正课

1. 组件化开发:

2. SPA

3. 脚手架

一. 组件化开发:

1. 组件间传递数据: 3种

(1). 父给子: 2步

a. 在父组件内，通过绑定语法，将父组件的数据绑定给子组件标签上的一个自定义属性。

父组件{  
 template:`  
 <子组件 :自定义属性名="父组件成员变量"></子组件>  
 `  
 }

b. 在子组件内，通过props可以访问到子组件标签上的自定义属性绑定的值:

var 子组件={

template:` ... `,

props: [ "自定义属性名" , ... ],

data(){...}, //是自己创建的或自己请求来的数据

...

}

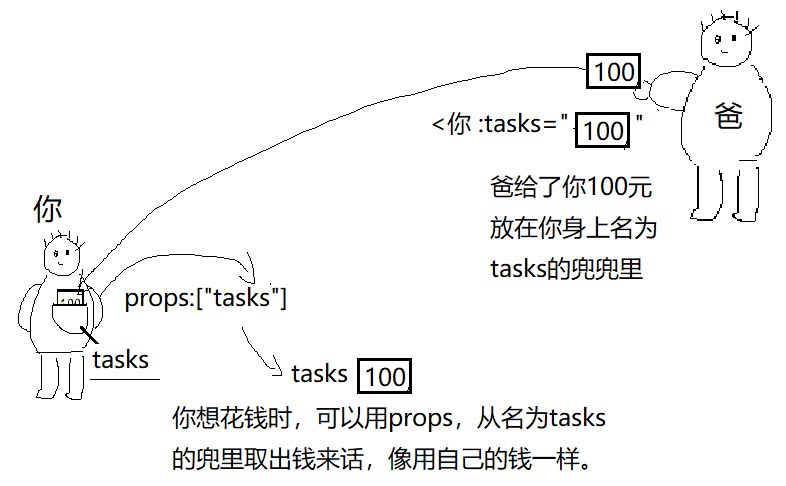
props中的变量和data中的变量:

相同点：都可用作绑定

不同点：来源不同:

data(){...}, //是自己创建的或自己请求来的数据

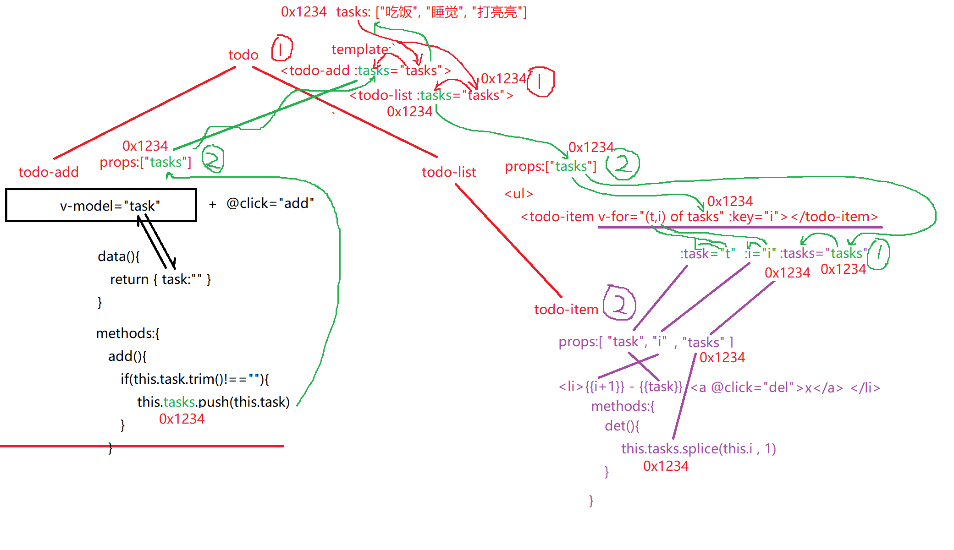
props:[], //父组件给的数据



c. 结果: 在子组件内，可用props中的自定义属性名，来使用父组件给的数据

1). 如果父组件给的是原始类型的值，则只能用，不能改。因为按值传递，原始类型的值传递的是副本。修改新变量的值，不影响原变量。

2). 如果父组件给的是引用类型的对象，则按照按值传递，传递的是对象地址。在子组件中修改新变量，等效于修改父组件中原对象。父组件同样受影响。



(2). 子给父:(看视频)

(3). 兄弟间:(看视频)

二. SPA: (Single Page Application)

1. 什么是: 整个应用程序，只有一个完整的.html页面。所谓的多个"页面"，其实是多个组件。切换不同的"页面", 其实.html文件没换，只是在一个.html文件内，替换不同的组件片段而已。

