正课:

1. VUE脚手架

2. 使用VUE脚手架开发学子商城项目

3. VUE组件生命周期:

一. VUE脚手架:

1. 什么是: 已经包含核心功能的半成品项目

2. 为什么:

(1). 简单: 已经包含了核心功能，避免了大量重复编码

(2). 规范: 文件夹结构和文件夹名，文件名都已标准化，降低了不同项目之间的差异。

3. 何时: 今后几乎所有前端项目的开发都是用脚手架完成的。

4. 如何:

(1). 安装生成脚手架代码的工具(老母鸡):

npm i -g @vue/cli

安装之后: 输入vue -V

看到版本号就算成功

(2). 用生成脚手架代码的工具为本次项目创建一套脚手架代码结构: (请老母鸡下一个蛋)

a. 在想要生成项目的目录，地址栏里写cmd，打开命令行

b. 在命令行中输入vue create xzvue

c. Your connection to the default npm registry seems to be slow.

你现在的链接是链接到默认的国外的npm仓库，看起来有些慢

Use https://registry.npm.taobao.org for faster installation? (Y/n)

是否使用国内的淘宝镜像来更快速安装 输入Y按回车

d. ? Please pick a preset: 请选择一个预置的设置

default (babel, eslint) //使用默认设置

> Manually select features //手动选择功能

按方向键下，选第二个，手动选择功能，按回车

e. ? Check the features needed for your project: (Press <space> to select, <a> to toggle all

选择你的项目需要的功能（按空格选中或取消选中，按a全选所有）

, <i> to invert selection)

(\*) Babel 将浏览器不认识的ES6甚至更高版本的js代码，翻译为ES5版本的js

( ) TypeScript

( ) Progressive Web App (PWA) Support

(\*) Router VUE中实现单页面应用的核心——路由器组件

(\*) Vuex 下个阶段要讲的vuex

( ) CSS Pre-processors

> ( ) Linter / Formatter Linter 是代码质量检查工具，会将格式不规范也报错！一定不要选择

( ) Unit Testing

( ) E2E Testing

按方向键上下移动，移动到想要选中的功能上，按空格键选中Babel、Router、Vuex，取消选中Linter

f. Use history mode for router? (Requires proper server setup for index fallback in produc

tion) (Y/n) 输入n 按回车

是否使用history模式作为路由器的标识（生产环境中需要配置首页重定向才能使用）

Vue-router默认采用#/相对路径 方式实现客户端路由地址:

比如: http://localhost:5500/index.html#/

http://localhost:5500/index.html#/details/2

history模式:

http://localhost:5500/

http://localhost:5500/details/2

默认浏览器会将所有地址栏中的地址发给服务器端请求资源，只要#地址留在客户端，不发给服务器。

但是，http://localhost:5500/details/2 我们本来想在客户端路由切换页面组件，但是浏览器也会发送给服务器端查找，因为服务器端没有这个资源，所以，很可能报错！

所以，将来，要么就用默认#/相对路径方式，要么可用histroy,必须同时请服务器端修改配置！

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

你想把本次的配置保存在哪个位置？（用方向键选择）

In dedicated config files //把每种组件的配置，放在各自单独的配置文件中

> In package.json //将所有配置集中放在一个package.json文件中

按方向键下，选in package.json，按回车

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

是否保存这次的配置为将来项目的预定义配置

输入N，按回车

i. 看到结果: 说明用脚手架工具生成项目源代码结构成功

...

Successfully created project xzvue.

