

# Vue第三天

## 今日内容介绍

- 路由服务vue-router在Vue1.0的写法
- 路由服务vue-router在Vue2.0的写法
- watch和计算属性的学习
- 总结这几天学习中知识点在Vue1.0与2.0的区别
- WebPack学习以及相关-loader的使用

## 今日内容学习目标

- 记住vue-router的Vue1.0和2.0版本的基本写法
- 知道使用vue-router实现url传入参数，用\$route.params接收参数
- 知道嵌套路由的使用，vue1.0用subRoutes 2.0使用 children
- 记住\$watch和计算属性computed的使用
- 记住webpack打包css，less,sass的相关-loader包
- 记住webpack es6转es5的写法和注意点
- 记住webpack完成 url()导入资源文件的打包

## 详细内容

### 1.0 路由vue-router

在一个系统中会由很多页面组成，在Vue开发中这些页面通常使用的是Vue中的组件来实现的，那么当在一个页面要跳转到另外一个页面的时候

是通过改变url路径来实现的，那么这个时候Vue需要知道当前url对应的是哪个组件页面，这个控制着就是vue-router

接下来，学习vue-router的相关写法，

注意的是：vue-router 在vue2.0版本中做了很大的改动，所以要注意Vue的版本来选择预期对应的vue-router版本

#### 1.0.1 vue-router资源和介绍

- 配合Vue1.0使用的版本的帮助文档地址：<https://github.com/vuejs/vue-router/tree/1.0/docs/zh-cn>
- 配合Vue1.0使用的vue-router下载地址：<https://cdnjs.cloudflare.com/ajax/libs/vue-router/0.7.10/vue-router.min.js>
- 配合Vue2.0使用的版本的帮助文档地址：<http://router.vuejs.org/zh-cn/installation.html>
- 配合Vue2.0使用的vue-router下载地址：<https://unpkg.com/vue-router/dist/vue-router.js>

#### 1.0.2 vue-router在 vue1.0中的使用

- 1、请下载匹配Vue1.0版本的vue-router文件

- 2、vue-router使用示例代码

```
<script src="../../vue1028.js"></script>
<script src="../../vue-router073.js"></script>
</head>
<body>
  <div id="app">
    <a v-link="{path: '/login'}">登录</a>
    <a v-link="{path: '/register'}">注册</a>

    <!-- 占位符，将来组件内容的替换这个占位符 -->
    <router-view></router-view>
  </div>

  <script>
    // 0.0 定义根组件
    var App = Vue.extend({});

    // 1.0 定义组件
    var register = Vue.extend({
      template: '<h2>注册</h2>'
    });

    // 2.0 定义组件
    var login = Vue.extend({
      template: '<h2>登录</h2>'
    });

    // 3.0 实例化vueRouter的对象
    var vueRouter = new VueRouter();

    // 4.0 定义路由规则
    // map是一个对象
    vueRouter.map({
      'login': {
        component: login
      },
      'register': {
        component: register
      }
    });

    // 5.0 是的vueRouter生效
    vueRouter.start(App, '#app');

    // 6.0 默认跳转到登录组件来显示
    vueRouter.redirect({'/': '/login'});

  </script>

</body>
</html>
```

注意导入js顺序

定义点击a标签以后跳转到哪个路由: /login或者/register  
浏览器地址栏的路径会跟着改变

当点击了注册a标签，则router-view被替换成<h2>注册</h2>

当点击了登录a标签，则router-view被替换成<h2>登录</h2>

### 1.0.3 vue-router在 vue2.0中的使用

- 1、请下载匹配Vue2.0版本的vue-router文件

- 2、vue-router使用示例代码

```
<script src="../../vue2.js"></script>
<script src="../../vue-router2.11.js"></script>
</head>
<body>

  <div id="app">
    <router-link to="/login">登录</router-link>
    <router-link to="/register">注册</router-link>

    <!-- 组件的显示占位区域 -->
    <router-view></router-view>
  </div>

  <script>
    // 定义根组件
    var App = Vue.extend({});

    // 1.0 定义注册组件
    var register = Vue.extend({
      template: '<h2>注册</h2>'
    });

    // 2.0 定义登录组件
    var login = Vue.extend({
      template: '<h2>登录</h2>'
    });

    // 3.0 定义路由对象并且注册路由规则
    var vueRouter = new VueRouter({
      routes: [
        {path: '/', redirect: '/login'},
        {path: '/login', component: login},
        {path: '/register', component: register}
      ]
    });

    // 4.0 使vueRouter生效
    new Vue({
      el: '#app',
      router: vueRouter
      // render: function(cfn) {cfn(App)} 实在webpack中才会使用
    });
  </script>

</body>
</html>
```

注意版本和顺序



#### 1.0.4 vue1.0的路由参数定义实现url的传值

- 1、请下载匹配Vue1.0版本的vue-router文件
- 2、vue-router路由参数示例代码

```

<script src="../vue1028.js"></script>
<script src="../vue-router073.js"></script>
</head>
<body>
  <div id="app">
    <a v-link="{path: '/login'}">登录</a>
    <a v-link="{path: '/register/ivan/123'}">注册</a>

    <!-- 组件的显示位置 -->
    <router-view></router-view>
  </div>
  <script>

```

两个参数的值：ivan和123

```

// 定义根组件
var App = Vue.extend({});

// 1.0 定义注册组件
var register = Vue.extend({
  template: '<h2>注册 ---{{unamevalue}} -> {{upwd}}</h2>',
  data: function() {
    return {
      unamevalue: '',
      upwd: ''
    }
  },
  created: function() {
    // 获取到url传入过来的和 /register/:uname路由规则匹配的参数值
    // /register/ivan
    var uname = this.$route.params.uname;
    this.unamevalue = uname;

    this.upwd = this.$route.params.upwd;
  }
});

// 2.0 定义登录组件
var login = Vue.extend({
  template: '<h2>登录</h2>'
});

// 3.0 实例化vueRouter的对象
var vueRouter = new VueRouter();

// 4.0 定义路由规则
// map是一个对象
vueRouter.map({
  'login': {
    component: login
  },
  'register/:uname/:upwd': {
    component: register
  }
});

```

获取浏览器中传入给uname  
的值和upwd的值  
此处uname的值为：ivan  
upwd的值为：123

在路由规则上定义参数名称，浏览器地  
址栏上传入真正的值

```
// 5.0 是的vueRouter生效
vueRouter.start(App, '#app');

// 6.0 默认跳转到登录组件来显示
vueRouter.redirect({'/':'/login'});

</script>
```

```
'body>
'html>
```

## 1.0.5 vue2.0的路由参数定义实现url的传值

- 1、请下载匹配Vue2.0版本的vue-router文件
- 2、vue-router路由参数示例代码

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script src="../vue2.js"></script>
  <script src="../vue-router2.1.1.js"></script>
</head>
<body>

  <div id="app">
    <router-link to="/login">登录</router-link>
    <router-link to="/register/rose" >注册</router-link>

    <!-- 组件的显示占位区域 -->      url传给uname的值为rose
    <router-view></router-view>
  </div>

  <script>
  // 定义根组件
  var App = Vue.extend({});

  // 1.0 定义注册组件
  var register = Vue.extend({
    template:'<h2>注册-->{{ uname }}</h2>',
    data:function() {
      return {
        uname:''
      }
    },
    created:function() {
      // 获取到url传入过来的和 /register/:uname路由规则匹配的参数值
      // /register/ivan
      var uname = this.$route.params.uname;  获取url传给uname的值为 rose
      this.uname = uname;
    }
  });
```

```

// 2.0 定义登录组件
var login = Vue.extend({
  template: '<h2>登录</h2>'
});

// 3.0 定义路由对象并且注册路由规则
var vueRouter = new VueRouter({
  routes: [
    {path: '/', redirect: '/login'},
    {path: '/login', component: login},
    {path: '/register/:uname', component: register}
  ]
});
// 在路由规则上定义名为 uname的参数

// 4.0 使vueRouter生效
new Vue({
  el: '#app',
  router: vueRouter
  // render: function(cfn) {cfn(App)} 实在webpack中才会使用
});
</script>
</body>
</html>

```

## 1.0.6 vue1中嵌套路由的写法

- 1、请下载匹配Vue1.0版本的vue-router文件
- 2、vue-router嵌套路由示例代码

```

<meta charset="UTF-8">
<title>Document</title>
<script src="../vue1028.js"></script>
<script src="../vue-router073.js"></script>
</head>
<body>
  <div id="app">
    <a v-link="{path: '/account/login'}">登录</a>
    <a v-link="{path: '/account/register'}">注册</a>
    <!-- 组件的显示位置 -->
    <router-view></router-view>
  </div>
  <script>
    // 定义根组件
    var App = Vue.extend({});

    // 定义account组件
    var account = {
      template: '<h1>账号组件</h1> <router-view></router-view>'
    };

```

在浏览器地址栏的url路径中显示的形式

```

// 1.0 定义注册组件
var register = Vue.extend({
  template: '<h2>注册</h2>'
});

// 2.0 定义登录组件
var login = Vue.extend({
  template: '<h2>登录</h2>'
});

// 3.0 实例化vueRouter的对象
var vueRouter = new VueRouter();

// 4.0 定义路由规则
// map是一个对象
vueRouter.map({
  'account': {component: account,
    subRoutes: {
      'login': {component: login},
      'register': { component: register }
    }
  }
});
// 5.0 是的vueRouter生效
vueRouter.start(App, '#app');

// 6.0 默认跳转到登录组件来显示
vueRouter.redirect({'/': '/account/login'});

</script>

```

在account路由下嵌套 login和register路由

```

</body>
</html>

```

## 1.0.7 vue1中嵌套路由的写法

- 1、请下载匹配Vue2.0版本的vue-router文件



- 2、vue-router嵌套路由示例代码

```
<script src="../vue2.js"></script>
<script src="../vue-router2.1.1.js"></script>
</head>
<body>
  <div id="app">
    <router-link to="/account/login">登录</router-link>
    <router-link to="/account/register">注册</router-link>
    <!-- 组件的显示占位区域 -->
    <router-view></router-view>
  </div>
  <script>
    // 定义根组件
    var App = Vue.extend({});

    // 定义account组件
    var account = {
      template: '<div><h1>账号组件</h1> <router-view></router-view></div>'
    };

    // 1.0 定义注册组件
    var register = Vue.extend({
      template: '<h2>注册</h2>'
    });

    // 2.0 定义登录组件
    var login = Vue.extend({
      template: '<div><h2>登录</h2><h3>其他内容</h3></div>'
    });

    // 3.0 定义路由对象并且注册路由规则
    var vueRouter = new VueRouter({
      routes: [{
        // 在vue2.0中一级组件必须有 /
        path: '/account',
        component: account,
        children: [
          //在vue2.0中子组件的path之前不能有 /
          {
            path: 'login',
            component: login
          }, {
            path: 'register',
            component: register
          }
        ]
      }
    ]
    });

    // 4.0 使vueRouter生效
    new Vue({
      el: '#app',
      router: vueRouter
      // render:function(cfn){cfn(App)} 实在webpack中才会使用
    });
  </script>
</body>
```

## 2.0 watch与计算属性computed

watch与computed均可以监控程序员想要监控的对象，当这些对象发生了改变以后，可以触发回调函数做一些逻辑处理

### 2.0.1 watch用法举例

- 监听data中定义的属性

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script src="../../vue1028.js"></script>
</head>
<body>
  <div id="app">
    <input type="text" v-model="firstName">
    <input type="text" v-model="lastName">
    {{ fullName }}
  </div>
</body>
</html>
```

```
new Vue({
  el: '#app',
  data: {
    firstName: 'ivan',
    lastName: '.zhang',
    fullName: 'ivan.zhang'
  },
  watch: {
    'firstName': function(newval, oldval) {
      // console.log(newval, oldval);
      this.fullName = this.firstName + this.lastName;
    }
  }
});
```

监控属性firstName，当其值发生改变的时候，就自动触发回调函数的执行

- 监听路由对象\$route

```

<script src="../../vue2.js"></script>
<script src="../../vue-router211.js"></script>
</head>
<body>
  <div id="app"><router-link to="/login">登录</router-link><router-link to="/register">注册</router-link>
  <!-- 组件的显示占位区域 --><router-view></router-view>
</div><script>
var App = Vue.extend({}); // 定义根组件
// 1.0 定义注册组件
var register = Vue.extend({
  template: '<h2>注册</h2>'
});
// 2.0 定义登录组件
var login = Vue.extend({
  template: '<h2>登录</h2>'
});
// 3.0 定义路由对象并且注册路由规则
var vueRouter = new VueRouter({
  routes: [
    {path: '/', redirect: '/login'},
    {path: '/login', component: login},
    {path: '/register', component: register}
  ]
});
// 4.0 使vueRouter生效
new Vue({
  el: '#app',
  router: vueRouter,
  watch: {
    '$route': function(newval, oldval) {
      console.log(newval, oldval);
    }
  }
});

```

监控\$route对象，当路由发生跳转的时候就会自动触发这个监控中的毁掉函数，可以查看到当前跳转的路由对象参数

## 2.0.2 computed用法举例

- 监听data中定义的属性

```
<script src="../../vue1028.js"></script>
</head>
<body>
  <div id="app">
    <input type="text" v-model="firstName">
    <input type="text" v-model="lastName">
    {{ fullName }}
  </div>
</body>
<script>
  new Vue({
    el: '#app',
    data: {
      firstName: 'ivan',
      lastName: 'zhang'
    },
    computed: {
      // 将fullname变成一个计算属性
      /*
      特点： 1、计算属性会依赖于他使用的data中的属性
      只要是依赖的属性值有改变，则自定重新调用一下计算属性

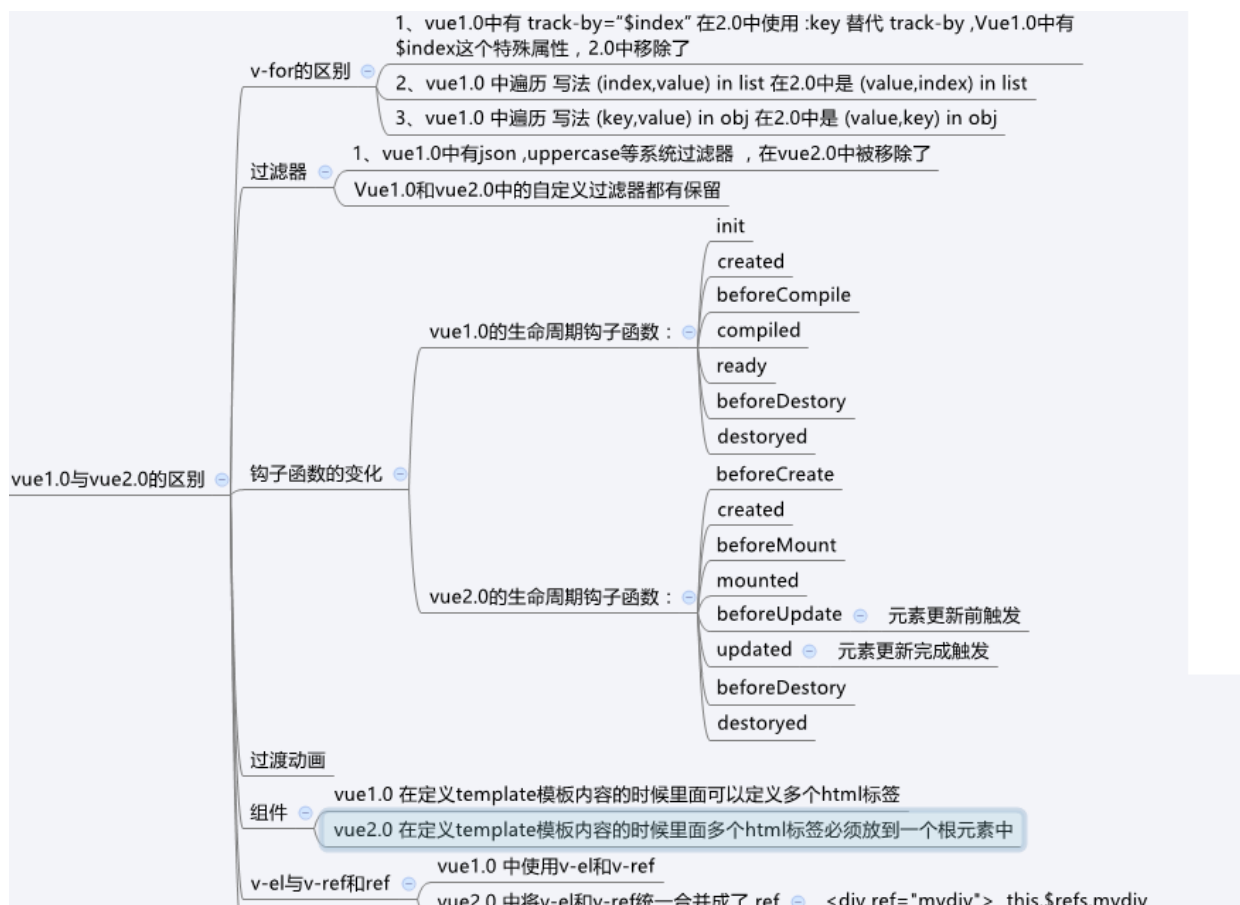
      2、如果它所依赖的这些属性值没有发生改变，那么计算属性的值
      是从缓存中来的，而不是重新编译，那么性能要高一些，所以vue中尽可能使用computed替代 watch
      */
      fullName: function() {
        return this.firstName + this.lastName;
      }
    }
  });
</script>
```

fullName的值会随着 firstName或者lastName的改变而改变

## 3.0 Vue1.0与Vue2.0区别总结

总结前3天中学习知识点中Vue1.0和2.0的区别

- 总结如下



## 4.0 webpack

### 4.0.1 webpack介绍

- webpack是一个资源的打包工具，分为1.0和2.0版本，可以将 .js, .css , image等静态资源当做一个模块来进行打包，那么每一种模块都是有一个对应的 loader来实现
- webpack 1.0版本官网：<https://webpack.github.io/docs/usage.html>
- webpack 2.0版本官网：<https://webpack.js.org/>
- 在这个项目中使用webpack 1.14.0
- node环境的安装

webpack是基于nodejs运行的，所以在安装webpack之前必须先安装nodejs环境,安装步骤如下

- 去 <https://nodejs.org/en/> 中下载当前操作系统匹配的版本,windows下软件名称通常叫做 node.exe
- 双击node.exe一路安装好，由于node.exe已经包含了npm工具，所以npm也能正常使用了
- 由于直接使用npm install 安装第三方包是去国外网站上下载，有可能会被墙而安装失败，所以我们要将下载源切换到国内淘宝上因此需要利用 npm install nrm -g安装一个全局的nrm
- 安装好nrm以后，在cmd命令面板中输入： nrm use taobao 将下载源切换到淘宝，可以使用 nrm ls 查看当前使用的下载源
- 也可安装淘宝提供的类似于npm的工具 cnpm来替代npm安装node包,安装包命令和npm一样，安装cnpm命令：  
npm install cnpm -g

- webpack的安装

安装webpack步骤:

第一种安装方式:

在cmd命令行面板中 执行: `npm install webpack@1.14.0 -g` 将webpack1.14.0版本安装为全局就能够在cmd命令行面板中使用webpack指令了

第二种安装方式:

在cmd命令行面板中 执行: `cnpm install webpack@1.14.0 -g` 将webpack1.14.0版本安装为全局就能够在cmd命令行面板中使用webpack指令了

## 4.0.1 webpack常用指令和webpack.config.js配置文件

- webpack常用指令

```
webpack 入口文件.js 输出文件.js
webpack          // 最基本的启动webpack的方法, 默认查找名称为 webpack.config.js文件
webpack --config webpack.config.js    // 指定配置文件

webpack -p       // 对打包后的文件进行压缩
webpack -d       // 提供source map, 方便调式代码
```

- webpack.config.js配置文件的作用

如果只在cmd命令面板中输入 `webpack`指令, 后面不跟任何参数的话, 则默认查找的是 `webpack.config.js`文件, 在这个文件中可以配置入口文件, 输出文件以及相关loader和插件等, 以增强webpack的功能

- 一个常用webpack1.0版本的webpack.config.js文件结构:

```
// 导入html-webpack-plugin 包，用来根据模板自动生成index.html
var htmlwp = require('html-webpack-plugin');

module.exports={
  entry: './src/main.js', // 1.0 定义打包的入口文件路径
  output:{
    path: './dist', //打包以后的文件存放目录
    filename: 'build.js' // 打包以后生成的文件名称
  },
  module:{
    loaders:[
      {
        // 将当前项目中所有的.js文件都要进行es6转es5操作，node_modules除外
        test: /\.js$/, //表示当前要打包的文件的后缀正则表达式
        // loader: 'babel-loader?presets[]=es2015', //如果写到这里，将来在打包.vue文件的时
        // 候会报错，表示先利用css-loader解析.css文件，再调用style-loader打包
        loader: 'babel-loader',
        exclude: /node_modules/ //node_modules中的所有.js文件不去转换，提高打包性能
      }
    ]
  },
  babel:{
    presets: ['es2015'], //表示es6转es5
    plugins: ['transform-runtime'] //这句代码就是为了解决打包.vue文件不报错
  },
  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(这个文件请你自己生成)
    })
  ]
}
```

## 4.0.2 webpack中loader介绍

- loader介绍

webpack本身不支持css,less,sass,js,image等相关资源的打包工作的，它仅提供了一个基础的框架，在这个框架上借助于相关的loader才可以实现css,less,sass,js,image等相关资源的打包工作

## 4.0.3 webpack相关配置

在使用loader之前需要在当前项目目录下打开cmd命令面板，输入：`npm init` 初始化一个 `package.json`文件来存放相关的 loader包

### 4.0.3.1 打包css资源演示

webpack中使用css-loader和style-loader这两个loader来处理css资源的打包工作，所以使用前必须在项目中先安装这两个包：

```
npm i css-loader style-loader --save-dev
```

- 在webpack.config.js中配置这两个loader

```
webpack.config.js
module.exports = {
  entry: './main.js',
  output: {
    path: './dist',
    filename: 'build.js'
  },
  module: {
    //rules: [ //也可以使用rules代替loaders: [
    多写 loaders: [
    法 {
      test: /\.css$/, //所有的.css结尾的文件被此loader处理
      //如果写成 style!css或者 css-loader!style-loader都会报错
      loader: 'style-loader!css-loader' //从右边向左边调用
      /* 也可以这样写:
      loader: ['style-loader', 'css-loader']
      loaders: ['style-loader', 'css-loader']
      */
    }
  ]
}
}
```

- 在项目中建立一个site.css文件，并且在main.js中导入

```
site.css
#res{
  border:solid 1px red;
}

main.js
//导包calc.js
var calc = require('./calc.js');
require('./statics/css/site.css');
```

- 在cmd中执行webpack命令



```

PS D:\web\打包loader配置> webpack
Hash: 633c9173f02e9bac9b64
Version: webpack 2.1.0-beta.27
Time: 1000ms
   Asset      Size  Chunks             Chunk Names
build.js    13 kB          0 [emitted]  main
   [0]    ./calc.js  48 bytes {0} [built]
   [5]    ./main.js  508 bytes {0} [built]
   + 4 hidden modules

```

### 4.0.3.2 打包sass资源演示

webpack中使用sass-loader, css-loader, style-loader来处理.scss文件的打包工作,而sass-loader需要依赖于node-sass所以使用前必须在项目中先安装这些包,并且node-sass的某些文件下载是需要去google上的,为了防止被墙而导致安装失败,所以建议使用cnpm来安装:

```
cnpm install node-sass sass-loader css-loader style-loader --save-dev
```

- 在webpack.config.js中配置这两个loader

```

{
  // 打包 sass文件
  test:/\.scss$/, //表示当前要打包的文件的后缀正则表达式
  loader:'style-loader!css-loader!sass-loader' //实现sass文件的打包
},

```

- 在项目中建立一个site1.scss文件,并且在main.js中导入

```

site1.scss
$color:blue;

#v1{
  border:1px solid $color;
}

```

项目入口文件 main.js

```
require('../statics/css/site1.scss');
```

- 在cmd中执行webpack命令

在项目根目录下打开cmd命令面板,输入: webpack 回车即可打包完成  
此时检查build.js文件的内容, sass语法是变成了css语法表示打包成功

### 4.0.3.3 打包less资源演示

需要安装的node包有:

css-loader: 编译css  
style-loader: 编译css  
less-loader: 编译less  
less: less-loader的依赖包

在项目根目录下打开cmd命令面板, 输入:

npm install less less-loader style-loader css-loader --save-dev 回车即可完成安装

- 在webpack.config.js中配置这两个loader

```
{  
  // 打包less文件  
  test: /\.less$/, //表示当前要打包的文件的后缀正则表达式  
  loader: 'style-loader!css-loader!less-loader' //实现sass文件的打包  
},
```

- 在项目中建立一个site1.scss文件, 并且在main.js中导入



The screenshot shows two code editors. The top editor displays `site2.less` with the following content:

```
1  
2 @color:yellow;  
3  
4 #v2{  
5   border:1px solid @color;  
6 }
```

A red arrow points to the `@color:yellow;` line with the annotation "less语法".

The bottom editor displays `main.js` with the following content:

```
1 // webpack的入口文件  
2 // 在此处main.js实现的是获取到dom对象, 实现一个加法  
3  
4 var add = require('./calc.js');  
5  
6 // 导入site.css 以后webpack才能将site.css打包  
7 require('../statics/css/site.css');  
8  
9 require('../statics/css/site1.scss');  
10  
11 require('../statics/css/site2.less');  
12
```

A red arrow points to the `require('../statics/css/site2.less');` line with the annotation "在webpack打包入口文件 main.js使用less语法的文件".

- 在cmd中执行webpack命令

在项目根目录下打开cmd命令面板, 输入: webpack 回车即可打包完成  
此时检查build.js文件的内容, less语法是变成了css语法表示打包成功

### 4.0.3.4 打包url()请求的资源

需要安装的node包有:

url-loader: 打包通过url()方式的请求资源

file-loader: url-loader的依赖loader

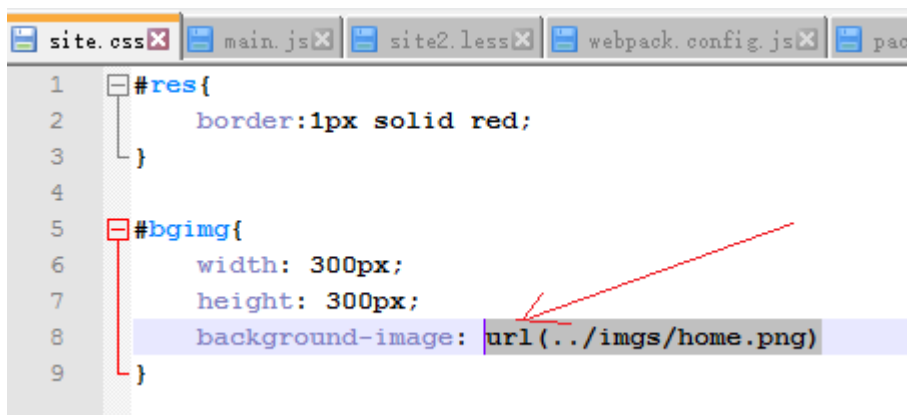
在项目根目录下打开cmd命令面板, 输入:

npm install url-loader file-loader --save-dev 回车即可完成安装

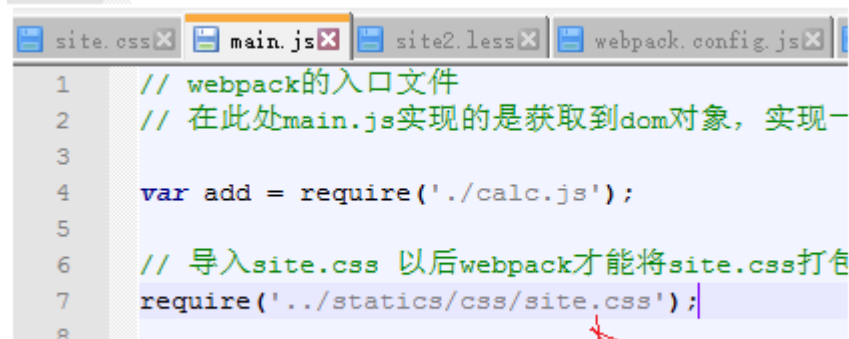
- 在webpack.config.js中配置这两个loader

```
{
  // 打包url()请求的资源文件 这个正则中的扩展名可以根据需求进行追加
  test: /\. (png|jpg|gif|tiff)$/ , // 注意url可能请求多个资源, 所以将来在项目中通过url导入的资源扩展名必须配置在这里
  // loader: 'url-loader' // 这种配置会将图片变成一个base64的字符串存储到build.js中,
  // 如果图片很大的话, 那么会造成build.js也比较大, 加载的时候会导致效率低下
  loader: 'url-loader?limit=40000' // 限制图片的大小如果<40k则把他当做base64字符串存储到build.js中
  // , 如果>40k 则单独将图片存放到磁盘上, 将路径打包进去bulid.js
},
```

- 在site.css文件导入一个图片



```
1 #res{
2   border:1px solid red;
3 }
4
5 #bgimg{
6   width: 300px;
7   height: 300px;
8   background-image: url(..../imgs/home.png);
9 }
```



```
1 // webpack的入口文件
2 // 在此处main.js实现的是获取到dom对象, 实现一
3
4 var add = require('./calc.js');
5
6 // 导入site.css 以后webpack才能将site.css打包
7 require('../statics/css/site.css');
8
```

- 在cmd中执行webpack命令

在项目根目录下打开cmd命令面板, 输入: webpack 回车即可打包完成

检查是否成功分两种情况:

- 1、如果打包的图片大小大于配置文件中 url-loader?limit= 中的limit值的话, 则会在目录下看到一张单独的一个图片
- 2、如果打包的图片大小小于等于配置文件中 url-loader?limit= 中的limit值的话, 则会将图片以base64格式存储在build.js中

请按照上述两种情况去验证是否打包成功

#### 4.0.3.5 ECMAScript6语法转ECMAScript5语法

需要安装的node包有:

babel-core

babel-loader

babel-preset-es2015

babel-plugin-transform-runtime: 这个包主要是在打包.vue组件页面中的es6语法需要

在项目根目录下打开cmd命令面板, 输入:

```
npm install babel-core babel-loader babel-preset-es2015 babel-plugin-transform-runtime --save-dev 回车即可完成安装
```

- 在webpack.config.js中配置这两个loader

```
loaders:[
  {
    // 将当前项目中所有的.js文件都要进行es6转es5操作, node_modules除外
    test:/\.js$/, //表示当前要打包的文件的后缀正则表达式
    // loader:'babel-loader?presets[]=es2015', //如果写到这里, 将来在打包.vue文件的时候会报错
    //, 表示先利用css-loader解析.css文件, 再调用style-loader打包
    loader:'babel-loader',
    exclude:/node_modules/ //node_modules中的所有.js文件不去转换, 提高打包性能
  },
],
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main.js中使用es6语法导入site.css

```
import '../statics/css/site.css'
```

- 在cmd中执行webpack命令

在项目根目录下打开cmd命令面板, 输入: webpack 回车即可打包完成

检查build.js文件中, 如果出现了类似于 require('../statics/css/site.css'); 但是看不到import  
'../statics/css/site.css' 表示转换成功