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
但是如果有多个文件的时候，我们就要使用另一种方式来完成

```
const path = require("path");
const HTMLWebapckPlugin = require("html-webpack-plugin");
const MiniCssExtractPlugin = require("mini-css-extract-plugin");
const webpack = require("webpack");
const CopyWebpackPlugin = require("copy-webpack-plugin");

/**
 * @type {import("webpack").Configuration}
 */

const config = {
    target: ["web", "es5"],
    mode: "production",
    //入口,多页面的入口 ，可以以对象的形式去完成
    entry: {
        index: "./js/index.js",
        login: "./js/login.js"
    },
    //出口
    output: {
        path: path.join(__dirname, "./dist"),
        filename: "js/[name].[fullhash:8].js",
        publicPath: "./",
        clean: true
    },
    //模块
    module: {
        rules: [
            {
                test: /\.js$/,
                exclude: /node_modules/, exclude: /node_modules/,
                // use:["babel-loader"]
                use: [
                    {
                        loader: "babel-loader"
                    }
                ]
            }, {
                test: /\.css$/,
                use: [
                    {
                        loader: MiniCssExtractPlugin.loader
                    },
                    {
                        loader: "css-loader",
                        options: {
                            importLoaders: 1
                        }
                    },
                    "postcss-loader"
```

```
                ]
            },
            {
                test: /\.s[ca]ss$/,
                use: [
                    {
                        loader: MiniCssExtractPlugin.loader
                    },
                    {
                        loader: "css-loader",
                        options: {
                            importLoaders: 2
                        }
                    },
                    "postcss-loader",
                    "sass-loader"
                ]
            },
            {
                test: /\.(jpe?g|png|svg|bmp|gif)$/,
                use: [
                    {
                        loader: "url-loader",
                        options: {
                            // 小于8KB以后的图片就转base64,否则就不转
                            limit: 8 * 1024,
                            name: "[name].[fullhash:8].[ext]",
                            outputPath: "img/",
                            esModule: false,
                            publicPath: "../img"
                        }
                    }
                ],
                type: "javascript/auto"
            },
            {
                test: /\.(ttf|eot|woff|woff2)$/,
                use: [
                    {
                        loader: "url-loader",
                        options: {
                            // 小于8KB以后的图片就转base64,否则就不转
                            limit: 8 * 1024,
                            name: "[name].[fullhash:8].[ext]",
                            outputPath: "fonts/",
                            esModule: false,
                            publicPath: "../fonts"
                        }
                    }
                ],
                type: "javascript/auto"
            }
        ]
    },
    //插件
    plugins: [
```

```
        new HTMLWebapckPlugin({
            filename: "index.html",
            template: path.join(__dirname, "./index.html"),
            inject: true,
            chunks: ["index"]
        }),
        new HTMLWebapckPlugin({
            filename: "login.html",
            template: path.join(__dirname, "./login.html"),
            inject: true,
            chunks: ["login"]
        }),
        new MiniCssExtractPlugin({
            filename: "css/[name].[fullhash:8].css",
            ignoreOrder: false
        }),
        //打包的时候会显示进度条
        new webpack.ProgressPlugin(),
        new CopyWebpackPlugin({
            patterns: [
                {
                    from: path.join(__dirname, "./static"),
                    to: path.join(__dirname, "./dist/static")
                }
            ]
        }),
        //定义全局变量，如jquery
        new webpack.ProvidePlugin({
            "$": "jquery",
            "jQuery": "jquery",
            "window.jquery": "jquery"
        })
    ],
    // 优化配置
    optimization: {
        // 分离模块
        splitChunks: {
            // 所有公共的模块，我们都分离出来
            chunks: "all"
        }
    }
}


module.exports = config;


//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
//最后再通过`CopyWebpackPlugin`来复制到目标目标就可以了
```