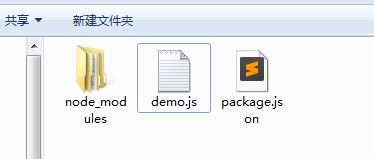
* 18

### **首先需要安装node.js**

### **安装webpack**

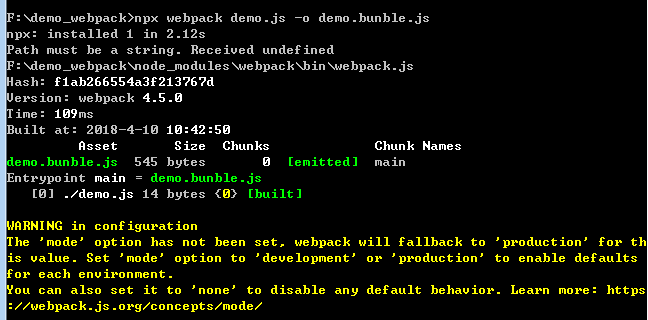
    命令行安装：npm install webpack webpack-cli --save-dev

### **在根目录下创建deme.js，看看webpack打包**



     用webpack将demo.js 打包成demo.bunble.js

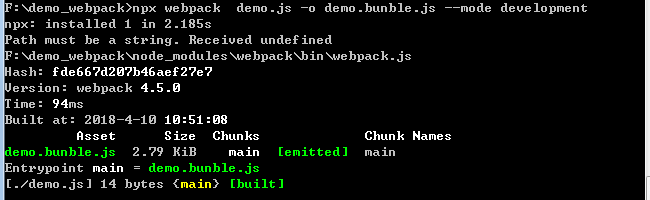
     npx webpack demo.js -o demo.bunble.js



    这里会出现黄色的警告：在打包时请指定mode 的取值，不给mode设定值的话，mode将被设为默认值production。mode的取值有：development，production，none。

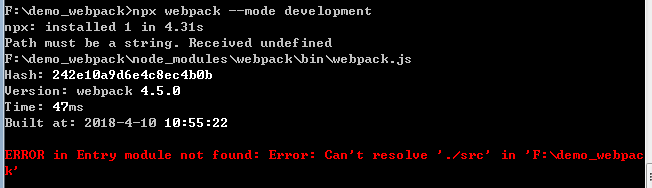
     现在为mode设置打包

     npx webpack demo.js -o demo.bunble.js --mode development



### **在不指定输入输出对象时**

        npx webpack --mode developement



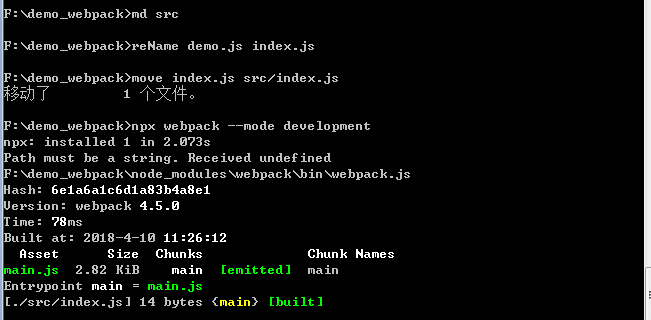
    报错是因为说在 ./src 下找不到 index.js 文件 - 这也是 webpack 4 引入的特性，默认输入文件位置在 src/index.js 下，现在建立src文件夹，将demo.js改名为index.js，将index.js移到src文件夹下

    md src

    reName demo.js index.js

    mv index.js src/index.js

    npx webpack --mode development



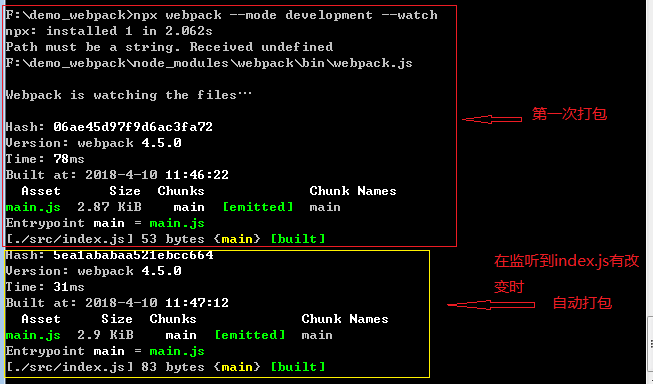
    打包成功，并且目录下多出 dist/main.js - 这也正是 webpack 4 开始默认的输出文件。

### 实时更新

##### **监控文件**

        webpack 有好几个解决办法，其中 watch 选项最直观，我们可以让 webpack 监控文件变化，一旦文件有变化，就重新构建。但默认情况下，watch 选项是禁用的。

    npx webpack --mode development --watch



##### **刷新浏览器**

        至于自动刷新浏览器的问题，webpack 提供 webpack-dev-server 来解决，它是一个基于 expressjs 的开发服务器，提供实时刷新浏览器页面的功能。不过目前 webpack-dev-server 已经进入维护模式，文档中推荐我们使用 webpack-serve。但这里暂时还是使用 webpack-dev-server 来说明。

        首先在根目录下 安装webpack-dev-server

            npm install webpack-dev-server --save-dev

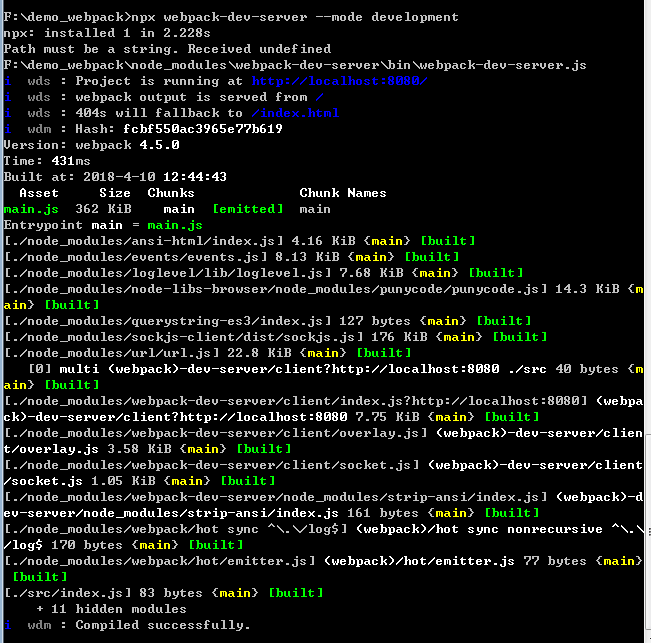
        启动webpack-dev-server

            npx webpack-dev-server --mode development

            可能会出现以下错误，再装一次webpack 和webpack-cli就好了



            成功启动后



            此时就算修改了index.js的内容，浏览器也不会响应

            原因：webpack-dev-server 构建的 main.js 其实是在 http://localhost:8080/main.js 的位置，而不是 http://localhost:8080/dist/main.js，而且，它存在于内存中，并不写入磁盘。而我们在 index.html 页面中引用的是 dist/main.js

            解决：在运行 webpack-dev-server 时指定 output.publicPath

               npx webpack-dev-server --mode development --output-public-path dist