正课：

1. 脚手架:

1. 脚手架

什么是: 已经包含半成品代码的项目结构

为什么: 统一框架的使用规范

何时: 今后使用框架，都是用脚手架创建项目的

如何:

1. 安装: 2个

1. 安装可以反复创建项目的脚手架命令行工具：

npm i -g @vue/cli

2. 用命令行创建一个项目的脚手架结构:

vue create 项目文件夹名

步骤:

1. ? Please pick a preset: (Use arrow keys)

default (babel, eslint)

>Manually select features

2. ? Check the features needed for your project: （上下键切换， 空格选中/取消选中）

(\*) Babel 翻译工具: Vue脚手架，采用了ES6的模块化开发。但是几乎所有浏览器都不认识！必须接住Babel将ES6翻译为ES5，才能在浏览器中运行

( ) TypeScript

( ) Progressive Web App (PWA) Support

(\*) Router

(\*) Vuex

( ) CSS Pre-processors

~~( ) Linter / Formatter~~ 代码格式和质量检查工具！太苛刻！即使代码正确，格式不标准，也报错！学习阶段不用要求这么严格！

( ) Unit Testing

( ) E2E Testing

3. Use history mode for router? (Requires proper server setup for index fallback in production) (Y/n)N

默认: vue中的路由地址是#/相对路径

问题: 如果页面中刚好有锚点地址！锚点地址也会修改地址栏！和vue的#/路径 冲突！

解决: 采用history模式的vue路由方式——不带#的地址方式。

问题: 不带#号的地址，会被误认为是服务端资源地址，而优先交给服务解析。但是，Vue的路由，在服务端极有可能没有对应的资源，就会报404。找不到。 比如: 本来想调转到Vue的首页，采用history方式不带#的路径跳转:

http://localhost:8080/index

但是，浏览器会直接将整个地址发给服务端！而不是交给Vue解析

解决: 请服务端配置一个重定向

凡是/index.html/xxx的地址，我服务端不解析，直接返回给客户端！请Vue解析。

所以，学习阶段或自己做项目时，都不启用history模式。都采用默认#/的模式。

4. Where do you prefer placing config for Babel, PostCSS, ESLint, etc.? (Use arrow keys)

In dedicated config files

> In package.json

5. ? Save this as a preset for future projects? (y/N) N

结果: 在硬盘上指定目录下创建了Vue项目的脚手架原代码

测试: 1. 进入项目文件夹: cd 项目文件夹

2. 运行npm run serve

1. Starting development server

启动 开发 服务器

临时模拟将来网页保存的真实服务器环境。同Live Server。

福利: 只要修改原代码，开发服务器自动编译，并重新运行！不用反复手动npm run serve

2. 编译和打包:

1. Vue脚手架代码中包含大量ES6的代码。浏览器不支持！就需要命令行工具将ES6代码，编译/翻译为ES5的代码，浏览器才认识。

2. 脚手架代码分组件分模块开发的，文件很多。所以，发布时必须打包为一个文件整体。还需要压缩代码！——打包！webpack

3. 为vscode安装vetur插件:

4. 安装并配置axios模块:

1. 安装axios:

在项目内本地安装axios模块：

npm i -save axios

2. 配置axios:

1. 将axios对象，保存到Vue的原型对象中，保证所有的Vue组件，都可以使用axios对象

src/main.js:

import axios from "axios"

Vue.prototype.axios=axios

结果: 所有组件中，任何位置:

this.axios.get("url",{ ...})

.then(result=>{ ... })

2. 脚手架目录结构:

public: 唯一不会被打包的文件夹！

今后所有不需要编译的文件，都放在public下。比如: 唯一完整的index.html页面, imgs文件夹以及图片，第三方的已经编译和压缩完的css和js（bootstrap.min.css; jquery-1.11.3.min.js; bootstrap.min.js）

且: 共用的资源: bootstrap.css，jquery.js等都要在index.html中统一引入

src: 所有自己编写的代码，一律放在src下

所有src中的代码，都会被编译，打包，压缩！

assets: 保存共用的css样式和js脚本

components: 全局组件(MyHeader.vue)

或独立的复杂组件(轮播图.vue)

views: 所有页面组件

Index.vue Details.vue Login.vue Products.vue

说明: .vue文件:

每个.vue文件，是脚手架中专门定义的保存一个组件HTML（template），CSS和JS代码的文件。

今后，有几个组件，就要创建几个.vue文件

App.vue 主界面: 相当于以前的<div id="app"></div>

其中用<router-view/>占位

其它页面组件将来都会替换App.vue中的<router-view/>以此来切换页面

main.js 是主界面的new Vue()

也是整个程序的入口

经常对Vue类型或new Vue对象进行配置。

其实: App.vue+main.js=以前的一个Vue页面

router.js 路由器和路由字典

store.js 用于存储Vuex组件的数据和方法的文件。——下个阶段讲！

3. .vue文件的内容：

每个.vue文件都是一个组件

无论根组件，全局组件，子组件，页面组件都是一个.vue

每个.vue组件都拥有专属的HTML , CSS, JS和数据

<template>

//组件的HTML片段

//要求: 唯一父元素包裹！

</template>

<script>

//ES6的语法抛出一个组件对象！

export default {

//除了没有template外，其余和之前讲的组件完全一样！

}

</script>

<style>

//1.可直接将组件的css拷贝到这里

// 但是，如果这样写，<style scoped>

//scoped属性，可自动避免组件间冲突

// 如何避免: 自动给组件内的选择器加[随机属性]

//2. 也可以引入现有的.css文件

// 但是，引入现有的css文件，则scoped无效！只能自己避免组件间样式冲突

</style>

单词列表:

1. manually 手动的

2. feature 功能

3. Compile 编译