JavaScript进阶

第1天课堂笔记

讲师：邵山欢

日期：2017年12月13日

# 一、这几天的课程

JavaScript进阶，讲解一些原生JS底层的事情，但是不要怕：

* jQuery有了简化的实现
* 工作中一定使用jQuery，所以**要注意练习接下来所有jQuery的案例**
* **原生JS是面试题的考点，听一耳朵，可以放到毕业前回看**。

第1天：DOM，原生JS和jQuery

第2天：事件

第3天：BOM、jQuery高级使用、插件

这几天过后就是10天的HTML5、CSS3、响应式移动端web制作，实际上是玩DOM的最后的课程。Ajax就不碰DOM了。

# 二、DOM元素的获得

## 2.1 什么是DOM

用自己话的回答：**让JS以操作节点树的方式操作HTML元素，而不是以操作字符串的方式**。

DOM = document object model文档对象模型，**实际上是HTML提供给JS调用的接口**。

<body>

<p>A</p>

<p>A</p>

<p>A</p>

<p>A</p>

<p>A</p>

<script type="text/javascript">

//得到那个元素

//改变它

</script>

</body>

DOM的基本操作（这也是今天课程枯燥无味的清单）：

* 得到DOM元素
* DOM节点关系
* 创建DOM元素
* 改变DOM元素的属性
* 移动DOM元素
* 删除DOM元素
* 克隆DOM元素

## 2.2 nodeType、nodeValue、nodeName（面试常考）

先来看下面的box盒子，内部有几个子节点？？

<div id="box">

<p>A</p>

<p>A</p>

<p>A</p>

</div>

得到子节点的属性是chileNodes，

<script type="text/javascript">

var childNodes = document.getElementById("box").childNodes;

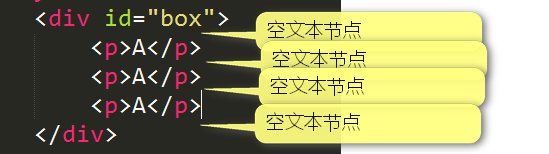
console.log(childNodes);

</script>

Chrome（高级浏览器）会认为box节点内部有7个子节点：

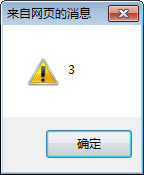


我们观察到空文本节点也被算作了节点。



浏览器兼容提醒：

在IE6、7、8中会认为空文本节点不是节点，所以IE8中会认为box中的子节点数量只有3个。



IE6、7、8对空文本节点是不是节点有歧义，**但是如果有文字，IE6、7、8和Chrome将都认为是节点**。

<div id="box">

字

<p>A</p>

字

<p>A</p>

字

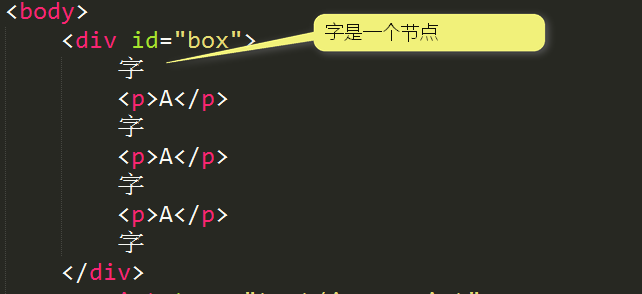
<p>A</p>

字

</div>



我们有一个属性叫做nodeType，表示节点的类型：



<script type="text/javascript">

var childNodes = document.getElementById("box").childNodes;

alert(childNodes[0].nodeType); //弹出3，说明文字节点的节点类型是3

alert(childNodes[1].nodeType); //弹出1，说明元素节点的节点类型是1

</script>

手册：<https://developer.mozilla.org/zh-CN/>



来自：<https://developer.mozilla.org/zh-CN/docs/Web/API/Node/nodeType>

需要背诵的是：

|  |  |
| --- | --- |
| 1 | 元素节点 |
| 3 | 文本节点（注意IE8不认为空节点是文本节点） |
| 8 | 注释节点 |
| 9 | 根节点document |

证明根节点是9：

alert(document.nodeType); //9

注意document这个词在任何浏览器中表示所有的元素节点，包括了文档的DTD（第一行文档类型定义声明头）：

document



**我们有的时候需要使用html节点（真实的有用的根）**，此时必须这么写：

**document.documentElement**



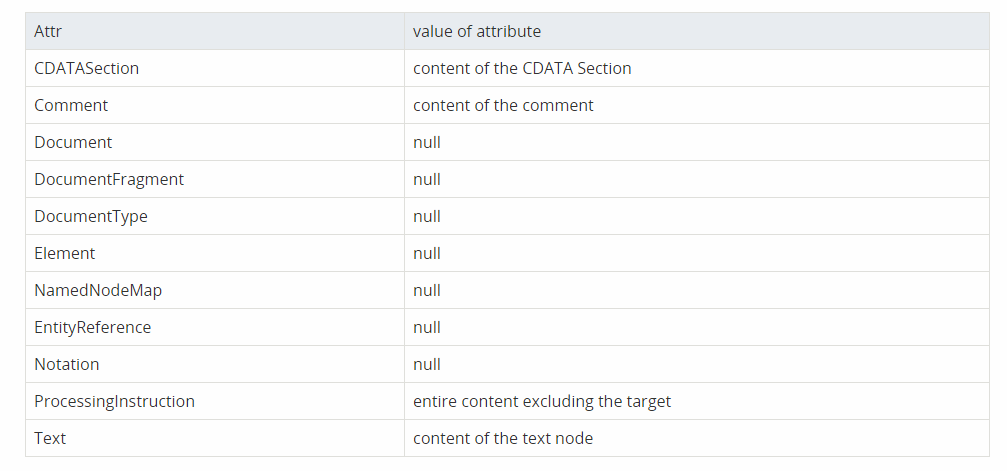
比如我们需要得到窗口的宽度：

alert(**document.documentElement.clientWidth**);

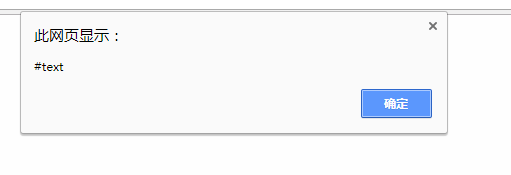
结论：

* document节点在开发中没有什么用，而根是document.documentElement。
* 事实上document是一个“虚构”的一个职位，里面封装了getElementById()、getElementsByTagName()方法等等。
* document节点的nodeType是9，document.documentElement的nodeType是1

文本节点、注释节点有nodeValue属性，值是它们文字的内容。其他节点是没有nodeValue属性的



节点有nodeName属性，如果是文本节点，则值是：



如果是元素节点，则值是：

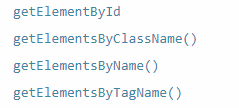


alert(childNodes[0].nodeName);

alert(childNodes[1].nodeName);

## 2.3 DOM元素的获得的方法

在原生JS中提供了4种得到元素节点（就是nodeType是1的那种节点）的方法：

 注意ES6中多了querySelector()、querySelectorAll()下一节讲。

**这些方法不是document的方法，而是任何nodeType是1的节点的方法（document.documentElement没有这个方法）**。所以可以连续打点调用：

<body>

<div id="box">

<p>

<span>0</span>

<span>1</span>

<span>2</span>

</p>

</div>

<script type="text/javascript">

document.getElementById("box").getElementsByTagName("p")[0].getElementsByTagName("span")[1].style.background = "red";

</script>

</body>

其中IE6兼容的只有两个：getElementById()、getElementsByTagName()，剩下两个是IE8开始兼容。

你们不知道的是，对于表单元素，DOM提供了快捷入口：

<form action="" **name="myForm"**>

<input type="text" **name="name"**/>

<input type="radio" **name="sex"** value="男"/>

<input type="radio" **name="sex"** value="女"/>

<input type="text" **name="age"**/>

</form>

只有表单（form标签），**并且必须加上name属性**，和里面的所有控件，都要加上name属性，可以快速用document打点获得。

<script type="text/javascript">

console.log(**document.myForm**)

console.log(**document.myForm.name**)

console.log(**document.myForm.sex**)

console.log(**document.myForm.age**)

</script>

## 2.4 jQuery做了什么

$()简化了节点的获得，原理是正则表达式。也就是说：

$(“#box p span”)

实际上jQuery在**用正则**识别了你想要ID为box里面的p标签里面的span。还得用递归。

2013年考拉老师还在班级中讲过“选择框架”，但是现在工作的时候不要求。

|  |
| --- |
| 实际上ES6中提出了新的两个方法：  querySelector();通过选择器选择符合条件的第一个节点  querySelectorAll(); 通过选择器选择符合条件的所有节点。 |

<body>

<div id="box">

<p>

<span>0</span>

<span>1</span>

<span>2</span>

</p>

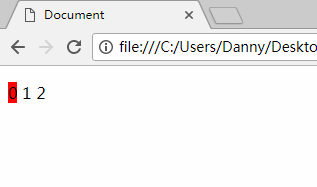
</div>

<script type="text/javascript">

**document.querySelector("#box p span").style.background = "red";**

</script>

</body>



<body>

<div id="box">

<p>

<span>0</span>

<span>1</span>

<span>2</span>

</p>

</div>

<script type="text/javascript">

var spans = **document.querySelectorAll(**"#box p span");

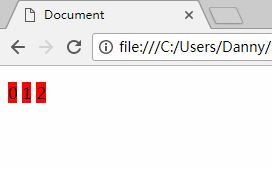
for(var i = 0 ; i < spans.length ; i++){

spans[i].style.background = "red";

}

</script>

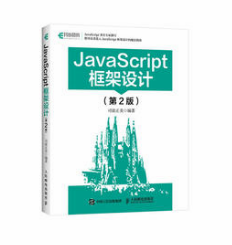
</body>



jQuery3中使用了querySelectorAll()来得到元素，所以不再使用正则表达式来分析词法了。

jQuery中的选择器算法（正则和递归）叫做sizzle，在github上开源了。

司徒正美也写了书，介绍了正则和递归选择元素的方法：



# 三、DOM元素的节点关系

## 3.1 childNodes、firstChild、lastChild、previousSibling、nextSibling、parentNode

|  |
| --- |
| childNodes 子节点  firstChild 第0号子节点  lastChild 最后的子节点  previousSibling 前一个兄弟  nextSibling 后一个兄弟  parentNode 父节点 |

原生JS提供的都是属性，不是方法，比如previusSibling类似jQuery中的prev()。

previousSibling表示上一个兄弟，**注意空文本节点也是节点**，**所以要连续两次**previousSibling才能找到真正的上一个兄弟元素节点。

<body>

<div>

<p></p>

<p></p>

<p id="xiaoming">小明</p>

<p></p>

<p></p>

</div>

<script type="text/javascript">

var xiaoming = document.getElementById("xiaoming");

**xiaoming.previousSibling.previousSibling**.style.background = "red";

**xiaoming.nextSibling.nextSibling**.style.background = "red";

</script>

</body>

几个轮子函数，这些轮子都被jQuery封装了，所以工作没用！就是面试用！

1. 如何得到上一个真正的元素节点

//找它前面的真正的元素节点兄弟

function finePreviousElementSibling(o){

while(o = o.previousSibling){

if(o.nodeType == 1) return o;

}

}

1. 如何得到所有兄弟

function findAllSiblings(o){

var arr = [];

var \_o = o;

//while语句有有一个特点，赋值语句可以写在条件圆括号中

//等号右边一旦是假性的东西（比如false、0、undefined、null、"")，此时将会停止循环

while(o = o.previousSibling){

if(o.nodeType == 1) arr.push(o);

console.log(\_o)

}

//恢复初始的o

o = \_o;

while(o = o.nextSibling){

if(o.nodeType == 1) arr.push(o);

}

return arr;

}

3）如何得到真元素儿子

function findAllElementChild(o){

var childNodes = o.childNodes;

var arr = [];

for(var i = 0 ; i < childNodes.length ; i++){

if(childNodes[i].nodeType == 1){

arr.push(childNodes[i]);

}

}

return arr;

}

## 3.2 jQuery做了什么

* jQuery丰富了节点关系：children()、find()、siblings()、prev()、prevAll()、next()、nextAll()、parent()、parents()、closest()、parentUntil()。
* 并且jQuery得到的都是nodeType为1的真的元素节点。

# 四、DOM元素的创建

## 4.1 原生中JS节点的创建createElement()、appendChild()、insertBefore()

* JS中创建的节点用createElement()来创建，圆括号中写”p”、”div”等等标签名，表示创建这样的节点。
* 创建的节点是孤儿节点，必须使用appendChild()或者inserBefore()来上树。

|  |
| --- |
| 三步走：   1. 创建节点 ： var nli = **document.createElement(“li”);** 2. 更改内容 ： **nli.innerHTML = “东西”;** 3. 上树 ： 两种方法：**父亲.appendChild()** 或者 **父亲.insertBefore(nli , 标杆)** |

<body>

<button id="btn">生孩子</button>

<ul id="myList">

</ul>

<script type="text/javascript">

var btn = document.getElementById("btn");

var myList = document.getElementById("myList");

btn.onclick = function(){

//创建一个节点

**var nli = document.createElement("li");**

//给节点加内容

**nli.innerHTML = Math.random();**

//创建的节点是孤儿节点

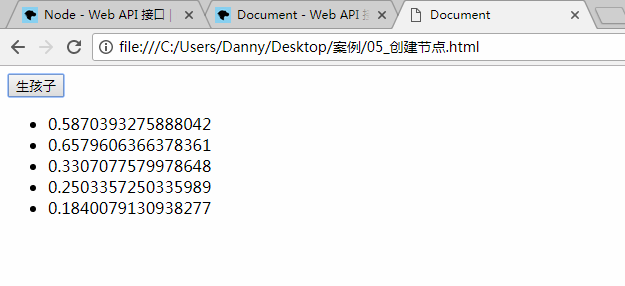
//我们可以使用appendChild来上树，appendChild表示“追加子节点”

**myList.appendChild(nli);**

}

</script>

</body>



在现有儿子之后插入是appendChild();

在现有儿子之前插入是insertBefore(); insertBefore实际上要选择一个标杆，插入到这个标杆的前面，但是所有程序员都习惯把标杆设为第一个子元素。在第一个子元素之前插入，不就是插入到现有儿子之前。

myList.insertBefore(nli , myList.firstChild);

nli 是要插入的孤儿节点

myList.firstChild是标杆

## 4.2 日历

我们要在页面上呈递每行7个格，6行的月历。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 周日 | 周一 | 周二 | 周三 | 周四 | 周五 | 周六 |
| 26 | 27 | 28 | 29 | 30 | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 1 | 2 | 3 | 4 | 5 | 6 |

决定一个月份的形态有三要素：

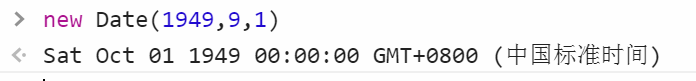
1. 本月1号星期几；
2. 本月有几天；
3. 上个月有几天；

快速得到他们，自己推敲去，别问老师了，求你们了。

年月日创建日期对象，月份从0开始。

共和国诞生日期就是：

new Date(1949,9,1)



得到这个日期对象的星期：

new Date(1949,9,1).getDay()



0表示星期天

1表示星期一

……

6表示星期六

我们来得到本月有几天，同学们肯定想到的是要判断闰年，因为如果是1、3、5、7、8、10、12月肯定是31天，4、6、9、11肯定是30天，2月要看是不是闰年来决定是28天还是29天。

老师告诉大家一个奇淫技巧，千万别问，先研究研究。

一个日期对象创建之后（用new Date()来创建之后）可以减去一个数字，减去的数字表示毫秒数。

得到一个毫秒数，表示这个日期距离1970年元月1日凌晨0点0分0秒000毫秒的毫秒数，叫做时间戳。

比如今天2017年12月13日，减去1毫秒得到昨晚的23:59:59 999毫秒



然后这个时间戳可以再次放到new Date()得到新的时间。

**所以本月一共几天，就是下个月1号减去1毫秒所在的天**。

new Date(new Date(2017 , 12 , 1) - 1).getDate()

上个月共几天，就是本月1号减去1毫秒所在的天。



袁培哲老师给大家的一个新方法：

**每月的0号就是上月的最后一天：**

new Date(year , month , 0).getDate();

## 4.3 innerHTML也可以用来创建元素节点

我们发现docuemnt.createElement()一次性只能创建一个节点

<div>

<p></p>

<div>

<ul>

<li></li>

<li></li>

<li></li>

<li></li>

</ul>

</div>

</div>

innerHTML也可以用来创建节点。

有一个工具，可以将复杂DOM结构瞬间变为join字符串形式：

<http://www.css88.com/tool/html2js/>

## 4.4 jQuery中的节点创建

jQuery的$()函数也可以肩负创建节点。

<script type="text/javascript" src="js/jquery-3.2.1.min.js"></script>

<script type="text/javascript">

$("button").click(function(){

**$("<li>" + Math.random() + "</li>")**.appendTo("#myList");

});

</script>

$()函数中如果没有<>就是选择元素，$()会认为你传入了一个选择器，要求它选择现有东西；

如果里面有<>，表示命令它创建元素，创建孤儿节点。

$(“li”) //选择所有li

$(“<li></li”> //创建孤儿节点li

jQuery中，上树有特别多种方法，有主动、被动的写法。

|  |
| --- |
| appendTo() ： 孤儿.appendTo(老爸),在现有儿子末尾  append() ： 老爸.append(孤儿) ,在现有儿子末尾  prependTo() ： 孤儿.prependTo(老爸)，在现有儿子之前  prepend() ： 老爸.prepend(孤儿) ，在现有儿子之前  after() ： 现有节点.after(孤儿)，在现有节点之后添加兄弟  insertAfter(); ： 孤儿.insertAfter(现有节点) ，在现有节点之后添加兄弟  before() : 现有节点.before(孤儿) ，在现有节点之前添加兄弟  insertBefore (); ： 孤儿. insertBefore (现有节点) ，在现有节点之前添加兄弟 |

比如：

$("#myList").append($("<li>" + Math.random() + "</li>"));

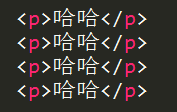
等价于

$("<li>" + Math.random() + "</li>").appendTo(“#myList”);

|  |  |  |
| --- | --- | --- |
|  | 原生JS | jQuery |
| 创建元素 | docuement.createElement  或者  innerHTML | $(“<li></li”) |
| 上树 | appendChild();  insertBefore(); | append  appendTo  prepend  prependTo  after  insertAfter  before  insertBefore |

jQuery中还有：

结构：



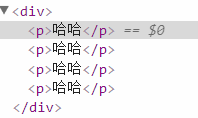
//表示给所有p每一个都创建一个div父亲：

$("p").wrap("<div></div>");



wrapAll表示给所有p创建一个div父亲：

$("p").wrapAll("<div></div>");



$()也可以用数组join的那种写法。

$(

[

'<div>',

' <p>我是p</p>',

' <div>',

' <ul>',

' <li>列表项</li>',

' <li>列表项</li>',

' <li>列表项</li>',

' <li>列表项</li>',

' </ul>',

' </div>',

'</div>'

].join("")

).appendTo();

jQuery的九九乘法表的：

<script type="text/javascript">

var $myTable = $("#myTable");

for(var i = 1 ; i <= 9 ; i++){

var $tr = $("<tr></tr>");

//创建内部9个td

for(var j = 1 ; j <= 9 ; j++){

$("<td>" + i + "\*" + j + "=" + (i \* j) +"</td>").appendTo($tr);

}

$tr.appendTo($myTable);

}

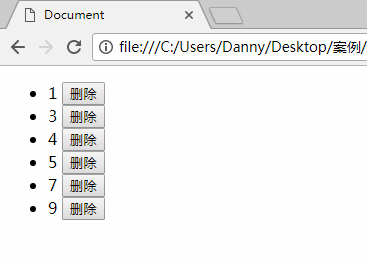
</script>

# 五、DOM元素的删除

## 5.1 .removeChild()

只有父亲能删除儿子，**也就是说如果你要自杀，找你爸把你杀死**。

父亲.removeChild(子元素)



原生JS中元素不能自杀，必须找爸爸把自己杀死。

<body>

<ul id="myList">

<li>1 <button>删除</button></li>

<li>2 <button>删除</button></li>

<li>3 <button>删除</button></li>

<li>4 <button>删除</button></li>

<li>5 <button>删除</button></li>

<li>6 <button>删除</button></li>

<li>7 <button>删除</button></li>

<li>8 <button>删除</button></li>

<li>9 <button>删除</button></li>

</ul>

<script type="text/javascript">

var buttons = document.querySelectorAll("button");

var myList = document.querySelector("#myList");

for(var i = 0 ; i < buttons.length ; i++){

buttons[i].onclick = function(){

//找爷爷把爸爸杀死

myList.removeChild(this.parentNode);

}

}

</script>

</body>

## 5.2 jQuery中的元素删除

**.remove();表示自己删除自己。**

上面的例子用jQuery实现很方便。

点击按钮之后，让按钮所在的li标签自杀：

<script type="text/javascript">

$("button").click(function(){

$(this).parent(“li”).remove();

});

</script>

**.empty()等价于.innerHTML = “”;**

//清空内部

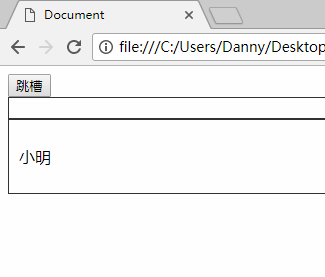
$("#myList").empty();

删除被包裹的p的div：

$("p").unwrap("div");

# 六、DOM元素的位置改变

原理很简单：一个DOM元素只能在DOM树有一个位置，**如果它再次上树，则会移动位置**。



<body>

<button id="btn">跳槽</button>

<div id="box1">

<p id="xiaoming">小明</p>

</div>

<div id="box2">

</div>

<script type="text/javascript">

var xiaoming = document.getElementById("xiaoming");

var box1 = document.getElementById("box1");

var box2 = document.getElementById("box2");

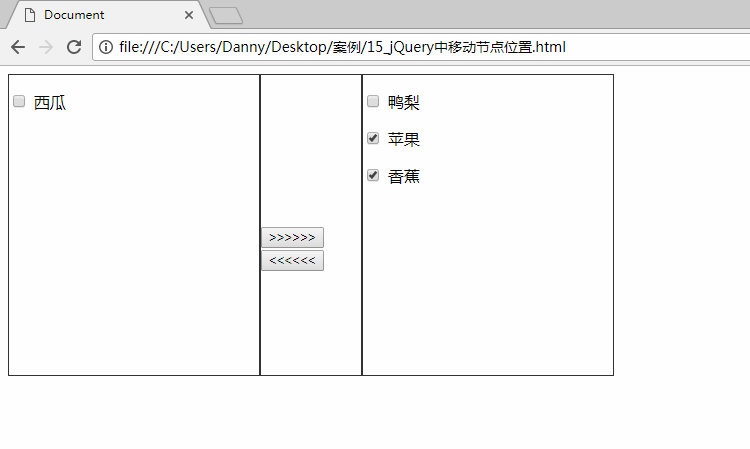
var btn = document.getElementById("btn");

**btn.onclick = function(){**

**box2.appendChild(xiaoming);**

**}**

</script>



<script type="text/javascript">

$("#btn1").click(function(){

//jquery能够特别快得到所有被勾选上的框框

$(".left input:checked").closest("p").appendTo(".right");

});

$("#btn2").click(function(){

//jquery能够特别快得到所有被勾选上的框框

$(".right input:checked").closest("p").appendTo(".left");

});

</script>

上面说的是位置改变，还有替换：

|  |
| --- |
| jQuery中有.replaceAll()、.replaceWith()方法自学。  <http://api.jquery.com/category/manipulation/dom-replacement/>  原生中，replaceChild()方法  replacedNode = parentNode.replaceChild(孤儿节点, oldChild); |

# 七、DOM元素的克隆

原生是cloneNode()方法，jQuery叫做clone()。

原生如果要克隆元素的所有内部元素，要加true参数。

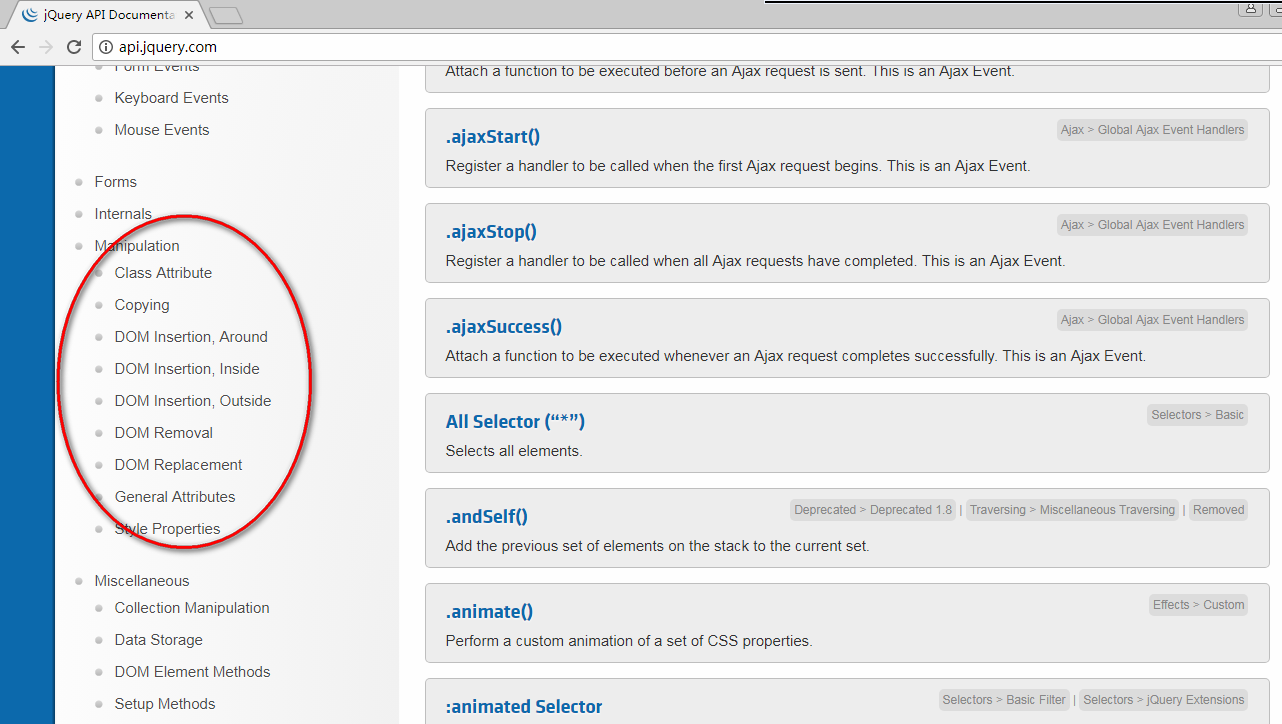
oDiv.cloneNode(true)

是孤儿节点，继续上树。

jQuery中不能更改是否要克隆内部元素，一定是克隆的。

$().clone().appendTo();

jQuery手册：



面试题：



