[Sudhakar Sharma Sir]

JavaScript

JavaScript

- JavaScript is Interpreted And light weight, JIT [Just-in-Time] compiled programming language.
- Interpreted :Line By Line Translation

What is JIT?

- There are 2 type of compiling techniques
    a) JIT [Just-in-Time]
    b) AOT [Ahead-of-Time]
- JIT And AOT : Compiling Techniques
- JIT compiles logic in browser.
- AOT compile logic at app level.
- Compiling is the process of translating all lines in program simultaneously at the same time.
- The comilers used for JavaScript are V8, Babel etc..
- JavaScript supports various programming techniques and approaches
    a) Functional Programming
    b) Structrual Programming
    c) Imperative Programming
    d) Object Oriented Programming etc..

- JavaScript is not an OOP language, it supports only few features of OOP.
- JavaScript is used
    a) Client Side HTML
    b) Server Side Node.js

c) Database	MongoDB
d) Animations	ActionScript, Flash, 3DS Max etc..
E)CAD	AutoCAD

Evolution of JavaScript:

- The first browser was "Mosaic"

Markup	: GML, SGML Presentation
Script	: ECMA Script Client Side

- 1990's Tim Berners Lee introduced HTML.
-1992's

HTML	:IETF [Internet Engineering Task Force]
Script	: ECMA Script unable handle HTML new features

- 1994 Netscape Communications - Developed browser "Netscape Communicator"
    - HTML
    - ECMA Script
    - Netscape appointed"Brendan Eich"[CERN][10 Days]
    - Mocha
    - Live Script
    - Sun Micro Systems - JavaScript
    - MDN-JavaScript [Mozilla Devolper Network]
- 1998 Microsoft Win-98 - Free Apps

  Internet Explorer

  Wordstar - Buy

  Lotus -

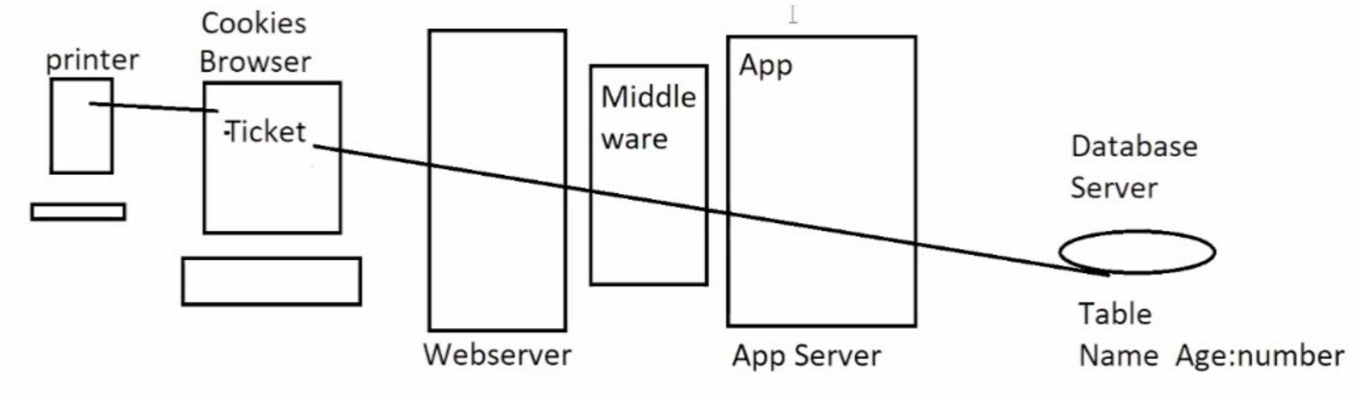
  FoxPro

  XP - Plug And Play [Free Browser & Tools]
- 2001's Every Browser Started adding features to JavaScript by its own.
- 2006 John Resing - Library For JavaScript

- jQuery
- 2004 Netscape stopped its browser, JavaScript -> ECMA
- ECMA Script = JavaScript = ES4 - ES20 [ECMA Script 2020]

JavaScript with HTML [Client Side]

- JavaScript is used Client Side Script with HTML.
- JavaScript can reduce burden on server by



- JavaScript handles
  - Validations Client Side
  - Interactions Client Side
  - Browser Animations
  - Browser Plugins etc..
  - DOM Manipulations
    a) Adding Elements
    b) Removing Elements
    c) Rendering new Data into Elements
    d) Update data in elements etc..

- Client Side Script Issues
  - Not Secured
  - Trozen Friendly [Virus, Trozen, Worms etc..]
  - Disabled by Browsers
- JavaScript Issues
  - Not Strongly Typed

        - Not Strictly Typed
        - Not Follow Duck Typing
        - Not Support All OOP Features
        - Extensibility Issues
        - Lot Of References
        - Heavy On Application
        - Compatibility Issues
    - Alternatives
        - atScript   [Obsolete]
        - TypeScript

FAQ: What is the role of JavaScript with HTML?

Ans: DOM Manipulations.

Integrating JavaScript into Page

1. Inline
2. Embedded
3. External File

Ex:- Inline

```
     <button onclick="window.print()"> Print </button>

    <!DOCTYPE html>
        <html>
          <head>
           <title>JavaScript - Inline</title>
         </head>
     <body>
        <h1>Click Print button to print page.</h1>
            <button onclick="window.print()">Print</button>
     </body>
    </html>
```

Ex:- Embeded

```
       <script type="text/javascript">
        function  name(){

        }
       </script>

        <button onclick="name()"> Insert </button>
```

```
<!DOCTYPE html>
<html>
   <head>
      <title>JavaScript - Inline</title>
      <script type="text/javascript" >
         function PrintPage(){
            window.print();
         }
      </script>
   </head>
   <body>
      <h1>Click Print button to print page.</h1>
      <button onclick="PrintPage()">Print</button>
   </body>
</html>
```

Ex:- External

1. Create a new file

```
      "printing.js"
     function PrintPage()
     {
  window.print();
 }
```

2. Link to HTML page

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

<button onclick="PrintPage()"> Print </button>
```

Note: Always use "Minified" script file in "Production".
jsminifier.com

- Minifying JavaScript File

Printing.min.js	Production
Printing.js	Development

- Media Type For JavaScript
  text/javascript
  language = "javascript"

Syntax:-

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

- Strict Mode For JavaScript

ESLint	JavaScript Language Analyzer

You Can Turn On JavaScript Strict Mode by Using "use strict;"

How JavaScript Refers HTML elements?

1. JavaScript can refer HTML elements by using DOM

hierarchy.

- If you change the position of any element in page, then every time you have to update its position in code.
- It is faster in rendering.

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>JavaScript - Inline</title>
      <script>
         function bodyload(){
            window.document.images[0].src =
"../public/images/laptop.png";
            window.document.forms[0].elements[1].value =
"Register";
           window.document.forms[1].elements[1].value = "Login";
         }
      </script>
   </head>
   <body onload="bodyload()">
      <img width="100" height="100" border="1">
      <div>
         <form>
            <h2>Register</h2>
            User Name : <input type="text"> <input type="button">
         </form>
      </div>
      <div>
         <form>
            <h2>Login</h2>
            Email : <input type="email"> <input type="button">
          </form>
        </div>
    </body>
</html>
```

2. JavaScript can refer elements by using "name".

- You can access any element directly by using its reference name.
- you can't access any child element directly without refering its parent.
- You have to refer both parent and child hierarchy.
- you can access element by using name.
- Name can be common for multiple elements.

Syntax:-

```
        <img name="pic" >
         pic.src="path";

        <form name="form1">
          <input type="button" name="btn1">
        </form>

        form1.btn1.value="Text";
```

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>JavaScript - Inline</title>
    <script>
      function bodyload(){
            pic.src="../public/images/neckband.png";
            frmRegister.btnRegister.value="Register";
            frmLogin.btnLogin.value = "Login";
        }
    </script>
  </head>
  <body onload="bodyload()">
    <img width="100" name="pic" height="100" border="1">
    <div>
        <form name="frmRegister">
            <h2>Register</h2>
            User Name :<input type="text" name="txtName">
<input name="btnRegister" type="button">
        </form>
    </div>
    <div>
        <form name="frmLogin">
            <h2>Login</h2>
            Email : <input name="txtEmail" type="email"> <input
name="btnLogin" type="button">
        </form>
    </div>
  </body>
</html>
```

3. You can refer by using "ID"

- Every element can be configured with only one ID.
- Every element can be defined with unique ID.
- youcan access element by using ID.
- ID have conflict with CSS reference
- It requires the method

document.getElementById()

- ID is not required if you are accessing any level directly document element.
- Issue with ID is, it is also used in CSS references where a common ID can be given formultiple elements.

Syntax:-

```
<img id="pic">
document.getElementById("pic").src = "path";
```

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>JavaScript - Inline</title>
    <script>
       function bodyload(){

 document.getElementById("pic").src="../public/images/mobile.
png";
           document.getElementById("btnRegister").value =
"Register";
           document.getElementById("btnLogin").value = "Login";
       }
    </script>
  </head>
  <body onload="bodyload()">
    <img width="100" id="pic" name="pic" height="100"
border="1">
    <div>
       <form name="frmRegister">
```

```
            <h2>Register</h2>
            User Name :<input type="text" name="txtName">
<input id="btnRegister" name="btnRegister" type="button">
         </form>
      </div>
      <div>
         <form name="frmLogin">
            <h2>Login</h2>
            Email : <input name="txtEmail" type="email"> <input
id="btnLogin" name="btnLogin" type="button">
         </form>
      </div>
   </body>
</html>
```

4. JavaScript can refer any element by using CSS selectors

- Query Selector can apply effects only to the first element.
- How ever it can handle data for multiple elements.
- It can also handle styles for multiple elements.
- JavaScript can refer HTML element by using all CSS selectors, which include.
    a) Type
    b) ID
    c) Class
    d) Rational
    e) Pseudo etc..
- You have to use the method document.querySelector()

Syntax:-

```
<img id="pic">
<input type="button" class="btn">
```

```
      document.querySelector("h2")         // type selector
      document.querySelector("#pic")     // id selector
      document.querySelector(".pic")     // class selector
```

- It can access and apply to element first occurance.

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>JavaScript - Inline</title>
      <script>
         function bodyload(){

document.querySelector("#pic").src="../public/images/laptop.pn
g";
     document.querySelector(".btn-register").value="Register";
           document.querySelector(".btn-login").value="Login";
         }
      </script>
   </head>
   <body onload="bodyload()">
      <img width="100" id="pic" name="pic" height="100"
border="1">
      <div>
         <form name="frmRegister">
            <h2>Register</h2>
            User Name :<input type="text" name="txtName">
<input id="btnRegister" name="btnRegister" class="btn-register"
type="button">
```

```
            </form>
        </div>
        <div>
            <form name="frmLogin">
                <h2>Login</h2>
                Email : <input name="txtEmail" type="email"> <input
id="btnLogin" name="btnLogin" class="btn-login"
type="button">
            </form>
        </div>
    </body>
</html>
```

Accessing Multiple Elements

```
        console.log(document.getElementsByTagName("input").
        length);
        console.log(document.getElementsByName("pay").length);
```

5. JavaScript can refer all elements having common name

```
        document.getElementsByName()
```

Ex:-

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>JavaScript - Inline</title>
        <script>
            function bodyload(){
               result = document.getElementsByName("pay");
               alert("Total Number of Payment Methods : " +
result.length);
```

```
        }
      </script>
   </head>
   <body onload="bodyload()">
      <fieldset>
         <legend>Payment Method</legend>
         <input type="radio" name="pay"> Cash
         <input type="radio" name="pay"> UPI
         <input type="radio" name="pay"> Credit Card
      </fieldset>
   </body>
</html>
```

6. JavaScript can refer all elements having common class Name

      document.getElementsByClassName()

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>JavaScript - Inline</title>
      <script>
         function bodyload(){
           result = document.getElementsByClassName("form-
check-input");
           alert("Radios with Input Class : " + result.length);
         }
      </script>
   </head>
   <body onload="bodyload()">
      <fieldset>
```

```
          <legend>Payment Method</legend>
          <input type="radio" class="form-check-input"
name="pay"> Cash
          <input type="radio" class="form-check-input"
name="pay"> UPI
          <input type="radio" name="pay"> Credit Card
        </fieldset>
    </body>
</html>
```

7. JavaScript can refer all elements by using Tag Name

```
        .getElementsByTagName()
```

Ex:-

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>JavaScript - Inline</title>
        <script>
          function bodyload(){
            result = document.getElementsByTagName("h2");
            alert("Total no. of Headings : " + result.length);
          }
        </script>
    </head>
    <body onload="bodyload()">
        <h2>Register</h2>
        <fieldset>
          <legend>Payment Method</legend>
          <input type="radio" class="form-check-input"
name="pay"> Cash
```

```
            <input type="radio" class="form-check-input"
name="pay"> UPI
            <input type="radio" name="pay"> Credit Card
        </fieldset>
        <h2>Login</h2>
    </body>
</html>
```

JavaScript Output Techniques

Output refers to the techniques used for rendering data into UI.

- alert()
- confirm()
- console.log(), warn(), success(), error(), debug(), info() etc..
- innerText
- innerHTML
- outerHTML
- document.write()

alert:

- It can display message in a message box.
- Window pops up a message box.
- You can't cancel.

Syntax:-

```
alert("message");
```

- alert is RC data type.
- You can display in multiple line using "\n"

Ex:-

```
<!DOCTYPE html>
```

```
        <html>
        <head>
<title>Demo</title>
<script>
        function InsertClick(){
                alert("Record Inserted\nYou check the database.");
        }
        </script>
        </head>
<body>
        <button onclick="InsertClick()">Insert</button>
</body>
</html>
```

confirm:

- It is similar to alert box but allows tocancel.
- confirm buttons are

```
                Ok              = true
                Cancel          = false
```

- confirm box returns true on OK and false on cancel.

Syntax:-

```
            confirm("your message");
```

Ex:-

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Demo</title>
        <script>
            function DeleteClick(){
                if(confirm("Record will be Deleted")==true){
                    alert("Record Deleted..");
                } else {
                    alert()"Action Canceled..");
                }
            }
            </script>
          </head>
        <body>
            <button onclick="DeleteClick()">Delete</button>
        </body>
      </html>
```

console methods:

- They are used by developers to display message in console of browser debugging tools.

```
console.log()
console.warn()
console.error()
console.debug()
console.info()
```

Syntax:-

```
console.log("your message");
console.error("message..");
```

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>Demo</title>
```

```
        <script>
            function DeleteClick(){
                console.warn("Warning : Delete Clicked");
                if(confirm("Record will be Deleted")==true) {
                    console.error("Delete Confirmed");
                    document.write("<h2><font color=red>Record
Deleted..</font></h2>");
                } else {
                    alert("Action Canceled..");
                }
            }
        </script>
    </head>
    <body>
        <button onclick="DeleteClick()" >Delete</button>
    </body>
</html>
```

innerText:

- It is an output property used for displaying content in any container like <p> <dd> <td> <div> <span> <h2> etc..

- It is RC data type, which will not support formats for text.

Syntax:-

```
document.querySelector("p").innerText="your text";
```

innerHTML:

It is similar to innerText property but allows formats for text.

Syntax:-

```
document.querySelector("p").innerHTML = "your markup";
```

outerHTML:

It is similar to innerHTML but will replace the parent container and render the markup defined.

Syntax:-

```
document.querySelector("p").outerHTML = "<target
 markup>";
```

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>Demo</title>
    <script>
       function DeleteClick(){

          if(confirm("Record will be Deleted")==true) {
             document.querySelector("p").outerHTML=
"<h2>Record Deleted Successfully..</h2>";
          } else {
             document.querySelector("p").innerText = "Action
Canceled..";
          }
       }
    </script>
  </head>
  <body>
    <button onclick="DeleteClick()" >Delete</button>
```

```
        <p align="center"></p>
    </body>
</html>
```

JavaScript Input Techniques

- Input is the process of accepting a value dynamically into application from user. Javascript based applications can handle input using

    a) Query String
    b) Prompt
    c) Form Input Elements

Query String:

- A query string is defined in address bar of browser as URL search parameter.

Syntax:-

http://127.0.0.1:5500/home.html?key=value&key=value

- You can access and use the querystring by using JavaScript location object search property.

Syntax:-

```
console.log(location.search);
```

Ex:-

```
<!DOCTYPE html>
<html>
<head>
  <title>Demo</title>
```

```
        <script>
           function bodyload(){
               document.querySelector("p").innerHTML =
location.search;
           }
        </script>
    </head>
    <body onload="bodyload()">
        <p align="center"></p>
    </body>
</html>
```

http://www.amazon.in/electronics.html?category=mobiles&model=samsung

http://www.amazon.in/electronics/mobiles/samsung

Prompt:

- It provides an input box from where you can input value.
- It can input only string.
- Prompt returns

OK without value	:	returns empty string [" "]
Cancel with/without value	:	returns null.
OK with value	:	returns the value
on cancel	:	null
on ok with value	:	value

Syntax:-

```
        prompt("Your Message", "Default _Value");
```

Ex:-1

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>Demo</title>
      <script>
         function CreateClick(){
           foldername = prompt("Enter Folder
Name","new_folder");
           if(foldername==""){
              document.querySelector("p").innerHTML ="Name
can't be Empty";
           } else if(foldername==null) {
             document.querySelector("p").innerHTML="You
canceled..";
           } else {
              document.querySelector("p").innerHTML+="Folder
Created : " + foldername + "<br>";
           }
         }
      </script>
   </head>
   <body>
       <button onclick="CreateClick()">Create New
Folder</button>
      <p></p>
   </body>
</html>
```

Ex:-2

```
      <!DOCTYPE html>
      <html>
```

```
   <head>
      <title>JavaScript - Inline</title>
      <script>
         function CreateClick(){
            foldername = prompt("Enter Folder
Name","New_folder");
            if(foldername==null){
               document.write("You canceled..");
            } else if (foldername=="") {
               document.write("Please provide folder name..");
            } else {
               document.querySelector("p").innerHTML+= "Folder
Created :" + foldername + "<br>";
            }
         }
      </script>
   </head>
   <body>
      <button onclick="CreateClick()">Create Folder</button>
      <p></p>
   </body>
</html>
```

Form Input Elements

- You can use form input elements like textbox, password, number, email, radio, checkbox, listbox, dropdown etc.

Step-1: Every form element must have a reference ID.

```
<input type="text"  id="txtName">
<select id="lstCities">
<input type="checkbox"  id="optStock">
```

Step-2: You can access element and use its properties

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

document.getElementById("optStock").checked==true/false
```

Ex:-1

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>JavaScript - Inline</title>
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
      <script>
          function SubmitClick(){

document.getElementById("detailsContainer").style.display="blo
ck";

document.getElementById("registerContainer").style.display="n
one";

              document.getElementById("lblName").innerHTML =
document.getElementById("txtName").value;
              document.getElementById("lblPrice").innerHTML =
document.getElementById("txtPrice").value;
              document.getElementById("lblCity").innerHTML =
document.getElementById("lstCities").value;

              stock = "";
```

```
            stockCheckBox =
document.getElementById("optStock");
            if(stockCheckBox.checked)
            {
               stock = "Available";
            } else {
               stock = "Out of Stock";
            }

            document.getElementById("lblStock").innerHTML =
stock;
         }
      </script>
   </head>
   <body class="container-fluid">
      <div class="mt-3" id="registerContainer">
         <button data-bs-target="#registerProduct" data-bs-
toggle="modal" class="btn btn-primary">Register New
Product</button>
      </div>
      <div class="modal fade" id="registerProduct">
         <div class="modal-dialog">
            <div class="modal-content">
               <div class="modal-header">
                  <h2>Reigster new Product</h2>
                  <button class="btn-close" data-bs-
dismiss="modal"></button>
               </div>
               <div class="modal-body">
                  <dl>
                     <dt>Name</dt>
```

```
                    <dd><input type="text" id="txtName"
class="form-control"></dd>
                    <dt>Price</dt>
                    <dd><input type="text" id="txtPrice"
class="form-control"></dd>
                    <dt>Shipped To</dt>
                    <dd>
                       <select id="lstCities" class="form-select">
                          <option>Delhi</option>
                          <option>Hyderabad</option>
                       </select>
                    </dd>
                    <dt>Stock</dt>
                    <dd class="form-switch">
                       <input id="optStock" type="checkbox"
class="form-check-input"> Available
                    </dd>
                 </dl>
              </div>
              <div class="modal-footer">
                 <button class="btn btn-primary"
onclick="SubmitClick()" data-bs-
dismiss="modal">Submit</button>
                 <button class="btn btn-danger" data-bs-
dismiss="modal">Cancel</button>
              </div>
           </div>
        </div>
     </div>

     <div id="detailsContainer" style="display: none;">
```

```
          <h2>Product Details</h2>
          <dl class="row">
             <dt class="col-3">Name</dt>
             <dd class="col-9" id="lblName"></dd>
             <dt class="col-3">Price</dt>
             <dd class="col-9" id="lblPrice"></dd>
             <dt class="col-3">Shipped To</dt>
             <dd class="col-9" id="lblCity"></dd>
             <dt class="col-3">Stock</dt>
             <dd class="col-9" id="lblStock"></dd>
          </dl>
          <button class="btn btn-info" data-bs-toggle="modal"
data-bs-target="#registerProduct">Edit</button>
       </div>

       <script src="../node_modules/jquery/dist/jquery.js">
       </script>
       <script
src="../node_modules/bootstrap/dist/js/bootstrap.bundle.js">
       </script>
   </body>
</html>

document.getElementById("IblName").innerHTML = name;
```

Ex:-2

```
       <!DOCTYPE html>
       <html>
   <head>
       <title>Demo</title>
       <link rel="stylesheet" href="../node_modules/bootstrap-
```

```
icons/font/bootstrap-icons.css">
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
      <script>
         function RegisterClick(){

document.getElementById("detailsContainer").style.display="blo
ck";
            document.getElementById("lblName").innerHTML =
document.getElementById("txtName").value;
            document.getElementById("lblPrice").innerHTML =
document.getElementById("txtPrice").value;
            document.getElementById("lblCity").innerHTML =
document.getElementById("lstCities").value;
            stock="";
            if(document.getElementById("optStock").checked) {
               stock = "Available";
            } else {
               stock = "Out of Stock";
            }
            document.getElementById("lblStock").innerHTML =
stock;
         }
      </script>
   </head>
   <body class="container-fluid">
       <div class="mt-2">
         <button data-bs-target="#registerContainer" data-bs-
toggle="modal" class="btn btn-primary">Register New
Product</button>
       </div>
```

```
        <div id="registerContainer" class="modal fade">
          <div class="modal-dialog">
            <div class="modal-content">
              <div class="modal-header">
                <h3>Register Product</h3>
                <button class="btn-close" data-bs-
dismiss="modal"></button>
              </div>
              <div class="modal-body">
                <dl>
                  <dt>Name</dt>
                  <dd><input type="text" class="form-control"
id="txtName"></dd>
                  <dt>Price</dt>
                  <dd><input type="text" class="form-control"
id="txtPrice"></dd>
                  <dt>City</dt>
                  <dd>
                    <select class="form-select" id="lstCities">
                      <option>Delhi</option>
                      <option>Hyderabad</option>
                    </select>
                  </dd>
                  <dt>Stock</dt>
                  <dd class="form-switch">
                    <input class="form-check-input"
id="optStock" type="checkbox"> Available
                  </dd>
                </dl>
              </div>
              <div class="modal-footer">
```

```
                  <button onclick="RegisterClick()" data-bs-
dismiss="modal" class="btn btn-primary">Register</button>
                </div>
              </div>
          </div>
        </div>
        <div id="detailsContainer" style="display: none;">
          <h3>Product Details</h3>
          <dl class="row">
             <dt class="col-3 mb-2">Name</dt>
             <dd class="col-9 mb-2" id="lblName"></dd>
             <dt class="col-3 mb-2">Price</dt>
             <dd class="col-9 mb-2" id="lblPrice"></dd>
             <dt class="col-3 mb-2">Shipped To</dt>
             <dd class="col-9 mb-2" id="lblCity"></dd>
             <dt class="col-3 mb-2">Stock</dt>
             <dd class="col-9 mb-2" id="lblStock"></dd>
          </dl>
          <div>
             <button data-bs-target="#registerContainer" data-bs-
toggle="modal" class="btn btn-link">Edit</button>
          </div>
        </div>

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

FAQ: What is strict mode of JavaScript?

Ans: Strict mode for JavaScript allows to reduce code in-consistency So that developers have to follow coding standards.

```
 <script>
      "use strict";
       x = 10;                    // invalid - x is not defined
       document.write("x=" + x);
</script>

var x;
x = 10;
```

FAQ: How to write JavaScript for Legacy Browser? [Old Version]

Ans: By enclosing the code in "HTML Comments"

Syntax:-

```
        <script>
                <!--
                "use strict";
                var x;
                x = 10;
                document.write("x=" + x);
                -->
        </script>
    <body>
      <!-- HTML Comments -->
    </body>
```

FAQ: How to add JavaScript Comments?

Ans:

//	Single line comment
/* */	Multi line comment
///	XML comment
<!-- -->	HTML comments

JavaScript Language Basics

- Variables
- Data Types
- Operators
- Statements
- Functions

Variables

- Variables are storage locations in memory where you can store a value and use it as a part of any expression.
- JavaScript allows to use variable directly if it is not in strict mode.

Ex:-

```
<script>
 username = prompt("Enter Name");
 document.write("Hello ! " + username);
 document.write(" How are you today ? " + username);
</script>
```

- In strict mode variable configuration comprises of 3 phases
  - a) Declaration
  - b) Assignment / Rendring
  - c) Initialization

- Declaring is defining scope and name for variable.

```
var  username;
```

- Assigning is rendering a value into variable after declaring.

```
username = "john";
```

- Initialization is rendering a value into variable while declaring.

```
var username = "john";
```

Note: Declaring variable is not mandatory in JavaScript if it is not in strict mode.
- Declaring or Initialization of variable is mandatory if JavaScript is in strict mode.

Ex:-

```
<script>
    "use strict";
     x = 10;
    document.write("X="+ x);
</script>
```

- Variables in JavaScript can be declared by using

    a) var
    b) let
    c) const

var:-

- It defines function scope variable
- You can declare in any block of a function and access

from any another block in function.
- var Supports declaring, assigning and initialization.
- var allows shadowing.

Syntax:-

```
    <script>
        function f1(){
    var x;
    x = 10;
    if(x==10)
    {
        var y = 20;
    }
    document.write("x=" + x + "<br>" + "y=" + y);
  }
  f1();
</script>
```

FAQ: What is Shadowing?

Ans : Shadowing is a technique of re-declaring same name identifier within the scope.

Ex:-

```
    <script>
        function f1(){
    var x;
    x = 10;
    if(x==10)
    {
        var y = 20;
        var y = 30;   // shadowing
        }
        document.write("x=" + x + "<br>" + "y=" + y);
    }
    f1();
</script>
```

- var allows hoisting.
- Hoisting is the mechanism of declaring a variable after using. there is no order dependency as "var" can hoist your variable so that compiler can catch and configure before using.

FAQ: What is hoisting?

Ans : It is a compiling technique, where compiler can find declaration of variable before using it. Hence you can use a variable before declaring.

Ex:-

```
<script>
        "use strict";
        function f1(){
          x = 10;
          document.write("X=" + x);
          var x;  // hoisting
        }
        f1();
</script>
```

let :-

- It is used to define block scope variable.
- It is accessible only in the block where it is declared or to

  its inner blocks.
- It allows declaring, assigning and initalization.
- It will not allow shadowing.
- It will not allow hoisting.

Ex:-

```
<script>
      "use strict";
       function f1(){
          let x;
          x = 10;
           if(x==10)
           {
      let y = 20;
      document.write("x=" + x + "<br>" + "y=" + y);
    }

    }
    f1();
</script>
```

const:-

- It is used to define block scope variable.
- It will allow only initialization.
- No declaring
- No assigning
- No shadowing
- No hoisting

FAQ: Why we need a const?

Ans : const is required to initialize memory.

At the time of loading application or component memory is initialized with some default value.

Note: If initialization is missing then by default value will be "undefined".

```
var x;
document.write("x=" + x);          // x = undefined

const x;                           // invalid
x = 10;                           // invalid
const x = 10;                    // valid
```

FAQ: Can't const change its value?
Ans : Dynamically yes.

Global Scope for Variable:

- You can declare a variable in module scope. So that it is global and accessible to all functions in module.

Ex:-

```
    <script>
          var x = 10;
        function f1()
        {
          document.write("x in function-1 :" + x + "<br>");
          }
           function f2()
          {
          document.write("x in function-2 :" + x);
          }
```

```
                f1();
                f2();
            </script>
```

FAQ: Can we declare a global variable inside function?

Ans : Yes. By using browser "window" object.

```
            function f1()
            {
              window.y = 20;
            }

            function f2() {
              document.write("y=" + y);
            }
```

Ex:-

```
        <script>
            var x = 10;
     function f1()
     {
         window.y = 20;
        document.write("x in function-1 :" + x + "<br> y in function-
1 : " + y + "<br>");
     }
     function f2()
     {
        document.write("x in function-2 :" + x + "<br> y in function-
2 :" + y);
     }
     f1();
```

```
    f2();
</script>
```

Variable Naming Conventions:

- Variable name must start with alphabet or underscore.

```
var  username;
var  _username;
var  2021Sales;                    // invalid
var  Sales2021;                    // valid
var  Sales 2021;                   // invalid
var  Sales.2021;                   // invalid
var  Sales$2021;                   // invalid
var  Sales_2021;                   // valid
```

- Always use camel case for variable name

```
var tetName;
```

- Variable name must start with alphabet or underscore"_"
- Variable name can't start with number, but can be aplpha numeric

or

- The special character "_ " is used to indicate that variable not implemented. It requires further implementation.
- Don't use any special char in variable name othr that "_"
- Unerscore is used todefine a component that requires implementation.

