JavaScript需知的七件事

JavaScript代码的启示

七步测试法找到正确的JavaScript解决方法

JavaScript的十个古怪之处和秘密

aScript的“七宗罪”

aScript动画计算详解

AJAX爬行算法的可搜索式动态信息

ery几个易混淆之处

jQuery 和 PHP GD 处理图片

jQuery 制作自己的书签

的 jQuery 插件模式

ry 插件清单：是否应该使用jQuery 插件?

THE
SMASHING
BOOK

众妙之门

JavaScript与jQuery技术精粹

[德] Smashing Magazine 著
吴达茹 芮鹏飞 译

人民邮电出版社
POSTS & TELECOM PRESS

众妙之门

JavaScript与jQuery技术精粹

[德] Smashing Magazine 著

吴达茄 芮鹏飞 译

人民邮电出版社

北京

图书在版编目（CIP）数据

众妙之门 : JavaScript与jQuery技术精粹 / 德国Smashing杂志著 ; 吴达茄, 芮鹏飞译. -- 北京 : 人民邮电出版社, 2013.8
ISBN 978-7-115-31811-4

Ⅰ. ①众… Ⅱ. ①德… ②吴… ③芮… Ⅲ. ①JAVA语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2013)第090241号

内 容 提 要

本书出自世界知名Web设计网站Smashing Magazine，其中的文章是来自全球顶级设计师的精华总结。全书共分为两大部分，第一部分阐述JavaScript的实战经验，共7章，内容涉及JavaScript初学者应掌握的知识，JavaScript代码复查的重要性，作者独创的七步测试法，JavaScript的十大秘密，如何避免在维护和移交代码时所发生的不必要麻烦，JavaScript动画教学，以及使用AJAX的关键技巧。第二部分介绍了jQuery的实战经验，共5章，内容涉及jQuery容易让人混淆的几个方面，如何使用jQuery和PHP GD处理图像，用jQuery制作书签，jQuery的插件模式，最后介绍了各种jQuery插件以及选择依据。

本书最大的价值在于其结合大量实例的生动方式，详细阐述了使用JavaScript和jQuery时应掌握的知识和技巧，以及作者通过实践掌握的各种秘诀，可帮助开发人员提升自身水平，向成功更近一步。相信广大读者读完这本书之后，一定会有一种相识恨晚的感觉。

◆ 著　[德] Smashing Magazine
译　吴达茄　芮鹏飞
责任编辑　赵　轩
责任印制　王　玮

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街14号
邮编　100061　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京隆昌伟业印刷有限公司印刷

◆ 开本：880×1230　1/32
印张：6.75
字数：214千字　2013年8月第1版
印数：1－3 000册　2013年8月北京第1次印刷
著作权合同登记号　图字：01-2012-7224号

定价：35.00元

读者服务热线：(010)67132692　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

对于网站开发设计人员而言，在面对选择解决方案时做出正确的决定并不容易。不论是在建立复杂的网站应用还是在改进网站的过程中，都会有很多前期解决方案可供选择，有时选择最合适的一款方案至关重要。本书着重讲述了在选择相应解决方案时务必要注意的事项，即是否稳定并易于定制、是否有实用性并易于理解、是否具有可维护性、兼容性，以及功能的可拓展性。

本书重点阐述了检验代码的重要性以及在执行 JavaScript 程序时需要避免的问题。所选择的解决方案应能符合较高的编码标准并能够剔除常见错误。本书将会告诉你如何获得此类解决方案。其中一部分内容介绍了利用专家系统审核代码并检查是否有其他方法解决当前问题。

在本书中，你将会熟悉 JavaScript 基本动画操作的黄金法则，了解 JSON 作为一种数据格式与 JavaScript 函数（数学、数组、字符串函数）共同存在，了解一些快捷符号等。当然，我们还会提到触发网站应用的 JS 事件、匿名函数的执行、模块模式、配置信息，以及与后台的交互及代码库的使用说明。对 AJAX 感兴趣的读者可以获得与动态可搜索内容相关的知识。在本书的后半部分，着重介绍了与 jQuery 有关的重要内容，能够帮助读者对 jQuery 的应用有更加透彻的认识。

——Andrew Rogerson，Smashing Magazine 编辑

版权声明

目　录

第一部分 JavaScript 基础篇

初学 JavaScript 需知的七件事

第 1 章

Christian Heilmann

我很早以前就开始编写 JavaScript 代码，很高兴看到这种语言在今天所取得的成功，能成为这个成功故事中的一部分我很开心。关于 JavaScript，我写过许多文章、章节以及一整本书，直到今天我仍在寻找新的东西。下文是一些我工作学习过程中激动时刻的记录，大家与其守株待兔，不如自己尝试去体会这种感受。

