我为大家准备了: 微信小程序——“WEB问题速查”——每天必须跟着看

1. 历届师兄师姐的问题清单: 小程序->首页->VUE->day04~day05 点击右侧列表中的问题，就可弹出窗口，查看问题的详情

2. 当天重点视频片段: 小程序->在线->VUE -> day04~day05

3. 当天的单词列表: 小程序->单词-> VUE -> day04~day05

如果tmooc上的视频有问题看不了，可以直接看小程序中的视频

小程序中内容到鄙视面试前看完就行！不用现在都看完！

关于提问: (剩1天)直接私信我是最快的解决途径——谁问的问题越多，谁的培训费交的越值

1. 上课时哪个问题没听明白，立刻问

2. 笔记中哪句话/哪个图不明白，立刻问

3. 小程序中问题清单里哪个问题看不懂，立刻问

4. 如果个人项目中功能不会做，立刻问

5. 刷笔试题时，遇到不会的笔试题，立刻问

正课：

1. 脚手架

2. 模块化开发

3. 懒加载

一. 脚手架:

1. 问题: 当项目中组件非常多时，都会用专门的文件夹来分类组织不同的组件文件。便于查找和维护。但是，人和人创建文件夹的习惯和命名差异很大！结果，一个人建立的项目结构和另一个人建立的项目结构，很可能完全不一样！给开发人员接手和维护新项目带来了巨大的障碍。

2. 解决: 今后前端的所有框架以及后端的nodejs express项目，都已经有了标准化的脚手架项目结构。今后公司中开发，都是基于脚手架的标准项目结构进行开发的。

3. 好处: 一个项目的结构和另一个项目的结构几乎完全相同。开发人员接手和维护新项目时，几乎是无缝衔接！

4. 什么是脚手架: 已经拥有半成品核心功能和标准化文件夹结构的项目源代码。

5. 为什么: 标准化和避免重复编码

6. 何时: 今后所有框架都是先创建脚手架，再在脚手架标准项目结构基础上添加个性化功能定制。

7. 如何:

(1). 创建脚手架项目: 2步

a. 先安装可反复创建脚手架项目的工具: 会下蛋的老母鸡——只安装一次就可反复使用

npm i -g @vue/cli

vue -V

看到版本号，说明安装成功

b. 再反复使用工具，反复创建不同项目的脚手架源代码: 老母鸡下的蛋——每做一个新项目都要重新创建一套标准的脚手架项目源代码

1). vue create 项目名 回车

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

选择 预设 箭头 键

default (babel, eslint)

默认

> Manually select features

手动 功能

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

(\*) Babel 将ES6,ES7代码翻译为所有浏览器都认识的传统的ES5以及以下代码

( ) TypeScript

( ) Progressive Web App (PWA) Support

(\*) Router VueRouter，是实现单页面应用的核心组件

( ) Vuex 第三阶段暂时不选。但是今后所有项目都要选

( ) CSS Pre-processors 如果项目中使用了sass或less，需要对css进行预编译，才选

~~() Linter / Formatter~~ 学习阶段取消选择！linter是代码质量检查工具。

( ) Unit Testing

( ) E2E Testing

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

默认VueRouter采用"#/相对路径"，作为路由地址。但是有人讨厌#，想去掉，就可以开启history模式。一旦开启history，VueRouter采用"/相对路径"作为路由地址。

不要开启history模式，因为需要服务器端首页重定向功能的支持——专业的服务器端技术。

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

期望 放 配置

In dedicated config files 放在多个不同的配置文件中

> package.json 集中放在一个package.json文件中

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

保存这些为 预设 为 今后的项目

因为每个项目的配置可能各不相同，所以保存一套预设，意义不大！

7). 看到successfully说明在硬盘上创建脚手架项目源代码成功！

🎉 Successfully created project xzvue.

👉 Get started with the following commands:

$ cd xzvue

$ npm run serve

(2). 首次运行脚手架项目: 用vscode运行

a. 用vscode打开新生成的项目文件夹xzvue

b. 右键选择package.json文件，选"在终端中打开"，看到打开了一个新终端窗口

c. 当终端窗口中出现: xxx\xzvue>，就输入: npm run serve

1). 自动启动了一个类似于live server和apache的临时服务器，用来保存项目的所有文件，供人们通过浏览器输入地址访问项目中的网页。

2). 将项目源代码编译为所有浏览器都认识的HTML+CSS+JS代码。

d. 看到: DONE Compiled successfully in 2982ms 编译成功

e. 按住Ctrl，点下方链接: Local: http://localhost:8080/，就打开浏览器查看默认的页面了

f. 在打开的浏览器中看到脚手架项目自带的首页

g. 其实脚手架已经包含了单页面应用的核心功能:

1). 包含两个页面: Home和About

2). 公共页头组件

3). 在页头点Home和About，可在两个页面间切换，说明已经创建了VueRouter路由！

8. 改造脚手架项目:

(1). 如果自己开发，不用git，可删除.git文件夹

(2). 脚手架默认不带axios，所以需要安装axios

a. 右键点击package.json，选择"在终端中打开"

b. 看到文字: xxx\xzvue> 输入 npm i -save axios

c. 看到+ axios@0.19.2，说明安装成功。打开node\_modules文件夹，可看到axios文件夹

(3). 试着修改脚手架默认首页的内容:

a. 打开src/views/Home.vue

b. <template>内是脚手架首页的HTML内容

1). 删除<HelloWorld msg="Welcome to Your Vue.js App"/>

2). 换成自己的内容: <h1>我的第一个脚手架项目</h1>

3). 删除<template>下方<script>元素及其内容。

4). 删除src/components/HelloWorld.vue组件文件

c. 如果刚才没有关闭npm run serve，则每修改一次脚手架中源代码，只要Ctrl+S保存修改后，都会自动重新编译，不用反复运行npm run serve。就可以在浏览器中查看到新的修改后的效果。除非不小心把npm run serve关了。才需要重新运行npm run serve。

9. 脚手架项目文件夹结构:

(1). 回顾: SPA应用程序的结构: 4部分:

a. 唯一完整的HTML页面

1). 引用:

i. 引入公共的bootstrap.css、jquery.js和bootstrap.js

ii. 引入Vue单页面应用必须的js文件: vue.js 和 vue-router.js

iii. 引入所有页面组件、全局组件

iv. 引入路由器对象.js

2). 唯一父元素:

<div id="app">

<router-view>停车位</router-view>

</div>

3). new Vue({

el:"#app",

router

})

b. 定义页面组件，习惯上所有页面组件.js放在一个专门的文件夹中, 习惯上名为views

c. 定义VueRouter对象，并定义路由字典，通常放在router.js中

var router=new VueRouter({

routes:[

{path:"/", component:组件名},

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

{path:"\*", component:组件名}

]

})

d. 多个页面共用的页头等组件通常定义为全局组件，习惯上全局组件或个别子组件.js放在一个专门的文件夹中，习惯上名为components

(2). 脚手架文件夹结构: 大致和SPA应用结构几乎完全一样。只不过个别地方划分的更细致而已

a. 唯一完整的HTML页面: 被拆为三部分

1). public/index.html ——几乎不需要我们修改

2). 公共的资源文件bootstrap.css应该放在public/css文件夹，并引入到index.html中

3). 公共的js文件jquery.js和bootstrap.js应该放在public/js文件夹，并引入到index.html中

4). 项目中用到的所有图片，应该放在public/imgs文件夹中

5). 不要在唯一完整的HTML页面中引入vue和vue-router.js了。因为脚手架采用模块化开发，稍后告诉大家vue和vue-router.js在哪里引入。

6). 页面组件和router对象.js不要在唯一完整的HTML页面中引入了。因为脚手架采用模块化开发，页面组件和router对象有专门的路径来保存

7). new Vue()也不要放在唯一完整的HTML页面中了

到此，今后index.html页面再不需要做任何更改！

8). 问题: <router-view>去哪儿了？

src/App.vue文件中保存了:

<div id="app">  
 <router-view></router-view>

</div>

运行时会用App.vue中的HTML内容代替index.html中的空<div id="app"></div>

9). 问题: new Vue()去哪儿了？

src/main.js

import Vue from 'vue' //以模块化方式引入vue.js

import App from './App.vue' //引入App.vue文件中的内容

import router from './router' //以模块化方式引入了new VueRouter()对象和路由字典

Vue.config.productionTip = false

new Vue({ //创建监控整个页面的大的Vue对象（快递员）

router,

render: h => h(App) //加载App中的HTML内容

}).$mount('#app')

挂载到index.html中代替<div id="app">位置

10). npm run serve 先执行main.js，引入App.vue中的内容，并用App.vue中的内容代替了index.html中空的<div id="app"></div>

b. 页面组件:

1). 脚手架规定所有页面组件都应该放在src/views文件夹下

2). 每个页面组件应该定义为一个xxx.vue文件，~~不是.js文件~~

a. .vue中可同时包含一个组件的HTML+JS+CSS

b. .js中只能包含js和简单的HTML，无法包含组件的css内容。

3). 每个.vue文件都可以包含三部分内容:

a. <template> <!--专门保存当前页面的HTML内容-->

<唯一父元素>

当前页面的HTML内容

</唯一父元素>

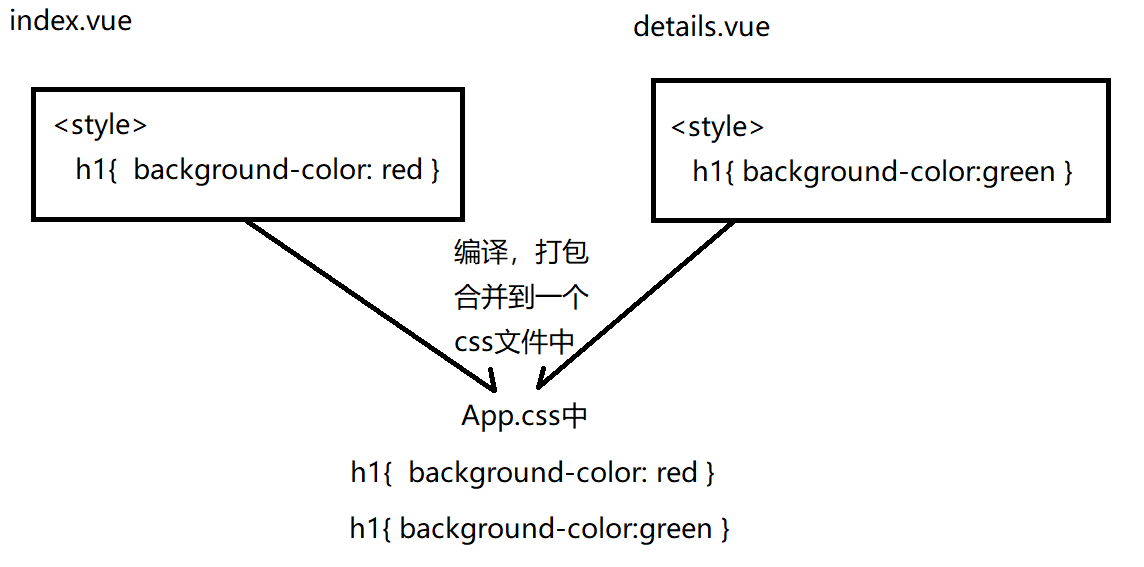
</template>

b. <style scoped>

只写当前页面用到的css样式

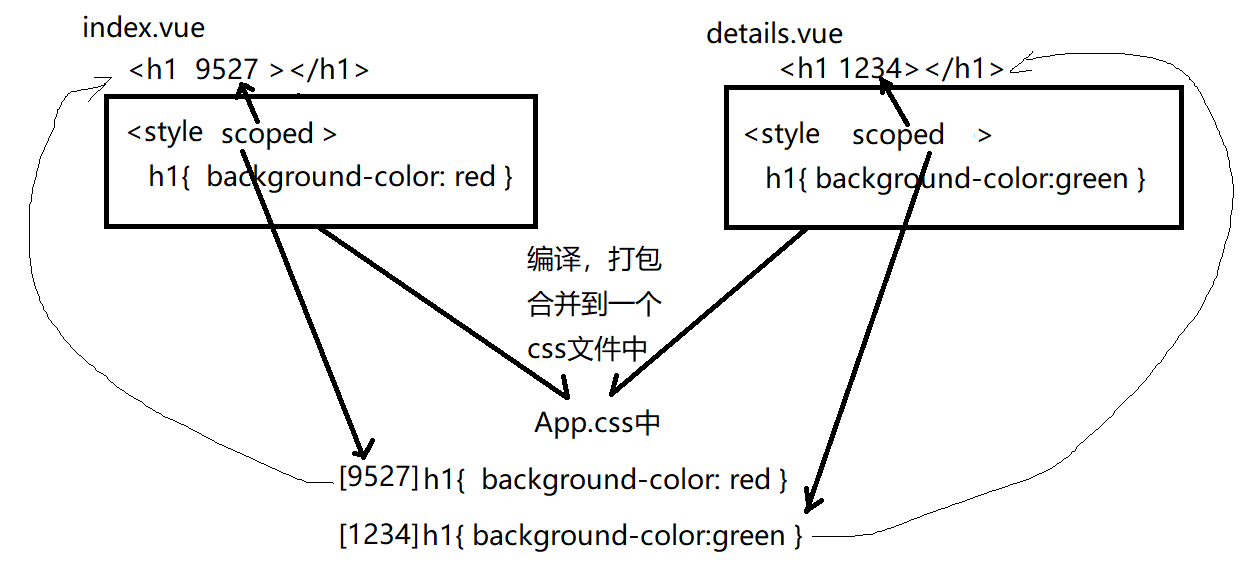
</style>

问题: 因为将来所有.vue文件中的<style>都会被编译到一个.css文件中。极有可能造成不同.vue中的选择器和样式发生冲突！



解决: 今后所有<style>都要写为: <style scoped>

scoped会自动保证不同组件的样式只局限于当前组件，不影响其他组件。



c. <script>

当前页面组件的组件对象js内容

</script>

c. 创建路由器对象和路由字典: src/router/index.js

import Vue from 'vue' //引入vue.js

import VueRouter from 'vue-router' //引入vue-router.js文件

import Home from '../views/Home.vue' //引入views文件夹下的首页.vue文件内容

Vue.use(VueRouter) //引入vue-router.js文件中的VueRouter对象

const routes = [

{ path: '/', component: Home }, //首屏加载的页面

{//其它页面只有在用到时才懒加载

path: '/about',

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

}

]

const router = new VueRouter({

routes //routes:routers

})

export default router //模块化开发: 抛出router对象，供其它文件引用

//如果没有这句话，其它文件，引用不了router对象。

d. 全局组件:

1). 集中创建在src/components文件夹下

2). 每个全局组件一个.vue文件

3). 在定义组件时，全局组件和页面组件的内容没有任何差别，也都是包含三部分

a. <template>组件的HTML内容</template>

b. <style scoped>组件的css内容</style>

c. <script>组件对象的js内容</script>

4). 必须在main.js中，引入components下的全局组件.vue文件，然后再用Vue.component()函数，将引入的组件内容变成全局组件:

import MyHeader from "./components/MyHeader"

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

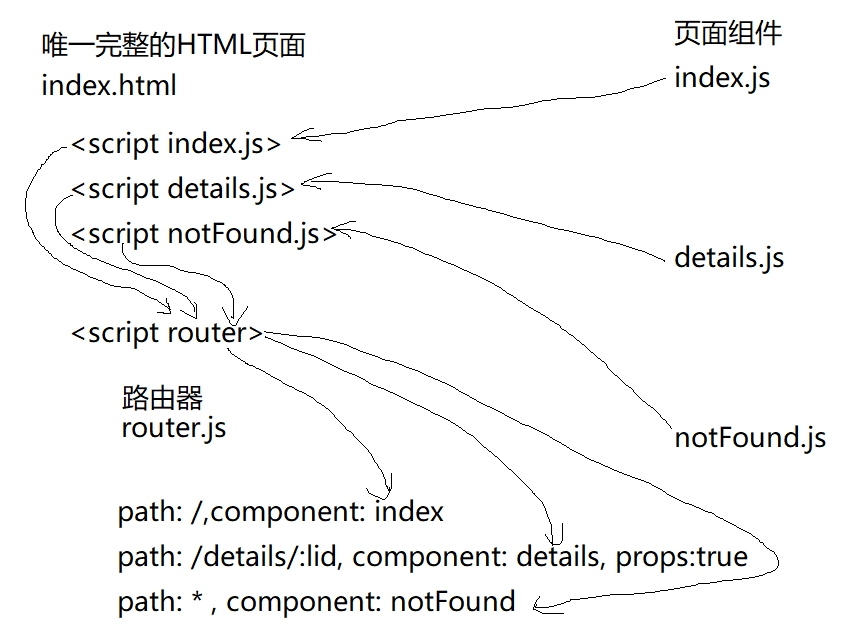
5). 结果在src/views文件夹下的页面.vue中可用

<my-header></my-header>引入全局组件页头

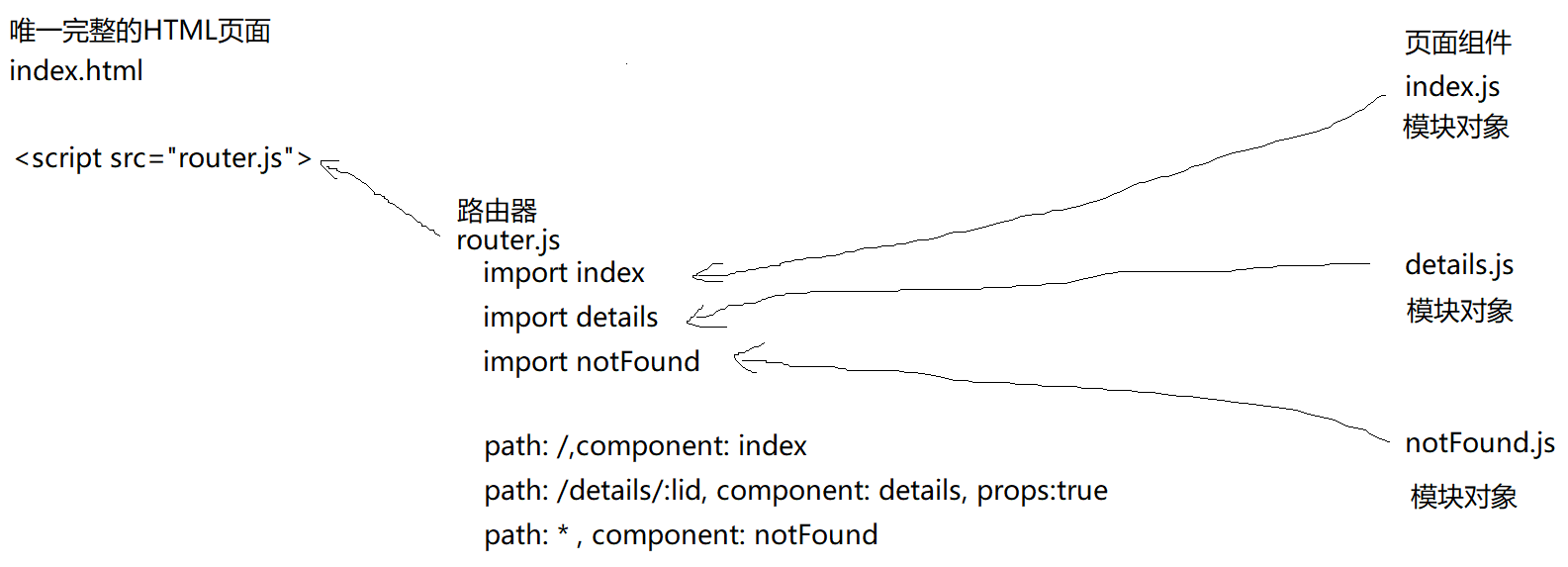
二. 模块化开发:

1. 什么是: 将每个页面和每个组件都看作是一个模块对象。

2. 问题: 没有模块化开发时，网页中一个对象要使用其他组件对象，必须先将所有组件对象都引入.html中集中起来。再一个页面内，挑选哪个对象使用哪个组件。——不直观，不便于理解，不便于维护



3. 模块化开发好处: 哪个模块想使用另一个模块中的内容，可直接引用！不需要全都引入到html文件中，再使用。——直观，便于理解，便于维护



4. 如何：2步:

(1). 组件对象需要创建js对象，并将js对象抛出到外部，供别人选择引用

export default {//new Object()

对象的js成员

}

(2). 其他组件可使用以下方式，引入另一个组件抛出的组件对象

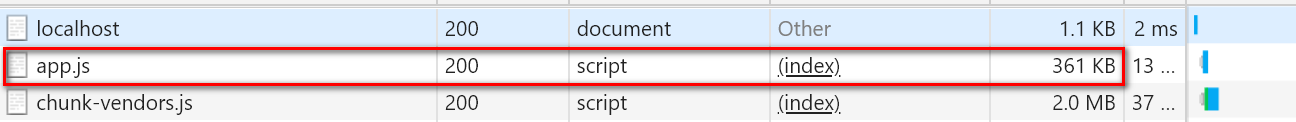
import 自定义变量名 from "另一个组件文件的相对路径"

(3). 结果: 在import所在的组件中，可通过自定义变量来使用引入的另一个组件的内容

5. 特例: 如果一个组件文件中没有js代码，则不需要export default，文件本身默认就是一个模块对象，就可以被别人import导入！

三. 懒加载:

1. 问题: 默认脚手架会将所有以import方式引入的组件的HTML+CSS+JS等编译到一个app.js文件中——这个文件会很大！且首屏下载慢！



2. 解决: 懒加载

3. 2种:

(1). 在编译时，将不同的组件编译到不同的js文件中，首屏加载时，先请求首屏组件的js，然后在低层悄悄异步下载其余组件的js——后续组件不影响首屏加载速度

a. 如何: 2步都是在src/router/index.js中修改

1). 除首页组件之外，不要用import方式过早引入其余页面组件

2). 除首页之外，其余路由字典中的对象，一律采用懒加载

{

path: '/details/:lid',

//只有当用户访问到/details/:lid路径时，才临时执行箭头函数，才临时执行import(...)

