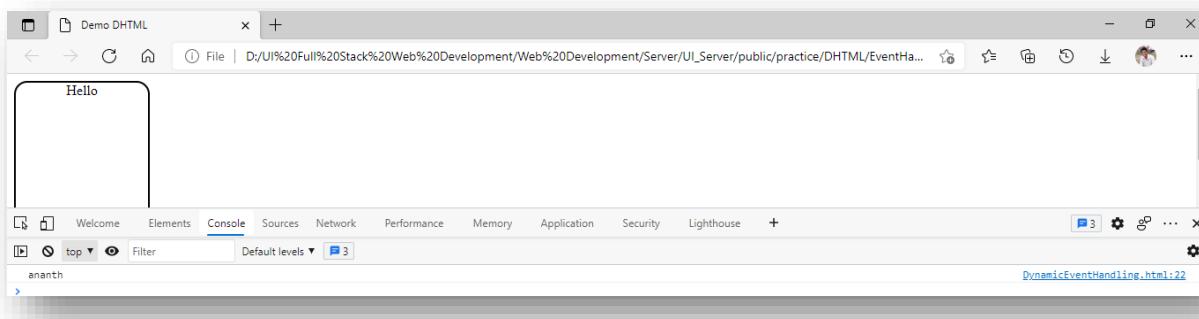
**Syntax: element.addEventListener(“typeofEvent”,<callbackmethod>);**

- This method attaches an event handler to an element without corresponding existing event handling.
- You can add many event handlers to any element.

**Ex:**



```
practice > DHTML > EventHandling > DynamicEventHandling.html > html > head > style > .sample
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="UTF-8">
5          <meta http-equiv="X-UA-Compatible" content="IE=edge">
6          <meta name="viewport" content="width=device-width, initial-scale=1.0">
7          <title>Demo DHTML</title>
8          <style>
9              .sample {
10                  border: 2px solid;
11                  border-radius: 15px;
12                  text-align: center;
13                  width: 150px;
14                  height: 200px;
15              }
16          </style>
17      </head>
18      <body>
19          <div class="sample">Hello</div>
20          <script>
21              document.querySelector('.sample').addEventListener('click',()=>{
22                  console.log('ananth');
23              })
24          </script>
25      </body>
26  </html>
```



## ➤ **Event Bubbling and Capturing.**

## Advanced JavaScript

### ➤ Inheritance:

The process of defining an object and accessing the data of it (data members and data functions) within another object is called inheritance.

**Inheritance** is a feature in which one class inherits the property of another class. A class which inherits the property is called a **derived class** or **subclass** or **child class** and from which derived class inherits property is called as a **base class** or **parent class**.

Inheritance makes **reusability** of code, which avoids rewriting the existing code again and again. A class can be driven from one class or from multiple base classes.

JavaScript supports two types of inheritance,

- I. Single level Inheritance.
- II. Multilevel Inheritance.

### \* Single level Inheritance:

Single Level inheritance - A class inherits properties from a single class. For example, Class B inherits Class A.



Ex:

```
1 var schoolDetails = {  
2     schoolName : 'Public School',  
3     area : 'all areas'  
4 }  
5  
6 var studentDetails = {  
7     sid: 15503,  
8     name: 'Ram',  
9     Age: 10,  
10    class : 4,  
11    displaystudentDetails: function() {  
12        document.querySelector('#school').innerHTML = 'School Name : ' + this.schoolName;  
13        document.querySelector('#state').innerHTML = 'Availability : ' + this.area;  
14        document.querySelector('#sid').innerHTML = 'Student ID : ' + this.sid;  
15        document.querySelector('#name').innerHTML = 'Student Name : ' + this.name;  
16        document.querySelector('#class').innerHTML = 'Student ID : ' + this.class;  
17        document.querySelector('#age').innerHTML = 'Student ID : ' + this.Age;  
18    }  
19};  
20  
21 studentDetails.__proto__ = schoolDetails;  
22 studentDetails.displaystudentDetails();
```

Here schoolDetails object is parent i.e. base class

Here studentDetails is child class or derived class

School Name : Public School
Availability : all areas
Student ID : 15503
Student Name : Ram
Student ID : 4
Student ID : 10

- **Note:**

Due to data ambiguity issues in JavaScript does not support Multiple Inheritance.

Following are the different ways between types of object inheritance can be implemented.

- Inheritance between two static objects.
- Inheritance between static object and dynamic object.
- Inheritance between two dynamic objects.

\* **Inheritance between two static objects:**

- Every static object been created in JavaScript holds a predefined property by default “**\_\_proto\_\_**”.
- Using “**\_\_proto\_\_**” we could able to extend the behaviour of any object to access another existing object with properties and behaviour.
- The main object from which data gets derived is called **parent object** or **base object**.
- The object which is accessing the data or got derived from the parent object is called **child object** or **derived object**.
- For Example Refer **Single Level Inheritance**.

\* **Inheritance between static object and dynamic object:**

- “**object.create()**” is a predefined way to create a dynamic object by inheriting data from a static object.

**Syntax: object.create(<existing static object>);**

**Ex: var data = {**

-----

-----

}

**var childobj = object.create(data);**

Here **childobj** holds its own properties along with it can point to existing data of “**data**” object.

```

var schoolData = {
    schoolName: 'Abc School',
    schoolLocation: 'Hyderabad',
    schoolRating: '4/5'
}

function registerStudent() {
    var studentData = Object.create(schoolData);
    // var studentData = {};
    studentData.name = document.querySelector("#uname").value;
    studentData.age = document.querySelector("#uage").value;
    studentData.class = document.querySelector("#uClass").value;
    studentData.address = document.querySelector("#uAddress").value;

    var ulTag = document.createElement("ul");
    for (var key in studentData) {
        var li = document.createElement('li');
        li.innerHTML = key + ' : ' + studentData[key];
        ulTag.appendChild(li);
    }
    document.querySelector(".detilsContainaner").append(ulTag);
}

```

## ➤ **JavaScript Class:**

Class indicates or represents a predefined structure of an object so that we can create any number of objects having same structure but with different data until **ECMA5**. Classes we are not directly supported in JavaScript, using functions we could indirectly get the feature of Class.

From **ECMA6**, Class is a predefined Keyword through which we could able to create classes directly.

## ✧ **ECMA5 Class Syntax:**

```

function <class name>(<optional params>){

    this.key = '..';

    this.key2='..';

    .....

    this.method = function() {

        .....

    }

}

```

## ✧ **ECMA-6:**

```

Class ClassName {

    constructor(optionalparams) {

        this.key = '..';

        this.key2='..';

        .....
    }
}

```

```

        }

methodName() {

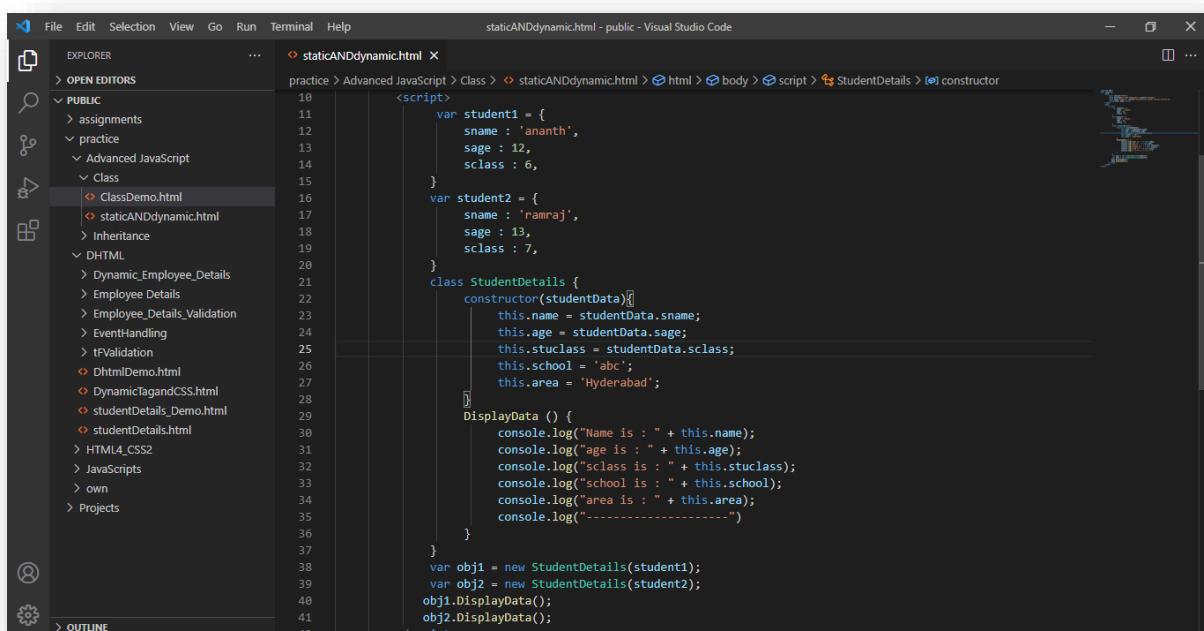
    .....

}

}

```

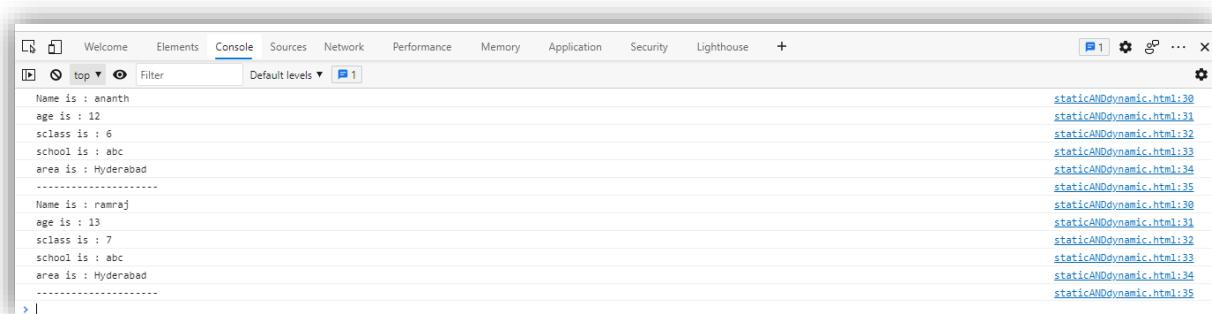
## Ex:



```

File Edit Selection View Go Run Terminal Help
staticANDdynamic.html - public - Visual Studio Code
EXPLORER OPEN EDITORS PUBLIC
> assignments
> practice
> Advanced JavaScript
> Class
> ClassDemo.html
> staticANDdynamic.html
> Inheritance
> DHTML
> Dynamic_Employee_Details
> Employee Details
> Employee_Details_Validation
> EventHandling
> tFValidation
> DhtmlDemo.html
> DynamicTagandCSS.html
> studentDetails_Demo.html
> studentDetails.html
> HTML4_CSS2
> JavaScripts
> own
> Projects
> OUTLINE
<script>
    var student1 = {
        sname : 'ananth',
        sage : 12,
        sclass : 6,
    }
    var student2 = [
        sname : 'ramraj',
        sage : 13,
        sclass : 7,
    ]
    class StudentDetails {
        constructor(studentData){
            this.name = studentData.sname;
            this.age = studentData.sage;
            this.sclass = studentData.sclass;
            this.school = 'abc';
            this.area = 'Hyderabad';
        }
        DisplayData () {
            console.log("Name is : " + this.name);
            console.log("age is : " + this.age);
            console.log("sclass is : " + this.sclass);
            console.log("school is : " + this.school);
            console.log("area is : " + this.area);
            console.log("-----");
        }
    }
    var obj1 = new StudentDetails(student1);
    var obj2 = new StudentDetails(student2);
    obj1.DisplayData();
    obj2.DisplayData();
</script>

```



```

Welcome Elements Console Sources Network Performance Memory Application Security Lighthouse +
top Filter Default levels 1
Name is : ananth
age is : 12
sclass is : 6
school is : abc
area is : Hyderabad
-----
Name is : ramraj
age is : 13
sclass is : 7
school is : abc
area is : Hyderabad
-----

```

- In Class first the **Constructor** is executed.
- **inherit from two Classes:**
- The **extends** keyword is used to create a child class of another class (parent). The child class inherits all the methods from another class. Inheritance is useful for code reusability: reuse properties and methods of an existing class when you create a new class.

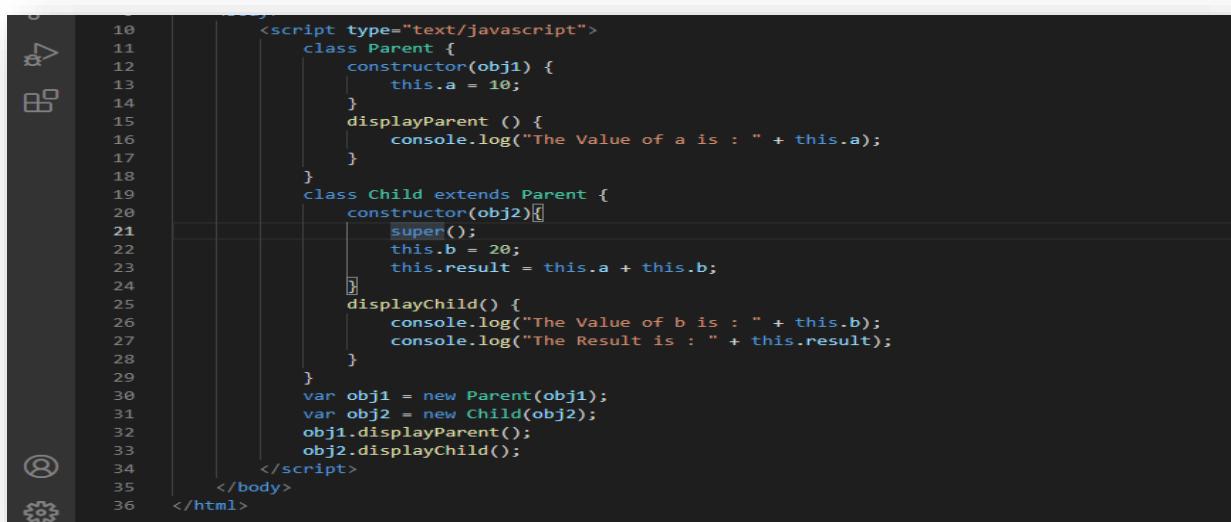
The super() method refers to the parent class. By calling the super() method in the constructor method, we call the parent's constructor method and gets access to the parent's properties and methods.

```
Syntax : class Parent {  
    Constructor() {  
        .....  
    }  
    .....  
}  
class child extends parent {  
    constructor () {  
        super();  
        .....  
    }  
}
```

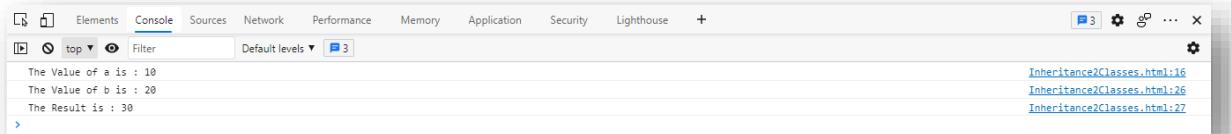
**Note:**

**Super()** keyword should be declared in constructor in the first line itself in the child class.

**Ex:**



```
10 <script type="text/javascript">  
11     class Parent {  
12         constructor(obj1) {  
13             this.a = 10;  
14         }  
15         displayParent () {  
16             console.log("The Value of a is : " + this.a);  
17         }  
18     }  
19     class Child extends Parent {  
20         constructor(obj2){  
21             super();  
22             this.b = 20;  
23             this.result = this.a + this.b;  
24         }  
25         displayChild() {  
26             console.log("The Value of b is : " + this.b);  
27             console.log("The Result is : " + this.result);  
28         }  
29     }  
30     var obj1 = new Parent(obj1);  
31     var obj2 = new Child(obj2);  
32     obj1.displayParent();  
33     obj2.displayChild();  
34     .....  
35     .....  
36 </script>
```



```
Elements Console Sources Network Performance Memory Application Security Lighthouse +  
top Filter Default levels ▾
```

The Value of a is : 10  
The Value of b is : 20  
The Result is : 30

Inheritance2Classes.html:16  
Inheritance2Classes.html:26  
Inheritance2Classes.html:27

➤ **Exception Handling:**

- **Error/Exception:**

A set of instructions or a line of instructions making the execution of application to stop abruptly.

The errors or exceptions might get raised while compiling the code or while executing the code.

- I. Runtime Exception.
- II. Compile Time Exception.

- **Compile Time Exception or Syntactical Exception:**

The errors which gets raised while we compile the code is called compile time errors.

**Ex:** Syntactical Errors.

JavaScript does not provide any way to handle compile time errors these errors are for sure need to be fixed to go to the further execution.

- **Run-Time Exceptions:**

The exceptions which gets raised while executing the code is called the runtime exceptions or the runtime errors.

**Ex:**

1. Array index out of bonds.
2. File not found exception.
3. Referring a DOM Element which does not exists.

In general when a runtime error gets raised the flow of execution gets stopped abruptly without going to the further execution.

JavaScript provide a feature of handling exception through which we could able to handle the exceptions being raised and continue the execution flow without stop.

Following are the predefined keyword through which we could able to handle Exceptions,

- i. Try
- ii. Catch
- iii. Finally
- iv. Throws.

Using **try, catch** statement we could able to handle exceptions get raised at the runtime.

➤ **Try Block:**

The set of instructions in which there is a chance of getting runtime errors has to be placed under the **TRY** block.

➤ **Catch Block:**

The set of instructions through which we could able to handle the exception has to be placed under the **Catch** block.

- **Catch** block is the immediate block should be placed after the **try** block.
- We cannot overload catch method in JavaScript, a single **try** block should have corresponding single catch method.
- We cannot invoke the **catch** method manually. It gets invoked automatically when there is an exception under the **try** block.
- Set of instructions through which we handle the error been raised at **try** block as to be placed under **catch** method.
- **Catch** method automatically gets involved holds an exception or error object with extra information about the current error been generated.

**Syntax:** **try {**

**:::/set of instructions in which there is a chance of getting runtime errors**

**} catch (error obj) {**

**....**

**// set of instructions to handle error been raised.**

**}**

## Ex:

```
10 <script type="text/JavaScript">
11     var add = () => {
12         var a = 10;
13         var b = 20;
14         try {
15             var c = a+b;
16             console.log("result is :" + c);
17             d = a + f + c;
18             console.log(d);
19         } catch (error) {
20             console.log(error);
21         }
22     }
23     add();
24 </script>
```

```
result is :30
ReferenceError: f is not defined
at add (Try_Catch.html:17)
at Try_Catch.html:23
```

## ➤ Finally Block:

An optional block can be placed after the catch block which gets executed for sure irrelevant of whether there is a exception being raised within the try block or not.

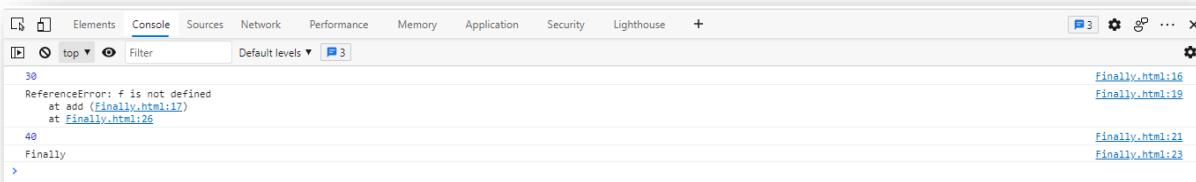
- The set of instructions which for sure need to be executed irrelevant of exception been raised or not with in the try block, has to be placed under the finally block.

**Syntax:** try {

```
..... // Set of instructions in which there is a chance of
..... // getting error.
} catch (error) {
    ..... // Code to handle error
} finally {
    ..... // set of instructions need to executed for sure,
    ..... // irrelevant of whether there is an exception
    ..... // raised or not.
};
```

**Ex:**

```
<script type="text/JavaScript">
    var add = () => {
        var a = 10;
        var b = 20;
        try {
            var c = a + b;
            console.log([c]);
            c = a+b+f+c;
        } catch (error) {
            console.log(error);
            b = a + c;
            console.log(b);
        } finally {
            console.log("Finally");
        }
    }
    add();
</script>
</body>
</html>
```

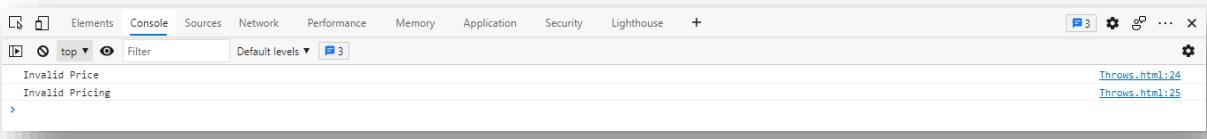


## ➤ Handling User Defined Exceptions:

JavaScript provides a feature of handling user-defined exceptions using “**throws**” keyword. Any time the controller reaches to the throw statement within the **try** block, it automatically treats that as an error and invokes the corresponding catch block.

**Ex:**

```
10     <script>
11         var pdprice = prompt("Enter the Value of a : ");
12         var tax = prompt("Enter the value of b : ");
13         pdprice = parseInt(a-pdprice);
14         tax = parseInt(tax);
15         var total = pdprice + tax;
16         try {
17             if (total > 20){
18                 console.log("valid price");
19             }
20             if (total < 20){
21                 throw "Invalid Price"
22             }
23         } catch(error) {
24             console.log(error);
25             console.log("Invalid Pricing")
26         }
27     </script>
```



## ➤ Closures:

Closure is a self-invoked functions gets invoked automatically once the controller reaches to it.

- Using closure, we could able to find set of JavaScript instructions as an individual module.
- The set of instructions within a closure module becomes a private data, and cannot be accessed outside of the closure.
- Within a closure we can able to find set of JavaScript instructions like set of variable methods, objects, classes etc...
- The data within a closure cannot be accessible outside of it even within the same page, because it adds accessibility security to that data.

**Syntax:** (function() {

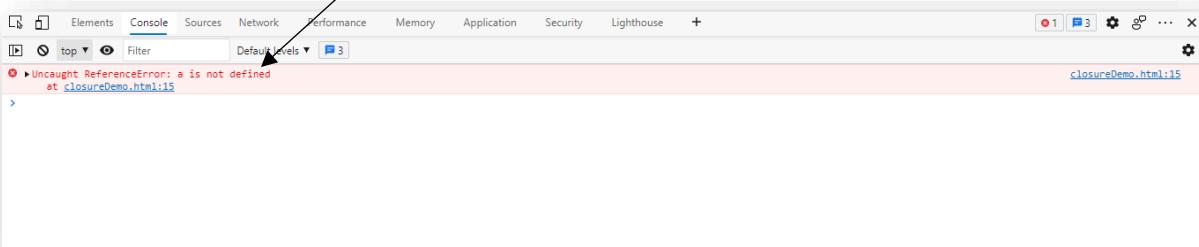
.....//set of instructions.

})();

**Ex:**

```
10 |     <script>
11 |     ( ()=> {
12 |         var a = 10;
13 |         var b = 20;
14 |     })();
15 |     var c = a + b;
16 |     console.log(c);
17 |   </script>
```

This is the closure block and we cannot use the variables/methods that are declared in it



➤ **Accessing data in closure outside of it:**

Steps to be followed to access the closure data,

- Create and assign a variable to the closure.
- Return the data which is needed to access outside of the closure as like an object.
- Using closure name, we could able to access closure returned object data outside of the closure.

**Note:**

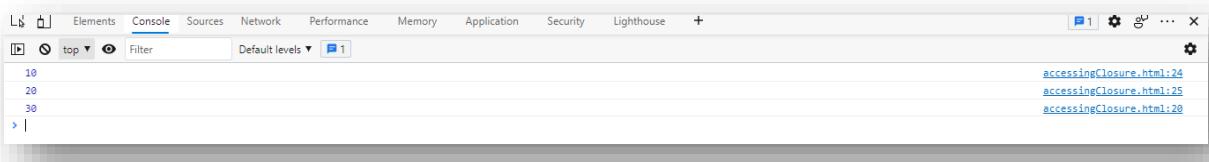
Not all the data within the closure can be accessible outside of it, only the data which is returned from the closure can be accessible through closure name.

**Syntax:** var closureName = ( () => {  
.....// set of instructions  
return {  
....// data to be accessed outside  
})();

**Ex:**

```
10 <script>
11     var sample = () => {
12         var a = 10;
13         var b = 20;
14
15         return {
16             firstval : a,
17             secondval : b,
18             result : () => [
19                 var result = a + b;
20                 console.log(result);
21             ]
22         }
23     })();
24     console.log(sample.firstval);
25     console.log(sample.secondval);
26     sample.result();
27 </script>
```

By using this we are accessing the data within the closure.



# **\*JQUERY\***

---

## **\*TONEKA\***

➤ **Introduction to jQuery:**

A predefined JavaScript library developed by Mr. John Resig in the Year 2006, comes with predefined method through which the DOM Operations performed within the page can be simplified.

