**JavaScript Tutorial: Events & Listening to Events | Web Development Tutorials #57**

In this tutorial, we are going to learn how we can make browser events come into action and how should we use them. An HTML event can be something the browser does or something a user does.

Here are some examples of HTML events:

* An HTML web page has finished loading
* An HTML input field was changed
* An HTML button was clicked

Often, when events happen, you may want to do something.

JavaScript lets you execute code when events are detected.

HTML allows event handler attributes,**with JavaScript code**, to be added to HTML elements.

Here is a list of some common HTML events-

* **Onchange-** An HTML element has been changed
* **Onclick-** The user clicks an HTML element
* **Onmouseover-**  The user moves the mouse over an HTML element
* **Onmouseout-**The user moves the mouse away from an HTML element
* **Onkeydown-**The user pushes a keyboard key
* **Onload-**The browser has finished loading the page

Make a new file as *tut57.html*and add the boilerplate to get the HTML template. Then give the title as **JS Events**under the <title> tag. Let us now write some HTML to begin the work. We will simply add the paragraph and heading and the result will be as follows-

Now will make a *button*and we want to hide that button if someone clicks on it. But before that we can add some CSS to it to make it look attractive.

If we want that while clicking the button the paragraph should hide and by again clicking, it should come back, then we can write as follows-

<button id="btn" onclick="toggleHide()">Show/Hide</button>

function toggleHide(){

let btn = document.getElementById('btn');

let para = document.getElementById('para');

if(para.style.display != 'none'){

para.style.display = 'none';

}

else{

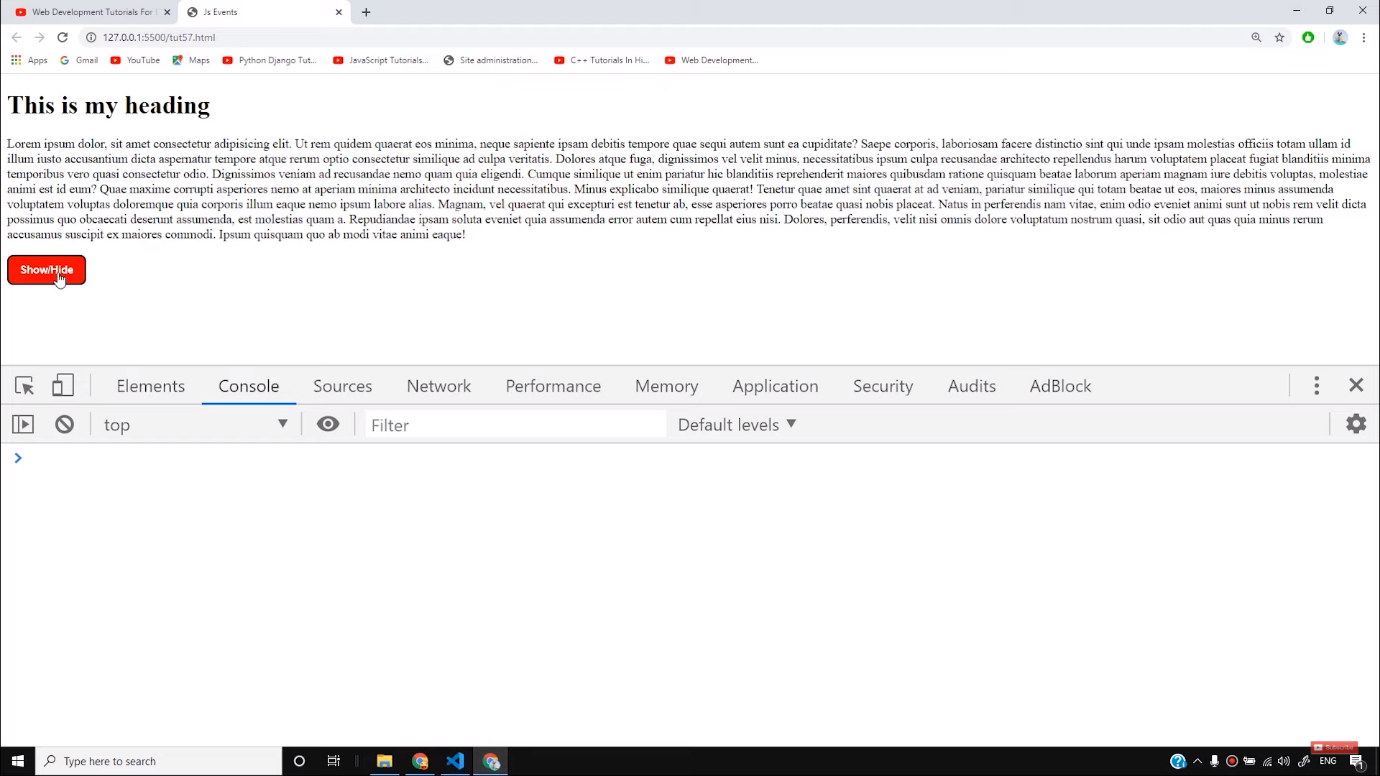
para.style.display = 'block';

