# Webpack 5.x - DAY02

# 课程介绍

* 核心loader【掌握】
* 插件plugins【掌握】
* 开发服务器devServer【了解】

# 核心loader【了解】

## 2.1 加载less

1. 下载对应的loader

|  |
| --- |
| **yarn add less-loader less -D // style-loader css-loader 前面处理css下载过了** |

1. 配置loader

|  |
| --- |
| // loader配置    module: {      rules: [        {          test: /\.less$/,  // 匹配所有.less文件          use: [            'style-loader', // 创建style标签 把css插入html的head内            'css-loader', // 解析加载css到js中（把css打成字符串）。            'less-loader', // 编译less为css          ]        }      ]    } |

## 2.2 加载img

1. **在css中加载img**
2. 下载对应loader

|  |
| --- |
| **yarn add url-loader file-loader -D** |

1. 配置loader

|  |
| --- |
| // loader配置    module: {      rules: [        {          test: /\.(jpg|png|gif)$/,          loader: 'url-loader',          options: {            name: '[hash:16].[ext]',  // 图片输出的名字hash长度16位 默认32位  limit: 10 \* 1024,  // 小于10kb base64处理            // 关闭url-loader的es Module  使用commonjs module解析 , url-loader默认使用es6模块化解析 而后续使用 html-loader 引入图片是 commonjs解析  不关闭会报错            esModule: false,     outputPath: 'img' // 图片输出文件夹        }        }      ]    }, |

1. **在html中加载img**
2. 下载对应loader 和 处理html的插件

|  |
| --- |
| **yarn add html-loader@1.3.2 html-webpack-plugin@4.5.0 -D** |

1. 配置loader

|  |
| --- |
| const HtmlWebpackPlugin = require('html-webpack-plugin') // 处理html的插件  // loader配置    module: {      rules: [        {          test: /\.html$/, **// 处理html中引入img**          loader: 'html-loader'        },      ]    }, |

1. 配置插件

|  |
| --- |
| plugins: [      // 打包html      new HtmlWebpackPlugin({        template: './src/index.html',  // 把index.html复制到dist中 自动引入资源文件  filename: ‘index.html’      })    ] |

1. 配置静态资源查找路径

|  |
| --- |
| // 2. 出口    output: {      path: path.resolve(\_\_dirname, 'dist'), // 输出路径 必须是绝对路径      filename: 'bundle.js',  // 输出的文件名      publicPath: './', // 查找静态资源的路径    }, |

1. **在JavaScript中加载本地img**

|  |
| --- |
| const imgBox = document.querySelector('#imgBox')  // 在js中引入本地图片 打包图片  let imgName = require('./xiaodc.jpg')  imgBox.src = imgName |

## 2.3. 加载字体图标

1. 下载对应loader

|  |
| --- |
| **yarn add file-loader -D // 之前下载过了 可以不用下载** |

1. 配置loader

|  |
| --- |
| {       test: /\.(eot|svg|ttf|woff|woff2)$/,  // 处理字体格式文件       loader: 'file-loader',       options: {            name: '[hash:16].[ext]', // 输出名字            outputPath: 'fonts' // 输出路径（输出到文件夹fonts中）       }  }, |

## 2.3 编译es6到es5

1. 下载对应依赖模块

|  |
| --- |
| **yarn add babel-core babel-loader@7.1.5 babel-preset-es2015** -D |

1. 配置loader

|  |
| --- |
| {          test: /\.js$/,          loader: 'babel-loader',    // loader 编译es6为es5          exclude: /node\_modules/  // 排除    } |

1. 在项目根目录 创建 **.babelrc** 文件, 内容如下。

|  |
| --- |
| {    "presets": [        "es2015"    ]  } |

# 3 插件plugins 【掌握】

1. 下载插件

|  |
| --- |
| **yarn add 插件名 -D**  // 安装如下3个插件  **html-webpack-plugin@4.5.0** // 打包html的插件。  **mini-css-extract-plugin** // 提取css的插件（把css提取成为独立文件）。  **optimize-css-assets-webpack-plugin** // 压缩css的插件 |

1. 引入插件

|  |
| --- |
| **const 变量 = require(‘插件名’)** |

1. 在plugins里面配置插件

|  |
| --- |
| // 处理html的插件  const HtmlWebpackPlugin = require('html-webpack-plugin')  // 提取css  const MiniCssExtractPlugin = require('mini-css-extract-plugin')  // 压缩css  const OptimizeCssAssetsWebpackPlugin = require('optimize-css-assets-webpack-plugin')  // 配置  plugins: [       // 处理html  把index.html复制到dist中 自动引入资源文件      new HtmlWebpackPlugin({        template: './src/index.html'      }),  // 提取css，把css中打包的js中提取出来成为独立文件      new MiniCssExtractPlugin({ // 提取css        filename: 'css/[name].css' // 输出到css文件夹里      }),      new OptimizeCssAssetsWebpackPlugin() // 压缩css    ], |

1. 修改之前的loader配置，把style-loader的配置修改成如下配置（css需要提取成独立文件，不再需要style-loader插入head标签里面了）

|  |
| --- |
| // 处理css        {          test: /\.css$/,          use: [            {              loader: MiniCssExtractPlugin.loader, // 提取css              options: {                publicPath: '../' // 资源查找路径              }            },            'css-loader'          ]        },    // 处理less        {          test: /\.less$/,          use: [            {              loader: MiniCssExtractPlugin.loader, // 从js中提取css              options: {                publicPath: '../'  // 资源查找路径              }            },            'css-loader',  // 解析css            'postcss-loader', // 处理css兼容性            'less-loader' // 编译less          ]        }, |

# 4 开发服务器devServer【掌握】

## 4.1 启动开发服务器

1. 局部安装 webpack-dev-server
2. 注意webpack-cli版本需要3.3.12

|  |
| --- |
| **yarn add [webpack-dev-server@](mailto:webpack-dev-server@3.11.2)**[3.11.2](mailto:webpack-dev-server@3.11.2) **-D** |

1. 配置

|  |
| --- |
| // 开发服务器    devServer: {      contentBase: path.resolve(\_\_dirname, 'dist'), // 启动服务器目录      compress: true, // 启动gzip      port: 666,  // 端口      open: true, // 自动打开服务      publicPath: '/', // 静态资源查找路径      openPage: 'index.html', // 打开的页面    },    target: 'web', // 目标是浏览器 |

1. 修改 package.json 配置

|  |
| --- |
| "scripts": {      "serve": "webpack-dev-server",      "build": "webpack"    }, |

1. 执行命令，启动开发服务器

|  |
| --- |
| **yarn serve** |

## 4.2 区分开发环境和生产环境

**1）安装 cross-env**

|  |
| --- |
| **yarn add cross-env -D** |

1. **修改 package.json 的配置**

|  |
| --- |
| "scripts": {      "serve": "cross-env NODE\_ENV=development webpack-dev-server",      "build": "cross-env NODE\_ENV=production webpack"    }, |

1. **修改模式配置**

|  |
| --- |
| mode: process.env.NODE\_ENV |

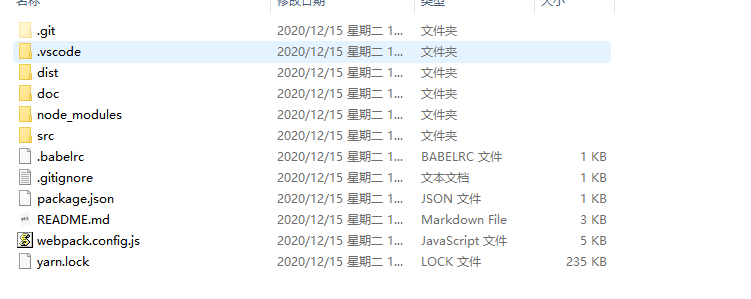
1. **最终命令**

|  |
| --- |
| // 开发执行  **yarn serve**  // 打包执行  **yarn build** |

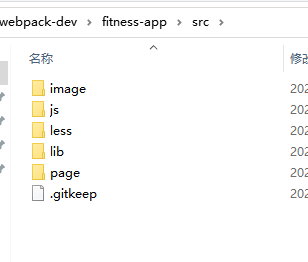
# 5 健身FitNessApp项目架构搭建

## 5.1 项目目录搭建

一级目录



Src开发目录



## 5.1 Wepack完整配置

### Webpack.config.js

|  |
| --- |
| /\*\*   \* webpack核心配置文件   \*/  const path = require('path') // 处理路径模块  node内置的  // 打包html  const HtmlWebpackPlugin = require('html-webpack-plugin')  // 从js中提取css成独立文件  const MiniCssExtractPlugin = require('mini-css-extract-plugin')  const OptimizeCssAssetsWebpackPlugin = require('optimize-css-assets-webpack-plugin') // 压缩css  module.exports = {    // 1. 入口    entry: {      commonCss: './src/js/common/commonCss.js', // 公用的css      iconfont: "./src/js/common/iconfont.js", // 字体图标      // 公用js      dom: './src/js/common/dom.js',      utils: './src/js/common/utils.js',      http: './src/js/common/http.js',      login: "./src/js/login.js",  // 登录      home: "./src/js/home.js",  // 首页    },    // 2. 出口    output: {      path: path.resolve(\_\_dirname, 'dist'), // 输出目录      filename: 'js/[name]-[hash:10].js', // 输出的文件名      publicPath: './',  // 资源路径    },    // 3. loader配置    module: {      rules: [        // 处理css        {          test: /\.css$/,          use: [            {              loader: MiniCssExtractPlugin.loader, // 提取css              options: {                publicPath: '../' // 输出路径              }            },            'css-loader'          ]        },        // 处理less        {          test: /\.less$/,          use: [            {              loader: MiniCssExtractPlugin.loader, // 从js中提取css              options: {                publicPath: '../'  // 资源查找路径              }            },            'css-loader',  // 解析css            'less-loader' // 编译less          ]        },        // 处理图片        {          test: /\.(jpg|png|gif)$/,          loader: 'url-loader',          options: {            limit: 10 \* 1024,            name: '[hash:10].[ext]',            esModule: false, // 关闭es module            outputPath: 'img',  // 打包后输出目录          }        },        // 处理html中引入的图片        {          test: /\.html$/,          loader: 'html-loader',        },        // 处理字体格式文件        {          test: /\.(eot|svg|ttf|woff|woff2)$/,  // 处理字体格式文件          loader: 'file-loader',          options: {            name: '[hash:16].[ext]',            outputPath: 'fonts'          }        },        // 处理js        {          test: /\.js$/,          loader: 'babel-loader',    // loader          exclude: /node\_modules/  // 排除        },        // 其他资源        {          exclude: /\.(html|js|json|css|less|jpg|png|gif|eot|svg|ttf|woff|woff2)$/,          loader: 'file-loader',          options: {            name: '[hash:16].[ext]',          }        },      ]    },    // 4. 模式    mode: process.env.NODE\_ENV    // 5. 插件    plugins: [      // 登录页      new HtmlWebpackPlugin({        template: './src/page/login.html', // 模板页面        filename: 'login.html', // 输出的文件名        chunks: ['iconfont', 'commonCss', 'dom', 'utils', 'http', 'login']  // 需要引入哪些资源      }),      // 首页      new HtmlWebpackPlugin({        template: './src/page/home.html', // 模板页面        filename: 'home.html', // 输出的文件名        chunks: ['iconfont', 'commonCss', 'home']  // 需要引入哪些资源      }),      // 提取css      new MiniCssExtractPlugin({        filename: 'css/[name]-[hash:5].css', // 重命名输出的css      }),      // 压缩css      new OptimizeCssAssetsWebpackPlugin(),      ],    // 6. 启动开发服务器   // 开发服务器    devServer: {      contentBase: path.resolve(\_\_dirname, 'dist'), // 启动服务器目录      compress: true, // 启动gzip      port: 666,  // 端口      open: true, // 自动打开服务      publicPath: '/', // 静态资源查找路径      openPage: 'index.html', // 打开的页面    },    target: 'web', // 目标是浏览器  } |

### package.json

|  |
| --- |
| {    "name": "fitness-app",    "version": "1.0.0",    "main": "index.js",    "license": "MIT",    "scripts": {      "serve": "cross-env NODE\_ENV=development webpack serve",      "build": "cross-env NODE\_ENV=production webpack"    },    "devDependencies": {      "babel-core": "^6.26.3",      "babel-loader": "7.1.5",      "babel-preset-es2015": "^6.24.1",      "css-loader": "^5.0.1",      "file-loader": "^6.2.0",      "html-loader": "^1.3.2",      "html-webpack-plugin": "^4.5.0",      "less": "^3.12.2",      "less-loader": "^7.1.0",      "mini-css-extract-plugin": "^1.3.2",      "optimize-css-assets-webpack-plugin": "^5.0.4",      "style-loader": "^2.0.0",      "url-loader": "^4.1.1",      "webpack": "^5.10.0",      "webpack-cli": "^4.2.0",      "webpack-dev-server": "^3.11.0"    },    "browserslist": {      "development": [        "last 1 chrome version",        "last 1 firefox version",        "last 1 safari version"      ],      "production": [        ">0.1%",        "not op\_mini all"      ]    }  } |

### .babelrc

|  |
| --- |
| {    "presets": [      "es2015"    ]  } |

# 作业

1. 完成完整的多页面开发 **webpac**k 完整配置。
2. 完成页面： **广告页 & 注册 & 登录 （flex布局）。**