# JavaScript DOM

第11天课堂笔记（本课程共10天）

目录

[JavaScript DOM 1](#_Toc476950354)

[一、节点关系 2](#_Toc476950355)

[1.1 原生JS中的节点关系 2](#_Toc476950356)

[二、节点操作 4](#_Toc476950357)

[2.1 原生JavaScript节点操作 4](#_Toc476950358)

[3.1.1 createElement()和appendChild() 4](#_Toc476950359)

[2.1.2 insertBefore 5](#_Toc476950360)

[2.1.3 removeChild() 5](#_Toc476950361)

[2.1.4 replaceChild() 5](#_Toc476950362)

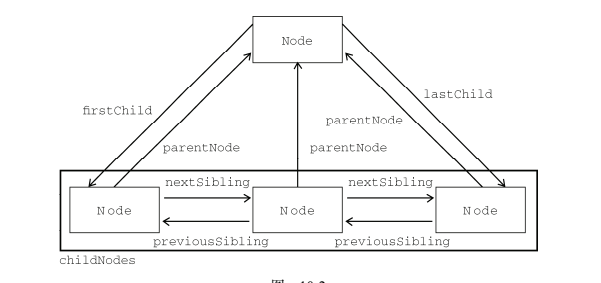
[2.1.5 cloneNode() 5](#_Toc476950363)

# 一、节点关系

## 1.1 原生JS中的节点关系

原生JS中提供的节点关系很少：

childNodes、firstChild、lastChild、parentNode、nextSibling、previousSibling



关于childNodes一定要记住，IE6、7、8和高级浏览器的不一致，高级浏览器视所有的换行为空文本节点，而IE6、7、8无视这个空文本节点。

|  |
| --- |
| 1. <div id="box"> 2. <p></p> 3. <p></p> 4. <p></p> 5. <p></p> 6. </div> |

|  |
| --- |
| 1. oDiv.childNodes.length; //chrome数值是9 IE6、7、8数值是4 |

为了解决兼容性问题（到底空文本算不算儿子，所以要用函数来解决）：

|  |
| --- |
| 1. //可以封装一个children函数，这个函数返回obj的所有节点儿子 2. function children(obj,num){ 3. var arr = []; //存放所有儿子 4. for(var i = 0 ; i < obj.childNodes.length ; i++){ 5. //寻找真正的HTML节点，不要考虑文本： 6. if(obj.childNodes[i].nodeType == 1){ 7. arr.push(obj.childNodes[i]); 8. } 9. } 10. //返回的是：如果用户传入了num，返回某一个儿子；如果没有num返回所有儿子 11. return num ? arr[num] : arr; 12. } |

同样的道理firstChild、lastChild也有兼容性问题。

这个兼容性问题一直存在，比如前一个兄弟：

|  |
| --- |
| 1. <div id="box"> 2. <p></p> 3. <p></p> 4. <p></p> 5. <p></p> 6. <p id="xiaoming">小明</p> 7. <p></p> 8. <p></p> 9. <p></p> 10. </div> |

兼容问题：

|  |
| --- |
| 1. xiaoming.previousSibling; //chrome是空文本节点； IE6、7、8就会认为是上一个p |

# 二、节点操作

HTML节点我们原来最多最多就是改改HTML属性，比如src属性改改；或者改改css样式，比如.style或者.css()。

现在的问题是，我们要增加节点、删除节点、移动节点、替换节点。

## 2.1 原生JavaScript节点操作

### 3.1.1 createElement()和appendChild()

|  |
| --- |
| 1. var ul = document.getElementsByTagName("ul")[0]; 2. //创建一个li标签，用变量oLi来表示。创建出来的节点不是任何节点的儿子， 3. //也就是说没有在DOM树上， 4. var oLi = **document.createElement**("li"); 5. oLi.innerHTML = "DDDD"; //改变这个节点里面的内容 6. //把新创建的节点，追加到DOM树上 7. **ul.appendChild(oLi);** |

创建节点的API：

|  |
| --- |
| 1. document.createElement() |

create创建，Element元素。接收一个参数，就是创建的标签是什么。

追加节点的API：

创建出来的节点不在DOM树上，所以就应该用appendChild()来添加到DOM树上：

|  |
| --- |
| 1. 父亲.appendChild(新儿子); |

appendChild()一般来说就是用来追加新创建的节点。如果你试图把页面上已经有的节点，appendChild()到别的地方，那么这个节点将移动。也就是说，同一个节点不可能在页面上的两个地方出现。

比如结构：

|  |
| --- |
| 1. <div id="box1"> 2. <p id="xiaoming">我是小明</p> 3. </div> 4. <div id="box2"> 5. </div> |

语句：

|  |
| --- |
| 1. box2.appendChild(xiaoming); |

将使xiaoming移动位置，从box1里面移动到box2里面。

事实上，工作的时候很少使用createElement。因为innerHTML足够好用，innerHTML也可以用来创建节点，**甚至效率比createElement还高**。

|  |
| --- |
| 1. box.innerHTML = "<ul><li>哈哈哈</li><li>做顿饭</li></ul>"; |

### 2.1.2 insertBefore

我们刚才说的appendChild是把新节点在父亲的所有儿子后添加，也就是说添加的节点就是父亲的最后一个儿子。

我们可以在任意一个位置添加节点。

|  |
| --- |
| 1. 父亲.insertBefore(新儿子,原有标杆儿子); |

会在原有标杆儿子之前插入。

如果想每次添加都在开头添加，那么就是：

|  |
| --- |
| 1. ul.insertBefore(oLi, lis[0]); |

lis这个变量是动态的，这次添加的li，下回就是lis[0]

### 2.1.3 removeChild()

|  |
| --- |
| 1. 父亲.removeChild(儿子); |

如果要自杀，也要找到爸爸

|  |
| --- |
| 1. this.parentNode.removeChild(this); |

### 2.1.4 replaceChild()

替换节点

|  |
| --- |
| 1. 父亲.replaceChild(新儿子, 老儿子); |

### 2.1.5 cloneNode()

克隆节点，参数true表示深复制，节点里面的所有内容一同复制。

复制之后的节点是个孤儿节点，所以也需要使用appendChild、inserBefore、replaceChild来添加上DOM树。

|  |
| --- |
| 1. ul.appendChild(lis[0].cloneNode(true)); |