webpack

---

webpack

webpack打包工具

babel转义es6 -> es5

babel-preset-es2015

babel-preset-stage-0

解析样式

less,sass,stylus(预处理语言)

解析图片

需要解析html插件

webpack-dev-server

---

webpack打包工具

- 前端用:import
- 后台用:require
    - 1.src:存储是源码

    - 2.dist:打包后代码(生产后代码)

- webpack必须采用commonjs写法
  安装

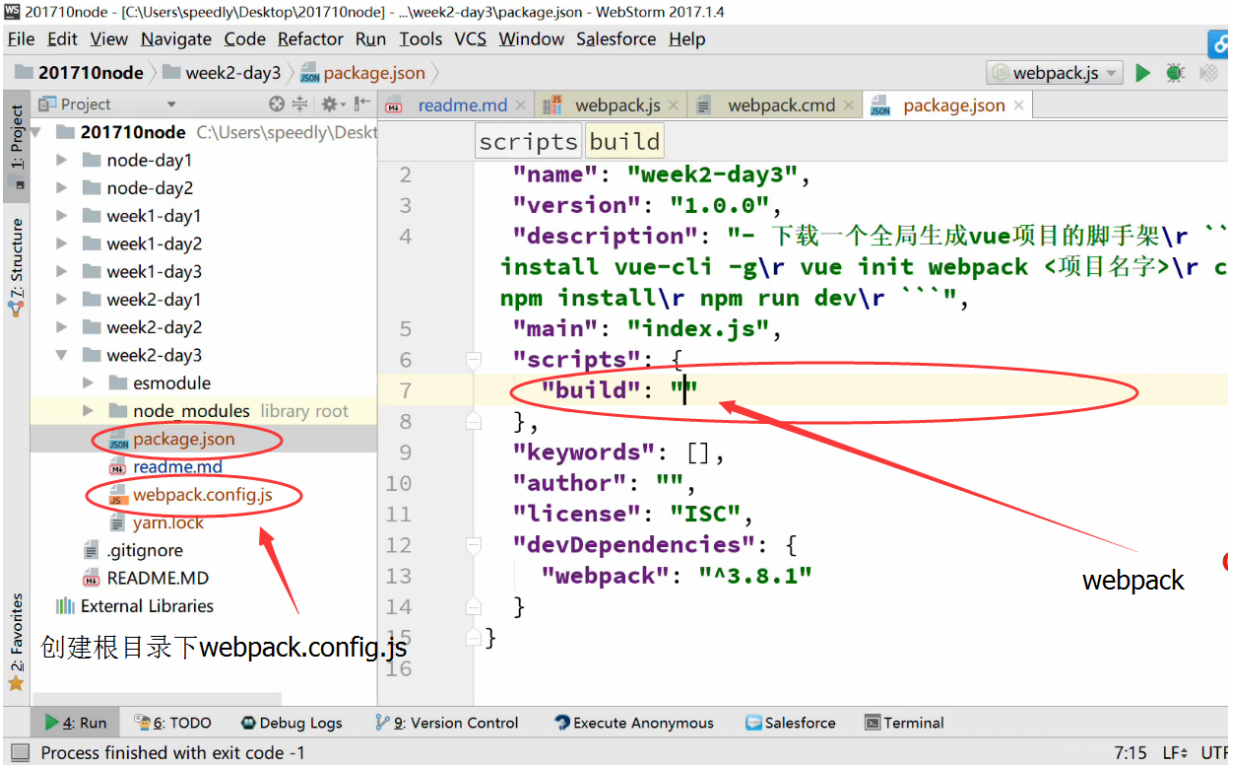
```
npm init -y
npm install webpack -save-dev
```

- 安装webpack或者less最好不要安装全局的，否则可能导致webpack的版本差异

> 1.项目根路径创建:webpack.config.js
> 2.在package.json中配置一个脚本，这个脚本用的命令是webpack.会去当前的node_modules下找bin对应的webpack名字让其执行，
> 3.执行的就是bin/webpack.js,webpack.js需要当前目录下有个名字叫
> 4.webpack.config.js的文件，我们通过npm run build执行的目录是当前文件的目录，所以会去当前目录下查找

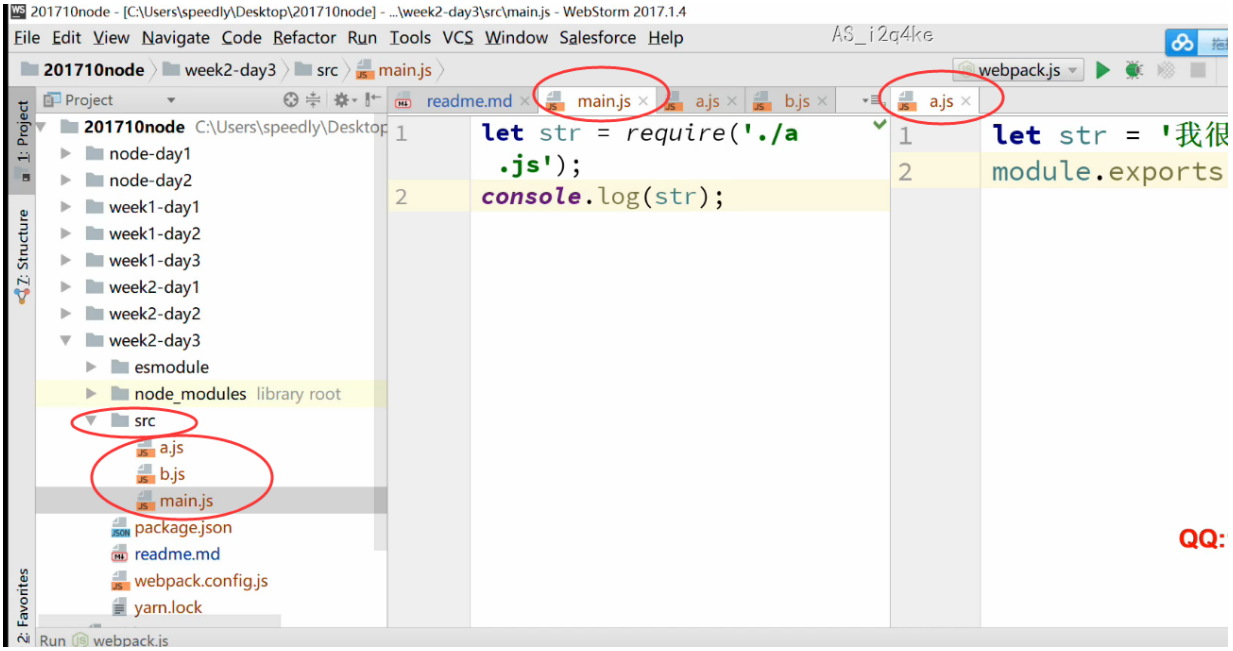
1.项目根路径创建:webpack.config.js

2.配置一个脚本



举个栗子

需要打包项目:



1.webpack.config.js内容:比如配合一个文件

```
// webpack必须采用commonjs写法
let path = require('path'); // 专门处理路径用的,以当前路径解析出一个绝对路径
console.log(path.resolve('./dist'));
module.exports = {
    entry:{
        entry:'./src/min.js',//打包的入口文件，webpack会自动查找相关的依赖进行打包
    output:{
        filename:'bundle.js',//打包后的名字
        path:path.resolve('./dist') //必须是一个绝对路径
    }
};
```

执行命令:

```
npm run build
```

src多个文件打包

```
同上....
module.exports = {
    entry:{
        main:'./src/main.js',
        main1:'./src/main1.js'
    }, // 打包的入口文件，webpack会自动查找相关的依赖进行打包
    output:{
        filename:'[name].js',//打包后的名字
        path:path.resolve('./dist') //必须是一个绝对路径
    }
};
```

