1. 全局安装：

npm install webpack –g

安装完成之后输入 webpack –help 通过查看提示信息来检验是否安装完成。

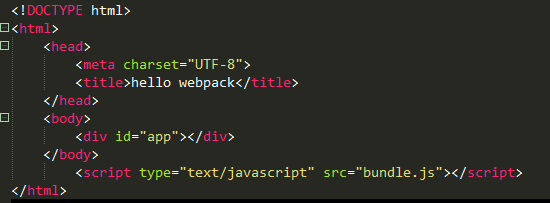
1. 在目标标目录下输入 npm init 进行一个npm的初始化。然后一路回车，然后再进行一个本地安装，和gulp的本地安装一样的命令
2. cnpm install webpack –-save-dev

**打包**

1. 在根目录下创建一个entry.js文件里面简单的写点东西



再在根目录下创建一个index页面，并简单的写点东西，并在index里面链入一个js文件，如：bundle.js，这个js文件会通过webpack生成，这是我们的输出文件。同时index里面的id要和entry.js里的id一致



1. 输入webpack entry.js bundle.js会 在根目录下生成一个bundle.js的文件,此时在entry.js中你自己写入的业务代码已经打包进了bundle.js中。你可以打开bundle.js文件去查看里面的代码，可以看到你在entry.js中写入的业务代码，在最下面。





此时你打开浏览器看看index，看到个hello就对了

1. 在根目录下创建一个name.js文件 在里面定义一个module并暴露，再简写点东西比如：



1. 在入口文件entry.js中引入name.js所暴露的借口

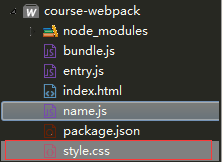


5.然后输入webpack entry.js bundle.js将入口文件的内容打包进bundle.js，然后在浏览器中打开页面看效果，会看到页面中会生成hellocourse webpack；

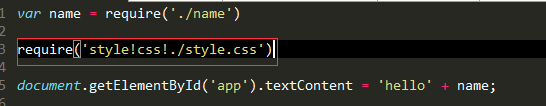
**Css文件的引入**

1.这个功能是靠webpack的loader功能实现的。所以我们需要输入：npm i css-loader style-loader –-D来安装这两个loader

C:\Users\Administrator\Desktop\QQ截图20161019170334.png2.在根目录下创建一个css文件，并写入一些简单的样式 如：body{background：red}

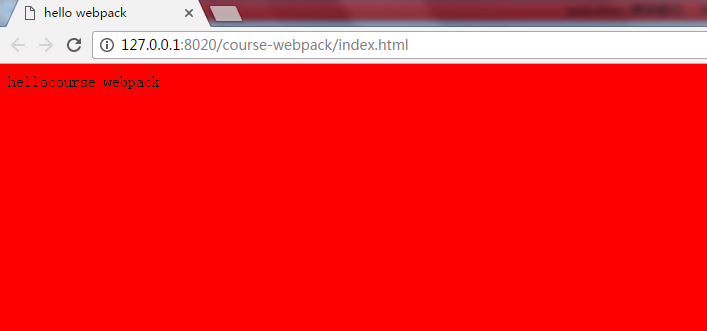


3在入口文件中写入引用css的代码



此处的style！和css！是指加载的我们刚才安装的loader

4.输入webpack entry.js bundle.js命令，打包到bundle.js，然后查看index网页



说明我们的css已经已经成功的引入，也可以通过查看bundle.js中的代码，可以看到引进了许多模块。

**Webpack的配置文件**

1. 在根目录下创建一个webpack.config.js文件，然后再里面写入webpack的配置，并暴露出去。



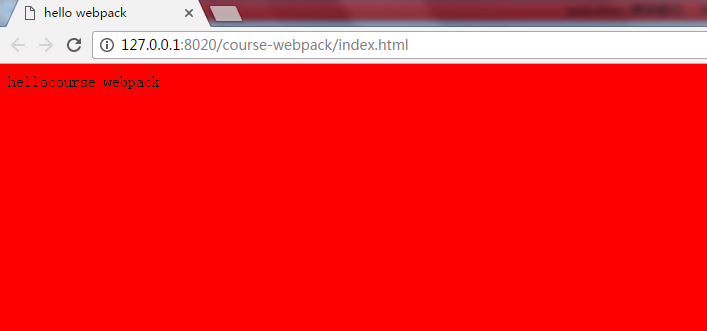
这其中由于已经配置好了loader，所以在入口文件entry.js中的引入的两个css和style的loader就可略掉了省略





同时，在命令行中也不用专门写webpack entry.js bundle.js这样的命了，直接输入webpack命令即可，他会根据我们的配置自动去打包到bundle.js中。

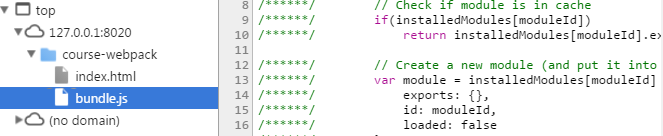
1. 输入webpack命令，然后查看index页面



引入成功

**通过source-map来调试代码**

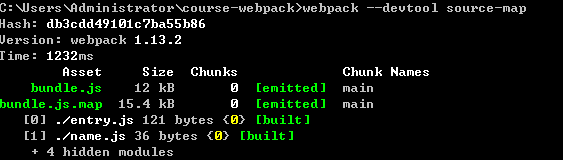
在webpack中必须使用source-map才能够生成这些模块的源文件，才能在浏览中查看源码并进行调试。



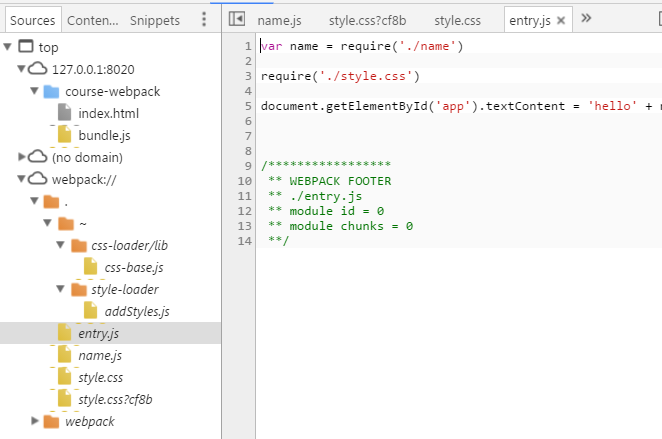
浏览器的Sources中我们只能看index.html和bundle.js两个源文件，我们创建的entry.js和name.js是看不到的

1. 输入webpack --devtool source-map

这样就会生成一个source-map

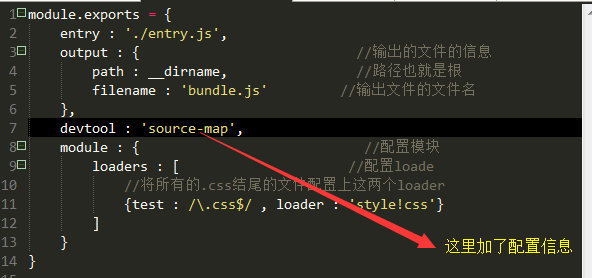


1. 回到浏览器查看Sources，会多出一个webpack文件夹，里面就有我们之前创建的文件了并且也是能看到源码的



可以在name中加入一个debugger来看一下变化

1. 同理，我们也可以在config文件中设置devtool来让我们像刚才一样，只需要输入一个webpack命令就能执行source-map的功能



**Webpack结合bable的使用**

**Javascript2015年第6次改版（ES6）**

Bable主要是用来处理ES2015也就是ES6的，由于ES6的一些代码有的浏览器是不能直接解析的，所以就必须用bable将其转换为浏览可以解析的代码。这就是bable。

1. 安装



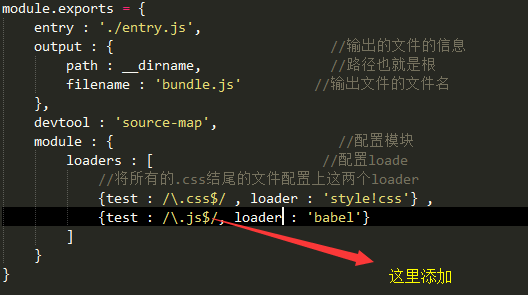
这里面分别安装了babel-loader这个不用解释吧，还有就babel-core，这个就是babel。

然后是babel-preset-es2015，这是babel的预设，就是他解析ES6的。

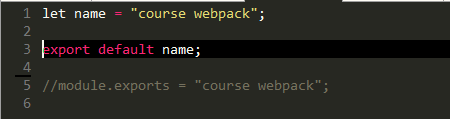
1. 添加预设 在根目录下创建一个名字为.babelrc的自定义文件，在里面添加预设

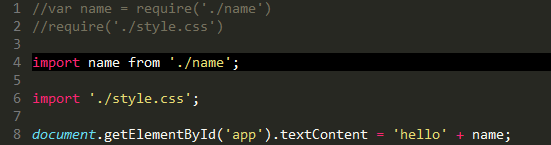
写成这样就行了

1. 配置文件中添加babel的配置

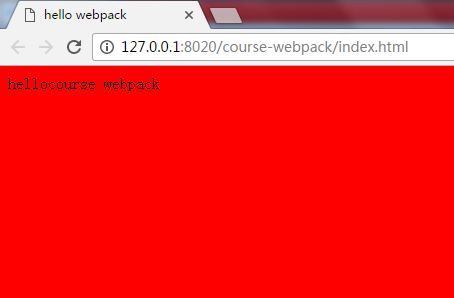


4,写点es6的代码，比如讲name.js中的和entry.js中的代码改成es6的





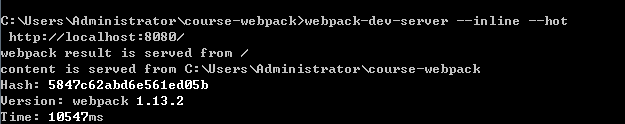
1. 然后运行webpack命令，看index网页



效果还能实现

**Webpack-dev-server模块热替换**

1. 先全局安装 webpack-dev-server 这里就不截图了
2. 再本地安装，也不截图了就
3. 启动webpack-dev-server并添加热替换这个参数



1. 这个时候热交换就有了，同时开了一个8080的端口，通过改变css里的背景颜色，不用刷新页面就能直接变色，不信你试试。

**Webpack与react+模块热交换**

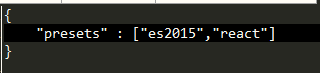
1. 安装一系列的组件





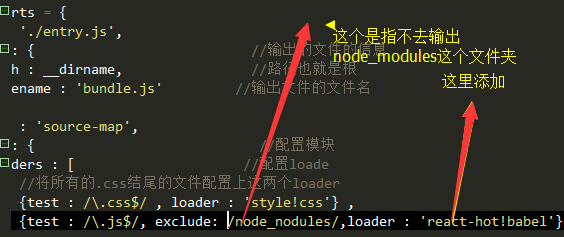
总共需要这么多，但是有的我们之前就装过，所以自己可以看哪个没有去装它，或者直接都再装一遍

1. 修改预设文件；

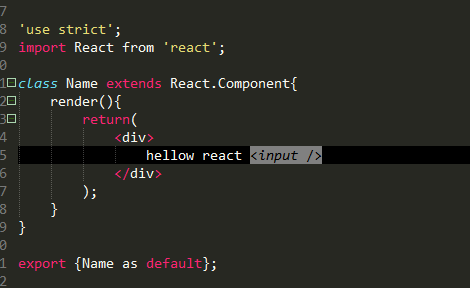


在这个预设文件的数组里添加react的预设

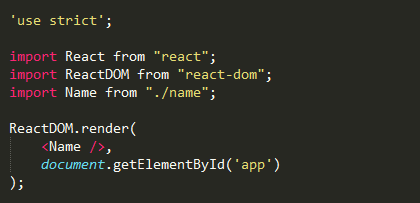
1. 在配置文件中加入react-hot这个loader



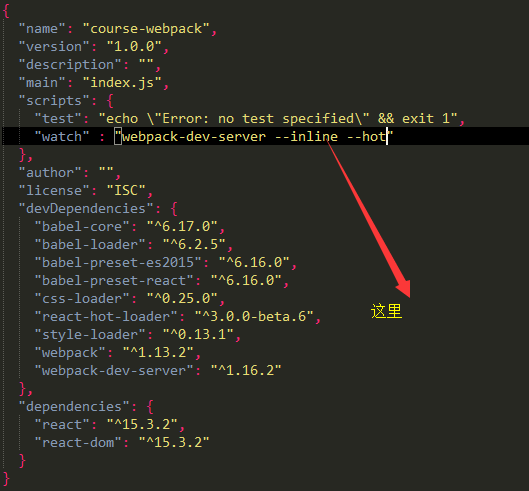
1. 把name.js改成react的



1. 再把entry.js改为react的



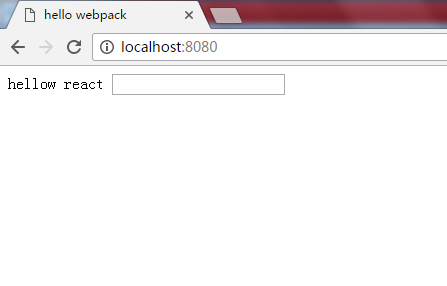
1. 在package.json文件夹中添加一对类似缩写或者是改变写法的一个参数



这样我们就不用在命令行中专门的去输入webpack-dev-server去启动端口了，

直接输入npm run watch即可。

1. 输入命令完成之后我们去看index



可以可以。