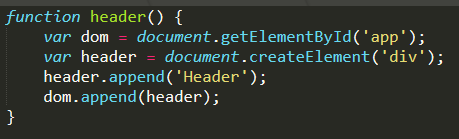
一. 为什么要使用Webpack, 什么是Webpack？

如果不使用任何框架，使用原生的js，如何实现在页面添加header,content,footer？



这样写虽然可以实现，但是如果页面很复杂，维护一个js就非常困难。于是我们可以将各个页面模块分开到不通的js中维护。比如header.js可以写成下面的方法



在html中引用该js，然后在index.js中调用header()方法，这样就形成了初步的代码分离。但是这样做也有很多问题，比如js的引用顺序，全局变量污染等问题，于是就有了amd，cmd，commonJs等模块化定义方法。我们最常用的就是commonJS和ES module两种，使用方法分别是：

commonJS： 导出 module.exports = header; 引用： var header = require('./header.js');

ES module： 导出 export default header; 引用： import header from './header.js';

这个时候又有一个问题，这些方法浏览器不认，跑不起来，于是就有了Webpack，可以将这些代码打包成一个浏览器可以运行的js，所以Webpack是一个模块打包工具。

二. 如何使用Webpack

1： 安装：

首先必须安装nodejs。

全局安装，可以通过npm命令，全局安装webpack 和 webpack-cli两个包



一般不用这么装，不通版本的Webpack配置可能不同。

本地安装，即在本项目中安装，也是通过npm命令



2: 使用

通过上面的代码，我们已经得到了一个分模块的代码，但是不能在浏览器上运行。现在我们可以对改代码进行打包，进入项目根目录，运行下面的指令即可实现打包。



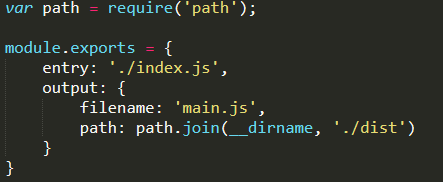
其中npx指的是运行本地Webpack，index.js是入口文件。

运行完成之后会生成一个目录dist和main.js文件，这个js即为Webpack打包之后浏览器可运行的文件。在html中引用该js就可以正常展示页面了。

虽然在这个项目中没有任何Webpack的配置，但是Webpack尽可能的简化的配置，给出了很多默认的配置。

3.Webpack配置文件

如果我们想要别的配置，就需要添加Webpack的默认配置文件，即在项目根目录下创建webpack.config.js文件即可。



在该文件中做一个最简单的配置，通过commonJs的形式导出配置，其中entry是入口文件配置，output是输出配置，filename即为输出js文件名，path是输出路径，path模块是nodejs自带的，所以可以直接引用，\_\_dirname为当前执行文件所在目录的完整目录名。

配置完成之后直接运行npx webpack即可。

4. package.json配置

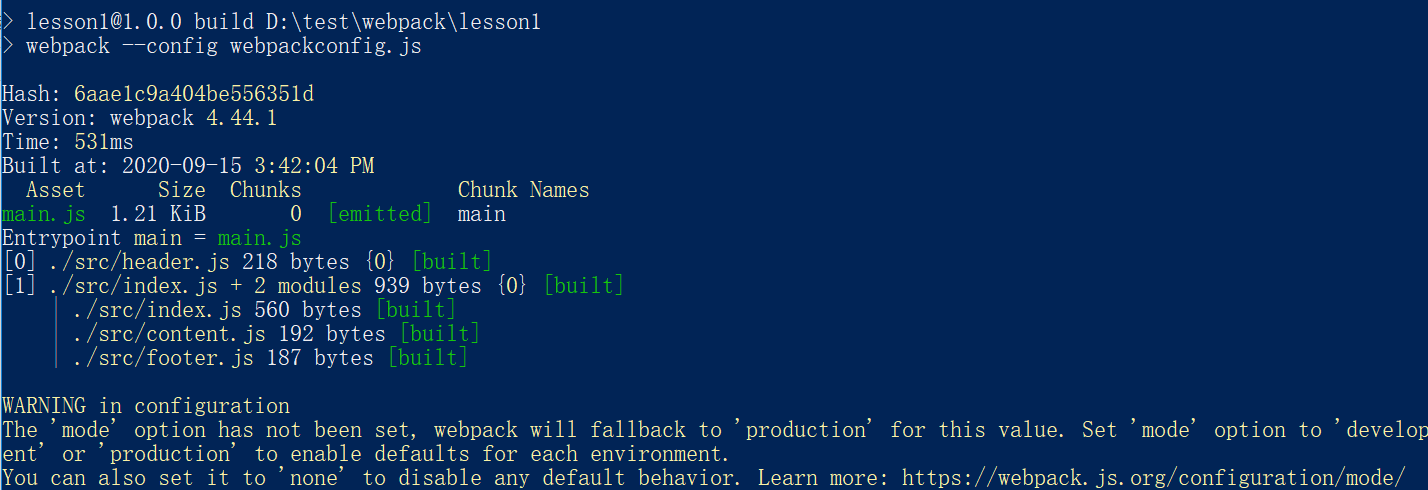
如果不想使用webpack.config.js作为默认文件，可以通过npm init初始化当前项目，会生成一个package.json文件，在其中的scripts下面配置



这样就可以指定任意文件，使用的时候用npm run build即可。

接下来我们再讲项目js文件放在src目录中，在webpack.config.js中讲entry改为’./src/index’，这样项目结构就和我们平时用的差不多了。

试运行一下



打包成功了，但是会有一个warning，需要添加打包的mode,可以设置为development或者production，区别在于打包后打代码会不会压缩。