**JavaScript**

**Session-43**

* Javascript is light weight JIT [just-in-time] compiled and interpreted language.
* Interpreted is the process of translating line-by-line .
* Compiled is the process of translating entire program.

1. JIT
2. AOT

* JIT [Just-In-Time] compiled in browser.
* AOT [Ahead-of-Time] compile at application level.
* Javascript is a language used to build large scale applications at both client side and server side.
* Javascript is used
* Client side HTML
* Server side Node.js
* Database MongoDB
* Animations[2D,3D] Flash, 3DS Max etc..

A close up of a text

Description automatically generated

Evolution of JavaScript  
==================

* The first browser “Mosaic”
* The first Markup languages
* GML
* SGML
* Scripting language : ECMA Script
* 1990 Tim Berner Lee introduced - HTML

HTML

Scripting – ECMA

* 1994 Netscape communications – Browser

Netscape Communicator.

HTML

ECMA

* In 1995 Netscape communications appointed a scripting expert to design a script for Netscape Browser.

“Brendan Eich” - MDN [Mozila developer Network]

He developed a script and named as “Mocha” after that it was renamed as LiveScript.

* In 1996 Netscape given the responsibility of LiveScript to company called “Sun-MicroSystems” -Java i.s why named as LiveScript – JavaScript.

Designed By – Brendan Eich

Designed For – Netscape

Maintained By - Sun Micro System.

* In 1998 microsoft developed an OS called Window-98 – for internet.

Free Browser : Internet Explorer

Opera,safari …

* In 2000 Netscape stopped its services due to free browser came.

Stopped JS -> ECMA [Kept only but not maintaining]

IE JS => Own Extensions to JS

Safari JS

Browser Compatibility issues arises.

ECMA – MDN = Started maintaining javascript = ES5, ES6, ES7 , ES2021.

JS = ES.

**Session-44 :**

Issues with JavaScript .

1. It is not strongly typed.
2. It will not support all OOP features.
3. Hard to extend
4. Hard to maintain and test
5. Less secured
6. Lot of compatibility issues
7. Lot of DOM manipulations
8. Lot of coding
9. Heavy on application.
10. It is slow

FAQ: Is JavaScript OOP language ?

Ans : NO. It supports few features of OOP.

FAQ: Why javascript is called as scripting language?

Ans : Scripting happened on the top of existing technologies.

Scripting cants be individual.

It cant build an application, it can only support application.

FAQ: How javascript is integrated into web applications ?

Ans : JavaScript can be

1. Inline
2. Embedded
3. External file

Inline JavaScipt

---------------------

* Javascript functions are written within the HTML tag.
* Faster in execution.
* But cant reuse.

EX:

<!DOCTYPE html>

<html>

    <head>

        <title>InLine JavaScript</title>

    </head>

    <body>

        <h2>Your Ticket - Click Print Button</h2>

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

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

    </body>

</html>

Embedded Script

* In this technique the script is defined in **head** or **body** section by using <script> element.
* You can reuse across various elements.

Ex:

<!DOCTYPE html>

<html>

    <head>

        <title>InLine JavaScript</title>

    </head>

    <script type=”text/javascript”>

        function printPage(){

            window.print();

        }

    </script>

    <body>

        <h2>Your Ticket - Click Print Button</h2>

        <button onclick="printPage()">Print</button>

        <button onclick="printPage()">Print Page</button>

    </body>

</html>

FAQ: What is the MIME type for javascript?

Ans : “text/javascript” or language = “javascript”.

<script type=”text/javascript”>

</script>

<script language=”javascript”>

</script>

FAQ : What is JavaScript strict mode ?

Ans : JavaScript mode allows only good code , it restricts bad code[not according to standard].

FAQ: How to enable javaScript strict mode ?

Ans : By using

“use strict”;

EX:

<script>

   "use strict"; //on first line

    x=10; //not declared directly assigned value to variable//invalid

    document.write("x="+x);

</script>

FAQ: How to target JavaScript for legacy browsers ?

Ans : By enclosing JavaScript in HTML comments.

<!—

Your code

-->

EX:

<script>

   "use strict";

   <!--

    var x=10;

    x = 20;

    document.write("x="+x);

    -->

</script>

3. JavaScript from External file

- JavaScript code is written in separate javascript file with extension “.js”

- Link to HTML page by using <script></script>

print.js

<script src=”print.js”></script>

EX:

Print.js

"use strict";

function printPage(){

    window.print();

}

Home.html

<!DOCTYPE html>

<html>

    <head>

        <title>InLine JavaScript</title>

    </head>

    <script src="./scripts/print.js">

    </script>

    <body>

        <h2>Your Ticket - Click Print Button</h2>

        <button onclick="printPage()">Print</button>

    </body>

</html>

Note : Using external file increase the number of requests and page load time.

FAQ: How to verify the status of JavaScript in HTML page?

Ans: By using HTML element <noscript>

Syntax :

<body>

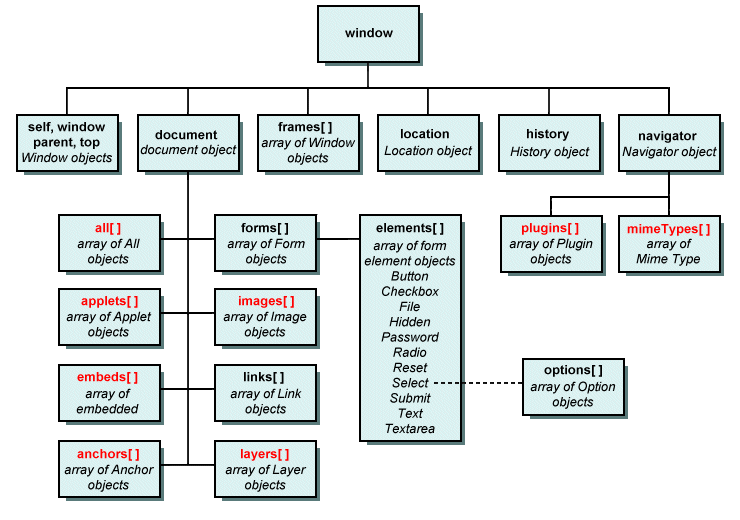
<noscript>Please Enable JavaScript</noscript>

</body>

**Session 45:**

How JavaScript refers HTML elements in Page?

1. By using DOM hierarchy



* You can refer HTML elements by using their hierarchy in DOM.
* You have to access with reference of index number and parent and child hierarchy.

Syntax:

Window.document.images[index].src



Good:

* Accessing with position is easy
* Accessing in sequential order is easy.
* Good for testing.

Bad :

* If you change the position of any element in page , then you have to update its index in code.

EX:

<!DOCTYPE html>

<html>

    <head>

        <title>Dom Hierarchy</title>

    </head>

    <script type="text/javascript">

        function bodyloads(){

            window.documnet.images[0].src = "../../assets/card1.jpg";

            window.document.forms[0].elements[1].value = "Register";

            document.write("hi ! how are you...");

        }

    </script>

    <body onload="bodyloads()">

        <div>

            <img width="100" height="100">

        </div>

        <div>

            <form>

                <h2>Register</h2>

                <input type="email">

                <input type="button">

            </form>

        </div>

        <div>

            <form>

                <h2>Login</h2>

                <input type="text">

                <input type="button">

            </form>

        </div>

    </body>

</html>

1. You can refer by using name

* Every element can have a reference name.
* You can access element by using the reference name.

Good:

* Easy to access.
* Faster in access .
* Even you change the position it will have the same values.

Bad

* Name can be common for multiple elements.
* If name is common then attributes are no applied.
* You cant refer child element

1. Refer by using ID

* Every element can be defined with ID.
* ID is unique for JavaScript but not for CSS.

Good:

* You can refer any HTML element directly.

Document.getElementById(“id);

* No more hierarchy issues.

Bad:

* Id can be used by CSS and JavaScript
* CSS id can be common for multiple elements.

