学习的是Webpack2.0版本

学习网址：

（1）<http://zhaoda.net/webpack-handbook/what-is-webpack.html>

（2）<https://doc.webpack-china.org/>

学习webpack之前先了解模块化定义：模块化指各个JS、组件、模块之间的依赖（一个模块就是一个js文件、一个模块就是一个文件），浏览器只认识js文件，此时我们需要进行编译（sass、less、ES6、JSX编译，都需要一个系统化的工具）。再者我们会把一个个的小功能或一个个小的函数，提供公共函数或提供公共功能的函数抽离出来单独形成一个文件，这个文件就称为一个模块。这个模块通过module.exports对外暴露一个接口。

如何定义模块：在ES6之前javascript不支持模块化系统，javascript没有export和import。我们现在所熟知的模块化更多的是javascript，我们希望图片是一个模块、css是一个模块，想用时用require方法调用。

Webpack阐述

**介绍：**webpack是模块化的打包机，自动分析项目结构、自动处理模块化之间的依赖

**工作方式：**把项目当作整体，通过给定的主文件（入口文件）中会require其它的模块，webpack将从这个主文件开始找到项目的所有的依赖文件，并使用loader进行全部加载、处理，最终打包生成浏览器识别的js文件

Webpack的使用

1. 全局安装，安装版本2

命令行输入：npm install -g webpack@2

webpack –v查看版本信息

1. 创建一个项目，并项目初始化

npm init

1. 项目本地安装webpack

npm install –save-dev webpack

-dev表示开发项目所依赖的库、文件

1. 创建主文件main.js、index.html文件
2. 创建2个模块module1.js和module2.js文件，并在这两个模块中使用module.exports暴露接口
3. 使用webpack处理main.js、module1.js、module2.js之间的依赖关系，然后打包成一个文件在index.html中进行引入

module1.js文件代码：

var changeColor=function(dom,color){

dom,style.color=color;

}

Module.exports=changeColor;//模块必须对外暴露接口

module2.js文件代码：

var color=”red”;

module.exports=color;//模块必须对外暴露接口

main.js文件代码（组文件，处理上面两个文件之间的依赖）：

var color=require(“./module2”);

var changeColor=require(“./module1”);

changeColor=(document.querySelector(“#app”),color);

1. 在根目录中创建webpack.config.js文件，并进行配置（告诉程序如何处理这些包的依赖），如下：

module.exports={

entry:\_\_dirname+”./main.js”,//告诉webpack入口文件是哪个，\_\_dirname表示项目的根目录，它是nodeJS中的一个全局变量

output:{

path:\_\_dirname+”./public”,//表示打包后的文件存放的地方，webpack会自动生成publlic目录

filename:”bundle.js”//打包后生成的文件

}

}

直接在index.html中引入bundle.js即可

1. 项目本地执行打包命令webpack
2. 也可以使用jquery（第三方提供的包），安装jquery

npm install –save jquery

在需要用jquery的模块中引入即可(var $=require(“jquery”)),如在module1.js文件中需要使用jquery，如下：



如果需要在浏览器将各个模块分别输出显示各个js文件，以便于调试（在单独的文件中产生文件映射）则需在webpack.config.js文件中增加devtool:”source-map”配置项，会自动生成webpack目录，里面放的就是各个模块

1. webpack的命令

webpack –p命令是压缩生成的bundle.js文件

loader（是nodeJS包）

**说明：**loader是nodeJS包，处理非javascript文件的，比如css、less文件等

Demo事例如下：

在项目中创建css、less文件和引入一张图片

（1）先处理main.css文件，在主文件引入main.css文件（require(“./main.css”)），处理css文件方法如下：

1）安装css-loader，执行命令npm install –save-dev css-loader

2）安装style-loader,执行命令npm i –save-dev style-loader

3）在webpack.config.js文件中配置module配置项，当我们处理好css之后加载到bundle.js文件中然后style.loader把css文件插入到index.html文件中，如下：

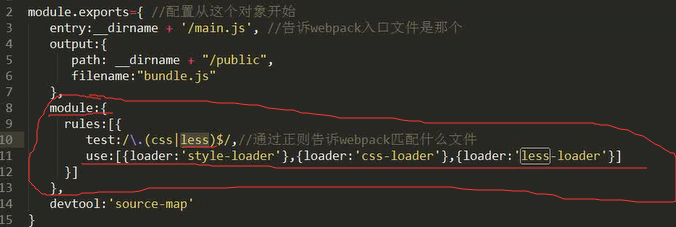


（2）如何处理main.less文件，在主文件中引入main.less文件（require(“./main.css”)），处理less文件如下：

1）安装less,执行命令npm install --save-dev less@2

2）安装less-loader,执行命令npm i –save-dev less-loader

2）在webpack.config.js文件中配置module，如下：



（3）如何编译二进制文件（图片），在主文件中引入图片var i=require(“图片路径”)，处理方法如下：

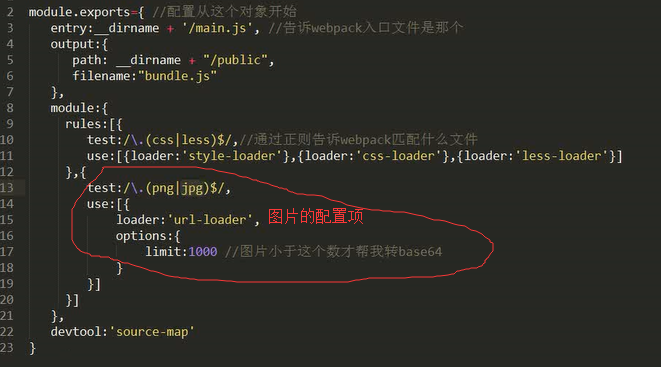
1）安装file-loader，npm I –save-dev file-loader

2）安装url-loader，npm I –save-dev url-loader

3）先要引入jquery文件， main.js主文件代码如下：



4）在webpack.config.js文件中配置module配置项，如下：



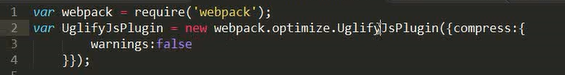
当配置options时，图片大于配置的属性值就会被转为路径（图片存在public中去了），此时路径应该是img.src=“./public/”+i;

Webpack的插件都在webpack.config.js中配置

1. UglifyJsPlugin压缩插件，是webpack自带的插件，运用如下：

1）在webpack.config.js文件中引入webpack

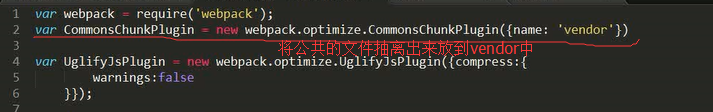
var webpack=require(“webpack”)

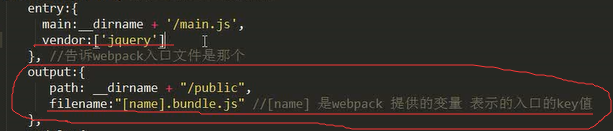
var UglifyJsPlugin=new webpack.optimize.UglifyJsPlugin();

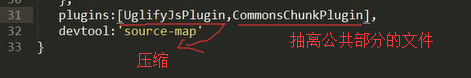
UglifyJsPlugin（）传的参数可以在文档中查找（直接百度即可）

2）在module.exports中配置

3）CommonsChunkPlugin：是webpack的内部插件，可以合并各个依赖包的公共文件，并单独打包出来。先引入插件，然后在entry中进行配置入口文件，修改输出保存的文件名，最后在module.exports中配置，如下：







webpack构建本地服务器

（1）安装webpack-dev-server，npm install –save-dev webpack-dev-server

（2）在主文件main.js中配置devServer配置项

devServer:{//配置的服务器

port:8090//port表示端口

historyApiFallback:true//告诉程序当url方式变化的时候永远指向index.html

}

（3）启动服务器，webpack-dev-server

（4）webpack-dev-server –hot –inline，--hot表示启动服务器的同时还能帮助开发者进行热替换（不需要刷新浏览器）内容；--inline表示把与服务器通讯的js文件注入到entry文件

（5）可以在package.json文件的script中配置快捷启动指令“dev”: webpack-dev-server –hot –inline，配置好后只需在命令行输入npm run dev即可启动与webpack-dev-server –hot –inline相同的指令

配置react的开发环境搭建开发环境

（1）安装处理react语法的loader包，npm i --save-dev babel-loader

（2）安装babel的核心库（babel-core），npm i --save-dev babel-core@6

（3）安装转ES6，npm i --save-dev babel-preset-es2015

（4）转ES7的是，npm i --save-dev babel-preset-stage-2

（5）转react的JSX语法的包，npm i --save-dev babel-preset-react

（6）安装在运行过程中转对象扩展的包，npm i --save-dev babel-plugin-transform-runtime

（7）在webpack.config.js文件中的module配置项中配置

{

test:/\.(js|jsx)$/,

exclude:/node\_modules/,// exclude表示告诉babel不要转这个目录下的文件

use:[{loader:’babel-loader’}]

}

（8）在根目录创建.babelrc文件，此文件表示告诉babel具体转哪些内容，在文件中配置

{

"presets": ["es2015", "stage-2","react"],

"plugins": ["transform-runtime"],

“comments”: false // 转码是否留注释

}

（9）安装react（15版本），npm install --save react@15