- It supports predefined methods through which we can add, remove (or) update any CSS Property of any HTML Element dynamically.
- It provides a set of predefined methods through which we can add, remove or update any html element dynamically.
- It provides a set of predefined methods through which we can handle dynamic events generated on page.
- It provides set of predefined methods and objects through which ajax communication within the page implemented easily.
- It supports methods of implement animations to the DOM element.
- It provides set of predefined methods through which we could able to refer existing elements on the page in different ways.
- It provides different set of selectors and travelling methods.
- 3.6.0 is the current and latest version of jQuery, <https://jquery.com> is the website where we can download, even provides API documentation of jQuery library.
- jQuery library is of two types
  - I. Compressed version
  - II. Uncompressed version

**Links for jQuery library**

- I. <https://code.jquery.com/jquery-3.6.0.js> //uncompressed
- II. <https://code.jquery.com/jquery-3.6.0.min.js> //compressed.

**Ex:**

```

8   <style>
9     .borders {
10       border: 2px solid;
11       border-radius: 10px;
12       padding: 10px;
13       margin: 10px;
14       width: 50px;
15     }
16     .changes {
17       color: #azure;
18       background-color: #black;
19       font-weight: bold;
20     }
21   </style>
22   <script src="../jQueryCompressed.js"></script>
23 </head>
24 <body>
25   <div class="borders" id="test">Hello jQuery</div>
26   <input type="button" value="Change" onclick="change()">
27   <script>
28     var change = () => {
29       $("#test").addClass("changes");
30     }
31   </script>

```



## ➤ Minified and non-minified code:

While loading the content on the browser, it is recommended that to have less number of files so that browser takes less time to load them which in turn increases performance of the page.

- It is recommended every JavaScript file of a webpage has to go under minification process before it gets deployed on production environment.
- Minification is the process of reducing the total JavaScript code been written.

## Following are the steps to be followed in minifying code:

- It removes all the extra spaces within the code.
- Replaces all the variable names with smaller ones.
- Optimises the logic as much as possible.
- Removes unnecessary commands and descriptions within the file.

## ➤ Steps the minify the JS code:

- Select the JavaScript file which you want to minify.
- Visit <https://javascript-minifier.com>
- Select the code and paste and press minify,

The screenshot shows a web-based minifier tool. On the left, under 'Input JavaScript', there is a multi-line code snippet. On the right, under 'Minified Output', the same code is shown in a highly compressed form. Below the code blocks are several buttons: 'Minify' (in blue), 'Download as File', 'RAW', and 'Clear'. At the bottom right are 'Copy to Clipboard' and 'Select All' buttons.

```

Input JavaScript
document.querySelector(".age").style.display = 'none';
return false;
}
}
var validateSal = (event) => {
if ((event.charCodeAt >= 48 && event.charCodeAt <= 57)) {
document.querySelector(".usa1").style.display = 'none';
document.querySelector(".sal").style.display = 'block';
return true;
} else {
document.querySelector(".usa1").style.display = 'block';
document.querySelector(".sal").style.display = 'none';
return false;
}
}

Minified Output
!").innerHTML=this.tsal;var
validatename=e==e.charCodeAt>=97&&e.charCodeAt<=122||e.charCodeAt>=65&&e.charCodeAt<
=90||32==e.charCodeAt?
(document.querySelector(".uname").style.display="none",document.querySelector(".na
me").style.display="block",!0);
(document.querySelector(".uname").style.display="block",document.querySelector(".na
me").style.display="none",!1);validateage=e==e.charCodeAt>=48&&e.charCodeAt<=57?
(document.querySelector(".age").style.display="none",document.querySelector(".age"
).style.display="block",!0);
(document.querySelector(".age").style.display="block",document.querySelector(".age"
).style.display="none",!1);validatesal=e==e.charCodeAt>=48&&e.charCodeAt<=57?
(document.querySelector(".usa1").style.display="none",document.querySelector(".sa
l").style.display="block",!0);
(document.querySelector(".usa1").style.display="block",document.querySelector(".sa
l").style.display="none",!1);

```

- This is the minified code,

The screenshot shows a browser's developer tools console window. It contains a massive amount of minified JavaScript code, which is mostly illegible due to its compact nature. The code appears to be a complex function, likely related to generating a payslip, given the context of the surrounding text.

```

1 var employeeDetails={};function generatePaySlip(){employeeDetails.getEmployeeDetails();}employeeDetails.
getEmployeeDetails=function(){this.ename=document.querySelector("#ename").value,this.eage=document.
querySelector("#eage").value,this.egender=document.querySelector("input[name=gender]:checked").value,this.
edept=document.querySelector("#edept").value,this.basicsal=document.querySelector("#basicsal").value,this.
basicsal=parseInt(employeeDetails.basicsal),this.getTotalSal(),employeeDetails.getTotalSal=();
{employeeDetails.hra=.4*employeeDetails.basicsal,employeeDetails.pf=.12*employeeDetails.basicsal,
employeeDetails.totalsal=employeeDetails.hra+employeeDetails.basicsal-employeeDetails.pf,employeeDetails.
getTax=();=>{"male"==employeeDetails.egender&&(employeeDetails.totalsal>4e5?employeeDetails.etax=.
15*employeeDetails.totalsal:employeeDetails.totalsal>2e5?employeeDetails.etax=.1*employeeDetails.
totalsal:employeeDetails.totalsal>1e5?employeeDetails.etax=.05*employeeDetails.totalsal:employeeDetails.
etax=0*employeeDetails.totalsal),"Female"==employeeDetails.egender&&(employeeDetails.totalsal>4e5?
employeeDetails.etax=.1*employeeDetails.totalsal:employeeDetails.totalsal>2e5?employeeDetails.etax=.
05*employeeDetails.totalsal:employeeDetails.totalsal>1e5?employeeDetails.etax=.02*employeeDetails.
totalsal:employeeDetails.etax=0*employeeDetails.totalsal)),employeeDetails.getTax(),employeeDetails.
tsal=employeeDetails.totalsal-employeeDetails.etax,employeeDetails.displayTotalsal()),employeeDetails.
displayTotalsal=function(){document.querySelector("#payslip").style.display="block",document.querySelector(
"#rname").innerHTML=this.ename,document.querySelector("#rage").innerHTML=this.eage,document.querySelector(
"#rgender").innerHTML=this.egender,document.querySelector("#rdept").innerHTML=this.edept,document.
querySelector("#rbasic").innerHTML=this.basicsal,document.querySelector("#rhra").innerHTML=this.hra,document.
querySelector("#rpfp").innerHTML=this.pf,document.querySelector("#rtotalsal").innerHTML=this.totalsal,document.
querySelector("#rtax").innerHTML=this.etax,document.querySelector("#tsal").innerHTML=this.tsal};var
validatename=e==e.charCodeAt>=97&&e.charCodeAt<=122||e.charCodeAt>=65&&e.charCodeAt<=90||32==e.charCodeAt?
(document.
querySelector(".uname").style.display="none",document.querySelector(".name").style.display="block",!0);
(document.querySelector(".uname").style.display="block",document.querySelector(".name").style.display="none",
!1);validateage=e==e.charCodeAt>=48&&e.charCodeAt<=57?(document.querySelector(".age").style.display="none",
document.querySelector(".age").style.display="block",!0):(document.querySelector(".age").style.
display="block",document.querySelector(".age").style.display="none",!1);validatesal=e==e.charCodeAt>=48&&e.
charCodeAt<=57?(document.querySelector(".usa1").style.display="none",document.querySelector(".sa
l").style.
display="block",!0):(document.querySelector(".usa1").style.display="block",document.querySelector(".sa
l").style.
display="none",!1);

```

## ➤ jQuery selectors and methods:

`$(*)` - To get any element with the dom

`$("#abc")` - Returns the reference of element with id abc

`$(".abc")` - Returns the reference of element with class abc

`$("div.abc")` -> Returns the reference of all div tags having the class as abc

`$("li:first")` -> Returns the reference of li tag which is in first child position

`$("input[type=text]")` -> Returns reference of all input elements having type as text

`$("a[target!=\"_blank\"]")` -> Returns reference of all "a" tags which are not having target as "\_blank"

`$("tr:even")` -> Returns reference of all "tr" tags which are in even position

`$("div:odd")` -> Returns reference of all "div" tags which are in odd position

`$("div > span")` -> Returns reference of all span tags which all are direct childrens of div tag

`$("li:eq(2)")` -> Returns reference of li tag which is in second position

`$("span:first-of-type")` -> selects all elements that are the first among siblings of the same element name

`$("div:gt(3)")` -> selects elements at an index greater than index within the matched set

`$("div:has(p)")` -> Returns div tag which has atleast one p tag

`$("div:hidden")` -> Returns reference of div tag which is in hidden state

`$(“span:last-of-type”)` -> Returns span which is last sibling

`$(“div, .abc, span, #pqr”)` -> Returns multiple selectors

`$(“input:disabled”)` -> Returns input element with disabled state

`$(“input:checked”)` -> Returns check box element which is in checked state.

etc..

➤ **Jquery Predefined methods:**

Following are list of predefined methods been supported in jquery can be applied on DOM elements.

`.addClass()` - to add single or multiple classes to an element

`.removeClass("classname")` - To remove a classname

`.hasClass("classname")` - Returns true /false based on, if element is having the provided class or not.

`.after()` -> To add an element as sibling after the existing element

eg:

```
$(“div”).after(“<p>test</p>”);
```

`.before()` -> To add an element before the existing element

eg:

```
$(“div”).before(“<p>test</p>”);
```

`.append()` -> to append an element to existing element

.attr() - To set or get any attribute of an element

.css() -> To set single or multiple css properties to elements

eg:

```
$("#abc").css("color", "green");  
$(".container").css({  
    color: 'blue',  
    'font-size': '20px'  
})
```

.val() -> To get or set value from input elements

.removeAttr() -> To remove and existing attribute from element

.toggleClass() -> to toggle class from existing element

.html("html text") -> adds required content as html content with in an element

.text("text content") -> adds required content as text content with in an element

.show() -> to show an hidden element

.hide() -> To hide an visible element

etc.

#### ➤ **\$( document ).ready():**

A page can't be manipulated safely until the document is "ready." jQuery detects this state of readiness for you. Code included inside \$( document ).ready() will only run once the page Document Object Model (DOM) is ready for JavaScript code to execute. Code included inside \$(

window ).on( "load", function() { ... }) will run once the entire page (images or iframes), not just the DOM, is ready.

**Ex:** \$( document ).ready(function() {  
 console.log( "ready!" );  
});

**Ex2:**

```
Line wrap []
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Demonstrating Jquery Methods</title>
8     <script src="../jquery.js"></script>
9     <script src="readyAndLoad.js"></script>
10    </head>
11    <body>
12      <div id="container">
13        Click here
14      </div>
15    </body>
16  </html>
```

## JS Code:

```
console.log("heelo");
$(document).ready(function(){
  document.querySelector("#container").addEventListener('click', () => {
    console.log("user clicked");
  });
});
```

## ➤ jQuery traversing Methods:

following are the predefined methods through which we could traverse between DOM Elements,

- **parent():** Returns the immediate parent of any DOM Element.

```
9   <style type="text/css">
10    .text {
11      border: 2px solid;
12      color: #aliceblue;
13      background-color: #black;
14      list-style: none;
15    }
16  </style>
17 </head>
18 <body>
19   <ul>
20     <li>one</li>
21     <li>two</li>
22     <li><button>three</button></li>
23     <li>four</li>
24     <li>five</li>
25   </ul>
26   <script>
27     $("li").on("click", () => [
28       $(event.target).parent().addClass("text");
29       $(event.target).css("border-radius", "15px");
30     ])
31   </script>
```

- **parents():** Returns all parents until HTML tag.
- **parentsUntil("pattern"):** Returns all the parents until it finds a parent with given pattern.

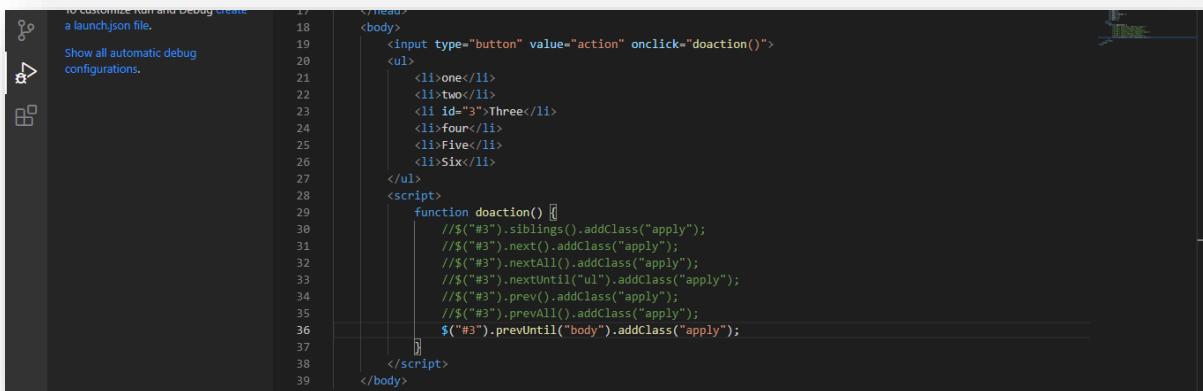
```

26     <script>
27         $("li").on("click", () => {
28             // $(event.target).parent().addClass("text");
29             // $(event.target).parents().addClass("text");
30             $(event.target).parentsUntil("body").addClass("text")
31             $(event.target).css("border-radius", "15px");
32         })
33     </script>

```

### ➤ Traversing siblings:

- **siblings():** Returns the list of all siblings of any DOM Element.
- **next():** Returns the reference of next sibling element.
- **prev():** Returns the reference of Previous sibling element.
- **prevAll():** Returns the reference of all Previous sibling element.
- **nextAll():** Returns the reference of all Next sibling element.
- **nextUntil("Pattern"):** Return all the next siblings until the provided pattern.
- **previousUntil("Pattern"):** Return all the previous siblings until the provided pattern.