ES6转化

```
import xxx from './b.js';
console.log(xxx);
```

babel转义es6 -> es5

- 安装babel

```
npm install babel-core --save-dev
npm install babel-loader --save-dev
```

2.快速安装

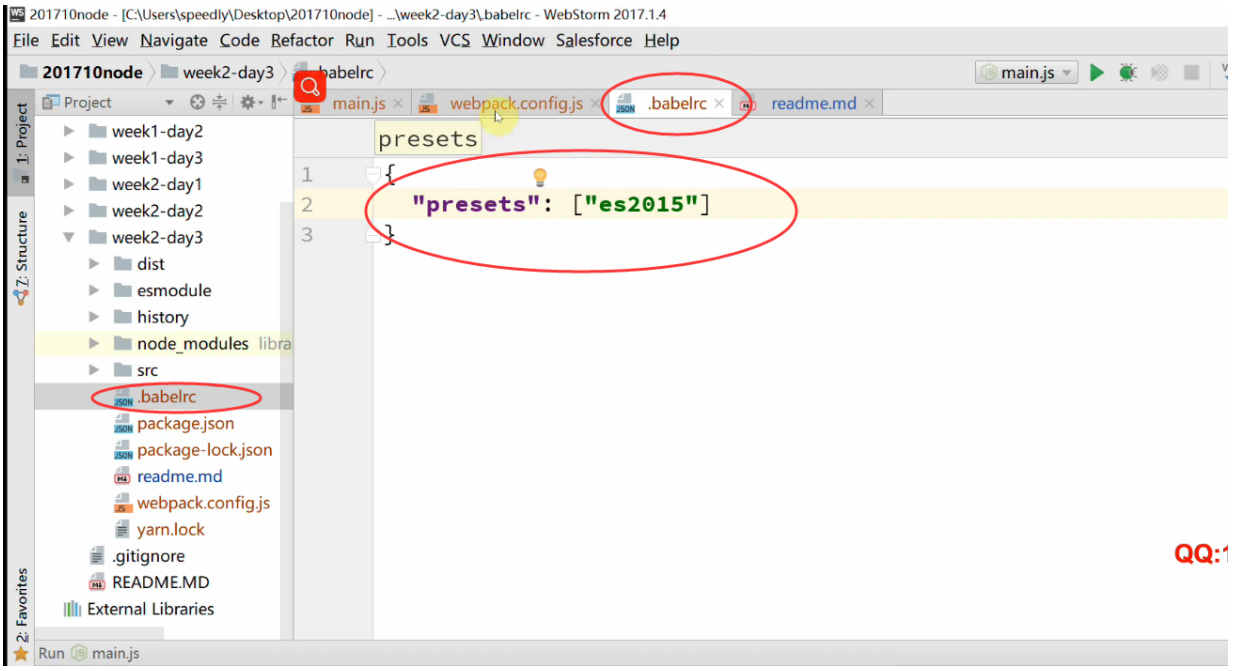
```
npm install babel-core babel-loader --save-dev
```

`npm run build`

babel-preset-es2015

- 让翻译官拥有解析es6语法的功能

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



- main.js

```
let a = b => c=> d=> b+c+d;
let obj ={school:'zfpx'};
let obj1 = {age:8};
```

- webpack.config.js内容:ES6转ES5

```
let path = require('path'); // 专门处理路径用的,以当前路径解析出一个绝对路径
console.log(path.resolve('./dist'));
module.exports = {
    entry:'./src/main.js', // 打包的入口文件，webpack会自动查找相关的依赖进行打包
    output:{
        filename:'build.js',// 打包后的名字
        path:path.resolve('./dist') //必须是一个绝对路径
    },
    // 模块的解析规则
    // - js 匹配所有的js 用babel-loader转译 排除掉node_modules
    module:{
        rules:[
            {test:/\.js$/,use:'babel-loader',exclude:/node_modules/}
        ]
```

