目录 Contents

全局配置


```
// 给所有请求指定统一网址
axios.defaults.baseURL = 'https://api.example.com';
// 给所有请求添加一个请求头Authorization
axios.defaults.headers.common['Authorization'] = AUTH_TOKEN;
// 给所有的post请求添加一个请求头Content-Type
axios.defaults.headers.post['Content-Type'] = 'application/x-www-form-urlencoded'
```

- 好处：只需设置一次即可

请求拦截器


```
axios.interceptors.request.use(function (config) {
      // 可以在这里对config里的数据做任何修改
      config.url = 'https://api.example.com' + config.url
      // 记得一定要return config，否则axios拿不到config数据，也就无法发送ajax请求
      return config
}, function (error) {
      // 当请求出错时会自动执行这里
      return Promise.reject(error)
})
```

- 什么时候会执行请求拦截器里的函数？
- 答：在发送每一个ajax请求前都会自动执行请求拦截器里的函数（举例：安检）
- 好处：可以在请求拦截器的函数里对config里的数据做任何修改

响应拦截器


```
axios.interceptors.response.use(function (response) {
      // 当响应正确时会自动执行这里
      return config
}, function (error) {
      // 当响应出错时会自动执行这里
      return Promise.reject(error)
})
```

- 什么时候会执行响应拦截器里的函数？
- 答：只要浏览器接收到响应，第一时间会自动执行响应拦截器里的函数
- 好处：在响应拦截器的函数里可以对服务器的响应做一些处理

拦截器的执行过程

1. 调用axios()函数
2. 自动触发请求拦截器（如果没bug就执行第一个函数，如果有bug就执行第二个函数）
3. 发生http请求
4. 浏览器接收到响应
5. 自动触发响应拦截器（如果响应状态码是2xx就执行第一个函数，否则执行第二个函数）
6. 自动触发then或catch里的函数（如果响应拦截器没有 return Promise.reject()则执行then里的函数，否则执行catch里的函数）

目录 Contents

- ajax语法总结

语法

请求配置

这些是创建请求时可以用的配置选项。只有 `url` 是必需的。如果没有指定 `method`，请求将默认使用 `get` 方法。

```
{
   // `url` 是用于请求的服务器 URL
  url: '/user',

  // `method` 是创建请求时使用的方法
  method: 'get', // default

  // `baseURL` 将自动加在 `url` 前面，除非 `url` 是一个绝对 URL。
  // 它可以通过设置一个 `baseURL` 便于为 axios 实例的方法传递相对 URL
  baseURL: 'https://some-domain.com/api/',

  // `transformRequest` 允许在向服务器发送前，修改请求数据
  // 只能用在 'PUT', 'POST' 和 'PATCH' 这几个请求方法
  // 后面数组中的函数必须返回一个字符串，或 ArrayBuffer，或 Stream
  transformRequest: [function (data, headers) {
    // 对 data 进行任意转换处理
    return data;
  }],
```

目录 Contents

- URL编码

URL编码 – 导入

- 问题：在html代码中，为什么不能在页面中直接显示"<>"？
- 答：浏览器分不清
- 如何解决：对特殊字符进行转义（即用别的字符去表示）

URL编码

- 为了安全起见，URL中不允许使用非ASCII 字符（如：中文、日文、韩文等）。如果URL中真的有非ASCII 字符，浏览器会对其进行URL编码*（我们程序员不用做任何事情）*

- 不过js也给我们提供了两个方法来对url进行编码和解码： encodeURI/decodeURI *（其实浏览器在对url进行自动编码时内部使用的就是encodeURI 方法）*

```
encodeURI('http://xx.cn/黑马?bookname=西游记')
// 输出字符串：  http://xx.cn/%E9%BB%91%E9%A9%AC?bookname=%E8%A5%BF%E6%B8%B8%E8%AE%B0
decodeURI('%E6%B8%B8')
// 输出字符串：  游
```

URL编码

- 在URL中 /、?、=、&、# 等都是有特殊含义的，如果想把这些字符变成普通字符，也需要进行编码*（浏览器对这些特殊字符不会进行编码， encodeURI 也不会对这些字符进行编码）*
- js提供了两种方法对这些特殊字符进行编码与解码： encodeURIComponent/decodeURIComponent

```
encodeURI('6&10')
// 输出字符串:  6&10
encodeURIComponent("6&10");
// 输出字符串:  6%2610
decodeURIComponent("6%2610");
// 输出字符串:  6&10
```

- 在工作中的作用：比如 https://baidu.com?wd=js中&符号的作用&q=xxx *（第一个&是用户要搜索的内容，第二个&是查询字符串的分隔符）*

```
let url = 'https://baidu.com?wd=' + encodeURIComponent('js中&符号的作用') + '&q=xxx';
```

提问

1. URL里不允许有哪些字符？怎么解决？我们需要做什么事情？
   - ✓ 非ASCII 字符（如：中文、日文、韩文等）
   - ✓ 对URL进行编码
   - ✓ 浏览器会自动对url进行编码，我们不需要做任何事情
2. 浏览器自动对url进行编码本质上使用的是js的哪个方法？
   - ✓ encodeURI
3. 工作中什么情况下我们需要手动对url进行编码？使用js的哪个方法？
   - ✓ 我们要把 /、?、=、&、# 等特殊字符当做普通字符时需要手动对其进行编码
   - ✓ encodeURIComponent
4. encodeURI和encodeURIComponent的区别？
   - ✓ encodeURI：只对非ASCII 字符进行编码，不对 /、?、=、&、# 等特殊字符进行编码
   - ✓ encodeURIComponent：对非ASCII 字符和 /、?、=、&、# 等特殊字符都进行编码

目录 Contents

- 跨域（现阶段了解即可）

跨域

学习路径：

① 同源

② 同源策略

③ 跨越

同源

什么是同源？答：如果两个页面的**协议**，**域名**和**端口号**都相同，则两个页面具有相同的源，简称同源

例如，下表给出了相对于 http://www.test.com/index.html 页面的同源检测：

URL	是否同源	原因
http://www.test.com/other.html	是	同源（协议、域名、端口相同）
https://www.test.com/about.html	否	协议不同（http 与 https）
http://blog.test.com/movie.html	否	域名不同（www.test.com 与 blog.test.com）
http://www.test.com:7001/home.html	否	端口不同（默认的 80 端口与 7001 端口）
http://www.test.com:80/main.html	是	同源（协议、域名、端口相同）

同源策略

什么是同源策略？

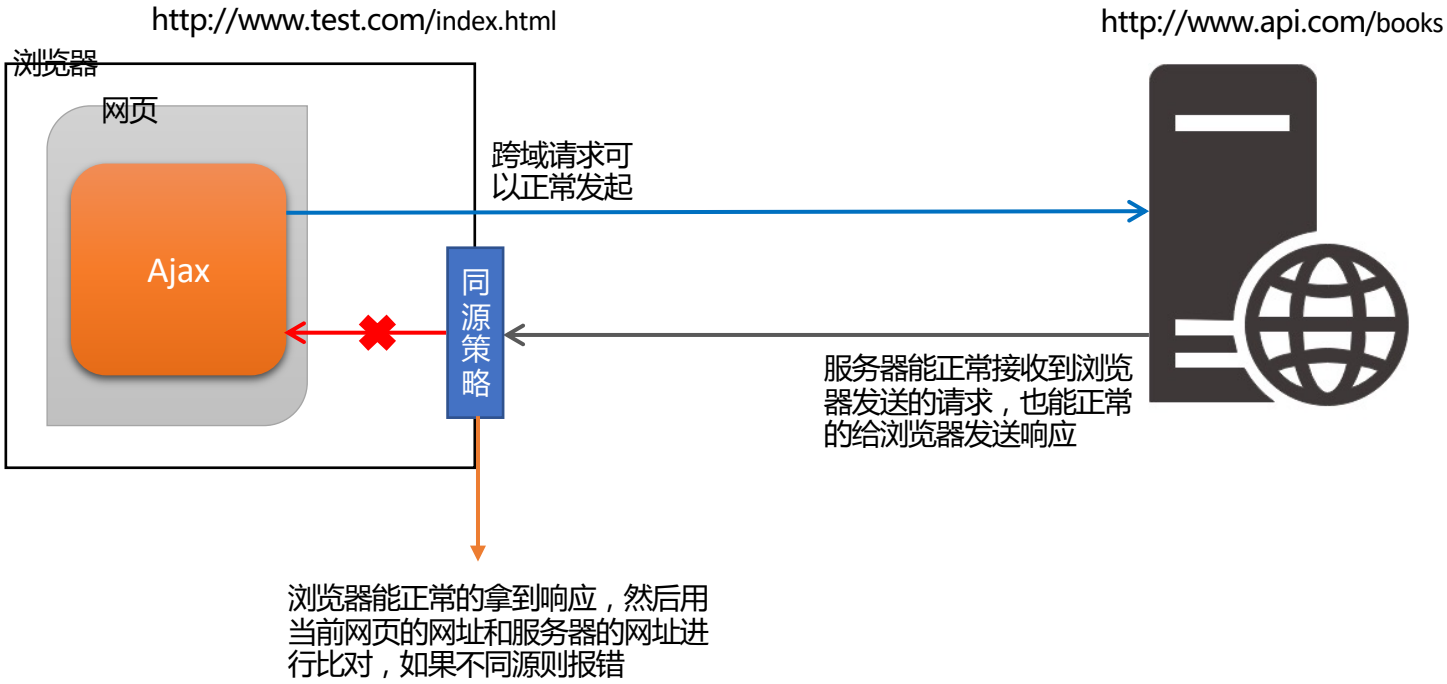
答：浏览器为了安全考虑，所以规定：非同源的两个网站之间不允许相互访问对方的数据。这些数据有：

① 无法读取非同源网页的 **Cookie、LocalStorage 和 IndexedDB**

② 无法获取非同源网页的 **DOM**

③ 无法向非同源地址发送 **Ajax** 请求 *（注意：浏览器的同源策略只限制ajax请求，对<link href="">、<script src="">、<img src="">不做限制）*

ajax请求同源策略的原理



提问

1. 怎么判断两个网址是否是同源？
   - ✓ 协议、域名、端口号都相同则是同源
2. 什么是同源策略？
   - ✓ 浏览器规定：非同源的两个网站之间不允许相互访问对方的数据
3. ajax请求同源策略的原理？
   - ✓ 浏览器在接收到响应时会去比对服务器的网址是否和当前页面的网址是否同源，如果不同源则报错
4. A网址可以请求B网址的js、css、图片等吗？
   - ✓ 可以，因为浏览器的同源策略只限制ajax请求，对<link href="">、<script src="">、<img src="">不做限制
5. qq、微信等软件可以发送http请求吗？他们有同源策略吗？
   - ✓ 可以，没有（只有浏览器有同源策略）
6. 浏览器为什么要弄一个同源策略，目的是什么？
   - ✓ 安全

跨域

- 什么是跨域？答： 通过一些技术手段实现不同源之间进行数据交互（一般专指http请求）
- 有哪些技术手段？*（其实还有很多，但最常见的就是下面这三种）*

1. jsonp *（2010左右最主流，现在已逐渐被淘汰）*
2. **CORS（cross origin sharing source）跨域资源共享** *（IE9-不支持。目前最主流）*
3. 服务器跨域 *（有的公司在用）*

JSONP

- 由来：2005年发布ajax请求，但不支持跨越。程序员们想出了一个临时解决方案，即jsonp。
- 原理：通过 <script src=""> 发送http请求 *（因为浏览器的同源策略只限制ajax请求，对<link href="">、<script src="">、<img src="">不做限制）*
- 缺点：只支持get请求 *（因为 <script src="">、<link href="">、<img src="">只能发送get请求，不能自定义 ）*

jQuery中的JSONP

jQuery 提供的 $.ajax() 函数，除了可以发起真正的 Ajax 数据请求之外，还能够发起 JSONP 数据请求，例如：

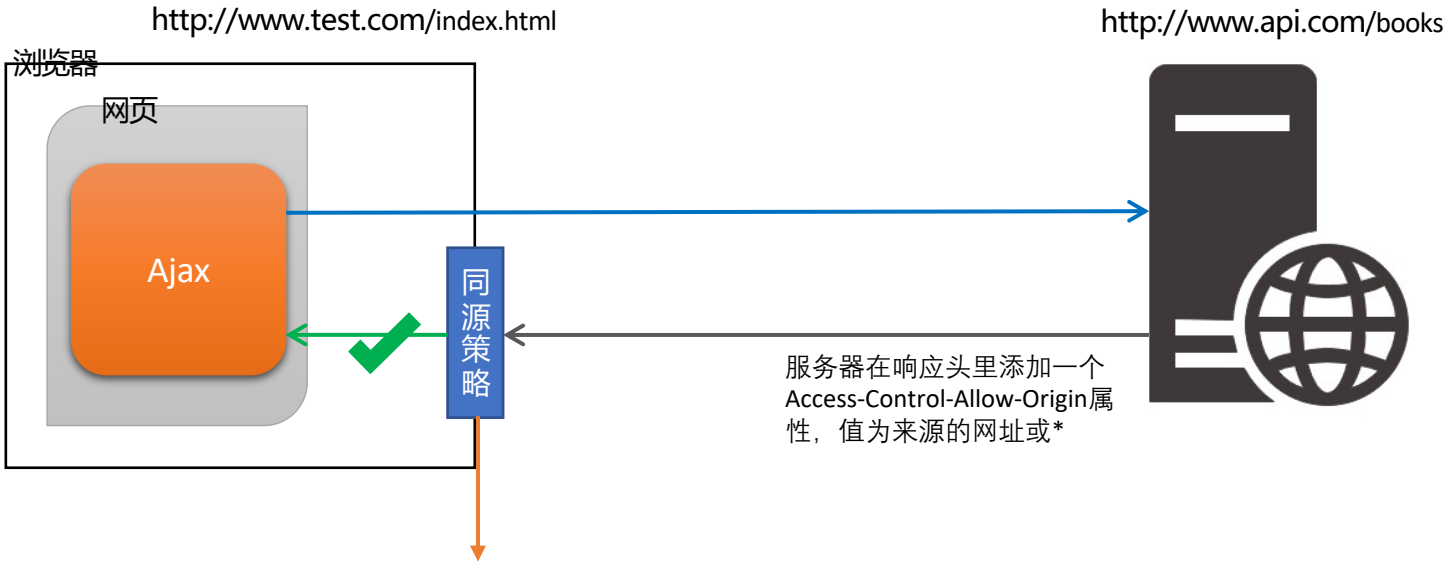
```
$.ajax({
    url: 'http://ajax.frontend.itheima.net:3006/api/jsonp?name=zs&age=20',
    // 如果要使用 $.ajax() 发起 JSONP 请求，必须指定 datatype 为 jsonp
    dataType: 'jsonp',
    success: function(res) {
        console.log(res)
    }
})
```

1. jsonp请求的原理？
2. jsonp的缺点？为什么？
3. jsonp属于ajax请求吗？

CORS

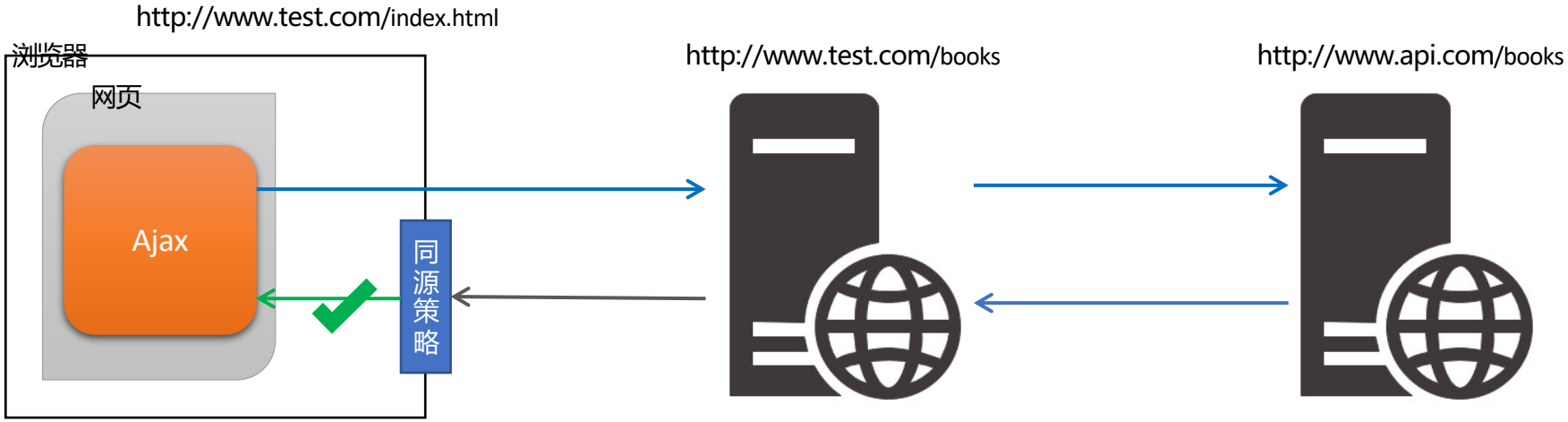
- 操作：完全由后端同学设置，我们前端不做任何设置
- 优点：是W3C官方提供的跨域解决方案，支持get、post、delete等各种请求
- 缺点：IE9-不支持

CORS的原理



1. cors的原理？
2. cors的优点？
3. cors的缺点？

服务器跨域



服务器跨域的原理：同源策略只是浏览器的安全策略，而服务器并没有同源策略限制

1. 服务器跨域的原理？