爱创课堂前端培训

# javascript进阶

第2天课堂笔记（本课程共5天）

班级：北京前端训练营27期

讲师：蔡久明

日期：2019年10月10日

爱创课堂官网 ：[www.icketang.com](http://www.icketang.com)

目录

**[一、 call与apply](#_Toc22662_WPSOffice_Level1)** **[2](#_Toc22662_WPSOffice_Level1)**

[复习：](#_Toc14725_WPSOffice_Level2) [4](#_Toc14725_WPSOffice_Level2)

**[二、 事件流程](#_Toc20306_WPSOffice_Level1)** **[5](#_Toc20306_WPSOffice_Level1)**

[2.1 DOM0 级事件](#_Toc13632_WPSOffice_Level2) [6](#_Toc13632_WPSOffice_Level2)

[2.2 DOM2级添加事件](#_Toc22629_WPSOffice_Level2) [7](#_Toc22629_WPSOffice_Level2)

[2.3 高级浏览器的DOM2和DOM0的区别](#_Toc20314_WPSOffice_Level2) [8](#_Toc20314_WPSOffice_Level2)

[2.4 this 指向的问题](#_Toc27579_WPSOffice_Level2) [9](#_Toc27579_WPSOffice_Level2)

[2.5 IE的高级绑定事件](#_Toc27533_WPSOffice_Level2) [9](#_Toc27533_WPSOffice_Level2)

[2.6 IE的DOM2和DOM0的区别和执行顺序](#_Toc9000_WPSOffice_Level2) [10](#_Toc9000_WPSOffice_Level2)

[2.7 this指向](#_Toc30033_WPSOffice_Level2) [10](#_Toc30033_WPSOffice_Level2)

[2.8 兼容绑定事件](#_Toc31064_WPSOffice_Level2) [11](#_Toc31064_WPSOffice_Level2)

[2.9 点和中括号的区别](#_Toc3442_WPSOffice_Level2) [12](#_Toc3442_WPSOffice_Level2)

[2.10 移除事件](#_Toc388_WPSOffice_Level2) [12](#_Toc388_WPSOffice_Level2)

**[三、 事件对象](#_Toc8539_WPSOffice_Level1)** **[13](#_Toc8539_WPSOffice_Level1)**

[3.1 DOM0](#_Toc26148_WPSOffice_Level2) [13](#_Toc26148_WPSOffice_Level2)

[3.2 DOM2](#_Toc8591_WPSOffice_Level2) [14](#_Toc8591_WPSOffice_Level2)

**[四、 事件对象的属性和方法](#_Toc11754_WPSOffice_Level1)** **[14](#_Toc11754_WPSOffice_Level1)**

[4.1 offsetX和offsetY](#_Toc31026_WPSOffice_Level2) [15](#_Toc31026_WPSOffice_Level2)

[4.2 clientX和clientY](#_Toc449_WPSOffice_Level2) [15](#_Toc449_WPSOffice_Level2)

[4.3 screenX和screenY](#_Toc13965_WPSOffice_Level2) [16](#_Toc13965_WPSOffice_Level2)

[4.4 pageX和pageY（IE不兼容）](#_Toc17326_WPSOffice_Level2) [17](#_Toc17326_WPSOffice_Level2)

**[五、阻止事件冒泡](#_Toc5357_WPSOffice_Level1)** **[18](#_Toc5357_WPSOffice_Level1)**

**[六、 阻止默认行为](#_Toc1469_WPSOffice_Level1)** **[19](#_Toc1469_WPSOffice_Level1)**

[e.preventDefault();](#_Toc23124_WPSOffice_Level2) [19](#_Toc23124_WPSOffice_Level2)

[e.returnValue = false;](#_Toc10600_WPSOffice_Level2) [19](#_Toc10600_WPSOffice_Level2)

# call与apply

call与apply都能改变this的指向

所有的函数天生都可以调用call与apply

区别:

传参方式不同

fn.call(target, arg1,arg2,....) 参数直接写在要改变this指向的target之后

fn.apply(target, [arg1, arg2, .......]) 参数使用数组的方式进行传递

call是多个参数，不固定的。apply只需要两个参数

执行代码：

|  |
| --- |
| 1. /\* 2. 相同：call与apply都能改变this的指向 3. 区别：调用函数的时候，传入参数的方式不同 4. \*/ 5. function demo(a, b) { 6. // console.log(this, arguments); 7. console.log(a + b); 8. } 9. demo(1, 2); 10. var obj1 = { 11. name: 'obj1' 12. } 13. var obj2 = { 14. name: 'obj2' 15. } 16. demo.call(obj1, 1, 2); 17. demo.apply(obj2, [1, 2]); |

