**JAVASCRIPT**

**Day 1:**

1. **J**s is a object oriented and object based scripting language
2. Which is used to develop functon part of the webpage
   1. Character of the JS
   2. It is a program language or scripting language which means it will create the response with respect to the program
   3. It is dynamic language which means the JS code can be update in incoming lines.

**Day 2**

    console.log()🡪 To print in the Html

**Write Functions**

document.write() 🡪

document.writenln()🡪

**Day 3**

JavaScript program can be performed in 2 ways  
 1) internal JS

2)externa; JS

**INTERNAL JS**

Internal JS is performed by using <Script> which we have to specify inside the body tag after writing the html part of the program

Ex:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1 id="heading"> </h1>

<script>

    document.getElementById("heading").innerHTML="This is Inside the H! tag"

</script>

</body>

</html>

**External JS**

1. It is perform by using external js document
2. We have to link the external js by using script tag
3. In the script tag we have to specify src attribute for the src attribute we have pass url of the external Js Document
4. We use differ attribute to specify the browser to run HTML document at first then JS document

Ex : In HTML Document

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <script src="external.js" defer></script>

    <title>Document</title>

</head>

<body>

    <h1 id="heading"> </h1>

</body>

Ex: In JS Document

document.getElementById("heading").innerHTML="This is Inside the H! tag"

**JS Printing Statements**

In JS we have the printing statements like:

Console.log()

Document.write()

**CONSOLE.LOG**

* 1. It is used to debug the code
  2. The colsole.log print the output imn browser colsole
  3. In col=nsole.log console is a object and log () is a function which is inside the console object and it is accessed by using **“.”**

**EX:**

console.log("This is in Browser");

console.log(3+4);

console.log(console);

**Document.Write()**

1. It is used to print the output of the user interface
2. In document.write document is obj which refers to current html document,write() is function which is inside the document obj and it is accessed by using “.”
3. We don’t use document.write() on real time project coz I have its disadvantages like
   1. It will replace the parent document
   2. The coument.write () will not work ig the program is Asyncoroncely imported

Ex: document.write("This is on<b><i style='color:red'>User</i></b>Interface");

**Docment.writeln()**

1. it is simalr to document.write() the difference btn doc.wrt and document.writeln() is it gives single space btn the statements
2. Ex: document.writeln("This is one")
3. document.writeln("This is Two")

**Window.Alert()**

1. It is used to the output on the dialog of the browser (or) pop up window of the browser
2. The dialog will wait until the user dismisses the dialog box by click on ok btn
3. In Window.alert window is a obj and alert is is a func which is inside the window to access the alert we have to use “.”
4. Whenever a func or a obj or a property which havwe direct relationship with the window obj there is no need specify the window obj

Ex: window.alert("This is alert");

alert("This is 2nd");

**InnerHTML()**

1. It is used to pass the value inside the HTML elements.
2. To use the innerHTML properties we have to use the DOM methods to target the element.
3. In innerHTML it will consider html tag as a tag.

Ex:In the html document

<h1 id=”heading”></h1>

In the JS doc

document.getElementById("heading").innerHTML("This is in h1 Tag");

**InnerText**

1. It is used to pass the value inside the html elements.
2. To use innerText we have to use the DOM methods to target the element.
3. In innerText if we specify HTML elements It will be considered as text.

Ex:   In HTML 🡪  <span id="heading"></span>

In JS 🡪 document.getElementById("heading").innerText("This is on<b><i style='color:red'>User</i></b>Interface");

**JS tokens**

Tokens is a smallest unit of the program language.

In JS we have tokens like

1. Variable
2. Identifier
3. Value

**Variable**:

Variable are used to allocate the memory block ,in JS we have 3 types of variables they are

1. Var
2. Let
3. Const

**Var:**

It is a function scope variable in which we can perform Hoisting in Var we can redeclare and re initialise the variable

Ex: var num1=45;

Console.log(num1);

**Let:**

it is a variable with block scope in let we can perform reinitialization but we cannot perform redeclaration and hoisting

Ex: let num2=12;

Console.log(num2);

**Const:**

It is a block scope var if we declare the const in the same statement we have to initialize , the const value must kept constant thorough out the program which means we cannot reinitialize and redeclare.

It const we cannot perform the variable hoisting

Ex: const num3=65;

Console.log(num3);

**Variable Hoisting:**

Using the variable before its declaration is called hoisting

In JS we can perform variable hoisting only ***‘VAR’***

**Different between var, let, const**

|  |  |  |  |
| --- | --- | --- | --- |
| **Status** | **Var** | **Let** | **Const** |
| **Declaration** and **initialisation** | Yes | Yes | Yes |
| **Only declare** | Yes | Yes | No |
| **1st declare the initialisation** | Yes | Yes | No |
| **Reinitialization** | Yes | Yes | No |
| **Redeclare** | Yes | No | No |
| **Hoisting** | Yes | No | No |
| **Scope** | Function | Block | Block |

**Identifier**

1. It is the name for the memory block
2. To write identifier we have to follow some rules
3. We have to write identifier with camel casing
4. We should not start identifier with number /special characters except ‘\_’ and “$”
5. We should not use keyword as identifier
6. We should not give space between the identifier
7. Ther is no limit for len(identifier ) but we may face some problem with some JS engines if the identifier is longer than 31 char

**Value**

1. It is nothing but the data which is stored in the memory block
2. In javascript we have 2 types of data type
   1. Primitive
   2. Non Primitive

**Primitive Data Types:**

**I**n JS in primitive data type it will nor return obj type and it represents single value and immutable

In JS we have primitive data types like

1. Number 🡪 var num =45;
2. String 🡪 var str=”apple”;
3. Undefined 🡪 var un=undefined;
4. Null 🡪 var nu=null;
5. Symbol 🡪 var foo=Symbol();
6. BigInt 🡪 var num1=BigInt(4555245454)
7. Boolean 🡪 var bool1=true/false;

Note: Null is Primitive Data Type even after the return type is object coz it returns empty object

**Non Primitive Data Type**

1. It is also called as object Type coz the type of () Non primitive data type is Object type.
2. In JS we have nonprimitive data types like
3. Function
4. Array
5. Object

**Type of():**

1. It is a operator which return the data type of JS values
2. The type of always return string type output.

Ex: console.log(typeof true);

console.log(typeof null);

console.log(null);

console.log(typeof typeof 5);

**Type casting**

1. It is also known as type conversion, type casting is a process where we convert 1 data type into another data type
2. In type casting we have 2 types

They are

1. Implicit Type conversion
2. Explicit type converson

**Implicit Type conversion**

The data is going to convert automatically without any external function or constructor

**Conversion to String type:**

var a=10;

console.log(typeof a,a);

var b=’10’;

console.log(typeof b,b);

**Conversion to Number type:**

var b=’10’;

var a=10;

var a=a\*b;

console.log(typeof a,a);

**Conversion to Boolean type:**

Var b=’10’;

Var a=10;

Console.log(typeof (a==b));

**Explicit Type Conversion:**

It is called as manual type conversion in which we use functions and constructors

**Conversion to Number type:**

To convert the data into Number type we use

1. Number()
2. parseInt()
3. parseFloat()

Ex: var b=Number(a);

console.log(typeof b,b);

var c=parseInt(a);

console.log(typeof c,c);

var d=parseFloat(a);

console.log(typeof d,d);

Note:

Whenever we try to convert other data type into number type if its fails it will return NaN which stands for Not A Number which is a special type of number.

**Conversion to String type:**

To Convert other data into string type we can use

1. String()
2. toString()

Ex: // conversion of string

var str=String(a);

var st=a.toString();

console.log(typeof str,str);

console.log(typeof st,st);

**Conversion to Boolean Type:**

To Convert the Data Into Boolean Type we use boolean constructor

Ex: // For the Boolean data

var bool=Boolean(num)

// false

var num=NaN

var num=""

var num=0

var num="0"

//Except that if there  is any values even space is true

**JavaScript Input Statement**

In JS we have 2 input statements they are

1. confirm()
2. prompt()

**Confirm():**

It is a function which is used to collect the Boolean value from the dialog box which returns **“True or False”**

Ex: var bool=window.confirm("This is POP up box")

**Prompt**()**:**

It is used to collect the string input from the dialog box

Ex: var num1=prompt("Enter num1");

**JS OPERATOR**

Operator are used to perform the operation btw the operands

In JS we have operators like

1. Arithmetic
2. Assignment
3. Comparision
4. Logical
5. Bitwise
6. Increment and Decrement

**Arithmetic Operator**

It is used to perform arithmetic operation btw the operands.

In arithmetic operator we have operators like

1. Addition(+)
2. Subtraction(-)
3. Multiplication(\*)
4. Division(/)
5. Modulas(%)
6. Exponentation(\*\*)

Examples

console.log(num1+num2);

console.log(num1-num2);

console.log(num1\*num2);

console.log(num1/num2);

console.log(num1%num2);

console.log(num1\*\*num2);

**Assignment Operator**

It is used to assign the value to the js variables

In assignment operator we have

1. +=
2. -=
3. \*=
4. /=
5. =
6. ==

Example:

num1+=num2

num1-=num2

num1\*=num2

num1/=num2

num1%=num2

num1\*\*=num2

console.log(num1,num2);

**Comparison Operator:**

It is used in logical statement to determine equality or diference btw varible or values

In comparision operator we have operators like

1. **==** 🡪 First operand value equal to second operand value
2. **===** 🡪 first oprand value and data type equals to compare with second operand and data type
3. **!=** 🡪 First operand value not equal to second operand value
4. **!==**🡪 first oprand value and data type not equals to compare with second operand and data type
5. **>🡪** First operand value grater than second operand value
6. **>=🡪** First operand value grater than or equal to second operand value
7. **<🡪** First operand value less than second operand value
8. **<=🡪** First operand value Less than or equal to second operand value

Example

var a=10

var b='10'

console.log(a==b);

console.log(a===b);

console.log(a!=b);

console.log(a!==b);

console.log(a>b);

console.log(a>=b);

console.log(a<b);

console.log(a<=b);

**Logical Operators**

It is used to perform the logical operatoration in the statements

It retuerns the either “true or false”

In JS we have logical operators like

1. &&
2. ||
3. !

Example:

var a=10

var b='10'

console.log(a==b && b==c);

console.log(a==b || b==c);

console.log(!true);

**Bitwise oprator:**

1. AND(‘&’)
2. OR ,‘|’
3. NOT(’<>’)
4. RIGHT SHIFFT( >>)
5. LEFT SHIFT( << )
6. UNSIGNEDD RIGHT SHIFT(->>)

**Increment and Decrement**

To increment a variable we use ++

To decrement a variable with 1 we use –

Example:

var a=10

console.log(a++);

console.log(a);

console.log(++a);

console.log(a--);

console.log(a);

console.log(--a);

console.log(a+=a++);

console.log(a+=++a);