Jquery

## 1、Jquery与jQuery UI 有啥区别？

\*jQuery是一个js库，主要提供的功能是选择器，属性修改和事件绑定等等。

\*jQuery UI则是在jQuery的基础上，利用jQuery的扩展性，设计的插件。

提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等

16.jquery 中如何将数组转化为json字符串，然后再转化回来？

jQuery中没有提供这个功能，所以你需要先编写两个jQuery的扩展：

$.fn.stringifyArray = function(array) {

return JSON.stringify(array)

}

$.fn.parseArray = function(array) {

return JSON.parse(array)

}

然后调用：

$("").stringifyArray(array)

## 2、请写出jquery绑定事件的方法，不少于两种。

.bind()是直接绑定在元素上

.live()则是通过冒泡的方式来绑定到元素上的。更适合列表类型的，绑定到document DOM节点上。和.bind()的优势是支持动态数据。(live由于性能原因已经被废弃)

.delegate()则是更精确的小范围使用事件代理，性能优于.live()

.on()则是最新的1.9版本整合了之前的三种方式的新事件绑定机制

1.无论使用bind、on、delegate、click(function())都是重复绑定，即绑定的同类型事件被放到一个事件队列中，依次执行，后绑定的事件不会替换之前绑定的，对于on使用off，delegate用undelegate，bind及click使用unbind来解除绑定，例如unbind(type)传递为事件类型，如果不传type则解出所有事件绑定；需要注意的是元素本身自带的事件无法unbind（如button1）

2.要绑定自定义事件，如'open'，以上函数都可以使用，但激活需要使用trigger

3.建议使用on函数

$('.myClass').on({

click:function(eleDom){

...do someting

},

dbclick:function(eleDom){

...do someting

}

})

## 3、你知道哪些针对jQuery的优化方法？

1.总是从ID选择器开始继承在jQuery中最快的选择器是ID选择器，因为它直接来自于JavaScript的getElementById()方法。

例如有一段HTML代码：

[](javascript:void(0);)

1 <div id="content">

2 <form method="post"action="#">

3 <h2>交通信号灯</h2>

4 <ul id="traffic\_light">

5 <li>

6 <input type="radio"class="on"name="light"value="red"/>红色

7 </li>

8 <li>

9 <input type="radio"class="off"name="light"value="yellow"/>黄色

10 </li>

11 <li>

12 <input type="radio"class="off"name="light"value="green"/>绿色

13 </li>

14 </ul>

15 <input class="button"id="traffic\_button"type="submit"value="Go"/>

16 </form>

17 </div>

比如需要选择红绿单选框，那么可以使用一个tag name来限制(修饰)class，如下所示：var active\_light=$(“input.on”);当然也可以结合就近的ID，如下所示：var active\_light=$(“#traffic\_light input.on”);  如果采用下面的选择器，那么效率是低效的。var traffic\_button=$(“#content.button”);因为button已经有ID了，我们可以直接使用ID选择器。如下所示：var traffic\_button=$(“#traffic\_button”);当然这只是对于单一的元素来讲。如果你需要选择多个元素，这必然会涉及到DOM遍历和循环，为了提高性能，建议从最近的ID开始继承。如下所示：var traffic\_lights=$(“#traffic\_light input”);

2.在class前使用tag(标签名)在jQuery中第二快的选择器是tag(标签)选择器(比如：$(“head”))。

跟ID选择器累时，因为它来自原生的getElementsByTagName()方法。继续看刚才那段HTML代码：

在使用tag来修饰class的时候，我们需要注意以下几点：（1）不要使用tag来修饰ID，如下所示：var content=$(“div#content”);这样一来，选择器会先遍历所有的div元素，然后匹配#content。（好像jQuery从1.3.1开始改变了选择器核心后，不存在这个问题了。暂时无法考证。）（2）不要画蛇添足的使用ID来修饰ID，如下所示：var traffic\_light=$(“#content#traffic\_light”);

3.将jQuery对象缓存起来把jQuery对象缓存起来就是要告诉我们要养成将jQuery对象缓存进变量的习惯。

下面是一个jQuery新手写的一段代码：

1 $("#traffic\_light input.on").bind("click",function(){});

2 $("#traffic\_light input.on").css("border","1px dashed yellow");

3 $("#traffic\_light input.on").css("background-color","orange");

4 $("#traffic\_light input.on").fadeIn("slow");

但切记不要这么做。我们应该先将对象缓存进一个变量然后再操作，如下所示：

记住，永远不要让相同的选择器在你的代码里出现多次.注：（1）为了区分普通的JavaScript对象和jQuery对象，可以在变量首字母前加上$符号。（2）上面代码可以使用jQuery的链式操作加以改善。如下所示：

1 var $active\_light = $("#traffic\_light input.on");

2

3 $active\_light.bind("click",function(){})

4 .css("border","1px dashed yellow")

5 .css("background-color","orange")

6 .fadeIn("slow");

4.如果你打算在其他函数中使用jQuery对象，那么你必须把它们缓存到全局环境中。

如下代码所示：

1 // 在全局范围定义一个对象(例如:window对象)

2 window.$my = {

3 head:$("head"),

4 trafficLight:$("#trafficLight"),

5 trafficButton:$("#trafficButton")

6 };

7

8 function doSomething(){

9 // 现在你可以引用存储的结果并操作它们

10 var script=document.createElement("script");

11 $my.head.append(script);

12

13 // 当你在函数内部操作是,可以继续将查询存入全局对象中去.

14 $my.coolResults=$("#some\_ul li");

15 $my.otherResults=$("#some\_table td");

16

17 // 将全局函数作为一个普通的jquery对象去使用.

18 $my.otherResults.css("border-color","red");

19 $my.trafficLight.css("border-color","green");

20 }

21 // 你也可以在其他函数中使用它.

这里的基本思想是在内存中建立你确实想要的东西，然后更新DOM。这并不是一个jQuery最佳实践，但必须进行有效的JavaScript操作。直接的DOM操作速度很慢。例如，你想动态的创建一组列表元素，千万不要这样做,如下所示：对直接的DOM操作进行限制。

1 var top\_100\_list = [];// 假设这里是100个独一无二的字符串

2 var $mylist = $("#mylist");// jQuery选择到<ul>元素

3

4 for(var i = 0,l = top\_100\_list.length;i<l;i++) {

5 $mylist.append("<li>" + top\_100\_list[i] + "</li>");

6 }

我们应该将整套元素字符串在插入进dom中之前先全部创建好，如下所示：

1 $("#entryform input").bind("focus",function() {

2 $(this).addClass("selected");

3 }).bind("blur",function(){

4 $(this).removeClass("selected");

5 });

5.冒泡除非在特殊情况下,否则每一个js事件(例如:click,mouseover等.)都会冒泡到父级节点。

当我们需要给多个元素调用同个函数时这点会很有用。代替这种效率很差的多元素事件监听的方法就是,你只需向它们的父节点绑定一次。比如,我们要为一个拥有很多输入框的表单绑定这样的行为:当输入框被选中时为它添加一个class传统的做法是，直接选中input，然后绑定focus等，如下所示：

1 $("#entryform").bind("focus",function(e) {

2 var $cell = $(e.target); // e.target捕捉到触发的目标元素

3 $cell.addClass("selected");

4 }).bind("blur",function(e) {

5 var $cell = $(e.target);

6 $cell.removeClass("selected");

7 });

当然上面代码能帮我们完成相应的任务，但如果你要寻求更高效的方法，请使用如下代码：

通过在父级监听获取焦点和失去焦点的事件，对目标元素进行操作。在上面代码中，父级元素扮演了一个调度员的角色,它可以基于目标元素绑定事件。如果你发现你给很多元素绑定了同一个事件监听,那么现在的你肯定知道哪里做错了。

6.推迟到$(window).load。

jQuery对于开发者来说有一个很诱人的东西,可以把任何东西挂到$(document).ready下。尽管$(document).rady确实很有用，它可以在页面渲染时，其它元素还没下载完成就执行。如果你发现你的页面一直是载入中的状态，很有可能就是$(document).ready函数引起的。你可以通过将jQuery函数绑定到$(window).load事件的方法来减少页面载入时的cpu使用率。它会在所有的html(包括iframe)被下载完成后执行。一些特效的功能，例如拖放,视觉特效和动画,预载入隐藏图像等等，都是适合这种技术的场合。

7.压缩JavaScript。

在线压缩地址:http://dean.edwards.name/packer/压缩之前，请保证你的代码的规范性，否则可能失败，导致Js错误。

8.尽量使用ID代替Class。

 前面性能优化已经说过，ID选择器的速度是最快的。所以在HTML代码中，能使用ID的尽量使用ID来代替class。看下面的一个例子：

1 // 创建一个list

2 var $myList = $('#myList');

3 var myListItems = '<ul>';

4

5 for(var i = 0; i < 1000; i ++) {

6 myListItems += '<li class = "listItem'+i+'">This is a list item</li>';

7 // 这里使用的是class

8 }

9

10 myListItems += '</ul>';

11 $myList.html(myListItems);

12 // 选择每一个li

13

14 for(var i = 0; i<1000; i++) {

15 var selectedItem = $('.listItem' + i);

16 }

在上段代码中，选择每个li总共只用了61毫秒，相比class的方式，将近快了100倍。       在代码最后，选择每个li的过程中，总共用了5066毫秒，超过5秒了。接着我们做一个对比，用ID代替class：

9.给选择器一个上下文。

jQuery选择器中有一个这样的选择器，它能指定上下文。jQuery(expression,context);通过它，能缩小选择器在DOM中搜索的范围，达到节省时间，提高效率。普通方式：$(‘.myDiv’)改进方式：$(‘.myDiv’,$(“#listItem”))。

10.慎用.live()方法（应该说尽量不要使用）。

这是jQuery1.3.1版本之后增加的方法，这个方法的功能就是为新增的DOM元素动态绑定事件。但对于效率来说，这个方法比较占用资源。所以请尽量不要使用它。例如有这么一段代码:

1 <script type="text/javascript">

2 $(function(){

3

4 $("p").click(function(){

5 alert($(this).text());

6 });

7

8 $("button").click(function(){

9 $("<p>this is second p</p>").appendTo("body");

10 });

11 })

12 </script>

13 <body>

14 <p>this is first p</p>

15 <button>add</button>

16 </body>

运行后，你会发现新增的p元素，并没用被绑定click事件。你可以改成.live(“click”)方式解决此问题，代码如下：

1 $(function() {

2 $("p").live("click",function() {

3 // 改成live方式

4 alert($(this).text());

5 });

6

7 $("button").click(function() {

8 $("<p>this is second p</p>").appendTo("body");

9 });

10 })

但我并不建议大家这么做，我想用另一种方式去解决这个问题，代码如下：

1 $(function() {

2 $("p").click(function() {

3 alert($(this).text());

4 });

5

6 $("button").click(function() {

7 $("<p>this is second p</p>").click(function() {

8 // 为新增的元素重新绑定一次

9 alert($(this).text());

10 }).appendTo("body");

11 });

12 })

虽然我把绑定事件重新写了一次，代码多了点，但这种方式的效率明显高于live()方式，特别是在频繁的DOM操作中，这点非常明显。

## 4、请说出你可以传递给jQuery方法的四种不同值。

选择器（字符串），HTML（字符串），回调函数，HTML元素，对象，数组，元素数组，jQuery对象等