# 前面的事件虚构标记,参考: [学习笔记4-事件处理1](coderwhy-jsdom学习笔记4-事件处理1.docx)

# 9.事件委托-案例1 p164

## 1>什么是事件委托

|  |
| --- |
|  |

## 小案例:新建一个html文件,然后添加一个ul元素,在里面添加10个li元素,然后我们有一个需求,就是被点击的li我们需要给他一个样式,比如修改字体颜色和背景颜色,此时我们可以先把这些li都获取到放到一个lis列表里面.然后我们利用循环来添加点击事件处理函数,在里面把每一个li的样式的classList列表添加一个active样式class,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>事件委托</title>      <style>          .active{              background-color: pink;              color: white;              font-size: 20px;              width: 200px;          }      </style>  </head>  <body>      <ul>          <li>1</li>          <li>2</li>          <li>3</li>          <li>4</li>          <li>5</li>          <li>6</li>          <li>7</li>          <li>8</li>          <li>9</li>          <li>10</li>      </ul>      <script>        let lis = document.querySelectorAll("li")        for(let li of lis){          li.addEventListener("click",function(){              this.classList.add("active") //这里只能用this,不能用li否则只有最后一个有效果          })        }      </script>  </body>  </html> |

### 效果

|  |
| --- |
|  |
|  |

## 虽然能够实现功能,但是有点啰嗦,我们可以使用事件委托的发生把事件交给ul处理

|  |
| --- |
|  |

### 这样子也能够实现效果,利用的就是事件冒泡的原理

|  |
| --- |
|  |

## 我们再提一个需求,就是如果不是当前点击的项目,需要把样式还原,一个方法可以实现,不过有点麻烦,就是先把所有的li是样式移除,然后再给当前选中的li添加样式

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |
|  |
|  |

### 我们还有一种方法,就是利用.active来查找元素,找到就把他的样式移除,因为我们在点击第二个li的时候就需要做了,所以他前面应该只有一个元素有这个样式

|  |
| --- |
|  |

### 这样子也是可行的,而且性能更高,另外还有一种方法,就是用变量记录上一次的选择,然后把它的样式移除

|  |
| --- |
|  |

#### 这种方法很好,效率也很高,也很常用.

## 不过这个案例有点问题,就是当我们点击ul的时候,所以的li都会被添加样式,这个问题的解决办法如下,只需要做一个判断,保证它是li

|  |
| --- |
|  |

## 案例完整代码如下

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>事件委托</title>      <style>          .active{              background-color: pink;              color: white;              font-size: 20px;              width: 200px;          }      </style>  </head>  <body>      <ul>          <li>1</li>          <li>2</li>          <li>3</li>          <li>4</li>          <li>5</li>          <li>6</li>          <li>7</li>          <li>8</li>          <li>9</li>          <li>10</li>      </ul>      <script>        //1.传统的方式      //   let lis = document.querySelectorAll("li")      //   for(let li of lis){      //     li.addEventListener("click",function(){      //         this.classList.add("active") //这里只能用this,不能用li否则只有最后一个有效果      //     })      //   }      //2.使用事件委托,统一在ul中监听      // let ul = document.querySelector("ul")      // ul.onclick = function(e){      //     // console.log("某个li被点击了...");     //    if(e.target !== ul){      //     e.target.classList.add("active")      //    }      // }      //3.新需求,从前选中的对象才添加样式,其他的样式把添加这个样式      // let lis = document.querySelectorAll("li")      // let ul = document.querySelector("ul")      // ul.onclick = function(e){      //     // console.log("某个li被点击了...");      //     //先吧所有的li是active样式移除,      //     for(let li of lis){      //         if(li.classList.contains("active"))      //         {      //             li.classList.remove("active") //这里不能用this      //         }        //     }      //     //然后再给当前选中的li添加样式      //   if(e.target !== ul){      //     e.target.classList.add("active")      //    }      // }      //4.新需求的第二种实现方法      // let ul = document.querySelector("ul")      // ul.onclick = function(e){      //     // console.log("某个li被点击了...");      //     //先找到有active样式的样式      //     let liAct = ul.querySelector(".active")      //     if(liAct){      //         liAct.classList.remove("active")      //     }      //     //然后再给当前选中的li添加样式      //   if(e.target !== ul){      //     e.target.classList.add("active")      //    }      // }       //5.新需求的第三种实现方法,用变量记录选中的元素,在下一次点击的时候把这个元素的样式移除      let ul = document.querySelector("ul")      let lastLi = null      ul.onclick = function(e){          // console.log("某个li被点击了...");          //存在才移除样式          if(lastLi && e.target.tagName ==="LI"){              lastLi.classList.remove("active")          }          //然后再给当前选中的li添加样式,ul是不添加这个样式的.          if(e.target !== ul){              e.target.classList.add("active")               //保存本次选中的项,然后在下一次点击的时候移除他的样式              lastLi = e.target          }          }      </script>  </body>  </html> |

# 10.事件委托-案例2 p165

## 比较简单,就是工具点击按钮的属性来判断点击了哪个按钮,完整代码如下

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>事件委托2</title>  </head>  <body>      <div class="box">          <button data-action="remove">移除</button>          <button data-action="new">新建</button>          <button data-action="search">搜索</button>      </div>      <script>         //在div的点击事件中(通过按钮的事件冒泡触发)根据点击的按钮的文本来输出不同的信息      //    方式1      //    let div = document.querySelector(".box")      //    div.onclick = function(e){      //        switch(e.target.textContent){      //           case "移除":      //             console.log("点击了移除按钮...");      //             break;      //           case "新建":      //             console.log("点击了新建按钮...");      //             break;      //           case "搜索":      //              console.log("点击了搜索按钮...");      //              break;      //        }      //    }      //    方式2      let div = document.querySelector(".box")      div.onclick = function(e){             switch(e.target.dataset.action){                case "remove":                  console.log("点击了移除按钮...");                  break;                case "new":                  console.log("点击了新建按钮...");                  break;                case "search":                   console.log("点击了搜索按钮...");                   break;             }         }      </script>  </body>  </html> |

### 效果

|  |
| --- |
|  |

# 11.鼠标常见事件p167

|  |
| --- |
|  |

## 注意:

### 1.如果你同时监听mousedown,mouseup和click事件,你会发现,先触发mousedown事件,再触发mouseup事件,然后才是click事件.如果你已经有了mousedown和mouseup,你就不需要使用click了

### 2.监听鼠标移动的时候,即使左键没有按下它还可以监听到移动事件,如果我们需要它只有鼠标左键按下并且移动,从监听鼠标移动事件,该这么做?

#### 答案:很显然我们需要先监听到鼠标左键被按下了.然后才监听移动.我们可以定义一个全局的标记isDown用来标记鼠标是否按下,默认是false,然后在onmousedown事件里面把它设置为true,在onmouseup事件里面把它设置为false,然后在onmousemove事件里面判断isDown是否为true,如果是,就执行一些代码,如果不是就什么都不做

### 3.mouseover和mouseout支持冒泡,所以在需要事件委托的时候就得用他们.

# 12,mouseover和mouseenter的区别p168

|  |
| --- |
|  |

## 1>小案例:

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>mouseover和mouseenter的区别</title>      <style>          .box{              width: 200px;              height: 200px;              background-color: yellowgreen;          }      </style>  </head>  <body>      <div class="box"></div>      <script>          let box = document.querySelector(".box")          box.onmouseenter=function(){              console.log("mouse enter...");            }          box.onmouseleave=function(){              console.log("mouse leave...");            }          box.onmouseover=function(){              console.log("mouse over...");          }          box.onmouseout=function(){              console.log("mouse out...");          }      </script>  </body>  </html> |
|  |

## 2>我们个div添加一个span元素,并且给他添加样式,然后给他添加mouseenter和mouseleave事件监听,为了方便学习,我们把mouseover和mouseout事件监听先注释了

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>mouseover和mouseenter的区别</title>      <style>          .box{              width: 200px;              height: 200px;              background-color: yellowgreen;              /\* display: flex;              justify-content: center;              align-items: center; \*/          }          .box p{              width: 100px;              height: 100px;              background-color: orange;              margin-left: 100px;          }      </style>  </head>  <body>      <div class="box">          <p></p>      </div>      <script>          let box = document.querySelector(".box")          let p = document.querySelector("p")          box.onmouseenter=function(){              console.log("box mouse enter...");            }          box.onmouseleave=function(){              console.log("box mouse leave...");            }          p.onmouseenter=function(){              console.log("p mouse enter...");            }          p.onmouseleave=function(){              console.log("p mouse leave...");            }          // box.onmouseover=function(){          //     console.log("box mouse over...");          // }          // box.onmouseout=function(){          //     console.log("box mouse out...");          // }      </script>  </body>  </html> |

### 此时我们把鼠标移动到p并且里面,会同时触发div和p的mouseenter事件,因为p是在div里面

|  |
| --- |
|  |

### 注意到了吗,先触发div的mouseenter事件,再触发p的事件

## 3>然后我们修改一些css样式使得p在div的中间,

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>mouseover和mouseenter的区别</title>      <style>          .box{              width: 200px;              height: 200px;              background-color: yellowgreen;              display: flex;              justify-content: center;              align-items: center;          }          .box p{              width: 100px;              height: 100px;              background-color: orange;              /\* margin-left: 100px; \*/          }      </style>  </head>  <body>      <div class="box">          <p></p>      </div>      <script>          let box = document.querySelector(".box")          let p = document.querySelector("p")          box.onmouseenter=function(){              console.log("box mouse enter...");            }          box.onmouseleave=function(){              console.log("box mouse leave...");            }          p.onmouseenter=function(){              console.log("p mouse enter...");            }          p.onmouseleave=function(){              console.log("p mouse leave...");            }          // box.onmouseover=function(){          //     console.log("box mouse over...");          // }          // box.onmouseout=function(){          //     console.log("box mouse out...");          // }      </script>  </body>  </html> |

### 此时把鼠标移动的到元素里面,效果是这样子的.

|  |
| --- |
|  |

### 看见了吗,是没有发生冒泡,因为mouseenter不支持事件冒泡

## 4>然后我们来看看mouseover和mouseout

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>mouseover和mouseenter的区别</title>      <style>          .box{              width: 200px;              height: 200px;              background-color: yellowgreen;              display: flex;              justify-content: center;              align-items: center;          }          .box p{              width: 100px;              height: 100px;              background-color: orange;              /\* margin-left: 100px; \*/          }      </style>  </head>  <body>      <div class="box">          <p></p>      </div>      <script>          let box = document.querySelector(".box")          let p = document.querySelector("p")          //1.mouseenter and mouseleave          // box.onmouseenter=function(){          //     console.log("box mouse enter...");            // }          // box.onmouseleave=function(){          //     console.log("box mouse leave...");            // }          // p.onmouseenter=function(){          //     console.log("p mouse enter...");            // }          // p.onmouseleave=function(){          //     console.log("p mouse leave...");            // }          //2.mouseover and mouseout          box.onmouseover=function(){              console.log("box mouse over...");          }          box.onmouseout=function(){              console.log("box mouse out...");          }          p.onmouseover=function(){              console.log("p mouse over...");          }          p.onmouseout=function(){              console.log("p mouse out...");          }      </script>  </body>  </html> |
|  |
|  |

### 注意到了吗,鼠标刚刚进入div就触发div的mouseover事件,然后当鼠标进入p并且,先触发div的mouseout事件,然后再触发p的mouseover事件,然后触发div的mouseover事件,因为p把mouseover通过冒泡传递给了div,当鼠标离开p标签,会先触发p的mouseout事件,然后会触发div的mouseout事件,然后再触发div的mouseover事件

## 5>我们把p标签的事件监听注释了,然后再来测试

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>mouseover和mouseenter的区别</title>      <style>          .box{              width: 200px;              height: 200px;              background-color: yellowgreen;              display: flex;              justify-content: center;              align-items: center;          }          .box p{              width: 100px;              height: 100px;              background-color: orange;              /\* margin-left: 100px; \*/          }      </style>  </head>  <body>      <div class="box">          <p></p>      </div>      <script>          let box = document.querySelector(".box")          let p = document.querySelector("p")          //1.mouseenter and mouseleave          // box.onmouseenter=function(){          //     console.log("box mouse enter...");            // }          // box.onmouseleave=function(){          //     console.log("box mouse leave...");            // }          // p.onmouseenter=function(){          //     console.log("p mouse enter...");            // }          // p.onmouseleave=function(){          //     console.log("p mouse leave...");            // }          //2.mouseover and mouseout          box.onmouseover=function(){              console.log("box mouse over...");          }          box.onmouseout=function(){              console.log("box mouse out...");          }          // p.onmouseover=function(){          //     console.log("p mouse over...");          // }          // p.onmouseout=function(){          //     console.log("p mouse out...");          // }      </script>  </body>  </html> |
|  |
|  |
|  |

### 可以看到,我们把p的事件监听注释了,但是div还是能够触发事件,上面发生了事件冒泡,指令p标签通过事件冒泡把事件委托给div响应.

## 注意,我们同时监听mouseenter和mouseover,鼠标进入,会先触发mouse over,然后再触发mouseenter,鼠标离开会先触发mouseout然后才触发mouseleave,其实只有一个元素的话,他们是没有太大的区别都是可以使用的,但是如果又嵌套元素,情况就不一样了.如果你需要使用事件委托,就只能够使用mouseover和mouseout

# 13. ,mouseover和mouseenter应用的区别p169

## 我们通过一个小案例来学习,新建一个html文件,然后新建一个div在它里面添加3个按钮,要求:当鼠标进入每一个按钮,需要把按钮的文本输出到控制台

## 1.>先来看方式一,代码比较多

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>mouseenter和mouseover应用的区别</title>      <style>          .box{              display: flex;              width: 300px;              height: 300px;          }          .box button{            flex: 1;            height: 50px;          }          .box button:nth-child(1){              background-color: yellow;          }          .box button:nth-child(2){              background-color: pink          }          .box button:nth-child(3){              background-color: cyan;          }      </style>  </head>  <body>     <div class="box">      <button class="btn1">删除</button>      <button class="btn2">新增</button>      <button class="btn3">搜索</button>     </div>     <script>        //方式一      //   let btn1 = document.querySelector(".btn1")      //   let btn2 = document.querySelector(".btn2")      //   let btn3 = document.querySelector(".btn3")        //   btn1.onmouseenter = function(){      //     console.log(this.textContent);      //   }      //   btn1.onmouseleave = function(){      //     console.log(this.textContent+" mouse leave...");      //   }      //   btn2.onmouseenter = function(){      //     console.log(this.textContent);      //   }      //   btn2.onmouseleave = function(){      //     console.log(this.textContent+" mouse leave...");      //   }      //   btn3.onmouseenter = function(){      //     console.log(this.textContent);      //   }      //   btn3.onmouseleave = function(){      //     console.log(this.textContent+" mouse leave...");      //   }      //当然,可以使用循环      let btns = document.querySelectorAll("button")      for(btn of btns){         btn.onmouseenter=function(){           console.log(this.textContent);         }         btn.onmouseleave=function(){           console.log(this.textContent+" mouse leave...");         }      }      //方式二      // let box = document.querySelector(".box")      // box.onmouseover = function (e) {      //     console.log(e.target.textContent);      // }      // box.onmouseout = function (e) {      //     console.log(e.target.textContent+"mouse out...");      // }     </script>  </body>  </html> |

### 可以看到,是可以正常工作的.

|  |
| --- |
|  |
|  |

## 2>然后我们来看看方式二

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>mouseenter和mouseover应用的区别</title>      <style>          .box{              display: flex;              width: 300px;              height: 300px;          }          .box button{            flex: 1;            height: 50px;          }          .box button:nth-child(1){              background-color: yellow;          }          .box button:nth-child(2){              background-color: pink          }          .box button:nth-child(3){              background-color: cyan;          }      </style>  </head>  <body>     <div class="box">      <button class="btn1">删除</button>      <button class="btn2">新增</button>      <button class="btn3">搜索</button>     </div>     <script>        //方式一      //   let btn1 = document.querySelector(".btn1")      //   let btn2 = document.querySelector(".btn2")      //   let btn3 = document.querySelector(".btn3")        //   btn1.onmouseenter = function(){      //     console.log(this.textContent);      //   }      //   btn1.onmouseleave = function(){      //     console.log(this.textContent+" mouse leave...");      //   }      //   btn2.onmouseenter = function(){      //     console.log(this.textContent);      //   }      //   btn2.onmouseleave = function(){      //     console.log(this.textContent+" mouse leave...");      //   }      //   btn3.onmouseenter = function(){      //     console.log(this.textContent);      //   }      //   btn3.onmouseleave = function(){      //     console.log(this.textContent+" mouse leave...");      //   }      //当然,可以使用循环      // let btns = document.querySelectorAll("button")      // for(btn of btns){      //    btn.onmouseenter=function(){      //      console.log(this.textContent);      //    }      //    btn.onmouseleave=function(){      //      console.log(this.textContent+" mouse leave...");      //    }      // }      //方式二      let box = document.querySelector(".box")      box.onmouseover = function (e) {          if(e.target!=box)          {              console.log(e.target.textContent);          }        }      box.onmouseout = function (e) {          if(e.target!=box)          {              console.log(e.target.textContent+"mouse out...");          }        }     </script>  </body>  </html> |
|  |
|  |

## 可以看到,效果是一样的,但是代码更加精简

# 14.常见键盘事件和案例

|  |
| --- |
|  |