Webpack的使用一

**一、什么是Webpack**

Webpack 是一个前端资源加载/打包工具。它将根据模块的依赖关系进行静态分析，然后将这些模块按照指定的规则生成对应的静态资源。

**二、NodeJs**

安装Webpack需要使用到npm命令，而使用这个命令需要我们优选安装NodeJs模块。

**三、Webpack的安装**

使用npm install [webpack@3.6.0](mailto:webpack@3.6.0) -g 命令在全局安装webpack功能。

**四、体验webpack的打包流程。**

Webpack可以将css、js、图片等打包为一个文件以便于调用。

打包命令：webpack 被打包的js文件路径 打包完成后的存放路径

例：webpack ./src/main.js ./dist/bundle.js

在一个webpack项目中包含有两个文件夹，分别是“dist”(distribution发布)和“src”。src中一般存放开发中的项目文件，而dist是用来存放开发完成打包完成之后的文件。

最终打包之后的js文件叫”bundle.js”，而我们也是一般引用这个文件作为我们的导入的JS库文件。

**五、webpack的配置文件。**

在项目中，我们有一个叫做webpack.config.js的文件，这个文件是webpack的配置文件，在这个文件中我们可以对webpack进行一系列的配置。



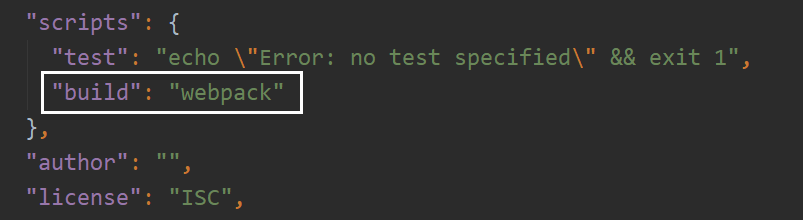
Webpack配置文件内容举例

如果我们要使用到nodeJS的组件的使用，需要使用：**npm init**进行初始化操作，在这个步骤执行完成之后，在文件夹中会生成一个叫做” package.json”的文件，当package.json文件中某些设置的功能需要外部的支持的时候需要使用**npm install**命令进行安装。

在完成上述操作，指定webpack默认的打包方式之后，就可以直接使用**webpack**命令进行打包了。

此时我们使用的webpack命令是全局的webpack命令，但是往往我们的项目的开发环境和全局的情况并不一致，所以我们还需要使用: npm install webpack@3.6.0 --save-dev 命令在当前项目中安装局部的webpack环境。

而安装完成之后，需要让项目使用当前项目的webpack环境进行文件的打包，还需要在” package.json”文件中添加一个设置项，然后使用:npm run build 完成打包。



Webpack环境配置

**六、webpack的loader拓展器**

Loader是webpack中非常核心的一个功能，可以完成文件、版本之间的互相转换。例如：es6转es5，less；scss转css；vue、jsx转js等。

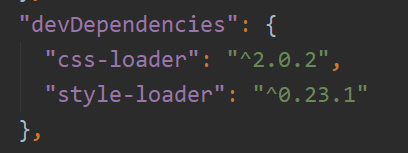
**1、下面我们以打包css为例：**

为了能让我们的webpack可以打包、解析css文件我们需要两个loader：

css-loader(打包loader): **npm install css-loader@2.0.2 --save-dev**

style-loader(解析loader): **npm install style-loader@0.23.1 --save-dev**

在安装完成之后，查看package.json文件可以查到了这两个依赖已经安装完成了.



