准备: 微信小程序: WEB问题速查 小恐龙图标

1. 历届师兄师姐在今天课程中曾经问过的高频的经典的问题清单和解答: 小程序->首页->VUE->day04、day05

自己可能没有发现问题，但是想知道别的同学都问过哪些问题

2. 今日课程对应的重点案例的视频以及鄙视面试所需的扩展知识视频：小程序->在线->VUE->day04、day 05

(学有余力!!!的同学或自己认为自己基础好的同学必须看!!!!!!)：

今日扩展视频：

小程序->在线->VUE->day04

模块化开发（7个视频）（基础一般的同学不用看！）

小程序->在线->VUE->day05

token jsonp keepalive缓存 vuex vue版轮播图

3. 今日课程中包含的英语单词的翻译：小程序->单词->VUE->day04、day05

英文不太好的同学，可以提前看一下第二天需要用到哪些英文单词。

约定: 2件事:

1. 第三阶段禁止上课跟着敲笔记和敲代码！谁抄笔记抄的勤，谁抄代码抄的勤，谁学的越烂！

2. 每个完整功能的案例/视频，至少写三遍: (如果需要画图，必须亲手画图)

(1). 连代码+注释抄一遍 —— 明白这个例子是要做什么事儿

(2). 保留注释，自己试着看着注释，将代码填回来 —— 复习函数和对象

(3). 删除所有代码和注释，自己试着用自己的话，把注释写回来，再翻译为代码！—— 只有自己可以写出人话的注释，这个例子才算自己会！

如果没有问题的同学: （享受vip一对一答疑的时间还剩1+3天，端午节我不休息）

1. 上课时那句话没听懂，笔记中那句话看不懂，案例中那句话看不懂，立刻问！

2. 小程序问题清单中哪个问题或解答看不懂，立刻问！

3. 小程序视频中哪个视频看不懂，立刻问

4. 个人项目中哪个功能不会做，立刻问

5. 平时刷笔试题时，遇到不会的笔试题，立刻问

要留级的同学，确定留级成功后！提前微信告知我！我改备注！否则有些重要通知可能收不到！

正课:

1. 脚手架:

2. 脚手架结构

3. ES6模块化开发

4. 懒加载

5. 为整个脚手架代码配置统一的axios

6. 做项目的流程:

一. 脚手架:

对应小程序视频: 小程序->在线->VUE->day04 2. 使用vue/cli工具创建学子商城脚手架

1. 什么是脚手架代码: 已经包含标准文件夹结构以及核心功能代码的半成品VUE框架项目

2. 为什么: 统一标准化所有VUE项目的结构！

3. 何时: 今后所有企业项目都在脚手架标准结构基础上继续开发

4. 如何: 2步:

(1). 安装用于反复生成脚手架代码的命令行工具（老母鸡）

！！！禁止上课跟着做操作！！！

a. 测试当前环境是否安装了vue/cli

命令行: vue -V

看到: 'vue' 不是内部或外部命令，说明vue/cli工具没有安装

b. 安装vue/cli

npm i -g @vue/cli

只要看到+ @vue/cli@4.4.5说明安装成功，版本号只要4.x以上都行

c. 测试当前环境是否安装vue/cli

命令行: vue -V

看到: @vue/cli 4.4.5 说明安装成功！版本号只要4.x以上都行

(2). 每做一个项目，都要用命令行工具生成一套脚手架代码（老母鸡下蛋）

a. 在要生成项目的文件夹，地址栏中输入cmd回车，在当前文件夹打开命令行

b. 运行vue create 项目名 按回车

1). ? Your connection to the default npm registry seems to be slow.

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

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

default (babel, eslint)

> Manually select features

3). ? Check the features needed for your project: (Press <space> to select, <a> to toggle all, <i> to invert selection)

>(\*) Babel 翻译: 脚手架代码中大量使用了ES6和ES7时髦的语法，大部分旧浏览器不支持！所以，最终交付给客户时，都要编译为等效的ES5代码，才能保证大部分浏览器都能正常使用！

( ) TypeScript VUE框架和react框架不主要使用ts，所以第三阶段不讲typescript。因为ng框架强制使用typescript，所以typescript第五阶段讲ng框架时再详细讲。

( ) Progressive Web App (PWA) Support

(\*) Router 单页面应用的核心组件，必选

( ) Vuex 以后必选！

( ) CSS Pre-processors 如果用less或sass才选

~~( ) Linter / Formatter~~ 代码规范和质量检查工具，就算程序没写错，功能可以正常执行，但是如果代码写的不规范或代码质量差，也会报错！

( ) Unit Testing

( ) E2E Testing

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

i. 问题: SPA应用中默认使用#/路径方式作为前端项目客户端导航的地址区分方式

比如: #/ 默认首页

#/details 详情页

如果在一个特别长的网页中也想使用锚点地址做页面内导航，就会和SPA应用的#发生冲突！

ii. 解决: 在new VueRouter({

mode: "history"，

routes

})

结果: SPA应用不再使用#/作为客户端导航地址

比如: / 默认首页

/details 详情页

iii. 问题: 浏览器懵逼了！

以前: 浏览器看到#/xxx 会发给前端的vue框架解析, 看到/xxx会发给后端服务器接口解析

现在: 都用/，浏览器会将所有/xxx都发给服务器端解析，不再发给vue框架解析，就会出错！

iv. 解决: 今后只要想开启history模式去#，必须请服务器端工程师用特殊的重定向手段协助解决才行！

v. 所以: 在达内学习时，不要开history模式，将来到了公司，再开history模式。

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

In dedicated config files

> In package.json

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

c. 看到以下内容，说明安装成功：

🎉 Successfully created project xzvue.