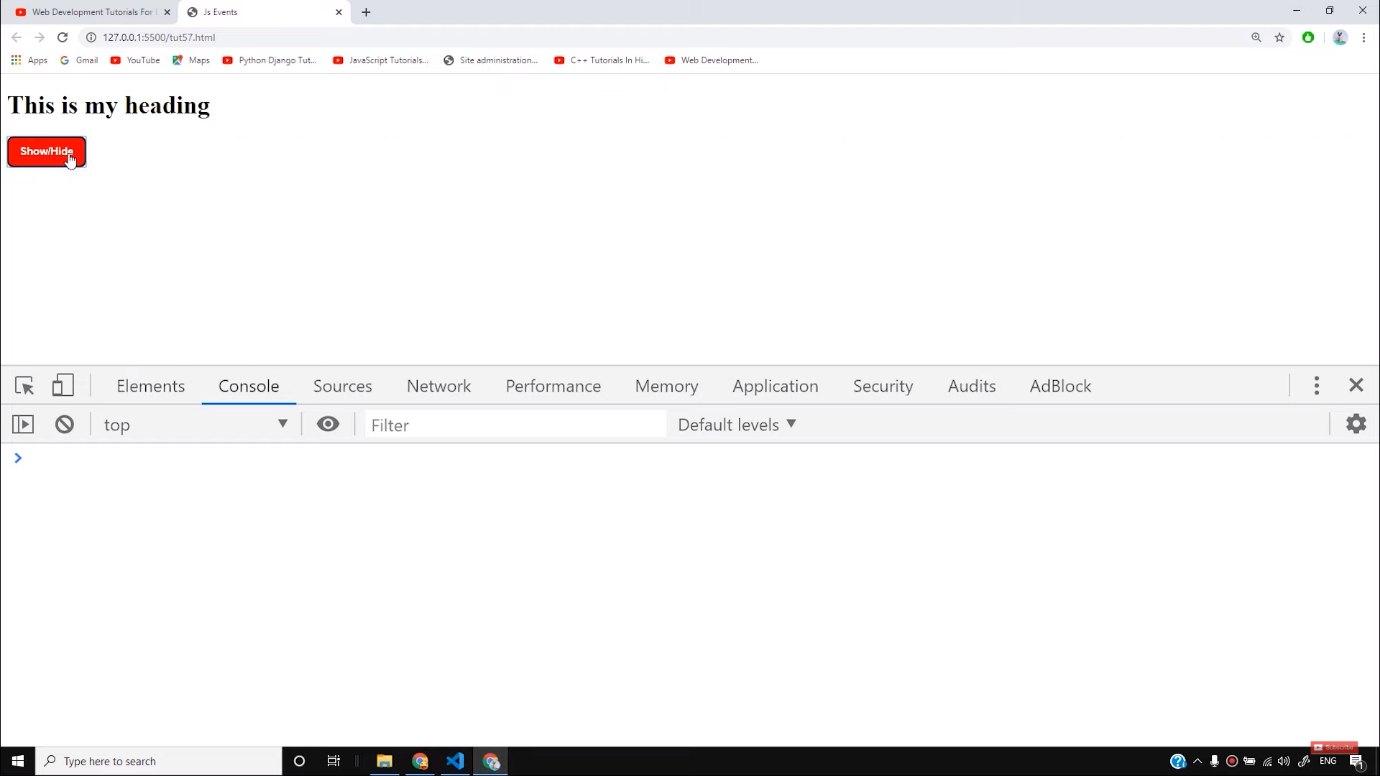
}

}

Copy

The result will be as follows-





In the same way, there is another event listener known as a **mouseover.**For example, if we want to alert when the mouse pointer is on the paragraph, then we can write the code as follows-

let para = document.getElementById('para');

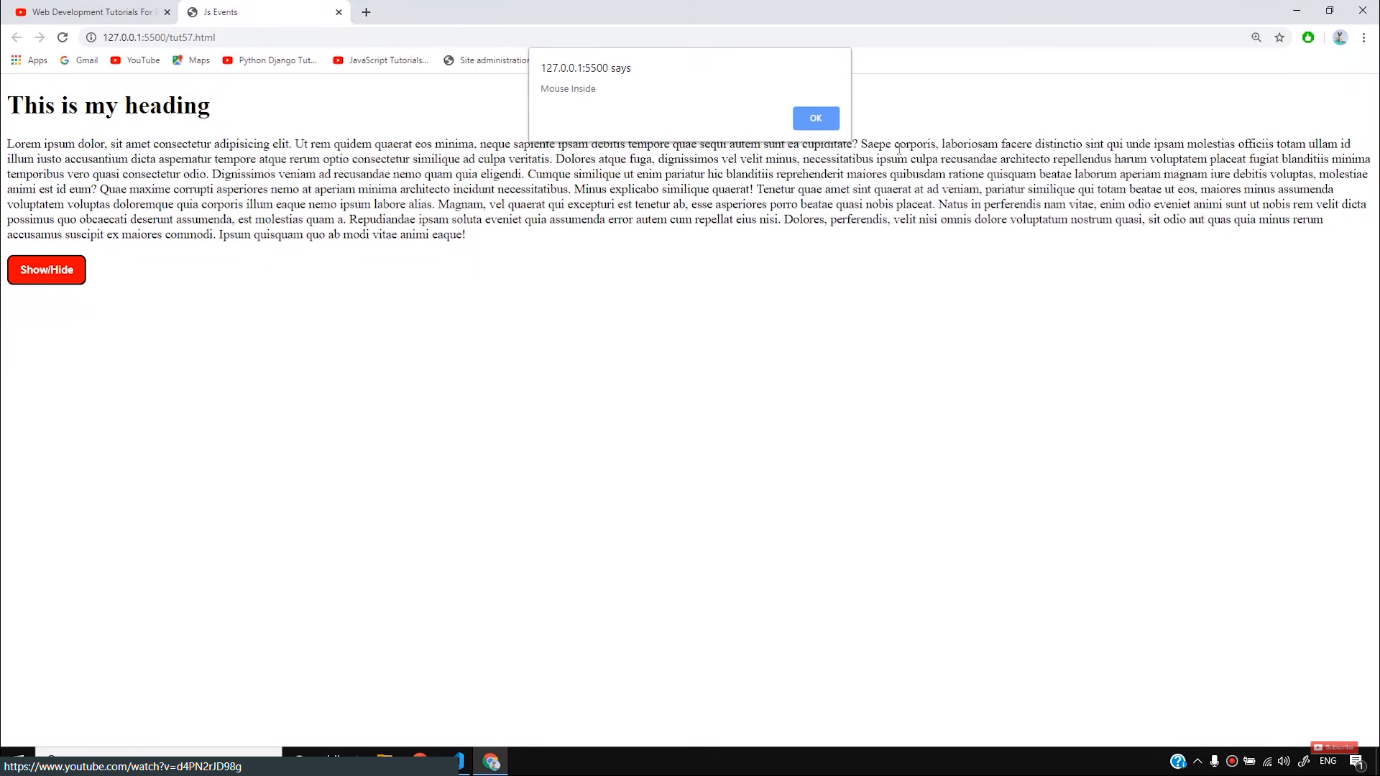
para.addEventListener('mouseover', function run(){

alert('Mouse Inside')

});

Copy

The result of the above code will be that whenever the mouse pointer will be over the paragraph, it will make an alert as shown below-



In the same way, if we want to alert when the pointer is outside the paragraph, we can write the code as follows-

para.addEventListener('mouseout', function run(){

alert('Mouse now went outside')

});

Copy

In the same way, you can try out with all the other events and practice more. In the further tutorials, we will practice some more event handlers in our websites. Till then stay with the tutorials.

**Code as described/written in the video**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Js Events</title>

</head>

<style>

#btn{

padding:10px 14px;

background-color: red;

border: 2px solid black;

color: white;

font-weight: bold;

border-radius: 8px;

cursor: pointer;

}

</style>

<body>

<!-- Browser events:

click

contextmenu

mouseover/mouseout

mousedown/mouseup

mousemove

submit

focus

DOMContentLoaded

transitionend -->

<div class="container">

<h1>This is my heading </h1>

<p id="para">Lorem ipsum dolor, sit amet consectetur adipisicing elit. Ut rem quidem quaerat eos minima, neque sapiente ipsam debitis tempore quae sequi autem sunt ea cupiditate? Saepe corporis, laboriosam facere distinctio sint qui unde ipsam molestias officiis totam ullam id illum iusto accusantium dicta aspernatur tempore atque rerum optio consectetur similique ad culpa veritatis. Dolores atque fuga, dignissimos vel velit minus, necessitatibus ipsum culpa recusandae architecto repellendus harum voluptatem placeat fugiat blanditiis minima temporibus vero quasi consectetur odio. Dignissimos veniam ad recusandae nemo quam quia eligendi. Cumque similique ut enim pariatur hic blanditiis reprehenderit maiores quibusdam ratione quisquam beatae laborum aperiam magnam iure debitis voluptas, molestiae animi est id eum? Quae maxime corrupti asperiores nemo at aperiam minima architecto incidunt necessitatibus. Minus explicabo similique quaerat! Tenetur quae amet sint quaerat at ad veniam, pariatur similique qui totam beatae ut eos, maiores minus assumenda voluptatem voluptas doloremque quia corporis illum eaque nemo ipsum labore alias. Magnam, vel quaerat qui excepturi est tenetur ab, esse asperiores porro beatae quasi nobis placeat. Natus in perferendis nam vitae, enim odio eveniet animi sunt ut nobis rem velit dicta possimus quo obcaecati deserunt assumenda, est molestias quam a. Repudiandae ipsam soluta eveniet quia assumenda error autem cum repellat eius nisi. Dolores, perferendis, velit nisi omnis dolore voluptatum nostrum quasi, sit odio aut quas quia minus rerum accusamus suscipit ex maiores commodi. Ipsum quisquam quo ab modi vitae animi eaque!</p>

</div>

<button id="btn" onclick="toggleHide()">Show/Hide</button>

<script>

let para = document.getElementById('para');

para.addEventListener('mouseover', function run(){

console.log('Mouse Inside')

});

para.addEventListener('mouseout', function run(){

console.log('Mouse now went outside')

});

function toggleHide(){

// let btn = document.getElementById('btn');

let para = document.getElementById('para');

if(para.style.display != 'none'){

para.style.display = 'none';

}

else{

para.style.display = 'block';

}

}

</script>

</body>

</html>

# JavaScript Events

[❮ Previous](https://www.w3schools.com/js/js_objects.asp)[Next ❯](https://www.w3schools.com/js/js_strings.asp)

HTML events are **"things"** that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can **"react"** on these events.

## **HTML Events**

An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:

* An HTML web page has finished loading
* An HTML input field was changed
* An HTML button was clicked

Often, when events happen, you may want to do something.

JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

With single quotes:

<element event=**'*some JavaScript*'**>

With double quotes:

<element event=**"*some JavaScript*"**>

In the following example, an onclick attribute (with code), is added to a <button> element:

### **Example**

<button onclick="document.getElementById('demo').innerHTML = Date()">The time is?</button>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onclick1)

In the example above, the JavaScript code changes the content of the element with id="demo".

In the next example, the code changes the content of its own element (using **this**.innerHTML):

### **Example**

<button onclick="this.innerHTML = Date()">The time is?</button>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onclick)

JavaScript code is often several lines long. It is more common to see event attributes calling functions:

### **Example**

<button onclick="displayDate()">The time is?</button>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_events1)

ADVERTISEMENT

## **Common HTML Events**

Here is a list of some common HTML events:

|  |  |
| --- | --- |
| **Event** | **Description** |
| onchange | An HTML element has been changed |
| onclick | The user clicks an HTML element |
| onmouseover | The user moves the mouse over an HTML element |
| onmouseout | The user moves the mouse away from an HTML element |
| onkeydown | The user pushes a keyboard key |
| onload | The browser has finished loading the page |

The list is much longer: [W3Schools JavaScript Reference HTML DOM Events](https://www.w3schools.com/jsref/dom_obj_event.asp).

## **JavaScript Event Handlers**

Event handlers can be used to handle and verify user input, user actions, and browser actions:

* Things that should be done every time a page loads
* Things that should be done when the page is closed
* Action that should be performed when a user clicks a button
* Content that should be verified when a user inputs data
* And more ...

Many different methods can be used to let JavaScript work with events:

* HTML event attributes can execute JavaScript code directly
* HTML event attributes can call JavaScript functions
* You can assign your own event handler functions to HTML elements
* You can prevent events from being sent or being handled
* And more ...

# JavaScript HTML DOM Events

[❮ Previous](https://www.w3schools.com/js/js_htmldom_animate.asp)[Next ❯](https://www.w3schools.com/js/js_htmldom_eventlistener.asp)

HTML DOM allows JavaScript to react to HTML events:

**Mouse Over Me**

**Click Me**

## **Reacting to Events**

A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.

To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

onclick=*JavaScript*

Examples of HTML events:

* When a user clicks the mouse
* When a web page has loaded
* When an image has been loaded
* When the mouse moves over an element
* When an input field is changed
* When an HTML form is submitted
* When a user strokes a key

In this example, the content of the <h1> element is changed when a user clicks on it:

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<h1 onclick="this.innerHTML = 'Ooops!'">Click on this text!</h1>  
  
</body>  
</html>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onclick2)

In this example, a function is called from the event handler:

### **Example**

<!DOCTYPE html>  
<html>  
<body>  
  
<h1 onclick="changeText(this)">Click on this text!</h1>  
  
<script>  
function changeText(id) {  
  id.innerHTML = "Ooops!";  
}  
</script>  
  
</body>  
</html>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onclick3)

ADVERTISEMENT

## **HTML Event Attributes**

To assign events to HTML elements you can use event attributes.

### **Example**

Assign an onclick event to a button element:

<button onclick="displayDate()">Try it</button>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_events1)

In the example above, a function named displayDate will be executed when the button is clicked.

## **Assign Events Using the HTML DOM**

The HTML DOM allows you to assign events to HTML elements using JavaScript:

### **Example**

Assign an onclick event to a button element:

<script>  
document.getElementById("myBtn").onclick = displayDate;  
</script>

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_events2)

In the example above, a function named displayDate is assigned to an HTML element with the id="myBtn".

The function will be executed when the button is clicked.

## **The onload and onunload Events**

The onload and onunload events are triggered when the user enters or leaves the page.

The onload event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

The onload and onunload events can be used to deal with cookies.

### **Example**

<body onload="checkCookies()">

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onload)

## **The onchange Event**

The onchange event is often used in combination with validation of input fields.

Below is an example of how to use the onchange. The upperCase() function will be called when a user changes the content of an input field.

### **Example**

<input type="text" id="fname" onchange="upperCase()">

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onchange)

## **The onmouseover and onmouseout Events**

The onmouseover and onmouseout events can be used to trigger a function when the user mouses over, or out of, an HTML element:

Mouse Over Me

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_events_mouseover)

## **The onmousedown, onmouseup and onclick Events**

The onmousedown, onmouseup, and onclick events are all parts of a mouse-click. First when a mouse-button is clicked, the onmousedown event is triggered, then, when the mouse-button is released, the onmouseup event is triggered, finally, when the mouse-click is completed, the onclick event is triggered.

Click Me

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_events_mousedown)

## **More Examples**

[onmousedown and onmouseup](https://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onmousedown)  
Change an image when a user holds down the mouse button.

[onload](https://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onload)  
Display an alert box when the page has finished loading.

[onfocus](https://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onfocus)  
Change the background-color of an input field when it gets focus.

[Mouse Events](https://www.w3schools.com/js/tryit.asp?filename=tryjs_event_onmouse)  
Change the color of an element when the cursor moves over it.

# JavaScript HTML DOM EventListener

[❮ Previous](https://www.w3schools.com/js/js_htmldom_events.asp)[Next ❯](https://www.w3schools.com/js/js_htmldom_navigation.asp)

## **The addEventListener() method**

### **Example**

Add an event listener that fires when a user clicks a button:

document.getElementById("myBtn").addEventListener("click", displayDate);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_displaydate)

The addEventListener() method attaches an event handler to the specified element.

The addEventListener() method attaches an event handler to an element without overwriting existing event handlers.

You can add many event handlers to one element.

You can add many event handlers of the same type to one element, i.e two "click" events.

You can add event listeners to any DOM object not only HTML elements. i.e the window object.

The addEventListener() method makes it easier to control how the event reacts to bubbling.

When using the addEventListener() method, the JavaScript is separated from the HTML markup, for better readability and allows you to add event listeners even when you do not control the HTML markup.

You can easily remove an event listener by using the removeEventListener() method.

## **Syntax**

element.addEventListener(event, function, useCapture);

The first parameter is the type of the event (like "click" or "mousedown" or any other [HTML DOM Event](https://www.w3schools.com/jsref/dom_obj_event.asp).)

The second parameter is the function we want to call when the event occurs.

The third parameter is a boolean value specifying whether to use event bubbling or event capturing. This parameter is optional.

Note that you don't use the "on" prefix for the event; use "click" instead of "onclick".

## **Add an Event Handler to an Element**

### **Example**

Alert "Hello World!" when the user clicks on an element:

element.addEventListener("click", function(){ alert("Hello World!"); });

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_add)

You can also refer to an external "named" function:

### **Example**

Alert "Hello World!" when the user clicks on an element:

element.addEventListener("click", myFunction);  
  
function myFunction() {  
  alert ("Hello World!");  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_add2)

ADVERTISEMENT

## **Add Many Event Handlers to the Same Element**

The addEventListener() method allows you to add many events to the same element, without overwriting existing events:

### **Example**

element.addEventListener("click", myFunction);  
element.addEventListener("click", mySecondFunction);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_add_many)

You can add events of different types to the same element:

### **Example**

element.addEventListener("mouseover", myFunction);  
element.addEventListener("click", mySecondFunction);  
element.addEventListener("mouseout", myThirdFunction);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_add_many2)

## **Add an Event Handler to the window Object**

The addEventListener() method allows you to add event listeners on any HTML DOM object such as HTML elements, the HTML document, the window object, or other objects that support events, like the xmlHttpRequest object.

### **Example**

Add an event listener that fires when a user resizes the window:

window.addEventListener("resize", function(){  
  document.getElementById("demo").innerHTML = sometext;  
});

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_dom)

## **Passing Parameters**

When passing parameter values, use an "anonymous function" that calls the specified function with the parameters:

### **Example**

element.addEventListener("click", function(){ myFunction(p1, p2); });

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_parameters)

## **Event Bubbling or Event Capturing?**

There are two ways of event propagation in the HTML DOM, bubbling and capturing.

Event propagation is a way of defining the element order when an event occurs. If you have a <p> element inside a <div> element, and the user clicks on the <p> element, which element's "click" event should be handled first?

In bubbling the inner most element's event is handled first and then the outer: the <p> element's click event is handled first, then the <div> element's click event.

In capturing the outer most element's event is handled first and then the inner: the <div> element's click event will be handled first, then the <p> element's click event.

With the addEventListener() method you can specify the propagation type by using the "useCapture" parameter:

addEventListener(event, function, useCapture);

The default value is false, which will use the bubbling propagation, when the value is set to true, the event uses the capturing propagation.

### **Example**

document.getElementById("myP").addEventListener("click", myFunction, true);  
document.getElementById("myDiv").addEventListener("click", myFunction, true);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_usecapture)

## **The removeEventListener() method**

The removeEventListener() method removes event handlers that have been attached with the addEventListener() method:

### **Example**

element.removeEventListener("mousemove", myFunction);