目录

[javascript进阶 1](#_Toc10822)

[复习： 3](#_Toc5364)

[一、 获取页面卷动值 5](#_Toc26843)

[二、 onscroll事件 5](#_Toc2947)

[三、 节流 5](#_Toc31203)

[四、 鼠标滚轮事件 6](#_Toc22379)

[4.1 事件对象 7](#_Toc452)

[五、 键盘事件 9](#_Toc17177)

[5.1 onkeydown 9](#_Toc25199)

[5.2 onkeyup 9](#_Toc11551)

[5.3 onkeypress 10](#_Toc6890)

[5.4 tabIndex属性 10](#_Toc20515)

[六、 面向对象 11](#_Toc15383)

# 复习：

事件绑定方式与区别：

dom0级事件绑定方式: dom.onclick = function() {}

只可以给一个元素的一个事件绑定一个函数

事件执行的顺序：

在高级浏览器中，代码的绑定顺序

在IE中，是优先执行

this指向触发事件的对象

dom2级绑定事件方式： dom.addEventListener(type, fn, bool)

可以给一个元素的一个事件绑定多个函数

执行事件的顺序按照代码的绑定顺序

this指向触发事件的对象

IE中的高级绑定方式: dom.attachEvent(type, fn)

可以给一个元素的一个事件绑定多个函数

执行事件的顺序，是逆序执行

this指向window

移除事件的方式：

DOM2级移除事件的方式：

dom.removeEventListener(type, fn, bool)

IE中的移除事件方式：

dom.detachEvent(type, fn)

事件对象与属性：

当某个事件执行的时候，会产生一些列的信息，这些信息被浏览器收集起来并且封装为一个对象传递到事件对象中

offsetX offsetY: 这两个属性标记的是鼠标位于元素内部的位置

clientX clientY: 这两个属性标记的是鼠标位于视口中的位置

pageX pageY: 这两个属性标记的是鼠标位于页面之间的位置

screenX screenY: 这两个属性标记的是鼠标位于屏幕中的位置

阻止冒泡：

在高级浏览器中： e.stopPropagation()

在IE中：e.cancelBubble = true

阻止默认事件：

在高级浏览器中：e.preventDefault()

在IE中：returnValue = false

如果使用的是dom0级事件绑定方式， 还可以使用return false

快捷尺寸：

原生：

clientWidth clientHeight: content + padding

offsetWidth offsetHeight: content + padding + border

clientLeft clientTop: 边框的厚度

jquery中：

$(dom).width() $(dom).height(): content

$(dom).innerWidth $(dom).innerHeight(): content + padding

$(dom).outerWidth() $(dom).outerHeight(): content + padding + border

$(dom).outerWidth(true) $(dom).outerHeight(true): content + padding + border + margin

定位父元素和定位值：

获取定位父元素： offsetParent

定位值：

offsetLeft: 获取距离定位父元素的左边的距离

offsetTop: 获取距离定位父元素上边的距离

jquery中的定位值：

$(dom).posoition()， 返回的是一个对象， 对象中包含了元素的left 和top

jquery中获取元素到页面之间的距离：

$(dom).offset()

# 获取页面卷动值

页面卷动值指的是视口上面看不到的部分（被卷起来）

获取页面卷动值的方式：

在新版本的chrome、火狐，IE中都支持： document.documentElement.scrollTop

在旧版本的chrmoe，其它浏览器中支持： document.body.scrollTop

# onscroll事件

页面中的视口是一定的，当页面的高度超出了视口之后，页面中会出现滚动条， 此时， 当滚动鼠标滚轮的时候，会改变滚动条的位置， 该事件就会触发

触发该事件的条件改变页面的卷动值

改变页面卷动值的方式：

滚动鼠标滚轮、 按下键盘上的空格键、按下键盘上的上下键、按下键盘上的PgUp、PgDn

举例：

当滚动鼠标滚轮的时候：

|  |
| --- |
|  |

# 节流

nscroll事件是一个高频率的事件， 如果把代码直接放入该事件中， 代码将会被执行很多次，没有必要，这个时候要进行节流， 我们开启一个延时器将代码放入， 还要清除之前的延时器， 最终确定开启一个延时器即可，从而保证代码只执行一次

举例：

|  |
| --- |
| 1. // 获取元素 2. var $box = $("#box"); 3. var timer = null; 4. // 给document添加onscroll事件 5. document.onscroll = function() { 6. // console.log(123); 7. // 获取页面的卷动值 8. var scrollTop = document.documentElement.scrollTop; 9. // console.log(scrollTop); 10. // 不断的判断页面的卷动值 11. if (scrollTop >= 2000) { 12. // console.log("导航栏要出现了"); 13. // 清除之前的延时器 14. clearTimeout(timer); 15. // 开启延时器赋值timer 16. timer = setTimeout(function() { 17. // 当符合条件的时候， 导航栏要出现 18. $box.animate({"opacity": 1}, 1000, function() { 19. console.log("出现了"); 20. }) 21. }, 200) 22. } else { 23. // 清除之前的延时器 24. clearTimeout(timer); 25. // 开启延时器赋值timer 26. timer = setTimeout(function() { 27. // 当符合条件的时候， 导航栏要出现 28. $box.animate({"opacity": 0}, 1000); 29. }, 200) 30. } 31. } |

# 鼠标滚轮事件

事件名称：onmousewheel

当滚动鼠标滚轮的时候会触发该事件

该事件存在兼容性问题：

在火狐中不支持onmousewheel事件， 支持DOMMouseScroll事件

在chrome中滚动鼠标滚轮：

|  |
| --- |
|  |

在IE中滚动鼠标滚轮：

|  |
| --- |
|  |

在火狐中滚动鼠标滚轮， 没有反应

原因是火狐中不支持该事件，支持方式:

|  |
| --- |
| 1. dom.addEventListener(“DOMMouseScroll”, function() {}) |

在火狐中滚动鼠标滚轮：

|  |
| --- |
|  |

## 4.1 事件对象

在鼠标点击事件，我们关心的是鼠标的位置

在鼠标滚轮事件中，我们关心的是滚轮的方向

在chrome中向下滚动滚轮：

|  |
| --- |
|  |

在chrome中向上滚动滚轮：

|  |
| --- |
|  |

在IE中向下滚动滚轮：

|  |
| --- |
|  |

在IE中向上滚动滚轮：

|  |
| --- |
|  |

**总结：**

**在非火狐中（chrome、IE）指示鼠标滚轮方向的属性： e.wheelDelta**

**如果是向下滚动是-120的倍数**

**如果是向上滚动是120的倍数**

在火狐中向下滚动：

|  |
| --- |
|  |

在火狐中向上滚动：

|  |
| --- |
|  |

**总结：**

**在火狐中，指示鼠标滚轮方向的属性： e.detail**

**如果是向下滚动：3的倍数**

**如果是向上滚动： -3的倍数**

# 键盘事件

## 5.1 onkeydown

该事件是当键盘被按下的时候触发

举例:

|  |
| --- |
| 1. // onkeydown 2. document.onkeydown = function() { 3. console.log("键盘按下了"); 4. } |

结果：

|  |
| --- |
|  |

## 5.2 onkeyup

该事件是在当键盘抬起的时候触发

举例：

|  |
| --- |
| 1. // onkeyup 2. document.onkeyup = function() { 3. console.log("键盘抬起了"); 4. } |

结果：

|  |
| --- |
|  |

## 5.3 onkeypress

该事件是在有字符输入的时候执行

举例：

|  |
| --- |
| 1. // onkeypress 2. document.onkeypress = function() { 3. console.log("有字符输入"); 4. } |

结果：

|  |
| --- |
|  |

## 5.4 tabIndex属性

该属性是html的标准属性，页面中的元素有很多， 当用户失去鼠标的情况下，想要获取元素可以通过键盘上tab键来获取， 此时添加tabIndex属性，它的属性值，决定了当用户按下tab键的时候获取元素焦点的顺序

按下tab键是按照tabIndex的值来获取焦点

按下shift + tab 可以逆序获取元素的焦点

按下tab键：

|  |
| --- |
|  |

按下shift + tab键：

|  |
| --- |
|  |

# 面向对象

我们之前书写的代码都是面向过程的方式

举例：

|  |
| --- |
| 1. var name1 = "老王"; 2. var sex = "男"; 3. var age = 40; 4. var length = 180; |

面向对象:

|  |
| --- |
| 1. var person = { 2. name: "老王", 3. sex: "男", 4. age: 40, 5. length: 180 6. } |

也就是说，面向对象是另一种书写代码的方式

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |