跨域&CORS

什么是跨域

协议+域名+端口

1 什么是“跨域”?

跨域指的是：浏览器出于安全考虑，不允许一个网页去请求 **不同源（域名/端口/协议不同）** 的资源。

这里的“源”是由三部分组成的：

复制 编辑

```
协议 + 域名 + 端口
```

只有三者都完全相同，才算同源。

只要这三者中的任何一个不同，就会被认为是“跨域”。

前端源	后端源	是否跨域?	理由
http://a.com	http://b.com	是	域名不同
http://a.com	https://a.com	是	协议不同
http://a.com:8080	http://a.com:9090	是	端口号不同
http://a.com	http://a.com	否	完全相同

导出到 Google 表格

实际开发问题

假设你写了一个前端请求接口：

```js
fetch('http://localhost:5000/api/user')
  .then(res => res.json())
  .then(data => console.log(data))
  .catch(err => console.error(err))
```

如果浏览器发现这是跨域请求，而且后端没有允许它访问，你会在浏览器控制台看到类似报错：

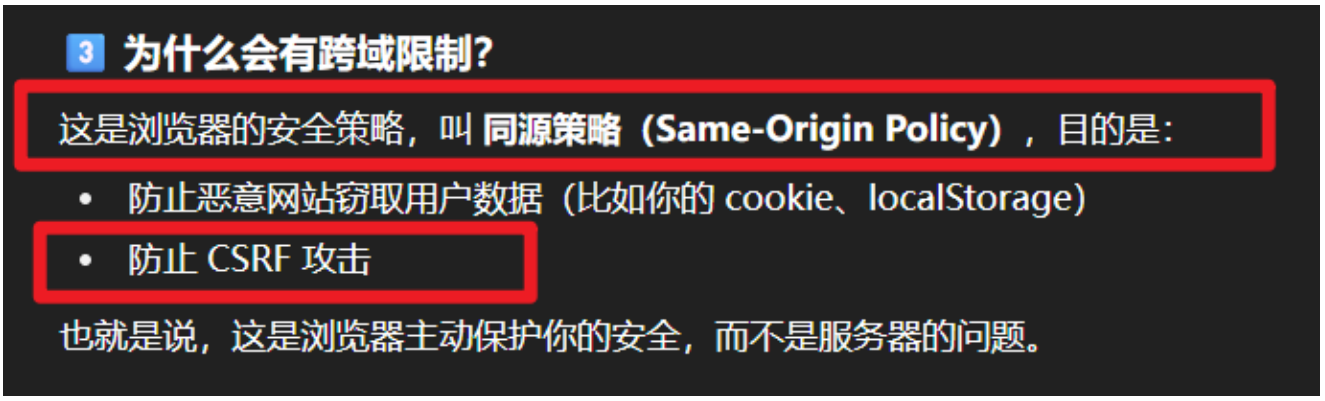
pgsql 复制 编辑

```
Access to fetch at 'http://localhost:5000/api/user' from origin 'http://localhost
```

简单理解就是：**浏览器拒绝访问了跨域的接口。**

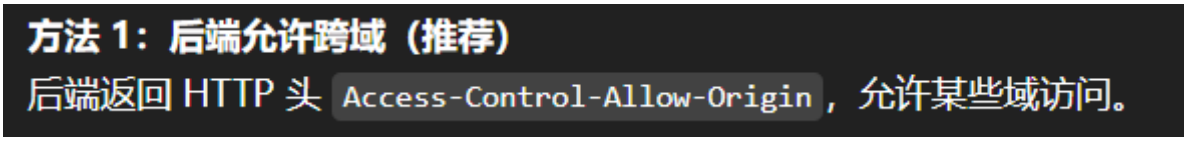
```
Access to fetch at 'http://api.example.com' from origin
'http://localhost:3000' has been blocked by CORS policy: No 'Access-
Control-Allow-Origin' header is present on the requested resource.
```

为什么有跨域限制



解决办法

后端允许跨域



示例：Node.js (Express)

js 复制 编辑

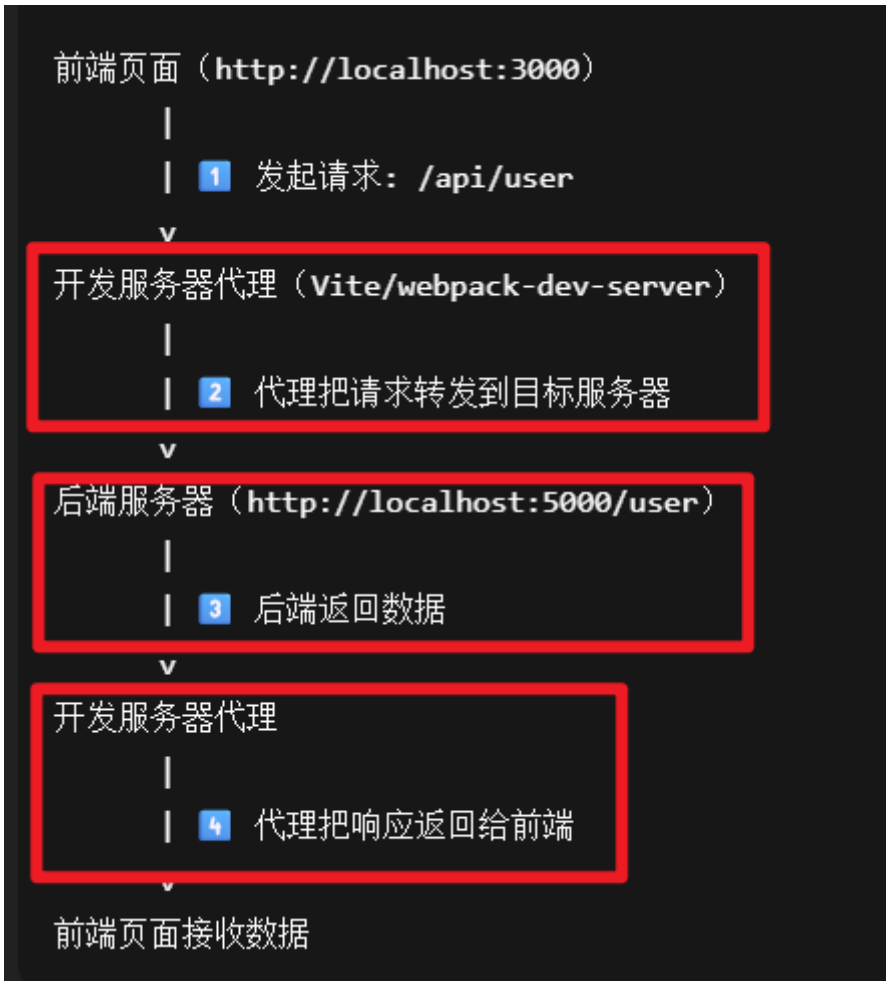
```js
const express = require('express')
const app = express()

// 允许 http://localhost:3000 跨域请求
app.use((req, res, next) => {
  res.setHeader('Access-Control-Allow-Origin', 'http://localhost:3000')
  res.setHeader('Access-Control-Allow-Methods', 'GET,POST,PUT,DELETE')
  res.setHeader('Access-Control-Allow-Headers', 'Content-Type')
  next()
})

app.get('/api/user', (req, res) => {
  res.json({ name: 'Alice' })
})

app.listen(5000)
```

⚠️ 如果你是前端开发，通常需要让后端帮忙加上。

前端使用代理


方法 2：前端使用 代理（开发环境常用）
比如你用 Vue/React 的开发服务器（webpack devServer 或 Vite），可以配置代理，把请求转发到后端：
Vue (vite.config.js)
js
复制
编辑
export default {
server: {
proxy: {
'/api': {
target: 'http://localhost:5000',
changeOrigin: true, // 修改请求头中的 origin
rewrite: path => path.replace(/^\/api/, '')
}
}
}
}
请求 http://localhost:3000/api/user 实际上会被代理到 http://localhost:5000/user，浏览器就不会报跨域。

流程



1. **前端请求** `/api/user`
   浏览器认为这是同源请求（因为和前端服务器同源），所以不会报跨域。
2. **代理转发**
   开发服务器把请求转到实际后端接口（`http://localhost:5000/user`）。
3. **浏览器不拦截**
   因为浏览器只看到同源请求，跨域问题就被绕过了。

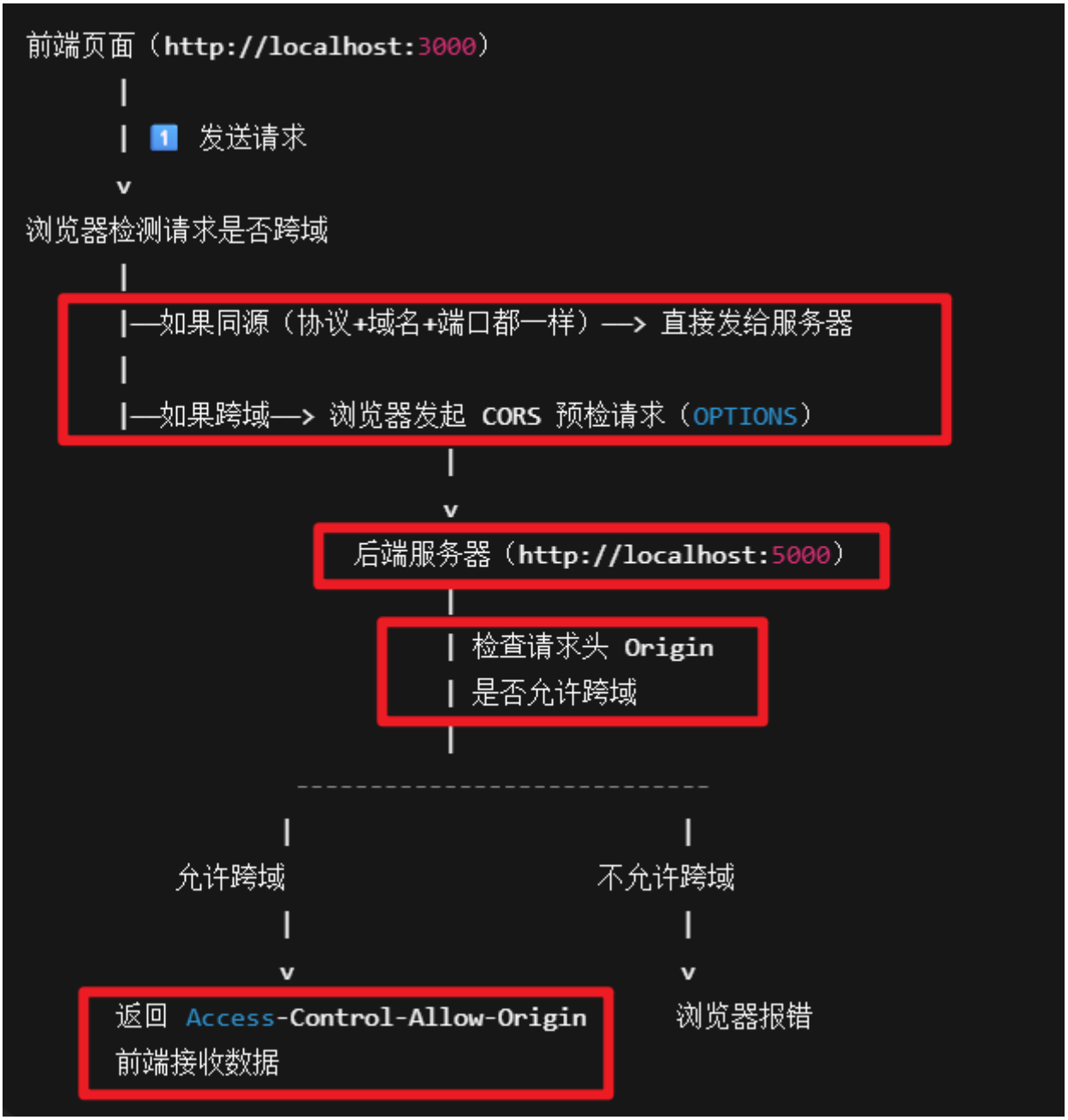
JSONP 方法

方法 3：JSONP（老方法，只能 GET 请求）
在 `<script>` 标签中请求接口，把数据返回包成函数调用。
现代开发很少用，除非接口特别老。

CORS插件

方法 4：使用 CORS 插件（开发临时解决方案）
浏览器插件可以临时绕过跨域限制，但**只适合开发调试**，生产不安全。

流程



1. **浏览器拦截**：跨域问题是浏览器报的，不是前端代码写错。
2. **预检请求**：当请求是复杂请求（比如 `POST` 带自定义 header）时，浏览器会先发一个 `OPTIONS` 请求确认是否允许。
3. **后端允许跨域**：通过返回 `Access-Control-Allow-Origin` 来告知浏览器可以访问。

总结

✅ **总结**

- **跨域问题**：浏览器安全策略阻止不同源的网页访问资源。
- **出现场景**：前端请求后端接口，浏览器报 `CORS policy` 错误。
- **解决办法**：
  1. 后端设置 `Access-Control-Allow-Origin`
  2. 前端开发环境使用代理
  3. 老方法：JSONP
  4. 临时调试：CORS 插件