Webpack 执行过程

1、查找 webpack 入口文件

实际入口文件 node_modules\webpack\bin\webpack.js

局部安装：在 package.json 中 bin 字段去提供的命令

2、webpack.js 中执行代码

查找本地依赖：webpack-cli or webpack-command

webpack-cli 具有全部特性，webpack-command 是简易版

启动后，webpack 最终找到 webpack-cli 或 webpack-command 其中之一

并且执行 CLI

- 关于命令行工具包 yargs

webpack-cli 会从 webpack.config.js 与命令行中生成 options

然后用 processOptions 处理 options

每个插件会接收 compiler 对象方法 apply

会根据 options 实例化 webpack 对象，然后执行构建流程

生成一个 compiler 对象

关于 webpack-cli 源码

① 引入 yargs，对命令行进行定制

② 分析命令行参数，对各个参数进行转换，组成编译配置项 → 输出配置项

③ 引用 webpack，根据配置项进行编译和构建

1、用 NON_COMPILATION_CMD 分析不需要编译的命令

↳ 默认声明了 8 个命令 prompt-command 直接执行

@webpack-cli/ + package → 包名直接做命令

未找到会询问是否下载

2、NON_COMPILATION_ARGS 的内容

webpack-cli 提供的不需要编译的命令

```
const NON_COMPILATION_ARGS = [
  "init",      // 创建一份 webpack 配置文件
  "migrate",   // 进行 webpack 版本迁移
  "add",       // 向 webpack 配置文件增加属性
  "remove",    // 向 webpack 配置文件删除属性
  "serve",     // 运行 webpack-serve
  "generate-loader", // 生成 webpack-loader 代码
  "generate-plugin", // 生成 webpack-plugin 代码
  "info"       // 返回与本地环境相关的一些信息
```

参数与插件

- 关于 webpack 核心：Tapable → 类似于 node 中的 eventEmiter

Compiler 类是继承 Tapable，HOOKS 由 Tapable 提供

webpack 可以理解一批基于事件流的编程范例，一系列的插件运行

核心 compiler 与 compilation 均继承 Tapable。

hook 绑定事件监听

提供钩子函数的发布与订阅

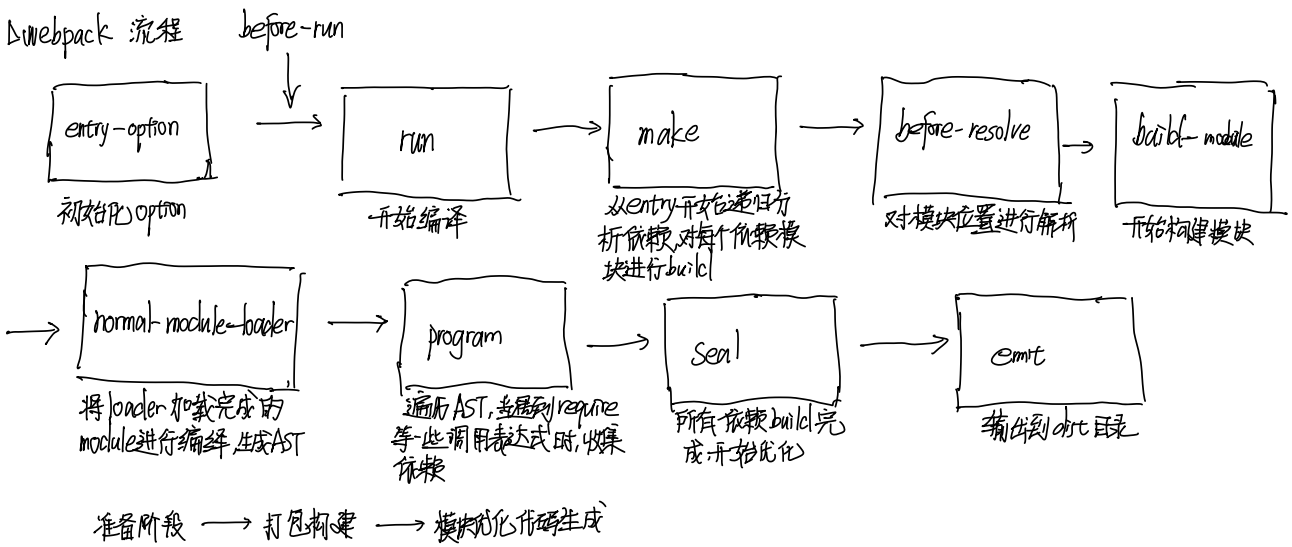
插件系统

每个插件有个 apply 方法，插件监听 compiler 对象上的 Hooks

Tapable hooks 类型

HOOK	所有钩子的后缀
Waterfall	同步方法，会传值给下个函数
Bail	熔断：函数有返回值就会停止
Loop	监听函数返回 true 继续循环 返回 undefined 表示结束循环
Sync	同步方法
AsyncSeries	异步串行钩子
Async Parallel	异步并行执行钩子

webpack 流程



准备阶段 → 打包构建 → 模块优化 代码生成

1、准备阶段 WebpackOptionsDefaulter

BeforeRun：NodeEnvironmentPlugin → 清理缓存

compiler.option = webpackOptionsApply →

将所有配置options参数转换为webpack内部插件

- output.library ⟶ LibraryTemplatePlugin
- externals ⟶ ExternalsPlugin ⟶ 会引入nodejs模块
- devtool ⟶ EvalDevtoolModulePlugin，SourceMapDevToolPlugin
- AMDPlugin，CommonJS Plugin
- RemoveEmptyChunksPlugin

EntryOptionPlugin 处理entry ⟹ 转化为入口

关于compiler

2、模块构建与chunk生成阶段

Compiler hook

流程相关

- before-run
- (before-/after-) compile
- make
  - done

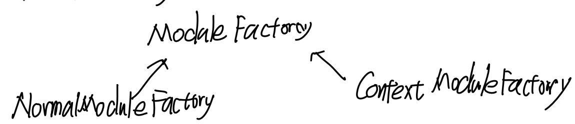
监听相关 (watch)

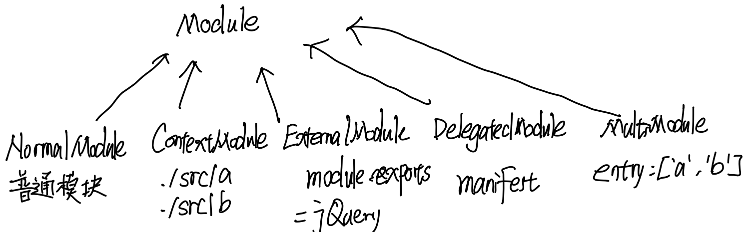
- watch-run
- watch-close

Compiler调用Compilation生命周期

- add Entry
- finish
- seal (输出&生成)

ModuleFactory





NormalModule

Build

- 使用loader-runner运行loaders
- 通过Parser解析
- ParserPlugin 增加依赖