EX:

<!DOCTYPE html>

<html>

    <head>

        <title>Dom Hierarchy</title>

        <script type="text/javascript">

            function bodyload(){

                document.getElementById("pics").src="../../assets/card1.jpg"

            }

        </script>

    </head>

    <body onload="bodyload()">

        <div>

            <img id="pics" width="100" name="pic" border="1">

        </div>

        <div>

            <form name="frmRegister">

                <h2>Register</h2>

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

                <input type="email">

            </form>

        </div>

    </body>

</html>

1. Refer by using CSS selectors.

* CSS provides selectors like

1. Id
2. Type
3. Class
4. Pseudo selector
5. Decendent
6. Attribute selectors etc…

* JavaScript uses

Document.querySelector();

Good:

* Various techniques to refer HTML elements to
* You can access by name , id , class etc..
* You can also access by using attribute , pseudo classes.

EX:

<!DOCTYPE html>

<html>

    <head>

        <title>Dom Hierarchy</title>

        <script type="text/javascript">

            function bodyload(){

                document.querySelector("img").src="../../assets/card1.jpg"

                document.querySelector("#btn").value = "Register";

            }

        </script>

    </head>

    <body onload="bodyload()">

        <div>

            <img id="pics" width="100" name="pic" border="1">

        </div>

        <div>

            <form name="frmRegister">

                <h2>Register</h2>

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

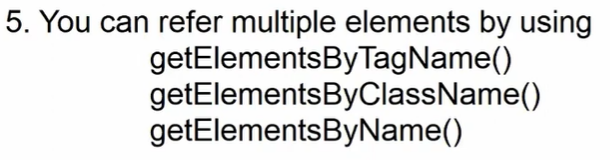
                <input type="email">

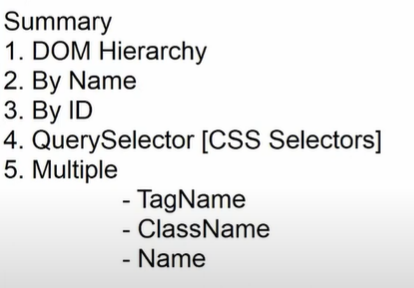
            </form>

        </div>

    </body>

</html>





**Session-46**

**Every programming language**

* Accepts input from user
* Process the input.
* Generate a response
* Render response as output

**Output Techniques:**

1. **Alert()**
2. **Confirm()**
3. **Document.write()**
4. **Console.log() , error() , debug() , warm() etc**
5. **innerHTML**
6. **outerHTML**
7. **innerText**

alert()

* it is used to display a message in message box that pops up in browser window.
* It will not allow to do any another task until or unless you confirm .
* It will not allow to cancel .

Syntax

alert(“your message”);

alert(“line1\nline2)”);

EX:

<!DOCTYPE html>

<html>

    <head>

        <title>OutPut</title>

        <script>

           function deleteClick(){

            alert("Delete Record\n Record will be deleted");

            }

        </script>

    </head>

    <body>

        <h2>Click Delete button to delete record</h2>

        <button onclick="deleteClick()" type="button">Delete</button>

    </body>

</html>

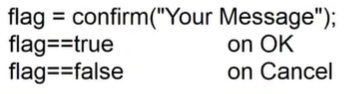
Confirm()

* It is similar to alert but allows to cancel.
* Cancel will not work directly , you have to define its functionality explicitly.
* confirm returns

True on OK

False on Cancel

* you have to capture the result of confirm and compare



EX:

<script>

           function deleteClick(){

                flag = confirm("Delete Record\n Record will be deleted");

                if(flag==true){

                    document.write(flag);

                    alert("Record deleted successfully..")

                }else{

                    alert("you Cancelled")

                }

            }

        </script>

Document.write():::

* it is used to display output on new screen
* you can use markup for output

Syntax

document.write(“your message”);

document.wirte(“<your markup>”);

EX: Custom confirm box :

<!DOCTYPE html>

<html>

    <head>

        <title>OutPut</title>

        <link rel="stylesheet" href="../../node\_modules/bootstrap-icons/font/bootstrap-icons.css">

        <link rel="stylesheet" href="../../node\_modules/bootstrap/dist/css/bootstrap.css">

        <script>

           function yesClick(){

            document.write("<b><i>Record deleted ....</i></b>");

           }

           function noClick(){

            document.write("You cancel..");

           }

        </script>

    </head>

    <body>

        <div class="container-fluid">

            <div class="mt-2">

                <button class="btn btn-danger" data-bs-target="#delete" data-bs-toggle="modal">Delete</button>

                <div class="modal fade" id="delete">

                    <div class="modal-dialog">

                        <div class="modal-content">

                            <div class="modal-header">

                                <h2>Delete record</h2>

                                <button class="btn-close" data-bs-dismiss="modal"></button>

                            </div>

                            <div class="modal-body">

                                <p>Are you sure want to delete?</p>

                            </div>

                            <div class="modal-footer">

                                <button onclick="yesClick()" class="btn btn-primary" data-bs-dismiss="modal">Yes</button>

                                <button onclick="noClick()" class="btn btn-danger" data-bs-dismiss="modal">No</button>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

        </div>

        <script src="../../node\_modules/jquery/dist/jquery.js"></script>

        <script src="../../node\_modules/bootstrap/dist/js/bootstrap.bundle.js"></script>

    </body>

</html>

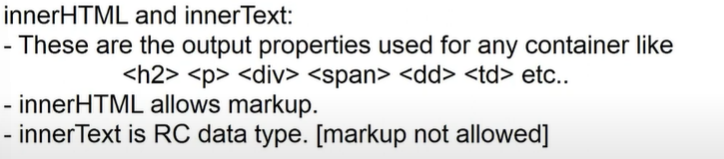
console methods ::

* console is a tool provided in every browser
* you can open developer tool and goto console.
* Console.log(),console.error(),console.warn() ,console.debug()..

Syntax:

Console.log(“message”);

Console.error(“message”);



Ex:

<!DOCTYPE html>

<html>

    <head>

        <title>OutPut</title>

        <link rel="stylesheet" href="../../node\_modules/bootstrap-icons/font/bootstrap-icons.css">

        <link rel="stylesheet" href="../../node\_modules/bootstrap/dist/css/bootstrap.css">

        <script>

           function yesClick(){

            document.getElementById("msg").innerText = "Record deleted..";

        }

           function noClick(){

            document.getElementById("msg").innerText= "You cancel";

           }

        </script>

    </head>

    <body>

        <div class="container-fluid">

            <div class="mt-2">

                <button class="btn btn-danger" data-bs-target="#delete" data-bs-toggle="modal">Delete</button>

                <div class="modal fade" id="delete">

                    <div class="modal-dialog">

                        <div class="modal-content">

                            <div class="modal-header">

                                <h2>Delete record</h2>

                                <button class="btn-close" data-bs-dismiss="modal"></button>

                            </div>

                            <div class="modal-body">

                                <p>Are you sure want to delete?</p>

                            </div>

                            <div class="modal-footer">

                                <button onclick="yesClick()" class="btn btn-primary" data-bs-dismiss="modal">Yes</button>

                                <button onclick="noClick()" class="btn btn-danger" data-bs-dismiss="modal">No</button>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

        </div>

        <div class="text-center mt-2">

            <h2 id="msg" style="color:green"></h2>

        </div>

        <script src="../../node\_modules/jquery/dist/jquery.js"></script>

        <script src="../../node\_modules/bootstrap/dist/js/bootstrap.bundle.js"></script>

    </body>

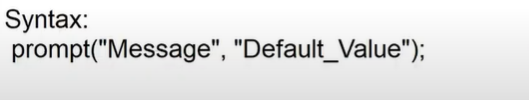
</html>

JavaScript Input Techniques ::

1. Prompt()
2. Form input elements

prompt():

* It is used to display input bot in browser.
* Input box will accept a string input.



EX:

<!DOCTYPE html>

<html>

    <head>

        <title>Input</title>

        <link rel="stylesheet" href="../../node\_modules/bootstrap/dist/css/bootstrap.css">

        <script>

            function createFolder(){

               name = prompt("Enter Name","New\_Name")

                document.write("Hello "+name);

            }

        </script>

    </head>

    <body>

        <div class="container-fluid">

            <button onclick="createFolder()" class="btn btn-link">Enter your name</button>

        </div>

    </body>

</html>

**Session- 47**

**Input using Form elements :**

* You can use form elements like

textbox , password , date , checkbox , radio , listbox , etc..

