addEventListener跟on的区别 - JavaScript

为什么需要addEventListener？

先来看一个片段：

html代码

<div id="box">追梦子</div>

用on的代码

**1 window.onload = function(){**

**2 var box = document.getElementById("box");**

**3 box.onclick = function(){**

**4 console.log("我是box1");**

**5 }**

**6 box.onclick = function(){**

**7 box.style.fontSize = "18px";**

**8 console.log("我是box2");**

**9 }**

**10 }  
　　　　　运行结果：“我是box2”**

看到了吧，第二个onclick把第一个onclick给覆盖了，虽然大部分情况我们用on就可以完成我们想要的结果，但是有时我们又需要执行多个相同的事件，很明显如果用on完成不了我们想要的，那不用猜，你们肯定知道了，对！addEventListener可以多次绑定同一个事件并且不会覆盖上一个事件。

用addEventListener的代码

**1 window.onload = function(){**

**2 var box = document.getElementById("box");**

**3 box.addEventListener("click",function(){**

**4 console.log("我是box1");**

**5 })**

**6 box.addEventListener("click",function(){**

**7 console.log("我是box2");**

**8 })**

**9 }  
　　　　运行结果：我是box1  
　　　　　　　　　我是box2**

addEventListenert方法第一个参数填写事件名，注意不需要写on，第二个参数可以是一个函数，第三个参数是指在冒泡阶段还是捕获阶段处理事件处理程序,如果为true代表捕获阶段处理,如果是false代表冒泡阶段处理，第三个参数可以省略，大多数情况也不需要用到第三个参数。

第三个参数的使用

有时候的情况是这样的

<body>  
　　<div id="box">  
　　　　<div id="child"></div>  
　　</div>  
</body>

如果我给box加click事件，如果我直接单击box没有什么问题，但是如果我单击的是child元素，那么它是怎么样执行的？（执行顺序）

**1 box.addEventListener("click",function(){**

**2 console.log("box");**

**3 })**

**4**

**5 child.addEventListener("click",function(){**

**6 console.log("child");**

**7 })  
　　执行的结果：  
　　　　　　　　child  
　　　　　　　　box**

也就是说，默认情况事件是按照事件冒泡的执行顺序进行的。

如果第三个参数写的是true，则按照事件捕获的执行顺序进行的。

**1 box.addEventListener("click",function(){**

**2 console.log("box");**

**3 })**

**4**

**5 child.addEventListener("click",function(){**

**6 console.log("child");**

**7 })**

**执行的结果：**

**box**

**child**

事件冒泡执行过程：从最具体的的元素（你单击的那个元素）开始向上开始冒泡，拿我们上面的案例讲它的顺序是：child->box

事件捕获执行过程：从最不具体的元素（最外面的那个盒子）开始向里面冒泡，拿我们上面的案例讲它的顺序是：box->child