jQuery语法

（1）引用jquery文件及下载库：

<http://jquery.com/download/>

下载

[Download the compressed, production jQuery 2.2.2](http://code.jquery.com/jquery-2.2.2.min.js)

这个是用户版的，已经被精简和压缩。

然后使用

<script src="jquery.js"></script>

来启动这个库文件，记得将下载的文件重命名为jquery.js

也可以使用谷歌和微软的CDN，不过这里略。

（2）jQuery语法

**$ (selector).action()**

其中$是必须的，表示使用jQuery语法；

selector表示你要对哪个进行操作；

action()表示进行的操作；

例如：$(“div”).hide() 其表示对所有div标签包含的内容进行隐藏，如果是<”p”>则表示对p标签包含的东西进行隐藏。

**其他：**

$(this) 表示对当前HTML元素进行操作

$(“#abc”) 表示对id为abc的元素进行操作

$(“.abc”) 表示对class=”abc”的进行操作

（3）$(document).ready

这个存在的意义在于，只有当文档加载完毕（不确定是指所有文档，还是指使用的文档的那一部分），才能运行jQuery代码

（4）选择器

①$(“p”) 选择元素为<p>的

②$(“p.abc”) 选择class=”abc”的<p>元素

③$(“p#abc”) 选择id=”abc”的<p>元素

④$(“[href]”) 选择所有带有href属性的元素

⑤$(“[href=’#’]”) 选择所有带href值等于“#”的元素

⑥$(“[href!=’#’]”) 选择所有带href值不等于“#”的元素

⑦$(“[href$=’.jpg’]”) 选择所有带href，末尾以.jpg结尾的元素

more：

|  |  |
| --- | --- |
| **语法** | **描述** |
| $(this) | 当前 HTML 元素 |
| $("p") | 所有 <p> 元素 |
| $("p.intro") | 所有 class="intro" 的 <p> 元素 |
| $(".intro") | 所有 class="intro" 的元素 |
| $("#intro") | id="intro" 的元素 |
| $("ul li:first") | 每个 <ul> 的第一个 <li> 元素 |
| $("[href$='.jpg']") | 所有带有以 ".jpg" 结尾的属性值的 href 属性 |
| $("div#intro .head") | id="intro" 的 <div> 元素中的所有 class="head" 的元素 |

（5）事件触发条件

①ready是完成后触发，一般用于文档。$(document).ready(执行的命令)

②click是点击后触发，一般用于按钮，也可以用于点击文档某区域。$(selector).click(执行的命令)

③dclick是双击后触发。$(selector).dclick(执行的命令);

④focus是元素获得焦点时触发的事件；

⑤mouseover是鼠标悬停后触发的事件；

更多事件触发参考：

<http://www.w3school.com.cn/jquery/jquery_ref_events.asp>

（6）名称冲突

例如jQuery和某个库名称冲突了（例如同样适用$符号作为开始），那么使用这个命令；

**var jq=jQuery.noConflict()**，帮助您使用自己的名称（比如 jq）来代替 $ 符号。

（7）解释

$(document).ready(function(){

$(".ex .hide").click(function(){

$(this).parents(".ex").hide("slow");

});

});

第一行是，当文件准备好了，执行函数，函数内容是后面的：

第二行是，class=”ex”的元素中的class=”hide”元素，点击后触发函数，函数内容为第三行；

第三行是，当前元素（this）的所有祖先元素（parents，一直到根元素html为止）中class=”ex”的元素，被隐藏（hide），速度为缓慢（slow）

疑问：

当有两个同样的东西时，为什么只隐藏自己当前这个，不隐藏另外一个？

推测是因为只遍历其祖先，不包含自己（且祖先中符合class=”ex”要求的）

（8）隐藏和显示

隐藏是hide()，而显示对应的是show()，

只要将隐藏的hide()改为显示的show()即可。

还可以规定速率：

速率有三种，slow（慢），fast（快），数字（毫秒）。

例如hide(“slow”); show(“fast”); hide(2000);

需要注意的是：无参数为瞬间完成，fast为快（但比瞬间慢），slow为慢，数字无需加引号。

另有toggle，用法同show和hide，效果是，如果是隐藏，点击后显示，如果是显示，点击后隐藏。也可以加时间和速度参数。

问题：假如多次点击，他会执行完每一次的行为，而不是只执行最近一次

（9）淡入和淡出

注意大写，对大小写是敏感的。

fadeIn(); //淡入（显示）

fadeOut(); //淡出（隐藏）

fadeToggle(); //类似toggle，如果隐藏则显示，如果显示则隐藏

fadeTo(速度, 透明度0~1, 返回函数); //可以调整不透明度，注意，不能大于1

如：

$("#img\_1st").fadeTo(1000,0.3);

以1000ms的速度，将id=”img\_1st”的东西设置为透明度30%

注：

假如有多个淡入同时执行，则分别执行自己的，而不是逐个执行。除非有设置设置其执行顺序。

更多特效参考；

<http://www.w3school.com.cn/jquery/jquery_ref_effects.asp>

（10）滑动

说明：非隐藏的情况下，滑动后就是隐藏了，隐藏的情况下，相反的滑动就是显示。

向上滑动：slideUp();

向下滑动：slideDown();

切换滑动（先向上，再点击则向下）：slideToggle()

其他效果：

<http://www.w3school.com.cn/jquery/jquery_ref_effects.asp>

（11）显示

display:none;

在css的style设置，这种设置为初始不显示。

（12）动画

$(selector).animate({params},speed,callback);

第一个是必须是，表示形成动画的CSS属性，简单的来说，例如初始字号大小为10px，然后这里填写50px，那么字体就会自动变为50px（这个变化的过程就是动画）

注意：

①这个可以操纵块，但是似乎是不能操作<p>标签。

②可以操纵CSS属性；

③移动位置的话，需要在style里这么设置：position:fixed

一些使用：

$(document).ready(function(){

$("#b\_3rd").click(function(){

$("#a\_3rd").animate(

{left:'550px', //左边的位置变为550px

width:'100px', //宽度变为100px

top:'-=200px', //向上的位置减少200px

fontSize:'50', //设置字体大小（注意没有-），并且S大写

});

});

});

并且由于top使用的是相对位置，因此每次点击top，都会移动位置

还可以设置为属性，例如：

width:’toggle’ //宽度形式的隐藏和显示切换

hide隐藏show显示。

不过貌似不能淡入淡出和改变透明度，是因为淡入淡出本身就是动画么？

队列：

$(document).ready(function(){

$("#b\_3rd").click(function(){

$("#a\_3rd").animate({left:'550px'}); //左边的位置变为550px

$("#a\_3rd").animate({width:'100px'});//宽度变为100px

$("#a\_3rd").animate({top:'-=200px'});//向上的位置减少200px

$("#a\_3rd").animate({fontSize:'50'});//设置字体大小（注意没有-），并且S大写

});

});

这种写法就是让其按顺序执行先移动位置，然后宽度变化，然后往上移动，再字体变大

（13）停止动画

语法：

$(selector).stop(stopAll,goToEnd);

假如有动画队列1、2、3、4，正在执行1；

默认无参数是1停止，下来执行2，3，4；

第一个参数存在表示1、2、3、4都停止；

第二个参数表示1停止，但是立刻出结果

参数用true和false来表示。

①false, false表示1停止，执行2、3、4；

②false, true表示直接完成1，然后执行2，3，4；

③true, false表示1停止，234取消；

④true,true 表示1直接出结果，然后其他停止；

如代码：

<divstyle="position:fixed"id="a\_3rd">

<p>用于测试动画的文字</p>

</div>

<div>

<pstyle="text-align:center">

<buttonid="b\_3rd"align="center">点击后左面的文字会移动</button>

<buttonid="b\_4th"align="center">点击停止旁边按钮的动画</button>

</p>

</div>

以下是对其进行操作：

$(document).ready(function(){

$("#b\_3rd").click(function(){

$("#a\_3rd").animate({left:'550px'},2000); //左边的位置变为550px

$("#a\_3rd").animate({width:'100px'},2000);//宽度变为100px

$("#a\_3rd").animate({top:'-=200px'},2000);//向上的位置减少200px

$("#a\_3rd").animate({fontSize:'50'},2000);//设置字体大小（注意没有-），并且S大写

});

$("#b\_4th").click(function(){

$("#a\_3rd").stop(true); //停止其的一切动画队列

});

});

（14）CallBack函数

称为回调函数，

其作用是，当前动画执行完100%后，执行该函数。

可以理解为，当某条件满足之后，执行该函数。

**（15）AJAX方法（异步请求）**

具体来说，是load()方法，语法是：

$(*selector*).load(*URL*,*data*,*callback*);

例如读取某个链接或者本地文件：

$("#a\_3rd").load("a.txt"); //这是标签为#a\_3rd的读取文本文档a.txt

①URL是希望加载的URL（链接）；

②data参数规定与请求一同发送的查询字符串键/值对集合（不懂，感觉是说这个是查询的时候顺便发个信息给对方）；

③callback是结束请求后执行的函数；

可以在URL里面加入选择器（例如#是选择id的，.是选择class的），来选择某一部分的内容（例如js文件的某个标签的）。

如：

load('/example/jquery/demo\_test.txt #p1') //读取id=”p1”的标签内的

也可以把#p1改为h2，读取h2标签内的

**load的返回值（注意大小写）：**

responseTxt 包含调用成功时的结果内容

statusTxt 包含返回状态，如果成功是success，如果失败是error

xhr 包含XMLHttpRequest对象（不懂）

**（16）get和post——两种http请求方法**

get——从指定的资源请求数据；

post——向指定的资源提交要被处理的数据；

**GET方法：**

查询字符串（名称/值 对）是在GET请求的URL中发送的；

①GET请求可被缓存；

②GET请求保留在浏览器历史记录中；

③GET请求可被收藏为书签；

④GET请求不应在处理敏感数据时使用；

⑤有长度限制；

⑥只应当用于取回数据；

**POST方法：**

①不会被缓存；

②不会被保存在浏览器记录中；

③不能被收藏为书签；

④对数据长度没有要求；

|  |  |  |
| --- | --- | --- |
|  | **GET** | **POST** |
| 后退按钮/刷新 | 无害 | 数据会被重新提交（浏览器应该告知用户数据会被重新提交）。 |
| 书签 | 可收藏为书签 | 不可收藏为书签 |
| 缓存 | 能被缓存 | 不能缓存 |
| 编码类型 | application/x-www-form-urlencoded | application/x-www-form-urlencoded 或 multipart/form-data。为二进制数据使用多重编码。 |
| 历史 | 参数保留在浏览器历史中。 | 参数不会保存在浏览器历史中。 |
| 对数据长度的限制 | 是的。当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）。 | 无限制。 |
| 对数据类型的限制 | 只允许 ASCII 字符。 | 没有限制。也允许二进制数据。 |
| 安全性 | 与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分。  在发送密码或其他敏感信息时绝不要使用 GET ！ | POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 web 服务器日志中。 |
| 可见性 | 数据在 URL 中对所有人都是可见的。 | 数据不会显示在 URL 中。 |

GET语法：

$.get(*URL*,*callback*);

URL是必须有的，表示链接；

callback是 **请求成功** 后执行的回调函数；

POST语法：

$.post(*URL*,*data*,*callback*);

data是连通请求发送的数据。

以下是收集的资料：

（1）GET用于信息获取，而且应该是安全的和幂等的。

“安全”大概意思是：

①不安全的信息（敏感的信息）不应该由GET获取；

②GET主要用于获取信息，而非修改信息；

③GET请求一般不应产生副作用；

“幂等”大概意思是：

①对同一个URL的多次请求的结果是相同的。

例如想要获得一个信息，多次请求，应该总能获得这个信息（而非突然变为另外一个信息）；

GET请求的数据会附在URL之后（把数据放在HTTP协议头中），以“?”分割URL和传输数据，参数之间以“&”相连，如：

login.action?name=hyddd&password=idontknow&%E4%BD%A0%E5%A5%BD

如果数据是英文字母/数字，则原样发送；

如果是空格，则转换为+；

如果是中文/其他字符，则把字符串用BASE64加密，得出%E4%BD%A0%E5%A5%BD，其中%XX中的XX为该符号以16进制表示的ASCII值

（2）POST表示可能改变服务器上的资源请求

①例如对某论坛的帖子进行回帖，服务器上的该贴多了一个回帖。因此应该通过POST来实现；

POST把提交的数据放置在HTTP包的包体之中。

（3）因此，POST的安全性（关于丢失敏感信息）比GET高，传输的数据更多一些，但是更复杂一点（因为data属性，要发一个表单）。

**（17）查看复选框是否被选中**

复选框为：

<input type="checkbox" name="yingmingliu" id="yingmingliu" value="yingmingliu"/>

jq语法为：

$("#yingmingliu").is(':checked')

如果选中，返回true，如果没有被选中，返回false。

**查看复选框的值：**

$("#yingmingliu").attr('value')

返回value

**两个复选框互斥：（选一个另外一个自动失效）**

$(document).ready(function()

{

$("#guluan").focus(function(){ $("#meiyinxiang").attr("checked",false);})

})

$(document).ready(function()

{

$("#meiyinxiang").focus(function(){ $("#guluan").attr("checked",false);})

})

效果：

①这是2个复选框按钮（checkbox）；

②当其中一个被选择时（成为焦点focus），另外一个被自动设置为未选择；

**使用attr的潜在问题：**

从jquery的1.9.1版本至2.0.3版本，若使用attr来设置checkbox，那么将无法多次重复操作。具体而言，第一次设置是正常的（如将checked取消），第二次设置就会失败（恢复checked）。因此，应使用prop替代attr。

另：经测试，v2.2.2版本是正常的。

**（18）如何设置某div块A一直在另一div块B的下部（这个应该属于CSS）**

首先：二者都在同一父块下，该父块class为B块的class（不确定是否必要）；

如：

<div class="dam"><!--显示区域-->

<div id="dam" class="dam"></div>

<div id="lastdam" class="lastdam">

</div>

其次，设置css属性

div .dam

{

float: left;

color:red;

font-size:20px;

text-align:left;

width:270;

}

div .lastdam

{

float: left;

color:blue;

font-size:20px;

text-align:left;

**width:100%;**

}

关键为红色部分。

然后就可以了。

**（19）利用jquery修改css属性（以display演示）**

代码：

$("#maincontent\_dafengche").css('display','none');

效果：

将ID为maincontent\_dafengche的css属性display的值，设置为none

**（20）使用jq的name选择器**

代码

$("[name='maincontent']").css('display','none');

效果：

选择所有name=" maincontent"的标签

**（21）同时选择多个标签，并对其进行操作**

代码：

function selectAll() //初始3个复选框被勾选

{

var objs = document.getElementsByName('twId');

var i;

for(i = 0; i < objs.length; i++)

{

if(objs[i].type == "checkbox")

{

objs[i].checked = true;

}

}

}

效果：

①选择所有name="twId"的项；

②如果其类型是复选框（checkbox）；那么将其的按钮类型设置为true；（被选择状态）；

③说明其是一个数组（变量objs），选择其中一个成员时使用objs[i]（i为数字）；

④推测其是根据name='twId"的顺序来依次成为数组成员的（但若是通过js或dojo添加的标签，那么不确定其顺序，也许也是根据其在添加后的位置，决定其顺序的？）；

**（22）确定一个数组的成员个数**

假如对象数组名是obj

那么其成员个数是obj.length

**（23）利用jquery来ajax异步加载文本**

**语法：**

$(*selector*).load(*URL*,*data*,*callback*);

代码：

$(document).ready(function()

{

$("#aa").click(function()

{

$(".another").load('a.txt');

})

})

html代码：

<div class="one">测试块</div>

<div class="another">对比块</div>

<input type="button" id="aa" value="点击启用ajax">

效果

①点击按钮后，读取同目录下的a.txt文件；

②用a.txt文件的内容，替代class="another"的标签的内容；

③假如有多个class="another"的标签，则每个都会被替换；

**进阶：加载文件中某一部分**

将代码更改为：

$(".another").load('a.txt #a');

效果：

①读取文件中id="a"的标签；

②如文件内容为：

你很帅！！

<div id="a">我很帅~~~</div>

则只读取第二行内容；

**进阶：回调函数**

将代码更改为：

$(".another").load('a.txt', function(responseTxt,statusTxt,xhr)

{

alert(responseTxt);

});

效果：

①函数为回调函数（load后执行的函数）；

②第一个参数为返回的文本（全部的文本，即使使用'a.txt #a'，显示的依然是全部的，而不是只有一部分），参数名可更改；

②第二个参数为load是否成功的状态。调用成功的话，其statusTxt值为“success”（字符串），失败的话为error。

③第三个是一个对象（object），其中xhr.status表示load的状态，xhr.statusText表示错误的情况是什么。

**（24）使用console来记录错误**

代码：

console.log(xhr.statusText);

将错误文本记录在日志里（可以利用火狐的开发者工具来显示）；

另外还有四种方法：

　　console.info("这是info");

　　console.debug("这是debug");

　　console.warn("这是warn");

　　console.error("这是error");

具体尚不是很清楚之间的区别，但可以用（至少log和error都能显示）

**（25）使用AJAX加载外部html后，使用加载到的按钮触发js事件**

**情况：**

①假设有一外部html，<div id="a"><input type="button" id="bb" value="我很帅~~~"></div>；

②通过ajax加载这部分内容；

③想通过点击加载的id为"bb"的按钮，触发某个js事件；

**问题：**

①常规写法是不行的，无法触发，根据查询到的资料，似乎是因为加载时便已经确定了。

如以下代码 **不可行**：

$(document).ready(function()

{

$("#bb").click(function()

{

alert("1");

})

})

**解决办法：**

①使用on写法，其选择器应为body或者document；

②代码例子：

$("body").on("click", "bb", function()

{

alert("1");

})

即可。其中第一个参数为click（事件），第二个参数为触发该事件的标签（bb），第三个参数是回调函数（触发事件后执行的内容）；

**（25）鼠标事件修改背景图片（针对div）**

img的比较简单，直接写

<img src="1.png" onMouseOver="src='2.png'" onMouseOut="src='1.png'">

即可

而div块的比较复杂。联合jq使用

**HTML代码：**

<div onMouseOver = "sideMouseOver(this.id)" onMouseOut = "sideMouseOut(this.id)">

**jQuery代码：**

function sideMouseOver(obj)

{

object = document.getElementById(obj);

object.style.cssText = "background:url(12.png)";

}

function sideMouseOut(obj)

{

object = document.getElementById(obj);

object.style.cssText = "background: ";

}

即可

**更简单写法：**

同样是上面的要求，鼠标移动上去的时候更改背景图片，鼠标挪走后取消。

①假设class="side" ，首先设置side的CSS属性，默认是没有背景图片的；

②再设置.side:hover{ background:url(xxx.png) } 即可；

③效果是，鼠标移动上去有变化，移动走变回来。

**（26）让某块随着浏览器的高度变化而自动变化**

假设该块的标签名为section

于是：

function ge()

{

var m = document.documentElement.clientHeight;

$("section").css("height",m-100);

}

window.onresize = function(){ge();} //浏览器高度变化时触发事件

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

效果：

①初始时自适应；

②浏览器高度变化时自适应。

③假如需要根据id来选择，更改$("section")即可。

④m-100是因为该块的上部有一个高度为100px的块。

**(27)按钮和标签绑定（按顺序绑定）触发事件**

前提：

假设有5个按钮，有5个事件。按钮的类名相同，事件的类名相同。按第一个按钮触发第一个事件，依次类推。

代码：

$(document).ready(function()

{

$(".side").click(function()

{

$(".iframe").css("display","none");

sles = document.getElementsByClassName("side");

var a;

for(a=0;a<sles.length;a++)

{

if(sles[a].id==this.id)

{

break;

}

}

var oo = document.getElementsByClassName("iframe");

oo[a].style.display="block";

})

})

**（28）初始不赋值src属性，后续触发事件时加载**

**适用范围：给标签添加src属性、单页面结构**

思路：

①首先，我们需要获得一个src的链接地址；

②某一个标签（例如iframe）需要使用这个链接（但不是即时使用，比如在点击某个按钮时使用）；

③我们通过javascript的class选择器（或者类似的），获取其标签（document.getElementsByClassName(xxxxx)），然后给标签添加一个value属性（即使该标签本身没有value属性），用于在使用该src地址之前，保存该链接地址；

④当触发事件时，将value属性赋值给src属性，即可。

代码：（使用了jquery库）

$(document).ready(function()

{

var oo = document.getElementsByClassName("iframe");

oo[0].value="http://www.baidu.com";

oo[1].value="https://www.so.com";

})

$(document).ready(function()

{

$(".side").click(function()

{

$(".iframe").css("display","none");

sles = document.getElementsByClassName("side");

var a;

for (a = 0; a<sles.length; a++)

{

if(sles[a].id == this.id)

{

break;

}

}

var oo = document.getElementsByClassName("iframe");

if(oo[a].src.indexOf(oo[a].value)<0) **//当索引值小于0时，说明src和value值不匹配**

{

oo[a].src=oo[a].value; **//将value值赋给src值**

}

})

})

解释：

①第一个函数用于模拟：当文档加载完毕后，给标签赋值（或者是触发事件，添加了该标签后赋值）；需要注意的是，标签个数是不确定的，确定的是，在导入标签时，一个iframe标签对应一个src地址和一个按钮；

②由于需要在按钮点击时，才触发事件，赋给对应的iframe标签src值。因此，首先要确定点击的是哪个按钮，然后找到该按钮对应的iframe标签；（就是第二个函数的a变量的值表示是第a+1个按钮/标签，

③然后确定该标签目前的src值，通过.indexOf()和value值进行匹配。

如果不匹配（返回值-1），那么将value值赋给该src（即加载链接地址，注意，其他的是没有被加载的）；

如果匹配（返回值大于0），那么跳过。

④于是实现了在触发事件才加载链接地址。

**（29）监视hashchange**

①什么是hashchange？

简单来说，假设我们设置一个超链<a href="#top">m1</a>

这个超链的作用，是跳转到页内id="top"处（注意，不是指页面顶端，而是页面内标签等于top的地方）

此时，在浏览器的地址处，最后会添加一个#top

②所谓的hashchange，就是监视这个链接的变化。

具体如下：

**$(window).on("hashchange",function(){回调函数});**

③效果是：当hashchange事件触发后，将调用回调函数。

④查看当前后缀地址（指的是例如点击上面的超链接后，在浏览器地址后面所加的#top），使用代码：

**window.location.hash**

以上面为例，返回值是：#top

⑤由于我们有时候不想要第一个“#”字符，于是，我们再使用命令：

**x=window.location.hash.substr(1);**

返回的是去除#之后的所有字符串。

**（30）利用hashchange()监测，实现单页面技术**

代码：

$(window).on("hashchange",function()

{

x=window.location.hash.substr(1);

y="iframe\_"+x;

m=document.getElementById(y);

if(m==null) //如果找不到，则添加该标签

{

$("#iframe\_section").append(function(){return "<iframe id='" + y + "' class='iframe' src='http://www." + x + ".com'>ff</iframe>"});

}

$(".iframe").css("display","none"); //设置所有为none

$("#"+y).css("display","block"); //读取id，修改css属性

});

解释：

①加入的iframe框架，都在一个id为iframe\_section的块之下。（所以必须要有这个块，或者修改这个id）；

②监测hash地址的变化，如果有变化，首先检查是否有等于该地址的id的块（例如地址是#abc，那么监测是否有id="abc"的块）；

③如果没有，创造一个这样的块，并给其赋予链接（链接地址是id，具体格式见src，可以根据实际需要修改）；

④有、或者创建之后，设置所有类名为.iframe的块的display属性为隐藏（倒数第二行代码）；

⑤然后设置id为hash地址（实际上这个id有一个前缀iframe\_ ，是在前面加入的）的块的display属性为block（显示这个块）。

⑥综合效果是：点击一个链接，哈希地址变化，显示对应的iframe或者创造他，不符合的被隐藏。切换不同的哈希地址时，不会导致之前操作的哈希地址的页面被重新加载。

**另一种实现方式：**

简单解释一下，是iframe和其按钮按顺序一一对应。不需要影响哈希地址，但相对实现就更复杂一些。

JS代码：

/\*功能是点击某个标签，显示该标签对应的iframe

原理：

1.先隐藏其他的，标签对应iframe逐个对应（即标签1对应1#iframe），然后利用其对应顺序，先找到标签是第N个，然后找到第N个iframe，并显示他

2.需要标签的类名是side，iframe的类名是iframe

3.假如标签开始没有被加载的话，读取其value属性，将其值赋给src属性

4.可能存在的缺陷：假如某iframe打开后，其链接地址在操作过程中被改变，name返回时，再重新加载原链接

\*/

/\*$(document).ready(function()

{

$(".side").click(function()

{

$(".iframe").css("display","none");

sles = document.getElementsByClassName("side");

var a;

for (a = 0; a<sles.length; a++)

{

if(sles[a].id == this.id)

{

break;

}

}

var oo = document.getElementsByClassName("iframe");

if(oo[a].src.indexOf(oo[a].value)<0)

{

oo[a].src=oo[a].value;

}

oo[a].style.display="block";

})

})

//这个是【模拟】在加载iframe标签的时候，给iframe标签赋值（value）

$(document).ready(function()

{

var oo = document.getElementsByClassName("iframe");

oo[0].value="http://www.baidu.com";

oo[1].value="https://www.so.com";

})\*/

HTML代码①按钮部分：

<div class="side" id="nav\_pane">

<img src="11.png" class="aside\_icon">

<span>我的工作台</span>

</div>

<div class="side" id="aside\_newcase">

<img src="13.png" class="aside\_icon">

<span>新建案件</span>

</div>

HTML代码②iframe部分：

<section id="iframe\_section">

<iframe class="iframe none" src="">

</section>

CSS部分：

.display\_block

{

display:block;

}

.display\_none

{

display:none;

}

注：实际使用的话，可能需要调整一下布局问题。并且本代码省略掉了实时创建iframe标签部分的代码，如果需要实时创建的话，需要在创建后，将src地址赋值给对应的iframe标签（通过获得标签对象后，加入到其value属性中）

**（31）iframe框架内触发事件影响父窗口**

简单来说，在iframe里点击一个按钮（触发一个事件），父窗口可以同时触发一个事件。

①最简单的例子：（修改父窗口某个元素的属性）

假设父窗口里有一个id="abc"的元素，iframe里有一个id="qwe"的按钮。

在点击qwe这个按钮后，父窗口abc这个元素的背景颜色被修改。

代码如下（放在iframe所在的页面中）：

$(document).ready(function(){

$("#qwe").click(function(){

$(window.parent.document).find("#abc").css("background-color", "red");

});

})

点击按钮，父窗口颜色变为红色。

很奇怪的是，可以修改css属性，可以remove，但是不能触发change()事件？可是我之前好像是触发成功过？我不知道为什么。

②进阶：

由于不知道为什么，我之前能成功触发change事件，但是后来又不行了。

所以另外一个变种方法就是修改父窗口某个元素的属性，利用jquery监视这个事件，当触发时调用一个回调函数即可。

~~假如我们想做更多的事，通过这个似乎并不直观，而且，把代码都放在子元素，未免有时候会有些不友好（例如我们有5个iframe，一次只加载一个，但都想触发同一个事件）。~~

~~如果放在子元素的话，要分别对5个子元素进行设置，但放在父元素的话，只需要设置一次即可。~~

~~因此，换个办法，iframe触发一个事件，然后父元素响应这个事件，然后可以执行一些代码做一些事情。~~

~~如代码~~

~~【iframe】：~~

~~$(document).ready(function(){~~

~~$("#qwe").click(function(){~~

~~$("#abc", parent.document.body).change();~~

~~});~~

~~})~~

~~【父元素】：~~

~~$("#abc").change(function(){~~

~~【做一些想做的事情】~~

~~})~~

~~解释：~~

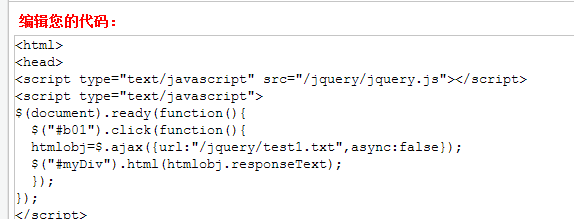
~~子元素qwe的点击，触发父元素id="abc"的change事件（abc可以是隐藏的，看不见的，只用于触发事件使用）；~~

~~父元素监视abc的change事件，当触发时，执行我们预先设定的代码。~~

~~这样的话，iframe就可以起到对父元素的操作效果了！~~

**(32)利用ajax来读取带html标签的json文件**

假如我们需要像w3school那样，在某个区域显示带html标签的内容（如下图）；

我们已知的方法有通过转义字符来显示。

但是假如这些内容并不是直接写在页面上的，而是通过读取另一个存放有这些文字的文件来获得，那么并不是那么简单。

例如，通过jquery的load方法是没有办法简单获取的。（例如$("#textarea").load("xxx/xxxx.json", function(data){....})这样是不可以的。

我们可以确定是需要通过ajax来解决这个问题，通过查询，我采用直接使用**$.ajax()**来解决。该方法说明见链接：

http://www.w3school.com.cn/jquery/ajax\_ajax.asp

**具体做法：(同步，旧）（异步请看后面）**

①目标html标签为：<textarea id="area\_1"></textarea>

②事件：点击<button id="html\_normal">html</button>触发；

③调用文件：路径（txt/tag\_query.json），文件内容：

<!DOCTYPE HTML>

<html>

<head>

<meta charset="UTF-8">

<link rel="stylesheet" type="text/css" href=".css" />

</head>

<body>

</body>

</html>

④代码：

$(document).ready(function(){

$("#html\_normal").click(function()

{

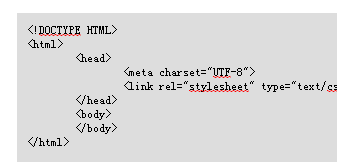
obj = $.ajax({url:"txt/tag\_query.json",async:false});

**$("#area\_1").text(obj.responseText);**

})

})

⑤效果：



**异步（新）：**

**①HTML标签：**

<div class="button">

<button id="html\_normal" class="clickit">html</button>标准html标签模板

</div>

<textarea class="output" id="area\_1">输出的内容区将显示在这里，黑色是你点击按钮之后，输出你需要的内容的区域</textarea>

<textarea class="output" id="area\_2"></textarea>

<textarea class="output" id="area\_3"></textarea>

<textarea class="output" id="area\_4"></textarea>

**②事件：**点击class="clickit"的按钮后触发；

**③调用文件**和原来一致

**④JavaScript代码：**

var num = 1;

$(document).ready(function(){

$(".clickit").click(function(){

ID=this.id;

$.ajax({

url: "txt/" + ID + ".json",

success: function(data){

$("#area\_"+num).text(data);

$(".output").css("background-color","#ddd");

$(".output").css("color","black");

$("#area\_"+num).css("background-color","black");

$("#area\_"+num).css("color","white");

},

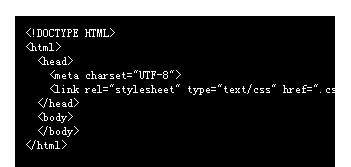
dataType: 'text'

});

});

});

效果（因为css属性被修改）：



**（33）遍历**

假设我们用class选择器$(".node")选择了所有class="node"的对象，然后我们需要对其进行逐个操作（而不是一次性批量操作），那么我们需要使用each方法

例子：

$(".node").each(function(index, item){

函数内容

})

其中，参数第一个是当前索引（比如第几个），item是当前元素（所以可以用this）

效果是对$(".node")这个对象的每一个元素，执行each的回调函数里的内容。

**(34)选择器**

①普通选择方法，略

②选择其中的c，要求是b里的c

<div class="a">  
 <div class="b">  
 <div class="c">  
 </div>  
 </div>  
 <div class="c">  
 </div>  
</div>

方法：

$(".a .b .c")

关键之处是.a、.b、.c之间要有空格。

③下面转一个帖子：

$("#myELement") 选择id值等于myElement的元素，id值不能重复在文档中只能有一个id值是myElement所以得到的是唯一的元素

$("div") 选择所有的div标签元素，返回div元素数组

$(".myClass") 选择使用myClass类的css的所有元素

$("\*") 选择文档中的所有的元素，可以运用多种的选择方式进行联合选择：例如$("#myELement,div,.myclass")

层叠选择器：

$("form input") 选择所有的form元素中的input元素

$("#main > \*") 选择id值为main的所有的子元素

$("label + input") 选择所有的label元素的下一个input元素节点，经测试选择器返回的是label标签后面直接跟一个input标签的所有input标签元素

$("#prev ~ div") 同胞选择器，该选择器返回的为id为prev的标签元素的所有的属于同一个父元素的div标签

基本过滤选择器：

$("tr:first") 选择所有tr元素的第一个

$("tr:last") 选择所有tr元素的最后一个

$("input:not(:checked) + span")

过滤掉：checked的选择器的所有的input元素

$("tr:even") 选择所有的tr元素的第0，2，4... ...个元素（注意：因为所选择的多个元素时为数组，所以序号是从0开始）

$("tr:odd") 选择所有的tr元素的第1，3，5... ...个元素

$("td:eq(2)") 选择所有的td元素中序号为2的那个td元素

$("td:gt(4)") 选择td元素中序号大于4的所有td元素

$("td:ll(4)") 选择td元素中序号小于4的所有的td元素

$(":header")

$("div:animated")

内容过滤选择器：

$("div:contains('John')") 选择所有div中含有John文本的元素

$("td:empty") 选择所有的为空（也不包括文本节点）的td元素的数组

$("div:has(p)") 选择所有含有p标签的div元素

$("td:parent") 选择所有的以td为父节点的元素数组

可视化过滤选择器：

$("div:hidden") 选择所有的被hidden的div元素

$("div:visible") 选择所有的可视化的div元素

属性过滤选择器：

$("div[id]") 选择所有含有id属性的div元素

$("input[name='newsletter']") 选择所有的name属性等于'newsletter'的input元素

$("input[name!='newsletter']") 选择所有的name属性不等于'newsletter'的input元素

$("input[name^='news']") 选择所有的name属性以'news'开头的input元素

$("input[name$='news']") 选择所有的name属性以'news'结尾的input元素

$("input[name\*='man']") 选择所有的name属性包含'news'的input元素

$("input[id][name$='man']") 可以使用多个属性进行联合选择，该选择器是得到所有的含有id属性并且那么属性以man结尾的元素

子元素过滤选择器：

$("ul li:nth-child(2)"),$("ul li:nth-child(odd)"),$("ul li:nth-child(3n + 1)")

$("div span:first-child") 返回所有的div元素的第一个子节点的数组

$("div span:last-child") 返回所有的div元素的最后一个节点的数组

$("div button:only-child") 返回所有的div中只有唯一一个子节点的所有子节点的数组

表单元素选择器：

$(":input") 选择所有的表单输入元素，包括input, textarea, select 和 button

$(":text") 选择所有的text input元素

$(":password") 选择所有的password input元素

$(":radio") 选择所有的radio input元素

$(":checkbox") 选择所有的checkbox input元素

$(":submit") 选择所有的submit input元素

$(":image") 选择所有的image input元素

$(":reset") 选择所有的reset input元素

$(":button") 选择所有的button input元素

$(":file") 选择所有的file input元素

$(":hidden") 选择所有类型为hidden的input元素或表单的隐藏域

表单元素过滤选择器：

$(":enabled") 选择所有的可操作的表单元素

$(":disabled") 选择所有的不可操作的表单元素

$(":checked") 选择所有的被checked的表单元素

$("select option:selected") 选择所有的select 的子元素中被selected的元素

选取一个 name 为”S\_03\_22″的input text框的上一个td的text值

$(”input[@ name =S\_03\_22]“).parent().prev().text()

名字以”S\_”开始，并且不是以”\_R”结尾的

$(”input[@ name ^='S\_']“).not(”[@ name $='\_R']“)

一个名为 radio\_01的radio所选的值

$(”input[@ name =radio\_01][@checked]“).val();

$("A B") 查找A元素下面的所有子节点，包括非直接子节点

$("A>B") 查找A元素下面的直接子节点

$("A+B") 查找A元素后面的兄弟节点，包括非直接子节点

$("A~B") 查找A元素后面的兄弟节点，不包括非直接子节点

1. $("A B") 查找A元素下面的所有子节点，包括非直接子节点

例子：找到表单中所有的 input 元素

HTML 代码:

<form>

<label>Name:</label>

<input name="name" />

<fieldset>

<label>Newsletter:</label>

<input name="newsletter" />

</fieldset>

</form>

<input name="none" />

jQuery 代码:

$("form input")

结果:

[ <input name="name" />, <input name="newsletter" /> ]

2. $("A>B") 查找A元素下面的直接子节点

例子：匹配表单中所有的子级input元素。

HTML 代码:

<form>

<label>Name:</label>

<input name="name" />

<fieldset>

<label>Newsletter:</label>

<input name="newsletter" />

</fieldset>

</form>

<input name="none" />

jQuery 代码:

$("form > input")

结果:

[ <input name="name" /> ]

3. $("A+B") 查找A元素后面的兄弟节点，包括非直接子节点

例子：匹配所有跟在 label 后面的 input 元素

HTML 代码:

<form>

<label>Name:</label>

<input name="name" />

<fieldset>

<label>Newsletter:</label>

<input name="newsletter" />

</fieldset>

</form>

<input name="none" />

jQuery 代码:

$("label + input")

结果:

[ <input name="name" />, <input name="newsletter" /> ]

4. $("A~B") 查找A元素后面的兄弟节点，不包括非直接子节点

例子：找到所有与表单同辈的 input 元素

HTML 代码:

<form>

<label>Name:</label>

<input name="name" />

<fieldset>

<label>Newsletter:</label>

<input name="newsletter" />

</fieldset>

</form>

<input name="none" />

jQuery 代码:

$("form ~ input")

结果:

[ <input name="none" /> ]

**（35）取消绑定的事件**

①假如我设置一个事件：

$("#p").click(function(evt){

//do something

}

需求：

当某个条件符合时，点击事件被取消；否则，点击事件依然生效。

因此，我不能使用one来取代click（就算可以，也不优雅）

②取消方法：

$("#p").click(function(evt){

//do something

**$(this).unbind(evt);**

}

即可取消本事件，this指$("#p")，evt指该事件。

**（35）鼠标事件时检测鼠标是否处于按下状态**

**$**("#Box\_Background").mousemove(function (evt) {  
 if (!evt.**buttons**) { // 假如鼠标不是按下状态，那么取消移动，然后返回；因为在移动中鼠标取消按下，无法触发mouseup事件  
 move = false;  
 return;  
 }

})

查看evt.buttons，如果为0，则未按下，否则则处于按下状态

**（36）切分条代码**

//切分条移动  
SplitDrag: function (topNode, BottomNode, splitLine, ParentNode) {  
 //注：四个参数都是已经通过选择器获取后的dom元素  
 var self = this;  
 var mouseStart;  
 var splitStart;  
 var gridStart;  
 var logAreaStart;  
 var canMove = false;  
  
 //允许移动和禁止移动  
 splitLine.mousedown(function (evt) {  
 mouseStart = evt.**clientY**;  
 canMove = true;  
  
 gridStart = parseInt(topNode.**css**("height"));  
 splitStart = parseInt(splitLine.**css**("top"));  
 logAreaStart = parseInt(BottomNode.**css**("height"));  
 })  
  
 splitLine.mouseup(function (evt) {  
 canMove = false;  
 })  
  
 //移动中  
 ParentNode.mousemove(function (evt) {  
 if (canMove && evt.**buttons**) {  
 //切分条的位置  
 var offset = evt.**clientY** - mouseStart;  
 topNode.**css**("height", gridStart + offset + "px");  
 BottomNode.**css**("height", logAreaStart - offset + "px");  
 }  
 })  
}

**（37）FormData**

简单来说，就是一个form表单。

因此，可以使用HTML表单来初始化一个FormData对象，或者是用FormData对象发送文件等。

参考url：

<https://developer.mozilla.org/zh-CN/docs/Web/Guide/Using_FormData_Objects>

**（38）通过jquery上传文件**

<http://www.jianshu.com/p/46e6e03a0d53>

通过jQuery Ajax使用FormData对象上传文件