爱创课堂前端培训

javascript第三天

班级：爱创课堂十七期

讲师：彭帅伟

日期：2018年9月29日

# 复习：

call和apply:

作用: 执行函数并改变函数中的this指向

call: 接收多个参数， 第一个是要改变的this指向， 从第二个参数开始都是原函数所需要的参数。

apply:接收两个参数，第一个是要改变的this指向，第二个参数是一个数组。

dom0级：

绑定方式： dom.onclick = function() {}

只能绑定一个事件函数

只能绑定到冒泡阶段

this指向触发事件的元素

事件对象可以传递到函数中

在IE中事件对象不能被传递

dom2级：

绑定方式：dom.addEventListener(type, fn, boolean)

可以绑定多个事件函数

可以与dom0级同时存在

this指向触发事件的元素

事件对象可以被传递进来

事件类型是不带on的

事件执行顺序是顺序执行

attachEvent:

绑定方式: dom.attachEvent(type, fn)

可以绑定多个事件函数

可以与dom0级同时存在

this指向window

事件对象可以被传递

事件类型是带on的

事件执行顺序 dom0优先执行，再逆序执行attachEvent事件

移除事件：

dom2级移除方式：

dom.removeEventListener(type, fn, boolean)

需要注意：匿名函数无法移除， 必须将函数提取为有名函数

detachEvent:

dom.detachEvent(type, fn)

使用方式 同上

事件对象和属性：

在事件触发的时候，会产生一系列的信息，这些信息会被浏览器收集起来并封装为对象传递到事件函数中。

offsetX offsetY: 这两个属性标记的是鼠标位于元素内部的位置

clientX clinetY: 标记的是鼠标位于视口中的位置

pageX pageY: 标记的是鼠标位于页面的位置

screenX screenY: 标记的是鼠标位于屏幕中的位置

traget: 触发事件的元素

currentTarget: 绑定事件的元素

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：

e.returnValue = false;

如果是dom0还可以使用：

return false;

# 快捷尺寸

## clientWidth clientHeight

样式：

|  |
| --- |
| 1. #box { 2. float: left; 3. padding: 10px; 4. background-color: red; 5. border: 10px solid blue; 6. } |

|  |
| --- |
|  |

输出clientWidth和clientHeight:

|  |
| --- |
|  |

总结：

clientWidth和clientHeight包含的是：content + padding区域

## offsetWidth offsetHeight

样式：

|  |
| --- |
| 1. #box { 2. float: left; 3. padding: 10px; 4. background-color: red; 5. border: 10px solid blue; 6. } |

|  |
| --- |
|  |

输出offsetWidth 和 offsetHeight:

|  |
| --- |
|  |

总结：

offsetWidth 和 offsetHeight包含的是： content + padding + border 区域

## clinetLeft clientTop

clientLeft: 左边框的宽度

clientTop: 上边宽的宽度

样式：

|  |
| --- |
| 1. #box { 2. float: left; 3. padding: 10px; 4. background-color: red; 5. border: 10px solid blue; 6. border-left-width: 20px; 7. } |

|  |
| --- |
|  |

## jquery中快捷尺寸

样式：

|  |
| --- |
| 1. #box { 2. float: left; 3. padding: 10px; 4. background-color: red; 5. border: 10px solid blue; 6. border-left-width: 20px; 7. margin: 20px; 8. } |

显示：

|  |
| --- |
|  |

content:

|  |
| --- |
| 1. // content 2. console.log("content宽是:" + $box.width()); 3. console.log("content高是:" + $box.height()); |



content + padding:

|  |
| --- |
| 1. // content + padding 2. console.log("innerWidth:" + $box.innerWidth()); 3. console.log("innerHeight:" + $box.innerHeight()); |



content + padding + border:

|  |
| --- |
| 1. // content + padding + border 2. console.log("outerWidth:" + $box.outerWidth()); 3. console.log("outerHeight:" + $box.outerHeight()); |



content + padding + border + margin:

|  |
| --- |
| 1. // content + padding + border + margin 2. console.log("outerWidth(true):" + $box.outerWidth(true)); 3. console.log("outerHeight(true):" + $box.outerHeight(true)); |



# 元素的位置和距离

## 2.1 定位父元素

子绝父相中的“父”

该元素不一定是父节点中的parentNode

DOM结构：

|  |
| --- |
| 1. <!DOCTYPE html> 2. <html lang="en"> 3. <head> 4. <meta charset="UTF-8"> 5. <title>Document</title> 6. <style type="text/css"> 7. \* { 8. margin: 0; 9. padding: 0; 10. } 11. #box { 12. position: relative; 13. width: 500px; 14. height: 500px; 15. margin: 0 auto; 16. background-color: red; 17. } 18. #box1 { 19. width: 250px; 20. height: 250px; 21. margin: 0 auto; 22. background-color: blue; 23. } 24. #box2 { 25. position: absolute; 26. width: 200px; 27. height: 200px; 28. left: 100px; 29. top: 100px; 30. background-color: orange; 31. } 32. </style> 33. </head> 34. <body> 35. <div id="box"> 36. <div id="box1"> 37. <div id="box2"></div> 38. </div> 39. </div> 40. <!-- 此时box2是相对于box进行定位，在节点关系中box2的parentNode才是box1 --> 41. </body> 42. </html> |

此时box2相对于box进行定位

如何通过box2得到定位父元素呢？

|  |
| --- |
| 1. console.log(box2.offsetParent); |

结果：

|  |
| --- |
|  |

## 2.2 定位值

元素的定位值也是经常要获取的值， 所以dom也提供了相应的属性来获取

快速获取元素的定位值的方式：offsetLeft offsetTop

举例：

|  |
| --- |
| 1. // 获取box2的定位值 2. console.log("距离自己定位父元素的左边距离是:" + box2.offsetLeft); // 距离自己定位父元素的左边距离 3. console.log("距离自己定位父元素的上边距离是:" + box2.offsetTop); // 距离自己定位父元素上边的距离 |

结果：

|  |
| --- |
|  |

该属性存在兼容性问题：

在高级浏览器中： 是从子元素的边框外到父元素的边框内

在IE中： 是从子元素的边框外到父元素的边框外 也就是说多算了一条父元素的边框

在IE中输出定位值：

|  |
| --- |
|  |

## 2.3 jquery中的定位值

使用方式：

$(dom).position() 返回值是一个对象， 对象中包含left就是dom的定位left 值 同时包含top就是dom的定位top值

执行代码:

|  |
| --- |
| 1. // 获取box2的定位值 2. var $box\_position = $("#box2").position(); 3. console.log($box\_position); |

输出结果：

|  |
| --- |
|  |

# 三、获取一个元素到页面的距离

|  |
| --- |
| 1. function offset(dom) { 2. /\* 3. 思路: 4. 获取元素的定位父元素，再获取距离 5. 再获取元素的定位父元素的定位父元素， 再获取距离 6. 再获取元素的定位父元素的定位父元素的定位父元素， 直到终点 => body元素 距离相累加 7. \*/ 8. // for (, dom != document.body; dom = dom.offsetPatent) 9. // 获取浏览器的信息 10. var str = window.navigator.userAgent; 11. // 定义变量简化书写 12. var isIE8 = null; 13. // 检测浏览器信息 14. if (str.indexOf("MSIE 8.0") === -1) { 15. // console.log("不是IE8"); 16. isIE8 = false; 17. } else { 18. // console.log("是IE8"); 19. isIE8 = true; 20. } 21. // 定义对象 22. // 一开始不应该加上自己的边框， 23. var result = { 24. top: dom.offsetLeft, 25. left: dom.offsetTop 26. } 27. // 使用while循环 28. while (dom != document.body) { 29. // console.log(dom); 30. dom = dom.offsetParent; 31. // 判断是不是IE8 32. if (isIE8) { 33. // 说明是IE8 不需要加上父元素的边框了 34. result.left += dom.offsetLeft; 35. result.top += dom.offsetTop; 36. } else { 37. // 说明不是IE8 需要额外加上复元素的边框 38. result.left += dom.offsetLeft + dom.clientLeft; 39. result.top += dom.offsetTop + dom.clientTop; 40. } 41. } 42. // 返回对象 43. return result; 44. } |

# 下午复习：

快捷尺寸：

原生:

clientWidth clientHeight: 包含的是content + padding区域

offsetWidth offsetHeight: 包含的是content + padding + border区域

clientLeft clientTop : 包含的元素的左边框宽度， 上边框的宽度

jquery:

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

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

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

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

元素的位置和距离：

定位父元素：dom.offsetParent

定位值：

dom.offsetLeft: 表示距离自己定位父元素的左边距离

dom.offsetTop: 表示距离自己定位父元素的上边距离

jquery中的定位值：

$(dom).position()

返回的是一个对象， 对象中包含dom的left定位值 和top定位值

jquery中的offset:

$(dom).offset()

返回一个对象， 对象中包含元素到页面的左边距离和上边距离

# 四、页面卷动值

页面卷动值指的是 在页面中，视口上面看不到的部分， （被卷起来的部分）

获取页面卷动的方式：

document.documentElement.scrollTop 新版本的chrome 火狐 IE 支持

document.body.scrollTop 老版本的chrome或者一些其他的浏览器支持的

# IE中的透明度

高就浏览器中：

设置透明度： opacity[0-1]

IE中：

filter: alpha(opacity=[0~100])

# onscroll事件

当页面卷动的时候会触发该事件

在一个页面中，视口是一定的，当页面的高度比视口高的时候，此时页面将出现滚动条，这个时候当我们改变滚动条位置的时候，会触发onscroll事件。

触发该事件的条件: 改变页面卷动值

触发该事件的条件有很多：

比如：鼠标滚轮滚动的时候，键盘上空格键， 键盘上的PgUp 和 PgDn， 键盘上的上下按键

绑定方式：

|  |
| --- |
| 1. // 给docuemnt.body添加scroll事件 2. document.body.onscroll = function() { 3. console.log(123); 4. } |

当鼠标滚轮滚动的时候：

|  |
| --- |
|  |

# 六、节流

onscroll事件是一个高频率的事件，如果将代码放入事件中，代码将会将会被执行很多次，很浪费，

为了避免代码执行多次，我们定义一个延时器，将要执行的代码放入延时器中，在频繁执行该事件的时候，先清除之前的延时器，再开启一个新的延时器，从而保证代码只执行一次。

|  |
| --- |
| 1. // 获取元素 2. var $box = $("#box"); 3. // 定义一个定时器 4. var timer = null; 5. // 给document添加scroll事件 6. document.onscroll = function() { 7. console.log(123); 8. // console.log("页面滚动了"); 9. // 如果页面卷动值大于或者等于2000 让box出现 10. if (document.documentElement.scrollTop >= 2000) { 11. // console.log("box该出现了"); 13. // 清除之前的定时器 14. clearTimeout(timer); 15. // 赋值定时器 16. timer = setTimeout(function() { 17. $box.animate({opacity: 1}, 2000, function() { 18. console.log("出现了"); 19. }) 20. }, 200) 21. } 22. } |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |