# Dynamic HTML: Overview of JavaScript

- JavaScript is a client and server-side object-based scripting language that is used to make interactive Web pages.
- A scripting language is a lightweight programming language with less complexity.
- JavaScript is the most usually used scripting language to add dynamism and interactivity to Web pages.
- This is because JavaScript, written on the client-side, executes on a client browser, thereby reducing the load on the server.

**Why to Learn JavaScript?**

There are the three languages, all web developers must know, these are the following:

- HTML - to define the content of web pages
- CSS - to define the layout of web pages
- JavaScript - to program the behavior of web pages

## Features of JavaScript:

- Light Weight Scripting language
- Dynamic Typing
- Object-oriented programming support
- Functional Style
- Platform Independent
- Prototype-based
- Interpreted Language
- Async Processing
- Client-Side Validation
- More control in the browser

## Using JavaScript in an HTML document:

- The HTML SCRIPT element (<script> tag) is used to define a client-side script (JavaScript).
- The SCRIPT element either contains script statements, or it points to an external script file.
- Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.
- The SCRIPT element contains five attributes as follow

| Attribute | Value | Description |
|---|---|---|
| src | URL | Specifies the address of a script file. |
| type | text/javascript<br>text/ecmascript<br>application/javascript<br>application/ecmascript<br>text/vbscript | Specifies the MIME(multipurpose internet mail extension) type |
| async | true or false | Specifies whether the script should be executes asynchronously or not |
| charset | Charset | Specifies character encoding used in script |
| defer | true or false | Specifies whether browser can continues parsing the web page or not |

- You can use SCRIPT element in three ways
  1. In the HEAD element
  2. In the BODY element
  3. As an external script file

**1. In the HEAD element:**

- You can place SCRIPT element inside the HEAD element of an HTML document.
- The script runs when you perform some action, such as click on the link, click on submit button.
- Syntax:

```
<HEAD>
   <SCRIPT type="text/javascript" >
         Script code here
   </SCRIPT>
</HEAD>
```

**2. In the BODY element:**

- You can place SCRIPT element inside the BODY element of an HTML document.
- The script runs when a web page starts loading in a web browser.
- Syntax:

```
<BODY>
   <SCRIPT type="text/javascript" >
         Script code here
   </SCRIPT>
</BODY>
```

**3. As an external script file:**

- You can store JavaScript code in external file and save that file using the .js extension.
- This external file was linked to HTML document by using scr attribute in SCRIPT element to access the script.
- Syntax:

```
<BODY>
   <SCRIPT src="URL of external file " >
         Script code here
   </SCRIPT>
</BODY>
```

## Exploring Lexical structure of JavaScript:

- Lexical structure of JavaScript provides set of rules to write programs.
- Lexical structure of JavaScript defines rules for following aspect:
  1. Character set
  2. Chase sensitivity
  3. White spaces and line breaks
  4. Optional semicolon
  5. Comments
  6. Literals
  7. Identifiers
  8. Reserved words

## Exploring popup boxes:

- A popup box is a window that displays message along with OK button.
- A popup box may also contain CANCEL button.
- JavaScript supports three types of popup boxes
  1. The alert box
  2. The confirm box
  3. The prompt box

### 1. The alert box:

- The alert box generally used to display an alert message while executing JavaScript.
- The alert box is used to display an error messages after you validate a form.
- The alert box contains OK button, which the user has to click to continue with the execution of the code.
- Syntax: alert (“alert message”);
- Example: alert (“please fill all mandatory fields.”);

### 2. The confirm box:

- The confirm box is used to verify the user activity regarding their operation.
- The confirm box is used to display a messages and return **true** or **false** value.
- The confirm box contains OK and CANCEL buttons.
- If the user clicks on OK button it return true value and if user click on CANCEL button it return false value.
- Syntax: confirm (“confirm message”);
- Example: confirm (“do you want proceed?”);

### 3. The prompt box:

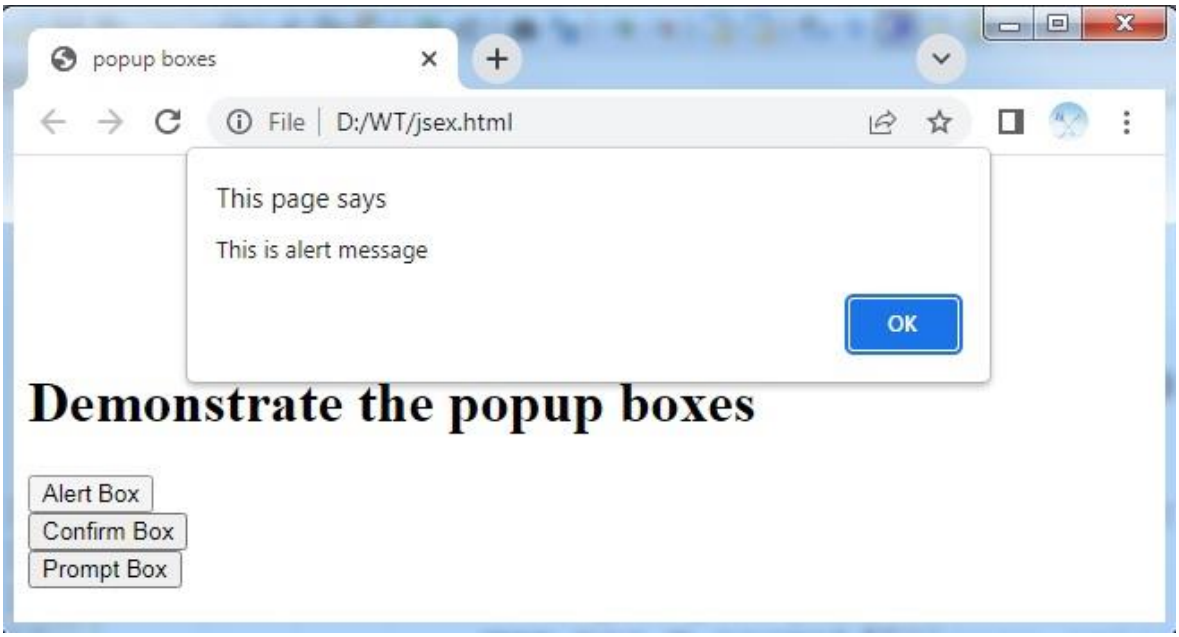
- The confirm box is used to input a value from the user.
- The confirm box contains text box and OK and CANCEL buttons.
- If the user clicks on OK button it return input value otherwise it return null value.

**Example:** Create a HTML file that demonstrates the popup boxes (alert, confirm, and prompt boxes).

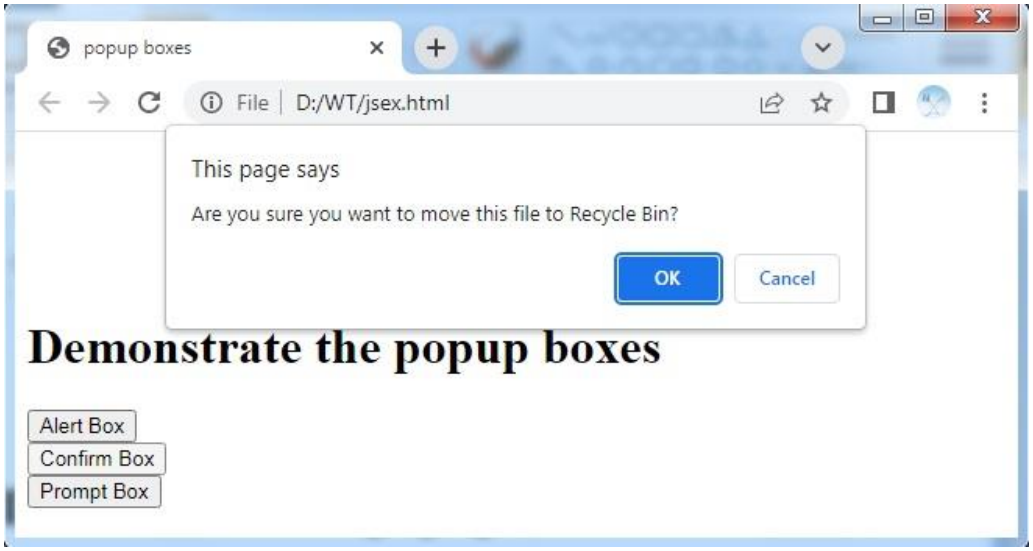
**Source code:**

```
<!DOCTYPE  html>
<html>
	<head>
		<title>popup boxes</title>
		<script type="text/javascript">
			function alertmsg(){
				alert("This is alert message");
			}
			function confirmmsg(){
				confirm("Are you sure you want to move this file to Recycle Bin?");
			}
			function promptmsg(){
				var msg = prompt("Please enter message");
				document.getElementById("pmsg").innerHTML="Your message: "+msg;
			}
		</script>
	</head>
	<body>
		<br><br><br><br><br><br><br><br>
		<h1>Demonstrate the popup boxes</h1>
		<button onclick="alertmsg()">Alert Box</button><br>
		<button onclick="confirmmsg()">Confirm Box</button><br>
		<button onclick="promptmsg()">Prompt Box</button><br>
		<p id="pmsg"></p>
	</body>
</html>
```

