Unit 5: JavaScript

7.1 Overview of JavaScript

- JavaScript is a powerful and flexible programming language. It can execute on a web browser that allows us to make interactive webpages such as popup menus, animations, form validation, etc.
- JavaScript (JS) is a single threaded, Synchronous and blocking programming Language.
- **JavaScript** (JS) is a lightweight, interpreted programming language.
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation).
- JavaScript was designed to add interactivity and programming to HTML web pages.
  - Performs calculations such as totaling the price or computing sales tax.
  - Verifies data such as with credit card applications.
  - Adapts the display to user needs.
- JavaScript is case sensitive.
- JavaScript is object-based language as it provides predefined objects.
- JavaScript is both dynamically and loosely typed language.
  - "Loosely typed" means language does not bother with types too much, and does conversions automatically.
  - In dynamic typing, types are associated with values not variables.
  - "Dynamically typed language" → We do not need to declare variable type before use.

How to Use JavaScript

- There are several ways to use JavaScript, including:
  1. In the browser: Most modern browsers have a JavaScript console that we can use to test and run our code.
  2. In HTML: We can add JavaScript to our HTML files and run it in the browser. This is a great way to add interactivity to our website.
  3. With a runtime environment: One popular runtime environment for JavaScript is Node.js. Node.js was created when a programmer named Ryan Dahl took Google's V8 JavaScript engine and wrapped it in C++. This allows us to run JavaScript code outside of the browser, which is useful for server-side programming.

Limitations of JavaScript

- Client-side JavaScript does not allow the reading or writing of files.
- It cannot be used for networking applications because there is no such support available.
- It doesn't have any multithreading or multiprocessor capabilities.

Uses of JavaScript

- Client-side validation
- Dynamic drop-down menus
- Displaying date and time
- Validate user input in an HTML form before sending the data to a server.
- Build forms that respond to user input without accessing a server.
- Change the appearance of HTML documents and dynamically write HTML into separate Windows.
- Open and close new windows or frames.
- Manipulate HTML "layers" including hiding, moving, and allowing the user to drag them around a browser

window.

- Build small but complete client-side programs.
- Displaying popup windows and dialog boxes (like alert dialog box, confirm dialog box and prompt dialog box)
- Displaying clocks etc.

7.2 Advantages of JavaScript

- JavaScript is an interpreted programming as well as a scripting language. JavaScript is often executed directly in a client's browser commonly utilized in web development.
- It was originally developed by Netscape as a way to feature dynamic and interactive elements on websites.
- JavaScript is influenced by Java with a similar syntax to C. JavaScript follows the ECMAScript specifications which were developed by Sun Microsystems.
- Following are the advantages of JavaScript
    1. Speed
    2. Reduces load on the server
    3. Ease of use
    4. Rich Interface
    5. Versatility
    6. Extended functionality
    7. Interoperability
    8. Popularity
    9. Platform independence
    10. Powerful frameworks
    11. Procedural programming features
    12. Response to user activity
    13. Updates

1. Speed
    - JavaScript is an "interpreted" language, so it cuts down on the time needed for compilation in other languages like Java. Additionally, it is a client-side script that shortens the time required to establish a server connection, which speeds up program execution.
2. Reduces Load on the Server.
    - The language runs on the client side rather than the server. Thus, the server doesn't have to deal with the stress of executing JavaScript. Once this burden is reduced, the server will function more quickly and concentrate on other tasks like data management.
3. Ease of Use
    - JavaScript is one of the simplest languages to learn, particularly for web programming. It has been designed to be simple for web developers to understand and use.
4. Rich Interface
    - JavaScript offers developers a variety of interfaces to build engaging websites. Websites with drag-and-drop elements or sliders may have a more robust user experience. This increases user interaction with the website.
5. Versatile
    - The versatility of JavaScript is its most valuable quality. There are many different ways to integrate it into our website. Since Node.js integrates effectively with MongoDB and MySQL, it can not only build and finish the front end of the website but also handle its back end.
6. Extended Functionality
    - To save time and money, third-party add-ons such as Greasemonkey (a Mozilla Firefox extension) enable developers to incorporate small sections of prefabricated code into their code. These add-ons make it easier and

faster for developers to create JavaScript applications than they could with other coding languages.

7. Interoperability
   - ❖ JavaScript seamlessly integrates with other programming languages, so many developers favour using it to create various apps. Any webpage or script of another computer language can incorporate it.
8. Popularity
   - ❖ JavaScript is one of the most popular languages for web development. It is an important factor for every working website. Even the most popular websites worldwide, including Amazon and Google, use JavaScript since it is regarded as a very powerful technology. Due to its steadily growing popularity, it is now simpler than ever to learn this language online through various courses.
9. Independent Platform
   - ❖ Most browsers support JavaScript, making it simple for any browser to comprehend and recognize JavaScript code. We don't need to go through any installation or setup procedures because it is an easily available technology. Simply use our browser to create several webpage editing zones.
10. JavaScript Has Powerful Frameworks
    - ❖ Around JavaScript, many commanding frameworks display ready-to-use codes. Such codes are all easy to comprehend and troubleshoot. Depending on the structure in question, we will have access to several additional capabilities that will exponentially increase our efficiency.

7.3. Implementing JavaScript code to HTML page using SCRIPT tag

Embedding JavaScript:

- ❖ JavaScript scripts are embedded, either directly or indirectly, in HTML documents.

1. **Between the <body> </body> tag of html (Inline JavaScript)**
   - When java script was written within the html element using attributes related to events of the element then it is called as inline java script.
   - Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <input type="button" value="click" onclick="alert('Button Clicked')">
</body>
</html>
```

2. **Between the <head> </head> tag of html (Internal JavaScript)**
   - When java script was written within the head section using <script> element then it is called as internal java script.
   - Example: <script type = "text/javascript"> </script>

3. **In .js file (External JavaScript)**
    - <script type = “text/javascript” src = “file.js” > </script>
    - It separates HTML code from JavaScript code
    - It makes HTML and JavaScript easier to read and maintain
    - Cached JavaScript files can speed up page loads

JavaScript Comments:

- Comments are used to explain the code. They are not executed by the computer. They are just for the programmer to understand the code. There are two types of comments:
    1. Single line comments
    2. multi-line comments

1. Single Line Comments

- Single line comments are used to comment a single line of code. They are denoted by //.
- Everything after // is a comment and will not be executed by the computer.
- Example

```
// This is a single line comment
console.log("Hello World"); // This is also a single line comment
```

2. Multi-Line Comments

- Multi-line comments are used to comment multiple lines of code. They are denoted by /* and */.
- Everything between /* and */ is a comment and will not be executed by the computer.
- Example

```
/*
This is
a multi-line
comment.
*/
```

7.4. Variables in JavaScript

Variables and Constants in JavaScript:

- In programming, a variable is a container (storage area) to hold data.
- JavaScript Variables can be declared in 4 ways:
    1. Automatically
    2. Using var
    3. Using let
    4. Using const
- In JavaScript, there are three keywords that are used to declare variables: var, let and const. The var keyword was used to declare variables in JavaScript before the let and const keywords were introduced.
- The let and const keywords were introduced in ES6 (ECMAScript 6). ES6 is the latest version of JavaScript. It was released in 2015. ES6 introduced a lot of new features to JavaScript.

1. Automatically

- In this first example, x, y, and z are undeclared variables.
- They are automatically declared when first used:
- Example

```
<!DOCTYPE html>
<html>
<head>
    <title> Assignment Variable Automatically</title>
</head>
<body>
    <p> Assignment Variable Automatically</p>
    <script type="text/javascript">
                x = 5;
                y = 7;
                sum = x + y;
                document.write("Sum =", sum);
    </script>
</body>
</html>
```

2. var

- The var keyword is used to declare variables in JavaScript.
- One important thing to note about var is that it has not block scope. This means that if a variable is declared with var inside a function or Block{}, it is accessible from outside the block.
- Example:

```
<!DOCTYPE html>
<html>
<head>
    <title>Document</title>
</head>
<body>
    <p> Declare Variable using var Keyword</p>
    <script type="text/javascript">
        var name = "Anjan";
        var surname = "KC";
        document.write("FullName :", name," ",surname);
    </script>
</body>
</html>
```

3. let

- The let keyword is used to declare variables in JavaScript and has block scope. This means that if a variable is declared with let inside a block, it is only accessible within that block.
- Here's an example:

```
<!DOCTYPE html>
<html>
<head>
    <title>Document</title>
</head>
<body>
    <p> Declare Variable using let Keyword</p>
    <script type="text/javascript">
        let num1 = 5;
        let num2=5;
        let product=num1*num2;
        document.write("Product :",product);
    </script>
</body>
</html>
```

4. const

- The const keyword is used to declare variables in JavaScript and is used when we don't want to reassign the variable. The value of a variable declared with const cannot be changed.
- Here's an example:

```
<!DOCTYPE html>
<html>
<head>
    <title>Document</title>
</head>
<body>
    <p> Declare Variable using const Keyword</p>
    <script type="text/javascript">
        const PI=3.145;
        document.write("The Value of PI is: ", PI);
    </script>
</body>
</html>
```

Difference between let, Var and Const

	Block scoped	Hoisting	Reassignment	Initialization
let	Yes	No	Yes	Optional
var	No	Yes	Yes	Optional
const	Yes	No	No	Required

- **Block scoped**: let and const are block-scoped, meaning they are only accessible within the block they were defined in, while var is function-scoped, meaning it is accessible within the entire function it was defined in.
- **Hoisting:** let and const are not hoisted, meaning they cannot be accessed before they are declared, while var is hoisted, meaning it can be accessed before it is declared (although it will have a value of undefined).
- **Reassignment:** let and var can be reassigned to a new value, while const cannot be reassigned.
- **Initialization:** let and var can be declared without being initialized, while const must be initialized with a value when it is declared.

JavaScript Output:

- JavaScript Output defines the ways to display the output of a given code.
- The output can be displayed by using different approaches which are listed below
    1. Writing into an HTML element, using innerHTML.
    2. Writing into the HTML output using document.write().
    3. Writing into an alert box, using window.alert().
    4. Writing into the browser console, using console.log().

1. Using innerHTML
- To access an HTML element, JavaScript can use the document.getElementById(id) method.
- The id attribute defines the HTML element. The innerHTML property defines the HTML content:
- Example

```
<!DOCTYPE html>
<html>
<head>
    <title>innerHTML</title>
</head>
<body>
    <p>My First Paragraph.</p>
    <p id="demo"></p>
    <script type="text/javascript">
        let a=5;
        let b=7;
        let sum=a+b;
        document.getElementById("demo").innerHTML =sum;
    </script>
</body>
</html>
```

2. Using document.write()
- For testing purposes, it is convenient to use document.write()
- The document.write() method should only be used for testing.
- Example

```
<!DOCTYPE html>
<html>
<head>
    <title> Using Document.write()</title>
</head>
<body>
    <p>My First Paragraph.</p>
    <script type="text/javascript">
        let a = 5;
        let b = 7;
        let sum = a + b;
        document.write("Sum =" + sum);
    </script>
</body>
</html>
```

3. Using window.alert()
   - ❖ We can use an alert box to display data.
   - ❖ Example

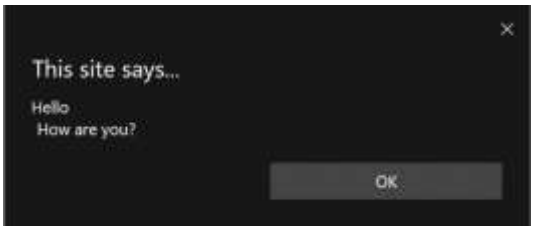
```
<!DOCTYPE html>
<html>
<head>
    <title>Document</title>
</head>
<body>
    <p>My First Paragraph.</p>
    <script type="text/javascript">
        window.alert("Kathmandu Institute of Technology");
    </script>
</body>
</html>
```

- ❖ window keyword is optional
- ❖ The window in which the browser displays an XHTML document is modeled with the Window object.
- ❖ The Window object represents the window in which the document containing the script is being displayed.
- ❖ The Window object is the default object for JavaScript, so properties and methods of the Window object may be used without qualifying with the class name
- ❖ Window includes three methods that create dialog boxes for three specific kinds of user interactions.
  - ➢ Alert() method
    - The alert() method displays an alert box with a specified message (string) and an OK button.
    - The string parameter of alert is plain text.

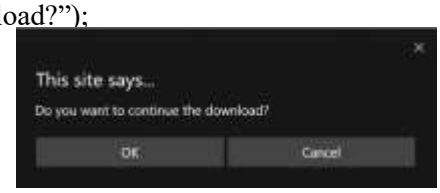
- Alert() box prevents the user from accessing other parts of the page until the box is closed.
- Syntax: alert(message);
- Example:
  alert("Hello\n How are you?");

  var sum=5;

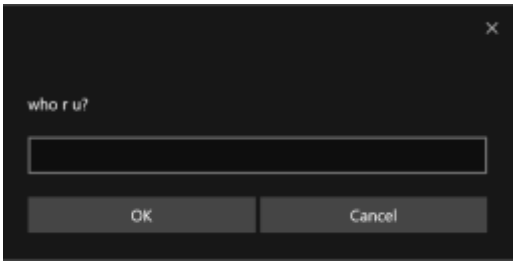
  alert("The sum is:" + sum + "\n");



- Confirm() method
  - The confirm() method displays a dialog box with a specified message, along with an OK and a Cancel button.
  - A confirm box is often used if you want the user to verify or accept something.
  - The confirm() method returns true if the user clicked "OK", and false otherwise.
  - Example:
    confirm("Press a button!\n Either OK or Cancel.");
    var question = confirm("Do you want to continue the download?");



- Prompt() method
  - The prompt() method displays a dialog box that prompts the visitor for input.
  - A prompt box is often used if you want the user to input a value.
  - When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
  - The prompt() method returns the input value if the user clicks "OK". If the user clicks "cancel" the method returns null.
  - If the user clicks OK without entering any text, an empty string is returned.
  - Example:
    Var name = prompt("What is your name?", " ");



4. Using console.log()
   - For debugging purposes, we can call the console.log() method in the browser to display data.
   - Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Document</title>
</head>
<body>
    <p>My First Paragraph.</p>
    <script type="text/javascript">
        console.log("Kathmandu Institute of Technology ");
        console.log("Sum=",5+6);
    </script>
</body>
</html>
```

7.5. JavaScript Data Type-Variant subtypes

JavaScript Data Types

- Data types in JavaScript define the data type that a variable can store. JavaScript includes primitive and non-primitive data types.
- The primitive data types in JavaScript include string, number, Boolean, undefined, null, and symbol.
- The non-primitive data type includes the object.
- A variable of primitive data type can contain only a single value.
- There are eight basic data types in JavaScript. They are:

Data Types	Description	Example
String	represents textual data	'hello', "hello world!" etc
Number	an integer or a floating-point number	3, 3.234, 3e-2 etc.
BigInt	an integer with arbitrary precision	900719925124740999n , 1n etc.
Boolean	Any of two values: true or false	true and false
undefined	a data type whose variable is not initialized	let a;
null	denotes a null value	let a = null;
Symbol	data type whose instances are unique and immutable	let value = Symbol('hello');
Object	key-value pairs of collection of data	let student = { };

1. **Primitive Data Types:**
   - The predefined data types provided by JavaScript language are known as primitive data types. Primitive data types are also known as in-built data types.
     - Number: JavaScript numbers are always stored in double-precision 64-bit binary format IEEE 754. Unlike other programming languages, we don't need int, float, etc to declare different numeric values.
     - String: JavaScript Strings are similar to sentences. They are made up of a list of characters, which is essentially just an "array of characters, like "Hello Nepal" etc.
     - Boolean: Represent a logical entity and can have two values: true or false.

- Null: This type has only one value that is null.
- Undefined: A variable that has not been assigned a value is undefined.
- Symbol: Symbols return unique identifiers that can be used to add unique property keys to an object that won't collide with keys of any other code that might add to the object.
- BigInt: BigInt is a built-in object in JavaScript that provides a way to represent whole numbers larger than 253-1.

2. **Non-Primitive Data Types:**
   - ❖ The data types that are derived from primitive data types of the JavaScript language are known as non-primitive data types. It is also known as derived data types or reference data types.
     - Object: It is the most important data type and forms the building blocks for modern JavaScript.
     - The Object Datatype can contain:
       - An object
       - An array
       - A date

JavaScript Operators:

- ❖ In JavaScript, an operator is a special symbol used to perform operations on operands (values and variables).
- ❖ For example

```
5+10; // Output: 15
```

- ❖ Here + is an operator that performs addition, and 5 and 10 are operands.

Different types of Operators in JavaScript

1. Assignment Operators
2. Arithmetic Operators
3. Comparison Operators
4. Logical Operators
5. Bitwise Operators
6. String Operators

1. JavaScript Assignment Operators

- ❖ Assignment Operators assign the values to the JavaScript variable.

Operator	Description	Example
=	Assignment	a = 5 Assign 5 to the variable a : a<=5
+=	Addition assignment	a += b is the same as a = a + b
-=	Subtraction assignment	a -= b is the same as a = a - b
*=	Multiplication assignment	a *= b is the same as a = a * b
/=	Division assignment	a /= b is the same as a = a / b
%=	Modulus assignment	a %= b is the same as a = a % b
**=	Exponential assignment	a **= b is the same as a = a ** b

2. JavaScript Arithmetic Operators

- Arithmetic operators are used to perform mathematical calculations. Here are the most common arithmetic operators in JavaScript

Operator	Description	Example
+	Addition	5 + 5 evaluates to 10
-	Subtraction	10 - 5 evaluates to 5
*	Multiplication	3 * 5 evaluates to 15
/	Division	25/ 5 evaluates to 5
%	Modulus	5 % 2 evaluates to 1
**	Exponentiation	2 ** 10 evaluates to 1024
++	Increment	++2 or 2++ evaluates to 3 (increments by 1)
--	Decrement	--2 or 2-- evaluates to 1 (Decrements by 1)

3. JavaScript Comparison Operators

- Comparison operators are used to compare two values and return a Boolean value (true or false) depending on whether the comparison is true or false. Here are the most common comparison operators in JavaScript:

Operator	Description	Example
==	Equal to	5 == 5 evaluates to true
===	Strict equal to (true=>if the operands are equal and of the same types)	5 === "5" evaluates to false
!=	Not equal to	5 != 4 evaluates to true
!==	Strict not equal to (true=>if the operands are equal but of different types or not equal at all)	5 !== "5" evaluates to true
>	Greater than	6 > 4 evaluates to true
<	Less than	6 < 4 evaluates to false
>=	Greater than or equal to	6 >= 6 evaluates to true
<=	Less than or equal to	6 <= 4 evaluates to false

