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=201 sentences=416, words=2546

1. 初始化项目

```
doskey /history
mkdir webpack-source
cd webpack-source
npm init -y
cnpm i webpack webpack-cli -D
cnpm i lodash -S
npx webpack
```

2.配置文件

2.1 src\index.js

src\index.js

```
import _ from 'lodash';
console.log(_.isArray([]));
```

2.2 webpack.config.js

webpack.config.js

```
const path = require('path');
module.exports = {
    mode:'development',
    entry: './src/index.js',
    output: {
        path:path.join(__dirname,'dist'),
        filename: 'bundle.js'
    }
};
```

3.可执行命令

- cli (https://webpack.js.org/api/cli/)

3.1 直接调用

- 可以直接打开 node_modules\webpack-cli\bin\cli.js文件
- 然后直接打开调试窗口，然后点击启动编译就可以了

3.2 如何生成调试文件

- 打开工程目录，点击调试按钮，再点击小齿轮的配置按钮系统就会生成launch.json配置文件
- 修改好了以后直接点击F5就可以启动调试

.vscode\launch.json

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "type": "node",
            "request": "launch",
            "name": "debug webpack",
            "cwd": "${workspaceFolder}",
            "program": "${workspaceFolder}/node_modules/_webpack-cli@3.3.10@webpack-cli/bin/cli.js"
        }
    ]
}
```

3.2 webpack.cmd

webpack-source\node_modules.bin\webpack.cmd

- %~dp0是批处理文件所在的盘符:+路径(%~dp0 C:\vipdata\vipproject\webpack-source\node_modules.bin)
- SETLOCAL主要针对临时环境变量，不会影响到系统的变量环境设置，应与endlocal联用
- PATHEXT当在一个相同的目录结构下，有相同的多个主文件名，不同的文件后缀名时，系统会根据PATHEXT中的后缀名，选择其中顺序最靠前的那一个

```
@IF EXIST "%~dp0\node.exe" (
"%~dp0\node.exe"  "%~dp0\..\_webpack@4.39.3@webpack\bin\webpack.js" %*
) ELSE (
@SETLOCAL
@SET PATHEXT=%PATHEXT:;.JS;=;%
node  "%~dp0\..\_webpack@4.39.3@webpack\bin\webpack.js" %*
)
```

3.3 webpack.js

node_modules\webpack\bin\webpack.js

```
const path = require("path");
const pkgPath = require.resolve(`${installedClis[0].package}/package.json`);

const pkg = require(pkgPath);

require(path.resolve(
    path.dirname(pkgPath),
    pkg.bin[installedClis[0].binName]
));
```

```
const path = require("path");
const pkgPath = require.resolve(`webpack-cli/package.json`);
const pkg = require(pkgPath);
require(path.resolve(path.dirname(pkgPath),pkg.bin['webpack-cli']));
```

- npx webpack就相当于用node执行cli.js脚本

```
npx webpack = node ./node_modules/webpack-cli/bin/cli.js
```

3.4 阅读思路

- 先折叠无关的分支的逻辑，只看主体流程代码
- 寻找关键路径，根据变量名和方法名猜测意图，然后通过阅读源码来验证想法
- debugger关键路径，理解整个执行过程

3.5 重要文件

- Webpack 类似于一个公司
- Compiler 类似于公司的董事长，只把握战略方向
- Compilation 就像各个事业部的总监,负责各个部门的管理
- ModuleFactory 就像各个具体部门了,负责具体工作产出产品

4.启动编译

- Compiler和Compilation都继承自Tapable
- Compiler是每个Webpack配置对应一个Compiler对象，记录着Webpack的生命周期
- 在构建的过程中，每次构建都会产生一个Compilation,Compilation是构建周期的产物
- Compiler模块是Webpack最核心的模块
- 每次执行构建的时候，都会先实例化一个 Compiler对象，然后调用它的 `run`方法来开启一次完整的编译

说明 代码 1.读取配置文件
requireConfig (https://github.com/webpack/webpack-cli/blob/v3.3.7/bin/utils/convert-argv.js#L125)

2.创建compiler
webpack(options) (https://github.com/webpack/webpack-cli/blob/v3.3.7/bin/cli.js#L272)

3.开始编译
compiler.run (https://github.com/webpack/webpack-cli/blob/v3.3.7/bin/cli.js#L352)

```
node debug.js
```

debug.js

```
const webpack = require("webpack");
const config = require('./webpack.config.js');
const compiler = webpack(config);
function compilerCallback(err, stats) {
    const statsString = stats.toString();
    console.log(statsString);
}
compiler.run((err,stats)=>{
    compilerCallback(err, stats);
});
```

5.钩子

5.1 Compiler

- L106

