预处理器

一切皆模块

对于Webpack来说所有静态资源都是模块
`import './style.css'`

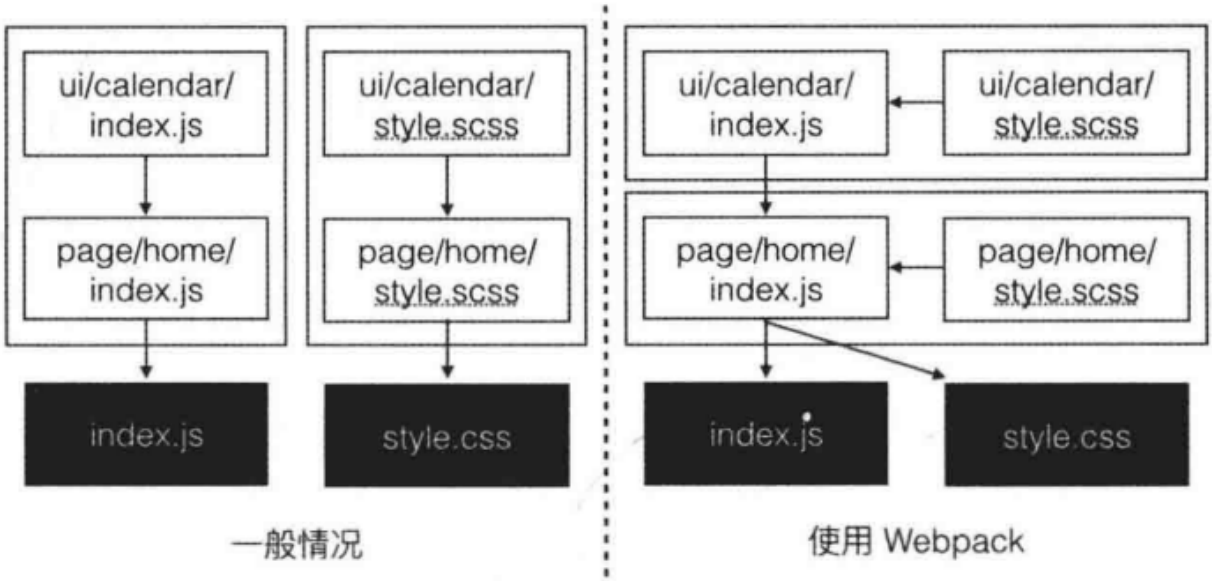

图 4-1　使用 Webpack 前后依赖关系图对比

loader

本质上每个loader是个函数
webpack4之前，函数的输入输出必需为字符串
webpack4之后，支持 抽象语法树 AST的传递

loader源码结构

```
module.exports = function loader (content, map, meta) {
    var callback = this.async();
    var result = handler(content, map, meta);
    callback(
        null,            // error
        result.content, // 转换后的内容
        result.map,      // 转换后的 source-map
        result.meta,     // 转换后的 AST
    );
};
```

loader配置

loader，装载器，webpack中其实就是 预处理器
webpack本身只认识JS
其他类型资源需要预先定义一个或者多个loader进行转译
输出为webpack能够接收的形式再进行

loader做的是预处理的工作

如果没有loader 打包遇到导入其他类型文件会报错
You may need an appropriate loader to handle this file type

loader都是第三方npm，webpack本省并不包含任何loader

```
module.exports = {
    module:{
        rules:[
            {
                test: /\.css$/,
                use:['css-loader']
            }
        ]
    }
}
```

- test 接收正则表达式或者一个元素为正则表达式的数组，只有匹配才会使用这条规则
- use 接收数组/字符串，包含该规则所使用的loader

链式loader

处理某一类资源可能需要用到多个loader
例:处理css类型
css-loader处理css语法
style-loader包装成style标签

`use:['style-loader','css-loader']`

最后生效的写在前面

loader options

预处理器的配置项，通过options传入

更多配置

exclude include

排除或包含指定目录下的模块，接收正则或者字符串(绝对路径)
一般都会排除 node_modules
一般node_modules都是已经编译好了的文件，无需再做额外处理

exclude优先级比include更高

resource issuer

被加载模块 esource
加载这 issuer

可以更加精确地确定模块规则作用范围

```
rules: [
    {
        test: /\.css$/,
        use: ['style-loader', 'css-loader'],
        exclude: /node_modules/,
        issuer: {
            test: /\.js$/,
            include: /src/pages/,
        },
    }
],
```

enforce

指定一个loader类，只接收 pre 或 post 两种字符串类型的值

loader执行顺序 pre inline normal post
直接定义的loader都属于normal
inline官方不推荐
pre post需要enforce指定

```
rules:[{
    test: /\.js$/,
```

```
    enfore:'pre',
    use:'enlint-loader'
}]
```

pre表示所有loader之前执行，可以确保代码没被其他loader更改
post 相反

其实没必要用enfore，我们只要保证loader顺序正确，即可

常用loader

babel-loader

处理ES6+,并编译为ES5
可以让我们使用最新的语言特性，同时不必关心不同平台的兼容问题

- babel-loader Babel与Webpack协同工作的模块
- @babel/core Babel编译器核心模块
- @babel/preset-evn Babel官方推荐预置器，可以根据用户设置的目标环境自动添加所需的插件和补丁来编译ES6+代码

注意:

- babel-loader 规则对所有js文件设置，所以需要排除node_modules
- babel-loader，有cacheDirectory配置项，启动缓存，重复打包时防止二次编译。
  可以自己设置路径，也可以true，默认路径node_modules/.cache/babel-loader
- @babel/preset-env 会把ES6 Module转成CommonJS
  会导致tree-shaking特性失效
  可以配置@babel/preset-env modules为false，让Webpack自己处理ES6 Module

配置文件 .babelrc

ts-loader

连接Webpack与Typescript模块
TS本身配置项不在ts-loader中，需要放在tsconfig.json中

html-loader

将HTML文件转为字符串并格式化，html片段可以通过js加载

handlebars-loader

处理handlebars

file-loader

用于文件类型（包括png,jpg等图片资源）的资源
返回publicPath

url-loader

类似file-loader
可以设置文件大小的阈值
大于 返回publicPath
小于 返回 base64

vue-loader

处理vue文件

自定义loader

loader初始化

npm/yarn 软链功能 本地调试
在项目中npm install 相对路径，则会在项目的node_modules中创建一个实际指向 之前的安装路径的一个软连接
这样就可以直接修改 安装文件，而不需要重复安装了

启用缓存

Webpack中 this.cacheable控制缓存，所以loader中可以加入缓存

```
if (this.cacheable) {
    this.cacheable();
}
```

options配置项

因为配置项实在webpack中配置，所以肯定需要用到第三方库来获取webpack中的option配置项。
loader-utils
`var options = loaderUtils.getOptions(this) || {}`

source-map

可以在控制台看源码
没有对source-map处理，就无法生成正确的map文件。控制台看到的可能就是乱码

小结

loader 预处理器
loader配置
常用loader