

Vue第三天

今日内容介绍

- webpack配置
- 项目中使用的es6语法总结
- webpack结合vue-router和vue-resource
- webpack结合移动端开发UI组件
- 源代码托管
- 业务功能的实现

今日内容学习目标

- 知道webpack-dev-server的作用以及配合webpack-html-plugin安装和配置
- 记住es6的几种常用写法
- 知道在webpack项目结构中集成了vue-router模块
- 能够在webpack中集成mint-ui和mui
- 了解vue官方提供的脚手架工具vue-cli
- 知道将项目源码利用git.oschina.net托管
- 知道在webpack项目结构中集成了vue-resource模块
- 能够实现Home.vue组件的轮播图功能

详细内容

1.0 webpack相关配置

1.0.1 利用webpack-dev-server实现热刷新配置

我们在修改了代码以后需要不断的重新执行webpack命令重新打包然后回到浏览器刷新页面去查看，这种开发效率很低下，

所以这里使用webpack-dev-server当代码更新的时候自动重新打包和刷新浏览器。

需要安装的node包有：

webpack-dev-server : webpack开发服务器

html-webpack-plugin : 结合webpack在内存中自动生成index.html的入口文件

在项目根目录下打开cmd命令面板，输入：

```
npm install webpack-dev-server html-webpack-plugin --save-dev 回车即可完成安装
```

- 在package.json文件中配置webpack-dev-server命令

```
"scripts": {
  "dev": "webpack-dev-server --inline --hot --open --port 4009"
}
```

参数说明:

--inline :自动刷新

--hot :热加载

--port 指定监听端口为 5200

-- open : 自动在默认浏览器中打开

-- host: 可以指定服务器的ip, 不指定默认为127.0.0.1(localhost)

- 配置html-webpack-plugin组件

webpack-dev-server要实现浏览器自动刷新, 必须要利用html-webpack-plugin在内存中生成index.html页面才能实现

html-webpack-plugin 配置步骤:

1、在webpack.config.js中加入如下代码:

```
// 导入html-webpack-plugin 包, 获取到插件对象
var htmlwp = require('html-webpack-plugin');
```

```
plugins:[
  new htmlwp({
    title: '首页', //生成的页面标题
    filename: 'index.html', //webpack-dev-server在内存中生成的文件名称, 自动将build注入到这个
    // 页面底部, 才能实现自动刷新功能
    template: 'index1.html' //根据index1.html这个模板来生成(这个文件请程序员自己生成)
  })
]
```

- index1.html 模板页面代码

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <meta name="viewport"
    content="width=device-width,initial-scale=1,minimum-scale=1,maximum-scale=1,user-
scalable=no"/>
</head>
<body>
  <div id="app">
  </div>
</body>
</html>
```

- 运行

在cmd中执行npm run dev 命令开启 webpack-dev-server服务器来运行vue项目
这时候可以随便修改一个css样式, 就会自动刷新看到效果

1.0.2 利用webpack解析和打包 .vue组件页面

Vue项目中的每个页面其实都是一个.vue的文件，这种文件，Vue称之为组件页面，必须借助于 webpack的vue-loader才能使用

所以必须安装相关包：

```
vue-loader          : .vue文件编译loader
vue-template-compiler : .vue模板编译,被vue-loader所依赖
babel-plugin-transform-runtime : es6实时转换成es5语法
```

1、在项目根目录下打开cmd命令面板，输入：

```
npm install vue-loader vue-template-compiler babel-plugin-transform-runtime --save-dev
```

回车即可完成安装

2、在webpack.config.js中添加如下配置（只能在webpack1.0中使用）：

```
babel:{
  presets: ['es2015'],
  plugins: ['transform-runtime'] //这句代码就是为了解决打包.vue文件不报错
}
```

在webpack2.0中在webpack.config.js中添加 babel:{}是不认识的，要改成如下方式：

在项目根目录下新建 .babelrc文件，内容填写如下：

```
{
  presets: ['es2015'],
  plugins: ['transform-runtime'] //这句代码就是为了解决打包.vue文件不报错
}
```

3、在webpack.config.js中的loaders中增加

