jQuery使用规范

jQuery虽好，但是不恰当的使用也会导致性能损耗。这里对如何更好的使用jQuery做一个归纳

缓存变量

DOM遍历是昂贵的，所以尽量将会重用的元素缓存。

Copy

// 糟糕 h = $('#element').height(); $('#element').css('height',h-20); // 建议 $element = $('#element'); h = $element.height(); $element.css('height',h-20);

避免全局变量

DOM遍历是昂贵的，所以尽量将会重用的元素缓存。

Copy

// 糟糕 $element = $('#element'); h = $element.height(); $element.css('height',h-20); // 建议 var $element = $('#element'); var h = $element.height(); $element.css('height',h-20);

使用匈牙利命名法

在变量前加$前缀，便于识别出jQuery对象。

Copy

// 糟糕 var first = $('#first'); var second = $('#second'); var value = first.val(); // 建议 - 在jQuery对象前加$前缀 var $first = $('#first'); var $second = $('#second'); var value = $first.val();

使用 Var 链(单 Var 模式)

将多条var语句合并为一条语句，建议将未赋值的变量放到后面。

Copy

var $first = $('#first'), $second = $('#second'), value = $first.val(), k = 3, cookiestring = 'SOMECOOKIESPLEASE', myArray = {}, i, j

请使用on

在新版jQuery中，更短的 on('click') 用来取代类似click() 这样的函数。 在之前的版本中on()就是bind()。  
自从jQuery 1.7版本后，on()附加事件处理程序的首选方法。 然而，出于一致性考虑，你可以简单的全部使用 on()方法。

Copy

// 糟糕 $first.click(function(){ ... }); // 建议 $first.on('click',function(){ ... });

精简语句

一般来说，最好尽可能合并函数。

Copy

// 糟糕 $first.on('click', function(){ $(this).css('border', '1px solid red') .css('color','blue'); }); // 建议 $first.on('click', function(){ $(this).css({ 'border': '1px solid red', 'color': 'blue' }); });

链式操作

jQuery实现方法的链式操作是非常容易的。下面利用这一点。

Copy

// 糟糕 $second.html(value); $second.on('click',function(){ alert('hello everybody'); }); $second.fadeIn('slow'); $second.animate({height: '120px'}, 500); // 建议 $second.html(value) .on('click', function(){ alert('hello everybody'); }) .fadeIn('slow') .animate({height: '120px'}, 500);

维持代码的可读性

伴随着精简代码和使用链式的同时，可能带来代码的难以阅读。添加缩紧和换行能起到很好的效果。

Copy

// 糟糕 $second.html(value); $second.on('click', function(){alert('hello everybody');}).fadeIn('slow').animate({height:'120px'},500); // 建议 $second.html(value) .on('click', function(){ alert('hello everybody'); }) .fadeIn('slow') .animate({height: '120px'}, 500);

选择短路求值

短路求值是一个从左到右求值的表达式，用 &&(逻辑与)或(逻辑或)操作符。

Copy

// 糟糕 function initVar($myVar) { if(!$myVar) { $myVar = $('#selector'); } } // 建议 function initVar($myVar) { $myVar = $myVar || $('#selector'); }

选择捷径

精简代码的其中一种方式是利用编码捷径。

Copy

// 糟糕 if(collection.length > 0){..} // 建议 if(collection.length){..}

繁重的操作中分离元素

如果你打算对DOM元素做大量操作(连续设置多个属性或css样式)，建议首先分离元素然后在添加。

Copy

// 糟糕 var $container = $("#container"), $containerLi = $("#container li"), $element = null; $element = $containerLi.first(); //... 许多复杂的操作 // 建议 var $container = $("#container"), $containerLi = $container.find("li"), $element = null; $element = $containerLi.first().detach(); //建议使用detach()将元素分离出来 //... 许多复杂的操作 $container.append($element);

使用子查询缓存的父元素

正如前面所提到的，DOM遍历是一项昂贵的操作。典型做法是缓存父元素并在选择子元素时重用这些缓存元素。

Copy

// 糟糕 var $container = $('#container'), $containerLi = $('#container li'), $containerLiSpan = $('#container li span'); // 建议 (高效) var $container = $('#container '), $containerLi = $container.find('li'), $containerLiSpan= $containerLi.find('span');

避免通用选择符

将通用选择符放到后代选择符中，性能非常糟糕。

Copy

// 糟糕 $('.container > \*'); // 建议 $('.container').children();

避免隐式通用选择符

通用选择符有时是隐式的，不容易发现。

Copy

// 糟糕 $('.someclass :radio'); // 建议 $('.someclass input:radio');

优化选择符

例如，Id选择符应该是唯一的，所以没有必要添加额外的选择符。

Copy

// 糟糕 $('div#myid'); $('div#footer a.myLink'); // 建议 $('#myid'); $('#footer .myLink');

避免多个ID选择符

在此强调，ID 选择符应该是唯一的，不需要添加额外的选择符，更不需要多个后代ID选择符。

Copy

// 糟糕 $('#outer #inner'); // 建议 $('#inner');

摒弃弃用方法

关注每个新版本的废弃方法是非常重要的并尽量避免使用这些方法。

Copy

// 糟糕 - live 已经废弃 $('#stuff').live('click', function() { console.log('hooray'); }); // 建议 $('#stuff').on('click', function() { console.log('hooray'); }); // 建议, 给动态创建的元素添加绑定事件 $(document).on('click', '.stuff', function() { console.log('hooray'); });