**JAVASCRIPT**

Contents

1. [Javascript](#java_script)
   1. [How javascript is integrated into HTML Page?](#javascript_Integrated_into_HTML)
   2. [Minification](#Minification)
   3. [How JavaScript refer HTML Elements?](#javascript_refer_html_elements)
      1. [JavaScript can refer elements by using DOM Hirarchy](#javascript_refer_element_by_dom)
      2. [JavaScript can refer elements by usingname](#javascript_refer_element_by_name)
      3. [JavaScript can refer elements by using ID](#javascript_refer_element_by_id)
      4. [JavaScript can refer elements by using CSS Selectors](#javascript_refer_element_by_css_selector)
2. [JavsScript Output Techniques](#javascript_output)
   1. [Alert](#javascript_output_alert)
   2. [Confirm](#javascript_output_confirm)
   3. [Document.write](#javascript_output_documentwrite)
   4. [InnerHTML](#javascript_output_innerHTML)
   5. [InnerText](#javascript_output_innerText)
   6. [OuterHTML](#javascript_output_outerHTML)
   7. [Console Methods](#javascript_output_console_methods)
3. [JavaScript Input Techniques](#javascript_input)
   1. [Query String](#javascript_input_query_string)
   2. [Prompt](#javascript_input_prompt)
   3. [Form Elements](#javascript_input_form_element)
4. [Variables](#variables)
5. [JavaScript Data Types](#javascript_data_type)
6. [Primitive Data Type](#primitive_type)
   1. [Number Type](#javascript_number_type)
   2. [String Type](#javascript_string_type)
      1. [String Manipulation](#javascript_string_manipulation)
   3. [Boolean Type](#boolean_type)
   4. [Undefined Type](#undefined_type)
   5. [Null Type](#null_type)
   6. [Symbol Type](#symbol_type)
7. [Non-Primitive Data Type](#Non_Primitive_type)
8. [Array Type](#array_type)
   1. [Array Manipulation](#array_manipulation)
      1. [Reading Values From Array](#array_manipulation_reading_values)
      2. [Adding Elements Into Array](#array_manipulation_adding_element)
      3. [Removing Elements from Array](#array_manipulation_removeing_element)
      4. [Sorting Array Elements](#array_manipulatioin_shorting_elements)
9. [Object Type](#javascript_object_type)
   1. [Array Of Objects](#object_array_of_object)
   2. [Object Manipulation](#object_manipulation)
   3. [NASA API Data Example](#nasa_api)
   4. [Nested Iterations And Map()](#nested_iterations_and_map)
   5. [Shopping Site Example Using API](#object_shopping_example_api)
10. [Map Type](#map_type)
11. [JavaScript Operators](#operators)
    1. [Arithemetic Operator](#operator_arithemetic)
    2. [Assignment Operator](#operator_assignment)
    3. [Comparision Operator](#operator_comparision_operator)
    4. [Logical Operator](#operator_logical)
    5. [Special Operators](#operator_special_operators)

# [JavaScript Statements](#statements)

* 1. [Selection Statements](#statements_Selection_statements)
     1. [Forward Jump](#statements_Selection_forward_jump)
     2. [Simple Decision](#statements_Selection_simple_decision)
     3. [Multi Level Decision](#statements_Selection_multi_level_)
     4. [Multiple Choices](#statements_Selection_multiple_choices)
     5. [Switch Selector](#statements_Selection_switch_selector)
  2. [Loop Control Statement](#statements_Looping_control)
     1. [For Loop](#statements_Looping_for)
     2. [While Loop](#statements_Looping_while)
     3. [Do While Loop](#statements_Looping_do_while)
  3. [Iteration Statements](#statements_Iteration)
  4. [Jump Statements](#statements_Jump)
  5. [Exception Handling Statement](#statements_Exception_handling)

1. [Javascript Functions](#functions)
   1. [Function Expression](#functions_expression)
   2. [Function Declaration](#functions_declaration)
   3. [Function Parameters](#functions_parameters)
   4. [REST Parameters](#functions_Rest_parameters)
   5. [Anonymous Function](#functions_Anonymous)
   6. [Function With Return](#functions_with_return)
   7. [Arrow Function](#functions_arrow)
   8. [Function Recursion](#functions_Recursion)
   9. [Function Closure](#functions_closure)
   10. [Function Generator](#functions_generator)
   11. [Function Debounce](#functions_Debounce)
   12. [Function Call Back & Function Promise](#functions_call_back_promise)
2. [Javascript OOPS](#oops)
   1. [OOPS](#oops_defination)
   2. [Javascript OOPs Features](#oops_features)
      1. [Modules in Javascript](#OOPS_features_modules)
      2. [Class in Javascript](#OOPS_features_classes)
3. [Property](#OOPS_features_classes_property)
4. [Accessors](#OOPS_features_classes_accessors)
5. [Method](#OOPS_features_classes_method)
6. Constructor
   * 1. [Code Extensibility](#code_extensibility)
     2. [Inheritance](#inheritance)
     3. [Polymorphism](#polymorphism)
7. [Javascript Data Structure](#js_data_structure)
8. [Javascript Events](#events)
   1. [Event Handler](#events_handler)
   2. [Event Arguments](#events_arguments)
   3. [Event Listener](#events_listener)
   4. [Events Types](#events_types)
9. [Javascript Browser Objects](#js_BrowserObjects)
   1. [Window](#js_BrowserObjects_window)
   2. [Location](#js_BrowserObjects_location)
   3. [Navigator](#js_BrowserObjects_navigator)
   4. [History](#js_BrowserObjects_history)
10. [AJAX](#ajax)
11. [Javascript State Management](#storage_management)
    1. [Local Storage](#localStorage)
    2. [Session Storage](#SessionStorage)
    3. [Cookies](#cookies)

1. [jQuery](#jQuery)
   1. [jQuery DOM Interation](#jQuery_DOMinteraction)
   2. [jQuery DOM Function](#jQuery_DOMfunctions)
   3. [jQuery Event](#jQuery_DOMEvents)
   4. [jQuery Effects](#jQuery_DOMEffects)
   5. [jQuery AJAX](#jQuery_ajax)
   6. [AJAX Phases](#jQuery_ajaxMethod)
   7. [jQuery UI](#jQuery_UI)

JavaScript  
Definition: JavaScript light weight interpreted and JIT compiled programming language.  
  
- Light weight occupies less memory.  
- Interpreted language translates line-by-line.  
- Interpreter is a translator.  
- Compiler is also a translator that translates all line in program simultaneously at the same time.  
- Compilers use 2 types compiling techniques  
        a) JIT    [Just-in-Time]  
        b) AOT [Ahead-of-Time]  
  
- JIT allows to translate in browser.  
- AOT allows to translate before reaching to browser. [Application Level]  
- Browser Engines [JIT]  
    V8  
    Webkit  
    Spider Monkey  
    Chakra etc...  
- AOT  
    - Node  
    - Babel etc..  
  
***Purpose:***  
    - JavaScript is a programming language used for various applications  
        a) Client Side                            HTML  
        b) Server Side                            Node JS, Express JS  
        c) Database                                MongoDB  
        d) 2D, 3D, Animation Softwares        Flash, 3DS Max etc..  
        e) Mobile Frameworks                    Cordova, Ionic, Native Script  
  
***FAQ: What is the role of JavaScript client side?***  
Ans:  JavaScript is used client side for handling  
        a) DOM manipulations  [ Document Object Model ]  
            - Adding a new element to page  
            - Removing element from page  
            - Updating data into element  
            - Formating element dynamically etc..  
        b) BOM Interactions  [ Browser Object Model ]  
            - Plugins  
            - MIME Types  
            - Extentions  
            - Location  
            - History etc...  
        c) Validations  
        d) Reduce burden on server  
  
***Programming Techniques:***  
        a) Structured Programming  
        b) Functional Programming  
        c) Imperative Programming  
        d) Object Oriented Programming etc..  
  
        JavaScript supports all types of programming techniques.  
  
Note: JavaScript is not an OOP language. It supports few features of OOP.  
  
**Evolution**  
------------  
- In early days of Internet [ 1970's ]  
  
            Browser        : Mosaic  
            Markup        : GML, SGML  
            Language    : ECMA Script  
  
- In early 1990's "Tim Berners Lee" introduced HTML and concept of Web  
  
            Browser        : Mosaic  
            Markup        : HTML  
            Lang            : ECMA Script  
  
- 1995 Netscape company developed  
  
            Browser        : Netscape Communicator  
            Markup        : HTML  
            Lang            : ECMA Script  
  
- 1996 Netscape appointed  "Brendan Eich" to design a script for browser [NC]  
  
            Script        : Mocha => Live Script  
  
- Netscape appointed "Sun MicroSystems" company to maintain "LiveScript"  
  
            Sun    => Java  
            Sun Micro => Named  Live Script => JavaScript  
  
- In early 2000  Netscape closed its service => JavaScript => ECMA International  
  
- ECMA Started developing JavaScript from 2014  
  
            Version JavaScript    =>  ECMA Script 2015  [ES5]  
                                         ECMA Script 2022  [ES2022]  
                                         ECMA Next

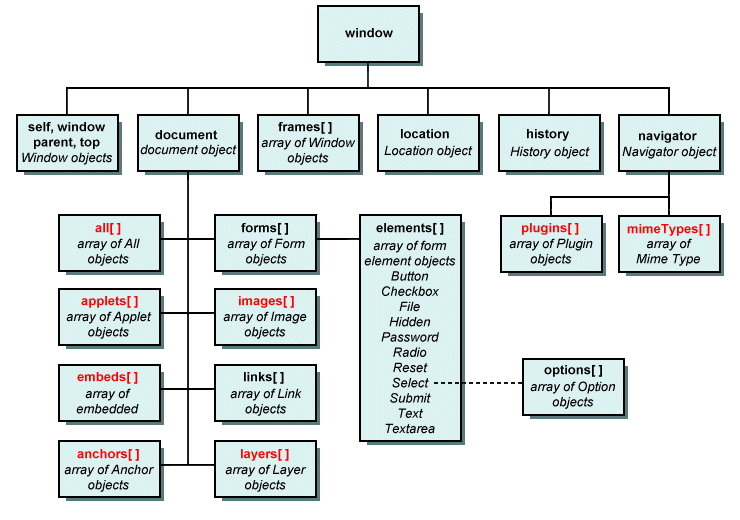
- What is JavaScript?  
- Where is JS is used?  
- Evolution of JS  
- Latest ES2022  [ECMA Script 2022]  
- What is the role of JS client side?  
 **What are the issues with JavaScript?**  
  
- JavaScript is not a strongly typed language.  
  
                price = 34000.33;  
                price = "TV";                // valid  
  
- Explicitly we have to verify the types and restrict the types.  
  
- JavaScript is not implicitly strictly typed. It will not use basic coding standards implicitly.  
  
- JavaScript is not an OOP language.  
  
        a) Extensibility issues  
        b) Maintainability issues  
        c) Testability issues  
        d) Code Security issues  
  
- JavaScript is not secured. It can be disabled by browsers.  
  
- Browser compatiblity issues  
  
Solution: TypeScript

How JavaScript is integrated into HTML Page?  
**1. Inline  
2. Embedded  
3. External File**  
  
**Ex: Inline**  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
</head>  
<body>  
    <h2>Welcome to JavaScript</h2>  
    <button onclick="window.print()">Print</button>  
</body>  
</html>  
  
**Ex: Embedded**  
1.  Code is embedded into page by using <script> element.  
2.  You can embed in head or body sections.  
3.  The MIME type for script is "text/javascript"  
  
11

Syntax:  
        <head>  
          <script type="text/javascript">  
  
          </script>  
        </head>  
  
Ex:   
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script type="text/javascript">  
        function PrintPage(){  
            window.print();  
        }  
    </script>  
</head>  
<body>  
    <h2>Welcome to JavaScript</h2>  
    <button onclick="PrintPage()">Print</button>  
    <button onclick="PrintPage()">Print Page</button>  
</body>  
</html>  
  
**Ex - External File**      
        ".js"  
- Create a new file in "src/scripts" folder  
  
          printing.js  
function PrintPage(){  
    window.print();  
}  
  
- Link to HTML page using <script> element "src" attribute.  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script type="text/javascript" src="../src/scripts/printing.js">  
         
    </script>  
</head>  
<body>  
    <h2>Welcome to JavaScript</h2>  
    <button onclick="PrintPage()">Print</button>  
    <button onclick="PrintPage()">Print Page</button>  
</body>  
</html>

Minification  
- It can compress JavaScript code  
- You need a JS Minifier tool  
- You can link minified file to page  
  
Ex:  
 printing.min.js  
  
function PrintPage(){window.print()}  
  
 index.html  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script type="text/javascript" src="../src/scripts/printing.min.js">  
         
    </script>  
</head>  
<body>  
    <h2>Welcome to JavaScript</h2>  
    <button onclick="PrintPage()">Print</button>  
    <button onclick="PrintPage()">Print Page</button>  
</body>  
</html>  
**Note: Uncompressed Script files are used in Development.  
        Minified Script files are used in Production.**

**How JavaScript takes control over HTML?**  
1. JavaScript can refer HTML elements by using DOM hierarchy



        window.document.images[]  
        window.document.forms[].elements[]  
  
    - It is the fastest way to take control over HTML.  
    - It is the native method of JavaScript.  
    - Refering to index of elements needs to change the index when ever position is  
     changed.  
    - It uses lengthy hierarchy.

Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script type="text/javascript">  
        function bodyload(){  
            window.document.images[0].src = "../public/images/asp.jpg";  
            window.document.forms[0].elements[0].value = "Register";  
            window.document.forms[1].elements[1].value = "Login";  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <div>  
        <img width="200" height="200" border="1">  
    </div>  
    <div>  
        <form>  
            <h2>Register</h2>  
            Your Mobile: <input type="button"> <input type="text">  
        </form>  
    </div>  
    <div>  
        <form>  
            <h2>Login</h2>  
           Your Email:  <input type="email"> <input type="button">  
        </form>  
    </div>  
</body>  
</html>

2. JavaScript can refer elements by using name  
  
 - Every element can have a reference name.  
 - You can access by using reference name  
 - You can't access child element directly.  
 - Name can be common for several elements.  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script type="text/javascript">  
        function bodyload(){  
            pic.src="../public/images/a1.jpg";  
            frmRegister.btnRegister.value = "Register";  
            frmLogin.btnLogin.value = "Login";  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <div>  
        <img width="200" name="pic" height="200" border="1">  
    </div>  
    <div>  
        <form name="frmRegister">  
            <h2>Register</h2>  
            Your Mobile:  <input type="text"> <input type="button" name="btnRegister">  
        </form>  
    </div>  
    <div>  
        <form name="frmLogin">  
            <h2>Login</h2>  
           Your Email:  <input type="email"> <input name="btnLogin" type="button">  
        </form>  
    </div>  
</body>  
</html>

3. JavaScript can refer element using ID  
     
            document.getElementById()  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script type="text/javascript">  
        function bodyload(){  
            document.getElementById("pic").src = "../public/images/a2.jpg";  
            document.getElementById("btnRegister").value = "Register";  
            document.getElementById("btnLogin").value = "Login";  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <div>  
        <img width="200" id="pic" name="pic" height="200" border="1">  
    </div>  
    <div>  
        <form name="frmRegister">  
            <h2>Register</h2>  
            Your Mobile:  <input type="text"> <input id="btnRegister" type="button" name="btnRegister">  
        </form>  
    </div>  
    <div>  
        <form name="frmLogin">  
            <h2>Login</h2>  
           Your Email:  <input type="email"> <input id="btnLogin" name="btnLogin" type="button">  
        </form>  
    </div>  
</body>  
</html>

4. JavaScript can refer elements by using CSS selectors  
  
            document.querySelector("cssSelector")  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script type="text/javascript">  
        function bodyload(){  
           document.querySelector("img").src= "../public/images/asp.jpg";  
           document.querySelector("#btnRegister").value = "Register";  
           document.querySelector(".btn-login").value = "Login";  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <div>  
        <img width="200"  height="200" border="1">  
    </div>  
    <div>  
        <form name="frmRegister">  
            <h2>Register</h2>  
            Your Mobile:  <input type="text"> <input id="btnRegister" type="button" name="btnRegister">  
        </form>  
    </div>  
    <div>  
        <form name="frmLogin">  
            <h2>Login</h2>  
           Your Email:  <input type="email"> <input id="btnLogin" name="btnLogin" class="btn-login" type="button">  
        </form>  
    </div>  
</body>  
</html>

JavaScript Output Techniques- Output is the process of rendering result on screen.  
- JavaScript output can be handled by using various techniques  
***1. alert()  
        2. confirm()          
        3. document.write()  
        4. innerHTML  
        5. outerHTML  
        6. innerText  
        7. console methods : log(), warn(), error(), debug(), info()...***  
1. Alert ()   
- It pops-up a message box in browser window.  
- It contains only "OK" to confirm.  
- It will not have cancel or close.  
  
Syntax:  
            alert("your message");        // single line alert  
            alert("line-1 \n line-2");        // multiline alert  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Output</title>  
    <script>  
        function DeleteClick(){  
            alert("Delete Record\nRecord will be deleted permanently");  
        }  
    </script>  
</head>  
<body>  
    <button onclick="DeleteClick()">Delete</button>  
</body>  
</html>  
  
Note: alert() is RC data type.

2. confirm()  
- It is similar to alert but allows to cancel.  
- It is a boolean type: return true on OK and false on Cancel.  
  
Syntax:  
        confirm("your message");  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Output</title>  
    <script>  
        function DeleteClick(){  
           result = confirm("Delete Record\nRecord will be deleted permanently");  
           if(result==true){  
              alert("Record Deleted..");  
           } else {  
              alert("You canceled..");  
           }  
        }  
    </script>  
</head>  
<body>  
    <button onclick="DeleteClick()">Delete</button>  
</body>  
</html>

3. document.write()- It is used to present output on a new screen.  
- It can present complex data.  
- It allows all formats using markup.  
  
Syntax:  
    document.write("<markup>text</markup>")  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Output</title>  
    <script>  
        function DeleteClick(){  
           result = confirm("Delete Record\nRecord will be deleted permanently");  
           if(result==true){  
              document.write("<font color='red'><i><b>Record Deleted..</b></i></font><br><a href='output.html'>Back</a>");  
           } else {  
              alert("You canceled..");  
           }  
        }  
    </script>  
</head>  
<body>  
    <button onclick="DeleteClick()">Delete</button>  
</body>  
</html>

4. innerHTML  
- It presents output in any container of HTML that can display text content.  
    <div> <span> <p> <h2> <blockquote> <dd> <td> <dt> <button>  
- It supports all formats for text.  
  
5. innerText  
- It is similar to innerHTML but will not allow formats.  
  
Syntax:  
        document.querySelector("div | p | span").innerText = "some text";  
                                                         .innerHTML = "some formated text";  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Output</title>  
    <script>  
        function DeleteClick(){  
           result = confirm("Delete Record\nRecord will be deleted permanently");  
           if(result==true){      
                document.querySelector("p").innerHTML = "<font color='red'>Record Deleted Successfully..</font>";  
           } else {  
               document.querySelector("p").innerText = "You canceled..";  
           }  
        }  
    </script>  
</head>  
<body>  
    <button onclick="DeleteClick()">Delete</button>  
    <p align="center"></p>  
</body>  
</html>

6. outerHTML  
- It is similar to innerHTML, but it will replace the old content with new content.  
- In innerHTML the new content is added into old content.  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Output</title>  
    <script>  
        function DeleteClick(){  
           result = confirm("Delete Record\nRecord will be deleted permanently");  
           if(result==true){      
                document.querySelector("p").outerHTML = "<h2>Record Deleted Successfully..</h2>";  
           } else {  
               document.querySelector("p").innerText = "You canceled..";  
           }  
        }  
    </script>  
</head>  
<body>  
    <button onclick="DeleteClick()">Delete</button>  
    <p align="center"></p>  
</body>  
</html>

7. console methods  
- Console is a CLI tool [Command Line Interface] for browser.  
- Developers use console to test the code and monitor is functions, errors, warning etc..  
  
Syntax:  
        console.log("message");  
  
- It is RC data type.  
  
        console.warn("line-1 \n line-2");  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Output</title>  
    <script>  
        function DeleteClick(){  
            now = new Date();  
           console.warn("Warning : Delete Button Clicked at " + now.toLocaleTimeString());  
           result = confirm("Delete Record\nRecord will be deleted permanently");  
           if(result==true){      
                document.querySelector("p").outerHTML = "<h2>Record Deleted Successfully..</h2>";  
                console.error("OK Button Clicked at "  + now.toLocaleTimeString());  
           } else {  
               document.querySelector("p").innerText = "You canceled..";  
               console.log("Cancel Button Clicked");  
           }  
        }  
    </script>  
</head>  
<body>  
    <button onclick="DeleteClick()">Delete</button>  
    <p align="center"></p>  
</body>  
</html>

JavaScript Input Techniques **1. Query String  
2. prompt()  
3. Form Input elements**  
  
  
1. Query String  
- It is a search string passed in the browser address bar.  
- It is appended to URL using "?referenceName"  
  
                ?ref=value  
  
- You can access querystring by using  "location.search"  
  
Ex:  
    location.search.substring(location.search.indexOf("=") + 1)  
  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function bodyload(){  
            document.getElementById("user").innerHTML = location.search.substring(location.search.indexOf("=")+1);  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <p>Hello ! <span id="user"></span></p>  
</body>  
</html>

2. prompt()                               
- It provides an input box, which is similar to an alert box.  
- It allows user to input a value.  
  
Syntax:  
            prompt("Your Message", "Default\_Value");  
            prompt("Your Message");  
  
- Prompt method returns  
            Empty String  " "        => when you click OK without value  
            Value                    => when you click OK with value  
            null                    => when you click Cancel  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Input | Prompt</title>  
    <script>  
        function CreateClick(){  
            result = prompt("Enter Folder Name","New\_Folder");  
            if(result==null) {  
                alert("You Canceled..");  
            } else if(result=="") {  
                alert("Folder Name Can't be Empty");  
            } else {  
                document.querySelector("p").innerHTML += "Folder Created : " + result + "<br>";  
            }  
        }  
    </script>  
</head>  
<body>  
    <button onclick="CreateClick()">Create Folder</button>  
    <p></p>  
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Input | Prompt</title>  
    <script>  
        function CreateClick(){  
            foldername = prompt("Enter Folder Name");  
            purpose = prompt("Enter the purpose of " + foldername + " Folder");  
            document.querySelector("p").innerHTML = "Folder Created :" + foldername + "<br>Used for =>" + purpose;  
        }  
    </script>  
</head>  
<body>  
    <button onclick="CreateClick()">Create Folder</button>  
    <p></p>  
</body>  
</html>

3. JavaScript Input using Form Elements  
  
- You can input value using form input elements like  
    text, password, number, date, url, email, color, file, select, textarea etc..  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Form Input</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 type="text/javascript">  
        function RegisterClick(){  
            document.getElementById("RegisterContainer").style.display = "none";  
            document.getElementById("DetailsContainer").style.display = "block";  
  
            document.getElementById("lblName").innerHTML = document.getElementById("txtName").value;  
            document.getElementById("lblPrice").innerHTML = document.getElementById("txtPrice").value;  
            document.getElementById("lblCity").innerHTML = document.getElementById("lstCities").value;  
  
            StockCheckBox = document.getElementById("optStock");  
            StockStatus = "";  
  
            if(StockCheckBox.checked) {  
                StockStatus = "Available";  
            } else {  
                StockStatus = "Out of Stock";  
            }  
  
            document.getElementById("lblStock").innerHTML = StockStatus;  
  
        }  
        function EditClick(){  
            document.getElementById("btnRegister").innerHTML = "Save";  
            document.getElementById("btnRegister").className = "btn btn-success";  
        }  
    </script>  
</head>  
<body class="container-fluid">  
    <div class="mt-4">  
        <div id="RegisterContainer">  
            <button class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#register">Register Product</button>  
        </div>  
        <div id="DetailsContainer" style="display:none">  
            <table class="table table-hover table-dark w-25 caption-top">  
                <caption>Product Details</caption>  
                <tr>  
                    <td>Product Name</td>  
                    <td id="lblName" class="bg-warning"></td>  
                </tr>  
                <tr>  
                    <td>Price</td>  
                    <td id="lblPrice" class="bg-warning"></td>  
                </tr>  
                <tr>  
                    <td>Shipped To</td>  
                    <td id="lblCity" class="bg-warning"></td>  
                </tr>  
                <tr>  
                    <td>Stock Status</td>  
                    <td id="lblStock" class="bg-warning"></td>  
                </tr>  
            </table>  
            <p>  
                <button onclick="EditClick()" data-bs-target="#register" data-bs-toggle="modal" class="btn btn-info">  
                    <span class="bi bi-pen-fill"></span> Edit  
                </button>  
                <a class="btn btn-primary" href="input.html">  
                    <span class="bi bi-file"></span> New Product  
                </a>  
            </p>  
        </div>  
        <div class="modal fade" id="register">  
            <div class="modal-dialog">  
                <div class="modal-content">  
                    <div class="modal-header">  
                        <h2>Register 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="number" id="txtPrice" class="form-control"></dd>  
                            <dt>Shipped To</dt>  
                            <dd>  
                                <select id="lstCities" class="form-select">  
                                    <option>Delhi</option>  
                                    <option>Hyd</option>  
                                    <option>Chennai</option>  
                                </select>  
                            </dd>  
                            <dt>Stock</dt>  
                            <dd class="form-switch"><input type="checkbox" id="optStock" class="form-check-input"> Available</dd>  
                        </dl>  
                    </div>  
                    <div class="modal-footer">  
                        <button id="btnRegister" class="btn btn-primary" onclick="RegisterClick()" data-bs-dismiss="modal">Register</button>  
                        <button class="btn btn-danger" data-bs-dismiss="modal">Cancel</button>  
                    </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>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Form Input</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 type="text/javascript">  
        function BookClick(){  
            document.getElementById("lblCinema").innerHTML = document.getElementById("lstCinema").value;  
            document.getElementById("lblMovie").innerHTML = document.getElementById("lstMovie").value;  
            document.getElementById("lblDate").innerHTML = document.getElementById("lstDate").value;  
            document.getElementById("lblTime").innerHTML = document.getElementById("lstTime").value;  
            document.getElementById("lblSeats").innerHTML = document.getElementById("lstSeats").value;  
  
            poster = document.getElementById("poster");  
  
            movieName = document.getElementById("lstMovie").value;  
             
            if(movieName=="John Wick - Chapter-4") {  
                poster.src = "../public/images/john.jfif";  
            }  
            if(movieName=="Everything Everywhere") {  
                poster.src = "../public/images/every.jfif";  
            }  
  
             
        }  
    </script>  
</head>  
<body class="container-fluid">  
    <button class="mt-3 btn btn-danger" data-bs-target="#book" data-bs-toggle="modal">  
        Book Ticket  
    </button>  
    <div class="modal" id="book">  
         
        <div class="modal-dialog modal-fullscreen">  
            <div class="modal-content">  
                <div class="modal-header">  
                    <h3>Book Movie Ticket</h3>  
                    <button class="btn-close" data-bs-dismiss="modal"></button>  
                </div>  
                <div class="modal-body">  
                    <div class="d-flex justify-content-between">  
                        <div>  
                            <select class="form-select" id="lstCinema">  
                                <option>Select Cinema</option>  
                                <option>PVR Cinemas</option>  
                                <option>Inox Movies</option>  
                            </select>  
                        </div>  
                        <div>  
                            <select class="form-select" id="lstMovie">  
                                <option>Select Movie</option>  
                                <option>John Wick - Chapter-4</option>  
                                <option>Everything Everywhere</option>  
                            </select>  
                        </div>  
                        <div>  
                            <select class="form-select" id="lstDate">  
                                <option>Select Date</option>  
                                <option>Today 23 March</option>  
                                <option>Tomorrow 24 March</option>  
                            </select>  
                        </div>  
                        <div>  
                            <select class="form-select" id="lstTime">  
                                <option>Select Time</option>  
                                <option>10:45 AM</option>  
                                <option>02:20 PM</option>  
                            </select>  
                        </div>  
                        <div>  
                            <select class="form-select" id="lstSeats">  
                                <option>Select Seats</option>  
                                <option>1</option>  
                                <option>2</option>  
                                <option>3</option>  
                            </select>  
                        </div>  
                        <div>  
                            <button onclick="BookClick()" class="btn btn-danger" data-bs-dismiss="modal">Book</button>  
                        </div>  
                    </div>  
                </div>  
            </div>  
        </div>  
    </div>  
    <div>  
        <table class="table table-hover">  
            <thead>  
                <tr>  
                    <th>Booking Summary</th>  
                </tr>  
            </thead>  
            <tbody>  
                <tr>  
                    <td>  
                        <img width="200" height="200" id="poster">  
                    </td>  
                </tr>  
                <tr>  
                    <td>Cinema</td>  
                    <td id="lblCinema"></td>  
                </tr>  
                <tr>  
                    <td>Movie</td>  
                    <td id="lblMovie"></td>  
                </tr>  
                <tr>  
                    <td>Date</td>  
                    <td id="lblDate"></td>  
                </tr>  
                <tr>  
                    <td>Time</td>  
                    <td id="lblTime"></td>  
                </tr>  
                <tr>  
                    <td>Seats</td>  
                    <td id="lblSeats"></td>  
                </tr>  
            </tbody>  
        </table>  
    </div>  
    <script src="../node\_modules/jquery/dist/jquery.js"></script>  
    <script src="../node\_modules/bootstrap/dist/js/bootstrap.bundle.js"></script>  
</body>  
</html>

Summary  
- How JavaScript is integrated into HTML?  
- How JavaScript refers HTML?  
- How JavaScript handles Input and Output?  
  
