WEB TECHNOLOGY NOTES

HTML

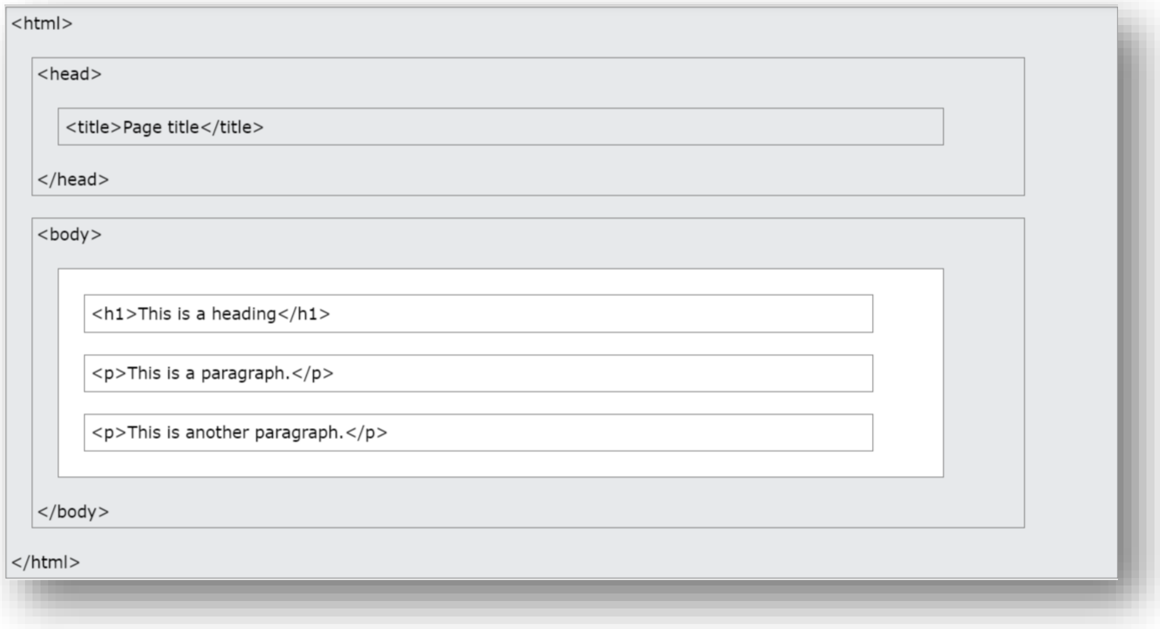
❖ What is HTML?

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

❖ What is an HTML Element?

- An HTML element is defined by a start tag, some content, and an end tag:
- <tagname> Content goes here... </tagname>
- The HTML **element** is everything from the start tag to the end tag:
- <h1>My First Heading</h1>
- <p>My first paragraph.</p>

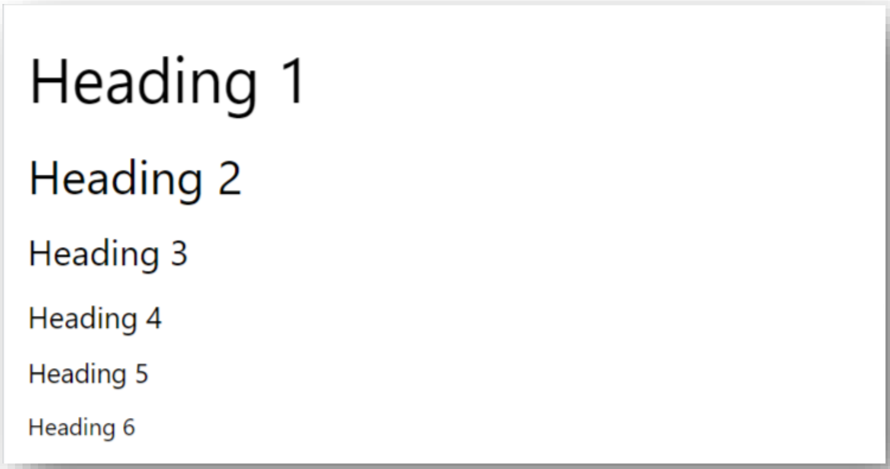
❖ HTML Page Structure :-



```
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

❖ HEADING TAGS :-

- `<h1>Heading 1</h1>`
- `<h2>Heading 2</h2>`
- `<h3>Heading 3</h3>`
- `<h4>Heading 4</h4>`
- `<h5>Heading 5</h5>`
- `<h6>Heading 6</h6>`



Heading 1

Heading 2

Heading 3

Heading 4

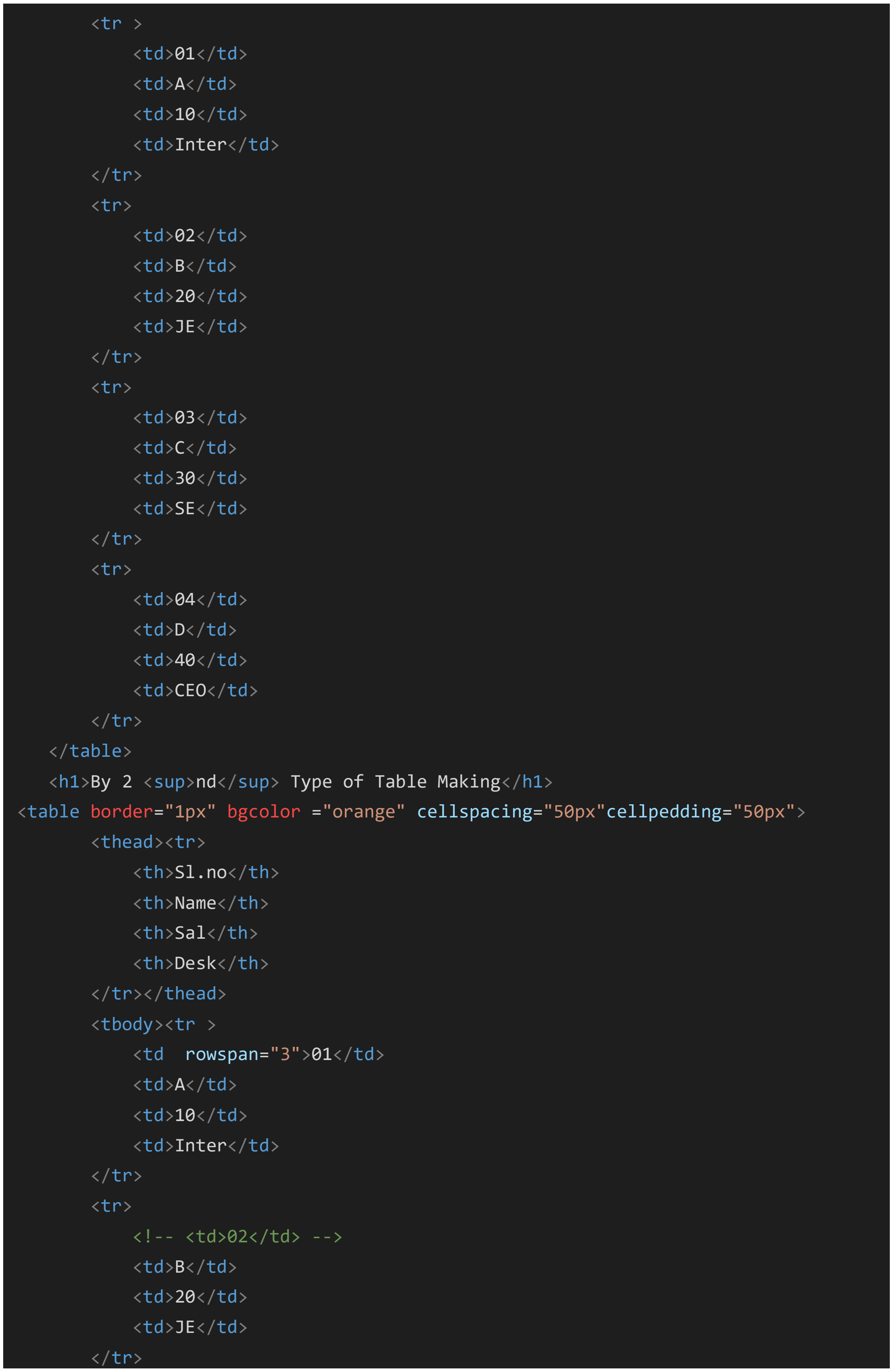
Heading 5

Heading 6

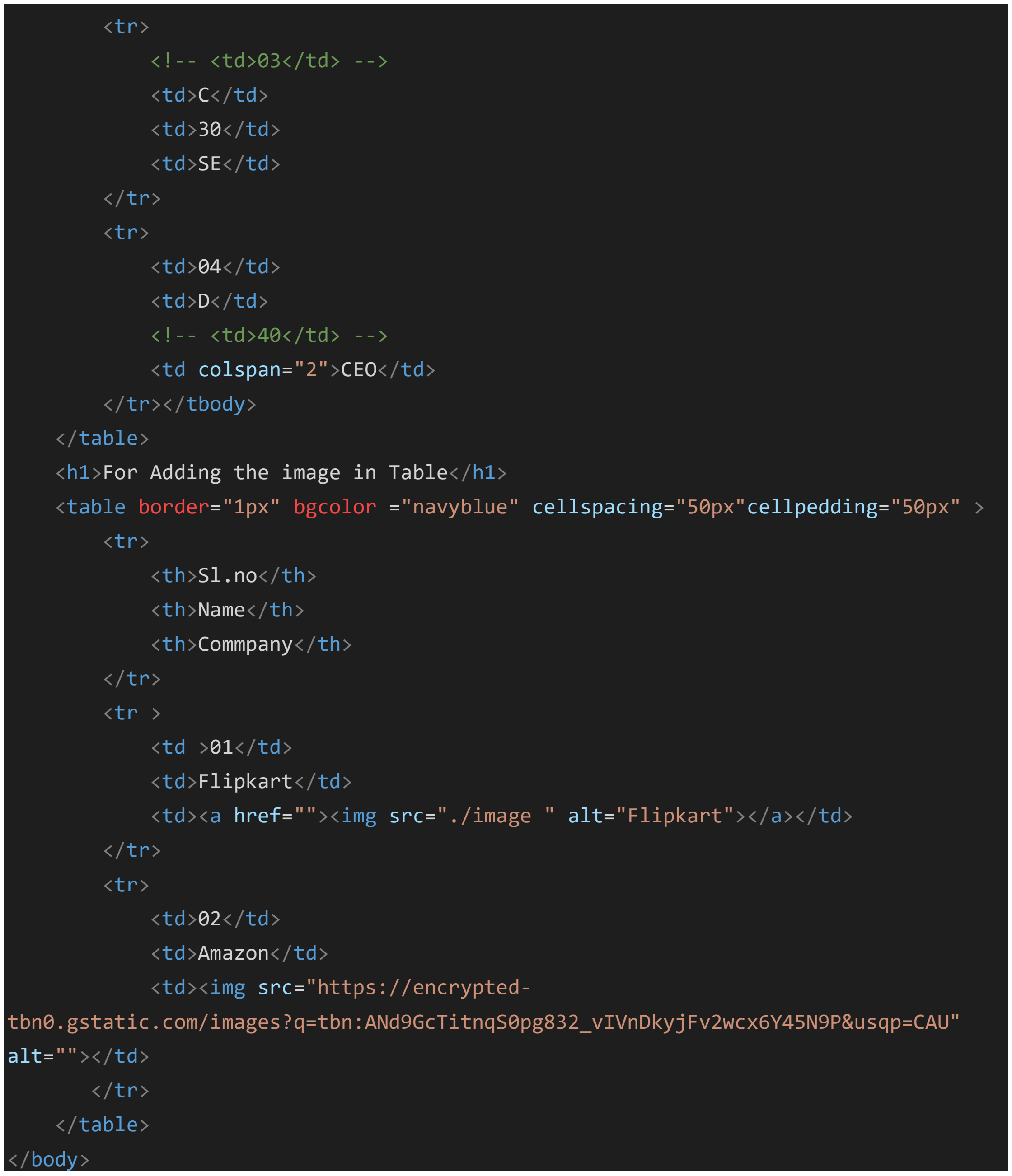
❖ TABLE MAKING :-

- border="1px"--for border]
- bgcolor = "green"--for back ground color to green]
- cellspacing ="50px"--for cell space to 50px
- <Table> --For making table
- <tr> --For writting inside the a row.
- <td> --For inserting table data
- cellpedding= "50px"--for space between content and border
- **E.g.:-**

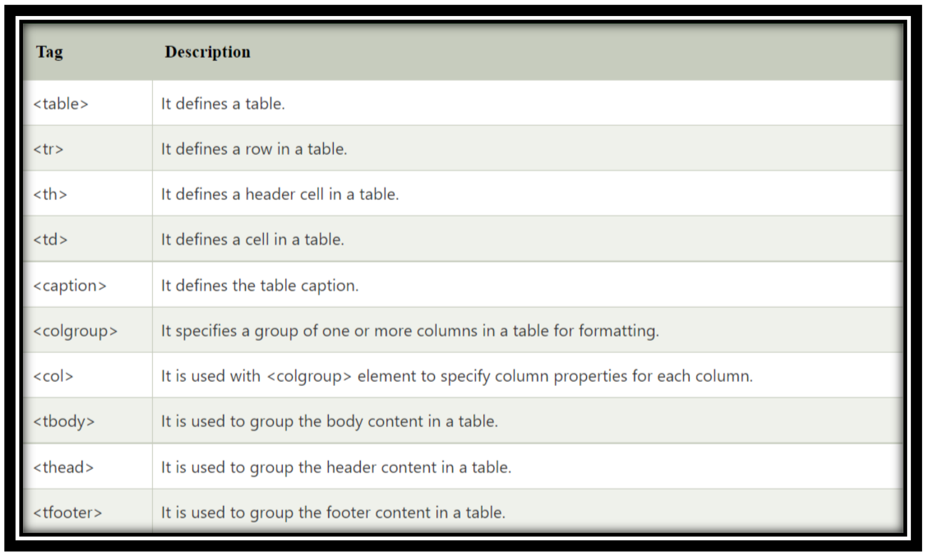
```
<body>
    <h1>By 1<sup>st</sup> Type of Table Making</h1>
    <table border="1px" bgcolor ="green" cellspacing="50px" cellpedding="50px">
        <tr>
            <th>Sl.no</th>
            <th>Name</th>
            <th>Sal</th>
            <th>Desk</th>
        </tr>
```



```
       <tr >
           <td>01</td>
           <td>A</td>
           <td>10</td>
           <td>Inter</td>
       </tr>
       <tr>
           <td>02</td>
           <td>B</td>
           <td>20</td>
           <td>JE</td>
       </tr>
       <tr>
           <td>03</td>
           <td>C</td>
           <td>30</td>
           <td>SE</td>
       </tr>
       <tr>
           <td>04</td>
           <td>D</td>
           <td>40</td>
           <td>CEO</td>
       </tr>
   </table>
   <h1>By 2 <sup>nd</sup> Type of Table Making</h1>
<table border="1px" bgcolor ="orange" cellspacing="50px"cellpedding="50px">
       <thead><tr>
           <th>Sl.no</th>
           <th>Name</th>
           <th>Sal</th>
           <th>Desk</th>
       </tr></thead>
       <tbody><tr >
           <td  rowspan="3">01</td>
           <td>A</td>
           <td>10</td>
           <td>Inter</td>
       </tr>
       <tr>
           <!-- <td>02</td> -->
           <td>B</td>
           <td>20</td>
           <td>JE</td>
       </tr>
```

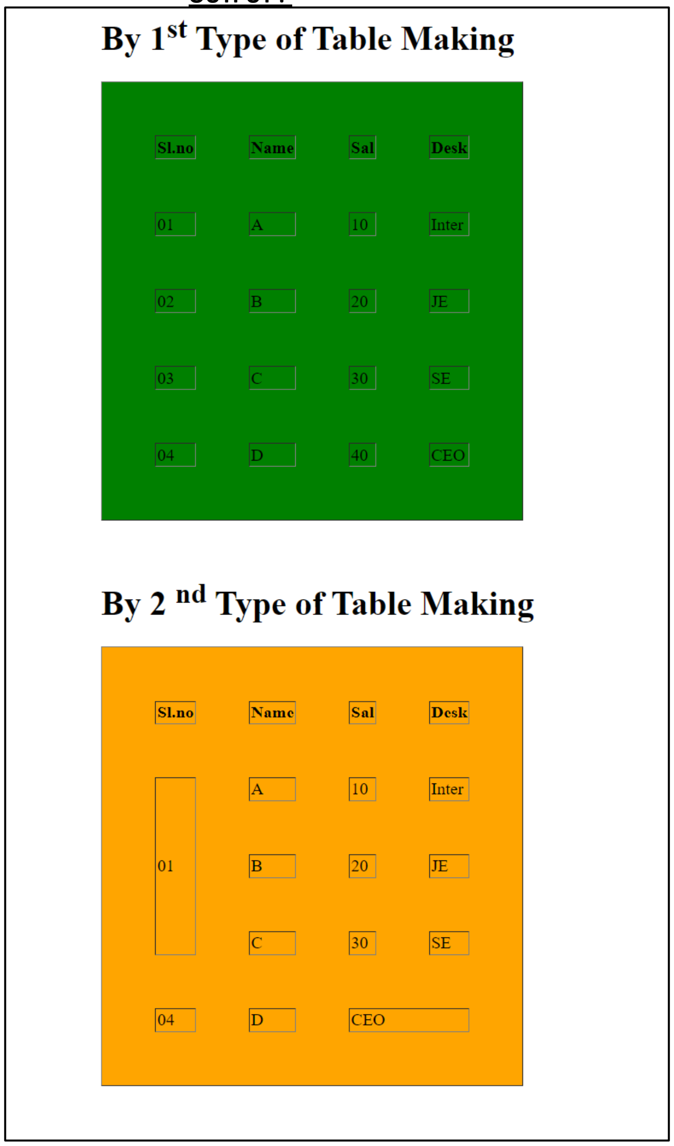