4. JavaScript Logical Operators

Operator	Description	Example
&&	Logical AND. Returns true if both operands are true, otherwise false.	true && true evaluates to true
||	Logical OR. Returns true if at least one of the operands is true, otherwise false.	true || false evaluates to true
!	Logical NOT. Returns the opposite of the operand. If it's true, it returns false. If it's false, it returns true.	!true evaluates to false

5. **JavaScript String Operators**
   - ❖ in JavaScript, We can also use the + operators to concatenate (join) two or more strings.
   - ❖ Here's an example:

```
// String Concatenation operator(+)
let name = "Anjan";
let surname = "KC";
let fullName = name + " " + surname;
console.log(fullName);       // Anjan KC
console.log(name + surname);    //AnjanKC
```

JavaScript Conditional Statements:

- ❖ Conditional statements are used to perform different actions based on different conditions.
- ❖ In JavaScript, there are three forms of the if else statements.
  - if statement
  - if... else statements
  - if..else if..else statement
- ❖ Use if to specify a block of code to be executed, if a specified condition is true
- ❖ Use else to specify a block of code to be executed, if the same condition is false
- ❖ Use else if to specify a new condition to test, if the first condition is false.

1. **The if Statement**
   - ❖ Use the if statement to specify a block of JavaScript code to be executed if a condition is true.
   - ❖ Syntax

```
if (condition) {
  //  block of code to be executed if the condition is true
}
```

   - ❖ Example: check if the number is positive

```
<!DOCTYPE html>
<html>
<head>
    <title>Document</title>
</head>
<body>
    <script type="text/javascript">
     // check if the number is positive
        let num =prompt("Enter any Number");
        if (num > 0) {
            document.write("The Number is Positive");
        }
    </script>
</body>
</html>
```

2. The if else Statement

- Use the if statement to specify a block of JavaScript code to be executed if a condition is true.
- Use the else statement to specify a block of code to be executed if the condition is false.
- **Syntax**

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

- For Example: Write a program that checks if a number is even or odd.

```
<!DOCTYPE html>
<html>
<head>
    <title>Document</title>
</head>
<body>
    <script type="text/javascript">
       let num = prompt("Enter Any Number");
        if (num % 2 === 0) {
            document.write(num,"is a Even Number")
        } else {
            document.write(num, "is a Odd number");
        }
    </script>
</body>
</html>
```

3. The else if Statement

- Use the else if statement to specify a new condition if the first condition is false.
- Syntax

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```

- Example: Write a program that assigns a letter grade based on a numerical grade.

Above 90 -->"A"
Above 80 -->"B"
Above 70 -->"C"
Above 60 -->"D"
otherwise "F"

```
<!DOCTYPE html>
<html>
<head>
    <title>Document</title>
</head>
<body>
    <script type="text/javascript">
      let obtainMarks = prompt("Enter Your Obtain Marks");
        let grade;
        if (obtainMarks >= 90) {
            grade = "A";
        } else if (obtainMarks >= 80) {
            grade = "B";
        } else if (obtainMarks >= 70) {
            grade = "C";
        } else if (obtainMarks >= 60) {
            grade = "D";
        } else {
            grade = "F";
        }
        document.write("Obtain Grade is ", grade)
    </script>
</body>
</html>
```

7.6. JavaScript Functions

JavaScript Functions:

- A JavaScript function is a block of code designed to perform a particular task.
- A function is a group of statements that perform specific tasks.
- Syntax:

```
function functionName() {
    // Code to be executed
}
```

- We can store function expression in a variable, and then the variable can be used as a function:

Example:

```
let getSum = function (num1, num2) {
  let total = num1 + num2;
  return total;
};
let sum = (getSum(5, 10));
document.write(sum)    // 0utputs: 15
```

- There are two aspects of a JavaScript function.

1. **Declaring a Function**
   - A function must be declared globally in a JavaScript program to tell the compiler about the function name and function parameters.
   - The syntax to declare a function is:

```
function nameOfFunction () {
   // function body
}
```

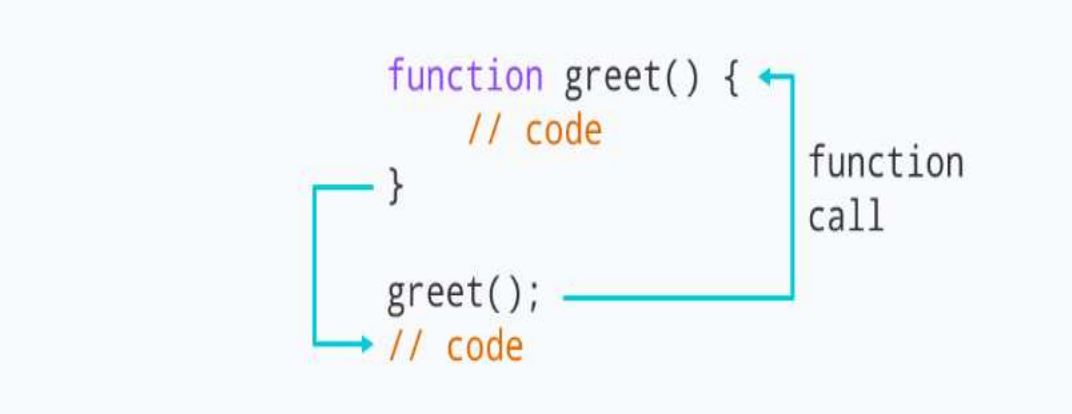
   - A function is declared using the function keyword.
   - The basic rules of naming a function are similar to naming a variable. It is better to write a descriptive name for our function.
   - For example, if a function is used to add two numbers, we could name the function add or addNumbers.
   - The body of function is written within {}.
   - For example,

```
// declaring a function named greet()
function greet() {
   console.log("Hello World !");
}
```

2. **Calling a Function**
   - Function can be called from anywhere in the program. The parameter list must not differ in function calling and function declaration. We must pass the same number of argument as it is declared in the function declaration.
   - In the above program, we have declared a function named greet (). To use that function, we need to call it.
   - Here's how we can call the above greet() function.

- Syntax:

```
// function call
greet();
```

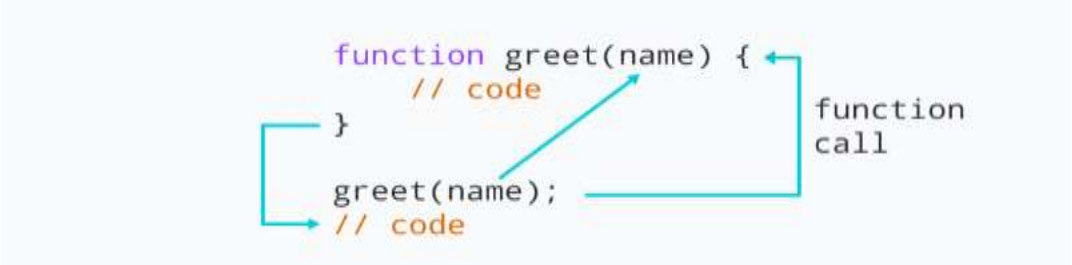


Example 1: Write a program to display "Welcome to KIT College" Using Function

```
<!DOCTYPE html>
<html>
<head>
    <title>Document</title>
</head>
<body>
    <script type="text/javascript">
        // declaring a function
        function displayMessage() {
            document.write("Welcome to KIT College");
        }
        // calling the function
        displayMessage();
    </script>
</body>
</html>
```

Function Parameters:

- A function can also be declared with parameters. A parameter is a value that is passed when declaring a function.



- Example 2: Write a Program to add two numbers using a function with Parameters

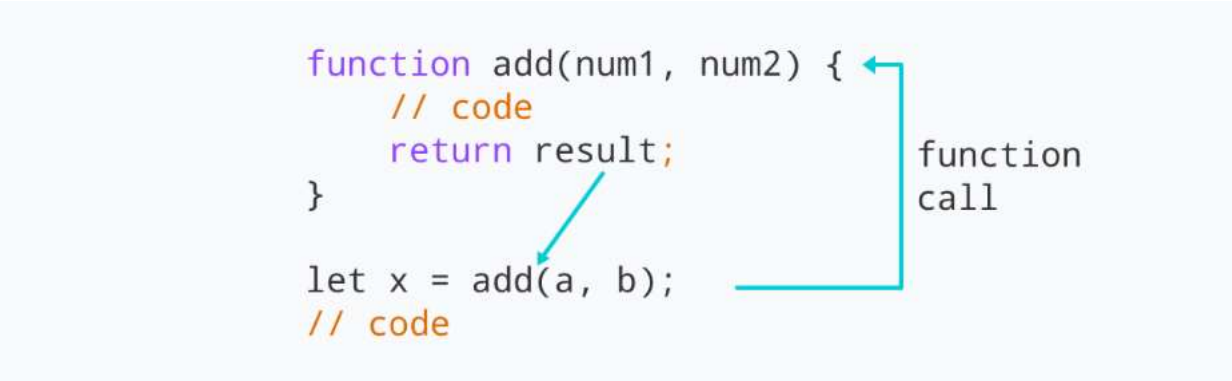
```
<!DOCTYPE html>
<html>
<head>
    <title>Function With Parameter</title>
</head>
<body>
    <script type="text/javascript">
        // declaring a function
        function addNumber(num1,num2) {
            let sum=num1+num2;
            document.write("Sum=",Number(sum));
        }
        // calling the function
       let firstNum=Number(prompt("Enter Fist Number"));
       let secondNum=Number(prompt("Enter Second Number"));
        addNumber(firstNum,secondNum);

    </script>
</body>
</html>
```

- In the above program, the addNumber() function is declared with num1 and num2 parameters. The user is prompted to enter two numbers (firstNum, secondNum). Then when the function is called, an argument is passed into the function.

Function Return

- The return statement can be used to return the value to a function call.
- The return statement denotes that the function has ended. Any code after return is not executed.
- If nothing is returned, the function returns an undefined value



Example 4: Write a Program to add two numbers using a function return with Parameters

```
<!DOCTYPE html>
<html>
<head>
    <title>Function With Parameter</title>
</head>
<body>
    <script type="text/javascript">
        // declaring a function
        function addNumber(num1, num2) {
            let sum = num1 + num2;
            return(sum);
        }
        // calling the function
        let firstNum = Number(prompt("Enter Fist Number"));
        let secondNum = Number(prompt("Enter Second Number"));
        let sum =addNumber(firstNum, secondNum);
        document.write("Sum=",sum)
    </script>
</body>
</html>
```

7.7. Event Handling and JavaScript objects

JavaScript Events:

- The change in the state of an object is known as an Event. In html, there are various events which represents that some activity is performed by the user or by the browser.
- When JavaScript code is included in HTML, JS react over these events and allow the execution. This process of reacting over the events is called Event Handling.
- Thus, JS handles the HTML events via Event Handlers.
- Examples: In HTML onclick is the event listener, myFunction is the event handler:

<button onclick="myFunction()">Click me</button>

Various Types of Event Handlers in JavaScript

- JavaScript offers a variety of event handlers that are activated in response to specific actions taken on the HTML elements.
- Some of the HTML events and their event handlers are:

1. **Mouse events:**

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

2. **Keyboard events**

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

3. **Form events**

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

4. **Window/Document events:**

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

Example: MouseOver Event

```
<html>
    <head>
        <script language="Javascript" type="text/Javascript">
    function mouseoverevent()
    {
        alert("Thi is Mouse Over Event");
    }
</script>
    </head>
<body>
    <h1> Javascript Events </h1>
<p onmouseover="mouseoverevent()"> Keep cursor over me</p>
</body>
</html>
```

7.8 Document Object Model (DOM) in JavaScript

DOM (Document Object Model):

- The Document Object Model (DOM) is a platform and language independent model to represent the HTML or XML documents.
- It defines the logical structure of the documents and the way in which they can be accessed and manipulated by an application program.
- In the DOM, all parts of the document, such as elements, attributes, text, etc. are organized in a hierarchical tree-like structure;
- With the HTML DOM, we can use JavaScript to
    - build HTML documents
    - navigate their hierarchical structure
    - add, modify, or delete elements and attributes or their content, and so on.
- The Document Object Model or DOM is basically a representation of the various components of the browser and the current Web document (HTML or XML) that can be accessed or manipulated using a scripting language such as JavaScript.
- The DOM is a W3C (World Wide Web Consortium) standard.
- "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."
- With the HTML DOM, JavaScript gets all the power it needs to create dynamic HTML:
    - JavaScript can change all the HTML elements in the page
    - JavaScript can change all the HTML attributes in the page
    - JavaScript can change all the CSS styles in the page
    - JavaScript can remove existing HTML elements and attributes
    - JavaScript can add new HTML elements and attributes

- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

➢ When a web page is loaded, the browser creates a Document Object Model of the page.

➢ The above diagram demonstrates the parent/child relationships between the nodes. The topmost node i.e. the Document node is the root node of the DOM tree, which has one child, the <html> element. Whereas, the <head> and <body> elements are the child nodes of the <html> parent node.

➢ Example: This example illustrates the dom-manipulation using getElementById() Method.

```
<!DOCTYPE html>
<html>
<head>
    <title>DOM manipulation</title>
</head>
<body>
    <label>Enter First Number: </label>
    <input type="text" id="num1" />
    <br>
    <br>
    <label>Enter Second Number: </label>
    <input type="text" id="num2" />
    <br>
    <button onclick="getAdd()">Click To Add</button>
    <p id="result"></p>
    <script type="text/javascript">
        function getAdd() {
            let num1 = Number(document.getElementById("num1").value);
            let num2 = Number(document.getElementById("num2").value);
            let add = num1 + num2;
            console.log(add);
            document.getElementById("result").innerHTML = "Addition : " +add;
            // Changes the color of paragraph tag with red
            document.getElementById("result").style.color = "red";
        }
    </script>
</body>
```

```
</html>
```

7.9 Dialog Box supported by JavaScript

- There are three types of dialog boxes supported in JavaScript that are alert, confirm, and prompt. These dialog boxes can be used to perform specific tasks such as raise an alert, to get confirmation of an event or an input, and to get input from the user

1. **Alert**
   - It is used to provide a warning message to users and mainly used for validation. It displays a message in the dialog box. It is one of the most widely used dialog box in JavaScript. It has only one 'OK' button to continue and select the next task.
   - Example:

```
<!DOCTYPE html>
<html>
    <head>
            <script>
        function show() {
            alert("This is Warring Message!");
}
</script>
    </head>
<body>
<h2>JavaScript Alert</h2>
<button onclick="show()"> Click Me</button>
</body>
</html>
```

2. **Confirm**
   - A confirm box is often used if we want to verify or accept something. When a confirm box pops up, the user will have to click either “OK” or “Cancel” to proceed.
   - If the user clicks on the OK button, the window method confirm() will return true. If the user clicks on the Cancel button, then confirm() returns false and will show null.
   - Example:

```
<!DOCTYPE html>
<html>
    <head>
    <script type="text/javascript">
    function Confirmation() {
        let value = confirm("Do you want to continue ?");
        if (value == true) {
            document.write(" CONTINUED!");
            return true;
```

```
        } else {
            document.write("NOT CONTINUED!");
            return false;
        }
    }
</script>
    </head>

<body>
<h2>Click the button to check the Confirm Box functionality</h2>
<form>
    <input type="button" value="Click Me" onclick="Confirmation();" />
</form>
</body>
</html>
```

3. Prompt

- The prompt dialog box is very useful when we want to pop-up a text box to get user input. Thus, it enables us to interact with the user. The user needs to fill in the field and then click OK.
- The prompt dialog box also has two buttons, which are OK and Cancel. The user needs to provide input in the textbox and then click OK. When a user clicks on the OK button, then the dialog box reads that value and returns it to the user. But on clicking the Cancel button, prompt() method returns null.

```
<html>
   <head>
      <script type = "text/javascript">
            function getValue() {
               let name = prompt("Enter your name : ", "your name here");
               document.write("You have entered : Name => " + name);
            }
      </script>
   </head>
   <body>
      <h2>Click the following button to see the result: </h2>
      <form>
         <input type = "button" value = "Click Me" onclick = "getValue();" />
      </form>
   </body>
</html>
```

7.10. Form validation

- It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user.
- JavaScript provides facility to validate the form on the client-side so data processing will be faster than server-side validation. Most of the web developers prefer JavaScript form validation.
- Through JavaScript, we can validate name, password, email, date, mobile numbers and more fields.
- The data entered into a form needs to be in the right format and certain fields need to be filled in to effectively use the submitted form. Username, password, and contact information are some details that are mandatory in forms.
- JavaScript Form Validation Example:

RegistrationForm.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title> Form validation using HTML and JavaScript</title>

</head>
<body>
    <h1> REGISTRATION FORM </h1>
    <form name="RegForm"
          onsubmit="return validateForm()">
            <label for="name">Name:</label>
            <input type="text" id="name" name="Name" placeholder="Enter Your Name"/>
        <br><br>
            <label for="address">Address:</label>
            <input type="text" id="address" name="Address" placeholder="Kathmandu,Nepal"/>
        <br><br>
            <label for="email">E-mail Address: </label>
            <input type="text" id="email" name="EMail" placeholder="example@gmail.com"/>
        <br><br>
            <label for="password">Password: </label>
            <input type="password" id="password" name="Password" />
        <br><br>
            <label for="repassword">Retype Password:</label>
            <input type="password" id="repassword" name="rePassword" />
            <br><br>
            <label for="course">Select Your Course: </label>
            <select value="" id="course" name="Course">
                <option>Select Course</option>
                <option>CIVIL</option>
```

```
                <option>Computer</option>
                <option>Electrical </option>
                <option>Architecture</option>
            </select>
            <br><br>
            <label for="message">Message :</label>
            <textarea cols="55" id="message" name="Message"> </textarea>
            <br><br>
            <input type="submit" value="Send" name="Submit" />
            <input type="reset" value="Reset" name="Reset" />
    </form>
    <script src="formValidation.js"></script>
</body>
</html>
```

formValidation.js

```
function validateForm() {
  const name = document.forms.RegForm.Name.value;
  const email = document.forms.RegForm.EMail.value;
  const what = document.forms.RegForm.Course.value;
  const password = document.forms.RegForm.Password.value;
  const rePassword = document.forms.RegForm.rePassword.value;
  const address = document.forms.RegForm.Address.value;
  console.log(name, email, what, password, address);

  if (name === "" || name == null) {
    window.alert("Name Field is Required");
    return false;
  }

  if (address === "") {
    window.alert("Please enter your address.");
    return false;
  }
  if (email === "" || !email.includes("@")) {
    window.alert("Please enter a valid e-mail address.");
    return false;
  }

  if (password === "") {
    alert("Please enter your password");
    return false;
  }
```

```
  if (password.length < 6) {
    alert("Password should be atleast 6 character long");
    return false;
  }
  if (password !== rePassword) {
    alert("Password Does Not Matched");
    return false;
  }

  if (what.selectedIndex === -1) {
    alert("Please enter your course.");
    return false;
  }

  return true;
}
```