👉 Get started with the following commands:

$ cd xzvue

$ npm run serve

(3). 脚手架代码已经是一套可以运行的网站代码了，包含了SPA应用三大部分内容和示例网页。用vscode运行:

！！！禁止上课跟着做操作！！！

a. 如果不用git管理代码，则可以删除.git隐藏文件夹:

1). 在操作系统文件夹窗口，选顶部菜单: 查看->选项->查看->选中 显示隐藏的文件、文件夹和驱动器，点应用

2). 进入刚创建的项目文件夹，删除.git文件夹

b. 用vscode打开刚刚命令行创建的项目文件夹

c. 右键单击package.json文件，选择"在终端中打开"

d. 当弹出终端窗口（稍等），看到>，输入npm run serve

1). DONE Compiled successfully in 3163ms

会编译/翻译脚手架中时髦的代码为ES代码，并压缩代码。

2). App running at:

- Local: http://localhost:8080/

启动一个简化版的服务器，临时保存脚手架项目中的网页，让开发人员可以在本地调试网页功能

~~从此，再不会用live server —— 退出历史舞台！~~

今后，只要是vue脚手架项目都用npm run serve运行！

3). 按住ctrl，点终端中Local后的连接，就可打开示例页面

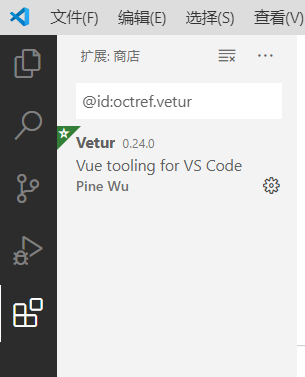
e. 你不用每次都反复运行npm run serve:

(1). npm run serve是热编译，只要代码一修改，一保存，自动重新编译！无需手动重新运行命令！

(2). 已经打开的页面无需刷新，就自动换成修改后的页面了！

5. （选装）安装vetur插件: 2个办法:

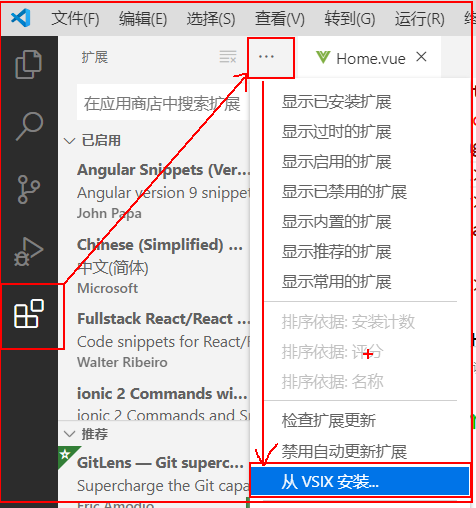
(1). 有外网: 点vscode左侧最下方“扩展”按钮，文本框输入octref.vetur，点安装



(2). 没有外网:

下载小程序->在线->VUE->day04 学校电脑版vetur插件

点vscode左侧最下方"扩展"按钮，点右上方...，选从VSIX安装，在打开的窗口中选择刚下载的vetur.vsix文件即可。



二. 脚手架结构:

对应小程序视频: 小程序->在线->VUE->day04 3. vue脚手架文件夹结构

1. 回顾: SPA应用3步+1步:

(1). 创建一个唯一完整的HTML页面，包含

a. <router-view>占位

b. new Vue({

...,

router

})

(2). 为每个"页面"都创建一个组件对象/文件

(3). 在专门的router.js文件中创建路由器对象和路由字典

(4). 如果有共用的页头，会创建全局组件，在需要的页面组件中引入

2. 脚手架项目结构就是SPA应用标准结构！只不过有的划分的更细致，有的放入专门的文件夹中。——脚手架项目结构绝不是新知识！

(1). 也有一个唯一完整的HTML页面: 但是被一分为三

a. 基础HTML部分，保留在了index.html中

public文件夹/ 将来做项目时

css文件夹/ 包含不是自己写的第三方的压缩过的css文件，比如bootstrap.min,css

js文件夹/ 包含不是自己写的第三方的压缩过的js文件，比如:

jquery-1.11.3.min.js

bootstrap.min.js

imgs文件夹/ 包含网站所需的所有图片

index.html 唯一完整的HTML页面

b. <div id="app">  
 <router-view>

被单独放在src/App.vue文件中

c. new Vue() 也被单独放在src/main.js文件中

d. 将来运行时，先会用App.vue中的<div id="app"><router-view>自动代替index.html中的空的<div id="app">，并用main.js中的new Vue()加载router，扫描index.html中的<div id="app">

e. public/index.html + src/App.vue + src/main.js 合起来等效于昨天SPA中的index.html

(2). 也要为每个“页面”分别创建组件对象/文件，只不过

a. 所有页面组件的文件都集中放在src/views文件夹中，

b. 且每个页面都是一个.vue文件，而不是一个.js文件。每个.vue文件包含三部分:

1). <template></template>内，包含页面组件的HTML片段

2). <script></script>内，包含页面组件的对象 —— 和昨天讲的页面组件对象相比，除了没有template属性之外，其余完全一样！

3). <style></style>内，包含这个页面组件所需的所有css定义。

c. 运行时: 也会根据地址栏变化，将对应组件的<template>中的html代替index.html中的<router-view>位置。用<script>中页面组件对象监控<template>中HTML内容的变化实现vue绑定和交互功能。

(3). 也会在专门的router.js文件中创建路由字典和路由器对象: 只不过:

a. 必须放在src/router文件夹/index.js

b. 内容几乎没变！

c. 除了增加了懒加载功能（下午讲）

(4). 如果有公共的页头组件，也应该创建为全局组件：有点变化！ 2步

a. 在src/components文件夹下，新建.vue文件保存全局组件的<template>+<script>+<style>

组件们

问题: 暂时只是一个普通的子组件对象，还不是Vue家的全局组件

b. 在main.js中，new Vue()之前,2步:

1). 引入src/components/全局组件.vue文件，

2). 再将引入的全局组件对象用Vue.component()函数转化成一个Vue大家庭中的全局组件。

c. 结果: 在项目任何一个组件中的<template>中使用<组件标签名>来引用全局组件

|  |
| --- |
| 练习: 自己封装有个公共的页头:  1. 先在src/components文件夹下创建MyHeader.vue文件，在.vue文件中添加<template></template>和<style scoped></style>，因为没有写js，所以暂时不用写<script>  2. 将App.vue中<router-view>上方的<div id="nav">xxx</div>剪切到MyHeader.vue中<template>内，将App.vue中<style>内#nav开头的所有样式，剪切到MyHeader.vue中<style>内  3. 在main.js中，new Vue()之前先用import 引入MyHeader.vue，然后用Vue.component()将引入的MyHeader组件对象转化为全局组件  暂时不要问import，稍后讲！  4. 回到App.vue中，在<router-view>上方添加<my-header><my-header>  5. npm run serve，按ctrl点localhost:8080  结果: Home页和About页，有了公共的页头组件 |

3. 样式冲突:

(1). 问题: 所有.vue中的<style>中的css样式最终会被编译到一个大的.css文件中集中保存，默认所有页面其实是共用这批样式的！极容易造成污染！

(2). 解决: 如何避免组件间样式冲突: 高频笔试题

a. 好的解决办法: 2步：

1). 在创建组件时，就为组件的唯一父元素定义一个专门的class名！跟别的组件都不一样！

2). 从此一个组件下的所有css选择器，都要以这个组件的统一class名开头

b. 偷懒的，不一定总是有效的方法: 1步

1). <style scoped>...</style>

2). scoped属性自动保证当前组件内的样式一定和别的组件的样式不冲突！

3). 原理:

i. scoped会给当前组件随机添加一个自定义属性

<template>

<div afasdfsdaf >

... ...

ii. 然后自动给<style>中每个选择器前加上这个自定义属性选择器

<style>

[afasdfsdaf]>xxx{ ... }

[afasdfsdaf]>xxx{ ... }

iii. 因为不同组件随机生成的自定义属性名不同，所以，也可避免组件间样式冲突

iv. 问题: 随机生成的自定义属性名没有意义，不可维护。不如自己主动定义的class名好用。

三. ES6模块化开发:

1. 什么是模块: 封装一个事物的属性和功能的对象

2. 在Vue脚手架中，每个.vue文件默认也都是一个模块对象

3. 一个模块对象可以随意被其它的模块或文件引入，如果一个模块中想使用另一个模块的内容和功能，都用import引入

import 自定义对象名 from "相对路径/文件名.vue"

结果: 将路径所指的.vue文件中的内容，打包为一个对象，引入当前文件中。

强调: 不管原来模块叫什么名字，from前的对象名是可以任意自定义的！

4. 抛出模块对象:

(1). .vue文件在不包含js内容时，默认就是一个模块对象，可被别人import引入

(2). 如果.vue文件中包含了组件对象(js代码)，则必须用export default { ... 组件js内容 ...}抛出，外人才能用import引入使用

四. 懒加载:

1. 问题: 旧单页面应用中，首屏加载时，只能一次性将所有页面组件，打包为一个大的js文件，下载下来。不管用户是否会看除首页之外的其它页面，都会全部下载所有页面内容。——首屏加载极慢，且不划算

2. 解决: 懒加载

3. 懒加载: 首屏只加载首页的内容(懒), 之后，每请求一个新页面，才临时下载新页面组件的内容。

4. 好处: 首屏加载快

5. 缺点: 后续页面加载稍慢！

6. 其实: 2种懒加载:

(1). 异步延迟下载: 当加载首页组件内容时，同时在底层发送异步请求，在不影响当前页面使用的情况下，悄悄的下载其余页面的内容

a. 优点: 当用户访问下个页面时，可能下个页面的组件已经提前异步下载好了！不影响后续页面的加载速度

b. 缺点: 偷跑流量！在用户不知道的情况下，多下载了内容。

c. 如何: 只在router/index.js中

1). 除首页之外的其它页面组件，不要提前import引入！

2). 路由字典中，除首页之外的其它页面，都可以采用:

{

path: '/about',

///\* webpackChunkName: "将来独立下载的js文件名" \*/不能删除，因为这是在为将来独立打包的js文件自定义文件名。

component: () => import(/\* webpackChunkName: "about" \*/ '../views/About.vue')

//结果: 不再把About.vue的内容和大js打包在一起。而是单独生成一个about.js文件

//下载时，异步单独下载about.js文件

}

(2). 完全懒加载: 只有当用户确实访问到这个页面时才动态加载这个页面的js。如果用户不访问。则永远不加载该页面.js

a. 优点: 不会偷跑流量

b. 缺点: 后续页面加载稍微慢

c. 如何:

1). 以上两步还是要做

2). 多做一件事: 在脚手架项目根目录创建一个文件vue.config.js，在其中添加如下代码:

module.exports={

chainWebpack:config=>{

config.plugins.delete("prefetch") //取消异步延迟下载

}

}

d. 结果: 所有懒加载的页面.js文件都不会提前异步延迟下载。只有用户访问这个页面对应的地址时，才临时下载页面组件.js。

练习:

1. 下载ftp: vue.config.js文件到xzvue项目根目录

2. 运行首页，打开network，刷新，看不到about.js

3. 点"关于"，在network中，才看到about.js

五. 为整个脚手架项目配置统一的axios：3步

！！！禁止跟着操作！！！

1. 安装axios: npm i -save axios

看到: + axios@0.19.2 说明安装完成

结果: node\_modules文件夹中多出一个axios文件夹

2. 引入并配置axios模块: 在main.js中， new Vue()之前

import axios from "axios" //引入node\_modules中的模块，什么路径/都不用加！

//配置服务端基础地质:

axios.defaults.baseURL="http://xzserver.applinzi.com"

3. 将配置好的axios对象放到Vue的原型对象中！

(1). 问题: 我们希望在所有页面组件或子组件或全局组件中都能随处使用axios对象发送ajax请求

(2). 原理: 其实，无论根组件new Vue()，还是全局组件，还是页面组件，还是子组件，底层都是Vue类型的子对象。只不过我们看不见new而已！——信任！

(3). 解决: 所有vue的孩子共用的东西，都要放在Vue的原型对象中

//强行给Vue.prototype添加一个axios属性，值为配置好的axios对象

Vue.prototype.axios=axios;

4. 结果: 所有组件中，都可用this.axios发送ajax请求！

六. 项目开发流程:

!!!不要跟着操作！！！

1. 先在public文件夹中准备好第三方的css和js，以及项目要用的所有图片，并在index.html页面顶部引入第三方的css和js

2. 如果有整个项目每个页面都需要的基础的css重置代码，应该放在App.vue中的style中！

因为App.vue相当于所有页面的一个外壳！

所有页面将来加载时，都会替换App.vue中<router-view>成为App.vue中的一部分

所有公共的css代码只要放在App.vue中，所有页面将来都会共用！

3. 在src/views文件夹下为每个页面创建独立的.vue文件:

<template>

这个页面的HTML片段（唯一父元素！）

</template>

<script>

export default {

就是以前的页面组件对象, 当前页面所需的所有变量，函数，ajax请求等！

props:[],

data(){

return {

变量

}

},

methods:{

函数

},

computed:{

计算属性

},

mounted(){

this.axios.get(...).then(result=>{ ... })

}

}

</script>

<style scoped>

当前页面所需的css

</style>

3. 在src/router/index.js中， 修改路由字典，换成我们网页的名称和路径！

(1). 引入我们自己页面的模块对象

(2). 修改routes中的路由字典

4. 先在src/components文件夹中创建页头组件.vue文件，还要在main.js中引入页头组件，并转化为全局组件。最后在需要加载页头的位置使用页头组件标签加载页头内容即可。

5. 在每个页面组件中发送axios请求，获取数据，绑定页面

总结:

1. MVVM: 界面View+模型对象Model+视图模型ViewModel

2. Vue的绑定原理: 访问器属性+虚拟DOM树(+观察者模式)

3. 虚拟DOM树的优点:

(1). 小，仅包含可能变化的元素

(2). 遍历查找快

(3). 更新效率高，只更新受影响的元素

(4). 避免重复编码，已封装了DOM的增删改查操作

总结: 今后所有VUE功能三步:

1. 创建界面(增强)，界面中标记变量和事件处理函数

2. 创建new Vue()对象，用el:"选择器"，找到自己负责的页面区域的唯一父元素

3. 在new Vue()中添加data和methods，分别保存界面所需的所有变量和事件处理函数

总结:

1. 绑定语法(1种)和指令(13种):

(1). 只要元素的内容随变量自动变化: {{}} v-text

(2). 只要元素的属性值随变量自动变化: :

(3). 控制一个元素显示隐藏: v-show （display:none）

(4). 控制两个元素二选一显示: v-if v-else （删除元素）

(5). 控制多个元素多选一显示: v-if v-else-if v-else

(6). 反复创建多个相同结果的元素: v-for :key="下标"

(7). 绑定事件: @ $event

(8). 防止用户短暂看到{{}}: v-cloak v-text

(9). 只要绑定的内容是HTML片段: v-html

(10). 只要希望一个元素只在首次加载时绑定一次，之后都不会自动更新: v-once

(11). 只要不希望元素内容中的{{}}被vue解析，而是保持原样显示: v-pre

(12). 只要希望随时获得页面上用户在表单元素中修改的新值: v-model

2. 绑定样式:

(1). 如果希望精确修改某一个css属性的值时，绑定style内联样式:

a. 为每个要修改的css属性分别设置变量

<元素 style="固定不变的css属性 " :style="{ css属性1: 变量1, css属性2: 变量2, ... }"

data:{

变量1: css属性值1,

变量2: css属性值2

}

b. 为整个style属性设置一个变量对象，其中包含多个css属性和属性值

<元素 style="固定不变的css属性 " :style="xxxStyle"

data:{

xxxStyle:{

css属性1: css属性值1,

css属性2: css属性值2

}

}

(2). 如果希望批量修改一个元素的多个css属性的值时，就要绑定class属性:

a. 为每个class分别设置条件

<元素 class="不变的class " :class="{class1: bool表达式或变量, class2:bool表达式或变量 }"

bool表达式或变量的值为true就应用，false就不应用

b. 为整个class属性设置一个变量对象，其中包含多个备选的class名极其对应判断条件

<元素 class="不变的class " :class="xxxClass"

data:{

xxxClass:{

class1: true或false, //true就应用，false就不应用

class2: true或false //true就应用，false就不应用

}

}

3. 自定义指令: 2步

(1). 希望在页面首次加载时就自动对元素执行一些初始化DOM操作，添加自定义指令:

Vue.directive("指令名",{

inserted(domElem){

对domElem执行DOM操作

}

})

(2). 使用自定义指令: <元素 v-自定义指令名>

4. 计算属性: 如果一个变量值不是现成的，需要根据其他变量的值经过复杂计算才能获得

(1). 定义:

new Vue({

... ,

computed:{

自定义属性名(){

计算过程

return返回值

} //Vue会缓存自定义属性的计算结果，反复使用，避免重复计算！

}

})

(2). 使用计算属性: <元素>{{自定义属性名}}</元素>

5. 过滤器: 如果变量的原始值不能直接给用户看，需要加工后才能给用户看时

(1)不带参过滤器:

a. 添加过滤器:

Vue.filter("过滤器名", function(oldVal){

return 根据不同的oldVal加工出不动的新值

})

b. 使用过滤器: <元素>{{ 变量 | 过滤器 }}</元素>

(2). 带参数过滤器：

a. 添加过滤器: 1 2

Vue.filter("过滤器名", function(oldVal, 自定义形参){

return 根据不同的oldVal加工出不动的新值

})

b. 使用过滤器: <元素>{{ 变量 | 过滤器(自定义实参) }}</元素>

(3). 多个过滤器连用: <元素>{{ 变量 | 过滤器1 | 过滤器2 | ...</元素>

6. axios: 只要在vue框架中发送ajax请求

(0). axios.defaults.baseURL="http://所有服务器端接口统一的域名部分"

(1). axios.get("/接口名", { params:{ 参数:值, 参数:值, ... } })

.then(result=>{

this.xxx=result.data ...

})

(2). axios.post("/接口名", "参数1=值1&参数2=值2&...")

.then(result=>{

this.xxx=result.data ...

})

7. vue生命周期: 4个阶段 8个钩子函数

beforeCreate() 既没有data和变量，又没有虚拟DOM树和页面元素

(1). 创建阶段 create

created() 虽然有data和变量，但是没有虚拟DOM树和页面元素

beforeMount() 虽然有data和变量，但是没有虚拟DOM树和页面元素

(2). 挂载阶段 mount

mounted(){ //此时既有了data对象和变量，又有了虚拟DOM树和页面元素

axios.get(...).then(result=>{

... ...

})

}

beforeUpdate()

(3). 更新阶段 update

updated()

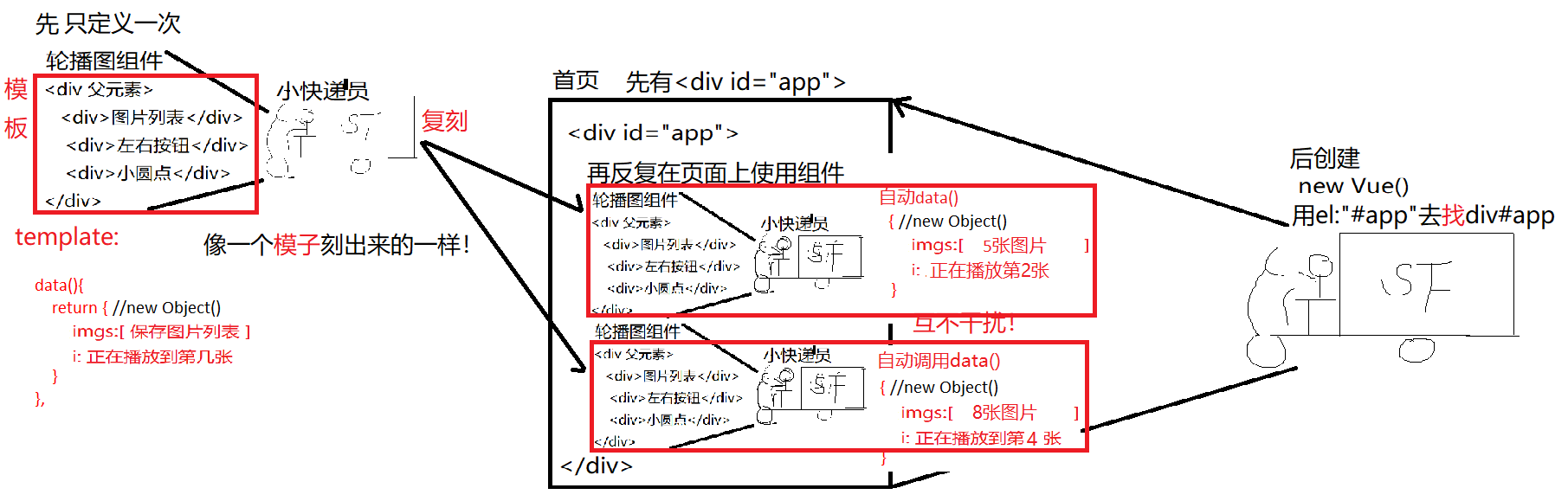
beforeDestroy()

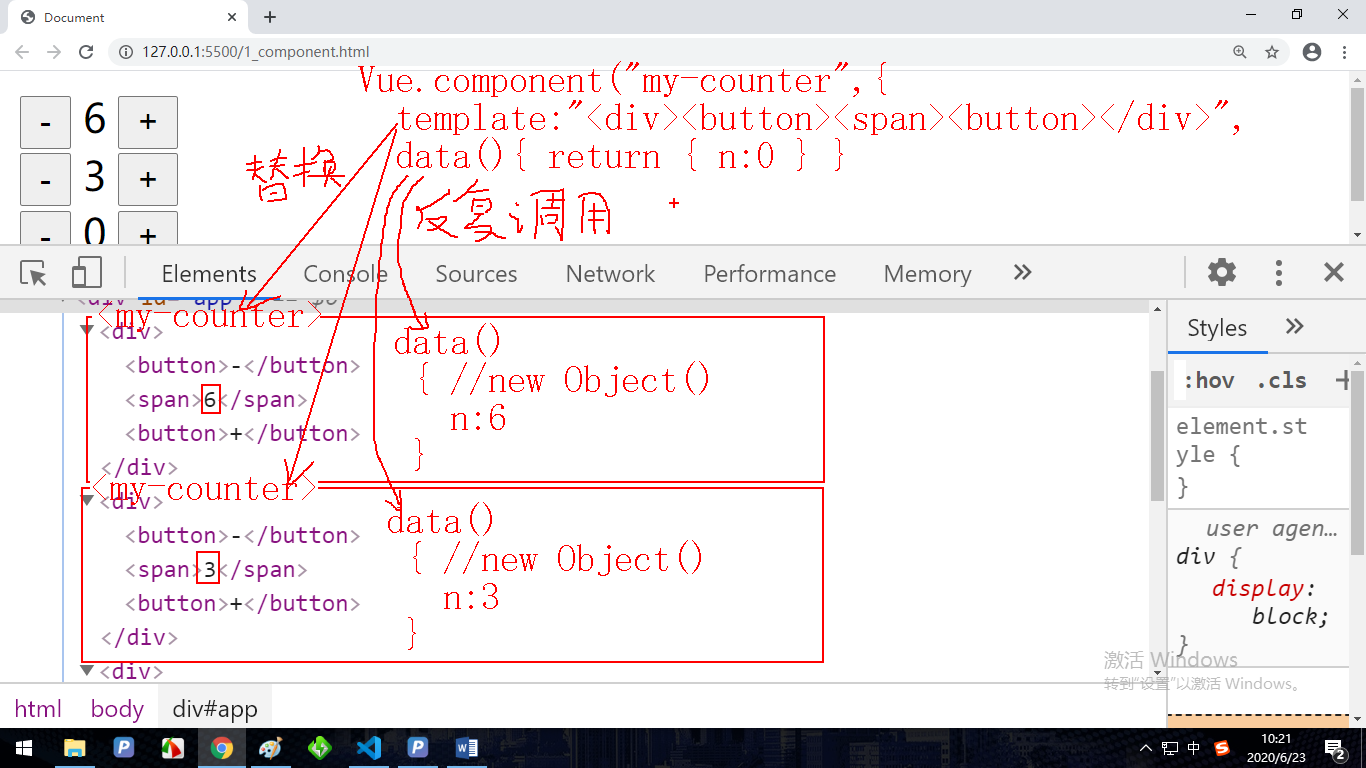
(4). 销毁阶段 destroy

destroyed()

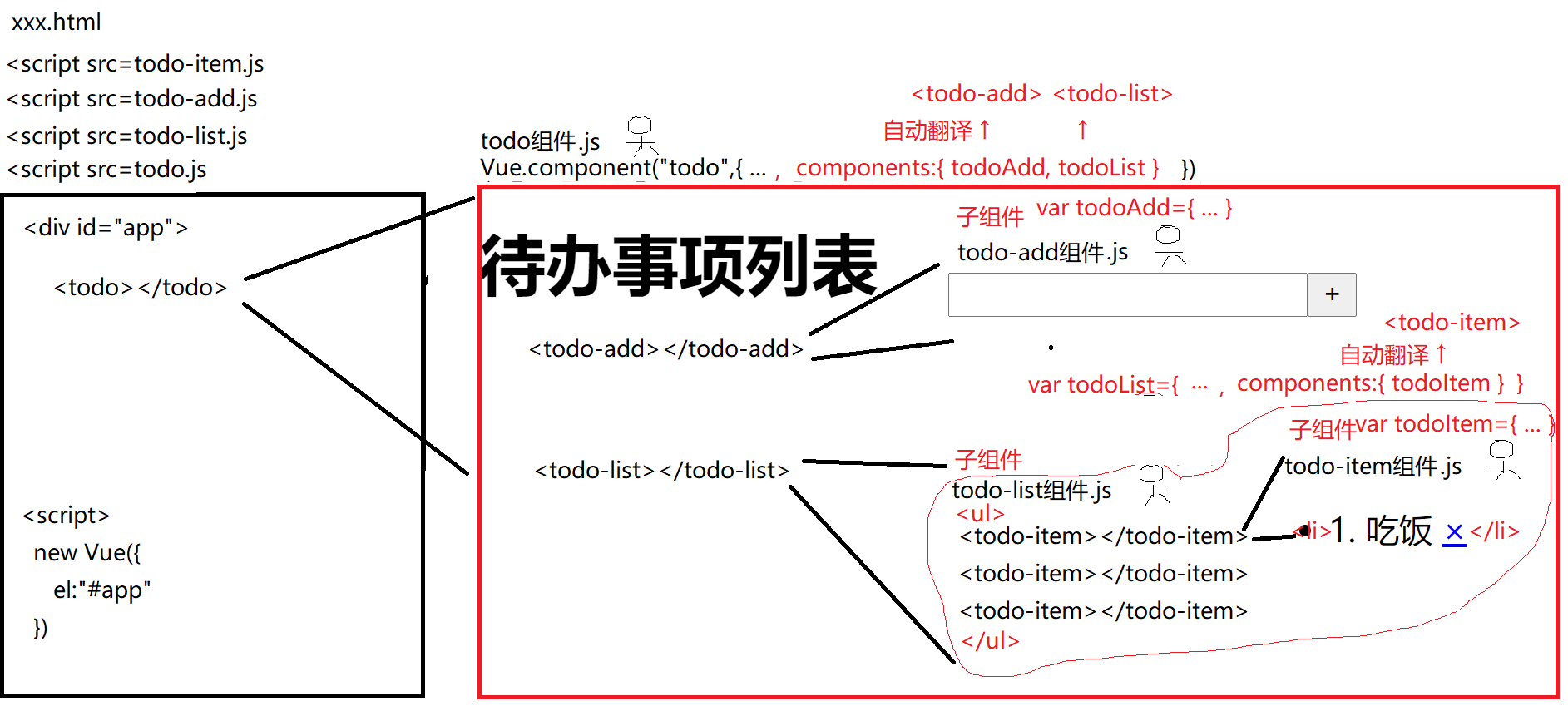
8. 组件化开发:

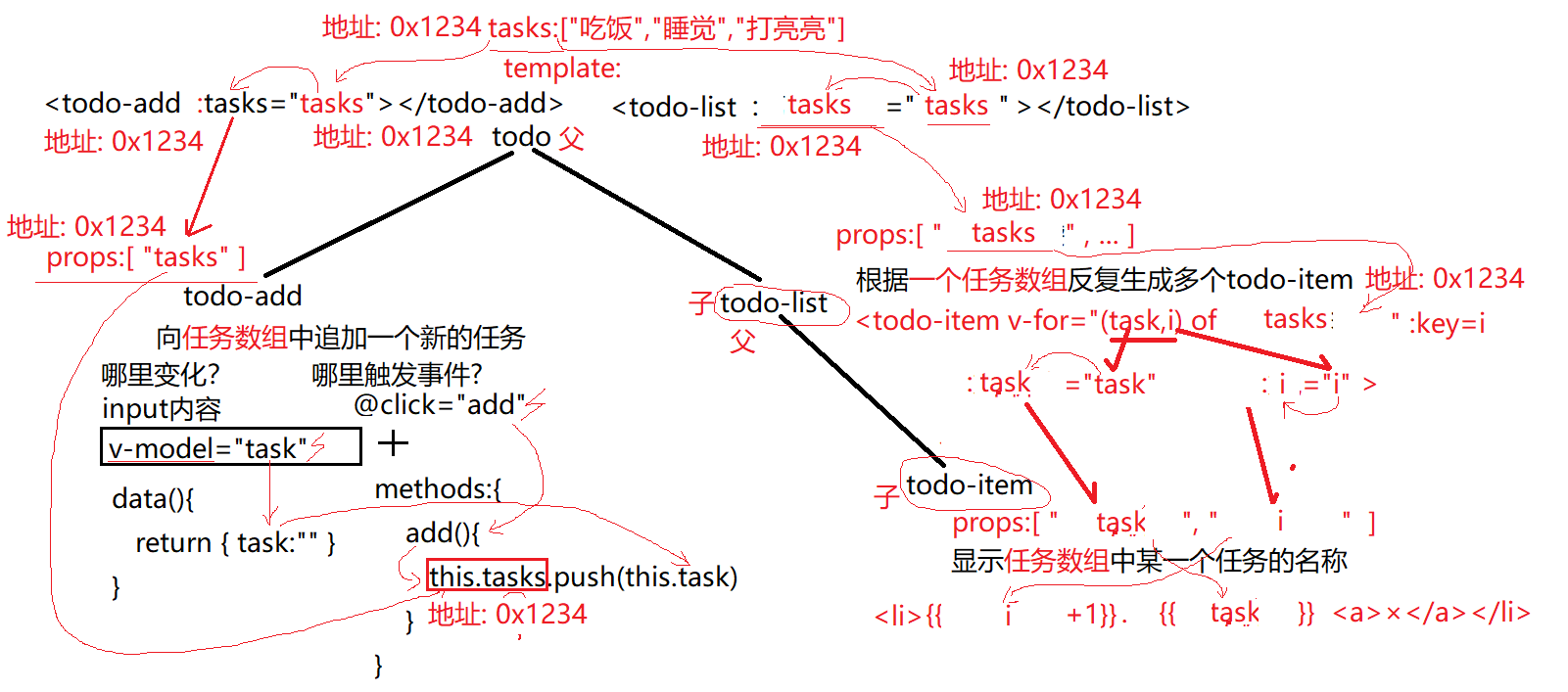
(1). 单个组件:



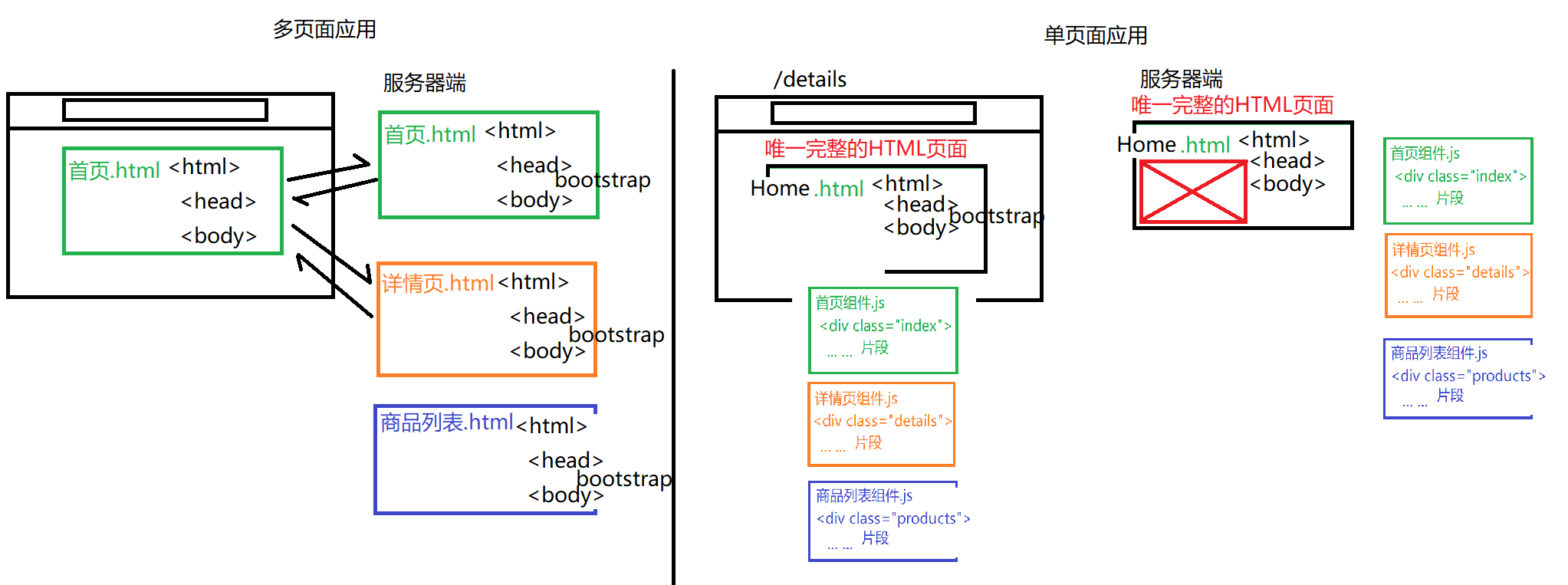


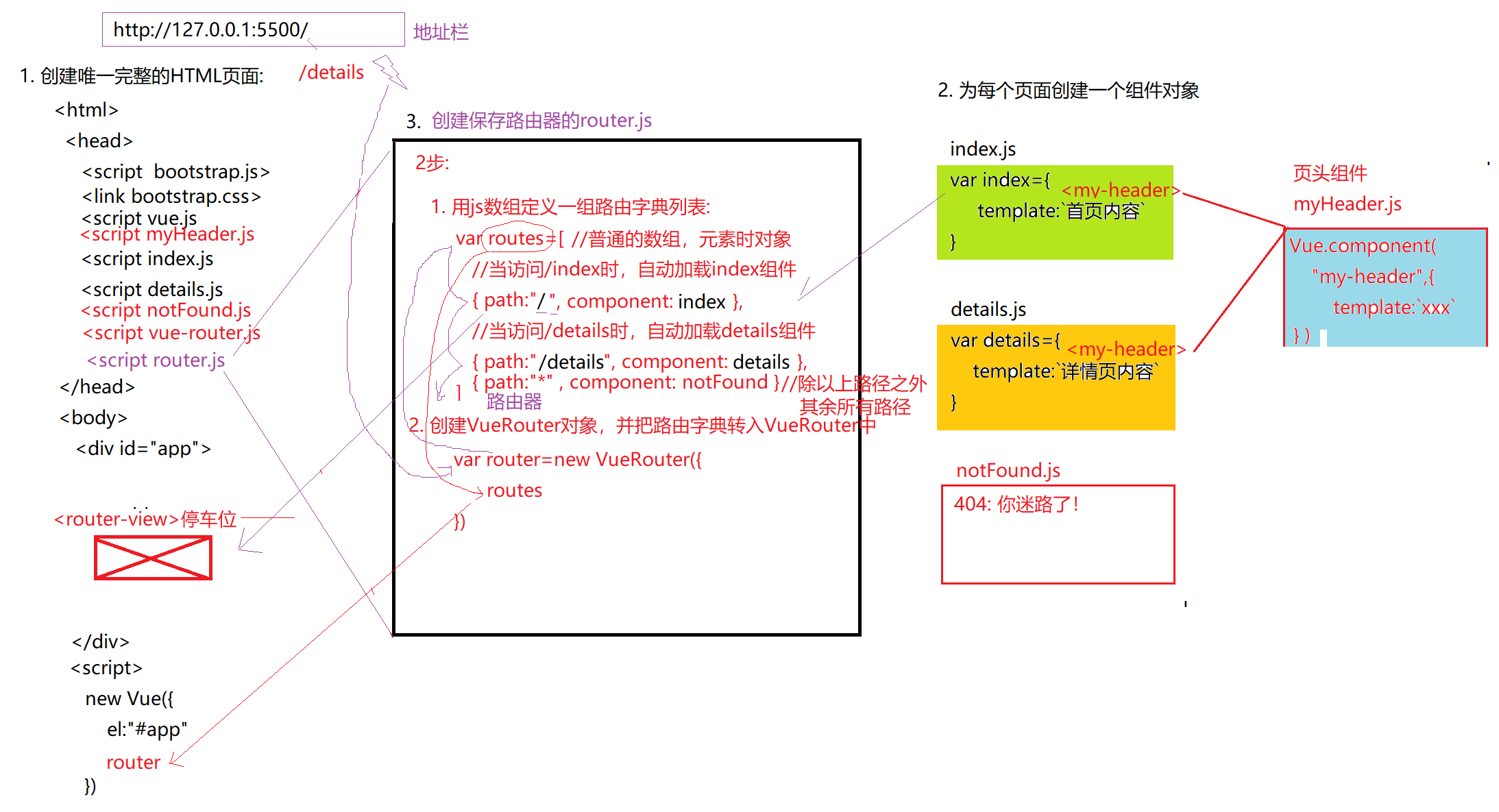
(2). 组件化开发:





9. SPA





路由跳转: 2种:

(1). 在HTML写死的跳转元素: <router-link to="/相对路径">文本</router-link>

(2). 在js中用程序跳转: this.$router.push("/相对路径")

路由传参: 3步:

(1). router.js中routes数组中:

{path:"/相对路径/:变量名", component:组件对象名, props:true}

(2). 前一个页面: 跳转时携带参数: /相对路径/变量值

(3). 后一个页面: props:[ "变量名" ]

10. 脚手架文件夹结构:

(1). 唯一完整的HTML页面:

SPA的index.html等效于public/index.html + src/App.vue + src/main.js 合并后

(2). 为每个“页面”分别创建一个组件对象/.vue文件，放在src/views文件夹下

SPA的页面.js等效于src/views/页面.vue

包含三部分:

<template> 页面的HTML片段

<script> 页面组件的对象

<style scoped> 页面所需的css

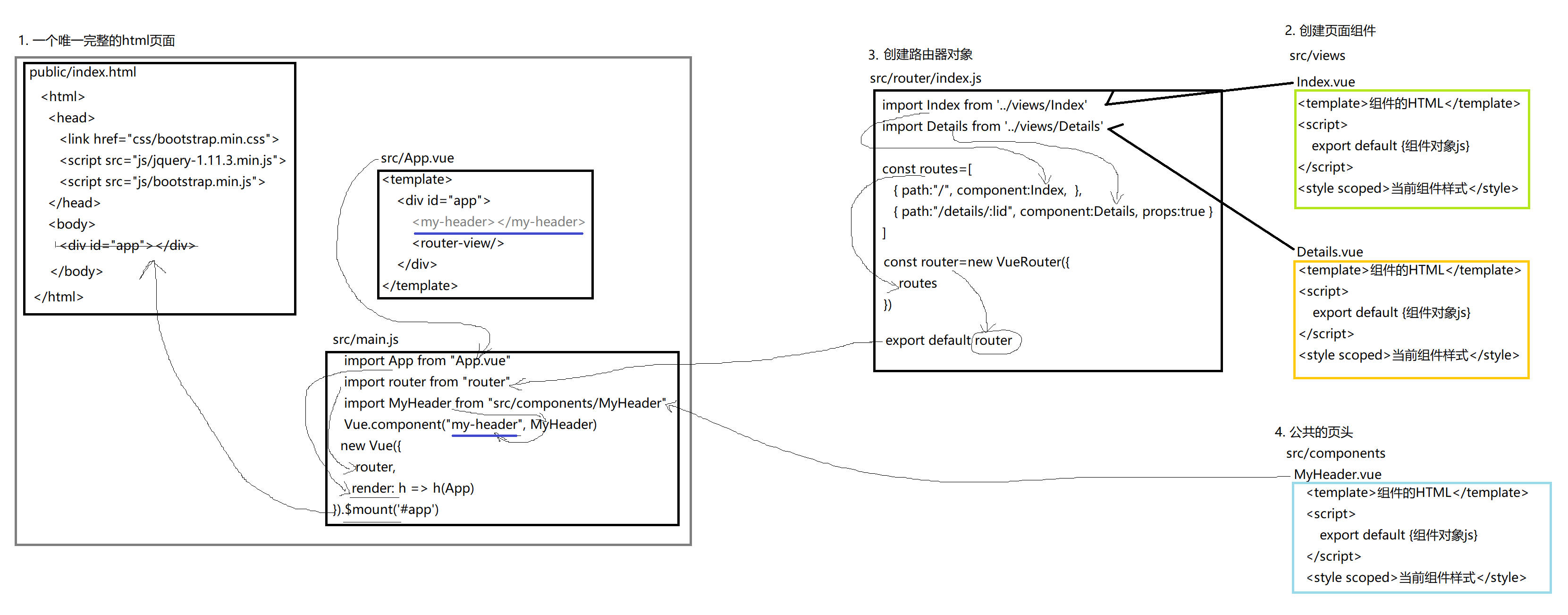
(3). 创建router对象: SPA的router.js等效于src/router/index.js 内容几乎完全一样

(4). 全局组件: 3步

a. 在src/components文件夹中创建组件.vue文件

b. 在src/main.js中引入组件.vue文件，并将引入的组件对象用Vue.component()转化为全局组件

c. 在其它任意组件的<template>中都可用<组件标签名>方式引入全局组件的内容



总结: this 7种:

1. obj.fun() this->obj

2. new Fun() this->new正在创建的子对象

3. 类型名.prototype.共有方法=function(){ ... } this->将来调用这个共有方法的.前的子对象

4. fun() 和 (function(){ })() 和回调函数中的this->window

5. 事件处理函数中的this->当前正在触发事件的这个元素对象

6. $jquery对象.click(function(){ this->将来点击的那个DOM元素对象})

7. new Vue()中的几乎所有this，都指向当前new Vue()对象。（回调函数要用箭头函数）

今日对应小程序视频:

小程序->在线->VUE->day04 2. 使用vue/cli工具创建学子商城脚手架

3. Vue脚手架文件夹结构

作业:

1. 复习今日小程序问题清单(小程序->首页-> VUE ->day04)

2. (选学)看小程序视频学习如何书写简历和面试

小程序->在线->VUE->day05 简历

3. 看小程序视频使用vue脚手架完成学子商城首页、详情页

先下载: 小程序->在线->VUE->day05 0. VUE版学子商城 起始空项目 中

旧js项目: public.zip

vue脚手架起始项目: xzvue\_start\_with\_axios.zip

观看并跟着做: 小程序->在线->VUE->day05

1. 学子商城首页

2. 学子商城详情页 。。。（不带放大镜）

3. 学子商城详情页 。。。（放大镜效果）

家里环境xampp和nodejs能用，且第一阶段学的好的同学:

可下载0. VUE版学子商城 起始空项目 中xzserver.zip，用自己的服务器端代码

家里环境xampp和nodejs不能用，或者第一阶段学的不好的同学：

无需下载xzserver.zip

视频中所有axios请求的服务器端接口地址，都改为我的新浪云服务器地址，就不需要你再运行服务器端了。也不需要你了解任何服务器端知识了:

http://localhost:5050/index 改为 http://xzserver.applinzi.com/index

http://localhost:5050/details 改为 http://xzserver.applinzi.com/details?lid=xxx