Monday, April 15, 2019

AJAX

Introduction

- AJAX = Asynchronous JavaScript and XML (不是一种新的编程语言，而是一种用于创建更好，更快，更具交互性的Web应用程序的技术。)
  - 使用AJAX，JavaScript可以使用JavaScript XMLHttpRequest对象直接与server通信
  - 使用此对象，JavaScript可以与Web server交换数据，而无需重新加载页面
  - AJAX在client和Web server之间使用异步数据传输（HTTP请求），允许Web页面从server而不是whole page请求少量信息。
  - AJAX技术使Internet应用程序更小，更快，更易于用户使用。是一种独立于Web服务器软件的浏览器技术。
- AJAX is Based on Web Standards
  - AJAX基于以下Web标准，这些标准定义明确，并得到所有主流浏览器的支持。 AJAX应用程序独立于浏览器/平台。
    - JavaScriptXML, HTML, CSS
- AJAX是关于更好的互联网应用程序
  - Web应用程序比桌面应用程序有许多好处:
    - 他们可以覆盖更多的观众，它们更易于安装和支持，更容易开发。
  - 但是，Internet应用程序并不总是像传统桌面应用程序一样“丰富”和用户友好。借助AJAX，可以使Internet应用程序更丰富，更加用户友好。

AJAX概念

- AJAX Uses HTTP Requests
  - 在传统的JavaScript coding中，如果要从server上的数据库或文件中获取任何信息，或将用户信息发送到server，必须将HTML表单和GET或POST数据发送到server
  - 用户必须点“提交”按钮发送/获取信息，等待服务器响应，然后新页面将加载结果
  - 每次用户提交输入时server都会返回一个新页面，因此传统的Web应用程序运行缓慢且对用户友好性较低。
  - 使用AJAX，JavaScript直接与server通信，通过JavaScript XMLHttpRequest对象和HTTP request，网页可以向Web server发出请求并从Web server获取响应，无需重新加载整个页面
  - 用户保持在同一页面，并且他不会注意到脚本请求页面或在后台将数据发送到服务器
- The XMLHttpRequest Object
  - 通过使用XMLHttpRequest对象，在页面加载后Web开发人员仍然可以可以使用服务器中的数据更新页面!
    - Google网页界面例子：在Google的搜索框中输入内容时，JavaScript会将这些字母发送到服务器，服务器会返回一个建议列表
    - AJAX - 浏览器支持：AJAX的基石是XMLHttpRequest对象

XMLHttpRequest对象的三个重要属性

Property	Description
onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 404: Page not found

- XMLHttpRequest对象用于和sever交换数据
- 1. The onreadystatechange Property
  - 在向server发出请求后，我们需要一个可以接收服务器返回的数据的function
  - onreadystatechange属性存储function，这个function处理来自服务器的响应【这不是一个method，该function被存储在自动调用的属性中】
  - 以下代码设置onreadystatechange属性并在其中存储一个空函数：
    - xmlHttp.onreadystatechange = function（）{//我们打算在这里写一些代码}
- 2. The readyState Property
  - 当发送对server的请求时，我们希望根据响应执行一些操作
  - readyState每次被更改后都会触发onreadystatechange事件
  - readyState属性保存XMLHttpRequest的状态（status）
  - XML XMLHttpRequest对象的三个重要属性：在onreadystatechange事件中，我们指定在当服务器响应准备好被处理时将发生的情况：当readyState为4且status为200时，响应就绪
    - xmlhttp.onreadystatechange=function() { if (xmlhttp.readyState==4 && xmlhttp.status==200){document.getElementById("myDiv").innerHTML=xmlhttp.responseText; } }
  - Using a Callback Function
    - 回调函数是作为参数传递给另一个函数的函数。
    - 如果您的网站上有多个AJAX任务，则应创建一个标准function来创建XMLHttpRequest对象，并为每个AJAX任务调用此函数
  - 函数调用应该包含URL以及onreadystatechange上要做什么（每个call可能不同）
    - function myFunction(){loadXMLDoc("ajax_info.php",function() { if (xmlhttp.readyState==4 && xmlhttp.status==200) { document.getElementById("myDiv").innerHTML=xmlhttp.responseText; } });}
- 3. The responseText Property
  - responseText Property检索从server传回的数据。在例子中把time设置为从server取回的值：
    - xmlHttp.onradystatechange=function(){if(xmlHttp.readyState==4){document.myForm.ome.value=xmlHttp.responseText;}}

```
<script type="text/javascript">
function ajaxFunction(strUser){
        var xmlHttp;
        try     // Firefox, Opera 8.0+, Safari{
                xmlHttp=new XMLHttpRequest();
        }catch (e){
                try  // Internet Explorer{
                        xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
                }catch (e){
                        try{
                                xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
                        }catch (e){
                                alert("Your browser does not support AJAX!");
                                return false;
                        }
                }
        }

        xmlHttp.onreadystatechange=function(){
                if(xmlHttp.readyState==4){
                        var infodiv = document.getElementById("infodiv");
                        infodiv.innerHTML = xmlHttp.responseText;
                        infodiv.setAttribute('class', 'visible');
                }
        }

xmlHttp.open("POST","suggest.php",true);
xmlHttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlHttp.send("user=" + strUser);
}
</script>
<form name=“”myForm>
        Name: <input type=“text” onkeyup=“ajaxFunction();” name=“userName”/>
        Time:<input type=“text” name=“time”/>
[There is no submit button. ]
</form></body></html>
```

AJAX - Sending a Request to the Server

- GET还是POST?
  - GET比POST更简单，更快，并且可以在大多数情况下使用。但是，在以下情况下始终使用POST请求：
    - 缓存文件不是一个选项（更新服务器上的文件或数据库）
    - 向服务器发送大量数据（POST没有大小限制）
    - 发送用户输入（可能包含未知字符），POST比GET更强大和安全

The XMLHttpRequest object is used to exchange data with a server.

Send a Request To a Server

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

```
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
```

Method	Description
open(*method,url,async*)	Specifies the type of request, the URL, and if the request should be handled asynchronously or not. *method*: the type of request: GET or POST *url*: the location of the file on the server *async*: true (asynchronous) or false (synchronous)
send(*string*)	Sends the request off to the server. *string*: Only used for POST requests

GET Requests

A simple GET request:

Example

```
xmlhttp.open("GET","demo_get.php",true);
xmlhttp.send();
```

In the example above, you may get a cached result.

To avoid this, add a unique ID to the URL:

Example

```
xmlhttp.open("GET","demo_get.php?t=" + Math.random(),true);
xmlhttp.send();
```

If you want to send information with the GET method, add the information to the URL:

Example

```
xmlhttp.open("GET","demo_get2.php?fname=Henry&lname=Ford",true);
xmlhttp.send();
```

POST Requests

A simple POST request:

Example

```
xmlhttp.open("POST","demo_post.php",true);
xmlhttp.send();
```

To POST data like an HTML form, add an HTTP header with setRequestHeader(). Specify the data you want to send in the send() method:

Example

```
xmlhttp.open("POST","ajax_test.php",true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlhttp.send("fname=Henry&lname=Ford");
```

Method	Description
setRequestHeader(*header,value*)	Adds HTTP headers to the request. *header*: specifies the header name *value*: specifies the header value

Monday, April 15, 2019

- Url - 服务器上的文件
  - open（）方法的url参数是服务器上文件的地址：xmlhttp.open（“GET”，“ajax_test.php”，true）；
  - 文件可以是任何类型的文件，如.txt和.xml，或服务器脚本文件，如.asp和.php（可以在发送响应之前在服务器上执行操作）
- Asynchronous - true or false
  - AJAX代表异步JavaScript和XML，并且对于XMLHttpRequest对象表现为AJAX，open（）方法的async参数必须设置为true：
  - 使用AJAX，JavaScript不必等待服务器响应，在等待服务器响应时执行其他脚本，在响应准备就绪时处理响应
  - async = TRUE
    - 使用async = true时，指定在onreadystatechange事件中响应准备好时要执行的函数：
      - xmlhttp.onreadystatechange = function（）{ if（xmlhttp.readyState == 4 && xmlhttp.status == 200）{document.getElementById（“myDiv”）.innerHTML = xmlhttp.responseText;}}
        xmlhttp.open（“GET”，“ajax_info.php”，true）;xmlhttp.send（）；
  - async = FALSE【不建议使用async = false，但对于一些小的请求，这可以】
    - 请记住，在server 的响应准备好之前，JavaScript不会继续执行。如果服务器忙或慢，应用程序将挂起或停止。
    - 注意：当你使用async = false时，不要编写onreadystatechange函数，只需将代码放在send（）语句之后：
      - xmlhttp.open("GET","ajax_info.txt",false);
        xmlhttp.send(); document.getElementById(“myDiv”).innerHTML=xmlhttp.responseText

Server Response

To get the response from a server, use the responseText or responseXML property of the XMLHttpRequest object.

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

The responseText Property

If the response from the server is not XML, use the responseText property.

The responseText property returns the response as a string, and you can use it accordingly:

Example

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

The responseXML Property

If the response from the server is XML, and you want to parse it as an XML object, use the responseXML property:

Example

Request the file cd_catalog.xml and parse the response:

```
xmlDoc=xmlhttp.responseXML;
txt="";
x=xmlDoc.getElementsByTagName("ARTIST");
for (i=0;i<x.length;i++)
  {
  txt=txt + x[i].childNodes[0].nodeValue + "<br>";
  }
document.getElementById("myDiv").innerHTML=txt;
```

JSON

```jsp
<%@page contentType="text/html; charset=UTF-8"%>
<%@page import="org.json.simple.JSONObject"%>
<%
  JSONObject obj=new JSONObject();
  obj.put("name","foo");
  obj.put("num",new Integer(100));
  obj.put("balance",new Double(1000.21));
  obj.put("is_vip",new Boolean(true));
  obj.put("nickname",null);
  out.print(obj);
  out.flush();
%>
```

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

<script type="text/javascript">
function createXMLHttpRequest(){
  // See http://en.wikipedia.org/wiki/XMLHttpRequest
  // Provide the XMLHttpRequest class for IE 5.x-6.x:
  if( typeof XMLHttpRequest == "undefined" ) XMLHttpRequest = function() {
    try { return new ActiveXObject("Msxml2.XMLHTTP.6.0") } catch(e) {}
    try { return new ActiveXObject("Msxml2.XMLHTTP.3.0") } catch(e) {}
    try { return new ActiveXObject("Msxml2.XMLHTTP") } catch(e) {}
    try { return new ActiveXObject("Microsoft.XMLHTTP") } catch(e) {}
    throw new Error( "This browser does not support XMLHttpRequest." )
  };
  return new XMLHttpRequest();
}

var AJAX = createXMLHttpRequest();

function handler() {
  if(AJAX.readyState == 4 && AJAX.status == 200) {
      var json = eval('(' + AJAX.responseText +')');
      alert('Success. Result: name => ' + json.name + ',' + 'balance => ' + json.balance);
  }else if (AJAX.readyState == 4 && AJAX.status != 200) {
    alert('Something went wrong...');
  }
}

function show(){
  AJAX.onreadystatechange = handler;
  AJAX.open("GET", "service.jsp");
  AJAX.send("");
};
</script>

<body>
  <a href="#" onclick="javascript:show();"> Click here to get JSON data from the server sid
</html>
```