Node.js 模块化

目录

Contents

什么是模块化？

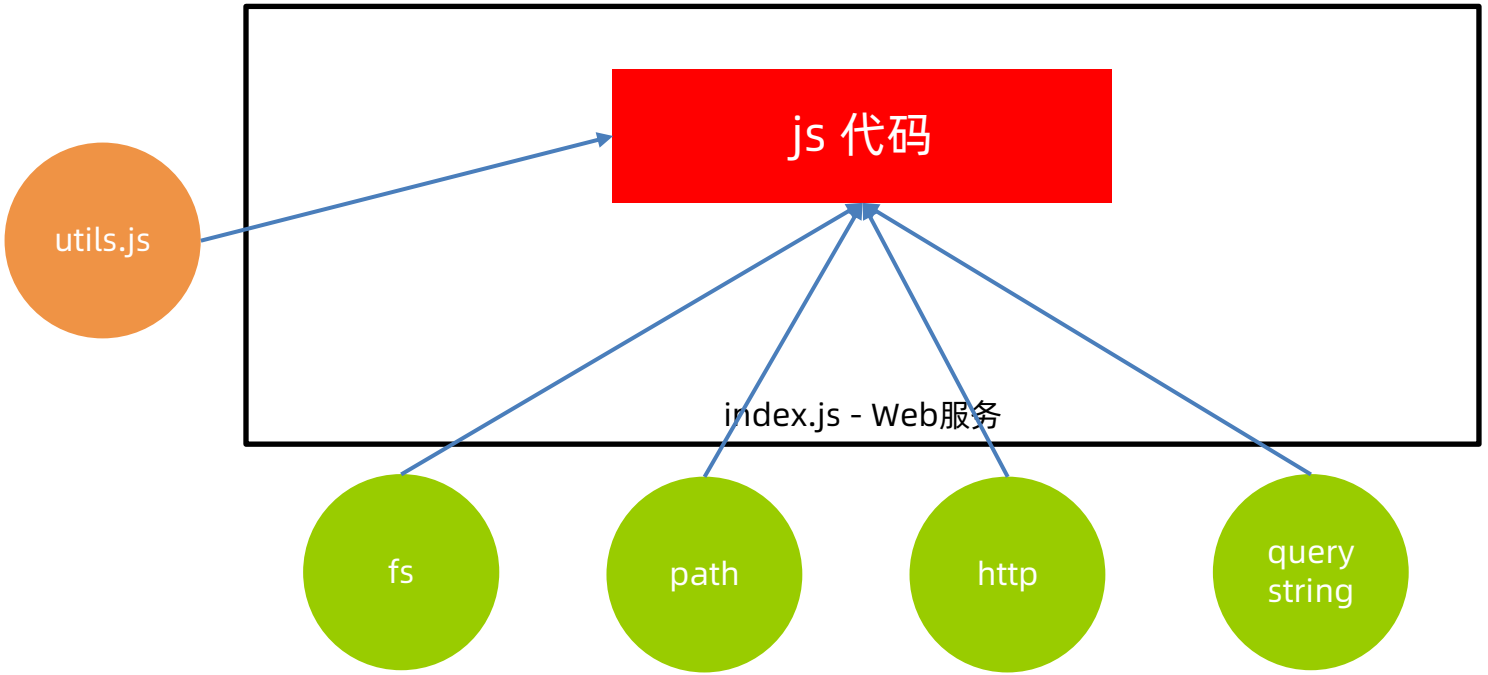
定义：

> CommonJS 模块是为 Node.js 打包 JavaScript 代码的原始方式。 Node.js 还支持浏览器和其他 JavaScript 运行时使用的 ECMAScript 模块 标准。
>
> 在 Node.js 中，每个文件都被视为一个单独的模块。

概念：项目是由很多个模块文件组成的

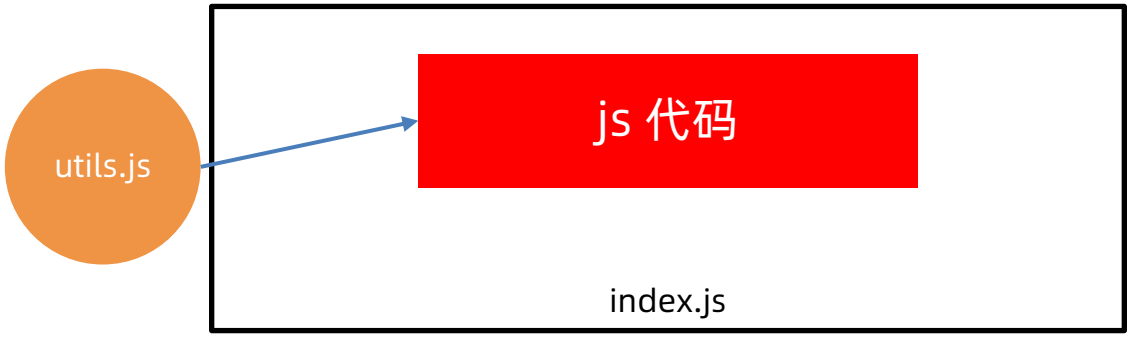
好处：提高代码复用性，按需加载，独立作用域

使用：需要标准语法导出和导入进行使用



CommonJS 标准

需求：定义 utils.js 模块，封装基地址和求数组总和的函数



使用：

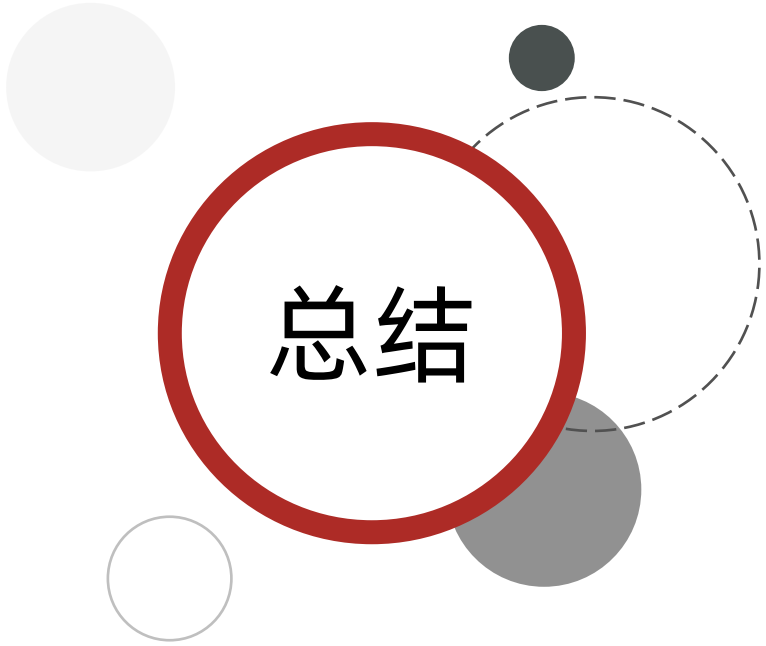
1. 导出：module.exports = {}
2. 导入：require('模块名或路径')

```
const baseURL = 'http://hmajax.itheima.net'
const getArraySum = arr => arr.reduce((sum, val) => sum += val, 0)

module.exports = {
  对外属性名1: baseURL,
  对外属性名2: getArraySum
}
```

模块名或路径：

- ✓ 内置模块：直接写名字（例如：fs，path，http）
- ✓ 自定义模块：写模块文件路径（例如：./utils.js）

```
const obj = require('模块名或路径')
// obj 就等于 module.exports 导出的对象
```



1. Node.js 中什么是模块化？
   - 每个文件都是独立的模块
2. 模块之间如何联系呢？
   - 使用特定语法，导出和导入使用
3. CommonJS 标准规定如何导出和导入模块呢？
   - 导出：module.exports = {}
   - 导入：require('模块名或路径')
4. 模块名/路径如何选择？
   - 内置模块，直接写名字。例如：fs，path，http等
   - 自定义模块，写模块文件路径。例如：./utils.js

ECMAScript 标准 - 默认导出和导入

需求：封装并导出基地址和求数组元素和的函数

```
const baseURL = 'http://hmajax.itheima.net'
const getArraySum = arr => arr.reduce((sum, val) => sum += val, 0)

export default {
  对外属性名1: baseURL,
  对外属性名2: getArraySum
}
```

默认标准使用：

1. 导出：export default {}
2. 导入：import 变量名 from '模块名或路径'

```
import obj from '模块名或路径'
// obj 就等于 export default 导出的对象
```

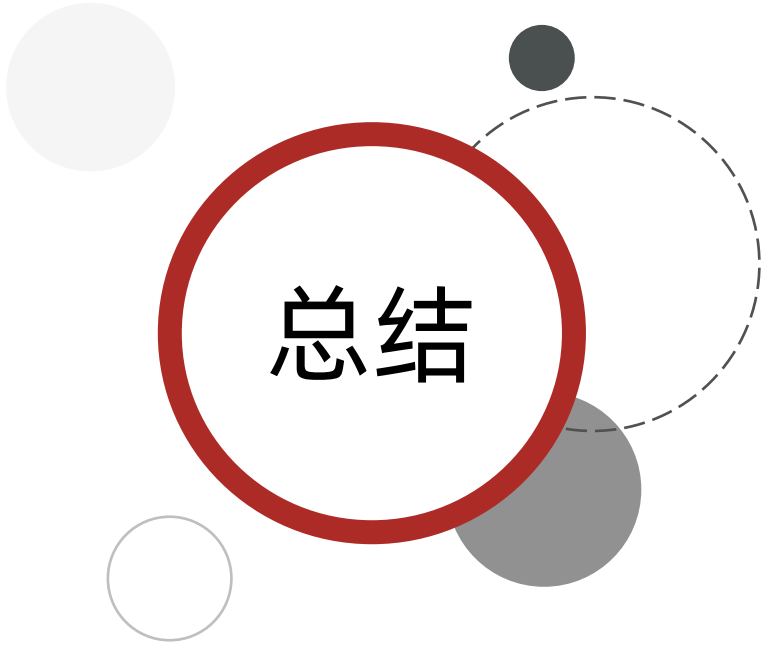
注意：Node.js 默认支持 CommonJS 标准语法

如需使用 ECMAScript 标准语法，在运行模块所在文件夹新建 package.json 文件，并设置 { "type" : "module" }

package.json

D: > 备课代码 > 2_node_3天 > Node_代码 > Day03_we

```
1  { "type": "module" }
```



1. ECMAScript 标准规定如何默认导出和导入模块呢？
   - 导出：export default {}
   - 导入：import 变量名 from '模块名或路径'
2. 如何让 Node.js 切换模块标准为 ECMAScript？
   - 运行模块所在文件夹，新建 package.json 并设置
   - { "type" : "module" }

ECMAScript 标准 - 命名导出和导入

需求：封装并导出基地址和求数组元素和的函数

```
export const baseURL = 'http://hmajax.itheima.net'
export const getArraySum = arr => arr.reduce((sum, val) => sum += val, 0)
```

命名标准使用：

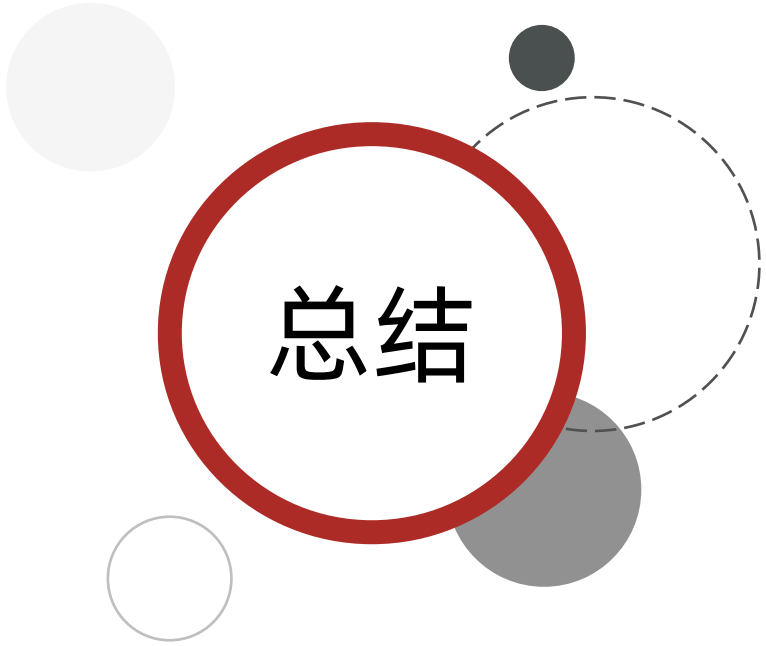
1. 导出：export 修饰定义语句
2. 导入：import { 同名变量 } from '模块名或路径'

```
import { baseURL, getArraySum } from '模块名或路径'
// baseURL 和 getArraySum 是变量，值为模块内命名导出的同名变量的值
```

如何选择：

按需加载，使用命名导出和导入

全部加载，使用默认导出和导入



1. Node.js 支持哪 2 种模块化标准？
   - CommonJS 标准语法（默认）
   - ECMAScript 标准语法
2. ECMAScript 标准，命名导出和导入的语法？
   - 导出：export 修饰定义的语句
   - 导入：import { 同名变量 } from '模块名或路径'
3. ECMAScript 标准，默认导出和导入的语法？
   - 导出：export default {}
   - 导入：import 变量名 from '模块名或路径'

包的概念

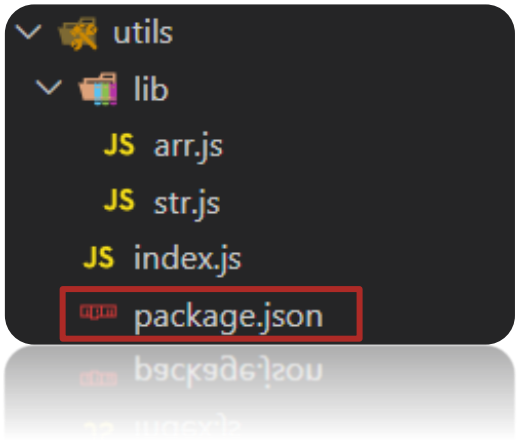
包：将模块，代码，其他资料聚合成一个文件夹

包分类：

- ✓ 项目包：主要用于编写项目和业务逻辑
- ✓ 软件包：封装工具和方法进行使用

package.json > ...

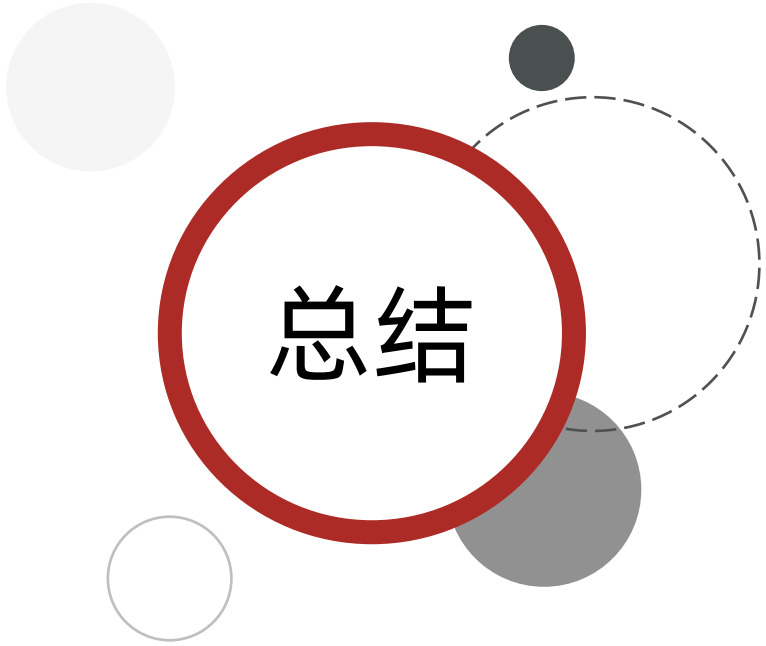
```
{
  "name": "cz_utils",          软件包名称
  "version": "1.0.0",          软件包当前版本
  "description": "一个数组和字符串常用工具方法的包",   软件包简短描述
  "main": "index.js",          软件包入口点
  "author": "itheima",         软件包作者
  "license": "MIT"             软件包许可证（商用后可以用作者名字宣传）
}
```



要求：根目录中，必须有 package.json 文件（记录包的清单信息）

注意：导入软件包时，引入的默认是 index.js 模块文件 / main 属性指定的模块文件

需求：封装数组求和函数的模块，判断用户名和密码长度函数的模块，形成成一个软件包



1. 什么是包？
   - 将模块，代码，其他资料聚合成的文件夹
2. 包分为哪 2 类呢？
   - 项目包：编写项目代码的文件夹
   - 软件包：封装工具和方法供开发者使用
3. package.json 文件的作用？
   - 记录软件包的名字，作者，入口文件等信息
4. 导入一个包文件夹的时候，导入的是哪个文件？
   - 默认 index.js 文件，或者 main 属性指定的文件

npm - 软件包管理器

定义：

npm 简介

npm 是 Node.js 标准的软件包管理器。

在 2017 年 1 月时，npm 仓库中就已有超过 350000 个软件包，这使其成为世界上最大的单一语言代码仓库，并且可以确定几乎有可用于一切的软件包。

它起初是作为下载和管理 Node.js 包依赖的方式，但其现在也已成为前端 JavaScript 中使用的工具。

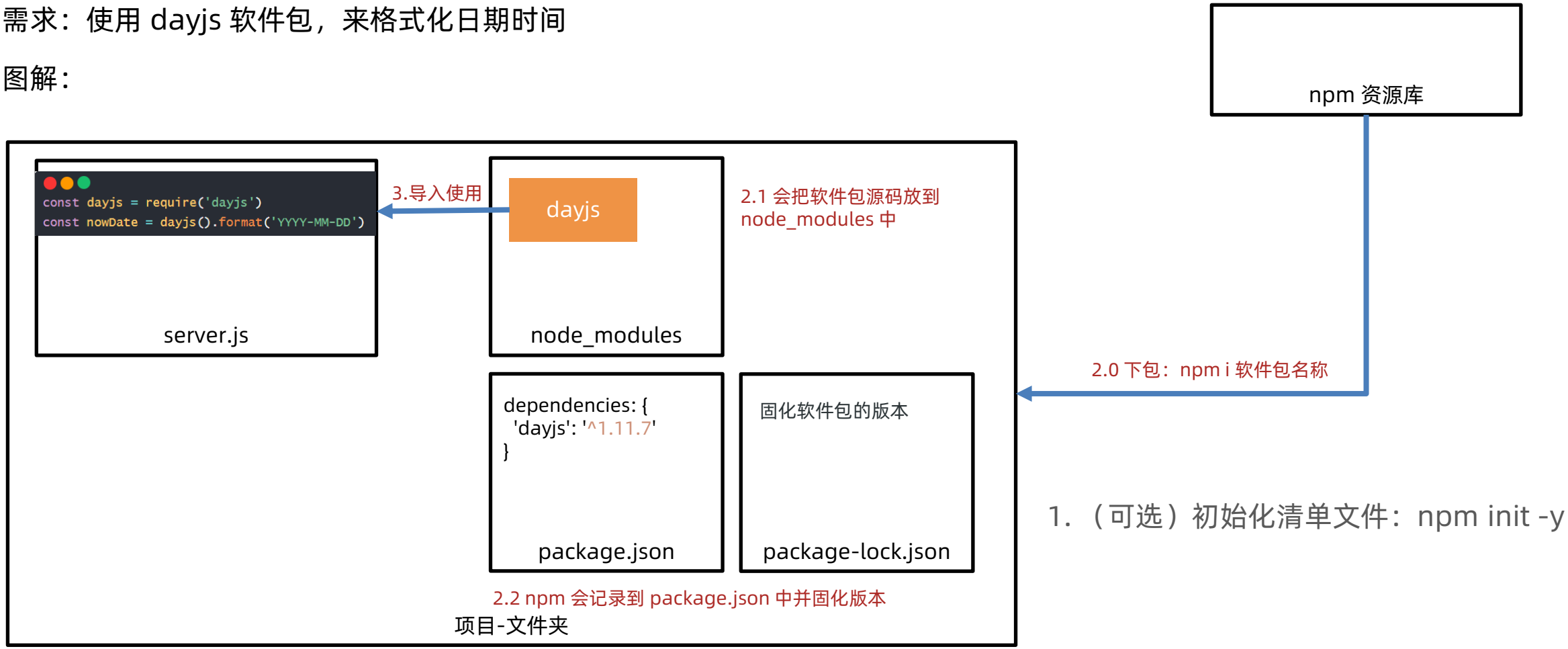
使用：

1. 初始化清单文件 ： npm init -y（得到 package.json 文件，有则略过此命令）
2. 下载软件包 ： npm i 软件包名称
3. 使用软件包

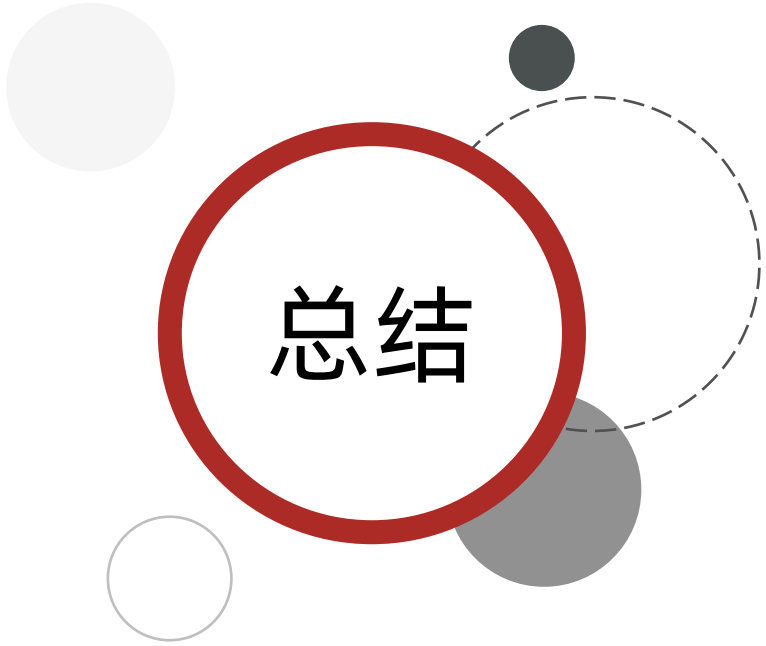
npm - 软件包管理器

需求：使用 dayjs 软件包，来格式化日期时间

图解：



1.（可选）初始化清单文件：npm init -y



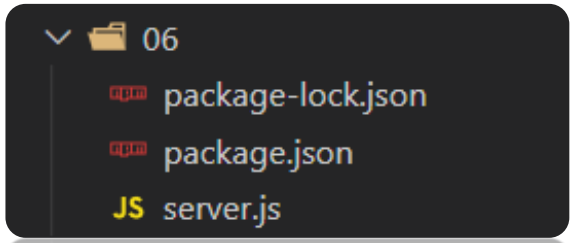
1. npm 软件包管理器作用？
   - 下载软件包以及管理版本
2. 初始化项目清单文件 package.json 命令？
   - npm init -y
3. 下载软件包的命令？
   - npm i 软件包名字
4. 下载的包会存放在哪里？
   - 当前项目下的 node_modules 中，并记录在 package.json 中

npm - 安装所有依赖

问题：项目中不包含 node_modules，能否正常运行？

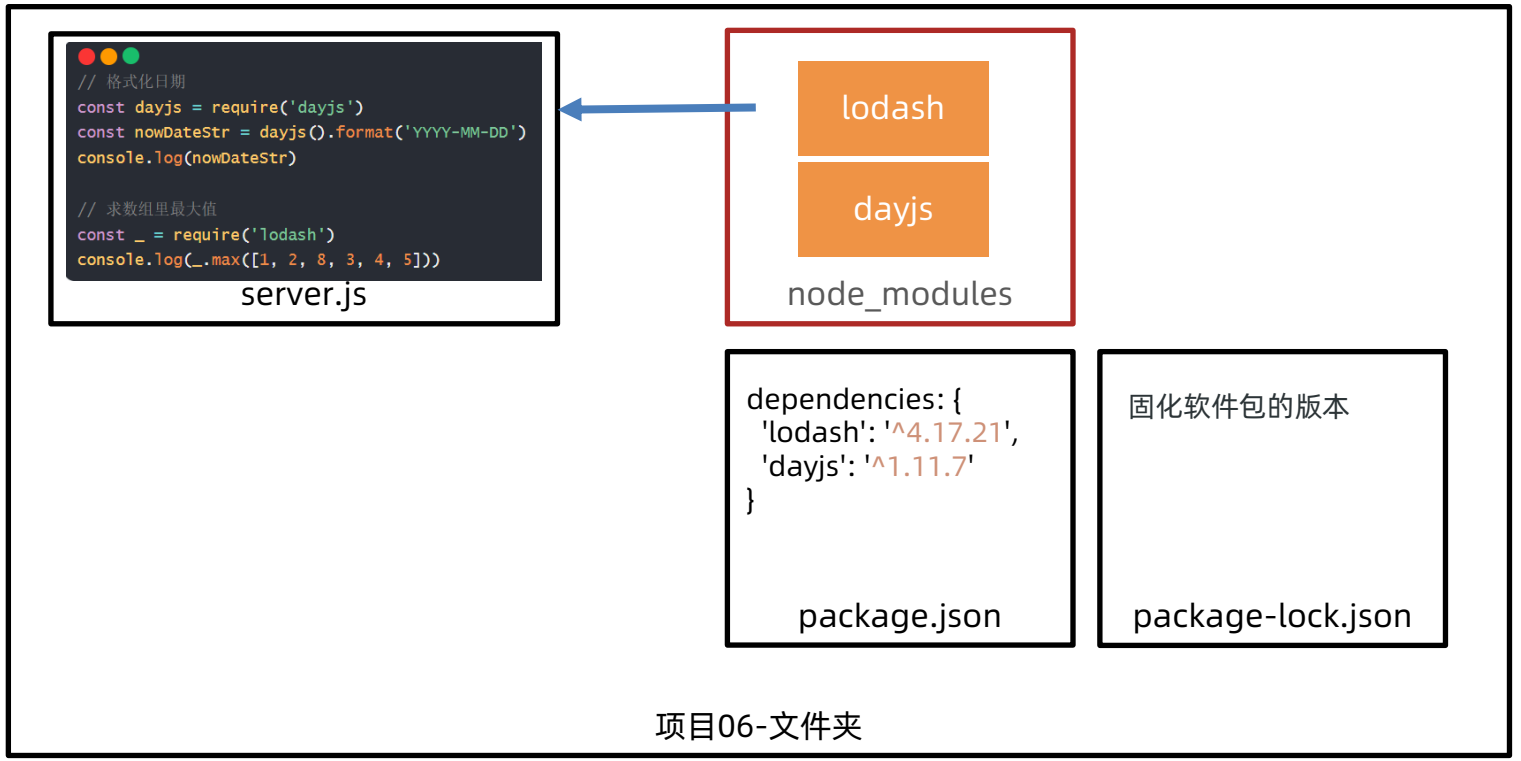
答案：不能，缺少依赖的本地软件包

原因：因为，自己用 npm 下载依赖比磁盘传递拷贝要快得多



解决：项目终端输入命令：npm i

下载 package.json 中记录的所有软件包



总结

1. 当项目中只有 package.json 没有 node_modules 怎么办？
   - 当前项目下，执行 npm i 安装所有依赖软件包
2. 为什么 node_modules 不进行传递？
   - 因为用 npm 下载比磁盘传递要快

npm - 全局软件包 nodemon

软件包区别：

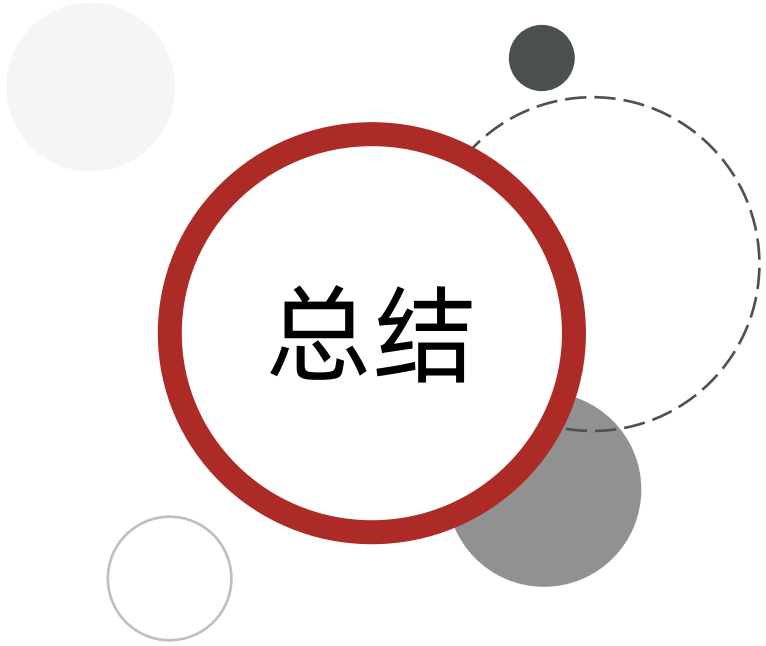
- 本地软件包：当前项目内使用，封装属性和方法，存在于 node_modules
- 全局软件包：本机所有项目使用，封装命令和工具，存在于系统设置的位置

nodemon 作用：替代 node 命令，检测代码更改，自动重启程序

使用：

1. 安装：npm i nodemon -g（-g 代表安装到全局环境中）
2. 运行：nodemon 待执行的目标 js 文件

需求：启动准备好的项目，修改代码保存后，观察自动重启应用程序



1. 本地软件包和全局软件包区别？
   - 本地软件包，作用在当前项目，封装属性和方法
   - 全局软件包，本机所有项目使用，封装命令和工具
2. nodemon 作用？
   - 替代 node 命令，检测代码更改，自动重启程序
3. nodemon 怎么用？
   - 先确保安装 npm i nodemon -g
   - 使用 nodemon 执行目标 js 文件

Node.js 总结

Node.js 模块化：

概念：每个文件当做一个模块，独立作用域，按需加载

使用：采用特定的标准语法导出和导入进行使用

CommonJS 标准

	导出	导入
语法	module.exports = {}	require('模块名或路径')

ECMAScript 标准

	导出	导入
默认	export default {}	import 变量名 from '模块名或路径'
命名	export 修饰定义语句	import { 同名变量 } from '模块名或路径'

CommonJS 标准：一般应用在 Node.js 项目环境中

ECMAScript 标准：一般应用在前端工程化项目中

Node.js 总结

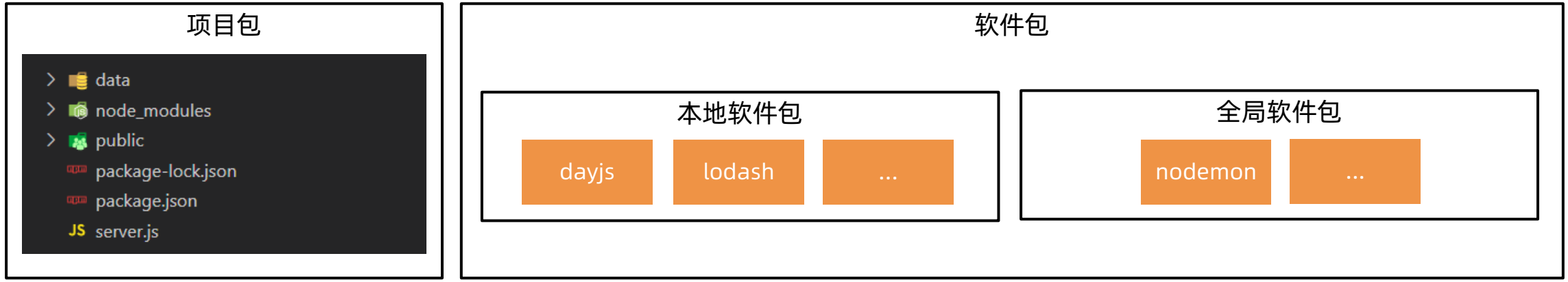
Node.js 包：

概念：把模块文件，代码文件，其他资料聚合成一个文件夹

项目包：编写项目需求和业务逻辑的文件夹

软件包：封装工具和方法进行使用的文件夹（一般使用 npm 管理）

- ✓ 本地软件包：作用在当前项目，一般封装的属性/方法，供项目调用编写业务需求
- ✓ 全局软件包：作用在所有项目，一般封装的命令/工具，支撑项目运行



Node.js 总结

常用命令：

功能	命令
执行 js 文件	node xxx
初始化 package.json	npm init -y
下载本地软件包	npm i 软件包名
下载全局软件包	npm i 软件包名 -g
删除软件包	npm uni 软件包名

Express - 框架

定义：

概念：使用 express 本地软件包，快速搭建 Web 服务（基于 http 模块）

功能：

- ✓ 提供数据接口
- ✓ 提供网页资源等

Express - 框架

使用：

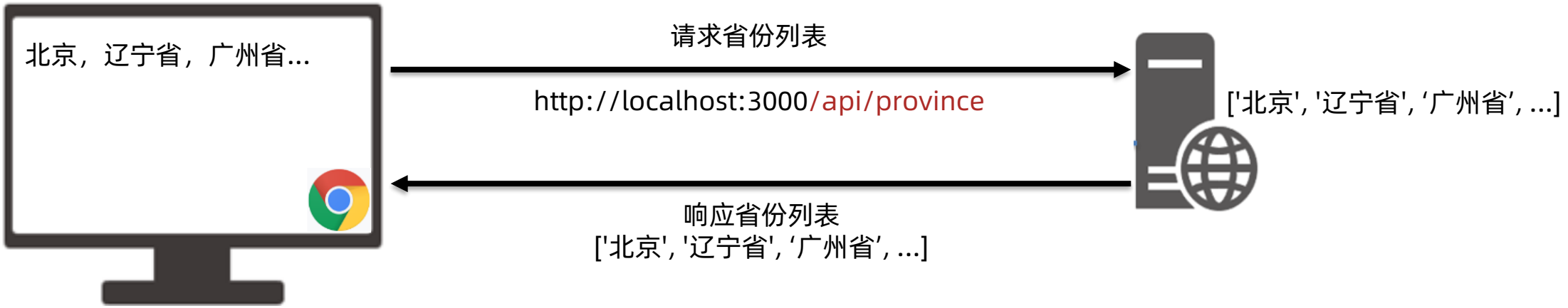
1. 下载 express 软件包
2. 导入 express 创建 Web 服务对象
3. 监听请求方法和请求路径
4. 对其他请求方法和请求路径，默认返回 404 提示
5. 监听端口号，启动 Web 服务，在浏览器请求测试

```
// 基于 express 创建 Web 服务对象
const express = require('express')
const server = express()

// 监听 get 请求和路径为 / 时触发回调函数，使用 send 响应内容
server.get('/', (req, res) => {
  res.send('你好，欢迎使用 Express')
})

// 监听所有请求方法和任意路径，统一返回 404 提示
server.all('*', (req, res) => {
  res.status(404)
  res.send('你要访问的资源路径不存在')
})

// 监听端口号
server.listen(3000, () => {
  console.log('Web 服务启动了')
})
```

案例 获取省份列表-接口开发

需求：基于 express，开发提供省份列表数据的接口

步骤：监听 get 请求方法的 /api/province 路径，并读取 province.json 里省份数据返回给请求方




```
server.get('/api/province', (req, res) => {
  fs.readFile(path.join(__dirname, 'data/province.json'), (err, data) => {
    res.send(data.toString())
  })
})
```

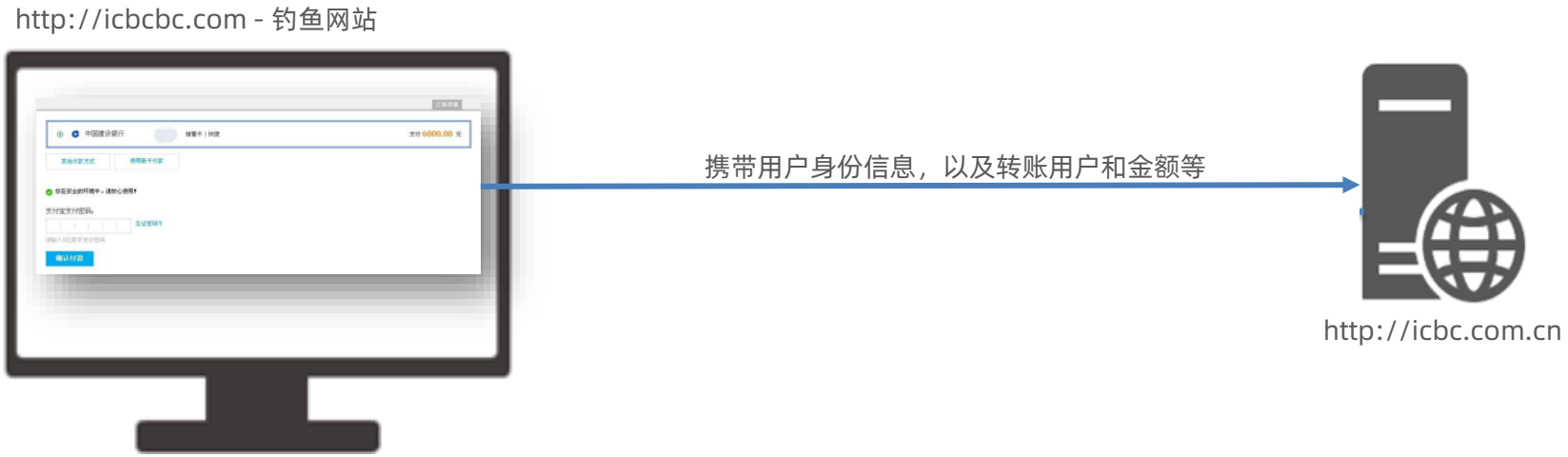
浏览器的同源策略

同源策略：

同源策略是一个重要的安全策略，它用于限制一个源的文档或者它加载的脚本如何能与另一个源的资源进行交互。

它能帮助阻隔恶意文档，减少可能被攻击的媒介。例如，它可以防止互联网上的恶意网站在浏览器中运行 JS 脚本，从第三方网络邮件服务（用户已登录）或公司内网（因没有公共 IP 地址而受到保护，不会被攻击者直接访问）读取数据，并将这些数据转发给攻击者。

例如：被钓鱼网站收集信息，使用 AJAX 发起恶意请求，传递转账信息给银行服务器



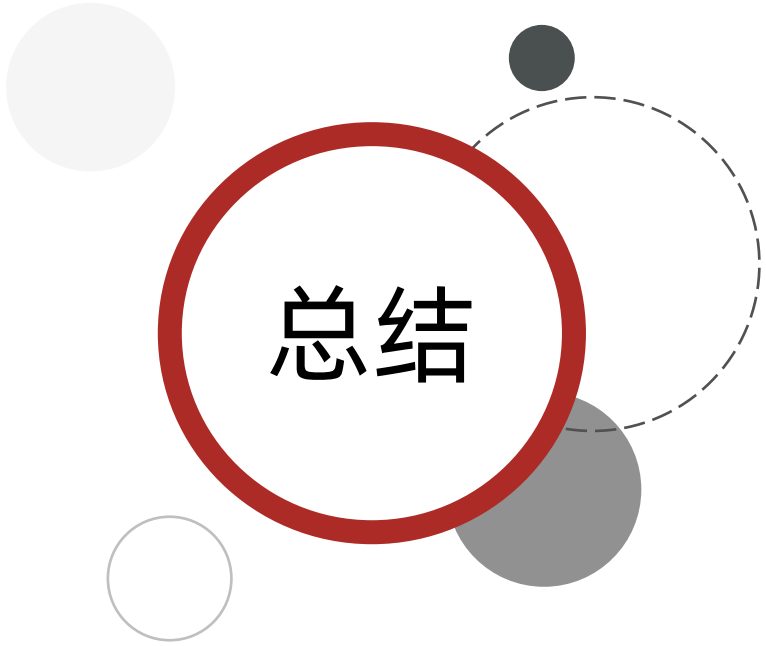
作用：保护浏览器中网站的安全，限制 AJAX 只能向同源 URL 发起请求

浏览器的同源策略

源：

> Web 内容的**源**由用于访问它的 URL 的*方案*（协议）、*主机名*（域名）和*端口*定义。只有当协议、主机和端口都匹配时，两个对象才具有相同的源。

同源：网页加载时所在源，和 AJAX 请求时的源（协议，域名，端口号）全部相同即为同源



1. 什么是同源策略？
   - 限制一个源对另外一个源资源交互
2. 同源策略限制的是？
   - 限制 AJAX 只能访问同源 URL
3. 什么情况下是同源？
   - 网页加载时所在源，和 AJAX 请求时的源（协议，域名，端口号）全部相同即为同源

跨域问题

跨域：从一个源的文档/脚本，加载另一个源的资源就产生了跨域

例如：网页所在源和 AJAX 访问的源（协议，域名，端口）有一个不同，就发生了跨域访问，请求响应是失败的


```
Access to XMLHttpRequest at 'http://localhost:3000/' from origin 'http://localhost:5500' has    index.html:1
been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
```