👉 Get started with the following commands:

$ cd xzvue

$ npm run serve

j. 在windows资源管理器中，进入当前文件家，就看到为这次项目生成的半成品项目源代码文件夹xzvue。（老母鸡为这次项目下的蛋）

k. 将来，每做一个新项目，这套a-h的步骤都要重新操作一遍。

l. 其实刚创建完的脚手架项目源代码包含示例网页:

用vscode打开刚创建的脚手架项目文件夹

右键点击项目中的package.json文件，选在终端/命令行中打开

在弹出的命令行窗口中，输入npm run serve

Npm run serve: 做了两件事:

编译脚手架项目代码为浏览器认识的ES5代码

同时启动简易开发服务器运行网站中网页

打开浏览器，在地址栏中输入: http://localhost:8080，可看到示例网页

或在命令行窗口中，按ctrl点连接地址: http://localhost:8080，也可查看示例网页

强调: 从此live server 退出历史舞台！不再使用live server运行！

所有vue项目都用npm run serve运行

(3). 为脚手架添加axios模块:

a. 在脚手架生成的项目文件夹内，比如: 这里是xzvue文件夹内

在地址栏输入cmd，打开命令行

b. 本地安装axios模块:

npm i -save axios

c. 在脚手架项目源代码的src/main.js中，new Vue()前引入axios模块

import axios from "axios" //node\_modules中安装的模块，引入时都不用加路径

d. 设置axios对象的基础路径属性:

axios.defaults.baseURL="http://服务器端域名"

比如: 学子商城项目中: axios.dafaults.baseURL="http://xzserver.applinzi.com"

e. 将axios对象放入Vue的原型对象中

Vue.prototype.axios=axios;

f. 结果: 因为所有组件对象都是Vue类型的子对象，所以在所有组件内，任何位置都可用this.axios.get()和this.axios.post()访问Vue.prototype中的axios对象里的函数。

5. 脚手架项目结构:

(1). .git文件夹(隐藏): 本地git仓库文件夹，如果git不熟，可以删掉。

(2). node\_modules文件夹: 保存脚手架项目运行的依赖包。

(3). public文件夹:

a. 保存着整个单页面应用唯一完整的index.html文件

b. 将来还可能保存所有页面都需要的公共的css文件和js文件，比如:

bootstrap.css jquery.js bootstrap.js

1). 可在public文件夹下新建css和js文件夹

2). 将所有页面都需要共用的第三方的css和js保存进去

3). 在唯一完整的index.html，顶部引入所有页面所需的第三方的css和js

c. 将来网页中用的所有图片，也都应该保存在public下！

所以，拷贝旧项目中的img文件夹到public下

(4). src文件夹: 放所有我们自己编写的HTML，css和js

a. 根组件: 原SPA中唯一完整的页面的HTML和new Vue()不再放在一个.html文件里，而是分为两个文件了:

1). App.vue 只放根组件<div id="app">的HTML内容和根组件的css样式

2). main.js中放根组件new Vue({}) 以及对new Vue()的配置

3). 结果: 运行时，App.vue中的<div id="app">和main.js中new Vue()会重新被编译回唯一完整的index.html页面内。

b. 多个页面组件:

1). 所有页面组件都要放在src/views/文件夹下

2). 每个页面组件都是一个.vue文件

3). 每个.vue文件中都包含三部分:

i. <template>

编写当前页面组件的HTML片段

</template>

ii. <script>//原页面组件对象中所有js内容，不再含template属性

{

data(){ return { ... } },

...

}

</script>

iii. <style>

//当前页面所需的css样式，注意组件间样式冲突

</style>

c. 路由器对象: src/router/index.js

同SPA中的路由器对象几乎完全一样

new VueRouter({

routers:[

{path:"/",component:Index},

{path:"/details/:lid",component:Details, props:true},

...

{path:"\*", component: NotFound }

]

