JavaScript

JavaScript is the most popular and widely used client-side scripting language. Client-side scripting refers to scripts that run within your web browser. JavaScript is designed to add interactivity and dynamic effects to the web pages by manipulating the content returned from a web server

JavaScript was originally developed as **LiveScript by Netscape** in the mid 1990s. It was later renamed to JavaScript in 1995, and became an ECMA standard in 1997. Now JavaScript is the standard client-side scripting language for web-based applications, and it is supported by virtually all web browsers available today, such as Google Chrome, Mozilla Firefox, Apple Safari, etc.

JavaScript is officially maintained by ECMA (European Computer Manufacturers Association) as ECMAScript. ECMAScript 6 (or ES6) is the latest major version of the ECMAScript standard.

What You Can Do with JavaScript

- You can modify the content of a web page by adding or removing elements.
- You can change the style and position of the elements on a web page.
- You can monitor events like mouse click, hover, etc. and react to it.
- You can perform and control transitions and animations.
- You can create alert pop-ups to display info or warning messages to the user.
- You can perform operations based on user inputs and display the results.
- You can validate user inputs before submitting it to the server.

Example 1

Embedding JavaScript

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Embedding JavaScript</title>
</head>
<body>
   <script>
   var greet = "Hello World!";
   document.write(greet); // Prints: Hello World!
   </script>
</body>
</html>
```

Example 2
Externatl JavaScript

Hello.js
```
// A function to display a message
```

```
function sayHello() {
   alert("Hello World!");
}

// Call function on click of the button
document.getElementById("myBtn").onclick = sayHello;
```

pro2.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Including External JavaScript File</title>
</head>
<body>
   <button type="button" id="myBtn">Click Me</button>
   <script src="js/hello.js"></script>
</body>
</html>
```

Example 3

Placing JavaScript Inline

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Inlining JavaScript</title>
</head>
<body>
   <button onclick="alert('Hello World!')">Click Me</button>
</body>
</html>
```

Difference between Client-side and Server-side Scripting

Client-side scripting languages such as JavaScript, VBScript, etc. are interpreted and executed by the web browser, while server-side scripting languages such as PHP, ASP, Java, Python, Ruby, etc. runs on the web server and the output sent back to the web browser in HTML format.

Client-side scripting has many advantages over traditional server-side scripting approach. For example, you can use JavaScript to check if the user has entered invalid data in form fields and show notifications for input errors accordingly in real-time before submitting the form to the web-server for final data validation and further processing in order to prevent unnecessary network bandwidth usages and the exploitation of server system resources.

JavaScript Syntax

A JavaScript consists of JavaScript statements that are placed within the <script></script> HTML tags in a web page, or within the external JavaScript file having .js extension.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>Example of JavaScript Statements</title>
</head>
<body>
   <script>
   var x = 5;
   var y = 10;
   var sum = x + y;
   document.write(sum); // Prints variable value
   </script>
</body>
</html>
```

Case Sensitivity in JavaScript

JavaScript is case-sensitive. This means that variables, language keywords, function names, and other identifiers must always be typed with a consistent capitalization of letters.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>JavaScript Case Sensitivity</title>
</head>
<body>
   <script>
   var myVar = "Hello World!";
   console.log(myVar);
   console.log(MyVar);
   console.log(myvar);
   </script>
   <p><strong>Note:</strong> Check out the browser console by pressing the f12 key on the
keyboard, you'll see a line something like this: "Uncaught ReferenceError: MyVar is not
defined".</p>
</body>
</html>
```

JavaScript Variables

What is Variable?

Variables are fundamental to all programming languages. Variables are used to store data, like string of text, numbers, etc. The data or value stored in the variables can be set, updated, and retrieved whenever needed. In general, variables are symbolic names for values.

You can create a variable with the var keyword, whereas the assignment operator (=) is used to assign value to a variable, like this: var varName = value;

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Creating Variables in JavaScript</title>
</head>
<body>
  <script>
  // Creating variables
  var name = "Peter Parker";
  var age = 21;
  var isMarried = false;

  // Printing variable values
  document.write(name + "<br>");
  document.write(age + "<br>");
  document.write(isMarried);
  </script>
</body>
</html>
```

Example 2

```
     <!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Declaring Variables in JavaScript</title>
</head>
<body>
  <script>
  // Declaring Variable
  var userName;

  // Assigning value
  userName = "Clark Kent";
```

```
    // Printing variable values
    document.write(userName);
    </script>
</body>
</html>
```

Example 3

```
 <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Declaring Multiple Variables in JavaScript</title>
</head>
<body>
    <script>
    // Declaring multiple Variables
    var name = "Peter Parker", age = 21, isMarried = false;

    // Printing variable values
    document.write(name + "<br>");
    document.write(age + "<br>");
    document.write(isMarried);
    </script>
</body>
</html>
```

The let and const Keywords (ES6)

ES6 introduces two new keywords let and const for declaring variables.

The const keyword works exactly the same as let, except that variables declared using const keyword cannot be reassigned later in the code.

Example

```
 <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Declaring Variables with let and const Keywords in JavaScript</title>
</head>
<body>
    <script>
    // Declaring variables
    let name = "Harry Potter";
    let age = 11;
    let isStudent = true;

    // Printing variable values
    document.write(name + "<br>");
    document.write(age + "<br>");
    document.write(isStudent + "<br>");

    // Declaring constant
    const PI = 3.14;

    // Printing constant value
    document.write(PI); // 3.14

    // Trying to reassign
    PI = 10; // error
    </script>
    <p><strong>Note:</strong> Please check out the browser console by pressing the f12 key on the
keyboard.</p>
</body>
</html>
```

JavaScript Generating Output

Generating Output in JavaScript

There are certain situations in which you may need to generate output from your JavaScript code. For example, you might want to see the value of variable, or write a message to browser console to help you debug an issue in your running JavaScript code, and so on.

In JavaScript there are several different ways of generating output including writing output to the browser window or browser console, displaying output in dialog boxes, writing output into an HTML element, etc.

Writing Output to Browser Console

You can easily outputs a message or writes data to the browser console using the console.log() method.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>Writing into the Browser's Console with JavaScript</title>
</head>
<body>
   <script>
   // Printing a simple text message
   console.log("Hello World!"); // Prints: Hello World!

   // Printing a variable value
   var x = 10;
   var y = 20;
   var sum = x + y;
   console.log(sum); // Prints: 30
   </script>
   <p><strong>Note:</strong> Please check out the browser console by pressing the f12 key on the
keyboard.</p>
</body>
</html>
```

Displaying Output in Alert Dialog Boxes

You can also use alert dialog boxes to display the message or output data to the user. An alert dialog box is created using the alert() method.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>Writing into an Alert Dialog Box with JavaScript</title>
</head>
```

```
<body>
   <script>
   // Displaying a simple text message
   alert("Hello World!"); // Outputs: Hello World!

   // Displaying a variable value
   var x = 10;
   var y = 20;
   var sum = x + y;
   alert(sum); // Outputs: 30
   </script>
</body>
</html>
```

Writing Output to the Browser Window

You can use the document.write() method to write the content to the current document only while that document is being parsed.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>Writing into an Browser Window with JavaScript</title>
</head>
<body>
   <script>
   // Printing a simple text message
   document.write("Hello World!"); // Prints: Hello World!

   // Printing a variable value
   var x = 10;
   var y = 20;
   var sum = x + y;
   document.write(sum); // Prints: 30
   </script>
</body>
</html>
```

Inserting Output Inside an HTML Element

You can also write or insert output inside an HTML element using the element's innerHTML property. However, before writing the output first we need to select the element using a method such as getElementById()

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="utf-8">
  <title>Writing into an HTML Element with JavaScript</title>
</head>
<body>
  <p id="greet"></p>
  <p id="result"></p>

  <script>
  // Writing text string inside an element
  document.getElementById("greet").innerHTML = "Hello World!";

  // Writing a variable value inside an element
  var x = 10;
  var y = 20;
  var sum = x + y;
  document.getElementById("result").innerHTML = sum;
  </script>
</body>
</html>
```

JavaScript Data Types

Data Types in JavaScript

Data types basically specify what kind of data can be stored and manipulated within a program.

There are six basic data types in JavaScript which can be divided into three main categories: primitive (or primary), composite (or reference), and special data types. String, Number, and Boolean are primitive data types. Object, Array, and Function (which are all types of objects) are composite data types. Whereas Undefined and Null are special data types.

Primitive data types can hold only one value at a time, whereas composite data types can hold collections of values and more complex entities.

The String Data Type

The string data type is used to represent textual data (i.e. sequences of characters). Strings are created using single or double quotes surrounding one or more characters

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>JavaScript String Data Type</title>
</head>
<body>
   <script>
   // Creating variables
   var a = 'Hi there!';  // using single quotes
   var b = "Hi there!";  // using double quotes

   // Printing variable values
   document.write(a + "<br>");
   document.write(b);
   </script>
</body>
</html>
```

Example 2

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>Including Quotes inside the JavaScript String</title>
</head>
<body>
   <script>
```

```
    // Creating variables
    var a = "Let's have a cup of coffee.";
    var b = 'He said "Hello" and left.';
    var c = 'We\'ll never give up.';

    // Printing variable values
    document.write(a + "<br>");
    document.write(b + "<br>");
    document.write(c);
    </script>
</body>
</html>
```

The Number Data Type

The number data type is used to represent positive or negative numbers with or without decimal place, or numbers written using exponential notation e.g. 1.5e-4 (equivalent to 1.5x10-4).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Number Data Type</title>
</head>
<body>
    <script>
    // Creating variables
    var a = 25;
    var b = 80.5;
    var c = 4.25e+6;
    var d = 4.25e-6;

    // Printing variable values
    document.write(a + "<br>");
    document.write(b + "<br>");
    document.write(c + "<br>");
    document.write(d);
    </script>
</body>
</html>
```

The Number data type also includes some special values which are: Infinity, -Infinity and NaN. Infinity represents the mathematical Infinity ∞, which is greater than any number. Infinity is the result of dividing a nonzero number by 0,
Example 2

```
    <!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Infinity</title>
</head>
<body>
  <script>
  document.write(16 / 0);
  document.write("<br>");
  document.write(-16 / 0);
  document.write("<br>");
  document.write(16 / -0);
  </script>
</body>
</html>
```

While NaN represents a special Not-a-Number value. It is a result of an invalid or an undefined mathematical operation, like taking the square root of -1 or dividing 0 by 0,
Example 3

```
      <!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript NaN</title>
</head>
<body>
  <script>
  document.write("Some text" / 2);
  document.write("<br>");
  document.write("Some text" / 2 + 10);
  document.write("<br>");
  document.write(Math.sqrt(-1));
  </script>
</body>
</html>
```

The Boolean Data Type

The Boolean data type can hold only two values: true or false. It is typically used to store values like yes (true) or no (false), on (true) or off (false)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
```

```
    <title>JavaScript Boolean Data Type</title>
</head>
<body>
    <script>
    // Creating variables
    var isReading = true;   // yes, I'm reading
    var isSleeping = false; // no, I'm not sleeping

    // Printing variable values
    document.write(isReading + "<br>");
    document.write(isSleeping);
    </script>
</body>
</html>
```

Example 2

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Comparisons</title>
</head>
<body>
    <script>
    var a = 2, b = 5, c = 10;

    document.write(b > a) // Output: true
    document.write("<br>");
    document.write(b > c) // Output: false
    </script>
</body>
</html>
```

The Undefined Data Type

The undefined data type can only have one value-the special value undefined. If a variable has been declared, but has not been assigned a value, has the value undefined.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Undefined Data Type</title>
</head>
<body>
    <script>
    // Creating variables
    var a;
    var b = "Hello World!"

    // Printing variable values
    document.write(a + "<br>");
    document.write(b);
    </script>
</body>
</html>
```

The Null Data Type

This is another special data type that can have only one value-the null value. A null value means that there is no value. It is not equivalent to an empty string ("") or 0, it is simply nothing.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Null Data Type</title>
</head>
<body>
    <script>
    var a = null;
    document.write(a + "<br>"); // Print: null

    var b = "Hello World!"
    document.write(b + "<br>"); // Print: Hello World!

    b = null;
    document.write(b) // Print: null
    </script>
</body>
</html>
```

The Object Data Type

The object is a complex data type that allows you to store collections of data.

An object contains properties, defined as a key-value pair. A property key (name) is always a string, but the value can be any data type, like strings, numbers, booleans, or complex data types like arrays, function and other objects.

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
   <meta charset="utf-8">
   <title>JavaScript Object Data Type</title>
</head>
<body>
   <script>
   var emptyObject = {};
   var person = {"name": "Clark", "surname": "Kent", "age": "36"};

   // For better reading
   var car = {
      "modal": "BMW X3",
      "color": "white",
      "doors": 5
   }

   // Print variables values in browser's console
   console.log(person);
   console.log(car);
   </script>
   <p><strong>Note:</strong> Check out the browser console by pressing the f12 key on the
keyboard.</p>
</body>
</html>
```

The Array Data Type

An array is a type of object used for storing multiple values in single variable. Each value (also called an element) in an array has a numeric position, known as its index, and it may contain data of any data type-numbers, strings, booleans, functions, objects, and even other arrays. The array index starts from 0, so that the first array element is arr[0] not arr[1].

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>JavaScript Array Data Type</title>
</head>
<body>
   <script>
   // Creating arrays
   var colors = ["Red", "Yellow", "Green", "Orange"];
   var cities = ["London", "Paris", "New York"];

   // Printing array values
   document.write(colors[0] + "<br>");   // Output: Red
```

```
    document.write(cities[2]);   // Output: New York
    </script>
</body>
</html>
```

The Function Data Type

The function is callable object that executes a block of code. Since functions are objects, so it is possible to assign them to variables

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Function Data Type</title>
</head>
<body>
    <script>
    var greeting = function(){
        return "Hello World!";
    }

    // Check the type of greeting variable
    document.write(typeof greeting) // Output: function
    document.write("<br>");
    document.write(greeting());     // Output: Hello World!
    </script>
</body>
</html>
```

Example 2

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Function Passed as Argument to Other Function</title>
</head>
<body>
    <script>
    function createGreeting(name){
        return "Hello, " + name;
    }
    function displayGreeting(greetingFunction, userName){
        return greetingFunction(userName);
    }
    var result = displayGreeting(createGreeting, "Vijay");
    document.write(result); // Output: Hello, Vijay
    </script>
</body>
</html>
```

The typeof Operator

The typeof operator can be used to find out what type of data a variable or operand contains. It can be used with or without parentheses (typeof(x) or typeof x).

The typeof operator is particularly useful in the situations when you need to process the values of different types differently, but you need to be very careful, because it may produce unexpected result in some cases

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript typeof Operator</title>
</head>
<body>
    <script>
    // Numbers
    document.write(typeof 15 + "<br>");  // Prints: "number"
    document.write(typeof 42.7 + "<br>");  // Prints: "number"
    document.write(typeof 2.5e-4 + "<br>");  // Prints: "number"
    document.write(typeof Infinity + "<br>");  // Prints: "number"
    document.write(typeof NaN + "<br>");  // Prints: "number". Despite being "Not-A-Number"

    // Strings
    document.write(typeof '' + "<br>");  // Prints: "string"
    document.write(typeof 'hello' + "<br>");  // Prints: "string"
    document.write(typeof '12' + "<br>");  // Prints: "string". Number within quotes is
document.write(typeof string

    // Booleans
    document.write(typeof true + "<br>");  // Prints: "boolean"
    document.write(typeof false + "<br>");  // Prints: "boolean"

    // Undefined
    document.write(typeof undefined + "<br>");  // Prints: "undefined"
    document.write(typeof undeclaredVariable + "<br>"); // Prints: "undefined"

    // Null
    document.write(typeof Null + "<br>");  // Prints: "object"
    // Objects
    document.write(typeof {name: "John", age: 18} + "<br>");  // Prints: "object"

    // Arrays
    document.write(typeof [1, 2, 4] + "<br>");  // Prints: "object"

    // Functions
    document.write(typeof function(){});  // Prints: "function"
    </script>
</body>
</html>
```

JavaScript Operators

What are Operators in JavaScript

Operators are symbols or keywords that tell the JavaScript engine to perform some sort of actions. For example, the addition (+) symbol is an operator that tells JavaScript engine to add two variables or values, while the equal-to (==), greater-than (>) or less-than (<) symbols are the operators that tells JavaScript engine to compare two variables or values

JavaScript Arithmetic Operators

The arithmetic operators are used to perform common arithmetical operations, such as addition, subtraction, multiplication etc

Operator	Description	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y.
*	Multiplication	$x * $y	Product of $x and $y.
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Arithmetic Operators</title>
</head>
<body>
  <script>
  var x = 10;
  var y = 4;
  document.write(x + y); // Prints: 14
  document.write("<br>");
  document.write(x - y); // Prints: 6
  document.write("<br>");
  document.write(x * y); // Prints: 40
  document.write("<br>");
  document.write(x / y); // Prints: 2.5
  document.write("<br>");
  document.write(x % y); // Prints: 2
  </script>
</body>
</html>
```

JavaScript Assignment Operators

The assignment operators are used to assign values to variables.

Operator	Description	Example	Is The Same As
=	Assign	x = y	x = y
+=	Add and assign	x += $	x = x + y
-=	Subtract and assign	x -= y	x = x - y
*=	Multiply and assign	x *= y	x = x * y
/=	Divide and assign quotient	x /= y	x = x / y
%=	Divide and assign modulus	x %= y	x = x % y

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Assignment Operators</title>
</head>
<body>
    <script>
    var x;    // Declaring Variable

    x = 10;
    document.write(x + "<br>"); // Prints: 10

    x = 20;
    x += 30;
    document.write(x + "<br>"); // Prints: 50

    x = 50;
    x -= 20;
    document.write(x + "<br>"); // Prints: 30

    x = 5;
    x *= 25;
    document.write(x + "<br>"); // Prints: 125

    x = 50;
    x /= 10;
    document.write(x + "<br>"); // Prints: 5
    x = 100;
    x %= 15;
    document.write(x); // Prints: 10
    </script>
</body>
</html>
```

JavaScript String Operators

There are two operators which can also used be for strings.

Operator	Description	Example	Result
+	Concatenation	str1 + str2	Concatenation of str1 and str2
+=	Concatenation assignment	str1 += str2	Appends the str2 to the str1

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript String Operators</title>
</head>
<body>
    <script>
    var str1 = "Hello";
    var str2 = " World!";

    document.write(str1 + str2 + "<br>"); // Outputs: Hello World!

    str1 += str2;
    document.write(str1); // Outputs: Hello World!
    </script>
</body>
</html>
```

JavaScript Incrementing and Decrementing Operators

The increment/decrement operators are used to increment/decrement a variable's value.

Operator	Name	Effect
++x	Pre-increment	Increments x by one, then returns x
x++	Post-increment	Returns x, then increments x by one
--x	Pre-decrement	Decrements x by one, then returns x
x--	Post-decrement	Returns x, then decrements x by one

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>JavaScript Incrementing and Decrementing Operators</title>
</head>
<body>
   <script>
   var x; // Declaring Variable

   x = 10;
   document.write(++x);
   document.write("<p>" + x + "</p>");

   x = 10;
   document.write(x++);
   document.write("<p>" + x + "</p>");
   x = 10;
   document.write(--x);
   document.write("<p>" + x + "</p>");
   x = 10;
   document.write(x--);
   document.write("<p>" + x + "</p>");
   </script>
</body>
</html>
```

JavaScript Logical Operators

The logical operators are typically used to combine conditional statements.

Operator	Name	Example	Result
&&	And	x && y	True if both x and y are true
||	Or	x || y	True if either x or y is true
!	Not	!x	True if x is not true

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>JavaScript Logical Operators</title>
</head>
<body>
   <script>
   var year = 2020;
```

```
    // Leap years are divisible by 400 or by 4 but not 100
    if((year % 400 == 0) || ((year % 100 != 0) && (year % 4 == 0))){
        document.write(year + " is a leap year.");
    } else{
        document.write(year + " is not a leap year.");
    }
    </script>
</body>
</html>
```

JavaScript Comparison Operators

The comparison operators are used to compare two values in a Boolean fashion.

Operator	Name	Example	Result
==	Equal	x == y	True if x is equal to y
===	Identical	x === y	True if x is equal to y, and they are of the same type
!=	Not equal	x != y	True if x is not equal to y
!==	Not identical	x !== y	True if x is not equal to y, or they are not of the same type
<	Less than	x < y	True if x is less than y
>	Greater than	x > y	True if x is greater than y
>=	Greater than or equal to	x >= y	True if x is greater than or equal to y
<=	Less than or equal to	x <= y	True if x is less than or equal to y

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Comparison Operators</title>
</head>
<body>
    <script>
    var x = 25;
    var y = 35;
    var z = "25";

    document.write(x == z);  // Prints: true
    document.write("<br>");
    document.write(x === z); // Prints: false
    document.write("<br>");

    document.write(x != y);  // Prints: true
    document.write("<br>");

    document.write(x !== z); // Prints: true
    document.write("<br>");

    document.write(x < y);   // Prints: true
    document.write("<br>");

    document.write(x > y);   // Prints: false
    document.write("<br>");

    document.write(x <= y);  // Prints: true
    document.write("<br>");

    document.write(x >= y);  // Prints: false
    </script>
</body>
</html>
```

JavaScript Events

An event is something that happens when user interact with the web page, such as when he clicked a link or button, entered text into an input box or textarea, made selection in a select box, pressed key on the keyboard, moved the mouse pointer, submits a form, etc. In some cases, the Browser itself can trigger the events, such as the page load and unload events.

When an event occur, you can use a JavaScript event handler (or an event listener) to detect them and perform specific task or set of tasks. By convention, the names for event handlers always begin with the word "on", so an event handler for the click event is called onclick, similarly an event handler for the load event is called onload, event handler for the blur event is called onblur, and so on.

There are several ways to assign an event handler. The simplest way is to add them directly to the start tag of the HTML elements using the special event-handler attributes

Mouse Events

The Click Event (onclick)

The click event occurs when a user clicks on an element on a web page. Often, these are form elements and links. You can handle a click event with an onclick event handler.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Handling the Click Event</title>
</head>
<body>
  <button type="button" onclick="alert('You have clicked a button!');">Click Me</button>
  <a href="#" onclick="alert('You have clicked a link!');">Click Me</a>
</body>
</html>
```

The Contextmenu Event (oncontextmenu)

The contextmenu event occurs when a user clicks the right mouse button on an element to open a context menu. You can handle a contextmenu event with an oncontextmenu event handler.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Handling the Contextmenu Event</title>
</head>
<body>
```

```
  <button type="button" oncontextmenu="alert('You have right-clicked a button!');">Right Click on Me</button>
  <a href="#" oncontextmenu="alert('You have right-clicked a link!');">Right Click on Me</a>
</body>
</html>
```

The Mouseover Event (onmouseover)

The mouseover event occurs when a user moves the mouse pointer over an element.
You can handle the mouseover event with the onmouseover event handler.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Handling the Mouseover Event</title>
</head>
<body>
  <button type="button" onmouseover="alert('You have placed mouse pointer over a button!');">Place Mouse Over Me</button>
  <a href="#" onmouseover="alert('You have placed mouse pointer over a link!');">Place Mouse Over Me</a>
</body>
</html>
```

The Mouseout Event (onmouseout)

The mouseout event occurs when a user moves the mouse pointer outside of an element.
You can handle the mouseout event with the onmouseout event handler.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Handling the Mouseout Event</title>
</head>
<body>
  <button type="button" onmouseout="alert('You have moved out of the button!');">Place Mouse Inside Me and Move Out</button>
  <a href="#" onmouseout="alert('You have moved out of the link!');">Place Mouse Inside Me and Move Out</a>
</body>
</html>
```

Keyboard Events

A keyboard event is fired when the user press or release a key on the keyboard. Here're some most important keyboard events and their event handler.

The Keydown Event (onkeydown)

The keydown event occurs when the user presses down a key on the keyboard.
You can handle the keydown event with the onkeydown event handler.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>JavaScript Handling the Keydown Event</title>
</head>
<body>
   <input type="text" onkeydown="alert('You have pressed a key inside text input!')">
   <hr>
   <textarea cols="30" onkeydown="alert('You have pressed a key inside textarea!')"></textarea>
           <p><strong>Note:</strong> Try to enter some text inside input box and textarea.</p>
</body>
</html>
```

The Keyup Event (onkeyup)

The keyup event occurs when the user releases a key on the keyboard.
You can handle the keyup event with the onkeyup event handler.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>JavaScript Handling the Keyup Event</title>
</head>
<body>
   <input type="text" onkeyup="alert('You have released a key inside text input!')">
   <hr>
   <textarea cols="30" onkeyup="alert('You have released a key inside textarea!')"></textarea>
           <p><strong>Note:</strong> Try to enter some text inside input box and textarea.</p>
</body>
</html>
```

The Keypress Event (onkeypress)

The keypress event occurs when a user presses down a key on the keyboard that has a character value associated with it. For example, keys like Ctrl, Shift, Alt, Esc, Arrow keys, etc. will not generate a keypress event, but will generate a keydown and keyup event.

You can handle the keypress event with the onkeypress event handler.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Handling the Keypress Event</title>
</head>
<body>
  <input type="text" onkeypress="alert('You have pressed a key inside text input!')">
  <hr>
  <textarea cols="30" onkeypress="alert('You have pressed a key inside textarea!')"></textarea>
  <p><strong>Note:</strong> Try to enter some text inside input box and textarea.</p>
</body>
</html>
```

Form Events

A form event is fired when a form control receive or loses focus or when the user modify a form control value such as by typing text in a text input, select any option in a select box etc. Here're some most important form events and their event handler.

The Focus Event (onfocus)

The focus event occurs when the user gives focus to an element on a web page.
You can handle the focus event with the onfocus event handler.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Handling the Focus Event</title>
</head>
<body>
  <script>
    function highlightInput(elm){
      elm.style.background = "yellow";
    }
  </script>
  <input type="text" onfocus="highlightInput(this)">
  <button type="button">Button</button>
</body>
</html>
```

The Blur Event (onblur)

The blur event occurs when the user takes the focus away from a form element or a window.
You can handle the blur event with the onblur event handler.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Handling the Blur Event</title>
</head>
<body>
  <input type="text" onblur="alert('Text input loses focus!')">
  <button type="button">Submit</button>
        <p><strong>Note:</strong> First click inside the text input box then click outside to see how
it works.</p>
</body>
</html>
```

The Change Event (onchange)

The change event occurs when a user changes the value of a form element.
You can handle the change event with the onchange event handler.

```
 <!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Handling the Change Event</title>
</head>
<body>
  <select onchange="alert('You have changed the selection!');">
    <option>Select</option>
    <option>Male</option>
    <option>Female</option>
  </select>
        <p><strong>Note:</strong> Select any option in select box to see how it works.</p>
</body>
</html>
```

The Submit Event (onsubmit)

The submit event only occurs when the user submits a form on a web page.
You can handle the submit event with the onsubmit event handler.

```
 <!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>JavaScript Handling the Submit Event</title>
</head>
<body>
   <form action="action.php" method="post" onsubmit="alert('Form data will be submitted to the
server!');">
       <label>First Name:</label>
       <input type="text" name="first-name" required>
       <input type="submit" value="Submit">
   </form>
</body>
</html>
```

Document/Window Events

Events are also triggered in situations when the page has loaded or when user resize the browser window, etc. Here're some most important document/window events and their event handler.

The Load Event (onload)

The load event occurs when a web page has finished loading in the web browser.
You can handle the load event with the onload event handler.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>JavaScript Handling the Load Event</title>
</head>
<body onload="window.alert('Page is loaded successfully!');">
   <h1>This is a heading</h1>
   <p>This is paragraph of text.</p>
</body>
</html>
```

The Resize Event (onresize)

The resize event occurs when a user resizes the browser window. The resize event also occurs in situations when the browser window is minimized or maximized.
You can handle the resize event with the onresize event handler.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Handling the Resize Event</title>
</head>
<body>
  <p id="result"></p>
  <script>
    function displayWindowSize(){
      var w = window.outerWidth;
      var h = window.outerHeight;
      var txt = "Window size: width=" + w + ", height=" + h;
      document.getElementById("result").innerHTML = txt;
    }
    window.onresize = displayWindowSize;
  </script>
  <p><strong>Note:</strong> Resize the browser window to see how the resize event works.</p>
</body>
</html>
```

JavaScript Strings

In this tutorial you will learn how to create and manipulate strings in JavaScript.

What is String in JavaScript

A string is a sequence of letters, numbers, special characters and arithmetic values or combination of all. Strings can be created by enclosing the string literal (i.e. string characters) either within single quotes (') or double quotes (")

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>Escaping Quotes inside JavaScript Strings</title>
</head>
<body>
   <script>
   // Creating variables
   var str1 = 'it\'s okay';
   var str2 = "He said \"Goodbye\"";
   var str3 = 'She replied \'Calm down, please\'';

   // Printing variable values
   document.write(str1 + "<br>");
   document.write(str2 + "<br>");
   document.write(str3);
   </script>
</body>
</html>
```

Getting the Length of a String

The length property returns the length of the string, which is the number of characters contained in the string. This includes the number of special characters as well, such as \t or \n.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>Get String Length in JavaScript</title>
</head>
<body>
   <script>
   var str1 = "This is a paragraph of text.";
   document.write(str1.length + "<br>");
   var str2 = "This is a \n paragraph of text.";
   document.write(str2.length);
</script>
```

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

Finding a String Inside Another String

You can use the indexOf() method to find a substring or string within another string. This method returns the index or position of the first occurrence of a specified string within a string.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>JavaScript Find the Position of Substring within a String</title>
</head>
<body>
   <script>
   var str = "If the facts don't fit the theory, change the facts.";
   var pos = str.indexOf("facts");
   document.write(pos); // 0utputs: 7
   </script>
</body>
</html>
```

Similarly, you can use the lastIndexOf() method to get the index or position of the last occurrence of the specified string within a string, like this:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>JavaScript Find the Position of Substring within a String</title>
</head>
<body>
   <script>
   var str = "If the facts don't fit the theory, change the facts.";
   var pos = str.lastIndexOf("facts");
   document.write(pos); // 0utputs: 46
   </script>
</body>
</html>
```

Both the indexOf(), and the lastIndexOf() methods return -1 if the substring is not found. Both methods also accept an optional integer parameter which specifies the position within the string at which to start the search.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Find the Position of Substring within a String</title>
</head>
<body>
  <script>
  var str = "If the facts don't fit the theory, change the facts.";

  // Searching forwards
  var pos1 = str.indexOf("facts", 20);
  document.write(pos1 + "<br>"); // 0utputs: 46

  // Searching backwards
  var pos2 = str.lastIndexOf("facts", 20);
  document.write(pos2); // 0utputs: 7
  </script>
</body>
</html>
```

Searching for a Pattern Inside a String

You can use the search() method to search a particular piece of text or pattern inside a string.

Like indexOf() method the search() method also returns the index of the first match, and returns -1 if no matches were found, but unlike indexOf() method this method can also take a regular expression as its argument to provide advanced search capabilities.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Search Text or Pattern inside a String</title>
</head>
<body>
  <script>
  var str = "Color red looks brighter than color blue.";

  // Case sensitive search
  var pos1 = str.search("color");
  document.write(pos1 + "<br>"); // 0utputs: 30

  // Case insensitive search using regexp
  var pos2 = str.search(/color/i);
```

```
   document.write(pos2); // 0utputs: 0
   </script>
</body>
</html>
```

Extracting a Substring from a String

You can use the slice() method to extract a part or substring from a string.

This method takes 2 parameters: start index (index at which to begin extraction), and an optional end index (index before which to end extraction), like str.slice(startIndex, endIndex).

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>JavaScript Slice Out a Portion of a String</title>
</head>
<body>
   <script>
   var str = "The quick brown fox jumps over the lazy dog.";
   var subStr = str.slice(4, 15);
   document.write(subStr); // Prints: quick brown
   </script>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>JavaScript Slice Strings Using Negative Indexes</title>
</head>
<body>
   <script>
   var str = "The quick brown fox jumps over the lazy dog.";
      document.write(str.length);
   document.write(str.slice(-28, -19) + "<br>"); // Prints: fox jumps
   document.write(str.slice(31)); // Prints: the lazy dog.
   </script>
</body>
</html>
```

You can also use the substring() method to extract a section of the given string based on start and end indexes, like str.substring(startIndex, endIndex). The substring() method is very similar to the slice() method, except few differences:

- If either argument is less than 0 or is NaN, it is treated as 0.
- If either argument is greater than str.length, it is treated as if it were str.length.
- If startIndex is greater than endIndex, then substring() will swap those two arguments; for example, str.substring(5, 0) == str.substring(0, 5).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Extract substring from a String</title>
</head>
<body>
  <script>
  var str = "The quick brown fox jumps over the lazy dog.";
  document.write(str.substring(4, 15) + "<br>"); // Prints: quick brown
  document.write(str.substring(9, 0) + "<br>"); // Prints: The quick
  document.write(str.substring(-28, -19) + "<br>"); // Prints nothing
  document.write(str.substring(31)); // Prints: the lazy dog.
  </script>
</body>
</html>
```

Extracting a Fixed Number of Characters from a String

JavaScript also provide the substr() method which is similar to slice() with a subtle difference, the second parameter specifies the number of characters to extract instead of ending index, like str.substr(startIndex, length). If length is 0 or a negative number, an empty string is returned.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Extract Fixed Number of Characters from a String</title>
</head>
<body>
  <script>
  var str = "The quick brown fox jumps over the lazy dog.";
  document.write(str.substr(4, 15) + "<br>"); // Prints: quick brown fox
  document.write(str.substr(-28, -19) + "<br>"); // Prints nothing
  document.write(str.substr(-28, 9) + "<br>"); // Prints: fox jumps
  document.write(str.substr(31)); // Prints: the lazy dog.
```

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

Replacing the Contents of a String

You can use the replace() method to replace part of a string with another string. This method takes two parameters a regular expression to match or substring to be replaced and a replacement string, like str.replace(regexp|substr, newSubstr).

This replace() method returns a new string, it doesn't affect the original string that will remain unchanged.

```
    <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Replace Part of a String with another String</title>
</head>
<body>
    <script>
    var str = "Color red looks brighter than color blue.";
    var result = str.replace("color", "paint");
    document.write(result); // 0utputs: Color red looks brighter than paint blue.
    </script>
</body>
</html>
```

Example 2

```
    <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Replace Part of a String with another String</title>
</head>
<body>
    <script>
    var str = "Color red looks brighter than color blue.";
    var result = str.replace(/color/i, "paint");
    document.write(result); // 0utputs: paint red looks brighter than color blue.
    </script>
</body>
</html>
```

Example 3

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Replace All Occurrences of a Substring in a String</title>
</head>
<body>
  <script>
  var str = "Color red looks brighter than color blue.";
  var result = str.replace(/color/ig, "paint");
  document.write(result); // 0utputs: paint red looks brighter than paint blue.
  </script>
</body>
</html>
```

Converting a String to Uppercase or Lowercase

You can use the toUpperCase() method to convert a string to uppercase

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Convert a String to Uppercase Characters</title>
</head>
<body>
  <script>
  var str = "Hello World!";
  var result = str.toUpperCase();
  document.write(result); // Prints: HELLO WORLD!
  </script>
</body>
</html>
```

toLowerCase() Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Convert a String to Lowercase Characters</title>
</head>
<body>
  <script>
```

```
    var str = "Hello World!";
    var result = str.toLowerCase();
    document.write(result); // Prints: hello world!
    </script>
</body>
</html>
```

Concatenating Two or More Strings

You can concatenate or combine two or more strings using the + and += assignment operators.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Join Two or More Strings</title>
</head>
<body>
    <script>
    var hello = "Hello";
    var world = "World";
    var greet = hello + " " + world;
    document.write(greet + "<br>"); // Prints: Hello World

    var wish  = "Happy";
        wish += " Birthday";
    document.write(wish); // Prints: Happy Birthday
    </script>
</body>
</html>
```

Accessing Individual Characters from a String

You can use the charAt() method to access individual character from a string, like str.charAt(index). The index specified should be an integer between 0 and str.length - 1. If no index is provided the first character in the string is returned, since the default is 0.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Extract a Single Character from a String</title>
</head>
<body>
    <script>
    var str = "Hello World!";
    document.write(str.charAt() + "<br>");  // Prints: H
    document.write(str.charAt(6) + "<br>"); // Prints: W
    document.write(str.charAt(30) + "<br>"); // Prints nothing
    document.write(str.charAt(str.length - 1)); // Prints: !
    </script>
</body>
</html>
```

Splitting a String into an Array

The split() method can be used to splits a string into an array of strings, using the syntax str.split(separator, limit). The seperator argument specifies the string at which each split should occur, whereas the limit arguments specifies the maximum length of the array.

If separator argument is omitted or not found in the specified string, the entire string is assigned to the first element of the array.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Split a String into an Array</title>
</head>
<body>
    <script>
    var fruitsStr = "Apple, Banana, Mango, Orange, Papaya";
    var fruitsArr = fruitsStr.split(", ");
    document.write(fruitsArr[0] + "<br>"); // Prints: Apple
    document.write(fruitsArr[2] + "<br>"); // Prints: Mango
    document.write(fruitsArr[fruitsArr.length - 1]); // Prints: Papaya
    document.write("<hr>");

    // Loop through all the elements of the fruits array
    for(var i in fruitsArr) {
        document.write("<p>" + fruitsArr[i] + "</p>");
    }
    </script>
</body>
</html>
```

Example 2

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="utf-8">
    <title>JavaScript Split a String Into an Array of Characters</title>
</head>
<body>
    <script>
    var str = "INTERSTELLAR";
    var strArr = str.split("");
    document.write(strArr[0] + "<br>"); // Prints: I
    document.write(strArr[1] + "<br>"); // Prints: N
    document.write(strArr[strArr.length - 1]); // Prints: R
    document.write("<hr>"); // Prints: N

    // Loop through all the elements of the characters array and print them
    for(var i in strArr) {
        document.write("<br>" + strArr[i]);
    }
    </script>
</body>
</html>
```