**Rollup基础配置篇**

# 一、Rollup简介

Rollup 是一个 JavaScript 模块打包器，可以将小块代码编译成大块复杂的代码，例如 library 或应用程序。

Rollup 对代码模块使用新的标准化格式，这些标准都包含在 JavaScript 的 ES6 版本中，而不是以前的特殊解决方案，如 CommonJS 和 AMD。ES6 模块可以使你自由、无缝地使用你最喜爱的 library 中那些最有用独立函数，而你的项目不必携带其他未使用的代码。ES6 模块最终还是要由浏览器原生实现，但当前 Rollup 可以使你提前体验。

rollup生成代码只是把我们的代码转码成目标js并无其他,同时如果需要,他可以同时帮我们生成支持umd/commonjs/es的js代码,vue/react/angular都在用他作为打包工具。

# 二、主流打包工具

目前主流的打包工具有：grunt，gulp，webpack，rollup。

**Grunt：**最老牌的打包工具，它运用配置的思想来打包脚本，一切皆配置，所以会出现比较多的配置项，诸如option，src， dest等等。而且不同的插件可能会有自己扩展字段，认知成本高，运用的时候需要明白各种插件的配置规则。

**Gulp：**用代码方式来写打包脚本，并且代码采用流式的写法，只抽象出了gulp.src，gulp.pipe，gulp.dest，gulp.watch接口，运用相当简单。更易于学习和使用，使用gulp的代码量能比grunt少一半左右。

**Webpack：**是模块化管理工具和打包工具。通过loader的转换，任何形式的资源都可以视为模块，比如CommonJs模块、AMD模块、ES6模块、CSS、图片等。它可以将许多松散的模块按照依赖和规则打包成符合生产环境部署的前端资源。还可以将按需加载的模块进行代码分隔，等到实际需要的时候再异步加载。它定位是模块打包器，而Gulp/Grunt属于构建工具。Webpack可以代替Gulp/Grunt的一些功能，但不是一个职能的工具，可以配合使用。

**Rollup：**下一代ES6模块化工具，最大的亮点是利用ES6模块设计，利用tree-shaking生成更简洁、更简单的代码。一般而言，对于应用使用Webpack，对于类库使用Rollup；需要代码拆分（Code Splitting），或者很多静态资源需要处理，再或者构建的项目需要引入很多CommonJS模块的依赖时，使用webpack。代码库是基于ES6模块，而且希望代码能够被其他人直接使用，使用Rollup。

# Rollup的特点

1. 使用tree-shaking

tree-shaking可以理解为通过工具"摇"我们的JS文件，将其中用不到的代码"摇"掉，是一个性能优化的范畴。具体来说，项目中，有一个入口文件，相当于一棵树的主干，入口文件有很多依赖的模块，相当于树枝。实际情况中，虽然依赖了某个模块，但其实只使用其中的某些功能。

通过 tree-shaking，将没有使用的模块摇掉，这样来达到删除无用代码的目的。

1. 可以一次输出多种格式

IIFE, AMD, CJS, UMD, ESM

1. Rollup打包后生成的bundle内容十分`干净`，没有什么多余的代码，只是将各个模块按照依赖顺序拼接起来，所有模块构建在一个函数作用域内（Scope Hoisting）, 执行效率更高。

# 三、快速开始

## 1. 本地安装

npm install rollup --save-dev

## 2. 配置文件

创建rollup.config.js文件。配置文件支持下面列出的选项。

// rollup.config.js

export default { // 可以是数组 (用于多个输入)

// 核心输入选项

external,

input, // 必须项

plugins,

// 高级输入选项

cache,

onwarn,

preserveEntrySignatures,

strictDeprecations,

// 危险选项

acorn,

acornInjectPlugins,

context,

moduleContext,

preserveSymlinks,

shimMissingExports,

treeshake,

// 实验性的选项

experimentalCacheExpiry,

perf,

output: { // 必须项(可以是数组， 用于多个输出)

// 核心输出选项

dir,

file,

format, // 必须

globals,

name,

plugins,

// 高级输出选项

assetFileNames,

banner,

chunkFileNames,

compact,

entryFileNames,

extend,

footer,

hoistTransitiveImports,

inlineDynamicImports,

interop,

intro,

manualChunks,

minifyInternalExports,

outro,

paths,

preserveModules,

preserveModulesRoot,

sourcemap,

sourcemapExcludeSources,

sourcemapFile,

sourcemapPathTransform,

validate,

// 危险选项

amd,

esModule,

exports,

externalLiveBindings,

freeze,

indent,

namespaceToStringTag,

noConflict,

preferConst,

strict,

systemNullSetters

},

watch: {

buildDelay,

chokidar,

clearScreen,

skipWrite,

exclude,

include

} | false

};

要使用相同的输入构建不同的包，可以为每个输入提供一组输出选项。

## 编写要打包的文件

### （1）新建src文件夹，并新建应用程序入口main.js

### （2）新建modules文件夹（模块文件）

## 4. 配置package.json中的打包命令

## 5. 执行npm run build 查看输出文件

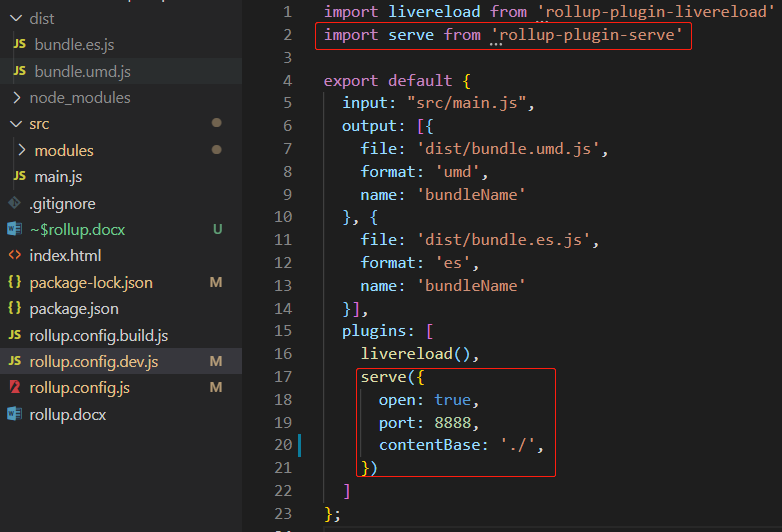
## 6. 使用打包后的文件

## 7. 开启本地的服务器

实际的开发过程中，需要运行代码，查看页面，方便调试代码

### 安装rollup-plugin-serve --save

### 配置rollup.config.js



## 8. 开启热更新

### （1）安装rollup-plugin-livereload

cnpm install rollup-plugin-livereload --save

### （2）配置rollup.config.js