```
            entryOption: new SyncBailHook(["context", "entry"])
        };
+        Object.keys(this.hooks).forEach(hookName => {
+            const hook = this.hooks[hookName];
+            if (hook.tap) {
+                hook.tap('flow', () => {
+                    console.log(`|compiler|${hookName}|${Object.getPrototypeOf(hook).constructor.name}|${hook._args}|`);
+                });
+            }
+        });
```

5.2 Compilation

- L437

```
            afterOptimizeExtractedChunks: new SyncHook(["chunks"])
        };
+        Object.keys(this.hooks).forEach(hookName => {
+            const hook = this.hooks[hookName];
+            if (hook.tap) {
+                hook.tap('flow', () => {
+                    console.log(`|compilation|${hookName}|${Object.getPrototypeOf(hook).constructor.name}|${hook._args}|`);
+                });
+            }
+        });
        this._pluginCompat.tap("Compilation", options => {
```

```
|compiler|environment|SyncHook||
|compiler|afterEnvironment|SyncHook||
|compiler|entryOption|SyncBailHook|context,entry|
|compiler|afterPlugins|SyncHook|compiler|
|compiler|afterResolvers|SyncHook|compiler|
|compiler|beforeRun|AsyncSeriesHook|compiler|
|compiler|run|AsyncSeriesHook|compiler|
|compiler|normalModuleFactory|SyncHook|normalModuleFactory|
|compiler|contextModuleFactory|SyncHook|contextModulefactory|
|compiler|beforeCompile|AsyncSeriesHook|params|
|compiler|compile|SyncHook|params|
|compiler|thisCompilation|SyncHook|compilation,params|
|compiler|compilation|SyncHook|compilation,params|
|compiler|make|AsyncParallelHook|compilation|
|compilation|addEntry|SyncHook|entry,name|
|compilation|buildModule|SyncHook|module|
|compilation|normalModuleLoader|SyncHook|loaderContext,module|
|compilation|succeedModule|SyncHook|module|
|compilation|buildModule|SyncHook|module|
|compilation|normalModuleLoader|SyncHook|loaderContext,module|
|compilation|succeedModule|SyncHook|module|
|compilation|buildModule|SyncHook|module|
|compilation|normalModuleLoader|SyncHook|loaderContext,module|
|compilation|buildModule|SyncHook|module|
|compilation|normalModuleLoader|SyncHook|loaderContext,module|
|compilation|succeedModule|SyncHook|module|
|compilation|succeedModule|SyncHook|module|
|compilation|succeedEntry|SyncHook|entry,name,module|
|compilation|finishModules|AsyncSeriesHook|modules|
|compilation|seal|SyncHook||
|compilation|optimizeDependenciesBasic|SyncBailHook|modules|
|compilation|optimizeDependencies|SyncBailHook|modules|
|compilation|optimizeDependenciesAdvanced|SyncBailHook|modules|
|compilation|afterOptimizeDependencies|SyncHook|modules|
|compilation|beforeChunks|SyncHook||
|compilation|dependencyReference|SyncWaterfallHook|dependencyReference,d
|compilation|dependencyReference|SyncWaterfallHook|dependencyReference,d
|compilation|dependencyReference|SyncWaterfallHook|dependencyReference,d
|compilation|dependencyReference|SyncWaterfallHook|dependencyReference,d
|compilation|log|SyncBailHook|origin,logEntry|
|compilation|log|SyncBailHook|origin,logEntry|
|compilation|afterChunks|SyncHook|chunks|
|compilation|optimize|SyncHook||
|compilation|optimizeModulesBasic|SyncBailHook|modules|
|compilation|optimizeModules|SyncBailHook|modules|
|compilation|optimizeModulesAdvanced|SyncBailHook|modules|
|compilation|afterOptimizeModules|SyncHook|modules|
|compilation|optimizeChunksBasic|SyncBailHook|chunks,chunkGroups|
|compilation|optimizeChunks|SyncBailHook|chunks,chunkGroups|
|compilation|optimizeChunksAdvanced|SyncBailHook|chunks,chunkGroups|
|compilation|afterOptimizeChunks|SyncHook|chunks,chunkGroups|
|compilation|optimizeTree|AsyncSeriesHook|chunks,modules|
|compilation|afterOptimizeTree|SyncHook|chunks,modules|
|compilation|optimizeChunkModulesBasic|SyncBailHook|chunks,modules|
|compilation|optimizeChunkModules|SyncBailHook|chunks,modules|
|compilation|optimizeChunkModulesAdvanced|SyncBailHook|chunks,modules|
|compilation|afterOptimizeChunkModules|SyncHook|chunks,modules|
|compilation|shouldRecord|SyncBailHook||
|compilation|reviveModules|SyncHook|modules,records|
|compilation|optimizeModuleOrder|SyncHook|modules|
|compilation|advancedOptimizeModuleOrder|SyncHook|modules|
|compilation|beforeModuleIds|SyncHook|modules|
|compilation|moduleIds|SyncHook|modules|
|compilation|optimizeModuleIds|SyncHook|modules|
|compilation|afterOptimizeModuleIds|SyncHook|modules|
