

硅谷课堂第四天-前端基础知识

硅谷课堂第四天-前端基础知识

一、NPM

1、NPM 简介

- 1.1、什么是 NPM
- 1.2、NPM 工具的安装位置

2、使用 npm 管理项目

- 2.1、创建文件夹 npm
- 2.2、项目初始化
- 2.3、修改 npm 镜像
- 2.4、npm install 命令的使用
- 2.5、其它命令

二、模块化开发（一）

1、模块化简介

- 1.1、模块化产生的背景
- 1.2、什么是模块化开发

2、ES5 模块化

- 2.1、创建“module”文件夹
- 2.2、导出模块
- 2.3、导入模块
- 2.4、运行程序

三、模块化开发（二）

1、ES6 模块化写法（一）

- 1.1、导出模块
- 1.2、导入模块
- 1.3、安装 Babel
- 1.4、配置 .babelrc
- 1.5、安装转码器
- 1.6、转码

- 1.7、运行程序

2、ES6 模块化写法（二）

- 2.1、导出模块
- 2.2、导入模块
- 2.3、转码
- 2.4、运行程序

四、搭建项目前端环境

- 1、vue-admin-template 模板
- 2、搭建环境

3、修改登录功能

- 3.1、创建登录接口
- 3.2、修改登录前端
 - 3.2.1、修改接口路径
 - 3.2.2、修改 js 文件

五、跨域问题

- 1、什么是跨域
- 2、配置

一、NPM

1、NPM 简介

1.1、什么是 NPM

NPM 全称 Node Package Manager，是 Node.js 包管理工具，是全球最大的模块生态系统，里面所有的模块都是开源免费的；也是 Node.js 的包管理工具，相当于前端的 Maven。

1.2、NPM 工具的安裝位置

我们通过 npm 可以很方便地下载 js 库，管理前端工程。

Node.js 默认安装的 npm 包和工具的位置：Node.js 目录\node_modules

- 在这个目录下你可以看见 npm 目录，npm 本身就是被 NPM 包管理器管理的一个工具，说明 Node.js 已经集成了 npm 工具

```
# 在命令提示符输入 npm -v 可查看当前npm版本  
npm -v
```

2、使用 npm 管理项目

2.1、创建文件夹 npm

2.2、项目初始化

```
# 建立一个空文件夹，在命令提示符进入该文件夹，执行命令初始化  
npm init  
# 按照提示输入相关信息，如果是用默认值则直接回车即可。  
# name: 项目名称  
# version: 项目版本号  
# description: 项目描述  
# keywords: {Array}关键词，便于用户搜索到我们的项目  
# 最后会生成package.json文件，这个是包的配置文件，相当于maven的pom.xml  
# 我们之后也可以根据需要进行修改。
```

```
# 如果想直接生成 package.json 文件，那么可以使用命令  
npm init -y
```

2.3、修改 npm 镜像

NPM 官方的管理的包都是从 <http://npmjs.com> 下载的，但是这个网站在国内速度很慢。

这里推荐使用淘宝 NPM 镜像 <http://npm.taobao.org/>，淘宝 NPM 镜像是一个完整 npmjs.com 镜像，同步频率目前为 10 分钟一次，以保证尽量与官方服务同步。

设置镜像地址：

```
# 经过下面的配置，以后所有的 npm install 都会经过淘宝的镜像地址下载  
npm config set registry https://registry.npm.taobao.org  
# 查看npm配置信息  
npm config list
```

2.4、npm install 命令的使用

```
# 使用 npm install 安装依赖包的最新版
# 模块安装的位置：项目目录\node_modules
# 安装会自动在项目目录下添加 package-lock.json文件，这个文件帮助锁定安装包的版本
# 同时package.json 文件中，依赖包会被添加到dependencies节点下，类似maven中的 <dependencies>
npm install jquery
# npm管理的项目在备份和传输的时候一般不携带node_modules文件夹
npm install # 根据package.json中的配置下载依赖，初始化项目
# 如果安装时想指定特定的版本
npm install jquery@2.1.x
# 局部安装
# devDependencies节点：开发时的依赖包，项目打包到生产环境的时候不包含的依赖
# 使用 -D参数将依赖添加到devDependencies节点
npm install --save-dev eslint
# 或
npm install -D eslint
# 全局安装
# Node.js全局安装的npm包和工具的位置：用户目录\AppData\Roaming\npm\node_modules
# 一些命令行工具常使用全局安装的方式
npm install -g webpack
# 或
npm install --global webpack
```

2.5、其它命令

```
# 更新包（更新到最新版本）
npm update 包名
# 全局更新
npm update -g 包名
# 卸载包
npm uninstall 包名
# 全局卸载
npm uninstall -g 包名
```

二、模块化开发（一）

1、模块化简介

1.1、模块化产生的背景

随着网站逐渐变成"互联网应用程序"，嵌入网页的 Javascript 代码越来越庞大，越来越复杂。



Javascript 模块化编程，已经成为一个迫切的需求。理想情况下，开发者只需要实现核心的业务逻辑，其他都可以加载别人已经写好的模块。

但是，Javascript 不是一种模块化编程语言，它不支持"类"（class），包（package）等概念，更遑论"模块"（module）了。

1.2、什么是模块化开发

传统非模块化开发有如下的缺点：

- 命名冲突
- 文件依赖

模块化规范：

- CommonJS 模块化规范
- ES6 模块化规范

2、ES5 模块化

每个文件就是一个模块，有自己作用域。在一个文件里定义的变量、函数、类，都是私有的，对其他文件不可见。

2.1、创建“module”文件夹

2.2、导出模块

创建 es5/四则运算.js

```
// 定义成员
const sum = function (a, b) {
  return parseInt(a) + parseInt(b);
};
const subtract = function (a, b) {
  return parseInt(a) - parseInt(b);
};
```

导出模块中的成员

```
// 导出成员
module.exports = {
  sum: sum,
  subtract: subtract,
};
```

简写

```
// 简写
module.exports = {
  sum,
  subtract,
};
```

2.3、导入模块

创建 es5/引入模块.js

```
// 引入模块，注意：当前路径必须写 ./
const m = require("./四则运算.js");
console.log(m);
const result1 = m.sum(1, 2);
const result2 = m.subtract(1, 2);
console.log(result1, result2);
```

2.4、运行程序

```
node es5/引入模块.js
```

CommonJS 使用 exports 和 require 来导出、导入模块。

三、模块化开发（二）

1、ES6 模块化写法（一）

ES6 使用 export 和 import 来导出、导入模块。

1.1、导出模块

创建 es6/userApi1.js

```
export function getList() {
  console.log("获取数据列表1");
}
export function save() {
  console.log("保存数据1");
}
```

1.2、导入模块

创建 es6/userComponent1.js

```
// 只取需要的方法即可，多个方法用逗号分隔
import { getList, save } from './userApi1.js';
getList();
save();
```

注意：这时程序无法运行，因为 ES6 的模块化无法在 Node.js 中执行，需要用 Babel 编辑成 ES5 后再执行。

1.3、安装 Babel

Babel 是一个广泛使用的转码器，可以将 ES6 代码转为 ES5 代码，从而在现有环境执行执行

安装命令行转码工具

Babel 提供 babel-cli 工具，用于命令行转码。它的安装命令如下：

```
npm install --global babel-cli
# 查看是否安装成功
babel --version
```

1.4、配置 .babelrc

Babel 的配置文件是 .babelrc，存放在项目的根目录下，该文件用来设置转码规则和插件，presets 字段设定转码规则，将 es2015 规则加入 .babelrc：

```
{
  "presets": ["es2015"],
  "plugins": []
}
```

1.5、安装转码器

在项目中安装

```
npm install --save-dev babel-preset-es2015
```


1.6、转码

```
# 整个目录转码
mkdir dist1
# --out-dir 或 -d 参数指定输出目录
babel es6 -d dist1
```

1.7、运行程序

```
node dist1/userComponent1.js
```

2、ES6 模块化写法（二）

2.1、导出模块

创建 es6/userApi2.js

```
export default {
  getList() {
    console.log("获取数据列表2");
  },
  save() {
    console.log("保存数据2");
  },
};
```

2.2、导入模块

创建 es6/userComponent2.js

```
import user from "./userApi2.js";
user.getList();
user.save();
```

2.3、转码

```
# 整个目录转码
mkdir dist2
# --out-dir 或 -d 参数指定输出目录
babel es6 -d dist2
```


2.4、运行程序

```
node dist2/userComponent2.js
```

四、搭建项目前端环境

1、vue-admin-template 模板

vue-admin-template 是基于 vue-element-admin 的一套后台管理系统基础模板（最少精简版），可作为模板进行二次开发。

 vue-admin-template.zip

GitHub 地址：<https://github.com/PanJiaChen/vue-admin-template>

2、搭建环境

```
# 解压压缩包
# 进入目录
cd vue-admin-template

# 安装依赖
npm install

# 启动执行后，浏览器自动弹出并访问 http://localhost:9528/
npm run dev
```

3、修改登录功能

3.1、创建登录接口

创建 LoginController

```
@RestController
@RequestMapping("/admin/vod/user")
@CrossOrigin
public class LoginController {
    /**
     * 登录
     * @return
     */
    @PostMapping("login")
    public Result login() {
        Map<String, Object> map = new HashMap<>();
        map.put("token", "admin");
        return Result.ok(map);
    }
    /**
     * 获取用户信息
     * @return
     */
    @GetMapping("info")
    public Result info() {
        Map<String, Object> map = new HashMap<>();
        map.put("roles", "[admin]");
        map.put("name", "admin");
        map.put("avatar", "https://oss.aliyuncs.com/aliyun_id_photo_bucket/default_handsome.jpg");
        return Result.ok(map);
    }
    /**
     * 退出
     * @return
     */
    @PostMapping("logout")
    public Result logout(){
        return Result.ok();
    }
}
```

3.2、修改登录前端

3.2.1、修改接口路径



3.2.2、修改 js 文件



```
import request from "@utils/request";

export function login(data) {
  return request({
    url: "/admin/vod/user/login",
    method: "post",
    data,
  });
}

export function getInfo(token) {
  return request({
    url: "/admin/vod/user/info",
    method: "get",
    params: { token },
  });
}

export function logout() {
  return request({
    url: "/admin/vod/user/logout",
    method: "post",
  });
}
```

五、跨域问题

1、什么是跨域

(1) 浏览器从一个域名的网页去请求另一个域名的资源时，域名、端口、协议任一不同，都是跨域。前后端分离开发中，需要考虑 ajax 跨域的问题。

(2) 跨域的本质：浏览器对 Ajax 请求的一种限制

(3) 这里我们可以从服务端解决这个问题

2、配置

在 **Controller** 类上添加注解

```
@CrossOrigin //跨域
```