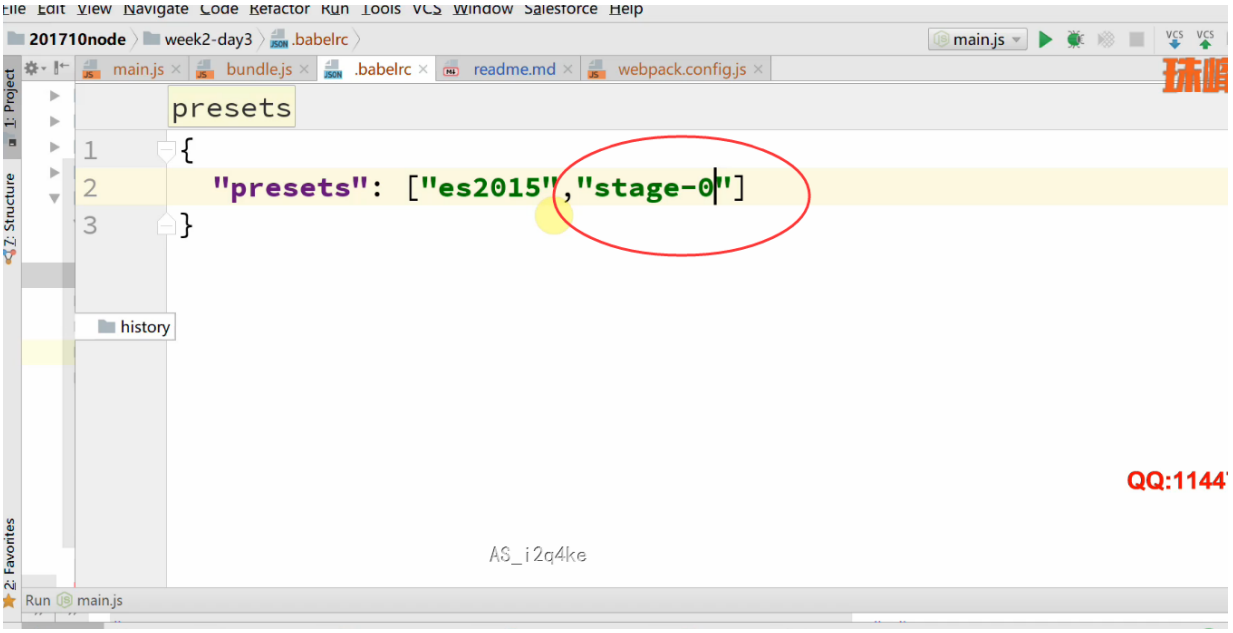
```
    },
```

`npm run build`

babel-preset-stage-0

- 解析es7语法的

```
npm install babel-preset-stage-0
--save-dev
```




```
let newObj = {...obj,...obj1}; //
es7语法
console.log(newObj);
```

`npm run build`

解析样式

- css-loader将css解析成模块，将解析的内容插入到style标签内(style-loader)

```
html {
  body:after{
    content:'我爱你';
    color:red
  }
}
```

```
npm install css-loader style-load
er --save-dev
```

```
import './index.css';  //引入css
```

```
        rules:[
            {test:/\.js$/,use:'ba
bel-loader',exclude:/node_module
s/}

            // use时从右往左写
            {test:/\.css$/,use:
['style-loader','css-loader']},
        ]
```

less,sass,stylus(预处理语言)

- less-loader less
- sass-loader
- stylus-loader

```
html {
  body:after{
    content:'我爱你';
    color:red
  }
}
```

```
npm install less less-loader --save-dev
```

```
import './style.less';  LESS
```

```
....
JS
rules:[
                // use时从右往左写
                {test:/\.css$/,use:['style-loader','css-loader']},

                {test:/\.less$/,use:['style-loader','css-loader','less-loader']},
            ]
```

解析图片

- file-loader url-loader(是依赖于file-loader的)

```
npm install file-loader url-loade
r --save-dev
```

```
...
JS
CSS
 rules:[
            {test:/\.less$/,use:
['style-loader','css-loader','les
s-loader']},

            // 转化base64只在8192
字节一下转化。其他情况下输出图片
            {test:/\.(jpg|png|gi
f)$/,use:'url-loader?limit=819
2'},
            {test:/\.(eot|svg|wof
f|woff2|wtf)$/,use:'url-loader'},

]
限制:?limit=8192 转化base64只在8192
字节一下转化。其他情况下输出图片
```

小知识:

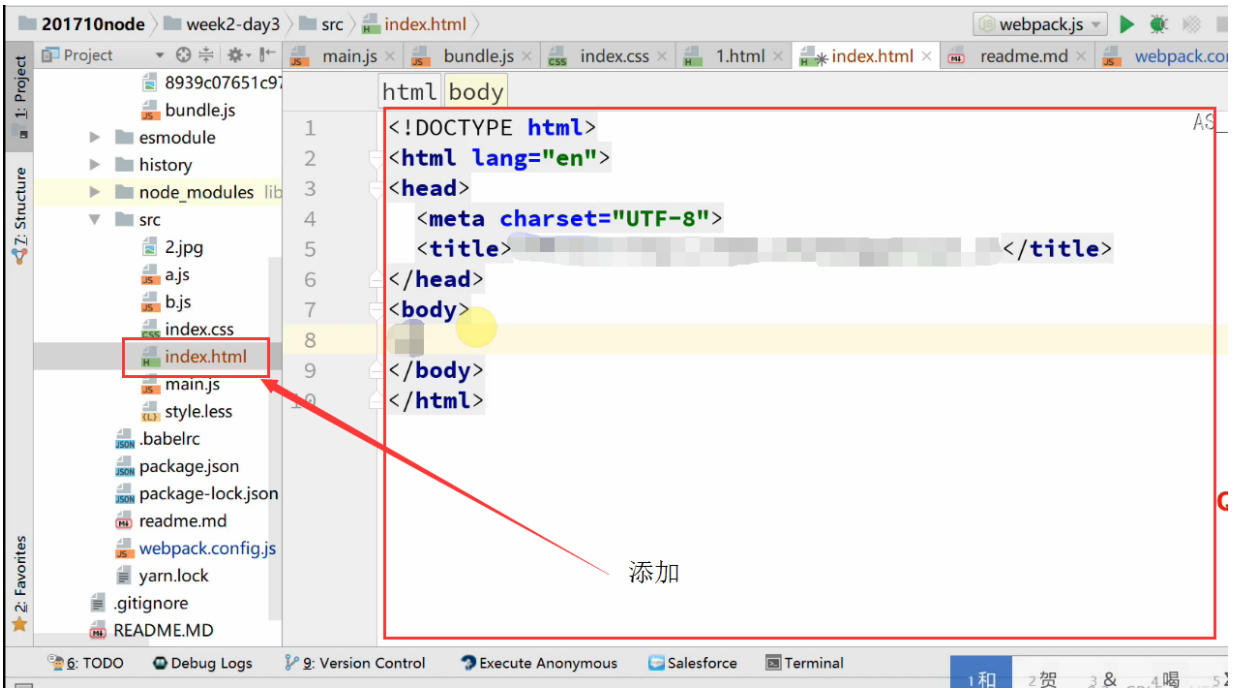
- 在main.js中引入图片需要import,或者写一个线上路径

```
import page from './2.jpg';
console.log(page); // page就是打包后图片的路径
let img = new Image();
img.src = page; // 写了一个字符串 webpack不会进行查找
document.body.appendChild(img);
```

需要解析html插件

- 插件的作用是以我们自己的html为模板将打包后的结果，`自动引入到html中产出到dist目录下`
- NPM：出现权限问题，npm执行以管理员身份运行

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



```
// webpack必须采用commonjs写法
let path = require('path'); // 专门处理路径用的,以当前路径解析出一个绝对路径
console.log(path.resolve('./dist'));
let HtmlWebpackPlugin = require('html-webpack-plugin');
module.exports = {
    entry:'./src/main.js', // 打包的入口文件，webpack会自动查找相关的依赖进行打包
    output:{
        filename:'build.js',// 打包后的名字
        path:path.resolve('./dist') //必须是一个绝对路径
    },
    // 模块的解析规则
    // - js 匹配所有的js 用babel-loader转译 排除掉node_modules
    module:{
        rules:[
            {test:/\.js$/,use:'ba
```

```
bel-loader',exclude:/node_module
s/},
                // use时从右往左写
                {test:/\.css$/,use:
['style-loader','css-loader']},
                {test:/\.less$/,use:
['style-loader','css-loader','les
s-loader']},
                // 转化base64只在8192
字节一下转化。其他情况下输出图片
                {test:/\.(jpg|png|gi
f)$/,use:'url-loader?limit=819
2'},
                {test:/\.(eot|svg|wof
f|woff2|wtf)$/,use:'url-loader'},
            ]
        },
        plugins:[
            new HtmlWebpackPlugin({
// 自动插入到dist目录中
                template:'./src/inde
x.html', // 使用的模板
                // filename:'login.htm
l' // 产出的文件名字
            })
```

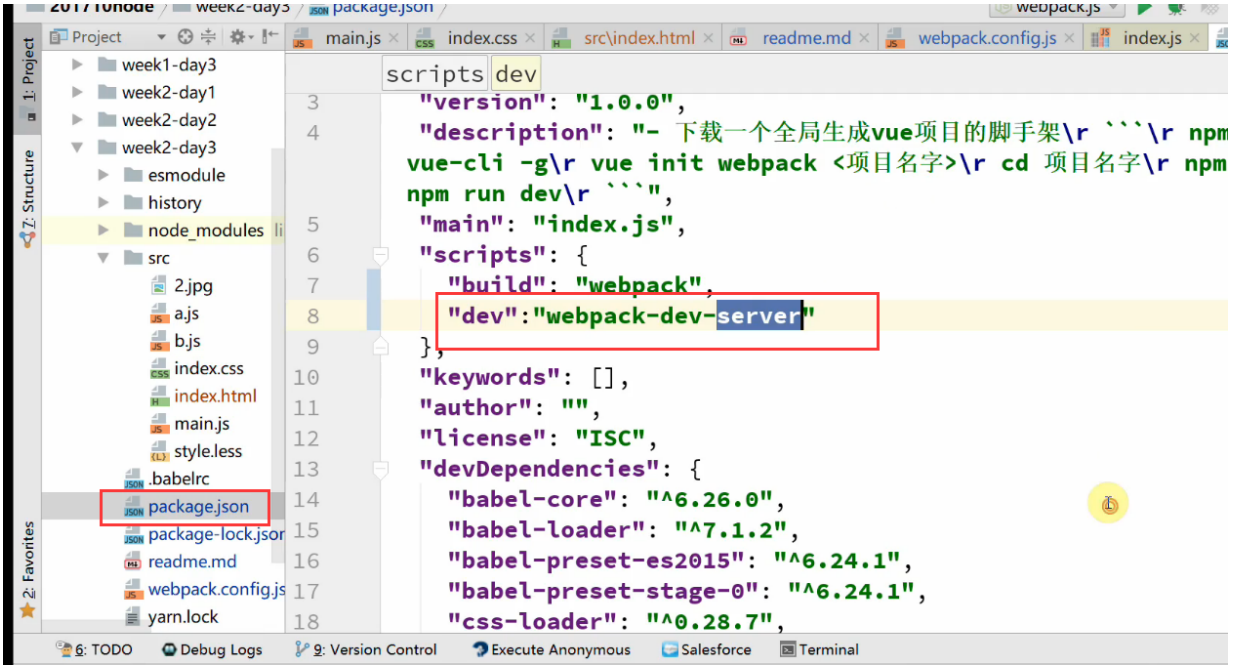
```
        ]
    };
```

webpack-dev-server

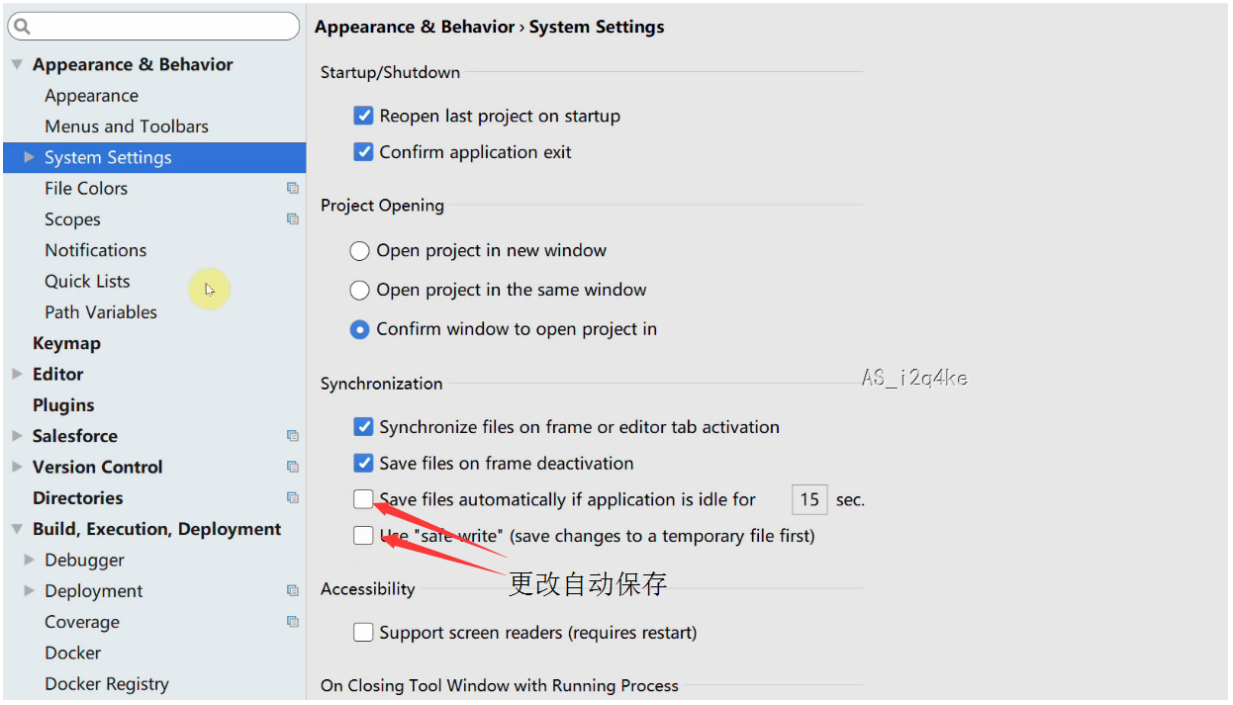
- 这里面内置了服务，可以帮我们启动一个端口号，当代码更新时，可以`自动打包`（内存中打包），代码有变化就重新执行

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

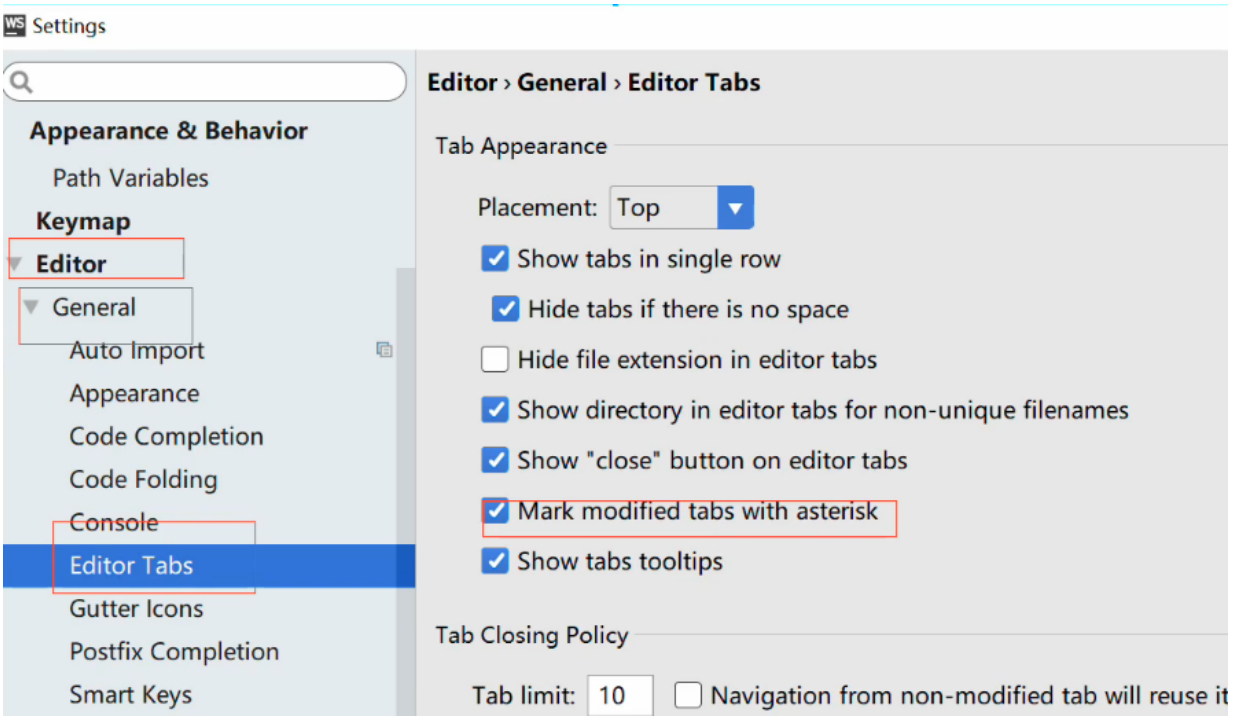
添加:



```
npm run dev
```


Appearance & Behavior › System Settings
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
File Colors
Scopes
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Salesforce
Version Control
Directories
Build, Execution, Deployment
Debugger
Deployment
Coverage
Docker
Docker Registry
Startup/Shutdown
Reopen last project on startup
Confirm application exit
Project Opening
Open project in new window
Open project in the same window
Confirm window to open project in
Synchronization
Synchronize files on frame or editor tab activation
Save files on frame deactivation
Save files automatically if application is idle for 15 sec.
Use "safe write" (save changes to a temporary file first)
更改自动保存
Accessibility
Support screen readers (requires restart)
On Closing Tool Window with Running Process


index.css


Settings
Appearance & Behavior
Path Variables
Keymap
Editor
General
Auto Import
Appearance
Code Completion
Code Folding
Console
Editor Tabs
Gutter Icons
Postfix Completion
Smart Keys
Editor › General › Editor Tabs
Tab Appearance
Placement: Top
Show tabs in single row
Hide tabs if there is no space
Hide file extension in editor tabs
Show directory in editor tabs for non-unique filenames
Show "close" button on editor tabs
Mark modified tabs with asterisk
Show tabs tooltips
Tab Closing Policy
Tab limit: 10
Navigation from non-modified tab will reuse it