# AJAX 的概念和使用

## 三、AJAX 异步交互

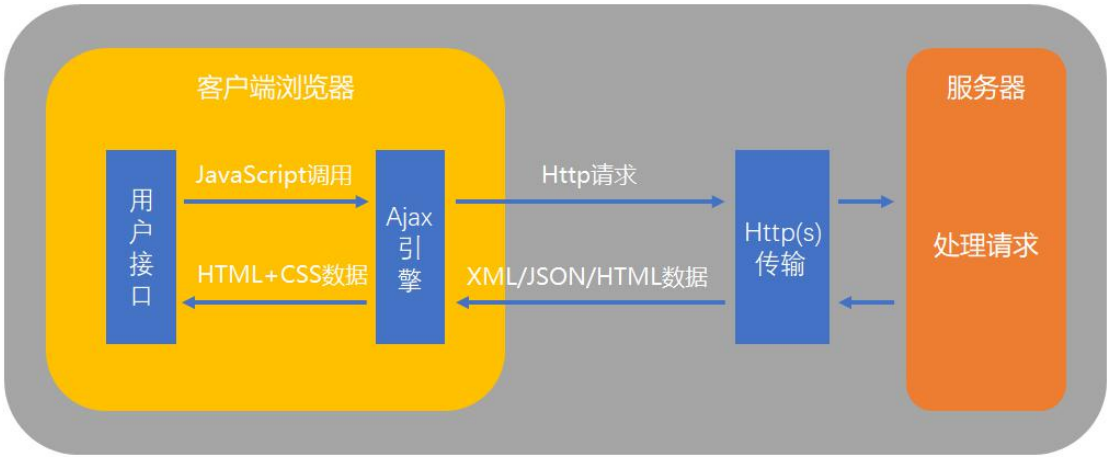
1.AJAX 的概念

- Ajax 全称为 Asynchronous JavaScript and XML，即异步的 JavaScript 和 XML。
- Ajax 不是新的编程语言，而是一种将现有的标准组合在一起使用的新方式。
- Ajax 最大的优点是在不重新加载整个页面的情况下，可以与服务器交换数据并更新部分网页内容。
- Ajax 不需要任何浏览器插件，但需要用户允许 JavaScript 在浏览器上执行。
- 简而言之，AJAX 是一个可以和后台沟通的技术。

2. Ajax 的工作原理

Ajax 的工作原理相当于在用户和服务器之间加了一个中间层(Ajax 引擎)，使用户操作与服务器响应异步化。

并不是所有用户请求都提交服务器，像一些数据验证和数据处理等都交给 Ajax 引擎自己来做，只有确定需要从服务器读取新数据时再由 Ajax 引擎代为向服务器提交请求。




3.Ajax 中所包含的技术

Ajax 并非是一种新的技术，而是几种原有技术的结合体。它由下列技术组合而成：

- 使用 CSS 和 HTML 来表示。
- 使用 DOM 模型来交互和动态显示。
- 使用 XMLHttpRequest 来和服务器进行异步通信。
- 使用 javascript 来绑定和调用。

## 二、AJAX 的使用

1. 使用步骤

（1）创建异步对象 XMLHttpRequest


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

需要创建一个专门用于做异步交互的对象 XMLHttpRequest （可以缩写为 xhr 但不是一定的）

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

2. 设置请求信息

创建个请求链接到服务器，先要使用我们创建的 XMLHttpRequest 对象。在这个对象里有个`open()`方法，这个方法的作用类似于初始化，并不会发起真正的请求。参数：

- method：请求的类型；GET 或 POST 等，大写小写不敏感
- url：文件在服务器上的位置（数据）
- async：true（异步）或 false（同步）

```
xhr.open(method, url, async);
```

3. 发送异步请求

`send()` 方法发送请求，并接受一个可选参数（body）。但根据请求方法的不同，参数也不同。

（1）GET 方式

当请求方式为 GET 时，可以不传或传入 null，如：send(null)

```
xhr.send(body);
```

（2）POST 方式

① 协议规定 POST 提交的数据必须放在消息主体中，但协议并没有规定数据必须使用什么编码方式。所以 POST 提交数据方案，包含了 Content-Type 和消息主体编码方式两部分。xhr.setRequestHeader("Content-Type":"编码格式")

② 因此，POST、PUT 请求需要置请求头，因为 POST 方式可以传递很多种格式以及文件，因此在发送之出需要告诉服务器发送的用什么格式解析发送的内容。

③ application/x-www-form-urlencoded

application/x-www-form-urlencoded 是最常见的 POST 提交数据的方式了。urlencoded 格式，又叫 form 格式、x-www-form-urlencoded 格式，它是一种表单格式。

设置请求头放在 open 和 send 中间。此方法可将参数转换为：`name=value`对之间放置`&`符号表现形式，如：id=1&names=tom

键值对组成：

键和值之间用 = ： name=poloyy

多个键值对之间用 & ： name=poloyy&age=19

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

④ 其他方式还有（本阶段无法使用以下三种格式）：

multipart/form-data

application/json

text/xml

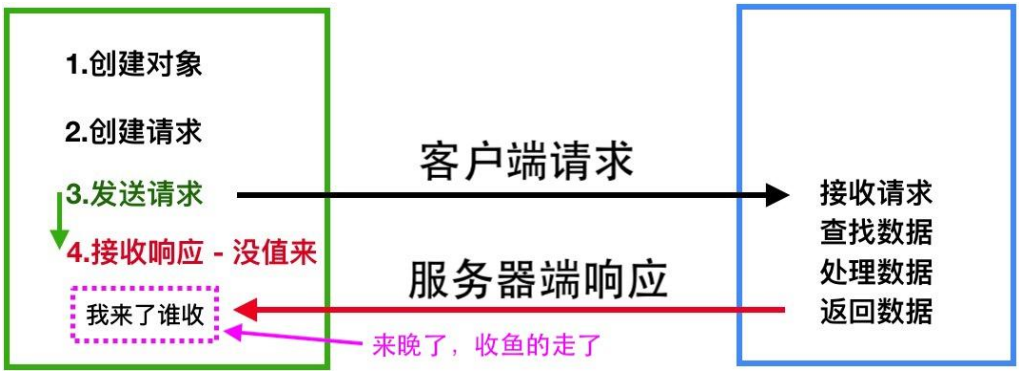
4. 接收响应

接收服务器发回的响应，有同步接收和异步接收两种

（1）同步接收

xhr.open(请求类型，请求接口的路径，同步); open 方法的最后一个参数需要设置为同步接收。（一般不会使用同步接收）

（2）异步接收




① readyState 属性

当发送异步请求时，可以检测到 XMLHttpRequest 对象的 readyState 属性。该属性表示请求/响应过程的当前活动阶段。readyState 的五个状态（XMLHttpRequest 对象的生命周期各个阶段）

- 0：未初始化。尚未调用`open()`方法。
- 1：启动。已经调用`open()`方法，但尚未调用`send()`方法。
- 2：发送。已经调用`send()`方法，但尚未接收到响应。
- 3：接收。已经接收到部分响应数据。
- 4：完成。已经接收到全部响应数据，而且已经可以在客户端使用了。


② onreadystatechange 事件属性

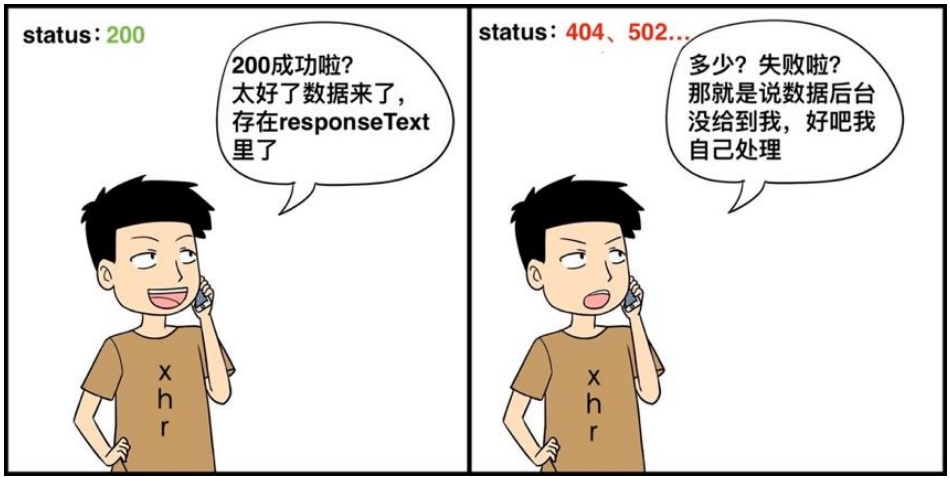
`readyState`属性值的变化是动态的，因此每一次的变化都会触发一次`readystatechange`事件属性。

可以利用这个`onreadystatechange`事件让浏览器识别并检测每次状态变化后`readyState`的值。（加了`on`浏览器就可以自动识别这是事件属性，并进行监听）

③ status 属性

XMLHttpRequest.status 返回了 XMLHttpRequest 响应中的数字状态码，状态码是一个三位数，用代码的形式表示处理的结果。如 200 是成功，404 没找到资源，500 是服务器错误，2xx 范围内都属于成功。

如果状态码不是 200-300 之间，也不是 304，那就说明服务器的反应是没找到你需要的数据，或者服务器出现问题无法给你。




④ responseText 属性

XMLHttpRequest.responseText 是响应主体被返回的文本，只能是文本形式，拿到后需转换格式。

```
xhr.onreadystatechange = function(){
  if (xhr.readyState == 4) {
    if (xhr.status >= 200 && xhr.status < 300) {
      var result = xhr.responseText;
      console.log(result);
    } else {
      console.log('服务器错误');
    }
  }
};
```

## ajax四步形成：

1.创建异步对象 2.设置请求信息 3.发送异步请求 4.接收响应

顺序需要调整，原因如下：

```
//第一步 创建对象
if (window.XMLHttpRequest) {
  var xhr = new XMLHttpRequest();
} else {
  var xhr = new ActiveXObject('Microsoft.XMLHTTP');
}
//第二步 创建链接
xhr.open('get', 'http://localhost:8082/get', true);
//第三步 发送请求
xhr.send();
//第四步 接收响应 创建监听
xhr.onreadystatechange = function () {
  console.log(xhr.readyState);
  if (xhr.readyState == 4 && xhr.status == 200) {
    var data = xhr.responseText;
    //  console.log(data);
  }
}
```

```
Elements
top
2
3
4
```

```
//第一步 创建对象
if (window.XMLHttpRequest) {
  var xhr = new XMLHttpRequest();
} else {
  var xhr = new ActiveXObject('Microsoft.XMLHTTP');
}
//第四步 接收响应 创建监听
xhr.onreadystatechange = function () {
  console.log(xhr.readyState);
  if (xhr.readyState == 4 && xhr.status == 200) {
    var data = xhr.responseText;
    // console.log(data);
  }
}
//第二步 创建链接
xhr.open('get', 'http://localhost:8082/get', true);
//第三步 发送请求
xhr.send();
```

```
Elements Console
top
1
2
3
4
▸XHR finished loading: GET
alhost:8082/get".
```

【重点的重点】ajax 知识总结：（4 步）

- 1、创建异步对象（1）
- 2、创建监听接收响应（4）
- 3、设置请求信息（2）

- 4、发送异步请求（3）

代码示例：

```
/*创建 ajax 函数*/
function ajax() {
    /*创建 XMLHttpRequest 对象*/
    var xhr = new XMLHttpRequest();

    /*监听 readyState 的变化*/
    xhr.onreadystatechange = function () {
        if (xhr.readyState == 4) {
            if ((xhr.status >= 200 && xhr.status < 300)) {
                var result = xhr.responseText;
                console.log(result);
            } else {
                console.log('服务器错误');
            }
        }
    };
    /*设置请求信息*/
    xhr.open('GET', 'http://localhost:5000', true);
    /*发送请求*/
    xhr.send(null);
}
/*调用 ajax 函数*/
ajax();
```