

MODULE-3

Introduction:

- JavaScript was designed to add interactivity to HTML pages.
- JavaScript is a scripting language (a scripting language is a lightweight programming language).
- A JavaScript consists of lines of executable computer code.
- A JavaScript is usually embedded directly into HTML pages.
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation).
- Everyone can use JavaScript without purchasing a license.

A simple syntax of your JavaScript will appear as follows.

```
<script language="javascript" type="text/javascript">  
JavaScript code  
</script>
```

Event Handling and Document Object model in JavaScript

An event handler is a bit of JavaScript code that allows us to capture each event as it happens, and respond to it by running some JavaScript code.

Event handling in the DOM (Document Object Model) is the process of detecting and responding to user interactions or system events on a web page. Events can be triggered by a variety of actions, such as clicking a button, submitting a form, scrolling the page, or resizing the window. Event handling allows web developers to create dynamic and interactive user interfaces that respond to user input in real-time. In the DOM, events are represented as objects, and event handling is implemented through event listeners.

Event Listeners

Event listeners are the foundation of event handling in the DOM. An event listener is a function that waits for a specific event to occur on an HTML element and executes a set of instructions when the event is triggered. The event listener is attached to the HTML element using the **addEventListener()** method, which takes two arguments:

1. The name of the event to listen for
2. The function to be executed when the event is triggered.

Example:

```
const button = document.querySelector('button');
button.addEventListener('click', () => {
  alert('Button clicked!');
});
```

Types of Events

There are many types of events that can be handled in the DOM, including:

1. **Mouse events:** click, dblclick, mouseover, mouseout, mousemove, mousedown, mouseup.
2. **Keyboard events:** keydown, keyup, keypress.
3. **Form events:** submit, reset, change, focus, blur.
4. **Window events:** load, unload, resize, scroll.
5. **Touch events:** touchstart, touchend, touchmove.

1. Mouse Events

JavaScript mouse events allow users to control and interact with web pages using their mouse. These events trigger specific functions or actions in response to user clicks, scrolls, drags, and other mouse movements.

To handle mouse events in JavaScript, you can use the **addEventListener()** method. The **addEventListener()** method takes two arguments: the event type and the event handler function. The event type is the name of the event that you want to handle, and the event handler function is the function that will be called when the event occurs.

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.

Ex:

```
const button = document.querySelector('button');
button.addEventListener('click', () => {
  alert('Button clicked!');
});
```

1. onclick event

The onclick event generally occurs when the user clicks on an element. It allows the programmer to execute a JavaScript's function when an element gets clicked.

Syntax

```
object.onclick = function() { myScript };
<script>
function fun() {
  alert("Welcome to the Web Technology");
}
</script>
<button onclick = "fun()">Click me</button>
```

2. Keyboard Events

The keyboard events in JavaScript provide a way to interact with a web page or application based on the user's keyboard input. These events allow developers to capture and respond to various keyboard actions, such as key presses, key releases, and character inputs. The primary keyboard events in JavaScript include keydown, keypress, and keyup.

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

1. Keydown Event – When a key on the keyboard is pressed down, it triggers the keydown event. This event equips developers with information about the specific key that was pressed:

2. Keypress Event – The keypress event triggers when a user types an actual character. Non-character keys, such as Shift or Ctrl, do not activate this event.

3. Keyup Event – Upon the release of a previously pressed key, the system initiates the firing of a keyup event.

Ex:

```
document.addEventListener('keypress', (event) => {
  console.log(`You pressed the ${event.key} key.`);
});
```

Form Events

Form events are used to handle interactions with HTML form elements. When the form is submitted, we prevent the default behaviour (which is to reload the page), and display the value of the input field in the console.

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

Ex:

```
const form = document.querySelector('form');
form.addEventListener('submit', (event) => {
  event.preventDefault();
  const input = document.querySelector('input');
  console.log(`You entered: ${input.value}`);
});
```

Window Events

Window events are used to handle interactions with the browser window. When the window is resized, we display the new width and height of the window in the console.