**=> JavaScript Language Basics**  
**1. Variables  
2. Data Types  
3. Operators  
4. Statements  
5. Functions**

Variables  
- Variables are storage locations in memory where you can store a value and use it as part of any expression.  
  
- Variable configuration in JavaScript comprises of 3 phases  
  
        a) Declaration            var x;  
  
        b) Initialization            var x = 10;  
  
        c) Assignment            var x;  
                                    x = 10;  
  
- In JavaScript you can directly assign a value into variable without declaring variable if JavaScript is not in strict mode.  
  
Ex:  
<script>  
     x = 10;                                // valid  
     document.write("x=" + x);  
</script>  
  
  
Ex:  
<script>  
     "use strict";  
     x = 10;                                // invalid  
     document.write("x=" + x);  
</script>  
  
- Variables in JavaScript are declared by using 3 keywords  
  
            a) var  
            b) let  
            c) const  
  
**var**-  
  
- It is used to define a function scope variable.  
- You can declare in any block of a function and access from any another block in function.  
- It allows declaring, assignment and initialization.  
  
Ex:  
<script>  
     "use strict";  
      function f1()  
      {  
         var x;                 // Declaring  
         x = 10;                // assignment  
         if(x==10)  
         {  
            var y = 20;         // initialization  
         }  
         document.write("x=" + x + "<br> y=" + y);  
      }  
      f1();  
</script>  
  
- var allows shadowing.  
- shadowing is the process of re-declaring or initializing a variable with in the scope again.  
  
Ex:  
<script>  
     "use strict";  
      function f1()  
      {  
         var x;                 // Declaring  
         x = 10;                // assignment  
         x = 12;  
         var x;                // shadowing - x  
         x = 14;  
         if(x==14)  
         {  
            var y = 20;         // initialization  
            y = 30;  
            y = 40;  
            var y = 50;         // shadowing - y  
         }  
         document.write("x=" + x + "<br> y=" + y);  
      }  
      f1();  
</script>  
  
- var allows hoisting.  
- Hoisting allows to use a variable and later declare.  
  
Ex:  
<script>  
     "use strict";  
      function f1()  
      {  
         x = 10;  
         document.write("x=" + x);  
         var x;                     // hoisting  
      }  
      f1();  
</script>

**Let**-  
  
- It defines a block scope variable.  
- It is accessible only in side the block where it is declared and all its inner block.  
- It supports declaring, assignment and intialization.  
- It will not support shadowing and hoisting.  
  
**const**  
  
- It defines a block scope variable.  
- It supports only initialization. No declaring and No assignment.  
- It will not support shadowing and hoisting.  
  
Syntax:  
        const x;                // invalid  
        const  x = 10;            // valid  
        x = 20;                // invalid  
  
- Global Scope for variable is defined by declaring variable outside function.  
  [ module scope ]  
  
Ex:  
<script>  
     "use strict";  
      var x = 10;  
      let y = 20;  
      const z = 30;  
  
      function f1(){  
        document.write("F1 x =" + x + "y=" + y + "z=" + z + "<br>");  
      }  
      function f2(){  
        document.write("F2 x =" + x + "y=" + y + "z=" + z + "<br>");  
      }  
      f1();  
      f2();  
</script>  
  
FAQ: Can we define a variable inside function with Global access?  
Ans : Yes. Only in HTML by using "window" object.  
  
Ex:  
<script>  
     "use strict";  
      var x = 10;  
      let y = 20;  
      const z = 30;  
  
      function f1(){  
        window.a = 40;  
        document.write("F1 x =" + x + "y=" + y + "z=" + z + "a=" + a + "<br>");  
      }  
      function f2(){  
        document.write("F2 x =" + x + "y=" + y + "z=" + z + "a=" + a + "<br>");  
      }  
      f1();  
      f2();  
</script>  
  
**Variable Naming  Rules**:  
- Name must  start with an alphabet.  
- You can start with \_ but it is not always recommended.  
  
                var \_product;        => It not final, It required furthur implmentation.  
  
- It can be alpha numeric.  
  
                var  sales2022;  
  
- Don't use special chars.  
  
- Variable name can be max 255 chars long.  
  
- Keep always variable name in Camel Case.  
  
            var  btnSubmit;  
 PMD, Sonar, ESLINT  => Code Analyzer

JavaScript Data Types  
  
- Data type defines data structure.  
- It specifies what type of data and range of data can be stored in a reference.  
- JavaScript is not strongly typed language.  
- It is implicitly typed language. Data type of variable is determined according to the value initialized or assigned.  
  
                        var x;            undefined  
                        x = 10;        number  
                        x = "A";        string  
  
- JavaScript data types are classified into 2 types  
  
**1. Primitive Types  
    2. Non Primitive Types**  
Primitive Types  
- They are immutable types.  
- They have a fixed structure.  
- Structure can't change dynamically.  
- They have fixed range for values.  
- JavaScript Primitive Types  
     
            **1. number  
            2. string  
            3. boolean  
            4. null  
            5. undefined  
            6. symbol  [ES6]**

1. JavaScript Number Type  
- A numeric type is reffered as number.  
- JavaScript number can be  
            Signed Integer            -10  
            Unsigned Integer        10  
            Floating Point            23.34  
            Double                    321.453  
            Decimal                  3400.33            [29]      
            Binary                        0b1010  
            Hexa                        0f0000  
            Octa                        0o763  
            Exponent                    2e3  [2000]   2 x 10[3]  
            BigInt                        2n  
  
Syntax:  
            var x = 2e3;  
            var y = 0b1010;  
            var z = 0o748;  
  
- Every input from HTML is considered as string type.  
- JavaScript provides parsing methods for converting a numeric string into number.  
  
                a) parseInt()  
                b) parseFloat()  
  
- A numeric string can start with a number and can contain string chars.  
  
                    parseInt("10A") + 20    = 30;  
                    parseInt("A10") + 20    = NaN  
                    parseInt("A") + 20        = NaN  
                    parseInt("10A20") + 20  = 30  
                    parseInt("10.22") + 20  = 30  
                    parseFloat("10.22") + 20 = 30.22  
  
- JavaScript can verify the input type as number by using the method "isNaN()".  
- It is a boolean method that returns true if input value is not a number.  
  
Ex:  
<script>  
      var rate = parseFloat(prompt("Enter Interest Rate"));  
      if(isNaN(rate)) {  
        document.write("Interest Rate must be a number.");  
      } else {  
      document.write("You will be " + (rate+1) + " Next year");  
      }  
</script>  
  
- JavaScript number can be converted into a string by using "toString()".  
  
                        var price = $560.56;        // invalid  
                        var price  = 54,560.45  // invalid  
Ex:  
<script>  
      var price = 4500.44;  
      var printPrice = "&#8377;" + price.toString();  
      document.write(printPrice);  
</script>  
  
- JavaScript requires a special library called "Math" to handle numbers in complex equations.  
                    Math.PI  
                    Math.cos()  
                    Math.tan()  
                    Math.sin()  
                    Math.sqrt()  
                    Math.pow()  
                    Math.avg()  
                    etc...  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta http-equiv="X-UA-Compatible" content="IE=edge">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>Document</title>  
  <script>  
     function bodyload(){  
          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;  
          var code = Math.round(a) + " " + Math.round(b) + " " + Math.round(c) + " " + Math.round(d) + " " + Math.round(e) + " " + Math.round(f);  
          document.getElementById("code").innerHTML = code;  
     }  
  </script>  
</head>  
<body onload="bodyload()">  
   <h2>User Login</h2>  
   <dl>  
    <dt>User Name</dt>  
    <dd><input type="text"></dd>  
    <dt>Password</dt>  
    <dd><input type="password"></dd>  
    <dt>Verify Code</dt>  
    <dd id="code"></dd>  
   </dl>  
   <button>Login</button>  
</body>  
</html>

EX: EMI Calculator  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>EMI calculator</title>  
    <link rel="stylesheet" href="../node\_modules/bootstrap/dist/css/bootstrap.css">  
    <script type="text/javascript">  
        function AmountChange(){  
            document.getElementById("txtAmount").value = document.getElementById("rangeAmount").value;  
        }  
        function YearChange(){  
            document.getElementById("txtYears").value = document.getElementById("rangeYear").value;  
        }  
        function RateChange(){  
            document.getElementById("txtRate").value = document.getElementById("rangeInterest").value;  
        }  
        function CalculateClick(){  
            var P = parseInt(document.getElementById("txtAmount").value);  
            var n = parseInt(document.getElementById("txtYears").value) \* 12;  
            var r = parseFloat(document.getElementById("txtRate").value)/12/100;  
  
            var emi = P \* r \* Math.pow((1+r),n) / Math.pow((1+r),n) - 1;  
  
            document.getElementById("result").innerHTML = "Your montly installment amount is &#8377; " + Math.round(emi) + " for " + (n/12) + " years with " + document.getElementById("txtRate").value + "% interest rate.";  
        }  
    </script>  
</head>  
<body class="container-fluid">  
    <h2>EMI Calculator</h2>  
    <div style="box-shadow:2px 2px 3px gray; padding: 20px;">  
        <div class="row mt-3">  
            <div class="col">  
                Amount you need &#8377; <input type="text" id="txtAmount" size="15">  
            </div>  
            <div class="col">  
                for <input type="text" id="txtYears" size="2"> years  
            </div>  
            <div class="col">  
                Interest rate <input type="text" id="txtRate" size="4"> %  
            </div>  
        </div>  
        <div class="row mt-4">    
            <div class="col">  
                <div class="input-group">  
                    <span>&#8377; 50,000</span> <div><input class="form-range mx-2" min="50000" max="1000000" value="50000" type="range" onchange="AmountChange()" id="rangeAmount"></div> <span class="ms-2">&#8377; 10,00,000</span>  
                </div>  
            </div>  
            <div class="col">  
                <div class="input-group">  
                    <span>1</span> <div><input class="form-range mx-2" min="1" max="5" value="1" type="range" onchange="YearChange()" id="rangeYear"></div> <span class="ms-2"> 5 </span>  
                </div>  
            </div>  
            <div class="col">  
                <div class="input-group">  
                    <span>10.45%</span> <div><input class="form-range mx-2" min="10.45" max="18.45" value="10.45" step="0.01" onchange="RateChange()" type="range" id="rangeInterest"></div> <span class="ms-2"> 18.45% </span>  
                </div>  
            </div>  
        </div>  
        <div class="row mt-4">  
            <div class="col text-end">  
                <button onclick="CalculateClick()" class="btn btn-primary">Calculate</button>  
            </div>  
        </div>  
    </div>  
    <h2 class="text-center mt-3" id="result">  
  
    </h2>  
</body>  
</html>  
  
Task : Design a BMI Calculator  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function GenderChange(){  
            var maleRadio = document.getElementById("optMale");  
            var femaleRadio = document.getElementById("optFemale");  
            if(maleRadio.checked) {  
                document.getElementById("msg").innerHTML = "Your Gender is Male and BMI Forumal for Male..";  
            }  
            if(femaleRadio.checked) {  
                document.getElementById("msg").innerHTML = "Your Gender is Female and BMI For female";  
                 
            }  
        }  
    </script>  
</head>  
<body>  
    <p>  
        <input type="radio" onchange="GenderChange()" name="gender" id="optMale"> Male  
        <input type="radio" onchange="GenderChange()" name="gender" id="optFemale"> Female  
    </p>  
    <p>  
        <span id="msg"></span>  
    </p>  
</body>  
</html>

2. String Type  
- String is a literal with group of characters enclosed in  
  
    a) Double Quotes            " "  
    b) Single Quotes                ' '  
    c) Back Ticks [ES5]            ` `  
  
- Double and Single quotes are used to switch outer and inner string.  
  
Ex:  
<script>  
     var link = "<a href='home.html'>Home</a>";  
     var link2 = '<a href="emi.html">EMI</a>'  
     document.write(link + "<br>");  
     document.write(link2);  
</script>  
  
- JS ES5+ versions introduced "back tick" which can handle data binding by using a binding expression "${ }"  
  
- Data binding expression is not allowed in single and double quotes.  
  
Syntax:  
                `string chars ${dynamic} string chars`  
                "string chars" + dynamic + "string chars"  
  
Ex:  
<script>  
    var title = prompt("Enter Title");  
    var controlType = prompt("Enter Control Type", "text|email|password");  
  
    var login = `  
        <h2>${title}</h2>  
        <span>Your ${controlType}</span> : <input type=${controlType}> <button>Login</button>  
    `;  
  
    document.write(login);  
</script>

Escape Sequence  
  
- Several string chars will escape printing.  
- You have to manually print the non-printing characters.  
- "\" is used as escape sequence character.  
  
Ex:  
<script>  
     var path = "\"D:\\project\\images\\banner.jpg\"";  
     document.write(path);  
</script>  
  
String Formatting Methods -  
bold()  
italics()  
sup()  
sub()  
fontcolor()  
fontsize()  
toUpperCase()  
toLowerCase()  
  
Note: Text formatting  can be applied using string methods only for elements which are not RC data type. RC Type elements can allow only changing of case not format.  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>String Format</title>  
    <script>  
        function ChangeCase(){  
            var username = document.getElementById("txtName").value;  
            document.getElementById("txtName").value = username.toUpperCase();  
        }  
        function VerifyName(){  
            var username = document.getElementById("txtName").value;  
            var error = document.getElementById("Error");  
            if(username=="") {  
                error.innerHTML = "User Name Required".bold().fontcolor('red').italics();  
            } else {  
                error.innerHTML = "";  
            }  
        }  
    </script>  
</head>  
<body>  
    <dl>  
        <dt> Your Name </dt>  
        <dd><input type="text" id="txtName" onblur="VerifyName()" onkeyup="ChangeCase()" placeholder="Name in Block Letters"></dd>  
        <dd id="Error"></dd>  
    </dl>  
  
</body>  
</html>  
  
**Formatting string with styles -**  
  
- Style can be applied to string by using "style" property.  
  
        document.getElementById("txtName").style.styleAttribute = value;  
         
        styleAttribute must be in "camelCase".  
  
        font-size                    fontSize  
        background-color        backgroundColor  
        text-align                    textAlign  
        color                        color  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>String Format</title>  
    <script>  
        function ChangeCase(){  
            var username = document.getElementById("txtName").value;  
            document.getElementById("txtName").value = username.toUpperCase();  
        }  
        function VerifyName(){  
            var username = document.getElementById("txtName");  
            var error = document.getElementById("Error");  
            if(username.value=="") {  
                error.innerHTML = "User Name Required".bold().fontcolor('red').italics();  
                username.style.border = "1px solid red";  
                username.style.boxShadow = "2px 2px 2px red";  
            } else {  
                error.innerHTML = "";  
            }  
        }  
    </script>  
</head>  
<body>  
    <dl>  
        <dt> Your Name </dt>  
        <dd><input type="text" id="txtName" onblur="VerifyName()" onkeyup="ChangeCase()" placeholder="Name in Block Letters"></dd>  
        <dd id="Error"></dd>  
    </dl>  
  
</body>  
</html>  
  
 **Format String with CSS classes -**  
  
- CSS class comprises of a set of effects.  
- You can apply the effects for any string using CSS classes.  
  
            .text-effect { }  
  
            document.getElementById("error").className = "text-effect";  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>String Format</title>  
    <style>  
        .error-style {  
            color:red;  
            font-style: italic;  
            font-weight: bold;  
        }  
    </style>  
    <script>  
        function ChangeCase(){  
            var username = document.getElementById("txtName").value;  
            document.getElementById("txtName").value = username.toUpperCase();  
        }  
        function VerifyName(){  
            var username = document.getElementById("txtName");  
            var error = document.getElementById("Error");  
            if(username.value=="") {  
                error.innerHTML = "User Name Required";  
                error.className = "error-style";  
                username.style.border = "1px solid red";  
                username.style.boxShadow = "2px 2px 2px red";  
            } else {  
                error.innerHTML = "";  
            }  
        }  
    </script>  
</head>  
<body>  
    <dl>  
        <dt> Your Name </dt>  
        <dd><input type="text" id="txtName" onblur="VerifyName()" onkeyup="ChangeCase()" placeholder="Name in Block Letters"></dd>  
        <dd id="Error"></dd>  
    </dl>  
  
</body>  
</html>  
  
Ex: Dark Theme  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</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  ThemeChange(){  
            var cbTheme = document.getElementById("cbTheme");  
            var form = document.querySelector("form");  
            var loginButton = document.querySelector("button");  
  
            if(cbTheme.checked) {  
                form.className = "bg-dark text-white p-4 border border-primary border-3";  
                loginButton.className = "btn btn-light w-100";  
            } else {  
                form.className = "";  
                loginButton.className = "btn btn-dark w-100";  
            }  
        }  
    </script>  
</head>  
<body class="container-fluid">  
    <div class="d-flex justify-content-center align-items-center" style="height:500px">  
        <form>  
            <div class="mt-3 mb-3 form-switch">  
                <input type="checkbox" onchange="ThemeChange()" id="cbTheme" class="form-check-input"> Dark Theme  
            </div>  
            <h2><span class="bi bi-person-fill"></span> User Login</h2>  
            <dl>  
                <dt>User Name</dt>  
                <dd><input type="text" class="form-control"></dd>  
                <dt>Password</dt>  
                <dd><input type="password" class="form-control"></dd>  
            </dl>  
            <button class="btn btn-dark w-100">Login</button>  
        </form>  
    </div>  
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function SizeChange(){  
            var size = parseInt(document.getElementById("sizeRange").value);  
            document.querySelector("p").style.fontSize = size + "px";  
        }  
        function ApplyColor(){  
            document.querySelector("p").style.color = document.getElementById("txtColor").value;  
        }  
        function AlignChange(){  
            document.querySelector("p").style.textAlign = document.getElementById("lstAlign").value;  
        }  
    </script>  
</head>  
<body>  
    <dl>  
        <dt>Font Size</dt>  
        <dd>10<input onchange="SizeChange()" id="sizeRange" type="range" min="10" max="100" value="10">100</dd>  
        <dt>Choose Color</dt>  
        <dd><input type="color" onchange="ApplyColor()" id="txtColor"></dd>  
        <dt>Select Alignment</dt>  
        <dd>  
            <select id="lstAlign" onchange="AlignChange()">  
                <option>left</option>  
                <option>center</option>  
                <option>right</option>  
            </select>  
        </dd>  
    </dl>  
    <p align="center">Sample Text</p>  
</body>  
</html>

String Manipulation Methods and Properties   
  
**1. length**            : It returns the total number of chars in a string.  
  
                     var str = "Welcome";  
                     str.length                    => 7  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function VerifyUser(){  
            var username = document.getElementById("UserName").value;  
            var UserError = document.getElementById("UserError");  
            if(username.length<4) {  
                UserError.innerHTML = "Name too short min 4 chars required".fontcolor('red');  
            }else if(username.length>10) {  
                UserError.innerHTML = "Name too long max 10 chars only".fontcolor('red');  
            }  
            else {  
                UserError.innerHTML = "";  
            }  
        }  
        function VerifyRequired(){  
            var username = document.getElementById("UserName").value;  
            if(username=="") {  
                document.getElementById("UserError").innerHTML = "User name required".fontcolor('red');  
            } else {  
                document.getElementById("UserError").innerHTML = "";  
            }  
        }  
    </script>  
</head>  
<body>  
    <dl>  
        <dt>User Name</dt>  
        <dd><input type="text" onkeyup="VerifyUser()" onblur="VerifyRequired()" id="UserName"></dd>  
        <dd id="UserError"></dd>  
    </dl>  
</body>  
</html>  
  
**2. charAt()**            : It returns the character at specified index.  
  
                            var str = "Welcome";  
                            str.charAt(1);                // e  
  
Ex:<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function VerifyUser(){  
            var username = document.getElementById("UserName").value;  
            var UserError = document.getElementById("UserError");  
            if(username.length<4) {  
                UserError.innerHTML = "Name too short min 4 chars required".fontcolor('red');  
            }else if(username.length>10) {  
                UserError.innerHTML = "Name too long max 10 chars only".fontcolor('red');  
            }  
            else {  
                UserError.innerHTML = "";  
            }  
        }  
        function VerifyRequired(){  
            var username = document.getElementById("UserName").value;  
            var userError = document.getElementById("UserError");  
            if(username.charAt(0)=="A") {  
                userError.innerHTML = "";  
            } else {  
                userError.innerHTML = "Name must start with 'A' Character.".fontcolor('red');  
            }  
        }  
    </script>  
</head>  
<body>  
    <dl>  
        <dt>User Name</dt>  
        <dd><input type="text" onkeyup="VerifyUser()" onblur="VerifyRequired()" id="UserName"></dd>  
        <dd id="UserError"></dd>  
    </dl>  
</body>  
</html>  
  
**3. charCodeAt()**        : It returns the code of character, which is ASCII code.  
                            A = 65, Z=90  
  
                            var name = "Ajay";  
                            name.charCodeAt(0);            65  
                            name.charAt(0);                    A  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function VerifyUser(){  
            var username = document.getElementById("UserName").value;  
            var UserError = document.getElementById("UserError");  
            if(username.length<4) {  
                UserError.innerHTML = "Name too short min 4 chars required".fontcolor('red');  
            }else if(username.length>10) {  
                UserError.innerHTML = "Name too long max 10 chars only".fontcolor('red');  
            }  
            else {  
                UserError.innerHTML = "";  
            }  
        }  
        function VerifyRequired(){  
            var username = document.getElementById("UserName").value;  
            var userError = document.getElementById("UserError");  
            if(username.charCodeAt(0)>=65 && username.charCodeAt(0)<=90) {  
                userError.innerHTML = "";  
            } else {  
                userError.innerHTML = "Name must start with uppercase letter".fontcolor('red');  
            }  
        }  
    </script>  
</head>  
<body>  
    <dl>  
        <dt>User Name</dt>  
        <dd><input type="text" onkeyup="VerifyUser()" onblur="VerifyRequired()" id="UserName"></dd>  
        <dd id="UserError"></dd>  
    </dl>  
</body>  
</html>  
  
**4. startsWith()**    : It returns boolean true if the given string starts with specified  
                              chars.  
  
**5. endsWith()**            : It returns true if the given string ends with specified chars.  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <link rel="stylesheet" href="../node\_modules/bootstrap/dist/css/bootstrap.css">  
    <script>  
        function VerifyCard(){  
            var card = document.getElementById("Card").value;  
            var pic = document.getElementById("pic");  
            if(card.startsWith("45")) {  
                pic.src= "../public/images/visa.png";  
            } else if (card.startsWith("54")) {  
                pic.src="../public/images/master.png";  
            } else {  
                pic.src="";  
                pic.alt="N/A";  
            }  
        }  
        function VerifySkype(){  
            var skype = document.getElementById("Skype").value;  
            var error = document.getElementById("SkypeError");  
            if(skype.endsWith("outlook.com")) {  
                error.innerHTML = "";  
            } else {  
                error.innerHTML = "Invalid Skype Id".fontcolor('red');  
            }  
        }  
    </script>  
</head>  
<body class="container-fluid">  
    <dl>  
        <dt>Your Card Number</dt>  
        <dd class="input-group"><input onkeyup="VerifyCard()" type="text" id="Card"><img id="pic" width="50"></dd>  
        <dt>Your Skype Id</dt>  
        <dd>  
            <input type="text" onblur="VerifySkype()" placeholder="outlook.com" id="Skype">  
        </dd>  
        <dd id="SkypeError"></dd>  
    </dl>  
</body>  
</html>  
  
**6. trim()**         : It is used to remove leading spaces in a string.  
                          [space before and after string]  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <link rel="stylesheet" href="../node\_modules/bootstrap/dist/css/bootstrap.css">  
    <script>  
        function VerifyCard(){  
            var card = document.getElementById("Card").value;  
            var pic = document.getElementById("pic");  
            if(card.startsWith("45")) {  
                pic.src= "../public/images/visa.png";  
            } else if (card.startsWith("54")) {  
                pic.src="../public/images/master.png";  
            } else {  
                pic.src="";  
                pic.alt="N/A";  
            }  
        }  
        function VerifySkype(){  
            var skype = document.getElementById("Skype").value;  
            var error = document.getElementById("SkypeError");  
            if(skype.endsWith("outlook.com")) {  
                error.innerHTML = "";  
            } else {  
                error.innerHTML = "Invalid Skype Id".fontcolor('red');  
            }  
        }  
        function VerifyId(){  
            var userid = document.getElementById("UserId").value;  
            var userError = document.getElementById("UserError");  
            if(userid.trim()=="john\_nit") {  
                userError.innerHTML = "";  
            } else {  
                userError.innerHTML = "Invalid User Id".fontcolor('red');  
            }  
        }  
    </script>  
</head>  
<body class="container-fluid">  
    <h2>Register</h2>  
    <dl>  
        <dt>User Id</dt>  
        <dd>  
            <input type="text" onblur="VerifyId()" id="UserId">  
        </dd>  
        <dd id="UserError"></dd>  
        <dt>Your Card Number</dt>  
        <dd class="input-group"><input onkeyup="VerifyCard()" type="text" id="Card"><img id="pic" width="50"></dd>  
        <dt>Your Skype Id</dt>  
        <dd>  
            <input type="text" onblur="VerifySkype()" placeholder="outlook.com" id="Skype">  
        </dd>  
        <dd id="SkypeError"></dd>  
    </dl>  
</body>  
</html>  
  
**7. split()**     :  It splits a string using any specified delimeter [separator] and returns  
                   an array of string.  
  
Ex:  
<script>  
    var contacts = "john-9983382838, david-9948288323, sam-99494992922";  
    var [john, david, sam] = contacts.split(',');  
    document.write(david);  
</script>  
  
Ex:  
<script>  
    var topics = "HTML:It is a markup language | CSS:It defines styles | JavaScript: It is a language";  
    var [html, css, js] =  topics.split('|');  
    document.write(js);  
</script>  
  
  
**8. slice()**                    : It defines chars to read between start  and end index.  
**9. substr()**                : It defines specified number of chars from start index.  
**10. substring()**       : It is specified number of chars from start in bi-directional.  
  
  
<script>  
    var msg = "Welcome to JavaScript";  
    document.write(msg.slice(7,14) + "<br>");  
    document.write(msg.substr(7,3)  + "<br>");  
    document.write(msg.substring(7,0));  
</script>

FAQ: What is difference between  
        slice()  
        substr()  
        substring()  
Ans:  
      - slice can read values between start and end index. End index must be a value  
        greater than start index.  
                 
            Syntax:  
                     slice(startIndex, endIndex);  
                     slice(7, 14);  
                     slice(7, 0);            // invalid  
                     slice(7);                // from 7 to end  
  
      - substr can read value from start index to specified number of chars. It is the value  
        greater than "0" only.  
  
            Syntax:  
                    substr(startIndex, numberOfChars);  
                    substr(7, 3);                // from 7 index 3 chars.  
                    substr(7,0);                // empty  
                    substr(7);                    // from 7 to end  
  
      - substring can read bi-directionaly from specified index.  
  
            Syntax:  
                    substring(startIndex, numberOfChars|index);  
                    substring(7, 14);             7 to 14  
                    substring(7,0);             7 to 0 index [left]  
                    substring(7);                 7 to end  
  
**11. indexOf()**  : It returns the index number of char. You can search for any  
                          character in a string using indexOf.  
                          It returns the first occurance index number.  
                          If the char not found, then it returns "-1".  
  
                        Syntax:  
                            var msg = "Welcome";  
                            msg.indexOf("e");            1  
                            msg.indexOf("a");            -1  
  
**12. lastIndexOf()**   : It returns the last occurance index number of specified char.  
  
                        Syntax:  
                            var msg = "Welcome";  
                            msg.lastIndexOf("e");        7  
                            msg.lastIndexOf("a");        -1  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function VerifyEmail(){  
            var email = document.getElementById("Email").value;  
            var error = document.getElementById("Error");  
            var dotpos = email.lastIndexOf(".");  
            var domain = email.substring(dotpos+1);  
            if(email.indexOf("@")==-1) {  
                error.innerHTML = "Invalid Email- Please include @".fontcolor('red');  
            } else {  
                error.innerHTML="";  
            }  
            document.getElementById("domain").innerHTML = `Your Domain extention is : ${domain}`;  
        }  
    </script>  
</head>  
<body>  
    <dl>  
        <dt>Your Email</dt>  
        <dd>  
            <input type="text" onblur="VerifyEmail()" id="Email">  
        </dd>  
        <dd id="Error"></dd>  
        <dd id="domain"></dd>  
    </dl>  
</body>  
</html>  
  
  
**13. match()**    :  It is used to match your value against a "Regular Expression" &  
                           return true if it is valid. [boolean]  
  
                           The regular expression in JavaScript is enclosed in "/  /".  
                            It is created by using meta characters and quantifiers.  
  
Syntax:  
        var value = "your value";  
        var regExp = /your expression/;  
  
        if(value.match(regExp))  
        {  
        }  
  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>String Match</title>  
    <link rel="stylesheet" href="../node\_modules/bootstrap/dist/css/bootstrap.css">  
    <style>  
        #flag {  
            position: absolute;  
            top: 60px;  
            right: 500px;  
        }  
    </style>  
    <script>  
        var regExp = / /;  
        function CountryChange(){  
            var countryName = document.getElementById("lstCountry").value;  
            var flag = document.getElementById("flag");  
            var mobileTextBox = document.getElementById("txtMobile");  
            var error = document.getElementById("error");  
  
            if(countryName=="India") {  
                flag.src="../public/images/india.png";  
                mobileTextBox.placeholder = "+91 and 10 digits";  
                regExp = /\+91\d{10}/;  
                error.innerHTML = "";  
  
            } else if (countryName=="US") {  
                flag.src="../public/images/us.png";  
                mobileTextBox.placeholder = "+(1)(425) 555-0100";  
                regExp = /\+\(1\)\(\d{3}\)\s\d{3}-\d{4}/;  
                error.innerHTML = "";  
  
            } else if (countryName=="UK") {  
                flag.src="../public/images/uk.png";  
                mobileTextBox.placeholder = "+(44)(20) 1234 5678";  
                regExp = /\+\(44\)\(\d{2}\)\s\d{4}\s\d{4}/;  
                error.innerHTML = "";  
  
            } else {  
                error.innerHTML = "Please Select your country".fontcolor('red');  
                flag.src="";  
                mobileTextBox.placeholder="Country not selected";  
            }  
        }  
        function VerifyMobile(){  
            var mobileNumber = document.getElementById("txtMobile").value;  
            var mobileError = document.getElementById("mobileError");  
            if(mobileNumber.match(regExp)) {  
                document.write("<h2>Mobile Verified Successfully..</h2>");  
            } else {  
                mobileError.innerHTML= `Invalid Mobile : <b>${document.getElementById("txtMobile").placeholder}</b>`.fontcolor('red');  
            }  
        }  
    </script>  
</head>  
<body class="container-fluid">  
    <img id="flag" width="100" height="100" class="rounded rounded-circle">  
    <h2>Verify Mobile Number</h2>  
    <dl class="w-25">  
        <dt>Select Your Country</dt>  
        <dd>  
            <select onchange="CountryChange()" id="lstCountry" class="form-select">  
                <option>Select Country</option>  
                <option>India</option>  
                <option>US</option>  
                <option>UK</option>  
            </select>  
        </dd>  
        <dd id="error"></dd>  
        <dt>Your Mobile Number</dt>  
        <dd>  
            <input type="text" id="txtMobile" class="form-control">  
        </dd>  
        <button onclick="VerifyMobile()" class="btn btn-primary w-100">Submit</button>  
    </dl>  
    <p class="mt-4 text-center" id="mobileError"></p>  
</body>  
</html>  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <link rel="stylesheet" href="../node\_modules/bootstrap/dist/css/bootstrap.css">  
    <script>  
        function VerifyPassword(){  
            var password = document.getElementById("txtPwd").value;  
            var regExp = /(?=.\*[A-Z])\w{4,10}/;  
            var error = document.getElementById("error");  
            var status = document.getElementById("status");  
  
            function SetStatus(min, max, value, low, high){  
                status.min = min;  
                status.max = max;  
                status.value = value;  
                status.low = low;  
                status.high = high;  
            }  
  
  
            if(password.match(regExp)){  
                error.className = "text-success";  
                error.innerHTML = "Strong Password";  
                SetStatus(1,100,100,0,0);  
            } else {  
                if(password.length<4) {  
                    error.className = "text-danger";  
                    error.innerHTML = "Poor Password - min 4 chars required";  
                    SetStatus(1,100,100,60,80);  
                } else {  
                    error.className = "text-warning";  
                    error.innerHTML = "Weak Password - atleast one uppercase required";  
                    SetStatus(1,100,100,40,80);  
                }  
            }  
        }  
    </script>  
    <style>  
        meter {  
            width: 190px;  
            height: 20px;  
        }  
    </style>  
</head>  
<body class="container-fluid">  
    <h3>Verify Password</h3>  
    <dl>  
        <dt>Password</dt>  
        <dd>  
            <input type="password" onkeyup="VerifyPassword()" id="txtPwd">  
            <div>  
                <meter id="status"></meter>  
            </div>  
        </dd>  
        <dd id="error"></dd>  
    </dl>  
</body>  
</html>  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <link rel="stylesheet" href="../node\_modules/bootstrap/dist/css/bootstrap.css">  
    <script>  
        function VerifyPassword(){  
            var password = document.getElementById("txtPwd").value;  
            var regExp = /(?=.\*[A-Z])\w{4,10}/;  
            var error = document.getElementById("error");  
            var status = document.getElementById("status");  
  
            function SetStatus(min, max, value){  
                status.min = min;  
                status.max = max;  
                status.value = value;  
               
            }  
  
  
            if(password.match(regExp)){  
                error.className = "text-success";  
                error.innerHTML = "Strong Password";  
                SetStatus(1,100,100);  
            } else {  
                if(password.length<4) {  
                    error.className = "text-danger";  
                    error.innerHTML = "Poor Password - min 4 chars required";  
                    SetStatus(1,100,20);  
                } else {  
                    error.className = "text-warning";  
                    error.innerHTML = "Weak Password - atleast one uppercase required";  
                    SetStatus(1,100,60);  
                }  
            }  
        }  
    </script>  
    <style>  
        meter {  
            width: 190px;  
            height: 20px;  
        }  
    </style>  
</head>  
<body class="container-fluid">  
    <h3>Verify Password</h3>  
    <dl>  
        <dt>Password</dt>  
        <dd>  
            <input type="password" onkeyup="VerifyPassword()" id="txtPwd">  
            <div>  
                <meter id="status"></meter>  
            </div>  
        </dd>  
        <dd id="error"></dd>  
    </dl>  
</body>  
</html>

3. Boolean Type  
- Boolean value is used in decision making.  
- JavaScript boolean value can be "true or false". But you can verify by using 1 or 0.  
  
                var  stock = true | false;  
  
                if(stock==1)            true = 1,   false = 0  
  
                true + true = ?        2  
  
                var   stock = 1;  
                if(stock==true)            // invalid  
  
  
Ex:  
<script>  
     var stock = false;  
     if(stock==1) {  
        document.write("stock is true");  
     } else {  
        document.write("stock is false");  
     }  
</script>  
  
- **Various HTML element attributes depend on boolean type**  
            readonly      
            disabled  
            required  
            checked  
            selected  
    default value of boolean properties is "true".  
  
     
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function VerifyName(){  
            var username = document.getElementById("UserName");  
            var btnSubmit = document.getElementById("btnSubmit");  
            if(username.value=="") {  
                btnSubmit.disabled = true;  
                username.style.border = "1px solid red";  
            } else {  
                btnSubmit.disabled = false;  
                username.style.border = "1px solid green";  
            }  
        }  
    </script>  
</head>  
<body>  
    <dl>  
        <dt>User Name</dt>  
        <dd><input type="text" onblur="VerifyName()" id="UserName"></dd>  
    </dl>  
    <button id="btnSubmit">Submit</button>  
</body>  
</html>  
  
**FAQ: How to design a Toggle?**  
Ans : Toggle refers to actions performed when same element is switched in control.  
        The logical conditions and decision making statements are required to design toggle.  
            Ex:    if(condition){ }  
                   (condition)?true:false        => Ternary Operator  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function StockChange(){  
            var StockCheckBox = document.getElementById("Stock");  
            var lblStock = document.getElementById("lblStock");  
            console.log(StockCheckBox.checked);  
            if(StockCheckBox.checked) {  
                lblStock.innerHTML = "Available";  
            } else {  
                lblStock.innerHTML = "Out of Stock";  
            }  
        }  
        function SortClick(){  
            var sortIcon = document.getElementById("sortIcon");  
            sortIcon.className = (sortIcon.className=="bi bi-sort-alpha-down")?"bi bi-sort-alpha-up":"bi bi-sort-alpha-down";  
        }  
    </script>  
    <link rel="stylesheet" href="../node\_modules/bootstrap-icons/font/bootstrap-icons.css">  
</head>  
<body>  
    <input type="checkbox" onchange="StockChange()" id="Stock"> <span id="lblStock">Out of Stock</span>  
    <br><br>  
    <button onclick="SortClick()">  
        <span id="sortIcon" class="bi bi-sort-alpha-down"></span>  
    </button>  
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function ToggleClick(){  
            var toggleButton = document.getElementById("toggleButton");  
            toggleButton.innerHTML = (toggleButton.innerHTML=="Show")?"Hide":"Show";  
  
            var preview = document.getElementById("preview");  
            preview.style.display = (preview.style.display=="none")?"block":"none";  
        }  
    </script>  
</head>  
<body>  
    <div>  
        <button id="toggleButton" onclick="ToggleClick()">Show</button>  
    </div>  
    <div id="preview" style="display:none; margin-top: 20px;">  
        <img src="../public/images/asp.jpg" width="200" height="200">  
    </div>  
</body>  
</html>  
  
**FAQ: How to parse string into boolean?**  
Ans : Explicitly you have to convert by using conditional statements or ternary operator.  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function SubmitClick(){  
            var choice = document.getElementById("Choice").value;  
            var c = (choice=="true")?true:false;  
            if(c==true) {  
                document.write("You Selected Yes");  
            } else {  
                document.write("You Selected No");  
            }  
        }  
    </script>  
</head>  
<body>  
    Your Choice :  
    <select id="Choice">  
        <option value="true">Yes</option>  
        <option value="false">No</option>  
    </select>  
    <button onclick="SubmitClick()">Submit</button>  
  
</body>  
</html>

4. Undefined Type  
- It specifies that there is no value defined in a variable.  
- You can verify using "undefined" keyword.  
  
Ex: Bad Code  
<script>  
    var Name = "Samsung TV";  
    var Price;  
    if(Price==undefined){  
        document.write("Name=" + Name);  
    } else {  
    document.write(`  
        Name = ${Name} <br>  
        Price= ${Price}  
    `);  
    }  
</script>  
  
Ex: Good Code  
  
<script>  
    var Name = "Samsung TV";  
    var Price =45000.33;  
    if(Price) {  
        document.write(`  
           Name =  ${Name} <br>  
           Price = ${Price}  
        `);  
    } else {  
        document.write(`Name = ${Name}`);  
    }  
</script>

FAQ: What is diffference between "undefined" and "not defined"?  
Ans : undefined specifies  that reference variable is available but value is not defined.  
        no-defined specifies that there is no reference  variable.  
  
Ex:  
<script>  
    var x;  
    document.write(`  
       X = ${x}  <br>                     x is undefined  
       Y = ${y}                            y is not defined  
    `);  
</script>

5. Null Type  
- If a value is expected into variable during compile time then it is used as undefined.  
- If a value is expected into variable during run time then it is used as null.  
  
                var x;                                         // undefined  
                var y = prompt("Enter Number");        // null  
  
- You can use "null" keyword to verify the value at runtime.  
  
Ex:  
<script>  
    var uname = prompt("Enter Name");  
    if(uname==null){  
        document.write("Please provide a name");  
    } else {  
    document.write(`Name=${uname}`);  
    }  
</script>

6. Symbol  
- It is a new type introduced into JavaScript from ES6.  
- It is used to defined a unique reference type for object.  
- It symbol type configure a member, which is hidden and not accesible using iterations.  
  
  
Ex: Input type hidden  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
</head>  
<body>  
    <form>  
        <dl>  
            <dd>  
                <input type="hidden" name="Id" value="1">  
            </dd>  
            <dt>Name</dt>  
            <dd><input type="text" name="Name"></dd>  
            <dt>Price</dt>  
            <dd><input type="text" name="Price"></dd>  
        </dl>  
        <button>Submit</button>  
    </form>  
</body>  
</html>  
  
Ex:  
<script>  
      var Id = Symbol();  
      var product = {  
         [Id]: 1,  
         Name: "TV",  
         Price: 45000.33,  
         Stock: true  
      }  
      for(var property in product) {  
        document.write(`${property} : ${product[property]}<br>`);  
      }  
      document.write("Id : " + product[Id]);  
</script>  
  
  
Non Primitive Types  
- They are mutable types.  
- Their structure can change dynamically.  
- They don't have fixed range for values.  
- Value range varies according to memory available.  
- JavaScirpt non-primitive types are  
        a) Array  
        b) Object  
        c) Map

Array  
- Arrays are used in computer programming to reduce overhead and complexity.  
- Array can store values in a sequential order, hence they can reduce overhead.  
- Array refers to a formation type, where you can store in sequential order, but can access in random.  
- Several technologies can't allocate sequential memory for different types, hence you can store only similar types.  
- JavaScript array can store various types of values.  
- Array size can change dynamically in JS.

Array  
- Arrays are used to reduce overhead and complexity  
- Array will reduce overhead by storing values in sequential order.  
- Array will reduce complexity by storing multiple values under one name.  
- JavaScript array can handle various types of values, where as array is restricted to similar type of values in other technologies like : C++, Java, C, C#, etc..  
- JavaScript array size can change dynamically, which is not possible in other programming languages.  
- Array refers to a formation, where contents are sequentially arranged but are accessed random.  
  
**1. Declaring Array**  
 - It is same as you declare any other variable.  
 - It is declared by using var, let or const keywords.  
  
        var  products;  
        var  categories;  
  
 - Array declaration is not enough to store data.  
  
  
**2. Initialize or assign memory for array**  
- Memory for array can be initialized or assigned by using 2 techniques  
        ***a) By using array meta character "[ ]"  
        b) By using array constructor "new Array()"***  
  
Syntax: Meta Character  
  
            var products = [];                initialization  
  
            var products;  
            products = [ ];                assignment  
  
                                (or)  
  
Syntax: Array Contructor  
  
            var products = new Array();        initialization  
  
            var products;  
            products = new Array();                assignment  
  
Note:  "new" is dynamic memory allocation operator.  
  
        [ ]                    - contineous memory  
        new Array()        - discreet memory  
  
**3. Storing data into array**- You can initialize values directly into array.  
  
        var collection = [10, "john", true];  
        var collection = new Array(10, "john", true);  
  
- You can assign values into array with reference of property that maps to index number.  
  
        collection[0]    = 10;              
        collection["1"] = 20;  
        collection[2] = 30;  
  
- Every property is  string type :   ["0"], [0] => string  
  
Ex:  
<script>  
      var collection = [];  
      collection[0] = 10;  
      collection[1] = "jonn";  
      collection[2] = true;  
      collection["3"] = 30;  
      for(var property in collection){  
            document.write(`[${property}](${typeof property})-${collection[property]}(${typeof collection[property]})<br>`);  
      }  
</script>  
  
- **You can store any type of data in array memory**        a) Primitive  
        b) Non Primitive  
        c) function  
  
Ex:  
<script>  
     var collection = [10, "John", true, ["Delhi", "Hyd"], function(){document.write("Hello from Array")}];  
     document.write(collection[3][1] + "<br>");  
     collection[4]();  
</script>  
  
Ex:      
        var collection = [];  
        collection[0] = 10;  
        collection[1] = 20;  
        collection[3] = 30;  
  
        document.write(collection[2]);                // undefined  
  
- JavaScript ES6+ versions introduced  Array de-structuring technique.  
- It is the process of separating the values of an array and storing in inidividual references.  
  
Ex:  
<script>  
     var collection = [1, "Samsung TV", 46000.44, true, ["Delhi", "Hyd"]];  
     var [id, name, price, stock, cities] = collection;  
       
     document.write(`  
            Id  : ${id} <br>  
            Name: ${name} <br>  
            Price: ${price}<br>  
            Stock: ${stock} <br>  
            Cities: ${cities}  
     `);  
</script>

JavaScript provides various methods to manipulate array  
1. Reading values from array  
        toString()  
        join()  
        slice()  
        find()  
        filter()  
        map()           : It is an iterator that reads all elements from collection in sequential order and returns using a markup.  
  
Ex:  
<script>  
    var collection = [23000, 61000, 21000, 63100, 35000];  
    document.write(collection.filter(function(value){  
        return value>60000;  
    }));  
</script>

Ex:  
<script>  
      var categories = ["Electronics", "Footwear", "Fashion"];  
      categories.map(function(value){  
            document.write(`<a href=#>${value}</a><br>`);  
      });  
</script>  
 **for loops for reading**  
- A loop requires initializaiton, condition and counter.  
  
    for(var i=0;  i<collection.length; i++)  
    {  
    }  
  
Ex:  
<script>  
      var categories = ["Electronics", "Footwear", "Fashion"];  
      for(var i=0; i<categories.length; i++)  
      {  
         document.write(`<button>${categories[i]}</button><br>`)  
      }  
</script>  
 **for.. iterators**  
- Iterator doesn't require initialization, codition and counter.  
  
Syntax: of operator is used for reading all values.  
  
         for(var item of collection)  
        {  
        }  
  
Syntax: in  operator is used for reading all properties  
  
        for(var property in collection)  
        {  
        }  
  
Ex:  
<script>  
      var categories = ["Electronics", "Footwear", "Fashion"];  
      for(var property in categories)  
      {  
         document.write(`${property} : ${categories[property]}<br>`);  
      }  
</script>  
  
**Presenting Values in HTML page:**  
  
1. You have to create a new HTML element  
  
        document.createElement("p | div | img | table | select | option..");  
  
2. Set properties for element  
  
        var pic = document.createElement("img");  
        pic.src = ""  
        pic.width=""  
        pic.alt=""  
  
3. Add element into page  
         
         ***appendChild()***  
        <body>  
  
        document.querySelector("body").appendChild(pic);  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function bodyload(){  
            var img = document.createElement("img");  
            img.src = "../public/images/a1.jpg";  
            img.width = "100";  
            img.height= "100";  
            document.querySelector("div").appendChild(img);  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <div>  
  
    </div>  
</body>  
</html>  
  
Ex: Creating Dropdown Dynamically  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function bodyload(){  
             var select = document.createElement("select");  
             var option1 = document.createElement("option");  
             var option2 = document.createElement("option");  
             option1.text = "Delhi";  
             option2.text = "Hyd";  
             select.appendChild(option1);  
             select.appendChild(option2);  
             document.querySelector("div").appendChild(select);  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <div>  
  
    </div>  
</body>  
</html>  
  
  
Ex: Creating list using map()  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var categories = ["All", "Electronics", "Footwear", "Fashion"];  
        function bodyload(){  
            categories.map(function(category){  
                  var li = document.createElement("li");  
                  li.innerHTML = category;  
                  document.querySelector("ol").appendChild(li);  
            })  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <ol>  
  
    </ol>  
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var categories = ["All", "Electronics", "Footwear", "Fashion", "Kids"];  
        function bodyload(){  
           for(var category of categories)  
           {  
                  var li = document.createElement("li");  
                  li.innerHTML = category;  
                  document.querySelector("ol").appendChild(li);  
           }  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <ol>  
  
    </ol>  
</body>  
</html>  
  
Ex: Checkbox List  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var categories = ["All", "Electronics", "Footwear", "Fashion", "Kids"];  
        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;  
                  option.value = 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);  
  
                  var ulLi = document.createElement("li");  
                  var input = document.createElement("input");  
                  input.type = "checkbox";  
                  var label = document.createElement("label");  
                  label.innerHTML = category;  
  
                  ulLi.appendChild(input);  
                  ulLi.appendChild(label);  
                  document.querySelector("ul").appendChild(ulLi);  
           }  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <ol>  
  
    </ol>  
    <select>  
  
    </select>  
    <br><br>  
    <table border="1" width="300">  
        <thead>  
            <tr>  
                <th>Categories</th>  
            </tr>  
        </thead>  
        <tbody>  
  
        </tbody>  
    </table>  
    <br><br>  
    <ul>  
  
    </ul>  
</body>  
</html>  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var categories = ["All", "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;  
                  option.value = 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);  
  
                  var ulLi = document.createElement("li");  
                  ulLi.innerHTML = `<input type="checkbox"> <label>${category}</label>`;  
                  document.querySelector("ul").appendChild(ulLi);  
           }  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <ol>  
  
    </ol>  
    <select>  
  
    </select>  
    <br><br>  
    <table border="1" width="300">  
        <thead>  
            <tr>  
                <th>Categories</th>  
            </tr>  
        </thead>  
        <tbody>  
  
        </tbody>  
    </table>  
    <br><br>  
    <ul>  
  
    </ul>  
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var categories = ["All", "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;  
                  option.value = 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);  
  
                  var ulLi = document.createElement("li");  
                  ulLi.innerHTML = `<input type="checkbox"> <label>${category}</label>`;  
                  document.querySelector("ul").appendChild(ulLi);  
  
                  var anchor = document.createElement("a");  
                  anchor.href="#";  
                  anchor.innerHTML = category;  
                  document.querySelector("nav").appendChild(anchor);  
                  var br = document.createElement("br");  
                  document.querySelector("nav").appendChild(br);  
           }  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <nav>  
  
    </nav>  
    <ol>  
  
    </ol>  
    <select>  
  
    </select>  
    <br><br>  
    <table border="1" width="300">  
        <thead>  
            <tr>  
                <th>Categories</th>  
            </tr>  
        </thead>  
        <tbody>  
  
        </tbody>  
    </table>  
    <br><br>  
    <ul>  
  
    </ul>  
</body>  
</html>  
  
2. Adding elements into Array  
        push()                : It adds new element(s) as last element.  
        unshift()            : It adds new element(s) as first element.  
        splice()            : It adds new element(s) at specified position.  
  
    Syntax:  
        collection.push("item1", "item2",...);  
        collection.unshift("item1", "item2",...);  
        collection.splice(1,0,"item1", "item2",...);  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var categories = ["All", "Electronics", "Footwear", "Fashion"];  
        function bodyload(){  
            categories.splice(1,0,"Men's Clothing","Kid's Fashion");  
           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;  
                  option.value = 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);  
  
                  var ulLi = document.createElement("li");  
                  ulLi.innerHTML = `<input type="checkbox"> <label>${category}</label>`;  
                  document.querySelector("ul").appendChild(ulLi);  
  
                  var anchor = document.createElement("a");  
                  anchor.href="#";  
                  anchor.innerHTML = category;  
                  document.querySelector("nav").appendChild(anchor);  
                  var br = document.createElement("br");  
                  document.querySelector("nav").appendChild(br);  
           }  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <nav>  
  
    </nav>  
    <ol>  
  
    </ol>  
    <select>  
  
    </select>  
    <br><br>  
    <table border="1" width="300">  
        <thead>  
            <tr>  
                <th>Categories</th>  
            </tr>  
        </thead>  
        <tbody>  
  
        </tbody>  
    </table>  
    <br><br>  
    <ul>  
  
    </ul>  
</body>  
</html>

3. Removing Elements from Array  
  
        pop()                : It removes and returns the last element.  
        shift()                : It removes and returns the first element.  
        splice()            : It removes and returns the specified element.  
  
Syntax:  
        collection.pop()  
        collection.shift()  
        collection.splice(indexNumber, deleteCount)  
  
4. Sorting Array Elements  
        sort()                : It arranges elements in ascending order  
        reverse()            : It arranges in reverse order  
  
Syntax:  
        collection.sort()  
        collection.reverse()  
  
**FAQ: How to empty array?**  
Ans:  
         var cities = ["Delhi", "Hyd"];  
        cities = [];  
        cities.length = 0;  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>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 cities = ["Delhi","Hyd","Chennai"];  
        function LoadCities(){  
            document.getElementById("lstCities").innerHTML = "";  
            for(var city of cities)  
            {  
                var option = document.createElement("option");  
                option.text = city;  
                option.value = city;  
                document.getElementById("lstCities").appendChild(option);  
            }  
            document.getElementById("lblCount").innerHTML = `Total No. of Cities : ${cities.length}`;  
        }  
        function bodyload(){  
            LoadCities();  
        }  
        function AddClick(){  
            var cityName = document.getElementById("txtCity").value;  
            if(cities.indexOf(cityName)==-1) {  
                cities.push(cityName);  
                alert(`${cityName} Added to List`);  
                LoadCities();  
                document.getElementById("txtCity").value = "";  
            } else {  
                alert(`${cityName} Exists`);  
            }  
        }  
        function SortAsc(){  
            cities.sort();  
            LoadCities();  
        }  
        function SortDesc(){  
            cities.sort();  
            cities.reverse();  
            LoadCities();  
        }  
        function RemoveClick(){  
            var selectedCityName = document.getElementById("lstCities").value;  
            var selectedCityIndex = cities.indexOf(selectedCityName);  
            var flag = confirm(`Delete ${selectedCityName}\nAre you sure want to delete?`);  
            if(flag==true){  
                cities.splice(selectedCityIndex,1);  
                LoadCities();  
            }  
        }  
        function ClearClick(){  
            cities = [];  
            LoadCities();  
        }  
        function SearchClick(){  
            var cityName = document.getElementById("txtSearch").value;  
            if(cities.indexOf(cityName)==-1){  
                alert(`${cityName} Not Found`);  
            } else {  
                var searchedCity = cities.indexOf(cityName);  
                 
            }  
        }  
    </script>  
</head>  
<body class="container-fluid" onload="bodyload()">  
    <div class="mt-3 w-25 bg-dark text-white p-3">  
        <div class="mt-3 mb-3">  
            <div class="input-group">  
                <input type="text" placeholder="Search city" class="form-control" id="txtSearch">  
                <button onclick="SearchClick()" class="btn btn-warning">  
                    <span class="bi bi-search"></span>  
                </button>  
            </div>  
        </div>  
        <div class="mt-3 mb-2">  
            <label class="form-label fw-bold">New City</label>  
            <div class="input-group">  
                <input type="text" id="txtCity" class="form-control">  
                <button onclick="AddClick()" class="btn btn-primary">Add</button>  
            </div>  
        </div>  
        <div>  
            <label class="form-label fw-bold">Cities List</label>  
            <div class="mt-2 mb-2">  
                <button class="btn btn-success" onclick="SortAsc()">  
                    <span class="bi bi-sort-alpha-down"></span>  
                </button>  
                <button class="btn btn-success" onclick="SortDesc()">  
                    <span class="bi bi-sort-alpha-up"></span>  
                </button>  
            </div>  
            <select size="3" class="form-select" id="lstCities">  
  
            </select>  
            <div class="mt-3">  
                <label id="lblCount"></label>  
            </div>  
            <div class="mt-3">  
                <button onclick="RemoveClick()" class="btn btn-danger">  
                    <span class="bi bi-trash-fill"></span> Remove City  
                </button>  
                <button class="btn btn-danger" onclick="ClearClick()">  
                    <span class="bi bi-trash2"></span> Clear All  
                </button>  
            </div>  
        </div>  
    </div>  
</body>  
</html>  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function bodyload(){  
            var listbox = document.querySelector("select");  
            listbox.options[2].selected = true;  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <select size="5" style="width:100px">  
        <option>Delhi</option>  
        <option>Hyd</option>  
        <option>Chennai</option>  
    </select>  
</body>  
</html>

JavaScript Object Type  
  
- Object is used in computer programming to keep all related data and logic together.  
- Object was introduced into computer programming by "Alan Kay" in early 1960's.  
- Object intention is to encapsulate all data and logic into one unit.  
  
            var tv = "Samsung TV";  
            var mobile = "Realme";  
            var price = 35000.44;  
  
- To keep all realeted data you have to design a collection of Keys and Values  
  
            {  
                Name: "Samsung TV",  
                Price: 35000.44  
            }  
  
- Object can contain both data and logic  
     
            {  
              Name: "Samsung TV",  
              Price: 32000.34,  
              Total: function() {  
  
               }  
             }  
  
- If object comprises of only Data, If it is mapping to data only then it is called as "JSON"  
  [JavaScript Object Notation]  
  
- The popular data formats understandable to every device are:  
  
            XML             => Extensible Markup Language  
            JSON            => JavaScript Object Notation  
  
- JavaScript object comprises of Keys and Values collection.  
- Every Key is a string type.  
- Value can be any type. [Primitive, Non-Primitive]  
- If you want to store object data in separate file then it must have extention ".json".  
  
Syntax:  
            var  product = {  
                    "Name": "TV",  
                    "Price": 46000.33,  
                    "Stock": true  
            }  
            product.Name  
            product.Price  
            product.Stock  
  
Ex:  
1. Add a new folder  
        "data"  
  
2. Add a new file  
  
        "product.json"  
  
{  
    "title": "realme C33 (Sandy Gold, 64 GB)  (4 GB RAM)",  
    "rating": {"rate":4.3, "ratings":64500,"reviews":5000},  
    "price": 10099,  
    "offers": [  
        "Bank Offer 5% Cashback on Flipkart Axis Bank CardT&C",  
        "Bank Offer 5% Cashback on Flipkart Axis Bank CardT&C",  
        "Special PriceGet extra ₹3000 off (price inclusive of cashback/coupon)T&C",  
        "Partner OfferSign up for Flipkart Pay Later and get Flipkart Gift Card worth up to ₹500\*Know More"  
    ],  
    "photo": "../public/images/realme.jpg"  
}  
  
3. Consume and present in HTML page using "fetch()" promise  
  
Syntax:  
            fetch("url")  
            .then(function(response){  
                return response.json();  
            })  
            .then(function(data){  
                //present..  
            })  
            .catch(function(err){  
                // display err  
            })  
  
product.html  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Flipkart</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 bodyload(){  
            fetch("../data/product.json")  
            .then(function(response){  
                return response.json();  
            })  
            .then(function(product){  
                document.getElementById("title").innerHTML = product.title;  
                document.getElementById("pic").src = product.photo;  
                document.getElementById("rating").innerHTML = `<span class="badge bg-success text-white rounded rounded-pill">${product.rating.rate} <span class="bi bi-star-fill text-white"></span></span> ${product.rating.ratings} Ratings & ${product.rating.reviews} Reviews`;  
                document.getElementById("price").innerHTML = "&#8377; " + product.price;  
                for(var offer of product.offers){  
                    var li = document.createElement("li");  
                    li.className = "bi bi-tag text-success";  
                    li.innerHTML = offer;  
                    document.getElementById("offers").appendChild(li);  
                }  
            })  
            .catch(function(err){  
                alert('Something went wrong: please contact site admin');  
            })  
        }  
    </script>  
    <style>  
        #title {  
            font-size: 24px;  
            font-weight: bold;  
        }  
        li {  
            margin-bottom: 20px;  
        }  
    </style>  
</head>  
<body class="container-fluid" onload="bodyload()">  
    <div class="mt-3 row">  
        <div class="col-3">  
            <img id="pic" width="250" height="300">  
        </div>  
        <div class="col-9">  
            <div id="title"></div>  
            <div id="rating"></div>  
            <h3 id="price"></h3>  
            <div>  
                <h4>Offers</h4>  
                <ul id="offers" class="list-unstyled">  
  
                </ul>  
            </div>  
        </div>  
    </div>  
</body>  
</html>

- JavaScript object comprises of both data and logic.  
- Logic is defined using function.  
  
Syntax:  
            var obj = {  
                    "Key": value,                    => Property  
                    "Key": function() {             => Method  
  
                     }  
             }  
  
- **Object comprises 3 basic elements**  
    a) Property        : To Store data  
    b) Method        : To Define Action  
    c) Event            : To Trigger the Action  
  
- The members of any object in JavaScript can be  
        a) Property  
        b) Method  
- You can access the members of any object within object by using "this" keyword.  
- You can access the members of any object outside object by using object name.  
  
        var product =  {  
            "Name": "TV",  
             "Print": function() {  
                document.write("Name=" + this.Name);  
             }  
        }  
  
        product.Name = "Samsung TV";  
        product.Print();  
  
Ex:  
<script>  
     var product = {  
         "Name":"",  
         "Price":0,  
         "Qty":0,  
         "Total": function(){  
            return this.Qty \* this.Price;  
         },  
         "Print": function(){  
            document.write(`Name=${this.Name}<br>Price=${this.Price}<br>Qty=${this.Qty}<br>Total=${this.Total()}`);  
         }  
     }  
     product.Name = prompt("Enter Product Name");  
     product.Price = parseFloat(prompt("Enter Price"));  
     product.Qty = parseInt(prompt("Enter Qty"));  
     product.Print();  
</script>

Array of Objects  
 Table  
  
  Name            Price  
  ----------------------------  
   TV                45000.33        => Binary  => JSON  
   Mobile            6300.44  
   Shoe            2000.44  
  
JSON  
    var products =  
    [  
        {"Name": "TV", "Price": 45000.33},  
           {"Name": "Mobile", "Price": 6300.44},  
           {"Name": "Shoe", "Price": 2000.44}  
    ]  
  
    Mobile Price :  products[1].Price  
  
Ex:  
1**. Add a new file into data folder  
        "products.json"**[  
    {  
        "Name": "Realme Aqua Blue",  
        "Price": 8999.33,  
        "Photo": "../public/images/realme.jpg"  
    },  
    {  
        "Name":"Realme Gold",  
        "Price":9999.33,  
        "Photo":"../public/images/realmegold.jpg"  
    },  
    {  
        "Name": "Kitchen Accessories",  
        "Price": 3400.33,  
        "Photo": "../public/images/a1.jpg"  
    }  
]  
  
**2. Products.html**  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Products</title>  
    <script>  
        function bodyload(){  
            fetch("../data/products.json")  
            .then(function(response){  
                return response.json();  
            })  
            .then(function(products){  
                products.map(function(product){  
                    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;  
                    tdPhoto.innerHTML = `<img width="100" height="100" src=${product.Photo}>`;  
  
                    tr.appendChild(tdName);  
                    tr.appendChild(tdPrice);  
                    tr.appendChild(tdPhoto);  
  
                    document.querySelector("tbody").appendChild(tr);  
  
                })  
            })  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <h2>Products Table</h2>  
    <table border="1" width="400">  
        <thead>  
            <tr>  
                <th>Name</th>  
                <th>Price</th>  
                <th>Preview</th>  
            </tr>  
        </thead>  
        <tbody>  
  
        </tbody>  
    </table>  
</body>  
</html>  
  
  
Ex: Nasa API    =>  api.nasa.gov  
  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Nasa</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=DEMO_KEY&quot>;)  
            .then(function(response){  
                return response.json();  
            })  
            .then(function(marsObject){  
                 marsObject.photos.map(function(item){  
                      var tr = document.createElement("tr");  
                      var tdId = document.createElement("td");  
                      var tdCamera = document.createElement("td");  
                      var tdPreview = document.createElement("td");  
                      var tdRover = document.createElement("td");  
  
                      tdId.innerHTML = item.id;  
                      tdCamera.innerHTML = item.camera.full\_name;  
                      tdPreview.innerHTML = `<a href=${item.img\_src} target="\_blank"><img width="100" height="100" src=${item.img\_src}></a>`;  
                      tdRover.innerHTML = item.rover.name;  
  
                      tr.appendChild(tdId);  
                      tr.appendChild(tdCamera);  
                      tr.appendChild(tdPreview);  
                      tr.appendChild(tdRover);  
  
                      document.querySelector("tbody").appendChild(tr);  
                 })  
            })  
        }  
    </script>  
</head>  
<body class="container-fluid" onload="bodyload()">  
    <h2>Mars Rover Photos</h2>  
    <table class="table table-hover">  
        <thead>  
            <tr>  
                <th>Photo Id</th>  
                <th>Camera Full Name</th>  
                <th>Preview</th>  
                <th>Rover Name</th>  
            </tr>  
        </thead>  
        <tbody>  
  
        </tbody>  
    </table>  
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Nasa</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=DEMO_KEY&quot>;)  
            .then(function(response){  
                return response.json();  
            })  
            .then(function(marsObject){  
                 marsObject.photos.map(function(item){  
                      var card = document.createElement("div");  
                      card.className = "card p-2 m-2";  
                      card.style.width = "200px";  
                      card.innerHTML = `  
                         <img src=${item.img\_src} class="card-img-top" height="150">  
                         <div class="card-header">  
                            <h2>${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.querySelector("main").appendChild(card);  
                 })  
            })  
        }  
    </script>  
</head>  
<body class="container-fluid" onload="bodyload()">  
    <h2>Mars Rover Photos</h2>  
    <main class="d-flex flex-wrap">  
  
    </main>  
</body>  
</html>

Object Manipulation  
FAQ's :  
**1. How to access the list of all keys in an object?**  
A. You can get the list of keys from any object by using  
***a) for..in  iterator  
        b) Object.keys()***  
  
Syntax:  
        for (var property  in  objectName)  
        {  
        }  
  
Ex:  
<script>  
    fetch("../data/product.json")  
    .then(function(response){  
      return response.json();  
    })  
    .then(function(product){  
        for(var key in product){  
           document.write(key + "<br>");  
        }  
    })  
</script>  
  
Syntax:  
            **Object.keys(objectName).map()**  
  
Ex:  
<script>  
    fetch("../data/product.json")  
    .then(function(response){  
      return response.json();  
    })  
    .then(function(product){  
        Object.keys(product).map(function(key){  
           document.write(`${key}`);  
        })  
    })  
</script>  
  
**2. How to read both keys and values of any object?**  
A. By using object and key reference  
  
Syntax:  
**object[key]        =>  value**  
Ex:  
<script>  
    fetch("../data/product.json")  
    .then(function(response){  
      return response.json();  
    })  
    .then(function(product){  
        Object.keys(product).map(function(key){  
           document.write(`${key}:${product[key]}<br>`);    
        })  
    })  
</script>  
  
**3. How to know the count of keys in object?**  
A. By accessing the length of key using "Object.keys()"  
  
Syntax:  
document.write(Object.keys(product).length);  
  
**4. How to know the data type of value present in a key?**  
A. By using "typeof" operator  
  
Ex:  
<script>  
    fetch("../data/product.json")  
    .then(function(response){  
      return response.json();  
    })  
    .then(function(product){  
        Object.keys(product).map(function(key){  
             document.write(`${key}: ${typeof product[key]}<br>`);  
        })  
    })  
</script>  
  
**5. How to verify the existance of any key in an object?**A.  By using "in" operator  
  
Syntax:  
            "keyName"  in  ObjectName;        => true / false  
  
Ex:  
<script>  
    fetch("../data/product.json")  
    .then(function(response){  
      return response.json();  
    })  
    .then(function(product){  
        var field = prompt('Enter Field Name');  
        document.write(`  
            ${( field in product)?`${field} Available`:`${field} Not Availabe`}  
        `);  
    })  
</script>  
  
  
**6. How to delete any key from object?**  
A. "delete" operator can remove any key from object.  
  
 Syntax:  
            delete   objectName.key  
  
Ex:  
<script>  
    fetch("../data/product.json")  
    .then(function(response){  
      return response.json();  
    })  
    .then(function(product){  
         delete product.price;  
         Object.keys(product).map(function(key){  
            document.write(key + "<br>");  
         })  
    })  
</script>

Nested Iterations and Map()  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
   <meta charset="UTF-8">  
   <meta http-equiv="X-UA-Compatible" content="IE=edge">  
   <meta name="viewport" content="width=device-width, initial-scale=1.0">  
   <title>Document</title>  
   <script>  
        var menu = [  
            {Category:"Electronics", Products: ["Televisions","Mobiles"]},  
            {Category:"Footwear", Products: ["Casuals", "Boots"]}  
        ]  
        function bodyload(){  
             for(var item of menu)  
             {  
                 var li = document.createElement("li");  
                 li.innerHTML = item.Category;  
                 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);  
                     document.querySelector("ol").appendChild(li);  
                 }  
             }  
        }  
   </script>  
</head>  
<body onload="bodyload()">  
   <ol>  
  
   </ol>  
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
   <meta charset="UTF-8">  
   <meta http-equiv="X-UA-Compatible" content="IE=edge">  
   <meta name="viewport" content="width=device-width, initial-scale=1.0">  
   <title>Document</title>  
   <script>  
        var menu = [  
            {Category:"Electronics", Products: ["Televisions","Mobiles"]},  
            {Category:"Footwear", Products: ["Casuals", "Boots"]}  
        ]  
        function bodyload(){  
            menu.map(function(item){  
                  var li = document.createElement("li");  
                  li.innerHTML = item.Category;  
  
                  var optgroup = document.createElement("optgroup");  
                  optgroup.label = item.Category;  
  
                  var details = document.createElement("details");  
                  var summary = document.createElement("summary");  
  
                  summary.innerHTML = item.Category;  
  
                  item.Products.map(function(product){  
                      var ul = document.createElement("ul");  
                      var ulLi = document.createElement("li");  
                      ulLi.innerHTML= product;  
                      ul.appendChild(ulLi);  
                      li.appendChild(ul);  
                      document.querySelector("ol").appendChild(li);  
  
                      var option = document.createElement("option");  
                      option.innerHTML = product;  
                      optgroup.appendChild(option);  
                      document.querySelector("select").appendChild(optgroup);  
  
                      var summaryList = document.createElement("ul");  
                      var summaryListItem = document.createElement("li");  
                      summaryListItem.innerHTML = product;  
                      summaryList.appendChild(summaryListItem);  
                      details.appendChild(summary);  
                      details.appendChild(summaryList);  
                      document.querySelector("nav").appendChild(details);  
                  })  
            })  
        }  
   </script>  
</head>  
<body onload="bodyload()">  
   <ol>  
  
   </ol>  
   <select>  
  
   </select>  
   <div>  
      <nav>  
  
      </nav>  
   </div>  
</body>  
</html>

## Ex- Shopping Example using API

<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Shopping | Online Store</title>  
    <style>  
        header a {  
            color:white;  
            text-decoration: none;  
        }  
    </style>  
    <link rel="stylesheet" href="../node\_modules/bootstrap/dist/css/bootstrap.css">  
    <link rel="stylesheet" href="../node\_modules/bootstrap-icons/font/bootstrap-icons.css">  
    <script type="text/javascript">  
        function LoadProducts(url){  
            document.querySelector("main").innerHTML = "";  
            fetch(url)  
            .then(function(response){  
                return response.json();  
            })  
            .then(function(products){  
                products.map(function(product){  
                     var card = document.createElement("div");  
                     card.className = "card m-2 p-2";  
                     card.style.width = "200px";  
                     card.innerHTML = `  
                        <img src=${product.image} class="card-img-top" height="150">  
                        <div class="card-header" style="height:140px">  
                            <p>${product.title}</p>  
                        </div>  
                        <div class="card-body">  
                            <dl>  
                                <dt>Price</dt>  
                                <dd>${product.price}</dd>  
                                <dt>Rating</dt>  
                                <dd> <span class="bi bi-star-fill text-success"></span> ${product.rating.rate} [${product.rating.count}]</dd>  
                            </dl>  
                        </div>  
                        <div class="card-footer">  
                            <button onclick="AddToCartClick(${product.id})" class="btn btn-danger w-100">  
                                <span class="bi bi-cart3"> </span> Add to Cart  
                            </button>  
                        </div>  
                     `;  
                     document.querySelector("main").appendChild(card);  
                })  
            })  
        }  
        function LoadCategories(){  
            fetch("<http://fakestoreapi.com/products/categories&quot>;)  
            .then(function(response){  
                return response.json();  
            })  
            .then(function(categories){  
                categories.unshift("all");  
                categories.map(function(category){  
                    var option = document.createElement("option");  
                    option.text = category.toUpperCase();  
                    option.value = category;  
  
                    document.getElementById("lstCategories").appendChild(option);  
                })  
            })  
        }  
        function bodyload(){  
            LoadProducts("<http://fakestoreapi.com/products&quot>;);  
            LoadCategories();  
            GetCartCount();  
        }  
        function CategoryChanged(){  
            var categoryName = document.getElementById("lstCategories").value;  
            if(categoryName=="all"){  
                LoadProducts("<http://fakestoreapi.com/products&quot>;);  
            } else {  
                LoadProducts(`<http://fakestoreapi.com/products/category/$>{categoryName}`);  
            }  
        }  
        var cartItems = [];  
        function GetCartCount(){  
            document.getElementById("cartCount").innerHTML = cartItems.length;  
        }  
        function AddToCartClick(id){  
            fetch(`<http://fakestoreapi.com/products/$>{id}`)  
            .then(function(response){  
                return response.json();  
            })  
            .then(function(product){  
                cartItems.push(product);  
                alert(`Add to Cart\n${product.title}\nAdded to Cart`);  
                GetCartCount();  
            })  
        }  
        function LoadCartItems(){  
            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 tdPreview = document.createElement("td");  
  
                tdTitle.innerHTML = item.title;  
                tdPrice.innerHTML = item.price;  
                tdPreview.innerHTML = `<img width="50" height="50" src=${item.image}>`;  
  
                tr.appendChild(tdTitle);  
                tr.appendChild(tdPrice);  
                tr.appendChild(tdPreview);  
  
                document.querySelector("tbody").appendChild(tr);  
            }  
        }  
    </script>  
</head>  
<body class="container-fluid" onload="bodyload()">  
    <header class="d-flex justify-content-between p-3 bg-dark text-white">  
        <div class="fw-bold fs-5">  
            Shopper  
        </div>  
        <div>  
            <span class="me-3"><a href="javascript:LoadProducts('[http://fakestoreapi.com/products')">Home</a></span&gt](http://fakestoreapi.com/products')%22%3EHome%3C/a%3E%3C/span&gt);  
            <span class="me-3"><a href="javascript:LoadProducts('[http://fakestoreapi.com/products/category/electronics')">Electronics</a></span&gt](http://fakestoreapi.com/products/category/electronics')%22%3EElectronics%3C/a%3E%3C/span&gt);  
            <span class="me-3"><a>Jewelery</a></span>  
            <span class="me-3"><a>Men's Clothing</a></span>  
            <span class="me-3"><a>Women's Clothing</a></span>  
        </div>  
        <div>  
            <span class="bi bi-search me-2"></span>  
            <span class="bi bi-heart me-2"></span>  
            <span class="bi bi-person me-2"></span>  
            <button data-bs-toggle="modal" data-bs-target="#cart" onclick="LoadCartItems()" class="position-relative btn btn-warning">  
               <span class="bi bi-cart4"></span> <span class="badge bg-danger rounded rounded-circle position-absolute" id="cartCount"></span>  
            </button>  
            <div class="modal fade" id="cart">  
                <div class="modal-dialog">  
                    <div class="modal-content">  
                        <div class="modal-header">  
                            <h2 class="text-primary">Your Cart Summary</h2>  
                            <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>  
                </div>  
            </div>  
        </div>  
    </header>  
    <section class="mt-3 row">  
        <nav class="col-2">  
            <div>  
                <label class="form-label fw-bold">Select Category</label>  
                <div>  
                    <select onchange="CategoryChanged()" class="form-select" id="lstCategories"></select>  
                </div>  
            </div>  
        </nav>  
        <main class="col-10 d-flex flex-wrap overflow-auto" style="height: 500px;">  
  
        </main>  
    </section>  
    <script src="../node\_modules/jquery/dist/jquery.js"></script>  
    <script src="../node\_modules/bootstrap/dist/js/bootstrap.bundle.js"></script>  
</body>  
</html>

**FAQ: What are the issues with Object?**Ans:  
 - Keys must be only "string" type.  
 - You need explicit iterators for reading keys and values.  
 - You need explicit operators to verify and remove keys.  
 - Object is slow in access and functionality.  
 - Good about object is, It is schema based.

Map Type  
  
- JS introduced Map type from ES5 version  
- Map() is a key and value collection same like object.  
- It provides all implicit methods for manipulations.  
- It is more faster than object.  
- Map() is structure less.  
- Key can be any type.  
  
Syntax:  
            var  data = new Map();  
  
Methods  
    set()                It used to store data with reference of key  
    get()                It used to access data with reference of key  
    has()                It verifies keys  
    delete()            It removes key and value  
    clear()                It removes all keys  
    keys()                It returns all keys  
    values()            It returns all values  
    entries()            It returns all keys and values  
    size                It returns the count of keys.  
  
  
Ex:  
<script>  
     var data = new Map();  
     data.set("html", "It is a markup language");  
     data.set(1, "Samsung TV - 34600.44 - Available");  
     document.write(data.get("html") + "<br>");  
     for(var ref of data.entries()){  
        document.write(ref + "<br>");  
     }  
</script>  
  
Date Type  
  
- JavaScript date and time values are stored in memory by using "Date()" constructor.  
- It allocates memory for store date and time type of values.  
  
Syntax:  
            var  ref  = new Date();        => It loads current date and time into memory.  
  
- To store date and time values in Date(), you have to follow the given format  
  
             Date("year-month-day  hrs:min:sec.milliSeconds");  
  
  
Ex:  
<script>  
     var Name = "Nike Casuals";  
     var Price = 4600.44;  
     var Stock = true;  
     var Mfd = new Date("2023-01-18 17:49:39.98");  
     document.write(`  
        Name        : ${Name} <br>  
        Price       : ${Price} <br>  
        Stock       : ${Stock} <br>  
        Mfd         : ${Mfd}  
     `);  
</script>  
  
**Date Methods**:  
  
    getHours()            : It returns hour number 0 to 23  
    getMinutes()            : It returns minutes number 0 to 59  
    getSeconds()            : It returns seconds number 0 to 59  
    getMilliSeconds()    : It returns milliseconds 0 to 99  
     
    getDate()                : It returns current date numer [1 to 31]  
    getMonth()            : It returns month number [0=Jan]  
      getDay()                 : It returns weekday number [0=Sunday]  
    getFullYear()            : It returns full year number      
    getYear()                : It is obsolete. [Returns year as per y2K]  
  
                                        12-31-99        =>  01-01-100      
    toLocaleDateString()  
    toLocaleTimeString()  
    toDateString()  
    toTimeString()  
  
  
Ex:  
<script>  
     var Name = "Nike Casuals";  
     var Price = 4600.44;  
     var Stock = true;  
     var Mfd = new Date("2023-01-17 17:49:39.98");  
     var weekdays = ["Sunday","Monday","Tuesday","Wednesday", "Thr","Fri","Sat"];  
     var months = ["January", "Feb", "March", "April","May"];  
     document.write(`  
        Name        : ${Name} <br>  
        Price       : ${Price} <br>  
        Stock       : ${Stock} <br>  
        Mfd         : ${weekdays[Mfd.getDay()]} ${Mfd.getDate()} ${months[Mfd.getMonth()]} ${Mfd.getFullYear()}  
     `);  
</script>  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Date and Time</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 GetTime(){  
            var now = new Date();  
            var date = now.toLocaleDateString();  
            var time = now.toLocaleTimeString();  
            document.getElementById("clock").innerHTML = date + "," + time;  
        }  
  
        function DisplayMessage(){  
            var msg = document.getElementById("msg");  
            var icon = document.getElementById("icon");  
            var pic = document.getElementById("pic");  
  
            var now = new Date();  
            var hrs = now.getHours();  
            if(hrs>=0 && hrs<=12) {  
                msg.innerHTML = "Good Morning";  
                icon.className = "bi bi-brightness-alt-high";  
                pic.src="../public/images/morning.gif";  
  
            } else if (hrs>12 && hrs<=17) {  
                msg.innerHTML = "Good Afternoon";  
                icon.className = "bi bi-brightness-high";  
                pic.src="../public/images/afternoon.gif";  
            } else if(hrs>17 && hrs<=23) {  
                msg.innerHTML = "Good Evening";  
                icon.className = "bi bi-brightness-alt-high-fill";  
                pic.src="../public/images/evening.gif";  
            }  
        }  
  
        function bodyload(){  
            setInterval(GetTime,1000);  
            DisplayMessage();  
        }  
    </script>  
</head>  
<body class="container-fluid" onload="bodyload()">  
    <div class="btn-toolbar mt-2 justify-content-between bg-danger">  
        <div class="btn-group">  
            <button class="btn btn-danger">Home</button>  
            <button class="btn btn-danger">About</button>  
            <button class="btn btn-danger">Contact</button>  
        </div>  
        <div class="btn-group">  
            <button class="btn btn-danger"> <span id="icon"></span> <span id="msg"></span> </button>  
        </div>  
        <div class="btn-group">  
            <button class="btn btn-danger"> <span id="clock"></span> </button>  
        </div>  
    </div>  
    <div class="d-flex mt-3 justify-content-center align-items-center">  
        <div>  
            <img id="pic" width="100" height="100">  
        </div>  
    </div>  
</body>  
</html>

JavaScript Operators  
  
- Operator is a object that evaluates a value based on give expression using operands.  
- Based on how many operands an operator can use, the operators are classified into  
**a) Unary  
            b) Binary  
            c) Ternary**  
                                x + y        expression  
                                x, y          are operands  
                                +              is operator  
  
- **Unary operator can handle only one operand**.  
  
                    x++  
                    --y  
  
        ++        is known as increment operator [it increments the current value by 1]  
        --        is know as decrement operator [it decrements the current value by 1]  
  
        x++    post increment  
        ++x    pre increment  
  
        x--        post decrement  
        --x        pre decrement  
  
Syntax: post increment  
  
            var x = 10;  
            var y = x++;                x = 11,   y=10  
         
post increment => first assign the current value to left operand and later increments  
                 
            var y = x++  [ x = x + 1 ]  
            y = 10;  
            x =  x + 1 = 11  
  
pre increment  => first increment and later assigns  
  
            var y = ++x;        [x = x + 1]  
            y = 11,  
            x = 11  
  
post decrement  
     
            var y = x--;        [x= x - 1]    y = 10, x = 9  
            var y = --x;        [x= x - 1]   y = 9, x = 9  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Fakestore</title>  
    <link rel="stylesheet" href="../node\_modules/bootstrap/dist/css/bootstrap.css">  
    <link rel="stylesheet" href="../node\_modules/bootstrap-icons/font/bootstrap-icons.css">  
    <style>  
        #price {  
            width: 100px;  
            height: 50px;  
            border-radius: 100px;  
            font-size: 20px;  
            font-weight: bold;  
            text-align: center;  
            background-color: darkcyan;  
            color:white;  
            padding: 5px;  
            position: absolute;  
            top: 200px;  
            right: 200px;  
        }  
    </style>  
    <script type="text/javascript">  
          function LoadProduct(id){  
                fetch(`<http://fakestoreapi.com/products/$>{id}`)  
                .then(function(response){  
                    return response.json();  
                })  
                .then(function(product){  
                    document.getElementById("title").innerHTML = product.title;  
                    document.getElementById("productImage").src = product.image;  
                    document.getElementById("price").innerHTML = `$ ${product.price}`;  
                })  
          }  
          function bodyload(){  
              LoadProduct(1);  
          }  
          var count = 1;  
          function NextClick(){  
              count++;  
              LoadProduct(count);  
          }  
          function PrevClick(){  
              count--;  
              LoadProduct(count);  
          }  
    </script>  
</head>  
<body class="container-fluid d-flex justify-content-center" onload="bodyload()">  
    <div id="price"></div>  
    <div class="mt-3 card w-50">  
        <div class="card-header text-center">  
            <p id="title"></p>  
        </div>  
        <div class="card-body text-center">  
            <img width="100%" height="300" id="productImage">  
        </div>  
        <div class="card-footer text-center">  
            <button class="btn btn-danger" onclick="PrevClick()">  
                <span class="bi bi-chevron-left"></span>  
            </button>  
            <button class="btn btn-danger" onclick="NextClick()">  
                <span class="bi bi-chevron-right"></span>  
            </button>  
        </div>  
    </div>  
</body>  
</html>  
  
**Binary Operator:  It handles operations on 2 operands.**  
                        x + y  
                        x + y + z  
  
**Ternary Operator:  It handles operations using 3 operands**  
                           ? :  
  
                        (operand1) ? (operand2) : (operand3)  
  
                         operand-1    => condition  
                         operand-2    => value on true  
                         operand-3    => value on false  
  
                            if (condition)  
                            {  
                              true;  
                            }  
                            else  
                            {  
                             false;  
                            }  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function ToggleClick(){  
             var toggle = document.getElementById("toggle");  
             toggle.innerHTML = (toggle.innerHTML=="ON")?"OFF":"ON";  
        }  
    </script>  
</head>  
<body>  
    <button id="toggle" onclick="ToggleClick()">ON</button>  
</body>  
</html>  
  
**- Operators are further classified into various groups based on the type of value they return**  
  
1. Arithematic operators  
2. Assignment Operators  
3. Comparision Operators  
4. Logical Operators  
5. Bitwise Operators  
6. Special Operators

1. Arithematic Operators  
- They returns a number.  
- They return a string if used over string concat [+].  
  
  
        +             Addition  
        -            Subtraction      
        \*            Multiplication  
        /            Division  
        %            Modulus Division  
        \*\*            Power [Math.pow()]        => ES5  
        ++            Increment  
        --            Decrement  
  
**Addition Operator**  
- It is used to find the sum of given value.  
  
            string + string        => string  
            string + number        => string  
            string + boolean        => string  
            number + number    => number  
            number + boolean    => number  
            boolean + boolean    => number  
  
**Subtraction Operator**  
- It is used to find the difference value.  
     
            string - string            => NaN  
            number - number    => number  
            number - boolean    => number  
            boolean - boolean    => number  
  
**Multiplication Operator**  
- It is used to find the product of given values.  
            string \* string            => NaN  
            number \* boolean    => number on true,  0 [infinity] on false  
  
**Division Operator**-  It is used to return the quotient value when  
  
            x / y  
  
**Modulus Division**  
- It is used to return remainder value.  
  
            x % y  
  
  
Exponent  
- It is used to find the power of a number  
  
            2\*\*3    => Math.pow(2,3)   => 8  
  
  
Task: Design a simple Calculator using JavaScript.

2. Assignment Operators  
        =                        assign  
        +=                        add and assign  
        -=                        subtract and assign  
        \*=                        multiply and assign  
        /=                        divide and assign  
        %=                        modulus and assign  
  
Note:   "=+"  invalid   "+="  valid  
  
**- They keep the exsiting value and adds new  value to the existing.  
   [It can be add, sub, multiply, divide]**  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var pixels = 1;  
        var left = 1;  
        function MoveClick(){  
            var pic = document.getElementById("pic");  
             left \*= pixels + 20;  
             pic.style.marginLeft = left + "px";  
        }  
    </script>  
</head>  
<body>  
    <button onclick="MoveClick()">Move right</button>  
    <br><br>  
    <div>  
        <img src="../public/images/asp.jpg" id="pic" width="100" height="100">  
    </div>  
</body>  
</html>  
  
**JavaScript can access multiple elements by using following DOM methods**-  
     
        ***getElementsByTagName()  
        getElementsByClassName()  
        getElementsByName()***        document.querySelector("input[type='checkbox']")  
  
  
EX:  
         
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function SubmitClick(){  
            document.getElementById("list").innerHTML="";  
            var categoryCheckBoxes = document.getElementsByName("category");  
            for(var checkbox of categoryCheckBoxes){  
                if(checkbox.checked) {  
                    document.getElementById("list").innerHTML += checkbox.value + "<br>";  
                }  
            }  
        }  
    </script>  
    <style>  
        ul {  
            list-style: none;  
        }  
    </style>  
</head>  
<body>  
    <ul>  
        <li><input type="checkbox" name="category" value="Electronics"> Electronics</li>  
        <li><input type="checkbox" name="category" value="Jewelery"> Jewelery</li>  
        <li><input type="checkbox" name="category" value="Footwear"> Footwear</li>  
        <li><input type="checkbox" name="category" value="Fashion"> Fashion</li>  
    </ul>  
    <button onclick="SubmitClick()">Submit</button>  
    <p id="list"></p>  
</body>  
</html>  
  
  
                        <https://seshajobs.com/>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function SubmitClick(){  
           var links = document.getElementsByTagName("a");  
           for(var link of links) {  
              document.getElementById("list").innerHTML += link.href  + "<br>";  
           }  
        }  
    </script>  
    <style>  
        ul {  
            list-style: none;  
        }  
    </style>  
</head>  
<body>  
    <a href="http://www.amazon.in">Amazon</a&gt;  
    <span>|</span>  
    <a href="http://fakestoreapi.com">Fakestore</a&gt;  
    <span>|</span>  
    <a href="http://api.nasa.gov">Nasa</a&gt;  
    <br><br>  
    <button onclick="SubmitClick()">Submit</button>  
    <p id="list"></p>  
</body>  
</html>  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var fee = 0;  
        function SubmitClick(){  
            document.getElementById("list").innerHTML="";  
            fee = 0;  
            var categoryCheckBoxes = document.getElementsByName("category");  
            for(var checkbox of categoryCheckBoxes){  
                if(checkbox.checked) {  
                    document.getElementById("list").innerHTML += checkbox.value + "<br>";  
                    fee += parseInt(checkbox.id);  
                    document.getElementById("fee").innerHTML = fee;  
                }  
            }  
        }  
    </script>  
    <style>  
        ul {  
            list-style: none;  
        }  
    </style>  
</head>  
<body>  
    <ul>  
        <li><input type="checkbox" name="category" id="2500" value="UI"> UI - 2500/-</li>  
        <li><input type="checkbox" name="category" id="4000" value="Java"> Java - 4000/-</li>  
        <li><input type="checkbox" name="category" id="3500" value="PHP"> PHP - 3500/- </li>  
        <li><input type="checkbox" name="category" id="1000" value="Python"> Python - 1000/- </li>  
    </ul>  
    <button onclick="SubmitClick()">Submit</button>  
    <p id="list"></p>  
    <p id="fee"></p>  
</body>  
</html>  
  
  
Ex:  
<script>  
     var products = [  
         {Name: "TV", Category:"Electronics"},  
         {Name:"Shoe", Category: "Footwear"},  
         {Name:"Shirt", Category: "Fashion"},  
         {Name: "Mobile", Category: "Electronics"}  
     ];  
     var result = products.filter(function(item){  
         return item.Category=="Fashion" || item.Category=="Footwear";  
     });  
     for(var item of result) {  
        document.write(item.Name + "<br>");  
     }  
</script>

3. Comparision Operators

==                        Equal  
===                    Identical Equal  
>                        Greater than  
>=                        Greater than or equal  
<                        Less than  
<=                        Less than or equal  
!=                        Not Equal  
!==                        Not Identical  
  
**FAQ: What is difference between "==" & "===" ?**Ans:  
      **"=="  can compare value of different types.**  
                 
            var x = "10";  
            var y = 10;  
  
            document.write(x==y);            // true  
  
**"===" can compare value of same type only.**  
            document.write(x===y);            // false  
  
Task: Find the largest among given 3 numbers.  
  
                a = 20,  
                b = 10,  
                c = 8  
Ex:  
<script>  
    function FindLargest(a,b,c){  
        if(a>b && a>c){  
            document.write(`${a} is greater`);  
        } else if (b>c) {  
            document.write(`${b} is greater`);  
        } else {  
            document.write(`${c} is greater`);  
        }  
    }  
    FindLargest(110,64,60);  
</script>

4. Logical Operators

&&            AND  
||            OR  
!            NOT  
  
Logical operator can bind multiple expressions or can return a negation of expression.  
  
            expression-1   &&  expression-2       =>  if both are true then statement is true  
            expression-1   ||  expression-2            =>  if any one is true then statement true  
  
Ex:  
1. Add a new file into "data" folder  
  
            "users.json"  
[  
    {  
        "UserId":"john\_ui",  
        "Mobile": "+919876543210",  
        "Password":"john@12"  
    },  
    {  
        "UserId":"sam\_nit",  
        "Mobile":"+919977554321",  
        "Password": "sam11"  
    }  
]  
  
2. Login.html  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function LoginClick(){  
            var userid = document.getElementById("UserId").value;  
            var password = document.getElementById("Password").value;  
            fetch("../data/users.json")  
            .then(function(response){  
                return response.json();  
            })  
            .then(function(users){  
                for(var user of users){  
                    if((user.UserId===userid||user.Mobile===userid) && user.Password===password)  
                    {  
                        document.write(`<h2>${user.UserId} Login Success..</h2>`);  
                        break;  
                    } else {  
                        document.getElementById("msg").innerHTML = "Invalid UserId / Password";  
                    }  
                }  
            })  
        }  
    </script>  
</head>  
<body>  
    <h3>User Login</h3>  
    <dl>  
        <dt>User Id or mobile</dt>  
        <dd><input type="text" id="UserId"></dd>  
        <dt>Password</dt>  
        <dd><input type="password" id="Password"></dd>  
    </dl>  
    <button onclick="LoginClick()">Login</button>  
    <p align="center" id="msg"></p>  
</body>  
</html>  
  
Ex: With Profile Pic  
  
users.json  
  
[  
    {  
        "UserId":"john\_ui",  
        "UserName":"John D",  
        "Mobile": "+919876543210",  
        "Password":"john@12",  
        "Photo": "../public/images/john.jpg"  
    },  
    {  
        "UserId":"sam\_nit",  
        "UserName":"Samson",  
        "Mobile":"+919977554321",  
        "Password": "sam11",  
        "Photo":"../public/images/sam.jpg"  
    }  
]  
  
Login.html  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function LoginClick(){  
            var userid = document.getElementById("UserId").value;  
            var password = document.getElementById("Password").value;  
            fetch("../data/users.json")  
            .then(function(response){  
                return response.json();  
            })  
            .then(function(users){  
                for(var user of users){  
                    if((user.UserId===userid||user.Mobile===userid) && user.Password===password)  
                    {  
                        document.write(`<h2>${user.UserName} </h2> <img src=${user.Photo} width="100" height="100"> <p> Login Success..</p>`);  
                        break;  
                    } else {  
                        document.getElementById("msg").innerHTML = "Invalid UserId / Password";  
                    }  
                }  
            })  
        }  
    </script>  
</head>  
<body>  
    <h3>User Login</h3>  
    <dl>  
        <dt>User Id or mobile</dt>  
        <dd><input type="text" id="UserId"></dd>  
        <dt>Password</dt>  
        <dd><input type="password" id="Password"></dd>  
    </dl>  
    <button onclick="LoginClick()">Login</button>  
    <p align="center" id="msg"></p>  
</body>  
</html>  
  
- Logical Not  is to define the negation of given expression  
  
                    var x = true;  
                    var y = !x;            y = false  
  
            if(!undefined) {  
  
            }

5. Special Operators

**1. Ternary Operator**  
         
            ? :  
  
**2. typeof**      
            It returns the data type of value stored in a reference.  
  
**3. instanceof**  
            It returns true if the object is derived for specified class.  
  
**4. delete**  
            It is used to remove any property from object.  
  
**5. in**            It returns true if the give property is available in object.  
  
**6. of**            It returns all values from a collection  
  
  
**7. void**          
            It is used to specify element without return value.  
            It discards the return value of element.  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
</head>  
<body>  
    <a href="javascript:void()">Home</a>  
</body>  
</html>

JavaScript Statements

- A statement is used to control the execution flow.  
- Statements are classified into various groups  
  
            a) Selection Statements  
                    if, else, switch, case, default  
            b) Looping Control Statements  
                    for, while, do while  
            c) Iteration Statements  
                    for..of, for..in  
            d) Jump Statements  
                    break, return, continue  
            e) Exception Handling Statements  
                    try, catch, throw, finally

1. Selection Statements  
- Selection statements are used in decision making.  
- Keywords  
            **if, else, switch, case, default**  
  
The IF select:  
- It is a selection statement used to execute a set of statements when condition evaluates to true and another set of statements when it evaluates to false.  
  
- It have various forms  
  
**1. Forward Jump  
            2. Simple Decision  
            3. Multi level decision  
            4. Multiple Choices**  
Forward Jump:  
- It is a decision making process where there is no alternative.  
- User can go to next step only when the given evaluates to true.  
  
Syntax:  
            if (condition)  
            {  
              statement on true;  
            }  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var userDetails = {  
             Card: "4444555566667890",  
             Cvv: "3456",  
             Expiry:"2024"  
        };  
        function VerifyCard(){  
            var card = document.getElementById("Card").value;  
            if(card===userDetails.Card) {  
                document.getElementById("cvvContainer").style.display = "block";  
                document.getElementById("Card").readOnly = true;  
            }  
        }  
        function VerifyCvv(){  
            var cvv = document.getElementById("Cvv").value;  
            if(cvv===userDetails.Cvv) {  
                document.getElementById("expiryContainer").style.display= "block";  
                document.getElementById("Cvv").readOnly = true;  
            }  
        }  
        function VerifyExpiry(){  
            var expiry = document.getElementById("Expiry").value;  
            if(expiry === userDetails.Expiry) {  
                document.getElementById("btnPay").style.display = "block";  
                document.getElementById("Expiry").disabled = true;  
            }  
        }  
    </script>  
</head>  
<body>  
    <fieldset>  
        <legend>Payment Details</legend>  
        <dl>  
            <dt>Card Number</dt>  
            <dd><input type="text" id="Card" onblur="VerifyCard()"></dd>  
            <div id="cvvContainer" style="display:none">  
                <dt>CVV</dt>  
                <dd><input type="text" id="Cvv" size="4" onblur="VerifyCvv()"></dd>  
            </div>  
            <div id="expiryContainer" style="display: none;">  
                <dt>Expiry</dt>  
                <dd>  
                    <select id="Expiry" onchange="VerifyExpiry()">  
                        <option>2023</option>  
                        <option>2024</option>  
                        <option>2025</option>  
                    </select>  
                </dd>  
            </div>  
        </dl>  
        <button id="btnPay" style="display: none;">Pay</button>  
    </fieldset>  
</body>  
</html>  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function SubmitClick(){  
            var optMale = document.getElementById("optMale");  
            var optFemale = document.getElementById("optFemale");  
            var msg = document.getElementById("msg");  
            if(optMale.checked) {  
                msg.innerHTML = `Gender : ${optMale.value}`;  
            }  
            if(optFemale.checked) {  
                msg.innerHTML = `Gender : ${optFemale.value}`;  
            }  
        }  
    </script>  
</head>  
<body>  
    <fieldset>  
        <legend>Select Gender</legend>  
        <input type="radio" name="gender" id="optMale" value="Male"> <label>Male</label>  
        <input type="radio" name="gender" id="optFemale" value="Female"> <label>Female</label>  
        <br><br>  
        <button onclick="SubmitClick()">Submit</button>  
    </fieldset>  
    <p id="msg"></p>  
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <link rel="stylesheet" href="../node\_modules/bootstrap/dist/css/bootstrap.css">  
    <script>  
         var userDetails = {  
            "Email": "[john11@gmail.com](https://mail.google.com/mail/?view=cm&fs=1&to=john11%40gmail.com&authuser=0)",  
            "Password": "john@123"  
         }  
    </script>  
</head>  
<body class="container-fluid">  
    <div class="d-flex justify-content-center align-items-center" style="height:500px">  
        <div class="w-25">  
            <div>  
                <h3>Signin</h3>  
                <div id="emailContainer">  
                    <label class="form-label fw-bold">Your Email</label>  
                    <div>  
                        <input type="text" class="form-control" id="txtEmail">  
                    </div>  
                    <div class="mt-2">  
                        <button class="btn btn-warning w-100">Continue</button>  
                    </div>  
                </div>  
                <div id="passwordContainer" style="display: none;">  
                    <label class="form-label fw-bold">Your Password</label>  
                    <div>  
                        <input type="password" class="form-control" id="txtPassword">  
                    </div>  
                    <div class="mt-2">  
                        <button class="btn btn-warning w-100">Login</button>  
                    </div>  
                </div>  
            </div>  
        </div>  
    </div>  
</body>  
</html>

IF Select  
1. Forward Jump  
  
            if (condition )  
            {  
             statement true;  
            }  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>KFC Online Order</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 type="text/javascript">  
        function OrderClick(){  
            document.getElementById("lblName").innerHTML = document.getElementById("txtName").value;  
            document.getElementById("lblMobile").innerHTML = document.getElementById("txtMobile").value;  
  
            var mealName = "";  
            var mealCost = 0;  
  
            var burgerRadio = document.getElementById("optBurger");  
            var rollerRadio = document.getElementById("optRoller");  
  
            if(burgerRadio.checked) {  
                mealName = burgerRadio.value;  
                mealCost = 120;  
            }  
            if(rollerRadio.checked) {  
                mealName = rollerRadio.value;  
                mealCost = 100;  
            }  
  
            var adonName = "";  
            var adonCost = 0;  
  
            var krusherCheckbox = document.getElementById("optKrusher");  
            var wingsCheckbox = document.getElementById("optWings");  
  
            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("lblTotal").innerHTML = `&#8377; ${mealCost}`.bold().fontcolor('green');  
  
        }  
    </script>  
</head>  
<body class="container-fluid">  
    <header>  
        <img src="../public/images/kfctop.PNG" width="100%" height="160">  
    </header>  
    <section>  
        <div class="accordion" id="kfc">  
            <div class="accordion-item">  
                <div class="accordion-header">  
                    <button data-bs-target="#customer" data-bs-toggle="collapse" class="btn btn-danger w-100">Customer Details</button>  
                </div>  
                <div class="accordion-collapse collapse show" id="customer" data-bs-parent="#kfc">  
                    <div class="accordion-body bg-danger text-white mt-2">  
                        <dl>  
                            <dt>Customer Name</dt>  
                            <dd><input type="text" class="form-control" id="txtName"></dd>  
                            <dt>Mobile</dt>  
                            <dd><input type="text" class="form-control" id="txtMobile"></dd>  
                        </dl>  
                    </div>  
                </div>  
            </div>  
            <div class="accordion-item">  
                <div class="accordion-header">  
                    <button data-bs-target="#meal" data-bs-toggle="collapse" class="btn btn-danger w-100">Select Your Meal</button>  
                </div>  
                <div class="accordion-collapse collapse" id="meal" data-bs-parent="#kfc">  
                    <div class="accordion-body">  
                        <div class="row">  
                            <div class="col text-center">  
                                <div class="card">  
                                    <img src="../public/images/omg1.PNG" class="card-img-top" height="200">  
                                    <div class="card-header">  
                                        <h3>OMG Burger</h3>  
                                    </div>  
                                    <div class="card-footer">  
                                        <h3><input type="radio" name="meal" id="optBurger" value="OMG Burger" class="form-check-input"> <label class="form-check-label"> &#8377; 120/-</label></h3>  
                                    </div>  
                                </div>  
                            </div>  
                            <div class="col text-center">  
                                <div class="card">  
                                    <img src="../public/images/omg2.PNG" class="card-img-top" height="200">  
                                    <div class="card-header">  
                                        <h3>OMG Roller</h3>  
                                    </div>  
                                    <div class="card-footer">  
                                        <h3><input type="radio" name="meal" id="optRoller" value="OMG Roller" class="form-check-input"> <label class="form-check-label"> &#8377; 100/-</label></h3>  
                                    </div>  
                                </div>  
                            </div>  
                        </div>  
                    </div>  
                </div>  
            </div>  
  
  
            <div class="accordion-item">  
                <div class="accordion-header">  
                    <button data-bs-target="#adon" data-bs-toggle="collapse" class="btn btn-danger w-100">Select Add-ON's</button>  
                </div>  
                <div class="accordion-collapse collapse" id="adon" data-bs-parent="#kfc">  
                    <div class="accordion-body">  
                        <div class="row">  
                            <div class="col text-center">  
                                <div class="card">  
                                    <img src="../public/images/krusher1.PNG" class="card-img-top" height="200">  
                                    <div class="card-header">  
                                        <h3>Krusher Brownie</h3>  
                                    </div>  
                                    <div class="card-footer">  
                                        <h3><input type="checkbox" id="optKrusher" value="Krusher Brownie" class="form-check-input"> <label class="form-check-label"> &#8377; 60/-</label></h3>  
                                    </div>  
                                </div>  
                            </div>  
                            <div class="col text-center">  
                                <div class="card">  
                                    <img src="../public/images/wings.PNG" class="card-img-top" height="200">  
                                    <div class="card-header">  
                                        <h3>Hot Wings</h3>  
                                    </div>  
                                    <div class="card-footer">  
                                        <h3><input type="checkbox"  id="optWings" value="Hot Wings" class="form-check-input"> <label class="form-check-label"> &#8377; 80/-</label></h3>  
                                    </div>  
                                </div>  
                            </div>  
                        </div>  
                    </div>  
                </div>  
            </div>  
        </div>  
        <button onclick="OrderClick()" data-bs-target="#summary" data-bs-toggle="modal" class="btn btn-danger w-100">View & Place Order</button>  
        <div class="modal fade" id="summary">  
            <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>  
                            <dt>Customer Name</dt>  
                            <dd id="lblName"></dd>  
                            <dt>Mobile</dt>  
                            <dd id="lblMobile"></dd>  
                            <dt>Meal</dt>  
                            <dd id="lblMeal"></dd>  
                            <dt>Ad-ON's</dt>  
                            <dd id="lblAdon"></dd>  
                            <dt>Total</dt>  
                            <dd id="lblTotal"></dd>  
                        </dl>  
                    </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: KFC  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>KFC Online Order</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 type="text/javascript">  
        function OrderClick(){  
            document.getElementById("lblName").innerHTML = document.getElementById("txtName").value;  
            document.getElementById("lblMobile").innerHTML = document.getElementById("txtMobile").value;  
  
            var mealName = "";  
            var mealCost = 0;  
            var mealPreview = "";  
  
            var burgerRadio = document.getElementById("optBurger");  
            var rollerRadio = document.getElementById("optRoller");  
            var pic = document.getElementById("mealPreview");  
  
            if(burgerRadio.checked) {  
                mealName = burgerRadio.value;  
                mealCost = 120;  
                mealPreview = "../public/images/omg1.png";  
            }  
            if(rollerRadio.checked) {  
                mealName = rollerRadio.value;  
                mealCost = 100;  
                mealPreview = "../public/images/omg2.png";  
            }  
  
            var adonName = "";  
            var adonCost = 0;  
  
            var krusherCheckbox = document.getElementById("optKrusher");  
            var wingsCheckbox = document.getElementById("optWings");  
  
            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("lblTotal").innerHTML = `&#8377; ${mealCost}`.bold().fontcolor('green');  
            pic.src= mealPreview;  
        }  
    </script>  
</head>  
<body class="container-fluid">  
    <header>  
        <img src="../public/images/kfctop.PNG" width="100%" height="160">  
    </header>  
    <section>  
        <div class="accordion" id="kfc">  
            <div class="accordion-item">  
                <div class="accordion-header">  
                    <button data-bs-target="#customer" data-bs-toggle="collapse" class="btn btn-danger w-100">Customer Details</button>  
                </div>  
                <div class="accordion-collapse collapse show" id="customer" data-bs-parent="#kfc">  
                    <div class="accordion-body bg-danger text-white mt-2">  
                        <dl>  
                            <dt>Customer Name</dt>  
                            <dd><input type="text" class="form-control" id="txtName"></dd>  
                            <dt>Mobile</dt>  
                            <dd><input type="text" class="form-control" id="txtMobile"></dd>  
                        </dl>  
                    </div>  
                </div>  
            </div>  
            <div class="accordion-item">  
                <div class="accordion-header">  
                    <button data-bs-target="#meal" data-bs-toggle="collapse" class="btn btn-danger w-100">Select Your Meal</button>  
                </div>  
                <div class="accordion-collapse collapse" id="meal" data-bs-parent="#kfc">  
                    <div class="accordion-body">  
                        <div class="row">  
                            <div class="col text-center">  
                                <div class="card">  
                                    <img src="../public/images/omg1.PNG" class="card-img-top" height="200">  
                                    <div class="card-header">  
                                        <h3>OMG Burger</h3>  
                                    </div>  
                                    <div class="card-footer">  
                                        <h3><input type="radio" name="meal" id="optBurger" value="OMG Burger" class="form-check-input"> <label class="form-check-label"> &#8377; 120/-</label></h3>  
                                    </div>  
                                </div>  
                            </div>  
                            <div class="col text-center">  
                                <div class="card">  
                                    <img src="../public/images/omg2.PNG" class="card-img-top" height="200">  
                                    <div class="card-header">  
                                        <h3>OMG Roller</h3>  
                                    </div>  
                                    <div class="card-footer">  
                                        <h3><input type="radio" name="meal" id="optRoller" value="OMG Roller" class="form-check-input"> <label class="form-check-label"> &#8377; 100/-</label></h3>  
                                    </div>  
                                </div>  
                            </div>  
                        </div>  
                    </div>  
                </div>  
            </div>  
  
  
            <div class="accordion-item">  
                <div class="accordion-header">  
                    <button data-bs-target="#adon" data-bs-toggle="collapse" class="btn btn-danger w-100">Select Add-ON's</button>  
                </div>  
                <div class="accordion-collapse collapse" id="adon" data-bs-parent="#kfc">  
                    <div class="accordion-body">  
                        <div class="row">  
                            <div class="col text-center">  
                                <div class="card">  
                                    <img src="../public/images/krusher1.PNG" class="card-img-top" height="200">  
                                    <div class="card-header">  
                                        <h3>Krusher Brownie</h3>  
                                    </div>  
                                    <div class="card-footer">  
                                        <h3><input type="checkbox" id="optKrusher" value="Krusher Brownie" class="form-check-input"> <label class="form-check-label"> &#8377; 60/-</label></h3>  
                                    </div>  
                                </div>  
                            </div>  
                            <div class="col text-center">  
                                <div class="card">  
                                    <img src="../public/images/wings.PNG" class="card-img-top" height="200">  
                                    <div class="card-header">  
                                        <h3>Hot Wings</h3>  
                                    </div>  
                                    <div class="card-footer">  
                                        <h3><input type="checkbox"  id="optWings" value="Hot Wings" class="form-check-input"> <label class="form-check-label"> &#8377; 80/-</label></h3>  
                                    </div>  
                                </div>  
                            </div>  
                        </div>  
                    </div>  
                </div>  
            </div>  
        </div>  
        <button onclick="OrderClick()" data-bs-target="#summary" data-bs-toggle="modal" class="btn btn-danger w-100">View & Place Order</button>  
        <div class="modal fade" id="summary">  
            <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>  
                            <dt>Customer Name</dt>  
                            <dd id="lblName"></dd>  
                            <dt>Mobile</dt>  
                            <dd id="lblMobile"></dd>  
                            <dt>Meal</dt>  
                            <dd><img id="mealPreview" width="150" height="150"></dd>  
                            <dd id="lblMeal"></dd>  
                            <dt>Ad-ON's</dt>  
                            <dd id="lblAdon"></dd>  
                            <dt>Total</dt>  
                            <dd id="lblTotal"></dd>  
                        </dl>  
                    </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

2. Simple Decision  
            if(condition)  
            {  
             true;  
            }  
            else  
            {  
             false;  
            }  
  
    If and else clause.  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function LoginClick(){  
            var username = document.getElementById("UserName").value;  
            var password = document.getElementById("Password").value;  
            fetch("../data/users.json")  
            .then(function(res){  
                return res.json();  
            })  
            .then(function(users){  
                for(var user of users){  
                    if(user.UserId===username && user.Password===password) {  
                        document.write("Login Success");  
                    } else {  
                        document.querySelector("p").innerHTML = "Invalid Credentials";  
                    }  
                }  
            })  
        }  
    </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>  
    <p></p>  
</body>  
</html>  
  
3. Multi Level Decisions  
- You have to verify only one set of condition using "if()"  
- So that you can validated every inidividual condition defined.  
  
Syntax:  
        if(condition1)  
        {  
            if(condition2)  
              {  
                statements when all conditions evaluate to true;  
            }  
            else {  
              statement on condition-2 false;  
            }  
        }  
        else  
        {  
           statement on condition-1 false;  
        }  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function LoginClick(){  
            var username = document.getElementById("UserName").value;  
            var password = document.getElementById("Password").value;  
            var msg = document.querySelector("p");  
  
            fetch("../data/users.json")  
            .then(function(res){  
                return res.json();  
            })  
            .then(function(users){  
                for(var user of users){  
                   if(username==""){  
                      msg.innerHTML = "User Name Required";  
                   } else {  
                         if(user.UserId===username){  
                            if(user.Password===password){  
                                document.write("Login Success");  
                            } else {  
                                msg.innerHTML = "Invalid Password";  
                                break;  
                            }  
                        } else {  
                            msg.innerHTML = "Invalid UserName";  
                             
                        }  
                   }  
                }  
            })  
        }  
    </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>  
    <p style="color:red"></p>  
</body>  
</html>  
  
4. Multiple Choices  
- It provides multiple decisions for same task to perform.  
  
Syntax:  
        if(condition-1)  
        {  
           statement on condition-1 true;  
        }  
         else if(condition-2)  
        {  
           statement on condition-2 true;  
        }  
        else {  
            if both conditions are false;  
        }  
         
Ex:  
users.json  
  
[  
    {  
        "UserId":"john\_ui",  
        "UserName":"John D",  
        "Mobile": "+919876543210",  
        "Password":"john@12",  
        "Photo": "../public/images/john.jpg",  
        "Email": "[john@gmail.com](https://mail.google.com/mail/?view=cm&fs=1&to=john%40gmail.com&authuser=0)"  
    },  
    {  
        "UserId":"sam\_nit",  
        "UserName":"Samson",  
        "Mobile":"+919977554321",  
        "Password": "sam11",  
        "Photo":"../public/images/sam.jpg",  
        "Email": "[sam@outlook.com](https://mail.google.com/mail/?view=cm&fs=1&to=sam%40outlook.com&authuser=0)"  
    }  
]  
  
amazon-login.html  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Amazon Login</title>  
    <link rel="stylesheet" href="../node\_modules/bootstrap/dist/css/bootstrap.css">  
    <script>  
         var flag = "";  
         function ContinueClick(){  
            var userid = document.getElementById("UserId").value;  
            var userContainer = document.getElementById("userContainer");  
            var passwordContainer = document.getElementById("passwordContainer");  
  
            function ToggleContainers(){  
                userContainer.style.display = "none";  
                passwordContainer.style.display = "block";  
            }  
  
            fetch("../data/users.json")  
            .then(function(res){  
                return res.json()  
            })  
            .then(function(users){  
                for(var user of users){  
                    if(user.Email===userid){  
                        flag = `Login Success and Email sent to ${userid}`;  
                        ToggleContainers();  
                    } else if(user.Mobile===userid){  
                        flag = `Login Success and OTP sent to your mobile ${userid}`  
                        ToggleContainers();  
                    } else {  
                        document.getElementById("userError").innerHTML = `${userid} doesn't exits`;  
                    }  
                }  
            })  
         }  
  
         function LoginClick(){  
            var password = document.getElementById("Password").value;  
  
            fetch("../data/users.json")  
            .then(function(res){  
                return res.json();  
            })  
            .then(function(users){  
                for(var user of users){  
                    if(user.Password===password) {  
                        document.write(`<h2>${flag}</h2>`);  
                    } 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: 500px;">  
        <div class="w-25">  
            <h2>Singin</h2>  
            <div id="userContainer">  
                <div class="mb-3">  
                    <label class="form-label fw-bold">Email or mobile phone number</label>  
                    <div>  
                        <input type="text" class="form-control" id="UserId">  
                        <div class="text-danger" id="userError"></div>  
                    </div>  
                </div>  
                <div>  
                    <button onclick="ContinueClick()" class="btn btn-warning w-100">Continue</button>  
                </div>  
            </div>  
            <div id="passwordContainer" style="display: none;">  
                <div class="mb-3">  
                    <label class="form-label fw-bold">Your Password</label>  
                    <div>  
                        <input type="password" id="Password" class="form-control">  
                        <div class="text-danger" id="pwdError"></div>  
                    </div>  
                </div>  
                <div>  
                    <button onclick="LoginClick()" class="btn btn-warning w-100">Login</button>  
                </div>  
            </div>  
        </div>  
    </div>  
</body>  
</html>

#Switch Selector  
- Switch in electronics is used to interrupt the flow of electrons.  
- Switches are various types  
        toggle switch  
        push button switch  
        joy stick switch  
        selector switch ...

FAQ's  
1. Can we define switch without default?  
2. Can we define default before or between cases?  
3. Can we define case without break?  
4. Can we define return as jump statement?  
5. What is difference between return and break?  
  
6. How to define multiple blocks for same condition?  
A. By removing break.  
  
7. How to define single block for multiple cases?  
A. By using multiple case statements or by using multiple conditions using "||" operator.  
  
Ex:  
<script>  
    var value = prompt("Enter Value");  
    switch(value)  
    {  
        case "red":  
        case "green":  
        case "blue":  
        document.write(`You selected ${value} color`);  
        break;  
        case "tv":  
        case "mobile":  
        case "watch":  
        document.write(`You selected ${value} which belongs to Electronics`);  
        break;  
        default:  
        document.write(`Please enter colors or electronic products..`);  
        break;  
    }  
</script>  
  
Ex:  
<script>  
    var choice = prompt("Enter Y or N");  
    switch(choice)  
    {  
        case "y":  
        case "Y":  
        document.write("You selected Yes..");  
        break;  
        case "n":  
        case "N":  
        document.write("You selected No..");  
        break;  
    }  
</script>  
  
8. How to write a case for range of values?  
A. By using a boolean expression in case and by passing boolean true into switch.  
  
Ex:  
<script>  
    var n = parseInt(prompt("Enter Number"));  
    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(`Your number ${n} is between 11 to 20`);  
       break;  
    }  
</script>  
  
9. Can we define "if" selector in switch?  
A. yes.  
  
  
    case "red":  
    case "green":  
    case "blue":  
        if(choice=="red") {   } else {  }  
     break;  
  
10. Can we define switch within case? Multi Level  
A.  Yes. But not recommended to directly write a switch, you can write in a function and     call.  
  
Note: If switch..case  is "case sensitive" then always convert the case and compare.  
  
Ex:  
<script>  
  var value = prompt("Enter Value");  
  switch(value.toLowerCase())  
  {  
      case "red":  
      case "green":  
      case "blue":  
      document.write(`You selected ${value} color`);  
      break;  
      case "tv":  
      case "mobile":  
      case "watch":  
      document.write(`You selected ${value} which belongs to Electronics`);  
      break;  
      default:  
      document.write(`Please enter colors or electronic products..`);  
      break;  
  }  
</script>  
  
     
**Ex: Cascading Dropdown**  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta http-equiv="X-UA-Compatible" content="IE=edge">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>Document</title>  
  <script>  
      var categories = ["Select a Category", "Electronics", "Footwear"];  
      var electronics = ["Select Electronics", "Televisions", "Mobiles"];  
      var footwear = ["Select Footwear", "Boots", "Sneakers"];  
      var products = [];  
      function LoadCategories(){  
         for(var category of categories) {  
             var option = document.createElement("option");  
             option.text = category;  
             option.value = category;  
             document.getElementById("lstCategories").appendChild(option);  
         }  
      }  
      function LoadProducts(){  
        document.getElementById("lstProducts").innerHTML = "";  
        for(var product of products) {  
             var option = document.createElement("option");  
             option.text = product;  
             option.value = product;  
             document.getElementById("lstProducts").appendChild(option);  
         }  
      }  
      function bodyload(){  
        LoadCategories();  
      }  
      function CategoryChange(){  
        var categoryName = document.getElementById("lstCategories").value;  
        switch(categoryName)  
        {  
           case "Electronics":  
           products = electronics;  
           LoadProducts();  
           break;  
           case "Footwear":  
           products = footwear;  
           LoadProducts();  
           break;  
           default:  
           products = ["Select Category"];  
           LoadProducts();  
           break;  
        }  
      }  
  </script>  
</head>  
<body onload="bodyload()">  
   <dl>  
    <dt>Select Category</dt>  
    <dd>  
      <select id="lstCategories" onchange="CategoryChange()"></select>  
    </dd>  
    <dt>Select Product</dt>  
    <dd>  
      <select id="lstProducts"></select>  
    </dd>  
   </dl>  
</body>  
</html>  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>String Match</title>  
    <link rel="stylesheet" href="../node\_modules/bootstrap/dist/css/bootstrap.css">  
    <style>  
        #flag {  
            position: absolute;  
            top: 60px;  
            right: 500px;  
        }  
    </style>  
    <script>  
        var regExp = / /;  
        function CountryChange(){  
            var countryName = document.getElementById("lstCountry").value;  
            var flag = document.getElementById("flag");  
            var mobileTextBox = document.getElementById("txtMobile");  
            var error = document.getElementById("error");  
  
            function SetExpression(pic, tip, pattern){  
                flag.src = pic;  
                flag.alt = tip;  
                mobileTextBox.placeholder = tip;  
                regExp = pattern;  
                error.innerHTML = "";  
            }  
  
            switch(countryName){  
                case "India":  
                    SetExpression("../public/images/india.png","+91 and 10 digits",/\+91\d{10}/);  
                break;  
                case "US":  
                    SetExpression("../public/images/us.png","+(1)(425) 555-0100",/\+\(1\)\(\d{3}\)\s\d{3}-\d{4}/);  
                break;  
                case "UK":  
                    SetExpression("../public/images/uk.png","+(44)(20) 1234 5678",/\+\(44\)\(\d{2}\)\s\d{4}\s\d{4}/);  
                break;  
                default:  
                error.innerHTML = "Please Select your country".fontcolor('red');  
                flag.src="";  
                mobileTextBox.placeholder="Country not selected";  
                break;  
  
            }  
        }  
        function VerifyMobile(){  
            var mobileNumber = document.getElementById("txtMobile").value;  
            var mobileError = document.getElementById("mobileError");  
            if(mobileNumber.match(regExp)) {  
                document.write("<h2>Mobile Verified Successfully..</h2>");  
            } else {  
                mobileError.innerHTML= `Invalid Mobile : <b>${document.getElementById("txtMobile").placeholder}</b>`.fontcolor('red');  
            }  
        }  
    </script>  
</head>  
<body class="container-fluid">  
    <img id="flag" width="100" height="100" class="rounded rounded-circle">  
    <h2>Verify Mobile Number</h2>  
    <dl class="w-25">  
        <dt>Select Your Country</dt>  
        <dd>  
            <select onchange="CountryChange()" id="lstCountry" class="form-select">  
                <option>Select Country</option>  
                <option>India</option>  
                <option>US</option>  
                <option>UK</option>  
            </select>  
        </dd>  
        <dd id="error"></dd>  
        <dt>Your Mobile Number</dt>  
        <dd>  
            <input type="text" id="txtMobile" class="form-control">  
        </dd>  
        <button onclick="VerifyMobile()" class="btn btn-primary w-100">Submit</button>  
    </dl>  
    <p class="mt-4 text-center" id="mobileError"></p>  
</body>  
</html>

2. Looping Control Statements  
  
- Looping is the process of executing a set of statement repeatedly until the given condition is statisfied.  
- Loop comprises of set of statements, which return same value everytime or differ.  
- Loops are created by using  
**a) for  
        b) while  
        c) do while**

1. The For loop  
- Developers use for loop when the number of iterations are known and the iteration counter will not change dynamically.  
  
Syntax:  
    for (initialization; condition; counter)  
    {  
      statements;  
    }  
  
    initialization        : it defines where to start.  
    condition            : when to stop, how many times to repeat  
    counter            : it defines how to move [forward, backward, step]  
  
Ex:  
<script>  
     for(var i=1; i<=10; i++)  
     {  
        document.write(i + "<br>");  
     }  
</script>  
  
Note: Initializer and condition can be defined outside  for()  
  
    var i=1;  
    for(  ; i<=10; i++)  
    {  
    }  
Ex:  
<script>  
     var i=5;  
     for( ;i<=10; i++)  
     {  
        document.write(i + "<br>");  
     }  
</script>  
  
Ex:  
<script>  
     for(var i=10 ;i>=1; i--)  
     {  
        document.write(i + "<br>");  
     }  
</script>  
  
Ex:  
<script>  
     for(var i=10 ;i>=1; i=i-2)  
     {  
        document.write(i + "<br>");  
     }  
</script>  
  
  
Ex:  
<script>  
    var categories = ["All", "Electronics", "Footwear", "Fashion"];  
    for(var i=0; i<categories.length;i++){  
        document.write(categories[i] + "<br>");  
    }  
</script>  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var menu = [  
            {Category: "Electronics", Products: ["TV", "Mobile"]},  
            {Category: "Footwear", Products: ["Boots", "Casuals"]}  
        ];  
        function bodyload(){  
            for(var i=0; i<menu.length; i++){  
                var li = document.createElement("li");  
                li.innerHTML = menu[i].Category;  
                for(var j=0; j<menu[i].Products.length; j++) {  
                    var ul = document.createElement("ul");  
                    var ulLi = document.createElement("li");  
                    ulLi.innerHTML = menu[i].Products[j];  
                    ul.appendChild(ulLi);  
                    li.appendChild(ul);  
                    document.querySelector("ol").appendChild(li);  
                }  
  
            }  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <ol>  
  
    </ol>  
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var menu = [  
            {Category: "Electronics", Products: ["TV", "Mobile"]},  
            {Category: "Footwear", Products: ["Boots", "Casuals"]}  
        ];  
        function bodyload(){  
            for(var item of menu)  
            {  
                var li = document.createElement("li");  
                li.innerHTML = item.Category;  
                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);  
                    document.querySelector("ol").appendChild(li);  
                }  
            }  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <ol>  
  
    </ol>  
</body>  
</html>  
  
Ex: 2D Array  
  
<script>  
     var values = [[10, 20], [30, 40]];  
     for(var i=0; i<values.length; i++)  
     {  
          for(var j=0; j<values[i].length; j++){  
              document.write(`${values[i][j]} &nbsp; &nbsp; &nbsp;`);  
          }  
          document.write("<br><br>");  
     }  
</script>  
  
Ex:  
<script>  
     var n = parseInt(prompt("Enter number"));  
     for(var i=1; i<=n; i++){  
        for(j=0; j<i; j++) {  
            document.write(`\* &nbsp; &nbsp;`);  
        }  
        document.write("<br>");  
     }  
</script>

2. While Loop  
- It is used when the number of iterations are not sure and the iteration counter may change dynamically.  
  
Syntax:  
        while (condition)  
        {  
          statements;  
          counter;  
        }  
  
- In while loop statements will not execute when condition is false.  
  
Ex:  
<script>  
    var i = 1;  
    while(i<=10)  
    {  
        document.write(i + "<br>");  
        i++;  
    }  
</script>

3. Do While  
- It is same as while loop, but it ensures that statements will execute atleast once even when the condition is false.  
  
Syntax:  
    do  
    {  
     statements;  
     counter;  
    } while(condition);  
  
Ex:  
<script>  
    var i = 1;  
    do {  
        document.write(i + "<br>");  
        i++;  
    }while(i<=10);  
</script>  
  
Ex: PIN Verification  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var userDetails = {  
            UserName: 'john',  
            PIN: 3530  
        }  
        var count = 0;  
        function VerifyClick(){  
            var pin = parseInt(document.getElementById("PIN").value);  
            var msg = document.getElementById("msg");  
            if(pin===userDetails.PIN) {  
                document.write("Transaction Completed Successfully..");  
            } else {  
                do {  
                   count++;  
                   if(count==3) {  
                      msg.innerHTML="Your card blocked".fontcolor('red');  
                      document.getElementById("PIN").disabled=true;  
                      break;  
                   } else {  
                     msg.innerHTML = `${(3-count)} Attempts lefts`.fontcolor('red');  
                     break;  
                   }  
                } while(count<=3)  
            }  
        }  
    </script>  
</head>  
<body>  
    <fieldset>  
        <legend>Verify You PIN</legend>  
        <dl>  
            <dt>PIN</dt>  
            <dd><input type="text" id="PIN"></dd>  
            <dd id="msg"></dd>  
        </dl>  
        <button onclick="VerifyClick()">Verify</button>  
    </fieldset>  
</body>  
</html>  
  
Task: Write a program to convert number into words.

3. Iteration Statements  
- Iteration is a software design pattern used to access elements from a collection in sequential order.  
- It uses a function generator that yeilds value.  
- You can access elements from collection without using initilizer, condition and counter.  
  
a) for..in  
b) for..of

for .. in  
- It is used to read all properties from collection in sequential order.  
  
Syntax:  
    for(var property in collection)  
    {  
    }  
  
  for.. of  
- It is used to read all values from a collection in sequential order.  
  
Syntax:  
    for(var value of collection)  
    {  
    }

4. Jump Statements  
  
- return            : It terminates the execution  
- break            : It terminates the block  
- continue            : It skips the counter.  
  
Ex:  
<script>  
    for(var i = 1; i<=10; i++) {  
        if(i==5 || i==8) {  
            continue;  
        }  
        document.write(i + "<br>");  
    }  
</script>  
  
  
Ex:  
<script>  
    var products = [  
        {Name: "TV", Category:"Electronics"},  
        {Name: "Shirt", Category: "Fashion"},  
        {Name: "Watch", Category: "Electronics"},  
        {Name: "Jeans", Category: "Fashion"},  
        {Name: "Shoe", Category:"Footwear"}  
    ];  
    for(var item of products){  
        if(item.Category=="Fashion" || item.Category=="Electronics"){  
            continue;  
        }  
        document.write(item.Name + "<br>");  
    }  
</script>

## 5. Exception Handling Statement

- In computer programming 2 types of errors occur  
     
    1. Compile Time Errors  
    2. Run Time Errors  
  
- Compile time errors are syntactical errors, due to which program fails to compile and run.  
- Run time errors are dynamic, which occur during run time of application.  
- If any run time error occurs then application terminates abnormally.  
- To avoid abnormal termination in application we need Exception Handling.  
  
Syntax:  
        var x  = 10;  
        var y  = parseInt(prompt("Enter Number"));    => 2 [5]  0[exception]  
        var z = x / y;  
        console.log(`Z=${z}`);  
  
- Exception handling statements are  
  
        a) try            - It is monitoring block  
        b) catch        - It is handler block  
        c) throw        - It is exception block  
        d) finally        - It is the block that executes always.  
  
Ex:  
<script>  
    try {  
        var x = parseInt(prompt("Enter Number-1"));  
        var y = parseInt(prompt("Enter Number-2"));  
        if(y==0){  
            throw "Can't Divide By Zero";  
        }  
        if(y>x) {  
            throw "Number is too large..";  
        }  
        var z  = x / y;  
        document.write(`Division = ${z}<br>`);  
    }  
    catch(ex)  
    {  
        document.write(ex);  
    }  
    finally {  
        console.log(`Program End`);  
    }  
   
</script>  
  
# JavaScript Functions  
                                 
- Functions are used to "Refactor" the code.  
- Refactor is the process of extracting a set of lines and storing in reference of file or any function.  
  
Configuration:  
- Function can be configured in 2 ways  
**a) Function Expression  
    b) Function Declaration**  
  
# Function Expression  
- It is a technique where we allocate memory for function, but the functionality may change according to situation.  
  
Syntax:  
        let   name = function() {  }  
  
Ex:  
<script>  
     var names = ["John", "David", "Sam"];  
     var result = prompt("Enter What you want to Print?","Numbers|Names");  
     let Print;  
     if(result==="Numbers") {  
          Print = function(){  
                for(var i=1; i<=10; i++){  
                    document.write(i + "<br>");  
                     }  
                }  
    } else {  
         Print = function() {  
            for(var item of names) {  
                document.write(item + "<br>");  
            }  
         }  
    }  
     Print();  
</script>  
  
# Function Declaration  
- It is the process of allocating memory for a function to handle specific operation.  
- Memory allocated when function requested and it is destroyed when function is unsubscribed.  
  
Syntax:  
    function  Name()  
    {  
  
    }  
  
**Structure of Function:**  
- Every function declaration comprises of 3 phases  
        a) Declaration  
        b) Signature  
        c) Definition  
  
Syntax:      
        function Welcome(params)  
        {  
        }  
  
        function Welcome(params)        => Declaration  
        Welcome(params)                    => Signature  
        {  }                                        => Definition

# Function Parameters  
  
- Parameters are used to modify the function.  
- Function Parameters are 2 types  
    a) Formal Parameters  
    b) Actual Parameters  
  
Syntax:  
        function Print(userName)  
        {  
  
        }  
        Print("john");  
  
- Formal parameters are defined in function declaration.  
        "userName" is formal  
  
- Actual parameters are defined in function call.  
  
          "john"  is actual  
  
        userName = john  
  
Ex:  
<script>  
    function PrintNumbers(howMany)  
    {  
        for(var i=1; i<=howMany; i++) {  
            document.write(i + "<br>");  
        }  
    }  
    PrintNumbers(4);  
    PrintNumbers(18);  
</script>

Function => "Refactor"  
Function Expression  
Function Signature  
Function Definition  
Function Declaration  
  
                            Function Parameters  
- Parameters are used to modify the function  
- Parameters are 2 types  
    a) Formal      
    b) Actual  
- Formal parameters are defined in function declaration  
  
        function f1(a, b)        => a, b are formal parameters  
  
        f1(10, 20)                => 10, 20 are actual parameters  
  
        a = 10, b=20  
  
- A function parameter can be any type  
        a) Primitive  
        b) Non Primitive  
  
Ex:  
<script>  
    function Details(id, name, price, stock, cities, rating){  
        document.write(`  
            Id : ${id} <br>  
            Name: ${name} <br>  
            Price: ${price} <br>  
            Stock: ${stock} <br>  
            Cities: ${cities.toString()} <br>  
            Rating: ${rating.rate} [${rating.count}]  
        `);  
    }  
    Details(1, "TV", 34500.44, true, ["Delhi", "Hyd"], {rate:4.2, count:5000});  
</script>  
  
- You can define a function as parameter.  
- ***Functions are used as parameters to configure "call backs"***  
- Callback is a technique where functions are defined, which execute according to state and situation.  
  
Ex:  
<script>  
    function VerifyUser(password, success, failure)  
    {  
         if(password=="admin") {  
            success();  
         } else {  
            failure();  
         }  
    }  
    VerifyUser(prompt("Enter Password"),function(){  
        document.write("Login Success..");  
    }, function(){  
        document.write("Invalid Password");  
    })  
</script>  
  
- Every parameter defined in function is mandatory [required] and parameters have order dependency.  
  
Ex:  
<script>  
   function Details(id, name, price)  
   {  
     if(price==undefined) {  
        document.write(`  
        Id      : ${id} <br>  
        Name    : ${name} <br>  
    `);  
     } else {  
        document.write(`  
        Id      : ${id} <br>  
        Name    : ${name} <br>  
        Price   : ${price}  
    `);  
     }  
   }  
   Details(1, "TV");  
</script>

# Rest Parameters  
                                         
- JavaScript E6 introduced "Rest" parameters.  
- A single Rest parameter can handle multiple arguments.  
- Rest parameter is an array type.  
- Rest parameter is defined by using "..." with parameter name.  
  
        function  Name (...paramName)  
        {  
        }  
        Name(a, b, c, d, e)  
  
Ex:  
<script>  
   function Details(...product)  
   {  
      var [id, name, price, stock, cities, rating] = product;  
      document.write(`  
            Id      : ${id} <br>  
            Name    : ${name} <br>  
            Price   : ${price} <br>  
            Stock   : ${stock} <br>  
            Cities  : ${cities.toString()} <br>  
            Rating  : ${rating.rate} [${rating.count}]  
      `);  
   }  
   Details(1, "TV", 34000.33, true, ["Delhi","Hyd"], {rate:4.2, count:4500});  
</script>  
  
- Every function can have only one Rest parameter.  
- Rest parameter can be defined along with other parameters.  
- Rest parameter must be last parameter in formal list.  
  
Ex:  
<script>  
   function Details(title,...product)  
   {  
      var [id, name, price, stock, cities, rating] = product;  
      document.write(`  
            <h2>${title}</h2>  
            Id      : ${id} <br>  
            Name    : ${name} <br>  
            Price   : ${price} <br>  
            Stock   : ${stock} <br>  
            Cities  : ${cities.toString()} <br>  
            Rating  : ${rating.rate} [${rating.count}]  
      `);  
   }  
   Details( "Product Details" ,1, "TV", 34000.33, true, ["Delhi","Hyd"], {rate:4.2, count:4500});  
</script>  
  
- JavaScript ES6 supports "Spread Syntax".  
- Spread is a mechanism where one actual parameter value can spread into multiple formal parameters.  
  
Syntax:  
            function Name(a, b, c)  
            {  
            }  
  
            var args = [10, 20, 30];  
            Name(...args);

# Anonymous Functions  
  
- A function without name is known as Anonymous.  
- Anonymous functions are used to build and expression or for call back.  
- Individual Anonymous functions are accessed by using a technique called  
    "IIFE" [Immediately Invoked Function Expression]  
  
***Ex: Anonymous***  
<script>  
    (function(){  
        document.write("Welcome to Functions");  
    })();  
</script>  
  
***Ex: Expression Anonymous***  
  
<script>  
    const welcome = function(){  
        document.write("Welcome to Functions<br>");  
    };  
    welcome();  
    welcome();  
</script>

## # Function with Return

Function Parameters  
        a) REST and Spread  
Function Return  
        b) Terminate and store result in function memory  
  
FAQ: Can a function have multiple return statements?  
Ans:  Yes. Multiple return statements are used as conditional rendering statements.  
  
Ex:  
<script>  
    function Result(value){  
        if(isNaN(value)) {  
            return `Hello ! ${value}`;  
        } else {  
            value++;  
            return value;  
        }  
    }  
    document.write(Result("john"));  
</script>  
  
  
FAQ: Can we define "return" keyword in void method?  
Ans:  Yes.  
  
FAQ: What is the purpose of return keyword in void method?  
Ans : It is used to add "stubs" in program. A stub is used to terminate the execution.  
  
  
FAQ: What type of value a function can return?  
Ans:  Any type, primitive or non primitve.  
  
Ex:  
<script>  
    function Products(){  
        return [  
            {Name: "TV", Price:56000.33},  
            {Name: "Mobile", Price:21000.33}  
        ]  
    };  
    for(var item of Products())  
    {  
        document.write(item.Name + "<br>");  
    }  
</script>  
  
Note: A function can return function.  
  
            function Login()  
            {  
                return(...function definition....);  
            }  
  
  
            return { }        object  
            return ( )        function  
            return [ ]        array  
            return " "        string  
  
Ex:  
<script>  
    function Login()  
    {  
        return(  
             `  
              <dl>  
                 <dt>UserId</dt>  
                 <dd><input type="text"></dd>  
              </dl>  
              <button> Login </button>  
            `  
        )  
    }  
    document.write(Login());  
</script>  
  
Ex:  
<script>  
    function Login(title, label, controlType, btnText)  
    {  
        return(  
             `  
             <h2> ${title} </h2>  
              <dl>  
                 <dt>${label}</dt>  
                 <dd><input type=${controlType}></dd>  
              </dl>  
              <button> ${btnText} </button>  
            `  
        )  
    }  
    document.write(Login("Admin Login", "OTP", "number", "Verify"));  
</script>  
  
# Arrow Functions  
  
- It is a short hand technique for writing function expression.  
- Arrow uses the following chars  
  
        ( )            = function parameters  
        =>            = function definition and return  
        { }            = function definition if multiple statements are defined  
  
Ex:      
        function  hello()  
        {  
           return  "hello javascript";  
        }  
  
  
        var hello = ()=> "Hello JavaScript";  
  
Ex:  
<script>  
    const hello = () => "Hello JavaScript";  
    const addition = (a,b) => a + b;  
    const categories = () => ["Electronics", "Footwear"];  
  
    document.write(categories() + "<br>");  
    document.write(addition(10,30) + "<br>");  
    document.write(hello());  
</script>  
  
Ex:  
<script>  
     const welcome = (username,age) => `Hello ! ${username} you are ${age}`;  
     document.write(welcome("John", 22));  
</script>  
  
  
Ex:  
<script>  
***// With Arrow***  
         
        fetch("<http://fakestoreapi.com/products/categories&quot>;)  
        .then(response=> response.json())  
        .then(categories=> {  
            categories.map((category)=> document.write(`<li> ${category} </li>`));  
        })  
  
***//Without Arrow***  
        fetch("<http://fakestoreapi.com/products/categories&quot>;)  
        .then(function(response){  
            return response.json();  
        })  
        .then(function(categories){  
            categories.map(function(category){  
                document.write(`<li>${category}</li>`);  
            })  
        })  
</script>  
  
Ex:  
<script>  
    const msg = ()=> { document.write("Welcome<br>"); document.write("to functions in JS"); }  
           
    msg();  
</script>  
  
# Function Recursion  
 - It is a technique of calling a function with in the context of same function.  
  
        function f1(){  
          f1();  
        }  
     
- It is used to create batch operations.  
  
Ex:  
<script>  
    function Factorial(n)  
    {  
        if(n<=0) {  
            return 1;  
        }  
        return n \* Factorial(n-1);  
    }  
    document.write("Factorial of 6 is " + Factorial(6));  
</script>

# Function Closure  
  
- Every  function have a scope.  
- The scope of function can define members like variables and functions [inner].  
  
Syntax:  
    function  Outer()  
    {  
        var x;  
        function Inner(){  
            x=10;  
        }  
    }  
  
- Closure is a technique that allows the inner function to access members of outer function.  
- Closure will not allow outer function to access members of inner function.  
  
Ex:  
<script>  
   function Outer(){  
      var x = 10;  
      function Inner(){  
         var y = 20;  
         return x + y;  
      }  
      document.write(`result =` + Inner());  
   }  
   Outer();  
</script>

# Function Generator  
- It is used to yeild a value.  
- It is a unique reference value created for sequence of elements.  
- Generator verifies the values in a collection, and initializes the sequence to access and reads upto end.  
- Every generator function is defined with "\*" meta character.  
  
             \*   zero or more occurances  
  
Syntax:  
        function\*  GetDetails()            => Generator  
        {  
  
        }  
  
- Generator returns a value by using yield.  
  
         yield  "value | any";  
  
- Generator function requires object that returns 3 members  
  
        a) next()        read current and move to next value  
        b) throw()        if there is any exception  
        c) return        it terminates the execution  
  
- The next() returns 2 values  
  
        a) done        returns true if there are no more values to read.  
        b) value        returns the yield value  
  
Ex:  
<script>  
    function\* GetCategories(){  
        yield  "Electronics";  
        yield  "Footwear";  
        yield  "Fashion";  
        yield  "Men's Clothing";  
    }  
    var categories = GetCategories();  
    document.write(categories.next().value + "<br>");  
    document.write(categories.next().value + "<br>");  
    document.write(categories.next().value + "<br>");  
    document.write(categories.next().value + "<br>");  
    if(categories.next().done==true) {  
        document.write(`No more records to read <br>`);  
    }  
    document.write(categories.next().value + "<br>");  
</script>  
  
# Function Debounce      
- Bounce is an electronic term used with keying chars from any devices.  
- If user delays releasing of button then the function triggers several time.  
- You can delay the execution of function by keeping function inactive for specific duration of time.  
- It is controlled by using JavaScript  
        setTimeout()  
        clearTimeout()  
  
                                      **setTimeout()**  
- It is used to delay the function by specified amount of time.  
- Function is loaded into memory but is in inactive mode for specific duration.  
  
Syntax:  
        setTimeout(function(){} , interval)  
        clearTimeout(functionReference)  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function msg1(){  
            document.querySelector("p").innerHTML = "Hello !";  
        }  
        function msg2(){  
            document.querySelector("p").innerHTML = "How are you ?";  
        }  
        function msg3(){  
            document.querySelector("p").innerHTML = "Welcome to JavaScript";  
        }  
        var m1, m2, m3;  
        function DisplayMessage(){  
           m1 =  setTimeout(msg1, 3000);  
           m2 =  setTimeout(msg2, 5000);  
           m3 =  setTimeout(msg3, 10000);  
        }  
        function Stop2nd(){  
            clearTimeout(m2);  
        }  
    </script>  
</head>  
<body>  
    <button onclick="DisplayMessage()">Display</button>  
    <button onclick="Stop2nd()">Stop 2nd Message</button>  
    <p align="center"></p>  
</body>  
</html>  
  
                                    **setInterval()**  
- It is an event that executes the specified function at regular time intervals.  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function Timer(){  
            var now = new Date();  
            document.querySelector("p").innerHTML = now.toLocaleTimeString();  
        }  
        function bodyload(){  
            setInterval(Timer,1000);  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <p align="center"></p>  
</body>  
</html>

setTimeout()  
setInterval()  
  
Ex: Spinner  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Timer</title>  
    <link rel="stylesheet" href="../node\_modules/bootstrap/dist/css/bootstrap.css">  
    <script>  
        var count = 0;  
        function Loading(){  
            count++;  
            document.getElementById("status").innerHTML = count + " % ";  
            if(count==100) {  
                document.getElementById("statusContainer").style.display = "none";  
                document.getElementById("imageContainer").style.display = "block";  
            }  
        }  
        function FetchClick(){  
            document.getElementById("fetchContainer").style.display = "none";  
            document.getElementById("statusContainer").style.display = "block";  
            setInterval(Loading,100);  
        }  
    </script>  
</head>  
<body class="container-fluid d-flex justify-content-center align-items-center" style="height: 500px;">  
    <div>  
        <div id="fetchContainer" class="text-center">  
            <button onclick="FetchClick()" class="btn btn-primary">Fetch Image</button>  
        </div>  
        <div id="statusContainer" class="text-center" style="display: none;">  
            <div class="spinner-border"></div>  
            <p id="status"></p>  
            <p>Loading..</p>  
        </div>  
        <div id="imageContainer" style="display: none;">  
            <img src="../public/images/realme.jpg" width="300" height="300">  
        </div>  
    </div>  
</body>  
</html>  
  
  
Ex: Progress  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Timer</title>  
    <style>  
        progress {  
            width: 300px;  
            height: 30px;  
        }  
    </style>  
    <link rel="stylesheet" href="../node\_modules/bootstrap/dist/css/bootstrap.css">  
    <script>  
        var count = 0;  
        function Loading(){  
            count++;  
            document.getElementById("status").innerHTML = count + " % Completed";  
            document.getElementById("progress").value = count;  
            if(count==100) {  
                document.getElementById("statusContainer").style.display = "none";  
                document.getElementById("imageContainer").style.display = "block";  
            }  
        }  
        function FetchClick(){  
            document.getElementById("fetchContainer").style.display = "none";  
            document.getElementById("statusContainer").style.display = "block";  
            setInterval(Loading,100);  
        }  
    </script>  
</head>  
<body class="container-fluid d-flex justify-content-center align-items-center" style="height: 500px;">  
    <div>  
        <div id="fetchContainer" class="text-center">  
            <button onclick="FetchClick()" class="btn btn-primary">Fetch Image</button>  
        </div>  
        <div id="statusContainer" class="text-center" style="display: none;">  
            <div>  
                <progress id="progress" min="1" max="100" value="1"></progress>  
            </div>  
            <p id="status"></p>  
        </div>  
        <div id="imageContainer" style="display: none;">  
            <img src="../public/images/realme.jpg" width="300" height="300">  
        </div>  
    </div>  
</body>  
</html>

# Function Call Back & Promise  
  
- Function callback is a technique where there are set of functions defined, and functions execute according to state and situation.  
- Callback uses synchronous technique.  
- It is slow.  
  
Ex:  
<script>  
    function FetchData(url,success, failure)  
    {  
        if(url=="http://fakestoreapi.com&quot;) {  
            success(`Data Fetched Successfully from - ${url}`);  
        } else {  
            failure(`Unable to fetch data - Invalid URL : ${url}`);  
        }  
    }  
    FetchData(prompt("Enter URL"),(res)=>{  
        document.write(`Success: ${res}`);  
    }, (res)=>{  
        document.write(`Error: ${res}`);  
    })  
</script>

# Function Promise  
- Promise is a proxy.  
- Proxy refers to uncertenity in executing functions.  
- Promise comprises of 3 stages  
     
        a) Pending    : It is initial stage, not fulfilled or rejected.  
        b) Resolved    : Promise is fulfilled.  
        c) Rejected    : Promise is broken.  
        d) Finally        : It is executed always  
  
Syntax:  
        var name = new Promise((resolved, rejected)=>{  
              if(condition) {  
                resolved();  
              } else {  
                rejected();  
              }  
        });  
  
        name.then(()=>{on resolved}).catch(()=>{on rejected}).finally(()=>{always});  
  
- It uses implicitly "async" technique.  
- It is an unblocking technique, it will execute function without blocking other functions.  
- It is faster and improves performance of application.  
  
Ex:  
<script>  
    var FetchData = new Promise((resolved, rejected)=>{  
        var url = prompt("Enter URL");  
        if(url=="http://fakestoreapi.com&quot;)  
        {  
            resolved(`Data Fetched Successfully from Fakestore - ${url}`);  
        } else if (url=="api.nasa.gov"){  
            resolved(`Data Fetched from Nasa API - ${url}`);  
        }  
        else {  
            rejected(`Unable to fetch data - Invalid URL : ${url}`)  
        }  
    })  
  
    FetchData.then((res)=>{  
        document.write(`Success : ${res}<br>`);  
    }).catch((res)=>{  
        document.write(`Error: ${res}<br>`);  
    }).finally(()=>{  
        document.write(`Testing URL Completed`);  
    })  
  
</script>  
  
- Promise provides actions like  
        a) all()  
        b) race()  
- all() can execute multiple promises at the same time asynchronously.  
- race() can execute only one - synchronous  
  
Ex:  
<script>  
    var GetCategories = new Promise((resolve)=>{  
        resolve('Gets all categories list');  
    })  
    var GetProducts = new Promise((resolve)=>{  
        resolve('Gets all Products from API');  
    })  
    var GetCart = new Promise((resolve)=>{  
        resolve('Gets your cart details');  
    })  
  
    Promise.all([  
        GetCategories,  
        GetProducts,  
        GetCart  
    ]).then(messages => {  
        console.log(messages);  
    })  
</script>  
  
  
=> JavaScript OOP  
- In real world project development we have 3 major programming systems  
  
a) POPS  
b) OBPS  
c) OOPS  
  
1. POPS  
- Process Oriented Programming System  
- It can directly interact with hardware services.  
- It supports low level features.  
- It uses less memory.  
- It is faster.  
  
Ex: C, Pascal  
  
- Reusability issues  
- Separation issues  
- Extensibility issues  
- Dynamic memory allocation issues.

2. OBPS  
- Object Based Programming System  
- It supports reusability, dynamic memory, separation  
Ex:  
 Visual Basic, JavaScript  
- Extensibility issues  
- Code level security issues  
- No dynamic polymorphism  
  
3. OOPS  
- Object Oriented Programming System  
- Supports Reusability  
- Supports separation  
- Dynamic memory  
- Extensibility  
- Code level security  
- Dynamic Polymorphism  
  
Ex: C++, Java, C#  
  
- Can't directly interact with hardware services  
- Don't have low level features  
- Need more memory  
- Slow  
- Complex in configurations  
  
Note: JavaScript is an OBPS supports some of the features of OOP.  
                                 
Evolution of OOP  
- 1960's Alan Kay introduced : "Object"  
- 1967   Johan Olay, Nygaard : SIMULA 67 => Concept OOP => Code Reusability  
- 1970's  Trygve introduced : Small Talk => Concept => Code Separation => MVC  
   [Model View Controller]  
                 
            Model => Data Logic  
            View    => UI  
            Controller => Business Logic  
  
    Java            Spring  
    .NET            ASP.NET MVC  
    Python        Django, Flask  
    Ruby            Ruby on Rails  
    PHP            Cake PHP, Code Igniter  
    JavaScript    SPINE, Angular, React  
  
- 1975  C++  
- 1990  Java  
- 2003      C# , TypeScript

JavaScript OOP Features  
  
Modules in JavaScript  
- A module comprises of set of values, functions and classes.  
- A module is required to build library for application.  
- A library is a set of values, functions and classes, which you can import and use from any location.  
  
Ex:  
        Product.js            => module  
function  PrintName()    => local in access  
{  
  
}  
  
export function PrintName()    => global in access  
{  
}  
  
- "export" keyword is used to configure any member in a class so that it can be accessed outside module.  
  
- You can access any module in a page by importing the module.  
  
    <script type="module">  
        import  { PrintProduct } from "path.js";  
     
        PrintProduct();  
  
    </script>  
  
- To handle modules you need a module system to install.  
- If you are using JavaScript in browser, then browser have in-built module system called "ES Module".  
- Other module systems of JavaScript  
        Common JS  
        Require JS  
        UMD [Universal Module Distribution]  
        AMD [Asyncrhonous Module Distribution]  
  
Ex:  
1. Add a new folder  
    "library"  
2. Add subfolder  
    "modules"  
3. Add following files  
        product.module.js  
  
var ProductName = "Samsung TV";  
  
export function PrintName(){  
    return `Name=${ProductName}`;  
}  
  
        student.module.js  
  
var Name = "John";  
var Course = "UI Web Development";  
var Subject = "JavaScript";  
  
export function PrintStudent(){  
    return `Name=${Name}<br>Course=${Course}<br>Subject=${Subject}`;  
}  
  
export function PrintTitle(title){  
    return `${title}`;  
}  
  
- Add new Index.html page  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script type="module">  
        import { PrintName } from "../library/modules/product.module.js";  
        import { PrintStudent, PrintTitle } from "../library/modules/student.module.js";  
  
        document.querySelector("p").innerHTML = `<h3>${PrintTitle('Product Details')}</h3>${PrintName()}<br><h3>${PrintTitle('Student Info')}</h3>${PrintStudent()}`;  
    </script>  
</head>  
<body>  
    <p></p>  
     
</body>  
</html>

JavaScript OOP  
JavaScript Modules  
- export  
- Every module can have one "default" member.  
  
Syntax:  
        export  default  function Name()  
        {  
        }  
  
        import   Name  from  "path";  
  
- The default member is eagerly loaded. It is in the memory even if it is not in use.  
- Every module can have only one member are default and all others are not default members.  
  
Ex:  
student.module.js  
  
  
var Name = "John";  
var Course = "UI Web Development";  
var Subject = "JavaScript";  
  
export default function PrintStudent(){  
    return `Name=${Name}<br>Course=${Course}<br>Subject=${Subject}`;  
}  
  
export function PrintTitle(title){  
    return `${title}`;  
}  
  
product.module.js  
  
  
var ProductName = "Samsung TV";  
  
export default function PrintName(){  
    return `Name=${ProductName}`;  
}  
  
index.html  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script type="module">  
        import PrintName  from "../library/modules/product.module.js";  
        import PrintStudent, { PrintTitle } from "../library/modules/student.module.js";  
  
        document.querySelector("p").innerHTML = `<h3>${PrintTitle('Product Details')}</h3>${PrintName()}<br><h3>${PrintTitle('Student Info')}</h3>${PrintStudent()}`;  
    </script>  
</head>  
<body>  
    <p></p>  
     
</body>  
</html>

JavaScript Class  
- Class is a program template.  
- A template provides data and logic which you can implement and customize according to the requirements.  
- Class is reffered as an Entity or a Model or Blue Print.  
  
Configuring Class:  
- Class can be configured by using 2 methods  
***a) class expression  
    b) class declaration***  
  
Syntax: Declaration  
  
class  Employee  
{  
  
}  
  
Syntax: Expression  
  
var  Employee = class {  
  
 }  
  
Class Members:  
- Every JavaScript class can have only following as class members  
  
***1. Property  
    2. Method  
    3. Accessor  
    4. Constructor***  
  
FAQ: Can we define a variable as class member?  
Ans: No.  
  
FAQ: Can we define a function as class member?  
Ans: No.  
  
FAQ: Why variable and function is not allowed in class?  
Ans: They are immutable types. Class member can be only mutable type.  
  
FAQ: Can a class have variables and functions inside?  
Ans: Yes. As member of any method.

1. Property  
- Data in a class is stored in property.  
  
class  Product  
{  
    Property = value;  
}  
  
- Property name can be same as variable name but without any keywords.  
- Property value can be any type  
    a) Primitive type  
    b) Non-Primitive type  
- Property is accessible with in the class by using "this" keyword and outside class by using instance [object] of class.  
  
Syntax:  
        let obj = new className();  
        obj.Property;                // read  
        obj.Property = value;    // write  
  
Ex:  
<script>  
     
    class Product  
    {  
       Name = "Samsung TV";  
       Price = 45000.44;  
       Stock = true;  
       Cities = ["Delhi", "Hyd"];  
       Rating = {Rate:4.2, Count: 320}  
    }  
    let tv = new Product();  
    document.write(`Name=${tv.Name}<br>Price=${tv.Price}<br>Stock=${tv.Stock}<br>Cities=${tv.Cities}<br>Rating=${tv.Rating.Rate} [${tv.Rating.Count}]`);  
</script>  
  
2. Accessors  
- Accessors will give a fine grained control over the property in class.  
- Accessors can control read and write operations on property.  
- Accessors are 2 types  
         
        a) Getter  get()  
        b) Setter    set()  
  
- Getter can read value from a property.  
- Setter can write value into a property.  
  
Syntax:  
        get  AliasName()  
        {  
         return value;  
        }  
  
        set  AliasName(newName)  
        {  
            property = newName;  
        }  
  
Ex:  
<script>  
    var username = prompt("Enter User Name");  
    var role = prompt("Enter Your Role", "Customer|Admin");  
    var productname = prompt("Enter Product Name");  
  
    class Product  
    {  
        \_productName;  
  
        get ProductName(){  
            return this.\_productName;  
        }  
        set ProductName(newName){  
           if(role=="Admin") {  
            this.\_productName = newName;  
           } else {  
             document.write(`Hello ! ${username} You are not authorized to set product name`);  
           }  
        }  
    }  
    let obj = new Product();  
    obj.ProductName = productname;  
    if(obj.ProductName){  
        document.write(`Name=${obj.ProductName}`);  
    }  
</script>  
  
Ex:  
<script>  
    class Product  
    {  
        Name = "Samsung TV";  
        Rating = {  
            CustomerRating : {Rate:4.2, Count: 520},  
            VendorRating  : {  
                 MumbaiVendor: {Rate:4.1, Count:35},  
                 HydVendor: {Rate:3.7, Count: 47}  
            }  
        }  
        get MumbaiRating(){  
            return this.Rating.VendorRating.MumbaiVendor.Rate;  
        }  
        get HydRating(){  
            return this.Rating.VendorRating.HydVendor.Rate;  
        }  
    }  
    let tv = new Product();  
    document.write(`Name=${tv.Name} <br> Mumbai Vendor Rating : ${tv.MumbaiRating}<br>Hyd Vendor Rating : ${tv.HydRating}`)  
</script>

3. Method  
- A method defines logic to perform.  
- Method is mutable and function is immutable.  
  
Syntax:  
class Name  
{  
  Property = value;  
  Method() {  
  
  }  
}  
  
- Everything about method is same as function  
    parameters  
    return  
    recursion  
    debounce  
    closure etc..  
  
Ex:  
<script>  
    class Product  
    {  
        Name = "Samsung TV";  
        Price = 54600.44;  
        Qty = 2;  
        Total(){  
            return this.Qty \* this.Price;  
        }  
        Print(){  
            document.write(`Name=${this.Name}<br>Price=${this.Price}<br>Qty=${this.Qty}<br>Total=${this.Total()}`);  
        }  
    }  
    let obj = new Product();  
    obj.Print();  
</script>  
  
Ex: Dynamically Adding Data into Table using Class Object  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var products = [];  
        function LoadProducts(){  
            document.querySelector("tbody").innerHTML="";  
            for(var item of products){  
                var tr = document.createElement("tr");  
                var tdName = document.createElement("td");  
                var tdPrice = document.createElement("td");  
                var tdQty = document.createElement("td");  
                var tdTotal = document.createElement("td");  
  
                tdName.innerHTML = item.Name;  
                tdPrice.innerHTML = item.Price;  
                tdQty.innerHTML = item.Qty;  
                tdTotal.innerHTML  = item.Total;  
  
                tr.appendChild(tdName);  
                tr.appendChild(tdPrice);  
                tr.appendChild(tdQty);  
                tr.appendChild(tdTotal);  
  
                document.querySelector("tbody").appendChild(tr);  
            }  
        }  
        class Product  
        {  
            Name = "";  
            Price = 0;  
            Qty = 0;  
            Total(){  
                return this.Qty \* this.Price;  
            }  
        }  
         
        function ClearElements(){  
            document.getElementById("Name").value="";  
            document.getElementById("Price").value="";  
            document.getElementById("Qty").value="";  
        }  
  
         
        function AddClick(){  
            let obj = new Product();  
            obj.Name = document.getElementById("Name").value;  
            obj.Price = parseFloat(document.getElementById("Price").value);  
            obj.Qty = parseInt(document.getElementById("Qty").value);  
            var product = {  
                Name: obj.Name,  
                Price: obj.Price,  
                Qty: obj.Qty,  
                Total:obj.Total()  
            }  
            products.push(product);  
            LoadProducts();  
            ClearElements();  
        }  
    </script>  
</head>  
<body>  
    <dl>  
        <h2>Add New Product</h2>  
        <dt>Name</dt>  
        <dd><input type="text" id="Name"></dd>  
        <dt>Price</dt>  
        <dd><input type="text" id="Price"></dd>  
        <dt>Qty</dt>  
        <dd><input type="number" id="Qty"></dd>  
    </dl>  
    <button onclick="AddClick()">Add Product</button>  
    <hr size="5" noshade >  
    <table border="1" width="500">  
        <thead>  
            <tr>  
                <th>Name</th>  
                <th>Price</th>  
                <th>Qty</th>  
                <th>Total</th>  
            </tr>  
        </thead>  
        <tbody>  
  
        </tbody>  
    </table>  
</body>  
</html>

4. Constructor                                  
- It is a design pattern used for instantiation.  
- It is the process of creating an object for class.  
- It is a special type of method that is responsible for creating an object for class.  
- Every class have constructor defined implicitly.  
- JavaScript constructor is anonymous type.  
  
Syntax:  
    class  ClassName  
    {  
      constructor() {  
  
      }  
    }  
  
- Constructor is a special type of sub-routine that executes automatically for every object.  
  
    let obj = new ClassName();  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        class Database  
        {  
            constructor(dbName){  
                document.write(`Connected with ${dbName}<br>`);  
            }  
            Insert(){  
                document.write("Record Inserted");  
            }  
            Delete(){  
                document.write("Record Deleted");  
            }  
        }  
        function InsertClick(){  
            let obj = new Database(document.querySelector("select").value);  
            obj.Insert();  
        }  
        function DeleteClick(){  
            let obj = new Database(document.querySelector("select").value);  
            obj.Delete();  
        }  
    </script>  
</head>  
<body>  
    <select>  
        <option>Select Database</option>  
        <option>Oracle</option>  
        <option>MySql</option>  
        <option>Sql Server</option>  
    </select>  
    <button onclick="InsertClick()">Insert</button>  
    <button onclick="DeleteClick()">Delete</button>  
</body>  
</html>  
  
Note: JavaScript constructor can't overload, It can't be private or static etc..

Code Extensibility  
  
- Code extensibility can acheived by using 2 techniques  
  
a) Aggregration  
b) Inheritance

Aggregration  
- Aggregation is the process of accessing the members of one class in another class by using an instance of class without creating any relation between classes.  
- It is knows "Has-A-Relation".  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        class HDFC\_Bank\_Version1  
        {  
            Personal = "Personal Banking Services";  
            NRI = "NRI Banking Services";  
            Print(){  
                document.write(`${this.Personal}<br>${this.NRI}<br>`);  
            }  
        }  
        class HDFC\_Bank\_Version2  
        {  
            Loans = "Car or Bike Loans";  
            Print(){  
                let obj = new HDFC\_Bank\_Version1();  
                obj.Print();  
                document.write(`${this.Loans}`);  
            }  
        }  
        function InstallClick(){  
            var ver = document.querySelector("select").value;  
            switch(ver){  
                case "ver1":  
                let obj1 = new HDFC\_Bank\_Version1();  
                obj1.Print();  
                break;  
                case "ver2":  
                let obj2 = new HDFC\_Bank\_Version2();  
                obj2.Print();  
                break;  
            }  
        }  
    </script>  
</head>  
<body>  
    <h2>Install Bank App</h2>  
    <select>  
        <option value="-1">Choose Version</option>  
        <option value="ver1">Version-1</option>  
        <option value="ver2">Version-2</option>  
    </select>  
    <button onclick="InstallClick()">Install</button>  
</body>  
</html>

Inheritance  
- It is the process of configuring relation between classes.  
- A class can extend another class.  
  
Syntax:  
    class A  
    {  
    }  
    class B extends A  
    {  
    }  
  
- class "A" is known as "super class" [base class].  
- class "B" is known as "derived class".  
- Derived class is in relation with Super class hence it is reffered as "Is-A-Relation".  
- Super class members are accessible to derived class by using "super" keyword.  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        class HDFC\_Bank\_Version1  
        {  
            Personal = "Personal Banking Services";  
            NRI = "NRI Banking Services";  
            Print(){  
                document.write(`${this.Personal}<br>${this.NRI}<br>`);  
            }  
        }  
        class HDFC\_Bank\_Version2 extends HDFC\_Bank\_Version1  
        {  
            Loans = "Car or Bike Loans";  
            Print(){  
                super.Print();  
                document.write(`${this.Loans}`);  
            }  
        }  
        function InstallClick(){  
            var ver = document.querySelector("select").value;  
            switch(ver){  
                case "ver1":  
                let obj1 = new HDFC\_Bank\_Version1();  
                document.write("<h2>Version-1 Features</h2>");  
                obj1.Print();  
                break;  
                case "ver2":  
                document.write("<h2>Version-2 Features</h2>");  
                let obj2 = new HDFC\_Bank\_Version2();  
                obj2.Print();  
                break;  
            }  
        }  
    </script>  
</head>  
<body>  
    <h2>Install Bank App</h2>  
    <select>  
        <option value="-1">Choose Version</option>  
        <option value="ver1">Version-1</option>  
        <option value="ver2">Version-2</option>  
    </select>  
    <button onclick="InstallClick()">Install</button>  
</body>  
</html>  
  
Note: OOP rule for inheritance is : If super class contains a constructor which is extended by derived class, then the derived class constructor must call super constructor.  
  
Syntax:  
    class Super  
    {  
        constructor(){ }  
    }  
    class Derived extends Super  
    {  
        constructor() {  
            super();  
        }  
    }  
    let obj = new Derived();  
  
Ex:  
<script>  
    class Super  
    {  
        constructor(){  
            document.write("Super Class Constructor<br>");  
        }  
    }  
    class Derived extends Super  
    {  
        constructor(){  
            super();  
            document.write("Derived Class Constructor");  
        }  
    }  
    let obj = new Derived();  
</script>  
  
- Class will support various inheritance approaches  
    a) Single  
    b) Multi Level  
  
- Single inheritance defines one derived class extending super class.  
- Multi Level inheritance is extending the derived class.  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        class HDFC\_Bank\_Version1  
        {  
            Personal = "Personal Banking Services";  
            NRI = "NRI Banking Services";  
            Print(){  
                document.write(`${this.Personal}<br>${this.NRI}<br>`);  
            }  
        }  
        class HDFC\_Bank\_Version2 extends HDFC\_Bank\_Version1  
        {  
            Loans = "Car or Bike Loans";  
            Print(){  
                super.Print();  
                document.write(`${this.Loans}<br>`);  
            }  
        }  
        class HDFC\_Bank\_Version3 extends HDFC\_Bank\_Version2  
        {  
            AGRI = "Govt. Bank Schemes";  
            Print(){  
                super.Print();  
                document.write(`${this.AGRI}`)  
            }  
        }  
        function InstallClick(){  
            var ver = document.querySelector("select").value;  
            switch(ver){  
                case "ver1":  
                let obj1 = new HDFC\_Bank\_Version1();  
                document.write("<h2>Version-1 Features</h2>");  
                obj1.Print();  
                break;  
                case "ver2":  
                document.write("<h2>Version-2 Features</h2>");  
                let obj2 = new HDFC\_Bank\_Version2();  
                obj2.Print();  
                break;  
                case "ver3":  
                document.write("<h2>Version-3 Features</h2>");  
                let obj3 = new HDFC\_Bank\_Version3();  
                obj3.Print();  
                break;  
            }  
        }  
    </script>  
</head>  
<body>  
    <h2>Install Bank App</h2>  
    <select>  
        <option value="-1">Choose Version</option>  
        <option value="ver1">Version-1</option>  
        <option value="ver2">Version-2</option>  
        <option value="ver3">Version-3</option>  
    </select>  
    <button onclick="InstallClick()">Install</button>  
</body>  
</html>

Polymorphism  
- Poly means Many  
- Morphos means forms.  
- Polymorphism in OOP is the process of creating a base class object which can use the memory of multiple derived classes.  
- It allows to overload the memory and handle various functionalities using one component.  
  
Ex:  
<script>  
    class Employee  
    {  
        FirstName;  
        LastName;  
        Designation;  
        Print(){  
            document.write(`${this.FirstName} ${this.LastName} - ${this.Designation}<br>`);  
        }  
    }  
    class Developer extends Employee  
    {  
        FirstName = "Raj";  
        LastName = "Kumar";  
        Designation = "Developer";  
        Role = "Developer Role : Build, Debug, Test";  
        Print(){  
            super.Print();  
            document.write(`${this.Role}`);  
        }  
    }  
    class Admin extends Employee  
    {  
        FirstName = "Kiran";  
        LastName = "Rao";  
        Designation = "Admin";  
        Role = "Admin Role : Authorizations";  
        Print(){  
            super.Print();  
            document.write(`${this.Role}`);  
        }  
    }  
    class Manager extends Employee  
    {  
        FirstName = "Tom";  
        LastName = "Hanks";  
        Designation = "Manager";  
        Role = "Manager Role : Approvals";  
        Print(){  
            super.Print();  
            document.write(`${this.Role}`);  
        }  
    }  
    let employees = new Array(new Developer(), new Admin(), new Manager());  
    var designation = prompt("Enter Designation");  
    for(var employee of employees){  
        if(employee.Designation==designation){  
            employee.Print();  
        }  
    }  
</script>

# JavaScript DS [Data Structure]  
- Data Structure in programming is about managing data for various situations.  
- It is the process of sorting, filtering, adding, removing etc..  
- You can create custom types to manage data.  
- The commonly used Data Structures in JavaScript are..  
        a) Stack  
        b) Queue  
        c) Linked List  
        d) HashTable  
        e) Binary Tree  
        f) Graph etc..  
  
Stack:  
- It uses LIFO.  
- Last-In-First-Out  
- Last Value added into collection will be the first value to read.  
  
Methods:  
    pop()  
    peek()  
    push()  
    size()  
  
Ex:  
stack.js  
  
  
export class Stack  
{  
    data = {};  
    length = 0;  
    pop(){  
      this.length--;  
      var value = this.data[this.length];  
      delete this.data[this.length];  
      return value;  
    }  
    push(value){  
       this.data[this.length] = value;  
       this.length++;  
    }  
    peek(){  
        return this.data[this.length-1];  
    }  
    size(){  
        return this.length;  
    }  
}  
  
ds.html  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script type="module">  
        import { Stack } from "../library/ds/stack.js";  
        let collection = new Stack();  
        collection.push("First");  
        collection.push("Second");  
        collection.push("Third");  
        console.log(collection.size());  
        console.log(collection.pop());  
        console.log(collection.size());  
        console.log(collection.peek());  
        console.log(collection.size());  
    </script>  
</head>  
<body>  
     
</body>  
</html>  
  
  
Queue:  
- It uses FIFO  
- First value pushed in will the value to read first.  
  
Methods:  
        enqueue()  
        dequeue()  
        size()  
  
Ex:  
queue.js  
  
  
export class Queue  
{  
    data = [];  
    length = 0;  
    enqueue(value){  
        this.data.push(value);  
        this.length++;  
    }  
    dequeue(){  
       return this.data.shift();  
    }  
    size(){  
  
    }  
}  
  
ds.html  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script type="module">  
        import { Queue } from "../library/ds/queue.js";  
        let collection = new Queue();  
        collection.enqueue("First");  
        collection.enqueue("Second");  
        collection.enqueue("Third");  
        console.log(collection.dequeue());  
        console.log(collection.length);  
    </script>  
</head>  
<body>  
     
</body>  
</html>

# JavaScript Browser Events  
  
What is an Event?  
- Event is a message sent by sender to its subscriber in order to notify the change.  
- Event follows a software design pattern called "Observer", which is a communication pattern.  
- Event uses a delegate mechanism, delegate is function pointer.  
  
Syntax:  
        function  InsertClick()                => Subscriber  
        {  
        }  
        <button onclick="InsertClick()">    => Sender  
  
- Subscriber comprises of actions to perform.  
- Sender sends notification to subscriber.

Event Handler  
What is Event Handler?  
- Every element uses Event and Event Handler  
- Event defines when the action is performed.  
- Event Handler defines when and which action to perform.  
  
Syntax:  
        onclick                        => event  
        onclick="InsertClick()"        => event handler  
  
What are Event Arguments?  
- Every event can have 2 default arguments  
  
**a) event  
        b) this**  
  
        <button onclick="InsertClick(this, event)">  
  
***- "this" is used to send information about current object to the subscriber***.  
         
         Current Object    :  <button>  
         Object Info.        :  id, name, className, width, height, value, disabled etc..  
  
***- "event" is used to send information about current event to the subscriber.***  
  
        Current Event    : onclick  
        Event Info        : clientX, clientY, shiftKey, ctrlKey, altKey, keyCode, which etc..  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function InsertClick(obj){  
            document.write(`  
                Button Id       : ${obj.id} <br>  
                Button Name     : ${obj.name} <br>  
                Button Class    : ${obj.className}  
            `);  
        }  
    </script>  
</head>  
<body>  
    <button id="btnInsert" name="Insert" class="btn btn-primary" onclick="InsertClick(this)">Insert</button>  
     
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function InsertClick(name, id){  
            document.write(`  
                Button Name : ${name} <br>  
                Button Id :  ${id}  
            `);  
        }  
    </script>  
</head>  
<body>  
    <button id="btnInsert" name="Insert" class="btn btn-primary" onclick="InsertClick(this.name, this.id)">Insert</button>  
     
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function InsertClick(e){  
            document.write(`  
                X Position      : ${e.clientX} <br>  
                Y position      : ${e.clientY} <br>  
                Ctrl Key        : ${e.ctrlKey}  
            `);  
        }  
    </script>  
</head>  
<body>  
    <button id="btnInsert" name="Insert" class="btn btn-primary" onclick="InsertClick(event)">Insert</button>  
     
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function InsertClick(e, obj){  
            document.write(`  
                X Position      : ${e.clientX} <br>  
                Y position      : ${e.clientY} <br>  
                Ctrl Key        : ${e.ctrlKey} <br>  
                Button Id       : ${obj.id}  
            `);  
        }  
    </script>  
</head>  
<body>  
    <button id="btnInsert" name="Insert" class="btn btn-primary" onclick="InsertClick(event, this)">Insert</button>  
     
</body>  
</html>  
  
- You can send custom arguments.  
- You can send custom argunment along with default arguments.  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function InsertClick(title, e, obj, product){  
            document.write(`  
                <h2>${title}</h2>  
                X Position      : ${e.clientX} <br>  
                Y position      : ${e.clientY} <br>  
                Ctrl Key        : ${e.ctrlKey} <br>  
                Button Id       : ${obj.id} <br>  
                Product Price   : ${product.Price}  
            `);  
        }  
    </script>  
</head>  
<body>  
    <button id="btnInsert" name="Insert" class="btn btn-primary" onclick="InsertClick('Welcome to Events',event, this, {Name:'TV', Price:56000.33})">Insert</button>  
     
</body>  
</html>  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
       function Player(action){  
         var p = document.querySelector("p");  
         switch(action){  
            case "Play":  
            p.innerHTML = "Paying Songs..";  
            break;  
            case "Pause":  
            p.innerHTML = "Paused..";  
            break;  
            case "Stop":  
            p.innerHTML = "Stopped Playing";  
            break;  
            case "Close":  
            p.innerHTML = "Player Closed..";  
            break;  
         }  
       }  
    </script>  
</head>  
<body>  
    <button onclick="Player('Play')">Play</button>  
    <button onclick="Player('Pause')">Pause</button>  
    <button onclick="Player('Stop')">Stop</button>  
    <button onclick="Player('Close')">Close</button>  
    <p></p>  
</body>  
</html>

Event Listener  
What is Event Listener?  
- It is a function defined with callback to configure event for any HTML element dynamically.  
- Element uses the method "addEventListener()" which adds a funcitonality for dynamically created element.  
  
