出处:http://blog.csdn.net/wansijie/

实战一

贝壳(就是本文作者了)也属于刚刚会用点 JQuery 的那一类型，在学习过程中也遇到挺多问题,特别是在开始入门的时候.一直准备写些有关 JQuery 的文章,但又恐自己文笔及表达能力有限而"误人子弟",最终还是鼓起 勇气. 如在文章中有错误或者不合适的理解望广大朋友直接指出批评.

本文的宗旨，授人鱼不如授人渔. 我只会讲关键部分,不可能 JQuery 的每个函数我都讲,因为有很多函数贝壳自己在实际应用中都从未使用过. 但当我们已经会渔了还用担心鱼吗!?

BTW:贝壳假设你已经掌了基本的 JavaScript 运用能力及基础的 CSS 知识.

JQuery 是什么

JQuery 只是一个 JS 文件

人对新鲜事业总是报着好奇与排斥的情感,贝壳刚开始学的时候也一样.想掌握又怕自己学不会. **其实 JQuery 很简单.**

JQuery 其实就是一个工具包,很多常用的功能已经被好心人封装好,我们真要直接调用就可以了(有点类似 SDK),具体内部原理和实现我们现在不去讨论.
记住 JQuery 很简单,很方便.

目前 jQuery 最新 Release 版本为 1.32,官方下载地址为:http://docs.jquery.com/Downloading_jQuery

本系列教程基于 1.32 版进行讲解.

JQuery 有两种版本:

一种为 uncompressed 版(未压缩版)主要用在开发中.

另一种为 Minified(迷您版)当开发完毕了,就可以改用这个版本.

两种版本的区别

文件大小不一样，最终用户在浏览时可以减少下载 JS 时的等待时间.

Uncompressed 版文件大小为 118KB

Minified 版文件大小为 56KB

JQuery 能带给我们什么

一, 站在巨人的肩上

别人已经发明了轮子,我们没必要自己再去造一个.

二, 强大的 BOM,DOM,CSS,事件 的操作能力

我最常用的是 CSS 及 DOM 的操作,真的太方便了.

三, 更加简洁的代码

自己的亲身经历,比如表格(Table)中的 TR 奇偶为两种颜色的代码如果自己用 JavaScript 去编号得用几十行代码,JQuery 只需要 2 行就可以了.

四, 与浏览器无关性

做过 JavaScript 开发的就知道 JavaScript 在不同的浏览器下行为表现有时不一样,我们不得不为不同的浏览器编写相应的代码. JQuery 在这方法已经内部处理,我们只管去用就是了.

五, 不错的性能

我自己有自知之明, 至少我写的 JavaScript 代码的运行效率不如 JQuery 的高.

如何使用 JQuery

万年开头难,我在第一次使用 JQuery 时被吓到过,不知道怎么使用,其实很简单,下面我们用一个例子来说明. :)

建立一个 html 文件,比如 index.html 并与 JQuery 库文件 jquery-1.3.2.js 放到同目录下(其实放那都可以)

首先我们解决 JQuery 的引用. 代码如下:

```
<html>
    <head>
        <title>JQuery 测试</title>
```

```
            <script language="javascript" src="jquery-
1.3.2.js"></script>

        </head>

        <body>

        </body>

    </thml>
```

红色的那一行代码就完成了 JQuery 的引用,简单吧!

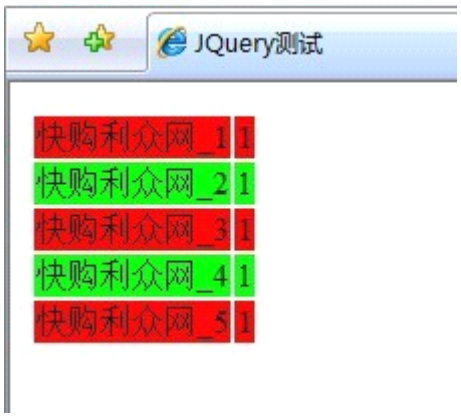
现在我们再扩充一个功能,使 index.html 显示一个表格,并且表格内各行的颜色奇偶不同.也就是奇数行是一种颜色,偶数行又是另一种颜色.

