**JavaScript HTML DOM EventListener**

**addEventListener() 方法**

**实例**

点用户点击按钮时触发监听事件：

*<button id="myBtn">点我</button>*

*<p id="demo"></p>*

*<script>*

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

*function displayDate() {*

*document.getElementById("demo").innerHTML = Date();*

*}*

*</script>*

addEventListener() 方法用于向指定元素添加事件句柄。

addEventListener() 方法添加的事件句柄不会覆盖已存在的事件句柄。

你可以向一个元素添加多个事件句柄。

你可以向同个元素添加多个同类型的事件句柄，如：两个 "click" 事件。

你可以向任何 DOM 对象添加事件监听，不仅仅是 HTML 元素。如： window 对象。

addEventListener() 方法可以更简单的控制事件（冒泡与捕获）。

当你使用 addEventListener() 方法时, JavaScript 从 HTML 标记中分离开来，可读性更强， 在没有控制HTML标记时也可以添加事件监听。

你可以使用 removeEventListener() 方法来移除事件的监听。

**语法**

*element*.addEventListener(*event, function, useCapture*);

第一个参数是事件的类型 (如 "click" 或 "mousedown").

第二个参数是事件触发后调用的函数。

第三个参数是个布尔值用于描述事件是冒泡还是捕获。该参数是可选的。

|  |  |
| --- | --- |
| **Note** | 注意:不要使用 "on" 前缀。 例如，使用 "click" ,而不是使用 "onclick"。 |

**向原元素添加事件句柄**

**实例**

当用户点击元素时弹出 "Hello World!" :

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

你可以使用函数名，来引用外部函数:

**实例**

当用户点击元素时弹出 "Hello World!" :

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

**向同一个元素中添加多个事件句柄**

addEventListener() 方法允许向同个元素添加多个事件，且不会覆盖已存在的事件：

**实例**

*<button id="myBtn">点我</button>*

*<script>*

*var x = document.getElementById("myBtn");*

*x.addEventListener("click", myFunction);*

*x.addEventListener("click", someOtherFunction);*

*function myFunction() {*

*alert ("Hello World!")*

*}*

*function someOtherFunction() {*

*alert ("函数已执行!")*

*}*

*</script>*

你可以向同个元素添加不同类型的事件：

**实例**

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

**向 Window 对象添加事件句柄**

addEventListener() 方法允许你在 HTML DOM 对象添加事件监听， HTML DOM 对象如： HTML 元素, HTML 文档, window 对象。或者其他支出的事件对象如: xmlHttpRequest 对象。

**实例**

当用户重置窗口大小时添加事件监听：

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

**传递参数**

当传递参数值时，使用"匿名函数"调用带参数的函数：

**实例**

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

*<p id="demo"></p>*

*<script>*

*var p1 = 5;*

*var p2 = 7;*

*document.getElementById("myBtn").addEventListener("click", function() {*

*myFunction(p1, p2);*

*});*

*function myFunction(a, b) {*

*var result = a \* b;*

*document.getElementById("demo").innerHTML = result;*

*}*

*</script>*

**事件冒泡或事件捕获？**

事件传递有两种方式：冒泡与捕获。

事件传递定义了元素事件触发的顺序。 如果你将 <p> 元素插入到 <div> 元素中，用户点击 <p> 元素, 哪个元素的 "click" 事件先被触发呢？

在 *冒泡*中，内部元素的事件会先被触发，然后再触发外部元素，即： <p> 元素的点击事件先触发，然后会触发 <div> 元素的点击事件。

在 *捕获*中，外部元素的事件会先被触发，然后才会触发内部元素的事件，即： <div> 元素的点击事件先触发 ，然后再触发 <p> 元素的点击事件。

addEventListener() 方法可以指定 "useCapture" 参数来设置传递类型：

addEventListener(*event*, *function*, *useCapture*);

默认值为 false, 及冒泡传递，当值为 true 时, 事件使用捕获传递。

**实例**

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

[尝试一下 »](http://www.w3cschool.cc/try/try.php?filename=tryjs_addeventlistener_usecapture)

**removeEventListener() 方法**

removeEventListener() 方法移除由 addEventListener() 方法添加的事件句柄:

**实例**

*element*.removeEventListener("mousemove", myFunction);

*<div id="myDIV"> div 元素添加了 onmousemove 事件句柄，鼠标在桔红色的框内移动时会显示随机数。*

*<p>点击按钮移除 DIV 的事件句柄。</p>*

*<button onclick="removeHandler()" id="myBtn">点我</button>*

*</div>*

*<p id="demo"></p>*

*<script>*

*document.getElementById("myDIV").addEventListener("mousemove", myFunction);*

*function myFunction() {*

*document.getElementById("demo").innerHTML = Math.random();*

*}*

*function removeHandler() {*

*document.getElementById("myDIV").removeEventListener("mousemove", myFunction);*

*}*

*</script>*

**浏览器支持**

表格中的数字表示支持该方法的第一个浏览器的版本号。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **方法                 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 |

**注意：** IE 8 及更早 IE 版本，Opera 7.0及其更早版本不支持 addEventListener() 和 removeEventListener() 方法。但是，对于这类浏览器版本可以使用 detachEvent() 方法来移除事件句柄:

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

**实例**

跨浏览器解决方法:

*<button id="myBtn">点我</button>*

*<script>*

*var x = document.getElementById("myBtn");*

*if (x.addEventListener) {*

*x.addEventListener("click", myFunction);*

*} else if (x.attachEvent) {*

*x.attachEvent("onclick", myFunction);*

*}*

*function myFunction() {*

*alert("Hello World!");*

*}*

*</script>*