```
        <tr>
            <!-- <td>03</td> -->
            <td>C</td>
            <td>30</td>
            <td>SE</td>
        </tr>
        <tr>
            <td>04</td>
            <td>D</td>
            <!-- <td>40</td> -->
            <td colspan="2">CEO</td>
        </tr></tbody>
    </table>
    <h1>For Adding the image in Table</h1>
    <table border="1px" bgcolor ="navyblue" cellspacing="50px"cellpedding="50px" >
        <tr>
            <th>Sl.no</th>
            <th>Name</th>
            <th>Commpany</th>
        </tr>
        <tr >
            <td >01</td>
            <td>Flipkart</td>
            <td><a href=""><img src="./image " alt="Flipkart"></a></td>
        </tr>
        <tr>
            <td>02</td>
            <td>Amazon</td>
            <td><img src="https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcTitnqS0pg832_vIVnDkyjFv2wcx6Y45N9P&usqp=CAU"
alt=""></td>
       </tr>
    </table>
</body>
```



Tag	Description
<table>	It defines a table.
<tr>	It defines a row in a table.
<th>	It defines a header cell in a table.
<td>	It defines a cell in a table.
<caption>	It defines the table caption.
<colgroup>	It specifies a group of one or more columns in a table for formatting.
<col>	It is used with <colgroup> element to specify column properties for each column.
<tbody>	It is used to group the body content in a table.
<thead>	It is used to group the header content in a table.
<tfooter>	It is used to group the footer content in a table.

✓ **OUTPUT :-**

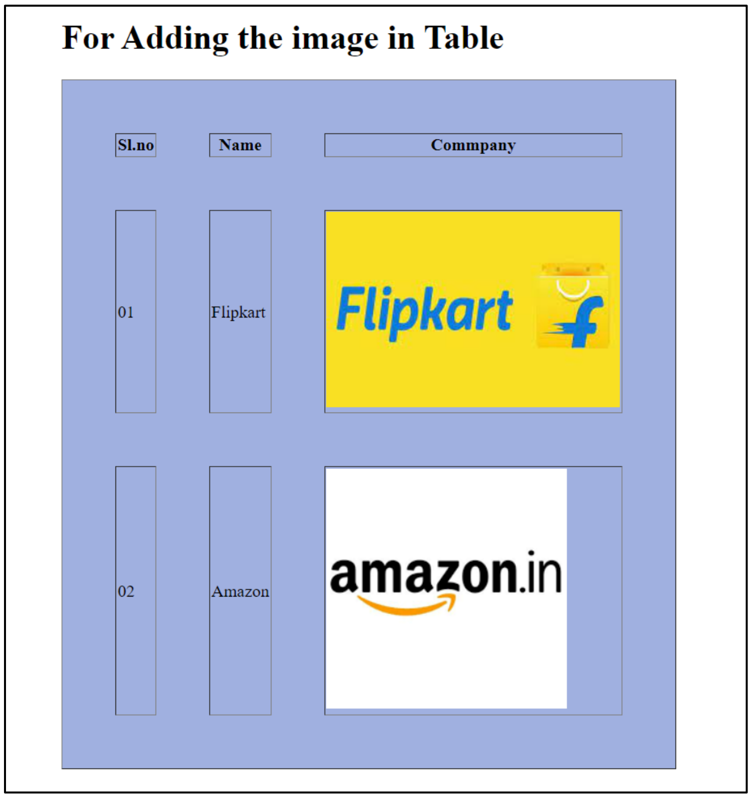


By 1st Type of Table Making

Sl.no	Name	Sal	Desk
01	A	10	Inter
02	B	20	JE
03	C	30	SE
04	D	40	CEO

By 2 nd Type of Table Making

Sl.no	Name	Sal	Desk
01	A	10	Inter
	B	20	JE
	C	30	SE
04	D	CEO	



For Adding the image in Table

Sl.no	Name	Commpany
01	Flipkart	
02	Amazon	

- ❖ **COLSPAN :-** It means murging two or more columns

- ❖ **ROWSPAN :-** It means murging two or more rows.

- ❖ **FORMS :-**

    - ➢ Form is used to collect data from user.
    - ➢ It is a paired tag.
    - ➢ Action which means collecting a data to server size path.

- ❖ **# :-** It will refresh in as server page.

- ❖ **GET** :-
    - ○ This method is for security purpose
        - ▪ Get is not a secure one because it save data in bookmark.

- ❖ **SEMANTIC TAG-**

  These are the tags introduce in HTML 5 and these semantic tag helps a programmer to write easy code.

    - ➢ H-Header
    - ➢ D-Div
    - ➢ A-Article
    - ➢ M-Main
    - ➢ N-Nav
    - ➢ S-Section
    - ➢ T-Time
    - ➢ A-Aside

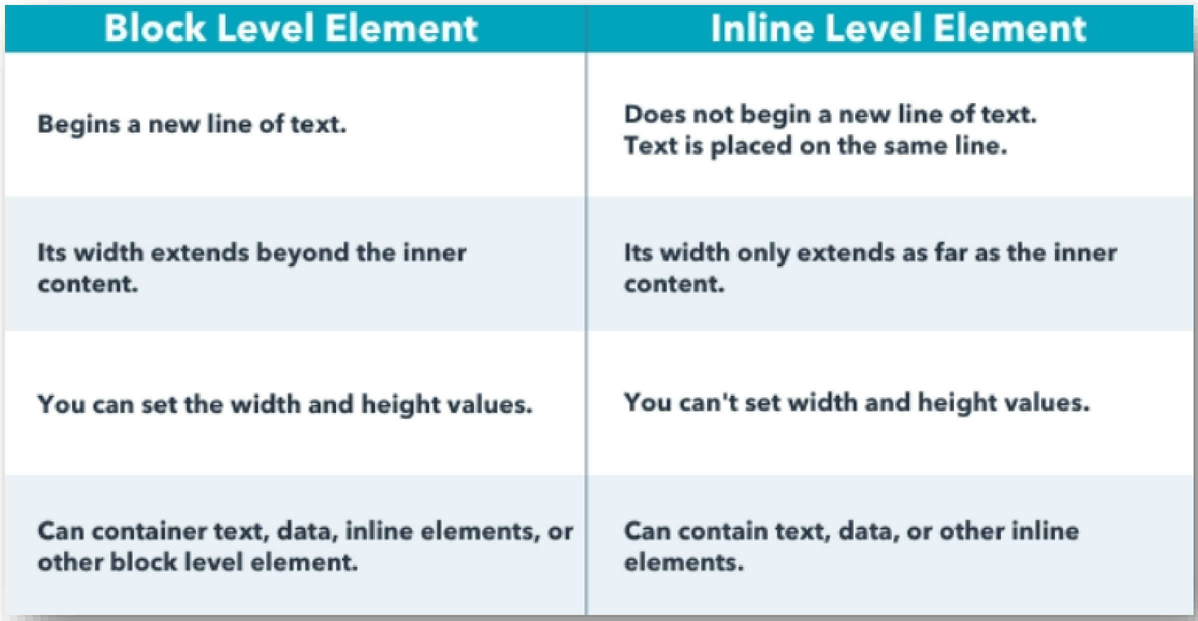
- F-Figure
- F-Figcaption
- F-Footer

❖ FIGURE-

```
 <figure>
    <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcR7ODD-
4dbySXd0pwY8sE9GCE-jg-CCmXmYDw&usqp=CAU" alt=""height=300px width="400">
    </figure>
```

Block level element & in length of element.

Block Level Element	Length of Element
1. It will acquire entire width of an web page	1. It will affect to the particular content
2. Article as DD(definition data)DT(definition term)DM, order list, Unorder list, paragraph, Table	2. Bold, big, small, EM, italic, input label, spam, subscript, super script, text area.



Block Level Element	Inline Level Element
Begins a new line of text.	Does not begin a new line of text. Text is placed on the same line.
Its width extends beyond the inner content.	Its width only extends as far as the inner content.
You can set the width and height values.	You can't set width and height values.
Can container text, data, inline elements, or other block level element.	Can contain text, data, or other inline elements.

❖ FORMS: -

- **What is HTML <form>?**
  - <form> is a HTML element to collect input data with containing interactive controls.
  - It provides facilities to input text, number, values, email, password, and control fields such as checkboxes, radio buttons, submit buttons, etc.
  - In other words, form is a container that contains input elements like text, email, number, radio buttons, checkboxes, submit buttons, etc.
  - Forms are generally used when you want to collect data from the user.\

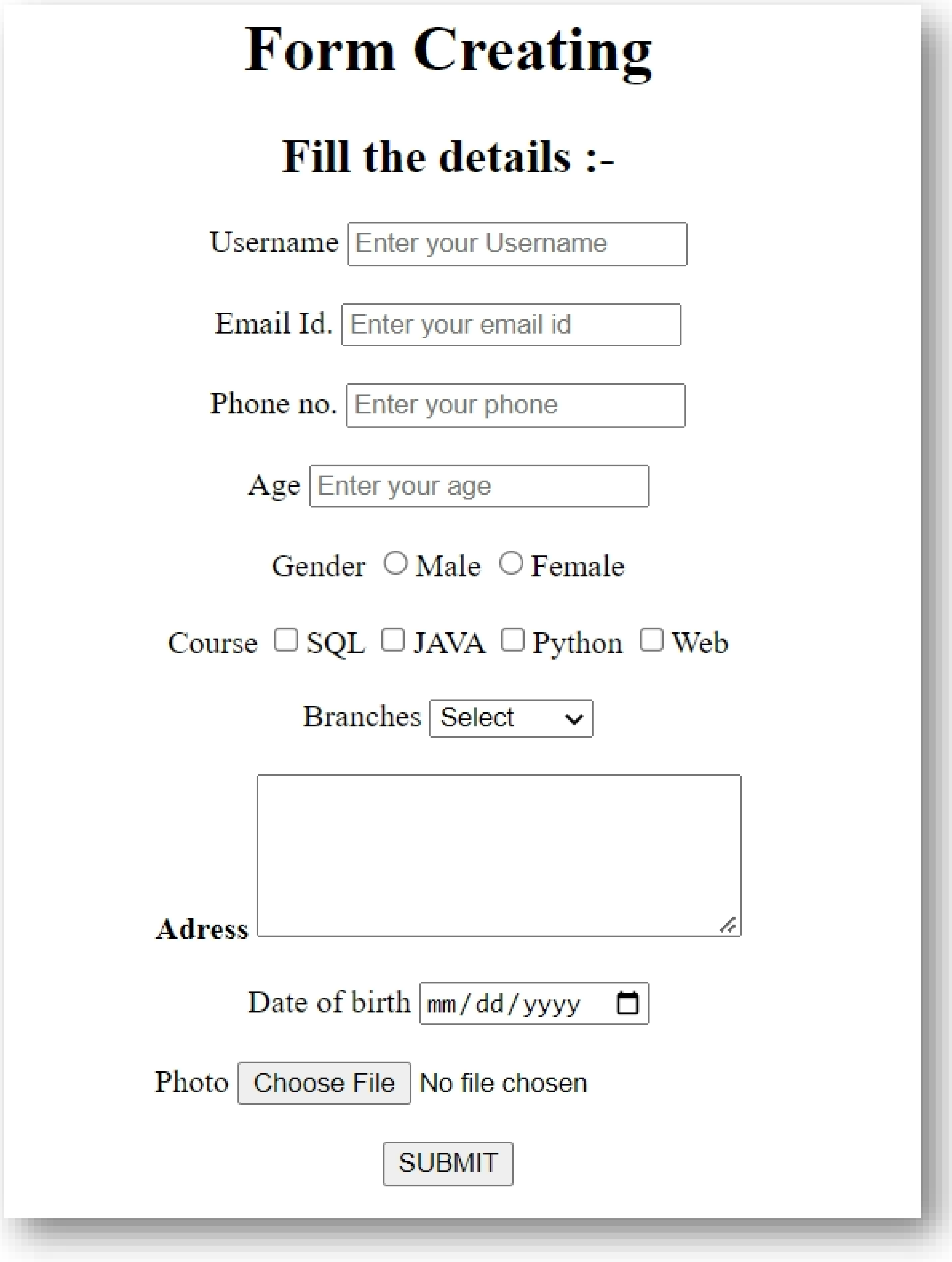
- **Form elements :-**

- These are the following HTML <form> elements:
    - **<label>:** It defines label for <form> elements.
    - **<input>:** It is used to get input data from the form in various types such as text, password, email, etc by changing its type.
    - **<button>:** It defines a clickable button to control other elements or execute a functionality.
    - **<select>:** It is used to create a drop-down list.
    - **<textarea>:** It is used to get input long text content.
    - **<fieldset>:** It is used to draw a box around other form elements and group the related data.
    - **<legend>:** It defines caption for fieldset elements.
    - **<datalist>:** It is used to specify pre-defined list options for input controls.
    - **<output>:** It displays the output of performed calculations.
    - **<option>:** It is used to define options in a drop-down list.
    - **<optgroup>:** It is used to define group-related options in a drop-down list.

E.g :

```
<body>
            <h1>Form Creating</h1>
            <h2>Fill the details :-</h2>
            <form action="#" method="get">
            <label for="Username">Username</label>
            <input type="text" id="Username"placeholder="Enter your Username">
            <br>
            <br>
            <label for="#">Email Id.</label>
            <input type="email" id="Email"placeholder="Enter your email id">
            <br>
            <br>
            <label for="#">Phone no.</label>
            <input type="" id="Phone"placeholder="Enter your phone">
            <br><br>
            <label for="Age">Age</label>
            <input type="number" id="Age" placeholder="Enter your age">
            <br>
            <br>
            <label for="Gender">Gender</label>
            <input type="radio" name="Gender" value="Male">Male
            <input type="radio" name="Gender" value="Femalee">Female
            <br>
            <br>
```

```html
            <label for="Course">Course</label>
            <input type="checkbox" name="Course" id="Course" value="SQL">SQL
            <input type="checkbox" name="Course" id="Course" value="JAVA">JAVA
            <input type="checkbox" name="Course" id="Course"
value="Python">Python
            <input type="checkbox" name="Course" id="Course" value="Web">Web
            <br><br>
            <label for="Branch">Branches</label>
            <select name="Branches" id="Branch">
                <option value="Select">Select</option>
                <option value="Hadapsar">Hadapsar</option>
                <option value="Wakad">Wakad</option>
                <option value="Deccan">Deccan</option>
                <option value="Other">Other</option>
            </select>
            <br>
            <br>
            <label for="Adress"> <b>Adress</b></label>
            <textarea name="Adress" id="Adress" cols="30" rows="5"></textarea>
            <br>
            <br>
            <label for="Date of birth">Date of birth</label>
            <input type="date" name="Date of birth" id="Date of birth">
            <br>
            <br>
            <label for="Photo">Photo</label>
            <input type="file" name="Photo" id="Photo">
            <br><br>
            <button>SUBMIT</button>
        </form>
```



Form Creating

Fill the details :-

Username Enter your Username

Email Id. Enter your email id

Phone no. Enter your phone

Age Enter your age

Gender ○ Male ○ Female

Course ☐ SQL ☐ JAVA ☐ Python ☐ Web

Branches Select

Adress

Date of birth mm/dd/yyyy

Photo Choose File No file chosen

SUBMIT

Css

❖ CSS –

CSS (Cascading Style Sheets)is used to apply styles to web pages. Cascading Style Sheets are fondly referred to as CSS. It is used to make web pages presentable. The reason for using this is to simplify the process of making web pages presentable. It allows you to apply styles on web pages. More importantly, it enables you to do this independently of the HTML that makes up each web page.

- CSS which means cascading styling which helps to give styling for the HTML page
- The current version of CSS is 3.
- I can create CSS file by .css
- We give styling give in 3(Three) ways.

1. **INLINE :** Inline CSS contains the CSS property in the body section attached with the element known as inline CSS.

```
</head>
<body>
    <p style="color:blue;">Welcome to css class</p>
</body>
</html>
```

2. **INTERNAL :** The CSS ruleset should be within the HTML file in the head section i.e the CSS is embedded within the HTML file.

```
 <title>Document</title>
    <style>
        p{
            color: brown;
        }
    </style>
</head>
<body>
</body>
</html>
```

3. **EXTERNAL-:** External CSS contains a separate CSS file that contains only style property with the help of tag attributes.
   - In html page-:

```
 <title>Document</title>
    <link rel="stylesheet" href="./abc.css">
</head>
<body>
    <p>Hello World</p>
</body>
</html>
```

- Now in css page -:

```
p{
    color: rgb(142, 201, 13);
}
```

❖ **CSS Syntax-:**

```
   <style>
        select{
            property value
              }
    </style>
```

E.g-

➢ Selectors which means it targets the html elements.

I. SIMPLE SELECTORS-

Simple-In simple selector, we know ID selector. It is for unique property.

a) ID (#)-:

- In html page-:

```
</head>
<body>
    <span id="span1">Q</span>spider
    <br>
    <span>Test</span>On monday
</body>
</html>
```

- Now in css page -:

```
#span{
    color: darkgreen;
}
```

b) Class (.)-:

- In html page-:

```
 <title>Document</title>
    <link rel="stylesheet" href="./abc.css">
</head>
<body>
    <img src="https://repository-
images.githubusercontent.com/378700960/737db180-d286-11eb-
8ce1-f3201eccd1b1" alt="" class="img1">
    <img src="https://repository-
images.githubusercontent.com/378700960/737db180-d286-11eb-
8ce1-f3201eccd1b1" alt=""class="img1">
```

```
    <img src="https://repository-
images.githubusercontent.com/378700960/737db180-d286-11eb-
8ce1-f3201eccd1b1" alt=""class="img1">
    <img src="https://repository-
images.githubusercontent.com/378700960/737db180-d286-11eb-
8ce1-f3201eccd1b1" alt=""class="img1">
</body>
```

- Now in css page -:

```
.img1{
    height: 50%;
    width: 49%;
}
```

c) Universal (*):-

```
<body>
    <h1>Good morning</h1>
    <p id="p1">para1</p>
    <p class="pc1"></p>
</body>
```

```
*{
    padding: 0;
    margin: 0;
    box-sizing: border-box;
}
```

- It means it is targeting entire web page

d) Grouping (,)-:

- In html page-:

```
<body>
    <h1>Good morning</h1>
    <p id="p1">para1</p>
    <p class="pc1"></p>
</body>
```

- Now in css page -:

```
h1,#p1,.pc1{
    font-size: 100px;
}
```

- Grouping selector which means targeting html elements, id, class, tagname.

e) Element (Tag name)-:

- In html page-:

```
</head>
<body>
    <p>Para1</p>
</body>
```

- in css page -:

```
p{
    color: red;
}
```

- Element selector which means ,it targets the html elements by the tag name

1. Combinator:-

It shows the relation sheet between parent and child selector.

1) Descendent selector-:

Which means it affecting to direct child and also for the indirect child.

```
<body>
    <div>1
        <2p>para1</p>
        <p>para2</p>
        <span><p>span para</p></span>
    </div>
</body>
```

```
div p{
    color: red;
}
```

2) Child combinator -:

Child combinator selector which mean it is going to affect only child.

```
<body>
    <div>1
        <2p>para1</p>
        <p>para2</p>
        <span><p>span para</p></span>
    </div>
</body>
```

```
div>p{
    color: red;
}
```

3) Adjacent sibling :-

Adjacent sibling combinator selector which means it is affecting to the first sibling.

```
<body>
    <div>
        <p>para1</p>
        <p>para2</p>
        <span><p>span para</p></span>
    </div>
        <p>outside para1</p>
        <p>outside para2</p>
</body>
```

```
div+p{
    color: red;
}
```

4) General sibling:-

- ➢ It is going to select all the siblings.
- ➢ It vice-versa to adjacent sibling combination

```
<body>
    <div>
        <p>para1</p>
        <p>para2</p>
        <span><p>span para</p></span>
    </div>
        <p>outside para1</p>
        <p>outside para2</p>
</body>
```

```
Div~p{
    color: red;
}
```

2. Pseudo class :-

(All code from below is in .css page)

1) First-line: Selects the first line of every <p> element

```
p::first-line{
    color: orange;
}
```

2) First letter : Selects the first letter of every <p> element

```
p::first-letter{
    color: green;
    font-size: 70px;
}
```

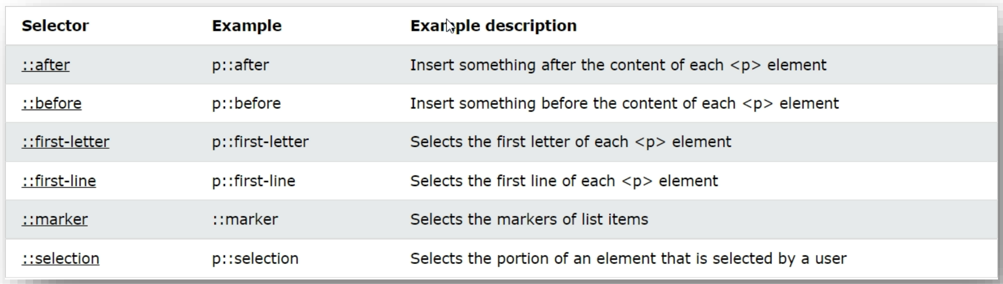
3) **After :** Selects the first letter of every <p> element.

```
p::after{
    color: red;
    font-size: 100px;
}
```

4) **Before:** Insert content before every <p> element.

```
P::before{
        color: blue;
        font-size: 50px;
}
```

3. Pseudo Elements :-



Selector	Example	Example description
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
::first-letter	p::first-letter	Selects the first letter of each <p> element
::first-line	p::first-line	Selects the first line of each <p> element
::marker	::marker	Selects the markers of list items
::selection	p::selection	Selects the portion of an element that is selected by a user

---

❖ Pseudo Class Selector :-

i. **Hover:**

Hover which helps to change the property when the cursor over on it.

```
img:hover{
    border-radius: 50%;
    height: 50%;
    width: 50%;
}
```

ii. **First-Child :-** First child which means it will affect only for first child.

(In html page)-

```
<p>Para 0</p>
<div>
```

```
    <p>Para1</p>
    <p>Para2</p>
    <p>Para3</p>
</div>
```

(In css page)-

```
p:first-child{
    background-color: aqua;
}
```

iii. **Last-Child :-** First child which means it will affect only for last child.

(In html page)-

```
<p>Para 0</p>
<div>
    <p>Para1</p>
    <p>Para2</p>
    <p>Para3</p>
</div>
```

(In css page)-

```
p:first-child{
    background-color: aqua;
}
```

iv. **First of type :-** The first of child is helps to all the first child.

(In css page)-

```
p:first-of-type{
    background-color: blanchedalmond;
}
```

v. **Lase of type :-**The last of child is helps to all the last child.

(In css page)-

```
p:last-of-type{
    background-color: blanchedalmond;
}
```

vi. **Nth-child :-** The nth of child is helps to nth child.

```
p:nth-child(2){
    background-color: blanchedalmond;
}
```

NOTE:- For continue refreshing after 5sec.

```
<meta charset="UTF-8">

    <meta http-equiv="refresh" content="5">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
```

❖ **Properties:-**

1. **Test-**
   a. Color
   b. Text-align
   c. Text-decoration
   d. Text-shadow (Horizontal verticle blurness color)
   e. Test-indent
   f. Word spacing
   g. Letter spacing

   e.g. :-

```
h1{
color: rgb(221, 255, 0);
text-align: center;
text-decoration: overline black;
text-shadow: 5px 4px 5px black;
text-indent: 100px;
word-spacing: 50px;
letter-spacing: 20px;
}
```

2. **Font-**
   a. Font – size
   b. Font – family
   c. Font – weight
   d. Font – variant
   e. Font – style

   E.g. :-

```
h1{
    font-size: large;
    font-family: Arial, Helvetica, sans-serif;
    font-weight: lighter;
    font-variant: small-caps;
    font-style: italic ;
}
```

3. **Background-**
   a. Background image
   b. Background position
   c. Background size
   d. Background repeat
   e. Height =”100px”;

   E.g, :

```
body{
background : linear-gradient(green, black);
background-position: cover;
background-size: 100%;
background-repeat: no-repeat;
height: 100%;
background : radial-gradient(orange);
}
```

❖ <u>Positions-:</u>

Positions which means targeting html element and changing the position

1. **<u>Static :</u>** Static position which does not change the place or position.(Nothing will change)

```
E.g-    div2{
    position: relative;
    top: 30px;
    left: 200px;
}
```

2. **<u>Absolute :</u>** Position absolute which means an html element moves form view port height.

```
E.g-    div2{
        position: relative;
        top: 30px;
        left: 200px;
        }
```

3. **<u>Relative :</u>**Position relative which means an html element move from original position.

```
E.g-    div2{
            position: relative;
            top: 30px;
            left: 200px;
        }
```

4. **<u>Fixed :</u>**Position fixed which means an html element moving with the screen.

```
E.g-    div2{
    position: relative;
    top: 30px;
    left: 200px;
        }
```

5. **<u>Sticky :</u>** Position sticky which means an html element fixed with the screen in same(top=”0” is mandatory).

```
E.g-    div2{
    position: relative;
    top: 30px;
    left: 200px;
        }
```

❖ **FLEX :-** Flex is an property which helps to make to create which is responsive web page.

- **Flex-wrap :** the flex property specify whether the flex item should wrap ort not.

E.g-:

- **In html page-**

```
<link rel="stylesheet" href="./new.css">
</head>
<body>
    <section>
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
    <div>6</div>
    <div>7</div>
    <div>8</div>
    </section>
</body>
```

- Now in css page -:

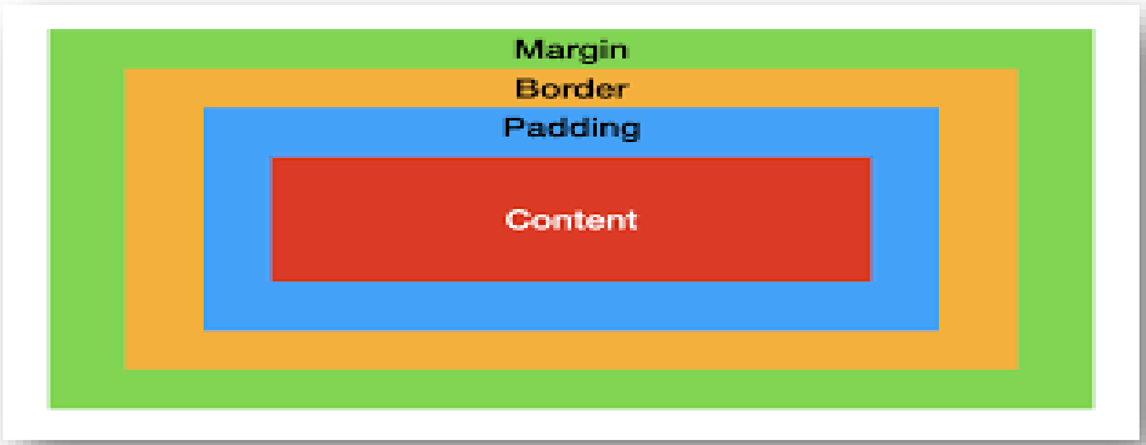
```
section{
    height: 100vh;
    width: 100%;
    background-color: black;
    display: flex;
    flex-wrap: wrap;
    align-items: center;
    justify-content: center;
    column-gap: 30px;
}
div{
    height: 250px;
    width: 250px;
    background-color: red;
    border: 1px solid;
}
```

❖ **Justify-content (space around) -:**

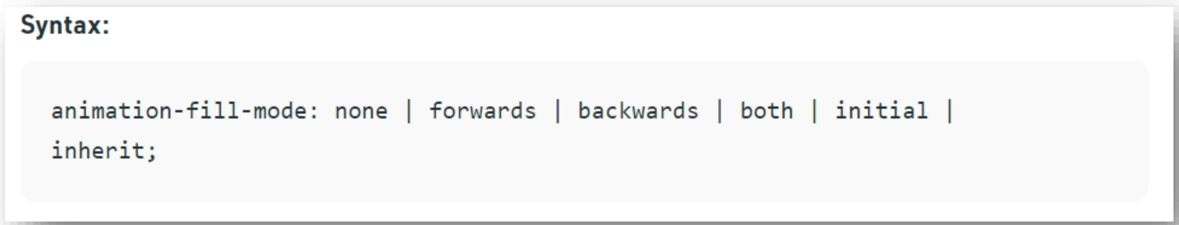
- getting space between around the blocks.
- Space between-getting space between the block.
- Space evenly- getting space equally.

❖ **BOX-MODEL :-**

- **Margin** -Which means the space between border and view port.
- **Border** -border which means the space between padding and Margin.
- **Padding** - padding which means the space between content and the border.
- **Content** – It is the content which is user given.



- **Transform**- Which means transformation to an element.
    - The **transform property** in CSS is used to change the coordinate space of the visual formatting model. This is used to add effects like skew, rotate, translate, etc on elements.
        - **Rotate**-It will rotate in clockwise and also in anti-clockwise.
        - **Skew**- To change the angle of an element.
            - **skewX(angle)**: It specifies the skew transformation along with the X-axis corresponding to the skew angle.
            - **skewY(angle)**: It specifies the skew transformation along with the Y-axis corresponding to the skew angle.
            - **skewZ(angle)**: It specifies the skew transformation along with the Z-axis corresponding to the skew angle.
        - **Scale**- The scale which helps to increase or decrease size of an element.
- **Transition-** which means setting a time for an html.
    - **Transition duration**
    - **Transition delay**
    - **Transition timing function**
        - **is**
        - **is-in**
        - **is-out**
        - **is-in-out**
- **ANIMATION :-**
    - The animation-fill-mode property is used to specify that values which are applied by the animation before and after it is executing. Before playing the first keyframe or after playing the last keyframe CSS animations do not affect the element. The animation-fill-mode property can override this behavior.


Syntax:

```
animation-fill-mode: none | forwards | backwards | both | initial |
inherit;
```

- **none:** It is the default value. The animation properties will not apply to any element before or after it is executed.
- **forwards:** The element will retain the same animation properties of the last keyframe after the animation completes.
- **backwards:** This property value is used to set the element to the first keyframe value before start the animation.
- **both:** This property is used to follow the rules for both forwards and backwards.
- **initial:** This property is used to set the property to its default value.
- **inherit:** This property is used to inherits this property from its parent element.

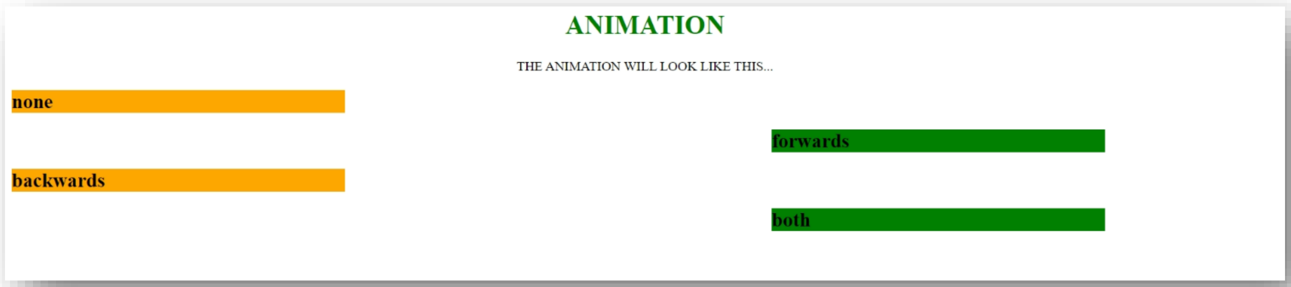
(IN HTML) :-

```html
<body>
     <h1>ANIMATION</h1>
     <p>THE ANIMATION WILL LOOK LIKE THIS...</p>
     <h2 id="one">none</h2>
     <h2 id="two">forwards</h2>
     <h2 id="three">backwards</h2>
     <h2 id="four">both</h2>
</body>
```

(IN CSS) :-

```css
h1 {
    color:green;
}
h1, p {
    text-align:center;
}
h2 {
    width: 400px;
    background-color: orange;
    animation-name: text;
    animation-duration: 3s;
}
#one {
    animation-fill-mode: none;
}
#two {
    animation-fill-mode: forwards;
}
#three {
    animation-fill-mode: backwards;
    animation-delay: 2s;
}
#four {
    animation-fill-mode: both;
    animation-delay: 2s;
}
@keyframes text {
    from {
        margin-left: 0%;
        background-color: #aaaaaa;
    }
    to {
        margin-left: 60%;
        background-color: #008000;
    }
}
```

✓ **OUTPUT :-**


ANIMATION

THE ANIMATION WILL LOOK LIKE THIS...

none

forwards

backwards

both

JAVASCRIPT

- ❖ **Consol.log()-** It will print on console(ctr+shift+I/J)
- ❖ **Alert()-** Alert which is providing a massage for word
    - ▪ In alert we can see only OK button.
- ❖ **Confirm()-** Getting confirmation from the user.
    - ▪ In confirm, you see 2 button i.e.-OK and Cancel
- ❖ **Document.write()-** which is helps to print an output on UI (user
- ❖ **Document.writeln()-** which is helps to print an output on a word it will provide space between each word

❖ VAR -: Declaration can be done.

```
E.g.- // Declaration

var a ;
console.log(a);
```

- Initialization can be done.

```
// Declaration
var a ;
console.log(a);
// intialization
var b= 20
console.log(b);

// declartion and reintialization
var c = 30;
console.log(c);

var d = "sudesh";
d="sudesh gowda";
console.timeLog(d);

var e = "hello";
var e = "bye";
console.log(e);

console.log(f);
var f = "i am hosting"
```

- LET-: Declaration can be done.

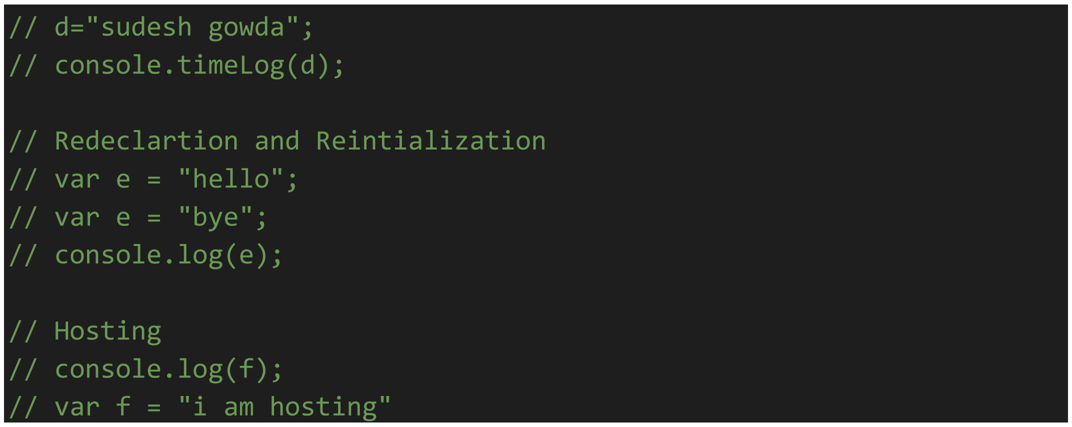
  E.g-:

```
// Declaration
var a ;
console.log(a);

// intialization
var b= 20
console.log(b);

// declartion and intialization
var c = 30;
console.log(c);

// Reintialization
var d = "sudesh";
d="sudesh gowda";
console.timeLog(d);

// Redeclartion and Reintialization
// var e = "hello";
// var e = "bye";
// console.log(e);

// Hosting
console.log(f);
var f = "i am hosting"
```

- CONST-: Only declaration and initialization can be done

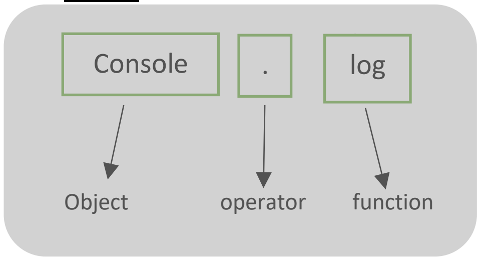
  E.g-:

```
// Declaration
// var a ;
// console.log(a);

// intialization
// var b= 20
// console.log(b);

// declartion and intialization
var c = 30;
console.log(c);

// Reintialization
// var d = "sudesh";
```



```
// d="sudesh gowda";
// console.timeLog(d);

// Redeclartion and Reintialization
// var e = "hello";
// var e = "bye";
// console.log(e);

// Hosting
// console.log(f);
// var f = "i am hosting"
```

NOTE :-



DATA TYPES

PRIMITIVE :-

1) String
2) Number
3) Boolean
4) Undefined
5) null

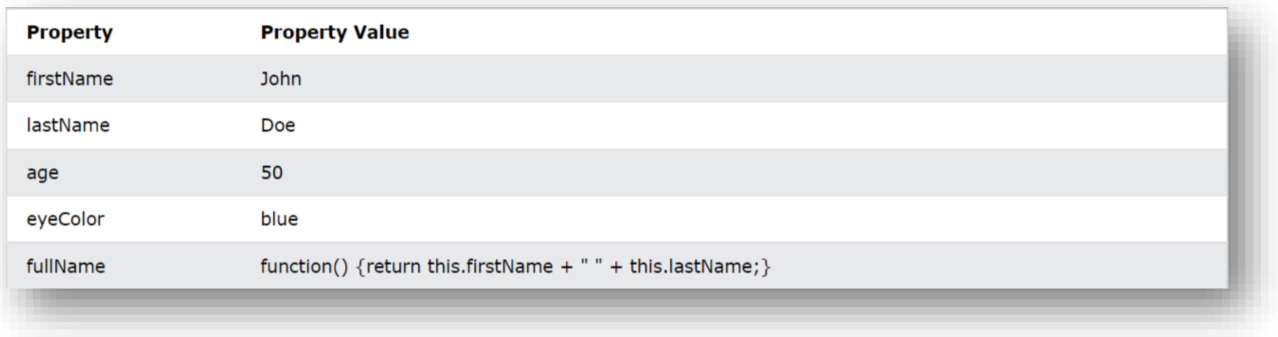
NON-PRIMITIVE :-

1) Function
2) Array
3) Object
4) Math
5) Date

- **Data type-:** It is a type of data
- **String -:** String is a primitive data type which is enclosed with single and double quote (‘_’ , “_”)

❖ Object methods :-

- Objects can also have **methods**.
- Methods are **actions** that can be performed on objects.
- Methods are stored in properties as **function definitions**.



Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue
fullName	function() {return this.firstName + " " + this.lastName;}

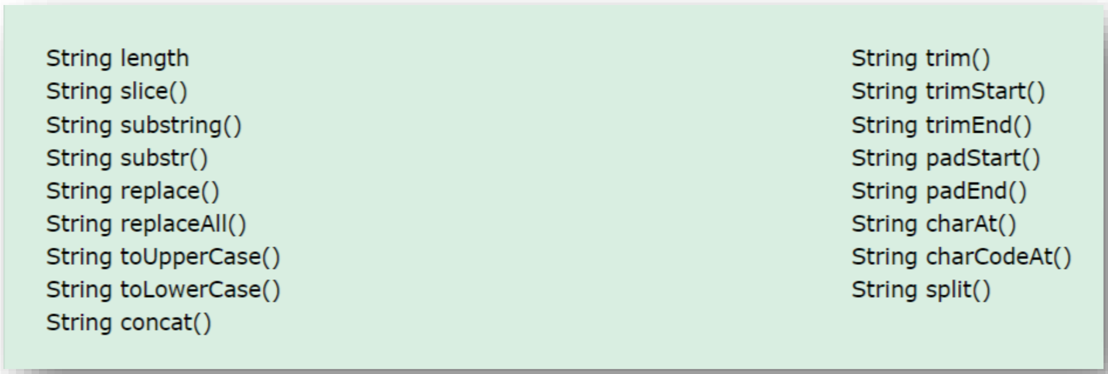
❖ String Methods :-

```
E.g.-   let a = "I'm Dinesh";

console.log(a);
console.log(typeof a);

let b ='I"m Dinesh'
console.log(a);
console.log(typeof b);
```

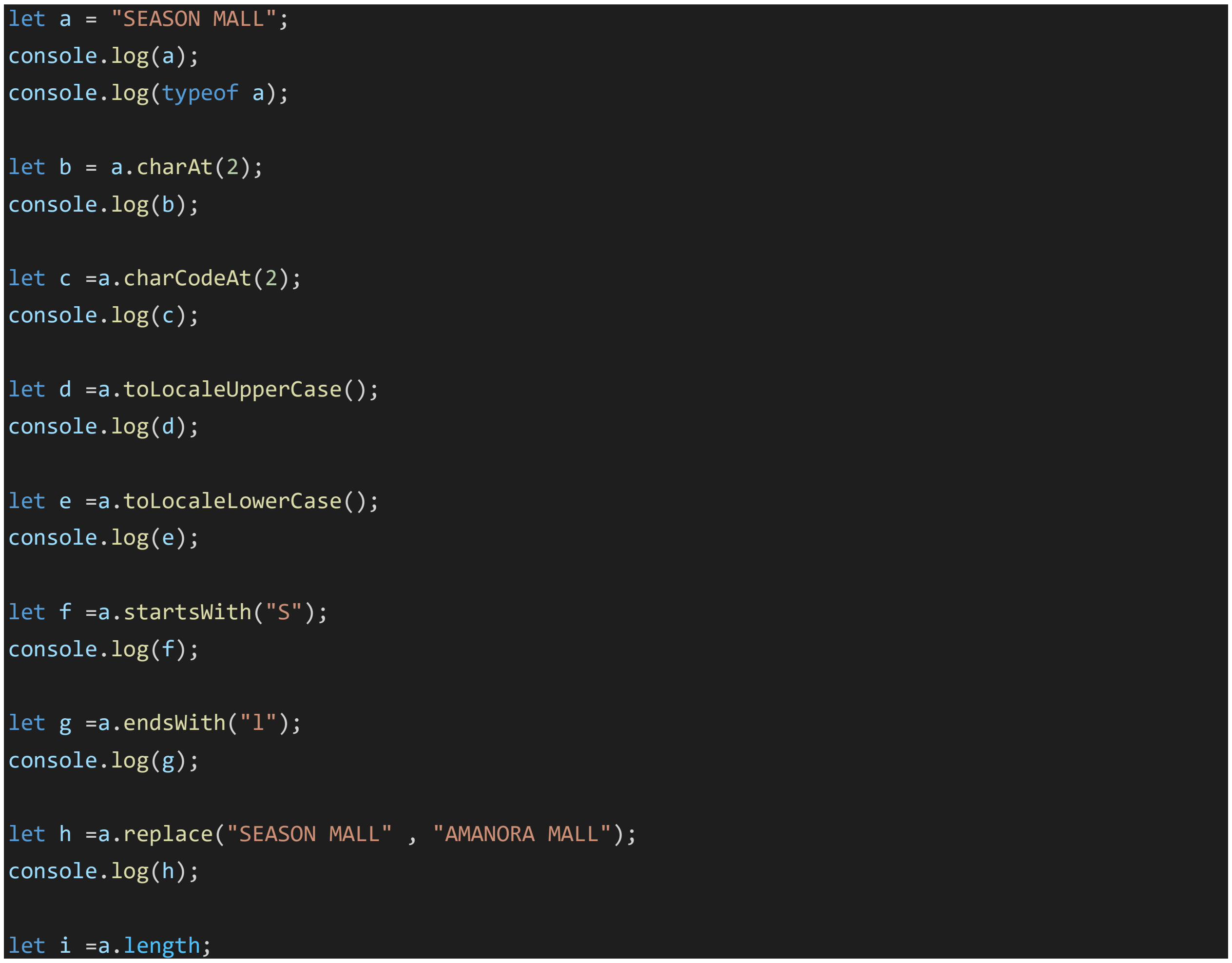
```
let a =
`I'm Dinesh
I'm form pune
I'm leaning javascript`
console.log(a);
```

➢ **${ _ _ _ _ } -: It is for calling the let.**

```

let name="Dinesh";
let id="Ty123";
let place="Pune";
    console.log(`I'm ${name}
                My id is ${id}
                My working place is ${place}`)
```



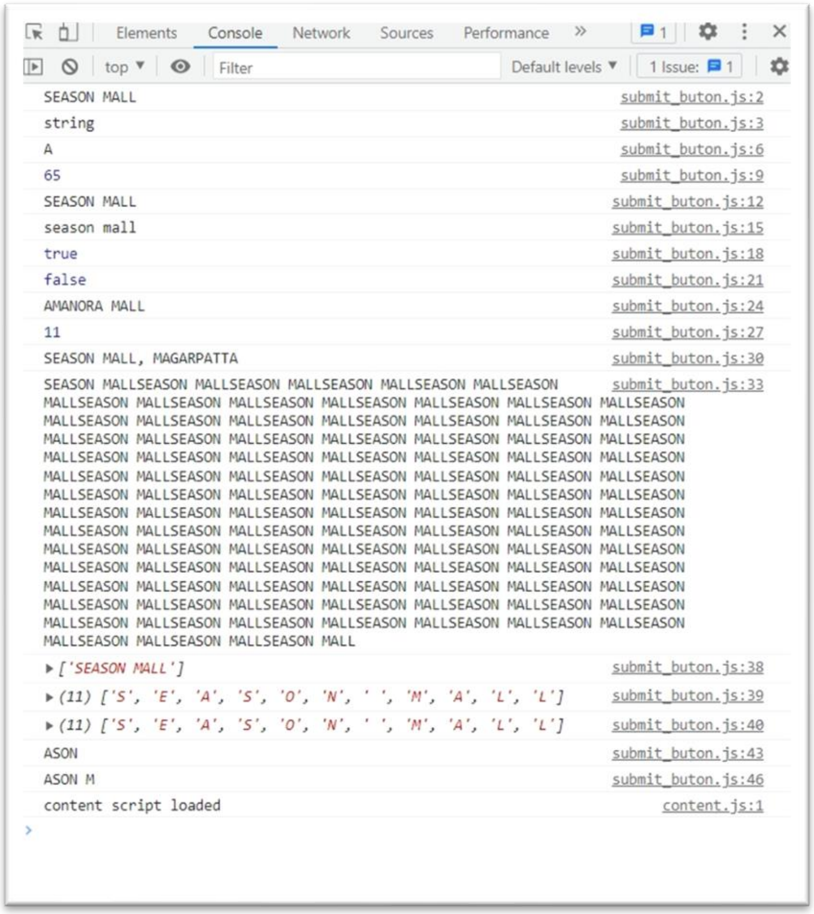
String length
String slice()
String substring()
String substr()
String replace()
String replaceAll()
String toUpperCase()
String toLowerCase()
String concat()
String trim()
String trimStart()
String trimEnd()
String padStart()
String padEnd()
String charAt()
String charCodeAt()
String split()



```
let a = "SEASON MALL";
console.log(a);
console.log(typeof a);

let b = a.charAt(2);
console.log(b);

let c =a.charCodeAt(2);
console.log(c);

let d =a.toLocaleUpperCase();
console.log(d);

let e =a.toLocaleLowerCase();
console.log(e);

let f =a.startsWith("S");
console.log(f);

let g =a.endsWith("l");
console.log(g);

let h =a.replace("SEASON MALL" , "AMANORA MALL");
console.log(h);

let i =a.length;
```

```
console.log(i);

let k =a.concat(", MAGARPATTA");
console.log(k);

let l = a.repeat(100);
console.log(l);

let m = a.split();
let n = a.split("");
let o = a.split('');
console.log(m);
console.log(n);
console.log(o);

let p = a.slice(2 , 6);
console.log(p);
let q = a.slice(2 , 8);
console.log(q);
```

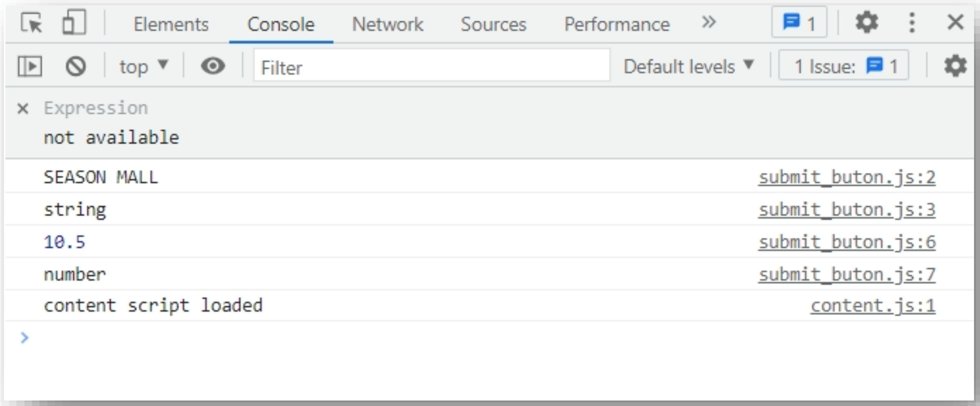
<u>IN Web page (ctr + shift + J/I)</u>



```
SEASON MALL                                                   submit_buton.js:2
string                                                        submit_buton.js:3
A                                                             submit_buton.js:6
65                                                            submit_buton.js:9
SEASON MALL                                                   submit_buton.js:12
season mall                                                   submit_buton.js:15
true                                                          submit_buton.js:18
false                                                         submit_buton.js:21
AMANORA MALL                                                  submit_buton.js:24
11                                                            submit_buton.js:27
SEASON MALL, MAGARPATTA                                       submit_buton.js:30
SEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON submit_buton.js:33
MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON
MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON
MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON
MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON
MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON
MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON
MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON
MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON
MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON
MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON
MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON
MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON
MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON MALLSEASON
MALLSEASON MALLSEASON MALLSEASON MALL
▸ ['SEASON MALL']                                             submit_buton.js:38
▸ (11) ['S', 'E', 'A', 'S', 'O', 'N', ' ', 'M', 'A', 'L', 'L'] submit_buton.js:39
▸ (11) ['S', 'E', 'A', 'S', 'O', 'N', ' ', 'M', 'A', 'L', 'L'] submit_buton.js:40
ASON                                                          submit_buton.js:43
ASON M                                                        submit_buton.js:46
content script loaded                                         content.js:1
>
```

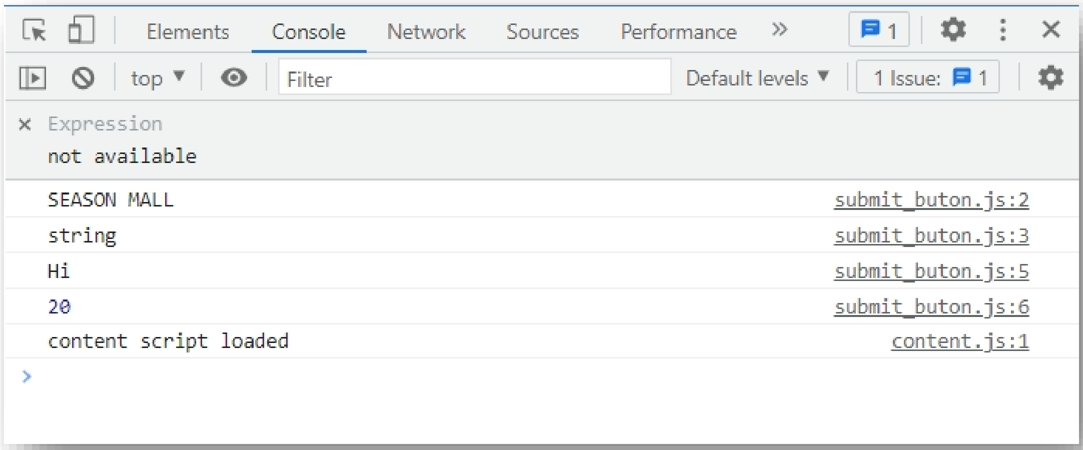
NUMBER :-

```
let a = "SEASON MALL";
console.log(a);
console.log(typeof a);

let b = 10.5;
console.log(b);
console.log(typeof b);
```



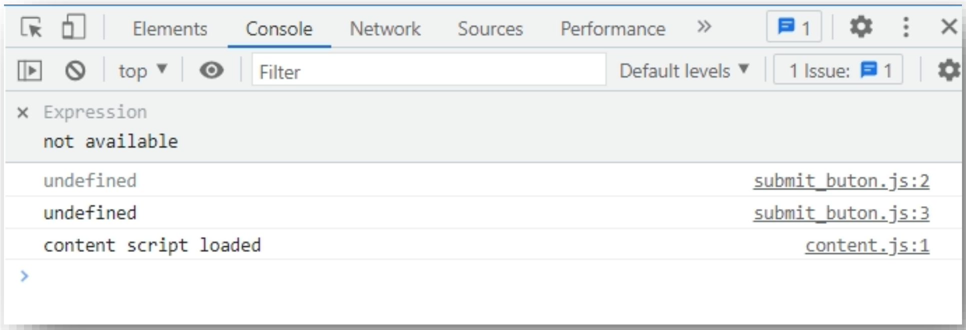
```
× Expression
  not available
  SEASON MALL                                         submit_buton.js:2
  string                                              submit_buton.js:3
  10.5                                                submit_buton.js:6
  number                                              submit_buton.js:7
  content script loaded                                     content.js:1
>
```

```
let a = "SEASON MALL";
console.log(a);
console.log(typeof a);

console.log(true? "Hi":"bye");
console.log(false? 10:20);
```



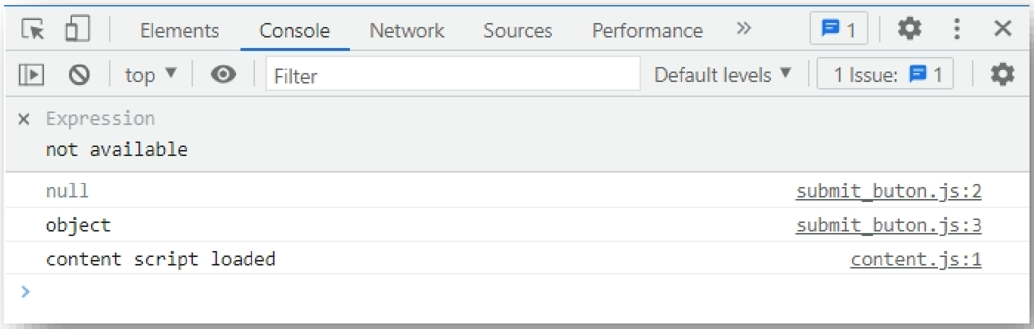
```
× Expression
  not available
  SEASON MALL                                    submit_buton.js:2
  string                                         submit_buton.js:3
  Hi                                             submit_buton.js:5
  20                                             submit_buton.js:6
  content script loaded                               content.js:1
>
```

```
let a = undefined
console.log(a);
console.log(typeof a);
```



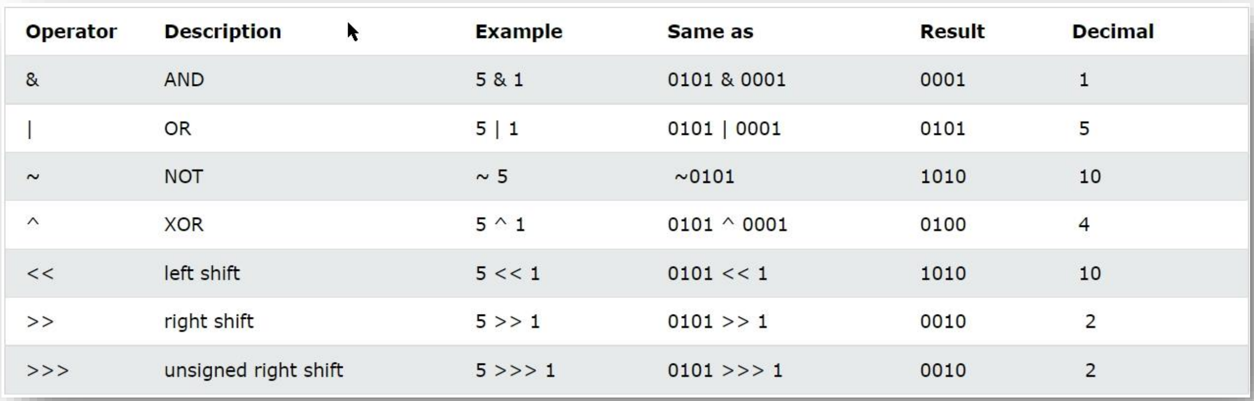
```
× Expression
  not available
  undefined                                      submit_buton.js:2
  undefined                                      submit_buton.js:3
  content script loaded                               content.js:1
>
```

```
let a = null;
console.log(a);
console.log(typeof a);
```



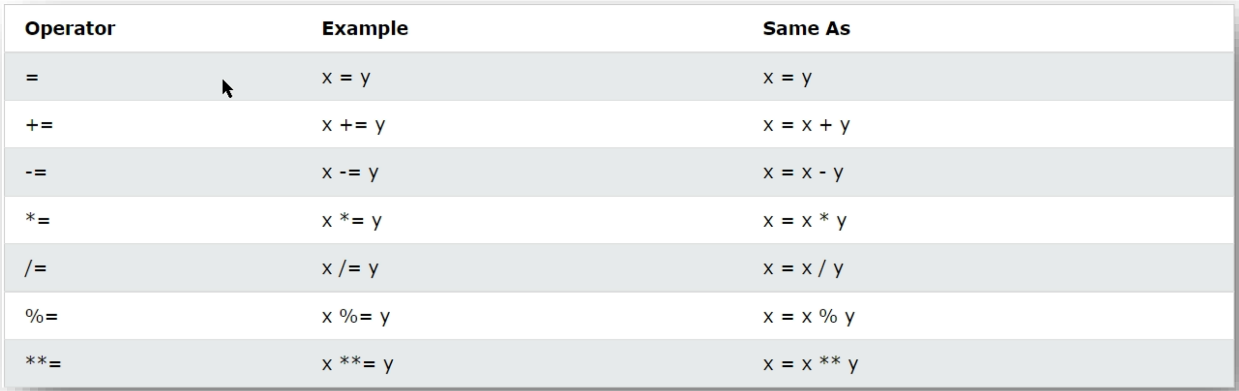
```
× Expression
  not available
  null                                   submit_buton.js:2
  object                                 submit_buton.js:3
  content script loaded                       content.js:1
>
```

❖ Bitwise operator :-



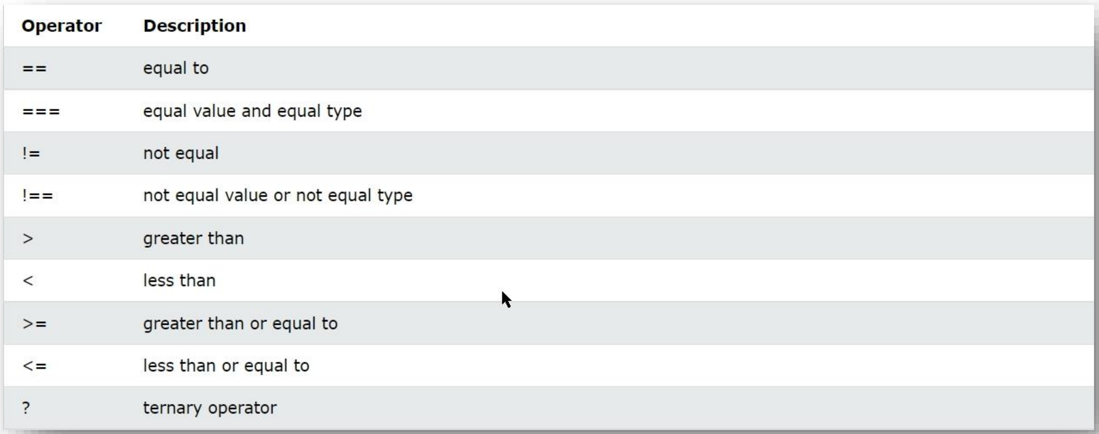
Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	left shift	5 << 1	0101 << 1	1010	10
>>	right shift	5 >> 1	0101 >> 1	0010	2
>>>	unsigned right shift	5 >>> 1	0101 >>> 1	0010	2

❖ Assignment operator :-



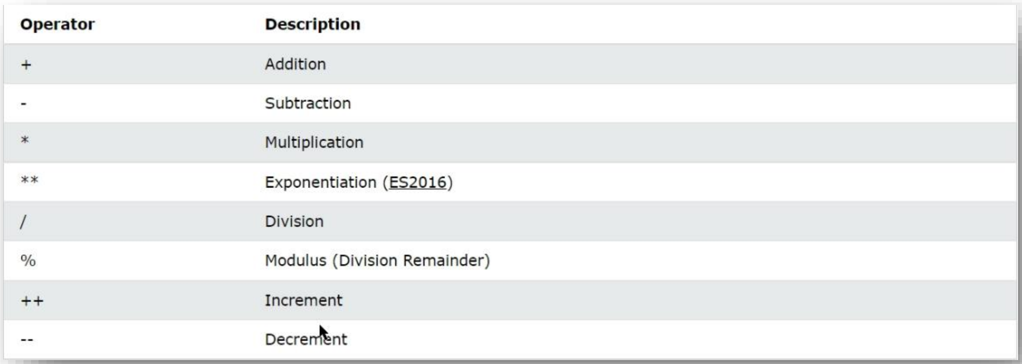
Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

❖ Comparison operator :-



Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

❖ Arithmetic operator :-



Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

❖ **NON-PRIMITIVE =>**

- **Function :-** Function which means in a block a set of code perform particular task.
  - **Named function :**
    Syntax-:

```
function function-Name(){

        //set of code
        }
        function name()
```

```
function add(a,b){
    console.log(`I'm doing addition`);
    console.log(`${a+b}`);
}
add(10,20)
```

  - **Anonymous function :**

```
function (){
    console.log(`I'm annanomous function`);
}
()
```

  - **Immediate involved function:-** Immediate involved function perform a task immediatly & this reduces the code.

```
(function (){
    console.log(`I'm immidiate involved function`);
})
()
```

  - **Function with expression** :-In function with expression the function is passing as an value to achieve anonymous function we are a declaring a variable.

```
let add=function (a,b){
    console.log(`I'm function with expression`);
    console.log(`${a+b}`);
}
add(10,20)
```

- **Arrow Function :-**

```
let a = ()=>{
    console.log(`I'm Arrow function`);

}
a()
```

If you want to print only hello then –

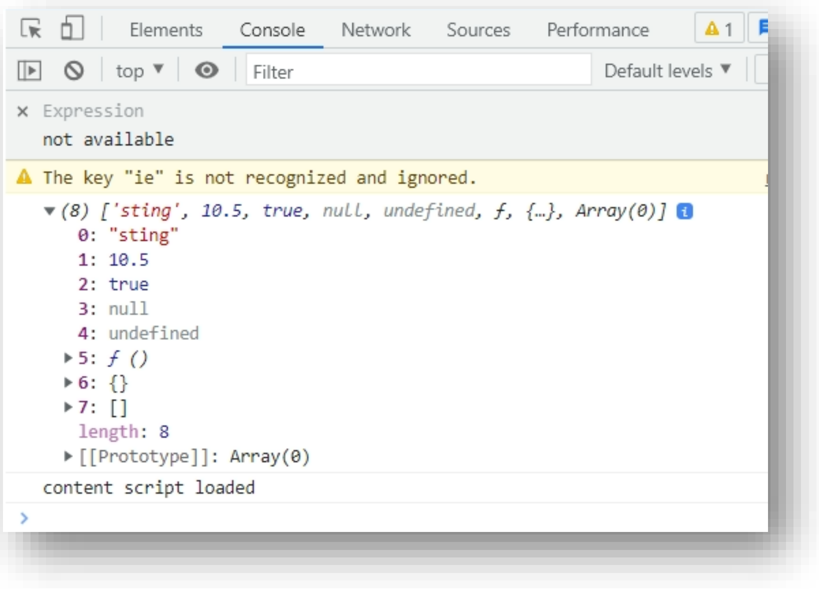
```
let b = ()=>"Hello"
    console.log(b());

b()
```

NOTE : In arrow function this keyword will not work in arrow function

❖ ARRAY :-

Array is a collection of homogeneous and heterogeneous data types.

```
let a = ["sting",10.5,true,null,undefined,function(){},{},[]]
console.log(a);
```



```
× Expression
  not available
⚠ The key "ie" is not recognized and ignored.
  ▼(8) ['sting', 10.5, true, null, undefined, ƒ, {…}, Array(0)]
     0: "sting"
     1: 10.5
     2: true
     3: null
     4: undefined
   ▶5: ƒ ()
   ▶6: {}
   ▶7: []
     length: 8
   ▶[[Prototype]]: Array(0)
  content script loaded
>
```

❖ NOTE :-

- If you use named indexes, JavaScript will redefine the array to an object.
- After that, some array methods and properties will produce **incorrect results**.

❖ Array methods :-

- ## Popping and pushing :-

  - When you work with arrays, it is easy to remove elements and add new elements.
  - This is what popping and pushing is:
  - Popping items **out** of an array, or pushing items **into** an array.
  - Push- adding element in the last index
    - **Array pop() :-**
      - The pop() method removes the last element from an array.
      - The push() method adds a new element to an array (at the end)
      - The shift() method removes the first array element and "shifts" all other elements to a lower index.
      - The unshift() method adds a new element to an array (at the beginning), and "unshifts" older elements
      - The length property provides an easy way to append a new element to an array(e.length).

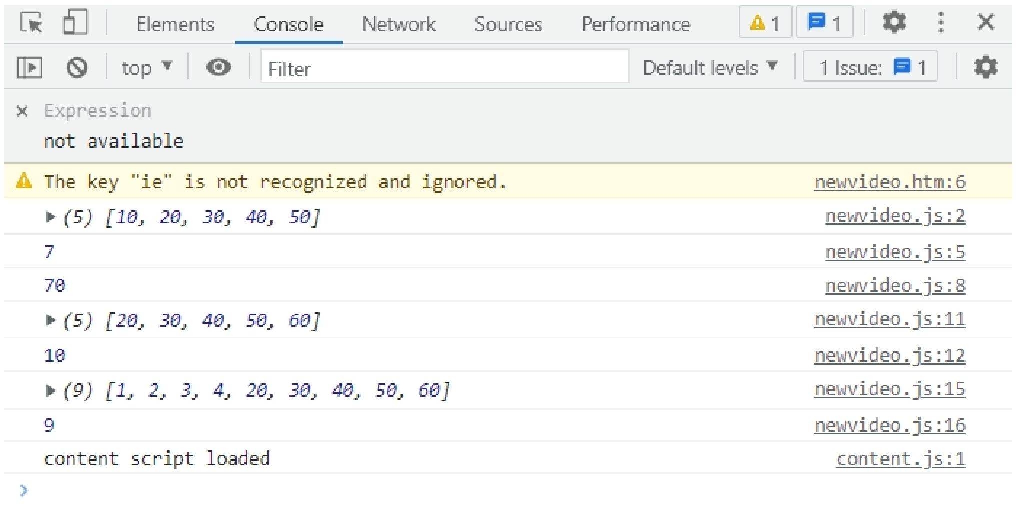
❖ **NOTE :-** Array elements can be deleted using the JavaScript operator `delete`.

Using `delete` leaves `undefined` holes in the array.

Use pop() or shift() instead.

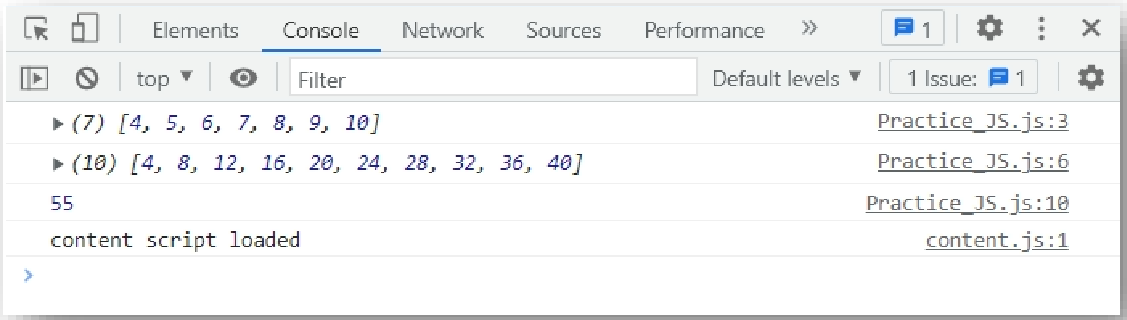
e.g.-

```
let a = [10,20,30,40,50]
console.log(a);

let b =a.push(60,70)
console.log(b);

let c = a.pop()
console.log(c);

let d = a.shift()
console.log(a);
console.log(d);

let e = a.unshift(1,2,3,4)
console.log(a);
console.log(e);
```



```
Expression
not available
⚠ The key "ie" is not recognized and ignored.      newvideo.htm:6
▶ (5) [10, 20, 30, 40, 50]                         newvideo.js:2
7                                                  newvideo.js:5
70                                                 newvideo.js:8
▶ (5) [20, 30, 40, 50, 60]                         newvideo.js:11
10                                                 newvideo.js:12
▶ (9) [1, 2, 3, 4, 20, 30, 40, 50, 60]             newvideo.js:15
9                                                  newvideo.js:16
content script loaded                              content.js:1
```

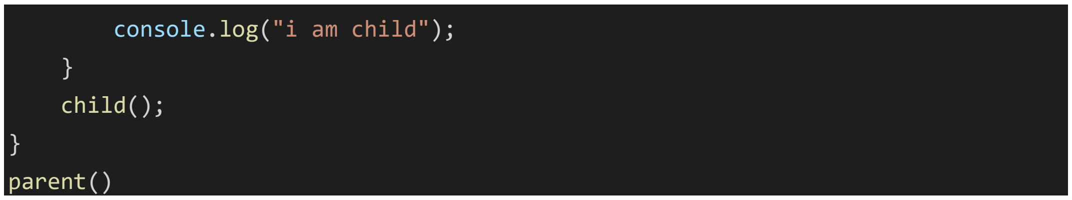
```
let aa = [1,2,3,4,5,6,7,8,9,10]
let bb =aa.filter((m)=>m>3)
console.log(bb);

let cc= aa.map((n)=>n*4);
console.log(cc);

let dd= aa.reduce((firstvalue,lastvalue)=>{
return firstvalue+lastvalue})
console.log(dd);
```



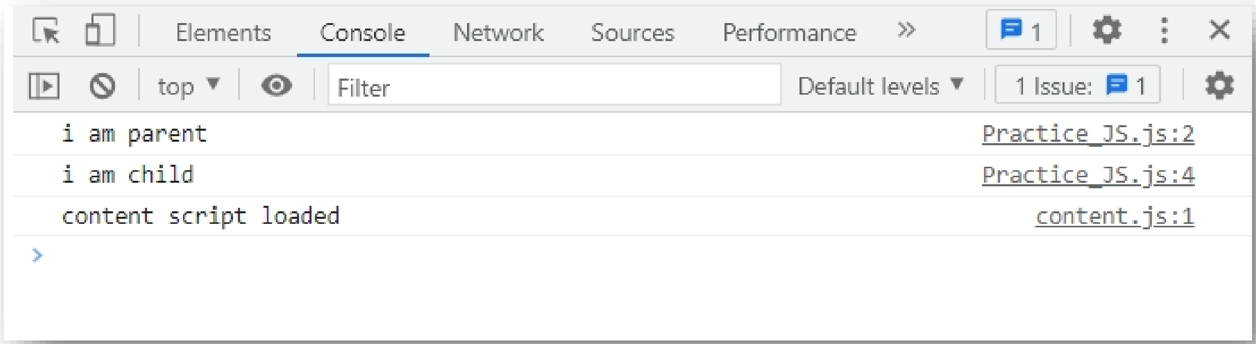
```
▶ (7) [4, 5, 6, 7, 8, 9, 10]                       Practice_JS.js:3
▶ (10) [4, 8, 12, 16, 20, 24, 28, 32, 36, 40]      Practice_JS.js:6
55                                                 Practice_JS.js:10
content script loaded                              content.js:1
```

❖ NESTED FUNCTION :-

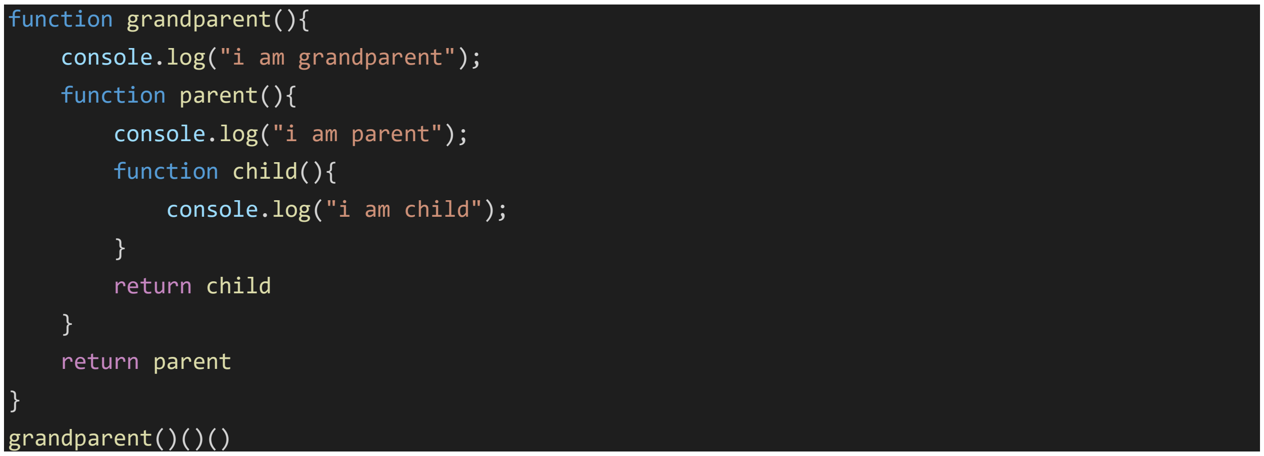
```
function parent(){
    console.log("i am parent");
    function child(){
```



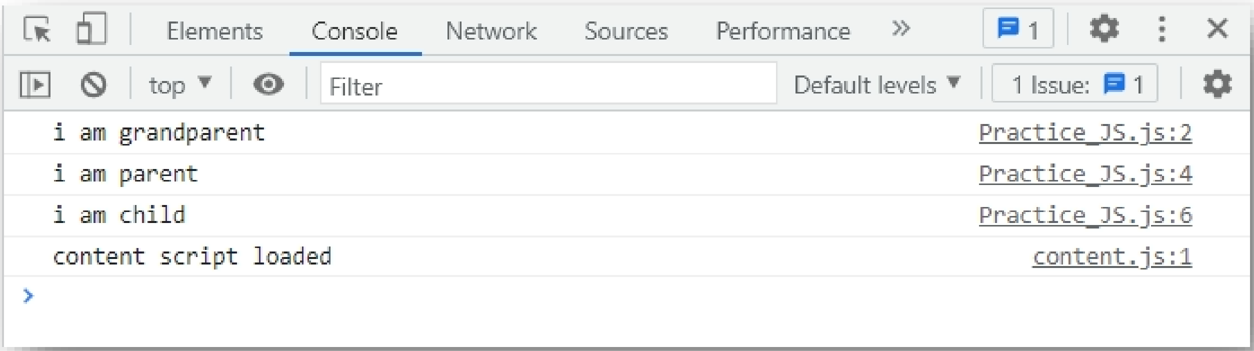
```
        console.log("i am child");
    }
    child();
}
parent()
```



```
i am parent                                    Practice_JS.js:2
i am child                                     Practice_JS.js:4
content script loaded                              content.js:1
>
```



```
function grandparent(){
    console.log("i am grandparent");
    function parent(){
        console.log("i am parent");
        function child(){
            console.log("i am child");
        }
        return child
    }
    return parent
}
grandparent()()()
```



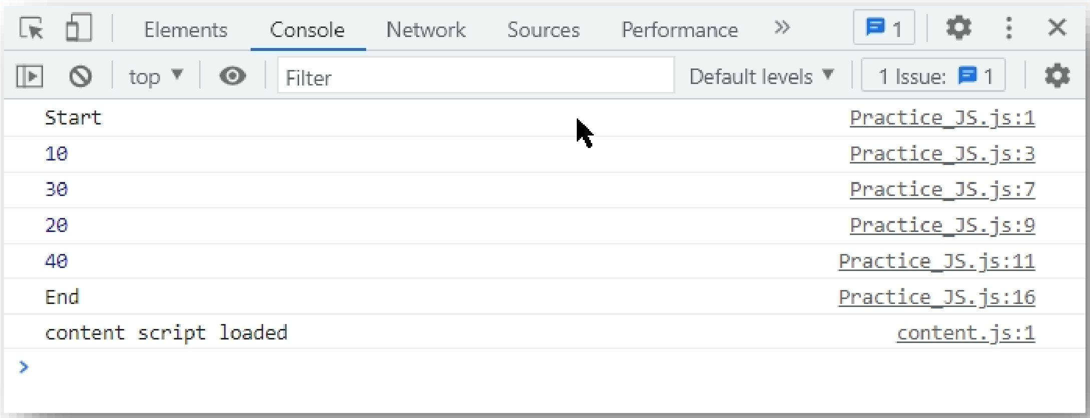
```
i am grandparent                               Practice_JS.js:2
i am parent                                    Practice_JS.js:4
i am child                                     Practice_JS.js:6
content script loaded                              content.js:1
>
```

- **Lexical scope :-** Lexical scope is the ability for a function scope to access a variable from the parent scope.

  We call the child function to be lexically bound by that of the parent function.

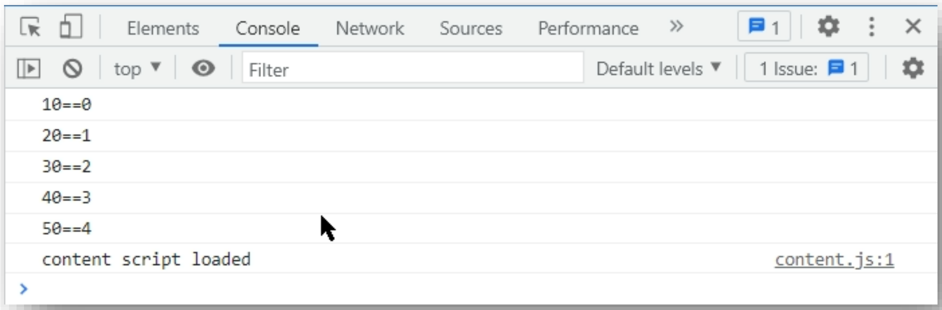
- **Closoure :-** it means holding address of holding address of parent function.
- **Call stack :-** It means last in first out (LIFO).

```
console.log("Start");
var a=10;
console.log(a);
    function x() {
        var b =20;
        var c =30;
        console.log(c);
            function y() {
                console.log(b);
                var d=40;
                console.log(d);
            }
            return y
    }
    x()()
    console.log("End");
```



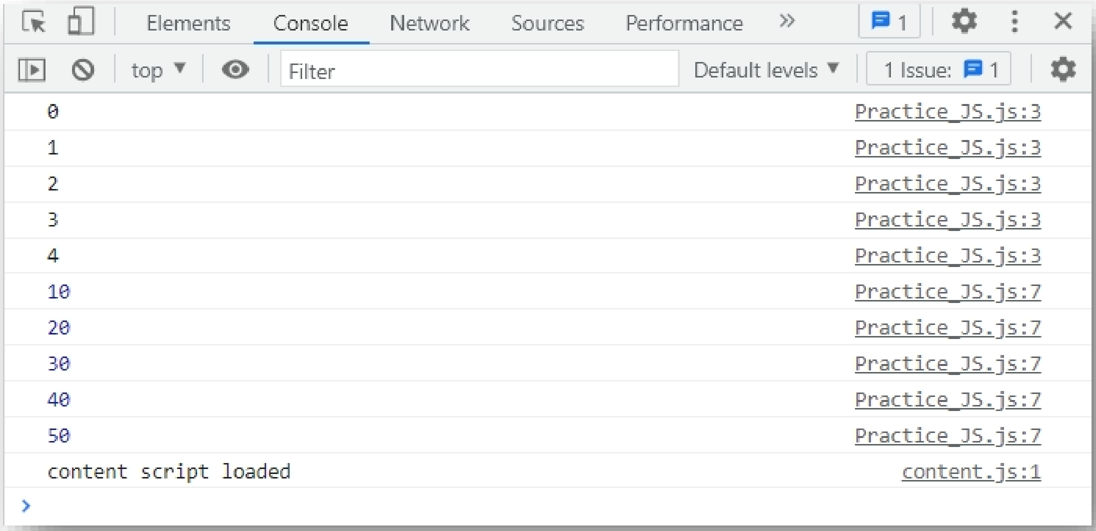
```
Start                                              Practice_JS.js:1
10                                                 Practice_JS.js:3
30                                                 Practice_JS.js:7
20                                                 Practice_JS.js:9
40                                                 Practice_JS.js:11
End                                                Practice_JS.js:16
content script loaded                              content.js:1
>
```

```
 let arr =[10,20,30,40,50];
arr.forEach((val,ind)=>{
    console.log(`${val}==${ind}`);
})
```



```
10==0
20==1
30==2
40==3
50==4
content script loaded                                   content.js:1
```

```
let arr =[10,20,30,40,50];
for(const arr1 in  arr){
    console.log(arr1);
}

for (const arr1 of arr){
    console.log(arr1);
}
```

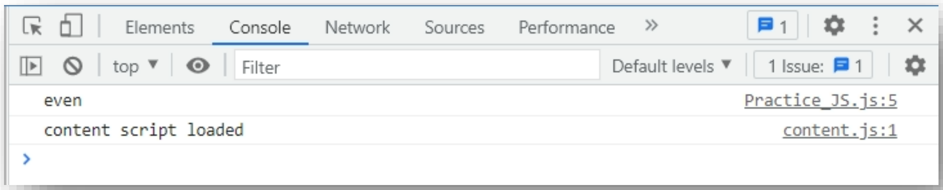


```
0                                                    Practice_JS.js:3
1                                                    Practice_JS.js:3
2                                                    Practice_JS.js:3
3                                                    Practice_JS.js:3
4                                                    Practice_JS.js:3
10                                                   Practice_JS.js:7
20                                                   Practice_JS.js:7
30                                                   Practice_JS.js:7
40                                                   Practice_JS.js:7
50                                                   Practice_JS.js:7
content script loaded                                    content.js:1
```

❖ **<u>CONDITIONS :-</u>**

1. **<u>If</u>**

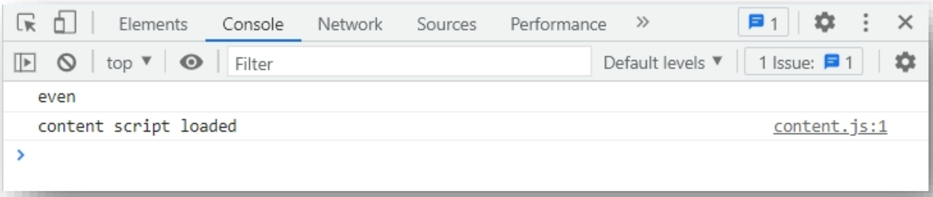
```
let b =3
if (b%2==0) {
    console.log("even");
```



```
even                                   Practice_JS.js:5
content script loaded                  content.js:1
```

2. If else:-

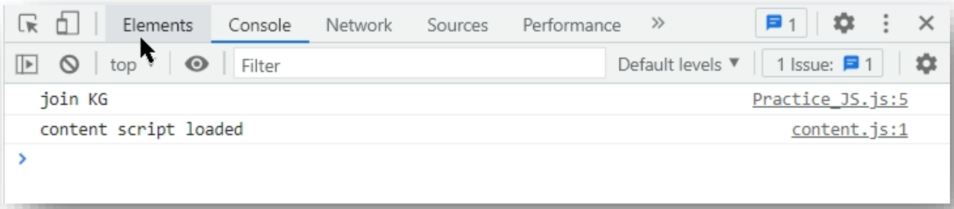
```
let b =3
if (b%2==0) {
    console.log("even");
} else {
    console.log("Odd");
}
```



```
even
content script loaded                  content.js:1
```

3. elseif

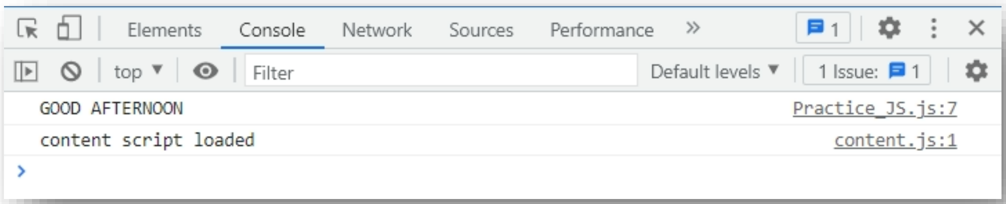
```
let age = prompt("Enter the age")
if (age==4) {
    console.log("join KG");
} else if (age==5) {
    console.log("join KG");

} else {
    console.log("Enter the correct one");
}
```



```
join KG                                Practice_JS.js:5
content script loaded                  content.js:1
```

4. Switch case

```
let time=12;
switch (time) {
    case 6:
        console.log("GOOD MORNING");
        break;
    case 12:
        console.log("GOOD AFTERNOON");
         break;
    case 118:
        console.log("GOOD EVENING");
        break;

    default:
        console.log("ENTER THE CORRECT ONE");
        break;
}
```



```
GOOD AFTERNOON                                    Practice_JS.js:7
content script loaded                                 content.js:1
```

❖ LOOPS :-

- Loops are handy, if you want to run the same code over and over again, each time with a different value.

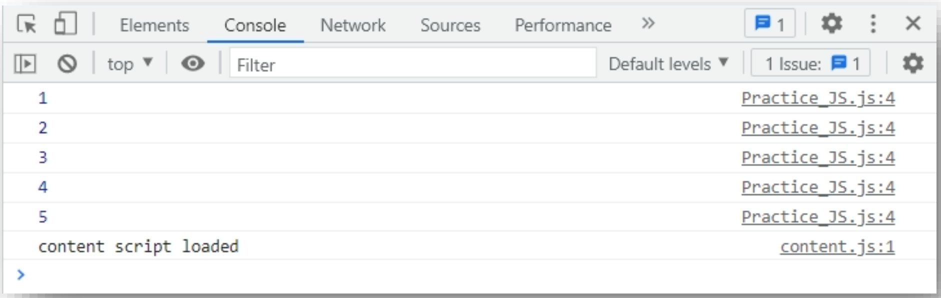
- Often this is the case when working with arrays.

❖ JavaScript supports different kinds of loops:

- `for` - loops through a block of code a number of times.
- `for/in` - loops through the properties of an object.
- `for/of` - loops through the values of an iterable object.
- `while` - loops through a block of code while a specified condition is true.
- `do/while` - also loops through a block of code while a specified condition is true.

• While Loop :-

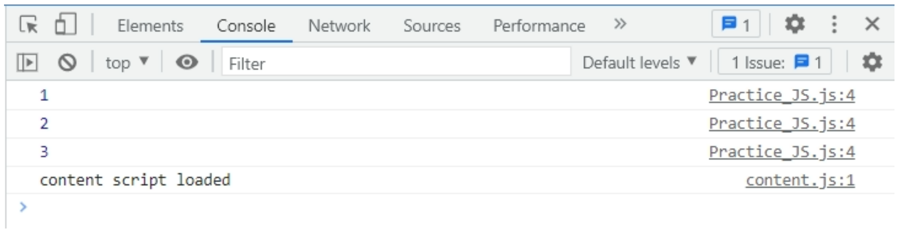
```
let i =1;
let n = 5;
while (i<=5) {
    console.log(i);
    i++;
}
```



```
1                                    Practice_JS.js:4
2                                    Practice_JS.js:4
3                                    Practice_JS.js:4
4                                    Practice_JS.js:4
5                                    Practice_JS.js:4
content script loaded                content.js:1
>
```

- **Do while loop:-**

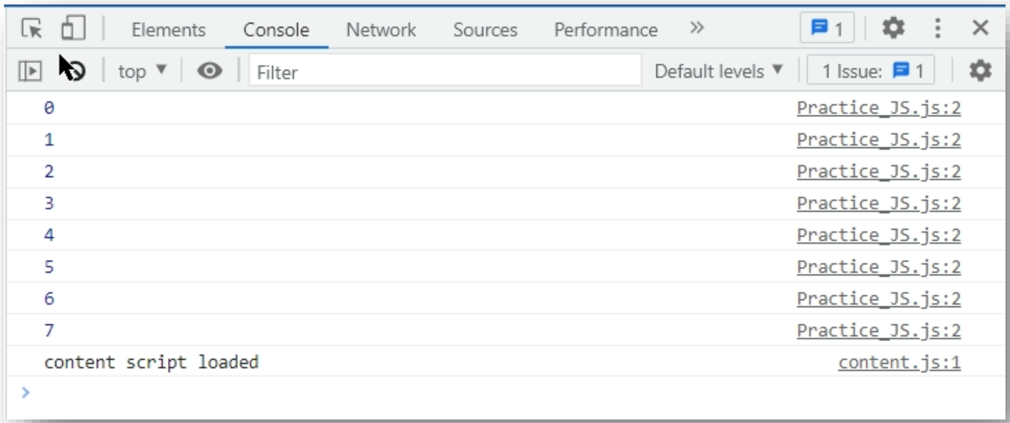
```
let i = 1;
let n = 5;
do {
    console.log(i);
    i++;
} while (i<=n);
```



```
1                                    Practice_JS.js:4
2                                    Practice_JS.js:4
3                                    Practice_JS.js:4
content script loaded                content.js:1
>
```

- **For loop**

```
for (let i = 0; i <= 7; i++) {
  console.log(i);
}
```



```
0                                    Practice_JS.js:2
1                                    Practice_JS.js:2
2                                    Practice_JS.js:2
3                                    Practice_JS.js:2
4                                    Practice_JS.js:2
5                                    Practice_JS.js:2
6                                    Practice_JS.js:2
7                                    Practice_JS.js:2
content script loaded                content.js:1
>
```

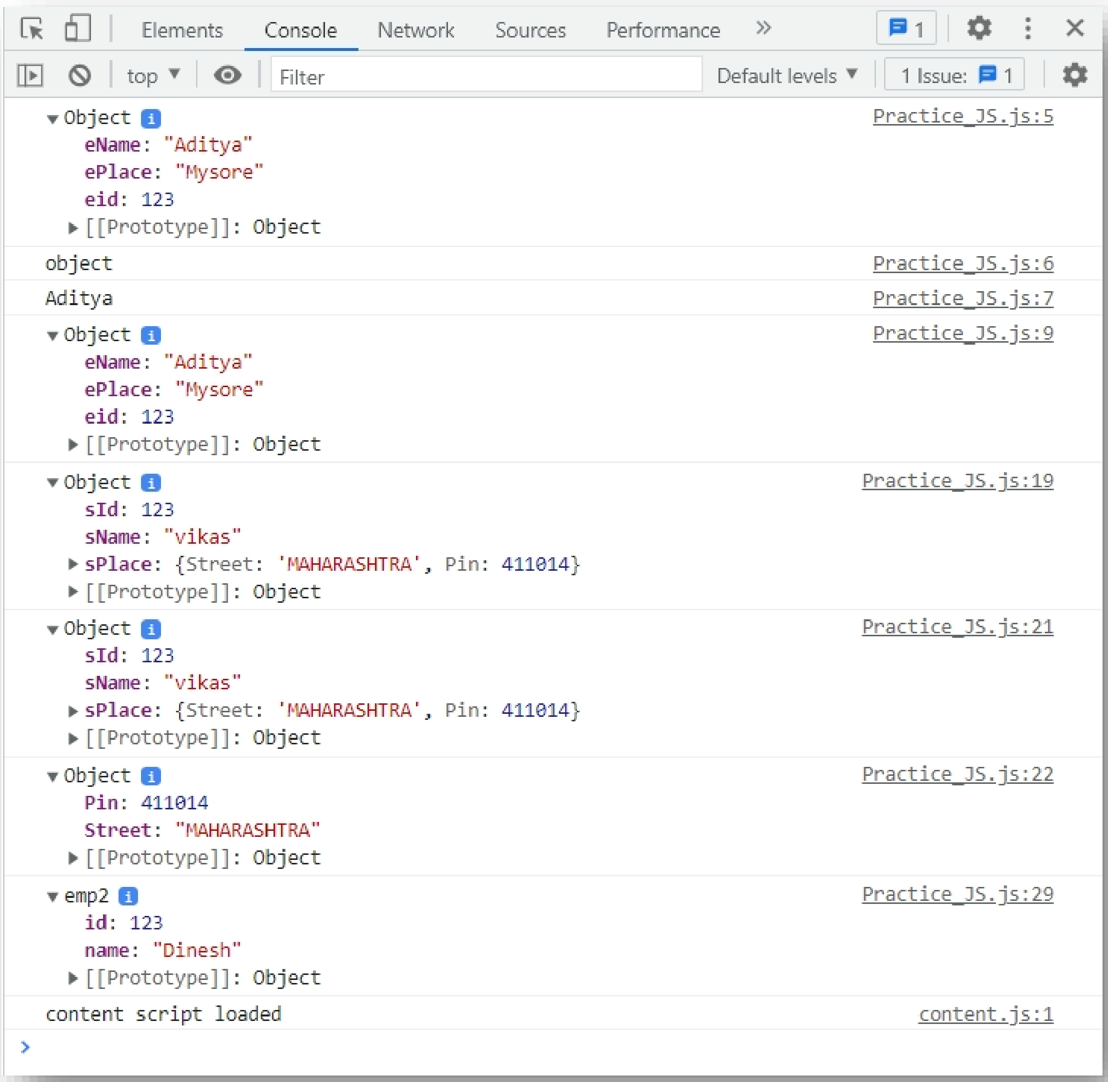
OBJECT :-

Object is combination of key and pair which enclosed with literals ({}).

We can create object in two ways.

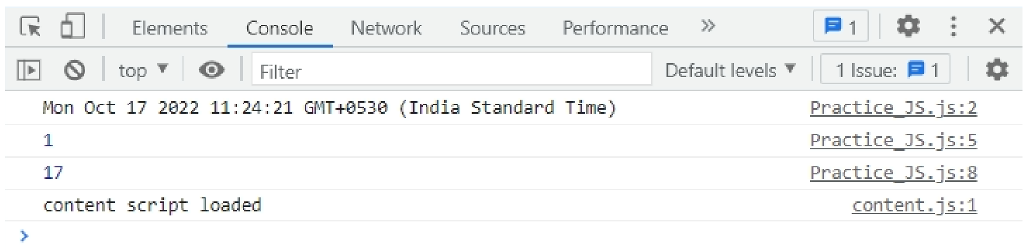
1) **Literals :** *Literals* are the constant values assigned to the constant variables.
2) **Constructer function :** A constructor gets called when an object is created using the new keyword. The purpose of a constructor is to create a new object and set values for any existing object properties.