index.html 代码结构:

```
<html>
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
    <title>JQuery 测试</title>
    <style type="text/css">

        /*table 中偶数行*/

        .tabEven {
            background: #ff0000;
        }

        /*table 中奇数行*/

        .tabOdd {
            background: #00ff00;
        }
    </style>
    <script language="javascript" src="jquery-1.3.2.js">
    </script>
    <script language = "JavaScript" type="text/javascript">
        //<![CDATA[
        $(document).ready(function(){
            $("#tabTest tr:even").addClass("tabEven");
```

```
            $("#tabTest tr:odd").addClass("tabOdd");
        });
        //]]>
    </script>
</head>
<body>
    <table id="tabTest">
        <tbody>
            <tr>
                <td>快购利众网_1</td>
                <td>1</td>
            </tr>
            <tr>
                <td>快购利众网_2</td>
                <td>1</td>
            </tr>
            <tr>
                <td>快购利众网_3</td>
                <td>1</td>
            </tr>
            <tr>
                <td>快购利众网_4</td>
                <td>1</td>
            </tr>
            <tr>
                <td>快购利众网_5</td>
                <td>1</td>
            </tr>
        </tbody>
    </table>
</body>
</thml>
```

效果如下:



代码解析:

```
$(document).ready(function(){
        $("#tabTest tr:even").addClass("tabEven");
        $("#tabTest tr:odd").addClass("tabOdd");
    });
```

$(document).ready 的 作用很简单,就是等待网页全部内部载入后再执行 JavaScript 代码(别忘了 JQuery 也是 JavaScript 代码哟), 这个函数几乎在所有使用 JQuery 的代码中都用到. 因为如果我们的 Javascript 代码中如果有 DOM 操作,在网页还没完全读取完时整个 DOM 框架还没有真正建立起来,在这之前的 DOM 操作都是无效 的.

$("#tabTest tr:even").addClass("tabEven"); 为 id 为 tabTest 的表格的偶数行增加名为 tabEven 的 CSS 样式

$("#tabTest tr:odd").addClass("tabOdd"); 为 id 为 tabTest 的表格的奇数行增加名为 tabEven 的 CSS 样式

实战二

就进入实战?是不是太快了?我还不知道怎么用 JQuery!! ...

我知道大家的想法, 放心好了,有些东西是不需要太多的理论知道做为基础, 其实在我们已经掌握 JavaScript 时,就已经掌握了 JQuery 的理论知识. 还是入门篇所说的那样,**其实 JQuery 很简单. :)**

今天的教程有些复杂(只是概念多点而已),但过了这一关,JQuery 的学习可谓是一马平川.

好了,进入正题. 再次申明 **JQuery 很简单,take easy!**

从那开始呢? 最好的切入地方就从 JQuery 的最基本的一个函数 ready 开始! 该函数的作用可以看做和 onLoad,至少现在可以这么看待.

定义

ready(fn);

功能

这是事件模块中**最重要**的一个函数,因为它可以极大地提高 web 应用程序的响应速度。

简单地说,这个方法纯粹是对向 **window.load 事件**注册事件**的替代方法**。通过使用这个方法,可以在 DOM 载入就绪能够读取并操纵时立即调用你所绑定的函数, 而 99.99%的 JavaScript 函数都需要在那一刻执行。

有一个参数——对 jQuery 函数的引用——会传递到这个 ready 事件处理函数中。可以给这个参数任意起一个名字,并因此可以不再担心命名冲突而放心地使用$别名。

请确保在 元素的 onload 事件中没有注册函数, 否则不会触发 $(document).ready()事件。

可以在同一个页面中无限次地使用$(document).ready()事件。其中注册的函数会按照(代码中的)先后顺序依次执行。

通过上面所述,我们可以把 ready 看做 onLoad 的替代方法. 这时有的朋友就会问了,有 onLoad 方法我们干什么还要用 ready 方法?

我个人的体会及看法是 onLoad 缺点是以后维护起来麻烦,到处都是 JavaScript 代码,很容易出问题的哟! 在<<ppk 谈 JavaScript>>中,ppk 针对这个问题的看法也是如此,尽量不要在标签中直接编写 JavaScript 代码.