1.1　缩略标记

在创建对象和数组过程中可以使用缩略标记是我喜欢 JavaScript 的重要原因之一。过去，当我们需要创建一个对象时，我们会这样写：

```
var car = new Object();
car.colour = 'red';
car.wheels = 4;
car.hubcaps = 'spinning';
car.age = 4;
```

现在也可以写成

```
var car = {
 colour:'red',
 wheels:4,
 hubcaps:'spinning',
 age:4
}
```

这样写更加简洁，并且不用重复写对象名。现在，car 运行良好，但是如果使用了 invalidUserInSession 会怎样呢？这种标记法中主要的缩略标记是 IE，在第二个大括号前千万不要写逗号，否则你将会遇到麻烦。

另一个使用缩略标记的地方是定义数组。老的定义方法是这样的：

```
var moviesThatNeedBetterWriters = new Array(
 'Transformers','Transformers2','Avatar','Indiana Jones 4'
);
```

更简洁的版本是这样的：

```
var moviesThatNeedBetterWriters = [
 'Transformers','Transformers2','Avatar','Indiana Jones 4'
];
```

关于数组，另一个要注意的是没有所谓的关联数组。你会在很多代码中看到这样定义 car：

```
var car = new Array();
car['colour'] = 'red';
car['wheels'] = 4;
car['hubcaps'] = 'spinning';
car['age'] = 4;
```

这不是 Sparta，这是一种疯狂的行为——但不要为此而困扰。“关联数组”是一种令人困惑的对象命名方式。

另一种非常有意思的缩略标记方法叫做三重标记法。如下语句：

```
var direction;
if(x < 200){
 direction = 1;
} else {
 direction = -1;
}
```

用三重标记法可以写成：

```
var direction = x < 200 ? 1 : -1;
```

该条件为 true 时执行问号后的内容，否则执行冒号后的内容。

1.2 JSON 数据格式

在我发现使用 JSON 存储数据之前，我试过使用各种 JavaScript 自带的格式来存储内容：带有控制字符进行分隔的数组、字符串等。Douglas Crockford 所发明的 JSON 彻底改变了这一切。运用 JSON，你可以使用 JavaScript 自带的格式存储各种复杂的数据并且不需要进行额外的转换。

JSON 是 JavaScript Object Notation 的缩写，使用了我们前面介绍的两种缩略标记。

例如，想要描述一个乐队的话，可以写成：

```
var band = {
 "name":"The Red Hot Chili Peppers",
 "members":[
  {
   "name":"Anthony Kiedis",
   "role":"lead vocals"
  },
  {
   "name":"Michael 'Flea' Balzary",
   "role":"bass guitar, trumpet, backing vocals"
  },
  {
   "name":"Chad Smith",
   "role":"drums,percussion"
  },
  {
   "name":"John Frusciante",
   "role":"Lead Guitar"
  }
 ],
 "year":"2009"
}
```

可以在 JavaScript 中直接使用 JSON，并且封装在函数调用中时可作为 API 的返回值。这称为 JSON-P 格式，被很多 API 函数支持。可以使用数据端点在脚本语句中直接返回 JSON-P 格式。

```
<div id="delicious"></div><script>
function delicious(o){
 var out = '<ul>';
 for(var i=0;i<o.length;i++){
```

```
  out += '<li><a href="' + o[i].u + '">' +
     o[i].d + '</a></li>';
 }
 out += '</ul>';
 document.getElementById('delicious').innerHTML = out;
}
</script>
<script src="http://feeds.delicious.com/v2/json/codepo8/javascript?
count=15&callback=delicious "></script>
```

这里调用了 Delicious Web 服务来获得最新的 JavaScript 书签（JSON 格式），然后将其显示为无序列表。

其实，JSON 可能是在浏览器运行中描述复杂数据最轻松的方式了，甚至可以在 PHP 中调用 json_decode() 函数。

1.3 JavaScript 自带函数（数学、数组以及字符串函数）

通读了 JavaScript 的数学、数组和字符串函数后，我意识到它们会让编程变得非常方便，使用它们可避免使用许多循环和条件。例如，当需要找到一组数中的最大数时，需要写这样一个循环：

```
var numbers = [3,342,23,22,124];
var max = 0;
for(var i=0;i<numbers.length;i++){
 if(numbers[i] > max){
  max = numbers[i];
 }
}
alert(max);
```