当然，做完这一步之后并不意味着这两个loader就已经生效了，我们还需要在webpack的配置文件中添加一个设置，保证这两个loader可以被读取到。



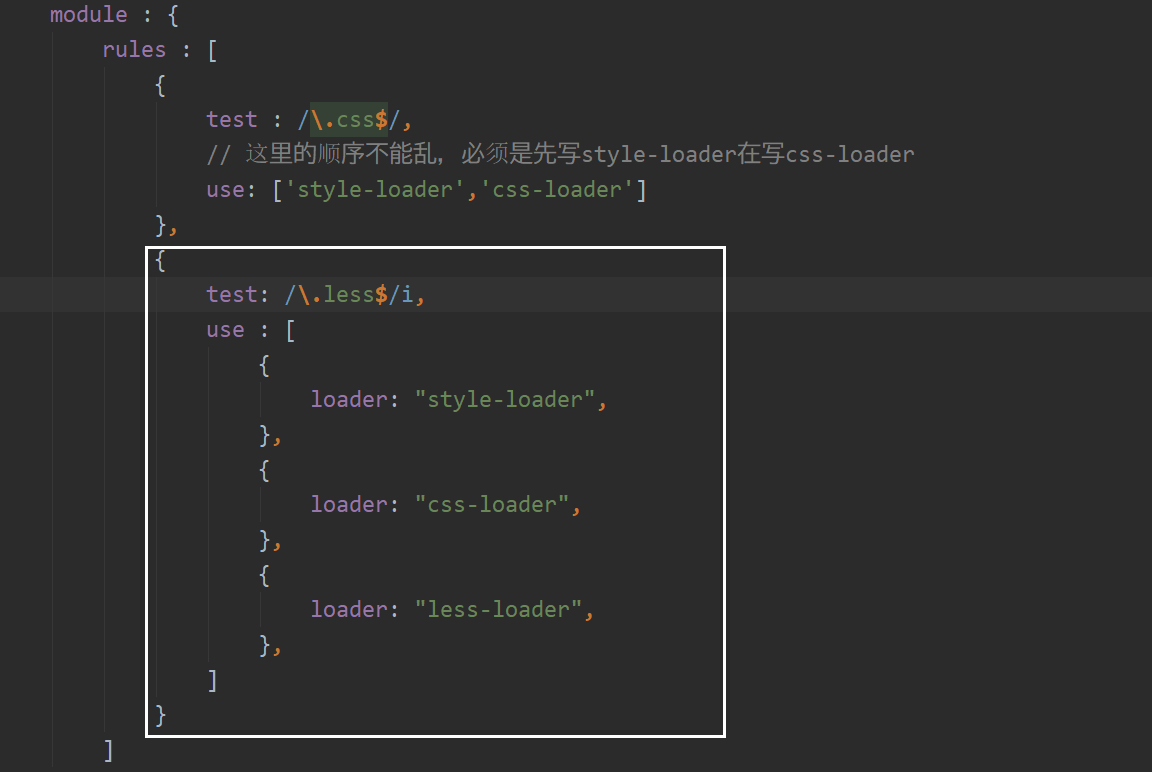
完成这一步之后，才可以算是这两个loader已经成功加载到了webpack上了。接下来就是在我们入口js文件中包含css文件。包含的方法很简单，如下图。



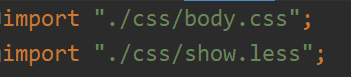
最后我们再次使用”npm run build”重新打包js文件后就可以在网页中看到css设置的效果了。

**2、追加处理less文件的loader：**

除了可以处理css文件之外，webpack同样可以处理less文件，其操作的流程和上方类似：首先应该使用**npm install --save-dev less-loader@4.1.0** [**less@3.9.0**](mailto:less@3.9.0) 命令安装依赖。在完成之后向webpack的配置文件中添加一个设置。



最后，在入口js文件中添加导入less文件的代码后。



再次使用”npm run build”重新打包js文件后就可以在网页中看到css设置的效果了。

**3、追加处理图片文件的loader**

如果在css中添加了背景图片，这时就需要对文件的打包添加文件打包的支持了。文件打包有两种loader。

第一种是url-loader：**npm install --save-dev** [**url-loader@1.1.2**](mailto:url-loader@1.1.2)。这类loader一般适用于比较小的图片，在处理这类文件的时候，打包的形式是将图片转换为base64格式的字符串进行保存。。

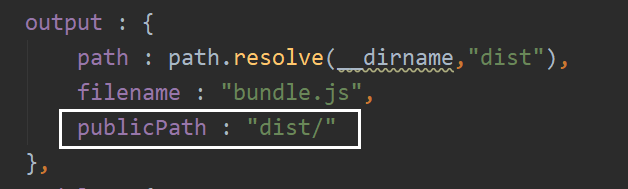
在安装完成之后，不要忘记在webpack文件名中写好运行规则。在规则中有一个属性是limit单位为B，示意了当前所允许的最大文件的尺寸。



同样的，在安装完url-loader之后，需要重新对整个项目进行打包后方可在网页中看到最终效果。

当图片的尺寸比较大的时候，url-loader就无法满足要求了，这时我们就需要使用file-loader模块进行加载。安装指令：**npm install --save-dev** [**file-loader@3.0.1**](mailto:file-loader@3.0.1) 。安装完成之后，不需要在配置文件中添加使用规则。因为超出文件大小后，会自动使用file-loader对图片进行打包。

操作完成之后，再次使用”npm run build”重新打包js文件后，就可以在dist文件中看到被打包好的图片了。但是这时你会发现，在网页中无法看到设置好的背景图片。这是因为背景图片的路径设置错误了，为了解决这个错误，让系统到dist文件夹中去寻找背景图片，需要修改webpack的配置文件。



修改完成后，重新打包js文件，就可以在网页中看到图片的效果了。

如果需要对文件进行重新命名也可以在配置中添加选项，指定文件保存的路径的具体格式。

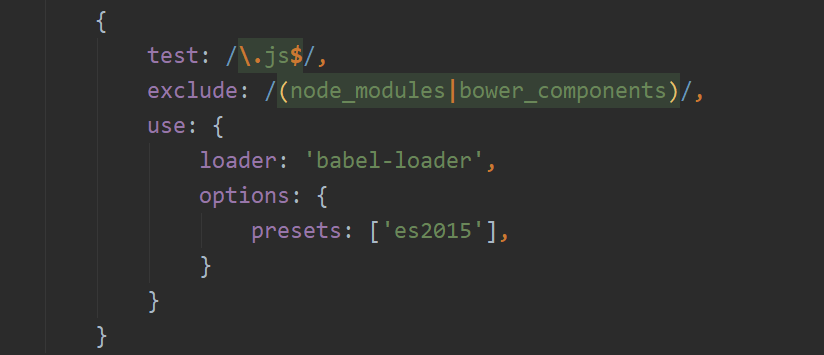


**4、ES6转ES5的方法**

为使得我们写的ES6可以和更多的浏览器兼容，在打包的时候，我们需要webpack可以帮助我们把ES6转换为ES5语法。

安装命令：npm install --save-dev babel-loader@7.1.5 babel-core@6.26.3 [babel-preset-es2015@6.24.1](mailto:babel-preset-es2015@6.24.1)

Webpack文件配置：



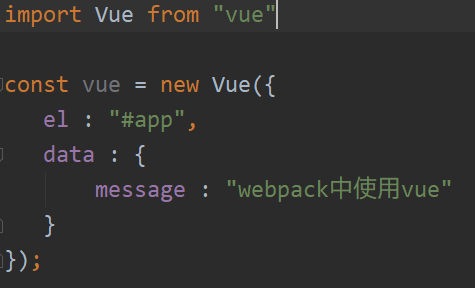
配置完成之后即可把JS文件中ES6转换成ES5了。

**5、Webpack中使用vue**

为了在webpack使用vue，应该先安装vue插件

命令：**npm install vue@2.5.21 --save**

在安装完成之后，我们就可以在webpack项目的入口JS文件中写Vue的代码了。



写完代码重新打包后，你会发现在网页中无法查到message中的内容，查看控制台发现可以查看到错误与runtime-only有关。而错误原因如下：

runtime-only中不可以包含任何的template。

runtime-compiler中可以有template，因为compiler可以编译template。

而我们现在使用的版本正是runtime-only，所以就无法通过编译，为了使得我们的vue可以正常工作，我们需要修改webpack的配置文件，让webpack以runtime-compiler的模式进行工作，修改的内容如下：



此时再次重新打包js文件，运行网页，就可以发现Vue已经正常跑起来了。

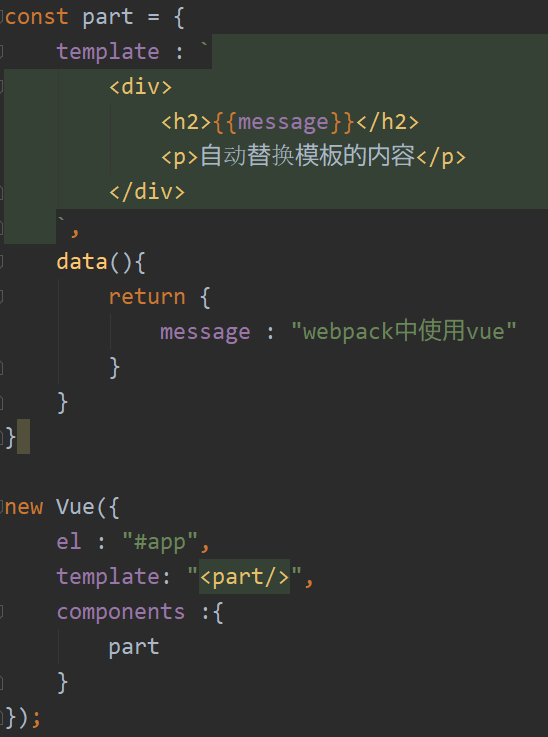
**6、el和template的关系**

在实际运用中，我们并不希望频繁的修改html文件中的内容，这个时候，我们就可以把Vue组件中的template中的内容设置为我们想要运用的代码。



这么设置完成后，在html文件中id为app的div的内容就会被template中的内容自动化替换。这样设置的好处就在于，我们不需要频繁的修改html的内容了，只需要变动template中的内容就可以完成对网页的修改了。

接下来，我们就可使用到组件对Vue进行开发了。我们稍稍对上述代码进行一些修改。下方的代码vue组件中的一部分抽离出来后，单独组成一个对象。然后在vue组件中注册它，并且在template中使用它。这样好处就在于，这样使vue变成了一个加载器，需要显示什么内容，我们就就插入什么组件，这样一来我们就可以便捷快速的完成对一个网页的渲染。



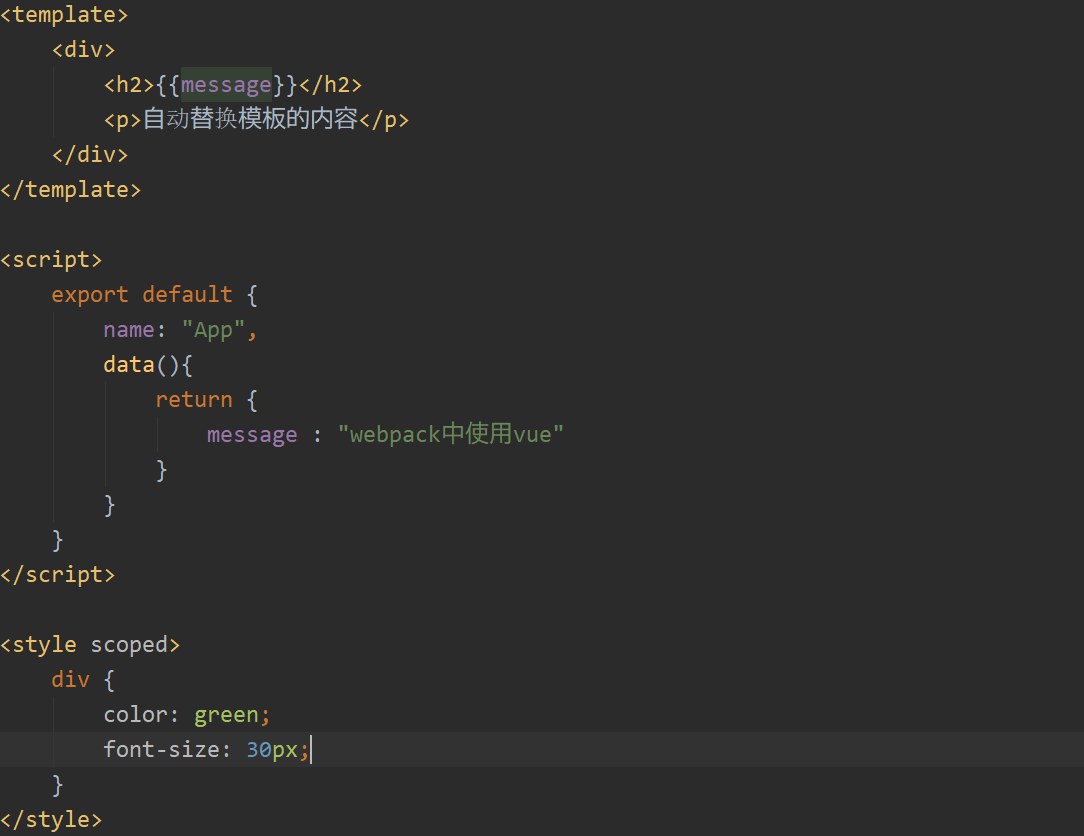
通过重新打包并且重新运行网页可以发现，这样操作的结果和之前操作的所得到的页面是一样的。

接下来，我们可以更进一步对上面的代码进行抽离。将上面的part对象单独写入到一个js文件中进行保存。最后再次引用抽离的内容，这样就完成了组件的分离操作。



**7、使用vue文件**

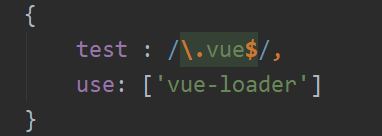
在上述流程体验完成之后，我们可以正式的来体验一下vue文件的的使用了。Vue文件才是我们使用vue抽离插件的指定文件，通过上方的js文件，我们可以创建一个这样的vue文件。



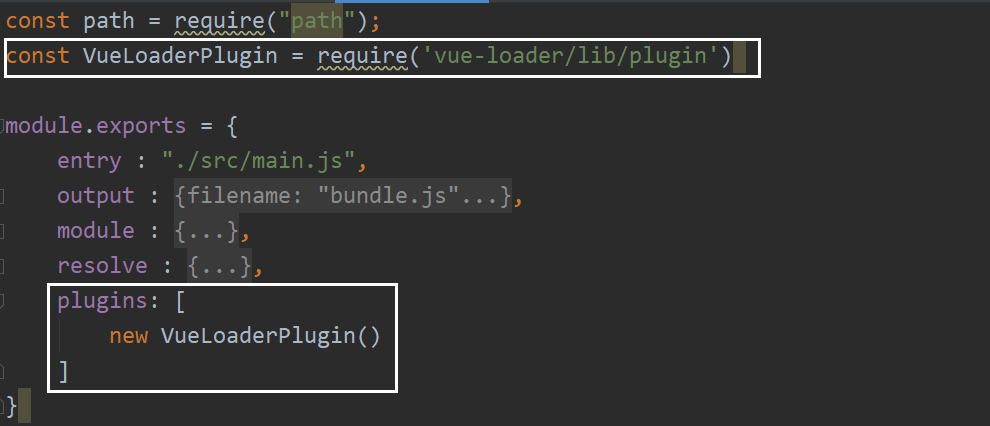
可以看到，我们在这个vue文件中把整个vue组件分离成了若干部分。分离完成之后，我们就可以在js中导入vue文件并且使用了。但是在此之前，我们还需要安装两个插件。

安装命令: **npm install vue-loader@15.4.2 vue-template-compiler@2.5.21 --save-dev**

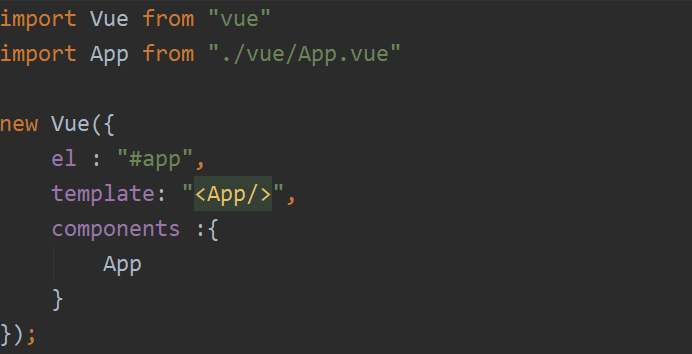
最后在修改wenpack的配置文件，增加使用规则如下。



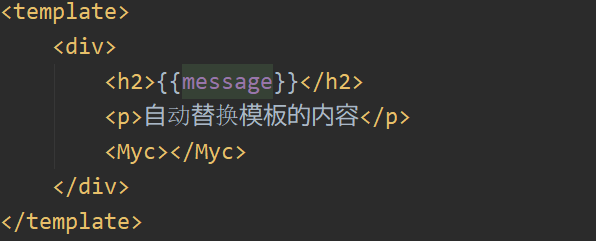
此时，我们的插件vue插件还不能正常使用。因为我们的vue插件的版本是大于15的，所以我们还需要在webpack的配置文件中增加一个插件配置。



经过上述的操作之后，就可以顺利通过打包的程序，正确的读取编译vue文件了。

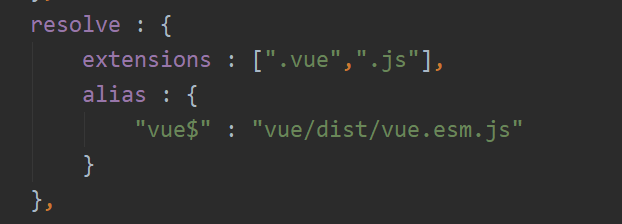


同时基于上述的内容，我们还可以在Vue文件中插入其他的Vue组件，以实现Vue组件的嵌套开发。





如上图所示，我们可以在一个组件中使用import导入另外一个组件。然后在template模板中直接使用，就可以完成组件的套用了。



使用小技巧：在导入vue文件的时候.vue的后缀是不可以省略的。如果向省略后缀名需要在webpack添加配置，让他可以识别后缀名。