///\*\*/注释不能删除，它为将来分块打包时，当前组件所在的js文件起文件名。

//比如: webpackChunkName: "details"

//结果: Details.vue会被编译为一个details.js文件

component: () => import(/\* webpackChunkName: "details" \*/ '../views/Details.vue'),

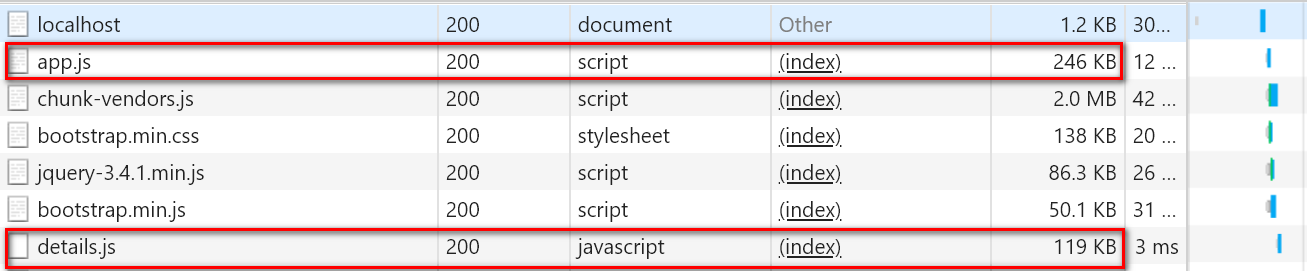
props:true

}

b. 结果:

1). 首屏index.vue和details.vue分别打包为两个js文件

2). 首屏先下载app.js，然后再低层异步下载details.js，不影响首屏加载速度



c. 问题: 不管用户是否会访问详情页，都始终悄悄下载details.js文件——偷跑流量

(2). 在编译时，将不同的组件编译到不同的js文件中，首屏加载时，先请求首屏组件的js，但是如果用户暂时不访问其他组件时，就不提前下载其他组件的js文件

a. 如何:

1). 前两步和第一种方法相同！

2). 但是需要多一步: 在脚手架项目的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") //取消预加载

}

}

b. 结果:

1). 首屏加载时只加载app.js，只包含首页组件内容

2). 只有用户访问/details/5时，才临时下载details.js文件

如果用户不访问/details/5时，是不会提前下载details.js文件的。——省流量

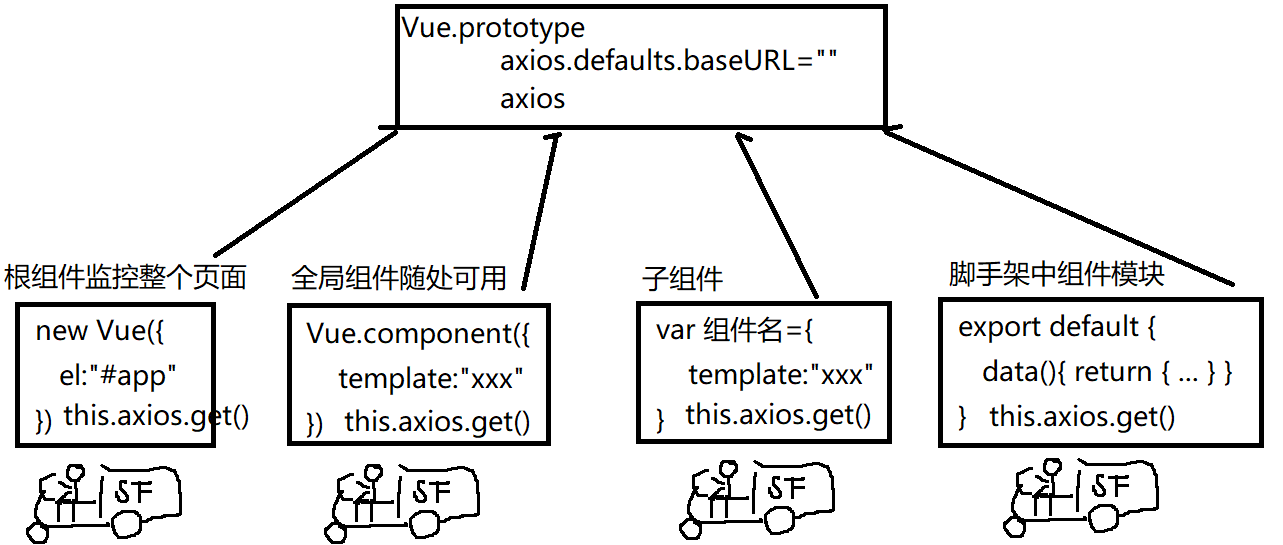
c. 问题: 其他组件加载可能稍慢

四. 脚手架中axios的使用:

1. 问题: 笨办法，可以在每个组件中都引入axios，繁琐！

2. 解决: 其实只在main.js中引入一次axios，然后将axios放入Vue的原型对象中。所有组件对象都可以使用this.axios方式来使用统一的axios对象。

3. 原理: 因为所有组件都是Vue类型的子对象，都继承自Vue的原型对象。所以，在vue的原型对象中添加一个axios，则所有子对象不用重复创建，就可直接使用。



4. 如何:

1). main.js中:

import axios from "axios"

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

Vue.prototype.axiso=axios;

2). 其余任何组件内:

this.axios.get("/相对路径").then(result=>{ ... result.data... })

总结:

1. MVVM:

(1). 界面(View) 增强版，支持变量、分支、循环等功能

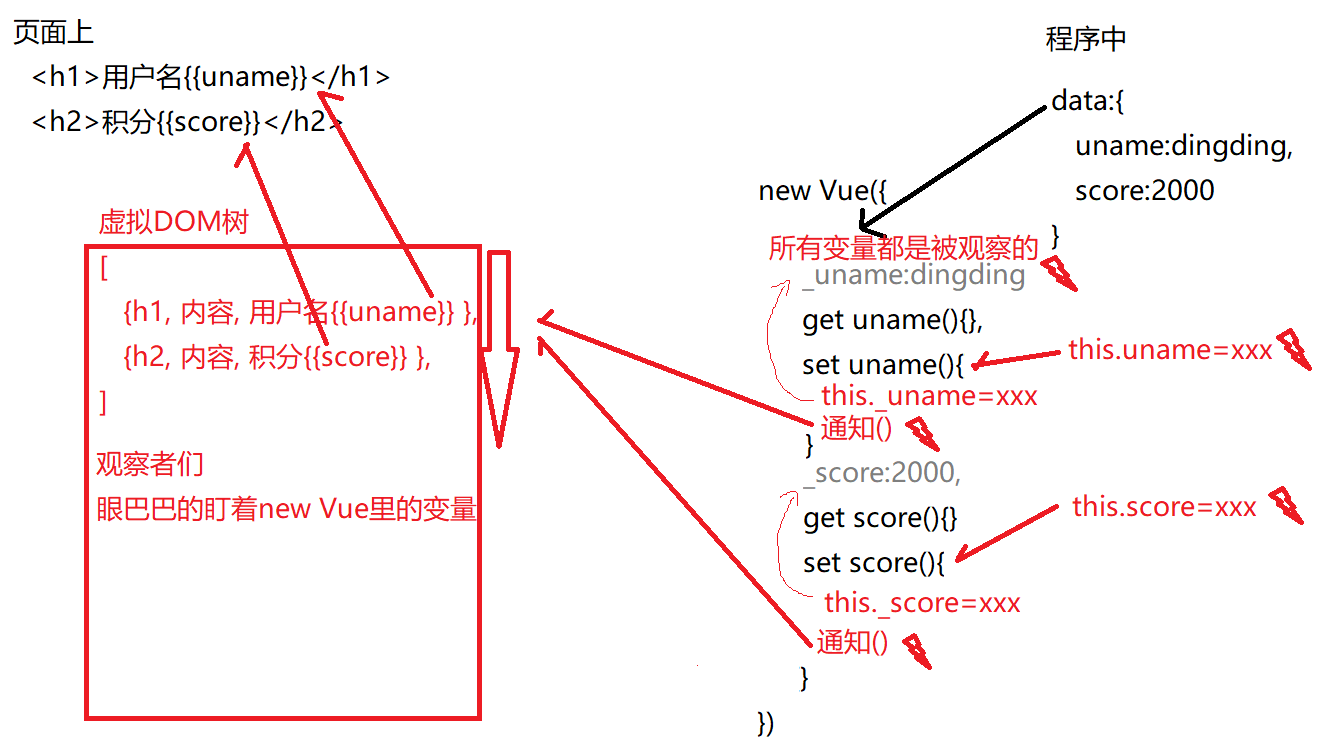
(2). 模型对象(Model) 保存界面所需的一切变量和函数:

(3). 视图模型(ViewModel) 快递员 模型对象中的变量值和函数自动送到界面中所需的位置。并始终保持同步！

2. Vue绑定原理:

(1). 简单: 访问器属性+虚拟DOM树

(2). 完整: 访问器属性+观察者模式+虚拟DOM树



3. 虚拟DOM树:

(1). 什么是: 仅保存可能变化的DOM元素的简化版DOM树  
 (2). 优点: 4个

a. 小: 仅保存可能变化的DOM元素

b. 快: 因为小，所以遍历快

c. 减少重复代码: 自动对受影响的元素执行增删改查操作

d. 更新效率高: 只更新受影响的元素。不受影响的元素，保持不变的！

4. Vue功能: 3步——招数

(1). 编写界面: 要求:

a. 整个界面必须包裹在一个唯一的父元素内，且父元素习惯上id="app"

b. 所有可能发生变化的地方，用{{变量名}}或指令方式标记

c. 所有触发事件的元素，用@事件名="函数名"标记

(2). 先创建new Vue()对象：

new Vue({

el:"#app", //选择器

... ...

})

(3). 为new Vue()添加data和methods模型对象

a. 界面中所需的所有变量都放在data对象中

b. 界面中所需的所有函数都放在methods对象中

c. methods中的函数，想操作data中的变量，必须加this.

5. 绑定语法和指令: ——招数

(1). 如果元素的内容可能随变量自动变化:

<元素>{{变量名}}</元素> 或 <元素 v-text="变量或js表达式"></元素>

(2).只要元素的属性值可能随变量自动变化: :属性名="js表达式"

(3). 控制一个元素显示隐藏: v-show="返回bool值的js表达式"

(4). 控制两个元素二选一显示: <元素1 v-if="返回bool值的js表达式">

<元素2 v-else>

(5). 控制多个元素多选一显示：<元素1 v-if="返回bool值的js表达式">

<元素2 v-else-if="返回bool值的js表达式">

... ...

<元素n v-else>

(6). 反复生成多个相同结构的元素：

<要反复生成的元素 v-for="(value,i) of 数组/对象/整数" :key="i">

(7). 绑定事件: <元素 @事件名="处理函数($event, 实参值)">

(8). 绑定原始HTML代码内容: <元素 v-html="变量"></元素>

(9). 防止用户短暂看到{{}}:

<元素 v-cloak>...</元素> 或 <元素 v-text="变量或js表达式"></元素>

(10). 今后只要一个绑定的值，在首次加载之后不再发生变化:

<元素 v-once>{{变量或js表达式}}</元素>

(11). 防止元素内容中的{{}}被new Vue()编译，保持{{}}原样显示:

<元素 v-pre>{{xxx}}</元素>

(12). 只要想获得用户在表单元素中输入的新值:

<元素 v-model ="变量名">

6. 绑定样式:

(1). 绑定内联样式: 只要仅单独修改一个css属性时，可首选绑定:style

<元素 style="固定不变的css属性" :style="变量">

new Vue({

data:{

变量: {

Css属性:"属性值",

... : ...

}

}

})

(2). 绑定class: 如果批量修改一个元素的多个css属性，首选绑定class

<元素 class="固定不变的class" :class="变量">

new Vue({

data:{

变量: {

Class1: true或false,

... : ...

}

}

})

7. 创建自定义指令: 只要希望在元素加载到页面上时，自动对元素执行一些初始化操作

(1). 定义

Vue.directive("指令名",{

inserted(elem){

对当前带有指令的elem元素执行原生的初始化DOM操作

}

})

(2). 使用: <元素 v-指令名>

8. 计算属性: 只要想要的值没有直接给出，需要根据其他属性值经过复杂的计算才能获得

(1). 定义:

new Vue({

el:"#app",

data:{ },

methods:{ },

computed:{

计算属性名(){

根据其他属性值经过复杂计算得到一个结果

return 计算结果

}

}

})

(2). 使用: <元素>{{计算属性名}}</元素>

(3). 如何选择:

a. 如果更关心执行结果的数值时，首选computed

b. 如果只关心执行过程，不关心执行结果，首选methods

9. 过滤器: 只要变量的原始值不能直接给人看，需要经过加工才能显示给人看时

(1). 定义:

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

return 根据每次自动传入的变量原始值oldVal，加工后生成新值返回

})

(2). 使用:

<元素>{{变量 | 过滤器1(实参值) | 过滤器2 | ...}}</元素>

10. axios:

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

(2). axios.get("/接口地址相对路径",{

params:{ 参数:值, ... }

}).then(result=>{

result.data ...

})

(3). axios.post("/接口地址相对路径","参数=值&...")

.then(result=>{

result.data ...

})

11. 生命周期:

beforeCreate

(1). Create

created

beforeMount

(2). Mount

mounted

beforeUpdate

(3). Update

updated

beforeDestroy

(4). destroy

destroyed

12. 组件:

(1).定义全局组件:

Vue.component("组件名",{

template:`<唯一父元素>

... ...

</唯一父元素>`,

data(){

return { ... }

},

//其余和new Vue都是完全一样的

})

(2). 使用: <组件名></组件名>

12. 组件化开发:

(1). 拿到一个页面先划分子组件

(2). 创建子组件:

a. var 子组件对象={

//和Vue.component()中内容完全一样

}

b. 在父组件对象内:

父组件:{

components:{ 子组件对象 }

}

(3). 父给子传值: 2步

a. 父亲给:

父组件:{

template:`

...

<子组件 :自定义属性="父组件变量"></子组件>

...

`

}

b. 孩子收:

子组件:{

props:["自定义属性"],

template:`

...将自定义属性用于绑定或指令

`,

methods:{

函数(){

... this.自定义属性 ...

}

}

}

13. 单页面应用:

(1). 先创建唯一完整的html页面

<html>

<head>

<link bootstrap.css

<script jquery.js

<script bootstrap.js

<script src="js/vue.js"

<script src="js/vue-router.js"

<body>

<div id="app">

<router-view>停车位</router-view>

<script>

new Vue({

el:"#app",

router

})

(2). 有几个页面，就定义几个子组件，额外多定义一个notFound组件

var index={ template:`...` }

var details={ template:`...` }

var notFound={ template:`...` }

以上这些组件也要引入到唯一完整的HTML页面开头

(3). 创建路由器对象和路由字典

var router=new VueRouter({

routes:[

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

{ path:"/details/:参数名" component: details, props:true },

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

]

})

一定要将路由字典对象加入到唯一完整的HTML页面中new Vue()内

(4). 多个页面都要使用的组件，可定义为全局组件，比如页头

Vue.component("my-header",{

... ...

})

将来在任何想加载页头的地方都可以<my-header></my-header>

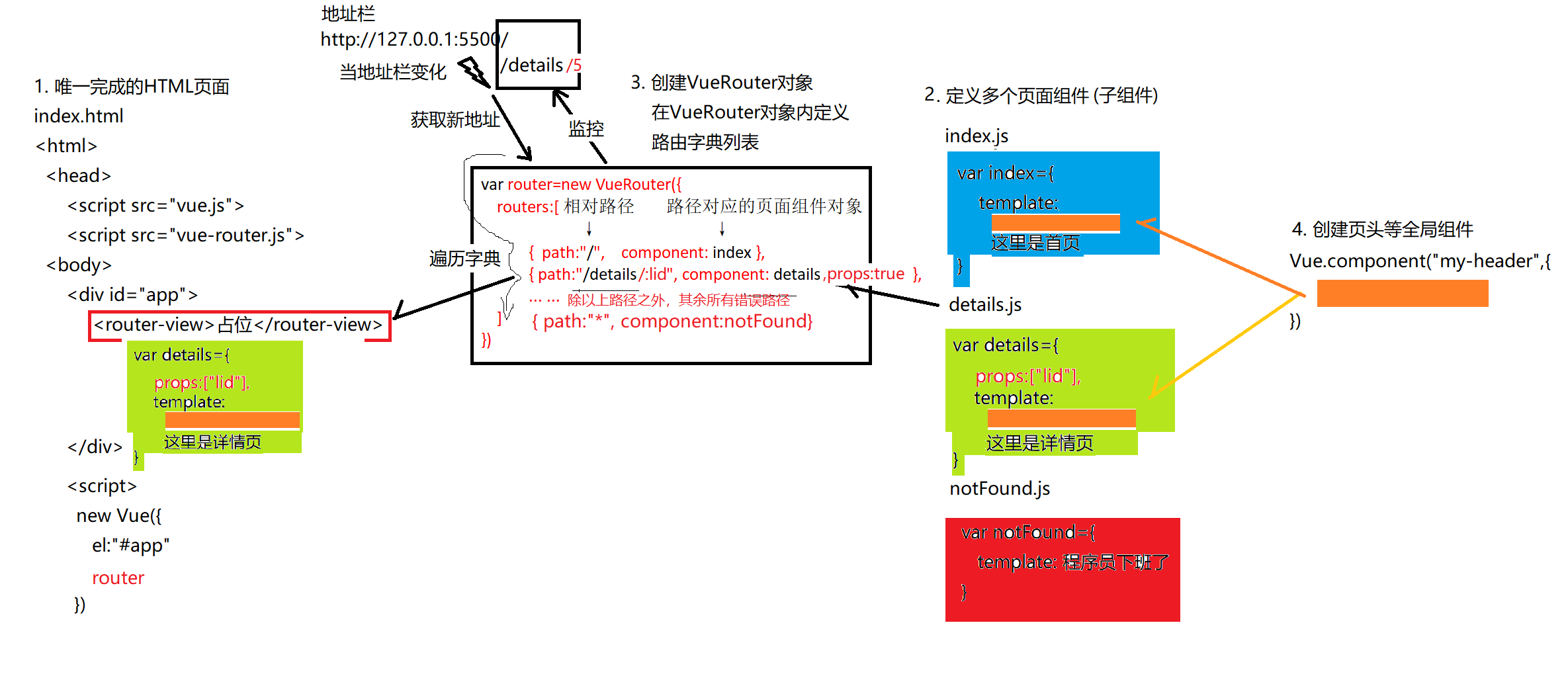
(5). 路由跳转:

a. HTML中写死链接: <router-link to="/相对路径/参数值"></router-link>

b. js中用程序自动跳转: this.$router.push("/相对路径/参数值")

c. 下个页面中如何接收上个页面传来的参数值:

props:["参数名"]



14. 脚手架文件夹结构:

(1). public/

css/ 第三方的css文件压缩版本

js/ 第三方的js文件的压缩版本

img/ 当前网站所需的一切图片

index.html 中需要先引入第三方的css文件和js文件

(2). src/

App.vue

<template>

<div id="app">

//如果有公共的页头，可以放这里

<router-view></router-view>

</div>

</template>

<style>

//不分页面，整个项目通用的样式

//比如css初始化等

</style>

main.js:

... ...

import 全局组件对象 from "./components/全局组件"

Vue.component("全局组件标签", 全局组件对象)

new Vue({

})

views/

页面组件.vue

<template>

<唯一父元素>

如果需要页头，可放这里

当前页面的HTML内容

</唯一父元素>

</template>

<style scoped>

仅当前页面可用的css

</style>

<script>

export default {

//没有template之外

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

methods:{ ... },

mounted(){ ... },

computed:{ ... }

}

</script>

router/

index.js

... ...

import 首页 from '../views/首页'

const routes=[

{ path:"/", component: 首页 },

{

path:"/details/:lid",

component:()=>import xxx,

props: true

},

{path:"\*", component: 404页面}

]

var router=new VueRouter({

routes

})

export default router

components/

全局组件.vue

作业:

1. 复习今日问题清单(vue->day05)

2. 看小程序视频完成学子商城项目(周四+周日做即可)

小程序->在线->VUE->day05

下载0. Vue版学子商城 起始空项目中

xzvue\_start\_with\_axios.zip，解压后，跟着视频做

public中是学子商城静态页面+css，用于移植到脚手架中

需要做:

0. 学子商城首页xzvue index ...

1. 学子商城详情页 静态HTML

2. 学子商城详情页

3. 学子商城放大镜效果

3. （学有余力）看小程序视频学习实现学子商城商品列表页面

小程序->在线->VUE->day05 4. 学子商城 商品列表页面 vue 分页...

4. 所有人，整理我微信朋友圈里师兄师姐反馈回来的鄙视面试重点关键词列表。并在小程序中找到对应的视频，在鄙视面试之前反复看，三遍起。