

之前的webpack配置是单页面的配置，它只有一个页面，并且配置起来是非常方便的，只要处理一些常用的文件类型就可以了，如js的兼容必一，CSS的兼容性以及图片及 `scss` 等文件就可以了
但是如果有多文件的时候，我们就要使用另一种方式来完成

```
1  const path = require("path");
2  const HTMLWebapckPlugin = require("html-webpack-plugin");
3  const MiniCssExtractPlugin = require("mini-css-extract-plugin");
4  const webpack = require("webpack");
5  const CopyWebpackPlugin = require("copy-webpack-plugin");
6
7  /**
8   * @type {import("webpack").Configuration}
9   */
10
11  const config = {
12    target: ["web", "es5"],
13    mode: "production",
14    //入口,多页面的入口，可以以对象的形式去完成
15    entry: {
16      index: "./js/index.js",
17      login: "./js/login.js"
18    },
19    //出口
20    output: {
21      path: path.join(__dirname, "./dist"),
22      filename: "js/[name].[fullhash:8].js",
23      publicPath: "./",
24      clean: true
25    },
26    //模块
27    module: {
28      rules: [
29        {
30          test: /\.js$/,
31          exclude: /node_modules/, exclude: /node_modules/,
32          // use:["babel-loader"]
33          use: [
34            {
35              loader: "babel-loader"
36            }
37          ]
38        }, {
39          test: /\.css$/,
40          use: [
41            {
42              loader: MiniCssExtractPlugin.loader
43            },
44            {
45              loader: "css-loader",
46              options: {
47                importLoaders: 1
48              }
49            },
50            "postcss-loader"
```

```

51     ]
52   },
53   {
54     test: /\.s[ca]ss$/,
55     use: [
56       {
57         loader: MiniCssExtractPlugin.loader
58       },
59       {
60         loader: "css-loader",
61         options: {
62           importLoaders: 2
63         }
64       },
65       "postcss-loader",
66       "sass-loader"
67     ]
68   },
69   {
70     test: /\.?(jpe?g|png|svg|bmp|gif)$/i,
71     use: [
72       {
73         loader: "url-loader",
74         options: {
75           // 小于8KB以后的图片就转base64, 否则就不转
76           limit: 8 * 1024,
77           name: "[name].[fullhash:8].[ext]",
78           outputPath: "img/",
79           esModule: false,
80           publicPath: "../img"
81         }
82       }
83     ],
84     type: "javascript/auto"
85   },
86   {
87     test: /\.?(ttf|eot|woff|woff2)$/i,
88     use: [
89       {
90         loader: "url-loader",
91         options: {
92           // 小于8KB以后的图片就转base64, 否则就不转
93           limit: 8 * 1024,
94           name: "[name].[fullhash:8].[ext]",
95           outputPath: "fonts/",
96           esModule: false,
97           publicPath: "../fonts"
98         }
99       }
100     ],
101     type: "javascript/auto"
102   }
103 ]
104 },
105 // 插件
106 plugins: [

```

```

107     new HTMLWebapckPlugin({
108         filename: "index.html",
109         template: path.join(__dirname, "../index.html"),
110         inject: true,
111         chunks: ["index"]
112     }),
113     new HTMLWebapckPlugin({
114         filename: "login.html",
115         template: path.join(__dirname, "../login.html"),
116         inject: true,
117         chunks: ["login"]
118     }),
119     new MiniCssExtractPlugin({
120         filename: "css/[name].[fullhash:8].css",
121         ignoreOrder: false
122     }),
123     //打包的时候会显示进度条
124     new webpack.ProgressPlugin(),
125     new CopyWebpackPlugin({
126         patterns: [
127             {
128                 from: path.join(__dirname, "../static"),
129                 to: path.join(__dirname, "../dist/static")
130             }
131         ]
132     }),
133     //定义全局变量, 如jquery
134     new webpack.ProvidePlugin({
135         "$": "jquery",
136         "jQuery": "jquery",
137         "window.jquery": "jquery"
138     })
139 ],
140 // 优化配置
141 optimization: {
142     // 分离模块
143     splitChunks: {
144         // 所有公共的模块, 我们都分离出来
145         chunks: "all"
146     }
147 }
148 }
149
150
151 module.exports = config;
152
153
154 //如果有一个文件之前已经被 webpack处理了, 不需要再次处理的时候 , 我们就可以放在`static`的目
录下面
155 //最后再通过`CopyWebpackPlugin`来复制到目标目标就可以了

```