2. 为什么:

|  |  |  |
| --- | --- | --- |
|  | 多页面 | 单页面 |
| 请求次数 | 每切换一次页面，一定会重新向服务器发送一次请求。——请求次数多 | 只在首次就把所有组件一次性加载到客户端，之后即使切换"页面"，也只是在客户端本地切换组件而已，无需重复向服务器发送请求——请求次数少 |
| 页面加载效率 | 每切换一次页面，都要重建整棵DOM树——效率低 | 只在第一次记载时创建整课DOM树，之后切换"页面"，只是局部更新DOM树的内容。——效率高 |
| 共享资源 | 每个页面都要重新引用共享的资源文件（jQuery, bootstrap, Qs, axios。。。）,只要切换页面，就要重新从服务器下载共享资源文件。——请求次数多 | 资源文件只在唯一完整的HTML文件上引用一次。之后每次切换页面，不需要重新从服务器下载共享资源文件。——请求次数少 |
| 页面切换动画 | 不可能实现页面切换动画。因为两个页面不可能同时存在于浏览器同一个窗口中，所以不可能实现页面过渡动画。 | 容易实现两个"页面"间切换的动画效果。 |

3. 如何:

(1). 创建一个唯一的完整的支持vue框架的html文件

在<body>中使用<router-view>为将来页面组件预留空间。

(2). 创建多个页面组件（当做一个页面使用的子组件）

(3). 创建路由字典数组:

a. 什么是路由字典: 保存多组相对路径和页面组件对象对应关系的数组。

b. 如何: var routes=[

{path:"/", component: 默认首页组件对象},

{path:"/index", component: 一个页面组件对象},

{path:"/details", component: 一个页面组件对象},

... ...

{path:"\*", component: 404页面组件对象}

]

比如: var routes=[

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

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

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

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

]

|  |
| --- |
| 强调: component: 后的对象名一定不要加引号，因为都是页面组件对象的变量名 |

(4). 创建路由器对象，将路由字典装入路由器对象中

a. 路由器对象:

1). 包含路由字典

2). 监控地址栏变化

3). 只要地址栏变化，就根据地址栏中的相对路径，自动查找对应的页面组件，替换到唯一完整的html页面中，替换<router-view>的位置。

b. var router=new VueRouter({ routes })

(5). 将router装入new Vue()中：

new Vue({

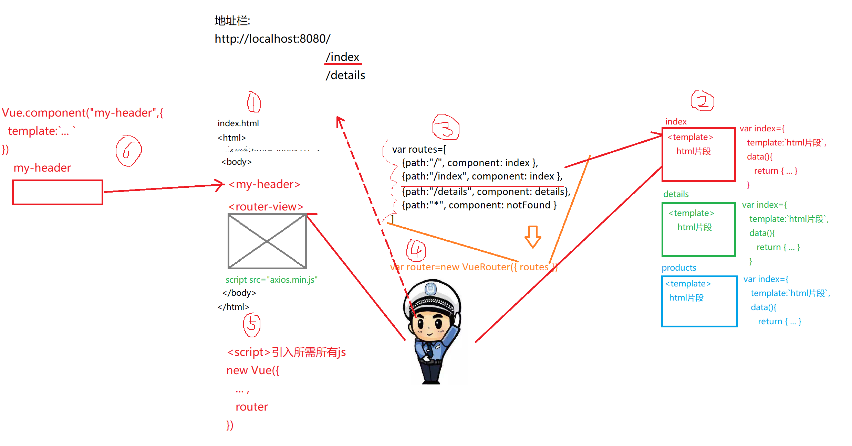
el:"#app",

router,

...

})

(6). 在唯一完整的HTML页面顶部，先引入vue-router.js，再引入所有页面组件.js和路由器.js



|  |
| --- |
| 所有页面公用的组件，比如页头部分，需要单独定义为全局组件。 |

4. 路由跳转：

(1). html中跳转：不要用a！用<router-link>代替  
 <router-link to="相对路径">文字</router-link>

在运行时会被自动翻译为:

<a href="完整路径">文字</a>

(2). js中跳转：this.$router.push("/相对路径")

$router就是new VueRouter() 路由器对象。除了可以自动切换页面组件外，还可以用于主动调用跳转新页面。

(3). 页面跳转时传参:

a. 在路由字典中，让路由地址支持传参:

var routes={

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

}

props:true，可让参数变量lid，自动成为props中一员属性，并在页面组件中用于绑定。

强调: 一旦定义了/details/:lid路径，再使用/details不带参数，就会被导到404.

b. 在下一个页面组件的对象中添加:

var details={

template: `html片段`,

props:["lid"],

...

}

c. 跳转时传值：

1). html中: <router-link to="/details/1"

如果遇到绑定: <router-link :to="`/details/${变量}`"

2). js中: this.$router.push("/details/2")

强调: /后不要加: ，直接加参数值即可！

c. 结果: 下一个页面的组件对象中的props中就会多出一个lid属性，值为1或2。且这个lid变量，可在下一个页面组建中用于绑定和程序计算。

三.脚手架：

1. 什么是脚手架: 一个已经包含了核心功能的半成品项目。我们只需要添加个性功能就可以。

2. 为什么: 避免大量重复的劳动。标准化项目的结构。

3. 如何:

(1). 安装生成脚手架代码的命令行工具: vue/cli

npm i -g @vue/cli

(2). 用脚手架生成工具vue/cli，为本次项目生成一个脚手架代码文件夹结构。

a. 进入想放置项目的文件夹下。

b. 在想放置项目的文件夹下，按shift点右键，选在此处打开命令行或PowerShell。

c. 输入: vue create 项目名

d. 安装过程，看视频: ftp vue/day04/videos/xzvue\_install.wmv

(3). 运行脚手架代码:

a. vs code中确保进入项目文件夹内，看到package.json

b. vs code中，右键选择package.json，点在终端/命令行打开

c. 在终端窗口中 运行: npm run serve

1). 启动了一个简易的开发服务器(作用类似于live server)。

2). 编译(compile)整个项目，为大部分浏览器可以支持的ES5代码和传统的html,css。

d. 按住ctrl，点击终端中出现的连接：http://localhost:8080.

包括两个页面: 首页(home)和关于(about)

|  |
| --- |
| 今后，不用频繁启动npm run serve。只要启动一次，今后，只要源代码一改变，就会自动编译，并会自动刷新页面 |

(4). 项目本地安装axios: vue脚手架代码默认不带axios

在项目文件夹内package.json同目录下:

npm i -save axios

4. 脚手架结构:

(1). public: 保存不需要编译就可直接使用的静态文件:

a. imgs/ 所有网站要用到的图片

b. css/ 一些不需要编译的第三方css bootstrap.min.css

c. js/ 第三方的不需要编译就可直接使用的js文件

比如: jquery-1.11.3.min.js

d. index.html 整个spa项目唯一完整的.html文件

需要手动将第三方的bootstrap.min.css和bootstrap.js和jquery.js都引入到index.html中。

(2). src: 所有自己写的html，css和js都要放在src内。只有src内的代码才会被脚手架编译并打包压缩

a. assets: 通常放整个项目共用的自己编写的css和js。

比如: base.css 和 访问所有接口的函数

b. views: 所有页面组件。将来程序中有几个"页面"，views下就要有几个文件

c. components: 所有页面公用的全局组件或子组件。比如: my-header。

d. router: 保存路由字典和路由器对象。

路由字典中要引入的所有页面组件，必须先通过import引入后，才能放入路由字典中保存。

e. app.vue 和 main.js: 其实就是将原来根组件的HTML，CSS和new Vue()，分开来放：

1). app.vue中放的是只有根组件的html和css

2). main.js中放的是new Vue()

|  |
| --- |
| 每个组件(页面组件，普通的子组件，全局组件)，在脚手架中都是一个.vue文件。.vue文件是集中保存一个组件的HTML，JS和CSS的专门的文件。  浏览器默认不认识.vue文件。只有经过npm run serve编译后，将.vue文件变成传统的html,js和css文件后，浏览器才可加载运行。  每个.vue中，都包含三部分：  1. <template></template>：里边专门放当前组件的HTML片段。以前写在template:``中的HTML片段，现在都要写在<template></template>标签内，且要求不变！只能有一个唯一的父元素。  2. <script></script>里写的是组件对象的js代码：  export default {  data(){ return { ... } },  methods:{ ... },  ... ... }  因为vue脚手架代码采用的是es6的模块化开发:  每个功能/组件都应该是一个模块对象  每个模块对象只有抛出/导出后才能被别的模块发现并引入使用: vue脚手架中: export default { ... }  等效于 ↓  node: module.exports={ 成员 }  一个模块要想使用其他模块的功能，必须先导入/引入  vue脚手架中: import对象名 from "模块文件所在路径"  node: var 对象名=require("模块文件所在路径")  3. <style></style> 专门放当前组件的css代码的区域。  <style scoped></style> 自动防止组件间同名class冲突  原理: 自动给当前组件的父元素加一个随机名称的属性选择器。然后给这个组件下的所有子元素的选择器自动以这个随机属性选择器开头。结果: 这个组件内的所有选择器和其它组件的选择器即使同名，也不会冲突。 |

作业:

1. 看组件间传参的视频

2. todo案例的添加任务