The screenshot shows a browser's developer tools console. The code is as follows:

```

<head>
<body>
    <input type="button" value="action" onclick="doaction()">
    <ul>
        <li>one</li>
        <li>two</li>
        <li id="3">Three</li>
        <li>four</li>
        <li>Five</li>
        <li>Six</li>
    </ul>
</body>
<script>
    function doaction() {
        //$("#3").siblings().addClass("apply");
        //$("#3").next().addClass("apply");
        //$("#3").nextAll().addClass("apply");
        //$("#3").nextUntil("ul").addClass("apply");
        //$("#3").prev().addClass("apply");
        //$("#3").prevAll().addClass("apply");
        $("#3").prevUntil("body").addClass("apply");
    }
</script>

```

### ➤ Traversing Children:

- **children():** Return the children of any DOM Element.
- **find(\*):** Return the every children of a DOM Element.

### ➤ Adding Event Handling to DOM Element through jQuery Methods:

Following are the predefined methods provided by the jQuery which can be applied on DOM Elements to handle events.

element.click();

element.focus();

element.change();

element.blur();

```
element.mouseover();
```

**Ex: \$("abc").click(callingMethod);**

- “on” is a method been provided by jQuery through which we can add event handling dynamically to DOM Elements.

**Syntax:** `$("abc").on("<typeofevent>", <callbackfunction()> {  
..... // lines to be executed;  
});`

- Using “on” we could able to add multiple events at a time to single DOM Element.

**Syntax:** `$("abc").on("<typeofevent>,"  
<callbackfunction()> {  
..... // lines to be executed;  
},  
<callbackfunction2()> {  
..... // lines to be executed;  
});`

```
26 |     <script>  
27 |         $("li").on("click", () => {  
28 |             // $(event.target).parent().addClass("text");  
29 |             // $(event.target).parents().addClass("text");  
30 |             $(event.target).parentsUntil("[body]").addClass("text")  
31 |             $(event.target).css("border-radius","15px");  
32 |         })  
33 |     </script>
```

## ➤ jQuery Animation:

Adding animations to DOM Elements using jQuery methods following are the different ways we could add animation to DOM Elements.

1. Sliding
2. Fade
3. Hide/Show/Toggle

- **Hide/Show/Toggle:**

```
<div id="demo">Hello World</div>  
    <input type="button" id="btn" value="show">  
    <input type="button" id="btn1" value="Hide">  
    <input type="button" id="Toggle" value="Toggle">  
<script type="text/JavaScript">  
    $("#btn").on("click",() => {  
        $("#demo").addClass("container").show();  
    });  
    $("#btn1").on("click",() => {  
        $("#demo").hide();  
    });  
    $("#Toggle").on("click",() => {  
        $("#demo").addClass("container").toggle();  
    });
```

- **Fade:**

```
<input type="button" value="Fade in" onclick="doAction('in')">
<input type="button" value="Fade out" onclick="doAction('out')">
<input type="button" value="Toggle" onclick="doAction('tog')">
<div class="container">
    CONTENT
</div>
<script>
    var doAction = (type) => {
        switch (type) {
            case 'in':
                $(".container").fadeIn(8000);
                break;
            case 'out':
                $(".container").fadeOut(8000);
                break;
            case 'tog':
                $(".container").fadeToggle(4000);
                break;
        }
    }
</script>
```

- **Sliding:**

```
<script>
    var doAction = (type) => {
        switch (type) {
            case 'up':
                $(".container").slideUp(8000);
                break;
            case 'down':
                $(".container").slideDown(8000);
                break;
            case 'tog':
                $(".container").slideToggle(4000);
                break;
        }
    }
</script>
```

- **Master Animation Method:**

Animate method is supported in jQuery, which can take multiple CSS Properties and add to DOM Elements with provided delay time.

## Syntax:

```
element.animate({  
    cssprop1 : value;  
    cssprop2 : value;  
    ....  
    ....  
},delaytime);
```

## Ex:

```
<script>  
    var doAction = (type) => {  
        //$(".container").css("left", "300px");  
        $(".container").animate({  
            left: '800px',  
            height: '200px',  
            width: '200px'  
        }, 3000, function() {  
            $(".container").animate({left: '10px'}, 3000);  
        });  
    }  
</script>
```

# **\*AJAX\***

---

➤ **Web Service:**

Set of instructions gets executed at the server side which are capable of considering a request from client side, process the request accordingly and response back is called webservice.

- In a single server, it can run any number of web Services.
- Every Web Service within the server gets identified with a unique **URL**.
- All the Web Services within the server runs independent to each other.

**Ex: Sample webservices URL's**

[https://www.amazon.in/ref=nav\\_logo](https://www.amazon.in/ref=nav_logo)

➤ **AJAX:**

The process of creating a request from client side to a server through a webservice **URL** in order to indirectly process operations on database is called **AJAX**.

➤ **Synchronous and Asynchronous Communication:**

While interacting to the server from the client, at a time we could able to create any number of requests, while communicating to the server, the communication type could be synchronous or asynchronous communication.

➤ **Synchronous Communication:**

In this type, once the request been sent to the server, from client side it doesn't execute further instructions until there is a response from the server.

➤ **Asynchronous Communication:**

In this type, once the request been sent to the server, client doesn't wait for its response and it continues executing further instructions. Call back function holds the set of instructions and executes automatically once there is response from the server.

➤ **Specifying the data type while Communicating:**

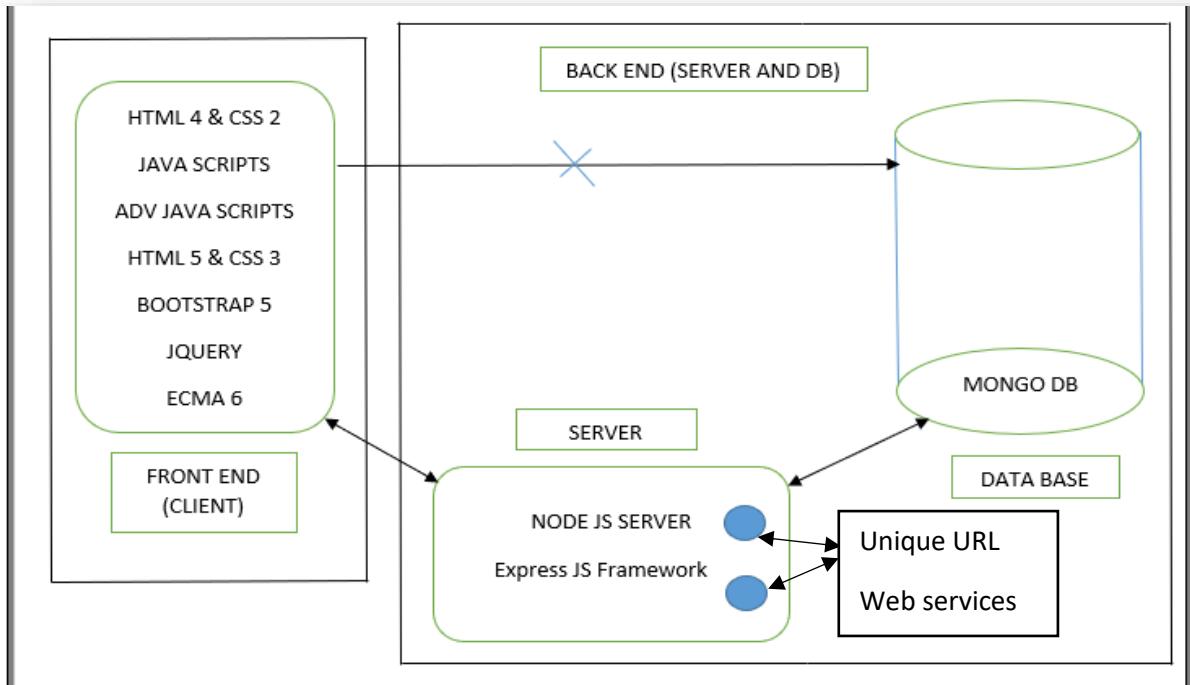
While interacting to the server, through a webserver, we need to specify the, type of the data format in which communication happens.

Following are the different data types in which communication can be done,

-Json      -String      -XML      -RSS etc...

Among all the datatypes **Json** is mostly used and recommended **Data Type**.

➤ **AJAX:**



➤ **Creating a AJAX call through jQuery AJAX Method:**

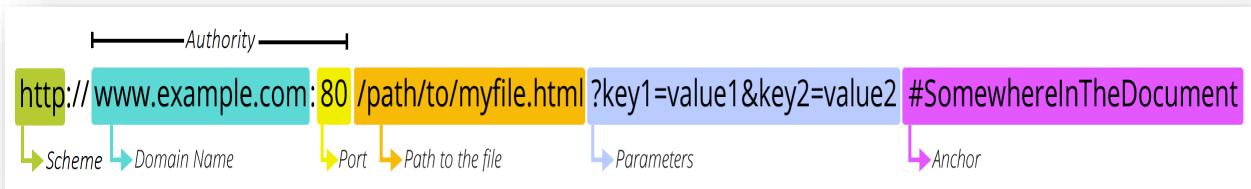
“`$.ajax()`” is a predefined method supported in jQuery using which we could able to create an AJAX call to a specific web service from client side.

It takes an object as a parameter with the following Properties,

\* **URL:**

**URL** stands for **Uniform Resource Locator** through which we can specify the data, type of data through which communication happens (XML/JSON/STRING etc...). A URL is nothing more than the address of a given unique resource on the Web. In theory, each valid URL points to a unique resource. Such resources can be an HTML page, a CSS document, an image, etc. In practice, there are some exceptions, the most common being a URL pointing to a resource that no longer exists or that has moved. As the resource represented by the URL and the URL itself are handled by the Web server, it is up to the owner of the web server to carefully manage that resource and its associated URL.

**Ex:** `https://developer.mozilla.org/en-US/search?q=URL`



#### \* Method:

Through which we need to specify the type of communication Secured/Non-Secured (Get/Post).

#### \* async:

Takes a Boolean value, through which we can specify whether the call is synchronous or asynchronous. By default, it is asynchronous.

#### \* Success():

Takes a call back method as input, and invokes the call back method automatically when there is a response from the server.

#### \* Error():

Takes a call back method as an input, and invokes the call back method automatically when there is an error while interacting with the server.

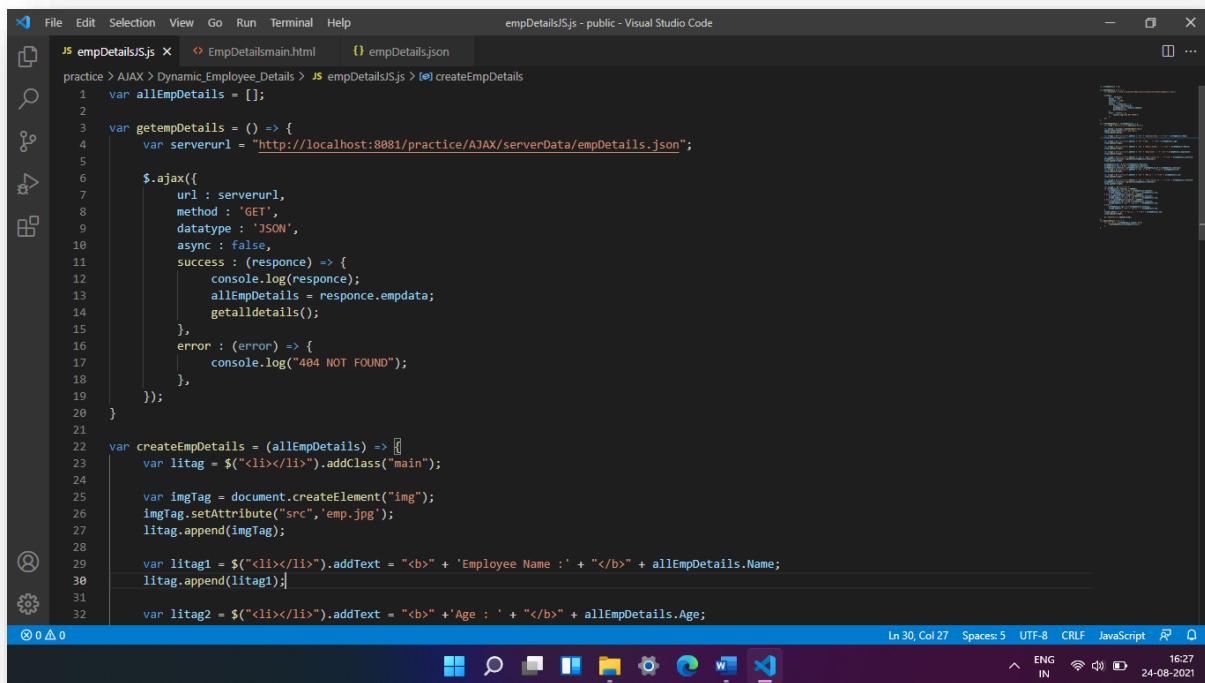
### Syntax:

```

$.ajax({
    url : 'webserviceurl',
    method : 'Get/Post',
    datatype : 'Json/xml/string',
    success : function (<response>) {
        .....
        ..... // code to handle response
    },
    error : function (error) {
        ....// code to handle error
    })
});
```

**Ex:**

## JS Program:



```
File Edit Selection View Go Run Terminal Help empDetailsJS.js - public - Visual Studio Code
practice > AJAX > Dynamic_Employee_Details > JS empDetailsJS.js > [e] createEmpDetails
1 var allEmpDetails = [];
2
3 var getempDetails = () => {
4     var serverurl = "http://localhost:8081/practice/AJAX/serverData/empDetails.json";
5
6     $.ajax({
7         url : serverurl,
8         method : 'GET',
9         datatype : 'JSON',
10        async : false,
11        success : (response) => {
12            console.log(response);
13            allEmpDetails = response.empdata;
14            getAllDetails();
15        },
16        error : (error) => {
17            console.log("404 NOT FOUND");
18        },
19    });
20 }
21
22 var createEmpDetails = (allEmpDetails) => [
23     var litag = $("<li></li>").addClass("main");
24
25     var imgTag = document.createElement("img");
26     imgTag.setAttribute("src","emp.jpg");
27     litag.append(imgTag);
28
29     var litag1 = $("<li></li>").addText = "<b>" + 'Employee Name : ' + "</b>" + allEmpDetails.Name;
30     litag.append(litag1);
31
32     var litag2 = $("<li></li>").addText = "<b>" + 'Age : ' + "</b>" + allEmpDetails.Age;
33
34 ];
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
218
219
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162

```

## HTML-5

- **LocalStorage and Session Storage:**

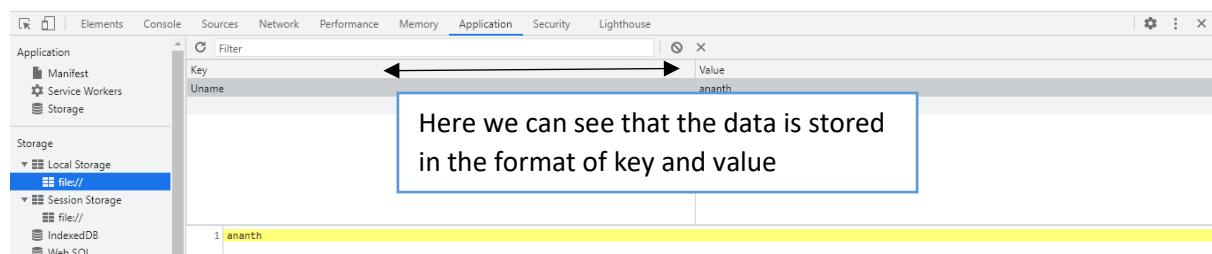
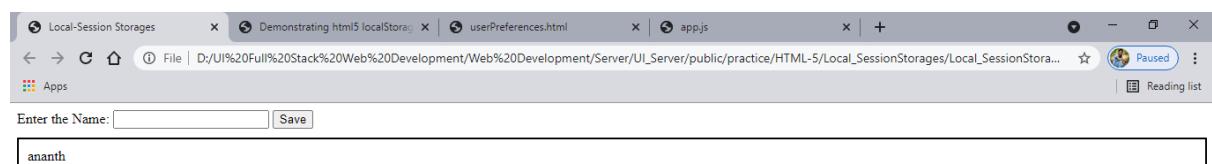
The two predefined objects supported in a **HTML-5** based browsers using which we could able to store user preferences within the browser cache itself.

Following are the predefined methods being supported or can be applied on both the Local Storage and Session Storage object through which we could able to add, remove or update data.

- **setItem("Key",<value>):** To set a value in a cache.
- **getItem("Key"):** Returns values been stored on a key.
- **removeItem("Key"):** Removes a key value.
- **removeAll():** Removes all values inside a object.

**Ex:**

```
var getUsername = () => {
    var name = $( "#uname" ).val();
    localStorage.setItem( "Uname", name );
    displayuname( "Uname" );
}
var displayuname = ( uname ) => {
    $(".displayUserName").text( uname ).show( 3000 );
}
if ( localStorage.getItem( "Uname" ) != null ) {
    displayuname( localStorage.getItem( "Uname" ) );
}
```



**The same is applicable for session storage but the only difference is**

- These two objects are almost similar used to store the user preferences within the browser cache which can be assessable even after the reload or even the reopen of the browser and the only difference between these two objects is data stored under local storage object will be available even after reloading or the reopen whereas data stored under the session object will be only available on the reload of the page, and will be flushed out when we close the page.

**Note:** These two objects have same set of predefined methods can be applied on them.

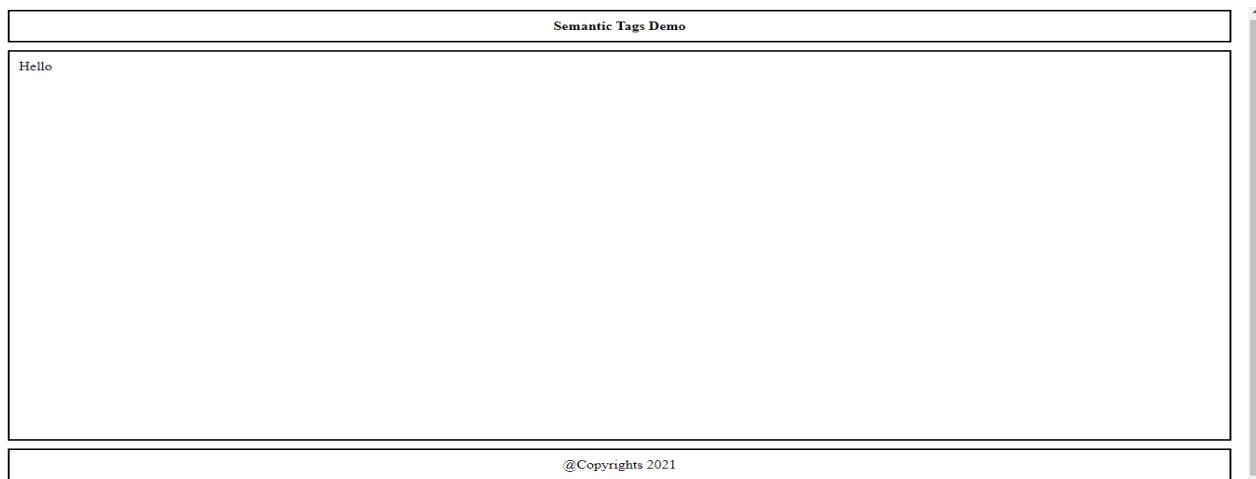
- **Semantic tags of HTML-5:**

Following are the predefined semantic tags been supported in the HTML-5. The name of the semantic tags describes the purpose of the element and type of the content that is within the tag.

-article	-aside	-header	-footer	-maincontainer
-details	-figure	-main	-section	-navbar      -nav
-summary	-title	etc...		

**Ex:**

```
<body>
    <header class="head"><b>Semantic Tags Demo</b></header>
    <main class="inner">
        <selection>
            <article>Hello</article>
        </selection>
    </main>
    <footer class="foot"> @Copyrights 2021 </footer>
</body>
```



- **HTML-5 Input Elements:**

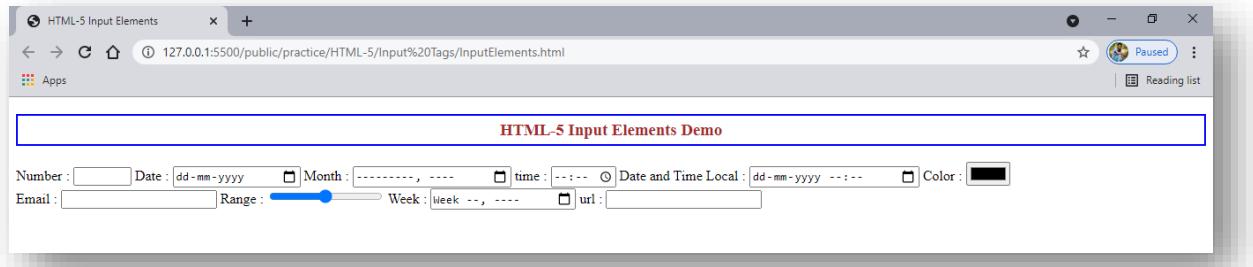
Following are the set of input elements been supported in **HTML-5** to read the different types of data from the user,

1. Input type = “color”
2. Input type = “date”
3. Input type = “datetime-local”
4. Input type = “email”
5. Input type = “search”
6. Input type = “month”
7. Input type = “number”
8. Input type = “range”
9. Input type = “tel”
10. Input type = “url”
11. Input type = “week”
12. Input type = “time” etc...

**Ex:**



```
<body>
  <h3>HTML-5 Input Elements Demo</h3>
  Number : <input type="number" name="Number" id="num" max="10" min="0">
  Date : <input type="date" name="date" id="dt">
  Month : <input type="month" name="month" id="mth">
  time : <input type="time" name="time" id="tm">
  Date and Time Local : <input type="datetime-local" name="dt" id="dtime">
  Color : <input type="color" name="color" id="clr"><br>
  Email : <input type="email" name="email" id="mail">
  Range : <input type="range" name="rn" id="rn" max="100" min="10">
  Week : <input type="week" name="week" id="week">
  url : <input type="url" name="ur" id="ur">
</body>
```



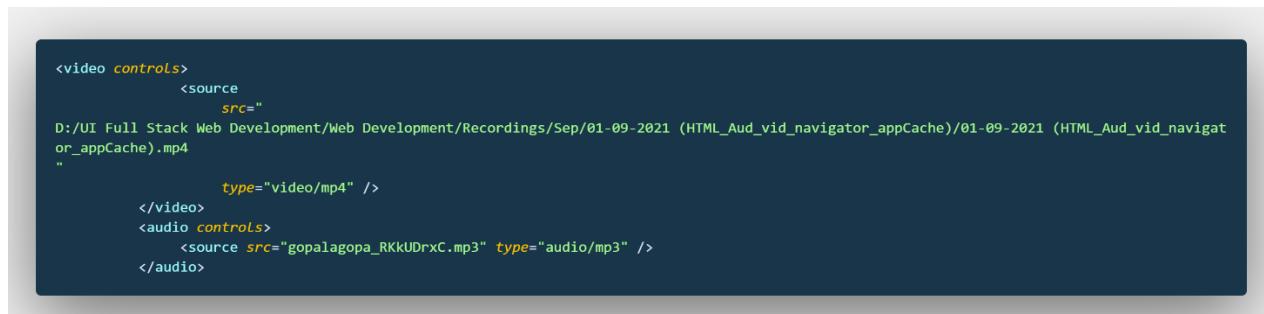
- **HTML-5 audio and video tags:**

HTML-5 provides a feature of adding audio and video without any dependences on the third-party plugins.

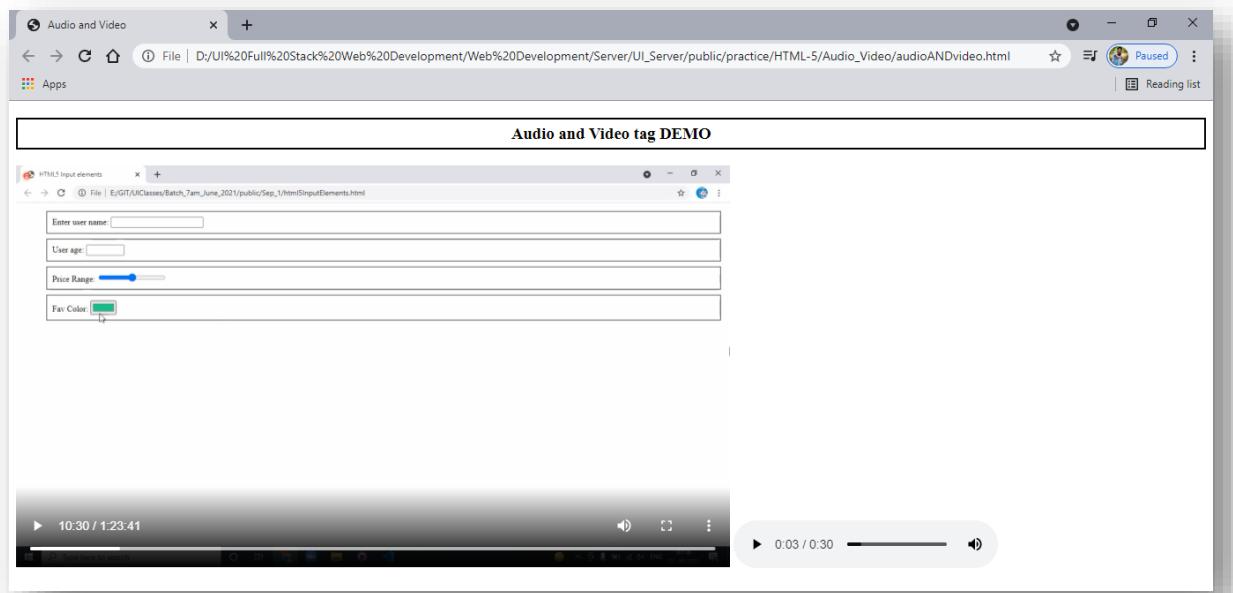
**Syntax: <audio autoplay controls>**

```
<source src ="music/data/song.mp3" type = "audio/mp3">  
</audio>  
  
<video autoplay controls>  
  <source src ="video/data/vd.mp4" type = "video/mp4">  
</audio>
```

**Ex:**



```
<video controls>  
  <source  
    src=  
      D:/UI Full Stack Web Development/Web Development/Recordings/Sep/01-09-2021 (HTML_Aud_vid_navigator_appCache)/01-09-2021 (HTML_Aud_vid_navigat  
      or_appCache).mp4  
      type="video/mp4" />  
</video>  
<audio controls>  
  <source src="gopalagopa_RKkUDrxC.mp3" type="audio/mp3" />  
</audio>
```



- **Attribute that can be added to video and audio tags:**

- **Controls:** Specifies that whether audio/video controls like play, pause etc... to show on the page.
- **autoplay:** It says that the audio or video to be played on the load of the page itself.
- **Height/width:** To set the height and width of the player.

- **Loop:** Boolean value specifies to keep playing audio or video file once it finishes.
- **Muted:** Specified audio/video output to be muted.
- **Poster:** Takes an image URL as input and shows the image before the video gets plays just like as thumbnail.
- **Preload:** Automatically buffers the video/audio file.
- **Application Cache:**

Its a new feature being supported in HTML-5 through which we could make the webpage resources to be accessible even when it is in offline.

Following are the steps to be followed to implement application cache to a webpage,

**Step 1:** Create an external app cache file (.appcache extension is recommended)

**Step 2:** Define set of rules in the appcache files which specifies which resources to be available offline, for which resources the network connection is mandatory.

**Step 3:** Within the HTML tag through manifest attribute specify the app cache file to be used to the current file.

```
<html lang="en" manifest="..../sample.appcache">
```

- **Creating an appcache file:**

Any appcache file contains the following three blocks,

- Cache management/ manifest
- Network
- Fallback

#### \* **Cache Management:**

Under this block we specify all the resource (HTML, CSS, JS, images...) which need to be accessed even when there is network connection.

#### \* **Network:**

Under this, we specify the list of resources which should never be cached, should have network connection to access these resources (Ex: LoginPage).

## \* **Fallback:**

Under this, we specify a custom user defined page not found file which will be automatically thrown when the user tries to access the resources which need an internet connection.

```
CACHE MANIFEST
# 2010-06-18:v3

# Explicitly cached entries
index.html
css/style.css

# offline.html will be displayed if the user is offline
FALLBACK:
/ /offline.html

# All other resources (e.g. sites) require the user to be online.
NETWORK:
*

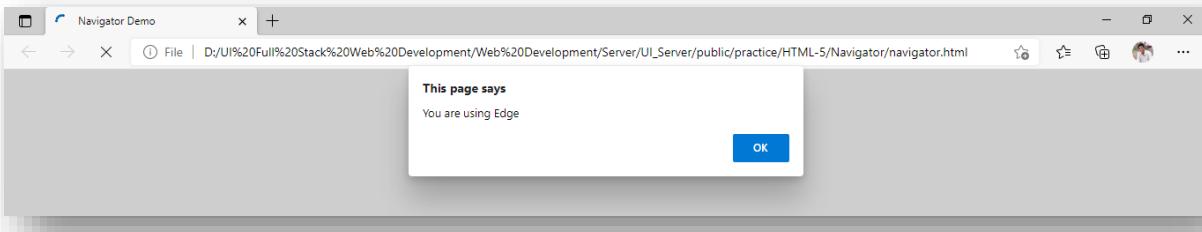
# Additional resources to cache
CACHE:
images/logo1.png
images/logo2.png
images/logo3.png
```

## • **Navigator object:**

A predefined object by default available in HTML-5 based page which holds extra information of the current browser and operating system of the current machine. It consists of,

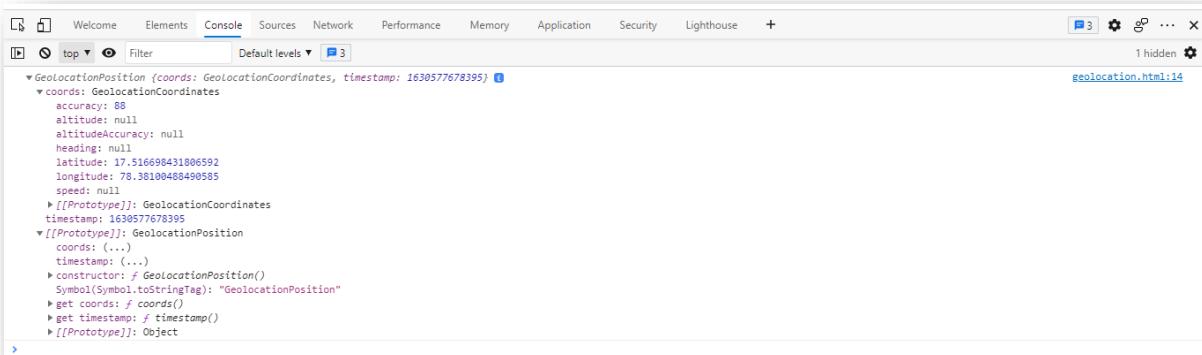
- List of languages that the browser is supporting.
- Whether the system is connected to the internet or not.
- Appcode and appname of the current browsers.
- Vendor name and version number of the current browser.
- List of plugins been installed by the browser.
- List of media devices been connected.
- Bluetooth information and geolocation information.

```
<script>
    console.log( navigator );
    if ( navigator.userAgent.includes( 'Edg' ) ) {
        alert( "You are using Edge" );
    } else if ( navigator.userAgent.includes( 'Firefox' ) ) {
        alert( "You are using firefox" );
    } else {
        alert( "You are using chrome" );
    }
</script>
```



- **Navigator Geolocation:**

```
<body>
    <script>
        navigator.geolocation.getCurrentPosition( ( point ) => {
            console.log( point );
        } );
    </script>
</body>
```



- **HTML5 Canvas Tag:**

A predefined tag been supported in HTML5 using which we could able to draw the graphical objects within the canvas container.

It supports set of predefined JavaScript methods through which we could draw the objects within the canvas container.

Following are the steps to be followed to draw the graphical objects within the canvas container,

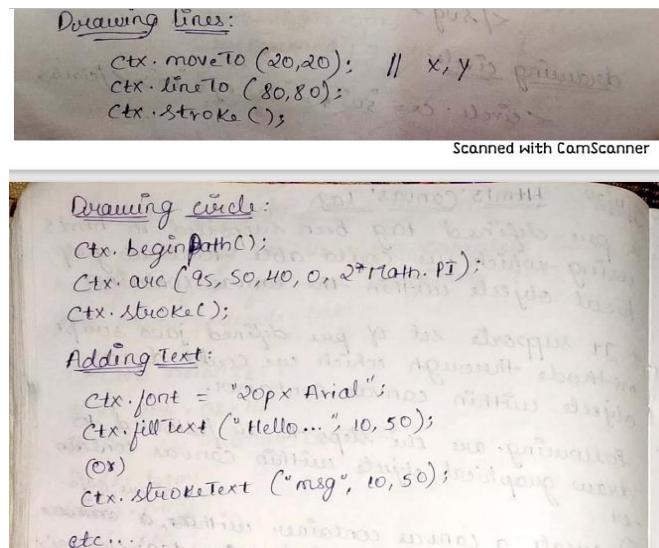
- Create a canvas tag with an ID Ex: <canvas id="usc"></canvas>
- Create a context object of canvas container.

**Ex:** var element = \$("#usc");

**//context object**

Var ctx = element.getContext("2d");

- Use the predefined methods on the context object through the objects can be drawn on the container.



- **SVG Tag:**

It is another way of creating graphical objects within the HTML5 Container, drawing graphical objects within the SVG is almost like a canvas tag, where the only difference is to draw objects within the canvas, we make use of JS methods where as in SVG we use predefined HTML Tags to draw objects.

Following are the predefined graphical objects can be used within the SVG container,

```
<body>
  <SVG>
    <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />
    <rect x="100" y="150" width="400" height="100"
      style="fill:rgb(0,0,255);stroke-width:10;stroke:rgb(0,0,0)" />
    <ellipse cx="200" cy="80" rx="100" ry="150" style="fill:yellow;stroke:purple;stroke-width:2" />
    <line x1="0" y1="0" x2="200" y2="200" style="stroke:rgb(255,0,0);stroke-width:2" />
    <text x="0" y="15" fill="red">Good morning all</text>
  </SVG>
</body>
```

- **HTML Drag and Drop Events:**

HTML5 supports handling new set of event types like drag and drop. We could able to invoke call back methods when an element is getting dragged or dropped.

- Draggable events are **dragstart**, **dragend** and attribute is **draggable = “true”**.
- Drop target events are **dragenter**, **dragover**, **dragleave**, **drop**.

```

<body>
    <div class="container" ondrop="elementDropped(event)" ondragover="dagIsOver(event)"></div>
    
    
    <img id="prod_3" ondragstart="draggingIsStarted(event)"
        src=
        https://www.cnet.com/a/img/xh7SVje9bq1_azlPW73EGrHHJVQ=/1200x630/2020/10/28/5a601f4a-06c8-4374-b10d-9f07fe183390/babyyoda-target-1.png"
        alt="">

    <script>
        var productId = '';
        var elementDropped = ( event ) => {
            event.preventDefault();
            productId = '#' + productId;
            event.target.append( document.querySelector( productId ) );
        }

        var dagIsOver = ( event ) => {
            event.preventDefault();
        }

        var draggingIsStarted = ( event ) => {
            console.log( "drag is started" );
            productId = event.target.getAttribute( "id" );
        }
    </script>
</body>

```

- **Multitasking and multithreading:**

The concept of process of executing multiple jobs at a time to increase the performance of the page is called as multitasking or multithreading.

**Note:** JavaScript does not support multitasking directly but we can make use of it indirectly.

- **HTML5 Web Worker:**

It is a new feature been supported in HTML5 through which we can indirectly achieve the support of multitasking and multithreading,

- Workers are a separate JS file gets initiated through the main thread, executes parallelly to the main thread.
- Even though the web worker gets executed independent through the main thread it can still pass or communicate to the main thread.
- Even the main thread is capable of receiving messages from a worker.
- In a single page we can use any number of web workers.
- Following are the steps to be followed to implement web workers in an application,

**Step-1:** Create an external JS file (web worker is a JS file) which executes independent to the main thread using the POST message method.

**Step-2:** Instantiate a web worker from the main thread through the predefined worker class.

**Syntax:** var worker = new Worker (“Worker JS File path”);

**Step-3:** Add the onmessage event handler on worker object which gets involved automatically when there is a response from corresponding web worker.

**Syntax:** worker.onMessage = function(event) {

....// call back method gets fixed automatically when there is a message from the web worker.

}

**Ex:**

```
var count = 1;
var doJob = () => {

    var myWorker = new Worker( 'dateWorker.js' );
    myWorker.onmessage = function ( event ) {
        document.querySelector( ".dateContainer span" ).innerHTML = event.data;
        // myWorker.terminate();
    }

    // Creating Container.
    setInterval( () => {
        var divTag = document.createElement( "div" );
        divTag.setAttribute( "class", 'content' )
        divTag.innerHTML = "Content Created " + count;
        count++;
        document.querySelector( ".main-container" ).append( divTag );
    }, 1000 );
}
```

```
var temp = 0
function getDateInFormat () {
    setInterval( function () {
        var date = new Date();
        var reqFormat = date.getMonth() + '/' + date.getDate() + '/' + date.getFullYear() + ' ' + date.getHours() + ':' + date.getMinutes()
        () + ':' + date.getSeconds();
        console.log( "Date format is " + reqFormat );
        temp++;
        //var data = {msg: 'one', data: 'two'};
        //postMessage(JSON.stringify(data));
        postMessage( reqFormat );
    }, 1000 );
}

getDateInFormat();
```

- **Border-Radius:**

The **border-radius** CSS property rounds the corners of an element's outer border edge. You can set a single radius to make circular corners, or two radii to make elliptical corners.

**Ex:** border-radius: 30px;

border-radius: 25% 10%;

border-radius: 10% 30% 50% 70%;

- The radius applies to the whole background, even if the element has no border; the exact position of the clipping is defined by the background-clip property.
- The border-radius property does not apply to table elements when border-collapse is collapse.

- **Constituent properties:**

This property is a shorthand for the following CSS properties:

- border-top-left-radius
- border-top-right-radius
- border-bottom-right-radius
- border-bottom-left-radius

The border-radius property is specified as:

- one, two, three, or four <length> or <percentage> values. This is used to set a single radius for the corners.
- followed optionally by "/" and one, two, three, or four <length> or <percentage> values. This is used to set an additional radius, so you can have elliptical corners.

### <length>:

Denotes the size of the circle radius, or the semi-major and semi-minor axes of the ellipse, using length values. Negative values are invalid.

### <percentage>:

Denotes the size of the circle radius, or the semi-major and semi-minor axes of the ellipse, using percentage values. Percentages for the

horizontal axis refer to the width of the box; percentages for the vertical axis refer to the height of the box. Negative values are invalid.

- **Multibackground Images:**

You can apply **multiple backgrounds** to elements. These are layered atop one another with the first background you provide on top and the last background listed in the back. Only the last background can include a background color.

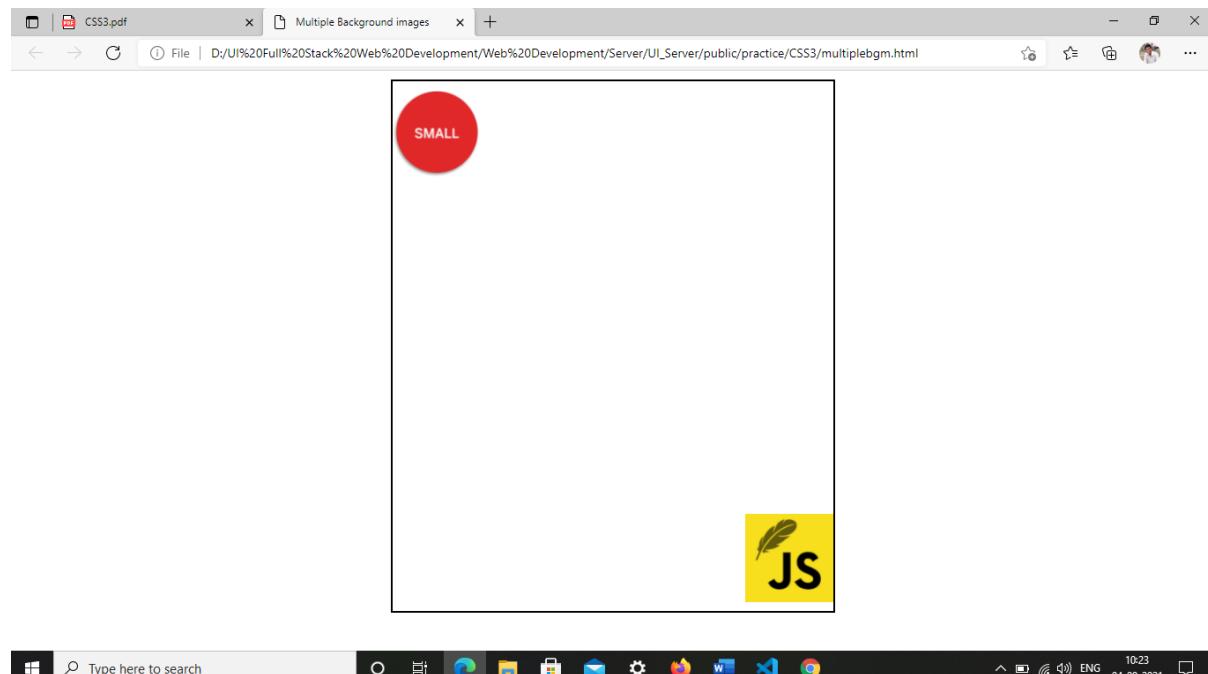
Specifying multiple backgrounds is easy:

**Background-image : url("abc.jpg"),url("abcd.png");**

**Background-position : left top, right bottom;**

**Ex:**

```
.containers {  
    border: 2px solid;  
    margin: 10px auto;  
    width: 500px;  
    height: 600px;  
    background-image: url("https://images.prismic.io/smallexchange-com/21260cc6-c545-4571-8f59-cd96052b327e_SMFE_Site_Buttons-SMALL@3x.png?auto=compress,format") ,  
    url("https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSpMHRgGhGu9aTtu0Te1mcG91JjK6v24lHbbw&usqp=CAU");  
    background-repeat: no-repeat;  
    background-size: 100px;  
    background-position: left top 10px, right bottom 10px;
```



- **CSS Box-Shadow:**

The box-shadow CSS property adds shadow effects around an element's frame. You can set multiple effects separated by commas. A box shadow is described by X and Y offsets relative to the element, blur and spread radius, and color.

**Ex:** box-shadow: 1px 2px 5px 5px red;

box-shadow: 12px 12px 2px 1px rgba(0, 0, 255, .2);

**Ex:**

```
/* Keyword values */
box-shadow: none;

/* offset-x | offset-y | color */
box-shadow: 60px -16px teal;

/* offset-x | offset-y | blur-radius | color */
box-shadow: 10px 5px 5px black;

/* offset-x | offset-y | blur-radius | spread-radius | color */
box-shadow: 2px 2px 2px 1px rgba(0, 0, 0, 0.2);

/* inset | offset-x | offset-y | color */
box-shadow: inset 5em 1em gold;

/* Any number of shadows, separated by commas */
box-shadow: 3px 3px red, -1em 0 0.4em olive;

/* Global keywords */
box-shadow: inherit;
box-shadow: initial;
box-shadow: revert;
box-shadow: unset;
```

- **CSS3 transitions and transformation:**

Transitions and transformations are the new set of CSS properties through which we could able to control the transformation of the DOM Element while rendering, adding the animation delay through transition.

- **CSS Transform property:**

Any DOM Element can be transformed in following three different ways,

- Rotate      - translate      - scale      - skew

- **Transform with Rotate property:**

Through which any DOM Element can be rotated within  $0^\circ$  to  $360^\circ$

**Ex: transform : rotate(90deg);**

**transform : rotate(x/y)(90deg);**

```
<style>
    .cont {
        border: 2px solid;
        width: 200px;
        height: 120px;
        margin: 10px auto;
        /*transform: rotate(90deg);*/
    }
</style>
</head>

<body>
    <div class="cont"></div>
    <script>
        var degree = 10;
        setInterval(() => {
            document.querySelector( ".cont" ).style.transform = "rotate(" + (" + degree + "deg" + ")";
            degree += 10;
        }, 0.1);
    </script>

```

- **Transform with translate property:**

Through which we could able to move DOM Element to a new position with provided values.

**Ex: transform: translate(30px, 20px);**

```
<style>
    .cont {
        border: 2px solid;
        border-radius: 100px;
        width: 200px;
        height: 120px;
        position: absolute;
        top: 100px;
        left: 200px;
        box-shadow: 1px 2px 5px 2px;
    }

    .cont:hover {
        transform: translate(100px, 200px);
    }

</style>
```

- **Transform with Scale:**

Through which we can redraw the elements with effects,

**Ex: transform: scale(3,4) ;**

**// redraws the DOM element with 3times of width and 4 times of height**

```
<style>
    .cont {
        border: 2px solid;
        border-radius: 100px;
        width: 200px;
        height: 120px;
        position: absolute;
        top: 300px;
        left: 200px;
        box-shadow: 1px 2px 5px 2px;
        text-align: center;
    }

    .cont:hover {
        transform: scale(2, 4);
    }

</style>
```

- **Transform with Scale:**

The **skew () CSS function** defines a transformation that skews an element on the 2D plane.

**Ex: transform: skew (15deg, 15deg);**

**transform: skew (0);**

```
<style>
    .cont {
        border: 2px solid;
        border-radius: 100px;
        width: 200px;
        height: 120px;
        position: absolute;
        top: 300px;
        left: 200px;
        box-shadow: 1px 2px 5px 2px;
        text-align: center;
    }

    .cont:hover {
        transform: skew(2deg, 14deg);
    }

</style>
```

- The skew () function is specified with either one or two values, which represent the amount of skewing to be applied in each direction. If you only specify one value it is used for the x-axis and there will be no skewing on the y-axis.

**Syntax:** `skew(ax);`

`//ax is an angle representing the angle to use to distort the element along the x-axis.`

`skew (ax, ay);`

`// ax is an angle representing the angle to use to distort the element along the x-axis. ay is an angle representing the angle to use to distort the element along the y-axis.`

- CSS Keyframes:**

The ‘@keyframes’ or the set of CSS rules which lets the user to control the intermediate steps in CSS animation sequence by establishing keyframes along with the animation sequence which must be reached to a certain point during the animation.

- In order to make use of keyframes we create a keyframe rule with the user defined name, we use CSS animation property to match its animation to its keyframe list.
- Each keyframe rule contains a style list a keyframe selectors each of which contains a percentage along with the animation at which the key frame occurs as well as a block containing transformation for that key frame.

**Syntax:**

`@keyframes slidein {`

`Value1 % {`

`transform: translateX(0%);`

`}`

`Value % {`

`transform: translateX(100%);`

`}`

`}`

**Ex:**

```
<style>
    @keyframes moving {
        10% {
            background-color: red;
            top: 200px;
            transform: rotate(360deg);
        }

        40% {
            background-color: blue;
            top: 100px;
            left: 300px;
        }

        100% {
            background-color: aqua;
            bottom: 100px;
            transform: rotate(360deg);
        }
    }

    .cont {
        border: 2px solid;
        width: 100px;
        height: 180px;
        position: absolute;
        animation: moving 5s infinite;
    }

</style>
```

- **CSS Linear Gradient:**

The **linear-gradient()** is a CSS function which creates an image consisting of a progressive transition between two or more colors along a straight line.

Using CSS3 **linear-gradient()** we could add smooth transitions two or more specified colors.

**Syntax:**

**background-image: linear-gradient (to <direction>, color1, color2);**

Direction could be,

to left, to right, to left bottom, to right bottom etc...

- **With direction as angle:**

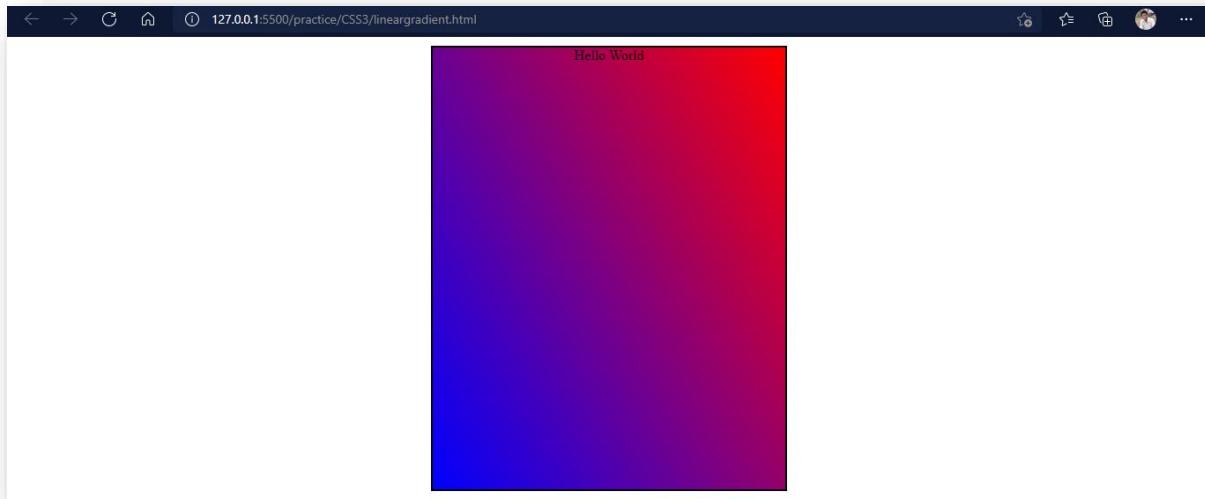
**background-image: linear-gradient (<angle>, color1, color2);**

- **Repeat linear-gradient():**

**background-image: repeating-linear-gradient (to <direction>/angle, color1, color2);**

**Ex:**

```
.inner {  
    border: 2px solid;  
    width: 400px;  
    height: 500px;  
    margin: 10px auto;  
    text-align: center;  
    background-image: linear-gradient(240deg, red, blue);  
}
```



- **Radial-gradients():**

Using which the gradients can be specified with in the centre.

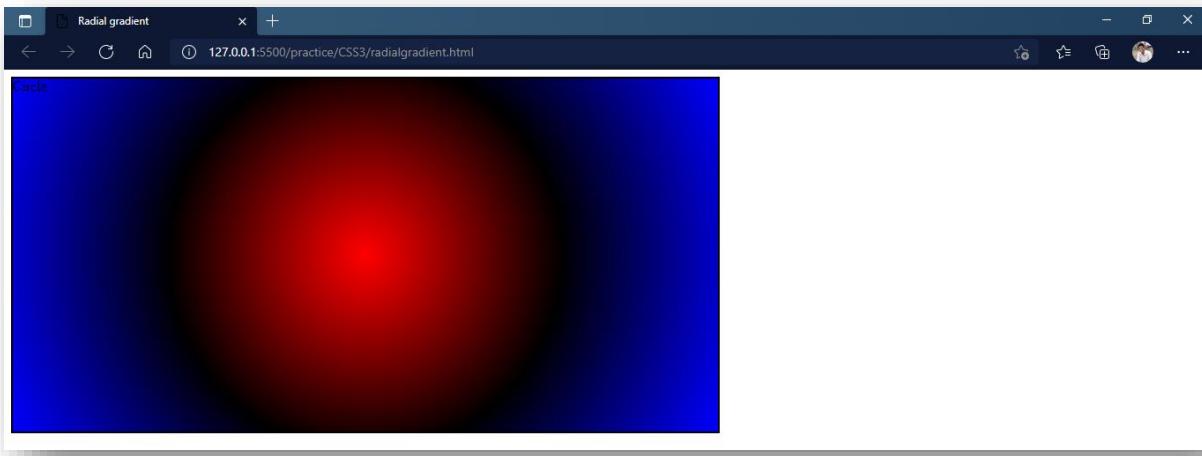
By default, the **Radial-gradients()** shape is ellipse and position is enter, we can still change the position and shape.

**background-image: radial-gradient (color1, color2,...);**

**background-image: radial-gradient (circle, color1 5%, color2 10%,...);**

**Ex:**

```
.rad {  
    border: 2px solid;  
    width: 800px;  
    height: 400px;  
    background-image: radial-gradient(circle, red, black, blue);  
}
```



- **CSS3 Media Queries:**

A predefined feature being supported in CSS3 through which we could able to design multiple set of CSS Classes for different device dimensions so that corresponding CSS will be applied for particular device dimensions.

“**@media**” is a keyword through which we define set of CSS Classes for particular dimensions.

**Syntax:** **@media <mediatype> and (dimension) {**

.....  
}

Following are the set of properties through which we can specify device dimensions within media queries,

- Min-width
- Min-height
- Max-height
- Max-width

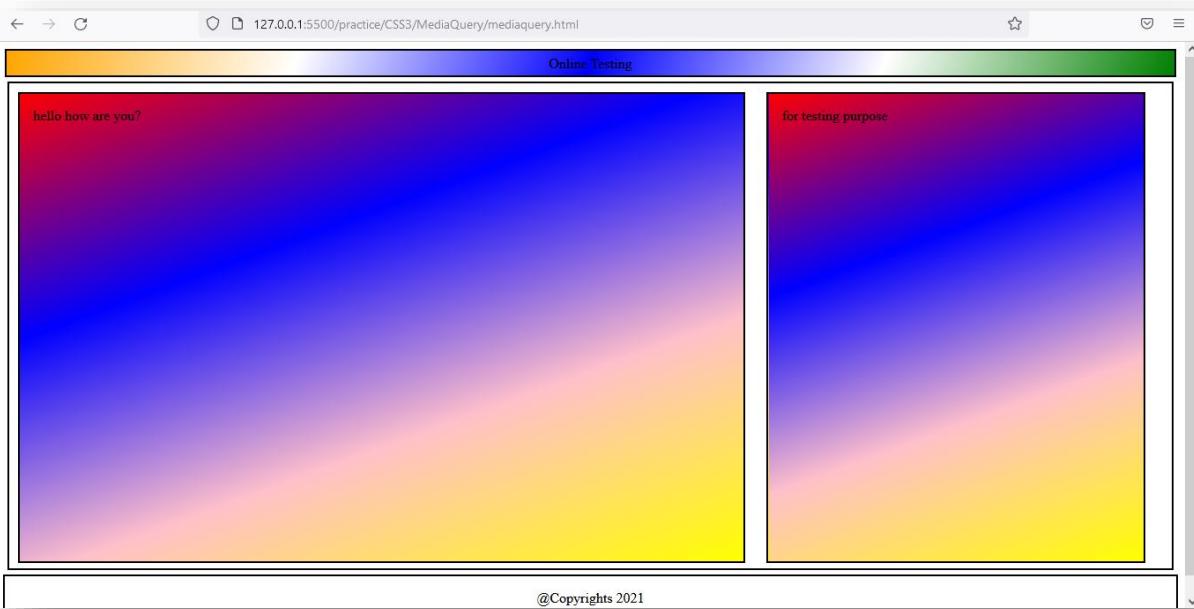
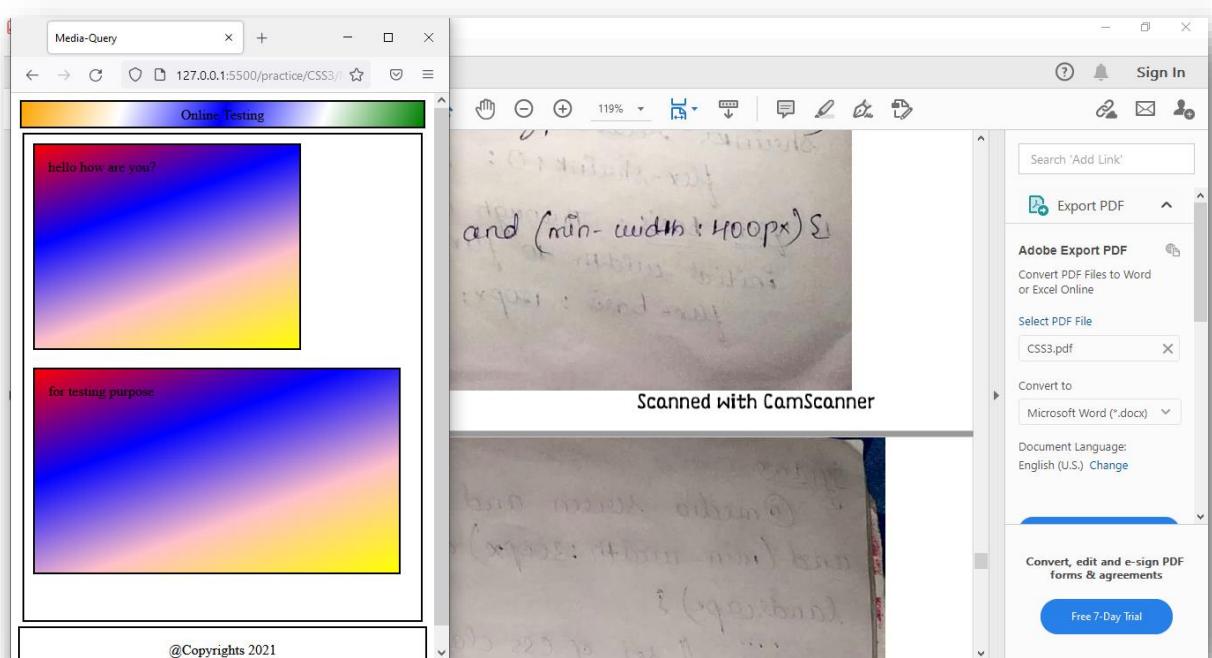
Following are the types of media can be specified while defining media queries,

#### **Media Types:**

- **Screen** – Any color screen
- **All** – indicates all screens
- **Handheld** – indicates all small screen projections
- **Print** – indicates print devices.

**Ex:**

```
@media screen and (min-width : 480px) and (max-width : 585px) {  
    .inner1 {  
        height: 200px;  
    }  
    .inner2 {  
        height: 200px;  
        width: 85%;  
    }  
}
```



- While defining the media-queries we can specify the device orientation using orientation property.
- “**orientation**” possible values “**Landscape/portrait**”.
- **Syntax:** @media Screen and (max-width:300px) and (min-width:200px) and (orientation : landscape/portrait) {

..... //set of CSS Classes

}

- The media queries can be injected into the website by @media queries or <link rel="stylesheet" media="max-width:200px; min-height:400px"; orientation : portrait href="data/app.css">

#### ➤ CSS Flex:

Following are the five layout models been supported in CSS,

- Block layout
- Inline layout
- Table layout
- Positioned layout
- Flex box layout

#### ➤ Flex Box Layout:

Flex container becomes flexible by setting the display property to flex. “**display:flex;**” through Flex Box Layout module we could design responsive layout structures without any dependency of either float or position.

Following are the properties can be applied to flex container.

- Flex-direction
- Flex-wrap
- Flex-flow
- Justify-content
- Align-items
- Align-content

#### ➤ Flex direction property:

Through which we can specify in which the direction of the flex items to be rendered.

Following are the possible values it takes,

- Column
- Row

- Column-reverse
- Row-reverse

➤ **Flex-Wrap:**

Through which the items with flex container gets wrapped if needed.

**flex-wrap : wrap / nowrap / wrap-reverse;**

➤ **Flex-flow:**

It's a shortcut way to define both direction and wrap property.

**Flex-flow : row wrap;**

➤ **Justify-content:**

Through which we can align the items under flex container.

**justify-content : center / flex-start / flex-end / stretch / baseline.**

➤ **Align-items:**

Through which we could align flex items.

**Align-items : center / flex-start / flex-end / stretch / baseline.**

➤ **Align-content:**

To align content inside flex container.

**flex-wrap : wrap;**

**align-content : space-between / space-around / stretch / center;**

```
<style>
    .outer {
        border: 2px solid;
        height: 500px;
        background-color: aqua;
        display: flex;
        flex-wrap: wrap;
        justify-content: center;
        /*center the items in row wise*/
        align-items: center;
        /*center the items in column wise*/
    }

    .inner {
        border: 2px solid;
        width: 160px;
        height: 160px;
        background-image: linear-gradient(270deg, red, yellow);
        padding: 10px;
        margin: 15px;
        text-align: center;
    }
</style>
```

The screenshot shows a Visual Studio Code interface with a file named 'CSSFlex.html' open. The code defines a flex container with a red-to-yellow gradient background, containing four items labeled 'Item 1' through 'Item 4'. The browser preview on the right shows the resulting layout.

```

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Flex</title>
    <style>
        .outer {
            border: 2px solid black;
            height: 500px;
            background-color: #aqua;
            display: flex;
            flex-wrap: wrap;
            justify-content: center;
            /*center the items in row wise*/
            align-items: center;
            /*center the items in column wise*/
        }

        .inner {
            border: 2px solid black;
            width: 160px;
            height: 160px;
            background-image: linear-gradient(270deg, red, yellow);
            padding: 10px;
            margin: 15px;
            text-align: center;
        }
    </style>
</head>
<body>
</body>

```

## ➤ Flex-item Properties:

Following are the properties can be applied to flex items,

- Order
- Flex-grow
- Flex-shrink
- Flex-basis
- Flex
- Align-self

## ➤ Order:

Through which we can control rendering order of flex item,

**Order: 1/3/5...;**

## ➤ Flex-grow:

Through which we can specify how much an item grows relative to other flex items in it,

**Flex-grow: 1/3/8...;**

## ➤ Flex-shrink:

Specifies how much item shrinks relative to other items,

**Flex-shrink: 0; //no Shrink**

➤ **Flex-basis:**

Through which we can specify initial width to the flex item.

**Flex-basis: 120px;**

➤ **Flex:**

Through which we can control flex-grow, shrink and basis at a time.

**Flex: 1 0 200px;**

➤ **Align-self:**

Through which we can make item to get aligned automatically within the container.

**Align-self: center;**

```
<style>
    .container {
        background-color: green;
        padding: 50px;
        display: flex;
    }

    .item {
        background-color: yellow;
        width: 100px;
        height: 100px;
        text-align: center;
        margin: 10px;
    }

</style>
</head>

<body>
    <div class="container">
        <div class="item" style="order:3;flex-shrink: 3; flex-wrap: nowrap;">1</div>
        <div class="item" style="order:1 ;flex-grow:1">2</div>
        <div class="item" style="order:4;flex-grow:1">3</div>
        <div class="item" style="order:2;flex-grow:1">4</div>
    </div>
</body>
```



## **BootStrap-V5.1.1**

➤ <https://getbootstrap.com/> Get full document here.