**jQuery原理和jQuery扩展**

**[一、jQuery的作用：](#_Toc11283)** [2](#_Toc11283)

**[二、jQuery如何实现声明？](#_Toc11743)** [2](#_Toc11743)

**[三、JQuery的核心：](#_Toc30261)** [2](#_Toc30261)

**[五、 调用jQuery的两种方式？](#_Toc499)** [3](#_Toc499)

**[1、高级的实现：](#_Toc1704)** [3](#_Toc1704)

**[2、较为低级的实现：](#_Toc15912)** [3](#_Toc15912)

**[六、 jQuery的实现原理](#_Toc20093)** [3](#_Toc20093)

**[1、通过jQuery原型属性的init方法来创建对象。](#_Toc8687)** [4](#_Toc8687)

**[2、对init方法内this指向的处理。](#_Toc24441)** [4](#_Toc24441)

**[3、jQuery对象通过extend方法来扩展自己的属性和方法](#_Toc28583)** [4](#_Toc28583)

**[4、$(selector)之后，返回的是jQuery对象。](#_Toc21354)** [5](#_Toc21354)

**[5、jQuery(selector)方法调用时返回的是数组。](#_Toc24365)** [5](#_Toc24365)

**[6、jQuery能做到jQuery对象属性/方法/事件的插件式扩展。](#_Toc31474)** [6](#_Toc31474)

**[七、实现自己的jQuery](#_Toc32231)** [6](#_Toc32231)

**一、jQuery的作用：**

jQuery是为了改变javascript的编码方式而设计的。从这段话可以看出jQuery本身并不是UI组件库或其他的一般AJAX类库。jQuery改变javascript编码方式！

**二、jQuery如何实现声明？**

这里，用以下的一段简短的使用流程：

1、查找(创建)jQuery对象：$("selector");

2、调用jQuery对象的方法完成我们需要完成的工作：$("selector").doOurWork();

jQuery就是以这种可以说是最简单的编码逻辑来改变javascript编码方式的。这两个步骤是jQuery的编码逻辑核心！

要实现这种简洁编码方式，创建jQuery对象这一环节至关重要。

**三、JQuery的核心：**

1、强悍的dom元素查找能力。

2、随心所欲的方法扩展

jQuery对象的方法肯定是有限的，有限的方法满足不了日益增长各有所需的要求，所以，必须提供jQuery对象方法的扩展能力。

**四、JQuery示例**

来一个简单的示例，来说明jQuery是如何工作的：

|  |
| --- |
| < !DOCTYPE>  <html>  <head>  <title>无标题页</title>  </head>  <body id=”bd”>  <a href=”http://www.baidu.com”>baidu</a>  </body>  </html>  <script type=”text/javascript” src=”../Script/jquery.js”></script>  <script type=”text/javascript” >  $(function(){  //1查找$(”a”); 2 jQuery对象事件click; 3 jQuery对象方法hide   $(“a“).click(function(e){  $(this).hide(“slow“);   return false;   });  });  </script> |

jQuery中有一个“配置”的思想，这一点使得对象的属性/事件等设置变得容易理解且十分简便。

1. **调用jQuery的两种方式？**

**1、高级的实现：**

通过传递一个参数实现DOM选择，如通过$(“h1″)选择所有的h1元素。

typeof $(‘h1′) ;//”object”

**2、较为低级的实现：**

如果通过$.ajax实现ajax的操作。

Typeof $.ajax ; //”function”

1. **jQuery的实现原理**

|  |
| --- |
| var jQuery = function( selector, context ) {   // The jQuery object is actually just the init constructor 'enhanced'   return new jQuery.fn.init( selector, context, rootjQuery );  }  jQuery.fn=jQuery.prototype={   jquery: core\_version,   init:function(selector,context){   //some code   return this;   }   //some code there   //......  }  jQuery.fn.init.prototype=jQuery.fn; |

**1、通过jQuery原型属性的init方法来创建对象。**

用init方法来创建对象可以达到不用new创建对象的目的。

**2、对init方法内this指向的处理。**

通过调用init返回一个jQuery的实例，那么这个实例就必须要继承jQuery.prototype的属性，那么init里面这个this, 就继承jQuery.prototype的属性。

但是init里面的this，受制于作用域的限制，并不能访问jQuery.prototype其它的属性。 jQuery通过一句'jQuery.fn.init.prototype=jQuery.fn'把它的原型指向jQuery.fn,这样以来，init产生的jQuery对象就拥有了jQuery.fn的属性。

**3、jQuery对象通过extend方法来扩展自己的属性和方法**

jQuery构造函数和jQuery.prototype均有各自的属性和方法，两者的调用方法各不一样,这两个对象都有一个extend方法，都是用来扩展自身的属性和方法，在jQuery内部，extend的实现实际是靠一样的代码。



**4、$(selector)之后，返回的是jQuery对象。**

从jQuery的源代码中，我们可以知道：var $ = jQuery 。

因此当我们$(selector)操作时，其实就是jQuery(selector)，创建的是一个jQuery对象。

当然正确的写法应该是这样的：var jq = new $(selector);

而jQuery使用了一个小技巧在外部避免了new，在jquery方法内部：if ( window == this ) return new jQuery(selector);

**5、jQuery(selector)方法调用时返回的是数组。**

在jQuery(selector)方法内部，最后返回的是一个数组：return this.setArray(a);

因此jQuery对象可以直接调用jQuery内置的数组的方法。

比如$(selector).each(function(index){…});进行遍历操作。  
 而each方法体内部是一个for循环，在循环体内是这样调用的：method.call(this[i]，i) 。

**6、jQuery能做到jQuery对象属性/方法/事件的插件式扩展。** jQuery.prototype原型对象上的扩展属性/方法和事件，将会给jQuery的对象“扩展”。

基于这一点，jQuery是这样写的：jQuery.fn = jQuery.prototype 。

所以，当我们扩展一个插件功能时，如下：

|  |
| --- |
| jQuery.fn.check = function() {   return this.each(function() {   this.checked = true;   });  }; |

其实就是：

|  |
| --- |
| jQuery.prototype.check = function() {   return this.each(function() {   this.checked = true;   });  }; |

综上所述,jQuery给我们带来了一个简洁方便的编码模型

(1>创建jQuery对象;2>直接使用jQuery对象的属性/方法/事件),

一个强悍的dom元素查找器($),插件式编程接口(jQuery.fn),以及插件初始化的”配置”对象思想.

**七、实现自己的jQuery**

|  |
| --- |
| < !DOCTYPE>   <html>  <head>  <title>Untitled Page</title>  </head>  <body>  <div id=”d”>divvv</div>  <div id=”fsd”>fdsf</div>  </body>  </html>  <script type=”text/javascript”>  //实现自己的MyQuery框架  var MyQuery = function(selector){   if ( window == this ) return new MyQuery(selector);   //这里只实现dom类型的简单查找,嘿嘿   var doms = document.getElementsByTagName(selector);   var arr = [];   for(var i=0; i<doms .length; i++){   arr.push(doms.item(i));   }   return this.setArray(arr);  }  MyQuery.prototype.setArray = function( arr ) {   this.length = 0;   [].push.apply( this, arr );   return this;  }  MyQuery.fn = MyQuery.prototype;  var $ = MyQuery;   //插件扩展1)each  MyQuery.fn.each = function(method){   for(var i=0,l=this.length; i<l; i++){   method.call(this[i],i);   }  }  //插件扩展2)show  MyQuery.fn.show = function(){   this.each(function(i){   alert(i+“:“+this.id+“:“+this.innerHTML);   });  }  //debugger  $(“div“).show();  </script>  </doms>  </script> |