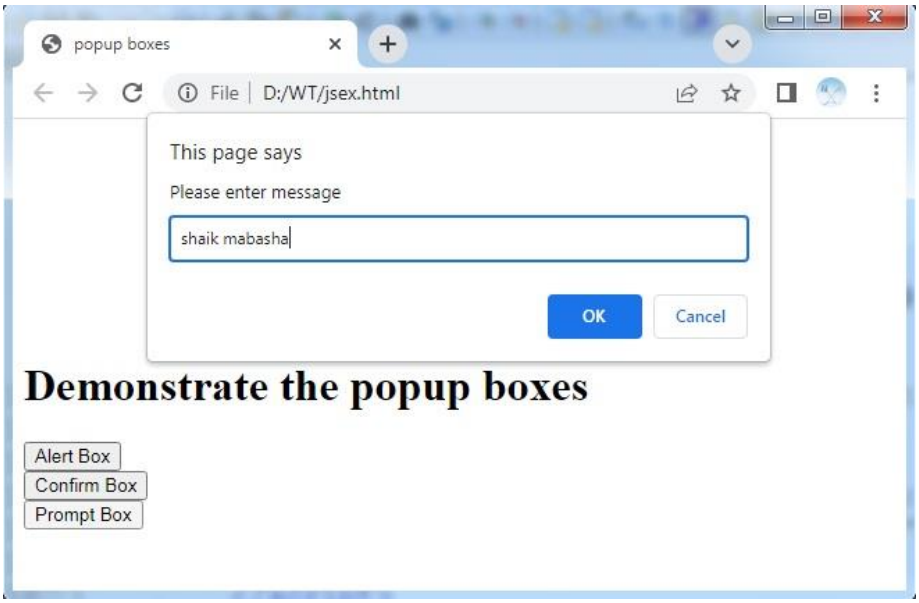
**Output 1:** popup box

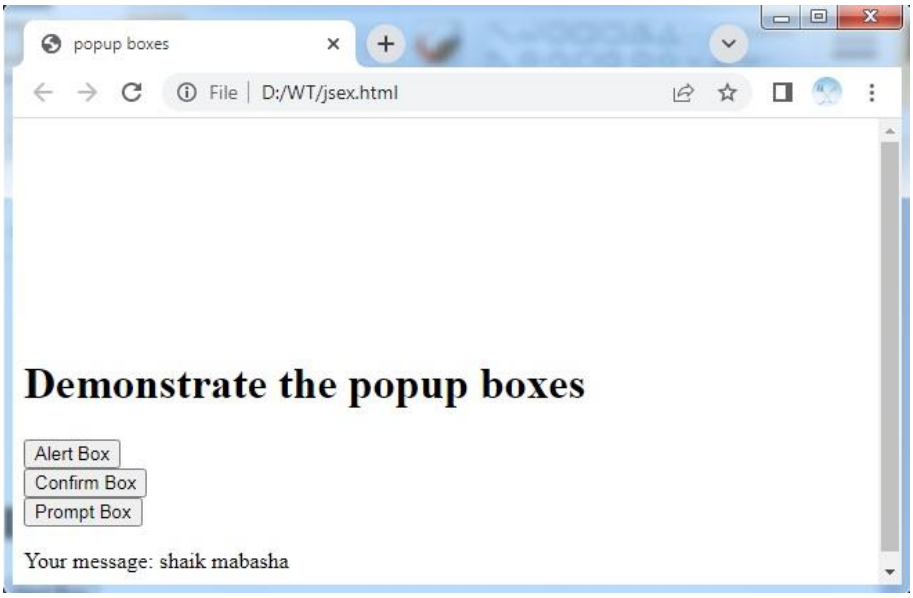


**Output 2:** confirm box




**Output 3:** prompt box



# JavaScript Functions, Events, Image Maps, and Animations

## Exploring Functions:

- A function is a collection of statements that is executed when it is called at some point in a program.
- JavaScript functions are nothing, but a block of code designed to be perform a specific task.
- The JavaScript functions are divided into two categories:
  a. **Function without parameters:** Does not contain parameters.
  b. **Function with parameters:** Contain parameters in parenthesis.

  JavaScript provides number of build-in global function; some of them are as follow

| Function | Description |
|---|---|
| alert() | Display the information in message box. This function displays error message when you validate a form. |
| prompt() | The confirm box contains OK and CANCEL buttons. If the user clicks on OK button it return true value and if user click on CANCEL button it return false value. |
| confirm() | The confirm box contains OK and CANCEL buttons. If the user clicks on OK button it return true value and if user click on CANCEL button it return false value. |
| eval() | Evaluate and execute a string and return a result. |
| isFinite() | Return Boolean value; indicate whether the argument passed to it is finite or infinite. |
| isNaN() | Determines whether or not a value is an illegal number. NaN stands for Not a Number. |
| parseInt() | Extract a integer number from beginning of a string. |
| parseFloat() | Extract a floating pint number from beginning of a string. |
| Number() | Converts a value of an object into a number. |
| escape() | Encodes special characters except *,@,-,+ and /. |
| unescape() | Decodes a string that is encoded by escape() function. |

## Defining Function in JavaScript

- To define a function in JavaScript, use **function** keyword, followed by the function name, which is followed by the parentheses (contains parameter list).
- Syntax to define a function in JavaScript:

```
function   funName (parameter1, parameter2, parameter3) {
        Code to be executed here
}
```

- Example:

```
<script type="text/javascript">
function  alertmsg()
{
        alert("Hello JavaScript, I am Statements of Function Definition");
}
</script>
```

## Calling (Invoking) Function in JavaScript

- To call a function in JavaScript, you have to simply write the name of the function which is going to be called.
- Example:

```
<script type="text/javascript">
        alertmsg()
</script>
```

## Calling function with Timer:

- In JavaScript the timer is a very important feature
- It allows us to execute a JavaScript function after a specified period, thereby making it possible to add a new dimension, time, to our website.
- With the help of the timer, we can run a command at specified intervals, run loops repeatedly at a predefined time, and synchronize multiple events in a particular time span.
- There are various methods for using it as in the following:
  1. setTimeout()
  2. clearTimeout()
  3. setInterval()
  4. clearInterval()

**1. The setTimeout() method:**

- Executes code at a specified interval.
- Syntax: **setTimeout(function, delayTime)**
- Here, **function** parameter specifies the method that the timer calls and the **delayTime** parameter specifies the number of milliseconds to wait before calling the method.

**2. The clearTimeout() method:**

- Deactivates or cancels the timer that is set using the setTime() method.
- Syntax: **clearTimeout(timer)**
- Here, **timer** is a variable that is created using the setTimeout() method.

**3. The setInterval() method:**

- Executes a function after a specified time interval.
- Syntax: **setInterval(function, intervalTime)**
- Here, function parameters specify the method to be called; whereas, the intervalTime parameter specifies the time interval between the function calls.

**4. The clearInterval() method:**

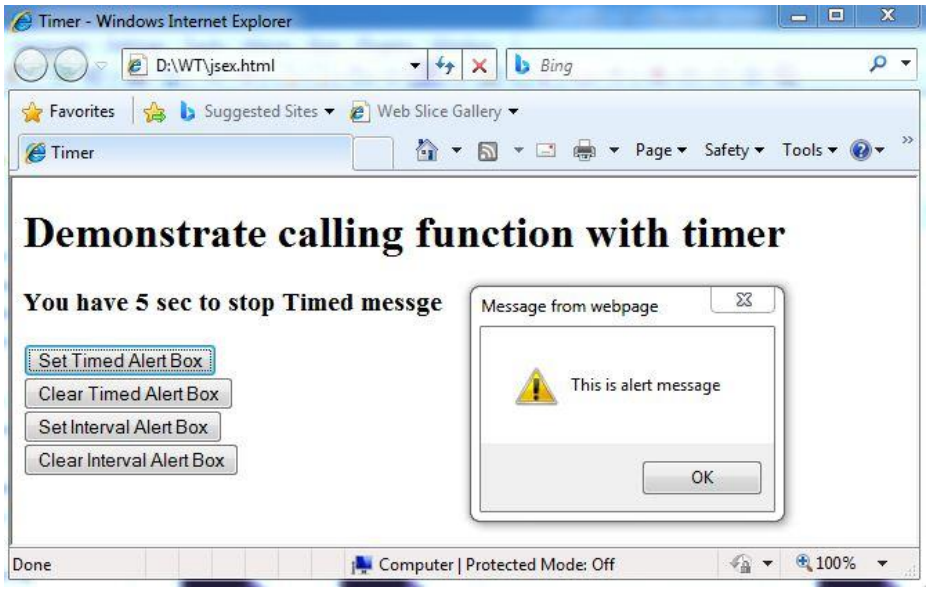
- Deactivates or cancels the timer that is set using the setInterval() method.
- Syntax: **clearInterval(timer)**
- The preceding syntax deactivates the inner timer variable that is created using the setInterval() method.

Example: create a HTML document that demonstrate the calling function with timer in JavaScript

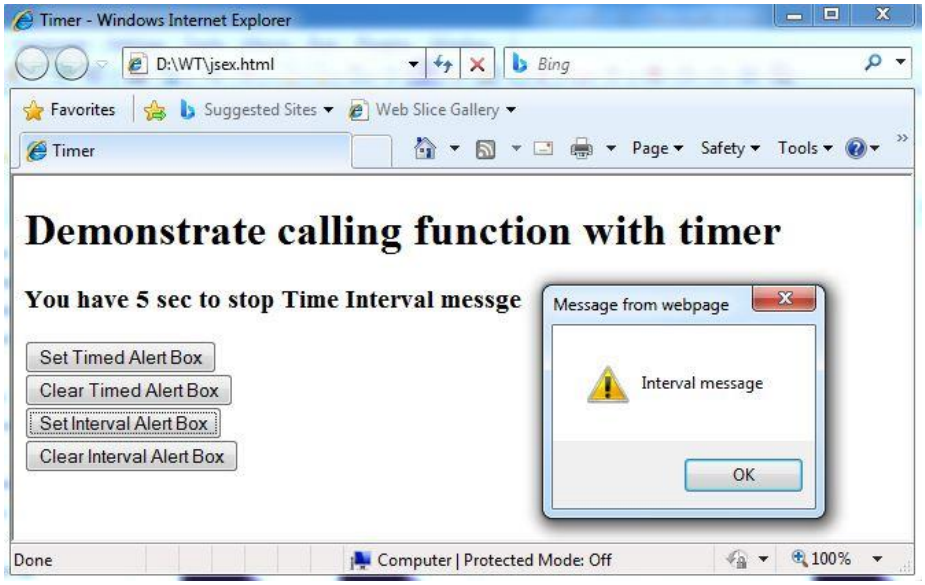
```
<!DOCTYPE html>
<html>
	<head>
		<title>Timer</title>
		<script type="text/javascript">
			var t,i;
			function timedmsg(){
		document.getElementById("msg").innerHTML="You have 5 sec to stop Timed messge";
				t = setTimeout("alert('This is alert message')",5000);
			}
			function cleartimedmsg(){
				clearTimeout(t);
			}
			function intervalmsg(){
	document.getElementById("msg").innerHTML="You have 5 sec to stop Time Interval messge";
				i = setInterval("alert('Interval message')",5000);
			}
			function clearintervalmsg(){
				clearInterval(i);
			}
		</script>
	</head>
```

```
        <body>
                <h1>Demonstrate calling function with timer</h1>
                <h3><p id="msg"></p></h3>
                <button onclick="timedmsg()">Set Timed Alert Box</button><br>
                <button onclick="cleartimedmsg()">Clear Timed Alert Box</button><br>
                <button onclick="intervalmsg()">Set Interval Alert Box</button><br>
                <button onclick="clearintervalmsg()">Clear Interval Alert Box</button><br>
        </body>
</html>
```

**Output 1:** setTimeout




**Output 2:** setInterval



## Exploring Events:

- Events in JavaScript, refer to actions that are detected by a JavaScript program when you perform a particular task.
- For example, onclick event is detected by the program when you click the mouse button.
- Syntax: **onEvent = "code to handle the event"**

### Form Events:

- Events triggered by actions inside a HTML form (applies to almost all HTML elements, but is most used in form elements):

| Attribute | Description | New in HTML5 |
|---|---|---|
| onblur | Fires the moment that the element loses focus | YES |
| onchange | Fires the moment when the value of the element is changed | YES |
| oncontextmenu | Script to be run when a context menu is triggered | YES |
| onfocus | Fires the moment when the element gets focus | NO |
| oninput | Script to be run when an element gets user input | YES |
| oninvalid | Script to be run when an element is invalid | YES |
| onreset | Fires when the Reset button in a form is clicked | YES |
| onsearch | Fires when the user writes something in a search field (for <input="search"> ) | YES |
| onselect | Fires after some text has been selected in an element | NO |
| onsubmit | Fires when a form is submitted | NO |

### Keyboard Events:

| Attribute | Description | New in HTML5 |
|---|---|---|
| onkeydown | Fires when a user is pressing a key | No |
| onkeypress | Fires when a user presses a key | No |
| onkeyup | Fires when a user releases a key | No |

### Mouse Events:

| Attribute | Description | New in HTML5 |
|---|---|---|
| onclick | Fires on a mouse click on the element | No |
| ondblclick | Fires on a mouse double-click on the element | No |
| onmousedown | Fires when a mouse button is pressed down on an element | No |
| onmousemove | Fires when the mouse pointer is moving while it is over an element | No |
| onmouseout | Fires when the mouse pointer moves out of an element | No |
| onmouseover | Fires when the mouse pointer moves over an element | No |
| onmouseup | Fires when a mouse button is released over an element | No |
| onmousewheel | Deprecated. Use the onwheel attribute instead | Yes |
| onwheel | Fires when the mouse wheel rolls up or down over an element | Yes |
| ondrag | Script to be run when an element is dragged | Yes |
| ondragend | Script to be run at the end of a drag operation | Yes |
| ondragenter | Script to be run when an element has been dragged to a valid drop target | Yes |
| ondragleave | Script to be run when an element leaves a valid drop target | Yes |
| ondragover | Script to be run when an element is being dragged over a valid drop target | Yes |
| ondragstart | Script to be run at the start of a drag operation | Yes |
| ondrop | Script to be run when dragged element is being dropped | Yes |
| onscroll | Script to be run when an element's scrollbar is being scrolled | Yes |