```
let Emp={
    eName : "Aditya" ,
    eid : 123
}
console.log(Emp);
console.log(typeof Emp);
console.log(Emp.eName);
Emp.ePlace="Mysore"
console.log(Emp);

let stu = {
    sName:"vikas",
    sId:123,
    sPlace:{
        Street:"MAHARASHTRA"
    }
}
console.log(stu);
stu.sPlace.Pin=411014
console.log(stu);
console.log(stu.sPlace);

function emp2(name,id) {
    this.name=name
    this.id=id
}
let e1 = new emp2("Dinesh",123)
console.log(e1);
```

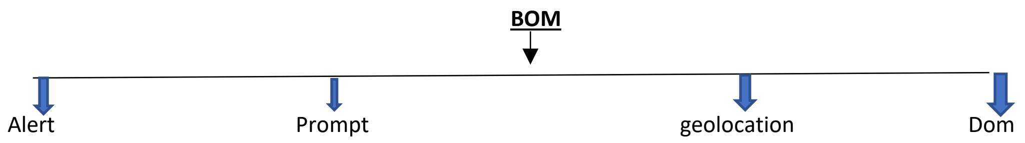


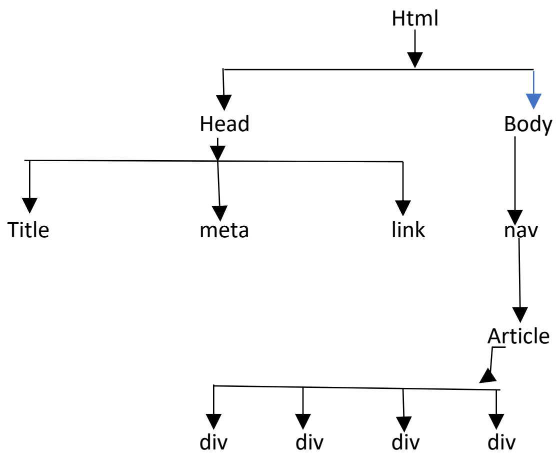
```
▼Object                                          Practice_JS.js:5
   eName: "Aditya"
   ePlace: "Mysore"
   eid: 123
  ▶[[Prototype]]: Object
object                                           Practice_JS.js:6
Aditya                                           Practice_JS.js:7
▼Object                                          Practice_JS.js:9
   eName: "Aditya"
   ePlace: "Mysore"
   eid: 123
  ▶[[Prototype]]: Object
▼Object                                         Practice_JS.js:19
   sId: 123
   sName: "vikas"
  ▶sPlace: {Street: 'MAHARASHTRA', Pin: 411014}
  ▶[[Prototype]]: Object
▼Object                                         Practice_JS.js:21
   sId: 123
   sName: "vikas"
  ▶sPlace: {Street: 'MAHARASHTRA', Pin: 411014}
  ▶[[Prototype]]: Object
▼Object                                         Practice_JS.js:22
   Pin: 411014
   Street: "MAHARASHTRA"
  ▶[[Prototype]]: Object
▼emp2                                           Practice_JS.js:29
   id: 123
   name: "Dinesh"
  ▶[[Prototype]]: Object
content script loaded                               content.js:1
>
```

```
let a = new Date()
console.log(a);

let b = a.getDay()
console.log(b);

let c = a.getDate()
console.log(c);
```



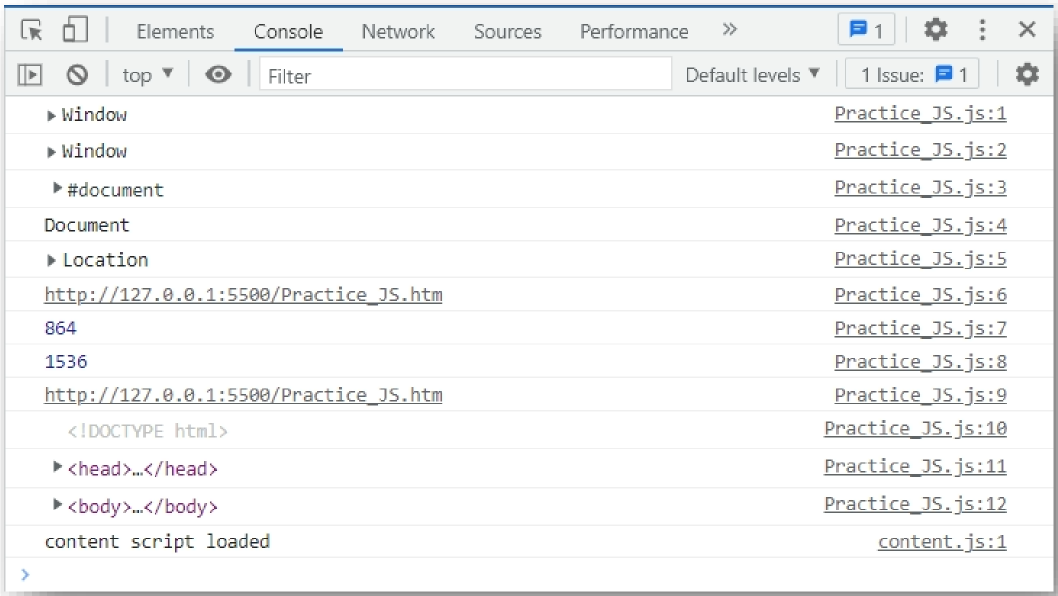
```
Mon Oct 17 2022 11:24:21 GMT+0530 (India Standard Time)      Practice_JS.js:2
1                                                            Practice_JS.js:5
17                                                           Practice_JS.js:8
content script loaded                                        content.js:1
>
```





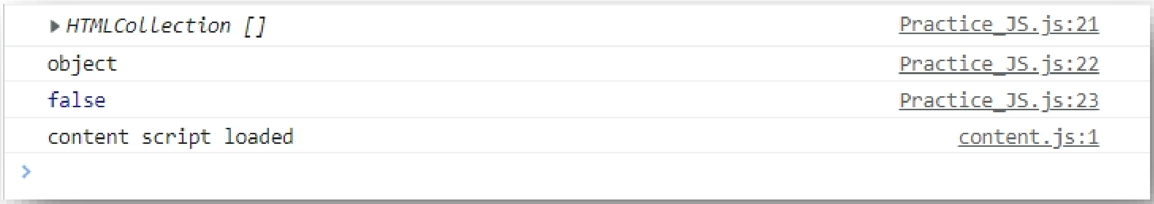
❖ **Dom - :**

- Dom is child of an **Bom (Browser object model).**
- Inside **bom**, we can find N no. of key and value pair and that is called object.
- Dom is a document object model. It describe Html page.



```
▸ Window                                          Practice_JS.js:1
▸ Window                                          Practice_JS.js:2
 ▸ #document                                      Practice_JS.js:3
Document                                          Practice_JS.js:4
▸ Location                                        Practice_JS.js:5
http://127.0.0.1:5500/Practice_JS.htm             Practice_JS.js:6
864                                               Practice_JS.js:7
1536                                              Practice_JS.js:8
http://127.0.0.1:5500/Practice_JS.htm             Practice_JS.js:9
   <!DOCTYPE html>                               Practice_JS.js:10
 ▸ <head>…</head>                                Practice_JS.js:11
 ▸ <body>…</body>                                Practice_JS.js:12
content script loaded                                 content.js:1
>
```

```
let b = document.getElementsByClassName('pc1');
console.log(b);
console.log(typeof b);
console.log(Array.isArray(b));
```



```
▸ HTMLCollection []                              Practice_JS.js:21
object                                           Practice_JS.js:22
false                                            Practice_JS.js:23
content script loaded                                 content.js:1
>
```

```
<section id="sec"></section>
    <script src="./Practice_JS.js"></script>
```

```
let a = document.getElementById('sec');
for(i=0;i<3;i++){
a.innerHTML += `<h1>hello</><br><br>
<img src="./images/heart-ge5827f0a4_1920.png" alt="" height="100px"
width="100px">
`
}
```

```
<section id="sec"></section>
    <script src="./Practice_JS.js"></script>
```

```
let a = document.getElementById('sec')
a.innerHTML+=`<table>
<tr>
    <th>NAME</th>
    <th>COMPANY</th>
</tr>
<tr>
    <td>Dinesh K.</td>
    <td>GOOGLE</td>
</tr>
<tr>
    <td>ABC</td>
    <td>abc</td>
</tr>
</table>`
```

```
<button onclick="clicked()">RED</button>
 <button onclick="clickedd()">WHITE</button>
 <br><br>
 <input type="text" onkeydown="keydown()" onkeyup="keyup()">
 <br><br>
 <button onmouseover="mouseover()" onmouseout="mouseout()">MOUSEOVER AND
OUT</button>
    <script src="./Practice_JS.js"></script>
```

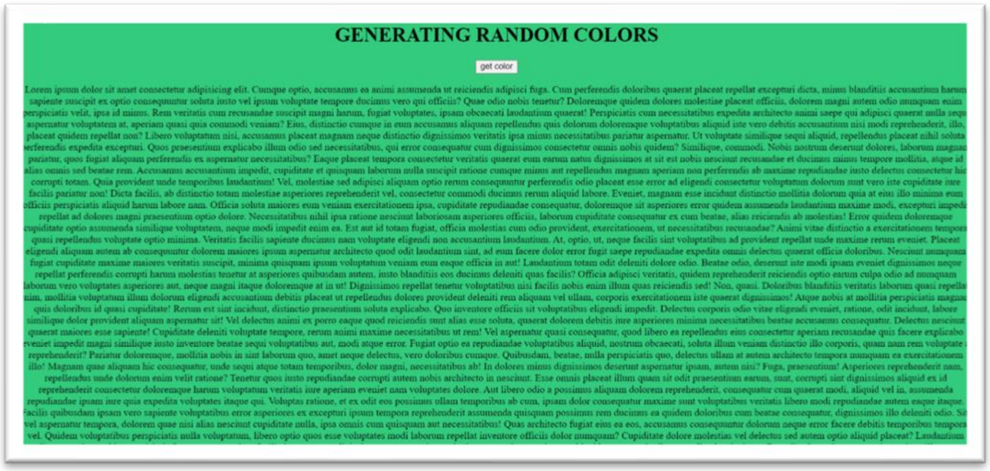
```
function clicked() {
    document.body.style.backgroundColor="red"
}

function clickedd() {
    document.body.style.backgroundColor="white"
}

function keydown() {
    document.body.style.backgroundColor="white"
}
function keyup() {
    document.body.style.backgroundColor="black"
}

function mouseover() {
    document.body.style.backgroundColor="GREEN"
}
function mouseout() {
    document.body.style.backgroundColor="WHITE"
}
```

```

 <button onclick="clicked()">BLACK</button>
 <button onclick="clickedd()">WHITE</button>
 <br><br>
 <p>Lorem1000
      </p>

 <script src="./Practice_JS.js"></script>
```

```
function clicked() {
    document.body.style.backgroundColor="black"
    document.body.style.color="white"
}

function clickedd() {
    document.body.style.backgroundColor="white"
    document.body.style.color="black"
}

```

```
    <h1>GENERATING RANDOM COLORS</h1>
 <button onclick="generateRandomColor()">get color</button>
<P>lorem1000</P>
<script>
    function generateRandomColor() {
        let arr = [1,2,3,4,5,6,7,8,9,"a","b","c","d","e","f"];
        let randomColor = "#";
        for (let i = 1; i <=6; i++) {
            let index = Math.floor(Math.random()*16)
            randomColor +=arr[index];=
        }
        console.log(randomColor);
        document.getElementsByTagName("body")[0].style.backgroundColor =
randomColor;
    }
</script>
<script src="./Practice_JS.js"></script>
```

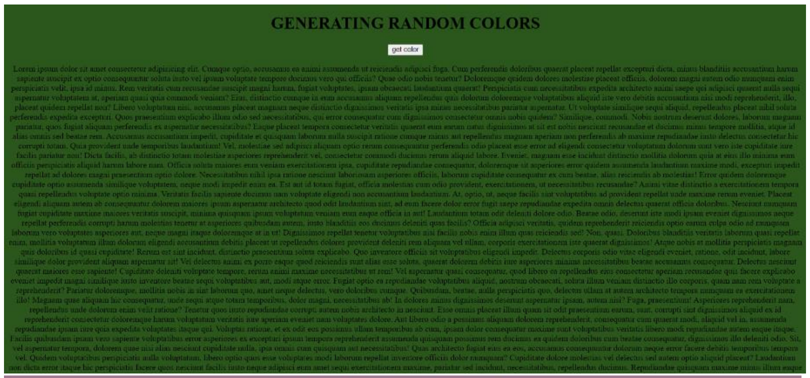
- ✓ **OUTPUT :-**



GENERATING RANDOM COLORS

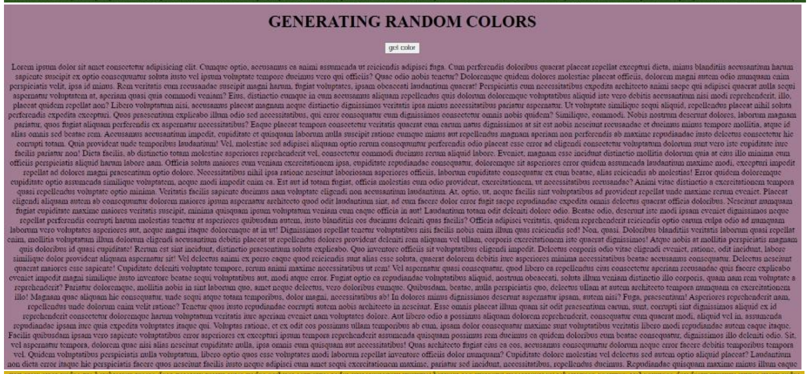
get color

Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque optio, accusamus ea animi assumenda ut reiciendis adipisci fuga. Cum perferendis doloribus quaerat placeat repellat excepturi dicta, minus blanditiis accusantium harum sapiente suscipit ex optio consequuntur soluta iusto vel ipsum voluptate tempore ducimus vero qui officiis? Quae odio nobis tenetur? Doloremque quidem dolores molestiae placeat officiis, dolorem magni autem odio numquam enim perspiciatis velit, ipsa id minus. Rem veritatis cum recusandae suscipit magni harum, fugiat voluptates, ipsam obcaecati laudantium quaerat! Perspiciatis cum necessitatibus expedita architecto animi saepe qui adipisci quaerat nulla sequi aspernatur voluptatem at, aperiam quasi quia commodi veniam? Eius, distinctio cumque in eum accusamus aliquam repellendus quis dolorum doloremque voluptatibus aliquid iste vero debitis accusantium nisi modi reprehenderit, illo, placeat quidem repellat non? Libero voluptatum nisi, accusamus placeat magnam neque distinctio dignissimos veritatis ipsa minus necessitatibus pariatur aspernatur. Ut voluptate similique sequi aliquid, repellendus placeat nihil soluta perferendis expedita excepturi. Quos praesentium explicabo illum odio sed necessitatibus, qui error consequatur cum dignissimos consectetur omnis nobis quidem? Similique, commodi. Nobis nostrum deserunt dolores, laborum magnam pariatur, quos fugiat aliquam perferendis ex aspernatur necessitatibus? Eaque placeat tempora consectetur veritatis quaerat eum earum natus dignissimos at sit est nobis nesciunt recusandae et ducimus minus tempore mollitia, atque id alias omnis sed beatae rem. Accusamus accusantium impedit, cupiditate et quisquam laborum nulla suscipit ratione cumque minus aut repellendus magnam aperiam non perferendis ab maxime repudiandae iusto delectus consectetur hic corrupti totam. Quia provident unde temporibus laudantium! Vel, molestiae sed adipisci aliquam optio rerum consequuntur perferendis odio placeat esse error ad eligendi consectetur voluptatum dolorum sunt vero iste cupiditate iure facilis pariatur non! Dicta facilis, ab distinctio totam molestiae asperiores reprehenderit vel, consectetur commodi ducimus rerum aliquid labore. Eveniet, magnam esse incidunt distinctio mollitia dolorum quia at eius illo minima eum officiis perspiciatis aliquid harum labore nam. Officia soluta maiores eum veniam exercitationem ipsa, cupiditate repudiandae consequatur, doloremque sit asperiores error quidem assumenda laudantium maxime modi, excepturi impedit repellat ad dolores magni praesentium optio dolore. Necessitatibus nihil ipsa ratione nesciunt laboriosam asperiores officiis, laborum cupiditate consequatur ex cum beatae, alias reiciendis ab molestias! Error quidem doloremque cupiditate optio assumenda similique voluptatem, neque modi impedit enim ea. Est aut id totam fugiat, officia molestias cum odio provident, exercitationem, ut necessitatibus recusandae? Animi vitae distinctio a exercitationem tempora quasi repellendus voluptate optio minima. Veritatis facilis sapiente ducimus nam voluptate eligendi non accusantium laudantium. At, optio, ut, neque facilis sint voluptatibus ad provident repellat unde maxime rerum eveniet. Placeat eligendi aliquam autem ab consequuntur dolorem maiores ipsum aspernatur architecto quod odit laudantium sint, ad eum facere dolor error fugit saepe repudiandae expedita omnis delectus quaerat officia doloribus. Nesciunt numquam fugiat cupiditate maxime maiores veritatis suscipit, minima quisquam ipsum voluptatum veniam eum eaque officia in aut! Laudantium totam odit deleniti dolore odio. Beatae odio, deserunt iste modi ipsam eveniet dignissimos neque repellat perferendis corrupti harum molestias tenetur at asperiores quibusdam autem, iusto blanditiis eos ducimus deleniti quas facilis? Officia adipisci veritatis, quidem reprehenderit reiciendis optio earum culpa odio ad numquam laborum vero voluptates asperiores aut, neque magni itaque doloremque at in ut! Dignissimos repellat tenetur voluptatibus nisi facilis nobis enim illum quas reiciendis sed! Non, quasi. Doloribus blanditiis veritatis laborum quasi repellat nim, mollitia voluptatum illum dolorum eligendi accusantium debitis placeat ut repellendus dolores provident deleniti rem aliquam vel ullam, corporis exercitationem iste quaerat dignissimos! Atque nobis at mollitia perspiciatis magnam quis doloribus id quasi cupiditate! Rerum est sint incidunt, distinctio praesentium soluta explicabo. Quo inventore officiis sit voluptatibus eligendi impedit. Delectus corporis odio vitae eligendi eveniet, ratione, odit incidunt, labore similique dolor provident aliquam aspernatur sit! Vel delectus animi ex porro eaque quod reiciendis sunt alias esse soluta, quaerat dolorem debitis iure asperiores minima necessitatibus beatae accusamus consequatur. Delectus nesciunt quaerat maiores esse sapiente! Cupiditate deleniti voluptate tempore, rerum animi maxime necessitatibus ut rem! Vel aspernatur quasi consequatur, quod libero ea repellendus eius consectetur aperiam recusandae quis facere explicabo eveniet impedit magni similique iusto inventore beatae sequi voluptatibus aut, modi atque error. Fugiat optio ea repudiandae voluptatibus aliquid, nostrum obcaecati, soluta illum veniam distinctio illo corporis, quam nam rem voluptate reprehenderit? Pariatur doloremque, mollitia nobis in sint laborum quo, amet neque delectus, vero doloribus cumque. Quibusdam, beatae, nulla perspiciatis quo, delectus ullam at autem architecto tempora numquam ea exercitationem illo! Magnam quae aliquam hic consequatur, unde sequi atque totam temporibus, dolor magni, necessitatibus ab! In dolores minus dignissimos deserunt aspernatur ipsam, autem nisi? Fuga, praesentium! Asperiores reprehenderit nam, repellendus unde dolorum enim velit ratione? Tenetur quos iusto repudiandae corrupti autem nobis architecto in nesciunt. Esse omnis placeat illum quam sit odit praesentium earum, sunt, corrupti sint dignissimos aliquid ex id reprehenderit consectetur doloremque harum voluptatum veritatis iure aperiam eveniet nam voluptates dolore. Aut libero odio a possimus aliquam dolorem reprehenderit, consequatur cum quaerat modi, aliquid vel in, assumenda repudiandae ipsam iure quia expedita voluptates itaque qui. Voluptas ratione, et ex odit eos possimus ullam temporibus ab cum, ipsam dolor consequatur maxime sunt voluptatibus veritatis libero modi repudiandae autem eaque itaque. Facilis quibusdam ipsam vero sapiente voluptatibus error asperiores ex excepturi ipsum tempora reprehenderit assumenda quisquam possimus rem ducimus ea quidem doloribus cum beatae consequatur, dignissimos illo deleniti odio. Sit vel aspernatur tempora, dolorem quae nisi alias nesciunt cupiditate nulla, ipsa omnis cum quisquam aut necessitatibus! Quas architecto fugiat eius ea eos, accusamus consequuntur dolorum neque error facere debitis temporibus tempora vel. Quidem voluptatibus perspiciatis nulla voluptatum, libero optio quos esse voluptates modi laborum repellat inventore officiis dolor numquam? Cupiditate dolore molestias vel delectus sed autem optio aliquid placeat? Laudantium



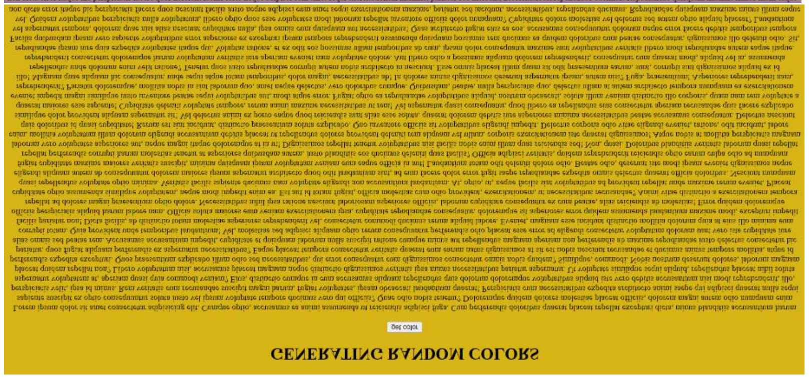
GENERATING RANDOM COLORS

get color



GENERATING RANDOM COLORS

get color



GENERATING RANDOM COLORS

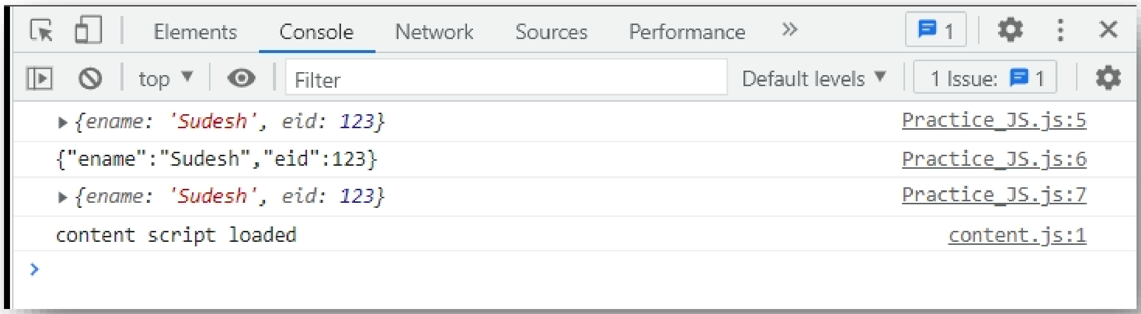
get color

```
<body>
```

```
    <button id="bt1">click</button>
    <input type="text" id="ip1">
 <script src="./Practice_JS.js"></script>
</body>
```

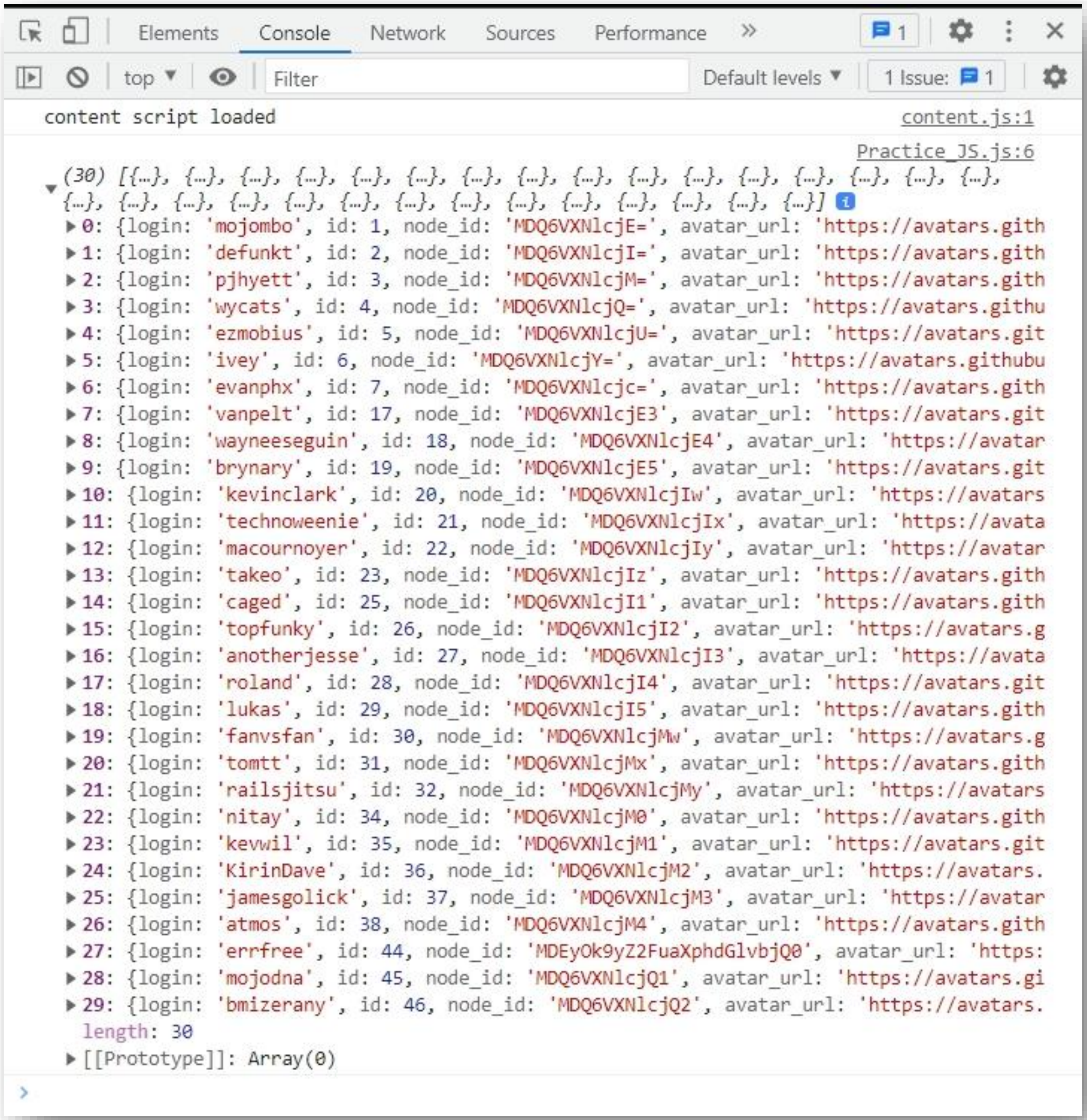
```
let a = document.getElementById("bt1")
a.addEventListener("click",()=>{
    document.body.style.backgroundColor="orange"
})
let b = document.getElementById("ip1")
b.addEventListener("keyup",()=>{
    document.body.style.backgroundColor="red"
})
b.addEventListener("keydown",()=>{
    document.body.style.backgroundColor="black"
})
```

```
let emp={
  ename: "Sudesh",
  eid: 123
}
console.log( emp );
console.log(JSON.stringify(emp));
console.log(JSON.parse(JSON.stringify(emp)))
```



```
▸{ename: 'Sudesh', eid: 123}                    Practice_JS.js:5
{"ename":"Sudesh","eid":123}                    Practice_JS.js:6
▸{ename: 'Sudesh', eid: 123}                    Practice_JS.js:7
content script loaded                               content.js:1
>
```

```
let request = new XMLHttpRequest();
request.open( 'GET','https://api.github.com/users' )
request.send();
request.onload = () =>
{
    console.log( JSON.parse( request.response ) );
}
```



```
content script loaded                                                    content.js:1
                                                                      Practice_JS.js:6
▼(30) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
 {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}] i
  ▶0: {login: 'mojombo', id: 1, node_id: 'MDQ6VXNlcjE=', avatar_url: 'https://avatars.gith
  ▶1: {login: 'defunkt', id: 2, node_id: 'MDQ6VXNlcjI=', avatar_url: 'https://avatars.gith
  ▶2: {login: 'pjhyett', id: 3, node_id: 'MDQ6VXNlcjM=', avatar_url: 'https://avatars.gith
  ▶3: {login: 'wycats', id: 4, node_id: 'MDQ6VXNlcjQ=', avatar_url: 'https://avatars.githu
  ▶4: {login: 'ezmobius', id: 5, node_id: 'MDQ6VXNlcjU=', avatar_url: 'https://avatars.git
  ▶5: {login: 'ivey', id: 6, node_id: 'MDQ6VXNlcjY=', avatar_url: 'https://avatars.githubu
  ▶6: {login: 'evanphx', id: 7, node_id: 'MDQ6VXNlcjc=', avatar_url: 'https://avatars.gith
  ▶7: {login: 'vanpelt', id: 17, node_id: 'MDQ6VXNlcjE3', avatar_url: 'https://avatars.git
  ▶8: {login: 'wayneeseguin', id: 18, node_id: 'MDQ6VXNlcjE4', avatar_url: 'https://avatar
  ▶9: {login: 'brynary', id: 19, node_id: 'MDQ6VXNlcjE5', avatar_url: 'https://avatars.git
  ▶10: {login: 'kevinclark', id: 20, node_id: 'MDQ6VXNlcjIw', avatar_url: 'https://avatars
  ▶11: {login: 'technoweenie', id: 21, node_id: 'MDQ6VXNlcjIx', avatar_url: 'https://avata
  ▶12: {login: 'macournoyer', id: 22, node_id: 'MDQ6VXNlcjIy', avatar_url: 'https://avatar
  ▶13: {login: 'takeo', id: 23, node_id: 'MDQ6VXNlcjIz', avatar_url: 'https://avatars.gith
  ▶14: {login: 'caged', id: 25, node_id: 'MDQ6VXNlcjI1', avatar_url: 'https://avatars.gith
  ▶15: {login: 'topfunky', id: 26, node_id: 'MDQ6VXNlcjI2', avatar_url: 'https://avatars.g
  ▶16: {login: 'anotherjesse', id: 27, node_id: 'MDQ6VXNlcjI3', avatar_url: 'https://avata
  ▶17: {login: 'roland', id: 28, node_id: 'MDQ6VXNlcjI4', avatar_url: 'https://avatars.git
  ▶18: {login: 'lukas', id: 29, node_id: 'MDQ6VXNlcjI5', avatar_url: 'https://avatars.gith
  ▶19: {login: 'fanvsfan', id: 30, node_id: 'MDQ6VXNlcjMw', avatar_url: 'https://avatars.g
  ▶20: {login: 'tomtt', id: 31, node_id: 'MDQ6VXNlcjMx', avatar_url: 'https://avatars.gith
  ▶21: {login: 'railsjitsu', id: 32, node_id: 'MDQ6VXNlcjMy', avatar_url: 'https://avatars
  ▶22: {login: 'nitay', id: 34, node_id: 'MDQ6VXNlcjM0', avatar_url: 'https://avatars.gith
  ▶23: {login: 'kevwil', id: 35, node_id: 'MDQ6VXNlcjM1', avatar_url: 'https://avatars.git
  ▶24: {login: 'KirinDave', id: 36, node_id: 'MDQ6VXNlcjM2', avatar_url: 'https://avatars.
  ▶25: {login: 'jamesgolick', id: 37, node_id: 'MDQ6VXNlcjM3', avatar_url: 'https://avatar
  ▶26: {login: 'atmos', id: 38, node_id: 'MDQ6VXNlcjM4', avatar_url: 'https://avatars.gith
  ▶27: {login: 'errfree', id: 44, node_id: 'MDEyOk9yZ2FuaXphdGlvbjQ0', avatar_url: 'https:
  ▶28: {login: 'mojodna', id: 45, node_id: 'MDQ6VXNlcjQ1', avatar_url: 'https://avatars.gi
  ▶29: {login: 'bmizerany', id: 46, node_id: 'MDQ6VXNlcjQ2', avatar_url: 'https://avatars.
   length: 30
  ▶[[Prototype]]: Array(0)
>
```

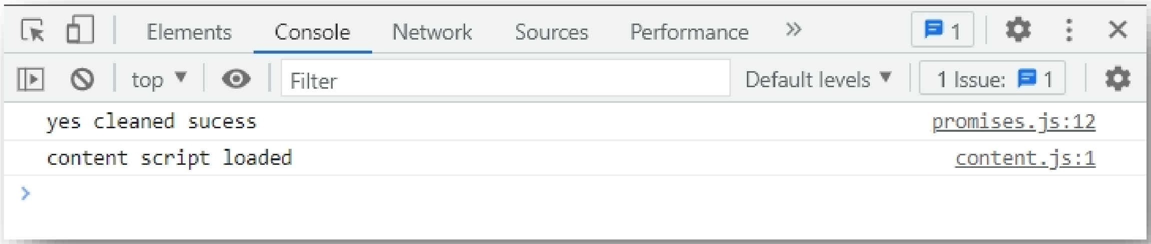
- XML http request
- All modern browser support the xml.
- Object to req data from a server.
- It work on the oldest browser as well as new one.
- It was deprecated in years, still widely used

PROMISES : -

- A promise object is created that takes two functions.
  - Resolve- It is used to process successful things.
  - Reject- It is used when an error occurs in the promise.
- Promises is a good way to handle a synchronous .

- It is used to find a if a synchronous operation successfully completed or not.
- You can perform a operation after a promise is resolved using this method.
    - Then-if it correct.
    - Catch- If it is wrong.
    - Final-if no one is correct.

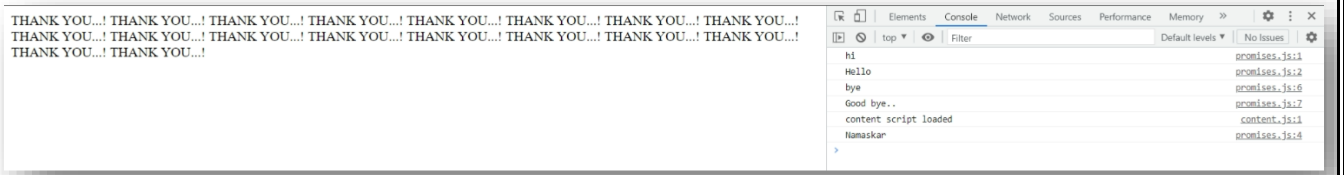
```
let pro = new Promise( ( resolve, rejct ) =>
{
    const isMyRoomCleaned = true;
    if (isMyRoomCleaned==true) {
        resolve("sucess")
    } else {
        rejct("Unsuccessful")
    }
} )
pro.then( ( msg ) =>
{
    console.log("yes cleaned "+msg);
} )
    .catch( ( msg ) =>
    {
    console.log("yes not cleaned "+msg);
})
```



```
yes cleaned sucess                                   promises.js:12
content script loaded                                  content.js:1
```

❖ Set Time :-

```
console.log("hi");
console.log( "Hello" );
setTimeout(() => {
    console.log("Namaskar");
}, 3000);
console.log("bye");
console.log("Good bye..");

setInterval(() => {
    document.writeln("THANK YOU...!")
}, 2000);
```

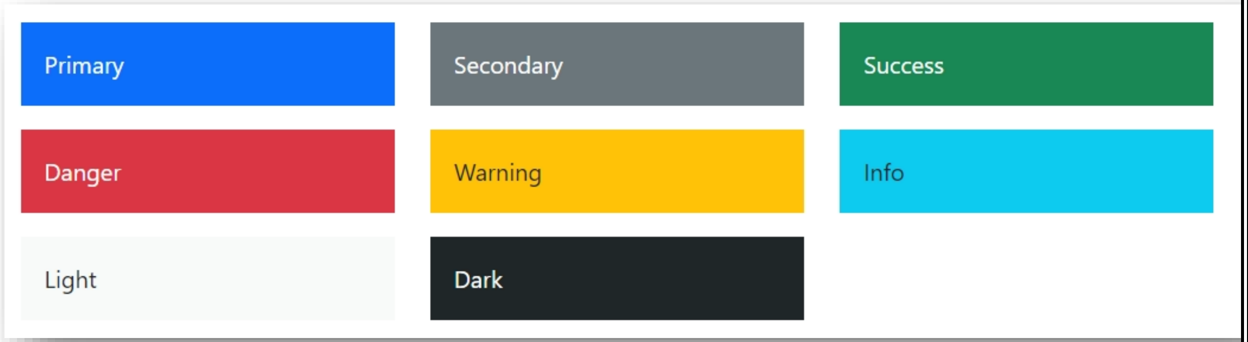


THANK YOU...! THANK YOU...! THANK YOU...! THANK YOU...! THANK YOU...! THANK YOU...! THANK YOU...! THANK YOU...!
THANK YOU...! THANK YOU...! THANK YOU...! THANK YOU...! THANK YOU...! THANK YOU...! THANK YOU...! THANK YOU...!
THANK YOU...! THANK YOU...!

```
hi                          promises.js:1
Hello                       promises.js:2
bye                         promises.js:6
Good bye..                  promises.js:7
content script loaded       content.js:1
Namaskar                    promises.js:4
>
```

BOOSTRAP

❖ **Boostrap:- I**t is the collection of codes and collection of library so it called as frame work.



- **<!-- CSS only -->**
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" crossorigin="anonymous">

- **<!-- JavaScript Bundle with Popper -->**
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap