AJAX DAY01:

* 基本内容
  * 同步交互与异步交互
    * 同步交互

      向服务器端发送请求,到服务器端进行响应,这个过程中,用户不能做任何其他事情.
    * 异步交互

      向服务器端发送请求,到服务器端进行响应,这个过程中,用户可以做任何其他事情.
  * 异步交互的场景
    * 百度一下的搜索框
    * 网页中的评论功能
    * 页面中局部数据更新
  * Ajax 的定义(不严格)
    * 客户端向服务器端发送请求,而无需刷新页面的技术
  * Ajax 的全称
    * Asynchronous JavaScript and Xml
    * 异步的 JavaScript 和 Xml
    * JavaScript 和 XML 的异步
  * Ajax 的特点
    * 并不是一种技术
  * Ajax 包含的技术
    * HTML、CSS、JavaScript、XML 和 XMLHttpRequest 等
  * 异步交互的技术
    * <iframe src="">元素
  * Ajax 的核心对象
    * XMLHttpRequest 对象
* XMLHttpRequest 对象
  * 基础
    * XMLHttpRequest 对象是一个 JavaScript 对象
      * 在使用上类似于 JavaScript 其他内置对象
    * XMLHttpRequest 对象的标准是 W3C
  * 方法
    * open(url,type)方法
      * 与服务器端建立连接
    * send()方法
      * 向服务器端发送数据
    * setRequestHeader(header,value)方法
      * 设置请求头信息
    * getResponseHeader(header)方法
      * 获取响应头信息
    * getAllResponseHeaders()方法
      * 获取所有响应头信息
  * 事件
    * onreadystatechange 事件
      * 用于监听服务器端的通信状态
  * 属性
    * readyState 属性
      * 获取服务器端的通信状态
    * status 属性
      * 获取服务器端的状态码
    * responseText 属性
      * 获取服务器端的响应数据(文本格式)
    * responseXML 属性
      * 获取服务器端的响应数据(XML 格式)
* 如何获取核心对象(XMLHttpRequest) - 固定写法

```
function getXhr(){
    var xhr = null;//定义核心对象
    // 完成浏览器的兼容性
    if(window.XMLHttpRequest){//其他浏览器
        xhr = new XMLHttpRequest();
    }else{//IE 浏览器
     xhr = new ActiveXObject("Microsoft.XMLHttp");
    }
    return xhr;//返回核心对象
}
```

* 实现 Ajax 的步骤
  * 创建 Ajax 的核心对象 XMLHttpRequest

```
function getXhr(){
    var xhr = null;
    if(window.XMLHttpRequest){
        xhr = new XMLHttpRequest();
    }else{
     xhr = new ActiveXObject("Mirosoft.XMLHttp");
    }
    return xhr;
}
```

  * 与服务器端建立连接
    * 使用 XMLHttpRequest 对象的 open()方法
      * type - 表示请求类型(GET 或 POST)
      * url - 表示请求地址
      * async - 表示是否异步
  * 向服务器端发送请求数据
    * 使用 XMLHttpRequest 对象的 send()方法
  * 接收服务器端响应的数据
    * 通过 XMLHttpRequest 对象绑定 onreadystatechange 事件
    * 判断 XMLHttpRequest 对象的 readyState 属性的值
    * 0 - (服务器端)未初始化
    * 1 - 正在连接
    * 2 - 正在接收(请求)
    * 3 - 正在响应
    * 4 - 响应完毕
    * 判断 XMLHttpRequest 对象的 status 属性的值
      * status 属性 - 表示服务器端的状态码
* Ajax 实现需要注意的
  * open(type,url,async)方法
    * 之前的标准 - async 的值为 true 或 false
    * 最新的标准 - async 的值只为 true
      * XMLHttpRequest 对象只能实现异步交互

* open(type,url)

* send()方法

* 如果请求类型为 GET

* 问题

* send()方法无法成功地发送请求数据

* 注意

* send()方法实现 Ajax 时,不能被省略的

* send(null) - 固定写法

* 请求数据增加在 URL 后面

* 如果请求类型为 POST

* 注意

* 在调用 send()方法之前,设置请求头信息

```
xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
```

* 代码段最后实现接收响应时

* readyState 属性值,只能 2,3,4 值

* 实现 Ajax 的 GET 与 POST 区别

* GET 方式

* open()方法的 type 参数设置为 GET

* 请求数据增加在 URL(请求路径)的后面

* XMLHttpRequest 对象的 send()方法无效

* send()方法不能被省略 - send(null)

* POST 方式

* open()方法的 type 参数设置为 POST

* 请求数据通过 send()方法进行发送

* 必须在 send()方法前,设置请求头信息

* Ajax 实现的步骤(另一种说法)

* 创建核心对象 XMLHttpRequest

* 注册监听 - onreadystatechange

* 建立连接

* 发送请求

* 实现 Ajax 异步交互的六步是什么?

* 创建核心对象

* 建立连接

* 发送请求

* 注册监听(接收响应)

* 获取状态 - readyState

* 获取状态码

* 扩展内容

HTML5 另一个标准 - https://whatwg.org/

```
<title>实现 Ajax 的步骤</title>
<body>
 <input type="button" id="btn" value="异步请求">
 <script>
    var btn = document.getElementById("btn");
    btn.onclick = function(){
        /* 1. 创建 XMLHttpRequest 对象*/
        var xhr = getXhr();
        /* 2. 建立与服务器端之间的连接
            XMLHttpRequest 对象的 open()方法
            open(type,url,async)方法
            * type - 请求类型(GET 或 POST)
            * url - 当前的请求地址
            * async - 表示是否异步,Boolean 类型
              * true - 默认值,表示异步
              * false - 表示同步 */
    xhr.open("get","01.php?user=zhangwuji&pwd=123456");
        /*  3. 向服务器端发送请求数据
            XMLHttpRequest 对象的 send()方法
            send(data)方法
            * data - 表示请求数据
            注意
            * 请求数据格式 - key=value
            send()方法的问题
            * 如果请求类型为 GET 方式的话
              * 该方法 send()方法不起作用
              * 该方法不能被省略 - xhr.send(null)
              * 请求数据增加在 URL?key=value*/
        xhr.send(null);
        /*  4. 接收服务器端的响应数据
            XMLHttpRequest 对象的 onreadystatechange 事件
            * 作用 - 该事件用于监听服务器端的通信状态
            XMLHttpRequest 对象的 readyState 属性
            * 作用 - 表示服务器端的通信状态
            * 选项
              * 0 - (服务器端)未初始化
              * 1 - (服务器端)正在连接
              * 2 - 正在接收(请求数据)
              * 3 - 正在响应(给客户端)
              * 4 - 响应完毕
            XMLHttpRequest 对象的 status 属性
            * 作用 - 表示服务器端的状态码
            XMLHttpRequest 对象的 responseText 属性
        * 作用 - 将服务器端响应的数据作为文本接收 */
        xhr.onreadystatechange = function(){
            if(xhr.readyState == 4){
                if(xhr.status == 200){
                    // 接收响应数据
                    var data = xhr.responseText;
```

```
                    console.log(data);
                }
            }
        }
    }

    function getXhr(){
        var xhr = null;
        if(window.XMLHttpRequest){
            xhr = new XMLHttpRequest();
        }else{
            xhr = new ActiveXObject("Microsoft.XMLHttp");
        }
        return xhr;
    }
</script>
<?php
    // 1. 接收客户端的请求数据
    $user = $_GET['user'];
    $pwd = $_GET['pwd'];
    // 2. 向客户端进行响应
    echo "$user : $pwd";
?>
```

<title>Ajax 的 POST 请求类型</title>

```
<meta charset="utf-8" />
<input type="button" id="btn" value="异步请求">
<script>
    var btn = document.getElementById("btn");
    btn.onclick = function(){
        // 1. 创建核心对象
        var xhr = getXhr();
        // 2. 建立连接
        xhr.open("post","02.php");
        /* 3. 发送请求
            注意 - 请求类型为 POST
            * 必须在调用 send()方法前,设置请求头信息
xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");*/
    xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
        xhr.send("user=zhangwuji&pwd=123456");
        // 4. 接收响应
        xhr.onreadystatechange = function(){
            if(xhr.readyState == 4 && xhr.status == 200){
                console.log(xhr.responseText);
            }
        }
    }
    function getXhr(){
        var xhr = null;
        if(window.XMLHttpRequest){
            xhr = new XMLHttpRequest();
        }else{
            xhr = new ActiveXObject("Microsoft.XMLHttp");
        }
        return xhr;
    }
</script>
<?php
    $user = $_POST['user'];
    $pwd = $_POST['pwd'];
    echo "$user : $pwd";
?>
<!--
    <form>表单
    * 除了常用的属性
        * id 属性
        * name 属性
        * action 属性
        * method 属性
    * 默认属性
        * enctype="application/x-www-form-urlencoded"
    表单的默认请求"Content-Type"
    Content-Type:application/x-www-form-urlencoded
-->
```

<title>Ajax 中的 readyState</title>

```
<input type="button" id="btn" value="异步请求">
<script>
    var btn = document.getElementById("btn");
    btn.onclick = function(){
        // 1. 创建核心对象 XMLHttpRequest
        var xhr = getXhr();
        /*2. 注册监听
            * 多得到的 1 状态,表示在 open 之前也有状态
        * open 之前,客户端与服务器端之间并没有建立连接
            * 在客户端无法获取到服务器端的通信状态
        * readyState 属性的值为 0 时,无法得到
            * 0 - 表示服务器端未初始化
    * 在实际开发中,最关心的是 readyState 属性值为 4 时*/
        xhr.onreadystatechange = function(){
            alert(xhr.readyState);
        }
        // 3. 建立连接
        xhr.open("get","04.php");
        // 4. 发送请求
        xhr.send(null);
    }
    function getXhr(){
        var xhr = null;
```

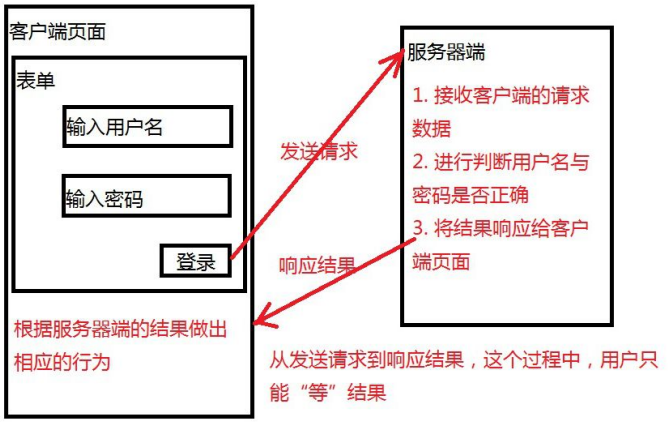
```
            if(window.XMLHttpRequest){
                xhr = new XMLHttpRequest();
            }else{
                xhr = new ActiveXObject();
            }
            return xhr;
        }</script>
<?php
    echo 'readystate';
?>
```

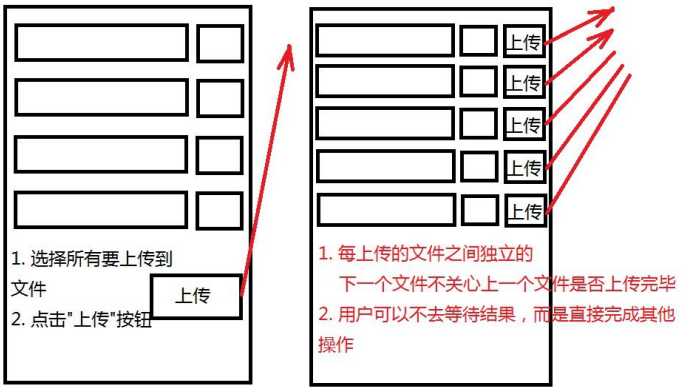
<title>二级联动</title>

```
    <select id="province">
        <option>请选择</option>
        <option>吉林省</option>
        <option>辽宁省</option>
        <option>山东省</option>
    </select>
    <select id="city"><option>请选择</option></select>
    <script>
        // 需求 - 根据用户选择省份信息,提供城市列表
        // 1. 获取省份下拉列表,绑定 onchange 事件
        var province = document.getElementById("province");
        province.onchange = function(){
            // 清空城市列表
            var cityEle = document.getElementById("city");
            var opts = cityEle.getElementsByTagName("option");
            for(var z=opts.length-1;z>0;z--){
                cityEle.removeChild(opts[z]);
            }
            /*2. 获取用户选择的省份信息
                * 获取 select 元素下的所有 option 元素
                 * 遍历获取的所有 option 元素
                  * 判断 option 哪个被选中
                * 利用 select 元素的特性
             * 直接通过 select 元素的 value 属性得到被选中的值 */
            var provinceValue = province.value;
            // 3. 利用 Ajax 的异步交互
                // a. 创建核心对象
                var xhr = getXhr();
                // b. 建立连接
                xhr.open("post","05.php");
        xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
                // c. 发送请求 - 将省份信息作为请求数据
                if(provinceValue != "请选择"){
                    xhr.send("province="+provinceValue);}
                // d. 接收响应
                xhr.onreadystatechange = function(){
                if(xhr.readyState == 4 && xhr.status == 200){
        var cities = xhr.responseText;
// 4. 解析获取的城市列表,写入到下拉列表中
    // a. 将字符串解析为数组
        var citiesArr = cities.split(",");
        // b. 遍历数组,得到每个城市信息
        for(var i=0;i<citiesArr.length;i++){
            // c. 得到每个城市信息
            var city = citiesArr[i];
            // d. 创建 option 元素
            var option = document.createElement("option");
/*f. 将文本内容插入到 option 元素内
        * 作为文本插入到 option 元素
        * textContent 或 innerText 属性可以获取或设置指定元素的文本内容
        * 作为节点方式
        * 创建文本节点 - createTextNode(text)
        * 将文本节点作为子节点*/
        var text = document.createTextNode(city);
        option.appendChild(text);
        // g. 插入到 select 元素内
        cityEle.appendChild(option);
                    }
                }
            }
        }
        function getXhr(){
            var xhr = null;
            if(window.XMLHttpRequest){
                xhr = new XMLHttpRequest();
            }else{
                xhr = new ActiveXObject("Microsoft.XMLHttp");
            }
            return xhr;
        }        </script>
<?php
    // 1. 接收客户端的请求数据
    $province = $_POST['province'];
    // 2. 判断省份是什么(不同省份响应不同的城市)
    switch ($province){
        case '吉林省':
        $cities = '长春市,通化市,松原市,白城市,辽源市';
            break;
        case '辽宁省':
        $cities = '沈阳市,大连市,铁岭市,锦州市,丹东市';
            break;
        case '山东省':
        $cities = '济南市,青岛市,威海市,日照市,德州市';
            break;}
        // 3. 向客户端进行响应
    echo $cities;
?>
```

```
<title>二级联动-修改</title>
    <select id="province">
        <option>请选择</option>
        <option>吉林省</option>
        <option>辽宁省</option>
        <option>山东省</option>
    </select>
    <select id="city">
        <option>请选择</option>
    </select>
    <script>
        var provinceEle = document.getElementById("province");
        provinceEle.onchange = function(){
            // 清空
            var cityEle = document.getElementById("city");
            var opts = cityEle.getElementsByTagName("option");
            for(var z=opts.length-1;z>0;z--){
                cityEle.removeChild(opts[z]);
            }
            // 获取选中的省份
            var provinceValue = provinceEle.value;
            // 实现 Ajax
            var xhr = getXhr();
            xhr.open("post","06.php");
        xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
            xhr.send("province="+provinceValue);
            xhr.onreadystatechange = function(){
                if(xhr.readyState==4&&xhr.status==200){
                    // 响应数据类型 - String
                    var cities = xhr.responseText;
                    cityEle.innerHTML = cities;
                }
            }
        }
        function getXhr(){
            var xhr = null;
            if(window.XMLHttpRequest){
                xhr = new XMLHttpRequest();
            }else{
                xhr = new ActiveXObject("Microsoft.XMLHttp");
            }
            return xhr;
        }
    </script>
<?php
        $province = $_POST['province'];
        switch ($province){
            case '吉林省':
                $cities = '<option>长春市</option>'.
                    '<option>通化市</option>'.
                    '<option>松原市</option>'.
                    '<option>白城市</option>'.
                    '<option>辽源市</option>';
                    break;
            case '辽宁省':
                $cities = '<option>沈阳市</option>'.
                    '<option>大连市</option>'.
                    '<option>铁岭市</option>'.
                    '<option>锦州市</option>'.
                    '<option>丹东市</option>';
                    break;
            case '山东省':
                $cities = '<option>济南市</option>'.
                    '<option>青岛市</option>'.
                    '<option>威海市</option>'.
                    '<option>日照市</option>'.
                     '<option>德州市</option>';
                    break;
        }
        // 应该是符合 HTML 代码的字符串
        echo $cities;
?>
<title>二级联动-final</title>
    <select id="province">
        <option>请选择</option>
    </select>
    <select id="city">
        <option>请选择</option>
    </select>
    <script>
        var provinceEle = document.getElementById("province");
        window.onload = function(){
            var xhr=getXhr();
            xhr.open("post","07province.php");
        xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
            xhr.send(null);
            xhr.onreadystatechange=function(){
                if (xhr.readyState==4 && xhr.status==200){
                    var province=xhr.responseText;
                    provinceEle.innerHTML+=province;
                }
            }
        }
        provinceEle.onchange = function(){
            var citySel=document.getElementById("city");
            var xhr=getXhr();
            var provinceValue = provinceEle.value;
            xhr.open("post","07cities.php");
```

```
        xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
            if (provinceValue!="请选择"){
                xhr.send("provinceEle="+provinceValue);
            }
            xhr.onreadystatechange=function(){
                if (xhr.readyState==4 && xhr.status==200){
                    var cities=xhr.responseText;
                    console.log(cities);
                    citySel.innerHTML+=cities;
                }
            }
        }
        function getXhr(){
            var xhr=null;
            if (window.XMLHttpRequest){
                xhr=new XMLHttpRequest();
            }else{
                xhr=new ActiveXObject("Microsoft.XMLHttp");
            }
            return xhr;
        }
    </script>
<?php
        echo '<option>吉林省</option>'.
             '<option>辽宁省</option>'.
             '<option>山东省</option>';
?>
<?php
        $provinceEle = $_POST['provinceEle'];
        switch ($provinceEle){
            case '吉林省':
                $cities='<option>请选择</option>'.
                '<option>长春市</option>'.
                '<option>通化市</option>'.
                '<option>白城市</option>'.
                '<option>辽源市</option>';
                break;
            case '辽宁省':
                $cities='<option>请选择</option>'.
                '<option>大连市</option>'.
                '<option>铁岭市</option>'.
                '<option>锦州市</option>'.
                '<option>丹东市</option>';
                break;
            case '山东省':
                $cities='<option>请选择</option>'.
                '<option>青岛市</option>'.
                '<option>威海市</option>'.
                '<option>日照市</option>'.
                '<option>德州市</option>';
                break;
        }
        echo $cities;
?>
```





AJAX DAY02:

* XML 格式
  * 基本内容
    * XML 被译为可扩展标记语言(标签)
    * XML 的标准是 W3C
    * XML 的语法类似于 HTML
      * HTML 的元素是预定义的
      * XML 的元素是自定义的
    * HTML、XHTML 和 XML 的区别
      * HTML - 超文本标记语言
        * HTML5 - 大融合(HTML 历代版本|XHTML|新元素)
      * XHTML - 严格版本的 HTML
        * 1.0 版本 - 目前还在使用的版本
      * 2.0 版本 - 几乎不使用
      * XML - 可扩展标记语言
    * XML 的用途
      * 配置文件 - 多用于服务器端
      * 数据格式 - 用于存储及传输数据内容
    * XML 的版本
      * 1.0 - XML 主流版本
      * 1.1 - 几乎不使用
  * 第一个 XML 文件
    * XML 文件的扩展名为".xml"
    * XML 声明
      <?xml version="1.0" encoding="UTF-8"?>
      * version - XML 的版本
      * encoding - XML 文件的编码
      * 注意
        * XML 的声明必须出现在 0 行 0 列
    * 根元素
      * 根元素必须是起始元素
      * 根元素只能一个(唯一)
    * 元素
      * 元素定义时,必须包含结束符"/"
    * 元素的属性
    * 元素的文本
  * JavaScript 解析 XML
    * DOM 的组成部分
      * DOM 核心
      * HTML DOM
      * XML DOM
    * 分类
      * 解析一个 XML 文件 - 不推荐
      * 解析一个 XML 字符串
    * 解析 XML 字符串

```
function getXML(xmlString){
    var xmlDoc = null;
  if(window.DOMParser){
    var parser = new DOMParser();
    xmlDoc = parser.parseFromString(xmlString,"text/xml");
  }else{
    xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
    xmlDoc.async = false;
    xmlDoc.loadXML(xmlString);
  }return xmlDoc;
  }
```

  * Ajax 中如何使用 XML 格式的数据
    * 客户端向服务器端发送的请求数据(XML 极少使用)
      * 客户端如何构建符合 XML 格式的数据
        * 定义一个符合 XML 格式的字符串(拼串)
      * 服务器端如何接收客户端的 XML 格式数据
        * 接收到的是符合 XML 格式的字符串
    * 利用 DOMDocument 对象的 loadXML()方法进行解析
    * 调用 DOMDocument、DOMElement 和 DOMNode 对象进行具体解析
    * 服务器端向客户端进行响应的数据
      * 服务器端如何构建符合 XML 格式的数据
      * 客户端如何接收服务器端的 XML 格式数据
* JSON 格式
  * 基本格式
    * 定义
      * JSON 的全称 JavaScript Object Notation
      * JSON 是一种轻量级的数据交换格式
    * 特点
      * 易于程序员阅读和编写
      * 易于计算机解析和生成
  * JSON 的结构
    * Array - 数组
    * Object - 对象
    * JSON 所支持的类型
* String * Number * Boolean * Object * Array * null
  * 建议
    * JSON 允许无限嵌套
    * 建议最多 3 层
  * Ajax 中的 JSON
    * 客户端向服务器端发送数据
      * 客户端构建符合 JSON 格式的字符串
      * 服务器端是以字符串类型进行接收
        * 再利用 json_decode()函数进行转换
    * 服务器端向客户端响应数据
      * 服务器端构建符合 JSON 格式的字符串
    * 将 PHP 中的 Object 或 Array 通过 json_encode()进行转换
      * 客户端通过 responseText 属性接收
        * 再利用 eval()函数进行转换
* Ajax 中的数据格式
  * 文本(HTML)格式
    * 优点 - 简单,数据量小
    * 缺点 - 涉及拼串和拆串的问题
  * XML 格式 - 传统软件

* 优点 - 结构化数据
* 缺点 - 数据量大,DOM 解析复杂

* JSON 格式 - 互联网
* 优点 - 轻量级(XML),数据量小,解析方便
* 缺点 - 嵌套过多影响性能

案例

```
<title>解析 XML</title>
<script>
    /*解析 XML 字符串 - 符合 XML 格式的字符串类型
        * 分类
 * 解析一个 XML 文件 - 使用 Ajax 的异步方式(设置为同步)
 * W3C 有关 XML 解析的最新规范 - 不允许读取 XML 文件
          * 解析一个 XML 字符串 */
    function getXml(xmlString){
        var xmlDoc = null;
        if(window.DOMParser){//其他浏览器
            // 创建解析器
            var parser = new DOMParser();
            // 通过解析器解析 XML
xmlDoc = parser.parseFromString(xmlString,"text/xml");
        }else{//IE 浏览器
xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
            xmlDoc.async = "false";//关闭异步加载
            xmlDoc.loadXML(xmlString);//读取 XML
        }return xmlDoc;
    }
    // 测试
var xml = "<province name='吉林省'><city>长春市</city><city>通化市</city><city>四平市</city><city>松原市</city><city>吉林市</city></province>";
    var xmlDoc = getXml(xml);
    console.log(xmlDoc);
    // DOM 解析 XML
var province = xmlDoc.getElementsByTagName("province")[0];
    var cities = province.getElementsByTagName("city");
    for(var i=0;i<cities.length;i++){
        console.log(cities[i].textContent);
    }  </script>
```

```
<title>DOM 解析 XML 文件</title>
<script>// 浏览器不允许读取本地的 XML 文件
    var xhr = null;
    if(window.XMLHttpRequest){
        xhr = new XMLHttpRequest();
    }else{
        xhr = new ActiveXObject("Microsoft.XMLHttp");
    }
    xhr.open("get","03_china.xml",false);
    xhr.send(null);
    var xmlDoc = xhr.responseXML;
console.log(xmlDoc);        </script>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<china>
    <province name="吉林省">
        <city>长春市</city>
        <city>通化市</city>
        <city>四平市</city>
        <city>松原市</city>
        <city>吉林市</city>
    </province>
    <province name="辽宁省">
        <city>沈阳市</city>
        <city>大连市</city>
        <city>铁岭市</city>
        <city>丹东市</city>
        <city>锦州市</city>
    </province>
    <province name="山东省">
        <city>济南市</city>
        <city>青岛市</city>
        <city>日照市</city>
        <city>威海市</city>
        <city>德州市</city>
    </province>
</china>
```

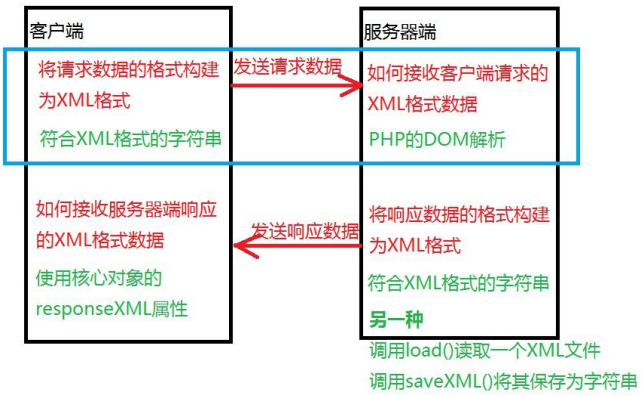
```
<title>Ajax 中的 XML 格式</title>
<input type="button" id="btn" value="异步交互">
<script>
    var btn = document.getElementById("btn");
    btn.onclick = function(){
        var xhr = getXhr();
        xhr.open("post","06.php");
    xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
        // 符合 XML 格式的字符串
var userXml =
 "<user><name>zhangwuji</name><pwd>123</pwd></user>";
        xhr.send("user="+userXml);
        xhr.onreadystatechange = function(){
            if(xhr.readyState==4&&xhr.status==200){
                var xmlDoc = xhr.responseXML;
                // 使用 DOM 直接解析
                var provinces =
xmlDoc.getElementsByTagName("province");
                for(var i=0;i<provinces.length;i++){
            console.log(provinces[i].getAttribute("name"));
                }
            }
        }
    }
```

```
        function getXhr(){
            var xhr = null;
            if(window.XMLHttpRequest){
                xhr = new XMLHttpRequest();
            }else{
                xhr = new ActiveXObject("Microsoft.XMLHttp");
            }return xhr;
        }        </script>
<?php
    // 服务器端接收客户端发送的 XML 格式数据
    $userXml = $_POST['user'];//符合 XML 格式的字符串
    //var_dump($userXml);
    /*在 PHP 如何解析 XML 字符串
     * 利用 DOMDocument 对象
       * $doc = new DOMDocument();
     * 调用 DOMDocument 对象的 loadXML(xmlString)方法
     * 利用 PHP 中的 DOM 解析 XML
    $doc = new DOMDocument();
    $doc->loadXML($userXml);
    //var_dump($doc);
    // <name>zhangwuji</name> - DOMElement 对象
    $name = $doc->getElementsByTagName('name')[0];
    //var_dump($name->firstChild->nodeValue);
    /**************************/
    header("Content-Type:text/xml;charset=utf-8");
    // 向客户端响应 XML 格式的数据
    $doc = new DOMDocument();
    $doc->load('03_china.xml');
    $xml = $doc->saveXML();
    // 符合 XML 格式的字符串
    //var_dump($xml);
    echo $xml;
    //echo '<province><city>长春市</city></province>';
?>
```

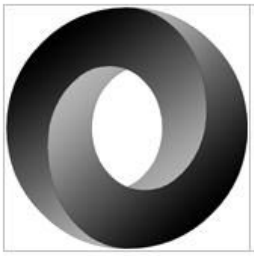
```
<title>用户登录功能</title>
<form id="login" action="07.php" method="post">
    用户名:<input type="text" id="user"><br>
    密码:<input type="text" id="pwd"><br>
    <input type="button" id="btn" value="登录">
</form>
<script>
    // 使用 Ajax 完成登录验证
    var btn = document.getElementById("btn");
    btn.onclick = function(){
        // 1. 获取必要的数据内容
        var form = document.getElementById("login");
        var url = form.action;
        var type = form.method;
        var user = document.getElementById("user").value;
        var pwd = document.getElementById("pwd").value;
        // 2. 实现 Ajax
        var xhr = getXhr();
        xhr.open(type,url);
    xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
        xhr.send("user="+user+"&pwd="+pwd);
        xhr.onreadystatechange = function(){
            if(xhr.readyState==4&&xhr.status==200){
                var data = xhr.responseText;
                console.log(data);
            }
        }
    }
    function getXhr(){
        var xhr = null;
        if(window.XMLHttpRequest){
            xhr = new XMLHttpRequest();
        }else{
            xhr = new ActiveXObject("Microsoft.XMLHttp");
        }return xhr;
    }        </script>
<?php
    $user = $_POST['user'];
    $pwd = $_POST['pwd'];
    $doc = new DOMDocument();
    $doc->load('users.xml');
    $users = $doc->getElementsByTagName('user');
    foreach($users as $userEle){
$name = $userEle->getElementsByTagName('name')[0];
        $nameValue = $name->firstChild->nodeValue;
$pwdEle = $userEle->getElementsByTagName('pwd')[0];
        $pwdValue = $pwdEle->firstChild->nodeValue;
        if($user==$nameValue&&$pwd==$pwdValue){
            echo 'success';
            return;//阻止后面的继续执行
        }
    }echo 'error';
?>
<?xml version="1.0" encoding="UTF-8"?>
<users>
    <user>
        <name>admin</name>
        <pwd>admin</pwd>
    </user>
    <user>
        <name>zhangwuji</name>
        <pwd>123</pwd>
    </user>
</users>
```

```
<title>Ajax 中的 JSON</title>
<input type="button" id="btn" value="异步请求">
<script>
    var btn = document.getElementById("btn");
    btn.onclick = function(){
        var xhr = getXhr();
        xhr.open("post","09.php");
    xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
        /*客户端向服务器端发送 JSON 格式数据
            问题 - PHP 的 json_decode()函数的问题
            * 最外层必须是单引号
            * 构建 Object 中的 key 必须使用双引号 */
var user = '{"username":"zhangwuji","password":123456}';
        xhr.send("user="+user);
        xhr.onreadystatechange = function(){
            if(xhr.readyState==4&&xhr.status==200){
                // 还是以文本格式接收
                var data = xhr.responseText;
                //console.log(data);
                /* 将符合 JSON 格式的字符串进行转换
                    * 不包裹"()",可能出现转换失败
                * 包裹了"()",eval()函数强制地进行转换
    注意
* 使用 eval()函数转换 JavaScript 代码还是 JSON 格式,包裹"()".*/
var json = eval("("+data+")");//将 JavaScript 字符串,转换为可执行的代码
                console.log(json);
            }
        }
    }
    function getXhr(){
        var xhr = null;
        if(window.XMLHttpRequest){
            xhr = new XMLHttpRequest();
        }else{
            xhr = new ActiveXObject("Microsoft.XMLHttp");
        }return xhr;
    }      </script>
<?php
    // 1. 是以字符串方式接收客户端发送的 JSON 格式数据
    $user = $_POST['user'];
    //"{username:'zhangwuji',password:123456}"
    //var_dump($user);
    /*2. 将符合 JSON 格式的字符串进行转换
        使用的是 json_decode()函数
        要求
        * 格式必须是'{"user":"zhangwuji","pwd":123456}'
        * 第二个参数表示转换为 Object 还是 Array
        * 默认值,false 表示 Object
        * true 表示 Array
    */
    $json = json_decode($user,true);
    //var_dump($json['username']);

    $userJSON = json_encode($json);
    echo $userJSON;
?>
```

在Ajax中如何使用XML格式数据



Ajax中的JSON格式数据



介绍 JSON

ECMA-404 The JSON Data Interchange Standard.

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。 易于人阅读和编写。同时也易于机器解析和生成。 它基于JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999的一个子集。 JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯（包括C, C++, C#, Java, JavaScript, Perl, Python等）。 这些特性使JSON成为理想的数据交换语言。

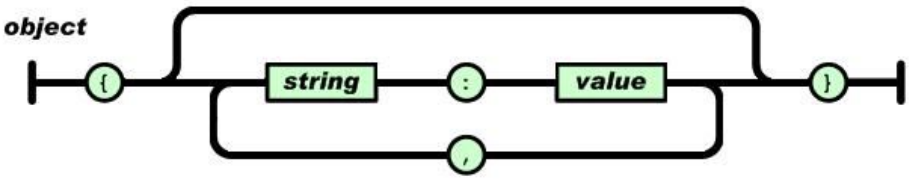
JSON建构于两种结构：

- “名称/值”对的集合（A collection of name/value pairs）。不同的语言中，它被理解为*对象（object）*，纪录（record），结构（struct），字典（dictionary），哈希表（hash table），有键列表（keyed list），或者关联数组 （associative array）。
- 值的有序列表（An ordered list of values）。在大部分语言中，它被理解为数组（array）。

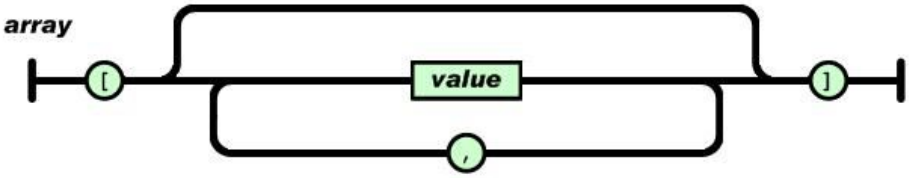
这些都是常见的数据结构。事实上大部分现代计算机语言都以某种形式支持它们。这使得一种数据格式在同样基于这些结构的编程语言之间交换成为可能。

JSON具有以下这些形式：

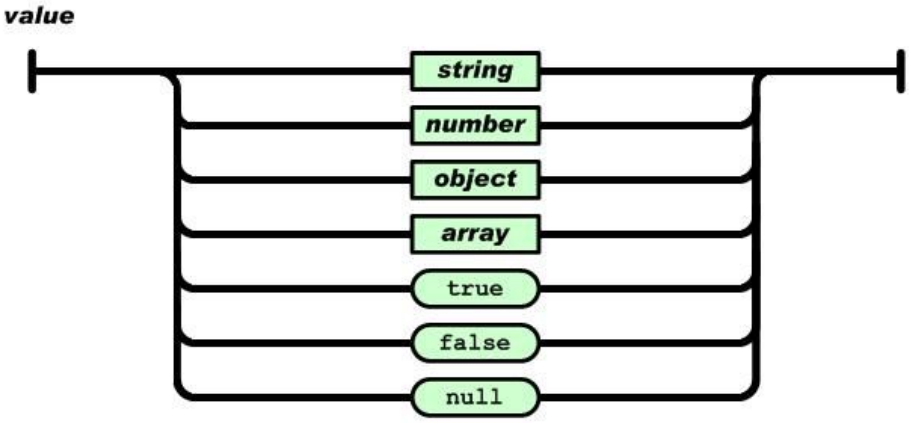
对象是一个无序的“‘名称/值’对”集合。一个对象以“{”（左括号）开始，“}”（右括号）结束。每个“名称”后跟一个“:”（冒号）；“‘名称/值’ 对”之间使用“,”（逗号）分隔。



数组是值（value）的有序集合。一个数组以“[”（左中括号）开始，“]”（右中括号）结束。值之间使用“,”（逗号）分隔。



值（*value*）可以是双引号括起来的字符串（*string*）、数值(number)、true、false、null、对象（object）或者数组（array）。这些结构可以嵌套。

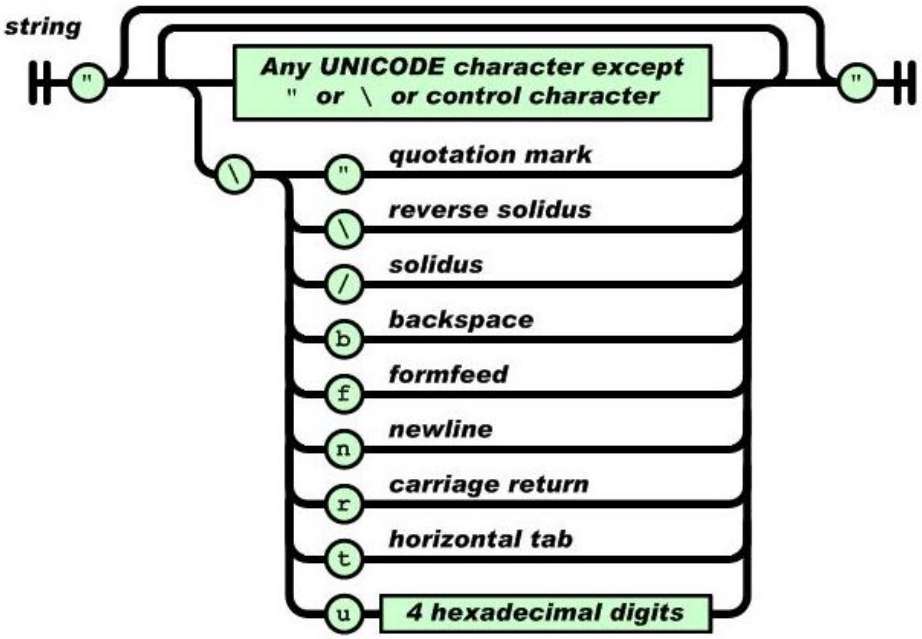


字符串（*string*）是由双引号包围的任意数量Unicode字符的集合，使用反斜线转义。一个字符（character）即一个单独的字符串（character string）。

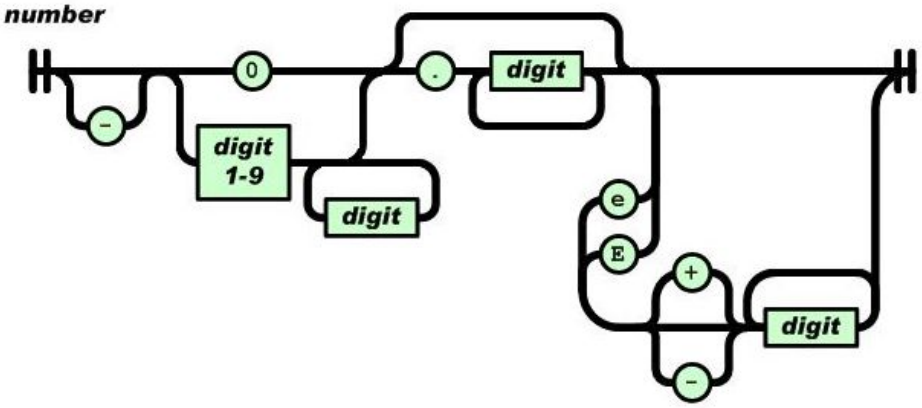
字符串（*string*）与C或者Java的字符串非常相似。

```
object
    {}
    { members }
members
    pair
    pair , members
pair
    string : value
array
    []
    [ elements ]
elements
    value
    value , elements
value
    string
    number
    object
    array
    true
    false
    null

---

string
    ""
    " chars "
chars
    char
    char chars
char
    any-Unicode-character-
        except-"-or-\-or-
        control-character
    \"
    \\
    \/
    \b
    \f
    \n
    \r
    \t
    \u four-hex-digits
number
    int
    int frac
    int exp
    int frac exp
int
    digit
    digit1-9 digits
    - digit
    - digit1-9 digits
frac
    . digits
exp
    e digits
digits
    digit
    digit digits
e
    e
```

e+
e-
E
E+
E-



数值（*number*）也与C或者Java的数值非常相似。除去未曾使用的八进制与十六进制格式。除去一些编码细节。



空白可以加入到任何符号之间。 以下描述了完整的语言。

来源： <http://www.json.org/json-zh.html>

Day03

AJAX DAY03:

* jQuery 中的 Ajax
  * Ajax 异步交互
    * $.ajax() - 最接近原生 Ajax 的用法,最复杂
      * url - 请求地址
      * type - 请求类型,默认为 GET
      * async - 是否异步,默认为 true
      * data - 请求数据,必须为 key:value
      * error - 请求失败后的回调函数
        function(XMLHttpRequest,textStatus,errorThrown){}
        * XMLHttpRequest - Ajax 核心对象
        * textStatus - 请求的状态
        * errorThrown - 错误异常
      * success - 请求成功后的回调函数
        function(data,textStatus){}
        * data - 服务器端响应的数据
        * textStatus - 服务器端的状态
      * dataType - 设置服务器端响应数据的格式
    * "text" - 文本格式,默认值
    * "xml" - XML 格式
    * "json" - JSON 格式
    * $().load(url,data,callback) - 最简单,局限性最大
      * 不能自定义请求类型
        * 不发送请求数据,GET
      * 发送请求数据,POST
      * 必须与 HTML 页面元素相关
      * 客户端与服务器端之间交互只能使用文本格式
    * $.get(url,data,callback,type) - 请求类型是 GET
      * 文本格式 - 默认
      * XML 格式
        * 服务器端
        * header("Content-Type:text/xml;charset=UTF-8");
        * 响应符合 XML 格式的字符串
      * 客户端 - 将 type 参数的值设置为"xml"
      * JSON 格式
        * 服务器端
        * 使用 json_encode()函数构建 JSON
        * 构建符合 JSON 格式的字符串
          * key 和 value 必须添加双引号
      * 客户端 - 将 type 参数的值设置为"json"
    * $.post(url,data,callback,type) - 请求类型是 POST
      * 文本格式 - 默认
      * XML 格式
        * 服务器端
        * header("Content-Type:text/xml;charset=UTF-8");
        * 响应符合 XML 格式的字符串
      * 客户端 - 将 type 参数的值设置为"xml"
      * JSON 格式
        * 服务器端
        * 使用 json_encode()函数构建 JSON
        * 构建符合 JSON 格式的字符串
          * key 和 value 必须添加双引号
      * 客户端 - 将 type 参数的值设置为"json"
    * $.getScript() - 动态获取脚本
    * $.getJSON() - 返回数据格式必须是 JSON 格式
  * 表单异步提交
    * 实现表单异步提交步骤
      * 获取表单元素的值
      * 实现 Ajax 的异步提交
      * 阻止表单默认提交
    * 阻止表单提交(默认行为)
      * 将 submit 按钮修改为 button,并绑定 click 事件
      * 表单绑定 submit 事件,在事件的处理函数中:
        * return false;
      * 事件对象 event.preventDefault();
    * 表单的序列化(jQuery)
      * 基本内容
        * 表单异步提交时,手动获取表单内所有元素的值
      * 只关注表单,而不关注表单内的元素
      * 方法
        * serialize()
      * serializeArray()
      * 注意
        * 必须要为表单元素定义 name 属性值
  * 跨域请求
    * 基本概念
      * 域 - IP:端口号
      * 域名 - 虚拟地址
      * 跨域
        * 完全跨域 - IP 不同
        * 跨子域 - IP 相同,端口号不同
      * 万维网(www)
        * 默认协议中,是不允许跨域请求的(同源策略)
        * 跨域请求中,默认情况下不允许响应
    * 实现跨域
      * JSONP(JSON With Padding)被称之为 JSON 的使用模式
      * 作用 - 可用于解决主流浏览器的跨域数据访问的问题
        * 描述
          * HTML 的<script>元素是一个例外
          * <script>元素不受同源策略的约束
          * <script>元素具有开放策略
          * 得到不同域动态产生的 JSON 格式数据
      * 实现方式
        * $.getJSON()方法
        * $.ajax()方法

```
<title>jQuery 的 load()方法</title>
 <body>
  <input type="button" id="btn" value="load 方法">
  <script>
      $("#btn").click(function(){
      /* jQuery 中的 load()方法
       $().load(url,data,callback)
       * url - 请求地址
       * data - 请求数据,格式必须为 key:value
       * callback - 回调函数
      function(responseText,textStatus,XMLHttpRequest){}
      * responseText - 服务器端响应的数据
      * textStatus - 服务器端的状态
       * error - 表示请求错误
       * success - 表示请求成功
       * timeout - 表示请求超时
       * notmodify - 表示服务器端资源未修改
       * XMLHttpRequest - Ajax 的核心对象
问题
       *  不能自定义请求类型
       *  没有发送请求数据时,load()方法的请求类型为 GET
       *  如果发送请求数据时,load()方法的请求类型为 POST
       *  必须与页面元素相关
            */
           var data = {
                user : "zhangwuji",
                pwd : 123456
           };
      $(this).load("01.php",data,function(responseText,textStatus){
                console.log(textStatus);
           });
      });
  </script>
<?php
      // 1. 接收客户端的请求数据
      $user = $_REQUEST['user'];
      $pwd = $_REQUEST['pwd'];
      // 2. 向客户端响应数据
      echo "$user : $pwd";
?>
<title>jQuery 中的 get 或 post 方法</title>
 <body>
  <input type="button" id="btn" value="get 或 post">
  <script>
      $("#btn").click(function(){
           /*jQuery 中的$.get()方法
               $.get(url,data,callback,type)
               * url - 请求地址
               * data - 请求数据,格式必须为 key:value
               * callback - 回调函数
               * type - 设置响应数据格式
                *  默认为文本格式
                 * xml 格式
                 * json 格式
            */
           var data = {
                user : "zhangwuji",
                pwd : 12345
           }
           /*
           $.get("02.php",data,function(responseText){
                console.log(responseText);
           },"json");
           */
           $.post("02.php",data,function(responseText){
                console.log(responseText);
           },"json");
      });
  </script>
<?php
      $user = $_GET['user'];
      $pwd = $_GET['pwd'];
      // 响应文本格式
      //echo "$user : $pwd";
      /* 响应 XML 格式
      header('Content-Type:text/xml;charset=UTF-8');
echo "<user><name>$user</name><pwd>$pwd</pwd></user>";
      /* 响应 JSON 格式
          由于 jQuery 中的 get()方法的问题
          * 服务器端手工构建符合 JSON 格式的字符串
           * 客户端识别为符合 JSON 格式的字符串,并不是 JSON
          * 服务器端使用 json_encode()函数进行转换*/
      $arr = array(
           'user'=>$user,
           'pwd'=>$pwd
      );
      //echo json_encode($arr);
      echo '{"user":"zhangwuji","pwd":"12345"}';
?>
<title>jQuery 中的 ajax 方法</title>
 <body>
  <input type="button" id="btn" value="ajax 方法">
  <script>
      $("#btn").click(function(){
           /* jQuery 中的$.ajax()方法
               $.ajax(options)
               * options 的格式为 {key:value}
               *  选项
                * url - 请求地址
                 * type - 请求类型,默认为 GET
```

```
                * async - 是否异步,默认为 true
                * data - 请求数据,必须为 key:value
                * error - 请求失败后的回调函数
    function(XMLHttpRequest,textStatus,errorThrown){}
                  * XMLHttpRequest - Ajax 核心对象
                  * textStatus - 请求的状态
                  * errorThrown - 错误异常
                * success - 请求成功后的回调函数
                  function(data,textStatus){}
                  * data - 服务器端响应的数据
                  * textStatus - 服务器端的状态
                * dataType - 设置服务器端响应数据的格式
                  * "text" - 文本格式,默认值
                  * "xml" - XML 格式
                  * "json" - JSON 格式*/
          var json = {
              user : "zhangwuji",
              pwd : 12345
          }
          $.ajax({
              url : "03.php",
              type : "get",
              data : json,
              error :
function(XMLHttpRequest,textStatus,errorThrown){
                  console.log(textStatus);
              },
              success : function(data,textStatus){
                  console.log(data);
              },
              datatype : "json"
          });
    });
  </script>
<?php
    // 1. 接收请求
    $user = $_REQUEST['user'];
    $pwd = $_REQUEST['pwd'];
    // 2. 响应数据
    //echo "$user : $pwd";
    //header('Content-Type:text/xml;charset=UTF-8');
    //echo
"<user><name>$user</name><pwd>$pwd</pwd></user>";
    echo '{"user":"zhangwuji","pwd":"12345"}';
?>
<title>jQuery 中的 getScript 方法</title>
 <body>
  <input type="button" id="btn" value="getScript">
  <div id="show"></div>
  <script>
    $("#btn").click(function(){
        /*jQuery 中的$.getScript()方法
            $.getScript(url,callback)
            * url - 请求地址(JS 文件)
            * callback - 回调函数 */
        //$.getScript("test.js");
        $.getScript("04.php");
    });
  </script>
<?php
    echo 'var comments = $("<div><h2> 张 三 </h2><p> 太 慢
了...</p></div>");$("#show").append(comments);';
?>
 <title>表单的异步提交</title>
 <body>
  <form name="login">
    用户名:<input type="text" id="user"><br>
    密码:<input type="text" id="pwd"><br>
    <input type="submit" value="登录">
  </form>
  <script>
    /*实现表单的异步提交
        * 将表单的 submit 按钮,修改为 button 按钮,并绑定
onclick 事件
        * 将表单绑定 onsubmit 事件,在事件的处理函数中
return false(表示阻止表单提交)
        * 获取表单,通过 JavaScript 代码绑定 onsubmit 事件
         在事件的处理函数中 return false(表示阻止表单提交)
        获取表单的几种方式
        * getElementById()通过表单 id 获取
        * getElementsByTagName()通过表单元素名获取
        * document.forms[索引值]
        * document.表单名称*/
    //$("#login").submit(function(){return false});
    document.login.onsubmit = function(){
        // 1. 获取表单元素的值
        var user = document.getElementById("user").value;
        var pwd = document.getElementById("pwd").value;
        // 2. 实现 Ajax 的异步提交
        var xhr = getXhr();
        xhr.open("post","05.php");
    xhr.setRequestHeader("Content-Type","application/x-www-f
orm-urlencoded");
        xhr.send("user="+user+"&pwd="+pwd);
        xhr.onreadystatechange = function(){
            if(xhr.readyState==4&&xhr.status==200){
            }
        }
        // 3. 阻止表单默认提交
        return false;
```

```
    }
    function getXhr(){
        var xhr = null;
        if(window.XMLHttpRequest){
            xhr = new XMLHttpRequest();
        }else{
            xhr = new ActiveXObject("Microsoft.XMLHttp");
        }
        return xhr;
    }
</script>
<?php
    //1 获取请求数据
    $user = $_POST['user'];
    $pwd = $_POST['pwd'];
    //2 响应数据
    echo "$user : $pwd";
?>
<title>jQuery 实现表单异步提交</title>
<form>
    用户名:<input type="text" id="user"><br>
    密码:<input type="text" id="pwd"><br>
    <input type="submit" value="登录">
</form>
<script>
    $("form").submit(function(event){
        // 1. 获取表单元素的值
        var data = {
            user : $("#user").val(),
            pwd : $("#pwd").val()
        }
        // 2. 实现 Ajax 的异步提交
        $.post("05.php",data,function(responseText){
            console.log(responseText);
        });
        // 3. 阻止表单默认提交
        event.preventDefault();
    });
<title>jQuery 的表单序列化</title>
<form id="login">
    <input type="text" id="ele1" name="user"><br>
    <input type="text" id="ele2" name="pwd"><br>
    <input type="text" id="ele3" name="ele3"><br>
    <input type="text" id="ele4" name="ele4"><br>
    <input type="text" id="ele5" name="ele5"><br>
    <input type="text" id="ele6" name="ele6"><br>
    <input type="text" id="ele7" name="ele7"><br>
    <input type="text" id="ele8" name="ele8"><br>
    <input type="text" id="ele9" name="ele9"><br>
    <input type="text" id="ele10" name="ele10"><br>
    <input type="submit">
</form>
<script>
    /* 需求 - 实现表单异步提交 */
    $("form").submit(function(){
        /* 1. 获取表单元素
        var formEles = {
            ele1 : $("#ele1").val(),
            ele2 : $("#ele2").val(),
            ele3 : $("#ele3").val(),
            ele4 : $("#ele4").val(),
            ele5 : $("#ele5").val(),
            ele6 : $("#ele6").val(),
            ele7 : $("#ele7").val(),
            ele8 : $("#ele8").val(),
            ele9 : $("#ele9").val(),
            ele10 : $("#ele10").val()
        };*/
        /*
            表单的序列化
            * 特点
                * 不再关注表单元素,只关注表单(只有一个)
                * 表单元素必须定义 name 属性值
            * 表单的序列化
                * serialize()
                    * key=value&key=value&key=value&...
                * serializeArray() - JSON 对象
                    * [{key:value},{key:value},{key:value}]
        */
        //var formEles = $("#login").serialize();
        var formEles = $("form").serialize();
        console.log(formEles);
        // 2. Ajax 异步提交
        $.post("05.php",formEles,function(data){
            console.log(data);
        });
        // 3. 阻止表单默认提交
        return false;
    });
    var json = [
        {name: 'firstname', value: 'Hello'},
        {name: 'lastname', value: 'World'},
        {name: 'alias'}, // this one was empty
    ]
var str = "[{name:'firstname',value:'Hello'},
        {name: 'lastname',value:'World'},
        {name:'alias'}]"
</script>
```

```
<title>jQuery 中的 getJSON 方法</title>
    <input type="button" id="btn" value="getJSON">
    <script>
        $("#btn").click(function(){
            /*$.getJSON(url,callback)方法*/
            $.getJSON("data.json",function(data){
                console.log(data);
            });
        });
    </script>
```

Json 文件:

```
[
    {
        "user" : "zhangwuji",
        "pwd" : 12345
    },
    {
        "user" : "zhouzhiruo",
        "pwd" : 12345
    }
]
```

```
<title>使用 getJSON 实现跨域</title>
    <input type="button" id="btn" value="跨域请求">
    <script>
        $("#btn").click(function(){
            /*使用 getJSON()实现跨域请求
                $.getJSON(url,[data],[callback])
                * url - 请求地址
                * data - 请求数据
                * callback- 回调函数
                * 注意
                  * 该方法的请求类型是 GET */
        $.getJSON("09.php?callback=?",function(responseText){
                /*服务器端向客户端响应以下内容
                  callback({"user":"zhangwuji","pwd":"12345"})
                  * 实际是一个函数的调用体,并传递了实参
                  * 如果当前 JavaScript 代码,具有这样一个函数的定义体的话 - 可以接受实参*/
                console.log(responseText);
            });
        }); </script>
<?php
        // 接收客户端发送的请求数据
        $callback = $_GET['callback'];
        // 测试
        //var_dump($callback);
        /*如果将 callback 当做一个函数来看
            通过这种方式 - $callback(data)
            调用该函数并传递实参
         */
        $data = '{"user":"zhangwuji","pwd":"12345"}';

        echo "$callback($data)";
?>
```

$.getScript()方法的作用