目录

[一、 jQuery简介 3](#_Toc401221196)

[1、 什么是jQuery 3](#_Toc401221197)

[2、 部署jQuery 3](#_Toc401221198)

[二、 选择器 4](#_Toc401221199)

[1、 基本 4](#_Toc401221200)

[2、 层级 4](#_Toc401221201)

[3、 简单 5](#_Toc401221202)

[4、 内容 5](#_Toc401221203)

[5、 可见性 6](#_Toc401221204)

[6、 属性 6](#_Toc401221205)

[7、 子元素 6](#_Toc401221206)

[8、 表单 7](#_Toc401221207)

[9、 表单对象属性 7](#_Toc401221208)

[10、 选择器例题 8](#_Toc401221209)

[三、 jQuery对象与dom对象 9](#_Toc401221210)

[1、 什么是dom对象 9](#_Toc401221211)

[2、 什么是jQuery对象 9](#_Toc401221212)

[3、 dom对象与jQuery对象的相互转换 10](#_Toc401221213)

[4、 例题 11](#_Toc401221214)

[四、 属性操作 11](#_Toc401221215)

[1、 基本属性 11](#_Toc401221216)

[2、 css 12](#_Toc401221217)

[3、 html/文本/值 13](#_Toc401221218)

[五、 css操作 14](#_Toc401221219)

[1、 基本css 14](#_Toc401221220)

[2、 位置 14](#_Toc401221221)

[3、 尺寸 14](#_Toc401221222)

[六、 事件编程 15](#_Toc401221223)

[1、 页面载入 15](#_Toc401221224)

[2、 基本事件 16](#_Toc401221225)

[3、 事件切换 17](#_Toc401221226)

[4、 事件处理 18](#_Toc401221227)

[5、 事件冒泡 19](#_Toc401221228)

[6、 默认行为 19](#_Toc401221229)

[七、 效果 20](#_Toc401221230)

[1、 基本 20](#_Toc401221231)

[2、 滑动 21](#_Toc401221232)

[3、 淡入淡出 22](#_Toc401221233)

[4、 一组图片的淡入淡出 22](#_Toc401221234)

[5、 菜单功能 23](#_Toc401221235)

[八、 文档处理 24](#_Toc401221236)

[1、 插入 24](#_Toc401221237)

[2、 删除 25](#_Toc401221238)

[3、 复制 25](#_Toc401221239)

[4、 替换 26](#_Toc401221240)

[5、 包裹 27](#_Toc401221241)

[6、 查找 27](#_Toc401221242)

[7、 案例 28](#_Toc401221243)

[九、 插件机制 29](#_Toc401221244)

[1、 插件机制语法： 29](#_Toc401221245)

[2、 插件机制案例 30](#_Toc401221246)

[十、 each语句 31](#_Toc401221247)

[1、 语法 31](#_Toc401221248)

[2、 例题 31](#_Toc401221249)

[十一、 jQuery中的Ajax应用 32](#_Toc401221250)

[1、 Ajax原生代码 32](#_Toc401221251)

[2、 jQuery中的Ajax方法 32](#_Toc401221252)

[3、 解析xml数据 35](#_Toc401221253)

[4、 解析json数据 36](#_Toc401221254)

1. jQuery简介
2. 什么是jQuery

一款免费且开放源代码的JavaScript代码库

事件编程

dom操作

Ajax

………

jQuery

dojo

prototype

ExtJS

1. 部署jQuery
2. 下载压缩包，并且解压



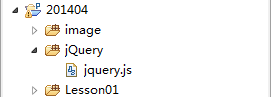
1. 进入文件夹，观察其中的文件



jQuery.min.js ：去掉了不必要的空格和换行，便于下载，项目上线后可以使用这个文件

jQuery.js ：完整的jQuery源代码，便于学习

1. 将其中的一个文件复制到项目，在网页中直接引用即可。



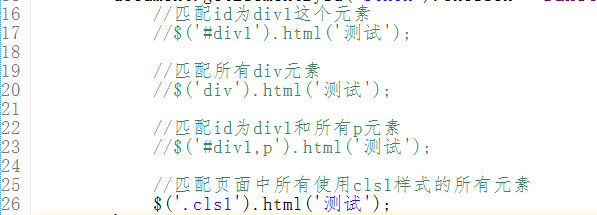
1. 选择器

在jQuery中，获取页面中元素的方式，我们称为选择器

* 基本
* 层级
* 简单
* 内容
* 可见性
* 属性
* 子元素
* 表单
* 表单对象属性

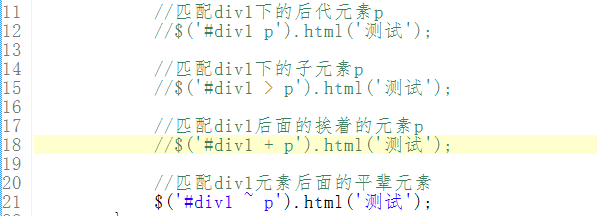
1. 基本

* #id ：通过id属性值获取元素
* Element ：通过元素名获取元素
* selector1,selector2 ：获取列表中指定的所有元素
* .class ：获取使用class样式的所有元素



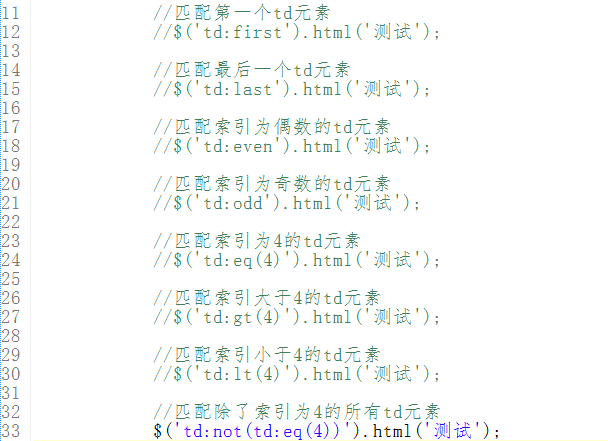
1. 层级

* ancetor descendant ：匹配祖先元素下的后代元素
* parent > child ：匹配父元素下的子元素
* prev + next ：匹配上一个元素的后面的元素，必须是挨着的
* prev~siblings ：匹配前一个元素的所有平辈元素，在prev元素后面才能匹配



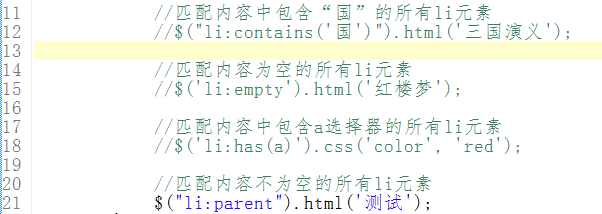
1. 简单

* :first ：匹配第一个元素
* :last ：匹配最后一个元素
* :even ：匹配索引为偶数的元素
* :odd ：匹配索引为奇数的元素
* :eq(index) ：匹配索引为指定值的元素
* :gt(index) ：匹配索引大于指定值的元素
* :lt(index) ：匹配索引小于指定值的元素
* :not(selector) ：匹配除了指定选择器之外的所有元素



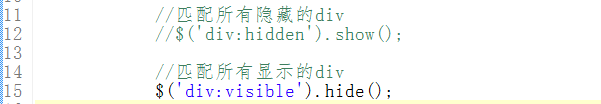
1. 内容

* :contains(text) ：匹配内容中包含指定内容的元素
* :empty ：匹配内容为空的元素
* :has(selector) ：匹配内容中包含指定选择器的元素
* :parent ：匹配内容不为空的元素



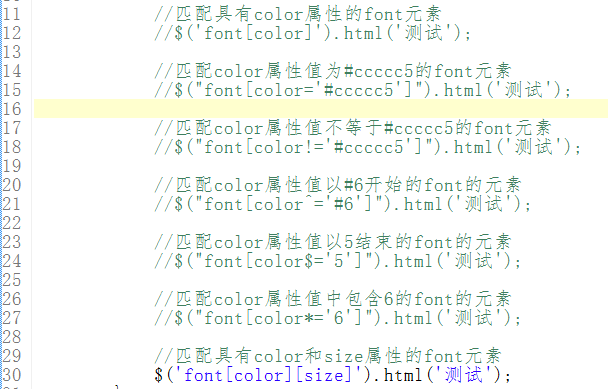
1. 可见性

* :hidden ：匹配隐藏的元素
* :visible ：匹配显示的元素



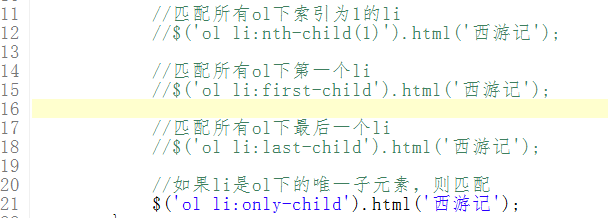
1. 属性

* [attribute] ：匹配具有指定属性的元素
* [attribute=value] ：匹配属性等于指定值的元素
* [attribute!=value] ：匹配属性不等于指定值的元素
* [attribute^=value] ：匹配属性值以指定值开始的元素
* [attribute$=value] ：匹配属性值以指定值结束的元素
* [attribute\*=value] ：匹配属性值中包含指定值的元素
* [selector1][selector2][selectorN] ：同时满足所有条件，则匹配



1. 子元素

* :nth-child(index/even/odd) ：匹配指定索引的元素 从1算起
* :first-child ：匹配第一个子元素
* :last-child ：匹配最后一个子元素
* :only-child ：如果子元素是父元素的唯一子元素，则匹配



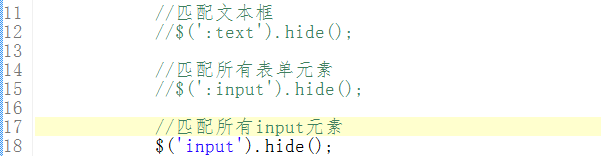
1. 表单

* :input ：匹配所有表单元素

$(“:input”) ：匹配所有表单元素 select、textarea

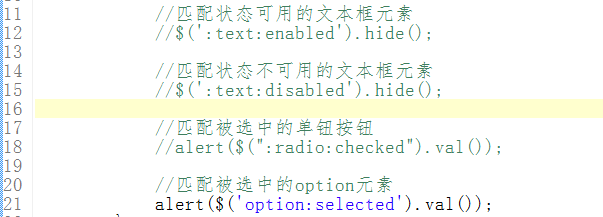
$(‘input’) ：匹配input标签

* :text ：文本框
* :password ：密码框
* :radio ：单选按钮
* :checkbox ：复选框
* :submit ：提交按钮
* :reset ：重置按钮
* :image ：图像按钮
* :button ：按钮
* :file ：文件框
* :hidden ：隐藏域

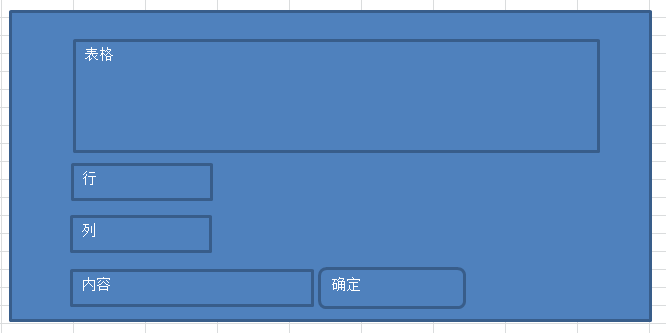


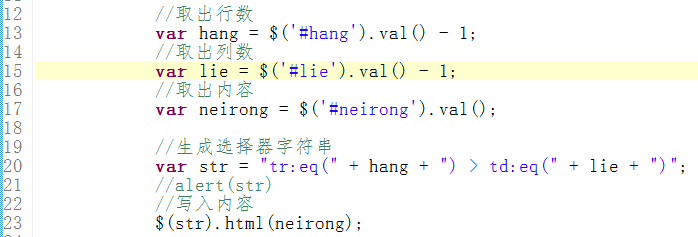
1. 表单对象属性

* :enabled ：匹配状态可用的表单元素
* :disabled ：匹配状态不可用的表单元素
* :checked ：匹配被选中的元素 匹配单选或复选
* :selected ：匹配被选中的元素 匹配的是option



1. 选择器例题





总结：

上午所用的选择器都是以$()形式获取的，

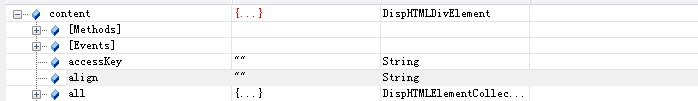
使用$()形式获取的都是jQuery对象

jQuery对象只能调用jQuery方法，不能调用其它的js方法。

1. jQuery对象与dom对象
2. 什么是dom对象

document.getElementById(‘content’);



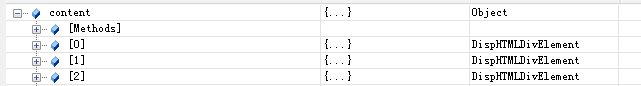


以上方式获取回来的是dom对象，dom对象可以调用dom属性和相关方法

dom对象不能调用jQuery方法

1. 什么是jQuery对象





以上方式获取回来的是jQuery对象，

其实，jQuery对象本身就是一个数组，选择器匹配回来几个数组元素，jQuery对象（数组）长度就是元素个数

每个jQuery对象下的某个元素都是一个dom对象

dom对象只能调用dom方法

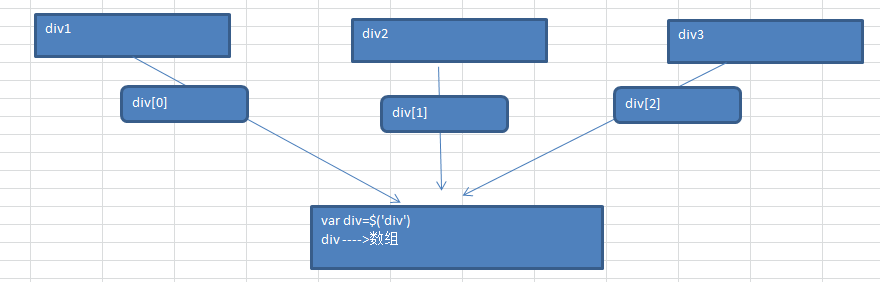
jQuery对象只能调用jQuery方法

它们之间能不能互相调用对方法

dom.jquery();

jquery.dom();

1. dom对象与jQuery对象的相互转换



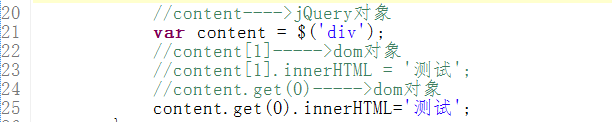
1）jQuery对象转dom对象

语法：

jQuery[0]

或

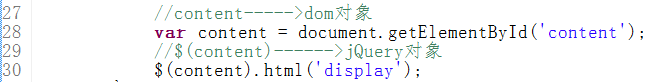
jQuery.get(0)



2）dom对象转jQuery对象

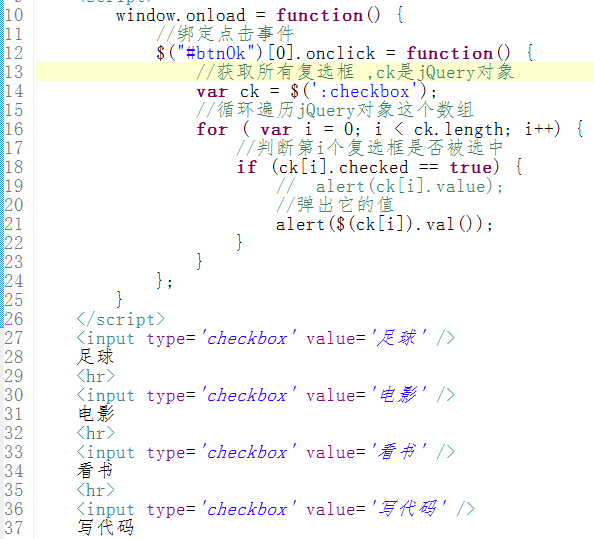
语法：

$(dom)



1. 例题

通过jquery获取所有复选框对象，再通过checked验证每一个是否被选中，如果选中，则弹出对应的值



建议：我们在写程序时，对于jQuery对象的变量尽量加$

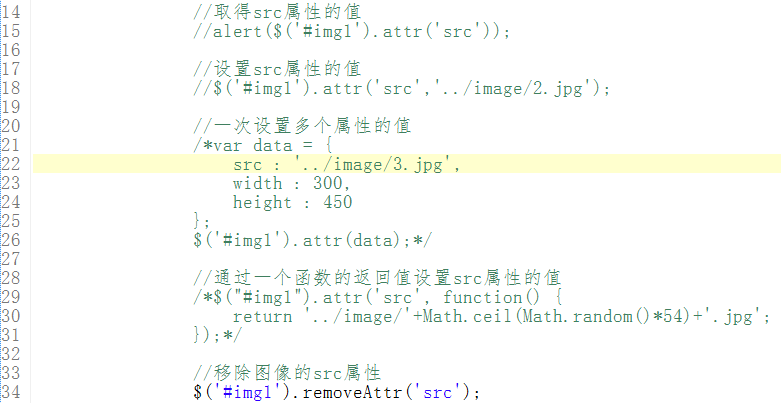


1. 属性操作

* 属性
* css
* html/文本/值

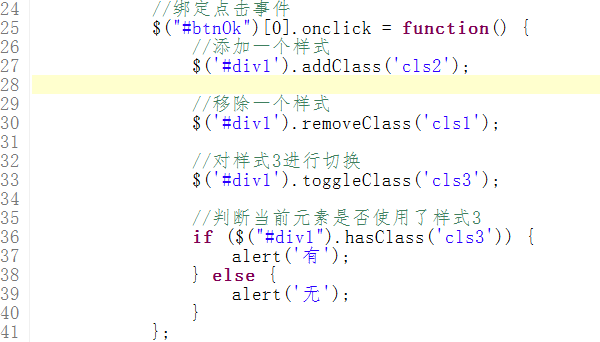
1. 基本属性

* attr(name) ：获取指定属性的值
* attr(key,value) ：设置指定属性的值
* attr(properties) ：一次设置多个属性的值，要求参数必须是json对象
* attr(key,fn) ：通过一个函数的返回值设置指定属性的值
* removeAttr(name) ：删除指定的属性



1. css

* addClass(class) ：添加样式
* removeClass(class) ：移除样式
* toggleClass(class) ：切换样式，如果有，就删除，如果没有，则添加
* hasClass(class) ：判断是否使用了某个样式 返回true：有 false：无



1. html/文本/值

innerHTML

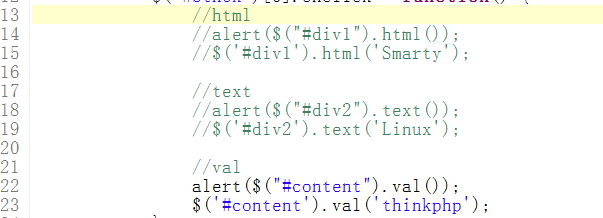
* html()
* html(val)

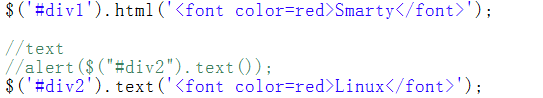
innerText

* text()
* text(val)

value

* val()
* val(val)







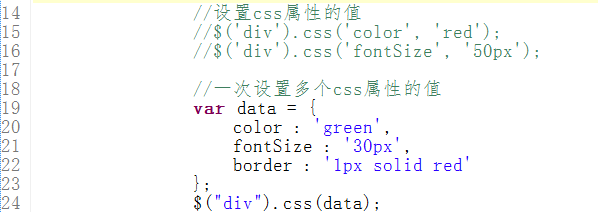
可以看出，innerHTML属性可以解析html标签,innerText是原样输出（忽略标签）

1. css操作

* css
* 位置
* 尺寸

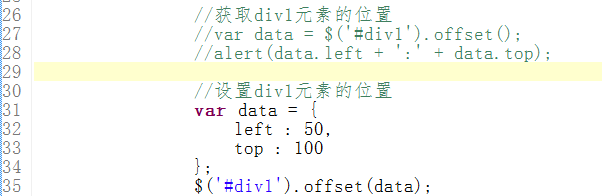
1. 基本css

* css(name) ：获取指定样式的值
* css(name,value) ：设置指定样式的值
* css(properties) ：一次设置多个样式的值，要求参数是json对象



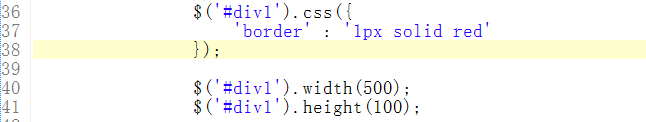
1. 位置

* offset() ：获取元素的位置，返回json对象，包含left和top属性
* offset(coordinates) ：设置元素的位置，要求参数json对象，必须包含left和top



1. 尺寸

* width() ：获取宽度
* width(value) ：设置宽度
* height() ：获取高度
* height(value) ：设置高度



1. 事件编程

* 页面载入
* 基本事件
* 事件切换
* 事件处理

1. 页面载入

以前，关于页面的初始化语句，我们定义在window.onload语句段内：

window.onload=function(){

//页面初始化

}

现在，在jQuery中，封装了一个方法用于初始化页面

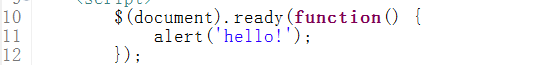
* ready(fn)

现在，可以改成：

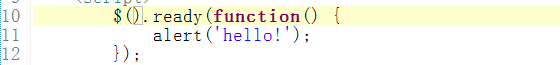
$(document).ready(function(){

//页面初始化

});



或者



window.onload与$().ready()有什么区别？

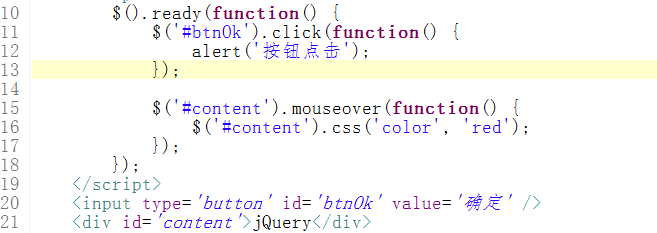
window.onload是页面载入后开始执行，必须等于所有网络资源加载完毕后才会触发

$().ready是页面载入后开始执行，但是只要将元素标签本身加载完毕后就可以直接运行

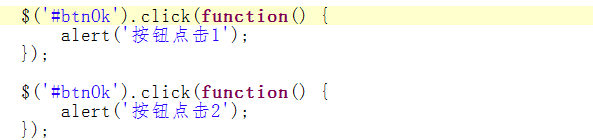
$().ready()相比较之下，执行要比window.onload要快一些。

1. 基本事件

* blur(fn) ：失去焦点
* change(fn) ：内容改变
* click(fn) ：点击
* dblclick(fn) ：双击
* focus(fn) ：获得焦点
* keydown(fn) ：键盘按下
* keyup(fn) ：键盘抬起
* keypress(fn) ：键盘点击
* load(fn) ：页面载入
* unload(fn) ：页面关闭
* mousedown(fn) ：鼠标按下
* mouseup(fn) ：鼠标抬起
* mousemove(fn) ：鼠标移动
* mouseover(fn) ：鼠标经过
* mouseout(fn) ：鼠标离开
* resize(fn) ：改变尺寸
* scroll(fn) ：滚动条滚动
* select(fn) ：选择内容
* submit(fn) ：提交按钮

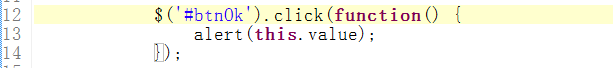


运行以下代码：



通过测试，可以看出，jQuery在绑定事件时使用了事件监听机制，以上两个方法都可以执行，同时调整事件监听的兼容性问题，并调整了事件处理程序的触发顺序，统一调整为顺序触发

运行以下代码：



可以看出，弹出的都是按钮的内容，说明jQuery对处理程序中的this的指向兼容性进行了调整，统一指向了当前dom对象（apply）。

1. 事件切换

* hover(over,out)

这个方法是针对鼠标经过和离开的事件绑定

over：鼠标经过时执行的事件处理程序

out：鼠标离开时执行的事件处理程序

* toggle(fn,fn)

当鼠标点击时的事件绑定

第一次点击时触发第一个事件处理程序

第二次点击时触发第二个事件处理程序

………

第一次



1. 事件处理

* bind(type,[data],fn)

绑定事件

type：事件名 事件名没有on前缀

[data] ：事件发生时传递的参数

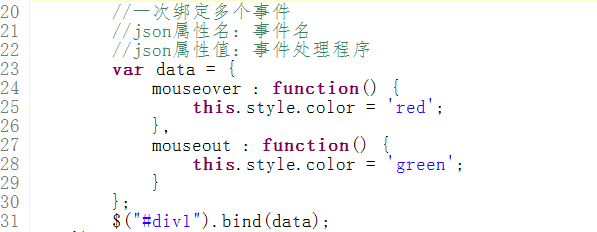
fn：事件处理程序



* bind({type:fn,type:fn})

一次绑定多个事件 要求参数必须是json对象

type：事件名 fn：事件处理程序



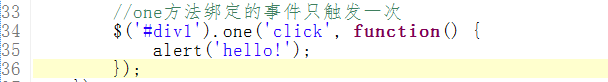
* one(type,[data],fn)

绑定事件，只触发一次

type：事件名 事件名没有on前缀

[data] ：事件发生时传递的参数

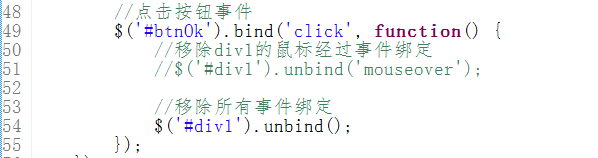
fn：事件处理程序



* unbind([type])

移除事件绑定

[type] ：要移除的事件名，如果不写此参数，表示移除该对象所有事件绑定



1. 事件冒泡

什么是事件冒泡

之所以称之为冒泡，是因为事件会按照dom层次结构向水泡一样上升至最顶端

<body>

<div>

<img />

</div>

</body>

取消事件冒泡

function display(event){

event.stopPropagation();

}



在js原生代码中，取消事件冒泡有两种方式：window.event.cancelBubble=true和event.stopPropagation()，可以看出,jQuery对以上两种形式调整了兼容性

1. 默认行为

什么是默认行为

有些html元素会有自己的行为，如：点击按钮时提交表单

<form>

<input type=“text”>

<input type=“submit”>

</form>

如果表单内有些数据不完整，那么需要取消提交按钮的默认行为

<a href=’1.php’>删除</a>

如果是用户的误操作，也需要取消超链接默认行为

w3c： event.preventDefault();

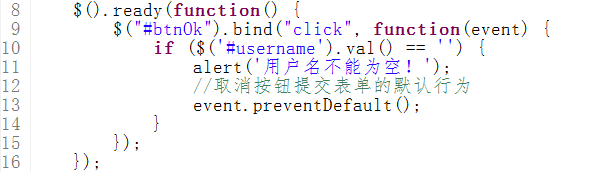
ie： window.event.returnValue=false;

阻止默认行为

function display(event){

event.preventDefault();

}



1. 效果

* 基本
* 滑动
* 淡入淡出

1. 基本

* show() 显示
* show(speed,[callback]) 以动画效果显示

speed：动画持续时间（毫秒）

[callback] ：动画执行完毕后所调用的函数

* hide() 隐藏
* hide(speed,[callback]) 以动画效果隐藏

speed：动画持续时间（毫秒）

[callback] ：动画执行完毕后所调用的函数

* toggle() 显示或隐藏（切换）
* toggle(switch) 显示或隐藏，要求参数是true或false
* toggle(speed,[callback]) 以动画效果显示或隐藏

speed：动画持续时间（毫秒）

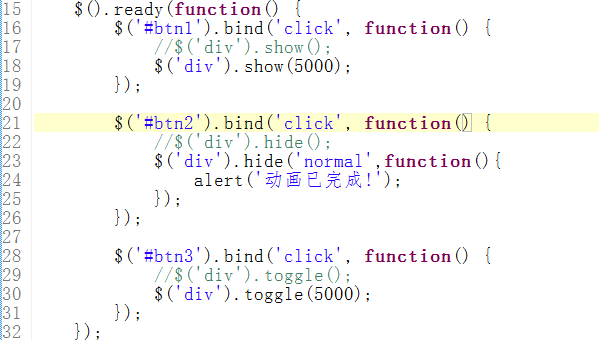
[callback] ：动画执行完毕后所调用的函数

同时以上函数speed参数也支持下面三个值：

"slow"：缓慢

"normal"：正常

"fast"：快速



1. 滑动

* slideDown(speed,[callback])

向下滑动 （显示）

* slideUp(speed,[callback])

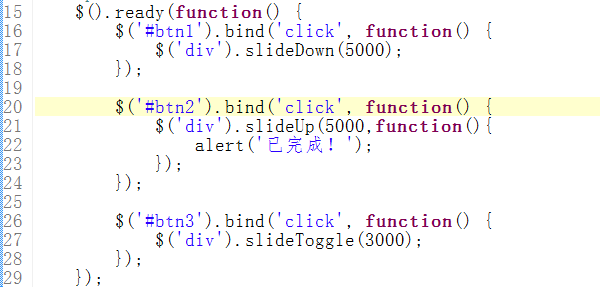
向上滑动 （隐藏）

* slideToggle(speed,[callback])

滑动切换

speed：动画持续时间（毫秒）

[callback] ：动画执行完毕后所调用的函数



1. 淡入淡出

* fadeIn(speed,[callback])

淡入（显示）

* fadeOut(speed,[callback])

淡出（隐藏）

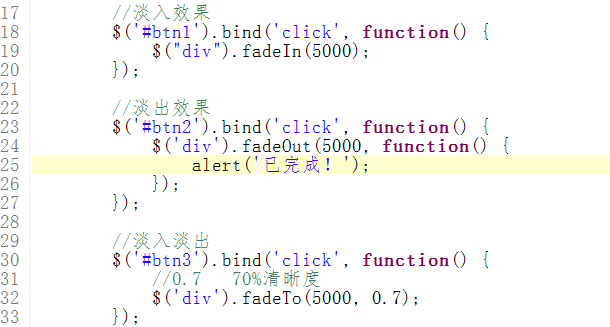
* fadeTo(speed,opacity,[callback])

淡入淡出到指定值

speed：动画持续时间

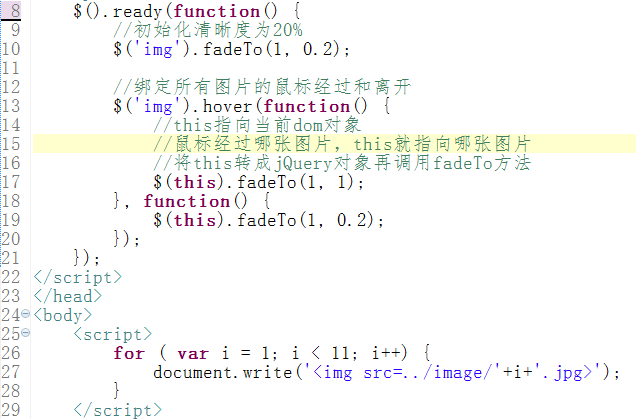
opacity：清晰度 0-----1 0：消息 1：显示 0.2 20%

[callback]：动画完毕后执行的函数



1. 一组图片的淡入淡出

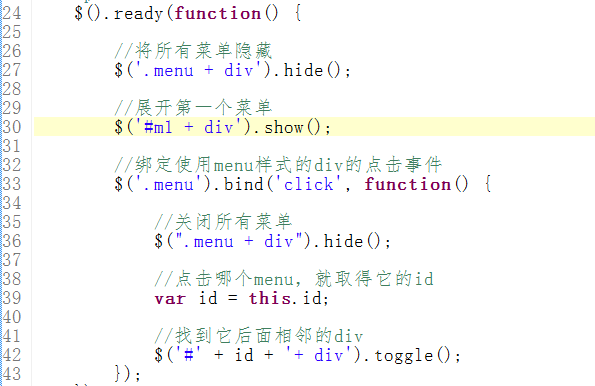
在浏览器放入一组图片，默认清晰度20%，当鼠标经过时清晰度100%，当离开时20%



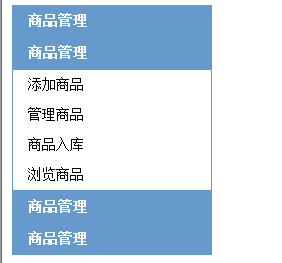
运行结果：



1. 菜单功能



运行效果：



1. 文档处理

jQuery中的文档处理就是dom操作

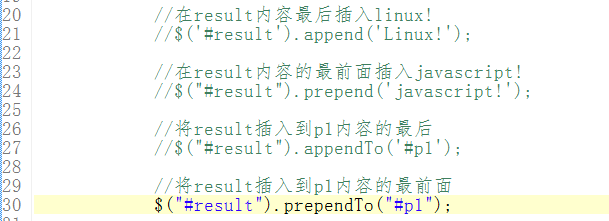
* 插入
* 删除
* 复制
* 替换
* 包裹
* 查找

1. 插入

内部插入：

* append(content) ：在元素内部的最后插入数据
* appendTo(content) ：将匹配到的元素插入到指定内容的后面
* prepend(content) ：在元素内部的前面插入数据
* prependTo(content) ：将匹配到的元素插入到指定内容的前面

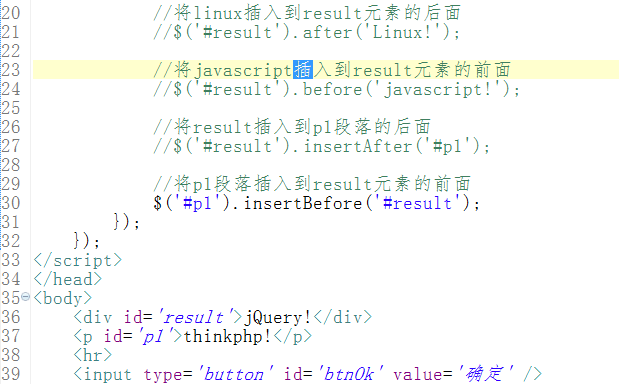
<div>abcjQuerydef</div>



外部插入：

abc<div>jQuery</div>def

* after(content) ：在元素的后面插入数据
* before(content) ：在元素的前面插入数据
* insertAfter(content) ：将匹配元素插入到指定内容的后面
* insertBefore(content) ：将匹配元素插入到指定内容的前面

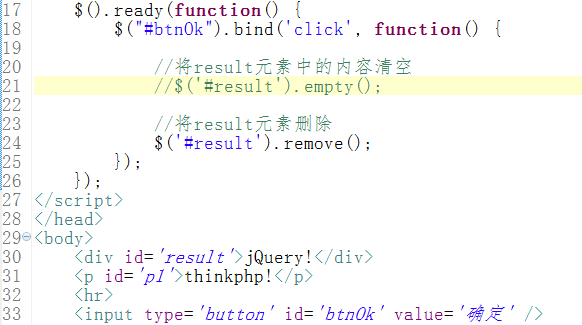


1. 删除

* empty() ：清空元素内容
* remove() ：删除元素

empty是将元素内容清空，但元素本身保留

remove是将元素本身删除，后面不允许再对元素进行操作



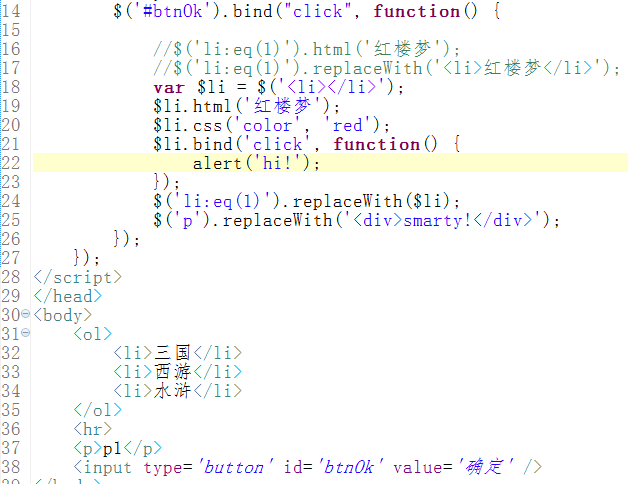
1. 复制

* clone() ：复制匹配的元素
* clone(true) ：复制匹配的元素，同时也复制它的事件机制



1. 替换

* replaceWith() ：将匹配到的元素替换成指定的内容



var div=document.createElement(‘div’);

$(div);

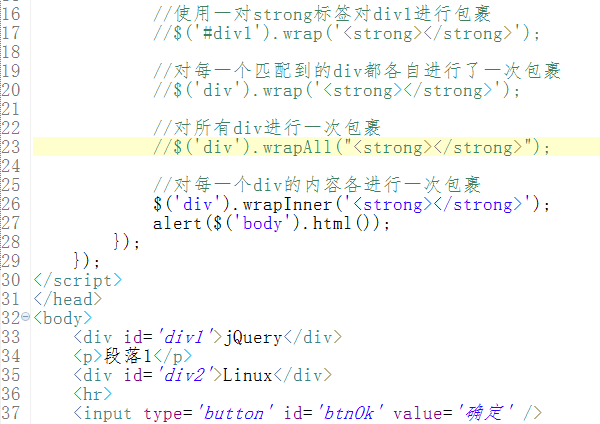
与

$(“<div></div”)效果一样的。

1. 包裹

<strong><div><b>jQuery</b></div></strong>

* wrap() ：对匹配的元素使用指定的内容进行包裹
* wrapAll() ：对匹配的元素使用指定的内容进行包裹（仅包裹一次）
* wrapInner() ：对匹配元素的内容进行包裹



1. 查找

* eq(index) ：匹配指定索引的元素

$(‘div’).eq(3) ：匹配索引为3的div

* filter(expr) ：匹配过滤的元素

$(‘div’).filter(‘.cls1’) ：匹配使用了cls1样式的所有div

* not(expr) ：匹配除了指定元素之外的所有元素

$(‘div’).not(‘.cls1’) ：匹配除了使用cls1样式的所有div

* children([expr]) ：匹配所有子元素

$(‘product’).children(‘name’).text();

* find(expr) ：查找指定的元素

$(xml).find(‘product’)

* next([expr])

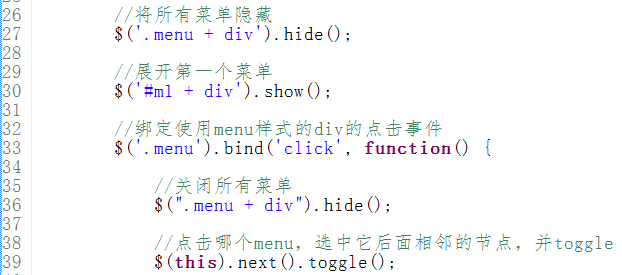
$(“#div1”).next() ：匹配div1相邻的下一个平辈节点

* prev([expr])

$(‘#div1’).prev() ：匹配div1相邻的上一个平辈节点

* parent([expr])

$(“#div1”).parent ：匹配div1元素的父节点



1. 案例





1. 插件机制
2. 插件机制语法：

如：我们在使用jQuery时，有时常用的方法jQuery并没有封装，我们还需要反复使用这样的方法，如何解决这个问题？

语法：

* jQuery.fn.extend(object)
* 语法：

jQuery.fn.extend({

fn1:function(){},

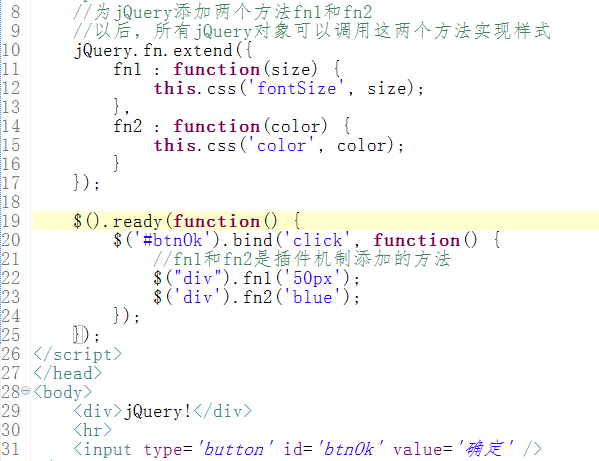
fn2:function(){},

......

});

通过jQuery.fn.extend方法对jQuery功能进行扩展，这个方法要求参数是一个json对象

对象的属性名是扩展方法名，属性值就是函数体。

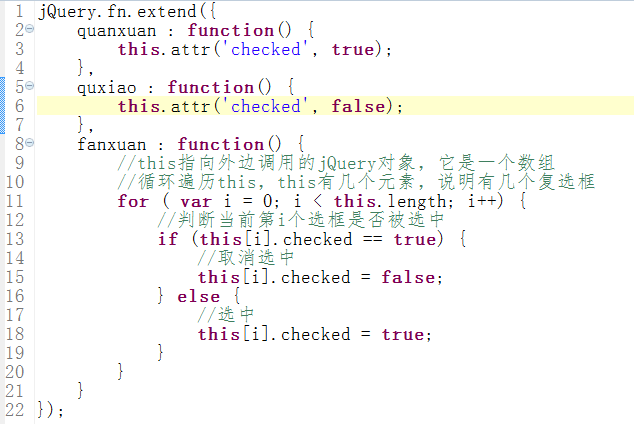


1. 插件机制案例

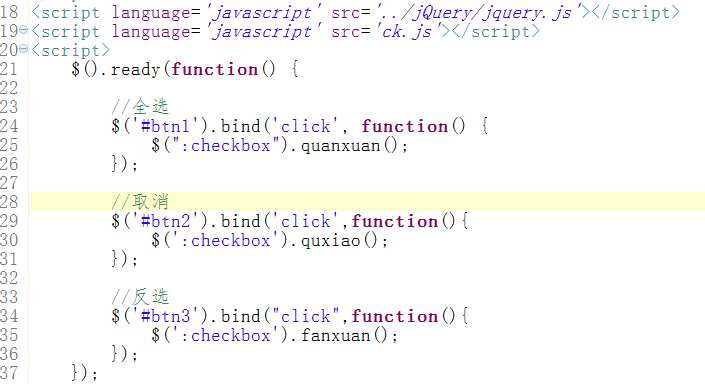


实现复选框的全选、取消、反选功能

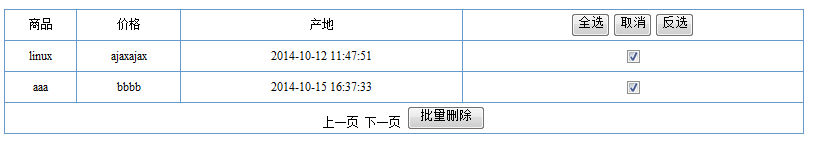
创建ck.js插件文件，用于扩展jQuery



调用：



运行效果：



1. each语句
2. 语法

each(callback) ：each方法只有一个参数，必须是一个函数

函数的形式如下：

function callback(i,item){

}

each方法的功能：

each方法用于遍历jQuery对象

jQuery对象有几个元素，那么each方法中指定的函数就会自动执行多少次

每次遍历时，将当前元素的下标放在参数i中

每次遍历时，将当前元素本身放在参数item中

1. 例题

为每个div中间添加不同的内容



为每个img添加不同的图片



1. jQuery中的Ajax应用
2. Ajax原生代码

var xhr=new XMLHttpRequest();

xhr.open(‘post’,’demo.php’);

xhr.setRequestHeader(‘content-type’,’application/x-www-form-urlencoded’);

xhr.onreadystatechange=function(){

if(xhr.readystate==4 && xhr.status==200){

//处理语句

}

};

xhr.send(‘id=10’);

1. jQuery中的Ajax方法

jQuery中实现Ajax功能的方法主要有以下三个：

* jQuery.ajax(options)
* jQuery.get(url,[data],[callback])
* jQuery.post(url,[data],[callback])

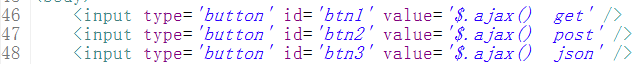
（1）jQuery中底层的ajax实现

jQuery.ajax(options);

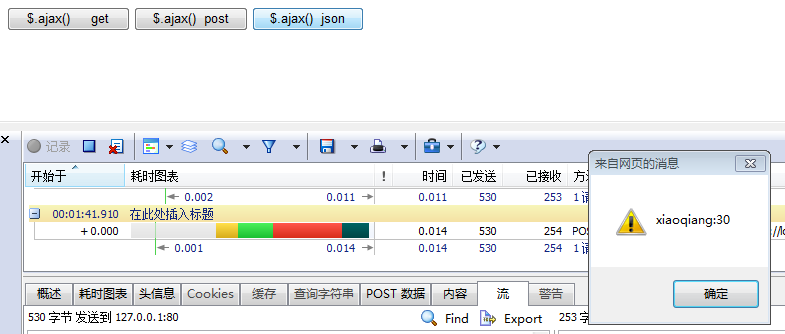
要求参数必须是一个json对象，json对象下的属性如下

* async ：是否异步 true：异步（默认） false：同步
* cache ：是否缓存
* complete ：当ajax对象的状态码为4时执行的函数
* contentType ：请求头
* data ：传递的参数，要求是一个字符串
* dataType ：期望得到的数据类型
* success ：ajax对象状态码为4并且http响应状态码为200时所执行的函数
* type ：请求类型 get post
* url ：请求地址





运行结果：



（2）jQuery中高级的ajax实现

jQuery.get(url,[data],[callback],[type])

发送get请求

url：请求地址

[data]：参数列表，要求是一个json对象，如果没有，可以不写

[callback]：处理函数

[type]：期望得到的数据类型 text、xml、json

jQuery.post(url,[data],[callback],[type])

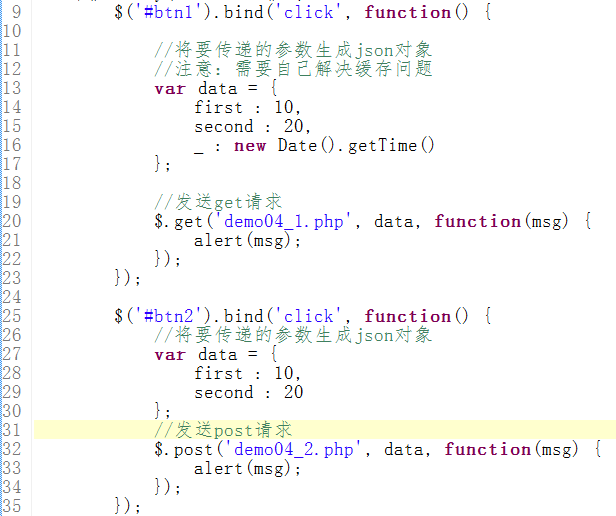
发送post请求

url：请求地址

[data]：参数列表，要求是一个json对象，如果没有，可以不写

[callback]：处理函数

[type]：期望得到的数据类型 text、xml、json





1. 解析xml数据

* var person=$(msg).find(“person”);
* person.children(“name”).text();
* person.children(“jian”).text();
* person.children(“cheng”).text();



返回结果：

<users>

<user>

<username>wangcai</username><password>222222</password>

</user>

<user>

<username>xiaoqiang</username><password>111111</password>

</user>

</users>

1. 解析json数据

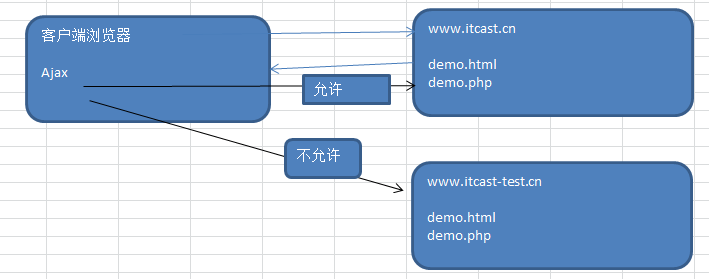
* var count=msg.length;
* for(var i=0;i<count;i++){
* msg[i].id;
* msg[i].name;
* }



返回结果：

[{"id":"3","username":"wangcai","password":"222222"},{"id":"2","username":"xiaoqiang","password":"111111"}]

1. 跨域请求
2. 什么是跨域请求



我们访问的是第一个域，ajax对象只能请求这个域上的资源，不能请求其它域上的资源

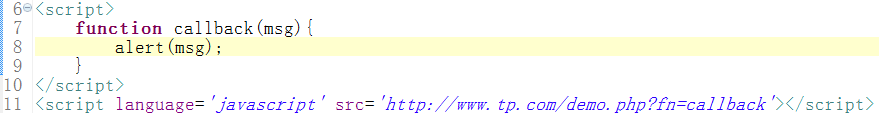
* Ajax技术由于受到浏览器的限制，该方法不允许跨域通信。
* 同源策略阻止从一个域上加载的脚本获取或操作另一个域上的文档属性。也就是说，受到请求的 URL 的域必须与当前 Web 页面的域相同。这意味着浏览器隔离来自不同源的内容，以防止它们之间的操作。



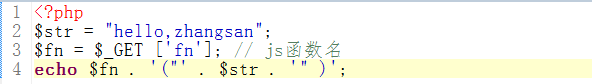
1. 解决跨域请求问题

（1）使用script标签向另一个域上的资源进行请求，传递参数fn，保存将来要执行的js函数名

html：



php：

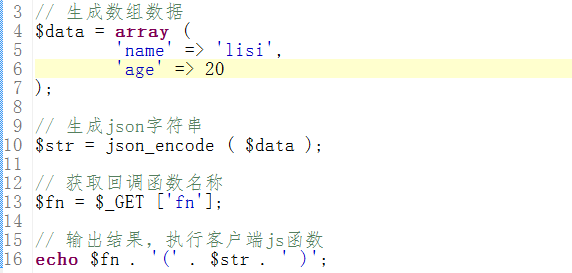


(2)JSONP是一个非官方的协议，它允许在服务器端集成script tags返回至客户端，通过javascript callback的形式实现跨域访问。

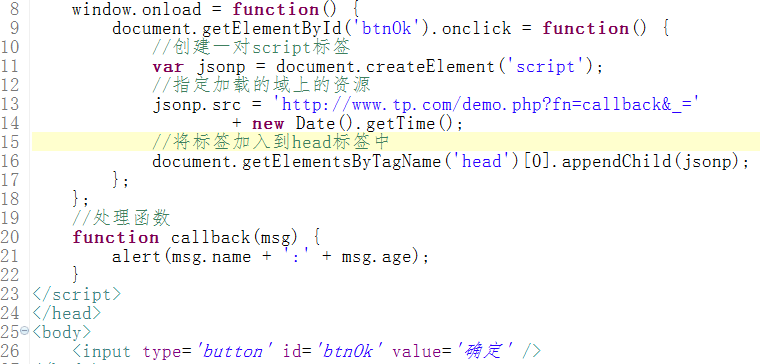


(3)为了向客户端返回更多数据，我们使用json

php：



html：

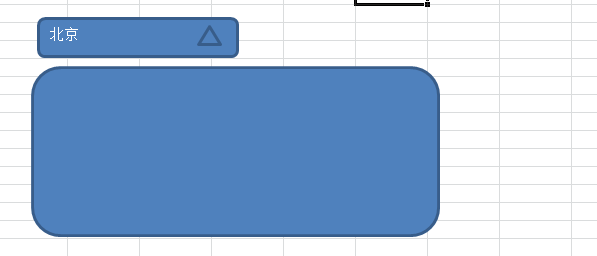


1. jQuery解决跨域问题

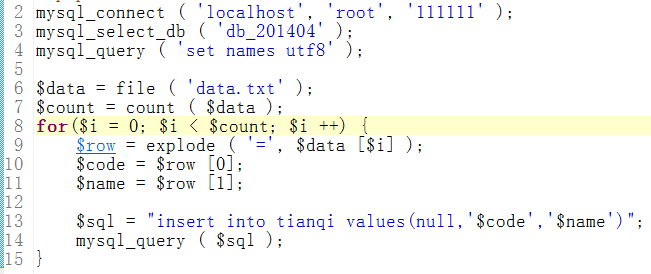
* $.ajax get
* $.get get
* $.getJSON get

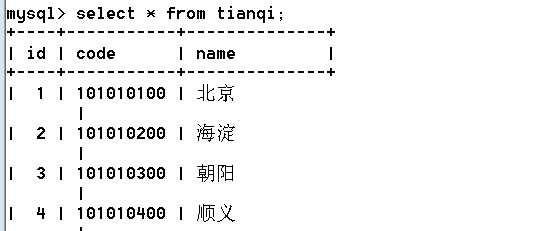


4、天气预报案例

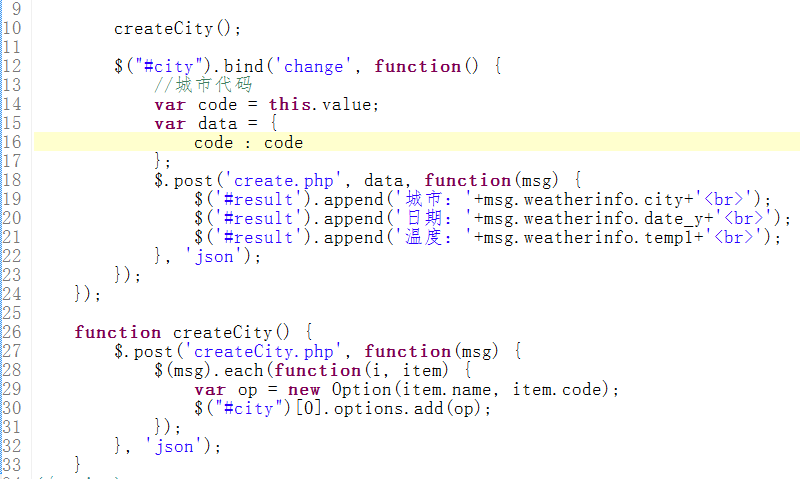


1. 编写导入程序





1. html文件：



1. php：

