# 一、AJAX简介

AJAX(Asynchronous Javascript And XML)=异步 JavaScript 和 XML。

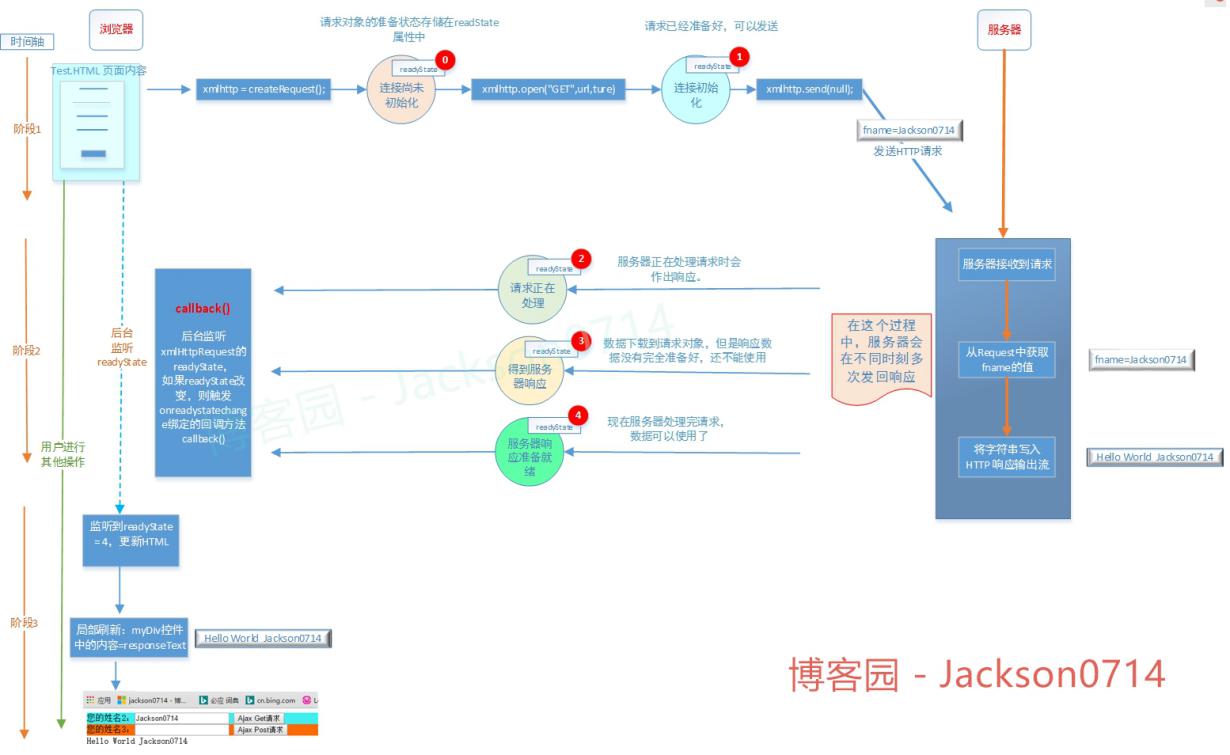
AJAX 是一种用于创建快速动态网页的技术。

通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

传统的网页（不使用 AJAX）如果需要更新内容，必须重载整个网页面。

有很多使用 AJAX 的应用程序案例：新浪微博、Google 地图、百度、淘宝等等。

# 二、AJAX的通信原理



# 三、AJAX的使用必备基础

## 1、HTML / XHTML（标签）

## 2、CSS（样式）

## 3、JavaScript / DOM（信息显示/交互）

# 四、AJAX的工作原理

## 1、创建 XMLHttpRequest 对象

XMLHttpRequest 用于在后台与服务器交换数据。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

所有现代浏览器（IE7+、Firefox、Chrome、Safari 以及 Opera）均内建 XMLHttpRequest 对象。

### 创建 XMLHttpRequest 对象的语法：

|  |
| --- |
| //创建xmlhttp对象  function createXmlHttpRequest(){  var xmlhttp=null;  if(window.XMLHttpRequest){  // code for IE7+, Firefox, Chrome, Opera, Safari  xmlhttp=new XMLHttpRequest();  }else {  xmlhttp=new ActiveXObject("Microsoft.XMLHttp")// code for IE6, IE5  }  return xmlhttp;  } |

## 2、XMLHttpRequest 对象用于和服务器交换数据。

如需将请求发送到服务器，我们使用 XMLHttpRequest 对象的 open() 和 send() 方法：

xmlhttp.open("GET","/polls/index/",true);

xmlhttp.send();

|  |  |
| --- | --- |
| **方法** | **描述** |
| open(method,url,async) | 规定请求的类型、URL 以及是否异步处理请求。  method：请求的类型；GET 或 POST  url：文件在服务器上的位置  async：true（异步）或 false（同步） |
| send(string) | 将请求发送到服务器。  string：仅用于 POST 请求 |

## 3、AJAX - 服务器响应

如需获得来自服务器的响应，请使用 XMLHttpRequest 对象的 responseText 或 responseXML 属性。

|  |  |
| --- | --- |
| **属性** | **描述** |
| responseText | 获得字符串形式的响应数据。 |
| responseXML | 获得 XML 形式的响应数据 |

### responseText 属性

如果来自服务器的响应并非 XML，请使用 responseText 属性。

responseText 属性返回字符串形式的响应，因此您可以这样使用：

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

### responseXML 属性

如果来自服务器的响应是 XML，而且需要作为 XML 对象进行解析

|  |
| --- |
| <script type="text/javascript">  function loadXMLDoc()  {  var xmlhttp;  var txt,x,i;  if (window.XMLHttpRequest)  {// code for IE7+, Firefox, Chrome, Opera, Safari  xmlhttp=new XMLHttpRequest();  }  else  {// code for IE6, IE5  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");  }  xmlhttp.onreadystatechange=function()  {  if (xmlhttp.readyState==4 && xmlhttp.status==200)  {  xmlDoc=xmlhttp.responseXML;  txt="";  x=xmlDoc.getElementsByTagName("title");  for (i=0;i<x.length;i++)  {  txt=txt + x[i].childNodes[0].nodeValue + "<br />";  }  document.getElementById("myDiv").innerHTML=txt;  }  }  xmlhttp.open("GET","/example/xmle/books.xml",true);  xmlhttp.send();  }  </script> |

book.xml内容

<bookstore><book category="children"><title lang="en">Harry Potter</title><author>J K. Rowling</author><year>2005</year><price>29.99</price></book><book category="cooking"><title lang="en">Everyday Italian</title><author>Giada De Laurentiis</author><year>2005</year><price>30.00</price></book>

</bookstore>

## 4、AJAX - onreadystatechange 事件

当请求被发送到服务器时，我们需要执行一些基于响应的任务。

每当 readyState 改变时，就会触发 onreadystatechange 事件。

readyState 属性存有 XMLHttpRequest 的状态信息。

下面是 XMLHttpRequest 对象的三个重要的属性：

|  |  |
| --- | --- |
| onreadystatechange | 存储函数（或函数名），每当 readyState 属性改变时，就会调用该函数。 |
| readyState | 存有 XMLHttpRequest 的状态。从 0 到 4 发生变化。   * 0: 请求未初始化 * 1: 服务器连接已建立 * 2: 请求已接收 * 3: 请求处理中 * 4: 请求已完成，且响应已就绪 |
| status | 200: "OK"  404: 未找到页面 |

在 onreadystatechange 事件中，我们规定当服务器响应已做好被处理的准备时所执行的任务。

当 readyState 等于 4 且状态为 200 时，表示响应已就绪：

|  |
| --- |
| xmlhttp.onreadystatechange=function()  {  if (xmlhttp.readyState==4 && xmlhttp.status==200)  {  document.getElementById("myDiv").innerHTML=xmlhttp.responseText;  }  } |

## AJAX请求案例

|  |
| --- |
| <html>  <body>  用户名：<input type="text" id="userName" onblur="getServerText()">  <span id="txtTip">\*</span>  </body>  </html>  <script>  //创建xhr对象  function createXhr(){  var xhr=null;  if(window.XMLHttpRequest){  xhr=new XMLHttpRequest();  }else {  xhr=new ActiveXObject("Microsoft.XMLHttp")  }  return xhr;  }  //异步请求AJAX方法  function getServerText(){  //1.获取xhr  var xhr=createXhr();  //2,创建请求  var userName=document.getElementById("userName").value;  var url="source/2.php?userName="+userName;  xhr.open("get",url);//如果有参数，写在1.php后面用&连接  //3,设置回调函数  xhr.onreadystatechange=function(){  if(xhr.readyState==4 && xhr.status==200){//必须按照这样的顺序  var resTxt=xhr.responseText;  if(resTxt=="通过!"){  document.getElementById("txtTip").innerHTML="ok";  }else{  document.getElementById("txtTip").innerHTML="这个名字你就别用了";  }  }  };  //4,发送请求  xhr.send();//get 请求无请求体  }  </script> |

# jQuery简介

jQuery 是一个 JavaScript 库。

jQuery 极大地简化了 JavaScript 编程。

jQuery 很容易学习。

jQuery 库可以通过一行简单的标记被添加到网页中。

jQuery库包含以下功能：

* HTML 元素选取
* HTML 元素操作
* CSS 操作
* HTML 事件函数
* JavaScript 特效和动画
* HTML DOM 遍历和修改
* AJAX
* Utilities

目前网络上有大量开源的 JS 框架, 但是 jQuery 是目前最流行的 JS 框架，而且提供了大量的扩展。

很多大公司都在使用 jQuery， 例如:

* Google
* Microsoft
* IBM
* Netflix

# jQuery 安装

可以通过多种方法在网页中添加 jQuery。 可以使用以下方法：

* 从 [jquery.com](http://jquery.com/download/) 下载 jQuery 库
* 从 CDN 中载入 jQuery, 如从 Google 中加载 jQuery

有两个版本的 jQuery 可供下载：

* Production version - 用于实际的网站中，已被精简和压缩。
* Development version - 用于测试和开发（未压缩，是可读的代码）

以上两个版本都可以从 [jquery.com](http://jquery.com/download/) 中下载。

jQuery 库是一个 JavaScript 文件，可以使用 HTML 的 <script> 标签引用它：

|  |
| --- |
| <head> <script src="jquery-1.12.4.min.js"></script> </head> |

**提示：** 将下载的文件放在网页的同一目录下，就可以使用jQuery。

如果不希望下载并存放 jQuery，那么也可以通过 CDN（内容分发网络） 引用它。

百度、又拍云、新浪、谷歌和微软的服务器都存有 jQuery 。

案例：百度CDN

|  |
| --- |
| <head> <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"> </script> </head>  <script src="http://libs.baidu.com/jquery/1.11.1/jquery.min.js"></script> |

# jQuery 语法

通过 jQuery，您可以选取（查询，query） HTML 元素，并对它们执行"操作"（actions）。

基础语法： **$(*selector*).*action*()**

* 美元符号定义 jQuery
* 选择符（selector）"查询"和"查找" HTML 元素
* jQuery 的 action() 执行对元素的操作

实例:

$(this).hide() - 隐藏当前元素

$("p").hide() - 隐藏所有 <p> 元素

$("p.test").hide() - 隐藏所有 class="test" 的 <p> 元素

$("#test").hide() - 隐藏所有 id="test" 的元素

## 文档就绪事件

为了防止文档在完全加载（就绪）之前运行 jQuery 代码，即在 DOM 加载完成后才可以对 DOM 进行操作。

如果在文档没有完全加载之前就运行函数，操作可能失败。解决该情况有以下两种方式：

（1）、$(document).ready(function(){ // 开始写 jQuery 代码... });

（2）、简洁写法（与以上写法效果相同）:

$(function(){ // 开始写 jQuery 代码... });

# jQuery 选择器

jQuery 选择器基于元素的 id、类、类型、属性、属性值等"查找"（或选择）HTML 元素。 它基于已经存在的 [CSS 选择器](https://www.runoob.com/cssref/css-selectors.html)，除此之外，它还有一些自定义的选择器。

jQuery 中所有选择器都以美元符号开头：$()。

## 标签选择器

jQuery 元素选择器基于元素名选取元素。

在页面中选取所有 <p> 元素:

|  |
| --- |
| <p></p>  $("p") |

## #id 选择器

jQuery #id 选择器通过 HTML 元素的 id 属性选取指定的元素。

页面中元素的 id 应该是唯一的，所以您要在页面中选取唯一的元素需要通过 #id 选择器。

通过 id 选取元素语法如下：

|  |
| --- |
| <p id=”test”></p>  $("#test") |

## .class 选择器

jQuery 类选择器可以通过指定的 class 查找元素。

语法如下：

|  |
| --- |
| <p class=”test”></p>  $(".test") |

## 其它常用选择器

|  |
| --- |
| $("\*") 选取所有元素  $(this) 选取当前 HTML 元素 |

# jQuery 事件

页面对不同访问者的响应叫做事件。

事件处理程序指的是当 HTML 中发生某些事件时所调用的方法。

## 常用的 jQuery 事件方法

### click()

click() 方法是当按钮点击事件被触发时会调用一个函数。

该函数在用户点击 HTML 元素时执行。

|  |
| --- |
| 案例：点击p标签隐藏  $("p").click(function(){ $(this).hide(); }); |

### dblclick()

当双击元素时，会发生 dblclick 事件。

dblclick() 方法触发 dblclick 事件，或规定当发生 dblclick 事件时运行的函数：

|  |
| --- |
| $("p").dblclick(function(){ $(this).hide(); }); |

### mouseenter()

当鼠标指针穿过元素时，会发生 mouseenter 事件。

mouseenter() 方法触发 mouseenter 事件，或规定当发生 mouseenter 事件时运行的函数：

|  |
| --- |
| $("#p1").mouseenter(function(){ alert('您的鼠标移到了 id="p1" 的元素上!'); }); |

### mouseleave()

当鼠标指针离开元素时，会发生 mouseleave 事件。

mouseleave() 方法触发 mouseleave 事件，或规定当发生 mouseleave 事件时运行的函数：

|  |
| --- |
| $("#p1").mouseleave(function(){ alert("再见，您的鼠标离开了该段落。"); }); |

### mousedown()

当鼠标指针移动到元素上方，并按下鼠标按键时，会发生 mousedown 事件。

mousedown() 方法触发 mousedown 事件，或规定当发生 mousedown 事件时运行的函数：

|  |
| --- |
| $("#p1").mousedown(function(){ alert("鼠标在该段落上按下！"); }); |

### mouseup()

当在元素上松开鼠标按钮时，会发生 mouseup 事件。

mouseup() 方法触发 mouseup 事件，或规定当发生 mouseup 事件时运行的函数：

|  |
| --- |
| $("#p1").mouseup(function(){ alert("鼠标在段落上松开。"); }); |

### hover()

hover()方法用于模拟光标悬停事件。

当鼠标移动到元素上时，会触发指定的第一个函数(mouseenter);当鼠标移出这个元素时，会触发指定的第二个函数(mouseleave)。

|  |
| --- |
| $("#p1").hover( function(){ alert("你进入了 p1!"); }, function(){ alert("拜拜! 现在你离开了 p1!"); } ); |

### focus()

当元素获得焦点时，发生 focus 事件。

当通过鼠标点击选中元素或通过 tab 键定位到元素时，该元素就会获得焦点。

focus() 方法触发 focus 事件，或规定当发生 focus 事件时运行的函数：

|  |
| --- |
| $("input").focus(function(){ $(this).css("background-color","#cccccc"); }); |

### blur()

当元素失去焦点时，发生 blur 事件。

blur() 方法触发 blur 事件，或规定当发生 blur 事件时运行的函数：

|  |
| --- |
| $("input").blur(function(){ $(this).css("background-color","#ffffff"); }); |

# jQuery DOM 操作

jQuery 中非常重要的部分，就是操作 DOM 的能力。

jQuery 提供一系列与 DOM 相关的方法，这使访问和操作元素和属性变得很容易。

## 获得内容 - text()、html() 以及 val()

三个简单实用的用于 DOM 操作的 jQuery 方法：

* text() - 设置或返回所选元素的文本内容
* html() - 设置或返回所选元素的内容（包括 HTML 标记）
* val() - 设置或返回表单字段的值

案例：

|  |
| --- |
| $("#btn1").click(function(){ alert("Text: " + $("#test").text()); }); $("#btn2").click(function(){ alert("HTML: " + $("#test").html()); });  $("#btn1").click(function(){ alert("值为: " + $("#test").val()); }); |

## 获取属性 - attr()

jQuery attr() 方法用于获取属性值。

下面的例子演示如何获得链接中 href 属性的值：

|  |
| --- |
| $("button").click(function(){ alert($("#runoob").attr("href")); }); |

## 设置内容 - text()、html() 以及 val()

我们将使用前一章中的三个相同的方法来设置内容：

* text() - 设置或返回所选元素的文本内容
* html() - 设置或返回所选元素的内容（包括 HTML 标记）
* val() - 设置或返回表单字段的值

案例：

|  |
| --- |
| $("#btn1").click(function(){ $("#test1").text("Hello world!"); }); $("#btn2").click(function(){ $("#test2").html("<b>Hello world!</b>"); }); $("#btn3").click(function(){ $("#test3").val("RUNOOB"); }); |

## jQuery - 添加元素

* append() - 在被选元素的结尾插入内容
* prepend() - 在被选元素的开头插入内容
* after() - 在被选元素之后插入内容
* before() - 在被选元素之前插入内容

|  |
| --- |
| $("p").append("追加文本");  $("p").prepend("在开头追加文本");  $("img").after("在后面添加文本");  $("img").before("在前面添加文本"); |

## jQuery - 删除元素

如需删除元素和内容，一般可使用以下两个 jQuery 方法：

* remove() - 删除被选元素（及其子元素）
* empty() - 从被选元素中删除子元素

|  |
| --- |
| $("#div1").remove();  $("#div1").empty(); |

## jQuery css() 方法

css() 方法设置或返回被选元素的一个或多个样式属性。

下面的例子将返回首个匹配元素的 background-color 值：

|  |
| --- |
| $("p").css("background-color"); |

如需设置指定的 CSS 属性，请使用如下语法：

css("*propertyname*","*value*");

|  |
| --- |
| $("p").css("background-color","yellow"); |

如需设置多个 CSS 属性，请使用如下语法：

css({"*propertyname*":"*value*","*propertyname*":"*value*",...});

|  |
| --- |
| $("p").css({"background-color":"yellow","font-size":"200%"}); |

# jQuery 与 AJAX

jQuery 提供多个与 AJAX 有关的方法。

通过 jQuery AJAX 方法，您能够使用 HTTP Get 和 HTTP Post 从远程服务器上请求文本、HTML、XML 或 JSON - 同时您能够把这些外部数据直接载入网页的被选元素中。

jQuery.ajax([*settings*])

|  |
| --- |
| **$**.**ajax**({   **url**:**"hello.php"**,  **type**: **"POST"**,  **data**: {**'search'**: first\_name},  success: **function** (data) {  *alert*(data + **"123"**);  *check*.preventDefault();  },  error: **function** () {  *alert*(**'error'**);  } }); |