正课:

1. 组件间通信——重点&难点

2. SPA应用

3. Vue-router组件

1. 组件间通信——重点&难点:

3大类:

1. 父->子: 属性下行:

1. 子: 定义props属性，其中包含接收父组件数据的模型变量

props属性：可被外部访问，并赋值的特殊模型数据

1. 在组件内的用法和data中的模型变量完全一样！

2. 可被其他组件发现并赋值。

var 子组件={

... ...

props:[ "模型变量", ... ]

}

2. 父组件的template中:

<子组件 :子组件props属性变量="父组件模型变量"

结果: 父组件将自己的模型变量赋值给子组件内的props属性变量。

2. 子->父: 事件上行

1. 父组件中:

<template>  
 <子组件 @自定义事件名="处理函数"

var 父组件={  
 props/data: 模型变量

methods:{

处理函数(参数){

根据参数修改this.模型变量

}

}

}

强调: @自定义事件名="处理函数"，在定义处理函数时不要加(参数)。因为参数会被认为是绑定模型变量，会报错。

2. 子组件任意位置，都可触发父组件中绑定的自定义事件

this.$emit("自定义事件名",参数值)

会触发父组件template中，提前绑定的@自定义事件，并将参数传递给父组件中的处理函数

3. 兄弟间: 采用先给父，再由父绑定到另一个子

强烈建议，今后，只要多个子组件共用一个数据时，都应将数据定义在父组件上。

2. SPA应用：

SPA: Single Page Application 单页面应用

什么是: 整个应用程序只有一个完整的HTML文件

所谓的"页面"，其实只是不同组件片段而已

在浏览器时，按需加载组件片段内容，替换HTML文件中指定位置。

多页面应用 vs 单页面应用

1. 请求次数:

多页面应用，会造成反复请求

单页面应用，只请求一次，就拿到所有内容

2. 页面跳转的本质:

多页面应用，删除整棵DOM树重建

单页面应用，只替换DOM树中局部节点对象即可

3. 共用资源:

多页面应用，多个页面共用的资源，也需要反复请求下载

单页面应用，只在首次加载HTML文件时，请求下载一次

4. 页面切换动画:

多页面应用，不可能实现页面切换动画

单页面应用，很容易实现页面切换动画

问题: 首屏加载慢！

解决: 将来Vue项目中可设置“懒加载”——按需加载。

单页面应用实现的原理：

1. 定义多个template

2. 定义一个唯一的完整的HTML文件

HTML中要有一个特殊的元素占位，表示将来要被不同template替换的位置。

3. 定义路由字典:

路由: 一个请求地址和一个组件名的对应关系

#index, tplIndex

#products, tplProducts

路由字典：包含一组路由键值对儿的集合

var routes=[

{#index, tplIndex}

{#products, tplProducts}

]

4. 根据当前url中相对路径或请求参数的不同，用不同的template代替占位符的位置。

1. 获得相对路径或请求参数

2. 遍历路由列表，找到对应的template，并读取其中内容

3. 用template内容代替占位符的位置

3. 路由组件: Vue-router

Vue-router: 是vue中专门实现单页面应用的组件

何时: 只要用vue实现单页面应用时，都用vue-router

如何:

Vue核心功能中不包含Vue-router，需要下载后才能使用

1. 下载并引入vue-router.js

<script src="js/vue.js"></script>

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

2. 将页面区域都定义在template中

3. <div id="#app"中添加占位元素<router-view></router-view>

<router-view></router-view>表示将来要被替换的区域

4. 所有template，都要创建对应的组件对象

共用的template，创建全局组件

替换router-view的组件，创建局部子组件

5. 建立路由字典:

var routes=[

{path:"/",component: 默认组件},

{path:"/路径名", component: 组件对象},

... ...,

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

]

仅是保存对应关系的数组，即不会解析地址，也不会替换router-view

6. 创建路由器对象，加载路由字典:

var router=new VueRouter({ routes })

7. 创建vue示例，并载入路由器对象

new Vue({ ... router ...})

如何路由跳转:

1. 在HTML中写死:

<router-link to="/相对路径">文本

原理: router-link用法和a几乎安全一样，因为router-link运行时会被自动翻译为a

2. 在Vue中用程序:

this.$router.push("/相对路径")

如何路由传参:

1. 在接收参数的组件中: props:[ "参数变量" ]

2. 配置路由地址：

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

3. 跳转时: 地址: /相对路径/参数值

强调: 不加:

获得当前路由地址对象: this.$route

this.$router vs this.$route

$router代表路由器，专门执行跳转任务

$route 封装当前路径信息的对象

this.$route.path 当前所在路径

嵌套路由:

如果一个router-view内的部分内容，也需要根据二级路径的不同，动态替换时，就用嵌套路由:

如何:

1. 定义子组件模板，封装子组件对象

2. 在父组件中添加<router-view>

3. 在路由字典中:

{path:"/上级路径",component:父组件对象,children:[

{path:"/上级路径/二级路径",component:子组件对象},

... ...

]}

4. 跳转: /上级路径/二级路径

其中: /上级路径决定根组件中router-view的内容

/二级路径决定父组件中router-view的内容