结果：

|  |
| --- |
|  |

# 复习：

节点

节点的属性：

节点的类型 nodeType

节点名称 nodeName

节点值 nodeValue

节点一共有12种

记住四种：

文档节点 9 #document null

注释节点 8 #comment 注释内容

文本节点 3 #text 文本内容

元素节点 1 大写标签名 null

js的节点操作

节点关系：

childNodes 获取所有的子节点

children 获取所有的子元素节点

firstChild 获取第一个子节点

firstElementChild 获取第一个子元素节点

lastChild 获取最后一个子节点

lastElementChild 获取最后一个子元素节点

nextSibling 获取下一个兄弟节点

previousSibling 获取上一个兄弟节点

parent 获取父元素节点

创建元素节点

document.createElement(标签名)

删除

removeChild

替换

replaceWith

添加节点

appendChild

insertBefore

克隆

cloneNode（bool）

jquery的节点操作

内部插入

append、appendTo

prepend、prependTo

外部插入

after、insertAfter

before、insertBefore

包裹

wrap、unwrap

wrapAll

wrapInner

删除

remove

empty

克隆

clone（bool）

替换

replaceWith、replaceAll

# 事件流程

事件的三要素：

事件源：发生事件的对象

事件类型：类型比如单击、双击、鼠标的移入、移除

事件处理程序: 触发事件之后做些什么，事件处理的函数

事件分为三个阶段：

事件捕获阶段：事件是由最外层的元素向目标元素传递的过程（IE不支持事件捕获）

事件处理阶段：事件正位于目标元素之上

事件冒泡阶段：有目标元素向外层元素传递的过程

事件捕获--》事件处理--》事件冒泡

事件的捕获和冒泡指的是当前元素的子元素跟当前元素拥有相同的事件，触发子元素的时候，当前元素对应事件发生在什么阶段

|  |
| --- |
|  |

## 2.1 DOM0 级事件

dom0级都属于冒泡阶段

结构：

|  |
| --- |
| 1. <div id="box1"> 2. <div id="box2"> 3. <div id="box3"></div> 4. </div> 5. </div> |

执行代码：

|  |
| --- |
| 1. var box1 = document.getElementById('box1'); 2. var box2 = document.getElementById('box2'); 3. var box3 = document.getElementById('box3'); 4. // 绑定事件 5. box1.onclick = function() { 6. this.style.backgroundColor = 'red'; 7. alert(1); 8. } 9. box2.onclick = function() { 10. this.style.backgroundColor = 'green'; 11. alert(2); 12. } 13. box3.onclick = function() { 14. this.style.backgroundColor = 'blue'; 15. alert(3); 16. } 17. document.body.onclick = function() { 18. alert('this is body'); 19. } |

结果：

|  |
| --- |
|  |

## 2.2 DOM2级添加事件

dom.addEventListener(type,fn,bool)

dom:要添加事件的对象

type：事件类型

fn：事件处理函数

bool：事件的触发阶段 默认值为false false表示冒泡阶段 如果为true则表示捕获阶段

结构：

|  |
| --- |
| 1. <div id="box1"> 2. <div id="box2"> 3. <div id="box3"></div> 4. </div> 5. </div> |

执行代码：

|  |
| --- |
| 1. // 获得元素 2. var box1 = document.getElementById('box1'); 3. var box2 = document.getElementById('box2'); 4. var box3 = document.getElementById('box3'); 5. // 事件捕获 6. // 绑定事件 7. box1.addEventListener('click',function() { 8. console.log('box1的事件捕获阶段'); 9. }, true); 10. box1.addEventListener('click',function() { 11. console.log('box1的事件冒泡阶段'); 12. }, false); 13. box2.addEventListener('click',function() { 14. console.log('box2的事件捕获阶段'); 15. }, true); 16. box2.addEventListener('click',function() { 17. console.log('box2的事件冒泡阶段'); 18. }, false); 19. box3.addEventListener('click',function() { 20. console.log('box3的事件冒泡阶段'); 21. }, false); 22. box3.addEventListener('click',function() { 23. console.log('box3的事件捕获阶段'); 24. }, true); |

