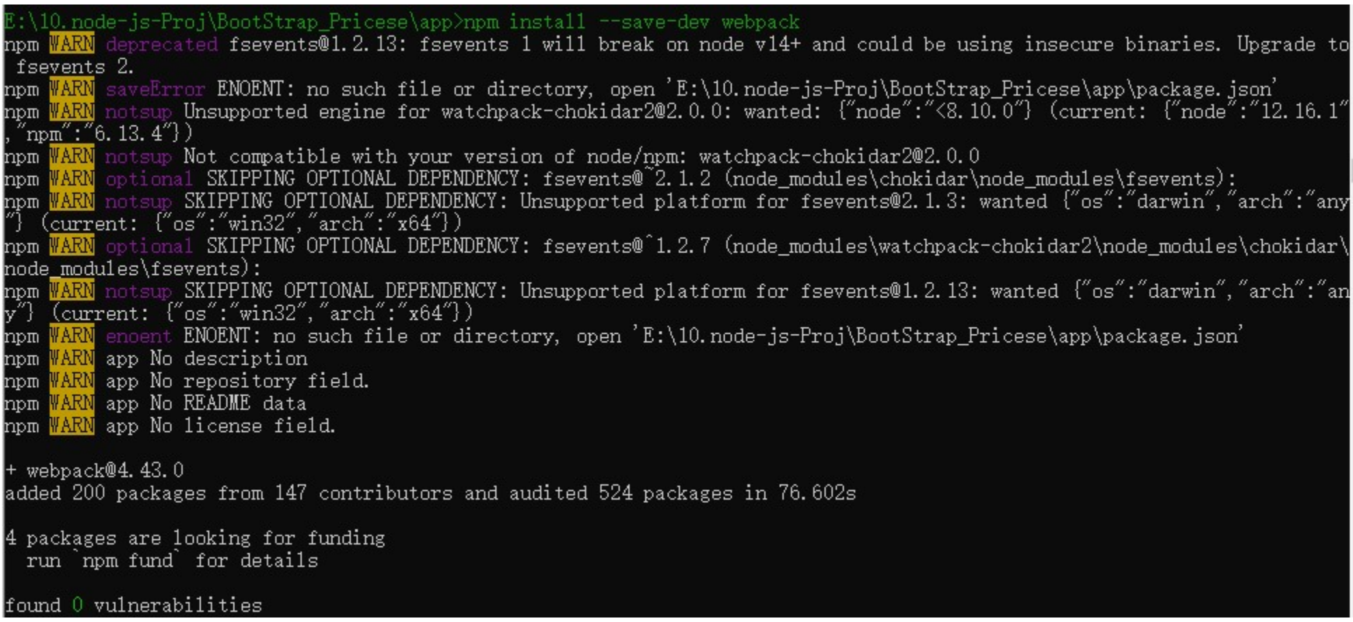
WebPack的安装使用(亲测)

Webpack 可以将多种静态资源 js、css、less 转换成一个静态文件，减少页面的请求。

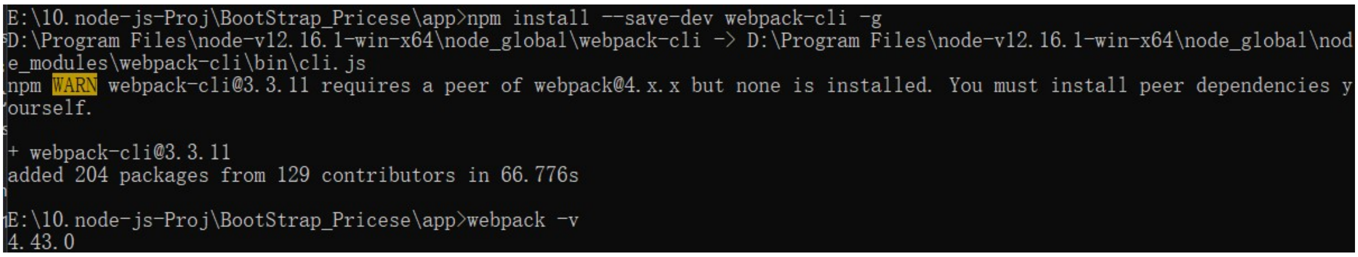
1.安装WebPack

```
- 本地环境要求node.js
- 建议使用cnmp进行安装，
  cnmp install webpack -g （菜鸟教程上推荐安装方式）
  使用淘宝镜像安装：npm install -g cnpm --registry=https://registry.npm.taobao.org
  如果安装过程报错：npm resource busy or locked...可以先删除之前安装的node_modules
  npm cache clean
  npm install

经过摸爬滚打，我总结了一下自己的安装过程：
1. 安装webpack:  npm install --save-dev webpack -g (建议全局安装)，安装后会有版本提示信息
```



```
E:\10.node-js-Proj\BootStrap_Pricese\app>npm install --save-dev webpack
npm WARN deprecated fsevents@1.2.13: fsevents 1 will break on node v14+ and could be using insecure binaries. Upgrade to
 fsevents 2.
npm WARN saveError ENOENT: no such file or directory, open 'E:\10.node-js-Proj\BootStrap_Pricese\app\package.json'
npm WARN notsup Unsupported engine for watchpack-chokidar2@2.0.0: wanted: {"node":"<8.10.0"} (current: {"node":"12.16.1"
,"npm":"6.13.4"})
npm WARN notsup Not compatible with your version of node/npm: watchpack-chokidar2@2.0.0
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.1.2 (node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.3: wanted {"os":"darwin","arch":"any
"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@^1.2.7 (node_modules\watchpack-chokidar2\node_modules\chokidar\
node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"an
y"} (current: {"os":"win32","arch":"x64"})
npm WARN enoent ENOENT: no such file or directory, open 'E:\10.node-js-Proj\BootStrap_Pricese\app\package.json'
npm WARN app No description
npm WARN app No repository field.
npm WARN app No README data
npm WARN app No license field.

+ webpack@4.43.0
added 200 packages from 147 contributors and audited 524 packages in 76.602s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

```
2. 安装webpack-cli: npm install --save-dev webpack-cli -g (建议全局安装)
```

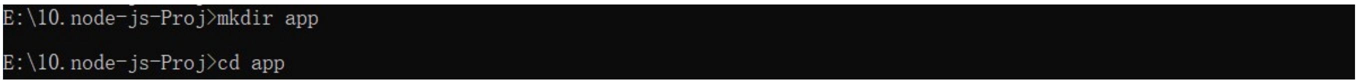


```
E:\10.node-js-Proj\BootStrap_Pricese\app>npm install --save-dev webpack-cli -g
D:\Program Files\node-v12.16.1-win-x64\node_global\webpack-cli -> D:\Program Files\node-v12.16.1-win-x64\node_global\nod
e_modules\webpack-cli\bin\cli.js
npm WARN webpack-cli@3.3.11 requires a peer of webpack@4.x.x but none is installed. You must install peer dependencies y
ourself.

+ webpack-cli@3.3.11
added 204 packages from 129 contributors in 66.776s

E:\10.node-js-Proj\BootStrap_Pricese\app>webpack -v
4.43.0
```

```
3. 测试安装是否成功: webpack -v
```

2.创建项目

新建一个目录，**mkdir app**



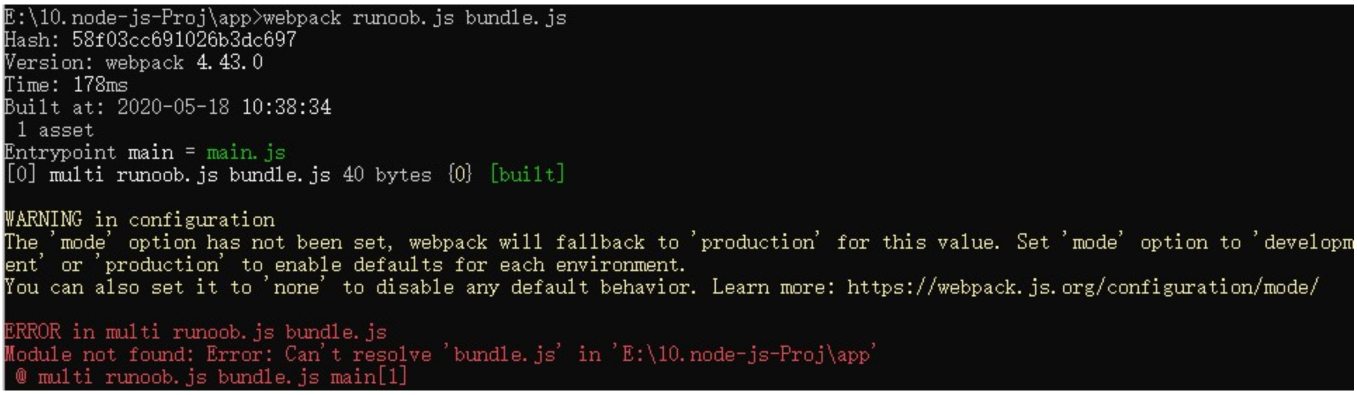
```
E:\10.node-js-Proj>mkdir app

E:\10.node-js-Proj>cd app
```

*创建1个js

在 **app** 目录下添加 **runoob1.js** 文件，代码：**document.write("the ocean");**
在 **app** 目录下添加 **index.html** 文件，代码如下：

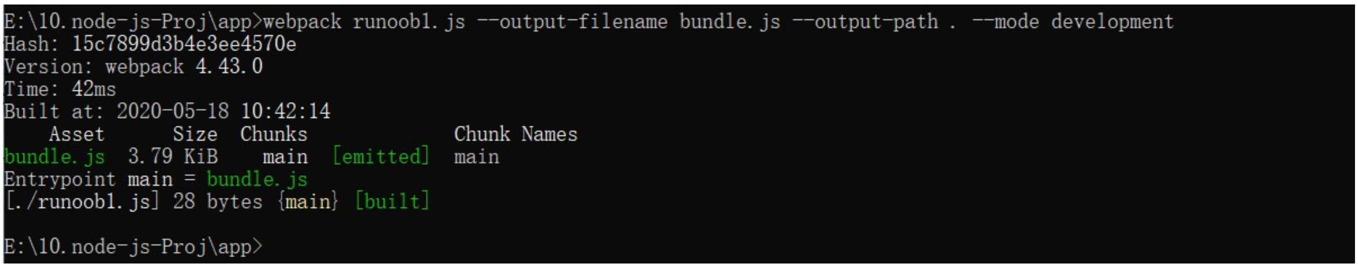
```
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <script type="text/javascript" src="bundle.js" charset="utf-8"></script>
    </body>
</html>
```

使用命令来进行打包：
webpack runoob1.js bundle.js会产生错误：



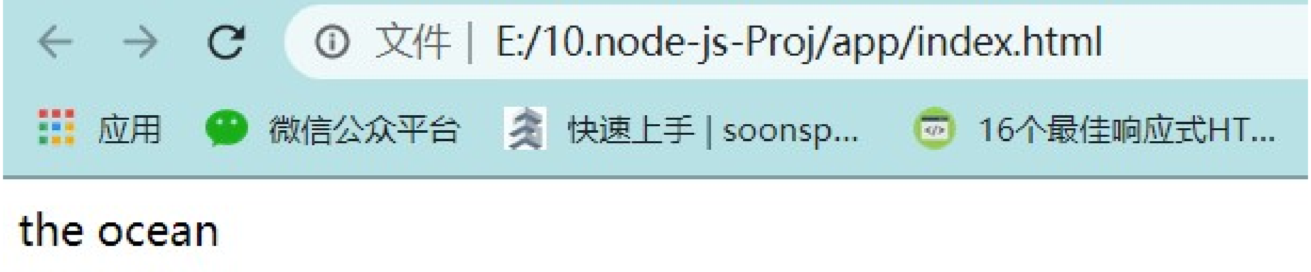
```
E:\10.node-js-Proj\app>webpack runoob.js bundle.js
Hash: 58f03cc691026b3dc697
Version: webpack 4.43.0
Time: 178ms
Built at: 2020-05-18 10:38:34
 1 asset
Entrypoint main = main.js
[0] multi runoob.js bundle.js 40 bytes {0} [built]

WARNING in configuration
The 'mode' option has not been set, webpack will fallback to 'production' for this value. Set 'mode' option to 'developm
ent' or 'production' to enable defaults for each environment.
You can also set it to 'none' to disable any default behavior. Learn more: https://webpack.js.org/configuration/mode/

ERROR in multi runoob.js bundle.js
Module not found: Error: Can't resolve 'bundle.js' in 'E:\10.node-js-Proj\app'
 @ multi runoob.js bundle.js main[1]
```

所以应该使用命令：**webpack runoob1.js --output-filename bundle.js --output-path . --mode deve**



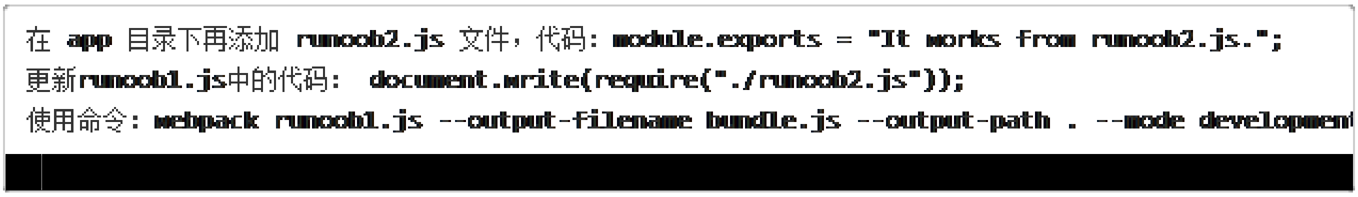
```
E:\10.node-js-Proj\app>webpack runoob1.js --output-filename bundle.js --output-path . --mode development
Hash: 15c7899d3b4e3ee4570e
Version: webpack 4.43.0
Time: 42ms
Built at: 2020-05-18 10:42:14
    Asset      Size  Chunks             Chunk Names
bundle.js  3.79 KiB    main  [emitted]  main
Entrypoint main = bundle.js
[./runoob1.js] 28 bytes {main} [built]

E:\10.node-js-Proj\app>
```

浏览器打开**index.html**

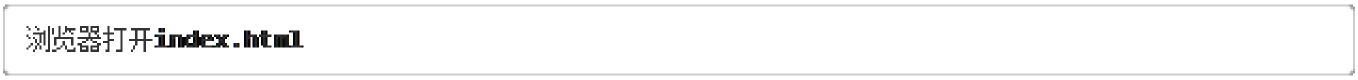


the ocean

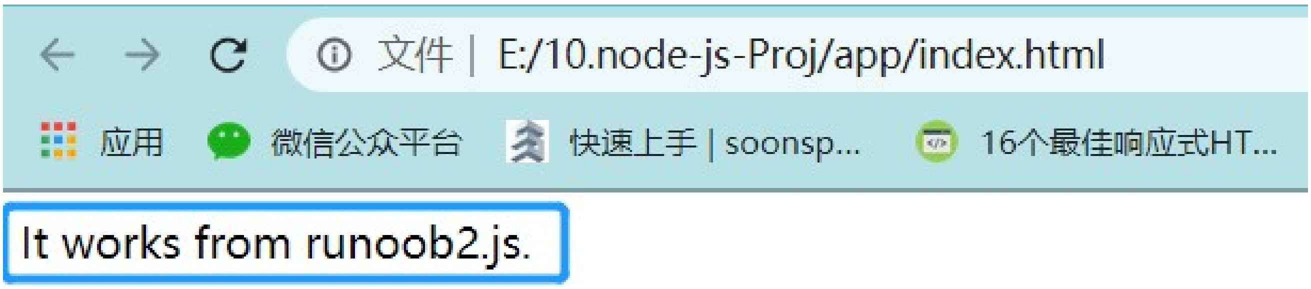
*创建2个js



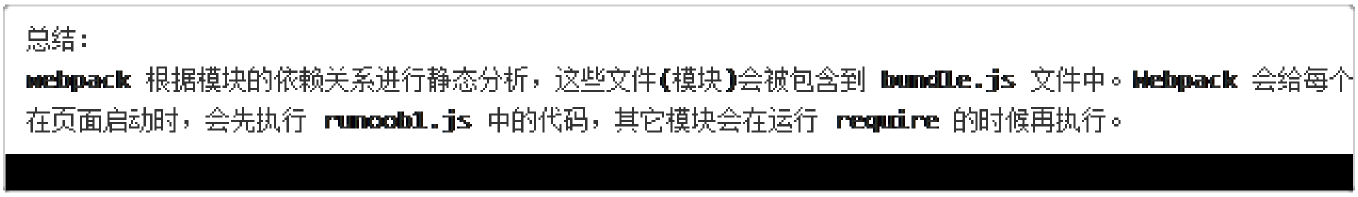
```
在 app 目录下再添加 runoob2.js 文件，代码: module.exports = "It works from runoob2.js.";
更新runoob1.js中的代码:  document.write(require("./runoob2.js"));
使用命令: webpack runoob1.js --output-filename bundle.js --output-path . --mode developmen
```



浏览器打开index.html

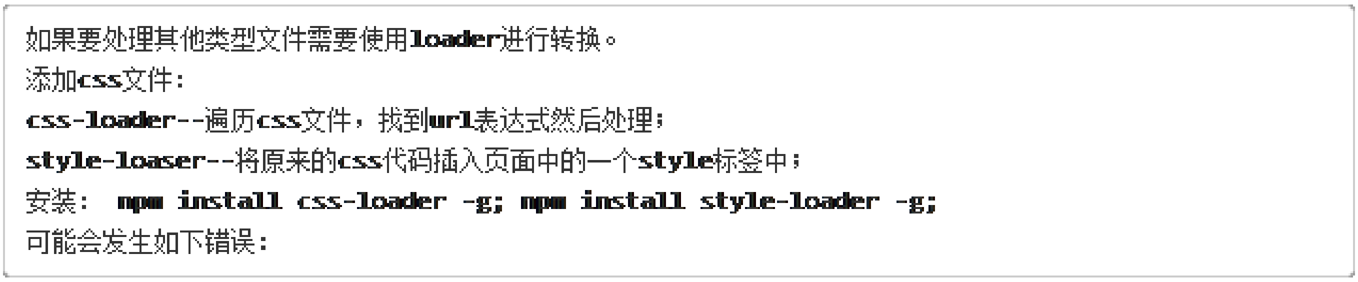


It works from runoob2.js.

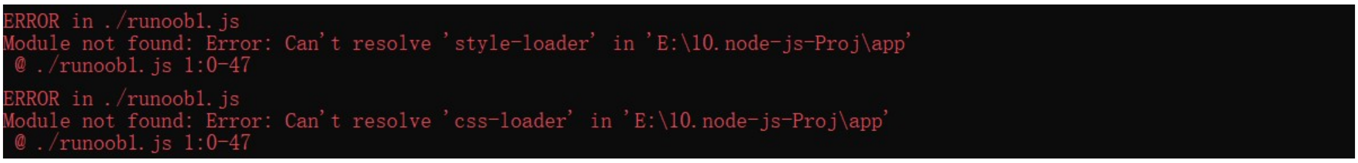


总结：
webpack 根据模块的依赖关系进行静态分析，这些文件(模块)会被包含到 bundle.js 文件中。Webpack 会给每个
在页面启动时，会先执行 runoob1.js 中的代码，其它模块会在运行 require 的时候再执行。

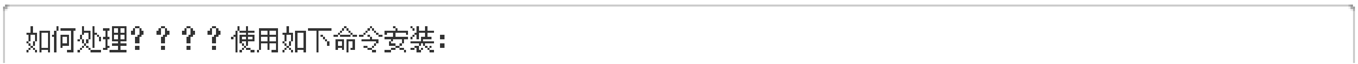
3.loader



如果要处理其他类型文件需要使用loader进行转换。
添加css文件：
css-loader--遍历css文件，找到url表达式然后处理；
style-loaser--将原来的css代码插入页面中的一个style标签中；
安装： npm install css-loader -g; npm install style-loader -g;
可能会发生如下错误：



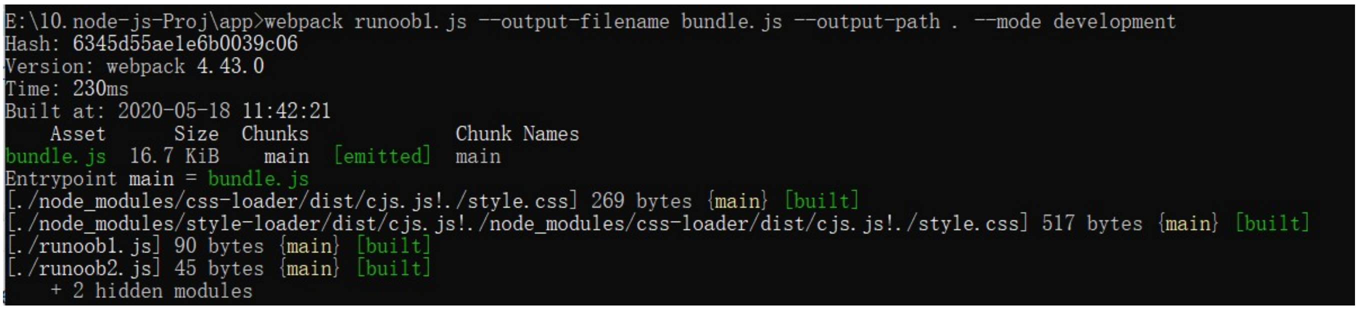
```
ERROR in ./runoob1.js
Module not found: Error: Can't resolve 'style-loader' in 'E:\10.node-js-Proj\app'
 @ ./runoob1.js 1:0-47

ERROR in ./runoob1.js
Module not found: Error: Can't resolve 'css-loader' in 'E:\10.node-js-Proj\app'
 @ ./runoob1.js 1:0-47
```



如何处理？？？？使用如下命令安装：

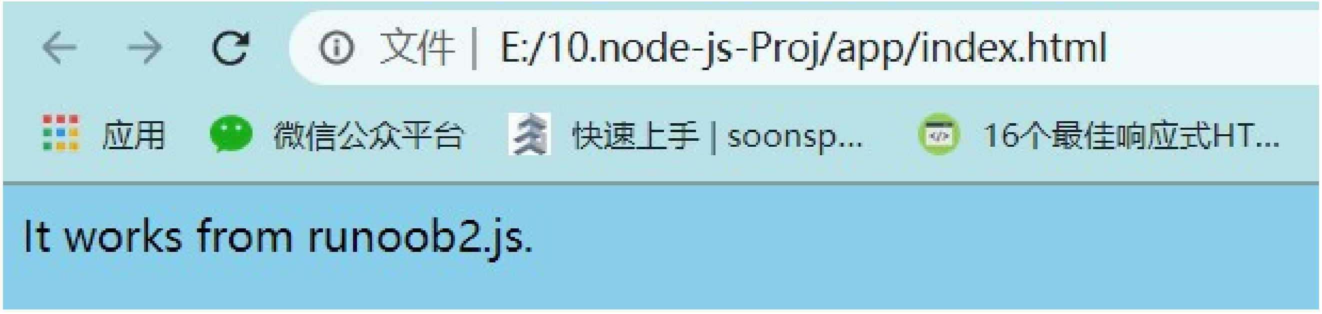
```
npm install style-loader --save-dev -g
npm install css-loader --save-dev -g
```

```
在app目录下新建style.css文件，代码：body{ background: skyblue; }
更新runoob1.js中的代码：
require("!style-loader!css-loader!./style.css");
document.write(require("./runoob2.js"));

使用命令：webpack runoob1.js --output-filename bundle.js --output-path . --mode developmen
发生错误：
ERROR in ./runoob1.js
Module not found: Error: Can't resolve 'css-loader' in 'E:\10.node-js-Proj\app'@ ./runoob
```



```
E:\10.node-js-Proj\app>webpack runoob1.js --output-filename bundle.js --output-path . --mode development
Hash: 6345d55ae1e6b0039c06
Version: webpack 4.43.0
Time: 230ms
Built at: 2020-05-18 11:42:21
    Asset      Size  Chunks             Chunk Names
bundle.js  16.7 KiB    main  [emitted]  main
Entrypoint main = bundle.js
[./node_modules/css-loader/dist/cjs.js!./style.css] 269 bytes {main} [built]
[./node_modules/style-loader/dist/cjs.js!./node_modules/css-loader/dist/cjs.js!./style.css] 517 bytes {main} [built]
[./runoob1.js] 90 bytes {main} [built]
[./runoob2.js] 45 bytes {main} [built]
    + 2 hidden modules
```

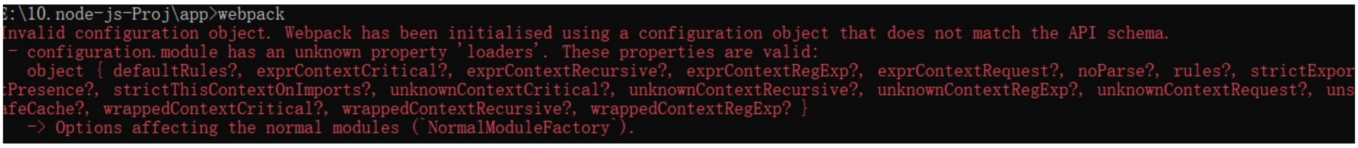
```
浏览器打开index.html
```



It works from runoob2.js.

4.配置文件

```
可以将一些编译选项放在配置文件中，便于管理：
app目录下信息新建webpack.config.js 文件，代码如下所示(webpack3.0的写法，如果版本不匹配的话会报错)：
module.exports = {
    entry: "./runoob1.js",
    output: {
        path: __dirname,
        filename: "bundle.js"
    },
    module: {
        loaders: [
            { test: /\.css$/, loader: "style-loader!css-loader" }
        ]
    }
};
执行webpack命令即可，可能会报错：
```



```
:\10.node-js-Proj\app>webpack
nvalid configuration object. Webpack has been initialised using a configuration object that does not match the API schema.
- configuration.module has an unknown property 'loaders'. These properties are valid:
  object { defaultRules?, exprContextCritical?, exprContextRecursive?, exprContextRegExp?, exprContextRequest?, noParse?, rules?, strictExpor
Presence?, strictThisContextOnImports?, unknownContextCritical?, unknownContextRecursive?, unknownContextRegExp?, unknownContextRequest?, uns
feCache?, wrappedContextCritical?, wrappedContextRecursive?, wrappedContextRegExp? }
  -> Options affecting the normal modules (`NormalModuleFactory`).
```

原因:这个配置文件的版本和我们当前安装的webpack的版本不匹配。接下来这段我们只需要看前面一句，意思是webpack.config.js这个配置文件里的module属性有一个未知的配置项loaders，原因就是我们当前安装的webpack版本已经去掉了这个配置。webpack的版本并没有向后兼容。

```
webpack.config.js匹配写法:

const path = require("path")
module.exports = {
    // 入口文件可以有多个，多个的时候用数组的形式写
    entry: {
        main: "./runoob1.js",
    },

    // 打包环境: 开发development/生产production
    mode: "development",

    // 出口，只能有一个
    output: {
        filename: "bundle.js", //[name]-bundle.js
        path: path.resolve(__dirname, "") //生成文件在app目录下,也可以../dist
    }
}
```

5.开发环境

```
当项目逐渐变大，webpack 的编译时间会变长，可以通过参数让编译的输出内容带有进度和颜色。代码：webpack -
监听模式：webpack --progress --colors --watch
通过一个 socket.io 服务实时监听它们的变化并自动刷新页面：
# 安装
cnpm install webpack-dev-server -g

# 运行
webpack-dev-server --progress --colors
```