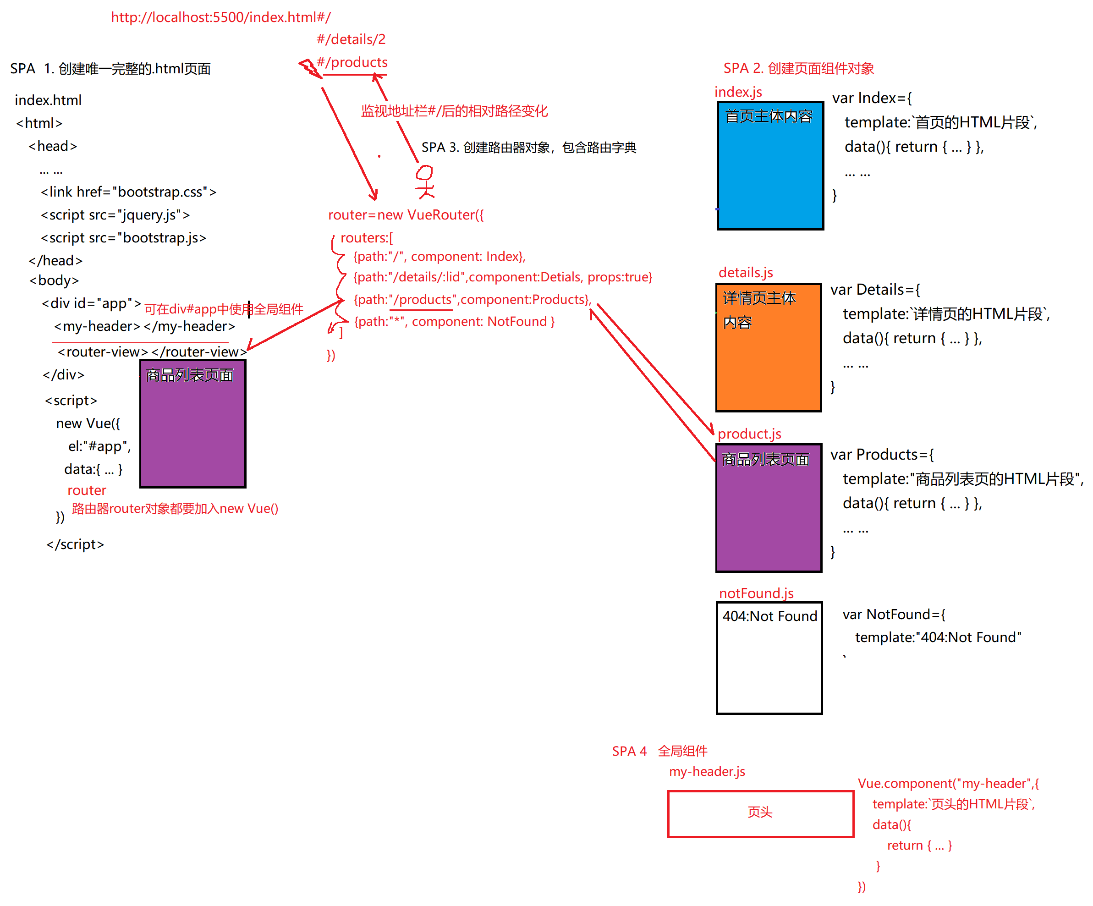
})

说明: router对象已经在main.js中被加入到new Vue()里了.

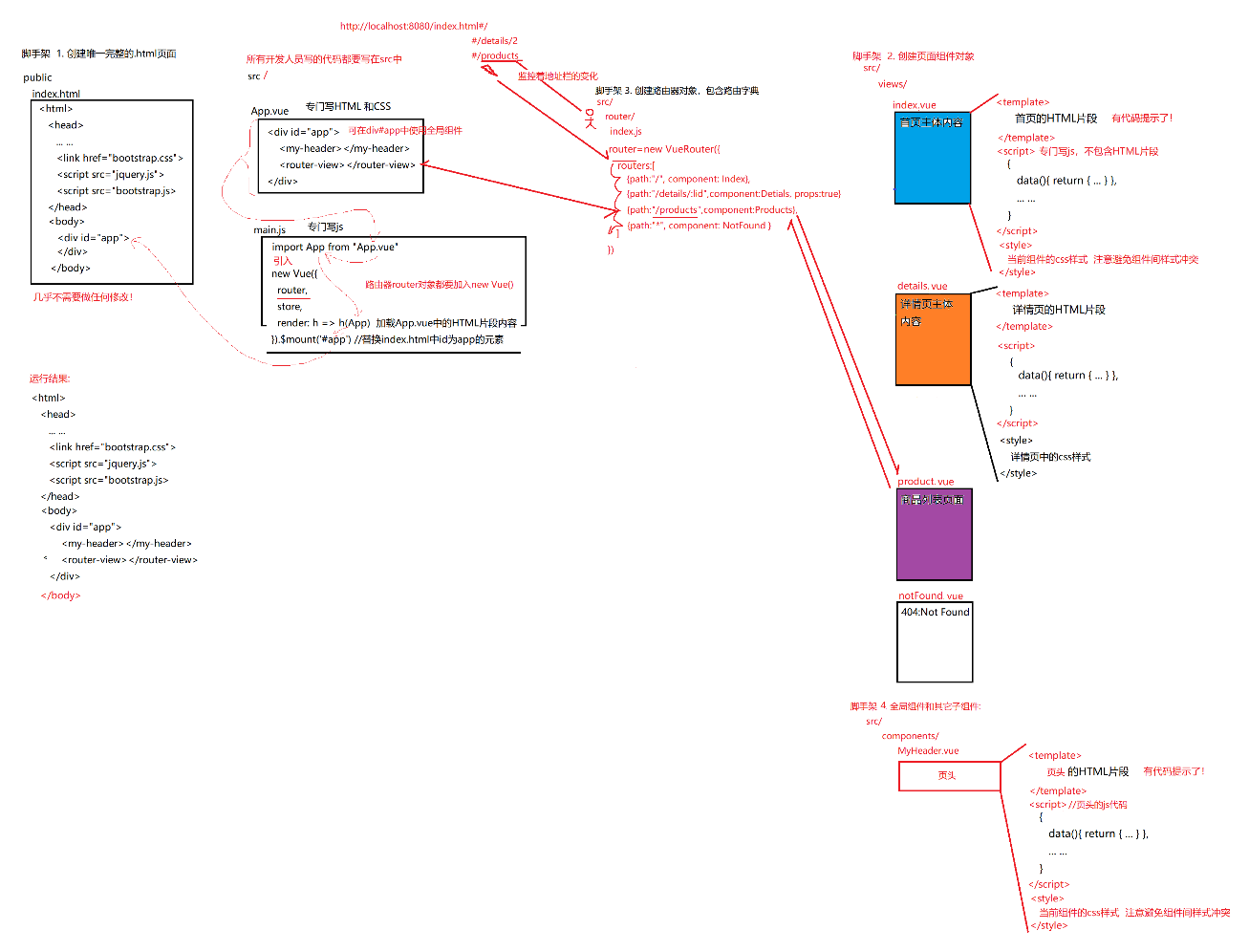
d. components文件夹: 集中保存所有全局组件对象文件和子组件的文件。

6. 其实: 脚手架代码的结构和SPA应用代码的结构本质是相同的。用已知释新知。

(1). 回顾: SPA代码结构:

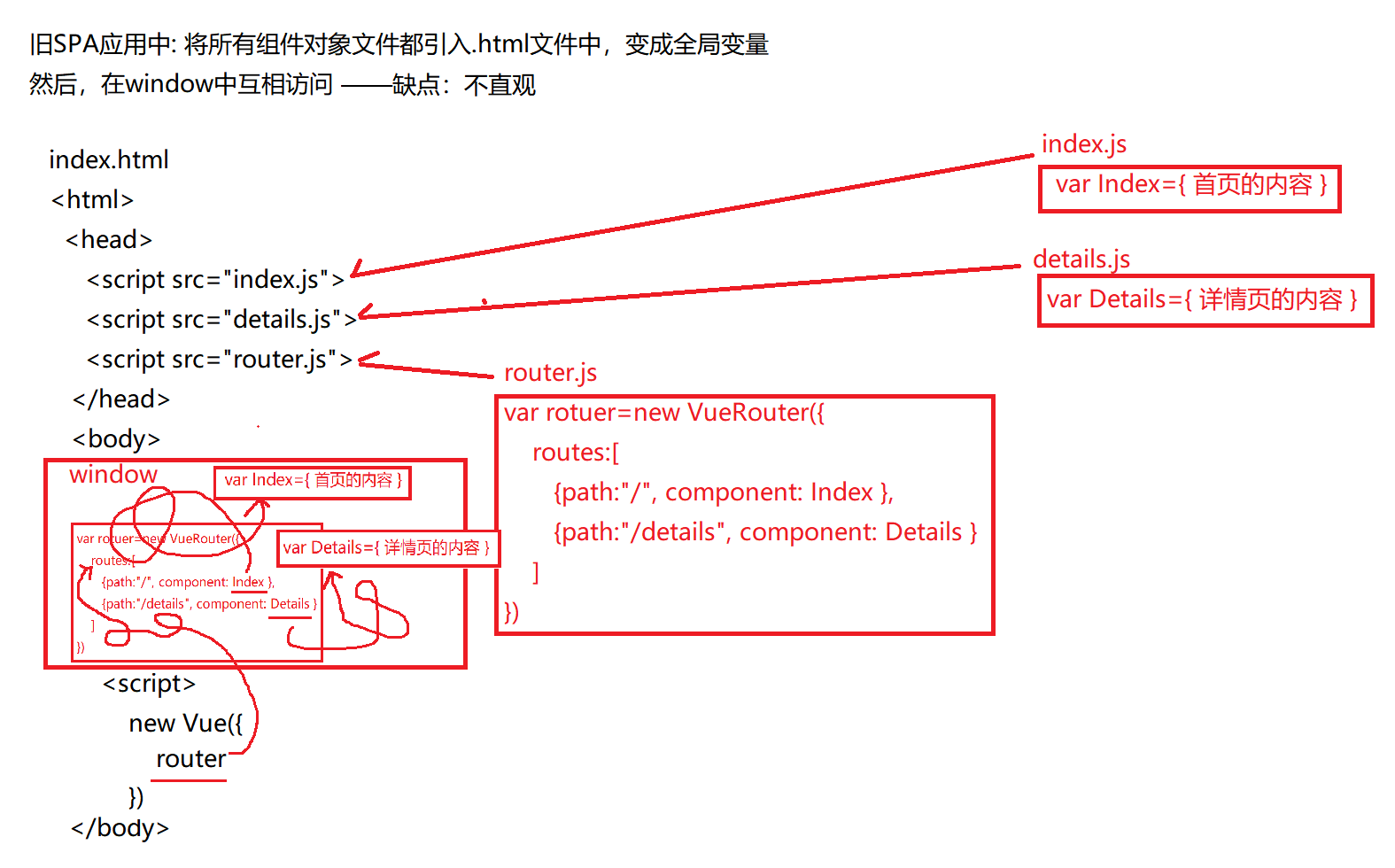
s

(2). 脚手架代码结构: 其实就是SPA的代码结构



7. es6模块化开发在脚手架中的应用

(1). 旧SPA应用中的引入问题: 所有组件对象都要先集中引入到Index.html中，再以全局对象方式使用。如果单看某一个组件对象文件，根本看不出依赖关系。——不直观



(2). 每个.vue文件和.js文件，默认都是一个模块对象

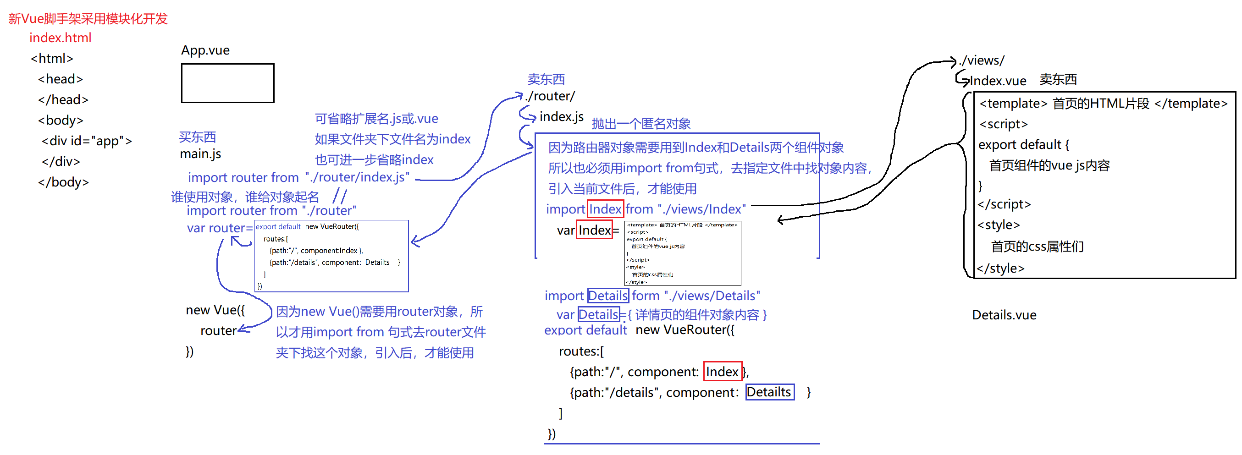
(3). 每个模块对象都可向外抛出自己内部的成员，供外部访问

export default {  
 要抛出的组件对象的内容  
 }

(4). 一个模块想使用另一个模块的内容时，无需经过第三方，即可直接找到模块文件引入其中的内容。

import 变量名 from "目标文件的相对路径"

(5). 结果: 就可把目标文件中抛出的内容，引入自己文件内，像使用自己的变量和对象一样，使用其他文件模块中的内容。——减少中间商赚差价！



二. 用VUE脚手架开发学子商城项目:

1. 在public/文件夹中添加网站所需的img文件夹和所有页面共用的第三方css和js，并且在唯一完整的index.html页面中，<head></head>中引入第三方的css和js

2. 在src/views/文件夹下，创建多个页面组件，将来程序中有几个页面，就创建几个.vue文件

每个页面.vue文件内都包含三部分内容:

(1). <template>中包含这个页面的HTML片段：

学子商城项目中: 回到旧jQuery项目中，找到对应的页面.html文件，复制其中的<main>部分代码到，Vue脚手架项目中的页面.vue中的<template>内

(2). <style>中包含这个页面的css代码:

学子商城项目中: 回到旧jQuery项目中，找到对应的页面.css文件，复制其中所有css代码到，Vue脚手架项目中的页面.vue中的<style>内

3. 在路由器对象中，添加每个页面的路由地址

在src/router/index.js中:

(1). 先引入所有页面的组件对象:

import 页面名 from "页面组件的相对路径"

...

(2). 再修改routes数组中的路由字典条目:

{path:"/路径名", component: 页面名}

4. 全局组件:

(1). 在src/components/文件下

新建一个普通的组件.vue文件，同样包含三部分  
 <template>内包含组件的HTML片段

<script>内包含组件的vue js代码

<style>内包含组件的css代码

(2). 在main.js中，在new Vue()前

a. 找到全局组件文件所在的位置，将全局组件的内容以模块对象方式引入到main.js中备用

import 组件名 from "./components/组件名.vue"

比如: 想引入页头组件:

import MyHeader from "./components/MyHeader"

b. 将引入的组件对象，变成一个真正的全局组件！

Vue.component("标签名", 组件名)

比如: 想将引入的页头组件对象，变成真正的全局组件

Vue.component("my-header",MyHeader);

c. 结果: 将来在任何页面或组件的任何位置都可通过<my-header></my-header>来引入全局组件。

比如: 想给将来所有页面上方都加页头: 在App.vue中

<div id="app">

     <my-header></my-header>

     <router-view/>

   </div>

5. 子组件:

(1). 在src/components/文件夹下新建子组件.vue文件

在子组件.vue文件中编写这个子组件的内容

比如: src/components/Carousel.vue

<template>

<script>

<style>

(2). 在父组件的<script>中用import引入子组件对象  
 <script>

import 组件名 from "子组件路径"

比如: import Carsouel from "../components/ Carsouel "

(3). 在父组件的<script>中的export default 中

export default {

... ...

components:{

组件名

比如: Carsouel

}

}

(4). 结果: 在当前组件或页面内，就可用<carsouel></carsouel>来引入子组件

6. 如果所有页面都需要的css代码写哪儿？App.vue中的<style>中

7. 使用axios动态获取数据:

(1). 在用脚手架工具生成完项目源代码后，就要为项目添加axios模块(见脚手架安装和配置部分)

(2). 在当前组件的生命周期钩子函数中的created()函数内，或mounted()函数内都可发送axios请求。

(3). 在axios.get().then(res=>{})会调函数中，将服务器返回的数据(res.data)，分门别类的赋值给data中的变量

(4). 在页面/组件的<template>中，找到需要这些变量的位置，使用vue第一天讲的绑定语法，绑定数据到页面指定位置

三. 组件的生命周期:

(1). 生命周期: 一个组件的加载过程

(2). 回顾: 网页加载过程: 也有生命周期

先加载HTML和JS，当HTML和JS加载完成后，提前触发DOMContentLoaded事件，

所以，我们可以在DOMContentLoaded中编写发送ajax请求的代码。这样，只要页面加载到这个阶段，事件触发，就会自动向服务器发送请求。

然后，当所有网页内容加载完，还会触发一个事件: window.onload。凡是写在window.onload事件中的代码，都会在整个页面加载完自动触发执行。

(3). 问题：组件不是页面，无法触发页面的加载完成事件，但是，我们也想在组件加载完自动发送ajax请求！

(4). 解决: 其实组件加载过程，也有生命周期的阶段，每个阶段也能自动触发一个回调函数。但是因为这个回调函数不是网页标准的事件，所以这种特殊的回调函数，称为生命周期中的钩子函数

(5). 列举: Vue组件加载过程共分四个阶段，每个阶段前后，都会自动触发一个钩子函数。共8个钩子函数:

a.0 beforeCreate(){ ... ~~axios.get()~~ ... }

a. 创建(create)阶段: 创建vue对象，并创建vue中的data对象

a.1 created(){ ... axios.get() ... }

b.0 beforeMount(){ ... }

b. 挂载(mount)阶段: 扫描网页内容，建立虚拟DOM树，并首次更新页面中的内容

b.1 mounted(){ ... axios.get() ... }

\*\*\*\*\*\*\*\*\*\*\*\*\*\*首次加载过程到此结束\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 c.0 beforeUpdate(){ ... }

c. 更新(update)阶段: 当组件的data中的变量被修改时，自动触发该阶段

c.1 updated(){ ... }

d.0 beforeDestroy(){ ... }

d. 销毁(destroy)阶段: 主动调用专门的函数销毁一个组件时才自动触发该阶段

d1 destroyed(){ ... }

(6). 鄙视题: 父组件和子组件生命周期函数的执行顺序:

父组件beforeCreate()

父组件created()

父组件 beforeMount()

父组件扫描页面内容时，发现不认识的子组件标签，开始进入子组件的加载过程

子组件beforeCreate()

子组件created()

子组件 beforeMount()

子附件 mounted()

父组件才继续先后扫描后续内容

父组件的mounted()

作业:

1. 看小程序中的视频，学习学子商城VUE版更多技巧:

小程序->在线->Vue->day05:

2. （必须）学子商城详情页xzvue details

3. （必须）学子商城详情页放大镜效果和懒加载

4. （可选）学子商城商品列表页面和按关键字搜索商品

2. 扩展: 企业要求，所有ajax请求不能写的到处都是，必须集中封装在函数中，然后在页面上调用函数——企业必备技能：

看小程序中的视频：小程序->在线->Vue->day05: 5. Vue脚手架 封装服务端接口函数 api

3. 扩展: 看视频学习vue版轮播图:

小程序->在线->Vue->day05: 扩展: vue版轮播图