---

link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=198 sentences=340, words=1972

---

1. webpack介绍

- webpack (https://webpack.js.org)是一个JavaScript 应用程序的静态模块打包工具

1.1 安装

```
npm install  webpack webpack-cli --save-dev
```

1.2 入口(entry)

- 入口起点(entry point)指示 webpack 应该使用哪个模块，来作为构建其内部 依赖图(dependency graph) 的开始。进入入口起点后，webpack 会找出有哪些模块和库是入口起点（直接和间接）依赖的
- 默认值是 `./src/index.js`，但你可以通过在 `webpack configuration` 中配置 `entry` 属性，来指定一个（或多个）不同的入口起点

1.2.1 src\index.js

```
import './index.css'
```

1.2.2 index.css

```
body{
  background-color:green;
}
```

1.2.3 webpack.config.js

```
const path = require('path');
module.exports = {
  entry: './src/index.js',
};
```

1.3 输出(output)

- `output` 属性告诉 webpack 在哪里输出它所创建的 bundle，以及如何命名这些文件
- 主要输出文件的默认值是 `./dist/main.js`，其他生成文件默认放置在 `./dist` 文件夹中。

webpack.config.js

```
const path = require('path');
module.exports = {
  entry: './src/index.js',
+  output: {
+    path: path.resolve(__dirname, 'dist'),
+    filename: 'main.js'
+  }
};
```

1.4 loader

- webpack 只能理解 `JavaScript` 和 `JSON` 文件
- loader 让 `webpack` 能够去处理其他类型的文件，并将它们转换为有效模块，以供应用程序使用，以及被添加到依赖图中

webpack.config.js

```
const path = require('path');
module.exports = {
  mode: 'development',
  devtool:false,
  entry: './src/index.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'main.js'
  },
+  module: {
+    rules: [
+      { test: /\.css$/, use: ['style-loader','css-loader']}
+    ]
+  }
};
```

1.5 插件(plugin)

- loader 用于转换某些类型的模块，而插件则可以用于执行范围更广的任务。包括：打包优化，资源管理，注入环境变量

1.5.1 src\index.html

src\index.html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>webpack5title>
head>
<body>
body>
html>
```

1.5.2 webpack.config.js

```
const path = require('path');
+const HtmlWebpackPlugin = require('html-webpack-plugin');
module.exports = {
  mode: 'development',
  devtool:false,
  entry: './src/index.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'main.js'
  },
  module: {
    rules: [
      { test: /\.css$/, use: ['style-loader','css-loader']}
    ]
  },
+  plugins: [
+    new HtmlWebpackPlugin({template: './src/index.html'})
+  ]
};
```

1.6 模式(mode)

- webpack 4.x 版本引入了mode (https://webpack.docschina.org/configuration/mode/) 的概念

1.6.1 环境差异

- 开发环境
    - 需要生成 sourcemap 文件
    - 需要打印 debug 信息
    - 需要 live reload 或者 hot reload 的功能
- 生产环境
    - 可能需要分离 CSS 成单独的文件，以便多个页面共享同一个 CSS 文件
    - 需要压缩 HTML/CSS/JS 代码
    - 需要压缩图片
- 其默认值为 production

1.6.2 区分环境

- --mode用来设置模块内的 process.env.NODE_ENV
- cross-env用来设置node环境的 process.env.NODE_ENV
- DefinePlugin用来设置模块内的全局变量

1.6.2.1 命令行配置

- webpack的mode默认为 production
- webpack serve的mode默认为 development
- 可以在模块内通过 process.env.NODE_ENV获取当前的环境变量,无法在webpack配置文件中获取此变量

```
  "scripts": {
    "build": "webpack",
    "dev": "webpack serve"
  },
```

index.js

```
console.log(process.env.NODE_ENV);
```

webpack.config.js

```
console.log('NODE_ENV',process.env.NODE_ENV);
```

1.6.2.2 命令行配置

```
  "scripts": {
    "build": "webpack --mode=production",
    "dev": "webpack --mode=development serve"
  },
```

1.6.2.3 mode配置

```
module.exports = {
  mode: 'development'
}
```

1.6.2.4 DefinePlugin

- 可以在任意模块内通过 process.env.NODE_ENV 获取当前的环境变量
- 但无法在 node环境(webpack 配置文件中)下获取当前的环境变量

```
plugins:[
   new webpack.DefinePlugin({
      'process.env.NODE_ENV':JSON.stringify(process.env.NODE_ENV)
   })
]
```

index.js

```
console.log(NODE_ENV);
```

webpack.config.js

```
console.log('process.env.NODE_ENV',process.env.NODE_ENV);
console.log('NODE_ENV',NODE_ENV);
```

1.6.2.5 cross-env

- 只能设置 node环境下的变量NODE_ENV

package.json

```
"scripts": {
  "build": "cross-env NODE_ENV=development webpack"
}
```

webpack.config.js

```
console.log('process.env.NODE_ENV',process.env.NODE_ENV);
```

2 开发服务器

2.1 安装服务器

```
npm install webpack-dev-server --save-dev
```

2.2 webpack.config.js

类别 配置名称 描述 output path 指定输出到硬盘上的目录 output publicPath 表示的是打包生成的index.html文件里面引用资源的前缀 devServer publicPath 表示的是打包生成的静态文件所在的位置(若是devServer里面的publicPath没有设置，则会认为是output里面设置的publicPath的值) devServer static 用于配置提供额外静态文件内容的目录

2.3 webpack.config.js

```
module.exports = {
  devServer: {
    static: path.resolve(__dirname, 'public'),
    port: 8080,
    open: true
  }
}
```

2.4 package.json

```
  "scripts": {
    "build": "webpack",
+   "dev": "webpack serve"
  }
```

3. 支持CSS

- css-loader (https://www.npmjs.com/package/css-loader)用来翻译处理@import和url()
- style-loader (https://www.npmjs.com/package/style-loader)可以把CSS插入DOM中

3.1 安装模块

```
npm i style-loader css-loader -D
```

3.2 webpack.config.js

```
const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');
module.exports = {
  mode: 'development',
  devtool:false,
  entry: './src/index.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: '[name].js'
  },
  module: {
    rules: [
     { test: /\.css$/, use: ['style-loader','css-loader'] }
    ]
  },
  plugins: [
    new HtmlWebpackPlugin({template: './src/index.html'})
  ]
};
```

3.3 src\bg.css

src\bg.css

```
body{
    background-color: green;
}
```

3.4 src\index.css

src\index.css

```
@import "./bg.css";
body{
    color:red;
}
```

3.5 src\index.js

src\index.js

```
+import './index.css';
```

4. 支持less和sass

- node-sass (https://www.npmjs.com/package/node-sass)

4.1 安装

```
npm i less less-loader -D
npm i node-sass sass-loader -D
npm rebuild node-sass
```

4.2 webpack.config.js

webpack.config.js

```
const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');
module.exports = {
  mode: 'development',
  devtool: false,
  entry: './src/index.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: '[name].js'
  },
  module: {
    rules: [
      { test: /\.css$/, use: ['style-loader', 'css-loader'] },
+     { test: /\.less$/, use: ['style-loader','css-loader', 'less-loader'] },
+     { test: /\.scss$/, use: ['style-loader','css-loader', 'sass-loader'] }
    ]
  },
  plugins: [
    new HtmlWebpackPlugin({ template: './src/index.html' })
  ]
};
```

4.3 src\index.html

src\index.html

```
    webpack5

+   less-container
+   sass-container
```

4.4 src\index.js

src\index.js

```
import './index.css';
+import './less.less';
+import './sass.scss';
```

4.5 src\less.less

src\less.less

```
@color:blue;
#less-container{
    color:@color;
}
```

4.6 src\sass.scss

src\sass.scss

```
$color:orange;
#sass-container{
    color:$color;
}
```

5. CSS兼容性

- 为了浏览器的兼容性，有时候我们必须加入-webkit,-ms,-o,-moz这些前缀
    - Trident内核：主要代表为IE浏览器, 前缀为-ms
    - Gecko内核：主要代表为Firefox, 前缀为-moz
    - Presto内核：主要代表为Opera, 前缀为-o
    - Webkit内核：产要代表为Chrome和Safari, 前缀为-webkit
- 伪元素 ::placeholder可以选择一个表单元素的占位文本，它允许开发者和设计师自定义占位文本的样式。

5.1 安装

- https://caniuse.com/ (https://caniuse.com/)
- postcss-loader (https://github.com/webpack-contrib/postcss-loader)可以使用PostCSS处理CSS
- postcss-preset-env (https://github.com/csstools/postcss-preset-env)把现代的CSS转换成大多数浏览器能理解的
- PostCSS Preset Env已经包含了 autoprefixer和 browsers选项

```
npm i postcss-loader postcss-preset-env -D
```

5.2 postcss.config.js

postcss.config.js

```
let postcssPresetEnv = require('postcss-preset-env');
module.exports={
    plugins:[postcssPresetEnv({
        browsers: 'last 5 version'
    })]
}
```

5.3 webpack.config.js

```
const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');
module.exports = {
  mode: 'development',
  devtool: false,
  entry: './src/index.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: '[name].js'
  },
  module: {
    rules: [
+     { test: /\.css$/, use: ['style-loader', 'css-loader','postcss-loader'] },
+     { test: /\.less$/, use: ['style-loader','css-loader','postcss-loader','less-loader'] },
+     { test: /\.scss$/, use: ['style-loader','css-loader','postcss-loader','sass-loader'] }
    ]
  },
  plugins: [
    new HtmlWebpackPlugin({ template: './src/index.html' })
  ]
};
```

5.4 src\index.css

src\index.css

```
@import "./bg.css";
body{
    color:red;
}
+#root {
+    background-color: red;
+    width: 100px;
+    height: 10px;
+    transform: rotate(10deg);
+}
```

5.5 src\index.html

src\index.html

```
    webpack5

  less-container
  sass-container
+
```

5.6 package.json

- browserslist (https://github.com/browserslist/browserslist)
- browserslist-example (https://github.com/browserslist/browserslist-example)
- .browserslistrc

```
{
+  "browserslist": {
+    "development": [
+      "last 1 chrome version",
+      "last 1 firefox version",
+      "last 1 safari version"
+    ],
+    "production": [
+      ">0.2%"
+    ]
+  }
}
```

6 资源模块

- 资源模块是一种模块类型，它允许使用资源文件（字体，图标等）而无需配置额外 loader
- `raw-loader` => `asset/source` 导出资源的源代码
- `file-loader` => `asset/resource` 发送一个单独的文件并导出 URL
- `url-loader` => `asset/inline` 导出一个资源的 data URI
- asset 在导出一个 data URI 和发送一个单独的文件之间自动选择。之前通过使用 `url-loader`，并且配置资源体积限制实现
- Rule.type (https://webpack.js.org/configuration/module/#rule)
- asset-modules (https://webpack.js.org/guides/asset-modules/)

6.1 webpack.config.js

```
module.exports = {
    module:{
        rules: [
            {
                test: /\.js$/,
                use: [
                    {
                        loader: 'babel-loader',
                        options: {
                            presets: [
                                "@babel/preset-react"
                            ]
                        },

                    }
                ],
                exclude:/node_modules/
            },
+           {
+               test: /\.png$/,
+               type: 'asset/resource'
+           },
+           {
+               test: /\.ico$/,
+               type: 'asset/inline'
+           },
+           {
+               test: /\.txt$/,
+               type: 'asset/source'
+           },
+           {
+               test: /\.jpg$/,
+               type: 'asset',
+               parser: {
+                   dataUrlCondition: {
+                     maxSize: 4 * 1024 // 4kb
+                   }
+               }
+           }
        ]
    },
  experiments: {
    asset: true
  },
};
```

6.2 src\index.js

src\index.js

```
+ import png from './assets/logo.png';
+ import ico from './assets/logo.ico';
+ import jpg from './assets/logo.jpg';
+ import txt from './assets/logo.txt';
+ console.log(png,ico,jpg,txt);
```

7 JS兼容性处理

- Babel其实是一个编译JavaScript的平台,可以把ES6/ES7,React的JSX转义为ES5

7.1 @babel/preset-env

- Babel默认只转换新的最新ES语法,比如箭头函数

7.1.1 安装依赖

- babel-loader (https://www.npmjs.com/package/babel-loader)使用Babel和webpack转译JavaScript文件
- @babel/@babel/core (https://www.npmjs.com/package/@babel/core)Babel编译的核心包
- babel-preset-env (https://www.babeljs.cn/docs/babel-preset-env)
- @babel/@babel/preset-react (https://www.npmjs.com/package/@babel/preset-react)React插件的Babel预设
- @babel/plugin-proposal-decorators (https://babeljs.io/docs/en/babel-plugin-proposal-decorators)把类和对象装饰器编译成ES5
- @babel/plugin-proposal-class-properties (https://babeljs.io/docs/en/babel-plugin-proposal-class-properties)转换静态类属性以及使用属性初始值化语法声明的属性

```
npm i babel-loader @babel/core @babel/preset-env @babel/preset-react  -D
npm i @babel/plugin-proposal-decorators @babel/plugin-proposal-class-properties @babel/plugin-proposal-private-property-in-object  @babel/plugin-proposal-private-methods -D
```

7.1.2 webpack.config.js

```
const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');
const MiniCssExtractPlugin = require('mini-css-extract-plugin');
module.exports = {
  mode: 'development',
  devtool: false,
  entry: './src/index.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'main.js',
  },
  module: {
    rules: [
+      {
+        test: /\.jsx?$/,
+        use: {
+          loader: 'babel-loader',
+          options: {
+            presets: ["@babel/preset-env", '@babel/preset-react'],
+            plugins: [
+              ["@babel/plugin-proposal-decorators", { legacy: true }],
+              ["@babel/plugin-proposal-private-property-in-object", { "loose": true }],
+              ["@babel/plugin-proposal-private-methods", { "loose": true }],
+              ["@babel/plugin-proposal-class-properties", { loose: true }],
+            ],
+          },
+        },
+      },
      { test: /\.css$/, use: ['style-loader', 'css-loader', 'postcss-loader'] },
      { test: /\.less$/, use: ['style-loader', 'css-loader','postcss-loader', 'less-loader'] },
      { test: /\.scss$/, use: ['style-loader', 'css-loader', 'postcss-loader','sass-loader'] },
      {
        test: /\.(jpg|png|gif|bmp|svg)$/,
        type:'asset/resource',
        generator:{
          filename:'images/[hash][ext]'
        }
      }
    ]
  },
  plugins: [
    new HtmlWebpackPlugin({ template: './src/index.html' })
  ]
};
```

7.1.3 src\index.js

src\index.js

```
+function readonly(target,key,descriptor) {
+    descriptor.writable=false;
+}
+
+class Person{
+    @readonly PI=3.14;
+}
+let p1=new Person();
+p1.PI=3.15;
+console.log(p1)
```

7.1.4 jsconfig.json

- jsconfig (https://code.visualstudio.com/docs/languages/jsconfig)

jsconfig.json

```
{
    "compilerOptions": {
        "experimentalDecorators": true
    }
}
```

8. ESLint代码校验

8.1 安装

- eslint (https://eslint.org/docs/developer-guide/nodejs-api#cliengine)
- eslint-loader (https://www.npmjs.com/package/eslint-loader)
- configuring (https://eslint.org/docs/user-guide/configuring)
- babel-eslint (https://www.npmjs.com/package/babel-eslint)
- Rules (https://cloud.tencent.com/developer/chapter/12618)
- ESlint 语法检测配置说明 (https://segmentfault.com/a/1190000008742240)

```
npm install eslint eslint-loader babel-eslint --D
```

8.2 webpack.config.js

```
const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');
const MiniCssExtractPlugin = require('mini-css-extract-plugin');
module.exports = {
  mode: 'development',
  devtool: false,
  entry: './src/index.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: '[name].js',
  },
  module: {
    rules: [
+      {
+        test: /\.jsx?$/,
+        loader: 'eslint-loader',
+        enforce: 'pre',
+        options: { fix: true },
+        exclude: /node_modules/,
+      },
      {
        test: /\.jsx?$/,
        use: {
          loader: 'babel-loader',
          options: {
            "presets": ["@babel/preset-env"],
            "plugins": [
              ["@babel/plugin-proposal-decorators", { legacy: true }],
              ["@babel/plugin-proposal-private-property-in-object", { "loose": true }],
              ["@babel/plugin-proposal-private-methods", { "loose": true }],
              ["@babel/plugin-proposal-class-properties", { loose: true }],
            ]
          }
        },
        include: path.join(__dirname, 'src'),
        exclude: /node_modules/
      },
      { test: /\.css$/, use: ['style-loader', 'css-loader', 'postcss-loader'] },
      { test: /\.less$/, use: ['style-loader', 'css-loader','postcss-loader', 'less-loader'] },
      { test: /\.scss$/, use: ['style-loader', 'css-loader', 'postcss-loader','sass-loader'] },
      {
        test: /\.(jpg|png|bmp|gif|svg)$/, use: [{
          loader: 'url-loader', options: {
            limit: 10
          }
        }]
      }
    ]
  },
  plugins: [
    new HtmlWebpackPlugin({ template: './src/index.html' })
  ]
};
```

8.3 src\index.html

src\index.html

```
    webpack5

+
```

8.4 src\index.js

src\index.js

```
+import React from "react";
+import ReactDOM from "react-dom";
+ReactDOM.render("hello",document.getElementById("root"));
+
+function readonly(target,key,descriptor) {
+    descriptor.writable=false;
+}
+
+class Person{
+    @readonly PI=3.14;
+}
+let p1=new Person();
+p1.PI=3.15;
```

8.5 .eslintrc.js

.eslintrc.js

```
module.exports = {
    root: true,
    parser:"babel-eslint",

    parserOptions: {
        sourceType: "module",
        ecmaVersion: 2015
    },

    env: {
        browser: true,
    },

    rules: {
        "indent": "off",
        "quotes":  "off",
        "no-console": "error",
    }
}
```

8.6 airbnb

- eslint-config-airbnb (https://github.com/airbnb/javascript/tree/master/packages/eslint-config-airbnb)

```
npm i eslint-config-airbnb eslint-loader eslint eslint-plugin-import eslint-plugin-react eslint-plugin-react-hooks and eslint-plugin-jsx-a11y -D
```

eslintrc.js

```
module.exports = {
    "parser":"babel-eslint",
    "extends":"airbnb",
    "rules":{
        "semi":"error",
        "no-console":"off",
        "linebreak-style":"off",
        "eol-last":"off"

    },
    "env":{
        "browser":true,
        "node":true
    }
}
```

8.7 自动修复

- 安装vscode的eslint (https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint)插件
- 配置自动修复参数

.vscode\settings.json

```
{
    "eslint.validate": [
        "javascript",
        "javascriptreact",
        "typescript",
        "typescriptreact"
    ],
    "editor.codeActionsOnSave": {
        "source.fixAll.eslint": true
    }
  }
```

9. 服务器代理

如果你有单独的后端开发服务器 API，并且希望在同域名下发送 API 请求，那么代理某些 URL 会很有用。

9.1 不修改路径

- 请求到 /api/users 现在会被代理到请求http://localhost:3000/api/users。 (http://localhost:3000/api/users。)

```
devServer: {
  proxy: {
    "/api": 'http://localhost:3000'
  }
}
```

9.2 修改路径

```
devServer: {
  proxy: {
      "/api": {
       target: 'http://localhost:3000',
       pathRewrite:{"^/api":""}
      }
  }
}
```

9.3 onBeforeSetupMiddleware

- onBeforeSetupMiddleware 在 webpack-dev-server 静态资源中间件处理之前，可以用于拦截部分请求返回特定内容，或者实现简单的数据 mock。

```
devServer: {
onBeforeSetupMiddleware(devServer){
    devServer.app.get('/api/users', (req, res) => {
      res.json([{ id: 1 }, { id: 2 }]);
    });
}
}
```

9.4 webpack-dev-middleware

webpack-dev-middleware (https://www.npmjs.com/package/)就是在 Express 中提供 webpack-dev-server 静态服务能力的一个中间件

```
npm install webpack-dev-middleware --save-dev
```

```
const express = require('express');
const app = express();
const webpack = require('webpack');
const webpackDevMiddleware = require('webpack-dev-middleware');
const webpackOptions = require('./webpack.config');
webpackOptions.mode = 'development';
const compiler = webpack(webpackOptions);
app.use(webpackDevMiddleware(compiler, {}));
app.listen(3000);
```

- webpack-dev-server 的好处是相对简单，直接安装依赖后执行命令即可
- 而使用 webpack-dev-middleware的好处是可以在既有的 Express 代码基础上快速添加 webpack-dev-server 的功能，同时利用 Express 来根据需要添加更多的功能，如 mock 服务、代理 API 请求等