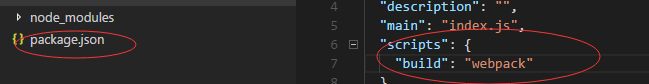
Webpack 配置

1. 先安装 npm install ---save-dev webpack

(在package.json中配置一个脚本，这个脚本的命令是webpack

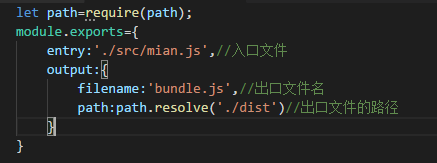


在命令行中输入，npm run build 即可进行打包

{在执行该命令后，会去node\_moudles下去的bin目录下去寻找对应的webpack命令执行，执行的就是wenpack.js,webpack.js又会去寻找一个在当前目录（即执行的npm run build这个目录）下的一个名叫webpack.config.js文件}

配置打包的文件，打包后的，打包方式

即配置webpack.config.js文件



**配置打包规则**

1. es6,es7转es5

1>先安装需要依赖的模块

npm install babel--core --save-dev

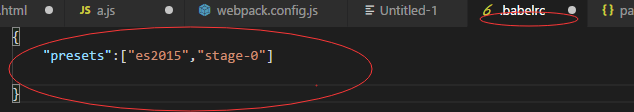
npm install babel-loader --save-dev

npm install babel-preset-es2015 --save-dev

npm install bable-preset-stage-0

2>在根目录下创建一个名为.babelrc的文件并进行配置

//在babel中，预设了6中，分别是es2015 ,stage-0 ,stage-1 ,stage-2 , stage-3 ,react



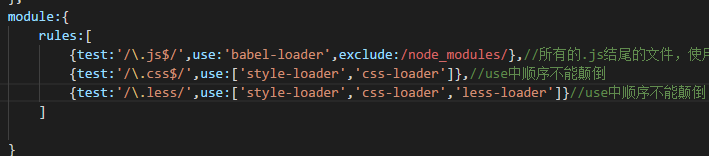
1. 在webpack-config.js中配置



1. css ,less,等文件打包方式配置
2. 先安装以来的模块

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

1. 在webpack-config.js中配置

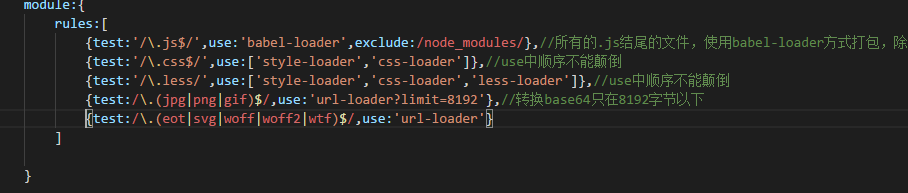


1. 图片打包方式
2. 安装依赖的模块
   1. npm install file-loader url-loader
   2. 配置webpack-config.js

url-loader依赖于file-loader，其他情况输出图片

加入规则{test:/\.(jpg|png|gif)$/,use:'url-loader?limit=8192'}

//转换base64只在8192字节以下

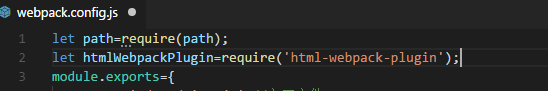
{test:/\.(eot|svg|woff|woff2|wtf)$/,use:'url-loader'

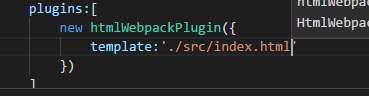
1. 让打包后的js文件自动导入到html文件中
2. 去安装以来的模块

npm install --save-dev html-webpack-plugin

1. 在webpack-config.js中的plugins中配置

先引入let htmlWebpackPlugin=require(‘html-webpack-plugin’)





1. webpack开服务时刻监视代码的变化

webpack-dev-server//webpack开发服务，内置了了个服务，可以帮我们启动一个端口号，当代码更新时，可以自动打包（内存中打包），时刻监控代码，代码有变化就重新执行

在package.json 的script中加入'dev':'webpack-dev-server'

命令行中输入 npm run dev 服务就跑起来了