AJAX学习笔记

什么时候该使用 AJAX

第一、请求的提交是为了页面数据的显示，这时候用户一般不希望看到页面的刷新，是使用 AJAX 的一个最佳时候。

第二、如果请求提交后，用户能从页面感觉到提交结果，这时候，也最好不要有页面刷新，推荐使用 AJAX 技术。

第三、如果请求提交后，用户不能从页面感觉到提交动作，如绝大多数时候的数据的增加和修改，这时候则需要页面刷新，不能使用 AJAX 技术。

第四、复杂的 UI，以前对于复杂的 C/S 模式的 UI，B/S 模式一向采取逃避的方法，现在则可以放心大胆的使用 AJAX 来加以解决。

常见问题解决：

1、 **第一、输入值校验的问题** 申请用户的时候检查用户名是否重复，用 AJAX 访问后台，既不需要刷新页面，也没有过多的 JS 代码

2、**第二、级联显示的问题** 访问后台吧，页面需要刷新；JS 代码量大，影响内存，数据不安全；所以常级联选择框，级联菜单，导航树等

3、**第三、请求结果只改变部分页面** 如，论坛的回复帖子和帖子列表在一个页面上的时候。这两个 UI 在一个页面上，用户体验比回复帖子在另外一个页面好。但回复后要对整个页面进行刷新，这种感觉就不好了。你看，那么大一个帖子列表，只增加你的一个回复，却要对

整个页面进行刷新，不管从哪个角度来看都不好。

4、**第四、由于技术原因而使用 iframe 的问题** 避免 iframe 的嵌套引入的技术难题

5、**第五、数据录入和列表显示在同一个页面** C/S 模式的 UI 中常常有数据录入和数据列表显示在同一个界面上，这样对于用户来说有很好的用户体验，用户录入的结果马上就能在同一界面显示。但是在 B/S 的 UI 上，由于需要提交刷新的问题，我们经常把数据的录入和数据显示分别放在两个不同的页面上。很显然，这样的用户体验肯定没有 C/S 模式来得好。像这样的 问题还有很多，在 B/S 模式下，都因为技术的原因而选择其他的解决办法。现在我们可以自豪的使用 AJAX 来宣告可以做出和 C/S 模式一样复杂的 UI 了

6、**第六、翻页问题** 不需要刷新的翻页

服务器

Ajax：用于 web 和服务器交换数据

服务器：文件服务器、邮件服务器、web 服务器

概述

- AJAX：Asynchronous JavaScript and XML 异步的 JS 和 xml
- 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，使网页实现异步更新，

在不重新加载页面的情况下就实现对网页的更新。

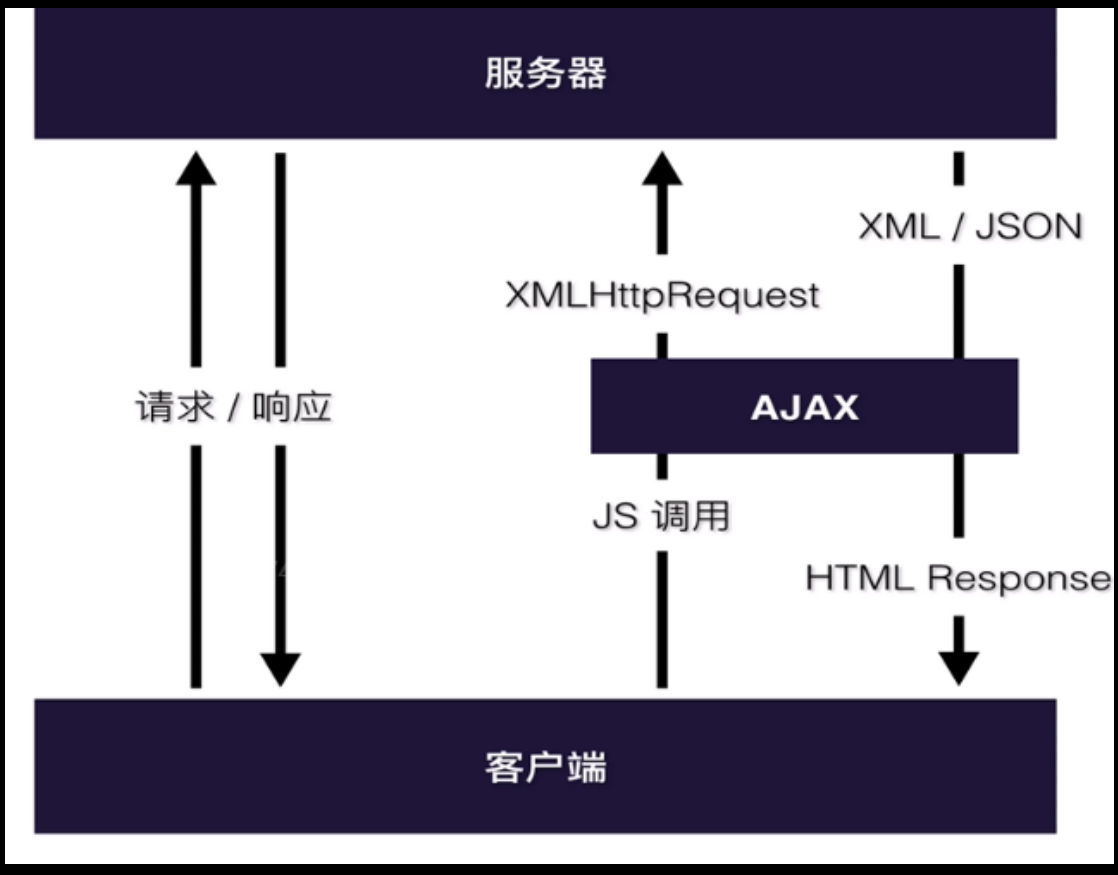
√ Asynchronous JavaScript & XML

√ web开发的一种技术

√ 异步发送 & 请求数据

√ 不需要重新刷新当前页面

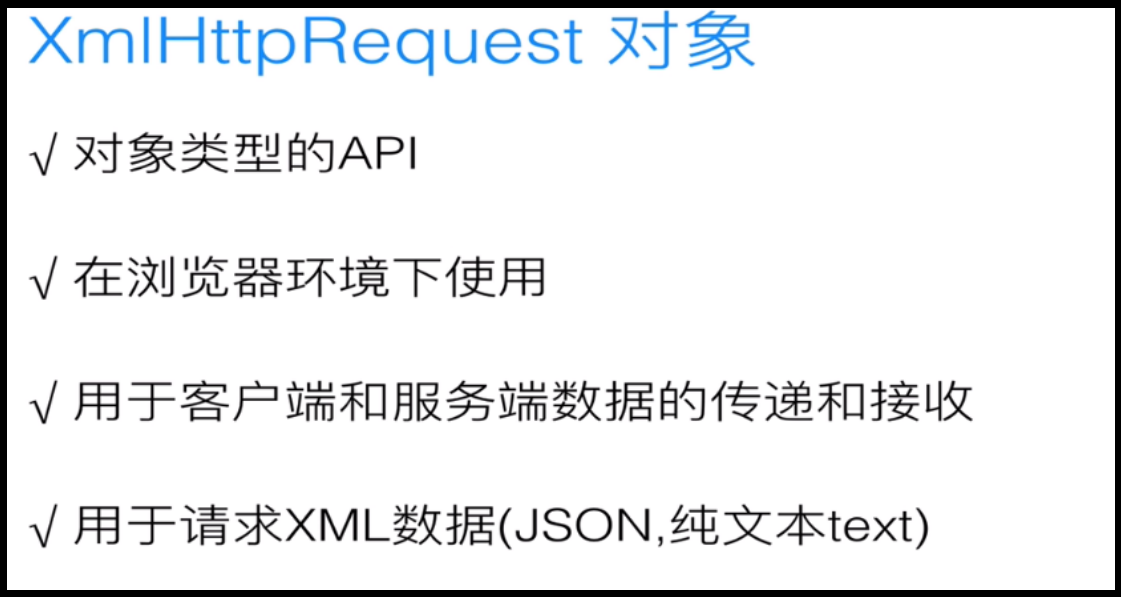
√ 目前JSON数据格式已经占据市场



XmlHttpRequast 对象

XmlHttpRequast 是 Ajax 的基础

所有浏览器都支持，用于在后台和服务器交换数据



创建：

```
var req = new XMLHttpRequest();
```

老版本（IE56）：

```
variable=new ActiveXObject("Microsoft.XMLHTTP");
```

所以为了保证绝对的兼容性：

```
var req ;
if (window.XMLHttpRequest){
    req =  new XMLHttpRequest();
}
else {
    req = new ActiveXObject("Microsoft.XMLHTTP");
}
```

使用 JQ 就不用考虑。

发送请求

1. Req.open(method, url, async)

a) Method: get or post

b) url: 文件在服务器上的位置

c) async: true 异步　false 同步

一般使用 true

2. req.send(string)

a) string: 仅用于 post

```
req.open("GET","www.baidu.com",true);
req.send();
```

可以添加附带信息：

```
req.open("GET","www.baidu.com?name=Edwin&age=21",true);
req.send();
```

GET VS POST

Get 更加简单，更快

以下情况使用 post：

1. 无法使用缓存文件（更新服务器上的文件或数据库）
2. 向服务器发送大量数据（post 没有数据量限制）
3. 发送包含未知字符的用户输入，post 更稳定、安全。

添加 HTTP 请求头

HTTP 请求头：有很多参数，可以赋值传递。

Req.setRequestHeader(header, value)

1. Header: 固定的请求头
2. Value:　请求头值

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

GET 请求

```
<form action="http://www.baidu.com" method="get">
    <input type="text" name="userName"><br>
    <input type="password" name="userPwd"><br>
    <input type="submit" value="提交"><br>
</form>
```

状态值 状态码

- 状态值：readyState

  运行 AJAX 所经历过的几种状态，无论访问是否成功都将响应的步骤，可以理解成为 AJAX 运行步骤。如：正在发送，正在响应等，由 AJAX 对象与服务器交互时所得；使用“ajax.readyState”获得。（由数字 1~4 单位数字组成）

  - 0 - (未初始化)还没有调用 send()方法，定义后自动具有。
  - 1 - (载入)已调用 send()方法，正在发送请求
  - 2 - (载入完成)send()方法执行完成，
  - 3 - (交互)正在解析响应内容
  - 4 - (完成)响应内容解析完成，可以在客户端调用了

- 状态码：status

  无论 AJAX 访问是否成功，由 HTTP 协议根据所提交的信息，服务器所返回的 HTTP 头信息代码，该信息使用“ajax.status”所获得；（由数字 1XX,2XX 三位数字组成）

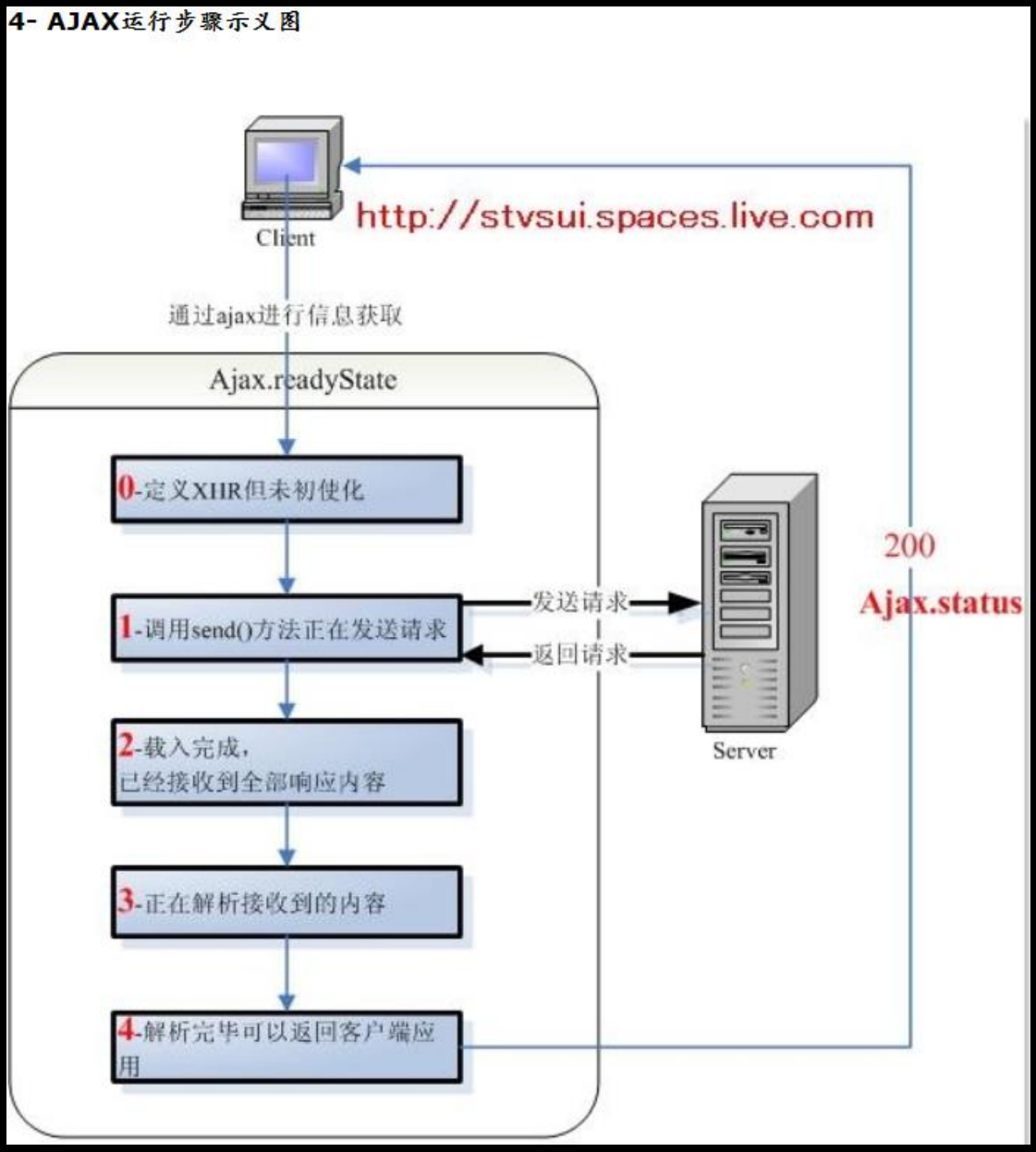
- 1**：请求收到，继续处理
- 2**：操作成功收到，分析、接受
- 3**：完成此请求必须进一步处理
- 4**：请求包含一个错误语法或不能完成
- 5**：服务器执行一个完全有效请求失败

当 readyState 等于 4 且状态为 200 时，表示响应已就绪：

100——客户必须继续发出请求
101——客户要求服务器根据请求转换HTTP协议版本
200——交易成功
201——提示知道新文件的URL
202——接受和处理、但处理未完成
203——返回信息不确定或不完整
204——请求收到，但返回信息为空
205——服务器完成了请求，用户代理必须复位当前已经浏览过的文件
206——服务器已经完成了部分用户的GET请求
300——请求的资源可在多处得到
301——删除请求数据
302——在其他地址发现了请求数据
303——建议客户访问其他URL或访问方式
304——客户端已经执行了GET，但文件未变化
305——请求的资源必须从服务器指定的地址得到
306——前一版本HTTP中使用的代码，现行版本中不再使用
307——申明请求的资源临时性删除
400——错误请求，如语法错误
401——请求授权失败
402——保留有效ChargeTo头响应
403——请求不允许
404——没有发现文件、查询或URl
405——用户在Request-Line字段定义的方法不允许
406——根据用户发送的Accept拖，请求资源不可访问
407——类似401，用户必须首先在代理服务器上得到授权
408——客户端没有在用户指定的饿时间内完成请求
409——对当前资源状态，请求不能完成
410——服务器上不再有此资源且无进一步的参考地址
411——服务器拒绝用户定义的Content-Length属性请求
412——一个或多个请求头字段在当前请求中错误
413——请求的资源大于服务器允许的大小

413——请求的资源大于服务器允许的大小
414——请求的资源URL长于服务器允许的长度
415——请求资源不支持请求项目格式
416——请求中包含Range请求头字段，在当前请求资源范围内没有range指示值，请求也不包含If-Range请求头字段
417——服务器不满足请求Expect头字段指定的期望值，如果是代理服务器，可能是下一级服务器不能满足请求
500——服务器产生内部错误
501——服务器不支持请求的函数
502——服务器暂时不可用，有时是为了防止发生系统过载
503——服务器过载或暂停维修
504——关口过载，服务器使用另一个关口或服务来响应用户，等待时间设定值较长
505——服务器不支持或拒绝支请求头中指定的HTTP版本

Ajax 运行过程



服务器响应

Req.responseText

responseText: 获取字符串形式的响应数据

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，请使用 responseXML 属性

onreadystatechange 事件

每当 readyState 改变时，就会触发 onreadystatechange 事件

```
req.onreadystatechange = function () {
    if (req.readyState == 4 && req.status == 200)
        alert(req.responseText);
    }
}
```

当您使用 async=false 时，请不要编写 onreadystatechange 函数

？？？

尚硅谷-佟刚-Ajax 视频

Ajax-原来是荷兰一只球队

也是泰坦尼克号的一艘姊妹舰

属性	描述
onreadystatechange	每个状态改变是都会触发这个事件处理器，通常会调用一个javaScript函数
readyState	请求的状态，有5个可取值：0=未初始化、1=正在加载、2=已经加载、3=交互中、4=完成。
responseText	服务器的响应，表示为一个串。
responseXML	服务器的响应，表示为XML。这个对象可以解析为DOM对象。
status	服务器的HTTP状态码（200对应OK、404对应NotFount、等）
statusText	HTTP状态码的相应文本（OK或NotFount等）

HelloWorld

```
<script>
    window.onload = function(){
        //1.获取节点，添加响应函数
        document.getElementsByTagName("a")[0].onclick = function(){
            //3、创建一个xmlHttpRequest对象
            var req = new XMLHttpRequest();
            //4.准备发送请求的数据：URL
            var url = this.href;
            var method = "GET";
            //5、调用open
            req.open(method,url);
            //6、send
            req.send();
            //7.添加onreadystatechange响应函数
            req.onreadystatechange = function(){
                //8.判断响应是否完成：readyState==4
                if(req.readyState==4){
                    //9、判断响应是否可用：status==200
                    if(req.status==200||req.status==304){
                        //10.打印响应结果。
                        alert(req.responseText);
                    }
                }
            }
            // 2.取消默认跳转行为
            return false;
        }

    }
</script>
</head>
<body>
    <a href = "temp.txt">点我吧！</a>
```

这里返回的是temp.txt文件中的内容。

Onreadystatechange

onreadystatechange:

- 该事件处理函数由服务器触发，而不是用户
- 在 Ajax 执行过程中，服务器会通知客户端当前的通信状态。这依靠更新 XMLHttpRequest 对象的 readyState 来实现。改变 readyState 属性是服务器对客户端连接操作的一种方式。每次 readyState 属性的改变都会触发 readystatechange 事件

Open

open(method, url, asynch)

- XMLHttpRequest 对象的 **open 方法允许程序员用一个Ajax调用向服务器发送请求。**
- method：**请求类型**，类似“GET”或”POST”的字符串。若只想从服务器检索一个文件，而不需要发送任何数据，使用GET(可以在GET请求里通过附加在URL上的查询字符串来发送数据，不过数据大小限制为2000个字符)。若需要向服务器发送数据，用POST。
- 在某些情况下，有些浏览器会把多个XMLHttpRequest请求的结果缓存在同一个URL。如果对每个请求的响应不同，就会带来不好的结果。**在此将时间戳追加到URL的最后，就能确保URL的唯一性，从而避免浏览器缓存结果。**
- url：路径字符串，指向你所请求的服务器上的那个文件。可以是绝对路径或相对路径。
- asynch：表示请求是否要异步传输，默认值为true。指定true，在读取后面的脚本之前，不需要等待服务器的相应。指定false，当脚本处理过程经过这点时，会停下来，一直等到Ajax请求执行完毕再继续执行。

加时间

```
var url = this.href + "?time=" + new Date();
```

加时间来唯一性 URL

SetRequestHeader

- setRequestHeader(header,value)
  - 当浏览器向服务器请求页面时，它会伴随这个请求发送一组首部信息。这些首部信息是一系列描述请求的元数据(metadata)。**首部信息用来声明一个请求是GET还是POST。**
  - Ajax 请求中，发送首部信息的工作可以由setRequestHeader该完成
  - 参数header： 首部的名字; 参数value：首部的值。
  - **如果用POST请求向服务器发送数据，需要将“Content-type”的首部设置为“application/x-www-form-urlencoded”.它会告知服务器正在发送数据，并且数据已经符合URL编码了。**
  - **该方法必须在open()之后才能调用**
  - 完整的 Ajax 的 POST 请求示例：

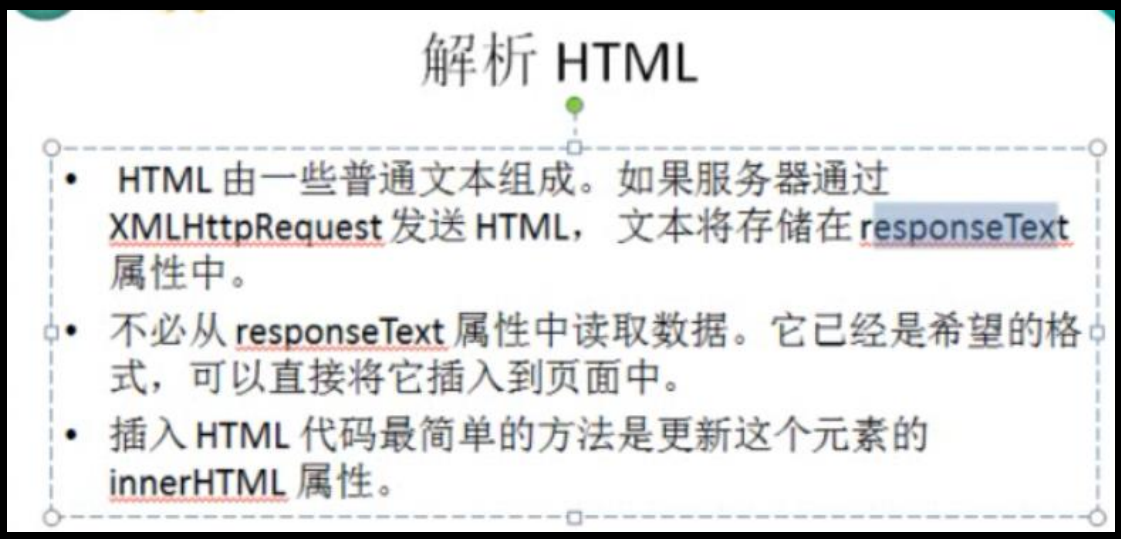
```
var url = "../jsp/forumServlet";
var nameValue = trim(document.forumAddForm.name.value);
xhr.open("POST",url);
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
xhr.send("method=name_isExist" + "&name=" + nameValue);
```

数据格式提要

有三种数据格式

1. Xml
2. Json 主流
3. HTML

HTML




```
request.onreadystatechange = function(){
    if(request.readyState == 4){
        if(request.status == 200 || request.status == 304){
            document.getElementById("details").innerHTML = request.responseText;
        }
    }
}
```

使用HTML数据格式，responseText返回的就是HTML格式，直接插入

XML 格式

需要使用 responseXML 获取

```
var result = request.responseXML;

//2. 结果不能直接使用，必须先创建对应的节点，再把节点加入到 #details 中
//目标格式为：
/*
<h2><a href="mailto:andy@clearleft.com">Andy Budd</a></h2>
<a href="http://andybudd.com/">http://andybudd.com/</a>
*/
var name = result.getElementsByTagName("name")[0].firstChild.nodeValue;
var website = result.getElementsByTagName("website")[0].firstChild.nodeV
var email = result.getElementsByTagName("email")[0].firstChild.nodeValue

//alert(name);
//alert(website);
//alert(email);
```

Json 格式

Json：JavaScript object Notation，一种简单的数据格式，比 xml 轻巧。

Json 是 JavaScript 的原生格式，因此 js 在处理时不需要任何的特殊 api 或工具

规则：

对象是一个无序的“名称/值”对象集合。

一个对象： {name : value …}

可以递归，json 的值可以是 json 对象

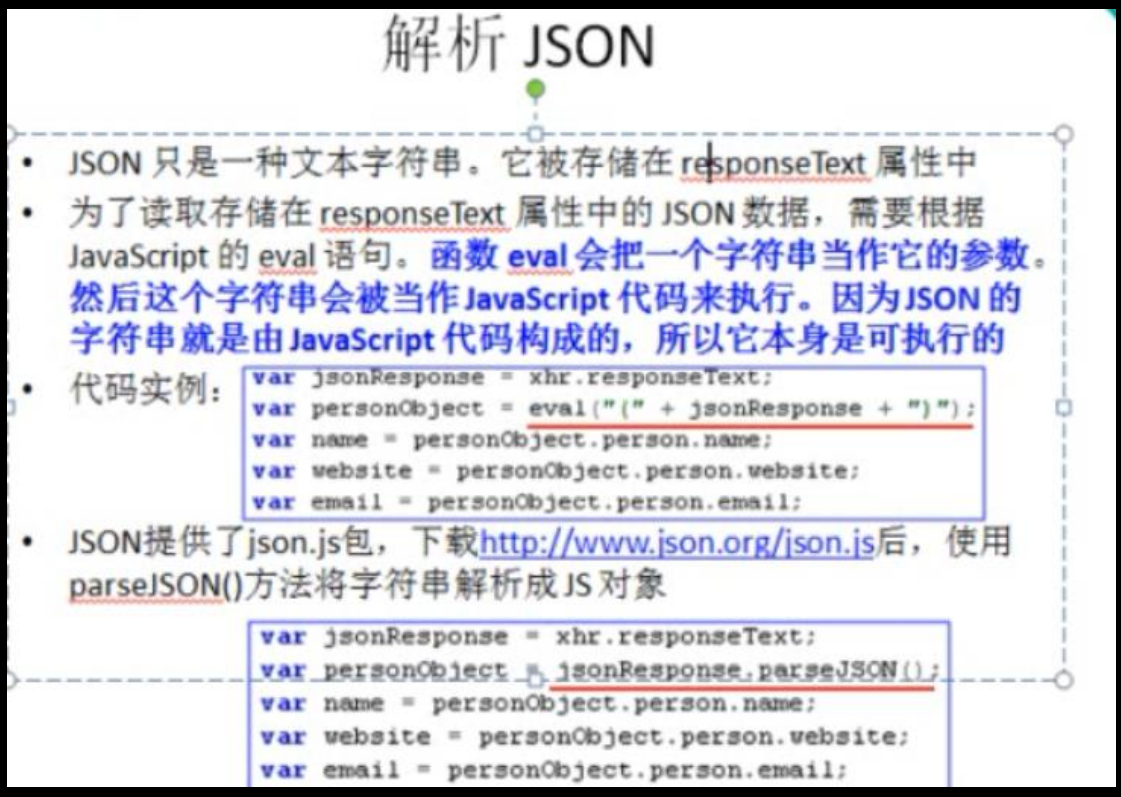
值还可以是方法，强

```
var jsonObject = {
        "name":"atguigu",
        "age":12,
        "address":{"city":"BeiJing", "school":"尚硅谷"},
        "teaching":function(){
            alert("JavaEE, Android...");
        }
};
```

Json 文件：

```
{"person": {
  "name":"Andy Budd",
  "website":"http://andybudd.com/",
  "email":"andy@clearleft.com"
  }
}
```

解析：



```
var jsonStr = "{'name':'atguigu'}";

//把一个字符串转为 JSON 对象!
//alert(jsonStr.name);

//使用 eval() 方法
var testStr = "alert('hello eval')";
//alert(testStr);

//eval 可以把一个字符串转为本地的 JS 代码来执行
eval(testStr);

//把 JSON 字符串转为 JSON 对象.
var testObject = eval("(" + jsonStr + ")");
alert(testObject.name);
```

- 优点：
  - 作为一种数据传输格式，JSON 与 XML 很相似，但是它更加灵巧。
  - JSON 不需要从服务器端发送含有特定内容类型的首部信息。
- 缺点：
  - 语法过于严谨
  - 代码不易读
  - eval 函数存在风险

对比：

JQuery 使用 Ajax

JQ 对 Ajax 操作进行了封装，在 jq 中

- 最底层的方法是：**$.ajax().**
- **第二层的方法是：load(), $.get(), $.post().**
- **第三层的方法是：$.getScript(), $.getJSON()**

Load()

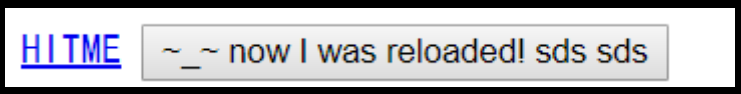
jsObj.load(url, *[data]*, *[callback]*) 简单 常用！

- **url: String** 待装入 HTML 网页网址。
- **data *(可选)*** **Map,String** 发送至服务器的 key/value 数据。在 jQuery 1.3 中也可以接受一个字符串了。
- **callback *(可选)*Callback** 载入成功时回调函数。

```
  <script>
      $(function () {
          $("#a").click(function () {
              const url = this.href ;
              let args = {"time":new Date()};//可以使用这个附加数据！
              $("#h").load(url);
              // $("#h").load(url,args,function () {
              //     alert("Loaded!")
              // });
              return false;
          })
      })
  </script>
head>
ody>
  <a id="a" href="temp.txt">HITME</a>
  <h1 id="h">还没有load</h1>
```

任何一个对象都可以使用load（）
使用该方法后，文件中的数据将
被加载到其中！

任何对象都可以加载，如按钮：




```
    $("#a").click(function () {
        // const url = this.href ;
        const url = this.href //this.href指向的是当前标签的href,一般a标签后href。
```

细节

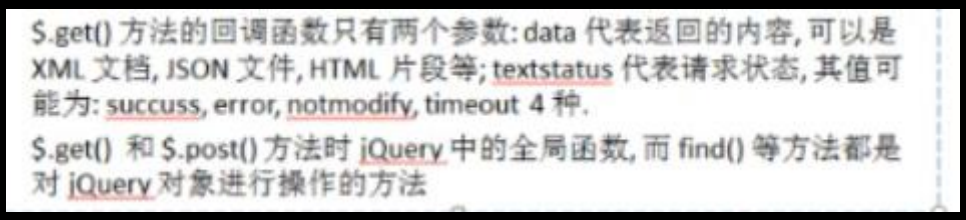
$.get()

通过远程 HTTP GET 请求载入信息,异步请求。这是一个简单的

GET 请求功能以取代复杂 $.ajax 。请求成功时可调用回调函数。如果需要在出错时执行函数，请使用 $.ajax。

$.get(url, *[data], [callback], [type]*)

- type *（可选）*String 返回内容格式，xml，html，script，json，text，_default。



示例

```
    <script>
        $(function () {
            $("#a").click(function () {
                const url = this.href;
                let args = {"time":new Date()};
                $.get(url,args,function (data) {
                    alert(data.name+data.age+data.version);
                });
                return false;
            });

            $("#b").click(function () {
                let url  = "temp.json";
                $.get(url,function (d) { //回调函数的参数表示一个调用get方法所返回的数据，名称任意。
                   alert(d.name+"  "+d.age);
                });
                return false;
            })

        })
    </script>
/head>
body>
    <a id="a" href="temp.json">HITME</a><br>
    <a id="b">ClickMe</a>
```

$.get(url, args, function(data){})

get是从URL中获取数据

$.post()

同$.get()

$.getJSON()

$.getJSON(url, *[data], [callback]*)

专门获取 json 数据，用法同 get

```
$("#c").click(function () {
    let url  = "temp.json";
    $.getJSON(url,function (d) {
        alert(d.name+"  "+d.age);
    });
    return false;
})
```

注意

Json 文档有两种形式：

- 没有对象名，相当于一个匿名对象

  格式为{name:value}

  ```
  {
    "name": "Edwin",
    "version": "1.0.0",
    "age": 21
  }
  ```

  使用$.get[JSON], $.post 获取时，直接 data.name 即可

  ```
  $.get(url,args,function (data) {
      alert(data.name+data.age+data.version);
  });
  ```

- 有对象，至少一个

  格式：{objName1:{name: value}, objName2: {name: value}}

```
{
  "user":
  {
    "name": "Edwin",
    "version": "1.0.0",
    "age": 21
  },

  "friends": {
    "name": "Edwin's Friends",
    "age": 1111,
    "motto": "Fight For Edwin Forever!"
  }
}
```

使用$.get[JSON], $.post 获取时，要先获取对象名，在获取数据

```
$.get(url,function (data)
    alert(data.user.name);
});
```

对象名

$.getScript（）

$.getScript(url, *[callback]*)

通过 HTTP GET 请求载入并执行一个 JavaScript 文件。

注意：方法会执行 URL 所表示的 JS 文件

```
$("#d").click(function () {
    let url  = "getScript.js";
    $.getScript(url,function () {
        alert("executed successfully!")
    });
    return false;
})
```

小结

1. 什么是 Ajax ？不用刷新页面，但可以和服务端进行通信的方式。使用 Ajax 的主要方式是 XMLHttpRequest 对象

2. 使用 XMLHttpRequest 对象实现 Ajax。[了解]

3. Ajax 传输数据的 3 种方式：

1). XML：笨重，解析困难。但 XML 是通用的数据交换格式。
2). HTML：不需要解析可以直接放到文档中。若仅更新一部分区域。但传输的数据不是很方便，且 HTML 代码需要拼装完成。
3). JSON：小巧，有面向对象的特征，且有很多第三方的 jar 包可以把 Java 对象或集合转为 JSON 字符串。

4. 使用 jQuery 完成 Ajax 操作

1). load 方法：可以用于 HTML 文档的元素节点，把结果直接加为对应节点的子元素。通常而言，load 方法加载后的数据时一个 HTML 片段。

```
var $obj = ...
var url = ...
var args = {key:value,...};
$obj.load(url, args);
```

2). $.get,$.post,$.getJSON：更加灵活。除去使用 load 方法的情况，大部分时候都使用这 3 个方法

I. 基本的使用

```
//url: Ajax 请求的目标 URL
//args: 传递的参数：JSON 类型
//data: Ajax 响应成功后的数据。可能是 XML，HTML，JSON
$.get(url, args, function(data){

})
```

II. 请求 JSON 数据

```
$.get(url, args, function(data){

}, "JSON");

$.get(url, args, function(data){

}, "JSON");
```

```
$.getJSON(url, args, function(data){

})
```

后台与 AJAX 结合

前端：

```
$.post(url,function(data){
    if(data.substring(0,1)=="Y"){

    }
```

前端发送请求

请求被响应后，在回调函数中执行后续步骤

后端：

```java
@RequestMapping("searchUser")
public String searchUser(HttpServletRequest request,HttpServletResponse response) throws IOExceptio
    String inp = request.getParameter("input");接受请求
    List<User> users = service.getAllUserBySidOrName(inp);
    String string = "";
    for(User u:users) {  //可以使用数组，算了。
        string+=(u.getName()+"\n||"+u.getSid()+"\n||");
    }
    response.setContentType("text;charset=utf-8");
    response.setCharacterEncoding("UTF-8");
    response.getWriter().print(string);

    return "Error";
}
```

响应

回传到请求发出方!

回传的数据也有几种格式。

JQ 的$.ajax()

jQuery.ajax(*[options]*)

最底层的实现，不常用，除非操作特殊对象。

$.ajax() 返回其创建的 XMLHttpRequest 对象。

最简单的情况下，$.ajax()可以不带任何参数直接使用。

所有的选项都可以通过$.ajaxSetup()函数来全局设置。

如果要处理$.ajax()得到的数据，则需要使用**回调函数**

- beforeSend 在发送请求之前调用，并且传入一个 XMLHttpRequest 作为参数。
- error 在请求出错时调用。传入 XMLHttpRequest 对象，描述错误类型的字符串以及一个异常对象（如果有的话）

  dataFilter 在请求成功之后调用。传入返回的数据以及"dataType" 参数的值。并且必须返回新的数据（可能是处理过的）传递给 success 回调函数。

  success 当请求之后调用。传入返回后的数据，以及包含成功代码的字符串。

complete 当请求完成之后调用这个函数，无论成功或失败。传入 XMLHttpRequest 对象，以及一个包含成功或错误代码的字符串。

参数

1.url:

要求为 String 类型的参数，（默认为当前页地址）发送请求的地址。

2.type:

要求为 String 类型的参数，请求方式（post 或 get）默认为 get。注意其他 http 请求方法，例如 put 和 delete 也可以使用，但仅部分浏览器支持。

3.timeout:

要求为 Number 类型的参数，设置请求超时时间（毫秒）。此设置将覆盖$.ajaxSetup()方法的全局设置。

4.async:

要求为 Boolean 类型的参数，默认设置为 true，所有请求均为异步请求。如果需要发送同步请求，请将此选项设置为 false。注意，同步请求将锁住浏览器，用户其他操作必须等待请求完成才可以执行。

5.cache:

要求为 Boolean 类型的参数，默认为 true（当 dataType 为 script 时，默认为 false），设置为 false 将不会从浏览器缓存中加载请求信息。

6.data:

要求为 Object 或 String 类型的参数，发送到服务器的数据。如果已经不是字符串，将自动转换为字符串格式。get 请求中将附加在 url 后。

防止这种自动转换，可以查看 processData 选项。对象必须为 key/value 格式，例如 {foo1:"bar1",foo2:"bar2"} 转换为 &foo1=bar1&foo2=bar2。如果是数组，JQuery 将自动为不同值对应同一个名称。例如{foo:["bar1","bar2"]}转换为&foo=bar1&foo=bar2。

7.dataType:

要求为 String 类型的参数，预期服务器返回的数据类型。如果不指定，JQuery 将自动根据 http 包 mime 信息返回 responseXML 或 responseText，并作为回调函数参数传递。可用的类型如下：

xml：返回 XML 文档，可用 JQuery 处理。

html：返回纯文本 HTML 信息；包含的 script 标签会在插入 DOM 时执行。

script：返回纯文本 JavaScript 代码。不会自动缓存结果。除非设置了 cache 参数。注意在远程请求时（不在同一个域下），所有 post 请求都将转为 get 请求。

json：返回 JSON 数据。

jsonp：JSONP 格式。使用 SONP 形式调用函数时，例如 myurl?callback=?，JQuery 将自动替换后一个“?”为正确的函数名，以执行回调函数。

text：返回纯文本字符串。

8.beforeSend:

要求为 Function 类型的参数，发送请求前可以修改 XMLHttpRequest 对象的函数，例如添加自定义 HTTP 头。在 beforeSend 中如果返回 false

可以取消本次 ajax 请求。XMLHttpRequest 对象是惟一的参数。

```
function(XMLHttpRequest){
  this;  //调用本次 ajax 请求时传递的 options 参数
}
```

9.complete:

要求为 Function 类型的参数，请求完成后调用的回调函数（请求成功或失败时均调用）。参数：XMLHttpRequest 对象和一个描述成功请求类型的字符串。

```
function(XMLHttpRequest, textStatus){
  this;  //调用本次 ajax 请求时传递的 options 参数
}
```

10.success：要求为 Function 类型的参数，请求成功后调用的回调函数，有两个参数。

(1)由服务器返回，并根据 dataType 参数进行处理后的数据。

(2)描述状态的字符串。

```
function(data, textStatus){
  //data 可能是 xmlDoc、jsonObj、html、text 等等
  this;  //调用本次 ajax 请求时传递的 options 参数
}
```

11.error:

要求为 Function 类型的参数，请求失败时被调用的函数。该函数有 3 个参数，即 XMLHttpRequest 对象、错误信息、捕获的错误对象(可选)。

ajax 事件函数如下：

```
function(XMLHttpRequest, textStatus, errorThrown){
  //通常情况下 textStatus 和 errorThrown 只有其中一个包含信息
  this;  //调用本次 ajax 请求时传递的 options 参数
}
```

12.contentType：

要求为 String 类型的参数，当发送信息至服务器时，内容编码类型默认为"application/x-www-form-urlencoded"。该默认值适合大多数应用场合。

13.dataFilter：

要求为 Function 类型的参数，给 Ajax 返回的原始数据进行预处理的函数。提供 data 和 type 两个参数。data 是 Ajax 返回的原始数据，type 是调用 jQuery.ajax 时提供的 dataType 参数。函数返回的值将由 jQuery 进一步处理。

```
function(data, type){
  //返回处理后的数据
  return data;
}
```

14.dataFilter：

要求为 Function 类型的参数，给 Ajax 返回的原始数据进行预处理的函数。提供 data 和 type 两个参数。data 是 Ajax 返回的原始数据，type 是调用 jQuery.ajax 时提供的 dataType 参数。函数返回的值将由 jQuery

进一步处理。

```
function(data, type){
    //返回处理后的数据
    return data;
}
```

15.global：

要求为 Boolean 类型的参数，默认为 true。表示是否触发全局 ajax 事件。设置为 false 将不会触发全局 ajax 事件，ajaxStart 或 ajaxStop 可用于控制各种 ajax 事件。

16.ifModified：

要求为 Boolean 类型的参数，默认为 false。仅在服务器数据改变时获取新数据。服务器数据改变判断的依据是 Last-Modified 头信息。默认值是 false，即忽略头信息。

17.jsonp：

要求为 String 类型的参数，在一个 jsonp 请求中重写回调函数的名字。该值用来替代在"callback=?"这种 GET 或 POST 请求中 URL 参数里的"callback"部分，例如{jsonp:'onJsonPLoad'}会导致将"onJsonPLoad=?"传给服务器。

18.username：

要求为 String 类型的参数，用于响应 HTTP 访问认证请求的用户名。

19.password：

要求为 String 类型的参数，用于响应 HTTP 访问认证请求的密码。

20.processData：

要求为 Boolean 类型的参数，默认为 true。默认情况下，发送的数据将被转换为对象（从技术角度来讲并非字符串）以配合默认内容类型 "application/x-www-form-urlencoded"。如果要发送 DOM 树信息或者其他不希望转换的信息，请设置为 false。

21.scriptCharset：

要求为 String 类型的参数，只有当请求时 dataType 为"jsonp"或者 "script"，并且 type 是 GET 时才会用于强制修改字符集(charset)。通常在本地和远程的内容编码不同时使用。

使用 Jackson

使用 BlockUI

Ajax 应用

- 登录、注册等表单填写时的格式错误提示
- 购物车类，总价等变化

Servlet

在 iDE 中可以新建服务器类：Servlet

设置 mapping URL 即可作为一个服务器。

Js 中的 const var let

- const: 声明创建一个只读的常量。这不意味着常量指向的值不可变，而是变量标识符的值只能赋值一次，必须初始化。
- **var：声明变量。**
- **let: 块级作用域，函数内部使用 let 对外部无影响。**

走进社会：认识社会，了解社会