结果：

|  |
| --- |
|  |

## 2.3 高级浏览器的DOM2和DOM0的区别

DOM0级事件：给同一个元素绑定相同类型的事件只能绑定一个，多个事件共存的时候，后面的会覆盖掉前面的

DOM2级事件：给同一个元素绑定相同类型的事件可以绑定多个，多个事件共存的时候，按照绑定的顺序依次执行

DOM0级事件和DOM2级事件可以共存，执行顺序跟绑定顺序一样

执行代码：

|  |
| --- |
| 1. // 获得元素 2. var box1 = document.getElementById('box1'); 3. var box2 = document.getElementById('box2'); 4. var box3 = document.getElementById('box3'); 5. /\* 6. DOM0级事件：给同一个元素绑定相同类型的事件只能绑定一个，多个事件共存的时候，后面的会覆盖掉前面的 7. DOM2级事件：给同一个元素绑定相同类型的事件可以绑定多个，多个事件共存的时候，按照绑定的顺序依次执行 8. DOM0级事件和DOM2级事件可以共存，执行顺序跟绑定顺序一样 9. \*/ 10. // 事件捕获 11. // box1.onclick = function() { 12. // console.log('DOM0的box1---1'); 13. // } 14. box1.addEventListener('click',function() { 15. console.log('DOM2的box1---1'); 16. }); 17. box1.addEventListener('click',function() { 18. console.log('DOM2的box1---2'); 19. }); 20. box1.onclick = function() { 21. console.log('DOM0的box1----2'); 22. } 23. box1.addEventListener('click',function() { 24. console.log('DOM2的box1---3'); 25. }); 26. box1.addEventListener('click',function() { 27. console.log('DOM2的box1---4'); 28. }); |

结果：

|  |
| --- |
|  |

## 2.4 this 指向的问题

指向绑定事件的对象

执行代码：

|  |
| --- |
| 1. box1.addEventListener('click',function() { 2. console.log('box1的事件捕获阶段'); 3. console.log(this); 4. }, true); 5. box1.addEventListener('click',function() { 6. console.log('box1的事件冒泡阶段'); 7. console.log(this); 8. }, false); 9. box2.addEventListener('click',function() { 10. console.log('box2的事件捕获阶段'); 11. console.log(this); 12. }, true); 13. box2.addEventListener('click',function() { 14. console.log('box2的事件冒泡阶段'); 15. console.log(this); 16. }, false); 17. box3.addEventListener('click',function() { 18. console.log('box3的事件冒泡阶段'); 19. console.log(this); 20. }, false); 21. box3.addEventListener('click',function() { 22. console.log('box3的事件捕获阶段'); 23. console.log(this); 24. }, true); |

结果：

|  |
| --- |
|  |

## 2.5 IE的高级绑定事件

IE不支持事件捕获，只有事件冒泡

dom.attachEvent(ontype, fn);

dom:要绑定事件的对象

type:事件类型

fn:处理函数

执行代码：

|  |
| --- |
| 1. // 获得元素对象 2. var div = document.getElementsByTagName('div')[0]; 3. // dom0的绑定事件 4. // div.onclick = function() { 5. // console.log('dom0事件----1'); 6. // } 7. // div.onclick = function() { 8. // console.log('dom0事件----2'); 9. // } 10. // dom2级事件绑定 11. div.attachEvent('onclick', function() { 12. console.log('dom2事件----1'); 13. }); 14. div.attachEvent('onclick', function() { 15. console.log('dom2事件----2'); 16. }); 17. // dom0的绑定事件 18. div.onclick = function() { 19. console.log('dom0事件----1'); 20. } 21. div.attachEvent('onclick', function() { 22. console.log('dom2事件----3'); 23. }); 24. div.attachEvent('onclick', function() { 25. console.log('dom2事件----4'); 26. }); |

结果：

|  |
| --- |
|  |

## 2.6 IE的DOM2和DOM0的区别和执行顺序

所有的DOM0，相同元素的同类型事件只能存在一个，后面的会覆盖掉前面

DOM2级，相同元素的同类型事件可以绑定多个

IE中的执行顺序：

1. 首先执行DOM0级事件
2. 按照绑定事件的倒序执行DOM2级事件

## 2.7 this指向

DOM0指向的是绑定事件的对象

DOM2指向window对象

执行代码：

|  |
| --- |
| 1. // 获得元素 2. var box1 = document.getElementById('box1'); 3. var box2 = document.getElementById('box2'); 4. var box3 = document.getElementById('box3'); 5. // 事件捕获 6. // 绑定事件 7. box1.attachEvent('onclick',function() { 8. console.log('box1的事件冒泡阶段'); 9. console.log(this); 10. }); 11. box2.attachEvent('onclick',function() { 12. console.log('box2的事件捕获阶段'); 13. console.log(this); 14. }); 15. box3.attachEvent('onclick',function() { 16. console.log('box3的事件冒泡阶段'); 17. console.log(this); 18. }); |

结果：

|  |
| --- |
|  |

## 2.8 兼容绑定事件

|  |
| --- |
| 1. // 获得元素对象 2. var div = document.getElementsByTagName('div')[0]; 3. /\* 4. 封装一个函数 5. 函数名： bindEvent 6. 调用方式：bindEvent(div,'click', function() { 7. this.backgroundColor = 'green'; 8. }) 9. 参数：bindEvent(dom, type, fn) 10. \*/ 11. function bindEvent(dom, type, fn) { 12. // 三类浏览器：高级浏览器 IE浏览器 杂牌浏览器 13. // 能力检测 14. if (dom.addEventListener) { 15. // 高级浏览器 16. dom.addEventListener(type, fn); 17. } else if (dom.attachEvent) { 18. // IE浏览器 19. dom.attachEvent('on' + type, function() { 20. // attachEvent绑定的事件this指向window 21. fn.call(dom); 22. }); 23. } else { 24. // 其他浏览器 25. dom['on' + type] = fn; 26. } 27. } 28. bindEvent(div,'click', function() { 29. this.style.backgroundColor = 'green'; 30. console.log(111); 31. }); |

## 2.9 点和中括号的区别

.不解析变量

[] 可以解析变量

执行代码：

|  |
| --- |
| 1. var obj = { 2. a: 1, 3. b: 2 4. } 5. var a = 'b'; 6. console.log(obj.a); // 1 7. console.log(obj[a]);// obj['b'] === obj.b = 2 8. console.log(obj['a']);// obj.a = 1 |

结果：

|  |
| --- |
|  |

## 2.10 移除事件

高级浏览器：

dom.removeEventListener(type, fn, bool)

type:事件类型

fn:要移除的函数（函数名）

bool：要移除的阶段 默认值为false 表示冒泡阶段，true表示捕获阶段

执行代码：

|  |
| --- |
| 1. // 获得元素对象 2. var div2 = document.getElementById('div2'); 3. var div1 = document.getElementById('div1'); 4. // 绑定事件 5. div2.onclick = function() { 6. console.log('div2被点击了'); 7. } 8. function demo1() { 9. console.log('div1被点击了----捕获'); 10. } 11. // div1绑定dom2事件 12. div1.addEventListener('click', demo1, true); 13. div1.addEventListener('click', function() { 14. console.log('div1被点击了---冒泡'); 15. }, false); 16. // 移除事件 17. div1.removeEventListener('click', demo1, true); |

结果：

|  |
| --- |
|  |

IE浏览器：

dom.detachEvent(ontype, fn)

type:事件类型

fn:要移除的函数（函数名）

执行代码：

|  |
| --- |
| 1. // 获得元素对象 2. var div2 = document.getElementById('div2'); 3. var div1 = document.getElementById('div1'); 4. // 绑定事件 5. div2.onclick = function() { 6. console.log('div2被点击了'); 7. } 8. function demo1() { 9. console.log('div1被点击了----冒泡1'); 10. } 11. // div1绑定dom2事件 12. div1.attachEvent('onclick', demo1); 13. div1.attachEvent('onclick', function() { 14. console.log('div1被点击了---冒泡2'); 15. }); 16. // 移除事件 17. div1.detachEvent('onclick', demo1); |

结果：

|  |
| --- |
|  |

# 事件对象

当事件触发的时候，会产生一个对象，该对象记录着发生事件相关的详细信息。

## 3.1 DOM0

DOM0级的情况：

高级浏览器会自动传递事件对象，只需要用形参接受即可使用

IE浏览器不会自动传递事件对象，需要手动接收，接受的方式为 e = window.event

DOM0兼容处理

var e = e || event

执行代码：

|  |
| --- |
| 1. // DOM0级的事件对象 2. var div = document.getElementsByTagName('div')[0]; 3. /\* 4. DOM0级的情况： 5. 高级浏览器会自动传递事件对象，只需要用形参接受即可使用 6. IE浏览器不会自动传递事件对象，需要手动接收，接受的方式为 e = window.event 7. DOM0兼容处理 8. var e = e || event 9. \*/ 10. // 绑定事件 11. div.onclick = function(e) { 12. var e = e || event; 13. console.log(e); 14. } |

结果：

|  |
| --- |
|  |

## 3.2 DOM2

DOM2级都可以传递事件对象，可以使用形参接收，然后直接使用

执行代码：

|  |
| --- |
| 1. // DOM0级的事件对象 2. var div = document.getElementsByTagName('div')[0]; 3. /\* 4. DOM2级都可以传递事件对象，可以使用形参接收，然后直接使用 5. \*/ 6. // 高级浏览器 7. if (div.addEventListener) { 8. div.addEventListener('click', function(e) { 9. console.log(e); 10. }); 11. } else { 12. // IE 13. div.attachEvent('onclick', function(e) { 14. console.log(e); 15. }); 16. } |

结果：

|  |
| --- |
|  |

# 四、 事件对象的属性和方法

查看 ：

高级浏览器直接查看

IE浏览器需要使用for in 进行查看

|  |
| --- |
| 1. // IE 2. div.attachEvent('onclick', function(e) { 3. // console.log(e); 4. for (var i in e) { 5. console.log(i, ' : ', e[i]); 6. } 7. }); |

## 4.1 offsetX和offsetY

当前鼠标位于元素内的位置，但注意受子元素的影响

执行代码：

|  |
| --- |
| 1. console.log('offsetX:', e.offsetX, ' - offsetY:', e.offsetY); |

结果：

|  |
| --- |
|  |

## 4.2 clientX和clientY

鼠标位于视口的x值和y值

执行代码：

|  |
| --- |
| 1. console.log('clientX:', e.clientX, ' - clientY:', e.clientY); |

结果：

|  |
| --- |
|  |

## 4.3 screenX和screenY

距离屏幕左上角的x和y

执行代码：

|  |
| --- |
| 1. console.log('screenX:', e.screenX, ' - screenY:', e.screenY); |

结果：

|  |
| --- |
|  |

## 4.4 pageX和pageY（IE不兼容）

距离整个网页的顶点的x和y

|  |
| --- |
|  |

|  |
| --- |
|  |

# 五、阻止事件冒泡

高级浏览器使用：

方法：stopPropagation()

属性: cancelBubble = true cancelBubble 是否阻止冒泡 默认值为false

IE浏览器：

属性: cancelBubble = true cancelBubble 是否阻止冒泡 默认值为false

执行代码：

|  |
| --- |
| 1. // 获得元素对象 2. var box = document.getElementById('box'); 3. var carousel = document.getElementById('carousel'); 4. // 绑定事件 5. carousel.onclick = function() { 6. console.log('carousel'); 7. } 8. box.onclick = function(e) { 9. console.log('box'); 10. // 事件的兼容 11. var e = e || event; 12. // console.log(e); 13. if ( e.stopPropagation) { 14. // 高级浏览器 15. e.stopPropagation(); 16. } else { 17. // IE浏览器 18. e.cancelBubble = true; 19. } 20. } |

# 阻止默认行为

a链接在点击的时候，会自动跳转，元素标签的默认行为。表单中的提交按钮，当其被点击的时候，就会提交表单的数据，这也是其默认行为。

DOM0级的方式,阻止默认行为 return false

执行代码：

|  |
| --- |
| 1. <a href="http://www.baidu.com">百度一下</a> 2. <script> 3. // 获得元素 4. var a = document.getElementsByTagName('a')[0]; 5. // 绑定点击事件 6. a.onclick = function() { 7. alert('百度一下'); 8. // DOM0级的方式,阻止默认行为 9. return false; 10. } 11. </script> |

DOM2级阻止默认行为

高级浏览器：

e.preventDefault();

e.returnValue = false;

IE浏览器：

event.returnValue = false;

return false;

执行代码：

|  |
| --- |
| 1. <a href="http://www.baidu.com">百度一下</a> 2. <script> 3. // 获得元素 4. var a = document.getElementsByTagName('a')[0]; 5. // 绑定点击事件 6. if (a.addEventListener) { 7. // 高级浏览器 8. a.addEventListener('click', function(e) { 9. alert('百度一下'); 10. // 方法 11. e.preventDefault();// 阻止默认行为 12. // e.returnValue = false; 13. }); 14. } else { 15. // IE浏览器 16. a.attachEvent('onclick', function() { 17. alert('百度一下'); 18. // return false; 19. // 事件对象的属性 20. event.returnValue = false; 21. }); 22. } 23. </script> |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |