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>、 <link>、<script>、 <img>等这些请求自定义弱，也比较简单。（比如不能任意设置请求方式method和请求头、也没有请求体）

<form>也能发送http请求（1995年左右），但最大问题是：**会刷新页面，给用户的体验非常不好，所以现在几乎没人用**

（2005年左右）重点学习

<form>标签的属性

不用练，快速过

属性	值	描述
action	URL地址（默认值为当前页面的 URL 地址）	浏览器会自动把action的值放到http协议中
method	get（默认值） post	浏览器会自动把method的值放到http协议中
enctype	application/x-www-form-urlencoded （普通信息）（默认值） multipart/form-data（上传文件、图片等） text/plain　普通字符串	浏览器会自动把enctype的值放到http协议中content-type的值
target	_blank　在新窗口中打开 _self　在相同的框架中打开（默认值） _parent　在父框架集中打开 _top　在整个窗口中打开 *framename*　在指定的框架中打开	规定在何处打开 action 的 URL，和http协议无关

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：**1、自定义强，可以任意设置请求行、请求头、请求体。2、不会刷新页面**
- 代码实现：
  - 原生js *（代码复杂且生涩，工作中几乎不用）*
  - 插件 *（书写代码简洁。有很多，比如：jquery、fetch、**axios（最主流）**）*

思考

1. ajax请求只是浏览器发送众多http请求中的一种，对吗？
2. 不利用webAPI中的 **XMLHttpRequest** 这个构造函数也可以发送ajax请求，对吗？
3. ajax请求的好处？form表单请求的坏处？

目录 Contents

- ajax - 原生js实现

语法

不用练，快速过


```
// 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
xhr.open('GET', 'http://ajax-base-api-t.itheima.net/api/getbooks')

// 3. 监听是否收到服务器发过来的响应，并打印这些数据
xhr.addEventListener('load', function () {
     console.log(this.status)        // 响应行里的状态码
     let data = JSON.parse(this.response)     // 解析响应体
     console.log(data)
})
// 4. 监听请求错误，并打印这些数据
xhr.addEventListener('error', function () {
     console.log(this.status)        // 响应行里的状态码
     console.log(this.response)      // 响应体
})
// 5. 调用 send() 函数     真正把http请求发出去了
xhr.send()
```

目录 Contents

语法


```
axios({
      method: 'GET',    // 大小写都可以
      url: 'http://ajax-base-api-t.itheima.net/api/getbooks'
}).then(res => {
      console.log(res)   // 打印所有的响应信息
}).catch(res => {
      console.log(err)   // 打印错误信息
})
```

语法规则：如果请求成功（即响应状态码为2xx），会自动执行then里的函数，如果失败会自动实现catch里的函数

axios响应信息


```
{
  // `data` 由服务器提供的响应
  data: {},

  // `status` 来自服务器响应的 HTTP 状态码
  status: 200,

  // `statusText` 来自服务器响应的 HTTP 状态信息
  statusText: 'OK',

  // `headers` 服务器响应的头
  headers: {},

   // `config` 是为请求提供的配置信息
  config: {},
 // 'request'
  // `request` is the request that generated this response
  // It is the last ClientRequest instance in node.js (in redirects)
  // and an XMLHttpRequest instance the browser
  request: {}
}
```

案例

Id	书名	作者	出版社	操作
1	西游记	吴承恩	北京图书出版社	删除
2	红楼梦	曹雪芹	上海图书出版社	删除
3	三国演义	罗贯中	北京图书出版社	删除

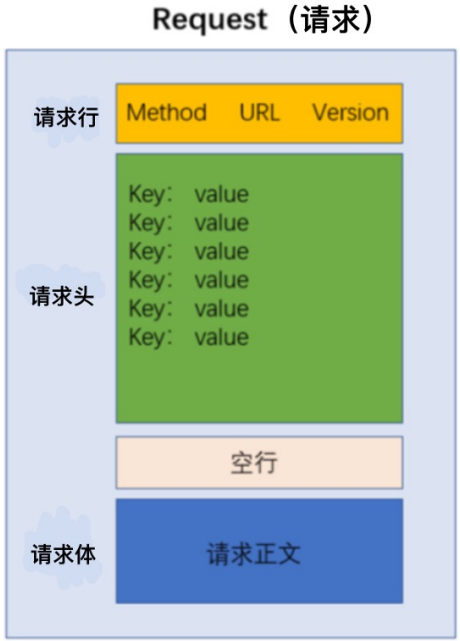
目录 Contents

- 如何携带数据？

案例

理论与现实

- **理论**上来说，http协议的**任何地方**都可以携带数据，而且**任何数据格式**都可以。
- 但在**现实**中，这样做势必会导致发送的数据**错乱复杂**。所以在工作中一般会在**URL、请求头、请求体**里以**某些特定的数据格式**来发送数据



位置

- url （IE浏览器大小限制：2k，服务器大小限制：4k往上一点）
- 请求头 （浏览器一般不限制，服务器大小限制：通常是512k）
- 请求体 （浏览器一般不限制，服务器大小限制：通常2M起步，多的可以接收几十G，甚至几十T）

- **注意：**

1. **http协议对数据大小没有做任何限制**
2. **由于请求体一般比较大，所以浏览器在发送http请求时，一般会先发送请求行和请求头，然后才发送请求体。**
   如果请求体过于庞大，则会一点点发送。服务器发送响应也是一样的（举例：视频）

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

目录 Contents

■ 特定数据格式（约定俗成的，不能更改）

- url参数
  - url查询参数
  - url动态参数
- 请求头
- 请求体
  - json格式
  - 查询字符串格式
  - FormData格式

语法


```
axios({
      method: 'GET',
      url: 'http://ajax-base-api-t.itheima.net/api/getbooks',
      params: {  // axios插件会自动将该数据转换成查询字符串格式，并自动拼接到url后面
          属性名: 'xxx',
          属性名: 'xxx'
      }
}).then(res => {
      console.log(res)  // 打印所有的响应信息
}).catch(res => {
      console.log(err)  // 打印错误信息
})
```

使用场景

- url查询参数的使用场景：**少量**、**结构简单**的数据
- 一般会通过url查询参数传输图片和视频吗？ 答：**不会**

目录 Contents

■ 如何携带数据？

- url参数
  - url查询参数
  - url动态参数
- 请求头
- 请求体
  - json格式
  - 查询字符串格式
  - FormData格式

语法


```
axios({
      method: 'GET',
      url: 'http://ajax-base-api-t.itheima.net/api/getbooks/xxx',
}).then(res => {
      console.log(res)   // 打印所有的响应信息
}).catch(res => {
      console.log(err)   // 打印错误信息
})
```

- 使用场景：只传输一个数据时通常采用这种方式（原因：懒）

目录 Contents

■ 如何携带数据？

- url参数
  - url查询参数
  - url动态参数
- 请求头
- 请求体
  - json格式
  - 查询字符串格式
  - FormData格式

语法


```
axios({
      method: 'GET',
      url: 'http://ajax-base-api-t.itheima.net/api/getbooks',
      headers: {    // axios插件会自动将该数据添加到http协议的请求头里
            属性名: 'xxx',
            属性名: 'xxx'
      }
}).then(res => {
      console.log(res)   // 打印所有的响应信息
}).catch(res => {
      console.log(err)   // 打印错误信息
})
```

- 使用场景：虽然可以用请求头传输数据，但工作中很少用。仅有一些特殊的数据需要通过请求头的方式传输给后端。
  比如：cookie、 Authorization

目录 Contents

■ 如何携带数据？

- url参数
  - url查询参数
  - url动态参数
- 请求头
- 请求体
  - json格式
  - 查询字符串格式
  - FormData格式

使用场景


```
axios({
      method: 'POST',  // 记得将method改为post，因为get请求设置请求体无效
      url: 'http://ajax-base-api-t.itheima.net/api/getbooks',
      data: {    // axios插件会自动将该数据转换为json格式，并添加到http协议的请求体里
          属性名: 'xxx',
          属性名: 'xxx'
      }
}).then(res => {
      console.log(res)  // 打印所有的响应信息
}).catch(res => {
      console.log(err)  // 打印错误信息
})
```

- **使用场景：json格式是现在工作中最主流的格式**

目录 Contents

■ 如何携带数据？

- url参数
  - url查询参数
  - url动态参数
- 请求头
- 请求体
  - json格式
  - 查询字符串格式
  - FormData格式

语法


```
axios({
      method: 'POST',
      url: 'http://ajax-base-api-t.itheima.net/api/getbooks',
      data: `属性名=xxx&属性名=xxx`     // 由于axios不能自动将data里的数据转换成查询字符串格式，
                                          所以只能我们自己写了~
}).then(res => {
      console.log(res)   // 打印所有的响应信息
}).catch(res => {
      console.log(err)   // 打印错误信息
})
```

- 使用场景：查询字符串格式是比较老的格式，现在几乎都不使用了

目录 Contents

■ 如何携带数据？

- url参数
  - url查询参数
  - url动态参数
- 请求头
- 请求体
  - json格式
  - 查询字符串格式
  - FormData格式

语法


```
let fd = new FormData()    // FormData是一个构造函数，它不是axios插件提供的，是浏览器提供的，属于浏览器的webAPIs。
fd.append('属性名', 'xxx')
fd.append('属性名', 'xxx')

axios({
      method: 'POST',
      url: 'http://ajax-base-api-t.itheima.net/api/getbooks',
      data: fd
}).then(res => {
      console.log(res)   // 打印所有的响应信息
}).catch(res => {
      console.log(err)   // 打印错误信息
})
```

- 使用场景：如果要传输文件（图片、视频等），一般使用FormData格式

练习


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

练习


```
// 1. 为按钮添加 click 事件监听
document.querySelector('#btnUpload').addEventListener('click', function() {
    // 2. 获取到选择的文件列表
    let files = document.querySelector('#file').files
    // 3. 判断用户是否选择了文件
    if (files.length <= 0) {
        alert('请选择要上传的文件！')
        return
    }
    // 4. 创建FormData对象
    let fd = new FormData()
    // 5. 向FormData对象中追加一个文件
    fd.append('avatar', files[0])
    // 6. 发送ajax请求
    axios({
        method: 'POST',
        url: 'http://ajax-base-api-t.itheima.net/api/upload/avatar',
        data: fd
    }).then(res => {}).catch(res => {})
})
```

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）
- 优点：简单（**尤其是url动态参数**），并且可以第一时间发送给服务器

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. 最大长度为512k左右
- 优点：可以第一时间发送给服务器

请求体

- 缺点：
  1. 不能是get请求
- 优点：可以发生**复杂结构**的（json）、**大体量**（理论上无限制，就看服务器性能了）、**任何类型**（图片、视频等）的数据

目录 Contents

■ ajax请求的额外功能

- 请求超时
- 文件上传进度
- 文件下载进度

语法


```
axios({
      method: 'GET',
      url: 'http://ajax-base-api-t.itheima.net/api/getbooks',
      timeout: 10000     // 如果该请求在10s内未接收到响应，则会触发catch里的函数
}).then(res => {
      console.log(res)   // 打印所有的响应信息
}).catch(res => {
      console.log(err)   // 打印错误信息
})
```

语法


```
axios({
      method: 'GET',
      url: 'http://ajax-base-api-t.itheima.net/api/getbooks',
      onUploadProgress: function (progressEvent) {  // 监听文件上传进度
          console.log(progressEvent)
      },
}).then(res => {
      console.log(res)  // 打印所有的响应信息
}).catch(res => {
      console.log(err)  // 打印错误信息
})
```

语法


```
axios({
      method: 'GET',
      url: 'http://ajax-base-api-t.itheima.net/api/getbooks',
      onDownloadProgress: function (progressEvent) {  // 监听文件下载进度
          console.log(progressEvent)
      },
}).then(res => {
      console.log(res)   // 打印所有的响应信息
}).catch(res => {
      console.log(err)   // 打印错误信息
})
```

1. 上面所有axios能实现的功能，原生js能实现吗？