实例

两种编写方式

一

```
$(document).ready(function(){alert("Hello World!");});
```

二

```
var myFun = function(){alert("Hello World!");}

$(document).ready(myFun);
```

到这儿我想大家对 ready 的用法应该是明白了,但对前面的 $(document) 应该很迷惑. 这是什么东东?别急... 现在只要记住这段代码的功能就是当整个文档载入完毕后再执行 ready 内的函数就够了.

看完下面的代码我们就明白了$()的用法.

index.html 代码结构如下:

```
<html>
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
    <title>JQuery 测试</title>

    <style type="text/css">

        .p1 {
            background: #ff0000;
        }

        .p2 {
            background: #00ff00;
        }

        .p3{

            background: #0000ff;

        }

        .myPCss{

            font-size:36px;

        }
    </style>
    <script language="javascript" src="jquery-1.3.2.js">
    </script>
```

```
    <script language = "JavaScript" type="text/javascript">
        //<![CDATA[
        $(document).ready(function(){
            $("p").addClass("p1");
            $("p").removeClass("p1");

            $("#myP").addClass("p2");

            $(".myPCss").addClass("p3");



            $("#myDiv  p").addClass("p3");

            $("#myDiv>p").addClass("p3");

            $("div+p").addClass("p3");

            $("div~p").addClass("p3");

            var aP = document.getElementById("myP");

            $(aP).addClass("p2");


        });
        //]]>
    </script>
</head>
<body>

  <p>快购利众网 1</p>

  <p id="myP">快购利众网 2</p>

  <p class="myPCss">快购利众网 3</p>

  <div id="myDiv">

      <div id="myDivInner">

          <p>快购利众网 4</p>

      </div>

      <div id="myDivTemp">
```

```
        </div>

        <p>快购利众网 5</p>

        <p>快购利众网 6</p>

    </div>

    <p>快购利众网 7</p>
</body>
</thml>
```

代码解析:

```
    $("p").addClass("p1");
    $("p").removeClass("p1");

    $("#myP").addClass("p2");

    $(".myPCss").addClass("p3");

    $("#myDiv  p").addClass("p3");

    $("#myDiv>p").addClass("p3");

    $("div+p").addClass("p3");

    $("div~p").addClass("p3");

    var aP = document.getElementById("myP");

    $(aP).addClass("p2");

    ////////////////////////////////////////////////////////////

    $("p").addClass("p1");

    $("p").removeClass("p1");
```

选择文档里**全部**的 元素对象,**不论** **在文档中所处的层次结构如何**, 最后选到了 **7** 个 元素对象 " 快购利众网 1 快购利众网 2 快购利众网 3 快购利众网 4 快购利众网 5 快购利众网 6 快购利众网 7 "

addClass("css name")函数很简单,就是为前面$()选择到的元素对象增加 CSS 样式.

removeClass("css name")函数也很简单,就是去掉前面$()选择到的元素对象的指定的样式.

在这儿,这两段代码功能抵消,相当什么也没做.

```
$("#myP").addClass("p2");
```

选择文档里 id 为"myP"的**指定**元素对象,并为该元素对象增加名称为"p2"的样式. 最后选到了 1 个 元素对象 " 快购利众网 2 "

知识点: 如果要选择指定 ID 号的元素对象,记得前面用"#"

```
$(".myPCss").addClass("p3");
```

选择文档里**样式名**为"myPCss"的**指定**元素对象,并为该元素对象增加名称为"p3"的样式. 最后选到了 1 个 元素对象 " 快购利众网 3 "

知识点: 如果要选择指定样式的元素对象,记得前面用"."

现在可以为大家正式介绍**$(),**在 JQuery 里,我们称她为"**选择器函数**",里面的内容称为"**选择器**".