EX:

<!DOCTYPE html>

<html>

    <head>

        <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">

    </head>

    <body class="container-fluid">

        <div class="mt-2">

            <button class="btn btn-primary" data-bs-target="#register" data-bs-toggle="modal">Register Product</button>

        <div class="modal fade" id="register">

            <div class="modal-dialog">

                <div class="modal-content">

                    <div class="modal-content">

                        <h3>Product Registration</h3>

                        <button class="btn-close"></button>

                    </div>

                    <div class="modal-body">

                        <div class="mb-2">

                            <label class="form-label">Name</label>

                            <div>

                                <input type="text" class="form-control" id="productName">

                            </div>

                        </div>

                        <div class="mb-2">

                            <label class="form-label">price</label>

                            <div>

                                <input type="text" class="form-control" id="productPrice">

                            </div>

                        </div>

                        <div class="mb-2">

                            <label class="form-label">Shipped To</label>

                            <div>

                                <select class="form-select" id="productCity">

                                    <option>Delhi</option>

                                    <option>Hyd</option>

                                    <option>Chennai</option>

                                </select>

                            </div>

                        </div>

                        <div class="mb-2">

                            <label class="form-label">Stock</label>

                            <div class="form-switch">

                                <input type="checkbox" class="form-check-input" class="form-control" id="productStock">

                            </div>

                        </div>

                    </div>

                   <div class="modal-footer">

                        <button class="btn btn-primary" data-bs-dismiss="modal">Register</button>

                        <button class="btn btn-danger" data-bs-dismiss="modal">Cancel</button>

                    </div>

                </div>

            </div>

        </div>

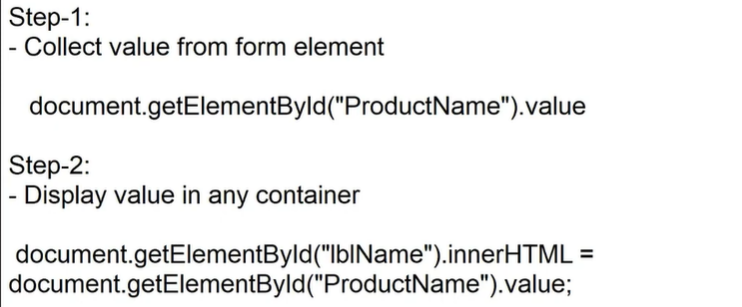
    </div>

        <script src="../../node\_modules/jquery/dist/jquery.js"></script>

        <script src="../../node\_modules/bootstrap/dist/js/bootstrap.bundle.js"></script>

    </body>

</html>

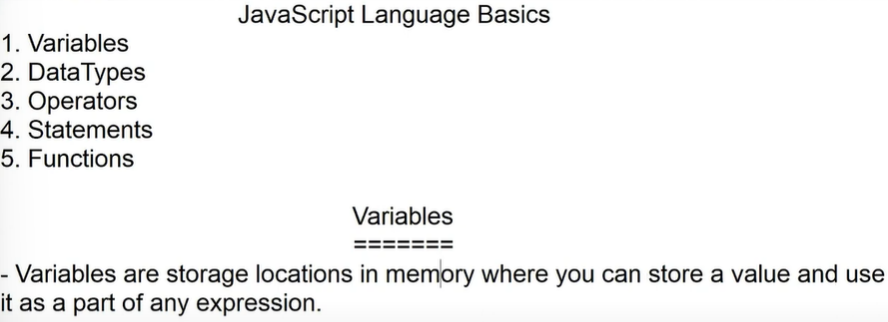




A close-up of a computer screen

Description automatically generated

**Session-48**

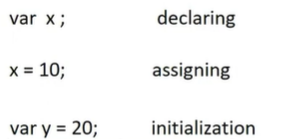


* Javascript can directly use variables and store value if it is not in strict mode.
* If javascript is in strict mode , it is mandatory to declare variables.
* Variables are declared by using 3 keywords.

1. var
2. let
3. const

* variable configuration comprises of 3 phases

1. declaring
2. initialization
3. assigning



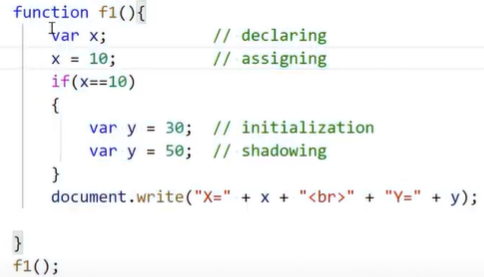
FAQ: If you don’t initialize or assign any value into variable , what is contains ?

Ans: “undefined”.

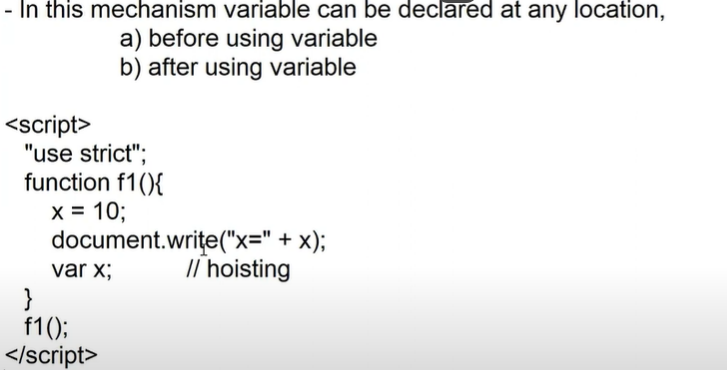
FAQ: What is difference between var , let and const ?

Ans: var :

* it used to define function scope variables.
* A function scope variable can be declared in any block and can be access from any another block in same function.
* Var allows declaring , assigning and initialization.
* Var allows shadowing .
* Shadowing is the process of re-declaring or initializing the same name variable in the scope.

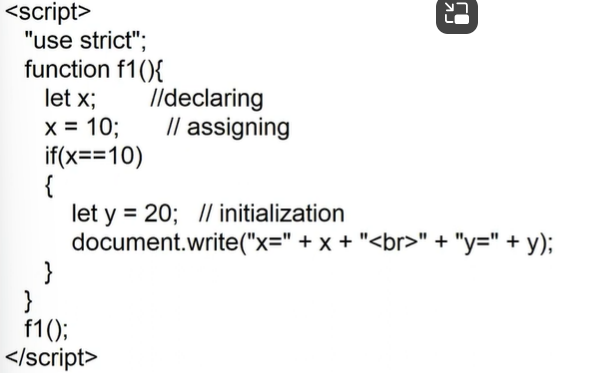


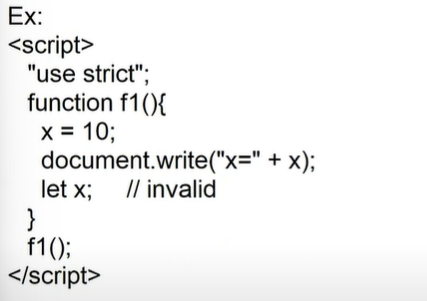
* Var allows hoisting.
* Hoisting is a technique used to indicate the location of carriable to compiler.



let :

* It is used to define block scope variable.
* A block scope variable can be accessed within the block or within its inner block .
* Inner block variable is not accessible to outer block.
* Allows declaring , assigning , initializing.



* Let will not allow shadowing.
* Let will not allow hoisting.
* 

const :

* It is also block scope .
* It allows only initialization.
* No declaring , no assigning.

const x ; //invalid

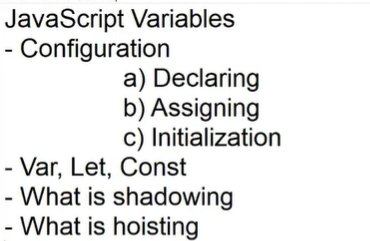
x = 10 ; //invalid

Const x = 10 ; // valid

x = 10 ; // invalid

* const will not allowing shadowing.
* const will not allowing hoisting.

**Session – 49**



Global Scope for Variables :

* variables are function or block scope.
* To define a global scope for variable , you have to declare at module level.

EX:

<script>

    //module scope

    var x = 10;

    let y = 20;

    const z = 30;

    function f1(){

        document.write("Function 1 -<br> x="+x + "<br> y="+y+"<br>z="+z);

    }

    function f2(){

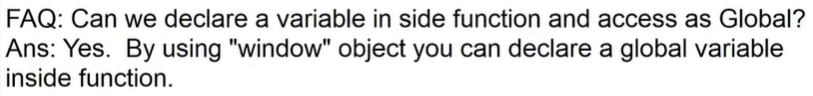
        document.write("Function 1 -<br> x="+x + "<br> y="+y+"<br>z="+z);

    }

    f1();

    f2();

</script>



EX:

function f1(){

        window.username = "John";

    }

function f2(){

        document.write('Hello ! '+username);

    }

Variable Naming Conventions :

* Variable name must start with alphabet or underscope.

var\_name;

var name;

var 2022 ; // invalid

var 2022year; //invalid

* Variable name can be alpha numeric.

var year2022 ; //valid

var year\_2022; //valid

var year.2022 ; // invalid

* Don’t use any special chars other than \_
* Don’t use blank spaces
* Always use camel case for naming

Var username;

* Variable name cant be more that 255 chars long;
* Variable name must speak what it is.

FAQ: Why and When to use “underscore” ?

Ans: it is used any compiling process.

It used for separation of name.

It used for implementation of name.

Var year\_2022;

Var \_productName; // marked for implementation means It can be modified.

JavaScript Data Types ::

* Data types defines the data structure.
* Data structures defines the type , range and behaviour.
* JavaScript is not strongly typed.
* You can store any type of value in a variable.
* It will not restrict implicitly.

var price = 45000; //number

price = “TV” //string

* The javascript data type are classified into 2 types

1. Primitive type
2. Non primitive types

Primitive Types :

* They are immutable types
* They cant change the structures.
* They have fixed range for values.
* They are stored in memory stack. [LIFO]
* JavaScript primitive types are

1. number
2. string
3. boolean
4. null
5. undefined

number :

* signed integer -8
* unsigned integer 8
* floating 34.32
* double 340.212, 44.232
* decimal 4256.1234
* exponent 2e3 = 2\*10[3]
* binary 0b1010 == 10
* hexa 0-f
* octa 0o743

Note: Every HTML element value is string , you have to explicitly convert into number by using

1. parseInt()
2. parseFloat()

var age = “22”;

document.write(age+1) ; //221

document.write(parseInt(age) +1); 23

EX:

<!DOCTYPE html>

<html>

    <head>

        <title>Number</title>

        <script>

            function calculate(){

                var age = document.getElementById("age").value;

                document.getElementById("msg").innerHTML = (parseInt(age)+1);

            }

        </script>

    </head>

    <body>

        Age: <input type="number" id="age" min="15" max="30">

        <button onclick="calculate()">Calculate</button>

        <span id="msg"></span>

    </body>

</html>

FAQ: How to check the number type input ?

ANS : By using “isNaN()”

It returns Boolean true if value is not a number type.

EX:

<!DOCTYPE html>

<html>

    <head>

        <title>Number</title>

        <script>

            function calculate(){

                var age = document.getElementById("age").value;

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

                if(isNaN(age)){

                    msg.innerHTML ="Please Enter a Number";

                }else{

                    msg.innerHtml = "Your Age:"+age;

                }

            }

        </script>

    </head>

    <body>

        Age: <input type="text" id="age" min="15" max="30">

        <button onclick="calculate()">Calculate</button>

        <span id="msg"></span>

    </body>

</html>

EX-1

Var x = “10”;

Var y =20;

Var z = x+y; //1020

Var z = parseInt(x) +y ; //30

EX-2

Var x = “AB”;

Var y =20;

Var z = x+y; //AB20

Var z = parseInt(x) +y ; //NaN

EX-3

Var x = “20AB”;

Var y =20;

Var z = x+y; //AB20

Var z = parseInt(x) +y ; //40

Note: parsing can be done from string to number if value starts with number . You can parse even it is having a string .

**Session-50**

String type

* string is a literal with groups of chars enclosed in

1. double quotes “ ”
2. single quotes ‘ ’
3. back tick ` `

* literal can contain group of characters like alphabet , number and special chars.
* Double quote and single quote can be used for inner and outer string.

“<a href = ’home.html’>Home</a>”

‘<a href = “home.html”>Home</a>’

* Any expression need to concat with string by using “+”.

“Age = ”+( 10 + 20 ) + “ ”

* ES5 introduced Back Tick, which can embed any expression in a string.
* Embedded expression is defined by using “${}”.
* It is allowed only in Back Tick.
* Some characters in string will escape printing.
* To print the non-printable chars you have to use “\”

JavaScript String Formatting Methods

* These methods are used to format a string dynamically.

bold()

italics()

fontsize(color);

sup()

sub()

touppercase()

tolowercase()

<!DOCTYPE html>

<html>

    <head>

        <title>String Formatting</title>

        <script>

            function verifyUser(){

                var username = document.getElementById("UserName").value;

                var usererror = document.getElementById("UserError");

                if(username == ""){

                    usererror.innerHTML = "User Name Required".fontcolor("red");

                }

            }

            function verifyName(){

                var username = document.getElementById("UserName").value;

                var usererror = document.getElementById("UserError");

                if(username!=""){

                    usererror.innerHTML = "User Verified".fontcolor("green");

                }

            }

        </script>

    </head>

    <body>

        <dl>

            <dt>User Name</dt>

            <dd><input onblur="verifyName()" type="text" id="UserName"></dd>

            <dd id="UserError"></dd>

        </dl>

        <button onclick="verifyUser()">Register</button>

    </body>

</html>

EX:

<!DOCTYPE html>

<html>

    <head>

        <title>String Formatting</title>

        <script>

            function changeCase(){

                var username = document.getElementById("UserName").value;

                document.getElementById("UserName").value = username.toUpperCase();

            }

        </script>

    </head>

    <body>

        <dl>

            <dt>User Name</dt>

            <dd><input  type="text" onkeyup="changeCase()" placeholder="Only Capital Leters" id="UserName"></dd>

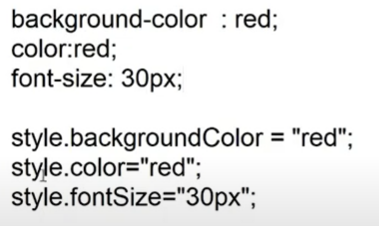
        </dl>

    </body>

</html>

Formatting a string By using styles

* You can use “style” property dynamically.
* Style attributes are defined in camel case.



EX:

<!DOCTYPE html>

<html>

    <head>

        <title>String Formatting</title>

        <script>

            function verifyUser(){

                var username = document.getElementById("UserName").value;

                var usererror = document.getElementById("UserError");

                if(username == ""){

                    usererror.innerHTML = "User Name Required";

                    usererror.style.color = "red";

                    usererror.style.fontStyle = "italic";

                }

            }

            function verifyName(){

                var username = document.getElementById("UserName").value;

                var usererror = document.getElementById("UserError");

                if(username!=""){

                    usererror.innerHTML = "User Verified";

                    usererror.style.color = "green";

                    usererror.style.fontWeight = "bold";

                }

            }

        </script>

    </head>

    <body>

        <dl>

            <dt>User Name</dt>

            <dd><input onblur="verifyName()" type="text" id="UserName"></dd>

            <dd id="UserError"></dd>

        </dl>

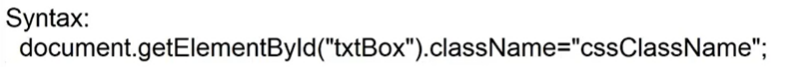
        <button onclick="verifyUser()">Register</button>

    </body>

</html>

Format String by using CSS Classes

* You can apply any CSS class instead of style by using “className” property.



EX:

<!DOCTYPE html>

<html>

    <head>

        <title>String Formatting</title>

        <link rel="stylesheet" href="../../node\_modules/bootstrap/dist/css/bootstrap.css">

        <style>

            .invalid-style{

                border:1px solid red;

                box-shadow: 2px 2px 2px red;

            }

            .valid-style{

                border:1px solid green;

                box-shadow: 2px 2px 2px green;

            }

        </style>

       <script>

            function verifyUser(){

                var username = document.getElementById("UserName").value;

                var usererror = document.getElementById("UserError");

                if(username == ""){

                    usererror.innerHTML = "User Name Required";

                    usererror.className="text-danger";

                    document.getElementById("UserName").className = "invalid-style";

                }

            }

            function verifyName(){

                var username = document.getElementById("UserName").value;

                var usererror = document.getElementById("UserError");

                if(username!=""){

                    usererror.innerHTML = "User Verified";

                    usererror.className="text-success fw-bold";

                    document.getElementById("UserName").className = "valid-style";

                }

            }

        </script>

    </head>

    <body class="container-fluid">

        <dl>

            <dt class="fw-bold">User Name</dt>

            <dd><input onblur="verifyName()" type="text" id="UserName"></dd>

            <dd id="UserError"></dd>

        </dl>

        <button onclick="verifyUser()">Register</button>

    </body>

</html>

EX:

<!DOCTYPE html>

<html>

    <head>

        <title>Register</title>

        <style>

            form{

                border: 2px solid gray;

                padding: 20px;

            }

            .container{

                display: flex;

                justify-content: center;

                align-items: center;

                height: 400px;

            }

        </style>

        <link rel="stylesheet" href="../../node\_modules/bootstrap/dist/css/bootstrap.css">

        <script>

            function changeMode(){

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

                if(theme.checked){

                    document.querySelector("form").className = "bg-dark p-3 text-white";

                    document.querySelector("button").className= "btn btn-primary"

                }else{

                    document.querySelector("form").className = "bg-light p-3 text-dark";

                    document.querySelector("button").className= "btn btn-light"

                }

            }

        </script>

    </head>

    <body class="container-fluid">

        <div class="container">

            <form>

                <div class="form-switch">

                    <input id="theme" onchange="changeMode()" class="form-check-input" type="checkbox"> Dark Mode

                </div>

                <h2>Register</h2>

                <dl>

                    <dt>User Name</dt>

                    <dd><input type="text"></dd>

                    <dt>Password</dt>

                    <dd><input type="password"></dd>

                    <dt>Email</dt>

                    <dd><input type="email"></dd>

                </dl>

                <button>Register</button>

            </form>

        </div>

    </body>

</html>

**Session-51**

String Formatting

* String methods
* Styles
* Class

String Manipulations

1. charAt : it returns the character at specified index.

Syntax:

string.charAt(0);

1. charCodeAt() : it returns the character code instead of char.

A – Z => 65 – 90

Syntax:

String.charCodeAt();

EX:

<!DOCTYPE html>

<html>

    <head>

        <title>String</title>

        <script>

            function verifyUserName(){

                var username = document.getElementById("UserName").value;

                var userError = document.getElementById("UserError");

                var firstCharCode = username.charCodeAt(0);

                if(firstCharCode >=65 && firstCharCode <=90){

                    userError.innerHTML = "";

                }else{

                    userError.innerHTML = "Name must start with Uppercase Letter".fontcolor("red");

                }

            }

        </script>

    </head>

    <body>

        UserName :

        <input type="text" onblur="verifyUserName()"  id="UserName">

        <div id="UserError"></div>

    </body>

</html>

1. slice() : it can extract the chars between specified index

it uses start and end index.

End index must be a number after start index.[end index > strat index]

1. substr() : it can extract specified number of chars from given index , it includes spaces also.

Syntax :

substr(startindex , how many chars)

substr(0,7)

substr(7,3)

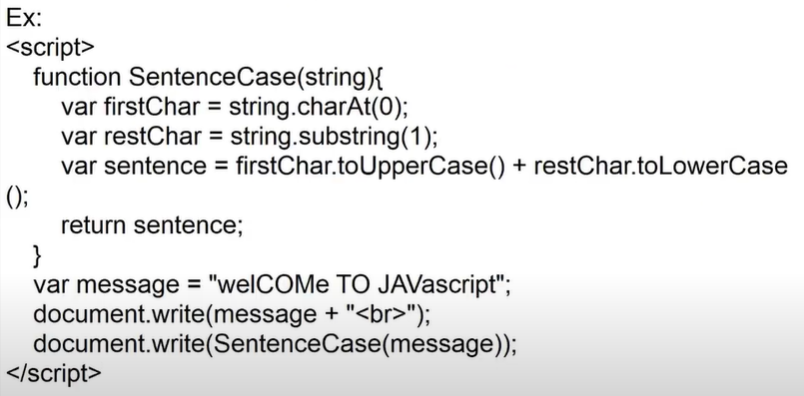
1. substring() : it can read the chars from specified index to either directions [start or end]

Syntax :

substring(startindex, noOfChars)

substring(7) 7 to end

substring(7,0) 7 to start



1. startWith() : it return true if string is starting with specified char(s).
2. endsWith() : it returns true if string is ending with specified char(s).
3. match() : it is used to verify a string against a pattern . In javascript pattern is a regular expression enclosed in “/ /”.
4. length : it returns the total count of chars in string.

**Session-52**

1. indexOf() : it returns the index number of specified char.

If char not found then it returns -1.

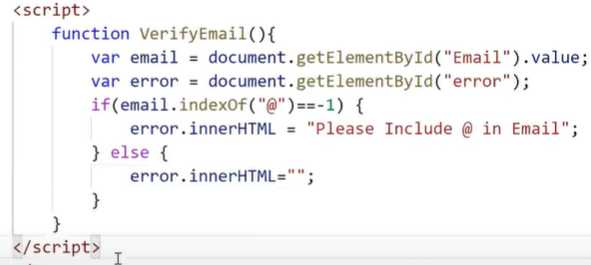
1. lastIndexOf : It returns the last occurrence index number.

Welcome

indexOf(“e”) 1

lastIndexOf(“e”) 6

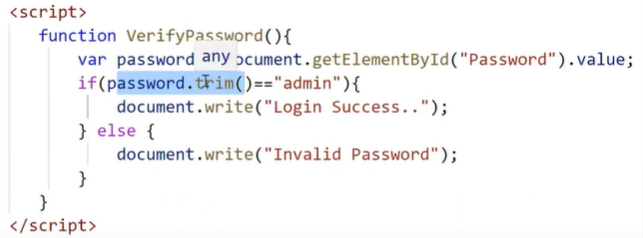
EX:



1. trim() : it is used to remove the leading spaces in string.

[space before and after string]

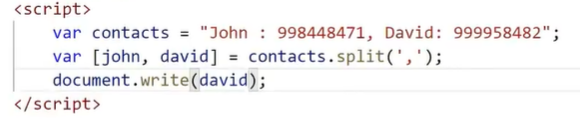
EX:

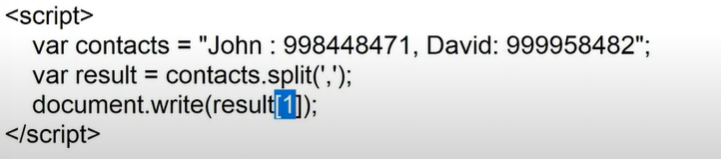


1. split() : it splits the string into multiple strings.

It can split by referring to any delimeter [separator]

EX:

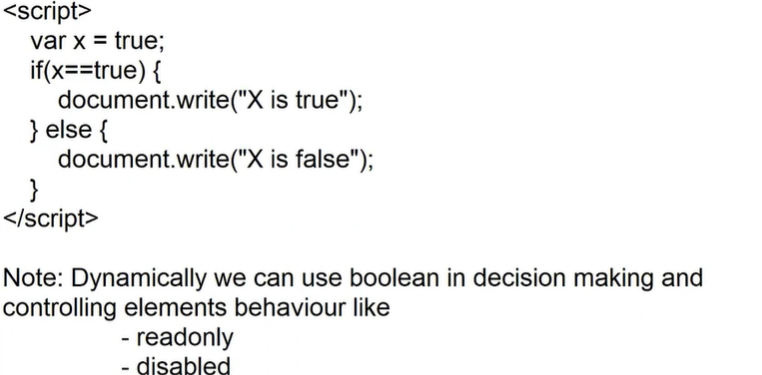




Boolean Type

* Boolean types are used in decision making.
* Boolean type can handle only in 2 values : true and false
* JavaScript Boolean true can refer to 1 and false to 0.

EX:



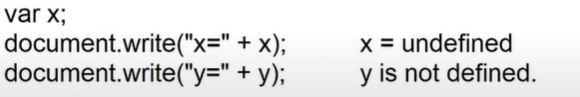
- checked

Undefined

* Undefined indicates that value is not defined into a variable or reference .
* You can use “undefined” keyword to verify value defined or not.

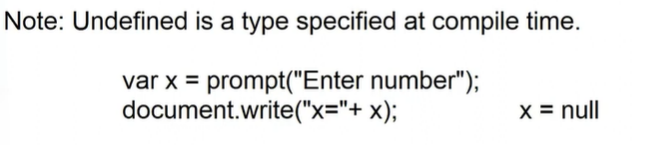
FAQ: What is undefined and not-defined ?

Ans : undefined is type , not-defined in reference error.



EX:



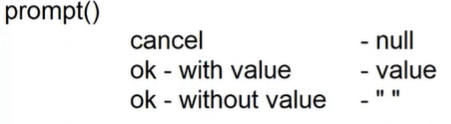


Null Type

* Null defines that values is not supplied during run time.
* Null is no value.

Var x = “”; blank space - &nbsp;

Var x = null; -nothing



EX:

A computer code with text

Description automatically generated with medium confidence

A white background with black text

Description automatically generated

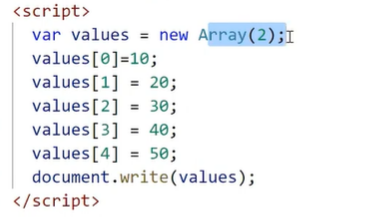
Non-Primitive Types

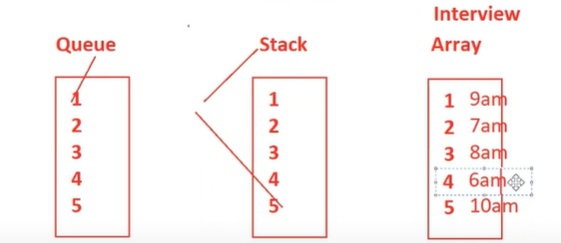
* These are mutable types.
* Their structure can change dynamically according to state and situation.
* They don’t have any fixed ranged for value
* Value range will change according to memory available.
* JavaScript non-primitive types are

1. Array
2. Object
3. Map

Array Type

* Arrays are used in computer programming to reduce overhead and complexity.
* Arrays can reduce overhead by storing values in sequential order.
* Arrays can reduce complexity by storing multiple values under one name.
* Arrays can store various types if values.
* Arrays size can change dynamically.





* We can access array elements randomly.

**Session - 53**

Declare Array :

* It is same like declaring any another variable .

var arrayName;

* Array must be initialized or assigned with memory .

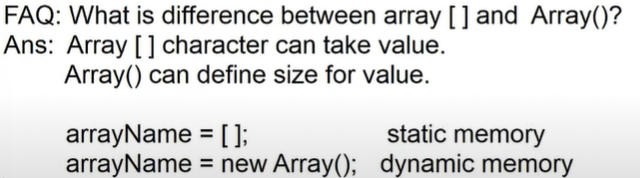
var arrayName = [] ;

var arrayName = new Array();

var arrayName;

arrayName = [];

arrayName = new Array();



[] continuous memory

Array() discreet/discontinuous memory

EX:

var categories = [“elecyronics”,”footwear”]

var categories = new Array([“elecyronics”,”footwear”);

Storing values into Array :

* Array comprises of index and property.
* Data is stored in the reference of memory index
* Data is accessed by using property reference
* You can use property for storing and accessing dat.
* Property name will be the same as index.
* Property type is string.

Syntax :

Var categories = [];

Categories[0] = “electronics”;

Categories[“1”] = 10;

Categories[“2”] = true;

EX:

<script>

    var categroies = [];

    categroies["0"] = "electronics" ;

    categroies[1] = 10;

    categroies["2"] = true

    for(var property in categroies){

        document.write(`[${property}] - ${typeof property} : ${categroies[property]}(${typeof categroies[property]})<br>`);

    }

</script>

FAQ: What types of value can be stored in array ?

Ans: you can store any type of value .

* primitive
* non primitive
* function

EX:

<script>

   var values = [10,"John",true,["hyd","delhi"],function(){document.write("function in array")}];

    document.write(values[3][1]+ "<br>")

    values[4]();

</script>

Note : if you are defining a function then it must be anonymous type i.e without name. you can access the function using property.

FAQ: What is array destruction ?

Ans: it is technique used to extract the values from array and store in various references.

EX:

<script>

   var values = [10,"John",true,["hyd","delhi"],function(){document.write("function in array")}];

   var [id , name , stock , cities, print] = values;

   print();

   document.write(cities)

</script>

Read values from array :

1. you can by using property name

EX:

categories[0];

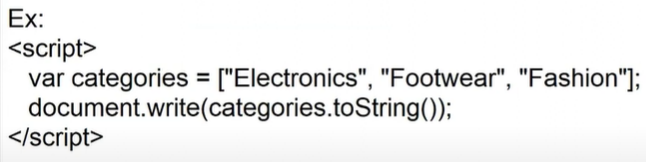
<script>

    var categories = ["electronics" , "footwear"];

    document.write(categories["1"]);

</script>

1. you can use “toString()” , which returns all elements separated with delimeter “,”.



1. you can use “join()” , which used custom delimeter.

EX:

<script>

   var categories = ["electronics" , "footwear"];

   document.write(categories.join("/"));

</script>

1. you can read by using map() , it is an iterator that reads every element from array and presents in the UI.

EX:

<script>

   var categories = ["electronics" , "footwear","fashion"];

   categories.map(function(category){

    document.write(`<h2>${category}</h2>`)

   })

</script>

1. you can read by using for..in statement , which returns all properties

Syntax :

for(var property in collection){

}

EX:

<script>

   var categories = ["electronics" , "footwear","fashion"];

   for(var property in categories){

    document.write(property + '<br>')

   }

</script>

1. you can read all values by using “for..of”

syntax:

for(var value of categories){

}

EX

<script>

   var categories = ["electronics" , "footwear","fashion"];

   for(var value of categories){

    document.write(value + '<br>')

   }

</script>

1. you can read both properties and values by using for..in

EX:

<script>

   var categories = ["electronics" , "footwear","fashion"];

   for(var property in categories){

    document.write(`[${property}] : ${categories[property]}`+"<br>")

   }

</script>

1. you can use “find()” method , which can search and return only the value the matches your conditions .it returns single value.

EX:

<script>

   var sales = [45000,66000,62000,24000,56000];

   var result = sales.find(function(value){

    return value<=55000;

   })

   document.write(result);

</script>

1. you can use “filter()” method , which returns all values that match the given condition.

EX:

<script>

   var sales = [45000,66000,62000,24000,56000];

   var result = sales.filter(function(value){

    return value>=55000;

   })

   document.write(result);

</script>

1. you can also use “for” loop for reading values

A math equation with black text

Description automatically generated

**Session-54**

FAQ: How to create a new HTML element dynamically ?

Ans: By using the method

document.createElement(“name”);

document.createElement(“img”);

document.createElement(“p”);

FAQ: How to configure element ?

var img = document.createElement(“img”);

img.width = “200”;

img.height = “100”;

img.src = “../public/images/shoe.jpg”;

FAQ: How to add new element into page ?

Ans: By using methods “appendChild()”

<div></div>

document.querySelector(“div”).appendChild(img);

Presenting Array Content Dynamically

* Create Element
* Set Properties
* Append to Parent

EX:

<!DOCTYPE html>

<html>

   <head>

      <title>Array</title>

      <script>

         var categories = ["All","Electronics","Footwear","Fashion","Men's Clothing"];

         function bodyLoad(){

            for(var item of categories){

               var li = document.createElement("li");

               li.innerHTML = item ;

               document.querySelector("ol").appendChild(li);

               var option = document.createElement("option");

               option.text = item;

               option.value = item;

               document.querySelector("select").appendChild(option);

               var tr = document.createElement("tr");

               var td = document.createElement("td");

               td.innerHTML = item;

               tr.appendChild(td);

               document.querySelector("tbody").appendChild(tr);

               var checkLi = document.createElement("li");

               checkLi.innerHTML = `<input type="checkbox"> ${item}`;

               document.querySelector("ul").appendChild(checkLi);

               var div = document.createElement("div");

               div.innerHTML = `<button style="margin-top:20px">${item}</button>`;

               document.querySelector("nav").appendChild(div);

            }

         }

      </script>

      <style>

         .container {

            display: flex;

         }

         .box {

            margin-right: 20px;

         }

         ul{

            list-style: none;

            height: 50px;

            border: 2px solid black;

            padding: 10px;

            overflow: auto;

         }

      </style>

   </head>

   <body onload="bodyLoad()">

      <div class="container">

         <div class="box">

            <h3>Categories List</h3>

            <ol>

            </ol>

         </div>

         <div class="box">

            <h3>Select Category</h3>

            <select>

            </select>

         </div>

         <div class="box">

            <h3>Categories Menu</h3>

            <table border="1" width="200">

               <thead>

                  <tr>

                    <th> Select Category</th>

                  </tr>

               </thead>

               <tbody>

               </tbody>

            </table>

         </div>

         <div class="box">

            <h3>Categories checklist</h3>

            <ul>

            </ul>

         </div>

         <div class="box">

            <h3>Nav Buttons</h3>

            <nav>

            </nav>

         </div>

      </div>

   </body>

</html>

Array Manipulations

1. Adding Elements into Array

Push() : it will add new items as last item.

Unshift() : it will add new items as first item.

Splice() : it will add new items at specific position.

Syntax

arrayName.push(“item1”,..);

arrayName.unshit(“item1”,”item2”)

arrayName.splice(startIndex,deleteCount,”item1”,”item2”)

1. Removing Elements from Array

pop() : removes and returns last item

shift() : removes and returns first item

splice() : removes specific item

EX:

<!DOCTYPE html>

<html>

   <head>

      <title>Array</title>

      <script>

         var categories = ["All","Electronics","Footwear","Fashion","Men's Clothing"];

         function loadCategory(){

            document.querySelector("ol").innerHTML = "";

            document.querySelector("select").innerHTML = "";

            document.querySelector("tbody").innerHTML = "";

            document.querySelector("ul").innerHTML = "";

            document.querySelector("nav").innerHTML = "";

            for(var item of categories){

               var li = document.createElement("li");

               li.innerHTML = item ;

               document.querySelector("ol").appendChild(li);

               var option = document.createElement("option");

               option.text = item;

               option.value = item;

               document.querySelector("select").appendChild(option);

               var tr = document.createElement("tr");

               var td = document.createElement("td");

               td.innerHTML = item;

               tr.appendChild(td);

               document.querySelector("tbody").appendChild(tr);

               var checkLi = document.createElement("li");

               checkLi.innerHTML = `<input type="checkbox"> ${item}`;

               document.querySelector("ul").appendChild(checkLi);

               var div = document.createElement("div");

               div.innerHTML = `<button style="margin-top:20px">${item}</button>`;

               document.querySelector("nav").appendChild(div);

            }

         }

         function bodyLoad(){

            loadCategory();

         }

         function addCategory(){

               var categoryname = document.getElementById("category").value;

               categories.push(categoryname);

               alert(`category added ${categoryname}`)

               loadCategory();

               document.getElementById("category").value = "";

            }

         function remove(){

            var categoryname = document.querySelector("select").value;

            var index = categories.indexOf(categoryname);

            var flag = confirm(`Are you sure , want to delete : ${categoryname}`)

            if(flag==true){

               categories.splice(index,1);

               loadCategory();

            }

         }

      </script>

      <style>

         .container {

            display: flex;

         }

         .box {

            margin-right: 20px;

         }

         ul{

            list-style: none;

            height: 50px;

            border: 2px solid black;

            padding: 10px;

            overflow: auto;

         }

      </style>

   </head>

   <body onload="bodyLoad()">

      <div style="margin-bottom: 20px;">

         New Category :

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

         <button onclick="addCategory()">Add</button>

      </div>

      <div class="container">

         <div class="box">

            <h3>Categories List</h3>

            <ol>

            </ol>

         </div>

         <div class="box">

            <h3>Select Category</h3>

            <select>

            </select>

            <br><br>

            <button onclick="remove()">Remove</button>

         </div>

         <div class="box">

            <h3>Categories Menu</h3>

            <table border="1" width="200">

               <thead>

                  <tr>

                    <th> Select Category</th>

                  </tr>

               </thead>

               <tbody>

               </tbody>

            </table>

         </div>

         <div class="box">

            <h3>Categories checklist</h3>

            <ul>

            </ul>

         </div>

         <div class="box">

            <h3>Nav Buttons</h3>

            <nav>

            </nav>

         </div>

      </div>

   </body>

</html>

**Session-55**

Sorting Array:

Sort() it arranges in ascending order

Reverse() it arranges in reverse order [bottom to top]

EX:

<script>

    var cities = ["Delhi","Hyd","Banglore","Chennai","Mumbai"]

    cities.sort();

    for(var city of cities){

        document.write(city + "<br>");

    }

</script>

Searching for element :

indexOf()

lastIndexOf()

find()

filter()

JavaScript Object Type

* In computer programming object is stored similar type of data and logic.
* It represents a pseudo class.
* Object was introduced in early 1990s by alen kay to keep all related data and logic together.

Syntax:

var object{

}

* Data is stored in properties and functionality in function.

Syntax:

var object = {

property : value ,

property : value,

action : function() {}

}

* Object comprises of “keys and values”
* Keys are string type and values are any type .

Syntax:

var object = {

“key” : value

}

* You can access outside object by using object name.

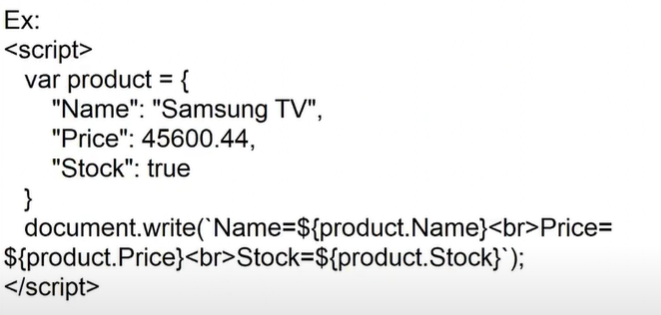
object.key

* You can access inside object by using “this”.

this.key

* If object comprises of only data then its refferes as “JSON”.

JavaScript Object Notation



EX:

Json file

{

    "name":"Iphone",

    "price": 55000.44,

    "photo":"../../assets/iphone.webp",

    "stock":true

}

Html file

<!DOCTYPE html>

<html>

    <head>

        <title>Object Demo</title>

        <link rel="stylesheet" href="../../node\_modules/bootstrap/dist/css/bootstrap.css">

        <script>

            var products = {};

            function getClick(){

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

                fetch("../../data/product.json")

                .then(function(response){

                    return response.json();

                })

                .then(function(product){

                    document.getElementById("lblName").innerHTML = product.name;

                    document.getElementById("lblPrice").innerHTML = product.price;

                    document.getElementById("productImg").src = product.photo;

                    document.getElementById("lblStock").innerHTML = (product.stock==true)?"Available":"Out of Stock";

                })

            }

        </script>

    </head>

    <body class="container-fluid">

        <div>

            <button onclick="getClick()" class="btn btn-primary">Get Products</button>

        </div>

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

            <h3>Product Details </h3>

            <dl>

                <dt>Name</dt>

                <dd id="lblName"></dd>

                <dt>Price</dt>

                <dd id="lblPrice"></dd>

                <dt>Preview</dt>

                <dd>

                    <img id="productImg" width="100" height="100">

                </dd>

                <dt>Stock</dt>

                <dd id="lblStock"></dd>

            </dl>

        </div>

    </body>

</html>

**Session-56**

EX: Shopping Cart

<!DOCTYPE html>

<html>

    <head>

        <title>Shopping | 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">

        <script>

            function loadCategory(){

                fetch("http://fakestoreapi.com/products/categories")

                .then(response=>response.json())

                .then(data=>{

                    data.unshift("ALL");

                    for(var item of data){

                        var option = document.createElement("option");

                        option.text=item;

                        option.value=item;

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

                    }

                })

            }

            function loadProducts(url){

                document.getElementById("catalog").innerHTML = "";

                fetch(url)

                .then(response=>response.json())

                .then(data=>{

                    for(var item of data){

                        var div= document.createElement("div");

                        div.className = "card m-2 p-2";

                        div.style.width = "200px";

                        div.innerHTML= `

                            <img src=${item.image} height="150" class="card-img-top">

                            <div class="card-header" style="height:170px">

                                <p>${item.title}</p>

                            </div>

                            <div class="card-body">

                                <h3>${item.price}</h3>

                            </div>

                            <div class="card-footer">

                               <button onclick="addToCartClick(${item.id})">

                                    <span class="bi bi-cart4">Add to Cart</span>

                                </button>

                            </div>

                        `;

                        document.getElementById("catalog").appendChild(div);

                    }

                })

            }

            function bodyLoad(){

                this.loadCategory();

                this.loadProducts("http://fakestoreapi.com/products");

                getCartItemsCount();

            }

            function categoryChange(){

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

                if(catName == "ALL"){

                    this.loadProducts("http://fakestoreapi.com/products")

                }else{

                    this.loadProducts(`http://fakestoreapi.com/products/category/${catName}`);

                }

            }

            var cartItems = [];

            function getCartItemsCount(){

                document.getElementById("lblCount").innerHTML = cartItems.length;

            }

            function addToCartClick(id){

                fetch(`https://fakestoreapi.com/products/1`)

                .then(response=>response.json())

                .then(function(data){

                    cartItems.push(data);

                    getCartItemsCount();

                    alert(`${data.title} \n Item added to cart`);

                })

            }

            function showCart(){

                document.querySelector("tbody").innerHTML = "";

                for(var item of cartItems){

                    var tr = document.createElement("tr");

                    var tdTitle = document.createElement("td");

                    var tdPrice = document.createElement("td");

                    var tdPhoto = document.createElement("td");

                    tdTitle.innerHTML = item.title;

                    tdPrice.innerHTML = item.price;

                    var img = document.createElement("img");

                    img.width="50";

                    img.height="50";

                    img.src=item.image;

                    tdPhoto.appendChild(img);

                    tr.appendChild(tdTitle);

                    tr.appendChild(tdPrice);

                    tr.appendChild(tdPhoto);

                    document.querySelector("tbody").appendChild(tr);

                }

            }

        </script>

    </head>

    <body onload="bodyLoad()" class="container-fluid">

        <header class="bg-danger text-danger text-white p-2 text-center">

            <h1><span class="bi bi-cart"></span>Fakestore Shopping</h1>

        </header>

        <section class="row mt-3">

            <div class="col-3">

                <div>

                    <label class="form-label">Select Category</label>

                    <div>

                        <select onchange="categoryChange()" id="lstCategories" class="form-select">

                        </select>

                    </div>

                </div>

            </div>

            <div class="col-7">

                <div id="catalog" class="d-flex overflow-auto flex-wrap" style="height: 500px;">

                </div>

            </div>

            <div class="col-2">

                <button onclick="showCart()" data-bs-target="#cart" data-bs-toggle="modal" class="btn btn-danger w-100">

                    <span class="bi bi-cart4"> [<span id="lblCount"></span>]</span>

                    Your Cart Items

                </button>

                <div class="modal fade" id="cart">

                    <div class="modal-dialog">

                        <div class="modal-content">

                            <div class="modal-header">

                                <h3>Your Cart Items</h3>

                                <button data-bs-dismiss="modal" class="btn-close"></button>

                            </div>

                            <div class="modal-body">

                                <table class="table table-hover">

                                    <thead>

                                        <tr>

                                            <th>Title</th>

                                            <th>Price</th>

                                            <th>Preview</th>

                                        </tr>

                                    </thead>

                                    <tbody>

                                    </tbody>

                                </table>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

        </section>

        <script src="../../node\_modules/jquery/dist/jquery.js"></script>

        <script src="../../node\_modules/bootstrap/dist/js/bootstrap.bundle.js"></script>

    </body>

</html>

**Session-57**

Nested Iteration

EX:

<!DOCTYPE html>

<html lang="en">

<head>

       <title>Nested List</title>

       <script>

        function bodyLoad(){

            fetch("../../data/menu.json")

            .then(res=>res.json())

            .then(function(data){

                for(var item of data){

                    var li = document.createElement("li");

                    li.innerHTML = item.category;

                    document.querySelector("ol").appendChild(li);

                    var optgroup = document.createElement("optgroup");

                    optgroup.label = item.category;

                    document.querySelector("select").appendChild(optgroup);

                    for(var product of item.products){

                        var ul = document.createElement("ul");

                        var ulItme = document.createElement("li");

                        ulItme.innerHTML= product;

                        ul.appendChild(ulItme);

                        li.appendChild(ul);

                        var option = document.createElement("option");

                        option.text=product;

                        option.value=product;

                        optgroup.appendChild(option);

                    }

                }

            })

        }

       </script>

</head>

<body onload="bodyLoad()">

    <ol>

    </ol>

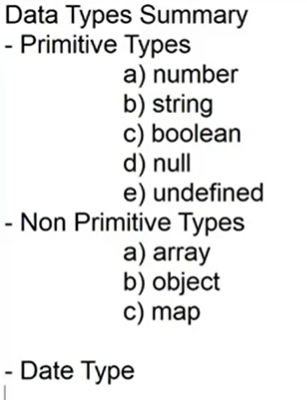
    <select>

    </select>

</body>

</html>

**Session-59**



FAQ : How to convert string into date ?

Ans: By using date constructor .

A close up of a number

Description automatically generated

JavaScript Operators

* Operator is an object that evaluates a value .
* Operator keeps data in operands .
* Operator defines functionality using entity “+, -, \*, /”

x+y -> x and y are operands

* Based on how many operands an operator can handle the operators are classified into

1. Unary - one operand

x++

1. Binary - two operands

x/y + z

1. Ternary - three operands

(operand1)?operand2:operand3

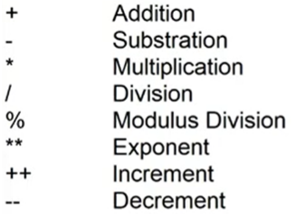
Ternary operator: “?:”

(condition) ? true : false;

* Operator are further classified into various types based on the type of value they return.

1. Arithmetic operators
2. Assignment operators
3. Conditional operators
4. Logical operators
5. Bitwise operators
6. Special operators

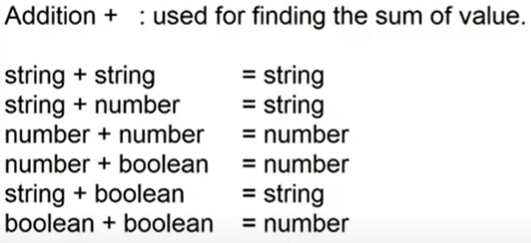
Arithmetic operators : They return always a number .



Math.pi

Math.cos

Math.sqrt



A group of black text

Description automatically generated

A close up of a text

Description automatically generated

A math equations on a white background

Description automatically generated

A white background with black text

Description automatically generated

**Session-60**

Practical done

**Session-61**

Comparison Operators

== equal

=== identical equal

!= not equal

!== not identical

> greater than

>= greater than equal

< less than

<= less than or equal

FAQ: What is difference between “==” and “===” ?

Ans: == can compare 2 values of various types.

Var x =”10” //string

Var y = 10 //number

X==y; //true

X===y ; //false

=== can compare 2 values of same type only.

Assignment Operator

+=

-+

\*\*=

Practicle done

**Session - 62**