```
var _productName;

get ProductName(){
  return _productName;
```

```
        }
```

- Double Underscore is used to define a component ready for testing.

```
        product.js                  // source file
        __product.js                //   test file
        __tests__                   //    test folder
```

- Variable name can't exceed more that 255 chars.
- Don't use language keywords for variable name.

```
        var for ;                        // invalid
        var switch;                      //   invalid
```

- Always variable must speak what it is.

```
        var  emp  = new Employee();             // not good
        var prod = new Product();               // not good
        var x = 1000;                           // not good

        var employee = new Employee();          //Good
        var product = new Product();            //Not Good
      var temporaryEmployee = new TemporaryEmployee();
```

Data Types

- It defines data structure. [DS]
- Data type determines the size and type of data.
- Data Structure comprises of information about
    - a) Type of Data
    - b) Size of Data
    - c) Memory type used for data etc..

- It will not have pre-defind data type for variables.
- Variable data type will change according to the value assigned or initialized.
- No fixed data type.

FAQ: What is Difference between strictly typed and strongly typed?

Ans : Strictly typed will reduce code inconsistancy.
Strongly typed will reduce variable data type inconsistancy.

- Data Types are classified into 2 types

    1) Primitive Data Types
    2) Non-Primitive Data Types

Primitive Data Types

- Primitive types are Immutable Types.
- Their structure will not change according state and situation.
- It have fixed structure.
- They have fixed range for value.
- Value range can't change.
- They use memory stack.
- Stack uses "LIFO" [Last-In-First-Out]
- JavaScript Primitive Types are

    a) number
    b) string
    c) boolean
    d) null
    e) undefined

Note: JavaScript is not strongly typed, It is implicitly typed.

```
var x  = 10;       x is number
x = "John";        x is string
```

Number Types

- JavaScript number type refers to

Signed Integer	-10
Unsigned Integer	10
Floating Point	24.53 or 45.34
Double	435.456 or 345.45, 45.563
Decimal	4560.55, 45.5695,[29 decimal places]
Exponent	2e3 [2 x 10^3] = 2000
Binary	0b1010
Octa	o740 or 0o742
Hexa	0fff00 or 000fd

Ex:-

```
<script>
      var x = 0b1010;
      document.write(`x is ${typeof x}<br>x=${x}`);
</script>
```

- JavaScript uses "isNaN()" method to verify the number type

Syntax:-

```
if(isNaN(value))
{
}
```

Ex:-

```
<script>
    var Age = prompt("Enter Age");
    if(isNaN(Age)) {
    document.write("Please Enter a numeric value");
 } else {
    document.write("Age=" + Age);
    }
</script>
```

- JavaScript can't identify a numeric value in string format, you have to explicitly convert into number by using following methods

```
a) parseInt()       x = 10;         parseInt(x);
b) parseFloat()     x = 10.56       parseFloat(x);
```

Ex:-

```
<script>
    var x = 10;
    var y = "46AB";        // valid to convert into number
```

FAQ: When you need parsing?

Ans: When you want to read and use as number.

Syntax:-

```
var  age  = "20";
document.write(age + 1);                    // 201
document.write(parseInt(age) + 1);        // 21
```

Ex:-

```
      <!DOCTYPE html>
      <html>
  <head>
    <title>Data Types</title>
    <script>
      function SubmitClick(){
        var age = document.getElementById("lstAge").value;
        document.write("You will be " + (parseInt(age) + 1) + "
next year");
      }
    </script>
  </head>
  <body>
    <dl>
      <dt>Select Age</dt>
      <dd>
        <select id="lstAge">
          <option value="15">15</option>
          <option value="20">20</option>
          <option value="25">25</option>
        </select>
      </dd>

    </dl>
    <button onclick="SubmitClick()">Submit</button>
  </body>
</html>
```

```
           var  a  = "10AB";
           var  b =  20;
```

```
        var  c  = parseInt(a) + b;       // 30
        a = "AB10";                      // invalid
        a = "10AB20";                    // 10
```

String Type

- String is a literal with group of characters enclosed in

  a) Single Quote   ' '
  b) Double Quote   " "
  c) Back Tick   ` `

- You can swap double and single quote for inner and outer string.
- Single and Double quote are used to configure inner and outer string combination.

Syntax:-

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

- ECMA2015 - ES5 introduced "backtick" for string, which can embed any expression by using data binding expression operayor "${}"
- Data Binding operator is allowed only in backtick.

Syntax:-

```
        var str =`your string ${expression}your string`;
```

- Back Tick allows a string with embedded expression "${ }".
  ${ }   => Data Binding Expression

Ex:- 1

```
      <script>
       var username = "john";
      var age =22;
      document.write("Hello !"+ username +" "+"you will  be"+"
"+(age+1)+" "+"+"next year.<br>");
      document.write(`Hello ! ${username}you will be
      ${age=+1} next year.1`);
      </script>
```

Ex:- 2

```
      <!DOCTYPE html>
      <html>
   <head>
     <title>Data Types</title>
     <script>
       function SubmitClick(){
         var username = prompt("Enter Your Name");
         var age = document.getElementById("lstAge").value;
         var msg1 = "Hello !" + " " + username + " " + "you will be"
+ " " + (parseInt(age)+1) + " " + "next year.<br>";
         var msg2 = `Hello ! ${username} you will be
${parseInt(age)+1} next year.`;
         document.write(msg1);
         document.write(msg2);
       }
     </script>
   </head>
   <body>
     <dl>
       <dt>Select Age</dt>
       <dd>
```

```
        <select id="lstAge">
          <option value="15">15</option>
          <option value="20">20</option>
          <option value="25">25</option>
        </select>
      </dd>

    </dl>
    <button onclick="SubmitClick()">Submit</button>
  </body>
</html>
```

Escape Sequence Issues:

- Special chars in a string can escape printing.
- You have to print the non-printable chars by using "\".

\n	new line in console
<br>	new line in HTML

Syntax:-

```
var path = "D:\images\pic.jpg";  => D:imagespic.jpg
var path = "D:\\images\\pic.jpg";  => D:\images\pic.jpg
```

Ex:-

```
  <script>
    var imagePath = "\"D:\\Images\\assets\\mobile.jpg\"";
    document.write(imagePath);
   </script>
```

- HTML string allows markup while presenting in non-rc data elements.

Syntax:-

```
document.write("First Line <br> <b> Second Line</b>");

alert("First line \n Second Line");
console.log("First line \n Second Line");
```

String Handling in JavaScript

JavaScript string object provides a set of methods and properties that are used to format and manipulate string.

String Formatting Methods:

- bold()
- italic()
- sup()
- sub()
- fontcolor()   fontcolor('colorName')
- fontsize()   fontsize('4')
- toUpperCase()
- toLowerCase()
- strike() etc..

Syntax:-

```
  var msg = "Welcome to JavaScript";
      msg.bold().italics().fontcolor('green');
  msg.toUpperCase();
```

Ex:-

```
<!DOCTYPE html>
<html>
<head>
<title>String Demo</title>
```

```
        <script>
            function VerifyUser(){
                var username =
document.getElementById("txtName").value;
                var msg = document.getElementById("msg");
                if(username.length>=4 && username.length<=10)
                {
                    msg.innerHTML = "User Name
Verified".bold().fontcolor('green').italics();
                } else {
                    msg.innerHTML = "Invalid - User Name 4 to 10 chars
Only".italics().fontcolor('red');
                }
            }
            function ChangeCase(){
                var username =
document.getElementById("txtName").value;
                document.getElementById("txtName").value =
username.toUpperCase();
            }
        </script>
    </head>
    <body>
        <h2>Register User</h2>
        <dl>
            <dt>User Name</dt>
            <dd><input type="text" placeholder="Block Letters"
id="txtName" onkeyup="VerifyUser()" onblur="ChangeCase()"
></dd>
            <dd id="msg"></dd>
        </dl>
        <button onclick="VerifyUser()">Submit</button>
    </body>
</html>
```

Events:

- onkeyup : actions to perform when key is released.
- onkeypress : actions when key finished.[keyin another char]
- onblur : actions to perform when control is blured. [lost focus]
- onclick : actions to perform when clicked.
- onload : actions to perform on page or image load.

Ex:- Formatting a String Dynamically using string format functions

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>String Demo</title>
        <script>
            function ChangeColor(){
                alert(document.querySelector("select").value);
                document.querySelector("p").innerHTML= "Welcome to
JavaScript".fontcolor(document.querySelector("select").value);
            }
        </script>
    </head>
    <body>
        <fieldset>
            <legend>Choose Effects</legend>
            <dl>
                <dt>Font Color</dt>
                <dd>
                    <select onchange="ChangeColor()">
                        <option>Red</option>
                        <option>Green</option>
                        <option>White</option>
                        <option>Yellow</option>
                    </select>
                </dd>
            </dl>
        </fieldset>
        <p></p>
    </body>
</html>
```

Ex:- Formatting string by using styles

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>String Demo</title>
    <script>
        function ChangeColor(){
          document.querySelector("p").style.color =
document.querySelector("select").value;
        }
    </script>
  </head>
  <body>
    <fieldset>
        <legend>Choose Effects</legend>
```

```
          <dl>
             <dt>Font Color</dt>
             <dd>
                <select onchange="ChangeColor()">
                   <option>Red</option>
                   <option>Green</option>
                   <option>White</option>
                   <option>Yellow</option>
               </select>
            </dd>
          </dl>
       </fieldset>
       <p>Welcome to JavaScript</p>
   </body>
</html>
```

Ex:- Dynamically applying CSS Class

```
       <!DOCTYPE html>
       <html>
   <head>
       <title>String Demo</title>
       <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
       <script>
          function ChangeTheme(){
             var themeCheckBox =
document.getElementById("theme");
             var formContainer =
document.getElementById("formContainer");
             if(themeCheckBox.checked) {
                formContainer.className = "dark-theme";
                document.querySelector("button").className="btn
btn-dark w-100";
            } else {
                formContainer.className = "light-theme";
                document.querySelector("button").className="btn
btn-primary w-100";
            }
        }
    </script>
    <style>
        .form {
            border: 2px solid black;
            padding: 20px;
            width: 250px;
            margin-top: 20px;
        }
        .dark-theme {
            border: 2px solid black;
            padding: 20px;
            width: 250px;
            margin-top: 20px;
            background-color: black;
            color:white;
        }
        .light-theme {
            border: 2px solid black;
            padding: 20px;
            width: 250px;
            margin-top: 20px;
            background-color: white;
            color:black;
        }
      </style>
    </head>
    <body class="container-fluid">
      <div class="form" id="formContainer">
        <div class="form-switch">
          <input type="checkbox" id="theme"
onchange="ChangeTheme()" class="form-check-input"> Dark
Theme
        </div>
        <h2>Register User</h2>
        <dl>
          <dt>User Name</dt>
          <dd><input type="text"></dd>
          <dt>Password</dt>
          <dd><input type="password"></dd>
          <dt>Email</dt>
          <dd><input type="email"></dd>
        </dl>
        <button class="btn w-100">Login</button>
      </div>
    </body>
</html>
```

Note: You can define styles and classes dynamically to any element.

```
     document.querySelector("p").style.color="green";
     document.querySelector("button").className = "btn
      btn-primary";
```

String Manipulations Methods And Properties

Length	It returns the total number of chars in given string. [count of chars]
indexOf()	It returns the index number of specified char. It returns -1 if the given char not found.
lastIndexOf()	It returns the last occurance index number of given char.
charAt()	It returns the character at specified index.
charCodeAt()	It returns the character ASCII code present at specified index.
slice()	It can returns the chars between specified index, which using onlystart and end index reference.
substr()	It can return the specified number of chars from given index.
substring()	It can return the chars from specific index in any direction.
startWith()	It returns boolean true or false by verfiying the string starting chars.
endWith()	It verfies the ending chars in string.
match()	It uses a regular expression to verify the format of value.
trim()	It removes the leading spaces.
split()	It splits the string at specified char.

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>String</title>
```

```
        <script>
           function ChangeCase(){
              var username =
document.getElementById("txtName").value;
              var firstChar = username.charAt(0);
              var restChars = username.substring(1);
              var sentence = firstChar.toUpperCase() +
restChars.toLowerCase();
              document.getElementById("txtName").value =
sentence;

           }
        </script>
     </head>
     <body>
        <fieldset>
           <legend>Test</legend>
           <div>
      <input type="text" id="txtName" onblur="ChangeCase()">
           </div>
        </fieldset>
     </body>
</html>
```

FAQ: What is charCodeAt()?

Ans: charAt() returns char at specified index
        charCodeAt() returns its character code as per UTF
        standards
        A=65, Z=90

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>String</title>
      <script>
         function VerifyName(){
            var username =
document.getElementById("txtName").value;
            if(username.charCodeAt(0)>=65 &&
username.charCodeAt(0)<=90) {
               document.querySelector("p").innerHTML = "";
            } else {
               document.querySelector("p").innerHTML = "Name
must start with uppercase letter".fontcolor('red');
            }
         }
      </script>
   </head>
   <body>
      <fieldset>
         <legend>User Name</legend>
         <div>
             <input type="text" size="40" placeholder="Name must
start with Uppercase letter" id="txtName"
onkeyup="VerifyName()" >
             <p></p>
         </div>
      </fieldset>
   </body>
</html>
```

FAQ: What is difference between slice(), substr() & substring()?

Ans:

- slice() : It can read the chars between specified start and end index.
  - If end is not defined, then it will read upto end of string.
  - End index must be greater than start index.
  - It can read in one direction only. [left to right]

```
string.slice(0,7)        // 0 to 7      valid
string.slice(7)          // 7 to end    valid
string.slice(7,0)        // invalid
```

Syntax:-

- slice(startIndex, endIndex);

- substr() : It can read specified number of chars from given index number.

Syntax: -

```
substr(startIndex, countOfChars|lengthOfChars);
```

```
string.substr(7);        // from 7 to end
string.substr(7,3);      // from 7 index it will read 3 chars.
```

- substring(): It can read from specified index to any direction.
  - it is bi-directional.
  - It can use the end index less than the start index for reading backward.

Syntax:-

```
substring(startIndex, endIndex);
```

Note: endIndex can be less than start index.

```
string.substring(0,7)          // 0 to 7 chars
string.substring(7)            // 7 to end
string.substring(7,0)          // 7 to start [0]
```

Ex:-

```
<script>
      function SentenceCase(str){
            var firstChar = str.charAt(0);
            var restChars = str.substring(1);
            return firstChar.toUpperCase() +
 restChars.toLowerCase();
      }
         var msg = "wELcoME to JaVAscriPT";
         document.write(msg + "<br>");
         document.write(SentenceCase(msg));
</script>
```

Ex:- Split words

```
<script>
   var msg = "Welcome-to-JavaScript";
   var result = msg.split('-');
   document.write(result[1] + "<br>");
   document.write(result[2]);
</script>
```

Task : Write a function that can convert the sentence into titlecase. [Every word first char must be capital letter]

Ans:

```
      <!DOCTYPE html>
      <html lang="en">
<head>
   <title>TitleCase</title>
   <script>
      function titleCase(){
         var msg = ("welcome to javascript");
          splitmsg=msg.split(" ");
         for(i=0;i<splitmsg.length;i++){
            document.write(splitmsg[i].charAt(0).toUpperCase() + 
splitmsg[i].slice(1) + " ");
         }
      }
   </script>
</head>
<body>
      <button onclick="titleCase()">
         Click
      </button>
</body>
</html>
```

IndexOf() & lastIndexOf()

- These are the functions, which can find any character in a string and return its index position.
- If character not found then it returns -1
- lastIndexOf() will return the last occurance index.

```
        Welcome     - indexOf("e")              // 1
        Welcome     - lastIndexOf("e")         // 6
```

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>String</title>
    <script>
      function VerifyEmail(){
        var email = 
document.getElementById("txtEmail").value;
        var msg = document.querySelector("h2");
        var atPos = email.indexOf("@");
        if(atPos<3) {
          msg.innerHTML = "Invalid Email";
        } else {
          msg.innerHTML = "Email Verified";
        }
      }
    </script>
  </head>
  <body>
    <fieldset>
      <legend>Your Email</legend>
      <input type="text" id="txtEmail"> <button 
onclick="VerifyEmail()">Submit</button>
    </fieldset>
    <h2></h2>
  </body>
</html>
```

startsWith() & endsWith():

- These are boolean functions that return true when given set of chars are defined as prefix or suffix of any any string
- These are the functions used to verify the starting and ending chars in a string.
- These functions return boolean true when string is starting or ending with specified chars.

Syntax:-

```
string.startsWith(chars);
string.endsWith(chars);
```

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>String</title>
    <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
    <script>
      function VerifyCard(){
        var cardnumber =
document.getElementById("txtCard").value;
        var cardImg = document.getElementById("cardImg");
        if(cardnumber.startsWith("4444")){
          cardImg.src = "../public/images/master.png";
        } else if(cardnumber.startsWith("5555")) {
          cardImg.src = "../public/images/visa.png";
        } else {
          cardImg.src="";
```

```
                cardImg.alt = "N/A";
            }
        }
        function VerifyEmail(){
            var email =
document.getElementById("txtEmail").value;

            if(email.endsWith("gmail.com")) {
                document.getElementById("msg").innerHTML =
"Email Verified";
            } else {
                document.getElementById("msg").innerHTML =
"Invalid Email";
            }
        }
    </script>
  </head>
  <body class="container-fluid">
    <fieldset>
        <legend>Verify Details</legend>
        <div>
            <h3>Card Number</h3>
            <div class="input-group">
                <input type="text" id="txtCard"
onkeyup="VerifyCard()" class="form-control">
                <img class="input-group-text" id="cardImg"
class="img-fluid" width="100" height="50">
            </div>
            <h3>Your Gmail</h3>
            <div>
                <input type="text" onkeyup="VerifyEmail()"
```

```
id="txtEmail" class="form-control">
          <div id="msg">
          </div>
        </div>
      </div>
    </fieldset>
  </body>
</html>
```

match() :

- It matches to verify and compare given string with any regular expression. It returns boolean true or false.
- if string format is as per regular expression.
- In JavaScript regular expression is enclosed in " / / ".

Syntax:-

```
/your Expression/;
    yourString.match(regularExpression);   //true - false
        or
var regExp = /pattern/;
var string = " ";

if(string.match(regExp))
{

}
```

Ex:-1

```
   <!DOCTYPE html>
   <html>
 <head>
```

```
    <title>MatchDemo</title>
    <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
    <style>
      meter {
        height: 30px;
      }
    </style>
    <script>
      function VerifyPassword(){
        var password =
document.getElementById("txtPwd").value;
        var regExp = /(?=.*[A-Z])\w{4,15}/;
        var msg = document.getElementById("msg");
        var grade = document.getElementById("grade");

        function ShowGrade(min, max, value, low, high) {
          grade.min = min;
          grade.max = max;
          grade.value = value;
          grade.low = low;
          grade.high = high;
        }
        if(password=="") {
          msg.innerHTML = "Password Required";
          msg.className = "text-danger";
        } else {
            if(password.match(regExp)) {
            msg.innerHTML = "Strong Password";
            msg.className = "text-success";
            ShowGrade(1,100,100,0,0);
```

```
                } else {
                if(password.length<4) {
                    msg.innerHTML = "Poor Password";
                    msg.className = "text-danger";
                    ShowGrade(1,100,100,60,80);
                } else {
                    msg.innerHTML = "Weak Password";
                    msg.className = "text-warning"
                    ShowGrade(1,100,100,40,80);
                }
            }
        }
    }
    function VerifyMobile(){
        var mobile =
document.getElementById("txtMobile").value;
        var mobileExpression = /\+91\d{10}/;
        var mobileError =
document.getElementById("mobileError");
        if(mobile.match(mobileExpression)) {
            mobileError.innerHTML = "";
        } else {
            mobileError.innerHTML = "Invalid Mobile";
            mobileError.className = "text-danger";
        }
    }
  </script>
 </head>
 <body class="container-fluid">
  <h2>Verify Password</h2>
  <input type="password" onkeyup="VerifyPassword()"
```

```
id="txtPwd" class="form-control">
        <div id="msg"></div>
        <div>
            <meter id="grade" min="1" max="100" class="w-
100"></meter>
        </div>
        <h2>Verify Mobile</h2>
        <input type="text" onkeyup="VerifyMobile()"
id="txtMobile" class="form-control">
        <div id="mobileError"></div>
    </body>
</html>
```

Ex:-2

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>String Match</title>
        <script>
            function VerifyPassword(){
                var password =
document.getElementById("txtPwd").value;
                var regExp = /(?=.*[A-Z])\w{4,10}/;
                var error = document.getElementById("error");

                function ShowGrade(min, max, value){
                    var grade = document.getElementById("grade");
                    grade.min = min;
                    grade.max = max;
                    grade.value = value;
                }
```

```
          if(password.match(regExp)) {
            error.innerHTML = "Strong Password";
            error.style.color= "green";
            ShowGrade(1,100,100);
          } else {
            if(password.length<4){
              error.innerHTML = "Poor Password";
              error.style.color = "red";
              ShowGrade(1,100,20);
            } else {
              error.innerHTML = "Weak Password";
              error.style.color = "goldenrod";
              ShowGrade(1,100,60);
            }
          }
        }
      </script>
    </head>
    <body>
      <fieldset>
        <legend>Your Password</legend>
        <input type="password" onkeyup="VerifyPassword()"
id="txtPwd">
        <div>
          <meter id="grade" style="width: 150px; height:
20px;"></meter>
        </div>
        <div id="error"></div>
      </fieldset>
    </body>
</html>
```

trim() & split() :

- trim() is used to remove the leading spaces in a string.
- split() is used to split the string at specified delimeter and return an array.

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>Trim demo</title>
      <script>
         function VerifyPassword(){
            var pwd = document.getElementById("txtPwd").value;
            if(pwd.trim()=="admin") {
               document.querySelector("h2").innerHTML =
"Verified";
            } else {
               document.querySelector("h2").innerHTML = "Invalid
Password";
            }
         }
      </script>
   </head>
   <body>
      Your Password :
      <input type="text" id="txtPwd"> <button
onclick="VerifyPassword()">Submit</button>
      <h2></h2>
   </body>
</html>
```

Ex:-

```
<script>
        var mobileNumbers = "9876543210, 9988877663, 
8893928111";
        var result = mobileNumbers.split(',');
        document.write(result[2]);
</script>
```

Boolean Type

- In computer programming boolean is used for decision making.
- Boolean can control the execution flow.
- Boolean type can handle only 2 keywords : true and false.
- However in JavaScript boolean types can be verified by using 1 and 0.

```
    true        = 1
    false       = 0
```

Syntax:-

```
var stock = true;

if(stock==true)               // valid
if(stock==1)                  // valid
```

Ex:-

```
<script>
```

```
        var stock = true;
        if(stock==1) {
      document.write("Stock : Available");
  } else {
      document.write("Stock : N/A");
  }
</script>
```

Undefined Type

- It is a type defined for variable or memory reference when value is not defined.

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

- The keyword "undefined" is used to verify the existiance of value in any reference.

Ex:-

```
    <script>
        var name = "Samsung TV";
        var price;
        if(price==undefined){
           document.write(`Name=${name}`);
        } else {
        document.write(`Name=${name}<br>Price=${price}`);
        }
    </script>
```

FAQ: What is difference between undefined and not-defined?

Ans:- undefined is a type.

- it specifies that memory reference is present but not assigned with value.
- notdefined is an exception
- it specifies that there is no memory reference that your are trying to use.

FAQ: What is difference between null and undefined?

Ans:- undefined is configured at compile time.
null is configured at run time.

- If there is no value during compile time then it is set to undefined.
- if there is no value during run time then it is set to null.

Null Type

- It is a data type defined for memory reference when value is not supplied during run time.
- You can verify the value during run time by using "null" keyword.

Ex:-

```
<script>
        var name = "Samsung TV";
        var price  = prompt("Enter Price");
        if(price==null) {
            document.write("Please Enter Price");
        } else if (price=="") {
            document.write("Price can't be Empty");
        } else {
         document.write(`Name=${name}<br>Price=${price}`);
        }
    </script>
```

Summary Primitive Data Types

- number	: numeric values
- string	: literals
- boolean	: true / false
- null	: no value at run time
- undefined	: no value at compile time

Non-Primtive Types

- They are mutable types.
- Their structure can change according to state and situation.
- No fixed range for value.
- Value range will change accroding to memory available.

- They are store in heap memory.
- JavaScript Non Primitive types are
    a) Array
    b) Object
    c) Map

Array Type

- Arrays are used in computer programming to reduce overhead and complexity.
- Arrays can reduce overhead by storing values in sequential order.
- Arrays can reduce complexity by storing multiple values under one name.
- Arrays can store various types of values.
- Arrays size can change dynamically.
- Array behaviour allow to store values in sequential order and access in random.
- In JavaScript array can have the behaviour of collection.
- It can handle LIFO, FIFO, access in random.

Note:- few Technologies can't allocate various types of memory in sequential order hence they restrict array to same type of values and size can't be changed dynamically. [c, Java, .net language]

Configure Array:-

1. Declaring Array
2. initialization of memory for Array.

Declaring Array:

- Array is declared same as other variables in JavaScript using var, let or const.

```
var  values;
let products;
const sales;
```

- Array declaration is not enough to store values.
- You have to initialize or assign memory for array.
- Memory is allocated for array by using
  - a) [ ]   - Meta character
  - b) Array()   - Array Constructor

Syntax:-

```
var categories;
categories = [ ];                    // assigning memory
var categories = [];             // initialization of memory
             (or)
var categories;
categories = new Array();
var categories = new Array();
```

FAQ: What is difference between "[]" and "Array()" ?

Ans: "[]" configures undefined structure for array.
 "Array()" configure a specified structure for array.
 However you can change the structure.

```
var categories = new Array(2);
var categories = [ ];
```

"new" => It is dynamic memory allocating operator.

Storing values into Array:

- You can store values into array by using the reference of array properties.
- Every array property is a string type.
- Each property maps to specific index number.

Syntax:-

```
var categories = [];
categories[0] = "Samsung TV";              // valid
categories["1"] = 45000.55;                // valid

0      - string            samsung tv     - string
1      - string            45000.55        - number
```

Ex:-

```
<script>
   var categories = [];
   categories[0] = "Samsung TV";
   categories[1] = true;
   categories[2] = 45000.55;
   categories["3"] = "Electronics";

   for(var property in categories)
   {
    document.write(`${property} [${typeof property}] -
${categories[property]} [${typeof
categories[property]}]<br>`);
   }
</script>
```

You can store any type of value in array

   a) Primitive Type
   b) Non Primitive Type
   c) Function

Syntax:-

```
var values = [number, string, boolean, array, function(){}];
```

Ex:-

```
    var values = [10, "TV", true, ['Delhi','Hyd'],
    function(){document.write('Function in Array')}];
```

Reading Values from Array

- You can read values from array by refering to the properties of array.

Syntax:-

```
    values[0];
    values["1"];

    values[0][]            - accessing array defined at  0.
```

```
values[0]()             - accessing function defined at 0.
```

Ex:-

```
<script>
   var values = [10, "TV", true, ['Delhi','Hyd'],
function(){document.write('Function in Array')}];
    document.write(values[3][1] + "<br>");
    values[4]();
</script>
```

- ES5 introduced array "destructing"

EX:-

```
<script>
    var values = [10, "TV", true, ['Delhi','Hyd'],
function(){document.write('Function in Array')}];
    var [id, name, stock, cities, print] = values;
    print();
    document.write("<br>"+ cities[1]);
</script>
```

FAQ: Why we need a function in Array?

Ans: Functions in array are used to handle call back mechanism.

- Call back mechanism allows function to execute according to situation.
- Call back function can't have a name. It must be anonymous.

```
function name(){ }        // invalid
function(){}              // valid - anonymous
```

Array Manipulations [Array Method and Properties]

Reading Array Elements:

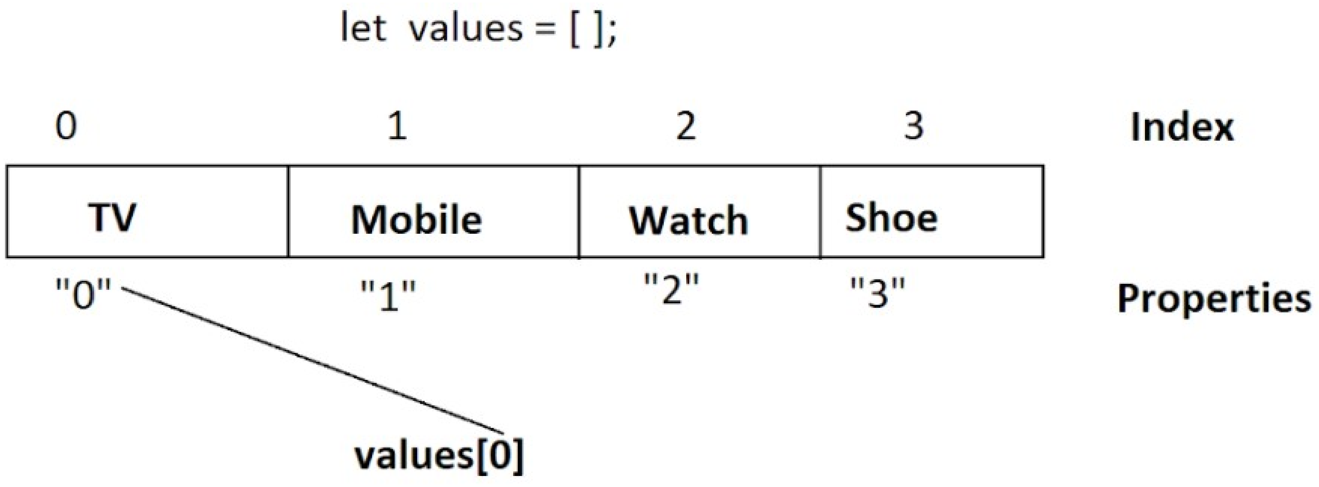
1. toString() - returns array elements separated with ","
2. join() - It is similar to string by uses custom delimeter.
3. slice() - It can access elements between specified index.
4. find() - returns only the first elements that match given condition
5. filter() - returns all elements that match given condition
6. loops()
7. Iterators()
8. map() - It is an iterator for presenting elements.

Ex:- toString(), join(), slice()

```
<script>
    var categories = ["Electronics", "Footwear", "Fashion"];
    document.write(categories.slice(0,2));
    document.write(categories.toString());
    document.write(categories.join('-->'));
</script>
```

Storing Values into Array:

- Values are stored and accessed from an array by using property.
- Property maps to Index in memory.



Syntax:-

```
let values = [];
let values[0] = 10;                  // valid
let values["1"] = 20;            // valid

document.write(values["0"]);
document.write(values[1];
```

Syntax:- Initialization of Values

```
let values = [10, 20, 30];
let values = new Array(10,20,30)
```

Ex1:-

```
<script>
    let sales = [34000, 32000, 60000, 42500, 52000];
```

```
        document.write(sales.toString());
    </script>
```

Ex2:-

```
    <script>
      let sales = [34000, 32000, 60000, 42500, 52000];
      document.write(sales.join("-->"));
    </script>
```

Ex3:-

```
    <script>
      let sales = [34000, 32000, 60000, 42500, 52000];
      document.write(sales.slice(1,3));
    </script>
```

Filtering and Finding Values:

- Array provides "find() and filter()" methods for searching of values in a collection.
- find() returns the first occurance value from collection that matches your condition.
- filter() returns all values that match your condition.

Syntax:-

```
        collection.find(function(value){
           return  value|condition;
        })
         collection.filter(function(value){
             return  value|condition;
           })
```

Ex:-

```
    <script>
      var sales = [34000, 56000, 70000, 23000, 42000];
      document.write(sales.filter(function(value){
         return value > 40000;
      }));
    </script>
```

Ex4:- Filter

```
    <script>
          let sales = [34000, 32000, 60000, 42500, 52000];
          function GetData(value){
          return value > 50000;
         }
           var result = sales.filter(GetData);
           document.write(result);
    </script>
```

or

```
    <script>
      let sales = [34000, 32000, 60000, 42500, 52000];
      document.write(sales.filter(function(value){return
value<50000}));
    </script>
```

Ex5: Find

```
	<script>
	  let sales = [34000, 32000, 60000, 42500, 52000];
	  document.write(sales.find(function(value){return
value>50000}));
	</script>
```

Map() :

- map() is an implicit iterator.
- It is faster in accessing value from collection.
- It uses a callback function to read values and present in UI.
- It is an Iterator - Software Design Pattern
- It can read elements from a collection in sequential order.
- It doesn't require any condition, initialization and counter.

Syntax:-

```
	collection.map(function(value){

	    -- present value --

	})
   <script>
```

```
        let sales = [34000, 32000, 60000, 42500, 52000, 56000];
        sales.map(function(value){
        document.write(`<li>${value}</li>`)
         })
     </script>
                         or

<script>

        var categories = ["Electronics", "Footwear", "Fashion"]

        categories.map(function(value)){

          document.write(`<li>${value}</li>`);

        })

</script>
```

Accessing Array Elements using Loops:

- A loop comprises set of statements to execute repeatedly until the condition is satisfied.
- It requires initialization, condition and counter.
- You define loops using for, while and do while statements.

Syntax: Loop

```
for (initialization ; condition; counter)
{
}
```

```
               or
        for(var i = 0; i< array.length; i++)
        {
            document.write(array[i]);
        }
```

Ex:-

```
        <script>
          var categories = ["Electronics", "Footwear", "Fashion"];
          for(var i=0; i<categories.length; i++) {
               document.write(categories[i] + "<br>");
           }
        </script>
```

Access Array Elements using Iterators:

- Iterator is a deisgn pattern used to access elements from a collection in sequential order.
- It doesn't require initialization, condition and counter.
- Iterators can be designed by using
  - "for..in"
  - "for..of"
- for..in statement is used to access all properties of collection.
- for..of statement is used to access all values from collection.
- Reading values by using loops and external iterators

```
            for()                          ]
```

```
        while()                ]  loops
        do while()             ]
        for..in                   // iterator of properties
        for..of                  // iterator of values
```

Syntax:-

```
    for(var property in collection)
    {
    }

    for(var value of collection)
    {
    }
```

Ex:- Read both properties and values from collection

```
    <script>
  var categories = ["Electronics", "Footwear", "Fashion"];
  for(var property in categories)
  {
    document.write(`[${property}]-${categories[property]}
<br>`);
  }
</script>
```

Ex: Iterator

```
   <script>
```

```
 let sales = [34000, 32000, 60000, 42500, 52000, 56000];
 for(var property in sales)
 {
  document.write(`[${property}] ${sales[property]}<br>`);
  }
</script>
```

Ex:-

```
 <script>
let sales = [34000, 32000, 60000, 42500, 52000, 56000];
 for(var value of sales)
 {
 document.write(value + "<br>");
 }
 </script>
```

Dynamically Creating and Adding DOM elements using Array

1. You can create any HTML element with JavaScript by using

   ```
   document.createElement("elementName");
   document.createElement("h2");
   document.createElement("img");
   ```

2. You can add element dynamically into page by using the method

   ```
   appendChild()
   ```

```
        append()
        prepend()
```

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>Dynamic</title>
    <script>
      function AddImage(){
        var pic = document.createElement("img");
        pic.src = "../public/images/neckband.png";
        pic.width = "100";
        pic.height = "100";
      document.getElementById("container").appendChild(pic);
      }
    </script>
  </head>
  <body>
    <div>
      <button onclick="AddImage()">Add Image to Page

      </button>
    </div>
    <div id="container">
    </div>
  </body>
```

```
</html>
```

Ex:-

```
     <!DOCTYPE html>
     <html>
  <head>
    <title>Dynamic</title>
    <script>
      var products =
["../public/images/neckband.png","../public/images/desktop.png
","../public/images/mobile.png"];
      function bodyload(){
        for(var path of products)
        {
          var pic = document.createElement("img");
          pic.src = path;
          pic.width = "100";
          pic.height = "100";

 document.getElementById("container").appendChild(pic);

          var option = document.createElement("option");
          option.text = path;
          document.querySelector("select").appendChild(option);

          var li = document.createElement("li");
          li.innerHTML = path;
```

```
            document.querySelector("ol").appendChild(li);
          }
        }
      </script>
   </head>
   <body onload="bodyload()">
      <div id="container">
      </div>
      <h2>Select Path</h2>
      <select>
      </select>
      <h2>Image Path List</h2>
      <ol>
      </ol>
   </body>
</html>
```

Presenting Array Elements in UI:

- Create a new element dynamically.

```
 document.createElement("h2, p, img, table, li, ol etc..");
```

- Define properties for element

```
      var pic  = document.createElement("img");
      pic.width="";
      pic.height="";
      pic.src="";
      pic.alt="";
```

- Add element into page.
- You need a container, which is the parent element that contains the dynamic element you created.

```
parentId.appendChild(yourDynamicElement);

container.appendChild(pic);
```

Ex:- Dynamically adding images into page.

```
<!DOCTYPE html>
<html>
<head>
  <title>Dynamic Elements</title>
  <script>
    var images = 
["../public/images/shoe.jpg","../public/images/shoe1.jpg","../pu
blic/images/neckband.png"];

    function AddClick(){
      for(var imagePath of images)
      {
        var img = document.createElement("img");
        img.width="100";
        img.height="100";
        img.border="1";
        img.src=imagePath;
document.getElementById("container").appendChild(img);
```

```
            }
        }
      </script>
   </head>
   <body>
      <div>
         <button onclick="AddClick()">Add Image</button>
      </div>
      <br>
      <div id="container">
      </div>
   </body>
</html>
```

Ex:- Presenting Array Elements into UI with <ol> <select><table>

```
     <!DOCTYPE html>
     <html>
   <head>
      <title>Array Manipulation</title>
      <script>
         var categories = ["All","Electronics", "Footwear",
"Fashion", "Men's Clothing"];
         function bodyload(){
            for(var category of categories)
            {
               var li = document.createElement("li");
               li.innerHTML = category;
```

```
                document.querySelector("ol").appendChild(li);
                var option = document.createElement("option");
                option.text = category;
document.querySelector("select").appendChild(option);

                var row = document.createElement("tr");
                var cell = document.createElement("td");
                cell.innerHTML = category;
                row.appendChild(cell);
   document.querySelector("tbody").appendChild(row);
              }
          }
      </script>
   </head>
   <body onload="bodyload()">
      <h3>Categories List</h3>
      <ol>
      </ol>
      <h3>Select Category</h3>
      <select>
      </select>
      <h3>Categories Menu</h3>
      <table border="1" width="200">
          <thead>
              <tr>
                  <th>Products Category</th>
              </tr>
          </thead>
          <tbody>
          </tbody>
        </table>
    </body>
</html>
```

Ex: CheckBox List

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Array Manipulation</title>
        <script>
            var categories = ["All","Electronics", "Footwear",
"Fashion", "Men's Clothing"];
            function bodyload(){
                for(var category of categories)
                {
                    var li = document.createElement("li");
                    li.innerHTML = `<input type="checkbox">
${category}`;
                    document.querySelector("ul").appendChild(li);
                }
            }
        </script>
        <style>
            ul {
```

```
            width: 200px;
            border:2px solid black;
            height: 50px;
            overflow: auto;
            padding: 20px;
          }
        </style>
    </head>
    <body onload="bodyload()">
        <h3>Categories List</h3>
        <ul style="list-style: none;">
        </ul>
    </body>
</html>
```

Reading Values from Array and Presenting:

- Creating a new Element

  ```
  document.createElement("tagName");
  ```

- Adding element into page

  ```
  appendChild(elementReference);
  ```

Ex:-

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Arrays</title>
        <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
        <script>
```

```
        var categories = ["Electronics", "Footwear", "Fashion"];
        function bodyload(){
          for(var category of categories)
          {
            var li = document.createElement("li");
            li.innerHTML = category;
            document.querySelector("ol").appendChild(li);

            var option = document.createElement("option");
            option.text = category;
document.querySelector("select").appendChild(option);
            var tr = document.createElement("tr");
            var td = document.createElement("td");
            td.innerHTML = category;
            tr.appendChild(td);
            document.querySelector("tbody").appendChild(tr);
          }
        }
      </script>
    </head>
    <body onload="bodyload()">
      <div class="container-fluid">
        <h2 class="text-center bg-danger text-white">
      Arrays</h2>
        <div class="row">
          <div class="col">
            <h3>Categories List</h3>
            <ol>
            </ol>
          </div>
          <div class="col">
```

```
                <h3>Select Category</h3>
                <select class="form-select">
                </select>
              </div>
              <div class="col">
                 <h3>Categories Table</h3>
                 <table class="table table-hover table-dark">
                    <thead>
                      <tr>
                         <th>Categories</th>
                      </tr>
                    </thead>
                    <tbody>
                    </tbody>
                 </table>
              </div>
           </div>
        </div>
     </body>
</html>
```

FAQ: What type of data we can store in Array?

Ans : Array can handle any type of data, both primitive, non primitive and functions.

Ex:-

```
     <script>
        var values = ["A",1000,true,['TV',
'Mobile'],function(){document.write('Function in Array')}];
   document.write(values[3][1] + "<br>");
   values[4]();
```

```
        </script>
```

FAQ: Why we need a function in Array?

Ans : Array is defined with functions to handle call back mechanism.
Call back is a technique where functions will execute according to situation.

Note: JavaScript E5+ introduced Array Destruction.

Ex:- Without Destruction

```
        var  values = [10, "John"];
        var  sno = values[0];
        var  name = values[1];
```

Ex:- With Destruction

```
        var  values = [10, "John"];
        var [sno, name] = values;
        var [sno, name, salary] = values;      // salary = undefined
        var [sno] = values;                    //  sno = 10
```

Ex:- Array Destruction and Callback

```
        <script>
   var authorize = ["admin",function(){document.write("Login
Success")}, function(){document.write("Invalid Password")}];

   var [password, success, failure] = authorize;

   var yourPassword = prompt("Enter Password");
   if(password==yourPassword){
      success();
   } else {
      failure();
   }
</script>
```

Note: Function in Array must be Anonymous.
        Anonymous functions will not have a name.

```
      function(){ }                          // Anonymous
      function success() { }                // invalid in array
```

Adding Values into Array:

1. push()   : Add new values as last elements.
2. unshift()   : Add new values as first elements.
3. splice()   : Add new values at any specific position.

Syntax:-

```
      arrayName.push("Item1", "Item2",..);

      arrayName.unshift("Item1", "Item2",..);

      arrayName.splice(startIndex, deleteCount, "Item1","Item2");
```

Ex:-

```
      categories.splice(1,0, "Men's Clothing");
      <script>
   var categories = ["electonics","Footwear","Fashion"];
   categories.unshift("All");
   categories.splice(2,0,"Men's Clothing","Women's Clothing");
   for(var property in categories)
   {
        document.write(`[${property}]-
${categories[property]}<br>`);
    }
</script>
```

Removing Values from Array:

1. pop() : It removes and returns the last item.
2. shift() : It removes and returns the first item.
3. splice() : It removes and returns item at specific index.

Syntax:-

```
        arrayName.pop()
        arrayName.shift()
        arrayName.splice(startIndex, deleteCount)
```

Ex:-

```
        categories.pop()
        categories.shift()
        categories.splice(1,2);        // removes 2 items from 1 index.
                                or
        <script>
    var categories = ["All","Electronics","Footwear","Fashion"];
    alert(categories.splice(2,2)+"-Removed");
    for(var property in categories)
    {
      document.write(`[${property}]-${categories[property]}<br>`);
    }
</script>
```

Ex:- Dynamically Adding and Removing Items from Array

```
        <!DOCTYPE html>
```

```
      <html>
   <head>
      <title>Array Manipulations</title>
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
      <script>
         var cities = ["Delhi", "Hyd"];
         function LoadCities(){
            document.getElementById("lstCities").innerHTML ="";
            for(var city of cities){
               var option = document.createElement("option");
               option.text = city;

document.getElementById("lstCities").appendChild(option);
            }
         }
         function bodyload(){
            LoadCities();
         }
         function AddClick(){
            var cityname =
document.getElementById("txtCity").value;
            var cityIndex = cities.indexOf(cityname);
            if(cityIndex==-1) {
               cities.push(cityname);
               alert(`${cityname} Added to List`);
               LoadCities();
               document.getElementById("txtCity").value = "";
            } else {
               alert("City Exists - Try Another");
            }
```

```
          }
          function RemoveClick(){
             var selectedCityName =
document.getElementById("lstCities").value;
             var selectedCityIndex =
cities.indexOf(selectedCityName);
             alert(`${cities.splice(selectedCityIndex,1)} Removed`);
             LoadCities();
          }
       </script>
    </head>
    <body class="container-fluid" onload="bodyload()">
       <h2>Array Manipulations</h2>
       <div class="row">
          <div class="col-3">
             <fieldset>
                <legend>Add New City</legend>
                <div class="input-group">
                   <input type="text" id="txtCity" class="form-
control">
                   <button onclick="AddClick()" class="btn btn-
primary">Add</button>
                </div>
             </fieldset>
          </div>
          <div class="col-9">
             <h3>Cities List</h3>
             <div class="input-group">
                <select class="form-select" id="lstCities">

                </select>
                <button onclick="RemoveClick()" class="btn btn-
danger">Remove City</button>
            </div>
        </div>
    </div>
  </body>
</html>
```

Ex:- 2

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>Array Manipulations</title>
    <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
    <link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
    <script>
      var categories = ["All", "Electronics", "Footwear"];
      function LoadCategories(){
        var lstCategories =
document.getElementById("lstCategories");
        lstCategories.innerHTML="";
        for(var item of categories)
        {
          var option = document.createElement("option");
          option.text = item;
          lstCategories.appendChild(option);
        }
      }
      function GetCategoriesCount(){
            document.getElementById("count").innerHTML =
categories.length - 1;
        }
        function bodyload(){
            LoadCategories();
            GetCategoriesCount();
        }
        function AddClick(){
            var categoryName =
document.getElementById("txtName");
            if(categories.indexOf(categoryName.value)==-1) {
                categories.push(categoryName.value);
                LoadCategories();
                alert(`${categoryName.value} Added to List`);
                categoryName.value = "";
                GetCategoriesCount();
            } else {
                alert(`${categoryName.value} Exists`);
            }
        }
        function RemoveClick(){
            var selectedCategory =
document.getElementById("lstCategories").value;
            var categoryIndex =
categories.indexOf(selectedCategory);
            var flag = confirm(`Are you sure?\nWant to delete
${selectedCategory}?`);
            if(flag==true){
                categories.splice(categoryIndex,1);
                alert(`${selectedCategory} Removed`);
                LoadCategories();
                GetCategoriesCount();
            }
        }
    </script>
  </head>
  <body class="container-fluid" onload="bodyload()">
    <fieldset>
        <legend>Array Manipulations</legend>
        <div class="row">
            <div class="col-6">
              <h3>Add New Category</h3>
              <div class="input-group">
                 <input type="text" id="txtName" class="form-
control">
                 <button onclick="AddClick()" class="btn btn-
primary">Add</button>
              </div>
            </div>
            <div class="col-6">
                <h3>Categories List</h3>
                <select size="3" class="form-select"
id="lstCategories">
                </select>
                <div class="mt-2">
                    <span class="bi bi-cart4 text-warning"></span>
                    <span>Categories Count <span class="badge bg-
dark text-light" id="count"></span> </span>
                </div>
                <div class="mt-2">
                    <button class="btn btn-danger"
onclick="RemoveClick()">
```

```
                         <span class="bi bi-trash-fill"></span> Remove
Category
                      </button>
                   </div>
                </div>
             </div>
          </fieldset>
   </body>
</html>
```

Sorting Array Elements

1. sort() : It sorts the items in ascending order.
2. reverse() : It sorts the items in reverse order of their occurance. [Last to First]

Syntax:-

```
arrayName.sort()
arrayName.reverse()
```

Ex:-

```
      <script>
   var cities =
["Delhi","Chennai","Bangalore","Hyd","Mumbai","Goa"];
   cities.sort();
   cities.reverse();
   for(var item of cities)
   {
      document.write(item + "<br>");
   }
</script>
```

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>Array Manipulations</title>
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
      <script>
         var cities = ["Delhi", "Hyd"];
         function LoadCities(){
            document.getElementById("lstCities").innerHTML ="";
            for(var city of cities){
               var option = document.createElement("option");
               option.text = city;

document.getElementById("lstCities").appendChild(option);
            }
         }
         function bodyload(){
            LoadCities();
         }
         function AddClick(){
            var cityname =
document.getElementById("txtCity").value;
            var cityIndex = cities.indexOf(cityname);
            if(cityIndex==-1) {
               cities.push(cityname);
               alert(`${cityname} Added to List`);
               LoadCities();
               document.getElementById("txtCity").value = "";
            } else {
                alert("City Exists - Try Another");
            }
        }
        function RemoveClick(){
            var selectedCityName =
document.getElementById("lstCities").value;
            var selectedCityIndex =
cities.indexOf(selectedCityName);
            alert(`${cities.splice(selectedCityIndex,1)} Removed`);
            LoadCities();
        }
        function SortReverse(){
            cities.reverse();
            LoadCities();
        }
        function SortAsc(){
            cities.sort();
            LoadCities();
        }
    </script>
  </head>
  <body class="container-fluid" onload="bodyload()">
    <h2>Array Manipulations</h2>
    <div class="row">
        <div class="col-3">
            <fieldset>
                <legend>Add New City</legend>
                <div class="input-group">
                    <input type="text" id="txtCity" class="form-
control">
```

```
                <button onclick="AddClick()" class="btn btn-
primary">Add</button>
              </div>
            </fieldset>
          </div>
          <div class="col-9">
            <h3>Cities List</h3>
            <div class="input-group">
              <select class="form-select" id="lstCities">
              </select>
              <button onclick="RemoveClick()" class="btn btn-
danger">Remove City</button>
              <button onclick="SortReverse()" class="btn btn-
info">Reverse Sort</button>
              <button onclick="SortAsc()" class="btn btn-
warning">Sort</button>
            </div>
          </div>
        </div>
      </body>
</html>
```

Object Types

- The concept of "object" into computer programming was introduced in early 1960's by "Alan Kay".
- To keep related data and logic under one reference.
- Object store data in properties and defines logic using function.

- Object stores data in properties.

- Object Encapsualtes data and logic i.e propertiesand functions.
- Object is a Key and Value collection.
- Keys are string type.
- Values can be any type.

Syntax:-

```
let object = {
              "Key": value,
              "Key": value,
              "Key": function(){ }
        }
```

- object comprises of "Keys" and "Values"
- The properties of object can be accessed within object by using "this" keyword.
- Outside object you can access with reference of object name.

```
object
{
    this.Key
}
object.Key;
```

- Object acts as a reusable template with sample data and logic which you can implement and customize according to requirement.

- Hence object is also known as "Pseudo Class".

- If only data is defined in object is representing a format of data then it is know as JSON - JavaScript Object Notation

Ex:-

```
<Script>
        vr tv = {
                "Name": "Samsung TV",
                "Price": 45000.55,
                "Stock": true
        };
        document.write(`Name : 
${tv.Name}<br>Price=${tv.Price}<br>Stock=${tv.stock}`);
</script>
```

Ex:-

```
<script>
    let product = {
        "Name": "",
        "Price": 0,
        "Stock": false,
        "Qty": 0,
        "Cities": [],
            "Rating": {Rate:0, Count:0},
            "Total": function(){
               return this.Qty * this.Price;
            },
            "Print": function(){

document.write(`Name=${this.Name}<br>Price=${this.Price}<br>
Stock=${this.Stock}<br>Quantity=${this.Qty}<br>Total=${this.Tot
al()}<br>Shipped
To=${this.Cities.toString()}<br>Rating=${this.Rating.Rate}<br>Rat
ing From ${this.Rating.Count} People <br>`);
      }
   }
   document.write(`<h2>TV Details</h2>`);
   product.Name = "Samsung TV";
   product.Price = 56000.55;
   product.Qty = 2;
   product.Stock = true;
   product.Rating.Rate = 4.5;
   product.Rating.Count = 300;
   product.Cities = ["Delhi", "Hyd"];
   product.Print();
   document.write(`<h2>Shoe Details</h2>`);
   product.Name = "Nike Casuals";
   product.Price = 5000.44;
   product.Qty = 3;
   product.Stock = true;
```

```
    product.Rating.Rate = 4.8;
    product.Rating.Count = 100;
    product.Cities = ["Mumbai", "Hyd", "Chennai"];
    product.Print();
</script>
```

Array of Objects

- It is a collection of objects

Syntax:-

```
    [
      { },
      { }
    ]
```

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>JSON</title>
    <script>
      var products = [
        {Name: "boAt Neckband", Price: 5600.45,
Photo:"../public/images/neckband.png"},
        {Name: "Laptop", Price: 67000.44, Photo:
"../public/images/laptop.png"},
        {Name: "Mobile", Price: 12000.44, Photo:
```

```
"../public/images/mobile.png"},
            {Name: "Desktop", Price: 34000.44,
Photo:"../public/images/desktop.png"}
         ];
         function bodyload(){
            for(var product of products)
            {
               var tr = document.createElement("tr");
               var tdName = document.createElement("td");
               var tdPrice = document.createElement("td");
               var tdPhoto = document.createElement("td");

               tdName.innerHTML = product.Name;
               tdPrice.innerHTML = product.Price;

               var img = document.createElement("img");
               img.src = product.Photo;
               img.width = "100";
               img.height="100";
               tdPhoto.appendChild(img);

               tr.appendChild(tdName);
               tr.appendChild(tdPrice);
               tr.appendChild(tdPhoto);
               document.querySelector("tbody").appendChild(tr);
            }
         }
```

```
        </script>
    </head>
    <body onload="bodyload()">
        <table border="1" width="700">
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Price</th>
                    <th>Preview</th>
                </tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </body>
</html>
```

Ex:- Card

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Cards</title>
        <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
        <link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
        <script>
```

```
            var products = [
              {Name: "boAt Neckband", Price: 5600.45,
Photo:"../public/images/neckband.png"},
              {Name: "Laptop", Price: 67000.44, Photo:
"../public/images/laptop.png"},
              {Name: "Mobile", Price: 12000.44, Photo:
"../public/images/mobile.png"},
              {Name: "Desktop", Price: 34000.44,
Photo:"../public/images/desktop.png"}
            ];
            function bodyload(){
              for(var product of products)
              {
                var card = document.createElement("div");
                card.className="card w-25 m-2 p-2";
                card.innerHTML = `
                 <img src=${product.Photo} height="200"
class="card-img-top">
                 <div class="card-header">
                   <h2>${product.Name}</h2>
                 </div>
                 <div class="card-body">
                  <p> &#8377; ${product.Price}</p>
                 </div>
                 <div class="card-footer">
                   <button class="btn btn-danger w-100">
                    <span class="bi bi-cart4"></span> Add to Cart
                    </button>
                  </div>
                 `;
document.getElementById("catalog").appendChild(card);
              }
            }
        </script>
    </head>
    <body class="container-fluid" onload="bodyload()">
        <div id="catalog" class="d-flex flex-wrap justify-content-between">

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

Ex:- Nested Interations

1. data/menu.json

```
            [
              {
                "Category": "Electronics",
                "Products": ["Samsung TV", "Mobile"]
              },
            {
                "Category": "Footwear",
                "Products": ["Nike Causual", "Lee Cooper Boot"]
            },
            {
                "Category": "Fashion",
                "Products": ["Shirt", "Jeans"]
            }
        ]
```

2. home.html

```
        <!DOCTYPE html>
        <html>
      <head>
        <title>Fetch Data</title>
        <script>
           function GetCategories(){
              fetch("../data/menu.json")
              .then(function(response){
                 return response.json();
              })
              .then(function(data){
                 for(var item of data)
                 {
                     var li = document.createElement("li");
                     li.innerHTML = item.Category;
                     document.querySelector("ol").appendChild(li);
                     var optionGroup =
document.createElement("optgroup");
                     optionGroup.label = item.Category;
document.querySelector("select").appendChild(optionGroup);

                    for(var product of item.Products)
                    {
                        var ul = document.createElement("ul");
                        var ulLi = document.createElement("li");
                        ulLi.innerHTML = product;
                        ul.appendChild(ulLi);
                        li.appendChild(ul);
                        var option =
document.createElement("option");
                        option.innerHTML = product;
                        optionGroup.appendChild(option);
                    }
                }
            })
        }
    </script>
  </head>
  <body>
    <div>
      <button onclick="GetCategories()">Get
Products</button>
    </div>
    <div>
      <h3>Categories List</h3>
      <ol>
```

```
            </ol>
            <h3>Select Category</h3>
            <select>

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

Ex:- 3 Nested Iterations

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Multi Level Iterations</title>
        <script>
            var data = [
                {Category: "Electronics", Products: ["TV", "Mobile"]},
                {Category: "Footwear", Products: ["Nike Casuals", "Lee
Boot"]}
            ];
            function bodyload(){
                for(var item of data)
                {
                    var outerli = document.createElement("li");
                    outerli.innerHTML = item.Category;
                    for(var product of item.Products)
                {
                    var ul = document.createElement("ul");
                    var innerli = document.createElement("li");
                    innerli.innerHTML = product;
                    ul.appendChild(innerli);
                    outerli.appendChild(ul);

document.querySelector("ol").appendChild(outerli);
                }
              }
            }
        </script>
    </head>
    <body onload="bodyload()">
        <ol>

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

Ajax Calls in JavaScript with "Fetch()" Method

- Ajax is "Asynchronous JavaScript And XML"
- It allows partial post back. It can post only a specific portion of page.
- It uses "XmlHttpRequest" object.
- JavaScript provides "fetch()" which uses asynchronous request for fetching data from API.

- The data is returned in binary format, you have to explicitly convert into JSON format by using "json()".
- we use ajax call for requesting data from API.

Note: The process of converting "Binary to Object" and "Object to Binary" is known as "COM-Marshaling"

Syntax:-

```
fetch("api_url")
 .then(function(reponse){
        return response.json();
})
 .then(function(data){
})
              or
fetch("url").then(function(){ get data
}).then(function(){convert to json});
```

Ex:-

1. Add a new folder by name "data"
2. Add a new file into data folder by name "products.json"

```
[
    {
      "Name": "Samsung TV",
      "Price": 45000.55
    },
    {
      "Name": "Nike Casuals",
      "Price": 5000.55
    },
    {
      "Name": "Mobile",
      "Price": 13000.44
    }
]
```

3. Add a new HTML file

   home.html

```
      <!DOCTYPE html>
      <html>
    <head>
      <title>Fetch Data</title>
      <script>
        function GetProducts(){
          fetch("../data/products.json")
          .then(function(response){
            return response.json();
          })
          .then(function(data){
            for(var item of data)
            {
                var tr = document.createElement("tr");
                var tdName = document.createElement("td");
                var tdPrice = document.createElement("td");

                tdName.innerHTML  = item.Name;
                tdPrice.innerHTML = item.Price;

                tr.appendChild(tdName);
                tr.appendChild(tdPrice);
         document.querySelector("tbody").appendChild(tr);
            }
         })
       }
    </script>
</head>
<body>
    <div>
        <button onclick="GetProducts()">Get Products</button>
    </div>
    <div>
        <table width="400" border="1">
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Price</th>
            </tr>
          </thead>
          <tbody>

          </tbody>
        </table>
      </div>
    </body>
  </html>
```

Distributed Computing

- Two applications running on two different mechnies can share information between them.
- Two applications running in two different process of same mechine can share information.
- There are various distributed computing technologies
    - a) CORBA
    - b) DCOM
    - c) RMI
    - d) EJB
    - e) Remoting
    - f) Web Services [API]

- There are 3 specifications
    - a) SOAP
    - b) REST
    - c) JSON

SOAP:

consumer => XML Request <=> XML Response <= provider

REST:

consumer => Query Request <=> XML Response <= provider

?name=tv

JSON:

consumer => JSON Request <=> JSON Response <= provider

"api.nasa.gov"

Ex:- Consuming Nasa API

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>Nasa API - Mars Photos</title>
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
      <script>
         function bodyload(){
            fetch("https://api.nasa.gov/mars-
photos/api/v1/rovers/curiosity/photos?sol=1000&api_key=DEM
O_KEY&quot;")
            .then(function(response){
               return response.json();
            })
            .then(function(data){
               for(var item of data.photos)
               {
                  var tr = document.createElement("tr");
                  var tdId = document.createElement("td");
                  var tdCameraName=
document.createElement("td");
                  var tdPhoto = document.createElement("td");
                  var tdRoverName = document.createElement("td");

                  tdId.innerHTML = item.id;
                  tdCameraName.innerHTML =
item.camera.full_name;

                  var img = document.createElement("img");
                  img.src= item.img_src;
                  img.width="200";
                  img.height="200";
                  tdPhoto.appendChild(img);

                  tdRoverName.innerHTML = item.rover.name;

                  tr.appendChild(tdId);
                  tr.appendChild(tdCameraName);
                  tr.appendChild(tdPhoto);
                  tr.appendChild(tdRoverName);
                  document.querySelector("tbody").appendChild(tr);
```

```
                    }
                })
            }
        </script>
    </head>
    <body class="container-fluid" onload="bodyload()">
        <h2>Nasa Mars Rover Photos</h2>
        <table class="table table-hover">
            <thead>
                <tr>
                    <th>Photo ID</th>
                    <th>Camera Name</th>
                    <th>Photo</th>
                    <th>Rover Name</th>
                </tr>
            </thead>
            <tbody>

            </tbody>
        </table>
    </body>
</html>
```

Fakestore API Methods

GET   http://fakestoreapi.com/products   - all products

GET http://fakestoreapi.com/products/categories - all ategories
GET http://fakestoreapi.com/products/category/electronics
GET http://fakestoreapi.com/products/3

Ex:- Shopping Cart

```
   <!DOCTYPE html>
   <html>
  <head>
    <title>Shopping</title>
    <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
    <link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
    <script>
        function GetCategories(){
          fetch("http://fakestoreapi.com/products/categories;")
          .then(function(response){
            return response.json();
          })
          .then(function(data){
            data.unshift("all");
            for(var item of data)
            {
              var option = document.createElement("option");
              option.text = item.toUpperCase();
              option.value = item;
```

```
 document.getElementById("lstCategories").appendChild(option
);
                }
              })
           }

           function GetProducts(url){

 document.getElementById("productsContainer").innerHTML=""
;
              fetch(url)
              .then(function(response){
                return response.json();
              })
              .then(function(data){
                 for(var item of data)
                 {
                    var card = document.createElement("div");
                    card.className = "card m-2 p-2";
                    card.style.width = "200px";
                    card.innerHTML = `
                      <img src=${item.image} height="150"
class="card-img-top">
                      <div class="card-header" style="height:140px">
                        <p>
                          ${item.title}
                        </p>
                      </div>
                      <div class="card-body">
                         <dl>
                           <dt>Price</dt>
                           <dd>${item.price}</dd>
                           <dt>Rating</dt>
                           <dd>${item.rating.rate}
[${item.rating.count}]</dd>
                         </dl>
                      </div>
                      <div class="card-footer">
                       <button onclick="AddToCartClick(${item.id})"
class="btn btn-danger w-100">
                         <span class="bi bi-cart4"></span> Add to Cart
                       </button>
                      </div>
                    `;
 document.getElementById("productsContainer").appendChild(c
ard);
              }
              })
           }
           function bodyload(){
              GetCategories();
              GetProducts("http://fakestoreapi.com/products;");
              GetCartCount();
```

```
        }
        function CategoryChanged(){

            var categoryName =
document.getElementById("lstCategories").value;

            if(categoryName=="all")
            {
                GetProducts("http://fakestoreapi.com/products;");
            } else {

GetProducts(`http://fakestoreapi.com/products/category/${cate
goryName}`);
            }
        }
        var count = 0;
        var cartItems = [];
        function GetCartCount(){
           document.getElementById("count").innerHTML =
cartItems.length;
        }
        function AddToCartClick(id){
            fetch(`http://fakestoreapi.com/products/${id}`)
            .then(function(response){
               return response.json();
            })
            .then(function(data){
```

```
            cartItems.push(data);
            alert(data.title + " Added to Cart");
            GetCartCount();
        })
    }
    function LoadCartItems(){
        document.getElementById("cartBody").innerHTML ="";
        for(var item of cartItems)
        {
            var tr = document.createElement("tr");
            var tdTitle = document.createElement("td");
            var tdPrice = document.createElement("td");
            var tdImage = document.createElement("td");

            tdTitle.innerHTML = item.title;
            tdPrice.innerHTML = item.price;

            var img = document.createElement("img");
            img.src=item.image;
            img.width="50";
            img.height="50";

            tdImage.appendChild(img);

            tr.appendChild(tdTitle);
            tr.appendChild(tdPrice);
            tr.appendChild(tdImage);
```

```
  document.getElementById("cartBody").appendChild(tr);
              }
            }
        </script>
    </head>
    <body class="container-fluid" onload="bodyload()">
        <header class="bg-danger text-white text-center p-2 mt-2">
            <h2><span class="bi bi-cart3"></span> Shopping Online
</h2>
        </header>
        <section>
            <div class="row">
                <div class="col-2">
                   <div>
                       <label class="form-label">Select Category</label>
                       <div>
                          <select onchange="CategoryChanged()"
class="form-select" id="lstCategories">

                          </select>
                       </div>
                   </div>
                </div>
                <div class="col-8">
                    <div class="d-flex flex-wrap" id="productsContainer"
style="height: 500px; overflow: auto;">
                    </div>
                </div>
                <div class="col-2">
                   <div class="mt-2">
                    <button data-bs-target="#cart"
onclick="LoadCartItems()" data-bs-toggle="modal" class="btn
btn-outline-danger">
                    <span class="bi bi-cart4"></span>
                    [<span id="count"></span>] Your Cart Items
                    </button>
                    <div class="modal fade" id="cart">
                    <div class="modal-dialog">
                        <div class="modal-content">
                            <div class="modal-header">
                                <h3>Your Cart Items</h3>
                                <button data-bs-dismiss="modal"
class="btn-close"></button>
                            </div>
                            <div class="modal-body">
                                <table class="table table-hover">
                                    <thead>
                                        <tr>
                                            <th>Title</th>
                                            <th>Price</th>
                                            <th>Preview</th>
                                        </tr>
                                    </thead>
                                    <tbody id="cartBody">
                                </tbody>
                             </table>
                          </div>
                          <div class="modal-footer">
                             <button data-bs-dismiss="modal"
class="btn btn-primary">OK</button>
                          </div>
                       </div>
                    </div>
                 </div>
                    </div>
                  </div>
               </div>
            </section>
            <script src="../node_modules/jquery/dist/jquery.js">

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

Ex:- Nasa API displayed in Card Style

```
       <!DOCTYPE html>
       <html>
```

```
   <head>
      <title>Nasa API - Mars Photos</title>
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
      <script>
         function bodyload(){
            fetch("https://api.nasa.gov/mars-
photos/api/v1/rovers/curiosity/photos?sol=1000&api_key=DEM
O_KEY&quot;")
            .then(function(response){
              return response.json();
            })
            .then(function(data){
               for(var item of data.photos)
               {
                  var div = document.createElement("div");
                  div.className = "card m-2 p-2";
                  div.style.width="200px";
                  div.innerHTML = `
                    <img src=${item.img_src} class="card-img-top"
height="200">
                    <div class="card-header">
                      <h2>${item.id}</h2>
                    </div>
                    <div class="card-body">
                      <dl>
                        <dt>Camera Name</dt>
                    <dd>${item.camera.full_name}</dd>
                    <dt>Rover Name</dt>
                    <dd>${item.rover.name}</dd>
                  </dl>
                </div>
               `;

document.getElementById("dataContainer").appendChild(div);
             }
          })
        }
     </script>
   </head>
   <body class="container-fluid" onload="bodyload()">
     <h2>Nasa Mars Rover Photos</h2>
     <div class="d-flex flex-wrap" id="dataContainer">
     </div>
   </body>
</html>
```

Ex:- Shopping Cart with "fakestoreapi.com"

Request	Purpose
fakestoreapi.com/products	returns all products [20]
/products/1	returns specific id related product.
/products/categories	returns all categories list

/products/category/jewelery   returns all products related to specific category

Ex:- shopping.html

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Shopping API</title>
        <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
        <link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
        <script>
            function GetCategories(){

fetch("http://fakestoreapi.com/products/categories&quot;")
                .then(function(response){
                    return response.json();
                })
                .then(function(data){
                    data.unshift("All");
                    for(var item of data)
                    {
                        var option = document.createElement("option");
                        option.text = item.toUpperCase();
                        option.value = item;
```

```
document.getElementById("lstCategories").appendChild(option);
          }
        })
      }
      function GetProducts(url){

document.getElementById("productsContainer").innerHTML="";
        fetch(url)
        .then(function(response){
          return response.json();
        })
        .then(function(data){
          for(var item of data) {
            var div = document.createElement("div");
            div.className="card m-2 p-2";
            div.style.width = "200px";
            div.innerHTML = `
             <img src=${item.image} class="card-img-top"
height="200">
             <div class="card-header" style="height:140px">
               <p>${item.title}</p>
             </div>
             <div class="card-body">
               <p>${item.price}</p>
               <p>Rating : ${item.rating.rate}</p>
               <p>Count  : ${item.rating.count}
             </div>
```

```
                    <div class="card-footer">
                     <button onclick="AddToCartClick(${item.id})"
class="btn btn-danger w-100">
                      <span class="bi bi-cart4"> </span>
                       Add to Cart
                     </button>
                    </div>
                   `;
document.getElementById("productsContainer").appendChild(di
v);
                }
              })
            }
            function bodyload(){
              GetCategories();
  GetProducts("http://fakestoreapi.com/products&quot;");
              GetCartItemsCount();
            }
            function  CategoryChanged(){
               var categoryName =
document.getElementById("lstCategories").value;
               if(categoryName=="All") {

GetProducts("http://fakestoreapi.com/products&quot;");
               } else {

GetProducts(`http://fakestoreapi.com/products/categories/${ca
tegoryName}`);
            }
            }
            var cartItems = [];
            var count = 0;
            function GetCartItemsCount(){
            count = cartItems.length;
            document.getElementById("cartCount").innerHTML =
count;
            }
            function AddToCartClick(id){
            fetch(`http://fakestoreapi.com/products/${id}`)
         .then(function(response){
                return response.json();
            })
            .then(function(data){
                cartItems.push(data);
                alert("Item Added to Cart");
                GetCartItemsCount();
            })
            }
            function ShowCartClick(){
            document.querySelector("tbody").innerHTML="";
            for(var item of cartItems){
                var tr = document.createElement("tr");
                var tdTitle = document.createElement("td");
                var tdPrice = document.createElement("td");
                var tdPhoto = document.createElement("td");

                tdTitle.innerHTML = item.title;
                tdPrice.innerHTML = item.price;

                var img = document.createElement("img");
                img.src= item.image;
                img.width="100";
                img.height="100";

                tdPhoto.appendChild(img);

                tr.appendChild(tdTitle);
                tr.appendChild(tdPrice);
                tr.appendChild(tdPhoto);

                document.querySelector("tbody").appendChild(tr);
            }
        }
    </script>
  </head>
  <body class="container-fluid" onload="bodyload()">
    <header class="bg-danger text-white text-center p-2">
        <h1> <span class="bi bi-cart4"></span> Shopping
Online</h1>
    </header>
    <div class="row mt-3">
```

```
          <div class="col-2">
           <h4>Select a Category</h4>
           <select class="form-select" id="lstCategories"
onchange="CategoryChanged()">
           </select>
          </div>
          <div class="col-8">
           <div id="productsContainer" class="d-flex flex-wrap
overflow-auto" style="height: 500px;">
           </div>
          </div>
          <div class="col-2">
            <button onclick="ShowCartClick()" class="btn btn-
warning" data-bs-target="#cart" data-bs-toggle="modal">
              [<span id="cartCount"></span>]
              <span class="bi bi-cart3"></span>
              Your Cart Items
            </button>
            <div class="modal fade" id="cart">
              <div class="modal-dialog modal-dialog-centered">
                <div class="modal-content">
                  <div class="modal-header">
                    <h3>Your Cart Items</h3>
                    <button class="btn-close" data-bs-
dismiss="modal"></button>
                  </div>
                  <div class="modal-body">
```

```
                    <table class="table table-hover">
                        <thead>
                            <tr>
                                <th>Title</th>
                                <th>Price</th>
                                <th>Preview</th>
                            </tr>
                        </thead>
                        <tbody>
                        </tbody>
                    </table>
                </div>
                <div class="modal-footer">
                     <button data-bs-dismiss="modal" class="btn
btn-success">OK</button>
                </div>
            </div>
        </div>
    </div>
  </div>
</div>
<script src="../node_modules/jquery/dist/jquery.js">

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

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

Non-Primitive Types

- Array
- Object
- Map

What is Map type?

- Map is similar to object with key and value collection.[ES5]
- It is an object with key and value collection

FAQ: What is difference between Map and Object?

Ans:

Object	Map
1. Key and Value collection	Key and Value collection
2. Can contain Keys is only of string type "Id" : 1 1 : "TV" // invalid	Key can be any type "Id": 1 // valid 1 : "TV" // valid
3. You need explicit iterators to reading all keys and values. for..in	Provides implicit iterators to read key and values. a)keys(), b) values(), c) entries()
4. Slow in accessing	Faster than object

5. Size of keys is unknown	Allows to access size of keys
6. No size forkeys, you can't get length of keys.	It provides built-in properties to access a) all keys, b) all values, c) size of keys
7. - - - - - - - - -	Map uses "async" technique
8. - - - - - - - - -	Faster in access
9. - - - - - - - - -	available from ES5

Map Methods AND Properties:

1. set()	: Adds a new value or can store value into specified and values
2. get()	: Fetch the value by using key reference
3. keys()	: returns the collection all keys
4. values()	: returns the collection of all values
5. entries()	: returns the colection of both key and values.
6. delete()	: deletes specific key
7. clear()	: deletes all entries
8. has()	: It returns boolean true when specified key exists.
9. size()	: returns the total count of keys

Syntax:-

```
if(data.has("Name"))
```

```
{

}

else

{

}
```

OR

```
<script>
    var collection = new Map();
    collection.set("TV", "Samsung TV");
    collection.set(1, "Lenonvo Laptop");
    document.write(collection.get("TV"));
</script>
```

Ex:-

```
<script>
    var collection = new Map();
    collection.set("TV", "Samsung TV");
    collection.set(1, "Lenonvo Laptop");
    for(var key of collection.entries()){
        document.write(key + "<br>");
    }
document.write("Total Count of Keys: " + collection.size);
</script>
```

Date Type

- JavaScript date values are defined by using "Date()" constructor.
- It loads the current date and time into memory.

```
var  now  = new Date();
```

- You can configure any specific date and time by using date value in constructor.

```
var  now = new Date("2021-01-10");
```

- You can access the date and time values by using following methods

getHours()	returns hour number in 24 hr format
getMinutes()	returns minutes number 0-59
getSeconds()	returns seconds number 0-59
getMilliSeconds()	returns milli seconds number 0-99
getDate()	returns date number [22]
getDay()	returns weekday number [0=sunday]
getMonth()	returns month number [0=January]
getFullYear()	returns year number [2021]
toLocaleDateString()	returns complete date
toLocaleTimeString()	returns complete time

Syntax:-

```
var mfd = new Date();        // will load current date and time

var mfd = new Date("2020-02-10");       // load specific date
```

Ex:-

```
<script>
    let Mfd = new Date("2021-08-15");
    let months = ["January",
"Feb","Mar","Apr","May","June","July","August","Sep"];
    let weekdays= ["Sunday","Monday","Tue","Wed",
"Thu", "Fir","Sat"];
    document.write(`
        Manufactured Date : ${Mfd.getDate()} <br>
        Manufactured Month: ${months[Mfd.getMonth()]}
<br>
      Manufactured Weekday: ${weekdays[Mfd.getDay()]} <br>
      Manufactured Year: ${Mfd.getFullYear()} <br>
      Manufactured Date : ${weekdays[Mfd.getDay()]},
${Mfd.getDate()} - ${months[Mfd.getMonth()] }
${Mfd.getFullYear()}
    `);
</script>
```

- JavaScript provides the following methods for setting a new date or time.

```
setHours()
setMinutes()
```

```
      setSeconds()
      setMilliSeconds()
      setDate()
      setDay()
      setMonth()
      setYear()
```

Ex:-

```
      <script>
    let Mfd = new Date("2021-08-15");
    let months = ["January",
"Feb","Mar","Apr","May","June","July","August","Sep"];
    let weekdays= ["Sunday","Monday","Tue","Wed", "Thu",
"Friday","Sat"];
    Mfd.setMonth(7);
    Mfd.setDate(20);
    document.write(`
        Manufactured Date : ${Mfd.getDate()} <br>
        Manufactured Month: ${months[Mfd.getMonth()]} <br>
        Manufactured Weekday: ${weekdays[Mfd.getDay()]} <br>
        Manufactured Year: ${Mfd.getFullYear()} <br>
        Manufactured Date : ${weekdays[Mfd.getDay()]},
${Mfd.getDate()} - ${months[Mfd.getMonth()] }
${Mfd.getFullYear()}
    `);
</script>
```

JavaScript Timer Events:

- setInterval()
- clearInterval()
- setTimeout()
- clearTimeout()

setInterval() : It is used to perform specified task repeatedly at given time interval.

Syntax:-

```
setInterval(functionName, timeInterval);

setInterval(GetTime, 1000);
    100 milli seconds = 1 sec        [ Clock ]

    1000 milli seconds = 1 sec       [ CPU ]
```

Ex:-

```
<!DOCTYPE html>
<html>
<head>
    <title>Time</title>
    <link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
    <style>
        #container {
            display: flex;
            justify-content: space-between;
            font-size: 25px;
            background-color: red;
            color:white;
            padding: 5px;
        }
    </style>
    <script>
        function Clock(){
            var now = new Date();
            document.getElementById("time").innerHTML =
now.toLocaleTimeString();
        }
        function bodyload(){
            setInterval(Clock, 1000);
            var now = new Date();
            var icon = document.getElementById("icon");
            var msg = document.getElementById("msg");
            var hrs = now.getHours();
            if(hrs>=0 && hrs<=12) {
                msg.innerHTML = "Good Morning";
                icon.className= "bi bi-brightness-alt-high";
            } else if (hrs>=13 && hrs<=17) {
                msg.innerHTML = "Good Afternoon";
                icon.className = "bi bi-brightness-high";
            } else {
                msg.innerHTML = "Good Evening";
            icon.className = "bi bi-cloud-sun-fill";
          }
        }
      </script>
    </head>
    <body onload="bodyload()">
      <div id="container">
        <span>Amazon Shopping</span>
        <span>
          <span id="icon"></span>  <span id="msg"></span>
        </span>
        <span id="time"></span>
      </div>
    </body>
  </html>
```

Note: You can access date from HTML date picker and convert into Date format by using "Date()".

Syntax:-

```
    var departure = new
  Date(document.getElementById("txtDate").value);
```

Ex:- Dynamic Date

```
    <!DOCTYPE html>
    <html>
  <head>
```

```
        <title>Date Demo</title>
        <script>
            function DateChanged(){
                var weekdays =
["Sunday","Mon","Tue","Wed","Thu","Fri","Sat"];

                var departure = new
Date(document.getElementById("txtDate").value);
                document.querySelector("h2").innerHTML = `Departure
Date : ${weekdays[departure.getDay()]},
${departure.toLocaleDateString()}`;
            }
        </script>
    </head>
    <body>
        Departure Date :
        <input type="date" onchange="DateChanged()"
id="txtDate">
        <br>
        <h2></h2>
    </body>
</html>
```

Summary:

1. Primitive Types

    - number

- string
- boolean
- null
- underfined

2. Non Primitive Types
    - array
    - object
    - map
    - date

Regular Expression Type

- Regular Expression is a group of meta characters and quantifiers enclosed in "/ /".

Syntax:-

```
let  regExp  = /\+91[0-9]{10}/;
```

- Regular Epxression is verified by using "match()" method.

Syntax:

```
let  mobile = "+919876543210";

if(mobile.match(regExp))
{
          }
```

Math Object

- It provides set of properties and methods to handle mathematical operations.

```
            Math.PI
            Math.sqrt()
            Math.sin()
            Math.tan()
            Math.pow()
            Math.round()
            Math.random()  etc..
```

Ex:-

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Login</title>
        <link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
        <style>
            dd,dt {
                margin-top: 10px;
            }
        </style>
        <script>
            function GenerateCode(){
                var a = Math.random() * 10;
                var b = Math.random() * 10;
                var c = Math.random() * 10;
                var d = Math.random() * 10;
                var e = Math.random() * 10;
                var f = Math.random() * 10;
                return `${Math.round(a)} ${Math.round(b)}
${Math.round(c)} ${Math.round(d)} ${Math.round(e)}
${Math.round(f)}`;
            }
            function bodyload(){
                document.getElementById("code").innerHTML =
GenerateCode();
            }
            function NewCode(){
                bodyload();
            }
        </script>
    </head>
    <body onload="bodyload()">
        <fieldset>
            <legend>User Login</legend>
            <dl>
                <dt>User Name</dt>
                <dd><input type="text"></dd>
                <dt>Password</dt>
                <dd><input type="password"></dd>
        <dt>Verify Code <button onclick="NewCode()"
class="btn"><span class="bi bi-arrow-
clockwise"></span></button> </dt>
        <dd><span id="code"></span></dd>
      </dl>
      <button>Login</button>
    </fieldset>
  </body>
</html>
```

JavaScript Language Basics

- Variables
- Data Types
- Operators

JavaScript Operators

- Operator is an object in computer programming.
- Operator is an object that evaluates and returns a value.
- Operator comprises of operands that store data.

```
x  +  y       x and y are operands
      +     => is operator
```

- Based on how many operands an operator can handle, the operators are classified into following 3 types

    a) Unary Operator

        x++

```
--y
```

b) Binary Operator

```
x + y
x * y
```

c) Ternary Operator

```
? :
(condition)?if_true:if_false
```

- Operators are again classified into various groups based on the type of value they return.

a) Arithematic Operators	: number
b) Conditional Operators	: boolean
c) Logical Operators	: boolean
d) Bitwise Operators	: Binary value
e) Special Operators	: vary in functionality

Arithematic Operators:

+	Addition
-	Substraction
*	Multiplication
/	Division
%	Modulus
**	Exponent
++	Increment
--	Decrement

Addition Operator: It returns the sum of given numbers.

number	+	number	= number
number	+	string	= string
number	+	boolean	= number
string	+	string	= string
string	+	number	= string
string	+	boolean	= string
boolean	+	string	= string
booelan	+	number	= number
boolean	+	boolean	= number

Note: Any operation with undefined will be "NaN" except string.
Any operation with null will return number except string.

Substraction Operator: It returns the difference value.

string	-	number	= NaN
string	-	string	= number [If both string are having a number value]
number	-	boolean	= number
boolean	-	boolean	= number
all other operations			= NaN

Note: "-" operator uses implicit parsing, but only for numeric representation in string format.

Multiplication Operator: It returns the product of given numbers

string * string = NaN [Not A Number]

string * string = number [both are numeric]

string * number = NaN

string * number = number [both are numeric]

number * number = number

number * boolean = number or infinity [exception]

boolean * boolean = number or infinity

all other operations = NaN

Division : It returns the quotient value of dividend divided by divisior. [/]

- same like multiplication [0 or false] - infinity

number / number = number

number / boolean = number [true]

Modulus : It returns the remainder value of division. [%]

- same like division

number % number = number

number % boolean = number [true]

Ex:-

```
<!DOCTYPE html>
<html>
```

```
    <head>
        <title>Modulus</title>
        <script>
            function SubmitClick(){
                var n = document.getElementById("txtEven").value;
                var msg = document.getElementById("msg");
                if(parseInt(n) %2 == 0){
                    msg.innerHTML = "Verified..";
                } else {
                    msg.innerHTML = "Not an Even Number";
                }
            }
        </script>
    </head>
    <body>
        Enter Even Number :
        <input type="text" id="txtEven">
        <button onclick="SubmitClick()">Submit</button>
        <br>
        <div id="msg"></div>
    </body>
</html>
```

Exponent Operator [**] : new in ES5+

It returns the value of base raised to power.

```
2**3            = 8            [ new ]
```

Math.pow(2,3)   = 8   [old technicque]

Increment and Decrement Operators: They increase or decrease the current value with 1 and store the returned value.

```
x++         x = x  + 1;
x--         x = x  - 1;
```

Post Increment : It assigns and then increments.

```
x++
x = 10;
y = x++;              y = 10, x = 11
```

Post Decrement : It assign and the decrements.

```
x--
x = 10;
y = x--;              x = 9,  y = 10
```

Pre Increment   : It increments and then assign

Pre Decrement : It decrements and then assign

```
++x
x = 10;
y = ++x;              x=11, y=11
y = --x;              x=9, y=9
```

Ex:-

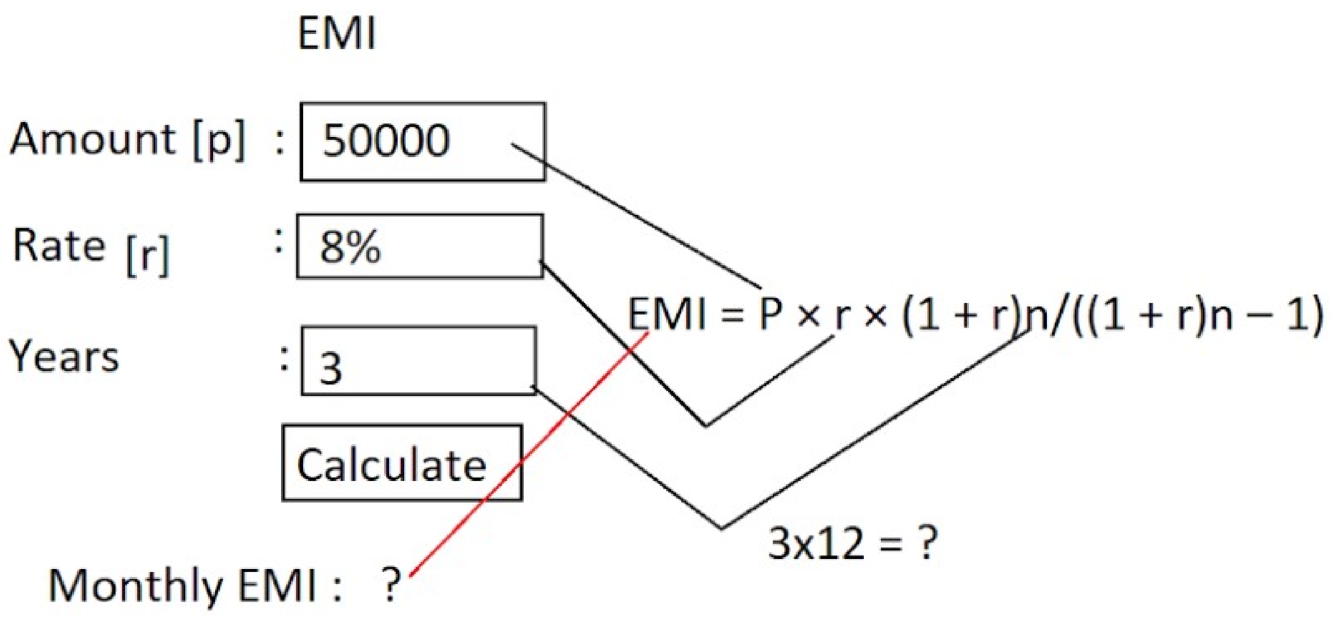
```
<!DOCTYPE html>
<html>
```

```
   <head>
      <title>Operators</title>
      <link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
      <script>
         function GetProduct(url) {
            fetch(url)
            .then(function(response){
               return response.json();
            })
            .then(function(data){
               document.getElementById("prodTitle").innerHTML =
data.title;
               document.getElementById("prodImg").src =
data.image;
               document.getElementById("prodDesc").innerHTML =
data.description;
            })
         }
         function bodyload(){
 GetProduct("http://fakestoreapi.com/products/1&quot;");
         }
         var count = 1;
         function NextClick(){
            count++;
GetProduct(`http://fakestoreapi.com/products/${count}`);
         }
         function PreviousClick(){
            count--;
GetProduct(`http://fakestoreapi.com/products/${count}`);
         }
      </script>
      <style>
         #prodDesc {
            position: fixed;
            right: 20px;
            top:550px;
            border:2px solid darkcyan;
            padding: 10px;
            width: 70px;
            height: 100px;
            background-color: darkcyan;
            color:white;
            font-size: 20px;
            overflow: auto;
         }
      </style>
   </head>
   <body onload="bodyload()">
      <div class="container-fluid" style="margin-top: 100px;">
         <div class="d-flex justify-content-center align-items-
center" style="height: 400px;">
```

```
          <div>
            <div class="card">
              <div class="card-header text-center">
                <h2 id="prodTitle"></h2>
              </div>
              <div class="card-body text-center">
                <img width="300" id="prodImg" height="300">
                <p id="prodDesc" class="w-50 text-center"></p>
              </div>
              <div class="card-footer text-center">
                <button class="btn btn-danger"
onclick="PreviousClick()">
                  <span class="bi bi-chevron-bar-left"></span>
                </button>
                <button class="btn btn-danger"
onclick="NextClick()">
                  <span class="bi bi-chevron-bar-right"></span>
                </button>
              </div>
            </div>
          </div>
        </div>
      </div>
    </body>
</html>
```

Ex:- EMI Calculator

EMI = P × r × (1 + r)n/((1 + r)n – 1)



Ex:- 1

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>EMI Calculator</title>
    <link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
    <link rel="stylesheet" href="../node_modules/bootstrap
/dist/css/bootstrap.css">
    <style>
      .form-element {
        width: 100px;
      }
    </style>
    <script>
```

```
          function AmountChange(){
             document.getElementById("txtAmount").value =
 document.getElementById("amountRange").value;
          }
          function CalculateClick(){
            var p =
parseInt(document.getElementById("txtAmount").value);
             var r =
parseInt(document.getElementById("txtInterest").value);
             var n =
parseInt(document.getElementById("txtYears").value);
             var emi = "";

          }
          function UpdateAmount(){
             document.getElementById("amountRange").value =
document.getElementById("txtAmount").value;
          }
       </script>
    </head>
    <body class="container-fluid">
       <div>
          <h2>EMI Calculator</h2>
          <div class="row">
             <div class="col">
                <div class="input-group">
                   Amount you need <span class="input-group-
```

```
text">&#8377;</span> <input type="text"
onchange="UpdateAmount()" id="txtAmount" class="form-
element">
          </div>
        </div>
        <div class="col">
          <div class="input-group">
            for <input type="text" id="txtYears" class="form-
element"> years
          </div>
        </div>
        <div class="col">
          <div class="input-group">
            Interest rate <input type="text" id="txtInterest"
class="form-element"> %
          </div>
        </div>
      </div>
      <div class="row mt-4">
        <div class="col">
          50,000 <input type="range"
onchange="AmountChange()" id="amountRange" min="50000"
max="4000000" value="50000"> 4000000
        </div>
        <div class="col">
          1 <input type="range" min="1" id="yearRange"
value="1" max="5"> 5
```

```
            </div>
            <div class="col">
              10.5% <input type="range" min="10"
id="interestRange" max="21" value="10" step="0.5"> 21%
            </div>
          </div>
          <div class="row mt-4">
            <div class="col">
                <button  onclick="CalculateClick()" class="btn btn-
primary">Calculate</button>
            </div>
          </div>
        </div>
    </body>
</html>
```

Ex:- 2 EMI Calculator

```
        <!DOCTYPE html>

        <html>

<head>

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

    <link rel="stylesheet" href="/node_modules/bootstrap-
icons/font/bootstrap-icons.css">
```

```
    <title>EMI Calculator</title>
</head>
<script>
    function AmountChange() {
        document.getElementById("Amount").value = 
document.getElementById("LoanAmount").value;
    }
    function IRchange() {
        document.getElementById("IR").value = 
document.getElementById("LoanIR").value;
    }
    function YearsChange() {
        document.getElementById("Years").value = 
document.getElementById("LoanYears").value;
    }
    function Calculate() {
        var time = document.getElementById("Years").value * 12;
        var rate = document.getElementById("IR").value / 1200;
        var money = document.getElementById("Amount").value;
        var result;
        result = money * rate * ((1 + rate) ** time) / (((1 + rate) **
time) - 1);
        error.innerHTML = `your monthly EMI payment is<b>
&#8377 ${result} </b>`;
    }
    function bodyLoad() {
        AmountChange();
        IRchange();
        YearsChange();
        Calculate();
    }
</script>
<style>
    button {
        position: relative;
        top: 50px;
        left: 1240px;
    }
    .content {
        display: grid;
        grid-template-columns: 4fr 4fr 4fr;
    }
</style>
<body class="container-fluid" onload="bodyLoad()">
    <h2>EMI Calculator</h2>
    <div class="content">
        <div class="input-group m-2 p-2">
            <span class="input-group-text">Amount you need</span>
            <input type="text" class="form-control" id="Amount">
            <input onchange="AmountChange()" type="range"
min="00" max="4000000" value="0" id="LoanAmount"
                class="form-range">
            <span class="col">0</span>
            <span class="col text-end">4000000</span>
        </div>
        <div class="input-group m-2 p-2">
            <span class="input-group-text">for</span>
            <input type="text" class="form-control" id="Years">
            <span class="input-group-text">years</span>
            <input class="form-range" onchange="YearsChange()"
type="range" id="LoanYears" min="1" max="5" value="1">
            <span class="col">1</span>
            <span class="col text-end">5</span>
        </div>
        <div class="input-group m-2 p-2">
            <span class="input-group-text">Interest rate</span>
            <input type="text" class="form-control" id="IR">
            <span class="input-group-text">%</span>
            <input class="form-range" type="range" min="0"
max="21" value="0" id="LoanIR" onchange="IRchange()">
            <span class="col">0</span>
            <span class="col text-end">21</span>
        </div>
    </div>
    <button onclick="Calculate()" class="btn btn-
primary">Calculate</button>
    <div id="error" class="bg-dark text-light w-50 p-2">
    </div>
</body>
</html>
```

Ex:- 3 EMI Calculator [HDFS]

```
    <!DOCTYPE html>
    <html lang="en-IN">
<head>
  <title>EMI Calculator</title>
  <link rel="stylesheet" href="/node_modules/bootstrap-
icons/font/bootstrap-icons.css">
  <link rel="stylesheet"
href="/node_modules/bootstrap/dist/css/bootstrap.css">
  <style>
    input {
      width: 100px;
    }
    #border {
      border: 5px solid black;
      padding: 20px;
      border-radius: 10px;
    }
  </style>
  <script>
```

```
        function AmountChange() {
            document.getElementById("txtAmount").value =
document.getElementById("rangeAmount").value;
        }
        function YearChange() {
            document.getElementById("txtYear").value =
document.getElementById("rangeYear").value;
        }
        function RateChange() {
            document.getElementById("txtRate").value =
document.getElementById("rangeRate").value;
        }
        function Calculate() {
            var amount =
document.getElementById("rangeAmount").value;
            var year = document.getElementById("rangeYear").value;
            var rate = document.getElementById("rangeRate").value;
            var EMI = amount * (rate / 100) * (1 + (rate / 100)) * year
* 12 / ((1 + (rate / 100)) * (year * 12) - 1);
```

```
          document.getElementById("msg").innerHTML = `Your
Monthly EMI will be &#8377;${EMI.toFixed(2)} per month`;

      }

   </script>

</head>

<body>

   <div class="m-5" id="border">

      <h2 class="bg-info text-white text-center p-2">EMI
Calculator</h2>

      <div class="row my-5">

          <div class="col">

              Amount <input type="text" id="txtAmount"
placeholder="&#8377;">

              <input type="range" id="rangeAmount"
onchange="AmountChange()" class="form-range w-75"
min="20000"

                  max="200000" value="10000">

              <div class="row w-75">

                  <div class="col">&#8377;10,000</div>

                  <div class="col text-end">&#8377;2,00,000</div>

              </div>
```

```
            </div>

            <div class="col">

                Select <input type="text" id="txtYear"
placeholder="Years">

                <input type="range" id="rangeYear"
onchange="YearChange()" class="form-range w-75" min="1"
max="10"

                    value="00">

                <div class="row w-75">

                    <div class="col">1 Year</div>

                    <div class="col text-end">10 Year</div>

                </div>

            </div>

            <div class="col">

                Interest <input type="text" id="txtRate"
placeholder="%">

                <input type="range" id="rangeRate"
onchange="RateChange()" class="form-range w-75" min="1"
max="20"

                    value="00" step="0.05">

                <div class="row w-75">
                    <div class="col">1%</div>

                    <div class="col text-end">20%</div>

                </div>

            </div>

        </div>

        <div class="mt-5">

            <button class="btn btn-success"
onclick="Calculate()">Calculate</button>

        </div>

        <p id="msg" class="mt-5 text-center text-white bg-danger p-
3 w-50"></p>

  </div>

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

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

</body>

</html>
```

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
```

```
        <title>Operators</title>
        <link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
        <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
        <script>
            function GetProduct(url) {
                fetch(url)
                .then(function(response){
                    return response.json();
                })
                .then(function(data){
                    document.getElementById("prodTitle").innerHTML =
data.title;
                    document.getElementById("prodImg").src =
data.image;
                    document.getElementById("prodDesc").innerHTML =
data.description;
                })
            }
            function bodyload(){
 GetProduct("http://fakestoreapi.com/products/1&quot;");
            }
            var count = 1;
            function NextClick(){
                count++;
 GetProduct(`http://fakestoreapi.com/products/${count}`);
```

```
        }
        function PreviousClick(){
          count--;
GetProduct(`http://fakestoreapi.com/products/${count}`);
        }
        function ProductSliderChanged(){
          var id =
parseInt(document.getElementById("productSlider").value);
          GetProduct(`http://fakestoreapi.com/products/${id}`);
        }
      </script>
      <style>
        #prodDesc {
          position: fixed;
          right: 20px;
          top:550px;
          border:2px solid darkcyan;
          padding: 10px;
          width: 70px;
          height: 100px;
          background-color: darkcyan;
          color:white;
          font-size: 20px;
          overflow: auto;
        }
      </style>
    </head>
    <body onload="bodyload()">
        <div class="container-fluid" style="margin-top: 100px;">
            <div class="d-flex justify-content-center align-items-
center" style="height: 400px;">
                <div>
                    <div>
                        <input type="range"
onchange="ProductSliderChanged()" id="productSlider" min="1"
max="20" value="1" class="form-range">
                    </div>
                    <div class="card">
                        <div class="card-header text-center">
                            <h2 id="prodTitle"></h2>
                        </div>
                        <div class="card-body text-center">
                            <img width="300" id="prodImg" height="300">
                            <p id="prodDesc" class="w-50 text-center">

                               </p>
                        </div>
                        <div class="card-footer text-center">
                            <button class="btn btn-danger"
onclick="PreviousClick()">
                                <span class="bi bi-chevron-bar-left"></span>
                            </button>
               <button class="btn btn-danger" onclick="NextClick()">
                             <span class="bi bi-chevron-bar-right"></span>
                            </button>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </body>
</html>
```

Conditional Operators

==	Equal
===	Identical Equal
!==	Not Identical
!=	Not Equal
>	Greater than
>=	Greater than or equal
<	Lesser than
<=	Lesser than or equal

Equal Opreator : [==]

- It can compare values of different types.

- It directly compares and uses implicit parsing.

```
10=="10"      // true
```

Identical Equal : [===]

- It can compare only values of same type.

- It will not use any implicit parsing.

```
10==="10"          // false
10===10            // true
"10"==="10"        // true
```

FAQ: What is the difference between =, ==, === operators ?

Ans:

=	is to assign a value
==	is to compare 2 values of different types
===	is to compare 2 values of same type

Syntax:-

```
x = 10;
y = "10";
x == y;              // true
x === y;             // false
x != y;              // false
x !== y;             // true
```

Note: All comparision operators return boolean value.

Logical Operators

&&	Logical AND
||	Logical OR
!	Logical NOT

Syntax:-

condition1 && condition2 : Expression will be true only when both conditions evaluate to true.

condition1 || condition2 : Expression will be true if any one condition is true.

!true : Negation of true is false.

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>Register</title>
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
      <script>
         var users = [
            {UserName: 'john'},
            {UserName: 'john_nit'},
            {UserName: 'david'},
            {UserName: 'david11'}
         ];
         function VerifyUser(){
            var username =
document.getElementById("txtName").value;
            var nameMsg =
document.getElementById("nameMsg");
            for(var user of users)
            {
               if(user.UserName===username) {
                  nameMsg.innerHTML = "User Name Taken - Try
Another";
                  nameMsg.className = "text-danger";
                  return;
               } else {
                  nameMsg.innerHTML = "User Name Available";
                  nameMsg.className = "text-success";
               }
            }
         }
      </script>
   </head>
   <body class="container-fluid">
      <h2>Register User</h2>
      <dl>
         <dt>User Name</dt>
         <dd><input type="text" onkeyup="VerifyUser()"
id="txtName"></dd>
         <dd id="nameMsg"></dd>
      </dl>
   </body>
```

```
</html>
```

FAQ: What is difference between break and return?

Ans: Both are Jump statements.
"break" will terminate the block but will stay in script.
"return" will terminate the script.

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>Register</title>
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
      <script>
         var users = [
            {UserName: 'john', Password: 'john@123'},
            {UserName: 'john_nit', Password: 'john11'},
            {UserName: 'david', Password: '12345'},
            {UserName: 'david11', Password: 'david@11'}
         ];
         function VerifyUser(){
            var username =
document.getElementById("txtName").value;
            var password =
document.getElementById("txtPwd").value;

            for(var user of users)
```

```
          {
            if(user.UserName===username &&
user.Password===password) {
              document.write("<h2>Login Success</h2>");
            } else {
              document.getElementById("msg").innerHTML =
"Invalid UserName / Password";
            }
          }
        }
      </script>
    </head>
    <body class="container-fluid">
      <h2>Register User</h2>
      <dl>
        <dt>User Name</dt>
        <dd><input type="text" id="txtName"></dd>
        <dt>Password</dt>
        <dd><input type="password" id="txtPwd"></dd>
      </dl>
      <button onclick="VerifyUser()">Login</button>
      <h2 id="msg" class="text-center text-danger"></h2>
    </body>
</html>
```

Assignment Operators

+=	Add and Assign
-=	

```
*=
/=
%=
```

Syntax:-

```
var x = 10;            // 10
var y = 20;          // 20
y += x;          // y = 20 + 10  = 30
```

Logical Opreators :

&&	Logical AND
||	Logical OR
!	NOT

```
expression1 && expression2   = true when both expression
                               evaluate to true.
expression1 && expression2   = true when any one
                               expression evaluate to true.
!expression                  = negation     !true = false
```

Ex: Kfc Order

```
<!DOCTYPE html>
<html>
<head>
<title>KFC Online Order</title>
<link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
<link rel="stylesheet" href="../node_modules /bootstrap
 /dist/css/bootstrap.css">
<style>
   body {
```

```
            background-color: maroon;
        }
    </style>
    <script>
        function OrderClick(){
            document.getElementById("lblName").innerHTML =
    document.getElementById("txtName").value;
            document.getElementById("lblMobile").innerHTML =
    document.getElementById("txtMobile").value;

   var burgerRadio = document.getElementById("optBurger");
   var rollerRadio = document.getElementById("optRoller");

var wingsCheckbox = document.getElementById("optWings");
var krusherCheckbox =
 document.getElementById("optKrusher");

            var mealName = "";
            var adonName = "";

            var mealCost = 0;
            var adonCost = 0;

            if(burgerRadio.checked) {
                mealName = burgerRadio.value;
                mealCost = 120;
            }

            if(rollerRadio.checked) {
                mealName = rollerRadio.value;
                mealCost = 100;
            }

            if(krusherCheckbox.checked){
               adonName += krusherCheckbox.value + "<br>";
               adonCost = 60;
               mealCost += adonCost;
            }
            if(wingsCheckbox.checked) {
               adonName += wingsCheckbox.value + "<br>";
               adonCost = 80;
               mealCost += adonCost;
            }
            document.getElementById("lblMeal").innerHTML
=mealName ;
            document.getElementById("lblAdon").innerHTML =
adonName;
            document.getElementById("lblAmount").innerHTML
="&#8377; " + mealCost;

         }
      </script>
   </head>
   <body class="container-fluid">
      <header>
         <img src="../public/images/kfctop.PNG" width="100%"
      height="160">
      </header>
      <section>
         <div class="accordion" id="kfcOrder">
            <div class="accordion-item">
              <div class="accordion-header">
              <button class="btn btn-danger w-100" data-bs-
            toggle="collapse" data-bs-target="#customer">
               <h3>Customer Details</h3>
              </button>
            </div>
            <div id="customer" class="accordion-body
          accordion-collapse collapse show" data-bs-
       parent="#kfcOrder">
              <dl>
                 <dt>Customer Name</dt>
                 <dd><input type="text" id="txtName" class=
"form-control"></dd>
                 <dt>Mobile</dt>
                 <dd><input type="text" id="txtMobile" class=
"form-control"></dd>
              </dl>
            </div>
          </div>
          <div class="accordion-item">
             <div class="accordion-header">
               <button class="btn btn-danger w-100" data-bs-
              toggle="collapse" data-bs-target="#meal">
                 <h3>Select Your Meal</h3>
               </button>
              </div>
              <div class="accordion-body accordion-collapse
      collapse" id="meal"  data-bs-parent="#kfcOrder" >
                <div class="row">
                  <div class="col text-center">
                     <img src="../public/images/omg1.PNG"
                 width="200" height="200">
                     <div>
                   <input id="optBurger" value="OMG Burger"
                     name="meal" type="radio" class="form-
               check-input" > OMG Burger [ &#8377; 120/-]
                     </div>
                   </div>
                   <div class="col text-center">
                     <img src="../public/images/omg2.PNG"
                   width="200" height="200">
                     <div>
                       <input type="radio" id="optRoller" value=
                         "OMG Roller" name="meal" class="form-
                     check-input"> OMG Roller [&#8377;
                  100/-]
                     </div>
                   </div>
                </div>
              </div>
          </div>
          <div class="accordion-item">
             <div class="accordion-header">
                <button class="btn btn-danger w-100" data-bs-
             toggle="collapse" data-bs-target="#adon">
               <h3>Select Ad-ON's</h3>
                </button>
              </div>
             <div class="accordion-body accordion-collapse
        collapse" id="adon"  data-bs-parent="#kfcOrder"  >
                <div class="row">
                   <div class="col text-center">
                     <img src="../public/images/krusher1.PNG"
                width="200" height="200">
                     <div>
              <input type="checkbox" class="form-check-input"
           id="optKrusher" value="Krusher Brownie">
        Krusher Brownie [&#8377; 60/-]
                     </div>
                </div>
                <div class="col text-center">
                     <img src="../public/images/wings.PNG"
               width="200" height="200">
                     <div>
              <input type="checkbox" class="form-check-input"
           id="optWings" value="6 pc Hot Wings"> 6 pc
       Hot wings [&#8377; 80/-]
                     </div>
                </div>
              </div>
              <div class="row">
                <div class="col">
        <button onclick="OrderClick()" data-bs-target=
         "#billsummary" data-bs-toggle="modal" class="btn
      btn-danger w-100">Place Order</button>
                </div>
                <div class="modal fade" id="billsummary">
                     <div class="modal-dialog">
                         <div class="modal-content">
                             <div class="modal-header">
                                 <h3>Your Bill Summary</h3>
              <button class="btn-close" data-bs-dismiss=
                "modal"></button>
          </div>
```

```
            <div class="modal-body">
             <dl class="row">
              <dt class="col-3">Customer Name</dt>
                <dd class="col-9" id="lblName"></dd>
                <dt class="col-3">Mobile</dt>
                <dd class="col-9" id="lblMobile"></dd>
                <dt class="col-3">Meal Name</dt>
                <dd class="col-9" id="lblMeal"></dd>
                <dt class="col-3">Ad-On's</dt>
                <dd class="col-9" id="lblAdon"></dd>
                <dt class="col-3">Total Amount</dt>
                <dd class="col-9" id="lblAmount"> </dd>
                </dl>
                </div>
                 <div class="modal-footer">
                  <button class="btn btn-primary" data-bs-dismiss
                     ="modal">OK</button>
                                </div>
                            </div>
                        </div>
                    </div>
                  </div>
              </div>
          </div>
       </div>
    </section>
    <script src="../node_modules/jquery/dist/jquery.js">
    </script>
    <script src="../node_modules/bootstrap/dist/js
     /bootstrap.bundle.js"></script>
</body>
```

```
</html>
```

Ex:- With Name Mandatory

```
    <!DOCTYPE html>
    <html>
    <head>
    <title>KFC Online Order</title>
    <link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
    <link rel="stylesheet" href="../node_modules/bootstrap
     /dist/css/bootstrap.css">
    <style>
       body {
          background-color: maroon;
       }
    </style>
    <script>
       function OrderClick(){
    var cname = document.getElementById("txtName").value;
         if(cname=="") {
          document.getElementById("msg").innerHTML  ="Name
Required";
         } else {
          document.getElementById("lblName").innerHTML =
    document.getElementById("txtName").value;
          document.getElementById("lblMobile").innerHTML =
     document.getElementById("txtMobile").value;

    var burgerRadio = document.getElementById("optBurger");
    var rollerRadio = document.getElementById("optRoller");
var wingsCheckbox = document.getElementById("optWings");
 var krusherCheckbox =
     document.getElementById("optKrusher");

            var mealName = "";
            var adonName = "";

            var mealCost = 0;
            var adonCost = 0;

            if(burgerRadio.checked) {
               mealName = burgerRadio.value;
               mealCost = 120;
            }

            if(rollerRadio.checked) {
               mealName = rollerRadio.value;
               mealCost = 100;
            }

            if(krusherCheckbox.checked){
               adonName += krusherCheckbox.value + "<br>";
               adonCost = 60;
               mealCost += adonCost;
            }
            if(wingsCheckbox.checked) {
               adonName += wingsCheckbox.value + "<br>";
               adonCost = 80;
               mealCost += adonCost;
            }
            document.getElementById("lblMeal").innerHTML
```

```
=mealName ;
            document.getElementById("lblAdon").innerHTML
=adonName;
            document.getElementById("lblAmount").innerHTML =
"&#8377; " + mealCost;

           }
         }
         function HideMessage(){
       var cname = document.getElementById("txtName").value;
            if(cname!="") {
               document.getElementById("msg").innerHTML="";
            } else {

document.getElementById("msg").innerHTML="Name
Required";
            }
         }
      </script>
   </head>
   <body class="container-fluid">
      <header>
         <img src="../public/images/kfctop.PNG" width="100%"
height="160">
      </header>
      <section>
         <div class="accordion" id="kfcOrder">
            <div class="accordion-item">
               <div class="accordion-header">
                  <button class="btn btn-danger w-100" data-bs-
toggle="collapse" data-bs-target="#customer">
```

```
                  <h3>Customer Details</h3>
                </button>
              </div>
              <div id="customer" class="accordion-body
           accordion-collapse collapse show" data-bs-
parent="#kfcOrder">
                <dl>
                   <dt>Customer Name</dt>
                   <dd><input type="text" onblur="HideMessage
                    ()" id="txtName" class="form-control"></dd>
                   <dd id="msg" class="text-danger"></dd>
                   <dt>Mobile</dt>
                   <dd><input type="text" id="txtMobile" class=
                       "form-control"></dd>
                </dl>
              </div>
           </div>
           <div class="accordion-item">
              <div class="accordion-header">
                 <button class="btn btn-danger w-100" data-bs-
              toggle="collapse" data-bs-target="#meal">
                    <h3>Select Your Meal</h3>
                 </button>
              </div>
              <div class="accordion-body accordion-collapse-
collapse" id="meal"  data-bs-parent="#kfcOrder" >
                 <div class="row">
                    <div class="col text-center">
                       <img src="../public/images/omg1.PNG"
                   width="200" height="200">
                       <div>
```

```
            <input id="optBurger" value="OMG Burger"
name="meal" type="radio" class="form-check-input" >
OMG Burger [ &#8377; 120/-]
                        </div>
                      </div>
                      <div class="col text-center">
                        <img src="../public/images/omg2.PNG"
                     width="200" height="200">
                        <div>
            <input type="radio" id="optRoller" value="OMG
         Roller" name="meal" class="form-check-input">
OMG Roller [&#8377; 100/-]
                        </div>
                      </div>
                    </div>
                  </div>
               </div>
               <div class="accordion-item">
                  <div class="accordion-header">
                    <button class="btn btn-danger w-100" data-bs-
                    toggle="collapse" data-bs-target="#adon">
                     <h3>Select Ad-ON's</h3>
                    </button>
                  </div>
                  <div class="accordion-body accordion-collapse
      collapse" id="adon"  data-bs-parent="#kfcOrder" >
                    <div class="row">
                      <div class="col text-center">
                        <img src="../public/images/krusher1.PNG"
                     width="200" height="200">
                        <div>
```

```
              <input type="checkbox" class="form-check-input"
         id="optKrusher" value="Krusher Brownie">
     Krusher Brownie [&#8377; 60/-]
                    </div>
                  </div>
                  <div class="col text-center">
                    <img src="../public/images/wings.PNG"
                  width="200" height="200">
                    <div>
          <input type="checkbox" class="form-check-input"
      id="optWings" value="6 pc Hot Wings"> 6 pc Hot
 wings [&#8377; 80/-]
                    </div>
                  </div>
                </div>
                <div class="row">
                  <div class="col">
         <button onclick="OrderClick()" data-bs-target=
         "#billsummary" data-bs-toggle="modal" class="btn
      btn-danger w-100">Place Order</button>
                  </div>
                  <div class="modal fade" id="billsummary">
                    <div class="modal-dialog">
                      <div class="modal-content">
                        <div class="modal-header">
                          <h3>Your Bill Summary</h3>
                          <button class="btn-close" data-bs-
                       dismiss="modal"></button>
                        </div>
                        <div class="modal-body">
                    <dl class="row">
                        <dt class="col-3">CustomerName
                       </dt>
                        <dd class="col-9" id="lblName">
                        </dd>
                        <dt class="col-3">Mobile</dt>
                        <dd class="col-9" id="lblMobile">
                        </dd>
                        <dt class="col-3">Meal Name</dt>
                        <dd class="col-9" id="lblMeal">
                        </dd>
                        <dt class="col-3">Ad-On's</dt>
                        <dd class="col-9" id="lblAdon">
                        </dd>
                        <dt class="col-3">Total Amount
                        </dt>
                        <dd class="col-9" id="lblAmount">
            </dd>
                    </dl>
                </div>
                <div class="modal-footer">
         <button class="btn btn-primary" data-bs-
     dismiss="modal">OK</button>
                </div>
            </div>
        </div>
     </div>
   </div>
  </div>
 </div>
</div>
```

```
        </section>
        <script src="../node_modules/jquery/dist/jquery.js">
        </script>
        <script src="../node_modules/bootstrap/dist/js
         /bootstrap.bundle.js"></script>
    </body>
</html>
```

Task Hotel Registration Form

Ex:-

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Hotel Registration</title>
        <link rel="stylesheet" href="/node_modules/bootstrap-
icons/font/bootstrap-icons.css">
        <link rel="stylesheet"
href="/node_modules/bootstrap/dist/css/bootstrap.css">
        <link rel="shot icon"
href="/images.folder/Five_star_Hotel.png">
        <style>
            body{
                background-image: url(/images.folder/Hotel.png);
            }
            header{
                border-radius: 20px;
                margin-top: 20px;
                height: 70px;
                background-image:
url(/images.folder/Five_star_Hotel.png);
            }
        dd{
           margin-left: -10px;
        }
        dt{
           margin-right: -80px;
        }
     </style>
     <script>
        function Registerclick(){
           document.getElementById("lblName").innerHTML =
document.getElementById("txtName").value;
           document.getElementById("lblDate").innerHTML =
document.getElementById("txtDate").value;
           document.getElementById("lblDays").innerHTML =
document.getElementById("txtDays").value;
           document.getElementById("lblPerson").innerHTML =
document.getElementById("txtPersons").value;

           var roomType;
           var roomCost;

           var amenityName = '';
           var amenityCost;

           var deluxroom =
document.getElementById("deluxroom");
           var suitroom = document.getElementById("suitroom");

           if(deluxroom.checked)
           {
               roomType = deluxroom.value;
```

```
                 roomCost = 2500;
            }
            if(suitroom.checked)
            {
                roomType = suitroom.value;
                roomCost = 4000;
            }

            var optac = document.getElementById("optac");
            var optlocker = document.getElementById("optlocker");

            if(optac.checked)
            {
                amenityName += optac.value + "<br>";
                amenityCost = 1000;
                roomCost += amenityCost;
            }

            if(optlocker.checked)
            {
                amenityName += optlocker.value + "<br>";
                amenityCost = 300;
                roomCost += amenityCost;
            }

            document.getElementById("lblType").innerHTML =
roomType;
            document.getElementById("lblAmenities").innerHTML =
amenityName;
            document.getElementById("lblAmount").innerHTML =
"&#8377;" + document.getElementById("txtAdvance").value;
```

```
            var total;
            var balance;

            var Persons = txtPersons.value;
            var days = txtDays.value;
            var advance = txtAdvance.value;

            if(txtPersons.value<=2)
            {
               total = roomCost * days;
               balance = total - advance;
               document.getElementById("lblBalance").innerHTML =
`${balance}`;
            }else{
               total = roomCost * days;
               balance = total - advance;
               document.getElementById("lblBalance").innerHTML =
`${balance}`;
            }

         }
      </script>
   </head>
   <body class="container-fluid">
      <header class="text-white text-center">
         <h3 class="p-2 mt-4">Hotel Registration Form</h3>
      </header>
      <section>
         <div class="accordion my-5" id="orderform">
            <div class="accordion-item">
```

```
                <div class="accordion-header">
                    <button class="btn btn-success w-100" data-bs-
toggle="collapse" data-bs-target="#customer">Customer
Info</button>
                </div>
                <div id="customer" class="accordion-body accordion-
collapse collapse show" data-bs-parent="#orderform">

            <dl class="row " style="text-align: center;" >
            <dt class="col-4 ms-4 mb-2">Customer Name</dt>
              <dd class="col-6"><Input type="text" id="txtName"
class="form-control" placeholder="text"></dd>
            <dt class="col-4 ms-4 mb-2">Check in Date</dt>
               <dd class="col-6"><Input type="date" id="txtDate"
class="form-control" placeholder="date"></dd>
               <dt class="col-4 ms-4 mb-2">Total No of Days</dt>
                                <dd class="col-6"><Input type="number"
id="txtDays" class="form-control" placeholder="days"></dd>
               <dt class="col-4 ms-4 mb-2">Total No of Persons</dt>
                                <dd class="col-6"><Input type="number"
id="txtPersons" class="form-control"
placeholder="Persons"></dd>
                        </dl>

                </div>
            </div>
            <div class="accordion-item">
                <div class="accordion-header">
                    <button class="btn btn-success w-100" data-bs-
toggle="collapse" data-bs-target="#room">Select Room
Type</button>
```

```
          </div>
          <div id="room" class="accordion-body accordion-
collapse collapse " data-bs-parent="#orderform">
            <div class="row">
              <div class="col">
                <div class="card" style="width: 250px;">
                  <img src="/images.folder/Hotel.png"
alt="delux" class="card-img-top" height="150">
                </div>
                <div class="card-header">
                  <input type="radio" id="deluxroom"
name="rooms" value="Delux Room" class="form-check-
input">Delux Hotel
                </div>
              </div>

              <div class="col">
                <div class="card" style="width: 250px;">
                  <img src="/images.folder/Suites_Hotel.png"
alt="delux" class="card-img-top" height="150">
                </div>
                <div class="card-header">
                  <input type="radio" id="suitroom"
name="rooms" value="Suite Room" class="form-check-
input">Hotel
                </div>
              </div>
              <div class="col">
                <div class="card" style="width: 250px;">
```

```
                              <img
src="/images.folder/Suites_Washaroom.png" alt="delux"
class="card-img-top" height="150">
                          </div>
                          <div class="card-header">
                              <input type="radio" id="suitroom"
name="rooms" value="Suite Room" class="form-check-
input">Washroom
                          </div>
                      </div>
                      <div class="col">
                          <div class="card" style="width: 250px;">
                              <img
src="/images.folder/Suites_bedroom.png" alt="delux"
class="card-img-top" height="150">
                          </div>
                          <div class="card-header">
                              <input type="radio" id="suitroom"
name="rooms" value="Suite Room" class="form-check-
input">Suite Room
                          </div>
                      </div>
                  </div>
              </div>
          </div>
          <div class="accordion-item">
              <div class="accordion-header">
                  <button class="btn btn-success w-100" data-bs-
toggle="collapse" data-bs-target="#amenities">Select
Amenities</button>
              </div>
```

```
                <div id="amenities" class="accordion-body accordion-
collapse collapse " data-bs-parent="#orderform">
                    <div class="row">
                        <div class="col">
                            <div class="card" style="width: 250px;">
                                <img src="/images.folder/AC.png.jpg"
class="card-img-top" height="150">
                            </div>
                            <div class="card-header">
                                <input type="checkbox" id="optac"
value="Ac Room" class="form-check-input">A/C
                            </div>
                        </div>
                        <div class="col">
                            <div class="card" style="width: 250px;">
                                <img src="/images.folder/Locker.room.png"
class="card-img-top" height="150">
                            </div>
                            <div class="card-header">
                                <input type="checkbox" id="optlocker"
value="Locker Room" class="form-check-input">Locker
                            </div>
                        </div>
                    </div>
                </div>
            </div>
            <div class="accordian-item">
                <div class="accordian-header">
                    <button class="btn btn-success w-100" data-bs-
toggle="collapse" data-bs-target="#advance">Advance
Amount</button>
```

```
                </div>
                <div id="advance" class="accordion-body accordion-
collapse collapse " data-bs-parent="#orderform">
                    <input type="number" id="txtAdvance"
class="form-control" placeholder="Advance Amount">
                </div>

                <div class="row mt-3">
                    <div class="col">
                        <button onclick="Registerclick()" class="btn btn-
danger w-100" data-bs-target="#summary" data-bs-
toggle="modal"> Register </button>
                        <div class="modal fade" id="summary">
                            <div class="modal-dialog">
                                <div class="modal-content">
                                    <div class="modal-header">
                                        <h3>Your Booking Bill</h3>
            <button class="btn-close" data-bs-dismiss="modal">
              </button>
              </div>
              <div class="modal-body">
              <dl class="row">
              <dt class="col-4 mb-2">Customer Name</dt>
              <dd class="col-8" id="lblName"></dd>
              <dt class="col-4 mb-2">Check In Date</dt>
               <dd class="col-8" id="lblDate"></dd>
                <dt class="col-4 mb-2">Total No of Days</dt>
                 <dd class="col-8" id="lblDays"></dd>
                  <dt class="col-4 mb-2">Total No of Persons</dt>
                  <dd class="col-8" id="lblPerson"></dd>
                  <dt class="col-4 mb-2">Room Type</dt>
            <dd class="col-8" id="lblType"></dd>
            <dt class="col-4 mb-2">Amenities</dt>
            <dd class="col-8" id="lblAmenities"></dd>
            <dt class="col-4 mb-2">Advance Amount</dt>
            <dd class="col-8" id="lblAmount"></dd>
            <dt class="col-4 mb-2">Balance Amount</dt>
            <dd class="col-8" id="lblBalance"></dd>
            </dl>
            </div>
            <div class="modal-footer">
      <button class="btn btn-success" data-bs-dismiss="modal">
      OK</button>
                        </div>
                      </div>
                    </div>
                  </div>
                </div>
            </div>
          </div>
        </div>
      </section>

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

JavaScript Special Operators

1. Ternary Operator   [?:]

It is used in decision making, where it defines a condition and statements to execute on true or false.

Syntax:-

```
(condition) ? statement_if_true : statement_if_false

if(condition)

{

}

else

{

}
```

Ex:-

```
<script>
   var stock = false;
   document.write(`${(stock==true)?"Available":"Out of 
Stock"}`);
</script>
```

2. new operator

- It is dynamic memory allocating operator.
- It allocates memory for object and loads its members into

memory.

```
var collection = new Array();
collection.length
collection.find()
collection.push() etc..
```

3. void Operator :

- It discards the return value.
- Element will not returns any value.

Syntax:-

```
<a href="javascript:void()"> Home </a>
```

Ex:-

```
<h2>Void Demo</h2>
<a href="javascript:void(location.href='home.html')">
  Home</a>
```

4. typeof

- It is to returns the data type of value stored in a reference.
- so that you can verify the data type.

Syntax:-

```
typeof    variableName;
typeof    object.Property;
            OR
```

```
        var price = 45000;
        document.write(typeof price);          // number
```

Ex:- 1

```
      <script>
        var product = {
            Name: "TV",
      Price: 45000.55,
      Stock: true
   }
   for(var property in product){
      document.write(`${property} : ${typeof product[property]}
<br>`);
   }
</script>
```

Ex:- 2

```
      <script>
   fetch('http://fakestoreapi.com/products/1&#39;`)
   .then(function(response){
      return response.json();
   })
   .then(function(data){
      for(var property in data)
      {
         document.write(`${property}-${typeof
data[property]}<br>`);
      }
   })
</script>
```

5. instanceof Operator

- It is used to check the class name, from where the given instance is derived. [instance is object]
- It is a boolean operator that returns true or false.
- It is a boolean operator that is used to verify the class name of the object defined.
- It returns true if object belongs to the class specified.

Syntax:-

```
    "object"  instanceof   className;          ture/ false
```

EX:-

```
      <script>
          class Employee
          {
          }
          class Student
         {
         }

   let david = new Employee();
   let john = new Student();
   let products = new Array();

   document.write(david instanceof Student       );       // false
   document.write(david instanceof Employee);         // true
   document.write(products instanceof Object);       // true
</script>
```

6. in operator

- It is used to verify the existance of any property in object.
- It is a boolean operator that returns true when given property is available in object.

Syntax:-

```
property   in  object;                         // true
"Price"     in   product;                      // true
```

EX:-

```
<script>
    let product = {
        Name: "TV",
    Price: 45000.44
  }
  document.write(`Product have Price ${"Price" in product}`);
</script>
```

7. of Operator

- It is used by itreatorto read the value of collection.

Syntax:-

```
for(var item of collection)
{
}
```

8. delete operator

- It is used to delete any property from object.

Syntax:-

```
delete  object.property;
```

- You can't delete properties of built-in objects, as they are readonly.

Ex:-

```
      <script>
         var product = {
       "Name": "TV",
       "Price": 45000.55,
       "Stock":true
    };
    delete product.Price;
   for(var property in product){
      document.write(property + "<br>");
   }
</script>
```

Syntax:-

```
        delete Math.PI;                    // invalid
        delete collection.length;      // invalid
```

JavaScript Statements

- A statement computer programming is used to control the execution flow.
- JavaScript statements are classified into following types
    1. Selection Statements
    2. Iteration and Looping Control Statements
    3. Jump Statements
    4. Exception Handling Statements

Selection Statements:

- The selection statements are used in decision making.
- The statements are executed according state and situation.
- The selection statement keywords are:

    if, else, switch, case, default

If Select:

- It is a decision making statement used to execute a block of code when given condition is satisfied and another set of code when condition evaluates of false.
- It is used in following forms
    a) Forward Jump
    b) Simple Decision
    c) Multi Level Decision
    d) Multiple Decisions

Forward Jump : It is a decision making technique where you will not configure alternate task.

- In this statements are executed only when condition is true.
- There will be no alternative.

Syntax:-

```
if ( condition )
{
  statement if true;
```

```
    }
```

The IF Selector

1. Forward Jump

```
if()
{
}
```

2.Simple Decision: In this a set of statements are executed when condition is true and another set of statements are executed when condition is false. It can contain only one alternative.

Syntax:-

```
if ( condition )
{
 statement if true;
}
else
{
 statement if false;
}
```

you configure only one alternative.

Note : if is a statement and else is a clause.

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>Login</title>
      <script>
         var userDetails = {
            UserName: 'john_nit',
            Password: 'john@12'
         };

         function LoginClick(){
            var UserName =
document.getElementById("UserName").value;
            var Password =
document.getElementById("Password").value;
            var msg = document.getElementById("msg");
            if(UserName==userDetails.UserName &&
Password==userDetails.Password)
            {
               document.write("Login Success..");
            } else {
               msg.innerHTML = "Invalid User Name / Password";
            }
         }
      </script>
   </head>
```

```
  <body>
    <h2>User Login</h2>
    <dl>
      <dt>User Name</dt>
      <dd><input type="text" id="UserName"></dd>
      <dt>Password</dt>
      <dd><input type="password" id="Password"></dd>
    </dl>
    <button onclick="LoginClick()">Login</button>
    <h2 id="msg" align="center"></h2>
  </body>
</html>
```

3. Nested Decision

- It defines a condition within another condition
- It is used to verify every condition and return individual set of statements.

Syntax:-

```
    if(condition1)
    {
        if(condition2)
        {
        }
        else
        {
        }
    }
    else
    {
    }
```

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>Login</title>
    <script>
       var userDetails = {
          UserName: 'john_nit',
          Password: 'john@12'
       };

       function LoginClick(){
          var UserName =
document.getElementById("UserName").value;
          var Password =
document.getElementById("Password").value;
          var msg = document.getElementById("msg");
          if(UserName==userDetails.UserName)
          {
             if(Password==userDetails.Password)
             {
                document.write("Login Success..");
            }
            else {
                msg.innerHTML="Invalid Password";
            }
        }
        else {
            msg.innerHTML="Invalid User Name";
        }
    }
    </script>
  </head>
  <body>
    <h2>User Login</h2>
    <dl>
      <dt>User Name</dt>
      <dd><input type="text" id="UserName"></dd>
      <dt>Password</dt>
      <dd><input type="password" id="Password"></dd>
    </dl>
    <button onclick="LoginClick()">Login</button>
    <h2 id="msg" align="center"></h2>
  </body>
</html>
```

Ex:-

```
    <!DOCTYPE html>
```

```
  <html>
<head>
  <title>Login</title>
  <script>
    var userDetails = {
      CardNumber: "4444555566667777834",
      Cvv: '5606',
      Expiry:'2023'
    };
    function VerifyCard(){
      var card = document.getElementById("Card").value;
      if(card==userDetails.CardNumber) {
        document.getElementById("Cvv").disabled=false;
        document.getElementById("Card").disabled=true;
      } else {
        document.getElementById("Cvv").disabled=true;
      }
    }
    function VerifyCvv(){
      var cvv = document.getElementById("Cvv").value;
      if(cvv==userDetails.Cvv) {
        document.getElementById("Expiry").disabled=false;
        document.getElementById("Cvv").disabled=true;
      } else {
        document.getElementById("Expiry").disabled=true;
      }
    }
```

```
        function VerifyExpiry(){
            var expiry = document.getElementById("Expiry").value;
            if(expiry==userDetails.Expiry) {
                document.getElementById("btnPay").disabled=false;
                document.getElementById("Expiry").disabled=true;
            } else {
                document.getElementById("btnPay").disabled=true;
            }
        }

        function PayClick(){
            document.write("Payment Success..");
        }
    </script>
  </head>
  <body>
    <fieldset>
        <legend>Payment Details</legend>
        <dl>
            <dt>Card Number</dt>
            <dd><input type="text" id="Card"
onkeyup="VerifyCard()"></dd>
            <dt>CVV</dt>
            <dd><input type="text" id="Cvv"
onkeyup="VerifyCvv()" disabled size="4"></dd>
            <dt>Expiry</dt>
            <dd>
```

```
                <select id="Expiry" onchange="VerifyExpiry()"
disabled>
                    <option>2022</option>
                    <option>2023</option>
                    <option>2024</option>
                </select>
            </dd>
        </dl>
        <button onclick="PayClick()" id="btnPay"
disabled>Pay</button>
    </fieldset>
  </body>
</html>
```

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>Login</title>
    <script>
        var userDetails = {
            CardNumber: "44445555666677778834",
            Cvv: '5606',
            Expiry:'2023'
        };
        function PayClick(){
            var card = document.getElementById("Card").value;
            var cvv = document.getElementById("Cvv").value;
            var expriy = document.getElementById("Expiry").value;
            var msg = document.getElementById("msg");

            if(card==userDetails.CardNumber)
            {
              if(cvv==userDetails.Cvv){
                if(expriy==userDetails.Expiry)
                {
                  document.write("<h2>Payment Success</h2>");
                }else {
                  msg.innerHTML="Invalid Expiry Year";
                }
              } else {
                msg.innerHTML="Invalid CVV";
              }

            } else {
              msg.innerHTML="Invalid Card Number";
            }
        }
    </script>
  </head>
  <body>
    <fieldset>
        <legend>Payment Details</legend>
        <dl>
```

```
            <dt>Card Number</dt>
            <dd><input type="text" id="Card""></dd>
            <dt>CVV</dt>
            <dd><input type="text" id="Cvv" size="4"></dd>
            <dt>Expiry</dt>
            <dd>
                <select id="Expiry" >
                    <option>2022</option>
                    <option>2023</option>
                    <option>2024</option>
                </select>
            </dd>
        </dl>
        <button onclick="PayClick()">Pay</button>
        <h2 align="center" id="msg"></h2>
    </fieldset>
  </body>
</html>
```

4. Multi Level Decision : In this a condition comprises of another condition with in the context.

Syntax:-

```
        if (conditon1 )
        {
            if(condition2)
            {
            }
```

```
   else
   {
   }
}
else
{
}
```

Ex:-

```
<!DOCTYPE html>
<html>
<head>
<title>Decision Making</title>
<script>
    var userdetails = {
        CardNumber: "4444555566667777893 8",
        Cvv:"4040",
        Expiry:"2023"
    }

function PayClick(){
  var card = document.getElementById("txtCard").value;
  var cvv = document.getElementById("txtCvv").value;
  var expiry = document.getElementById("lstExpiry").value;
  var msg = document.getElementById("msg");
  if(userdetails.CardNumber==card){
          if(userdetails.Cvv==cvv) {
            if(userdetails.Expiry==expiry)
            {
                document.write("Payment Success...");
            } else {
                msg.innerHTML = "Invalid Expiry";
            }
        } else {
            msg.innerHTML = "Invalid CVV";
        }
    }
    else {
        msg.innerHTML = "Invalid Card Number";
    }
}
function VerifyCard(){
   var card = document.getElementById("txtCard").value;
    if(userdetails.CardNumber==card) {
      document.getElementById("txtCvv").disabled = false;
    } else {
       document.getElementById("txtCvv").disabled = true;
    }
}
function VerifyCvv(){
    var cvv = document.getElementById("txtCvv").value;
    if(userdetails.Cvv==cvv) {
  document.getElementById("lstExpiry").disabled = false;
    }else {
          document.getElementById("lstExpiry").disabled = true;
           }
       }
       function VerifyExpiry(){
     var expiry = document.getElementById("lstExpiry").value;
           if(userdetails.Expiry==expiry) {
           document.getElementById("btnPay").disabled = false;
           } else {
            document.getElementById("btnPay").disabled = true;
           }
       }
   </script>
</head>
<body>
   <fieldset>
       <legend>Payment Details</legend>
       <dl>
           <dt>Card Number</dt>
          <dd><input type="text" onkeyup="VerifyCard()" id=

                 "txtCard"></dd>
           <dt>CVV</dt>
           <dd><input type="text" onkeyup="VerifyCvv()" id=

                 "txtCvv" disabled size="4"></dd>
           <dt>Expiry</dt>
           <dd>
    <select id="lstExpiry" disabled onchange="VerifyExpiry()">
```

```
                <option>2022</option>
                <option>2023</option>
                <option>2024</option>
              </select>
            </dd>
          </dl>
          <button disabled id="btnPay" onclick="PayClick()">Pay

          </button>
        </fieldset>
        <h2 id="msg" align="center"></h2>
    </body>
</html>
```

5. Multiple Conditions or Decisions :

- In this more than one alternative is provided for a set of statements.
- It is defined to handling multiple conditions.
- multiple conditions are required for multiple choices for same set of statements.

Syntax:-

```
    if (condition1)
    {
    statements_if_condition1_true;
    }
```

```
else if (condition2)
{
statements_if_condition2_true;
}
else if (condition3)
{
statements_if_both conditions_false;
}
else
{
}
```

Ex:- Amazon Login

```
<!DOCTYPE html>
<html>
<head>
<title>Amazon Login</title>
<link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
<link rel="stylesheet" href="../node_modules
  /bootstrap/dist/css/bootstrap.css">
<script>
   var userdetails = {
      Email: "david@gmail.com",
      Mobile: '9876543211',
      Password: 'john@12'
```

```
        }
        var userid = "";
        function ContinueClick(){
            userid = document.getElementById("txtUser").value;
            if(userid==userdetails.Email){

                document.getElementById("password").style.display
= "block";
      document.getElementById("userid").style.display = "none";
            }
            else if(userid==userdetails.Mobile) {

                document.getElementById("password").style.display
= "block";
      document.getElementById("userid").style.display = "none";
            } else {
          document.getElementById("userError").innerHTML =
       "Please Enter valid Email or Mobile";
            }
        }
        function ConfirmClick(){
      var password = document.getElementById("txtPwd").value;
            if(password==userdetails.Password) {
                document.write(`Login Success ${userid}`);
            } else {
                document.getElementById("pwdError").innerHTML
            = "Invalid Password";
            }
        }
    </script>
  </head>
  <body class="container-fluid">
    <div class="d-flex justify-content-center align-items-
center" style="height: 600px;">
        <div>
            <h3>Amazon Signin</h3>
            <div id="userid">
                <label class="form-label">Sign in with Email or
            Mobile</label>
                <div>
            <input type="text" id="txtUser" class="form-control">
                    <div class="text-danger" id="userError"></div>
                </div>
                <div class="mt-2">
                    <button class="btn btn-warning w-100" onclick=

                      "ContinueClick()">Continue
                    </button>
                </div>
            </div>
            <div id="password" style="display: none;">
                <label class="form-label">Your Password</label>
                <div>
                    <input type="password" id="txtPwd" class=
                "form-control">
              <div class="text-danger" id="pwdError"></div>
            </div>
            <div class="mt-2">
              <button class="btn btn-warning w-100" onclick=

                "ConfirmClick()">Confirm
              </button>
            </div>
          </div>
        </div>
      </div>
    </body>
</html>
```

Note : Issue with "If" selector while using multiple choices is verfiying every condition unitl the given criteria is not satisfied.

- It will take more time for compiling and rendering output.

Switch Selector

- Switch In electronics is used to control the flow of electrons.
- In electronics switches are classified into various types
- There in electronics switches are various types
    a) Push Button Switch
    b) Toggle Switch
    c) Selector Switch
    d) Joystick Switch etc..
- Switch can control the flow of execution by selecting only

the block that matches given condition.

Syntax:-

```
switch(value/expression)
{
 case:
   statement
   jump;
 default:
   statement
   jump;
}
```

Ex:-

```
<script>
     var n=3;
     switch(n)
{
   case 1:
   document.write("One");
   break;
   case 2:
   document.write("Two");
   break;
   case 3:
   document.write("Three");
   break;
   default:
   document.write("Please Enter 1 to 3");
   break;
}
```

</script>

FAQ: Can we define switch without default?

Ans: Yes. if default is not defined it will be void when switch value is not matching with any case.

FAQ: Can we define default above or before case? or in between the cases?

Ans: Yes.

FAQ: Can we define case without break? or jump statement?

Ans: Yes. But execution continues to next case and stops when break occured or will stop at the end of switch.

FAQ: Can we define case with return as jump statement or default?

Ans: Yes.

FAQ: What is difference between break and return?

Ans: break will stop the block and stays in script.
return will stop the script. It quits the block.
- return is a jump statement, which terminates the compiling the compiling and exits the function and script.

```
var n = 1;
    switch(n)
        {
            case 1:
            document.write("One");
            return;
            case 2:
            document.write("Two");
            break;
            case 3:
            document.write("Three");
            break;
            default:
            document.write("Please Enter number 1 to 3");
            break;
        }
    }
    f1();
```

FAQ: Can we define case value as string type?
Ans: Yes. but it is case sensitve.

FAQ: How to handle multiple case values for a set of statements?
Ans: a) You can define multiple cases for one block.
      b) You can change the capitalization for text.

Ex:-

```
        <script>
              function f1(){
                var choice = "N";
                switch(choice)
            {
                case "y":
                case "Y":
                document.write("You selected Yes");
                break;
                case "n":
                case "N":
```

```
                document.write("You selected No");
                break;
                default:
                document.write("Please enter y or n");
                break;
                }
            }
        f1();
    </script>
```

Ex:-

```
        <script>
          function f1(){
            var choice = "NO";
            switch(choice.toLowerCase())
        {
            case "yes":
            document.write("You selected Yes");
            break;
            case "no":
            document.write("You selected No");
            break;
            default:
            document.write("Please enter y or n");
            break;
              }
        }
        f1();
    </script>
```

FAQ : How to handle a lengthy string as it is case sensitive?

Ans : We convert the case and verify.

Ex:-

```
<script>
  function f1(){
    var choice = "YeS";
    switch(choice.toLowerCase())
    {
       case "yes":
       document.write("You Selected Yes");
       break;
       case "no":
       document.write("You Selected No");
       break;
       default:
       document.write("Please Enter yes or no only");
       break;
    }
  }
  f1();
</script>
```

FAQ : How to define case for range of values?

Ans : By using boolean expressions.

Switch can have only boolean true.

- If multiple conditions are matching, then it will execute only the first.

Ex:-

```
<script>
```

```
        function f1(){
          var n = 6;
    switch(true)
    {
       case (n>=1 && n<=10):
       document.write(`Your number ${n} is between 1 to 10`);
       break;
       case (n>10 && n<=20):
       document.write(`You number ${n} is between 11 to 20`);
       break;
       case (n>5 && n<=20):
       document.write(`Your number ${n} is between 5 to 20`);
       break;
       default:
       document.write(`Your number ${n} is -ve or above 20`);
       break;
    }
  }
  f1();
</script>
```

FAQ : If switch is using boolean expression then what should be the switch value?

Ans : only "true"

Ex:-

```
      <script>
       function f1(){
          var n = 12;
          switch(true)
          {
```

```
        case n>=1 && n<=10:
        document.write(`Your number ${n} is between 1 to 10`);
        break;
        case n>11 && n<=20:
        document.write(`Your number ${n} is between 11 to 20`);
        break;
      }
    }
    f1();
</script>
```

FAQ : How to define a regular expression as case?

Ans : Regular Expression can be defined using "/ /", We verify using match() method.

FAQ: Can we define "if" condition in switch case?

Ans: Yes

Syntax:-

```
        case (condition):
         if() { }
         break;
```

Ex:-

```
        <script>
          function f1(){
            var n = 5;
      switch(true)
      {
         case (n>=1 && n<=10):
          if(n==5) {
             document.write("You Entered 5");
          } else {
             document.write(`Your number ${n} is between 1 to 10`);
          }
         break;
         case (n>10 && n<=20):
         document.write(`You number ${n} is between 11 to 20`);
         break;
         case (n>5 && n<=20):
         document.write(`Your number ${n} is between 5 to 20`);
         break;
         default:
         document.write(`Your number ${n} is -ve or above 20`);
         break;
      }
    }
    f1();
</script>
```

FAQ: Can we define a switch inside case?

Ans: No. Not recommended. Always defined a function and call using function.

Ex:- Switch and Cascading Dropdown

```
       <!DOCTYPE html>
       <html>
       <head>
       <title>Shopping</title>
       <link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
       <link rel="stylesheet" href="../node_modules
```

```
          /bootstrap/dist/css/bootstrap.css">
       <script>
          var categories = ["Select a Category", "Electronics",
"Footwear", "Fashion"];
          var electronics = ["Select Electronic Product","Samsung
Mobile", "boAt NeckBand"];
          var footwear = ["Select Footwear", "Nike Casuals", "Lee
Cooper Boot"];
          var fashion = ["Select Fashion", "Backpack", "Mens
Cotton Jacket"];
          var products = [];

          var data = [
             {Name: 'Samsung Mobile', Price: 13000.44, Photo:
        '../public/images/mobile.png'},
             {Name: 'boAt NeckBand', Price: 4000.44, Photo:
        '../public/images/neckband.png'},
             {Name: 'Nike Casuals', Price: 6000.44, Photo:
        '../public/images/shoe.jpg'},
             {Name: 'Lee Cooper Boot', Price: 3000.44, Photo:
        '../public/images/shoe1.jpg'},
             {Name: 'Backpack', Price: 2000.44, Photo:
        '../public/images/backpack.jpg'},
             {Name: 'Mens Cotton Jacket', Price: 1200.44, Photo:
        '../public/images/jacket.jpg'},
          ];

          function LoadProducts(){
             document.getElementById("lstProducts").innerHTML
        ="";
             for(var item of products)
```

```
            {
                var option = document.createElement("option");
                option.text = item;
                option.value = item;

document.getElementById("lstProducts").appendChild(option);
            }
        }
        function bodyload(){
            for(var item of categories)
            {
                var option = document.createElement("option");
                option.value = item;
                option.text = item;

document.getElementById("lstCategories").appendChild(option);
            }
            GetCartItemsCount();
        }
        function CategoryChanged(){
            var categoryName = document.getElementById
                   ("lstCategories").value;
            switch(categoryName)
            {
                case "Electronics":
                products = electronics;
                LoadProducts();
                break;
                case "Footwear":
                products = footwear;
                LoadProducts();
                break;
                case "Fashion":
                products = fashion;
                LoadProducts();
                break;
                default:
                products = ["Please Select a Category"];
                LoadProducts();
                break;
            }
        }
        var searchedProduct = {};
        function ProductChanged(){
            var productName = document.getElementById
              ("lstProducts").value;
            searchedProduct = data.find(function(product){
                return product.Name=productName;
            });
            document.getElementById("lblName").innerHTML =
            searchedProduct.Name;
            document.getElementById("lblPrice").innerHTML =
            searchedProduct.Price;
            document.getElementById("imgProduct").src =
            searchedProduct.Photo;
        }
        var cartItems = [];

        function GetCartItemsCount(){
            document.getElementById("lblCart").innerHTML =
    cartItems.length;
        }
```

```
function AddToCartClick(){
   cartItems.push(searchedProduct);
   alert(`${searchedProduct.Name} Added to Cart`);
   GetCartItemsCount();
}
function LoadCartItems(){
   document.querySelector("tbody").innerHTML="";
   for(var item of cartItems)
   {
      var tr = document.createElement("tr");
      var tdName = document.createElement("td");
      var tdPrice = document.createElement("td");
      var tdPhoto = document.createElement("td");
      var tdRemove = document.createElement("td");

      tdName.innerHTML = item.Name;
      tdPrice.innerHTML = item.Price;

      var img = document.createElement("img");
      img.height= "50";
      img.width="50";
      img.src=item.Photo;

      tdPhoto.appendChild(img);

      tdRemove.innerHTML = `
        <button class="btn btn-outline-danger"> <span
   class="bi bi-trash-fill"></span> </button>`;

      tr.appendChild(tdName);
```

```
            tr.appendChild(tdPrice);
            tr.appendChild(tdPhoto);
            tr.appendChild(tdRemove);

            document.querySelector("tbody").appendChild(tr);
          }
        }
    </script>
</head>
<body class="container-fluid" onload="bodyload()">
    <header class="bg-danger text-center text-white p-2">
        <h1> <span class="bi bi-cart3"></span> Amazon
    Shopping</h1>
    </header>
    <div class="row mt-2">
        <div class="col-3">
          <div class="mt-2">
            <label class="form-label">Select a Category</label>
            <div>
                <select onchange="CategoryChanged()"
          class="form-select" id="lstCategories">

                </select>
            </div>
          </div>
          <div class="mt-2">
            <label class="form-label">Select a Product</label>
            <div>
                <select onchange="ProductChanged()" class="form-
          select" id="lstProducts">
                </select>
```

```
              </div>
            </div>
            <div class="mt-2">
              <label class="form-label">Preview</label>
              <div class="card">
                <div class="card-header  text-center">
                  <h2 id="lblName"></h2>
                </div>
                <div class="card-body text-center">
                  <img id="imgProduct" width="200"
                 height="200">
                  <p id="lblPrice"></p>
                </div>
                <div class="card-footer">
                  <button onclick="AddToCartClick()" class="btn
                btn-danger w-100">
                      <span class="bi bi-cart4"></span>
                      Add to Cart
                  </button>
                </div>
              </div>
            </div>
          </div>
          <div class="col-7">
          </div>
          <div class="col-2">
            <div class="mt-2">
              <button onclick="LoadCartItems()" class="btn btn-
          warning w-100" data-bs-target="#cart" data-bs-
toggle="modal" >
                <span class="bi bi-cart2"></span>
```

```
   [<label id="lblCart"></label>]
    Your Cart Items
</button>
<div class="modal fade" id="cart">
  <div class="modal-dialog">
     <div class="modal-content">
        <div class="modal-header">
           <h3>Your Cart Items</h3>
           <button class="btn-close" data-bs-
      dismiss="modal"></button>
        </div>
        <div class="modal-body">
           <table class="table table-hover">
              <thead>
                 <tr>
                    <th>Name</th>
                    <th>Price</th>
                    <th>Preview</th>
                 </tr>
              </thead>
              <tbody>
              </tbody>
           </table>
        </div>
        <div class="modal-footer">
           <button class="btn btn-primary" data-bs-
          dismiss="modal">OK</button>
        </div>
     </div>
  </div>
</div>
```

```
        </div>
       </div>
     </div>
     <script src="../node_modules/jquery/dist/jquery.js">
     </script>
     <script src="../node_modules/bootstrap/dist/js
      /bootstrap.bundle.js"></script>
   </body>
</html>
```

Summary - Selection Statements
- if, else, switch, case, default
- Forward jump
- Simple
- Multiple
- Multi Level

Looping Control and Iteration Statements

- Looping is the process of executing a set of statements repeatedly until given condition is satisfied.
- Looping requires
    - a) Initialization
    - b) Condition
    - c) Counter
- Loops are created by using

    for, while, do while
- initialization defines where to start.
- condition specifies the condition to loop and exit.
- counter defines the looping counter next or previous [increment and decrement]

JavaScript Loops are defined by using

a) for
b) while
c) do while

The "for" loop:

- It is used when the exact iterations are known to developer and iteration count will not change dynamically.
- It requires initialization, condition and counter.

****

Syntax:-

```
for(initialization; condition; counter)
{
}
```

- Initialization, condition, counter values are optional in 'for' declaration.

Syntax:-

```
for( ; ;)        for(var i=4;  ; i++)
{       {
}       }
```

- You can have counter decrement or counter with step value more than one.

```
i++,      i=i+2,
i--;
```

Ex:-

```
      <head>
   <script>
      function bodyload(){
         var data = [
            {Category: "Electronics", Products: ["TV","Mobile"]},
            {Category: "Footwear", Products: ["Nike Casuals",
        "Lee Boot"]}
           ];
         for(var i=0; i<data.length; i++)
         {
            var olLi = document.createElement("li");
            olLi.innerHTML = data[i].Category;
            document.querySelector("ol").appendChild(olLi);
            for(var j=0; j<data[i].Products.length; j++)
            {
               var ul = document.createElement("ul");
               var ulLi = document.createElement("li");
               ulLi.innerHTML = data[i].Products[j];
               ul.appendChild(ulLi);
               document.querySelector("ol").appendChild(ul);
            }
         }
      }
   </script>
</head>
<body onload="bodyload()">
   <ol>

   </ol>
</body>
```

The "while" Loop:

- It is used when iterations are unknown and iteration counter may change dynamically.
- It will start the loop only when condition is true.

Syntax:-

```
 while(condition)
 {
   statements;
   counter;
 }
```

Ex:-

```
var  i = 1;
while(i<=10)
{
  document.write(i + "<br>");
  i++;
}
```

The "do" loop:

- It is similar to while loop but ensures that the statements are executed atleast once even when the condition is false.

Syntax:-

```
do
{
  statements;
  counter;
} while (condition)
```

Ex:-

```
<script>
   function f1(){
      var i=11;
      do
      {
         document.write(i + "<br>");
         i++;
      }while(i<=10);
   }
   f1();
</script>
```

Note: Don't use loops for reading data from collection. Always use them for a counter to execute.

Task-1: Validate PIN for 3 attempts and block if all 3 went wrong.

```
   <!DOCTYPE html>
   <html>
<head>
   <title>Loop</title>
   <script>
      var userdetails = {
         Pin: '67042'
      }
      function SubmitClick(){

      }
   </script>
   </head>
   <body>
      <fieldset>
         <legend>Your PIN [Max 3 attempts]</legend>
         <input type="text" id="txtPin" size="5">
<button>Submit</button>
      </fieldset>
   </body>
</html>
```

Task-2: Write a program to convert number to words

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>Loop</title>
      <script>
          function Convert(){
            var amount = document.getElementById
             ("txtAmount").value;
            var hundreds = ['One', 'Two', 'Three', 'Four'];
            for(var i=0; i<hundreds.length; i++)
            {
               if(hundreds[i]==amount) {
                  document.getElementById("msg").innerHTML =
             hundreds[i] + " Hundred";

               }
            }
          }
      </script>
```

```
    </head>
    <body>
        <fieldset>
            <legend>Enter Amount</legend>
            <input type="text" onkeyup="Convert()"
        id="txtAmount" size="6">
            <p>In Words : <span id="msg"></span> </p>
        </fieldset>
    </body>
</html>
```

Iterators

- Iterator is a software design pattern used to read elements from a collection in sequential order.
- It doesn't require initialization, condition and counter.
- Iterators can be defined with

for..in	to read all properties from collection
for..of	to all values from collection

Syntax:-

```
for(var property in collection/object)
{
}
for(var value of collection)
{
}
```

Jump Statements

- continue
- break

- return

Continue: It skips the current counter and continues to next counter.

Ex:-

```
    <script>
       function f1(){
    var products = [
       {Name:"TV", Category:"Electronics"},
       {Name:"Nike", Category:"Footwear"},
       {Name:"Mobile", Category:"Electronics"},
       {Name:"Shirt", Category:"Fashion"}
    ];
    for(var i=0; i<products.length; i++){
       if(products[i].Category=="Electronics" || products[i]
       .Category=="Footwear"){
          continue;
       }
       document.write(products[i].Name + "<br>");
    }
  }
  f1();
</script>
```

Exception Handling Statements in JavaScript

- In computer programming we have to handle 2 types of errors
    - a) Compile Time Errors
    - b) Runtime Errors

Compile Time Errors

* These are the errors related to syntax declaration.
* Application fails to compile.
* You can't view the output.

Runtime Errors

* These are the errors identified during runtime of application.
* Application successfully compiles.
* Application starts working.
* Application faces catestrophic failures.
* This leads to abnormal termination of application.

```
var x  = 10;
var y = 0;
var z = x / y;        compile time error
```

FAQ: What is the purpose of Exception Handling?

Ans: It is used to avoid abnormal termination.

- JavaScript exception handling statements are
    a) try
    b) catch
    c) throw
    d) finally

try : It is monitoring block. It contains the statements to execute.

catch : It is handler block. It will catch the exeception thrown by application.

throw : It explicitly throws exception & message.

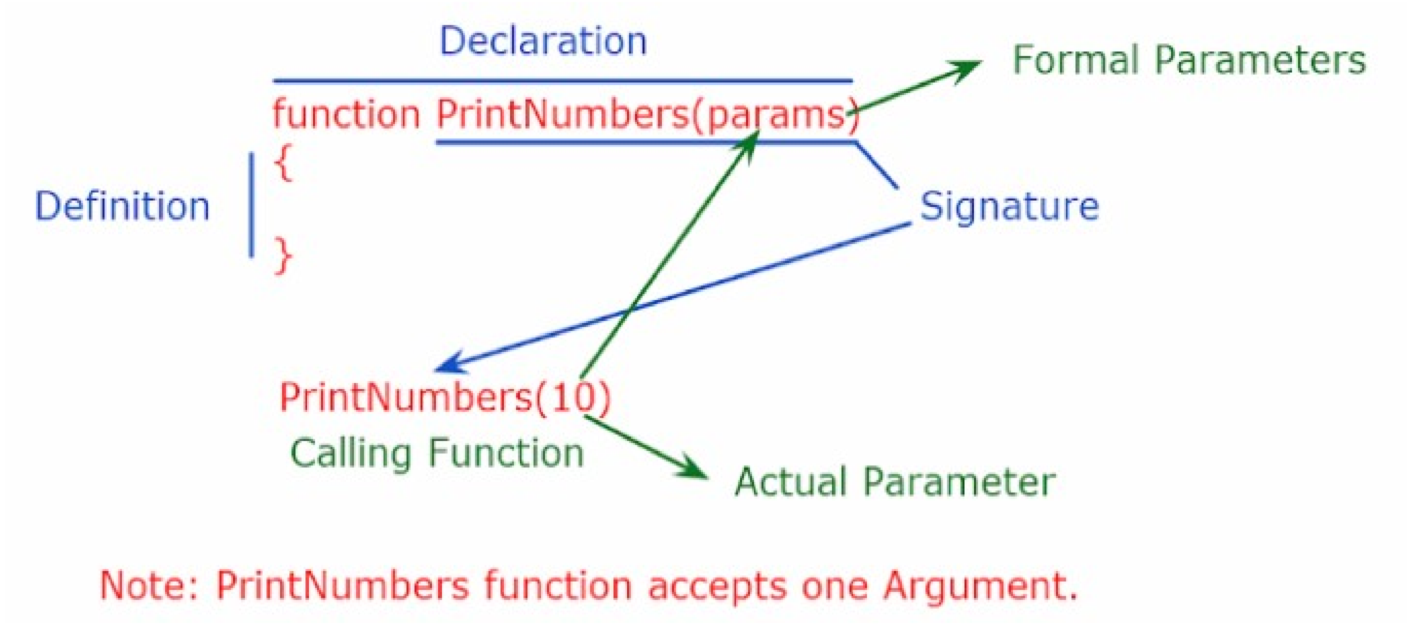
finally : It comprises of statements to execute in all situations.

Ex:-

```
      <script>
         function f1(){
      try
      {
         var x = 10;
         var y = -1;
         var z = x / y;
         if(y==0) {
            throw "DivideByZeroError: Can't Divide By Zero <br>";
         }
         if(y<0) {
            throw "NoNegativeValue: You Can't Divide by -ve
      Number<br>";
         }
      }
      catch(exceptionDetails)
      {
        document.write(exceptionDetails);
      }
      finally
      {
         document.write(`Division=${z}`);
      }
   }
   f1();
</script>
```

JavaScript Functions

- A function is used for "Refactor".
- Refactor is a technique used to extract set of statements into a function.
- It allows encapsulation and reusability.
- Function configuration comprises of following



FAQ: What is Formal Parameter?

Ans: It is the parameter defined in function declaration.

FAQ: What is Actual Parameter?

Ans: It is the parameter defined into function while calling the function.

FAQ: What are Arguments?

Ans: Arguments define how many parameters are configured in function.

FAQ: How a function is called?

Ans : By using its signature.
What a function can do, is defined in definition.

Syntax:-

```
function  PrintNumbers()
{
  for(var i=1; i<=10; i++)
  {
   document.write(i + "<br>");
  }
}
```

Function Parameters

- Function can be parameter less or parameterized.
- Function requires parameter in order to modify the functionality.
- Every parameter defined in function is mandatory.

Ex:-

```
<script>
        function PrintNumbers(totalCount) {
        for (var i = 1; i <= totalCount; i++) {
            document.write(i + "<br>");
          }
         }
         PrintNumbers(5);
         PrintNumbers(10);
</script>
```

- Function can have multiple parameters.
- Function parameters have order dependency.

Ex:-

```
<script>
    function PrintNumbers(startNumber, endNumber) {
         for (var i = startNumber; i <= endNumber; i++) {
            document.write(i + "<br>");
         }
        }
        PrintNumbers(22,28);
      </script>
```

- Function can have any type of parameter. Both primitive and non-primitive.

Ex:-

```
      <script>
   function PrintProduct(id, name, price, stock, cities, rating)
   {
      document.write(`
         Product Id : ${id} <br>
         Name: ${name} <br>
         Price: ${price} <br>
         Stock : ${(stock==true)?"Available":"Out of Stock"} <br>
         Shipped To: ${cities.toString()} <br>
         Rating : ${rating.rate} <br>
         From : ${rating.count}
      `);
   }
   PrintProduct(101, "Samsung TV", 56000.55, true, ['Delhi',
      'Hyd'],{rate:4.3, count:5600});
</script>
```

Function as Parameter

- A function can use functions are parameters to handle call back mechanism.

- Call back is a technique where function is executed according to state and situation.

Ex:-

```
      <script>
         function Login(userid, password, success, failure)
    {
       if(userid=="john" && password=="john11")
       {
          success();
       } else {
          failure();
       }
    }
    Login('john11', 'john11', function(){document.write('Login
Success')}, function(){document.write('Invalid Credentials')});
</script>
```

FAQ: How many parameters are allowed in a function?

Ans: There is no limit for parameter, but as per ECMA standards max 1024 parameters are recommened.

Rest Parameters

- JavaScript ES5 and higher versions support "Rest" parameter.
- A single rest parameter can allow multiple arguments.
- A Rest parameter is defined by using "...paramName".

Syntax:-

```
      function  Name(...restParam)
      {
      }
      Name(arg1, arg2, arg3, arg4..);
```

- Every function can have only one rest parameter.
- Rest parameter must be the last parameter in formal list.

Syntax:-

```
function Name(...rest1, ...rest2)          // invalid.
function Name(...rest1,  id);              // invalid.
function Name(id, ...rest1);               // valid
function Name(...rest1);                   // valid
```

Ex:-

```
  <script>
function Login(msg,...loginDetails)
{
   const [userid, password, success, failure] = loginDetails;
   document.write(msg);
   if(userid=="john" && password=="john11")
   {
      success();
   } else {
      failure();
   }

}
Login("<h2>Rest Parameters</h2>",'john', 'john11',
function(){document.write('Login Success')},
function(){document.write('Invalid Credentials')});
</script>
```

FAQ: Why rest parameter must be last parameter?
Ans: As it have to read values upto end.

FAQ: Why function can't have multiple rest parameters?
Ans: As one rest parameter will read upto end, you don't need multiple rest parameters and function will not allow multiple.

Function with Return Type Defined

- Function usually discards the return type, hence memory is only allocated to perform funcitonality.
- If a function is defined with "return" along with functionality, then it can allocate memory to store the result returned by function.

Syntax:-

```
function  Name()
{
   return  operation;
}
```

Ex:-

```
<script>
     function Add(a ,  b)
     {
        return  a  + b;
     }
     document.write(`Type of Add : ${typeof Add(10,20)}
       <br>Addition=${Add(10,20)}`);
</script>
```

- If return is not using any operation then it is used to terminate the execution.
- Any statement defined after retrun statement, is not reachable to compiler.

Ex:-

```
<script>
      function PrintDetails()
      {
          document.write("Statement1");
          return;
          document.write("Statement2");

          document.write("Statement3");
      }
      PrintDetails();
</script>
```

FAQ: Can we define return keyword in a function that doesn't return value?

Ans: Yes. It is used as Jump statement.

Syntax:-

```
function f1(a, b)
{
  return
  a + b;              // un-reachable code
}
f1() =  undefined
```

Function Recurrsion

- It is the technique of calling a function with in the context of current function.
- It is used for batch operations.

Ex:-

```
<script>
  function Factorial(n)
  {
    if(n==0){
      return 1;
    }
    return n * Factorial(n-1);
  }
  document.write(`Factorial of 5 is ${Factorial(5)}`);
</script>
```

Function Closure

- It is a technique of configuring a function inside another function.
- The members of outer function are accessible to inner function.
- The members of inner function are not directly accessible to outer function.
- Closure technique allows the outer function to access the members of inner functions.

Syntax:-

```
function outer()
{
  function inner()
```

```
         {
           return value;
         }
         return inner();
       }
       outer();
```

Ex:-

```
      <script>
         function Outer()
         {
       var msg = "Message from Outer Function";
       function Inner(){
           var outerMessage = msg;
           var innerMessage = "Message from Inner Function";
           return outerMessage + "<br>" + innerMessage;
       }
       return Inner();
   }
   document.write(Outer());
</script>
```

Anonymous Function

- It is a function without name.
- Anonymous functions are used in callbacks.

```
       array = ['TV', true, function(){}]
       array[2]();

       function Login(password, success, failure)
       {
       }
       Login('1234], function(){}, function(){});
```

Ex:-

```
        <script>
            (function()
            {
                document.write('Hello ! JavaScript');
            })()
        </script>
```

- Callback is a technique of accessing function and loading into memory according to state and situation.

Ex:-

```
<script>
    var hello = function(){
        document.write('Hello ! World');
    }
    hello();
</script>
```

Arrow Functions

- Arrow functions are used to define a function in short hand technique.
- You can minify the function declaration and definition.

( )	:	configure function with parameters
=>	:	configure return value
{ }	:	configure set of statements

Syntax:-

```
function hello(msg)
{
   return  `Hello ! ${msg}`;
}
var hello = msg => `Hello ! ${msg}`;              // short hand
```

Syntax:-

```
function welcome()
{
  document.write("Welcome");
}
var welcome = () => document.write("Welcome");   //
short hand
```

Syntax:-

```
function add(a, b)
{
 return  a + b;
}
var  add = (a,b) => a + b;
```

Syntax:-

```
function print()
{
  document.write("statement1");
  document.write("statement2");
}
var print = () => {
                 document.write("statement1");
                 document.write("statement2");
```

```
                    }
```

Ex:-

```
      <script>
        var add = (a,b) => a + b;
        var print = () => document.write("Print Method");
        document.write(add(10,20) + "<br>");
        print();
      </script>
```

Ex:-

```
      <script>
         var products = [
            {Name: 'TV', Category: 'Electronics'},
            {Name: 'Shoe', Category: 'Footwear'},
            {Name: 'Mobile', Category: 'Electronics'}
         ];
         var electronics = products.filter( item=>
      item.Category=='Electronics');
         for(var item of electronics){
      document.write(item.Name + "<br>");
         }
      </script>
```

Ex:-

```
      <script>
   var products = [
      {Name: 'TV', Category: 'Electronics'},
      {Name: 'Shoe', Category: 'Footwear'},
      {Name: 'Mobile', Category: 'Electronics'}
   ];
```

```
    document.write("Total Count of Electronics :" +
       products.filter(product=>
       product.Category=='Electronics').length);
</script>
```

Ex:- Events

```
      <!DOCTYPE html>
      <html>
      <head>
      <title>Events</title>
      <script>
         var InsertClick = () => document.write(`RecordInserted`);
      </script>
   </head>
   <body>
      <button id="btnInsert" onclick="InsertClick()">Insert
      </button>
   </body>
</html>
```

JavaScript OOP

- In real-time development various programming systems are used, like
    - a) POPS
    - b) OBPS
    - c) OOPS

POPS :

- Process Oriented Programming System.
- It supports low level features.
- It can directly interact with hardware services.

- It uses less memory.
- It is faster in communication.

Ex:- C, Pascal, COBOL etc..

- Code reusability issues
- Code separation issues
- No dynamic memory allocations
- Code security issues

OBPS:

- Object Based Programming System
- Supports code separation
- Supports code reusability
- Dynamic memory allocations
- Supports extensions

Ex:- JavaScript, Visual Basic etc..

- Code level security issues
- No contracts
- No templates
- No dynamic polymorphism

OOPS:

- Object Oriented Programming System
- Supports contracts, template, dynamic polymorphism
- Support code security

Ex: TypeScript, C++, Java, .NET Languages etc..

- They are tidious
- They use more memory
- They are complex in configuration

- They are slow

FAQ: Is JavaScript OOP Language?
Ans : No. It supports certain features of OOP.

Evoution of OOP:

- "Alan Kay" introduced Object into computer programming in early 1960's.
- "Johan Oly, Kristian Nygaard", introduced OOP in early 1967.
- The first OOP language is "SIMULA 67".
- Code reusability.
- "Trygve" introduced code separation in early 1970's by using a framework called "MVC". - Small Talk.

Java	- Spring
PHP	- Cake PHP, Code Igniter
Python	- Django, Fask
Ruby	- Ruby on Rails
.NET	- ASP.NET MVC
JavaScript	- SPINE, Angular, React, Vue

- 1970's C++
- 1990's Java
- 2000   .NET Languages

Features of OOPS:

- Code reusability
- Code separation
- Code Extensibility
- Code Security

Characterstics of OOPS:

- Inheritance
    - Encapsulation
    - Polymorphism
    - Abstraction

JavaScript Object Oriented Features:

Modules in JavaScript

- Module is a set of functions, values and classes.
- Modules are used to build library for application.
- You can import and implement the library in your project.

- Modules provide
    - reusability
    - mantainability
    - testability
    - extensibility
    - separation
- To use modules in a project you need a module system working in PC.
- There are various module systems like
    - Common JS
    - AMD [Asynchronous Module Distribution]
    - UMD [Universal Module Distribution]
- Node.js provides a module system called UMD.
- JavaScript can use the existing module system and implement in your application.

Steps:-

1. You have create a module, which is JavaScript file with set of functions, classes and varibles.
2. You have to mark the members with "export" , only exported members can be imported.

Syntax:-

```
export function name()
 {
}
```

3. You have to import the module and members of module in any page.

```
import { member }  from  'module_file_path';
```

4. You can implement the member at same location or in any another script source.

Note: The script type for module system must be defined a "module".

```
<script type="module"> </script>
```

- Every module can have a default member to export.

```
export default  function Name() { }
```

- The default member is directly imported

```
import  Name  from  'path';
import  {Name} from 'path';        // if not default.
```

- Every module can have only one default.

- You can't define directly constant values in a module if you want further implementation. Always use formal declarations.

Ex:-

1. Add a new folder "library"
2. Add a new file into folder

ProductModule.js

```
  export default function Title(msg){
return msg;
}

export function Details(name, price){
return {Name: name, Price: price};
}
```

3. Add a new HTML file

Project.html

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>Module System</title>
    <script type="module">
      import Title, {Details} from '../library/ProductModule.js';
      document.querySelector("h1").innerHTML = Title
        ("Product Details");
      document.querySelector("p").innerHTML = `Name=$
        {Details("Smasung TV", 45000.44).Name}<br>Price=$
        {Details("Samsung TV", 56000.42).Price}`;
    </script>
  </head>
  <body>
    <h1 align="center"></h1>
    <p></p>
  </body>
```

```
</html>
```

Class in OOP

- In computer programming class is a "Program Template".
- A class comprises of sample data and logic which you can implement and customize according to the requirements.
- Class have various behaviours
    a) Model
    b) Entity
    c) Blue Print
- Class is reffered as Model when it is representing data.
- Class is reffered as Entity when it is representing business requirements.
- Class is always a Blue Print as it contains set of data and logic, which you can implement at any location.

Configuring Class:

1. Class Declaration
2. Class Expression

- Class Declaration is used for classes that are requested according to requirement.

```
class  className
{
}
```

- Class Expression is used for classes that are loaded according state and situation.

```
var refName = class {  };
```

Class Members

- A class comprises of following members
    a) Properties
    b) Methods
    c) Accessors
    d) Constructor

FAQ: Can we define a Variable in class?

Ans : No.

FAQ: Can we define a function in class?

Ans : No.

FAQ: Why wen can't define variables and functions in class?

Ans: They are immutable. Class can contain only mutable members.

FAQ: What is difference between Variable and Property?

FAQ: What is difference between Function and Method?

Properties

- Properties are references in class memory where you can store data.
- Data is stored in properties.
- You can store any type of data in property. [primitive or non-primitive]

Syntax:-

```
class className
{
 Property = Value;
}
```

- In order to access members of a class, you have to create an instance for class.

```
    let  instance = new className;
```

- "new" is dynamic memory allocating operator.
- "constructor" is responsible for creating an object for class.

Ex:-

```
<script>
   class Product
   {
       Name = "Samsung TV";
       Price = 56000.66;
       Stock = true;
       ShippedTo = ["Delhi", "Hyd"];
       Rating = {Rate:3.4, Count:500};
   }
 let  tv  =  new Product;
 document.write(`Name=${tv.Name}<br>Price=${tv.Price}
< br>Rating=${tv.Rating.Rate}`);
</script>
```

- Properties are mutable, you can control the behaviour of property using "Accessors".

Accessors

- Accessors will provide a fine grained control over

properties.
- Accessors define the restrictions for reading and writing data into properties.
- Accessors are used for reading and writing data into properties, at the same time they can define restrictions.
- The accessors used in class are:

    a) get()   [Getter]
    b) set()   [Setter]

- get() accessor is used for reading values from a property.
- set() accessor is used for writing data into property.

Syntax:-

```
_propertyName;

get Property()
{
  return  this._propertyName;
}
set Property(newName)
{
   this._propertyName = newName;
}
```

Ex:-

```
<script>
    var username = prompt("Enter User Name");
    var password  = prompt("Enter Password");
    var productname = prompt("Enter Product Name");
    class Product
```

```
     {
        _productName;
        get ProductName()
        {
           return this._productName;
        }
        set ProductName(newName)
        {
           if(username=="john" && password=="john@11")
           {
              this._productName = newName;
           } else {
              document.write("Unauthorized : You are not
         authorized to set product name");
           }
        }
     }
     let obj = new Product;
     obj.ProductName = productname;
     if(obj.ProductName) {
        document.write(`Product Name :
  ${obj.ProductName}`);
     }
  </script>
```

Methods

- Methods define the actions to perform.
- All method specifications are similar to functions.

FAQ: What is difference between functions, methods and procedures?

Ans:

functions : always intended to return value.
methods : always intended to discard the return value.
procedure: changes according to situation. it can discard memory or memory according to return value.

Ex:-

```
<script>
        class Product
        {
                Name = "TV";
                Price = 56000;
                Qty = 3;
                Total(){
                return this.Qty * this.Price;
        }
        Print(){
          document.write(``);
        }
      }
        let obj = new Product;
         obj.Print();
  </script>
```

Constructor

- It is a special type of method in class, which is used for instantiation.
- Constructor is a sub-routine that executes automatically for every object.
- It initializes memory to store values at the time of allocating memory for class.

- In JavaScript constructor is anonymous [without name]
- Class name is used for constructor implicitly.
- It is defined by using "constructor" keyword.
- Every class have a default constructor.
- You can configure explicity constructor to define explicit actions.

Syntax:-

```
class ClassName
{
   constructor()
   {
   }
}

 let  obj = new ClassName;   constructor reffered implicitly.
 let  obj = new ClassName();   ( ) => refering to constructor.
```

FAQ: Is "()" mandatory beside class name?

Ans: No. It is required only when constructor is parameterized.

Ex:-

```
    <!DOCTYPE html>
    <html>
<head>
    <title>Constructor</title>
    <script>
        class Database
        {
            constructor(dbName){
                console.log(`Connected to ${dbName}`);
            }
            Insert(){
                document.write('Record Inserted');
            }
            Delete(){
                document.write('Record Deleted');
            }
        }
        let db = new Database("Oracle");
        function InsertClick(){

            db.Insert();
        }
        function DeleteClick(){

            db.Delete();
        }
    </script>
  </head>
  <body>
    <button onclick="InsertClick()">Insert</button>
    <button onclick="DeleteClick()">Delete</button>
  </body>
</html>
```

- JavaScript class can't overload constructor.
- JavaScript class can't have private constructor.
- JavaScript class can't have static constructor.
- A derived class constructor must have super call.

Note: According to the rules of OOP, If classes are configure with

relationship then a derived class constructor must call base class constructor.
- In other OOP language, it is done implicitly but in JavaScript you need an explicit calling.

Code Reusability

- It is one of the key feature of OOP.
- You can configure code reusability by using 2 techniques
    a) Aggregation
    b) Inheritance

Aggregation:

- It is a technique used to access the code of one class in another by using an instance of class.
- It is reffered as "Has-A-Relation".
- Without configuring any relation between classes, you can access the members of one class in another class.
- It is know as "Object-to-Object" communication.

Ex:-

```
    <script>
        class BaseClass
  {
     Print(){
         document.write(`Base Class Print Method<br>`);
     }
  }
  class Derived
  {
     Print(){
        let obj = new BaseClass();
        obj.Print();
        document.write(`Derived Class Print Method<br>`);
      }
    }
    let obj = new Derived;
    obj.Print();
</script>
```

Inheritance:

- In this technique a class and extend another class.
- You can configure relation between classes.
- Members of base class are accessible in derived class by using "super" keyword.
- Without creating instance of base class you can access its members using "super" keyword.
- Inheritance is code Extensibility.
- This is reffered as "Is-A-Relation".

Ex:-

```
    <script>
       class BaseClass
       {
     Print(){
        document.write(`Base Class Print Method<br>`);
     }
   }
   class Derived extends BaseClass
   {
     Print(){
        super.Print();
```

```
            document.write(`Derived Class Print Method<br>`);
        }
    }
    let obj = new Derived;
    obj.Print();
</script>
```

Ex:-

```
        <script>
            class BaseClass
            {
                constructor(uname){
                    document.write(`Base Class Constructor ${uname}
        <br>`);
        }
        Print(){
            document.write(`Base Class Print Method<br>`);
        }
    }
    class Derived extends BaseClass
    {
        constructor(){
            super('john');
            document.write(`Derived Class Constructor<br>`);
        }
        Print(){
            document.write(`Derived Class Print Method<br>`);
            super.Print();
        }
    }
    let obj = new Derived;
```

```
   obj.Print();
</script>
```

- Single Inheritance
- Multi Level Inheritance

Ex:-

```
      <script>
         class SBI_BankApp_Version1
   {
       CustomerModule = "Bank Customer Module for
Customers";
       AdminModule = "Bank Admin Module to Manage
Customers";
   }
   class SBI_BankApp_Version2 extends SBI_BankApp_Version1
   {
      CreditCardModule = "Bank Credit Card Payments";
   }
   class SBI_BankApp_Version3 extends SBI_BankApp_Version2
   {
      NIRModule = "Bank NRI Module for NRI's";
   }
   let windows10User = new SBI_BankApp_Version1();
</script>
```

Types of Inheritance

- Single Inheritance
  A Base class is extended by using derived class.
- Multi Level Inheritance

A Derived class is exteneded by another dervied class.

- Multiple Inheritance

    A single derived class implements multiple super classes.

    It is not supported for classes in OOP.

    It is supported for Interfaces in OOP.

    JavaScript don't have Interface.

Syntax:-

```
class Dervied extends  super1, super2         // invalid
{
}
```

FAQ: Why multiple inheritance is not supported for classes?

Ans: Because of Constructor Deadlock.