**Clipboard Events:**

| Attribute | Description |
|---|---|
| oncopy | Fires when the user copies the content of an element |
| oncut | Fires when the user cuts the content of an element |
| onpaste | Fires when the user pastes some content in an element |

**Media Events:**

- Events triggered by medias like videos, images and audio (applies to all HTML elements, but is most common in media elements, like <audio>, <embed>, <img>, <object>, and <video>).

| Attribute | Description | New in HTML5 |
|---|---|---|
| onabort | Script to be run on abort | Yes |
| oncanplay | Script to be run when a file is ready to start playing (when it has buffered enough to begin) | Yes |
| oncanplaythrough | Script to be run when a file can be played all the way to the end without pausing for buffering | Yes |
| oncuechange | Script to be run when the cue changes in a <track> element | Yes |
| ondurationchange | Script to be run when the length of the media changes | Yes |
| onemptied | Script to be run when something bad happens and the file is suddenly unavailable (like unexpectedly disconnects) | Yes |
| onended | Script to be run when the media has reach the end (a useful event for messages like "thanks for listening") | No |
| onerror | Script to be run when an error occurs when the file is being loaded | Yes |
| onloadeddata | Script to be run when media data is loaded | Yes |
| onloadedmetadata | Script to be run when meta data (like dimensions and duration) are loaded | Yes |
| onloadstart | Script to be run just as the file begins to load before anything is actually loaded | Yes |
| onpause | Script to be run when the media is paused either by the user or programmatically | Yes |
| onplay | Script to be run when the media is ready to start playing | Yes |
| onplaying | Script to be run when the media actually has started playing | Yes |
| onprogress | Script to be run when the browser is in the process of getting the media data | Yes |
| onratechange | Script to be run each time the playback rate changes (like when a user switches to a slow motion or fast forward mode) | No |
| onseeked | Script to be run when the seeking attribute is set to false indicating that seeking has ended | Yes |
| onseeking | Script to be run when the seeking attribute is set to true indicating that seeking is active | Yes |
| onstalled | Script to be run when the browser is unable to fetch the media data for whatever reason | Yes |
| onsuspend | Script to be run when fetching the media data is stopped before it is completely loaded for whatever reason | Yes |
| ontimeupdate | Script to be run when the playing position has changed (like when the user fast forwards to a different point in the media) | Yes |
| onvolumechange | Script to be run each time the volume is changed which (includes setting the volume to "mute") | Yes |
| onwaiting | Script to be run when the media has paused but is expected to resume (like when the media pauses to buffer more data) | No |

**Browser (Window) Events:**

- Events triggered for the window object (applies to the <body> tag):

| Attribute | Description | New in HTML5 |
|---|---|---|
| onafterprint | Script to be run after the document is printed | Yes |
| onbeforeprint | Script to be run before the document is printed | Yes |
| onbeforeunload | Script to be run when the document is about to be unloaded | Yes |
| onerror | Script to be run when an error occurs | Yes |
| onhashchange | Script to be run when there has been changes to the anchor part of the a URL | Yes |
| onload | Fires after the page is finished loading | No |
| onmessage | Script to be run when the message is triggered | Yes |
| onoffline | Script to be run when the browser starts to work offline | Yes |
| ononline | Script to be run when the browser starts to work online | Yes |
| onpagehide | Script to be run when a user navigates away from a page | Yes |
| onpageshow | Script to be run when a user navigates to a page | Yes |
| onpopstate | Script to be run when the window's history changes | Yes |
| onresize | Fires when the browser window is resized | Yes |
| onstorage | Script to be run when a Web Storage area is updated | Yes |
| onunload | Fires once a page has unloaded (or the browser window has been closed) | Yes |
| onundo | Triggers at the time of performing the undo action in a document. | Yes |
| onblur | Trigger when a window loses focus | No |
| onfocus | Trigger when a window gets focus | No |
| onredo | riggers at the time of performing the redo action in a document. | Yes |