# 在react使用sass的情况下使用css modules

2018年09月07日 17:41:30 [sinat\_33895623](https://me.csdn.net/sinat_33895623" \t "https://blog.csdn.net/sinat_33895623/article/details/_blank) 阅读数：253更多

个人分类： [前端](https://blog.csdn.net/sinat_33895623/article/category/8008221" \t "https://blog.csdn.net/sinat_33895623/article/details/_blank)[css modules](https://blog.csdn.net/sinat_33895623/article/category/8023740" \t "https://blog.csdn.net/sinat_33895623/article/details/_blank)[sass](https://blog.csdn.net/sinat_33895623/article/category/8023741" \t "https://blog.csdn.net/sinat_33895623/article/details/_blank)

### **1. 暴露相关配置文件**

yarn eject

### **2. 添加依赖**

yarn add css-hot-loader node-sass sass-loader --dev

### **3. 修改config/webpack.config.dev.js文件**

*//声明ExtractTextPlugin插件*const ExtractTextPlugin = require("extract-text-webpack-plugin");

...*//在exclude中添加scss*

{

excluedes:[

/\.html$/,

/\.(js|jsx)$/,

/\.css$/,

/\.json$/,

/\.bmp$/,

/\.gif$/,

/\.jpe?g$/,

/\.png$/,

/\.scss$/ *//+++*

]

...

}

...*//修改css的loader*

{

test: /\.css$/,

use: ["css-hot-loader"].concat(

ExtractTextPlugin.extract({

fallback: "style-loader",

use: [

{

loader: "css-loader",

options: {

modules: true,

localIdentName: "[name]\_\_[local]\_\_\_[hash:base64:5]"

}

},

"postcss-loader"

]

})

)

}

...*//添加scss的loader*

{

test: /\.scss$/,

use: ["css-hot-loader"].concat(

ExtractTextPlugin.extract({

fallback: "style-loader",

use: [

{

loader: "css-loader",

options: {

modules: true,

sourceMap: true,

importLoaders: 2,

localIdentName: "[name]\_\_[local]\_\_\_[hash:base64:5]"

}

},

"sass-loader"

]

})

)

}

...*//添加ExtractTextPlugin*

{

plugins :[

...

new ExtractTextPlugin({

filename: "styles.css",

allChunks: true,

disable: process.env.NODE\_ENV !== "production"

})

...

]

}

### 

### **4. 效果**

#### **源代码**

App.scss

.App {

text-align: center;

background: HotPink;

.logo {

animation: App-logo-spin infinite 20s linear;

height: 80px;

}

.header {

background-color: #222;

height: 150px;

padding: 20px;

color: white;

}

.title {

font-size: 1.5em;

}

.intro {

font-size: large;

.code {

font-size: 40px;

}

}

@keyframes App-logo-spin {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

}

App.jsx

import React, { Component } from "react";

import logo from "./logo.svg";

import styles from "./App.scss";

class App extends Component {

render() {

return (

<div className={styles.App}>

<header className={styles.header}>

<img src={logo} className={styles.logo} alt="logo" />

<h1 className={styles.title}>Welcome to React</h1>

</header>

<p className={styles.intro}>

To get started, edit <code className={styles.code}>src/App.js</code>{" "}

and save to reload.

</p>

</div>

);

}

}

export default App;

#### **webpack转码之后的效果**

html

<div class="App\_\_App\_\_\_wmyqV">

<header class="App\_\_header\_\_\_1wrze">

<img src="/static/media/logo.5d5d9eef.svg" class="App\_\_logo\_\_\_2xfX3" alt="logo">

<h1 class="App\_\_title\_\_\_3vGKJ">Welcome to React</h1>

</header>

<p class="App\_\_intro\_\_\_1ICPI">To get started, edit <code class="App\_\_code\_\_\_3Pk1a">src/App.js</code> and save to reload.</p></div>

css

.App\_\_App\_\_\_wmyqV {

text-align: center;

background: HotPink;}

.App\_\_App\_\_\_wmyqV .App\_\_logo\_\_\_2xfX3 {

animation: App\_\_App-logo-spin\_\_\_35CiA infinite 20s linear;

height: 80px;}

.App\_\_App\_\_\_wmyqV .App\_\_header\_\_\_1wrze {

background-color: #222;

height: 150px;

padding: 20px;

color: white;}

.App\_\_App\_\_\_wmyqV .App\_\_title\_\_\_3vGKJ {

font-size: 1.5em;}

.App\_\_App\_\_\_wmyqV .App\_\_intro\_\_\_1ICPI {

font-size: large;}

.App\_\_App\_\_\_wmyqV .App\_\_intro\_\_\_1ICPI .App\_\_code\_\_\_3Pk1a {

font-size: 40px;}

@keyframes App\_\_App-logo-spin\_\_\_35CiA {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

## **注**

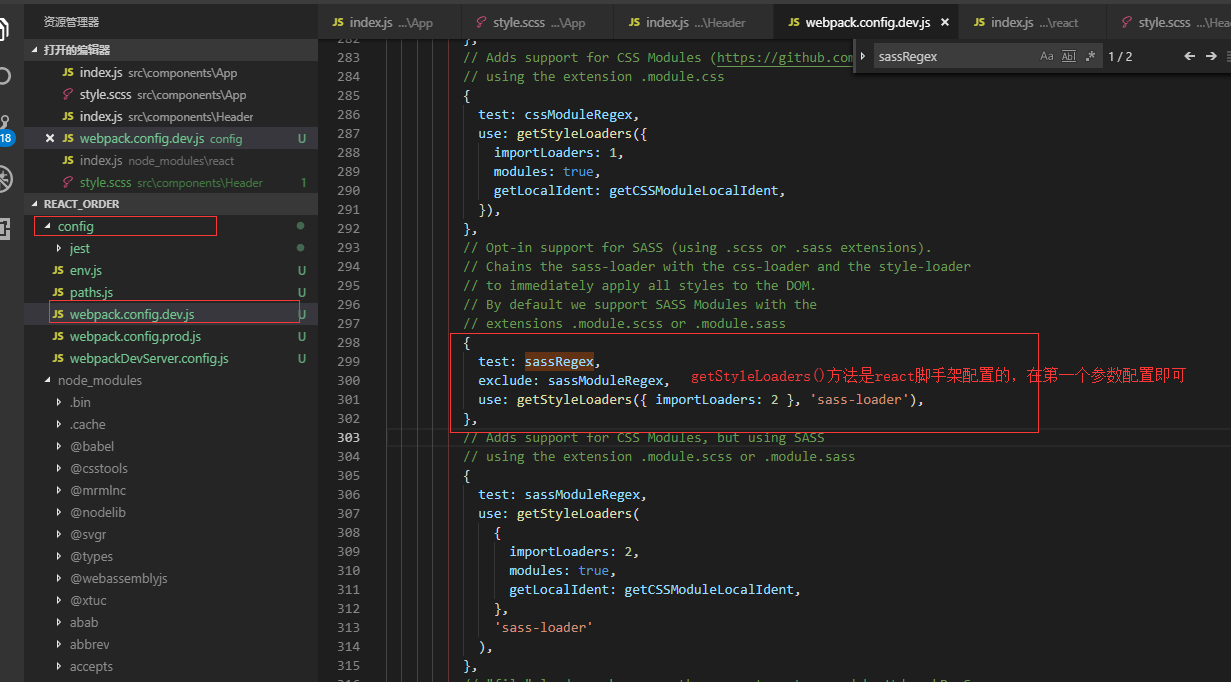
* 因为css module推荐使用composes来做嵌套相关的工作，composes最终结果是影响了js，在对应的className位置添加了多个class, sass则最终将自己的嵌套转化为多选择器的css，这点是不同的
* 正是因为上一点的不同，composes不能用在多选择器的作用域下，只能在最外层的单选择器的作用域下使用composes

**PS:个人配置过程**

注：css module配置与使用都有点麻烦，大项目样式模块化可以减少样式冲突，中小项目用sass就行。

运行【npm run eject】将webpack反编译出来。

然后在【config 》 webpack.config.dev.js】修改配置，找到sassRegex



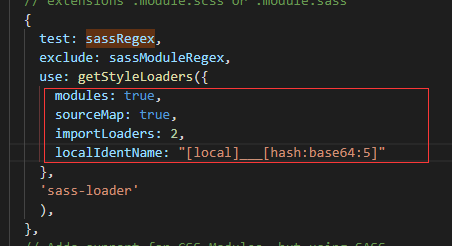
modules: true,

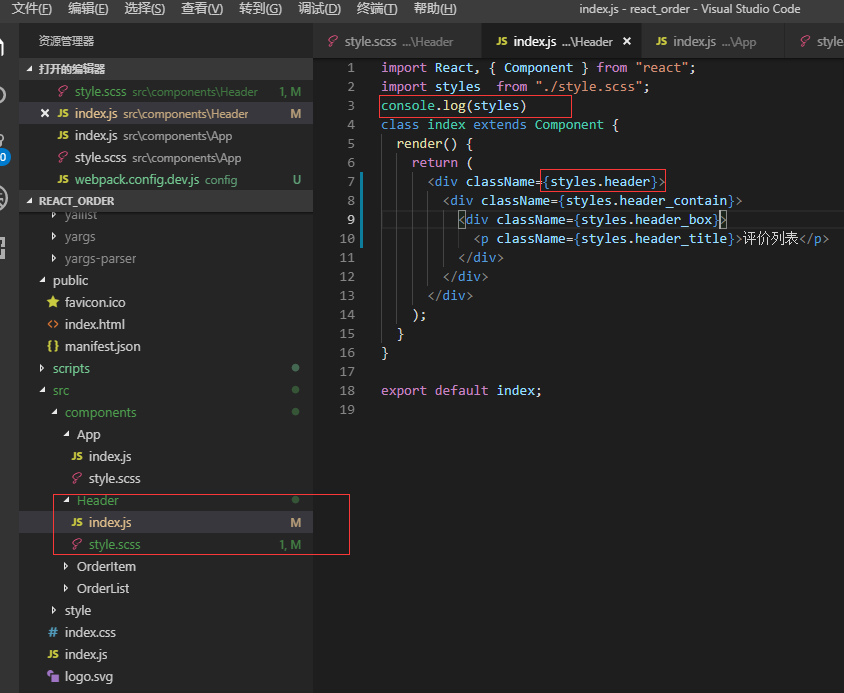
sourceMap: true,

importLoaders: 2,

localIdentName: "[local]\_\_\_[hash:base64:5]"

注：【localIdentName:"[path]\_\_\_[name]\_\_[local]\_\_\_[hash:base64:5]"】全部的配置参数，可根据实际需求更改。





Css命名：不能使用横杠，对象变量有横杠获取不到值的，必须使用【\_下划线】，要符合js的变量规范。

{

header\_contain: "header\_contain\_\_\_3mcTv",

header: "header\_\_\_3loT1",

header\_box: "header\_box\_\_\_1Ohw6",

header\_title: "header\_title\_\_\_1sO9s"

}