Syntax:  
        element.addEventListener("event", function(){ })  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function bodyload(){  
            var button = document.createElement("button");  
            button.innerHTML = "Insert";  
            button.addEventListener("click",()=>{  
                document.write("Record Inserted");  
            })  
            document.getElementById("container").appendChild(button);  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <div id="container">  
  
    </div>  
</body>  
</html>  
  
Browser Events or Events Types:-  
  
1. Mouse Events  
2. Keyboard Events  
3. Button Events  
4. Form Events  
5. Element State Events  
6. Touch Events  
7. Clipboard Events  
8. Timer Events etc..  
  
                                        Mouse Events  
onmouseover  
onmouseout  
onmousedown  
onmouseup  
onmousemove  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        function GetPosition(e){  
            var flag = document.getElementById("flag");  
            flag.style.position = "fixed";  
            flag.style.left = e.clientX + "px";  
            flag.style.top = e.clientY + "px";  
            document.querySelector("div").innerHTML = `X = ${e.clientX} <br> Y=${e.clientY}`;  
        }  
    </script>  
</head>  
<body onmousemove="GetPosition(event)">  
    <div style="height:1000px"></div>  
    <img id="flag" src="../public/images/flag.gif" width="50" height="50">  
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <style>  
        @keyframes ZoomIn  
        {  
            from {  
                width: 100px;  
                height: 100px;  
            }  
            to {  
                width: 300px;  
                height: 400px;  
            }  
        }  
    </style>  
    <script>  
        function ZoomIn(){  
            var img = document.querySelector("img");  
            img.style.animationName = "ZoomIn";  
            img.style.animationDuration= "3s";  
        }  
    </script>  
</head>  
<body>  
    <img onmousedown="ZoomIn()" onmouseup="ZoomOut()" src="../public/images/realme.jpg" width="100" height="100">  
</body>  
</html>

Mouse Events  
    onmouseover  
    onmouseout  
    onmousedown  
    onmouseup  
    onmousemove  
Keyboards  
    onkeyup        ]    Good for dealing with chars  
    onkeydown    ]  
    onkeypress    ]    Good for dealing with char codes  
Button Events  
    onclick                single click  
    ondblclick                double click  
    oncontextmenu        right click  
    onselectstart            drag selection  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        document.onselectstart = function(){  
            return false;  
        }  
        document.oncontextmenu = function(){  
            alert('Right Click Disabled');  
            return false;  
        }  
    </script>  
</head>  
<body>  
    <img ondblclick="window.open('../public/images/asp.jpg','ASP','width=400 height=400')" src="../public/images/asp.jpg" width="100" height="100">  
    <p>double click to open</p>  
</body>  
</html>  
  
Clipboard Events  
- oncut  
- oncopy  
- onpaste  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
       function cut(){  
        document.querySelector("p").innerHTML = "Removed and Copied to Clipboard";  
       }  
       function copy(){  
        document.querySelector("p").innerHTML = "Copied to Clipboard";  
       }  
       function paste(){  
        document.querySelector("p").innerHTML = "Inserted from Clipboard";  
       }  
         
    </script>  
</head>  
<body>  
    <textarea oncut="cut()" oncopy="copy()" onpaste="paste()" rows="4" cols="40">  
         
    </textarea>  
    <p></p>  
    <textarea rows="3" cols="30" oncut="return false" oncopy="return false">some text...</textarea>  
</body>  
</html>  
  
Element State Events:  
onblur  
onfocus  
onchange  
  
Touch Events  
ontouchstart  
ontouchend  
ontouchmove  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
       function GetDetails(course){  
           var p = document.querySelector("p");  
          switch(course){  
            case "asp":  
             p.innerHTML = "ASP.NET - Server Side Programming";  
             break;  
            case "ui":  
             p.innerHTML = "UI Web Development - Building Web Applications";  
             break;  
          }  
       }  
    </script>  
</head>  
<body>  
    <img ontouchstart="GetDetails('asp')" src="../public/images/asp.jpg" width="100" height="100">  
    <img ontouchstart="GetDetails('ui')" src="../public/images/ui.jpg" width="100" height="100">  
    <p></p>  
</body>  
</html>  
  
Timer Events  
- setInterval()  
- clearInterval()  
- setTimeout()  
- clearTimeout()  
  
Form Events  
- onsubmit  
- onreset  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
       
    </script>  
</head>  
<body>  
    <form onsubmit="alert('Form will submit data to server')" onreset="alert('It resets form data')">  
        User Name : <input type="text" name="UserName">  
        <button>Submit</button>  
        <button type="reset">Reset</button>  
    </form>  
</body>  
</html>

# JavaScript Browser Objects  
- JavaScript browser objects  
    window  
    location  
    navigator  
    history  
    document

1. window  
- It provides a set of propeties and methods to control browser window  
    open()      
    close()  
    print()  
    alert()  
    confirm()  
    prompt() etc..  
  
 2. location  
- It provides properties and methods that are used to get client location details.  
  
    host  
    href  
    pathname  
    port  
    protocol  
    search  
    hash  
  
Ex:  
<script>  
    document.write(`  
        IP Address      : ${location.host}  <br>  
        Port Number     : ${location.port}  <br>  
        Protocol        : ${(location.protocol=="http:")?"You are Using Live Server":"Please run app on live server"} <br>  
        URL             : ${location.href} <br>  
        Path Name       : ${location.pathname}  
    `);  
</script>  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <link rel="stylesheet" href="../node\_modules/bootstrap/dist/css/bootstrap.css">  
    <script>  
         var userDetails = {  
            "Email": "[john11@gmail.com](https://mail.google.com/mail/?view=cm&fs=1&to=john11%40gmail.com&authuser=0)",  
            "Password": "john@123"  
         }  
         function ContinueClick(){  
            var email = document.getElementById("txtEmail").value;  
            if(email==userDetails.Email) {  
                location.href = "shopping-online.html";  
            } else {  
                location.href="error.html";  
            }  
         }  
    </script>  
</head>  
<body class="container-fluid">  
    <div class="d-flex justify-content-center align-items-center" style="height:500px">  
        <div class="w-25">  
            <div>  
                <h3>Signin</h3>  
                <div id="emailContainer">  
                    <label class="form-label fw-bold">Your Email</label>  
                    <div>  
                        <input type="text" class="form-control" id="txtEmail">  
                    </div>  
                    <div class="mt-2">  
                        <button onclick="ContinueClick()" class="btn btn-warning w-100">Continue</button>  
                    </div>  
                </div>  
                <div id="passwordContainer" style="display: none;">  
                    <label class="form-label fw-bold">Your Password</label>  
                    <div>  
                        <input type="password" class="form-control" id="txtPassword">  
                    </div>  
                    <div class="mt-2">  
                        <button class="btn btn-warning w-100">Login</button>  
                    </div>  
                </div>  
            </div>  
        </div>  
    </div>  
</body>  
</html>  
  
Ex:  location.search  
  
search.html  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Search</title>  
</head>  
<body>  
    <div align="center">  
      <form action="results.html">  
        <h1>Google</h1>  
        <input type="text" name="searchstring" size="40">  
        <p>  
            <button>Search</button>  
        </p>  
      </form>  
    </div>  
</body>  
</html>  
  
results.html  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var data = [  
            {Topic: "JavaScript", Contents: ["Variables", "DataTypes"]},  
            {Topic: "CSS", Contents: ["Selectors", "Units"]}  
        ];  
        function bodyload(){  
            var str = location.search;  
            var word = str.substring(str.indexOf("=") + 1);  
            var topic = data.find(item => item.Topic==word);  
            topic.Contents.map(item=>{  
                document.querySelector("p").innerHTML += item + "<br>";  
            })  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <h3>Search Results</h3>  
    <p></p>  
</body>  
</html>  
  
Ex: location.hash  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <style>  
        .topic {  
            width: 350px;  
            border:1px solid black;  
            background-color: black;  
            color:white;  
            padding: 10px;  
            margin:10px;  
        }  
        .main {  
            display: flex;  
        }  
        .menu {  
            font-size: 25px;  
            text-align: center;  
        }  
        .topic:target {  
            background-color: yellow;  
            color:black;  
            font-size: 40px;  
            transition: 2s;  
        }  
    </style>  
    <script>  
        function RecentClick(){  
            var topic = location.hash;  
            var p = document.querySelector("p");  
            switch(topic){  
                case "#html":  
                p.innerHTML = "Recently viewed HTML";  
                break;  
                case "#css":  
                p.innerHTML = "Rate experince with CSS";  
                break;  
                case "#js":  
                p.innerHTML = "Next you try Advanced JS";  
                break;  
            }  
        }  
    </script>  
</head>  
<body>  
    <div class="menu">  
        <span><a href="#html">HTML</a></span> |  
        <span><a href="#css">CSS</a></span> |  
        <span><a href="#js">JavaScript</a></span>  
    </div>  
    <div class="main">  
        <div class="topic" id="html">  
            <h2>HTML</h2>  
            <ol>  
                <li>Semantic Elements</li>  
                <li>Normal Elements</li>  
                <li>Text formatting</li>  
            </ol>  
        </div>  
        <div class="topic" id="css">  
            <h2>CSS</h2>  
            <ol>  
                <li>Selectors</li>  
                <li>Units</li>  
                <li>Rules</li>  
            </ol>  
        </div>  
        <div class="topic" id="js">  
            <h2>JavaScript</h2>  
            <ol>  
                <li>Variables</li>  
                <li>Operators</li>  
                <li>Statements</li>  
            </ol>  
        </div>  
    </div>  
    <div>  
        <button onclick="RecentClick()">Recently Viewed</button>  
        <p></p>  
    </div>  
</body>  
</html>

3. Navigator  
- It provides properties and methods that are used to get client browser details  
  
    appName                    Browser Family  
    appVersion                Browser Version  
    cookieEnabled  
    geoLocation  
    plugins[]  
    mimeTypes[]  
  
Ex:  
<script>  
    if(navigator.cookieEnabled){  
        document.write("Cookies Allowed on your browser");  
    } else {  
        document.write("Cookies Disabled - Please Enable cookies on your browser to signin");  
    }  
</script>  
  
Ex:  
<script>  
    navigator.geolocation.getCurrentPosition(position=>{  
        document.write(`  
            Latitude : ${position.coords.latitude} <br>  
            Longitude: ${position.coords.longitude}  
        `);  
    })  
</script>

4. history  
- It is a JavaScript object that provides set of properties and methods to control current browsing history.  
  
        length  
        back()  
        forward()  
        goto()  
     
Ex:  
<script>  
    if(history.length>3){  
        alert('Please register for more');  
        location.href="../public/register.html";  
    } else {  
        document.write("You can view max 3 pages for free");  
    }  
</script>  
  
Syntax:  
    <button onclick="history.back()"> Back </button>

# JavaScript AJAX  
  
- Asynchronous JavaScript And XML  
- It allows to handle Partial Post back.  
- It allows to add new details to page without reloading the page.  
- It can submit only specific portion of page.  
- It improves the performance of application.  
- JavaScript AJAX request is managed by using "XMLHttpRequest" object.  
  
            jQuery            $.ajax()              
            react            axios                 =>  XMLHttpRequest  
            angular        HttpClient  
  
Syntax:  
        var http = new XMLHttpRequest();  
  
**- JavaScript Ajax object properties and method**  
  
**1. onreadystatechange**        : It is a function that executes when Ajax request is made  
    **2.** **readyState**                    : It defines the current Ajax status  
                     
                                            1        : Initial State - before request starts  
                                            2        : request started  
                                            3        : request processed  
                                            4        : response ready  
  
**3. status**                            : It returns status code    => 404  
                                            2xx        success  
                                            3xx        redirection  
                                            4xx        client side issues  
                                            5xx        server side issues  
  
**4. statusText**                    : It returns status text => Not Found  
  
**5. open()**                            : It makes a request to any URL or File  
  
**6. send()**                            : It sends data as response to client.  
                                      plain text, XML, JSON  
  
    **7. response**                        : It gets all response details.  
  
    **8. responseText**                : It returns plain text as response.  
  
**9. responseXML**                : It returns XML data as response.  
  
Note: JSON data is sent as response by using "JSON.parse()"  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script>  
        var http = new XMLHttpRequest();  
        function FetchClick(){  
            var now = new Date();  
            document.getElementById("fetchedTime").innerHTML = `Data Fetched on : ${now.toLocaleTimeString()}`;  
             
            http.open("get","../data/help.txt");  
            http.onreadystatechange = function(){  
                if(http.readyState==4){  
                    document.querySelector("p").innerHTML = `<pre>${http.responseText}</pre>`;  
                }  
            }  
            http.send();  
        }  
        function GetData(){  
            http.open("get", "../data/product.json");  
            http.onreadystatechange = function(){  
                if(http.readyState==4){  
                    var data = JSON.parse(http.responseText);  
                    document.querySelector("p").innerHTML = `<pre>${data.title}<br><img src=${data.photo} width="200" height="300"></pre>`;  
                }  
            }  
            http.send();  
        }  
        function bodyload(){  
            var now = new Date();  
            document.getElementById("time").innerHTML = `Page Last Loaded On : ${now.toLocaleTimeString()}`;  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <div id="time"></div>  
    <button onclick="FetchClick()">Fetch Details</button>  
    <button onclick="GetData()">Get Data</button>  
    <div id="fetchedTime"></div>  
    <p></p>  
</body>  
</html>

# JavaScript State Management  
  
    a) Local Storage  
    b) Session Storage  
    c) Cookies  
    d) Query String  
  
Local Storage  
- It is a member of window object.  
- It provides methods  
        **setItem()  
        getItem()  
        removeItem()  
        clear()**- It allocates memory which is permanent.  
- It is accessible to other tabs.  
- It is available even when the browser is closed.  
- It is not accessible across browsers.  
  
EX:  
**login.html**  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Login</title>  
    <script>  
        function LoginClick(){  
            var userName = document.getElementById("UserName").value;  
            localStorage.setItem("username", userName);  
            location.href = "home.html";  
        }  
    </script>  
</head>  
<body>  
    User Name : <input type="text" id="UserName"> <button onclick="LoginClick()">Login</button>  
</body>  
</html>  
  
**home.html**  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Home</title>  
    <script>  
        function bodyload(){  
            var user = localStorage.getItem("username");  
            if(user==null){  
                location.href = "login.html";  
            } else {  
                document.getElementById("user").innerHTML = `Hello ! ${user}`;  
            }  
             
        }  
        function LogoutClick(){  
            localStorage.removeItem("username");  
            location.href="login.html";  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <h2>Home -  <span id="user"></span></h2>  
    <button onclick="LogoutClick()">Logout</button>  
</body>  
</html>  
  
Session Storage  
- It is a member of window object.  
- It is temporary  
- It is erased when browser is closed.  
- It is not accessible across tabs and browsers.  
- Methods are same as localStorage  
  
Ex:  
**login.html**  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Login</title>  
    <script>  
        function LoginClick(){  
            var userName = document.getElementById("UserName").value;  
            sessionStorage.setItem("username", userName);  
            location.href = "home.html";  
        }  
    </script>  
</head>  
<body>  
    User Name : <input type="text" id="UserName"> <button onclick="LoginClick()">Login</button>  
</body>  
</html>  
  
**home.html**  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Home</title>  
    <script>  
        function bodyload(){  
            var user = sessionStorage.getItem("username");  
            if(user==null){  
                alert(`Your Session Expired - Login Again`);  
                location.href = "login.html";  
            } else {  
                document.getElementById("user").innerHTML = `Hello ! ${user}`;  
            }  
             
        }  
        function LogoutClick(){  
            sessionStorage.removeItem("username");  
            location.href="login.html";  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <h2>Home -  <span id="user"></span></h2>  
    <button onclick="LogoutClick()">Logout</button>  
</body>  
</html>  
  
Note: Local and Session storage are not accessible to server.  
  
Cookies  
- It is a member of document object.  
- It stores client details and appends in client computer.  
- It can save details upto specified date and  time.  
- It is accessible to server.  
  
Ex:  
login.html  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Login</title>  
    <script>  
        function LoginClick(){  
            var userName = document.getElementById("UserName").value;  
            document.cookie = `UserName=${userName}`;  
            location.href = "home.html";  
        }  
    </script>  
</head>  
<body>  
    User Name : <input type="text" id="UserName"> <button onclick="LoginClick()">Login</button>  
</body>  
</html>  
  
  
home.html  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Home</title>  
    <script>  
        function bodyload(){  
           var str = document.cookie;  
           var uname = str.substring(str.indexOf("=") + 1);  
           if(uname=="") {  
              alert("Session Expired");  
              location.href="login.html";  
           } else {  
               document.getElementById("user").innerHTML = `Hello ! ${uname}`;  
           }  
             
        }  
        function LogoutClick(){  
            document.cookie="";  
            location.href="login.html";  
        }  
    </script>  
</head>  
<body onload="bodyload()">  
    <h2>Home -  <span id="user"></span></h2>  
    <button onclick="LogoutClick()">Logout</button>  
</body>  
</html>

# jQuery  
  
- jQuery is a JavaScript library for building UI.  
- A library provides pre-defined functions.  
- You can implement and customize according to requirement.  
- John Resig introduced jQuery in 2006.  
- Aim of  jQuery is "write less - do more".  
- It provides functions for  
**a) DOM Manipulations  
    b) Ajax  
    c) Validations  
    d) Effects  
    e) UI Components**  
Setup and Install jQuery Library for Project:  
  
    > npm install jquery --save  
  
Configure jQuery in page:  
  
    <script src="../node\_modules/jquery/dist/jquery.js"> </script>  
     
- jQuery library is defined in anonymous function, which you can access using "$" reference.  
  
    <script>  
        $(function(){  
            // all jquery functions are accessible here..  
        })  
    </script>  
                        (or)  
  
     <script>  
        $(document).ready(function(){  
  
        })  
    </script>  
  
jQuery DOM interactions:  
- jQuery uses all CSS selectors to access HTML elements.  
     
            $("type")  
            $("#id")  
            $(".class")  
            $("parent > child")  
             
- jQuery doesn't require event handler, It uses all event listners.  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script src="../node\_modules/jquery/dist/jquery.js"></script>  
    <script>  
        $(function(){  
             $("h1").text("jQuery");  
             $("#subtitle").html(`<b>Write Less -</b> <i> Do More</i>`);  
             $(".description").text("It is a JavaScript library for building UI");  
        })  
    </script>  
</head>  
<body>  
    <h1></h1>  
    <p id="subtitle"></p>  
    <p class="description"></p>  
</body>  
</html>  
  
jQuery DOM functions:  
  
    text()                        innerText  
    html()                        innerHTML  
    append()                    add as suffix  
    appendChild()            add as child element  
    prepend()                    add as prefix  
    before()                    add above  
    after()                        add below  
    css()                        defines styles  
    $.each()                    iterator  
    val()                        get or set value  
    attr()                        get or set attribute  
    prop()                        get or set property

### jQuery Events

- All JavaScript events are same , they are added using event listner.  
  
    <button> Insert </button>  
  
    $("button").click(function(event){  
        event.clientX;  
        event.clientY;  
        event.keyCode;  
        event.target.id;  
        event.target.name;  
        event.target.className;  
    })  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script src="../node\_modules/jquery/dist/jquery.js"></script>  
    <script>  
        $(function(){  
            $("button").click(()=>{  
                var gender = $("input[type=radio]");  
                var genderType = "";  
                for(var item of gender){  
                    if(item.checked){  
                        genderType = item.value;  
                    }  
                }  
                $("p").html(`  
                   User Name : ${$("#UserName").val()} <br>  
                   Age: ${$("#Age").val()} <br>  
                   Gender : ${genderType}  
                `);  
            })  
        })  
    </script>  
</head>  
<body>  
    <dl>  
        <dt>User Name</dt>  
        <dd><input type="text" id="UserName"></dd>  
        <dt>Age</dt>  
        <dd><input type="number" id="Age"></dd>  
        <dt>Gender</dt>  
        <dd>  
            <input type="radio" name="gender" value="Male"> Male  
            <input type="radio" name="gender" value="Female"> Female  
        </dd>  
    </dl>  
    <button>Submit</button>  
    <p></p>  
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script src="../node\_modules/jquery/dist/jquery.js"></script>  
    <script>  
        var categories = ["Electronics", "Footwear", "Fashion"];  
        $(function(){  
            categories.map((value)=>{  
                $(`<li>${value}</li>`).appendTo("ol");  
                $(`<option value=${value}>${value}</option>`).appendTo("select");  
            })  
        })  
    </script>  
</head>  
<body>  
    <ol>  
  
    </ol>  
    <select></select>  
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script src="../node\_modules/jquery/dist/jquery.js"></script>  
    <script>  
         
        $(function(){  
            fetch("../data/products.json")  
            .then(res=> res.json())  
            .then(products=>{  
                products.map((item)=>{  
                    $(`<tr>  
                        <td>${item.Name}</td>    
                        <td>${item.Price}</td>  
                        <td><img src=${item.Photo} width="100" height="100"></td>  
                     </tr>`).appendTo("tbody");  
                })  
            })  
        })  
    </script>  
</head>  
<body>  
   <table border="1" width="600">  
        <thead>  
            <tr>  
                <th>Name</th>  
                <th>Price</th>  
                <th>Preview</th>  
            </tr>  
        </thead>  
        <tbody>  
  
        </tbody>  
   </table>  
</body>  
</html>

jQuery Effects  
slide()  
slideToggle()  
fade()  
fadeToggle()  
show()  
hide()  
toggle() etc..  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script src="../node\_modules/jquery/dist/jquery.js"></script>  
    <script>  
         
        $(function(){  
            $("button").click(()=>{  
                $("img").fadeToggle(3000);  
            })  
        })  
    </script>  
</head>  
<body>  
    <div>  
        <button>Toggle</button>  
    </div>  
    <div>  
        <img src="../public/images/asp.jpg">  
    </div>  
</body>  
</html>

jQuery Ajax  
$.ajax()        : It is jquery ajax method.  
  
Fetch                                              JQuery Ajax  
---------------------------------------------------------------------------  
Data returned in Binary.                    Data returned will be JSON, HTML, Text..  
  
You have to convert to JSON                No conversion required.  
  
Slow                                            Fast  
  
It uses catch() to catch issues, which        It is good in error handling. It provides  
is not good in error handling.                life cycle methods.  
  
  
Syntax:  
    $.ajax({  
        method : "get | post | put | delete",  
        url : "path",  
        success: ( data ) => { },  
        error: (err) => { }  
    })  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script src="../node\_modules/jquery/dist/jquery.js"></script>  
    <script>  
         
        $(function(){  
            $.ajax({  
                method: "get",  
                url: "<http://fakestoreapi.com/product&quot>;,  
                success: (data)=>{  
                    data.map((item)=>{  
                        $(`<li>${item.title}</li>`).appendTo("ol");  
                    })  
                },  
                error: (jqXhr) => {  
                    console.log(jqXhr.statusText);  
                }  
            })  
        })  
    </script>  
</head>  
<body>  
    <ol>  
  
    </ol>  
</body>  
</html>  
  
AJAX Life Cycle Method Phases:  
1. ajaxStart()  
2. ajaxSuccess()  
3. ajaxComplete()  
4. ajaxStop()  
5. ajaxError()  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <script src="../node\_modules/jquery/dist/jquery.js"></script>  
    <script>  
         
        $(function(){  
             $("button").click(()=>{  
                $.ajax({  
                    method:"get",  
                    url:"../data/help.txt",  
                    success: (data)=>{  
                        $("p").html(`<pre>${data}</pre>`);  
                    }  
                })  
             })  
        })  
        .ajaxStart(()=>{  
            alert(`Your Request for Help document Initiated`);  
        })  
        .ajaxSuccess(()=>{  
            alert(`Your request processed successfully..`);  
        })  
        .ajaxComplete(()=>{  
            alert(`Your Request completed and response ready`);  
        })  
        .ajaxStop(()=>{  
            alert(`Your data sent as response and memory cleaned`);  
        })  
        .ajaxError(()=> {  
            alert(`Unable to Fetch Data - Contact your site Admin`);  
        })  
    </script>  
</head>  
<body>  
    <button>Get Data</button>  
    <p></p>  
</body>  
</html>  
  
  
Note: Add events to elements inside dynamic content by using "on()"  
  
        $(document).on("eventName", "elementRef", (event)=>{  
  
        })  
  
Ex: Shopping-Template  
  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</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 src="../node\_modules/jquery/dist/jquery.js"></script>  
    <script>  
        function LoadCategories(){  
            $.ajax({  
                method: "get",  
                url: "<http://fakestoreapi.com/products/categories&quot>;,  
                success:(data)=>{  
                    data.unshift("all");  
                    data.map((item)=>{  
                        $(`<option value=${item}>${item.toUpperCase()}</option>`).appendTo("#lstCategories");  
                    })  
                }  
            })  
        }  
  
        function LoadProducts(url){  
            $("main").html("");  
            $.ajax({  
                method:"get",  
                url:url,  
                success: (data)=>{  
                    data.map((item)=>{  
                        $(`  
                         <div class="card p-2 m-2" style="width:200px">  
                            <img src=${item.image} class="card-img-top" height="150">  
                            <div class="card-header" style="height:150px">  
                                <p>${item.title}</p>  
                            </div>  
                            <div class="card-body">  
                                Price: ${item.price} <br><br>  
                                Rating: ${item.rating.rate} [${item.rating.count}]  
                            </div>  
                            <div class="card-footer">  
                                <button name=${item.id} id="btnAdd" class="btn btn-danger w-100">  
                                  <span class="bi bi-cart4"></span>  
                                  Add to Cart  
                                </button>  
                            </div>  
                         </div>  
                        `).appendTo("main");  
                    })  
                }  
            })  
        }  
  
        $(function(){  
            LoadCategories();  
            LoadProducts("<http://fakestoreapi.com/products&quot>;);  
            $("#lstCategories").change((event)=>{  
                if(event.target.value=="men's"){  
  
                } else if(event.target.value=="women's"){  
  
                }  
                if(event.target.value=="all"){  
                    LoadProducts("<http://fakestoreapi.com/products&quot>;);  
                } else {  
                    LoadProducts(`<http://fakestoreapi.com/products/category/$>{event.target.value}`);  
                }  
            })  
  
            $(document).on("click","#btnAdd",(e)=>{  
                alert(`Add Clicked - ${e.target.name}`);  
            })  
        })  
    </script>  
</head>  
<body class="container-fluid">  
    <header class="mt-2 bg-dark text-white text-center fw-bold p-4">  
        <h1>Shopper</h1>  
    </header>  
    <section class="mt-3 row">  
        <nav class="col-2">  
            <label class="form-label fw-bold">Select Category</label>  
            <div>  
                <select id="lstCategories" class="form-select">  
  
                </select>  
            </div>  
        </nav>  
        <main class="col-10 d-flex flex-wrap overflow-auto" style="height:500px">  
  
        </main>  
    </section>  
</body>  
</html>

jQuery UI  
- It is similar to bootstrap.  
- It is a separate library for building UI.  
- It provides  
    a) Interactions  
    b) Widgets  
    c) Plugin's  
  
<https://jqueryui.com/>   => download stable library => extract file and copy into your project folder.  
  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <link rel="stylesheet" href="../node\_modules/jquery-ui/jquery-ui.css">  
    <script src="../node\_modules/jquery/dist/jquery.js"></script>  
    <script src="../node\_modules/jquery-ui/jquery-ui.js"></script>  
    <script>  
        $(function(){  
            $("img").resizable();  
        })  
    </script>  
</head>  
<body>  
    <img src="../public/images/asp.jpg" width="100" height="100">  
</body>  
</html>  
  
Ex:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE=edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    <link rel="stylesheet" href="../node\_modules/jquery-ui/jquery-ui.css">  
    <script src="../node\_modules/jquery/dist/jquery.js"></script>  
    <script src="../node\_modules/jquery-ui/jquery-ui.js"></script>  
    <script>  
        $(function(){  
            $("#faqs").accordion();  
            $("#dept").datepicker();  
        })  
    </script>  
</head>  
<body>  
    <div>  
        Departure : <input type="text" id="dept">  
    </div>  
    <div id="faqs">  
        <h2>What is Netflix?</h2>  
        <div>  
            <p>something about netflix</p>  
        </div>  
        <h2>How to use Netflix?</h2>  
        <div>  
            <p>something about usage..</p>  
        </div>  
    </div>  
</body>  
</html>