现在大家自己试试,**记住选择器选择出的对象有可能是多个哟.**

上面的内容是不是挺简单的!! 嗯,革命才刚刚开始,下面的内容稍稍复杂些因为牵涉到层次的概念,但大家别急,**只要记住"选择器"选择出的元素对象可能是多个这点就不用怕了.**

```
$("#myDiv  p").addClass("p3");
```

功能:在给定的祖先元素下匹配所有的后代元素

分成两部分来分析

一,$("#myDiv") 根据上面所学的知识,选择出 1 个 元素对象, " "

二,$("#myDiv p") 在上面 2 个 元素对象中的**任意层**中选择 元素对象. 几个? 3 个 " 快购利众网 4 快购利众网 5 快购利众网 6 "

其中" 快购利众网 4 快购利众网 5 快购利众网 6< /p>"都是在" "内部定义的

虽然" 快购利众网 4 "是在 id 为"myDivInner 的"div"内部定义的.但因为 id 为"myDivInner 的"div" 也是 id 为"myDiv 的下层,所以" 快购利众网 5 "也属于 id 为"myDiv 的下层. 有些绕口,记住一点就行. A,B 操作器中如果是用空格连接,那么表示 B 属于 A 的下层(可以为任意层)之中

最后为这 3 个 元素对象增加名为"p3"的样式

知识点: **对于这类有层次的选择表达式$("A B"),A 选择器和 B 选择器可以是"标签名","#id",".css"三种中的任意一种,中间用空格分开,空格表示任意层次.**

右边的 B 选择器是在左边 A 选择器选择的结果上进行内部任意层次中选择, 记住是在左边选择出的元素对象(可能是多个)的内部进行再次选择(可能是多个). <- 这个知识点一定要理解透!

```
$("#myDiv>p").addClass("p3");
```

功能:在给定的父元素下匹配所有的子元素

>代表#myDiv 下的子元素对象(多个并只是下一层),最后选择出 2 个 元素对象,"" 快购利众网 5 快购利众网 6 ",并为该 对象增加名为"p3"的样式

```
$("div+p").addClass("p3");
```

功能:匹配所有**紧接在 div 元素后的 第一个同辈 p 元素**

+代表**紧接着** div **同层**的第一个子元素对象.

id 为"myDivInner"同层后面第一个因为是" ",所以后没有紧接着的" "

id 为"myDiv"同层后面正好是一个" "

最后选择出 1 个 元素对象,"" 快购利众网 7 ",并为该 对象增加名为"p3"的样式

知识点: 是紧接着,如果 A 与 B 之间有其它元素都无法匹配.

```
$("div~p").addClass("p3");
```

功能:匹配 #myDiv 元素之后的**所有同辈** p 元素

该功能与 + 类似,不同的有两处.

一, +为同辈并且紧跟, ~为同辈不需要一定紧跟

二, +为同辈并且第一个, ~为同辈多个.

```
var aP = document.getElementById("myP");

$(aP).addClass("p2");
```

$(aP).addClass("p2") 其实就是$("#myP").addClass("p2") 的另一种形式.

$()中处了可以用字符串的表达式选择器,还可以使用 DOM 对象

当你能看到这句话,我想对你说"辛苦了."

学习的过程本来就是艰辛难耐的,唯为坚持才能战胜一切.

到现在我们应该明白之前的"$(document)"代表什么意思了吧.

好了, 今天先讲到这儿. 要快速熟悉选择器的用法只有多加练习.更多的高级应用我相信大家都能自己学会,掌握.

实战三

这套教程在我博客以及 CSDN 论坛里发布后，个人感觉挺受大家欢迎的，对次我很开心，虽然有些怀疑是否原创的声音，但我想本来就是自己写的也不去管了。

在实战篇上发布后很多朋友建议用真正的“实战”例子。 好吧，我尽量用我自己的项目中的例子来讲解，但如果开篇说的那样，我自己也是刚刚开始学 JQuery，所以自己项目中的例子也许是很简单的。

有些朋友希望我讲解一些有关 JavaScript,CSS 方面的知识,说实话我本没有准备在本套教程里讨论与此有关的知识, 但挺多朋友们提出建议,那我准备粗略讨论一下,一带而过.,权当起个抛砖引玉的作用.

JavaScript 方面

最基础的核心功能 如运算符(+-*/等) 控制流程语句(if,for,while 等) 等等我就不说了. 这是最最基本的知识.

BOM 知识

BOM Browser Object Model 浏览器对象模型.

window 对象是 BOM 最重要的对象. 我们可以简单的把一个 IE 窗口理解为一个 window 对象.

window.location

window.history

window.screen

window.document 这个是我们最常用的

事件

其实挺简单的,但要记住事件的驱动有两模式, **"冒泡"与"捕获"**

冒泡: 激活顺序是事件从它(事件)触发点开始向上层逐级冒泡(触活)直至 document(上面所说的 window.document)为止.

捕获: 激活顺序与冒泡相反,事伯从第一层(document)逐级向下直止找到最终的事件激活目标.

上面两个概念看过去是麻烦,我们只要记住一般都用"冒泡"就可以了.

DOM

DOM Document Object Model 文档对象模型.

我非常喜欢这个功能(模型), 因为她实现了 WEB 界面的千变万化.(个人想法)

window.document 就是 DOM 要操作的对象, DOM 把 document 内的节点(<head><title><body><form><table> 等等)分析成为一棵文档结构树.然后我们就可以对这棵树进行增加,修改,删除. 对这棵树里各节点的各种属性同样进行增加,修改,删除. 比如绑定 CSS 的 class 属性. 通过这些操作,一个 HTML 的展显就完全在我们控制当中. 我们可以动态的增加节点(比如<tr>, 等),也可以在运 行中控制节点是否可见(display:none)等等..

CSS 方面

CSS Cascading Style Sheet 层叠样式表单

如果说 HTML 是骨架和肉体,JavaScript 是思想与行为的话,那 CSS 就是衣服.

定义 CSS

三种

节点名 {各种样式属性} 如: p{} 作用范围:所有 p 节点, body{} 整个 body 节点, table{} 各所 table 节点

#节点 ID 名 {各种样式属性} 如: #myID{} 作用范围 ID 为"newNode"的节点.(只要 ID 为"newNode"就符合要求)

.样式名 {各种样式属性} 如: .newStyle{} 作用范围 所有元素(节点)的 class 属性包含了该样式名("newStyle")的节点.

总结:

以上是我对 JS 以及 CSS 的一些理解,更多知识还是有待自己进一步学习. JavaScript 推荐书籍.<<JavaScript 高级程序设计>> <<ppk 谈 JavaScript>>.

<<JavaScript 高级程序设计>> 这本要注意,因为很多学习 JavaScript 的书都叫这个名字,因为我看的是电子版,没有有关作者的信息,所以就把目录结构说一下,以方面我们在选择时有个考虑.

目录结构 第一章:JavaScript 是什么 第二章 ECMAScript 基础 第三章 对象基础,第四章 继承.

CSS 的书我真的没怎么看过,都是从网上找的资料.

好,现在进入我们今天的正题.

今天我从我自己的网站(http://www.kuigood.com)里拿了一个实例与大家一起分析.**在此申明一下, 不讨论代码写的好不好,只讨论如何实现,代码优化是一件很漫长的事情.呵.**

该实例的功能是我网站右上方排序方式的自适应显示是如何实现的. 记住**今天说的是选择排序方式时的自适应显示,而不是排序.**

目前我网站上对搜索出的淘宝商品的显示有三种排序方式,每种排序方式里又有一个可选的子排序. 我现在的讨论的功能是当一个主排序方式被选择,

那么对应的可选子排序就会显示出来提供给用户选择,其它的两个不显示. 呵,很简单吧. 但我们今天是用 JQuery 来实现.

效果如下:

三种方式的界面上的排版在这儿不多说,我把三种排序单选框以及对应的可选子排序复选框的 ID 在这儿说明一下.

方式一

单选框为 cbType1CreditDesc

复选框为 cbType1PriceAsc

复选框后的标签 labType1PriceAsc

方式二

单选框为 cbType2PriceAsc

复选框为 cbType2CreditDesc

复选框后的标签　labType2CreditDesc

方式三

单选框为 cbType3PriceDesc

复选框为 cbType3CreditDesc

复选框后的标签　labType3CreditDesc

JS 代码

```
//当网页被完全 LOAD 后所执行函数 load;

$(document).ready(function(){
        load();
});

function load(){

        var controlShow = function(){
           if ($("#cbType1CreditDesc").attr("checked") == true) {
               $("#cbType1PriceAsc").css("display", "inline-block");
               $("#labType1PriceAsc").css("display", "inline-block");
           }
           else {
               $("#cbType1PriceAsc").css("display", "none");
               $("#labType1PriceAsc").css("display", "none");
           }


           if ($("#cbType2PriceAsc").attr("checked") == true) {
               $("#cbType2CreditDesc").css("display", "inline-block");

               $("#labType2CreditDesc").css("display", "inline-
block");
           }
           else {
               $("#cbType2CreditDesc").css("display", "none");
               $("#labType2CreditDesc").css("display", "none");
           }

           if ($("#cbType3PriceDesc").attr("checked") == true) {
```

```
                $("#cbType3CreditDesc").css("display", "inline-block");

                $("#labType3CreditDesc").css("display", "inline-
block");
                $("#cbType3CreditDesc>lable").css("display", "inline-
block");
            }
            else {
                $("#cbType3CreditDesc").css("display", "none");
                $("#labType3CreditDesc").css("display", "none");
             }

          }

         //当方式一的卖家信用从高至低被选择后进行的操作
        $("#cbType1CreditDesc").click(function(){
            controlShow();

        });

        //当方式二的商品价格从低至高被选择后的操作
        $("#cbType2PriceAsc").click(function(){
            controlShow();

        });

        //当方式 3 的商品价格从高至代被选择后的操作
        $("#cbType3PriceDesc").click(function(){
            controlShow();

        });

}
```

蓝色代码的分析

因为三小段代码的功能结构是一样的,所以我只拿第一小段的代码来分析说明

```
        $("#cbType1CreditDesc").click(function(){
            controlShow();

        });
```

该段的代码有一个**新的知识点, JQuery 的事件绑定**

$().click(执行函数) 在每一个匹配元素的 click 事件中绑定一个处理函数。

$("#cbType1CreditDesc") 该代码是选择 ID 为 cbType1CreditDesc 的元素(DOM 里称为节点),最后结果是方式一的单选框.

那上面整段代码的意思是, 为 ID 为 cbType1CreditDesc 元素的 click 事件绑定一个函数 function(){controlShow();}

这是是 JQuery 对 click 事件绑定的二种方法中的一种,另一种的实现为

```
$("#cbType1CreditDesc").bind('click',function(){
          controlShow();

 }
```

第一种是以函数的方式现实 click 事件,另一种是以参数的方式来绑定

相关知识, **解除事件绑定**

$("#cbType1CreditDesc").bind('click'); 这样我们就解除了 id 为 cbType1CreditDesc 元素的 click 事件的绑定了.

桔红色代码的分析

同样是因为三小段代码的功能结构是一样的,所以只拿第一小段代码来分析

```
            if ($("#cbType1CreditDesc").attr("checked") == true) {
                $("#cbType1PriceAsc").css("display", "inline-block");
                $("#labType1PriceAsc").css("display", "inline-block");
            }
            else {
                $("#cbType1PriceAsc").css("display", "none");
                $("#labType1PriceAsc").css("display", "none");
            }
```

该段的代码有二个**新的知识点, JQuery 的元素属性.**

$().attr("属性名") 取得第一个匹配元素的属性值。通过这个方法可以方便地从第一个匹配元素中获取一个属性的值。如果元素没有相应属性，则返回 undefined 。

$().css("样式属性","属性值") 在所有匹配的元素中,设置一个样式属性的值

那第一小段代码的意思就是

当 ID 为 cbType1CreditDesc 的元素的属性 checked 为 true(也就是说被选定了)时 为 ID 为 cbType1PriceAsc 的元素设置名为 display 样式属性的值为

"inline-block"(**显示**),如果该元素未被选择则设置该元素的 display 样式属性的值为"none"(**不显示**)

实战四

不好意思啊，离上一篇文章已经过去 12 天了，自己的事情太多，我这儿的天气也太热了 36~7 度．无法心静啊.呵．好了.Let's moving.

在我博客里有朋友留言说 JQuery 的 API 他都看了,但在实际应用时不知道如何下手了．　嗯...　这个问题提的很好,我认为也具有普遍性，我在学 JQuery 中并没有遇到这种感觉，当然我不是夸自己多牛 X，因为我从学 JQuery 的第一天起就一直在"实战",就一直想着怎么把 JQuery 应用到我的项目里去.

以下是我个人对于学 JQuery 的一些想法，供朋友参考.

一，JQuery 只是工具．(抛弃对 JQuery 的畏惧)

JQuery 其实与我们平时用的 JavaScript 没有根本上的区别，只不过是一样优化后的工具．要实现同样的功能,JavaScript 可以实现,JQuery 也能实现.只不过表现形式(代码)不一样罢了．所以我相信能用 JavaScript 写代码的朋友换成 JQuery 也是很快的.

二，多多练习,熟悉 JQuery 的语法．(熟悉 JQuery)

很多知识当我们在书上看后或者听别人讲解后都感觉自己似乎懂了,但到自己去做时却发现自己根本没懂．做为一个老程序员,我对这个感受颇深.

那怎么练习?

1，自己想些功能,然后用 JQuery 实现．(这个用的人比较少)

2，把自己以前的或者现在正在开发中的代码换用 JQuery 来实现 (个人比较推荐这个)，除了现在快购利众网 ,以前还有一个 GoogleMap 的网站我也是用 JQuery 重新写过主要功能．也正是在这些修改过程中,让我更加深刻的理解了 JQuery 的语法,使用,注意事项等．　这也不正是**重构**的开发思想吗?

好，进入我们今天的正题，实战开始.

前三篇文章说的都是有关"选择器"方面的知识, 虽然还有很多地方没有讲到, 但我想"选择器"方面剩下的知识点大家都能搞定. **今天我准备和大家一起交流一下 JQuery 中的"Ajax". 还是以前那句话,JQuery 很容易的.**

JQuery 中的 Ajax

在开始之前我还得事先申明一下,我对 Ajax 的理解目前只限于网页与服务器端动态交互这一程度,如果下面文章有理解有误或者不当的地方请大家直接指出来.

用 JavaScript 实际过 Ajax 功能功的朋友们一定不会忘记使用 XMLHttpRequest 对象的步骤,先生前 XMLHttpRequest 对象,如果浏览器不支持还要使用 ActiveXObject, 然后就是绑定回调函数,接下来.open 操作,有时还要设置 请求头 然后 .send 操作. 步骤太多了. 如果使用 JQuery 就方便多了. 当然使用 JQuery 时也不要忘记,JQuery 的背后其实也是按上述步骤来实际的.

JQuery 中有关 Ajax 的操作方法有很多,但我一般只有 $.ajax()

好,现在就用一个实际来讲解这个方法的使用.

快购利众网下面有一条主题广告,这广告其实是在网页载入后再从服务器端取回具体数据. 下面是实现代码.

第一种实现

```
//取新的主题图片    getNextTheme 其实是一个 JavaScript 的类
function getNextTheme(){
}

//完成后的回调函数,其实也可以不用这个,JQuery 还提供两个更简单的

getNextTheme.prototype.callback_Complete = function(XMLHttpRequest,
textStatus){
    if (XMLHttpRequest.readyState == 4) {
        var data = XMLHttpRequest.responseText;
        this.instance.finished(data);
    }
    else {
        this.instance.finished("搜索线程发生错误" + textStatus);
    }
}


getNextTheme.prototype.finished = function(data){
```

```
        //对服务器返回的数据 data 进行处理,因为处理代码与本课无关,所以忽略.

        // 对 data 进行处理
    }


getNextTheme.prototype.begin = function(){

    $.ajax({
        instance: this,
        url: PROXY_URL_THEME, //这个就是远程用来处理请求的页面,可以是PHP 写的,也可以是 JSP,也可以是 ASP
        cache: false,
        async: true,
        dataType: "html",
        timeout: 15000,
        complete: this.callback_Complete
    });
}

$(document).ready(function(){
        var getTheme = new getNextTheme();
        getTheme.begin();

});
```

代码分析

首先得说明一下, 以代码其实是一个 JavaScript 中的类. getNextTheme 就是这个类的类名

```
var getTheme = new getNextTheme();
getTheme.begin();
```

先生成 getNextTheme 的实例,然后调 begin()方法就开始向服务端"要"数据了.

具体有关 JavaScript 类的知识请朋友自行搜索,查找.

橙色代码解析:

```
    $.ajax({
        instance: this,
```

```
        type:"GET",
        url: PROXY_URL_THEME, //这个就是远程用来处理请求的页面,可以是
PHP 写的,也可以是 JSP,也可以是 ASP
        cache: false,
        async: true,
        dataType: "html",
        timeout: 15000,
        complete: this.callback_Complete
    });
```

type: 选择是以"GET"或"POST"的方式向服务器发送数据.

url: 服务器端的接收 URL

cache: 会不会从浏览器缓存中加载请求信息, 选择有 true,false,默认 false

async: (默认: true) 默认设置下，所有请求均为异步请求。如果需要发送同步请求，请将此选项设置为 false。注意，同步请求将锁住浏览器，用户其它操作必须等待请求完成才可以执行。

dataType:预期服务器返回的数据类型

timeout: 设置超时,单秒毫秋季

complete: 请求完成后回调函数 (请求成功或失败时均调用)。

还有更多更详细的参数说明请大家自己参考 JQuery 手册, 上面讲的比我说的还要详细.

知识点

在这儿有一个 instance 参数,这个参数是我自己"造"的, 我们除了使用 JQuery 内置的参数外还可以自己定义一些参数.instance 这个参数就是我自己定义的, 怎么定义? 直接写上去就是了,参数可以自己命名.

这个 instance 里放的东西可不会发送到服务端哟, 这个参数里的数据我**是给回调函数使用**的.

为什么要这样做? 因为当请求被响应后 JQuery 就会去调用回调函数, 这时回调函数被传入的参数只有 JQuery 预定给出的类型, 有时我们是需要使用

自己的数据,怎么办? 可以在 $.ajax({}) 中把我们的数据放进去,这样回调函数就能直接访问到了.

instance 使用代码如下:

```
getNextTheme.prototype.callback_Complete = function(XMLHttpRequest,
textStatus){
    if (XMLHttpRequest.readyState == 4) {
        var data = XMLHttpRequest.responseText;
        this.instance.finished(data);
    }
    else {
        this.instance.finished("搜索线程发生错误" + textStatus);
    }
}
```

这里面的 **this.instance** 就是我们在$.ajax({})中给入的 **instance**. 记住,名字可以自己取,可以是 abc,也可以是 myVar.

第二种实现

```
//取淘宝主题广告 getNextTheme 其实是一个 JavaScript 的类
function getNextTheme(){
}

//响应成功的回调函数

getNextTheme.prototype.callback_Success = function(data){
        this.instance.finished(data);
        this.instance.finished("搜索线程发生错误" + textStatus);
}

//响应失败的回调函数

getNextTheme.prototype.callback_Error = function(){
        this.instance.finished("搜索线程发生错误" + textStatus);
}


getNextTheme.prototype.finished = function(data){
      //对服务器返回的数据 data 进行处理,因为处理代码与本课无关,所以忽略.

      // 对 data 进行处理
 }
```

```
getNextTheme.prototype.begin = function(){

    $.ajax({
        instance: this,
        url: PROXY_URL_THEME, //这个就是远程用来处理请求的页面,可以是
PHP 写的,也可以是 JSP,也可以是 ASP
        cache: false,
        async: true,
        dataType: "html",
        timeout: 15000,
        success: this.callback_Success,

        error:this.callback_Error


    });
}

$(document).ready(function(){
        var getTheme = new getNextTheme();
        getTheme.begin();

});
```

代码分析:

因为其它的都没变化,就$.ajax({})中的参数有小小的变化,所以我只说说这个变化.

success: this.callback_Success 如果成功响应则调用名为 callback_Success 的回调函数

error:this.callback_Error 如果无响应或者响应有问题则调用名为 callback_Error 的回调函数