黑马

前端工程化

- 模块化 (js模块)
- 组件化(组件复用)
- 规范化(目录结构、编码规范、结构规范、 文档规范、git分支管理)
- 自动化(自动构建、部署、测试)
- 遵守 前端开发所需的工具、技术、流程、经验等
- Vue和Receat基于工程化开发

Webpack

- 一般是不自己配置webpack 使用第三方工具
- 前端工程化的具体解决方案 功能：代码压缩混淆、处理浏览器端js的兼容性、性能优化
- 提高了开发效率 项目可维护性
- vue-cli自动配置好了 webpack 解决人工配置的繁琐操作

基本使用

- 在新文件夹中 npm init -y 初始化包管理文件 新建src
- -s：dependencies 上线时也需要这些包 -d： 上线不需要的包
- 安装：npm i webpack@5.42.1 webpack-cli -D(版本最新)
- 配置： webpack.config.js 文件
- 使用：引入 npm run dev 生成 dist中的 main.js

```
//js  导出语法
module.exports = {
    mode: "development",
    //mode  运行模式 development 开发阶段 production 上线(压缩代码)
}
//在package.json的 scripts节点下, 新增dev脚本
"名字": "webpack"   使用 npm run  名字
```

- 默认 入口 为 src/index.js 出口为 dist/main.js

  修改默认: entry：指定打包入口 output:打包出口

```
//entry: 指定打包入口   output:打包出口
const path = require('path') //导入 node中 path 包
//node 导出语法
module.exports = {
    mode: "development",
    //mode  运行模式 development 开发阶段 production 上线
    entry: path.join(__dirname, './src/index.js'),//打包入口
    output: {
        path: path.join(__dirname, './dist'),//打包出口
        filename: 'js/bundle.js',//文件名字
    }
}
```

插件

- webpack-dev-server 类似 node 中 的 nodemon 修改源代码 自动打包构建
  - npm i webpack-dev-server@4.11.1 -d
  - 将dev 修改为 "dev": "webpack server"
  - 停掉：ctrl + c
  - 直接引入：/main.js   或/bundle.js
- html-webpack-plugin 自定制index.html 页面内容 直接通过端口号 访问到首页
  - npm i html-webpack.plugin -D
  - 功能：复制页面 自动 添加注入打包好的bundle.js
  - 配置

```
//1. 导入html 插件
const HtmlPlugin = require('html-webpack-plugin')
//2.创建html文件
const htmlPlugin = new HtmlPlugin({
    template: './src/index.html',//指定源文件存放路径   复制
    filename: 'index.html' //指定生成文件存放路径    粘贴
})
//node 导出语法
module.exports = {
    mode: "development",
     plugins: [htmlPlugin],//3通过节点 使插件生效

}
```

- devServer 打包自动打开页面

```
//在webpack-config.js
   devServer: {
       open: true, //初次打包完成后，自动打开浏览器
       port: 8080, //实时打包使用的端口号
       host: '127.0.0.1' //主机地址

   },
```

- loder 加载器 打包处理css文件
  - npm i style-loader css-loader -D
  - css 在index.js 导入 css路径
  - 配置：

```
module: { //书写第三方文件模块匹配规则
        rules: [ //文件后缀名匹配规则
            {
                test: /\.css$/,
                //从后往前调
                use: ['style-loader', 'css-loader']
            }
            //test 文件类型 .css类型    use：对应要调用的loader
        ]
    }
```

  - 处理流程：
    - 当代码中包含css文件时 会查找module.rules数组中是否配置了对应的loader加载器
    - 先交给最后一个loader加载器（css-loder），处理完毕后交给前一个以此类推 最后一个loader交给webpack合并到bundle.js文件中
- 处理less：npm i less-loader -D use: ['style-loader', 'css-loader'，'less-loader']
- img：base64格式 - 防止浏览器发出无畏的请求
- 处理url路径 npm i url-loader file-loader -D

```
module: { //书写第三方文件模块匹配规则
        rules: [ //文件后缀名匹配规则
            {
                test: /\.jpg|png|gif$/,
                //?之后时loader参数项    limit指定图片大小 单位字节 只有小于等于limit规定的大小的图片 才转
                //多个参数时使用&风格
                use: 'url-loader?limit=470& outputPath='images'',
                }
            }
            //test 文件类型 .css类型    use：对应要调用的loader
        ]
    }
```

  - 使用:在index.js 引入img import xxx from '路径' 使用js给img标签src属性赋值
- 处理js中的高级语法 babel-loader
  - npm i bable-loader @bable/core @bable/plugin-proposal-decorators -D
  - 注意：必须使用exclude 指定排除项；node_nodules 第三方包不需要打包

```
{test: /\.js$/,use:'bable-loader',exclude:/node_modules}
```

- 配置: 创建babel.config.js文件配置如下

```
module.exports={
  //将来，webpack在调用 bable-loader是 回加载plugins中的插件
    plugins:[[' @bable/plugin-proposal-decorators'],{legacy:true}]
}
```

打包发布

- 在package.json中scripts节点下新增build命令

```
"bulid":"webpack --mode production"
```

- clean-webpack-plugin 打包时自动删除旧的dist 重新新建
  - npm install --save-dev clean-webpack-plugin
  - 用法

```
//在webpack.config.js的引入
const { CleanWebpackPlugin } = require('clean-webpack-plugin');
//在webpack.config.js的plugins数组中
//添加
    new CleanWebpackPlugin(),


```

Source Map

- 信息文件-存着位置信息 解决代码混淆后 还能记得转换前的位置
- 开发环境下后 默认显示生成后的代码位置 不显示转换前的位置
- 在webpack.config.中添加： devtool:'eval-source-map' 与mdoe同级 运行时报错的行数与源代码行数保持一直
- 发布时 为了安全考虑 则需关闭Source Map 删除 devtool配置项
- 不暴露源码 有需要 报错行号 devtool:'nosources-source-map'

拓展

- 告诉程序员 @代表src 目录 vue默认配置了此项

```
resolve: {
    alias: {
        '@':path.join(__dirname,'./src/')
    }
}
```

注意：在实际开发中无需自己开发webpack 一些cli工具一键生成了webpack