可以不通过循环而这样实现：

```
var numbers = [3,342,23,22,124];
numbers.sort(function(a,b){return b - a});
alert(numbers[0]);
```

需要注意的是，不能对一个数值数组使用 sort() 函数，因为它会按照词法排序。

另一个有趣的方法是利用 Math.max() 函数，返回一列参数中的最大值：

```
Math.max(12,123,3,2,433,4); // returns 433
```

因为这个函数可以测试数据并返回最大值，因此可以用它来测试浏览器支持的默认属性：

```
var scrollTop= Math.max(
 doc.documentElement.scrollTop,
 doc.body.scrollTop
);
```

这解决了一个 IE 问题。我们可以读出当前文件的特性，但是对于该文件不同的文档类型，两个属性中其中之一将被赋予该值。而使用 Math.max() 则可以获得正确的值，因为只有一个属性有返回值，另一个将是未定义。

其余操作字符串的常用函数是 split() 和 join()。最经典的例子可能就是利用一个函数将 CSS 的类添加到元素中。

现在的问题是，当需要在 DOM 元素中添加一个类时，要么是将它作为第一个类添加，要么是将它和一个空格键一起加在已经存在的类前面。当删除该类时，也需要删除相应的空格（这在过去更为重要，因为有些浏览器会因为多余的空格报错）。

因此，原始方程应该写成这样：

```
function addclass(elm,newclass){
 var c = elm.className;
 elm.className = (c === '') ? newclass : c+' '+newclass;
}
```

可以运用 split() 和 join() 函数来自动实现：

```
function addclass(elm,newclass){
 var classes = elm.className.split(' ');
 classes.push(newclass);
 elm.className = classes.join(' ');
}
```

这样操作可以保证类与空格自动分离且结果被附加在最后。

1.4 事件代理

事件使得网络应用可以工作，我最爱事件，尤其是定制事件。它的存在，使得用户不需要接触核心代码就可以使产品具有更好的可拓展性。但主要的问题（其实也是它的优势）在于，事件会被 HTML 删除：对元素添加了事件监视器后它将被激活，但在 HTML 中无法表示这种情况。可以这样抽象地来考虑（这对初学者可能有困难）：诸如 IE6 之类的浏览器内存问题较多，事件处理量大，因此不要使用太多的事件处理是明智的选择。

这里就是事件代理的来源。当某一特定的元素或者其上 DOM 层的所有元素发生某一事件时，可以通过单一的处理程序对父元素进行处理来简化事件处理过程，而不是使用大量的程序。

我的意思是什么？比如说想要获得一个链接列表，而且想要通过函数的调用而不是通过加载来获得，其 HTML 实现方法如下：

```
<h2>Great Web resources</h2>
<ul id="resources">
 <li><a href="http://opera.com/wsc ">Opera Web Standards Curriculum</a></
li>
 <li><a href="http://sitepoint.com ">Sitepoint</a></li>
 <li><a href="http://alistapart.com ">A List Apart</a></li>
 <li><a href="http://yuiblog.com ">YUI Blog</a></li>
 <li><a href="http://blameitonthevoices.com ">Blame it on the voices</a></
li>
 <li><a href="http://oddlyspecific.com ">Oddly specific</a></li>
</ul>
```

通常事件处理程序是在整个链接中使用循环：

```
// Classic event handling example
(function(){
 var resources = document.getElementById('resources');
 var links = resources.getElementsByTagName('a');
 var all = links.length;
 for(var i=0;i<all;i++){
  // Attach a listener to each link
  links[i].addEventListener('click',handler,false);
 };
 function handler(e){
  var x = e.target; // Get the link that was clicked
  alert(x);
  e.preventDefault();
 };
})();
```

也可通过一个事件处理程序来实现：

```
(function(){
 var resources = document.getElementById('resources');
 resources.addEventListener('click',handler,false);
 function handler(e){
  var x = e.target; // get the link tha
  if(x.nodeName.toLowerCase() === 'a'){
   alert('Event delegation:' + x);
   e.preventDefault();
  }
 };
})();
```

因为单击事件发生在列表中所有的元素之上，所以你所要做的就是将节点 Name 与需要响应事件的元素进行对比。

说明：以上例子在 IE6 浏览器中会运行失败。对于 IE6，需要使用事件模型而不是 W3C，这就是我们在这种情况下使用库的原因。

这种方法的好处在于可以使用单独的事件处理程序。例如，想要在列表中动态地进行添加操作，如果使用事件代理，则不需要进行任何改变，只需在事