目录 Contents

浏览器发送请求的方式

- **浏览器向服务器发送请求的方式有很多：**
  - 在浏览器地址栏输入url按回车
  - 点击<a> 链接
  - js中的location.href="新的url地址"
  - <link>、<script>、<img>
  - css里的background-image：url()
  - form表单
  - js中的**XMLHttpRequest （即：ajax）**
  - etc.....

<a>、<img>等这些请求自定义弱，也比较简单，所以ajax是我们的重点学习对象

了解Ajax

- Ajax 的全称是 Asynchronous Javascript And XML（异步 JavaScript 和 XML）。
- 通俗的理解：在网页中利用webAPI中的 **XMLHttpRequest** 这个构造函数发出的请求，就叫Ajax请求
- 最大优点：自定义强，**可以任意设置请求行、请求头、请求体**。而且还**不会刷新页面**
- 代码实现：
  - 原生js *（代码复杂且生涩，但对理解ajax有帮助，所以理解即可，工作中100%不用）*
  - 插件 *（目前最流行的是axios，书写代码简洁）*

目录 Contents

- ajax - 原生js实现

js 如何发送ajax请求


```
// 1. 创建 XHR 对象  xhr 就是一个变量名（取自XMLHttpRequest 的首字母）
var xhr = new XMLHttpRequest()

// 2. 调用 open 函数     只是启动，http请求并没有发出去        有三个参数，最后一个参数表示是否异步发送http请求，默认值为true
xhr.open('GET', 'http://www.liulongbin.top:3006/api/getbooks', false)

// 3. 调用 send 函数     真正把http请求发出去了
xhr.send()

// 4. 打印服务器响应回来的数据
console.log(xhr.response)
```

异步ajax请求


```
// 1. 创建 XHR 对象
var xhr = new XMLHttpRequest()
// 2. 调用 open 函数     最后一个参数表示是否异步发送http请求，默认值为true，可以不写
xhr.open('GET', 'http://www.liulongbin.top:3006/api/getbooks')
// 3. 调用 send 函数
xhr.send()
// 4. 怎么能知道请求回来了呢？答：xhr 对象中有一个 readyState 属性，它表示当前 Ajax 请求所处的状态
```

readyState属性

xhr 对象中的 readyState 属性，用来表示**当前 Ajax 请求所处的状态**。每个 Ajax 请求必然处于以下状态中的一个：

值	状态	描述
0	UNSENT	XMLHttpRequest 对象已被创建，但尚未调用 open方法。
1	OPENED	open() 方法已经被调用。
2	HEADERS_RECEIVED	send() 方法已经被调用，响应头也已经被接收。
3	LOADING	数据接收中，此时 response 属性中已经包含部分数据。
4	DONE	Ajax 请求完成，这意味着数据传输已经彻底完成或失败。

异步ajax请求


```
// 1. 创建 XHR 对象
var xhr = new XMLHttpRequest()
// 2. 调用 open 函数    最后一个参数表示是否异步发送http请求，默认值为true，可以不写
xhr.open('GET', 'http://www.liulongbin.top:3006/api/getbooks')
// 3. 调用 send 函数
xhr.send()
// 4. on  readystate  change 监听readystate值，只要发生变化就会自动触发这个函数
xhr.onreadystatechange = function() {
    // 4.1 当readyState的值为4，代表请求回来了
    if (xhr.readyState === 4) {
        // 4.2 再判断status的值是2xx，代表请求成功
        if (xhr.status >= 200 && xhr.status < 300) {
            console.log(xhr.response)
        }
    }
}
```

提问

1. 工作中ajax请求使用同步发送还是异步发送？
2. 异步ajax请求发送过程？

目录 Contents

- ◆ 如何携带数据
  - url参数
  - 请求头
  - 请求体

方式一：查询字符串（也叫url参数）


```
xhr.open('GET', 'http://www.xx.com/api/getbooks?username=xxx&password=xxx')
```

练习


```
var xhr = new XMLHttpRequest()
xhr.open('GET', 'http://www.liulongbin.top:3006/api/getbooks?id=1&bookname=西游记')
xhr.send()
xhr.onreadystatechange = function() {
    if (xhr.readyState === 4) {
        if (xhr.status >= 200 && xhr.status < 300) {
            console.log(xhr.response)
        }
    }
}
```

URL 大小限制

浏览器最大长度限制参考：

浏览器	最大长度（字符数）
Internet Explorer	2048
Edge	4035
Firefox	65536
Chrome	8182
Safari	80000
Opera	190000

服务器最大长度限制参考：

服务器	最大长度（字符数）
Apache(Server)	8192
IIS	16384
Nginx	4096
Tomcat	65536

注意：http协议本身并没有对url做大小限制

目录 Contents

- ◆ 如何携带数据
  - url参数
  - 请求头
  - 请求体

方式二：请求头


```
xhr.setRequestHeader('username', 'xxx')
xhr.setRequestHeader('password', 'xxx')
```

练习


```
var xhr = new XMLHttpRequest()
xhr.open('GET', 'http://www.liulongbin.top:3007/my/userinfo')
xhr.setRequestHeader('Authorization', 'Bearer token')    // 注意：Bearer和token之间有个空格，这是规定
xhr.send()
xhr.onreadystatechange = function() {
    if (xhr.readyState === 4) {
        if (xhr.status >= 200 && xhr.status < 300) {
            console.log(xhr.response)
        }
    }
}
```

方式二：请求头

- http协议没有对请求头做大小限制
- 但是服务器一般会设置请求头的大小限制，一般为512k

目录 Contents

- ◆ 如何携带数据
  - url参数
  - 请求头
  - 请求体

方式三：请求体

问题：如果数据结构比较复杂怎么办？

```
// 改为post请求
xhr.open('POST', 'http://www.liulongbin.top:3006/api/addbooks')
// 把数据放到send()函数中即可
xhr.send(数据)
```

自定义（强烈不推荐）


```
// 改为post请求
xhr.open('POST', 'http://www.liulongbin.top:3006/api/addbooks')
xhr.send( 'xxx|xxx' )    //用任意特殊字符分隔开，但前提一定要和后端java同学商量好
```

查询字符串（不太推荐）


```
// 改为post请求
xhr.open('POST', 'http://www.liulongbin.top:3006/api/addbooks')
// 当请求体的数据格式为查询字符串格式时，Content-Type 属性的值必须改为 application/x-www-form-urlencoded，否则后端java无法正确解析数据
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded')
xhr.send('username=xxx&password=xxx')
```

练习


```
var xhr = new XMLHttpRequest()
xhr.open('POST', 'http://www.liulongbin.top:3006/api/addbooks')
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded')
xhr.send('bookname=水浒传&author=施耐庵&publisher=上海图书出版社')
xhr.onreadystatechange = function() {
    if (xhr.readyState === 4) {
        if (xhr.status >= 200 && xhr.status < 300) {
            console.log(xhr.response)
        }
    }
}
```

json（推荐）


```
// 改为post请求
xhr.open('POST', 'http://www.liulongbin.top:3006/api/getbooks')
// 当请求体的数据格式为json格式时，Content-Type 属性的值必须改为 application/json，否则后端java无法正确解析数据
xhr.setRequestHeader('Content-Type', 'application/json')
xhr.send( '{"username":"xxx","password":"xxx"}' )
```

练习


```
var xhr = new XMLHttpRequest()
xhr.open('POST', 'http://www.liulongbin.top:3006/api/formdata')
xhr.setRequestHeader('Content-Type', 'application/json')
xhr.send('{"username":"xxx","password":"xxx"}')
xhr.onreadystatechange = function() {
    if (xhr.readyState === 4) {
        if (xhr.status >= 200 && xhr.status < 300) {
            console.log(xhr.response)
        }
    }
}
```

文件（包括视频、图片、其他类型文件）


```
<!-- 1. 文件选择框 -->
<input type="file" id="file1" />
<!-- 2. 上传按钮 -->
<button id="btnUpload">上传文件</button>
```

```
// 1. 获取到选择的文件列表，值为一个数组
var files = document.querySelector('#file1').files
// 2. 创建一个FormData对象，这是浏览器提供的内置构造函数，专门用来提交表单数据
var fd = new FormData()
// 3. 将文件放到FormData对象中
fd.append('avatar', files[0])
// 4. 设置content-type的值为：Content-Type: multipart/form-data;  （这里不设置也可以，因为浏览器会自动设置）
xhr.setRequestHeader('Content-Type', 'multipart/form-data')
// 5. 发起请求
xhr.send(fd)
```

练习


```
// 1. 获取上传文件的按钮
var btnUpload = document.querySelector('#btnUpload')
// 2. 为按钮添加 click 事件监听
btnUpload.addEventListener('click', function() {
    // 3. 获取到选择的文件列表
    var files = document.querySelector('#file1').files
    if (files.length <= 0) {
        return alert('请选择要上传的文件！')
    }
    var fd = new FormData()
    fd.append('avatar', files[0])
    // 1. 创建 xhr 对象
    var xhr = new XMLHttpRequest()
    // 2. 调用 open 函数，指定请求类型与URL地址。其中，请求类型必须为 POST
    xhr.open('POST', 'http://www.liulongbin.top:3006/api/upload/avatar')
    // 3. 发起请求
    xhr.send(fd)
})
```

方式三：请求体

- http协议没有对请求体做大小限制
- 但是服务器一般会设置请求头的大小限制，一般为2M ~ ?G不等

比较

URL

- 缺点：
  1. 只能发送字符型数据，不能发生二进制数据（图片、视频等）
  2. url的最大长度为2k左右（因为得兼顾ie浏览器）
  3. 相对来说不太安全
- 优点：简单

请求头

- 缺点：
  1. 只能发送字符型数据，不能发生二进制数据（图片、视频等）
  2. 如果要发生多个字段非常麻烦
- 优点：只发生一两个字段比较简单

请求体

- 缺点：
  1. 必须设置content-type，而且要和请求体里的数据格式一致
- 优点：可以发生复杂结构的（json）、大体量（理论上无限制，但服务器一般会设置为2M ~ ?G）、任何类型（图片、视频等）的数据

目录 Contents

设置HTTP请求时限

如果网速很慢，用户可能要等很久。XMLHttpRequest 对象还提供了一个 timeout 属性，可以设置 HTTP 请求的时限：

```
xhr.timeout = 15000 (一般设置为15s左右)
```

上面的语句，将最长等待时间设为 15000 毫秒。过了这个时限，就自动停止HTTP请求。与之配套的还有一个 timeout 事件，用来指定回调函数：

```
xhr.ontimeout = function(event){
    alert('请求超时！请稍后再试！')
}
```

目录 Contents

显示文件上传进度

html代码

css代码

显示文件上传进度


```
// 创建 XHR 对象
var xhr = new XMLHttpRequest()
// 监听 xhr.upload 的 onprogress 事件
xhr.upload.onprogress = function(e) {
    // e.lengthComputable 是一个布尔值，表示当前上传的资源是否具有可计算的长度
    if (e.lengthComputable) {
        // e.loaded 已传输的字节
        // e.total  需传输的总字节
        var percentComplete = Math.ceil((e.loaded / e.total) * 100)
    }
}
```

目录 Contents

综合案例

图片上传案例

- 两种：
  - 第一种：选择图片--------点击上传按钮----------显示上传进度---------上传完成后显示服务器里的图片
  - 第二种：选择图片--------显示该图片--------点击上传按钮----------显示上传进度---------上传完成后提示成功

第一种


```
xhr.onreadystatechange = function() {
  if (xhr.readyState === 4 && xhr.status === 200) {
    var data = JSON.parse(xhr.response)
    if (data.status === 200) { // 上传文件成功
      // 将服务器返回的图片地址，设置为 <img> 标签的 src 属性
      document.querySelector('#img').src = 'http://www.liulongbin.top:3006' + data.url
    } else { // 上传文件失败
      console.log(data.message)
    }
  }
}
```

第二种


```
// 1. 为input绑定事件处理函数
document.querySelector('input').addEventListener('change', function () {
    // 2. 获取到用户选择的文件列表
    var files = document.querySelector('#file1').files
    // 3. 通过浏览器内置函数 URL.createObjectURL() 获取到图片本地路径
    var path = URL.createObjectURL(files[0])
    // 4. 显示用户选择的图片
    document.querySelector('#img').src = path
})
```