**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>

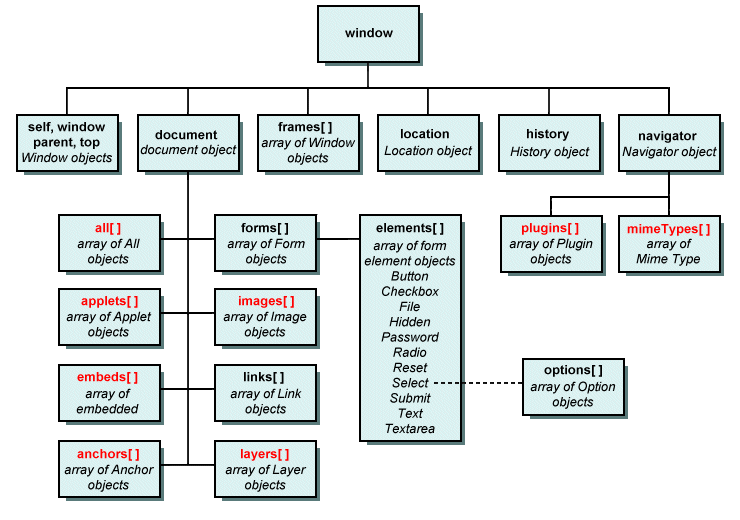
<noscript>Please Enable JavaScript</noscript>

</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">

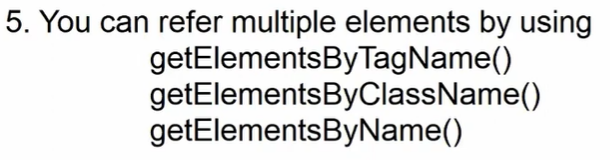
                <input type="email">

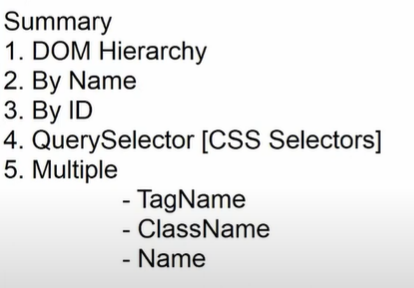
            </form>

        </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>

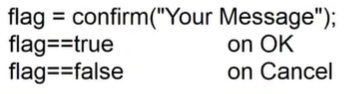
Confirm()

* 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>

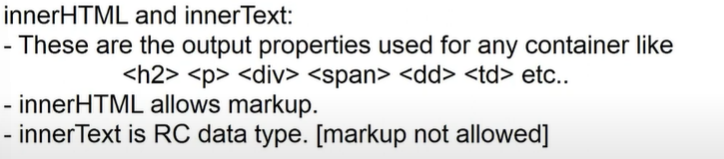
console methods ::

* 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>

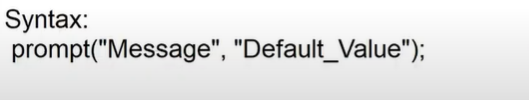
</html>

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>

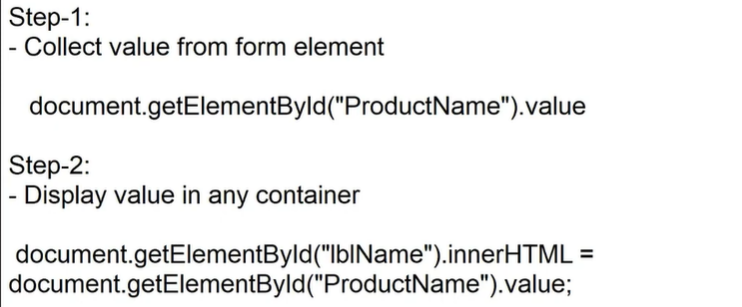
    </div>

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

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

    </body>

</html>

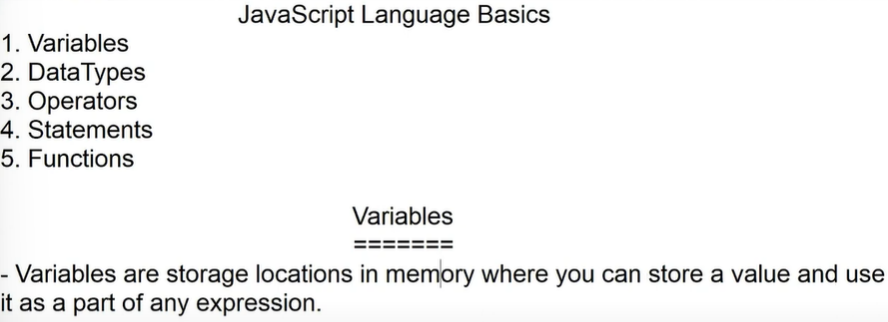




A close-up of a computer screen

Description automatically generated

**Session-48**

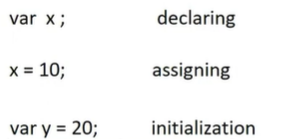


* Javascript can directly use variables and store value if it is not in strict mode.
* If javascript is in strict mode , it is mandatory to declare variables.
* Variables are declared by using 3 keywords.

1. var
2. let
3. const

* variable configuration comprises of 3 phases

1. declaring
2. initialization
3. assigning



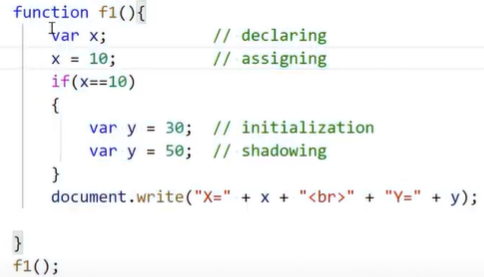
FAQ: If you don’t initialize or assign any value into variable , what is contains ?

Ans: “undefined”.

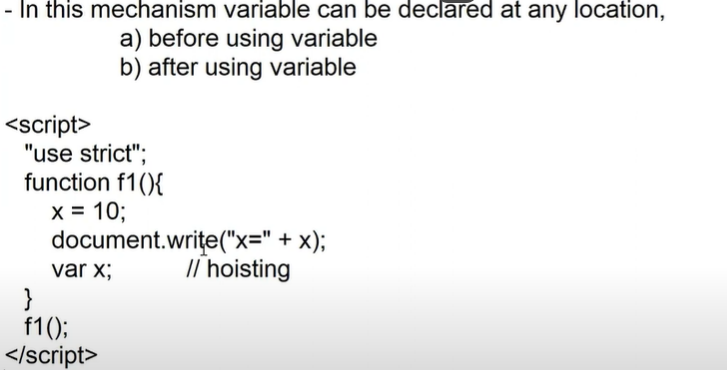
FAQ: What is difference between var , let and const ?

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.

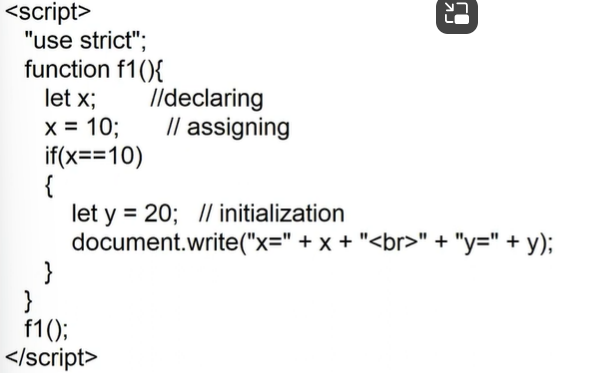


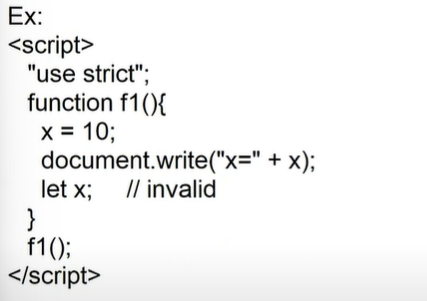
* Var allows hoisting.
* Hoisting is a technique used to indicate the location of carriable to compiler.



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

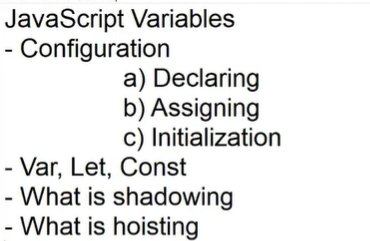
x = 10 ; //invalid

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(){

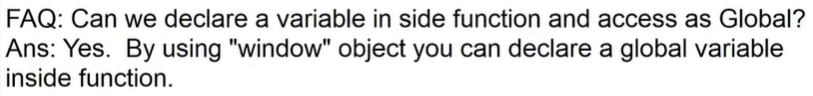
        document.write("Function 1 -<br> x="+x + "<br> y="+y+"<br>z="+z);

    }

    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 ?