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
Reset	onreset	When the window size is resized
Scroll	onscroll	When the visitor scrolls a scrollable area

Ex:

```
window.addEventListener('resize', () => {
  console.log(`Window size changed to ${window.innerWidth}x${window.innerHeight}`);
});
```

Touch Events

Touch events are used to handle interactions with touchscreens on mobile devices. When the user touches the box, we display the coordinates of the touch in the console.

Ex:

```
const box = document.querySelector('.box');
box.addEventListener('touchstart', (event) => {
  console.log(`Touch started at (${event.touches[0].clientX},${event.touches[0].clientY})`);
});
```

Document Object model in JavaScript

The Document Object Model (DOM) is a programming API for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. With the Document Object Model, programmers can create and build documents, navigate their structure, and add, modify, or delete elements and content.

JavaScript String

The **JavaScript string** is an object that represents a sequence of characters.

There are 2 ways to create string in JavaScript

1. By string literal
2. By string object (using new keyword)

1) By string literal

The string literal is created using double quotes. The syntax of creating string using string literal is given below:

1. `var stringname="string value";`

Syntax:

```
<script>
var str="This is string literal";
document.write(str);
</script>
```

2) By string object (using new keyword)

The syntax of creating string object using new keyword is given below:

1. `var stringname=new String("string literal");`

Here, **new keyword** is used to create instance of string.

```
<script>
var stringname=new String("hello javascript string");
document.write(stringname);
</script>
```

JavaScript String Methods

JavaScript strings are the sequence of characters. They are treated as **Primitive data types**. In JavaScript, strings are automatically converted to string objects when using **string methods** on them.

The following are methods that we can call on strings.

	Methods	Description
1	charAt()	It provides the char value present at the specified index.
2	concat()	It provides a combination of two or more strings.
3	indexOf()	It provides the position of a char value present in the given string.
4	replace()	It replaces a given string with the specified replacement.
5	substr()	It is used to fetch the part of the given string on the basis of the specified starting position and length.

6	slice()	It is used to fetch the part of the given string. It allows us to assign positive as well negative index.
7	toLowerCase()	It converts the given string into lowercase letter.
8	toUpperCase()	It converts the given string into uppercase letter.
9	split()	It splits a string into substring array, then returns that newly created array.
10	trim()	It trims the white space from the left and right side of the string.

1) charAt(index) Method

The JavaScript String **charAt()** method **retrieves the character at a specified index in a string**. The index is passed as an argument to the method, and it returns the character at that position.

Syntax:

```
character = str.charAt(index);
```

Ex:

```
<script>
var str="WEBTECHNOLOGY";
document.write(str.charAt(2));
</script>
```

Output:

B

2) concat(str) Method

The **concat()** method in JavaScript join or concatenate two or more strings together.

Syntax

```
str1.concat(str2, str3, str4, ..., strN)
```

Ex:

```
<script>
var s1="WEB ";
var s2="TECHNOLOGY";
var s3=s1.concat(s2);
document.write(s3);
</script>
```

Output:

WEB TECHNOLOGY

3. indexOf(str) Method

The **indexOf()** method in JavaScript is used to find the index of the first occurrence of a specified value within a string.

Syntax

```
str.indexOf(searchValue , index);
```

Ex:

```
<script>
var s1="javascript from web technology";
var n=s1.indexOf("from");
document.write(n);
</script>
```

Output:

11

4. JavaScript string replace() Method

JavaScript **replace()** method is used for manipulating strings. It allows you to search for a specific part of a string, called a *substring*, and then replace it with another *substring*.

Syntax:

```
str.replace(value1, value2);
```

Ex:

```
<script>
var str="WebTech";
document.writeln(str.replace("WebTech","WT"));
</script>
```

Output:

WT

5. substr() Method

The **substr()** method in JavaScript **extracts a portion of a string**, starting from a specified index position and extending for a given number of characters.

Syntax:

```
str.substr(start , length)
```


Ex:

```
<script>
var str="WebTechnology";
document.writeln(str.substr(0,4));
</script>
```

Output:

WebT

6. slice() Method

The slice() method in JavaScript is used to **extract a portion of a string** and create a new string without modifying the original string.

Syntax:

```
string.slice(startingIndex, endingIndex);
```

Ex:

```
<script>
var str = "WebTechnology";
document.writeln(str.slice(2,5));
</script>
```

Output:

bTe

7. toLowerCase() Method

The **JavaScript toLowerCase() method** converts all characters in a string to lowercase, returning a new string without modifying the original.

Syntax:

```
str.toLowerCase();
```

Ex:

```
<script>
var str = "WEB TECHNOLOGY";
document.writeln(str.toLowerCase());
</script>
```

Output:

Web technology

8. toUpperCase() Method

The JavaScript string toUpperCase() method is used to convert the string into uppercase letter. This method doesn't make any change in the original string.

Syntax:

```
str.toUpperCase()
```

Ex:

```
<script>
var str = "web technology";
document.writeln(str.toUpperCase());
</script>
```

Output:

WEB TECHNOLOGY

9. split() Method

The **split()** method in JavaScript splits the string into the array of substrings, puts these substrings into an array, and returns the new array. It does not change the original string.

Syntax

```
str.split( separator, limit );
```

Ex:

```
<html>
<head>
<script>
var str = 'Welcome to the Web Technology Lab'
var arr = str.split(" ", 3);
document.write(arr);
</script>
</head>
<body>

</body>
</html>
```

Output:

Welcome,to,the

10. trim() Method

The **trim() method** is used to remove the whitespace from both sides of a string.

Syntax

```
str.trim();
```

Ex:

```
<html>
<body>
<script>
function func_trim() {
    //original string with whitespace in beginning and end
    var str = "  Web  Technology  ";

    //string trimmed using trim()
    var trimmedstr = str.trim();
    document.write(trimmedstr);
}
func_trim();
</script>
</html>
</body>
```

Output:

Web Technology

JavaScript - Window Object

The JavaScript **window** object represents the browser's window. In JavaScript, a 'window' object is a global object. It contains the various methods and properties that we can use to access and manipulate the current browser window.

Window is the object of browser; **it is not the object of javascript**. The javascript objects are string, array, date etc.

For example, showing an alert, opening a new window, closing the current window, etc.

Method	Description
window.alert()	displays the alert box containing message with ok button.

--	--

<code>window.confirm()</code>	displays the confirm dialog box containing message with ok and cancel button.
<code>window.prompt()</code>	displays a dialog box to get input from the user.
<code>window.open()</code>	opens the new window.
<code>window.close()</code>	closes the current window.
<code>window.setTimeout()</code>	performs action after specified time like calling function, evaluating expressions etc.

1. `window.alert()` Method

The `window.alert()` method allows you to show the pop-up dialog containing the message, warning, etc. It takes the string text as an argument.

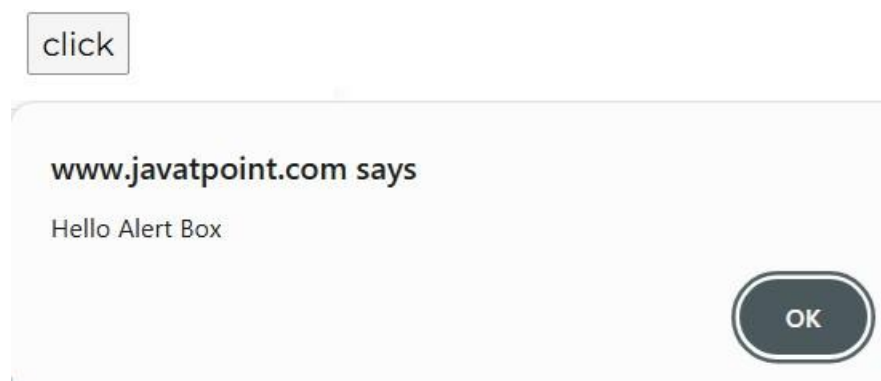
Syntax

```
window.alert(message)
```

Ex:

```
<script type="text/javascript">
function msg(){
window.alert("Hello Alert Box");
}
</script>
<input type="button" value="click" onclick="msg()"/>
```

Output:



2. `confirm()` Method

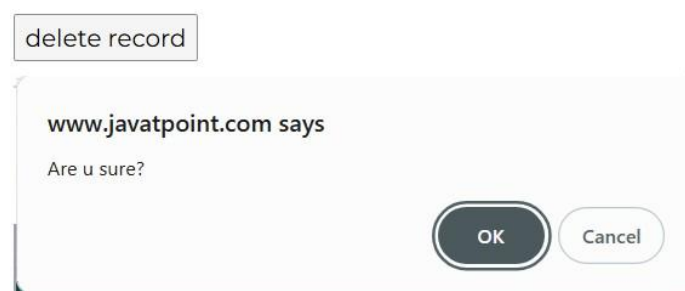
The **`confirm()` method** in JavaScript displays a dialog box with a message and two buttons: OK and Cancel. It is often used to get user confirmation before an action, returning true if OK is clicked, and false if Cancel is clicked.

Syntax

```
window.confirm(message);
```

Ex:

```
<script type="text/javascript">
function msg(){
var v= confirm("Are u sure?");
if(v==true){
window.alert("ok");
}
else{
window.alert("cancel");
}
}
</script>
<input type="button" value="delete record" onclick="msg()" />
```

Output:**3. window.prompt()**

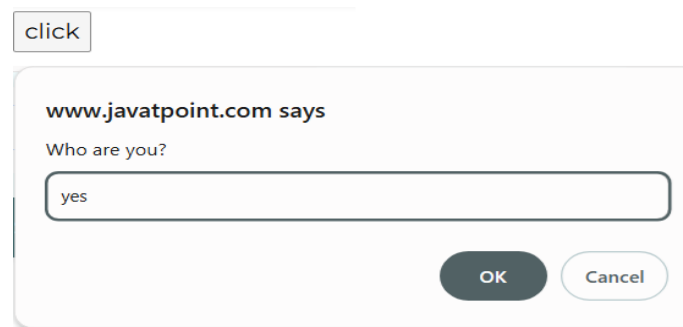
The prompt() method displays a dialog box that prompts the user for input. This method returns the input value if the user clicks "OK", otherwise it returns null.

Syntax

```
window.prompt(text, defaultText)
```

Ex:

```
<script type="text/javascript">
function msg(){
var v= prompt("Who are you?");
window.alert("I am "+v);
}
</script>
<input type="button" value="click" onclick="msg()" />
```

Output:**4. window.open()**

The open() method opens a new browser window, or a new tab, depending on your browser settings and the parameter values.

Syntax

```
window.open(URL, name, specs, replace)
```

Ex:

```
<html>
<body>
<script type="text/javascript">
function msg(){
window.open("http://www.google.com");
}
</script>
<input type="button" value="Google" onclick="msg()" />
</body>
</html>
```

Output:

5. window.close()

The close() method closes a window.

Syntax

```
window.close()
```

6. window.setTimeout()

The setTimeout() method calls a function after a number of milliseconds. 1 second = 1000 millisecond.

Syntax

```
window.setTimeout(function, milliseconds, param1, param2, ...)
```

Ex:

```
<script type="text/javascript">
function msg(){
setTimeout(
function(){
window.alert("Welcome to Javatpoint after 2 seconds")
},2000);
}
</script>
<input type="button" value="click" onclick="msg()" />
```

Output: