[**webpack之傻瓜式教程及前端自动化入门**](http://www.cnblogs.com/liqiyuan/p/6246870.html)

接触webpack也有挺长一段时间了，公司的项目也是一直用着webpack在打包处理，但前几天在教新人的情况下，遇到了一个问题，那就是：尽管网上的webpack教程满天飞，但是却很难找到一个能让新人快速入手的、接地气的、傻瓜式的教程，自己手把手教又太浪费时间。想了一想，决定自己写一篇webpack的傻瓜式教程，旨在教会尚未了解webpack的人迅速入手懂得基本原理，而不是懵逼一昧用着老员工搭好的环境天天敲着“webpack -p -w”却不知这到底是干嘛的。

对于webpack的概念、用途、好处等等一律省略，想了解的上网搜索即可，这里直接演示如何新建一个webpack的实例。

1、安装好nodejs，安装过程网上找。

2、在D盘、E盘或任意一个磁盘中新建文件夹，命名为webpack\_demo：

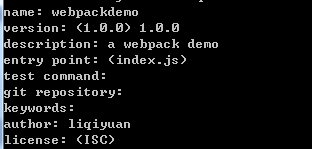


3、在webpack\_demo文件下按住Shift键后点击鼠标右键，再左键点击“在此处打开命令行窗口”：



在打开的命令行窗口中键入：

npm init



name表示项目名，version表示版本等等，不一一赘述了，看字段很容易明白，一一填完或直接一路按回车键，最后出现“is this ok？”时，敲击回车键后webpack\_demo的文件夹下便生成了package.json文件。

4、在命令行中分别执行以下命令，安装对应的插件（--save-dev作用是安装的插件的同时，将插件名写入package.json的devDependencies列表中）：

[复制代码](javascript:void(0);)

npm install webpack --save-dev

npm install jquery --save-dev

npm install style-loader css-loader --save-dev

npm install extract-text-webpack-plugin –-save-dev

[复制代码](javascript:void(0);)

安装完成后package.json的代码如下（name、version、description等都是在npm init步骤中手动添加而成）：

[复制代码](javascript:void(0);)

{

　　"name": "webpackdemo",

　　"version": "1.0.0",

　　"description": "a webpack demo",

　　"main": "index.js",

　　"scripts": {

　　　　"test": "echo \"Error: no test specified\" && exit 1"

　　},

　　"author": "liqiyuan",

　　"license": "ISC",

　　"devDependencies": {

　　　　"css-loader": "^0.26.1",

　　　　"extract-text-webpack-plugin": "^1.0.1",

　　　　"jquery": "^3.1.1",

　　　　"style-loader": "^0.13.1",

　　　　"webpack": "^1.14.0"

　　}

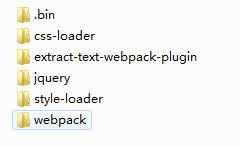
}

[复制代码](javascript:void(0);)

5、此时查看webpack\_demo文件夹，会发现多出了一个文件夹：

https://images2015.cnblogs.com/blog/1089038/201701/1089038-20170105164924191-1323397631.png

双击打开，文件夹下的内容如下：



说明所需要的插件都已安装成功。

6、在webpack\_demo文件下新建文件webpack.config.js：



编辑代码如下：

[复制代码](javascript:void(0);)

var ExtractTextPlugin = require("extract-text-webpack-plugin");

module.exports = {

　　entry: './js/main.js',

　　output: {

　　　　path: './dist',

　　　　filename: 'bundle.js'

　　},

　　plugins: [

　　　　new ExtractTextPlugin('./style.css')

　　],

　　module: {

　　　　//加载器配置

　　　　loaders: [

　　　　　　{

　　　　　　　　test: /\.css$/,

　　　　　　　　loader:ExtractTextPlugin.extract("style-loader","css-loader")

　　　　　　}

　　　　　　// {

　　　　　　　　// test: /\.(png|jpg)$/,

　　　　　　　　// loader: 'url-loader?limit=8192'

　　　　　　// }

　　　　]

　　}

};

[复制代码](javascript:void(0);)

以上代码的意思是：

引入extract-text-webpack-plugin这个插件（用于支持打包css文件，默认只支持打包js文件），读取当前目录下js文件夹中的main.js（入口文件）内容，把对应的js和css文件打包，打包后的文件放入当前目录的dist文件夹下，打包后的css文件命名为style.css，打包后的js文件名为bundle.js。

7、webpack\_demo下新建js文件夹，里面新建main.js，代码编辑如下：

[复制代码](javascript:void(0);)

require('./index/index.js');

require('./index/index2.js');

require('../css/index/index1.css');

require('../css/index/index2.css');

require('../css/common/public.css');

[复制代码](javascript:void(0);)

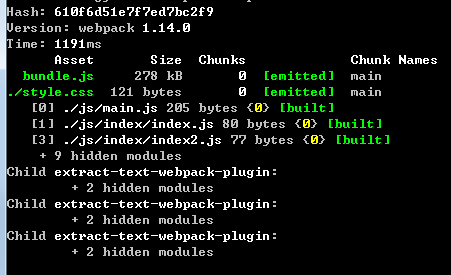
以上代码意思为：引入当前目录index文件夹下的index.js，index2.js，引入上一层目录css文件夹下index文件下的index1.css，index2.css和common文件夹下的public.css。

8、新建上一步中对应路径的css、js、index、common文件夹，里面分别新建对应的css或js文件，每个css、js文件里写入不同的代码。

9、窗口命令行中执行代码：

webpack

出现如下提示索命打包成功：



此时打开webpack\_demo文件夹，发现多了dist文件夹，打开，里面文件如下：



10、新建一个index.html文件，只引入bundle.js和style.css：

[复制代码](javascript:void(0);)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>webpack教程</title>

<meta http-equiv="X-UA-Compatible" content="IE=edge"/>

<meta name="viewport" content="width=device-width, initial-scale=1.0,user-scalable = no"/>

<link rel="stylesheet" type="text/css" href="./dist/style.css">

</head>

<body>

　　<div>webpack</div>

<script src="./dist/bundle.js"></script>

</body>

</html>

[复制代码](javascript:void(0);)

双击用浏览器打开这个html，会发现入口文件main.js里面引入的所有js和css文件都已经执行了。

**---------------------------------------------- 我是分割线----------------------------------------------**

上边是webpack的基本用法，可见它的基本作用是把一堆js和css压缩成一个js和css，但其实它的用法远不仅如此，这里再说两个常见的用法：

**1、自动加css3前缀。**

我们在写一些css3的样式时，为了良好的兼容性，往往会把css3代码写成如下这个样子：

.test{

　　display: flex;

　　display: -webkit-box;

　　display: -ms-flexbox;

}

如果每个css3的代码都要手动写一堆前缀，那不仅要奔溃还容易出错，这种事情还是交给工具来干，会效率和靠谱的多。

**2、ES6转码。**

ES6将会是以后的趋势了，但兼容性很差，转码后兼容性就大大提高了。

下面通过另一个小demo，把这一串东西给跑起来。

demo示例文件结构如下：



文件目录结构就不画图了，文字说明：

　　app：开发文件（编译前的文件）

　　　　css：手动编写的css文件集

　　　　　　css-entry.js：css入口文件，即说明需要打包哪些css文件

　　　　　　index.css：css示例文件1

　　　　　　index2.css：css示例文件2

　　　　js：手动编写的js文件集

　　　　　　index.js：js示例文件1

　　　　　　index.js：js示例文件2

　　dist：生产文件（编译后的文件）

　　　　css：编译后生成的css文件集

　　　　　　bundle.css：编译后生成的css文件，可直接引入使用

　　　　　　css.js：编译后生成的另一种css文件的用法，可先不管

　　　　js：编译后生成的jss文件集

　　　　　　bundle.js：编译后生成的js文件，可直接引入使用

　　node\_modules：webpack自动安装的各依赖包，不用管

　　package,json：配置文件，主要告诉电脑需要安装哪些依赖包

　　postcss.config.js：css3自动添加前缀的配置文件

　　webpack.config.js：webpack的配置文件

这个是什么意思呢，就是我们把自己写好的代码分别放进app的css和js文件夹里，打包以后，webpack会自动的将代码进行添加css3前缀、打包压缩、ES6转码等全部搞定后，将处理好的代码文件生成在dist的css和js里，我们在html里直接引用处理好以后的代码就行。

demo实战：

1、按上边的目录结构一一将文件新建好，node\_modules除外。

2、编写package.json，写明需要安装哪些依赖包，这些依赖包分别是干嘛的不一一说明了，想了解详细的可以直接复制依赖包名称去百度上搜。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | {    "name": "webpack",    "version": "1.0.0",    "description": "webpack, es6, css autoprefixer",    "author": "liqiyuan",    "license": "ISC",    "devDependencies": {      "autoprefixer": "^7.1.1",      "babel-core": "^6.24.1",      "babel-loader": "^7.0.0",      "babel-polyfill": "^6.23.0",      "babel-preset-env": "^1.6.1",      "css-loader": "^0.28.1",      "extract-text-webpack-plugin": "^2.1.0",      "postcss-loader": "^2.0.5",      "style-loader": "^0.17.0",      "webpack": "^2.5.1"    }  } |

3、编写webpack.config.js

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57 | // ES6转码和打包打包  // module.exports = {  //     entry: [  //         'babel-polyfill',        // 添加了这个东西，才能完美的将ES6转码  //         './app/js/index.js',     // 进行转码的js文件1  //         './app/js/index2.js',    // 进行转码的js文件2  //     ],  //     output: {  //         path: \_\_dirname + '/dist/js',    // 编译后的js文件输出目录  //         filename: 'bundle.js'            // 编译后的js文件名  //     },  //     module: {  //         loaders: [  //             {  //                 test: /\.jsx?$/,  //                 loader: 'babel-loader',     // 'babel-loader'也可写成'babel'  //                 query: {  //                     presets: ['env']  //                 }  //             }  //         ]  //     }  // };          // CSS自动添加前缀并打包  const ExtractTextPlugin = require("extract-text-webpack-plugin");  const extractCSS = new ExtractTextPlugin({      filename: './bundle.css',       // 编译后生成的css文件名      disable: false,      allChunks: true  });  module.exports = {      entry: {          app: ['./app/css/css-entry.js'],    // 需要进行打包处理的css入口文件      },      output: {          path: \_\_dirname + '/dist/css',      // 编译后生成的css文件输出目录          filename: 'css.js'      },      module: {          rules: [              {                  test: /\.css$/,                  use: extractCSS.extract({                      fallback: "style-loader",                      use: ['css-loader', 'postcss-loader']                  })              }          ]      },      plugins: [          extractCSS      ]  } |

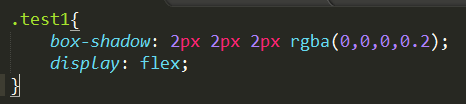
4、根目录下打开命令行窗口，输入“cnpm install”，即可按照package.json的配置内容自动安装所需插件（使用cnpm之前要先把淘宝镜像安装一下：npm install -g cnpm --registry=https://registry.npm.taobao.org），不然就用“npm install”。安装完成后，根目录下会生成一个文件夹node\_modules，这就是安装好的依赖包。

5、编辑postcss.config.js，这是css3自动添加前缀所需要的配置文件

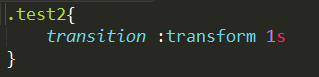
|  |  |
| --- | --- |
| 1  2  3  4  5 | module.exports = {    plugins: [      require('autoprefixer')({browsers:['last 5 versions']})    ]  } |

6、编辑app/css下面的css文件

index.css（css示例文件1）



index2.css（css示例文件2）

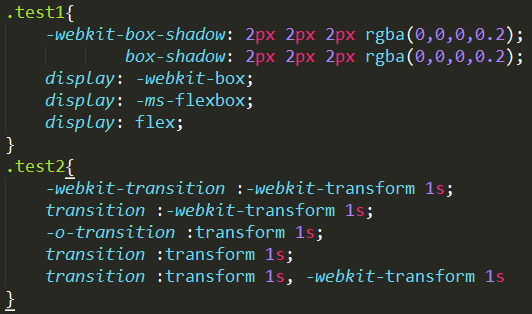


css-entry.js（css入口文件，即说明需要打包哪些css文件）

|  |  |
| --- | --- |
| 1  2 | require('./index.css');  require('./index2.css'); |

7、命令行中输入“webpack”。

8、神奇的情况发生，根目录下会生成dist文件夹，dist下有css文件夹，css下有bundle.css，打开bundle.css：



发现木有，本来index.css和index2.css里的css3代码是没有加任何前缀的，打包以后css3前缀都已经给加好了。

9、打包js文件和es6转码：编辑app/js目录下的index.js和index2.js，写上es6代码。再打开配置文件webpack.config.js，将 "ES6转码和打包" 部分代码放开，将 "CSS自动添加前缀并打包" 部分代码注释掉，命令行里敲击 "webpack"，打包完成后可以看见dist/js目录下会生成一个bundle.js的文件，里面正是被打包和转码好的js代码，可完美被IE9及以上的浏览器支持。

附上此例子的giuhub地址：<https://github.com/Leeqly/webpack-es6-autoprefixer>

可前往下载，然后直接cnpm install安装好依赖包后就可以使用了。

顺便提一下webpack几个常用命令

webpack：普通打包

webpack -p：压缩打包

webpack -p -w：监听所有需要打包的代码，只要有代码被改动并保存后，就会自动进行打包