JavaScript HTML DOM EventListener

The addEventListener() method

Example

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

```
"color:black">"color:red">
document."color:black">getElementById("color:brown">"myBtn")."c
olor:black">addEventListener("color:brown">"click",
displayDate);
```

Try it Yourself »

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").

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:

```
"color:black">"color:red">
element."color:black">addEventListener("color:brown">"click",
"color:mediumblue">function(){ alert("color:brown">"Hello
World!"); });
```

Try it Yourself »

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

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

```
"color:black">"color:red">
element."color:black">addEventListener("color:brown">"click",
myFunction);
"color:red">
element."color:black">addEventListener("color:brown">"click",
mySecondFunction);
"color:red">
```

Try it Yourself »

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

Example

```
"color:black">
element."color:black">addEventListener("color:brown">"mouseover
", myFunction);
element."color:black">addEventListener("color:brown">"click",
mySecondFunction);
element."color:black">addEventListener("color:brown">"mouseout"
, myThirdFunction);
```

Try it Yourself »

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:

```
"color:black">
window."color:black">addEventListener("color:brown">"resize",
"color:mediumblue">function(){
"color:red">
    document."color:black">getElementById("color:brown">"demo")
."color:black">innerHTML = sometext;
"color:red"> });"color:red">
```

Passing Parameters

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

Example

```
"color:black">
element."color:black">addEventListener("color:brown">"click",
"color:mediumblue">function(){ myFunction(p1, p2);
});"color:red">
```

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

```
"color:black">
document."color:black">getElementById("color:brown">"myP")."col
or:black">addEventListener("color:brown">"click", myFunction,
"color:mediumblue">true);
document."color:black">getElementById("color:brown">"myDiv")."c
olor:black">addEventListener("color:brown">"click", myFunction,
"color:mediumblue">true);"color:red">
```

Try it Yourself »

The removeEventListener() method

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

Example

```
"color:black">
element."color:black">removeEventListener("color:brown">"mousem
ove", myFunction);
```

Try it Yourself »

Browser Support

The numbers in the table specifies the first browser version that fully supports these methods.

Method	Chrome	IE	Firefox	Safari	Opera
addEventListener()	1.0	9.0	1.0	1.0	7.0
removeEventListener()	1.0	9.0	1.0	1.0	7.0

Note: The addEventListener() and removeEventListener() methods are not supported in IE 8 and earlier versions and Opera 6.0 and earlier versions. However, for these specific browser versions, you can use the attachEvent() method to attach an event handlers to the element, and the detachEvent() method to remove it:

```
element.attachEvent(event, function);
element.detachEvent(event, function);
```

Example

Cross-browser solution:

```
var x = document.getElementById("myBtn");
if (x.addEventListener) {                   // For all major browsers, except IE 8 and earlier
    x.addEventListener("click", myFunction);
} else if (x.attachEvent) {                   // For IE 8 and earlier versions
    x.attachEvent("onclick", myFunction);
}
```

HTML DOM Event Object Reference

For a list of all HTML DOM events, look at our complete HTML DOM Event Object Reference.

COLOR PICKER

HOW TO

Tabs
Dropdowns
Accordions
Convert Weights
Animated Buttons
Side Navigation
Top Navigation
Modal Boxes
Progress Bars
Parallax
Login Form
HTML Includes
Google Maps
Range Sliders
Tooltips
Slideshow
Filter List
Sort List

SHARE

CERTIFICATES

HTML, CSS, JavaScript, PHP, jQuery, Bootstrap and XML.

Read More »

REPORT ERROR
PRINT PAGE
FORUM
ABOUT

Top 10 Tutorials

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
SQL Tutorial
PHP Tutorial
jQuery Tutorial
Angular Tutorial
How To Tutorial

Top 10 References

HTML Reference
CSS Reference
JavaScript Reference
W3.CSS Reference
Bootstrap Reference
SQL Reference
PHP Reference
HTML Colors
jQuery Reference
AngularJS Reference

Top 10 Examples

HTML Examples
CSS Examples

JavaScript Examples
W3.CSS Examples
Bootstrap Examples
HTML DOM Examples
PHP Examples
jQuery Examples
Angular Examples
XML Examples

Web Certificates

HTML Certificate
CSS Certificate
JavaScript Certificate
jQuery Certificate
PHP Certificate
Bootstrap Certificate
XML Certificate