# JavaScript进阶

第6天课堂笔记

# 目录

[JavaScript进阶 1](#_Toc19767)

[目录 2](#_Toc19263)

[一、JavaScript代码的组织 3](#_Toc22762)

[1.1 外链JS文件 3](#_Toc29078)

[1.2 页面中的JS与HTML 4](#_Toc6079)

[1.3 window.onload 5](#_Toc29995)

[二、外链JS文件策略 7](#_Toc19422)

[2.1 js文件中IIFE，HTML不需要调用 7](#_Toc7885)

[2.2 js文件中定义函数，在HTML页面上用()调用 8](#_Toc29977)

[2.3 js文件中定义类，HTML中用new来调用函数 9](#_Toc18519)

# 一、JavaScript代码的组织

JS代码放到哪里？

## 1.1 外链JS文件

外链JavaScript是一个组织JS的非常好的方法。

比如：

|  |
| --- |
| 1. <script type="text/javascript" **src=**"js/1.js"></script> 2. <script type="text/javascript" **src=**"js/2.js"></script> |

JS的执行顺序，就是引包顺序的先后顺序。

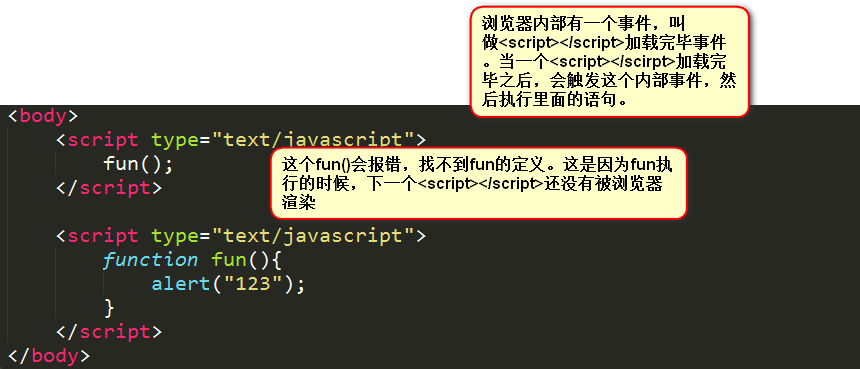
**包不能隔开作用域的**：



a能够正常输出100。JS文件（JS包)是不能隔开作用域的。**在JavaScript中，唯一能够隔开作用域的就是function**。

JS包里面定义的变量、函数，将在HTML外部失去变量声明的提升、函数声明头的提升。

**本质上，写在不同的<script></script>标签对儿里面的变量、函数，就已经失去变量声明的提升、函数声明头的提升。**



正因为如此，我们必须：

|  |
| --- |
| 1. <script type="text/javascript" src="1.js"></script>  **→ 这个包里面定义了函数fun** 2. <script type="text/javascript"> 3. fun(); **→ 执行fun的时候，浏览器已经认识了fun** 4. </script> |

## 1.2 页面中的JS与HTML

页面上会出现很多很多的特效，那么此时JS的摆放的位置，应该放到HTML结构的后面，并且习惯上紧跟HTML结构。这样做的好处是，我们就知道每个JavaScript的含义了，就知道这个JavaScript是负责上面HTML特效的。



## 1.3 window.onload

延迟加载，当页面中的所有元素加载完毕之后，触发这个事件。你要注意，什么叫所有元素加载完毕呢？页面上的img图片，都是一点一点通过HTTP传输过来，当所有的img都传输完毕之后，才触发window.onload事件。

所以，之前讲过chrome有个小问题，就是一个盒子如果是用img撑宽的，那么oDiv.clientWidth就是0。这是因为读取这个盒子的宽度的时候，这个img还没有加载完毕。解决方法就是写在window.onload里面：

|  |
| --- |
| 1. window.onload = function(){ 2. var box = document.getElementsByTagName("div")[0]; 3. alert(box.clientWidth); 4. } |

之前说过，DOM0级添加的事件，不能出现同名的。

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

以后出现的为准，所以页面上只能有一个window.onload。

但是，如果使用DOM2级，

|  |
| --- |
| 1. window.addEventListener(”load”,function(){},false); 2. window.addEventListener(”load”,function(){},false); |

就能够添加多个window.onload事件的处理函数。依次执行（attachEvent里面是反着的）。

jQuery中对应的事件叫做：

|  |
| --- |
| 1. $(document).ready(function(){ 3. }); |

jQuery中的所有事件，都是通过DOM2级的机制添加的，所以jQuery中所有的事件，都能够同名。下午再说。

window.onload在工作中是不允许使用的。因为这个会把JS提前，与自己的HTML结构分开了，所以还是应该一个结构、一个JS文件；一个结构、一个JS文件这么写。

但是，window.onload在应对几个东西的时候，有非常大的作用：

1） 表单验证

2） onscroll事件

3） ……

**所有的刚性的、事关穿不穿帮、事关安全的，都要写在window.onload里面。**

我们先学习一下表单事件，onsubmit事件表示表单试图提交的那个瞬间。这个事件只能加给form对象。当标签里面的submit按钮点击的时候，触发。

|  |
| --- |
| 1. //当表单提交的时候 2. form.onsubmit = function(){ 3. alert("哈哈啊，你要提交表单啦？我不让你提交"); 4. return false; //onsubmit事件里面，如果写了return false就不能提交表单了 5. } |

表单提交非常的“刚性”，事关网站的安全，所以一定一定要写在window.onload里面，不要单独写js文件，不要写在HTML结构后面。让浏览器第一时间知道form有一个onsubmit事件。

|  |
| --- |
| 1. window.onload = function(){ 2. var box = document.getElementsByTagName("div")[0]; 3. // 浏览器必须在第一时间知道，这个页面滚动条滚动，是有事件的。 4. window.onmousewheel = function(event){ 5. event.preventDefault(); 6. } 7. } |

# 二、外链JS文件策略

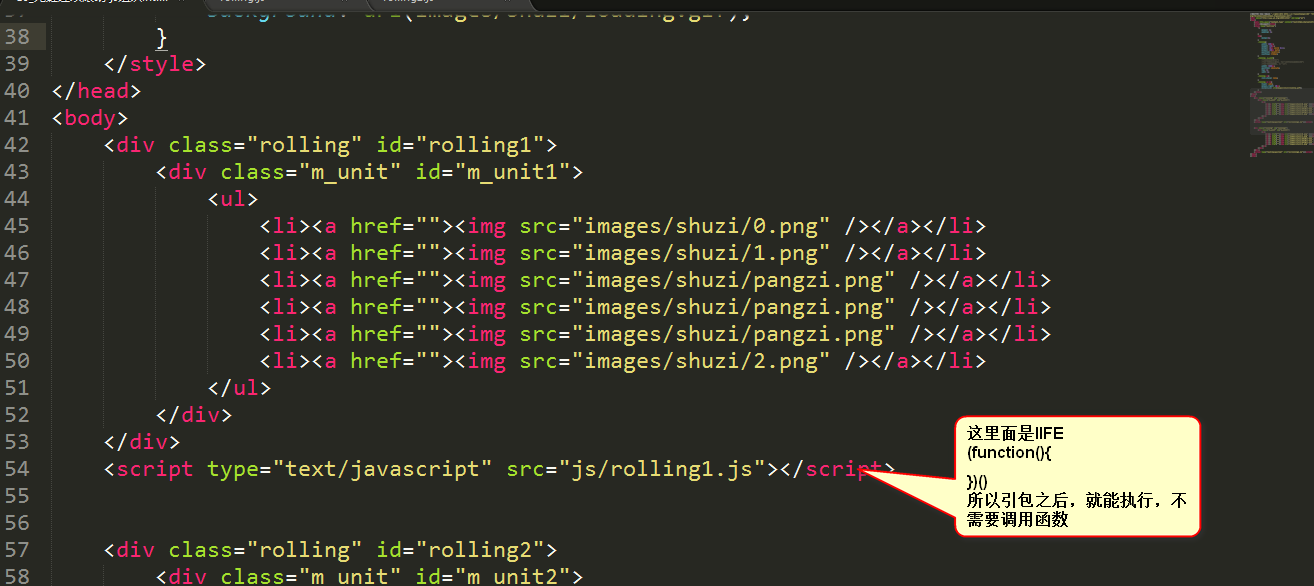
上午说了，一个js文件不是一个作用域，但是我们仍然希望一个js文件就是独立的作用域。所以，就需要把语句写在function里面。

没有任何的理由，将js文件里面的语句散着写，因为散着写作用域被污染的严重，多个效果会冲突。

必须把js里面的所有语句放到函数里面，有三种方法：

## 2.1 js文件中IIFE，HTML不需要调用

把js文件中的所有语句放到IIFE中，就能够关注作用域。并且HTML文件中不用调用函数。



IIFE是个好东西，能够管理住作用域，函数的闭包能够保护函数的每个变量与别人不冲突

没毛病，有瑕疵，不方便二次使用这个函数。当页面上有两个轮播图的时候，同样的业务，必须要定义两次函数。而函数天生是要复用的，所以IIFE有缺陷。

## 2.2 js文件中定义函数，在HTML页面上用()调用



这种调用函数的方式，就是用()来进行普通调用。没毛病，函数确实被复用了。

最近学过一种新的函数调用方式，就是用new来调用函数。实际上new就叫做组件开发。

我们现在来稍微了解一下JS的垃圾回收机制，来看下面的语句：

|  |
| --- |
| 1. function fun(){ 2. var a = 1; 3. a++; 4. a\*=9; 5. alert(a); 6. } 7. fun(); |

当函数运行的时候，a变量将开辟一小块内存空间，存入1这个数字。然后数值发生变化，弹出数字18。然后js发现a变量没有被任何的定时器、事件监听“持有”，所以为了清理内存，JS就把a当做垃圾了。就回收了这个空间。

你看下面的语句：

|  |
| --- |
| 1. function fun(){ 2. var a = 1; 3. document.onclick = function(){ 4. a++; 5. alert(a); 6. } 7. } 8. fun(); |

变量a被事件监听持有，所以fun虽然运行完毕了，但是浏览器不敢去清除a变量的空间。

所以每次点击页面，会弹出2、3、4、5、……。

## 2.3 js文件中定义类，HTML中用new来调用函数

轮播图是new出来的。实际上就叫做组件开发。组件开发没别的，无非就是代码组织的一种方式。不学组件开发，页面也嗷嗷做，用函数风格， 也没啥问题，但是逼格不高。现在越来越流行把代码做成组件。

轮播图，与打气球、挡板球、贪吃蛇比，太简单了！

杂碎知识点：

● 写在prototype里面的函数，对象的实例是可以调用的。但是一些函数是内部函数，我们只希望函数自己调用自己。不希望实例来调用，习惯上写\_开头命名。但是\_不能阻止外部实例调用，没事儿，这就是一个约定。

|  |
| --- |
| 1. Slider.prototype.**\_**findMyElemsByClassName = function(className){ 2. } |

加上\_的函数，说白了不向外暴露这个API。

● I’m so sorry，之前讲正则表达式没有说构造函数法和字面量法的区别。

字面量法，必须是正则表达式的字面量，用/ /来界定。里面不能有变量，不能有斩断链接。

但是，new RegExp()构造函数的语法：

|  |
| --- |
| 1. var reg = new RegExp(正则字符串,描述符字符串); |

比如：

|  |
| --- |
| 1. var reg = new RegExp("^1[\d]{10}$","g"); |

好处就是可以用字符串的斩断链接，来拼接字符串，使用变量：

|  |
| --- |
| 1. var a = "kaola"; 2. var reg = new RegExp("^" + a + "{1,2}$" , "g"); 3. reg.test("woaikaolahahaha") |

●组件开发有一个天大的好处，就是不用依赖任何的框架。我们不需要使用jQuery，也不需要使用运动框架。

运动框架的哲学是什么？就是把任何的运动都封装：

|  |
| --- |
| 1. animate(oDiv,{"top":700,"left":700},100); |

但是组件开发有一个思路，就是自己只管自己，不需要更多的复用。你想想看，传统轮播图中，能够运动的业务，就是left值。只涉及left值的改变，不涉及其他属性的改变，那为什么还要使用万能的运动框架呢？

组件的哲学和通用轮子不一样。通用轮子，要把所有浏览器、所有属性都要封装进来，注重复用、易用。组件开发只针对特定功能！！

● 当一个变量内部是一个setInterval、setTimeout的时候，那么这个变量被console.log输出的时候，变量的值是1。当定时器积累的时候，实际上值会变。不管怎么样，值正数。

这是一种非常简单的函数截流：

|  |
| --- |
| 1. if(this.timer){return;} |