

在 webpack 中，默认只能处理 一部分 ES6 的新语法，一些更高级的ES6语法或者 ES7 语法，webpack 是处理不了的；这时候，就需要 借助于第三方的 loader，

来帮助webpack 处理这些高级的语法，当第三方loader 把 高级语法转为 低级的语法之后，会把结果交给 webpack 去打包到 bundle.js 中

通过 Babel ，可以帮我们将 高级的语法转换为 低级的语法

1. 在 webpack 中，可以运行如下两套 命令，安装两套包，去安装 Babel 相关的loader 功能：

1.1 第一套包： `cnpm i babel-core babel-loader babel-plugin-transform-runtime -D`

1.2 第二套包： `cnpm i babel-preset-env babel-preset-stage-0 -D`

2. 打开 webpack 的配置文件，在 module 节点下的 rules 数组中，添加一个 新的 匹配规则：

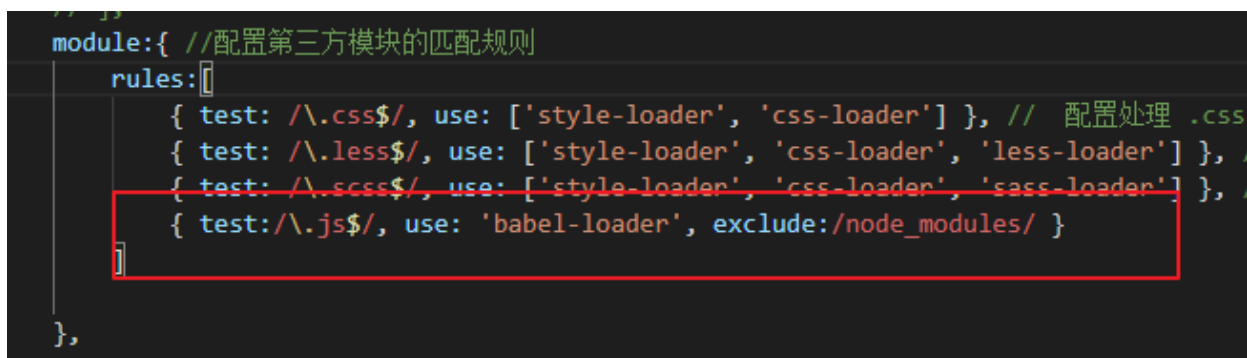
2.1 `{ test:/\.js$/, use: 'babel-loader', exclude:/node_modules/ }`

2.2 注意： 在配置 babel 的 loader规则的时候，必须 把 node_modules 目录，通过 exclude 选项排除掉：原因有两：

2.2.1 如果 不排除 node_modules， 则Babel 会把 node_modules 中所有的 第三方 JS 文件，都打包编译，这样，会非常消耗CPU，同时，打包速度非常慢；

2.2.2 哪怕，最终，Babel 把 所有 node_modules 中的JS转换完毕了，但是，项目也无法正常运行！

3. 在项目的 根目录中，新建一个 叫做 .babelrc 的Babel 配置文件，这个配置文件，属于JSON格式，所以，在写 .babelrc 配置的时候，必须符合JSON语法规则： 不能写注释，字符串必须用双引号



```

module:{ //配置第三方模块的匹配规则
  rules:[
    { test: /\.css$/, use: ['style-loader', 'css-loader'] }, // 配置处理 .css
    { test: /\.less$/, use: ['style-loader', 'css-loader', 'less-loader'] },
    { test: /\.scss$/, use: ['style-loader', 'css-loader', 'sass-loader'] },
    { test: /\.js$/, use: 'babel-loader', exclude: /node_modules/ }
  ]
},

```

3.1 在 .babelrc 写如下的配置： 大家可以把 preset 翻译成 【语法】 的意思

```

{
  "presets": ["env", "stage-0"],
  "plugins": ["transform-runtime"]
}

```



```
JS main.js 6 .babelrc webp
6 .babelrc ▸ ...
1  {
2    "presets": [
3      "env",
4      "stage-0"
5    ],
6    "plugins": [
7      "transform-runtime"
8    ]
9  }
```

4. 了解： 目前，我们安装的 `babel-preset-env`，是比较新的ES语法， 之前， 我们安装的是 `babel-preset-es2015`，现在，出了一个更新的 语法插件，叫做 `babel-preset-env`，它包含了 所有的 和 `es***`相关的语法