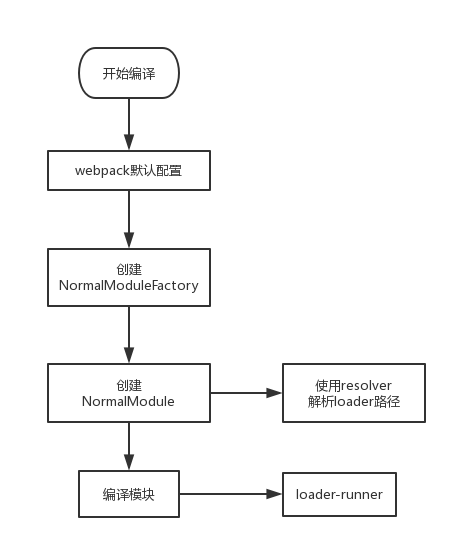
## **1. loader运行的总体流程**

* Compiler.js中会为将用户配置与默认配置合并，其中就包括了loader部分
* webpack就会根据配置创建NormalModuleFactory,它可以用来创建NormalModule
* 在工厂创建NormalModule实例之前还要通过loader的resolver来解析loader路径
* 在NormalModule实例创建之后，则会通过其build方法来进行模块的构建。构建模块的第一步就是使用loader来加载并处理模块内容。而loader-runner这个库就是webpack中loader的运行器
* 最后，将loader处理完的模块内容输出，进入后续的编译流程



## **2.babel-loader**

* [babel-loader](https://github.com/babel/babel-loader/blob/master/src/index.js)
* [@babel/core](https://babeljs.io/docs/en/next/babel-core.html)
* [babel-plugin-transform-react-jsx](https://babeljs.io/docs/en/babel-plugin-transform-react-jsx/)

| **属性** | **值** |
| --- | --- |
| this.request | /loaders/babel-loader.js!/src/index.js |
| this.userRequest | /src/index.js |
| this.rawRequest | ./src/index.js |
| this.resourcePath | /src/index.js |

$ cnpm i @babel/preset-env @babel/core -D

**const** babel = require("@babel/core");**function** **loader**(source,inputSourceMap) {

//C:\webpack-analysis2\loaders\babel-loader.js!C:\webpack-analysis2\src\index.js

**const** options = {

presets: ['@babel/preset-env'],

inputSourceMap:inputSourceMap,

sourceMaps: true,//ourceMaps: true 是告诉 babel 要生成 sourcemap

filename:**this**.request.split('!')[1].split('/').pop()

}

//在webpack.config.js中 增加devtool: 'eval-source-map'

**let** {code,map,ast}=babel.transform(source,options);

**return** **this**.callback(null,code,map,ast);

}module.exports = loader;

resolveLoader: {

alias: {//可以配置别名

"babel-loader": resolve('./build/babel-loader.js')

},//也可以配置loaders加载目录

modules: [path.resolve('./loaders'), 'node\_modules']

},

{

test: /\.js$/,

use:['babel-loader']

}

## **3.pitch**

* 比如a!b!c!module, 正常调用顺序应该是c、b、a，但是真正调用顺序是 a(pitch)、b(pitch)、c(pitch)、c、b、a,如果其中任何一个pitching loader返回了值就相当于在它以及它右边的loader已经执行完毕
* 比如如果b返回了字符串"result b", 接下来只有a会被系统执行，且a的loader收到的参数是result b
* loader根据返回值可以分为两种，一种是返回js代码（一个module的代码，含有类似module.export语句）的loader，还有不能作为最左边loader的其他loader
* 有时候我们想把两个第一种loader chain起来，比如style-loader!css-loader! 问题是css-loader的返回值是一串js代码，如果按正常方式写style-loader的参数就是一串代码字符串
* 为了解决这种问题，我们需要在style-loader里执行require(css-loader!resources)

pitch与loader本身方法的执行顺序图

|- a-loader `pitch`

|- b-loader `pitch`

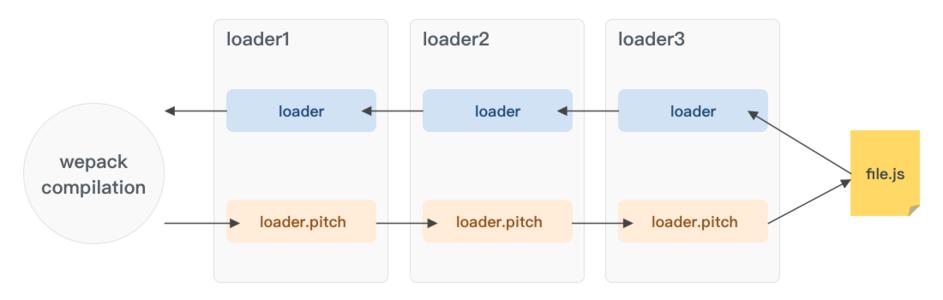
|- c-loader `pitch`

|- requested module is picked up **as** a dependency

|- c-loader normal execution

|- b-loader normal execution

|- a-loader normal execution



### **3.1 loaders\loader1.js**

loaders\loader1.js

**function** **loader**(source) {

console.log('loader1',**this**.data);

**return** source+"//loader1";

}

loader.pitch = **function** (remainingRequest,previousRequest,data) {

data.name = 'pitch1';

console.log('pitch1');

}module.exports = loader;

### **3.2 loaders\loader2.js**

loaders\loader2.js

**function** **loader**(source) {

console.log('loader2');

**return** source+"//loader2";

}

loader.pitch = **function** (remainingRequest,previousRequest,data) {

console.log('remainingRequest=',remainingRequest);

console.log('previousRequest=',previousRequest);

console.log('pitch2');

//return 'console.log("pitch2")';

}module.exports = loader;

### **3.3 loaders\loader3.js**

loaders\loader3.js

**function** **loader**(source) {

console.log('loader3');

**return** source+"//loader3";

}

loader.pitch = **function** () {

console.log('pitch3');

}module.exports = loader;

### **3.4 webpack.config.js**

{

test: /\.js$/,

use: ['loader1', 'loader2', 'loader3']

}

## **4.loader-runner**

### **4.1 loader类型**

* [loader的叠加顺序](https://github.com/webpack/webpack/blob/v4.39.3/lib/NormalModuleFactory.js" \l "L159-L339) = post(后置)+inline(内联)+normal(正常)+pre(前置)

### **4.2 特殊配置**

* [loaders/#configuration](https://webpack.js.org/concepts/loaders/" \l "configuration)

| **符号** | **变量** | **含义** |
| --- | --- | --- |
| -! | noPreAutoLoaders | 不要前置和普通loader | Prefixing with -! will disable all configured preLoaders and loaders but not postLoaders |
| ! | noAutoLoaders | 不要普通loader | Prefixing with ! will disable all configured normal loaders |
| !! | noPrePostAutoLoaders | 不要前后置和普通loader,只要内联loader | Prefixing with !! will disable all configured loaders (preLoaders, loaders, postLoaders) |

### **4.2 查找规则执行**

**let** path = require("path");**let** nodeModules = path.resolve(\_\_dirname, "node\_modules");**let** request = "-!inline-loader1!inline-loader2!./styles.css";//首先解析出所需要的 loader，这种 loader 为内联的 loader**let** inlineLoaders = request

.replace(/^-?!+/, "")

.replace(/!!+/g, "!")

.split("!");**let** resource = inlineLoaders.pop();//// 获取资源的路径**let** resolveLoader = loader => path.resolve(nodeModules, loader);//从相对路径变成绝对路径

inlineLoaders = inlineLoaders.map(resolveLoader);**let** rules = [

{

enforce: "pre",

test: /\.css?$/,

use: ["pre-loader1", "pre-loader2"]

},

{

test: /\.css?$/,

use: ["normal-loader1", "normal-loader2"]

},

{

enforce: "post",

test: /\.css?$/,

use: ["post-loader1", "post-loader2"]

}

];**let** preLoaders = [];**let** postLoaders = [];**let** normalLoaders = [];**for**(**let** i=0;i<rules.length;i++){

**let** rule = rules[i];

**if**(rule.test.test(resource)){

**if**(rule.enforce=='pre'){

preLoaders.push(...rule.use);

}**else** **if**(rule.enforce=='post'){

postLoaders.push(...rule.use);

}**else**{

normalLoaders.push(...rule.use);

}

}

}

preLoaders = preLoaders.map(resolveLoader);

postLoaders= postLoaders.map(resolveLoader);

normalLoaders = normalLoaders.map(resolveLoader);

**let** loaders = [];//noPrePostAutoLoaders 忽略所有的 preLoader / normalLoader / postLoader**if**(request.startsWith('!!')){

loaders = inlineLoaders;//只保留inline//noPreAutoLoaders 是否忽略 preLoader 以及 normalLoader

}**else** **if**(request.startsWith('-!')){

loaders = [...postLoaders,...inlineLoaders];//只保留post和inline//是否忽略 normalLoader

}**else** **if**(request.startsWith('!')){

loaders = [...postLoaders,...inlineLoaders,...preLoaders];//保留post inline pre

}**else**{

loaders = [...postLoaders,...inlineLoaders,...normalLoaders,...preLoaders];

}console.log(loaders);

### **4.4 run-loader**

* [LoaderRunner](https://github.com/webpack/loader-runner/blob/v2.4.0/lib/LoaderRunner.js)
* [NormalModuleFactory-noPreAutoLoaders](https://github.com/webpack/webpack/blob/v4.39.3/lib/NormalModuleFactory.js" \l "L180)
* [NormalModule-runLoaders](https://github.com/webpack/webpack/blob/v4.39.3/lib/NormalModule.js" \l "L292)

**let** readFile = require("fs");**let** path = require("path");**function** **createLoaderObject**(loader) {

**let** obj = { data: {} };

obj.request = loader;

obj.normal = require(loader);

obj.pitch = obj.normal.pitch;

**return** obj;

}**function** **runLoaders**(options, callback) {

**let** loaderContext = {};

**let** resource = options.resource;

**let** loaders = options.loaders;

loaders = loaders.map(createLoaderObject);

loaderContext.loaderIndex = 0;

loaderContext.readResource = readFile;

loaderContext.resource = resource;

loaderContext.loaders = loaders;

**let** isSync = true;

**var** innerCallback = (loaderContext.callback = **function**(err, args) {

loaderContext.loaderIndex--;

iterateNormalLoaders(loaderContext, args, callback);

});

loaderContext.async = **function** **async**() {

isSync = false;

**return** innerCallback;

};

Object.defineProperty(loaderContext, "request", {

get: **function**() {

**return** loaderContext.loaders

.map(**function**(o) {

**return** o.request;

})

.concat(loaderContext.resource)

.join("!");

}

});

Object.defineProperty(loaderContext, "remainingRequest", {

get: **function**() {

**return** loaderContext.loaders

.slice(loaderContext.loaderIndex + 1)

.map(**function**(o) {

**return** o.request;

})

.concat(loaderContext.resource || "")

.join("!");

}

});

Object.defineProperty(loaderContext, "currentRequest", {

enumerable: true,

get: **function**() {

**return** loaderContext.loaders

.slice(loaderContext.loaderIndex)

.map(**function**(o) {

**return** o.request;

})

.concat(loaderContext.resource || "")

.join("!");

}

});

Object.defineProperty(loaderContext, "previousRequest", {

get: **function**() {

**return** loaderContext.loaders

.slice(0, loaderContext.loaderIndex)

.map(**function**(o) {

**return** o.request;

})

.join("!");

}

});

Object.defineProperty(loaderContext, "data", {

get: **function**() {

**return** loaderContext.loaders[loaderContext.loaderIndex].data;

}

});

iteratePitchingLoaders(loaderContext, callback);

**function** **iteratePitchingLoaders**(loaderContext, callback) {

**if** (loaderContext.loaderIndex >= loaderContext.loaders.length) {

loaderContext.loaderIndex--;

**return** processResource(loaderContext, callback);

}

**let** currentLoaderObject = loaderContext.loaders[loaderContext.loaderIndex];

**let** fn = currentLoaderObject.pitch;

**if** (!fn) **return** iteratePitchingLoaders(options, loaderContext, callback);

**let** args = fn.apply(loaderContext, [

loaderContext.remainingRequest,

loaderContext.previousRequest,

currentLoaderObject.data

]);

**if** (args) {

loaderContext.loaderIndex--;

**return** iterateNormalLoaders(loaderContext, args, callback);

} **else** {

loaderContext.loaderIndex++;

iteratePitchingLoaders(loaderContext, callback);

}

**function** **processResource**(loaderContext, callback) {

**let** buffer = loaderContext.readResource.readFileSync(

loaderContext.resource,

"utf8"

);

iterateNormalLoaders(loaderContext, buffer, callback);

}

}

**function** **iterateNormalLoaders**(loaderContext, args, callback) {

**if** (loaderContext.loaderIndex < 0) **return** callback(null, args);

**var** currentLoaderObject = loaderContext.loaders[loaderContext.loaderIndex];

**var** fn = currentLoaderObject.normal;

**if** (!fn) {

loaderContext.loaderIndex--;

**return** iterateNormalLoaders(loaderContext, args, callback);

}

args = fn.apply(loaderContext, [args]);

**if** (isSync) {

loaderContext.loaderIndex--;

iterateNormalLoaders(loaderContext, args, callback);

}

}

}

**let** entry = "./src/world.js";

**let** options = {

resource: path.join(\_\_dirname, entry),

loaders: [

path.join(\_\_dirname, "loaders/loader1.js"),

path.join(\_\_dirname, "loaders/loader2.js"),

path.join(\_\_dirname, "loaders/loader3.js")

]

};

runLoaders(options, (err, result) => {

console.log(result);

});

## **5. file**

* file-loader 并不会对文件内容进行任何转换，只是复制一份文件内容，并根据配置为他生成一个唯一的文件名。

### **5.1 file-loader**

* [loader-utils](https://github.com/webpack/loader-utils)
* [file-loader](https://github.com/webpack-contrib/file-loader/blob/master/src/index.js)
* [public-path](https://webpack.js.org/guides/public-path/" \l "on-the-fly)

**const** { getOptions, interpolateName } = require('loader-utils');**function** **loader**(content) {

**let** options=getOptions(**this**)||{};

**let** url = interpolateName(**this**, options.filename || "[hash].[ext]", {content});

**this**.emitFile(url, content);

**return** `module.exports = ${JSON.stringify(url)}`;

}

loader.raw = true;module.exports = loader;

* 通过 loaderUtils.interpolateName 方法可以根据 options.name 以及文件内容生成一个唯一的文件名 url（一般配置都会带上hash，否则很可能由于文件重名而冲突）
* 通过 this.emitFile(url, content) 告诉 webpack 我需要创建一个文件，webpack会根据参数创建对应的文件，放在 public path 目录下
* 返回 module.exports = ${JSON.stringify(url)},这样就会把原来的文件路径替换为编译后的路径

### **5.2 url-loader**

**let** { getOptions } = require('loader-utils');**var** mime = require('mime');**function** **loader**(source) {

**let** options=getOptions(**this**)||{};

**let** { limit, fallback='file-loader' } = options;

**if** (limit) {

limit = parseInt(limit, 10);

}

**const** mimetype=mime.getType(**this**.resourcePath);

**if** (!limit || source.length < limit) {

**let** base64 = `data:${mimetype};base64,${source.toString('base64')}`;

**return** `module.exports = ${JSON.stringify(base64)}`;

} **else** {

**let** fileLoader = require(fallback || 'file-loader');

**return** fileLoader.call(**this**, source);

}

}

loader.raw = true;module.exports = loader;

### **5.3 样式处理**

* [css-loader](https://github.com/webpack-contrib/css-loader/blob/master/lib/loader.js) 的作用是处理css中的 @import 和 url 这样的外部资源
* [style-loader](https://github.com/webpack-contrib/style-loader/blob/master/index.js) 的作用是把样式插入到 DOM中，方法是在head中插入一个style标签，并把样式写入到这个标签的 innerHTML里
* [less-loader](https://github.com/webpack-contrib/less-loader) 把less编译成css
* [pitching-loader](https://webpack.js.org/api/loaders/" \l "pitching-loader)
* [loader-utils](https://github.com/webpack/loader-utils)
* [!!](https://webpack.js.org/concepts/loaders/" \l "configuration)

$ cnpm i less postcss css-selector-tokenizer -D

#### **5.3.2 使用less-loader**

##### **5.3.2.1 index.js**

src\index.js

**import** './index.less';

##### **5.3.2.2 src\index.less**

src\index.less

@color:red;#root{

color:@color;

}

##### **5.3.2.3 src\index.html**

src\index.html

<div id="root">hello</div><div class="avatar"></div>

##### **5.3.2.4 webpack.config.js**

webpack.config.js

{

test: /\.less$/,

use: [

'style-loader',

'less-loader'

]

}

##### **5.3.2.5 less-loader.js**

**let** less = require('less');**function** **loader**(source) {

**let** callback = **this**.async();

less.render(source, { filename: **this**.resource }, (err, output) => {

callback(err, output.css);

});

}module.exports = loader;

##### **5.3.2.6 style-loader**

**function** **loader**(source) {

**let** script=(`

let style = document.createElement("style");

style.innerHTML = ${JSON.stringify(source)};

document.head.appendChild(style);

module.exports = "";

`);

**return** script;

} module.exports = loader;

#### **5.3.5 两个左侧模块连用**

##### **5.3.5.1 less-loader.js**

**let** less = require('less');**function** **loader**(source) {

**let** callback = **this**.async();

less.render(source, { filename: **this**.resource }, (err, output) => {

callback(err, `module.exports = ${JSON.stringify(output.css)}`);

});

}module.exports = loader;

##### **5.3.5.2 style-loader.js**

**let** loaderUtils = require("loader-utils");**function** **loader**(source) {

}//https://github.com/webpack/webpack/blob/v4.39.3/lib/NormalModuleFactory.js#L339

loader.pitch = **function** (remainingRequest, previousRequest, data) {

//C:\webpack-analysis2\loaders\less-loader.js!C:\webpack-analysis2\src\index.less

console.log('previousRequest', previousRequest);//之前的路径

//console.log('currentRequest', currentRequest);//当前的路径

console.log('remainingRequest', remainingRequest);//剩下的路径

console.log('data', data);

// !! noPrePostAutoLoaders 不要前后置和普通loader

//\_\_webpack\_require\_\_(/\*! !../loaders/less-loader.js!./index.less \*/ "./loaders/less-loader.js!./src/index.less");

**let** style = `

var style = document.createElement("style");

style.innerHTML = require(${loaderUtils.stringifyRequest(**this**, "!!" + remainingRequest)});

document.head.appendChild(style);

`;

**return** style;

}module.exports = loader;

#### **5.3.6 css-loader.js**

* css-loader 的作用是处理css中的 @import 和 url 这样的外部资源
* [postcss](https://github.com/postcss/postcss" \l "usage)
* Avoid CSS @import CSS @importallows stylesheets to import other stylesheets. When CSS @import isused from an external stylesheet, the browser is unable to downloadthe stylesheets in parallel, which adds additional round-trip timesto the overall page load.

##### **5.3.6.1 src\index.js**

src\index.js

require('./style.css');

##### **5.3.6.2 src\style.css**

**@import** './global.css';.avatar {

width: 100px;

height: 100px;

background-image: url('./baidu.png');

background-size: cover;

}div{

color:red;

}

##### **5.3.6.3 src\global.css**

body {

background-color: green;

}

##### **5.3.6.4 webpack.config.js**

+ {+ test: /\.css$/,+ use: [+ 'style-loader',+ 'css-loader'+ ]+ },+ {+ test: /\.png$/,+ use: [+ 'file-loader'+ ]+ }

##### **5.3.6.5 css-loader.js**

loaders\css-loader.js

**var** postcss = require("postcss");**var** loaderUtils = require("loader-utils");**var** Tokenizer = require("css-selector-tokenizer");

**const** cssLoader = **function** (inputSource) {

**const** cssPlugin = (options) => {

**return** (root) => {

root.walkAtRules(/^import$/i, (rule) => {

rule.remove();

options.imports.push(rule.params.slice(1, -1));

});

root.walkDecls((decl) => {

**var** values = Tokenizer.parseValues(decl.value);

values.nodes.forEach(**function** (value) {

value.nodes.forEach(item => {

**if** (item.type === "url") {

item.url = "`+require(" + loaderUtils.stringifyRequest(**this**, item.url) + ")+`";

}

});

});

decl.value = Tokenizer.stringifyValues(values);

console.log(decl);

});

};

}

**let** callback = **this**.async();

**let** options = { imports: [] };

**let** pipeline = postcss([cssPlugin(options)]);

pipeline.process(inputSource).then((result) => {

**let** importCss = options.imports.map(url => "`+require(" + loaderUtils.stringifyRequest(**this**, "!!css-loader!" + url) + ")+`").join('\r\n');

callback(

null,

'module.exports=`' + importCss + '\n' + result.css + '`'

);

});

};

module.exports = cssLoader;