# 1.事件处理内容概述p156

## 1>认识事件

|  |
| --- |
|  |

# 2.事件处理的3种选择方案p157

## 事件处理的3种方式: 在元素中监听,使用dom元素的onclick绑定函数,和给元素添加事件监听也就是addEventListener,其中第一种方法基本不使用.

|  |
| --- |
|  |
|  |

## 第二种是onclick=function(){},需要在script标签里面写

|  |
| --- |
|  |

### 效果如下

|  |
| --- |
|  |

### 注意:这种方法只能够绑定一个函数.如果你绑定2次,后面的会覆盖前面的.比如我们修改一下代码,绑定2次函数

|  |
| --- |
|  |

### 运行代码,你会发现,只有后面绑定的函数有用

|  |
| --- |
|  |

## 第三种就是添加事件监听addEventListener

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

### 这种方式的好处就是可以多次添加,而且不会覆盖,一个事件可以触发多个函数,我们把代码修改一下,如图

|  |
| --- |
|  |

### 点击一下按钮3.可以同时触发这两个函数

|  |
| --- |
|  |

### 而且这种方式非常灵活,我们可以添加各种不同的事件,如鼠标进入事件

|  |
| --- |
|  |

### 还可以添加鼠标离开事件

|  |
| --- |
|  |

### 效果,把鼠标移动到按钮上面,触发mouseenter事件,把鼠标移出按钮,触发mouseleave事件

|  |
| --- |
|  |

# 3.事件流-事件冒泡和事件捕获p158

|  |
| --- |
|  |

## 都有哪些常见的事件呢

|  |
| --- |
|  |

## 事件流的概念

|  |
| --- |
|  |

## 冒泡就是当你点击子元素,你不仅仅可以触发子元素的事件处理函数,还会触发父亲元素和爷爷甚至爷爷的爷爷的事件处理函数,如我们在body里面添加一个div然后在div里面添加一个span,注意:需要对他们设置样式

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>Dom事件冒泡和事件捕获</title>      <style>          .box{              display: flex;              justify-content: center;              align-items: center;              width: 200px;              height: 200px;              background-color: rgb(243, 243, 243);          }          .box span{              width: 100px;              height: 100px;              background-color: rgb(255, 47, 47);          }      </style>  </head>  <body>        <div class="box">          <span></span>      </div>      <script>           let span = document.querySelector("span")           let boxDiv = document.querySelector(".box")           let body = document.body           span.onclick = function(){              console.log("span标签被点击了...");           }           boxDiv.onclick = function(){              console.log("div标签被点击了...");           }           body.onclick = function(){              console.log("body被点击了...");           }      </script>  </body>  </html> |

## 然后我们在页面上面点击span元素,你会发现,div和body的事件处理函数也一并触发了

|  |
| --- |
|  |

## 冒泡是从子元素往父元素传递,是单向的,当父元素被点击,这个事件默认不会传递给子元素,也就是从里往外的传递方式,如果你想让事件从父元素传递给子元素,你需要使用事件捕获来实现

|  |
| --- |
|  |

## 如果你先使用事件捕获,你需要额外传递一个参数给addEventListener方法

|  |
| --- |
|  |

## 刷新页面,然后点击span元素,效果如下

|  |
| --- |
|  |

## 在开发中使用的是事件冒泡,其实我们也可以既使用冒泡,又使用捕获,不过是捕获先触发,然后才是冒泡,我们把冒泡的代码打开,然后添加冒泡,在捕获函数里面添加捕获,代码如下

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>Dom事件冒泡和事件捕获</title>      <style>          .box{              display: flex;              justify-content: center;              align-items: center;              width: 200px;              height: 200px;              background-color: rgb(243, 243, 243);          }          .box span{              width: 100px;              height: 100px;              background-color: rgb(255, 47, 47);          }      </style>  </head>  <body>        <div class="box">          <span></span>      </div>      <script>           let span = document.querySelector("span")           let boxDiv = document.querySelector(".box")           let body = document.body          //  span.onclick = function(){          //     console.log("span标签被点击了...");          //  }          //  boxDiv.onclick = function(){          //     console.log("div标签被点击了...");          //  }          //  body.onclick = function(){          //     console.log("body被点击了...");          //  }          //默认方式是事件冒泡          //另外一种方式          span.addEventListener("click",()=>{              console.log("span标签被点击了...冒泡");          })          boxDiv.addEventListener("click",()=>{              console.log("div标签被点击了...冒泡");          })          body.addEventListener("click",()=>{              console.log("body被点击了...冒泡");          })          //如果你想使用事件捕获,你需要给addEventListener添加一个参数,是true就说明使用事件捕获          span.addEventListener("click",()=>{              console.log("span标签被点击了...捕获");          },true)          boxDiv.addEventListener("click",()=>{              console.log("div标签被点击了...捕获");          },true)          body.addEventListener("click",()=>{              console.log("body被点击了...捕获");          },true)      </script>  </body>  </html> |

### 刷新网页,点击span标签,效果如下

|  |
| --- |
|  |

# 4.事件对象解析p159

## 1>连接事件对象

|  |
| --- |
|  |

### 事件对象其实就是元素的事件处理函数的形参,它里面又很多信息,我们可以获取事件的类型,事件在哪个元素上面发生的等等很多信息.注意:当发生了事件冒泡后,在父元素的事件对象里面的target仍然是子元素,因为事件是由子元素触发的.

## 2>事件对象的常见的属性和方法

|  |
| --- |
|  |
|  |

### 其中target和currentTarget是事件对象中最重要的属性,target是指触发事件的元素,currentTarget是当前处理事件的元素.他们是不一样的.比如我们在div的事件处理函数里面添加下面的代码

|  |
| --- |
|  |

### 次时当我们点击按钮时,这个函数输出的target是按钮.currentTarget是div

|  |
| --- |
|  |

### 但是当我们点击div,你会发现他们其实是一样的

|  |
| --- |
|  |

# 5.阻止默认行为 (stopPropagation和preventDefault) p160

## 有些元素比如超链接,它是由默认行为的,就是会跳转到我们指定的链接.这时候,我们其实还是可以给他添加事件监听,在里面做额外的事情,然后,它仍然会执行它的默认行为,比如有下面的代码:

|  |
| --- |
|  |

### 运行程序,点击超链接,会先弹出一个选项卡,然后跳转到Google网站.

|  |
| --- |
|  |
|  |

## 假如我们不想让它跳转,可以在监听函数里面用一个形参来接收事件对象,然后利用事件对象的preventDefault方法来阻止默认行为,我们可以修改代码如下

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

#### 可以看到此时默认的跳转行为就被阻止了.

## 然后我们在页面中添加一个div,div里面添加一个span,在span里面添加一个按钮,并且给他们设置一些样式

|  |
| --- |
|  |
|  |

## 然后我们先获取这几个元素,然后分别给他们添加事件监听函数

|  |
| --- |
|  |

## 默认情况下,他会有事件冒泡的现象,我们点击按钮,然后它会触发这三元素的事件监听函数

|  |
| --- |
|  |

## 如果我们需要阻止事件冒泡,同样需要用一个形参来接收事件对象,然后调用他的stopPropagation方法,我们修改一些代码

|  |
| --- |
|  |

## 此时点击按钮,就只会触发按钮的点击事件

|  |
| --- |
|  |

## 注意:此时点击p标签还是会冒泡的,因为我们没有在他里面阻止冒泡

## 同理如果我们要阻止事件捕获的传递,就需要在div的事件里面使用stopPropagation方法,比如我们添加一些事件捕获代码,如下

|  |
| --- |
|  |

### 效果:虽然我们在按钮的事件事件里面阻止了事件冒泡,但是默认还是会发生事件捕获的传递.

|  |
| --- |
|  |

## 要阻止事件捕获的传递,可以在div的事件捕获函数里面接收一个event对象,然后调用他的stopPropagation方法,因为事件捕获是从外往里传递的.

|  |
| --- |
|  |

### 效果:神奇的一幕发生了,点击按钮,只会触发div的捕获事件....

|  |
| --- |
|  |

### 这个其实不奇怪,事件的传递顺序: div的捕获->p的捕获->按钮的捕获->按钮的冒泡->p的冒泡->div的冒泡.如果在div的捕获里面阻止了事件流的传递,那么后面的都不会发生

### 注意:只有在特定的情况下,我们才会去阻止事件对象的默认行为

# 6.事件处理函数中的this p161

|  |
| --- |
|  |

## 新建一个按钮,在script标签中获取这个按钮,然后给他添加点击事件监听,在监听函数里面输出this,发现当我们点击按钮,this其实就是这个按钮

|  |
| --- |
|  |
|  |

## 然后我们修改一下代码,把按钮放大div里面,然后我们也给div绑定事件处理函数,

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

### 说明:这个事件处理函数绑定到哪一个元素上面,this就是指这个元素

# 7.EventTarget中的removeEv P162

|  |
| --- |
|  |

## 1>我们新建一个html文件,在里面添加一个按钮,然后我们在script里面给他添加点击事件监听

|  |
| --- |
|  |

### 然后当我们点击这个按钮,这个监听事件就会触发,浏览器就会调用我们的事件处理函数

|  |
| --- |
|  |

## 2>如果我们现在又不需要这个事件处理函数,我们其实可以把它移除,使用removeEventListener方法

|  |
| --- |
|  |

### 刷新页面,然后你会发现,点击按钮已经没有反应了,说明监听事件函数已经被移除了

|  |
| --- |
|  |

## 3>当然我们也可以先添加事件监听,然后在一个时间后如5秒,再移除事件监听

|  |
| --- |
|  |
|  |

## 注意.如果你需要移除事件监听函数,你必须给这个函数起名字,在addEventListener中也使用这个命名函数,不能使用匿名函数

# 8.EventTarget中的dispatch P163

## dispatch方法是用来派发事件的,比如我们需要window来监听一个不存在的事件,

|  |
| --- |
|  |

## 那么这个监听肯定是失败的,那我们然后让window来监听自定义事件呢?需要先派发事件.

|  |
| --- |
|  |

### 运行程序,3秒后就会触发事件处理函数

|  |
| --- |
|  |