```
{
  // 打包.vue文件
  test: /\.vue$/, //表示当前要打包的文件的后缀正则表达式
  loader: 'vue-loader' //
}
```

- .vue组件页面的写法结构

1、<template><div class="tmp1"></div>由于是vue2.0 所以这个里面一定要放一个根元素，可以放vue的指令 v-</template>

2、<script> export default{data:{}} -> new Vue({data:{}}) 就是导出一个 Vue的实例 </script>

3、<style></style> 中的样式是全局的
<style scoped></style> 中的样式是当前组件的

- 将.vue中的内容解析编译并且展示在浏览器中

```
1、npm install vue --save
2、在main.js中编写解析.vue的代码
// 1.0 导入vue这个包
import Vue from 'vue';

// 2.0 导入 App.vue文件
import App from './App.vue';

// 3.0 将App中的内容编译解析出来替换index.html中的<div id="app"></div>
new Vue({
  el: '#app',
  // render:function(create){create(App);} es5语法
  render:create=>create(App) //es6语法
});
```

2.0 项目中使用的ECMAScript6语法总结

1、对象的写法

es5中对象: `{add:add,substract:substract}`

es6中对象: `{add,substract}` 注意这种写法的属性名称和值变量是同一个名称才可以简写, 否则要想es5那样的写法, 例如: `{addFun:add}`

2、在对象中的方法的写法

es5中对象: `{add:function(){},substract:function(){}}`

es6中对象: `{add(){},substract(){}}`

3、对象的导出写法

es5两种形式:

1、`module.exports = fucntion (){};`

2、`exprots.add = fucntion (){};`

es6中写法:

1、`export default{
 add(){
}`

2、`export fucntion add(){}` 相当于 将add方法当做一个属性挂在到exports对象

4、对象的导入

es5: `var add = require('./calc.js');`

es6:

如果导出的是: `export default{ add(){}}`

那么可以通过 `import obj from './calc.js'`

如果导出的是:

`export fucntion add(){}`

`export fucntion substract(){}`

`export const PI=3.14`

那么可以通过按需加载 `import {add,substract,PI} from './calc.js'`

5、es6中的箭头函数的写法

箭头的演变过程:

//需求: 利用函数实现倒序排列

`[2,1,3].sort(function(x,y){return y - x;});`

//用箭头函数实现 =>读 goes to

`[2,1,3].sort((x,y)=>{return y - x;});`

`[2,1,3].sort((x,y)=> {x++;y++; y - x;});`

`[2,1,3].forEach(x=> {console.log(x)});`

3.0 webpack项目中集成vue-router步骤 (Vue2.0写法)

- 1、安装vue-router: `npm install vue-router --save`

- 2、在webpack打包入口js文件中(entry指定的那个文件main.js)配置如下

```
main.js x
//1.0 导入vue包
import Vue from 'vue'
import VueRouter from 'vue-router'

//在Vue中使用路由
Vue.use(VueRouter)

//2.0 导入.vue组件
import App from './app.vue'
import Home from './components/Home.vue'
import About from './components/About.vue'

//2.0.1 定义路由规则
var router = new VueRouter({
  routes:[
    {path: '/Home', component: Home},
    {path: '/About', component: About}
  ]
});

//3.0 导入css
import './statics/css/site.less'

//4.0 实例化vue对象并且挂在到id=app的div元素上
let vm = new Vue({
  el: '#app',
  router: router,
  render: create => create(App)
});
```

- 3、在入口App.vue组件中添加如下代码

```
<template>
  <div id="app">
    {{ msg }}
    <br>
    <ul>
      <li>
        <router-link to="/Home">首页</router-link>
      </li>
      <li>
        <router-link to="/About">关于</router-link>
      </li>
    </ul>
    <div>
      <router-view></router-view>
    </div>
  </div>
</template>
```

路由
导航
链接

根据不同的路由规则
加载不同的组件，替换这个占位符
完成 显示

- 4、执行 npm run dev 就可以打开系统里面的两个超链接，点击即可实现页面跳转

4.0 Vue移动组件mint-ui使用

Vue拥有很多的第三方开发的PC端或者移动端UI组件，此项目中主要用到了Vue移动端组件：mint-ui

类似的移动端组件还有：

1、vux

- vux关于升级到vue2.0:https://vux.li/?x-page=github_readme#/zh-CN/upgrade-to-2
- vux 组件使用文档: https://vux.li/?x-page=github_readme#/zh-CN/components
- vux 组件效果演示: <https://vux.li/demos/v2/?x-page=v2-doc-home#/>
- vux github:<https://github.com/airyland/vux>

2、淘宝团队开发的：SUI

- 地址: <http://m.sui.taobao.org/components/#toolbar>

3、muse-ui

- 地址: <https://museui.github.io/#/gridList>

PC端组件：

1、饿了么团队开发的：element

- 地址: <http://element.eleme.io/#/zh-CN/component/installation>

2、iView 是一套基于 Vue.js 的开源 UI 组件库，主要服务于 PC 界面的中后台产品

- 地址: <http://element.eleme.io/#/zh-CN/component/installation>
- iView2 来了，全面支持 Vue.js 2.x:

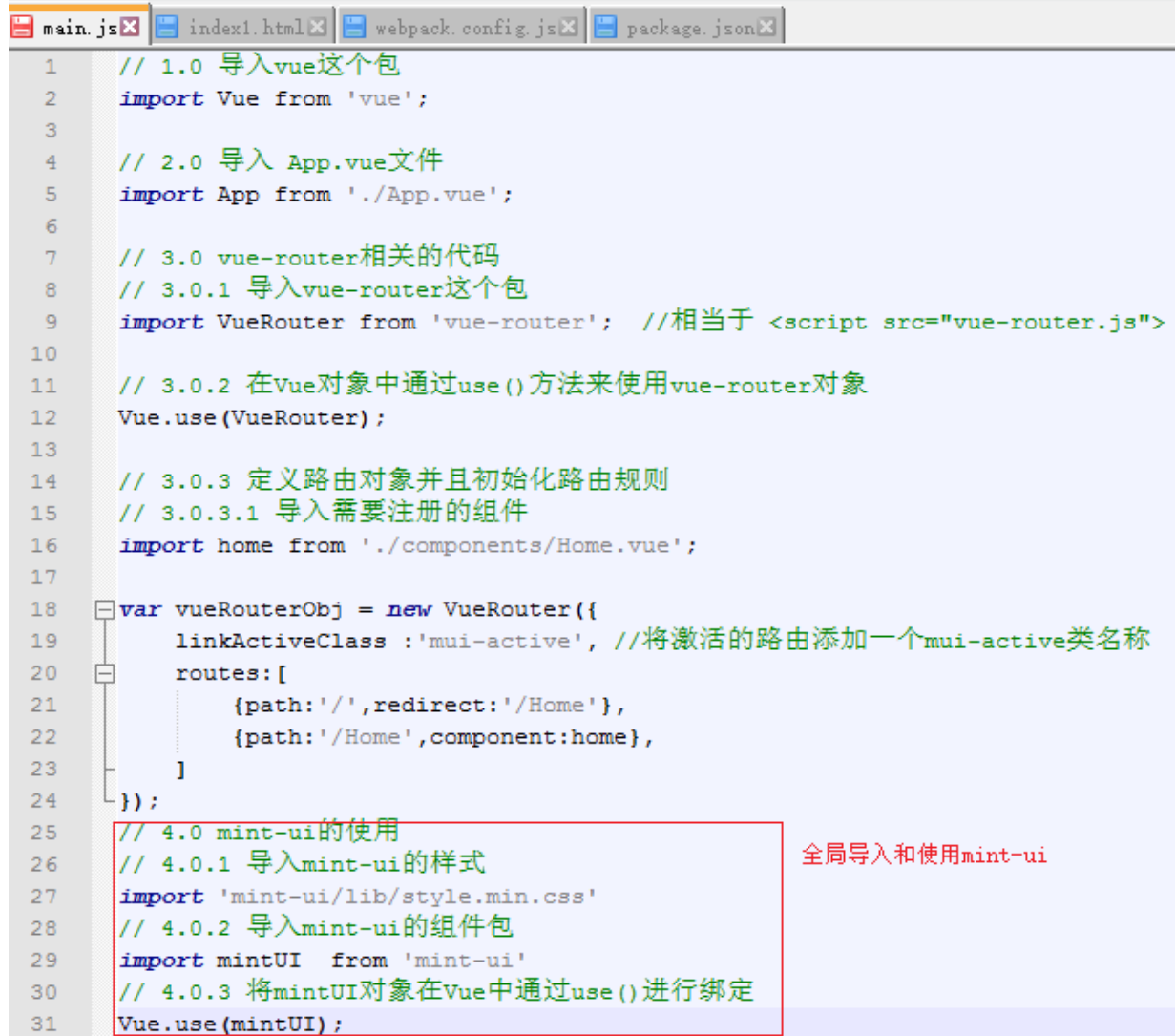
<https://www.talkingcoder.com/article/6395692494071220203>

4.0.1 mint-ui资源介绍

- 官网: <http://mint-ui.github.io/#/zh-cn>
- 在次项目中是与Vue2.0结合使用, 所以请看: <http://mint-ui.github.io/docs/#/zh-cn2>

4.0.2 安装mint-ui集成到项目中

- 1、利用: `npm install mint-ui --save` 命令将mint-ui安装到项目中
- 2、在main.js中全局导入mint-ui和它的css后即可在任何组件的中使用mint-ui组件了



```
1 // 1.0 导入vue这个包
2 import Vue from 'vue';
3
4 // 2.0 导入 App.vue文件
5 import App from './App.vue';
6
7 // 3.0 vue-router相关的代码
8 // 3.0.1 导入vue-router这个包
9 import VueRouter from 'vue-router'; //相当于 <script src="vue-router.js">
10
11 // 3.0.2 在Vue对象中通过use()方法来使用vue-router对象
12 Vue.use(VueRouter);
13
14 // 3.0.3 定义路由对象并且初始化路由规则
15 // 3.0.3.1 导入需要注册的组件
16 import home from './components/Home.vue';
17
18 var vueRouterObj = new VueRouter({
19   linkActiveClass: 'mui-active', //将激活的路由添加一个mui-active类名称
20   routes: [
21     {path: '/', redirect: '/Home'},
22     {path: '/Home', component: home},
23   ]
24 });
25 // 4.0 mint-ui的使用
26 // 4.0.1 导入mint-ui的样式
27 import 'mint-ui/lib/style.min.css'
28 // 4.0.2 导入mint-ui的组件包
29 import mintUI from 'mint-ui'
30 // 4.0.3 将mintUI对象在Vue中通过use()进行绑定
31 Vue.use(mintUI);
```

全局导入和使用mint-ui

- 3、举例使用mint-ui的按钮组件：可以参考文档：<http://mint-ui.github.io/docs/#/zh-cn2/button>

```

1 <template>
2   <div class="tpl">
3     <hr />
4     1.0 mint-ui中的button组件使用<br />
5     <mt-button type="danger" size="large" plain @click="tip">点击</mt-button>
6
7     <mt-button type="primary" size="large" @click="tip1">点击</mt-button>
8   </div>
</template>

<script>
  // 将MessageBox导入
  import { MessageBox, Toast } from 'mint-ui';

  // 利用es6语法定义并且导出一个vm对象
  export default {
    methods: {
      tip1() {
        Toast({
          message: '提示',
          position: 'bottom',
          duration: 1000
        });
      }
    }
  }
</script>

```

使用mint-ui的button按钮组件

像这种需要通过js代码调用的，则需要利用import将组件导入进来

否则在执行 Toast() 或者MessageBox() 的时候会报没有定义的错误

5.0 Vue中使用MUI

MUI是最接近原生APP体验的高性能前端框架，MUI不依赖任何第三方JS库，压缩后的JS和CSS文件仅有100+K和60+K
我们项目中主要使用它的css布局，对于js特效没有用到

- MUI官网：<http://dev.dcloud.net.cn/mui/>
- MUI控件使用文档：<http://dev.dcloud.net.cn/mui/ui/>
- MUI在线Demo：<http://www.dcloud.io/hellomui/>
- hellomui源码下载（完全可以直接拷贝里面的代码来实现自己的功能）：<https://github.com/dcloudio/mui>
- 在Vue+webpack项目中集成MUI步骤

1、从 <https://github.com/dcloudio/mui/tree/master/dist> 中下载所有的资源放到项目的statics\mui目录中

2、在main.js中import mui相关css

![d5-6.png](imgs/d5-6.png "")

3、要实现某个功能只需要按照MUI在线DEMO，找到案例源码页面将效果迁移到项目中

6.0 将项目源码利用git.oschina.net托管

码云 (<http://git.oschina.net/>) 是一个类似于github的中国的源码托管网站, 可以自己创建仓库以后, 使用git进行源码的管理
使用步骤如下:

- 1、去 <https://git.oschina.net/signup> 页面注册一个账号, 如果有则登录
- 2、创建仓库和提交源码



步骤1: 点击新建项目

创建项目

步骤2: 输入项目名称

项目名称: / ! ✓

http://git.oschina.net/ivanyb/Vuetest

项目介绍
非必填

Vue2.0单页程序 输入描述

选择语言 !

添加.gitignore !

添加开源许可证 !

是否公开

公开的 !

公开的话所有人都可以看到

- ☐ 使用Readme文件初始化这个项目
- ☐ 使用Issue模板文件初始化这个项目 !
- ☐ 使用Pull Request模板文件初始化这个项目 !

导入已有项目

创建

步骤3: 点击创建 成功后, 出现如下界面

快速设置— 如果你知道该怎么操作, 直接使用下面的地址

点击右边按钮复制地址

HTTPS SSH

https://git.oschina.net/ivanyb/Vuetest.git !

我们强烈建议所有的git仓库都有一个 README, LICENSE, .gitignore 文件

Git入门? 查看 帮助, Visual Studio / TortoiseGit / Eclipse / Xcode 下如何连接本站, 如何导入项目

简易的命令行入门教程:

Git 全局设置:

步骤4: 安装指令一条条往下执行即可将源码提交到码云上

```
git config --global user.name "ivanyb"
git config --global user.email "ivanyb@qq.com"
```

创建 git 仓库:

```
mkdir Vuetest
cd Vuetest
git init
```

这个步骤完毕以后, 请将源码拷贝到这个文件夹中再往下执行

在执行这个命令之前请先在Vuetest目录中建立 .gitignore

内容:

```
lib-cov
*.seed
*.log
```

```
touch README.md
git add README.md
git commit -m "first commit"
git remote add origin https://git.oschina.net/ivanyb/Vuetest.git
git push -u origin master
```

已有项目?

```
cd existing_git_repo
git remote add origin https://git.oschina.net/ivanyb/Vuetest.git
git push -u origin master
```

```
*.csv
*.dat
*.out
*.pid
*.gz
pids
logs
results
npm-debug.log
node_modules
dist
.idea
```

7.0 Vue 官方命令行工具快速搭建大型单页应用

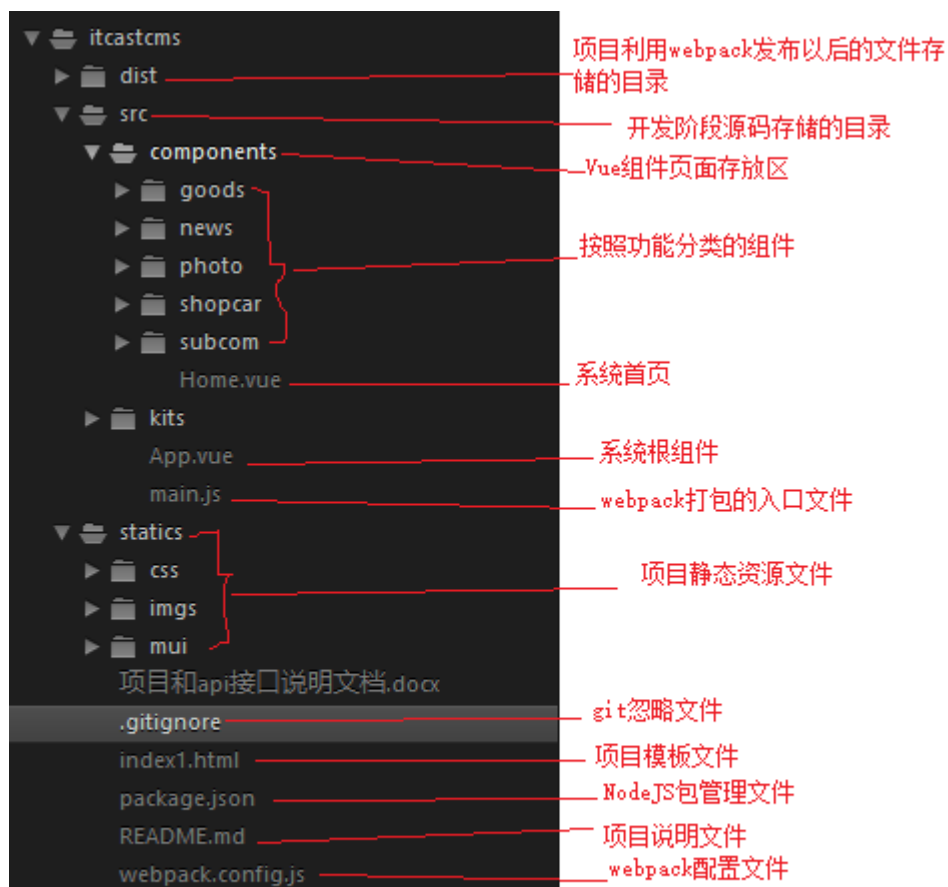
咱们前面从0到1带领大家一步步搭建了属于自己的Webpack+Vue项目，但其实Vue官方提供了一个快速搭建大型单页应用的工具Vue-cli

该工具提供开箱即用的构建工具配置，带来现代化的前端开发流程。只需几分钟即可创建并启动一个带热重载、保存时静态检查以及可用于生产环境的构建配置的项目

- Vue-cli使用步骤

- 1、在cmd命令面板中执行：npm install --global vue-cli 全局安装 vue-cli
- 2、利用：vue init webpack projectName(自定义项目名称) 创建一个基于webpack模板的新项目
- 3、进入到项目名称文件夹中执行 npm install 安装项目所需依赖
- 4、运行 npm run dev 运行项目

8.0 项目结构



还有一个隐藏的node_modules文件夹，它负责存放webpack相关的loader以及vue，vue-router，vue-resource，mint-ui等包文件

9.0 mian.js文件基本内容结构

通过前面慢慢的演变，main.js文件中的基本内容结构如下：

```
// 1.0 导入vue包
import Vue from 'vue';

// 2.0 导入 App.vue文件
import App from './App.vue';

// 3.0 vue-router相关的代码
// 3.0.1 导入vue-router这个包
import VueRouter from 'vue-router'; //相当于 <script src="vue-router.js">

// 3.0.2 在Vue对象中通过use()方法来使用vue-router对象
Vue.use(VueRouter);

// 3.0.3 定义路由对象并且初始化路由规则
// 3.0.3.1 导入需要注册的组件
import home from './components/Home.vue';

var vueRouterObj = new VueRouter({
  linkActiveClass : 'mui-active', //将激活的路由添加一个mui-active类名称
  routes:[
    {path: '/', redirect: '/Home'},
    {path: '/Home', component: home},
  ]
});

// 4.0 mint-ui的使用
// 4.0.1 导入mint-ui的样式
import 'mint-ui/lib/style.min.css'
// 4.0.2 导入mint-ui的组件包
import mintUI from 'mint-ui'
// 4.0.3 将mintUI对象在Vue中通过use()进行绑定
Vue.use(mintUI);

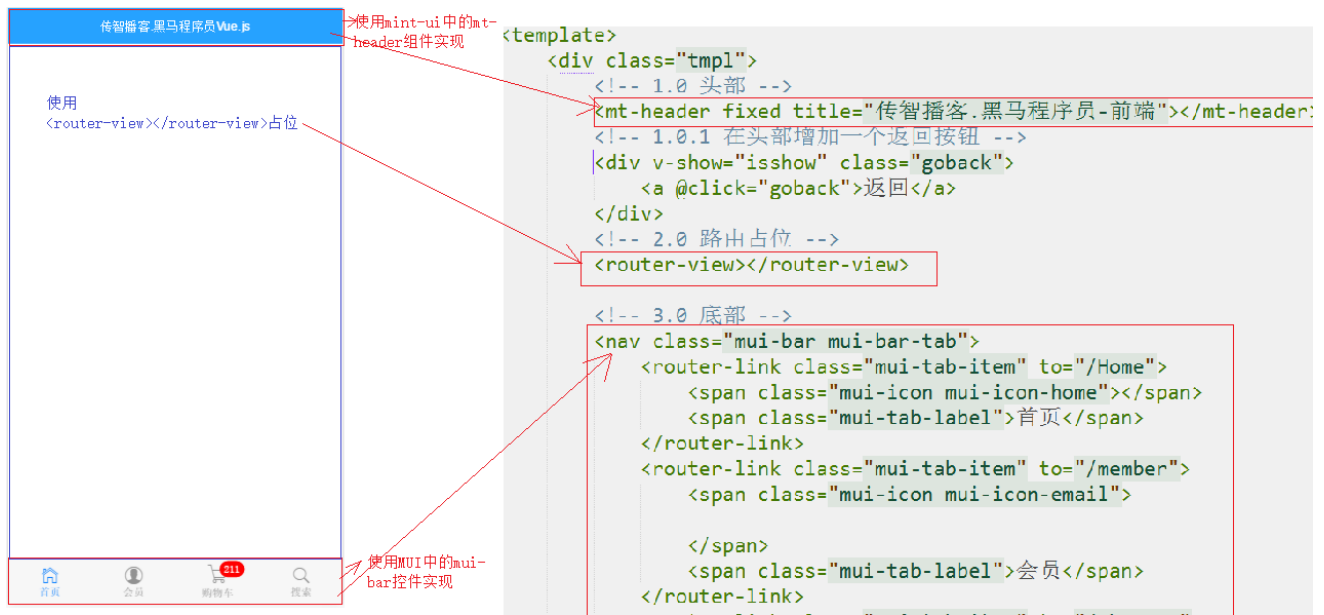
// 5.0 导入mui的css
import '../statics/mui/css/mui.css'
import '../statics/mui/css/icons-extra.css'

// 6.0 使用vue-resource
// 6.0.1 导入vue-resource
import vueResource from 'vue-resource'
// 6.0.2 使用
Vue.use(vueResource);

import '../statics/css/site.css'
// 最后： 将App中的内容编译解析出来替换index.html中的<div id="app"></div>
new Vue({
  el: '#app',
  router: vueRouterObj, //使用路由规则对象
  // render: function(create){create(App);} es5语法
  render: create => create(App) //es6语法
});
```

10.0 创建App.vue根组件

App.vue是整个系统的根组件，其他所有组件将来都是通过vue-router控制将此组件中的<router-view>
</router-view>替换成相应组件的内容



11.0 创建Home.vue根组件和实现



使用mint-ui
中的Swipe组件
实现图片轮播

源码

```
<!-- 1.0 轮播使用的mint-ui中的swipe组件 -->
<!-- <silder :swipeList="swipeList"></silder> -->
<mt-swipe :auto="2000">
  <mt-swipe-item v-for="item in swipeList">
    <a :href="item.url">
      
    </a>
  </mt-swipe-item>
</mt-swipe>
```



Home.vue组件页面

轮播图数据动态获取代码

```
export default{
  data(){
    return {
      swipeList:[]
    }
  },
  components:{silder},
  methods:{
    // 1.0 获取轮播图数据
    getimglist(){
      this.$http.get(common.apiDomain+ '/api/getlunbo')
        .then(res=>{
          // 将数据赋值给swipeList属性，自动导致v-for重新执行
          this.swipeList = res.body.message;
        });
    }
  },
  created(){
    //当进入Home.vue组件的时候请求数据
    this.getimglist();
  }
}
</script>
```