Deadlock is a situation where every constructor waits for each other.

- Interface in OOP supports multiple inheritance as it doesn't have constructor.

Polymorphism

- The term Poly means "Many" and Morphos means"Forms".
- It is the ability of a component to change its behaviour according to state and situation.
- Technically ploymorphism is a single base class object using the memory of multiple derived classes.
- In JavaScript a single object can work for various classes.

Ex:-

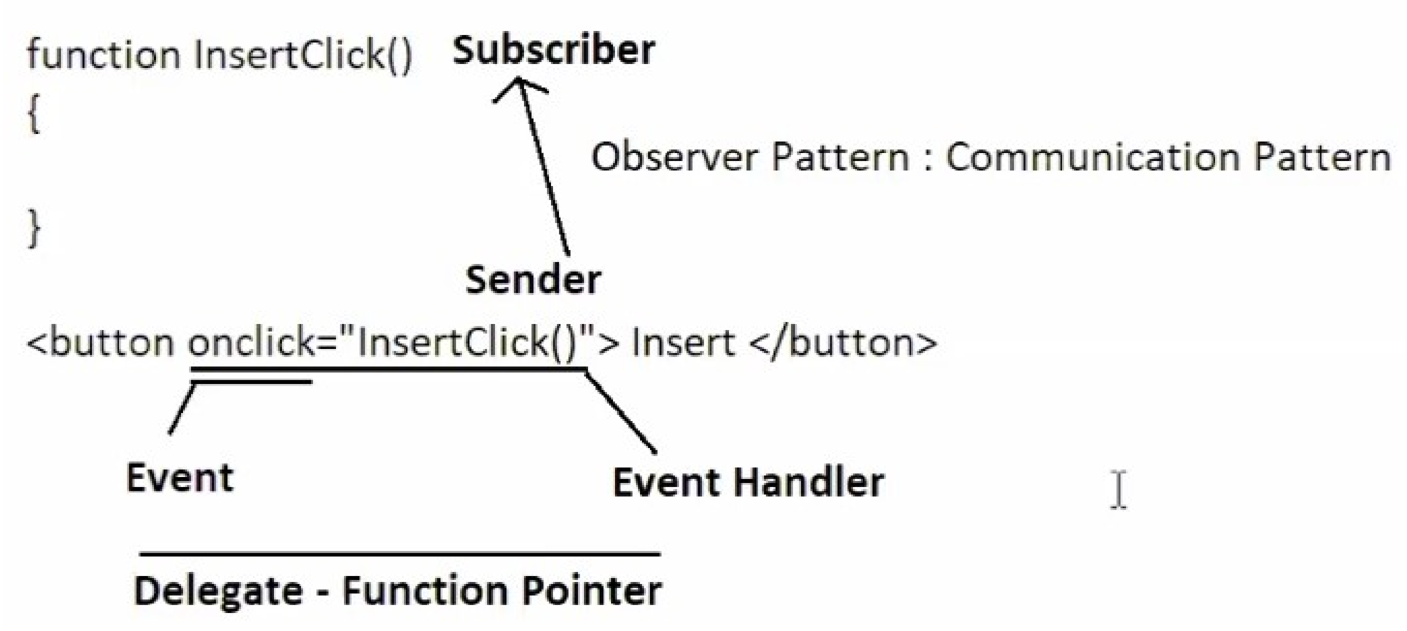
```
      <script>
   class Employee
   {
      FirstName;
      LastName;
      Designation;
      Print() {
         document.write(`${this.FirstName} ${this.LastName} -
${this.Designation}<br>`);
      }
   }
   class Developer extends Employee
   {
      FirstName = "Raj";
      LastName = "Kumar";
      Designation = "Developer";
      Role = "Developer Role : Build, Debug, Test, Deploy <br>";
      Print(){
         super.Print();
         document.write(this.Role);
      }
   }
   class Admin extends Employee
   {
      FirstName = "Kiran";
      LastName = "Kumar";
```

```
        Designation = "Admin";
        Role = "Admin Role : Authentication, Authorization <br>";
        Print(){
            super.Print();
            document.write(this.Role);
        }
    }
    class Manager extends Employee
    {
        FirstName = "Tom";
        LastName = "Hanks";
        Designation = "Manager";
        Role = "Manager Role : Approvals <br>";
        Print(){
            super.Print();
            document.write(this.Role);
        }
    }
    let employees = new Array(new Developer(), new Admin(), new
Manager());
    let designation = prompt("Enter Desigation");
    for(var employee of employees)
    {
        if(employee.Designation==designation) {
            employee.Print();
        }
    }
```

```
</script>
```

JavaScript Browser Events

- Event is a message sent by sender to its subscriber in order to notfiy the change.
- Event follows a software design pattern called "Observer".


function InsertClick()
{
}
Subscriber
Observer Pattern : Communication Pattern
Sender
<button onclick="InsertClick()"> Insert </button>
Event
Event Handler
Delegate - Function Pointer

- Sender sends notification
- Subscriber contains actions to perform.
- Event uses a delegate mechanism [Function Pointer]

onclick	=> Event
onclick="InsertClick()"	=> Event Handler

- Every event handler contains default arguments

a) this	: contains information about object [button]
b) event	: contains information about event [onclick]

- Object Information contains
  name, id, class, value etc..

- Event Information contains
  clientX, clientY, keyCode, charCode, which, etc..

Event Arguments

- Default Arguments [this, event]
- Custom Arguments

Ex:- Default Arguments

```
    <head>
  <script>
     function InsertClick(obj, e){
        if(e.ctrlKey){
           location.href = 'http://www.amazon.in&#39;;
        } else {
           document.write(`
          Button Id : ${obj.id} <br>
          Button Class: ${obj.className} <br>
          Clicked At X : ${e.clientX} <br>
          Ctrl Key : ${e.ctrlKey}
         `)
        }
     }
  </script>
</head>
<body>
  <button id="btnInsert" class="btn btn-primary"
onclick="InsertClick(this,event)">Insert - Ctrl+Click to Visit
```

```
Amazon</button>
</body>
```

Ex:- Custom Arguments

```
    <head>
  <script>
    function InsertClick(e, obj, msg){
      if(e.ctrlKey){
        location.href = 'http://www.amazon.in&#39;;
      } else {
        document.write(`
       Button Id : ${obj.id} <br>
       Button Class: ${obj.className} <br>
       Clicked At X : ${e.clientX} <br>
       Ctrl Key : ${e.ctrlKey} <br>
       Message  : ${msg}
      `)
      }
    }
  </script>
</head>
<body>
  <button id="btnInsert" class="btn btn-primary"
onclick="InsertClick(event,this,'Hello ! You Clicked Insert
Button')">Insert - Ctrl+Click to Visit Amazon</button>
</body>
```

Ex:-

```
      <head>
   <script>
        function DatabaseOperation(buttonName){
          switch(buttonName){
             case 'Insert':
             document.write(`Record Inserted`);
             break;
             case 'Update':
             document.write(`Record Updated`);
             break;
             case 'Delete':
             document.write(`Record Deleted..`);
             break;
          }
        }
   </script>
</head>
<body>
   <button name="Insert"
onclick="DatabaseOperation(this.name)">Insert</button>
   <button name="Update"
onclick="DatabaseOperation(this.name)">Update</button>
   <button name="Delete"
onclick="DatabaseOperation(this.name)">Delete</button>
   <select onchange="DatabaseOperation(this.value)">
      <option>Insert</option>
```

```
        <option>Update</option>
        <option>Delete</option>
    </select>
</body>
```

Various Categories of Browser Events

1. Keyboard Events
2. Mouse Events
3. Clipboard Events
4. Button Events
5. Timer Events
6. Touch Events etc..

Keyboard Events

- Handle interactions with regard to keys.
- They specify actions to perform while user is keying in the keys.

Event	Description
onkeyup	Actions when key is released
onkeydown	Actions when user hold down a key
onkeypress	Actions when user keys-in another key.

Event Properties

- keyCode	code of every key ASCII [A=65 to Z=90]
- charCode	code as per UTF standards [only char keys]
- which	similar to keyCode [Enchanced Keyboard]
- shiftKey	]
- ctrlKey	] return true or false
- altKey	]

Syntax:-

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Demo</title>
        <script>
            function GetCode(e){
                document.querySelector("p").innerHTML = `
                   KeyCode  : ${e.keyCode} <br>
                   charCode : ${e.charCode} <br>
                   which    : ${e.which}  `;
            }
        </script>
    </head>
    <body>
        <input type="text" id="txtName"
onkeypress="GetCode(event)">
        <p></p>
    </body>
</html>
```

Ex:-

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Demo</title>
        <link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
        <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
```

```
    <script>
        var userdetails = [
            {UserName: 'john'},
            {UserName: 'john12'},
            {UserName: 'david'}
        ];
        function VerifyUser(){
            var username =
document.getElementById("txtName").value;
            var msg = document.getElementById("msg");
            for(var user of userdetails)
            {
                if(user.UserName==username) {
                    msg.innerHTML = 'User Name Taken - Try Another';
                    msg.style.color = 'red';
                    break;
                } else {
                    msg.innerHTML = 'User Name Available';
                    msg.style.color = 'green';
                }
            }
        }
        function CapsWarning(e) {
            if(e.keyCode>=65 && e.keyCode<=90){

document.getElementById("capsWarning").style.display =
"block";
            } else {

document.getElementById("capsWarning").style.display =
"none";
```

```
            }
          }
        </script>
    </head>
    <body class="container-fluid">
        <fieldset>
            <legend>Register</legend>
            <dl>
                <dt>User Name</dt>
                <dd>
                    <input type="text" id="txtName"
onkeyup="VerifyUser()">
                    <div id="msg"></div>
                </dd>
                <dt>Password</dt>
                <dd>
                    <input type="password" id="txtPwd"
onkeypress="CapsWarning(event)">
                    <div class="text-warning" id="capsWarning"
style="display: none;">
                       <span class="bi bi-exclamation-triangle"></span>
Warning : CAPS ON
                    </div>
                </dd>
            </dl>

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

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>Demo</title>
      <link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootstrap.css">
      <script>
         var userdetails = [
            {UserName: 'john'},
            {UserName: 'john12'},
            {UserName: 'david'}
         ];
         function VerifyUser(){
            var username =
document.getElementById("txtName").value;
            var msg = document.getElementById("msg");
            for(var user of userdetails)
            {
               if(user.UserName==username) {
                  msg.innerHTML = 'User Name Taken - Try Another';
                  msg.style.color = 'red';
                  break;
               } else {
                  msg.innerHTML = 'User Name Available';
                  msg.style.color = 'green';
               }
            }
         }
         function ShowStatus(min, max, value) {
            var meter = document.getElementById("meter");
            meter.min = min;
            meter.max = max;
            meter.value = value;
        }

        function VerifyPassword() {
            var password =
document.getElementById("txtPwd").value;
            var regExp = /(?=.*[A-Z])\w{4,10}/;
            var passwordMsg =
document.getElementById("passwordMsg");

            if(password.match(regExp)) {
                passwordMsg.innerHTML = "Strong Password";
                ShowStatus(1,100,100);
            } else {
                if(password.length<4) {
                    passwordMsg.innerHTML = "Poor Password";
                    ShowStatus(1,100,20);
                } else {
                    passwordMsg.innerHTML = "Weak Password";
                    ShowStatus(1,100,60);
                }
            }
        }
    </script>
  </head>
  <body class="container-fluid">
    <fieldset>
        <legend>Register</legend>
```

```
          <dl class="w-25">
             <dt>User Name</dt>
             <dd>
                <input type="text" id="txtName"
onkeyup="VerifyUser()">
                <div id="msg"></div>
             </dd>
             <dt>Password</dt>
             <dd>
                <input type="password" id="txtPwd"
onkeyup="VerifyPassword()">
                <div>
                   <meter id="meter" style="width: 100%; height:
20px;" min="1" max="100"></meter>
                </div>
                <div id="passwordMsg">
                </div>
             </dd>
          </dl>
       </fieldset>
   </body>
</html>
```

Mouse Events

- Specifies actions to perform with regard to mouse interactions.

Event	Description
onmouseover	Actions when pointer is over element
onmouseout	Actions when pointer move out of element

onmousedown	Actions when mouse button is down
onmouseup	Actions when mouse button is released
onmousemove	Actions while mouse pointer is moving over element.

Event Properties

- clientX	Gets x position
- clientY	Gets y position

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>Mouse</title>
    <script>
      var images =
["../public/images/mobile.PNG","../public/images/shoe.jpg","../p
ublic/images/shoe1.jpg","../public/images/neckband.PNG"]
      function bodyload(){
        for(var image of images)
        {
          var img = document.createElement("img");
          img.src = image;

document.getElementById("gallery").appendChild(img);
        }
      }
    </script>
    <style>
      body {
        display: flex;
```

```
            align-items: center;
            height: 500px;
        }
        #gallery img {
            height: 150px;
            width: 150px;
            z-index: -1;
            opacity: 0.4;
            transition: 2s;
        }
        #gallery img:hover {
            transform: scale(2);
            transition: 2s;
            z-index: 1;
            opacity: 1;
        }
        marquee {
            height: 300px;
        }
    </style>
  </head>
  <body onload="bodyload()">
    <marquee  onmouseover="this.stop()"
onmouseout="this.start()">
        <div id="gallery">

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

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>Mouse</title>
      <script>
         function GetPosition(e){
            var flag = document.getElementById("flag");
            flag.style.position = "fixed";
            flag.style.top = e.clientY + "px";
            flag.style.left = e.clientX + "px";
            document.querySelector("h3").innerHTML =
`X=${e.clientX} <br> Y=${e.clientY}`;
         }
      </script>
   </head>
   <body onmousemove="GetPosition(event)">
      <div style="height: 1000px; display: flex;">
         <h3></h3>
      </div>
      <img id="flag" src="../public/images/flag.gif" width="50"
height="50">
   </body>
</html>
```

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>Mouse</title>
      <script>
```

```
          function ShowOffer(){

document.getElementById("pic").src="../public/images/speakero
ffer.png";
          }
          function HideOffer(){

document.getElementById("pic").src="../public/images/offer.png
";
          }
       </script>
   </head>
   <body>
       <p>Hold Down Mouse Button on Offer</p>
       <img onmousedown="ShowOffer()"
onmouseup="HideOffer()" src="../public/images/offer.png"
id="pic">

   </body>
</html>
```

Browser Events

- Keyboard
- Mouse
- Clipboard Events

oncut	removed and kept in clipboard
oncopy	copied to clipboard
onpaste	inserted from clipboard

Ex:-

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Events</title>
        <script>
            function Cut(){
                document.querySelector("h2").innerHTML = "Removed
: Copied to Clipboard";
            }
            function Copy(){
                document.querySelector("h2").innerHTML = "Copied :
Content copied to Clipboard";
            }
            function Paste(){
                document.querySelector("h2").innerHTML = "Inserted :
Content inserted from Clipboard";
            }
        </script>
    </head>
    <body>
        <fieldset>
            <legend>Your comments</legend>
            <textarea oncut="Cut()" oncopy="Copy()"
onpaste="Paste()" rows="4" cols="30">

            </textarea>
        </fieldset>
    <h2 align="center"></h2>
  </body>
</html>
```

FAQ: How to disable right click, cut, copy and paste like operations in page?

Ans: By disabling the event handler. You can disable event handler by returning false.

"oncut=return false"

onconextmenu	for right click
onselect	for selection
onselectstart	for extending selector

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>Events</title>
    <script>
      document.oncontextmenu = function(){
        alert('Right Click Disabled on this page.');
        return false;
      }
      document.onselectstart = function(){
        return false;
      }
    </script>
```

```
   </head>
   <body oncopy="return false">
       <h2>Page is secure - You can't right click, cut, copy or
select.</h2>
   </body>
</html>
```

Button Events:

onclick	single click
ondblclick	double click
oncontextmenu	right click

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>Events</title>
      <script>
         function OpenImage(){

window.open('../public/images/shoe.jpg','nike','width=400
height=300');
         }
      </script>
   </head>
   <body>
      <img ondblclick="OpenImage()"
```

```
src="../public/images/shoe.jpg" width="100" height="100">
        <p>double click to view large</p>
    </body>
</html>
```

FAQ: How to handle click event for <a> element?

Ans:

```
        <a onclick="function()">                    // invalid
        <a href="javascript:function()">            // valid
```

Ex:-

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Events</title>
        <script>
            function OpenImage(){

window.open('../public/images/shoe.jpg','nike','width=400
height=300');
            }
        </script>
    </head>
    <body>
        <img ondblclick="OpenImage()"
src="../public/images/shoe.jpg" width="100" height="100">
        <p>double click to view large</p>
    <a href="javascript:OpenImage()">Click</a>
  </body>
</html>
```

Form Events:

onsubmit	: defines actions when form is submitted
onreset	: defines actions when form resets.

Note: These are the events written for <form> element, but they require a submit and reset button in form.

FAQ: How a form can be submitted on any another element event other than submit button?

Ans. By using "form.submit()" method.

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>Events</title>
    <script>
      function SubmitClick(){
        alert('Data will be submitted to API');
       location.href="http://fakestoreapi.com/products&quot;
      }
      function CancelClick(){
        alert('Form will reset');
```

```
          }
          function CityChanged(){
             frmRegister.submit();
          }
       </script>
    </head>
    <body>
       <form name="frmRegister" id="frmRegister"
onsubmit="SubmitClick()" onreset="CancelClick()" >
          <dl>
             <dt>User Name</dt>
             <dd><input type="text" name="username"></dd>
             <dt>Age</dt>
             <dd><input type="number" name="age"></dd>
             <dt>City</dt>
             <dd>
                <select name="city" onchange="CityChanged()">
                   <option>Delhi</option>
                   <option>Hyd</option>
                </select>
             </dd>
          </dl>
          <button type="submit">Submit</button>
          <button type="reset">Cancel</button>
       </form>
    </body>
</html>
```

FAQ: Can a form have multiple submit buttons?

Ans: Yes.

FAQ: Why you need multiple submit buttons? As all submit buttons do the same work.

Ans: You can submit data to various API's. or You can handle multiple functionalities.

FAQ: Form can have only on "onsubmit" then how it will manage various functionalities?

Ans: By accessing submit button value.

FAQ: Can we have multiple form in a page?

Ans: Yes.

Focus Events

onfocus	: when element gets focus
onblur	: when element lost focus

Ex:-

```
    <!DOCTYPE html>
    <html>
  <head>
    <title>Events</title>
    <script>
      function Focus(){
              document.getElementById("msg").innerHTML = "Name
in Block Letters Only";
          }
          function Blur(){
              var username =
document.getElementById("txtName").value;
              document.getElementById("txtName").value =
username.toUpperCase();
              document.getElementById("msg").innerHTML = "";
          }
       </script>
   </head>
   <body>
      <fieldset>
         <legend>User Name</legend>
         <input type="text" id="txtName" onfocus="Focus()"
onblur="Blur()">
         <div id="msg"></div>
      </fieldset>
   </body>
</html>
```

Miscellaneous Events:

onchange	: when value changes
onselect	: when selected

Touch Events

```
        ontouchstart
        ontouchend
        ontouchmove
```

Ex:-

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Events</title>
        <script>
            function GetImageId(id){
                switch(id){
                    case 'nike':
                    document.querySelector("p").innerHTML = "Nike
Casuals - 5600.55";
                    break;
                    case 'boat':
                    document.querySelector("p").innerHTML = "boAt
Neckband - 6600.55";
                    break;
                }
            }
        </script>
    </head>
    <body>
        <img width="100" height="100"
ontouchstart="GetImageId(this.id)"
```

```
src="../public/images/shoe.jpg" id="nike">
        <img width="100" height="100"
 ontouchstart="GetImageId(this.id)"
src="../public/images/neckband.PNG" id="boat">
        <p></p>
    </body>
</html>
```

Timer Events:

```
        setInterval()
        clearInterval()
        setTimeout()
        clearTimeout()
```

setTimeout : It makes any task inactive for specific duration of time and invokes after the time interval.
- It can execute the given task only once for every object.

Syntax:- `setTimeout(functionName, interval);`

clearTimeout : It clears the task from memory and will not allow to execute

Syntax:- `clearTimeout(referenceOfFunction);`

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>Events</title>
      <script>
         function msg1(){
            document.querySelector("h2").innerHTML = "Hello ! - 
msg1";
         }
         function msg2(){
            document.querySelector("h2").innerHTML = "How are 
your? - msg2";
         }
         function msg3(){
            document.querySelector("h2").innerHTML = "Welcome 
to JavaScript - msg3";
         }
         var m1, m2, m3;
         function bodyload(){
           m1 =  setTimeout(msg1, 4000);
           m2 = setTimeout(msg2, 6000);
           m3 =  setTimeout(msg3,10000);
         }
         function Cancel2(){
            clearTimeout(m2);
         }
      </script>
```

```
    </head>
    <body onload="bodyload()">
        <h2 align="center"></h2>
        <div align="center">
            <button onclick="Cancel2()">Cancel Message 2</button>
        </div>
    </body>
</html>
```

setInterval : It preforms the specfied task at regular time intervals
- It is kept in the memory and sent to process.
- It is not removed from memory, hence it repeats until cleared from memory.

Syntax:- `setInterval(functionName, interval);`

clearInterval : It clears the task from memory.

Syntax:- `clearInterval(functionReferenceName);`

Ex:-

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Events</title>
        <link rel="stylesheet"
```

```
href="../node_modules/bootstrap/dist/css/bootstrap.css">
    <link rel="stylesheet" href="../node_modules/bootstrap-
icons/font/bootstrap-icons.css">
    <script>
      var products = [];
      var index = 0;
      function GetProduct() {

        document.getElementById("productTitle").innerHTML =
products[index].title;
        document.getElementById("productImage").src =
products[index].image;
        index++;
      }
      function bodyload(){
        fetch("http://fakestoreapi.com/products&quot;)
        .then(response=> response.json())
        .then(data=>{
          products = data;
          GetProduct(0)
        })
      }
      var slideshow;
      function PlayClick(){
        slideshow = setInterval(GetProduct,3000);
        document.querySelector("#status").innerHTML = "Slide
Show - Started";
```

```
          }
          function PauseClick(){
             clearInterval(slideshow);
             document.querySelector("#status").innerHTML = "Slide
Show - Paused";
          }
       </script>
    </head>
    <body class="container-fluid" onload="bodyload()">
      <div class="card p-3">
        <div class="card-header text-center">
          <p id="status"></p>
          <p id="productTitle"></p>
        </div>
        <div class="card-body text-center">
          <img id="productImage" width="80%" height="300">
        </div>
        <div class="card-footer text-center">
          <button class="btn btn-success" onclick="PlayClick()">
             <span class="bi bi-play"></span>
          </button>
          <button class="btn btn-danger" onclick="PauseClick()">
             <span class="bi bi-pause"></span>
          </button>
        </div>
      </div>
    </body>
```

</html>

Browser Objects

- JavaScript BOM is Browser Object Model which provides a set of objects to control browser
    a) window
    b) location
    c) navigator
    d) history
    e) document [DOM]

window:

- It provides properties and methods that are used to control browser window.
    a) open()
    b) close()
    c) print()

Syntax:-

```
<button  onclick="window.close()">
<button  onclick="window.print()">
<button onclick="window.open('path','title', 'features')">
```

location:

- It provides the properties and methods to control browser location details

host	: returns server name or IP address

port	: returns the port number
protocol	: return the protocol
href	: gets and sets url dynamically.
hash	: gets the current reference "id".
search	: gets the query string.
pathname	: gets the current file path

Syntax:- location.host

location.reload() : refresh the page

Ex:-

```
<!DOCTYPE html>
<html>
<head>
  <title>Location</title>
  <style>
    dt {
      background-color: gray;
      color:white;
    }
  </style>
  <script>
    function GetLocation(){
      document.getElementById("host").innerHTML =
location.host;
      document.getElementById("protocol").innerHTML =
(location.protocol=='http:')?`You are accessing
${location.protocol} from Web Server`:"You are accessign from
File Server";
            document.getElementById("port").innerHTML =
location.port;
            document.getElementById("path").innerHTML =
location.pathname;
            document.getElementById("href").innerHTML =
location.href;
         }
      </script>
   </head>
   <body>
      <fieldset>
         <legend><button onclick="GetLocation()">Get Location
Details</button></legend>
         <dl>
            <dt>Server Name / IP Address</dt>
            <dd id="host"></dd>
            <dt>Protocol</dt>
            <dd id="protocol"></dd>
            <dt>Port</dt>
            <dd id="port"></dd>
            <dt>URL</dt>
            <dd id="href"></dd>
            <dt>Path</dt>
            <dd id="path"></dd>
         </dl>
```

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

Ex:- Location Search

```
        search.html
        <!DOCTYPE html>
        <html>
    <head>
        <title>Search</title>
    </head>
    <body>
        <form method="get" action="results.html">
            <div align="center">
                <h1>Google</h1>
                <input type="text" name="search" size="40"
placeholder="Your search string">
                <p>
                    <button>Search</button>
                </p>
            </div>
        </form>
    </body>
</html>
```

Ex:- results.html

```
        <!DOCTYPE html>
        <html>
    <head>
        <title>Results</title>
        <script>
            var topics = [
                {title: "Oracle", Details: "It is a database.."},
                {title: "Html", Details: "It is a markup language"}
            ];
            function bodyload(){
                var term  =
location.search.substring(location.search.indexOf("=")+1);
                document.querySelector("p").innerHTML = `
                    Topic : ${term} <br>
                    Details:
${topics.find(item=>item.title==term).Details}`;
            }
        </script>
    </head>
    <body onload="bodyload()">
        <h2>Results</h2>
        <p></p>
    </body>
</html>
```

FAQ: How to access the current location details?

Ans:

href	complete URL

path	current file path
hash	current reference ID
search	query string

location.hash : It is used to access the current location by using "id" reference.
- It can access a named location within the page.

Ex:-

```
      <!DOCTYPE html>
      <html>
   <head>
      <title>Tutorial</title>
      <script>
         function GetTopic(){
            var topic = location.hash;
            switch(topic)
            {
               case "#html":
               var now = new Date();
               console.log(`You Recently Viewed HTML Tutorial :
${now.toString()}`);
               break;
               case "#css":
               var now = new Date();
               console.log(`You Recently Viewed CSS Examples :
${now.toString()}`);
```

```
                break;

            }
        }
    </script>
</head>
<body>
    <button onclick="GetTopic()">Get Previous Topic</button>
    <div>
        <a href="#html">HTML</a>
        <span>|</span>
        <a href="#css">CSS</a>
    </div>
    <h2 id="html">HTML</h2>
    <p>Your use of this software is subject to the terms and
conditions of the license agreement by which you acquired this
software.  If you are a volume license customer, use of this
software is subject to your volume license agreement.  You may
not use this software if you have not validly acquired a license
for the software from Microsoft or its licensed distributors.</p>

    <h2 id="css">CSS</h2>
    <p>Your use of this software is subject to the terms and
conditions of the license agreement by which you acquired this
software.  If you are a volume license customer, use of this
software is subject to your volume license agreement.  You may
```

not use this software if you have not validly acquired a license for the software from Microsoft or its licensed distributors.</p>

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

Navigator Object

- It is used to access the current browser details.
- It includes
    - Browser Family
    - Version
    - Plugin's
    - Mime Types
    - Geo Location etc..
- It uses the properties
    - appName
    - language
    - platform
    - plugins[]
    - mimeTypes[]
    - userAgent etc..
    - cookieEnabled

Ex:- General Browser Details

```
      <!DOCTYPE html>
      <html>
   <head>
```

```
        <title>Navigator</title>
        <script>
            function GetDetails(){
                document.querySelector("p").innerHTML= `
                   Browser Family : ${navigator.appName} <br>
                   Language  : ${navigator.language} <br>
                   Platform  : ${navigator.platform} <br>
                   Cookie Enabled :
${(navigator.cookieEnabled)==true?"Yes":"No-Please Enabled
Cookies on Your Browser"}
                `;
            }
        </script>
    </head>
    <body>
        <button onclick="GetDetails()">Get Details</button>
        <p></p>
    </body>
</html>
```

Ex:- To Get all plugins

```
        <script>
            function f1(){
                for(var item of navigator.plugins)
                {
                    document.write(item.name + "<br>");
                }
            }
            f1();
        </script>
```

Ex:- To Check any specific plugin

```
    <script>
        function f1(){
    if(navigator.plugins['PDF Viewer']==undefined){
        alert("You Browser is not enabled for PDF - Please
Install");
        location.href="http://www.adobe.com/downloads&quot;
    } else {
        document.write("You can watch PDF documents");
    }
  }
  f1();
</script>
```

FAQ: How to get the status of JavaScript in browser?

Ans: By using HTML element

```
    <noscript>
```

Ex:-

```
    <h2>JavaScript</h2>
        <noscript>
            Please Enable JavaScript on your browser.
        </noscript>
```

FAQ: How to get MIME Types?

Ans:

```
      <script>
         function f1(){
            for(var item of navigator.mimeTypes)
            {
               document.write(item.type + "<br>");
            }
         }
         f1();
      </script>
```

FAQ: How to access Geo Location?

Ans: By using "navigator.geoLocation.getCurrentPosition()"

Ex:-

```
      <script>
         function f1(){
 navigator.geolocation.getCurrentPosition(function(position){
           document.write(`
              Latitude : ${position.coords.latitude} <br>
              Longitude: ${position.coords.longitude}
           `);
       })
   }
   f1();
</script>
```

History Object

- It is used to access browser history details
- It allows

history.length	returns the total count of pages in current history
history.back()	moves to previous page in browser history.
history.forward()	moves to next page
history.go()	moves to specific page

Syntax:-

```
history.go('home.html')
history.go(1)            one page forward
history.go(-1)           one page back
```

Ex:-

```
<script>
   function f1(){
if(history.length<3) {
   document.write("You can view Max 3 Pages for free..");
} else{
   alert("Please Register for More..");
   location.href="login.html";
```

```
        }
    }
    f1();
</script>
```

jQuery