需求：在 LiveServer 的 Web 服务启动网页，用 AJAX 访问本机 Web 服务提供的省份列表接口，体验下跨域问题

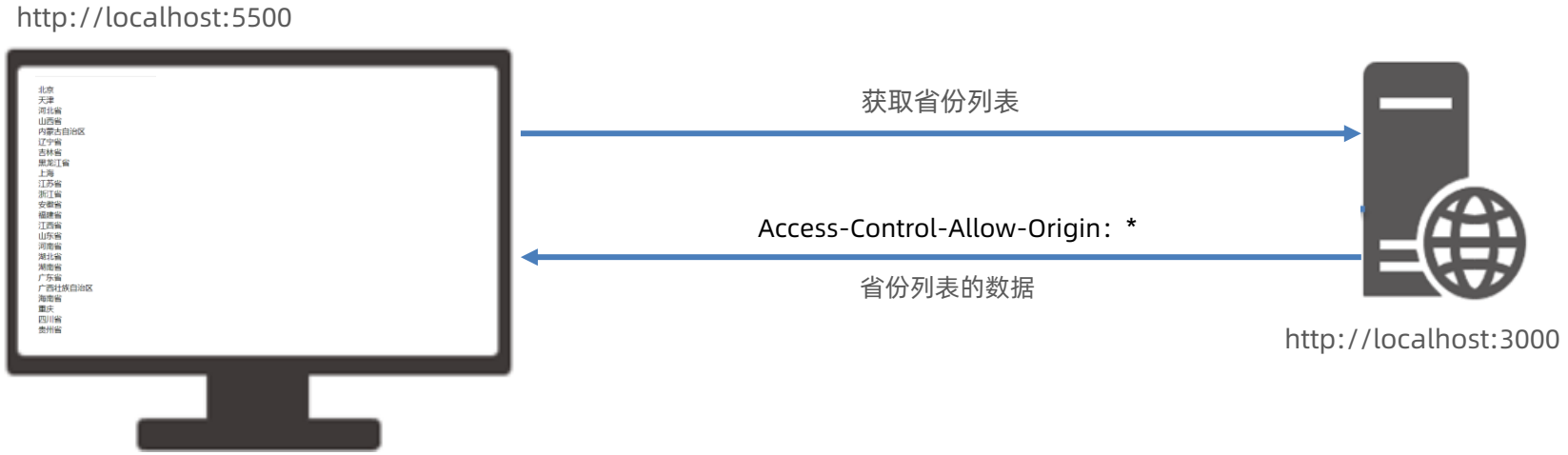
总结

1. 什么是跨域访问？

- 网页所在源和 AJAX 请求的源（协议，域名，端口号）不完全相同，就发生跨域访问

跨域问题 - 解决方案1 - CORS

目标：前后端分离的项目，前端和后端不在一个源，还要保证数据通信



解决：采用 CORS （跨域资源共享），一种基于 HTTP 头的机制，该机制通过允许服务器标示除了它自己以外的其他源（域、协议或端口），使得浏览器允许这些源访问加载自己的资源

步骤：

- ✓ 服务器端：设置 Access-Control-Allow-Origin 响应头字段，允许除了它自己以外的源来访问自己的资源
- ✓ 前端：正常发起 AJAX 请求，无需额外操作

跨域问题 - 解决方案1 - CORS

步骤：

1. 下载 cors 本地软件包
2. 导入 cors 函数
3. 使用 server.use() 给 Web 服务添加插件功能
4. 把 cors 函数调用传入给 Web 服务，启动测试

```
// 1. 下载 cors 本地软件包（专门给响应头设置跨域资源共享）
// 2. 引入 cors 并调用传入 server.use() 给 Web 服务注册插件
// 原理：会给响应头加一个Access-Control-Allow-Origin: *
// 允许哪些源来跨域访问服务器上资源，*代表：该资源可以被任意外源访问。
const cors = require('cors')
server.use(cors())

server.get('/api/province', (req, res) => {
  fs.readFile(path.join(__dirname, 'data/province.json'), (err, data) => {
    res.send(data.toString())
  })
})
})
```

总结

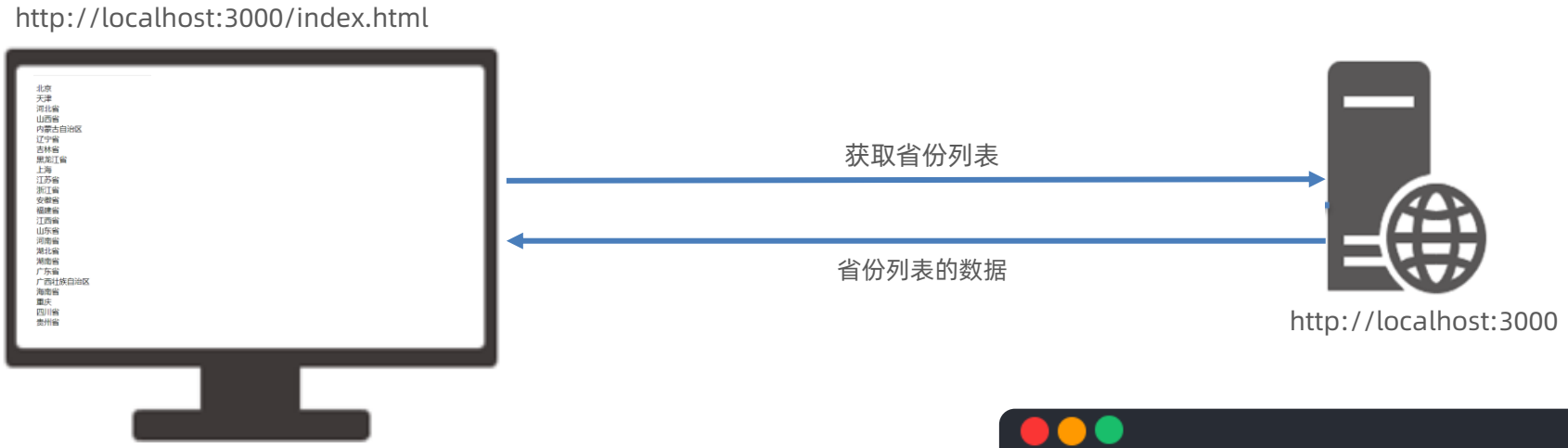
1. 为什么要解决跨域问题？
   - 因为前后端分离的项目，不在同一个源去开发项目
   - 需要保证数据之间通信
2. 跨域问题如何解决？
   - 让后端开启 CORS 跨域资源共享
   - 在响应头设置 Access-Control-Allow-Origin: *

跨域问题 - 解决方案2 - 同源访问

目标：开发环境用 cors，上线部署关闭 cors，并采用同源访问方式

做法：让后端 Web 服务既可以提供数据接口，也可以返回网页资源

好处：安全，后端的接口不允许非同源来访问




```
// 给 Web 服务注册静态网站访问功能：
// public 文件夹下的资源都可以通过http://localhost:3000/ 往下拼接去访问
server.use(express.static('./public'))
```

总结

1. CORS 只适用于什么阶段的项目？
   - 本地开发阶段项目
2. 项目上线，如何解决跨域问题？
   - 把前端项目和后端项目部署到同一个源下访问

传智教育旗下高端IT教育品牌