# JavaScript基础入门

第7天课堂笔记

# 目录

[JavaScript基础入门 1](#_Toc19065)

[目录 2](#_Toc7540)

[一、上次课知识复习 3](#_Toc2318)

[1.1 得到元素 3](#_Toc7884)

[1.2 更改HTML属性 3](#_Toc348)

[1.3 更改CSS样式 4](#_Toc9727)

[1.4 绑定事件监听 4](#_Toc27666)

[1.5 作业讲解 5](#_Toc29462)

[二、getElementsByTagName 6](#_Toc12994)

[2.1 概述 6](#_Toc5191)

[2.2 连续打点调用get 7](#_Toc29199)

[2.3 批量添加监听 8](#_Toc26082)

[2.4 对应和排他 9](#_Toc26202)

[三、计算后样式 10](#_Toc1740)

[3.1 高级浏览器和低级浏览器的不同写法 10](#_Toc24277)

[3.2 能力检测 12](#_Toc3532)

[3.3 关于opacity 12](#_Toc29217)

# 一、上次课知识复习

DOM ： Document Object Model文档对象模型，所有的HTML标签都是对象，非常方便得到、操作。不是操作字符串，而是操作对象。

## 1.1 得到元素

|  |
| --- |
| 1. document.getElementById(“id”); |

检索到就是HTML标签object，如果没有就是null。

大小写严格区分，IE6、7下有怪癖，name属性也被当做了id。

同一个页面不能有相同id，如果有用第一个。

## 1.2 更改HTML属性

得到元素之后能干嘛？能够更改HTML标签属性：

有什么，点什么。点语法。

setAttribute()、getAttribute()

|  |
| --- |
| 1. oImg.src = “images/2.jpg”; 2. oLink.href = “2.html”; |

点语法要避讳保留字、关键词，class要叫做className

|  |
| --- |
| 1. oDiv.className = “big”; |

id是不能更改。

使用点语法，不能得到自定义的属性，

|  |
| --- |
| 1. <div data-i=”8”></div> |

data-i属性自定义的属性，无法用点语法得到。

点语法得到的style是一个样式对象，可以继续打点，后面换成驼峰写法：

|  |
| --- |
| 1. oDiv.style.paddingLeft |

只能得到行内样式。

setAttribute()就不会避讳保留字、关键词了，

|  |
| --- |
| 1. oDiv.setAttribute(“class”,”big”); |

getAttribute()能够得到自定义的属性

|  |
| --- |
| 1. oDiv.getAttribute(“data-i”); |

getAttribute()得到style属性，得到字符串，不方便读取，同样的只能得到行内。

一般情况，我们更改HTML元素的属性，就用点语法。

## 1.3 更改CSS样式

读、设都是在行内，也就是说，现在的知识，是无法得到嵌入、外链样式表的样式的。

|  |
| --- |
| 1. oDiv.style.paddingLeft = “80px”; |

单位不能少。

|  |
| --- |
| 1. var a = 100; 2. oDiv.style.width = a + “px”; |

css怎么写，我们这个就怎么写：

|  |
| --- |
| 1. oDiv.style.backgroundColor = “rgb(111,222,123)”; |

这是因为css中：

|  |
| --- |
| 1. backgrond-color:rgb(111,222,123); |

## 1.4 绑定事件监听

先说一下什么是监听？计算机，如果发现一个盒子身上有一个onclick，就会在每个计算机时钟频率，都会检测用户是否点击了这个盒子。计算机这时候在监听这个盒子是否被点击。所以称为“事件监听”。

|  |
| --- |
| 1. oDiv.onclick = function(){ 2. } |

这个匿名函数，平时永远不会执行，必须等到事件的发生才执行。

现在我们就知道了函数执行的两种方法：

方法1： ()运算符调用,fun();

方法2： 绑定事件，事件发生了

如果绑定一个有名的函数，当做事件处理程序：

|  |
| --- |
| 1. function fun(){ 2. } 3. oDiv.onclick = fun; //绑定的是函数，而不是函数的执行，所以不加圆括号。 |

事件有哪些：

onclick、ondblclick、onmousedown、onmouseup、onmouseover、onmouseout、onfocus、onblur、onload。

|  |
| --- |
| 1. window.onload = function(){ 2. } |

当页面中所有HTML节点都加载完毕的时候执行，所有JS写在里面，就等于写在最后。

## 1.5 作业讲解

显示和隐藏：

|  |
| --- |
| 1. oTips.style.display = "block"; 2. oTips.style.display = "none"; |

信号量(semaphor)的思维模式，一定要养成。当多个按钮，共同操作一个对象的时候，马上就要想到信号量这种模式。信号量就不用添加行内样式了。

if语句有两种策略：先判断、后验收。

新玩家都喜欢先判断，老司机都是喜欢后验收。

先判断： 结构非常复杂，需要有一个else，并且<还不能写<=5

|  |
| --- |
| 1. if(index < 5){ 2. index++; 3. }else{ 4. index = 1; 5. } |

后验收：很清晰

|  |
| --- |
| 1. index++; 2. if(index > 5){ 3. index = 1; 4. } |

# 二、getElementsByTagName

## 2.1 概述

昨天我们学习的是得到元素的方法

|  |
| --- |
| 1. document.getElementById() |

得到一个元素。事实上，还有一个方法可以得到元素，并且得到的是多个元素：

|  |
| --- |
| 1. document.getElementsByTagName(); |

全线浏览器兼容的，得到元素的方法，就这两个：

document.getElementById() 通过id得到元素

document.getElementsByTagName(); 通过标签名得到元素们

以后还会学习更多的得到元素的方法：

document.getElementsByClassName(); 通过类名得到元素

document.querySelector(); 通过选择器得到元素

这些不是全线浏览器兼容。

getElementsByTagName就是通过标签名得到元素，得到的是页面上所有该种标签元素，得到的是数组，数组可以有下标，开始是0，最后一项是.length-1

|  |
| --- |
| 1. var ops = document.getElementsByTagName("p"); 2. ops**[0]**.style.backgroundColor = "rgb(111,222,123)"; 3. ops**[3]**.style.backgroundColor = "rgb(111,222,123)"; 4. ops**[ops.length - 1]**.style.backgroundColor = "rgb(111,222,123)"; |

既然是数组，就能通过循环语句批量控制，比如表格的隔行变色：

|  |
| --- |
| 1. var otrs = document.getElementsByTagName("**tr**"); 2. for(var i = 0 ; i **<** **otrs.length** ; i+=2){ 3. otrs[**i**].style.backgroundColor = "yellowgreen"; 4. } |

HTML标签从上到下，依次是0、1、2、3……

|  |
| --- |
| 1. <div> 2. <div> 3. <p>0</p> 4. <div> 5. <p>1</p> 6. </div> 7. <p>2</p> 8. </div> 9. <p>3</p> 10. </div> 11. <p>4</p> |

假如页面上只有一个p标签，那么得到的也是数组。

|  |
| --- |
| 1. var op = document.getElementsByTagName("p")**[0]**; 2. op.style.backgroundColor = "red"; |

## 2.2 连续打点调用get

先去选择一个HTML标签，然后选择这个HTML标签中的所有p标签：

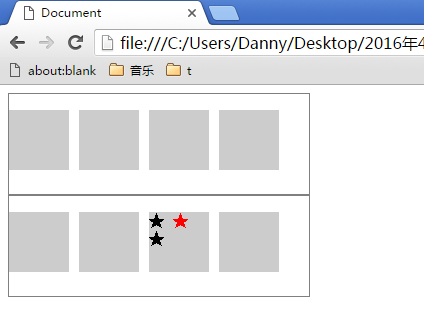
|  |
| --- |
| 1. var ps = document.getElementById("box2").getElementsByTagName("p"); |

不要多写一个document

|  |
| --- |
| 1. var ps = document.getElementById("box2").**~~document.~~**getElementsByTagName("p"); |

连续get可以很疯狂，但是如果是数组，一定要加上下标，才能变成对象：

|  |
| --- |
| 1. document.getElementsByTagName("div")**[1]**.getElementsByTagName("p")**[2]**.getElementsByTagName("span")**[1]**.style.color = "red"; |



举例：

表格的隔行变色

全选复选框

## 2.3 批量添加监听

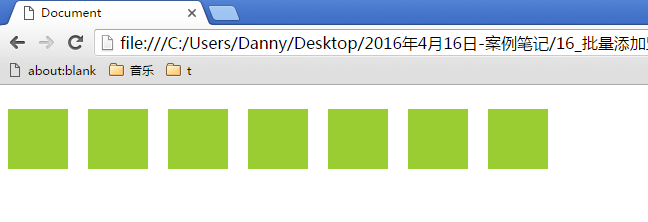
页面上的所有的p，现在都要有click事件监听。

就要用循环语句去加。

|  |
| --- |
| 1. for(var i = 0 ; i < ps.length ; i++){ 2. ps[i].onclick = function(){ 3. alert("你好"); 4. } 5. } |

for循环包裹添加监听的语句。

对序号的影响，闭包的影响又出现了：



|  |
| --- |
| 1. for(var i = 0 ; i < ps.length ; i++){ 2. ps[i].onclick = function(){ 3. alert("我是老" + i + "点我干嘛？！！"); 4. } 5. } |

所以每个盒子点击之后，都是7了，而不是我们预想的0、1、2、3、4、5、6。

闭包的影响：匿名函数定义的时候，已经记住了自己认识i，但是认识i不是表示把i的值赋值了一份记住，而是真真正正的认识了鲜活的、有生命的i。所以匿名函数执行的时候（就是事件发生时）i已经是7。

解决这个事情：

|  |
| --- |
| 1. for(var i = 0 ; i < ps.length ; i++){ 2. **(function(m){** 3. ps[**m**].onclick = function(){ 4. alert("我是老" + **m** + "点我干嘛？！！"); 5. } 6. **})(i);** 7. } |

除了IIFE之外，还有一种方法，就是强制添加属性的方法：

任何一个对象，我们都可以通过点语法来设置新的属性。

|  |
| --- |
| 1. for(var i = 0 ; i < ps.length ; i++){ 2. ps[i].**idx**= i; **//先不要着急设置监听，先编号** 4. ps[i].onclick = function(){ 5. alert("我是老" + **this.idx**+ "点我干嘛？！！"); **//自己的编号** 6. } 7. } |

## 2.4 对应和排他

对应：点击第一排的p，第二排对应的p变红：

|  |
| --- |
| 1. box2ps[this.idx].style.backgroundColor = “red”; |

排他：

|  |
| --- |
| 1. for(var i = 0 ; i < ps.length ; i++){ 2. ps[i].onmouseover = function(){ 3. //让所有的p都是灰色 4. for(var j = 0 ; j < ps.length ; j++){ 5. ps[j].style.backgroundColor = "#ccc"; 6. } 7. //让自己变红 8. this.style.backgroundColor = "red"; 9. } 10. } |

补充一个小知识点，ele.innerHTML属性就是HTML标签内部的文本。可以读，可以设。

# 三、计算后样式

## 3.1 高级浏览器和低级浏览器的不同写法

现在我们只能得到行内的样式，事实上DOM提供了可靠的API，得到计算后的样式。

W3C制定的标准API，所有现代浏览器（包括IE9，但不包括之前的版本）都实现了window.getComputedStyle()，该方法接收一个要进行样式计算的元素，并返回一个样式对象。样式对象提供了一个名为getPropertyValue()的方法，用于检索特定样式属性的计算样式。getPropertyValue方法接收css属性名称，而不是驼峰式的名称。getPropertyValue()可以不写，直接用方括号来检索属性也可以。

get得到，computed计算后，style样式

get得到，property属性，vaule值

比如：

|  |
| --- |
| 1. window.getComputedStyle(oDiv).getPropertyValue("width") |

现在我们要学习一个小知识，所有window对象的方法，都可以不用写window。至于window对象是什么，我们后面课程介绍。

比如，alert方法是window对象的方法，所以我们可以简写：

|  |
| --- |
| 1. window.alert("哈哈哈"); |

等价于

|  |
| --- |
| 1. alert("哈哈哈"); |

得到计算后的样式，可以直接使用getComputedStyle函数，而不用写window.

|  |
| --- |
| 1. getComputedStyle(oDiv).getPropertyValue("width") |

还要注意，引号里面不是驼峰：

|  |
| --- |
| 1. getComputedStyle(oDiv).getPropertyValue("**padding-left**") |

getPropertyValue看上去就像打它，看着就不爽，所以可以简写：

|  |
| --- |
| 1. getComputedStyle(oDiv)["padding-left"] |

计算后样式是综合的结果，就是这个元素此时的状态：

现在有css：

|  |
| --- |
| 1. background: url(images/songhuiqiao.jpg) no-repeat 10px 10px; |

虽然没有显式指定background-position，但是有值：

|  |
| --- |
| 1. getComputedStyle(oDiv)["background-position"] // 10px 10px; |

DOM提供给JS的API非常牛叉，一个元素此时的状态，完完全全可以被得到。好用的东西，一定不兼容。所以IE6、7、8不兼容getComputedStyle.getPropertyValue()的写法，另外一套写法：附加在元素身上的currentStyle属性，它表现和style点语法一样，使用驼峰式访问。

|  |
| --- |
| 1. oDiv.currentStyle.width |

现在要注意，它必须使用驼峰：

|  |
| --- |
| 1. oDiv.currentStyle.paddingTop |

并且不能被文字撑出高，得到”auto”这个值。颜色值在高级浏览器中是rgb()格式，低级浏览器中就是原样输出。

超级无敌大坑，可以不写点语法，可以使用方括号，但是里面也要写驼峰。

|  |
| --- |
| 1. oDiv.currentStyle["paddingLeft"] |

总结：

|  |
| --- |
| 1. 高级： 2. window.getComputedStyle(oDiv).getPropertyValue("padding-left"); 3. getComputedStyle(oDiv).getPropertyValue("padding-left"); 4. getComputedStyle(oDiv)["padding-left"]; 5. IE6、7、8： 6. oDiv.currentStyle.paddingLeft; 7. oDiv.currentStyle["paddingLeft"]; |

**练习理解的样式书写**

background-color:rgb(111,222,124);

oDiv.style.backgroundColor = "rgb(111,222,123)";

oDiv.onclick = function(){}

function fun(){

}

oDiv.onclick = fun;

window.onload = function(){}

oTips.style.display = "block";

oTips.style.display = "none";

document.getElementsByTagName("p");

document.getElementById("div");

getComputedStyle(oDiv).getPropertyValue("padding-left");

getComputedStyle(oDiv).["padding-left"];

oDiv.currentStyle.paddingLeft;

oDiv.currentStyle["paddingLeft"];

## 3.2 能力检测

IE不认识getComputedStyle视为错误

Chrome不认识currentStyle视为错误

所以，我们现在就要进行一个兼容性写法，新玩家会认为，要检测浏览器版本，如果版本是IE6、7、8那么……。

实际上，老司机都不这么做。我们不关心你的版本是什么，我只关心你的能力。

把某个方法往if语句里面做判断，就是能力检测。

|  |
| --- |
| 1. if(window.getComputedStyle){ 2. alert("我会getComputedStyle"); 3. }else{ 4. alert("我不会getComputedStyle，那四个啥？") 5. } |

所以使用能力检测，我们可以在不同浏览器中得到兼容性的写法，输出padding-left的值：

|  |
| --- |
| 1. if(window.getComputedStyle){ 2. alert(getComputedStyle(oDiv)["padding-left"]); 3. }else{ 4. alert(oDiv.currentStyle.paddingLeft); 5. } |

现在我们要在一个轮子，就是封装一个函数，这个函数接收两个参数，第一个是对象，第二个是属性名。

|  |
| --- |
| 1. fetchComputedStyle(obj,”padding-left”); 2. fetchComputedStyle(obj,”paddingLeft”); |

这个函数返回的是这个属性值的计算后的样式。更牛逼的是，我们无论用户输入的是驼峰还是非驼峰，都让这个函数鲁棒。

## 3.3 关于opacity

尽管IE8和早期版本不支持opacity，通过style.opacity或者ele.currentStyle.opacity属性取值时，返回的依然是opacity值，即使浏览器完全忽略了opatity值。这是一个好事儿，当我们能够保证opactiy、filter中设置的属性是一致的，则JavaScript读取opactiy值就算是兼容的。

|  |
| --- |
| 1. div.currentStyle.opacity //可以得到css清单中的opactiy里面写的值。而不是filter的值。 |

支持opacity的浏览器，总会将.6这种写法规范为0.6 。 而不支持opacity的浏览器则会返回原有的形式.6。这是一个小坑，进行比较的时候需要注意。

值差了100倍：

