Title: Web Application Development –III
Course Code: US03IDBCA04
UNIT-1 Basics of JavaScript

- **TOPICS:-**
- JavaScript basics Syntax, Data Types, Variables, Literals, Type Casting, Operators, User interaction through dialog boxes, Built-in functions, Flow Control statements: Decision-Making and Looping, Arrays, User-defined functions

JAVASCRIPT BASICS : SYNTAX

- HTML script element offer to write JavaScript between opening <script> tag and closing </script> tag.
- The <script> tag alerts the browser program to start interpreting all the text between these tags as a script
- A simple syntax of your JavaScript will appear as follows:

```
<script language = "javascript"   type = "text/javascript">
          JavaScript code writes on here....
</script>
```

EXAMPLE:-

```
<html>
<head>
  <title> Your first JavaScript program </title>
<head>
<body>
  <script language = "javascript"  type = "text/javascript">
          document.write("Hello World!")
  </script>
</body>
</html>
```

- The script tag takes two important attributes
  - **Language:-**This attribute must be specify with valid value "javascript".
  - **Type:-**This attribute must be specify with valid value "Text/javascript".
- JavaScript written in
  - HEAD section.
  - BODY section.
  - Both HEAD section & BODY section.

JavaScript place on <body> tag

- You can write JavaScript directly body section inside opening <script> tag or closing </script> tag.
- Put JavaScript code at end of the body section is improve the page loading speed, Because HTML content loading is not blocking JavaScript code.

EXAMPLE:-

```
<html >
<head>
```

```
<title>JavaScript place on body tag</title>
</head>
<body>
   <script type="text/javascript">
      document.write("Hello World!");
      document.write("<h2> Hello World! </h2>");
   </script>
   </body>
</html >
```

JavaScript place on <head> tag

- JavaScript write on head section inside opening <script> tag or closing </script> tag.
- Sometimes page content depend on this JavaScript code. So we need to load
- JavaScript code before the Body Section.

EXAMPLE:-

```
<html >
<head>
<title> JavaScript place on body tag </title>
<script type="text/javascript">
          document.write("Hello World!");
          document.write("<h2> Hello World! </h2>");
  </script>
  </head>
<body>
</body>
</html>
```

COMMENTS LINE

- A comment is single or multiple lines, which give some information about the current program.
- Comments are not for execution.
- Write comment after double slashes // or write multiple lines of comments between /* and */

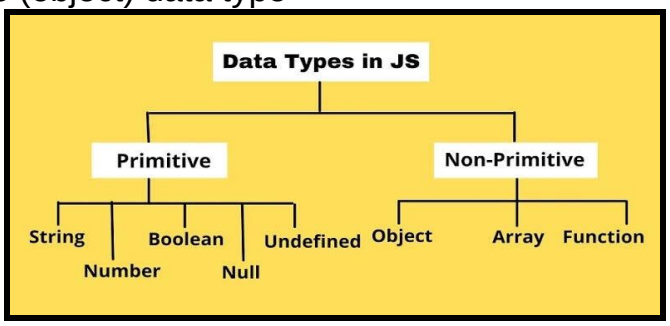
Example: Comment JavaScript Code

```
<script>
    var one =1; // this is a single line comment
    /* this
is multi line
comment*/
    var two = 2;
```

```
    var three = 3;
</script>
```

JAVASCRIPT DATA TYPES

- Data types in JavaScript refer to the types of the values that we are storing or working with.
- These are the type of values that can be represented and manipulated in a programming language.
- JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.
  1. Primitive data type
  2. Non-primitive (object) data type



1. Primitive data type

- Primitive data types are predefined (already defined) in JavaScript.
- Primitive types cannot be used to call methods to perform certain operations.
- Primitives are known as being immutable data types because there is no way to change a primitive value once it gets created.
- There are five types of primitive data types in JavaScript.
- They are as follows:
  - Number
  - String
  - Boolean
  - Symbol
  - Null
  - Undefined

- **Number Data types**
  - JavaScript has only one Number (numeric) data types.
  - Number data type can store normal integer number, floating-point number values.
  - A floating-point represents a decimal integer number.

- Extra large or extra small numbers can be written with scientific (exponential) notation

Example

```
var a = 5;              // Numeric integer value
var b = 10.5;           // Numeric float value
var c = -30.47;         // Negative numeric float value
var d = -35             //  Negative Numeric integer value
var y = 123e5;    // 12300000 //Exponential Notation
var z = 123e-5;   // 0.00123 //Exponential Notation
```

- **String Data types**
  - In JavaScript, the string is a sequence of characters and can be created using different ways given below −
    - Using the single quote
    - Using the double quote
  - Each character is representing as a element that occupies the position of that string.
  - Index value 0, starting from first character of the string.

Example

```
var name = 'Hello, I am run this town.!';    // Single quote
var name = "Hello, I am run this town.!";   // Double quote
var name = "Hello, I'm run this town.!";  // Single quote use
inside string
var name1 = "";                                  // empty string
```

- **Boolean Data types**
  - JavaScript Boolean type can have two values true or false.
  - Boolean type is use to perform logically operator to determine condition/expression is true.

Example

```
var val1 = true;
var val2 = false;
```

- **Null Data types**
  - JavaScript Null specifies variable is declared but values of this variable are empty.
  - When any variable's value is unknown, you can use the null.

Example

```
var str = null;                          // Value assign null
var car = " ";                           //  The value is "", the type of is "string"
document.write(str == undefined);        // Returns true
document.write(null == undefined);       // Equality check Returns true
document.write(null === undefined);      // Equality with type check Returns false
```

- **Undefined Data types**
  - In JavaScript, a variable without a value has the value undefined. The type is also undefined.
  - JavaScript uninitialized variables value are undefined. Uninitialized variable (value undefined) equal to null.
  - JavaScript uninitialized variables Boolean context return false.

Example

```
var str;            // Declare variable without value. Identify as undefined value.
var car;            // Value is undefined, type is undefined
car = undefined;    // Value is undefined, type is undefined
```

2. Non-primitive data type (object data type):-

- These data types are not actually defined by the programming language but are created by the programmer.
- Non-primitive data types are called **reference types** because they refer to objects.
- Non-primitive types can be used to call methods to perform certain operations,
- Non-Primitives **are known as mutable data types** because we can change the value after creation.
- The object data type can contain both built-in objects, and user defined objects:
- **Built-in object types** can be:
  - Objects
  - Maths objects
  - Functions objects
  - Dates objects
  - Arrays objects
  - string object

Creating/ Declare /Define object data type is create following different way

Syntax:-

```
var objectname = new Object();// Declare object
var  objectname = { }
var objectname = new Object;         // Declare object
var objectname = new Object( [ param1, param2, ..., paramN ] );  //
with parameter
```

Example

```
// Create an Array
var cars=new Array();
var cars = new Array("Saab", "Volvo", "BMW");
var cars = ["Saab", "Volvo", "BMW"];
const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
// Create a Date
var d = new Date();
const d = new Date("2022-03-25");
// Create a String
let x = "John";
let y = new String("John");
// Create a MATH
let x = Math.PI;
let y = Math.sqrt(16);
```

NOTE: Math can be called by using Math as an object, without creating it: **NOTE:** creating a JavaScript Object Using the **new Keyword**

NOTE: JavaScript Variables can be declared **Using var ,let, const Keyword.**

JAVASCRIPT VARIABLES

- Variables are used to store data in JavaScript.
- Variables are used to store reusable values.
- All JavaScript variables must be identified with unique names.
- These unique names are called identifiers (variable name).

The general rules for constructing names for variables (unique identifiers) are:

1. Variable names can contain letters (A TO Z, a To z), digits (0 To 9), underscores (_), and dollar signs ($).
2. Variable names must begin with a letter (A TO Z, a To z) underscores (_), and dollar signs ($) should not start with a number (0-9).
3. Variable names are case sensitive (y and Y are different variables).

4. Variable names cannot be used Reserved words (JavaScript keywords) (IF, SWITCH, ELSE, FOR, VAR, LET, NEW).

EXAMPLE:

```
var _abc = "Hi!";
var abc = "Hello!";
var 9abc = "Bye!";  // This is invalid
var for = "9"   // This is invalid keywords
```

Declaring a JavaScript Variable

- Creating a variable in JavaScript is called "declaring" a variable.
- JavaScript Variables can be declared in 4 ways:
  - Automatically
  - Using **var** keyword
  - Using **let** keyword
  - Using **const** keyword

Syntax: var variable_name;

Example:-

```
x = 5; // automatically declared
var y = 6; // declared Using var keyword
let z = 12; declared Using let keyword
const  a = 3.14; declared Using let keyword
```

Variable Initialization

- Storing a value in a variable is called variable initialization.
- The values of the variables are allocated using the assignment operator(“=”).

Syntax:

var variable_name = value;

you can declare a one or more variables in single statement separated by comma.

var var1 = val1, var2 = val2, ..., varN = valN;

EXAMPLE:

A JavaScript variable is a container for storing data.

```
var num = 5;
```

One or more variables declare and initialize

```
var num1 = 5, num2 = 6, num3 = 7;
```

JavaScript Variable Scope

- In JavaScript, the scope of a variable determines where it can be accessed within the code.
- In JavaScript, scope can be specified where you have to access JavaScript variables (functions or objects).

- Variables can be declared in different scopes:
  - Global Scope
  - Local Scope

Global scope:

- If you declare variables outside of the function, those variables scopes globally.
- You should access variable within function as well as outside of the function.

EXAMPLE:-

```
<script>
var data=200; //gloabal variable
function a()
{
document.write(data);
}
function b()
{
document.write(data);
}
a(); //calling JavaScript function
b();
</script>
```

Local variable:-

- A JavaScript local variable is declared inside block or function. It is accessible within the function or block only.

EXAMPLE:-

```
<script>
        var myVar = "global";      // Declare a global variable
        function abc( )
      {
              var myVar = "local";    // Declare a local variable
              document.write(myVar);
        }
  </script>
```

JAVASCRIPT LITERALS

- Literals in JavaScript mean values. When you create a variable in JavaScript, you assign it a value, that value is a literal.
- Literal is a fixed value that cannot be changed, you do not need to specify any keyword along with it.
- JavaScript supports various types of literals which are listed below:

- Numeric Literal
- Floating-point Literal
- Boolean Literal
- String Literal
- Array Literal
- Regular Expression Literal
- Object Literal

Numeric Literal in JS

- It can be a decimal value (base 10), a hexadecimal value (base 16), or an octal value(base 8).
- Decimal numeric literals consist of a sequence of digits (0-9) without a leading 0(zero).
- These are integer values that you will be using mostly in your JS code.

Numeric Literals Example:

```
120 // decimal literal
021434 // octal literal
0x4567 // hexadecimal literal
```

Floating-Point Literal

- It contains a decimal point, for example, the value 1.234
- A fractional value is a floating-point literal.
- It may contain an Exponent.

Floating-Point Literal Example:

```
6.99689 // floating-point literal
-167.39894 // negative floating-point literal
```

Boolean Literal in JS

- Boolean literal can have two values, either true or false.

```
true // Boolean literal
false // Boolean literal
```

String Literal in JS

- A string literal is a combination of characters (alphabets or numbers or special characters) enclosed within single(') or double quotation marks ("").

```
"Study" // String literal
'tonight' // String literal
```

Array Literal in JS

- An array literal is an array in JavaScript created using the square bracket ([]) with or without values.
- Whenever you create an array using an array literal, it is initialized with the elements specified in the square bracket.

Array Literal Example:

```
["Abhishek","Supriya","Joey"];  // Array literal
let students = ["Abhishek","Supriya","Joey"];  // Array literal
```

- Even when you assign an array to a variable, if you create the array directly using the square brackets, it is an array literal.
- In JavaScript, you can create an array using the Array object or using an array literal.

Object Literal in JS

- It is a collection of key-value pairs enclosed in curly braces ({}).
- The key-value pairs are separated using a comma.

Object Literal Example:

```
var games = {cricket :11, chess :2, carom: 4}  // Object literal
```

JAVASCRIPT TYPE CONVERSION (CASTING)

- Type Conversions in JavaScript refer to the automatic or explicit process of converting data from one data type to another in JavaScript.
- There are two types of type conversion in JavaScript:
  - **Implicit Conversion** - Automatic type conversion.
  - **Explicit Conversion** - Manual type conversion.

Implicit Type Conversion

- When type conversion is done by JavaScript automatically,
- it is called implicit type conversion.
- For example, when we use the '+' operator with the string and number operands, JavaScript converts the number to a string and concatenates it with the string.
- In programming, type conversion is the process of converting data of one type to another.

For example, converting string data to number.

```
<script>
    document.write("100" + 24 + "<br/>");
    document.write('100' + false + "<br/>");
    document.write("100" + null+ "<br/>");
    document.write("100" + undefined+ "<br/>");
  </script>
```

OUTPUT:-

```
10024,100false,100null,100undefined
```

In this example, we performed implicit type conversion using the + operator with a string and another data type.

Explicit Type Conversion

- In many cases, programmers are required to convert the data type of the variable manually. It is called the explicit type conversion.

EXAMPLE:-

```
<script>
    document.write(Number("200") + "<br/>");
    document.write(Number("1000e-2") + "<br/>");
    document.write(Number(false) + "<br/>");
    document.write(Number(null) + "<br/>");
    document.write(Number(undefined) + "<br/>");
    document.write(+"200" + "<br/>");
  </script>
```

OUTPUT:-

200 10 0 0 NaN 200

JAVASCRIPT OPERATORS

- JavaScript operators are special symbols that perform operations on one or more operands (values).

JavaScript has following types operators,

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators
- Conditional Operator (Ternary Operator)
- String Concatenation

JavaScript Arithmetic Operators

- The JavaScript arithmetic operators are used to perform mathematical calculations such as addition, multiplication, subtraction, division, etc. on numbers.
- JavaScript supports the following arithmetic operators

Operator	Description	Example

+ (Addition)	Adds two operands.	var x = 10, y = 5;
		document.write(x + y);
		RESULT: 15

- (Subtraction)	Subtracts the second	var x = 10, y = 5;

	operand from the first.	document.write(x - y); **RESULT:** 5
* (Multiplication)	Multiplies both operands.	var x = 10, y = 5; document.write(x * y); **RESULT**: **50**

/ (Division)	Divides the numerator by the denominator.	var x = 10, y = 5;
		document.write(x / y);
		RESULT:**Division: 2**

% (Modulus)	Outputs the remainder of an integer division.	var x = 10, y = 5;
		document.write(x % y);
		RESULT:: **0**

++ (Increment)	Increases an integer value by one.	**POST INCREMENT**
		var x = 10;
		document.write(x++);
		RESULT: x: 10,after increment x become now 11
		PRE INCREMENT
		var x = 10;
		document.write(++x);
		RESULT: x: 11,before increment x become now 11

-- (Decrement)	Decreases an integer value by one.	**POST DECREMENT**
		var x = 10;
		document.write(x--);
		RESULT: x: 10,after decrement x become now 9
		PRE DECREMENT
		var x = 10;
		document.write(--x);
		RESULT: x: 9,before decrement x become now 9

Assignment Operators

- In JavaScript, an assignment operator is used to assign a value to a variable.
- JavaScript supports the following assignment operators .

Operator	Description	Example

= (Simple Assignment)	Assigns values from the right side operand to the left side operand	var y=10;
		var x= y;
		document.write(("the value of",x);
		OUTPUT:x=10

+= (Add and Assignment)	It adds the right operand to the left operand and assigns the result to the left operand.	<script>
		var x = 10,y=5;
		x += y;
		document.write("the value of",x);
		</script>
		OUTPUT:x=15

−= (Subtract and Assignment)	It subtracts the right operand from the left operand and assigns the result to the left operand.	<script>
		var x = 10,y=5;
		x -= y;
		document.write("the value",x);
		</script>
		OUTPUT:x=5

*=(Multiply and Assignment)	It multiplies the right operand with the left operand and assigns the result to the left operand.	<script>
		var x = 10,y=5;
		x *= y;
		document.write("the value of",x);
		</script>
		OUTPUT:x=50

/= (Divide and Assignment)	It divides the left operand with the right operand and assigns the result to the left operand.	<script>
		var x = 10,y=5;
		x /= y;
		document.write("thee value",x);
		</script>
		OUTPUT:x=2

%= (Modules and Assignment)	It takes modulus using two operands and assigns the result to the left operand.	var x = 10,y=5;
		x %= y;
		document.write("thee value",x);
		</script>
		OUTPUT:x=0

Comparison Operators

- The JavaScript comparison operators compare two values and return a boolean result (True or False).
- JavaScript supports the following comparison operators

Operator	Description	Example
== (Equal)	Checks if the value of two operands is equal or not. If yes, then the condition becomes true.	var a=5,b=5; document.write(a == b); **OUTPUT: true**
!= (Not Equal)	Checks if the value of two operands is equal or not. If the values are not equal, then the condition becomes true.	var a=5,b=10; document.write(a != b); **OUTPUT: true**
> (Greater than)	Checks if the value of the left operand is greater than the value of the right operand. If yes, then the condition becomes true.	var a=5,b=10; document.write(a > b); **OUTPUT: false**
< (Less than)	Checks if the value of the left operand is less than the value of the right operand. If yes, then the condition becomes true.	var a=5,b=10; document.write(a< b); **OUTPUT: true**
>= (Greater than or Equal to)	Checks if the value of the left operand is greater than or equal to the value of the right operand. If yes, then the condition becomes true.	var a=5,b=5; document.write(a >= b); **OUTPUT: true**
<= (Less than or Equal to)	Checks if the value of the left operand is less than or equal to the value of the right operand. If yes, then the condition becomes true.	var a=5,b=5; document.write(a <= b); **OUTPUT: true**

Logical Operators

- The logical operators are generally used to perform logical operations on Boolean values (TRUE/FALSE).
- The logical operators are used to combine two or more conditions.
- JavaScript supports the following logical operators :

Operator	Sign	Description	EXAMPLE
Logical AND	**&&**	If first operand evaluates and returns a true only that evaluate the second operand otherwise skips. Return true if both are must be true, otherwise return false.	var a = 5, b = 10; (a != b) && (a < b); **OUTPUT:/ returns true**
Logical OR	**||**	Evaluate both operands, Return true if either both or any one operand true, Return false if both are false.	var a = 5, b = 10; (a > b) || (a == b); **OUTPUT: returns false** **(**a < b) || (a == b); **OUTPUT: returns true**
Logical NOT	**!**	Return the inverse of the given value result true become false, and false become true.	!(a < b); **OUTPUT:returns false** !(a > b); **OUTPUT:returns true**

Conditional Operator (also call Ternary Operator)

- JavaScript provides a special operator called ternary operator :? that assigns a value to a variable based on some condition.
- This is the short form of the if else condition.

Syntax:
<condition> ? <value1> : <value2>;

- The ternary operator starts with conditional expression followed by the ? operator.
- The second part (after ? and before :) will be executed if the condition turns out to be true.

- Suppose, the condition returns false, then the third part (after :) will be executed.

Example: Ternary operator

```
var  a = 10, b = 5;
var  c = a > b? a : b;  OUTPUT: value of c would be 10
var  d = a > b? b : a; OUTPUT: value of d would be 5
```

String Concatenation

- The + operator performs concatenation operation when one of the operands is of string type.
- The following example demonstrates string concatenation even if one of the operands is a string.

Example: + Operator with String

```
let a = 5, b = "Hello ", c = "World!", d = 10;
a + b; //returns "5Hello "
b + c; //returns "Hello World!"
a + d; //returns 15
b + true; //returns "Hello true"
c - b; //returns NaN; - operator can only used with numbers
```

USER INTERACTION THROUGH DIALOG BOXES(POPUP BOX)

- There are three types of dialog boxes provided by JavaScript.
- JavaScript provides built-in global functions to display popup message boxes for different purposes.
  - alert(message): Display a popup box with the specified message with the OK button.
  - confirm(message): Display a popup box with the specified message with OK and Cancel buttons.
  - prompt(message, defaultValue): Display a popup box to take the user's input with the OK and Cancel buttons.

1. Alert Box:-

- An alert box can be used to display a message (string passed to the alert method) or display some information as well as OK BUTTON.
- Javascript code will not continue processing until the OK BUTTON is clicked.
  - **Syntax:** alert("your message here");

EXAMPLE:-

```
<script language="javascript">
```

```
function abc()
{
        alert("Onclick event on button");
}
</script>
```

2. Prompt Box:-

- The Prompt box displays a predefine message.
- The prompt box also displays a single data entry field (textbox), which accepts user input.
- Prompt box displays OK BUTTON and CANCLE BUTTON
- If the OK BUTTON is clicked the text typed inside the textbox to be display.
- If the CANCEL BUTTON is clicked a NULL value is display.

Syntax: prompt("your message", "default value");

```
<script language="javascript">
function abc()
{
var a;
a=prompt("Enter your Name ","");
alert("Hello! How are you  "+a);
}
</script>
```

3. Confirm Box:-

- Confirm box displays predefine message and OK and CANCLE BUTTONS
- The confirm box, is program execution when the user will have to click either OK BUTTON or CANCLE BUTTON.
- If the user clicked OK button, the box returns true. If the user clicked Cancel button the box returns false.

Syntax: confirm("your message ");

Example:

```
<script type="text/javascript">
var r;
var r=confirm("Press a button");
if (r==true)
  {
  alert("You pressed OK!");
  }
```

```
else
  {
  alert("You pressed Cancel!");
  }
</script>
```

output: You pressed OK!

BUILT-IN FUNCTIONS

parseInt()

- The JavaScript Number parseInt() method is used to convert a string into an integer according to a specified 'radix'.
- The Radix parameter represents the base in a mathematical numeral system, and must be an integer within the range of 2 to 36, inclusive.
- If the input string is invalid, or if the specified radix is outside of this range, the method returns 'NaN'.

Syntax: parseInt(string, radix)

Parameter	Description
value	The string starts with an intger to be parsed.
radix	It is an integer represents base in mathematical numeral system.

EXAMPLE:-

```
<script>
parseInt("10",10) + "<br>" +
parseInt("10.00") + "<br>" +
parseInt("10.33") + "<br>" +
parseInt("34 45 66") + "<br>" +
parseInt("   60   ") + "<br>" +
parseInt("40 years") + "<br>" +
parseInt("He was 40");
</script>
```

OUTPUT: 10, 10, 10, 34, 60, 40, NaN

parseFloat()

- The JavaScript Number parseFloat() method is used for converting a string argument into a floating-point number.
- If the string does not start with a valid number, or if the first character of the passed string cannot be converted, "NaN" is returned.
- It ignores any leading or trailing white spaces before and after the string.

Syntax:parseFloat(value)

Parameter	Description
value	value Required. The value to be parsed.

EXAMPLE:-

```
<script>
document.getElementById("demo").innerHTML =
parseFloat("40.00") + "<br>" +
parseFloat("   40   ") + "<br>" +
parseFloat("40 years") + "<br>" +
parseFloat("40H") + "<br>" +
parseFloat("H40");
</script>
```

OUTPUT:40,40,40,40,NaN

isNAN() :-

- isNaN() is used to determine whether a given number is a "NaN" or not.
- The "NaN" stands for 'Not a Number'.
- The method check the value if the value returns a 'true', the given value is not a number, if the value returns a 'false' the given value is .

- **Syntax:-**isNaN(value)
- **Parameters:** The value to be checked.

EXAMPLE:-

```
<script>
  var value = 1234 ,value1=NaN;
     document.write("<br>Value is  ", isNaN(value));//return false
     document.write("<br>Value1 is  ", isNaN(value1));//return true
</script>
```

OUTPUT:-

Value is false
Value1 is true

Number() Function

- The Number() function converts the variable into a number.
- You can use it to change the data type of the variable.

SYNTAX:-var num = Number(val)

- Here val is a variable or value to convert into a number.
- It doesn't create a number object instead it returns a primitive value.

EXAMPLE

```
var x = 3.14;
var y = 3;
var z= "ABC";
document.write( "the value of",Number(x));
document.write( "the value of",Number(y));
document.write( "the value of",Number(z));
</script>
```

OUTPUT:

```
the value of x: 3.14
the value of y: 3
the value of z: NaN
```

FLOW CONTROL STATEMENTS: DECISION-MAKING AND LOOPING

Decision-Making(Conditional statements)

- Conditional statements are used to perform different actions based on different conditions.
- JavaScript Conditional Statements are
  - If Statement
  - If Else Statement
  - Else If Statement
  - switch Statement
  - break
  - continue

If Statement

- Here a JavaScript expression is evaluated.
- If the resulting value is true, the given statement(s) are executed.
- If the expression is false, then no statement would be not executed(otherwise skip the statement).

SYNTAX:-

```
if (expression)
{
  Statement(s) to be executed if expression is true
}
```

Example:

```
<script>
        if( 1 > 0)
        {
```

```
                alert("1 is greater than 0");
        }
        if( 1 < 0)
        {
                alert("1 is less than 0");
        }
</script>
```

if...else statement

- Here JavaScript expression is evaluated.
- If the resulting value is true, the given statement(s) in the 'if' block, are executed.
- If the expression is false, then the given statement(s) in the else block are executed.

Syntax:

```
if (expression)
{
  Statement(s) to be executed if expression is true
}
else
{
  Statement(s) to be executed if expression is false
}
```

EXAMPLE:

```
<script>
        var mySal = 500;
        var yourSal = 1000;
        if( mySal > yourSal)
        {
                alert("My Salary is greater than your salary");
        }
        else
        {
                alert("My Salary is less than or equal to your salary");
        }
   </script>
```

else if... statement

- The if...else if... statement (also called as if...else ladder)is an advanced form of if…else that allows JavaScript to make a correct decision out of several conditions.
- statement(s) are executed based on the true condition, if none of the conditions is true, then the else block is executed.

- **Syntax:** The syntax of an if-else-if statement is as follows −

```
if (expression 1)
 {
   Statement(s) to be executed if expression 1 is true
}
else if (expression 2)
{
   Statement(s) to be executed if expression 2 is true
}
else if (expression 3)
 {
   Statement(s) to be executed if expression 3 is true
}
else
{
   Statement(s) to be executed if no expression is true
}
```

EXAMPLE:-

```
<script>
var a=20;
if(a==10){
document.write("a is equal to 10");
}
else if(a==15)
{
document.write("a is equal to 15");
}
else if(a==20)
{
document.write("a is equal to 20");
}
Else
{
document.write("a is not equal to 10, 15 or 20");
}
```

```
</script>
```

Switch Case statement

- switch case is a conditional statement is used to execute different blocks of code depending on the value of an expression.
- The expression is evaluated, and if it matches the value of one of the case labels, the code block associated with that case is executed.
- If none of the case labels match the value of the expression the default case is executed.
- break − The statement keyword indicates the end of a particular case.Use break keyword to stop the execution and exit from the switch.
- default − The default keyword is used to define the default expression.
- When any case doesn't match the expression of the switch-case statement, it executes the default code block.

SYNTAX:-

```
switch (expression)
{
   case condition 1: statement(s)
   break;
   case condition 2: statement(s)
   break;
   ...

   case condition n: statement(s)
   break;
      default: statement(s)
}
```

EXAMPLE:-

```
<script>
var weekname = "wednesday";
switch (weekname) {
    case "monday":
        document.writeln("Monday working day.");
        break;
    case "tuesday":
        document.writeln("Tuesday working day.");
        break;
```

```
    case "wednesday":
        document.writeln("Wednesday government holiday.");
        break;
    case "thursday":
        document.writeln("Thursday working day.");
        break;
    case "friday":
        document.writeln("Friday is a last working day of the week.");
        break;
    case "saturday":
        document.writeln("Saturday week end.");
        break;
    case "sunday":
        document.writeln("Sunday week end.");
        break;
    default:
        document.writeln("No any case found.");
        // last line end of the case statement, no need to write break; keyword
}
</script>
```

Break Statement

- JavaScript break is special statement that can be used inside the loops, JavaScript break keyword terminates the current loop and execution control transfer after the end of loop

Syntax: The syntax of break statement in JavaScript is as follows −

```
break;
```

OR

```
break [label];
```

EXAMPLE:-

```
<script>
        EXAMPLE:-
         var i = 0;
        while(i <= 10)
{
        if (i == 5)
{
        document.write("terminated, i = 5");
```

```
        break;
        }
        document.write(i);
        i++;
        }
</script>
```
OUTPUT:-0 1 2 3 4 terminated, i = 5

Continue Statement

- JavaScript continue statement used inside the loops, continue statement skip the current iteration and execute to the next iterations of the loop.

Syntax: continue;
OR continue label;

EXAMPLE:-

```
<script>
   var i = 0;
   while(i < 10)
{
       i++;
       if (i == 5)
     {
          document.write("skipped, i = 5");
          continue;
       }
       document.write(i);
   }
</script>
```
OUTPUT:-1 2 3 4 skipped, i = 5 6 7 8 9 10

LOOPING

There are 3 types of loops in JavaScript.

- for loop
- while loop
- do-while loop

For Loop

- JavaScript for loop check condition and executes block of code for a specific number of times, until specified condition argument become false.
- JavaScript for loop accept 3 arguments separated by semicolons. All 3 arguments are optional.

Syntax:-

```
for (initialization; condition; increment/decrement)
{
   statements;     // Do stuff if for loop condition is true
}
```

- **Initialization**: var keyword to define and initialize counter variable. You can initialize the values of one or more counter variables. One or more counter variables separated by colon.
- **Condition**: Check the condition expression, for given condition true or not. If the condition true, execute the block of code otherwise terminates the for loop.
- **Increment/decrement**: It is specify end of loop iteration counter variable are count by 1.

Example:

```
<script>
   for( var num = 1; num <= 10; num++ )
  {
        document.write(num + ": iteration");
   }
         document.write("End of for loop");
</script>
```

OUTPUT:-

```
1: iteration
2: iteration
3: iteration
4: iteration
5: iteration
6: iteration
7: iteration
8: iteration
9: iteration
10: iteration
```

while Loop

- JavaScript while Loop check condition and execute while block for a specify number of times, until while condition become false.
- In JavaScript while loop check first condition. If condition becomes true then execute block of code.

- **Syntax:-**

```
while (condition)
 {
   statements;     // Do stuff if while true
   increment;      // Update condition variable value
}
```

EXAMPLE:-

```
<script>
            number = 1;
              while (number <= 10)
           {
               document.write(number + " times");
               number++;
            }
</script>
```

OUTPUT:-

1 times 2 times 3 times 4 times 5 times 6 times 7 times 8 times 9 times 10 times

Do While:-

- JavaScript do...while loop check condition and execute a block of code for a specify number of times, until the condition become false.
- JavaScript do...while block of statements executed before evaluated the condition.
- At least one time block of statements executes if the condition doesn't satisfy.

SYNTAX:-

```
do
{
   statements;     // Do stuff if while true
   increment;      // Update condition variable value
  }
while (condition)
```

Example:

```
<script>
   number = 1;
   do
   {
      document.writeln(number + " times");
      number++;
   }
while (number <= 10)
```

OUTPUT:-

```
1 times
2 times
3 times
4 times
5 times
6 times
7 times
8 times
9 times
10 times
```

ARRAY

- The JavaScript Array object lets you store multiple values in a single variable.
- It can hold various data types, including numbers, strings, objects, and even other arrays.
- It is often used when we want to store a list of elements and access them by a single variable.

Create an JavaScript Array

- Array syntax has three different forms, you can use any one to create an Array.

Syntax

```
my_array = [value1, value2, ..., valueN];
my_array = new Array(value1, value2, ..., valueN);
my_array = new Array(array_size);
```

Arguments

my_array: Required, Assign array values to this specified variable name.
value: Optionally, list of array values separated by comma.

array_size: Optionally, specifies an array size. This size indicates you'll store a maximum size of values.

EXAMPLE

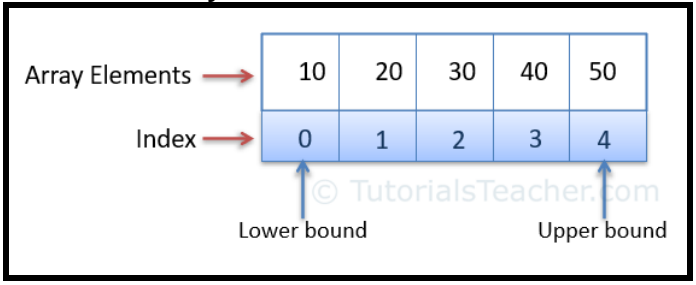
Array Literal Method You can use the array literal method to create an array. Specifies an array items, and assign to a variable.

```
var numArr = [10, 20, 30, 40, 50];
var domain = [".com", ".net", ".org", ".gov"];
```

- JavaScript Built-in Array Constructor Method You can also use built-in array constructor new Array() method to create an array. Specifies an array items, and assign to a variable.

```
var numArr = new Array(10, 20, 30, 40, 50);
var domain = new Array(".com", ".net", ".org", ".gov");
```

- In the above array, numArr is the name of an array variable.
- Multiple values are assigned to it by separating them using a comma inside square brackets as [10, 20, 30, 40, 50].
- Thus, the numArr variable stores five numeric values.
- The numArr array is created using the literal syntax and it is the preferred way of creating arrays.
- Every value is associated with a numeric index starting with 0. The following figure illustrates how an array stores values.



- JavaScript size of array length If you pass only one numeric argument to the Array constructor, the arguments assume the length of the array. you can assign values to an array variable with array indexes.

```
var domain = new Array(10);     // an empty array of length 10
```

Accessing Array Elements

- Array elements (values) can be accessed using an index.
- Specify an index in square brackets with the array name to access the element at a particular index

SYNTAX:-arrayName[index].

Note:- that the index of an array starts from zero.

EXAMPLE:-

```
<script type="text/javascript">
          // Create  empty array
          var domain = new Array();
          // Initializing(Assign) with Values
         domain[0] = ".com";
        domain[1] = ".net";
        domain[2] = ".org";
        domain[3] = ".gov";
         // Accessing  Array Elements
        document.write( domain [0] );
        document.write( domain[1] );
        document.write( domain[2] );
        document.write( domain[3] );
</script>
```

OUTPUT:- .com .net .org .gov

USER-DEFINED FUNCTIONS

- A function in JavaScript is a group of reusable code that can be called anywhere in your program.
- It eliminates the need of writing the same code again and again.
- It helps programmers in writing modular codes.
- Functions allow a programmer to divide a big program into a number of small and manageable functions.

Function declare(define/create)

Function Syntax

```
function functionName(arguments/ parameter)
{
   statements
}
```

EXAMPLE:

```
<script type="text/javascript">
   function msgfun()    //   function functionName
 {
      document.write("hello world!");     //  statements
   }
   msgfun();
```

Arguments
Function: function, is JavaScript reserved keyword
Function Name: Unique function name that is unique within programs.
Arguments (parameter): Optional, list of arguments separated by comma.

Calling a Function

- you would simply need to write the **name of that function with the parentheses ()** as shown in the following code.
- You can call or invoke the greet function by using the function name followed by the () , as shown below.
- When you call a function, JavaScript will execute the codes written inside the calling function.

EXAMPLE:-

```
<script>
            function greet()
           {
            alert("Hello World!");
            }
            greet();  // calling function
  </script>
```

Function Parameters(Arguments)

- You can pass values to a function using parameters.
- The function parameters in JavaScript are variables listed inside the parentheses in the function definition.
- A function can have multiple parameters separated by commas.
- The function can have one or more parameters and the values will be passed to function when it is called.
- We define function listing the parameters and call the function passing the arguments.

- **Syntax:-**The syntax to use function parameters in JavaScript is as follows −

```
function functionName (parameter1, parameter2, parameter3)
{
        //statements
}
```

In the above syntax, the function parameters are 'parameter1', 'parameter2', and 'parameter3'.

EXAMPLE:- Function Parameters

```
<script>
            function greet(firstName, lastName)  // define function parameters
```

```
                {
                      alert("Hello " + firstName + " " + lastName);
                }
                greet("Bill", "Gates"); // values will be  passed to function
</script>
```

OUTPUT:- Hello Bill Gates

Return a Value From a Function

- A function can return a value to the calling code using the return keyword followed by a variable or a value.

EXAMPLE: - Return a value of a Function

```
<script>
        var add = function (num1, num2)
    {
            return num1 + num2;  // Return a value
  };
        var result = add(10, 20);  // values will be  passed
        document.getElementById("p1").innerHTML = result;
</script>
```

OUTPUT:30

NOTE: This material was prepared using references from the websites:
https://www.tutorialspoint.com/javascript/index.htm
https://way2tutorial.com/javascript/tutorial.php
https://www.w3schools.com/js/
https://www.javatpoint.com/
https://www.geeksforgeeks.org/javascript/
https://www.tutorialsteacher.com/javascript