|compilation|reviveChunks|SyncHook|chunks,records|
|compilation|optimizeChunkOrder|SyncHook|chunks|
|compilation|beforeChunkIds|SyncHook|chunks|
|compilation|optimizeChunkIds|SyncHook|chunks|
|compilation|afterOptimizeChunkIds|SyncHook|chunks|
|compilation|recordModules|SyncHook|modules,records|
|compilation|recordChunks|SyncHook|chunks,records|
|compilation|beforeHash|SyncHook||
|compilation|chunkHash|SyncHook|chunk,chunkHash|
|compilation|contentHash|SyncHook|chunk|
|compilation|afterHash|SyncHook||
|compilation|recordHash|SyncHook|records|
|compilation|beforeModuleAssets|SyncHook||
|compilation|shouldGenerateChunkAssets|SyncBailHook||
|compilation|beforeChunkAssets|SyncHook||
|compilation|chunkAsset|SyncHook|chunk,filename|
|compilation|additionalChunkAssets|SyncHook|chunks|
|compilation|record|SyncHook|compilation,records|
|compilation|additionalAssets|AsyncSeriesHook||
|compilation|optimizeChunkAssets|AsyncSeriesHook|chunks|
|compilation|afterOptimizeChunkAssets|SyncHook|chunks|
|compilation|optimizeAssets|AsyncSeriesHook|assets|
|compilation|afterOptimizeAssets|SyncHook|assets|
|compilation|needAdditionalSeal|SyncBailHook||
|compilation|afterSeal|AsyncSeriesHook||
|compiler|afterCompile|AsyncSeriesHook|compilation|
|compiler|shouldEmit|SyncBailHook|compilation|
|compiler|emit|AsyncSeriesHook|compilation|
|compiler|assetEmitted|AsyncSeriesHook|file,content|
|compiler|afterEmit|AsyncSeriesHook|compilation|
|compilation|needAdditionalPass|SyncBailHook||
|compiler|done|AsyncSeriesHook|stats|
```

类型 事件名称 类型 参数 说明 发射事件代码 对应插件 compiler environment SyncHook 空 准备编译环境，webpack plugins配置初始化完成之后 [webpack.js#L55 (https://github.com/webpack/webpack/blob/v4.39.3/lib/webpack.js#L55)](https://github.com/webpack/webpack/blob/v4.39.3/lib/webpack.js#L55) [NodeEnvironmentPlugin (https://github.com/webpack/webpack/blob/v4.39.3/lib/node/NodeEnvironmentPlugin.js)](https://github.com/webpack/webpack/blob/v4.39.3/lib/node/NodeEnvironmentPlugin.js)

compiler afterEnvironment SyncHook 空 编译环境准备好之后 [webpack.js#L56 (https://github.com/webpack/webpack/blob/v4.39.3/lib/webpack.js#L56)](https://github.com/webpack/webpack/blob/v4.39.3/lib/webpack.js#L56)

compiler entryOption SyncBailHook context,entry 在 webpack 中的 entry 配置处理过之后
WebpackOptionsApply.js#L291 (https://github.com/webpack/webpack/blob/v4.39.3/lib/WebpackOptionsApply.js#L291) EntryOptionPlugin (https://github.com/webpack/webpack/blob/v4.39.3/lib/EntryOptionPlugin.js) SingleEntryPlugin (https://github.com/webpack/webpack/blob/v4.39.3/lib/SingleEntryPlugin.js)

compiler afterPlugins SyncHook compiler 初始化完内置插件之后
WebpackOptionsApply.js#L506 (https://github.com/webpack/webpack/blob/v4.39.3/lib/WebpackOptionsApply.js#L506)

compiler afterResolvers SyncHook compiler resolver 完成之后
WebpackOptionsApply.js#L541 (https://github.com/webpack/webpack/blob/v4.39.3/lib/WebpackOptionsApply.js#L541)

compiler beforeRun AsyncSeriesHook compiler 开始正式编译之前
Compiler.js#L312 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L312)

compiler run AsyncSeriesHook compiler 开始编译之后，读取 records 之前；监听模式触发watch-run
Compiler.js#L315 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L315)

compiler normalModuleFactory SyncHook normalModuleFactory NormalModuleFactory 创建之后
Compiler.js#L631 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L631)

compiler contextModuleFactory SyncHook contextModulefactory ContextModuleFactory 创建之后
Compiler.js#L631 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L637)

compiler beforeCompile AsyncSeriesHook params compilation 实例化需要的参数创建完毕之后
Compiler.js#L652 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L652)

compiler compile SyncHook params 一次 compilation 编译创建之前
Compiler.js#L652 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L652)

compiler thisCompilation SyncHook compilation,params 触发 compilation 事件之前执行
Compiler.js#L620 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L620)

compiler compilation SyncHook compilation,params compilation创建成功之后
Compiler.js#L620 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L620)

compiler make AsyncParallelHook compilation 完成编译之前
Compiler.js#L659 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L659) SingleEntryPlugin (https://github.com/webpack/webpack/blob/v4.39.3/lib/SingleEntryPlugin.js)

compilation addEntry SyncHook entry,name 增加入口
Compilation.js#L1106 (https://github.com/webpack/webpack/blob/master/lib/Compilation.js#L1106)

compilation buildModule SyncHook module 在模块构建开始之前触发
Compilation.js#L701 (https://github.com/webpack/webpack/blob/master/lib/Compilation.js#L701)

compilation normalModuleLoader SyncHook loaderContext,module 普通模块 loader，真正（一个接一个地）加载模块图（graph）中所有模块的函数
NormalModule.js#L233 (https://github.com/webpack/webpack/blob/master/lib/NormalModule.js#L233)

compilation succeedModule SyncHook module 模块构建成功时执行
Compilation.js#744 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#744)

compilation succeedEntry SyncHook entry,name,module
Compilation.js#L1147 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1147)

compilation finishModules AsyncSeriesHook modules 所有模块都完成构建
Compilation.js#L1216 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1216)

compilation seal SyncHook 编译（compilation）停止接收新模块时触发
Compilation.js#L1246 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1246)

compilation optimizeDependenciesBasic SyncBailHook modules 基础依赖优化开始时触发
Compilation.js#L1249 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1249)

compilation optimizeDependencies SyncBailHook modules 依赖优化开始时触发
Compilation.js#L1250 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1250)

compilation optimizeDependenciesAdvanced SyncBailHook modules 高级依赖优化开始时触发
Compilation.js#L1251 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1251)

compilation afterOptimizeDependencies SyncHook modules 优化结束
Compilation.js#L1255 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1255)

compilation beforeChunks SyncHook 开始生成代码块
Compilation.js#L1257 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1257)

compilation dependencyReference SyncWaterfallHook dependencyReference,dependency,module 依赖引用
Compilation.js#L1565 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1565)

compilation log SyncBailHook origin,logEntry 打印日志
Compilation.js#L542 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L542)

compilation afterChunks SyncHook chunks 代码块生成之后
Compilation.js#L1282 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1282)

compilation optimize SyncHook 优化阶段开始时触发
Compilation.js#L1284 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1284)

compilation optimizeModulesBasic SyncBailHook modules 基础模块的优化
Compilation.js#L1284 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1287)

compilation optimizeModules SyncBailHook modules 模块的优化
Compilation.js#L1284 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1288)

compilation optimizeModulesAdvanced SyncBailHook modules 高级模块的优化
Compilation.js#L1284 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1289)

compilation afterOptimizeModules SyncHook modules 模块优化结束时触发
Compilation.js#L1293 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1293)

compilation optimizeChunksBasic SyncBailHook chunks,chunkGroups 基础chunk优化
Compilation.js#L1296 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1287)

compilation optimizeChunks SyncBailHook chunks,chunkGroups 优化 chunks
Compilation.js#L1297 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1297)

compilation optimizeChunksAdvanced SyncBailHook chunks,chunkGroups 高级chunk优化
Compilation.js#L1298 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1298)

compilation afterOptimizeChunks SyncHook chunks,chunkGroups chunk 优化完成之后触发
Compilation.js#L1302 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1302)

compilation optimizeTree AsyncSeriesHook chunks,modules 异步优化依赖树
Compilation.js#L1304 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1304)

compilation afterOptimizeTree SyncHook chunks,modules 异步优化依赖树完成时
Compilation.js#L1309 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1309)

compilation optimizeChunkModulesBasic SyncBailHook chunks,modules 基础优化单个chunk中的 modules 开始
[Compilation.js#L1312 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1312)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1312)

compilation optimizeChunkModules SyncBailHook chunks,modules 优化单个chunk中的 modules 开始
[Compilation.js#L1313 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1313)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1313)

compilation optimizeChunkModulesAdvanced SyncBailHook chunks,modules 高级优化单个chunk中的 modules 开始
[Compilation.js#L1314 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1314)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1314)

compilation afterOptimizeChunkModules SyncHook chunks,modules 优化单个chunk中的 modules结束后
[Compilation.js#L1318 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1318)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1318)

compilation shouldRecord SyncBailHook 是否应该记录
[Compilation.js#L1320 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1320)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1320)

compilation reviveModules SyncHook modules,records 从 records 中恢复模块信息
[Compilation.js#L1322 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1322)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1322)

compilation optimizeModuleOrder SyncHook modules 将模块从最重要的到最不重要的进行排序
[Compilation.js#L1323 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1323)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1323)

compilation advancedOptimizeModuleOrder SyncHook modules 高级将模块从最重要的到最不重要的进行排序
[Compilation.js#L1324 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1324)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1324)

compilation beforeModuleIds SyncHook modules 处理 modulesId 之前
[Compilation.js#L1325 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1325)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1325)

compilation moduleIds SyncHook modules 处理 modulesId
[Compilation.js#L1326 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1326)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1326)

compilation optimizeModuleIds SyncHook modules 优化 modulesId
[Compilation.js#L1328 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1328)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1328)

compilation afterOptimizeModuleIds SyncHook modules 优化 modulesId之后
[Compilation.js#L1329 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1329)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1329)

compilation reviveChunks SyncHook chunks,records 从 records 中恢复 chunk 信息
[Compilation.js#L1333 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1333)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1333)

compilation optimizeChunkOrder SyncHook chunks 将 chunk 从最重要的到最不重要的进行排序
[Compilation.js#L1334 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1334)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1334)

compilation beforeChunkIds SyncHook chunks chunk id 优化之前触发
[Compilation.js#L1335 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1335)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1335)

compilation optimizeChunkIds SyncHook chunks chunk id 优化开始触发
[Compilation.js#L1337 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1337)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1337)

compilation afterOptimizeChunkIds SyncHook chunks chunk id 优化结束触发
[Compilation.js#L1338 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1338)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1338)

compilation recordModules SyncHook modules,records 将模块信息存储到 records
[Compilation.js#L1343 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1343)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1343)

compilation recordChunks SyncHook chunks,records 将 chunk 信息存储到 records
[Compilation.js#L1344 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1344)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1344)

compilation beforeHash SyncHook 在编译被哈希（hashed）之前
[Compilation.js#L1347 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1347)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1347)

compilation chunkHash SyncHook chunk,chunkHash 生成chunkHash
[Compilation.js#L1937 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1937)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1937)

compilation contentHash SyncHook chunk 生成contentHash
[Compilation.js#L1941 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1941)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1941)

compilation afterHash SyncHook 在编译被哈希（hashed）之后
[Compilation.js#L1349 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1349)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1349)

compilation recordHash SyncHook records 记录hash
[Compilation.js#L1352 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1352)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1352)

compilation beforeModuleAssets SyncHook 在创建模块的资源之前
[Compilation.js#L1355 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1355)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1355)

compilation shouldGenerateChunkAssets SyncBailHook 是否要生成chunk资源
[Compilation.js#L1357 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1357)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1357)

compilation beforeChunkAssets SyncHook 在创建 chunk 资源（asset）之前
[Compilation.js#L1358 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1358)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1358)

compilation chunkAsset SyncHook chunk,filename 一个 chunk 中的一个资源被添加到编译中
[Compilation.js#L2019 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L2019)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L2019)

compilation additionalChunkAssets SyncHook chunks additionalChunkAssets
[Compilation.js#L1361 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1361)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1361)

compilation record SyncHook compilation,records 将 compilation 相关信息存储到 records 中
[Compilation.js#L1364 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1364)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1364)

compilation additionalAssets AsyncSeriesHook 为编译（compilation）创建附加资源（asset）
[Compilation.js#L1367 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1367)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1367)

compilation optimizeChunkAssets AsyncSeriesHook compilation 优化所有 chunk 资源（asset）
[Compilation.js#L1343 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1343)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1343)

compilation afterOptimizeChunkAssets SyncHook chunks chunk 资源（asset）已经被优化
[Compilation.js#L1371 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1343)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1343)

compilation optimizeAssets AsyncSeriesHook assets 优化存储在 compilation.assets 中的所有资源（asset）
[Compilation.js#L1371 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1343)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1343)

compilation afterOptimizeAssets SyncHook assets 优化compilation.assets 中的所有资源（asset）之后
[Compilation.js#L1380 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1380)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1380)

compilation needAdditionalSeal SyncBailHook 是否需要额外的seal
[Compilation.js#L1381 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1381)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1381)

compilation afterSeal AsyncSeriesHook seal之后
[Compilation.js#L1383 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1383)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js#L1383)

compiler afterCompile AsyncSeriesHook compilation 完成编译和封存（seal）编译产出之后
[Compiler.js#L668 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L668)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L668)

compiler shouldEmit SyncBailHook compilation 发布构建后资源之前触发，回调必须返回true/false，true则继续
[Compiler.js#L267 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L267)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L267)

compiler emit AsyncSeriesHook compilation 生成资源到 output 目录之前
[Compiler.js#L481 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L481)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L481)

compiler assetEmitted AsyncSeriesHook file,content assetEmitted
[Compiler.js#L437 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L437)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L437)

compiler afterEmit AsyncSeriesHook compilation 生成资源到 output 目录之后
[Compiler.js#L472 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L472)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L472)

compilation needAdditionalPass SyncBailHook 是否需要额外的
[Compiler.js#L281 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L281)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L281)

compiler done AsyncSeriesHook stats compilation完成之后
[Compiler.js#L304 (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L304)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js#L304)

5.Stats 对象

- 在 Webpack 的回调函数中会得到stats对象
- 这个对象实际来自于 `Compilation.getStats()`，返回的是主要含有 `modules`、`chunks`和 `assets`三个属性值的对象。
- Stats对象本质上来自于[lib/Stats.js (https://github.com/webpack/webpack/blob/v4.39.3/lib/Stats.js)](https://github.com/webpack/webpack/blob/v4.39.3/lib/Stats.js)的类实例

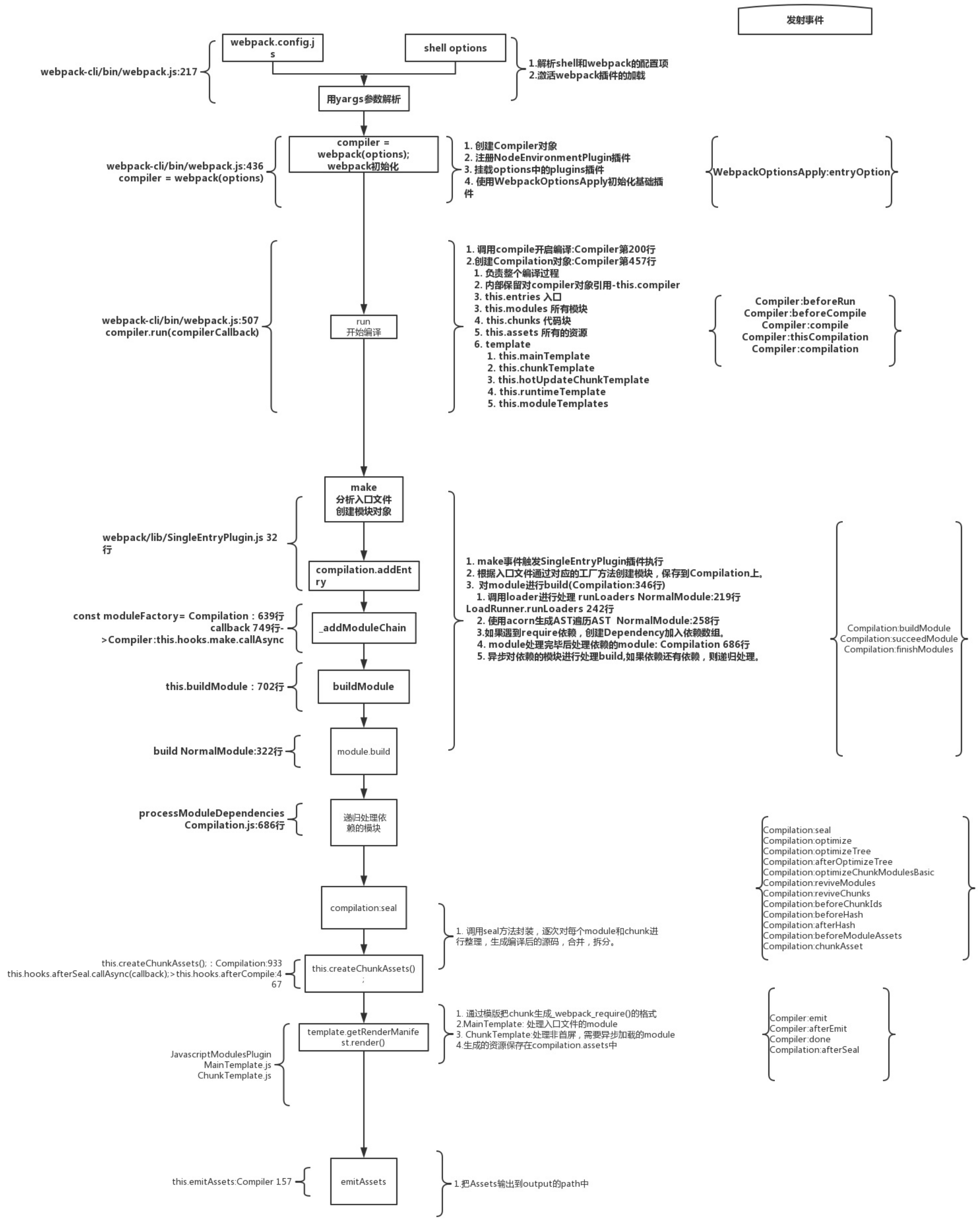
字段 含义 modules 记录了所有解析后的模块 chunks 记录了所有chunk assets 记录了所有要生成的文件

```
npx webpack --profile --json > stats.json
```

```
{
  "errors": [],
  "warnings": [],
  "version": "4.39.3",
  "hash": "3e945ec6b2c56d0b010e",
  "time": 66,
  "builtAt": 1567225465347,
  "publicPath": "",
  "outputPath": "C:\\vipdata\\vipproject\\webpack-source\\dist",
  "assetsByChunkName": {
    "lazy": "lazy.bundle.js",
    "main": "bundle.js"
  },
  "assets": [
    {
      "name": "bundle.js",
      "size": 9043,
      "chunks": [
        "main"
      ],
      "chunkNames": [
        "main"
      ],
      "emitted": true
    },
    {
      "name": "lazy.bundle.js",
      "size": 336,
      "chunks": [
        "lazy"
      ],
      "chunkNames": [
        "lazy"
      ],
      "emitted": true
    }
  ],
  "filteredAssets": 0,
  "entrypoints": {
    "main": {
      "chunks": [
        "main"
      ],
      "assets": [
        "bundle.js"
      ],
      "children": {},
      "childAssets": {}
    }
  },
  "namedChunkGroups": {
    "main": {
      "chunks": [
        "main"
      ],
      "assets": [
        "bundle.js"
      ],
      "children": {},
      "childAssets": {}
    },
    "lazy": {
      "chunks": [
        "lazy"
      ],
      "assets": [
        "lazy.bundle.js"
      ],
      "children": {},
      "childAssets": {}
    }
  },
  "chunks": [
    {
      "id": "lazy",
      "rendered": true,
      "initial": false,
      "entry": false,
      "size": 24,
      "names": [
        "lazy"
      ],
      "files": [
        "lazy.bundle.js"
      ],
      "hash": "d08a8b502d30324f81e1",
      "siblings": [],
      "parents": [
        "main"
      ],
      "children": [],
      "childrenByOrder": {},
      "modules": [
        {
          "id": "./src/lazy.js",
          "identifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\lazy.js",
          "name": "./src/lazy.js",
          "index": 2,
          "index2": 2,
          "size": 24,
          "cacheable": true,
          "built": true,
          "optional": false,
          "prefetched": false,
          "chunks": [
            "lazy"
          ],
          "issuer": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
          "issuerId": "./src/index.js",
          "issuerName": "./src/index.js",
          "issuerPath": [
            {
              "id": "./src/index.js",
              "identifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
              "name": "./src/index.js",
              "profile": {
                "factory": 18,
                "building": 14
              }
            }
          ],
          "profile": {
            "factory": 4,
            "building": 2
          },
          "failed": false,
          "errors": 0,
          "warnings": 0,
          "assets": [],
          "reasons": [
            {
              "moduleId": "./src/index.js",
              "moduleIdentifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
              "module": "./src/index.js",
              "moduleName": "./src/index.js",
              "type": "import()",
              "userRequest": "./lazy",
              "loc": "2:0-46"
            }
          ],
          "providedExports": null,
          "optimizationBailout": [],
          "depth": 1,
          "source": "module.exports = 'lazy';"
        }
      ],
      "filteredModules": 0,
      "origins": [
        {
          "moduleId": "./src/index.js",
          "module": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
          "moduleIdentifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
          "moduleName": "./src/index.js",
          "loc": "2:0-46",
          "request": "./lazy",
          "reasons": []
        }
      ]
    },
    {
      "id": "main",
      "rendered": true,
      "initial": true,
      "entry": true,
      "size": 162,
      "names": [
        "main"
      ],
      "files": [
        "bundle.js"
      ],
      "hash": "263cadc0459e8470151b",
      "siblings": [],
      "parents": [],
      "children": [
        "lazy"
      ],
      "childrenByOrder": {},
      "modules": [
        {
          "id": "./src/hello.js",
          "identifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\hello.js",
          "name": "./src/hello.js",
          "index": 1,
          "index2": 0,
          "size": 25,
          "cacheable": true,
          "built": true,
          "optional": false,
          "prefetched": false,
          "chunks": [
            "main"
          ],
          "issuer": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
          "issuerId": "./src/index.js",
          "issuerName": "./src/index.js",
          "issuerPath": [
            {
              "id": "./src/index.js",
              "identifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
              "name": "./src/index.js",
              "profile": {
                "factory": 18,
                "building": 14
              }
            }
          ],
          "profile": {
            "factory": 4,
            "building": 2
          },
          "failed": false,
          "errors": 0,
          "warnings": 0,
          "assets": [],
          "reasons": [
            {
              "moduleId": "./src/index.js",
              "moduleIdentifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
              "module": "./src/index.js",
              "moduleName": "./src/index.js",
              "type": "cjs require",
              "userRequest": "./hello",
              "loc": "1:12-30"
            }
          ],
          "providedExports": null,
          "optimizationBailout": [],
          "depth": 1,
          "source": "module.exports = 'hello';"
        },
        {
          "id": "./src/index.js",
          "identifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
          "name": "./src/index.js",
          "index": 0,
          "index2": 1,
          "size": 137,
          "cacheable": true,
          "built": true,
          "optional": false,
          "prefetched": false,
          "chunks": [
            "main"
          ],
          "issuer": null,
          "issuerId": null,
          "issuerName": null,
          "issuerPath": null,
          "profile": {
            "factory": 18,
            "building": 14
          },
          "failed": false,
          "errors": 0,
          "warnings": 0,
          "assets": [],
          "reasons": [
            {
              "moduleId": null,
              "moduleIdentifier": null,
              "module": null,
              "moduleName": null,
              "type": "single entry",
              "userRequest": "./src/index.js",
              "loc": "main"
            }
          ],
          "providedExports": null,
          "optimizationBailout": [],
          "depth": 0,
          "source": "let hello = require('./hello');\r\nimport(/* webpackChunkName: \"lazy\" */'./lazy').then(result=>{\r\n
console.log(hello,resut.default)\r\n});"
        }
      ],
      "filteredModules": 0,
      "origins": [
        {
          "module": "",
          "moduleIdentifier": "",
          "moduleName": "",
          "loc": "main",
          "request": "./src/index.js",
          "reasons": []
        }
      ]
    }
  ],
  "modules": [
    {
      "id": "./src/hello.js",
      "identifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\hello.js",
      "name": "./src/hello.js",
      "index": 1,
      "index2": 0,
      "size": 25,
      "cacheable": true,
      "built": true,
      "optional": false,
      "prefetched": false,
      "chunks": [
        "main"
      ],
      "issuer": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
      "issuerId": "./src/index.js",
      "issuerName": "./src/index.js",
      "issuerPath": [
        {
          "id": "./src/index.js",
          "identifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
          "name": "./src/index.js",
          "profile": {
            "factory": 18,
            "building": 14
          }
        }
      ],
      "profile": {
        "factory": 4,
        "building": 2
      },
      "failed": false,
      "errors": 0,
      "warnings": 0,
      "assets": [],
      "reasons": [
        {
          "moduleId": "./src/index.js",
          "moduleIdentifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
          "module": "./src/index.js",
          "moduleName": "./src/index.js",
          "type": "cjs require",
          "userRequest": "./hello",
          "loc": "1:12-30"
        }
      ],
      "providedExports": null,
      "optimizationBailout": [],
      "depth": 1,
      "source": "module.exports = 'hello';"
    },
    {
      "id": "./src/index.js",
      "identifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
      "name": "./src/index.js",
      "index": 0,
      "index2": 1,
      "size": 137,
      "cacheable": true,
      "built": true,
      "optional": false,
      "prefetched": false,
      "chunks": [
        "main"
      ],
      "issuer": null,
      "issuerId": null,
      "issuerName": null,
      "issuerPath": null,
      "profile": {
        "factory": 18,
        "building": 14
      },
      "failed": false,
      "errors": 0,
      "warnings": 0,
      "assets": [],
      "reasons": [
        {
          "moduleId": null,
          "moduleIdentifier": null,
          "module": null,
          "moduleName": null,
          "type": "single entry",
          "userRequest": "./src/index.js",
          "loc": "main"
        }
      ],
      "providedExports": null,
      "optimizationBailout": [],
      "depth": 0,
      "source": "let hello = require('./hello');\r\nimport(/* webpackChunkName: \"lazy\" */'./lazy').then(result=>{\r\n
console.log(hello,resut.default)\r\n});"
    },
    {
      "id": "./src/lazy.js",
      "identifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\lazy.js",
      "name": "./src/lazy.js",
      "index": 2,
      "index2": 2,
      "size": 24,
      "cacheable": true,
      "built": true,
      "optional": false,
      "prefetched": false,
      "chunks": [
        "lazy"
      ],
      "issuer": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
      "issuerId": "./src/index.js",
      "issuerName": "./src/index.js",
      "issuerPath": [
        {
          "id": "./src/index.js",
          "identifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
          "name": "./src/index.js",
          "profile": {
            "factory": 18,
            "building": 14
          }
        }
      ],
      "profile": {
        "factory": 4,
        "building": 2
      },
      "failed": false,
      "errors": 0,
      "warnings": 0,
      "assets": [],
      "reasons": [
        {
          "moduleId": "./src/index.js",
          "moduleIdentifier": "C:\\vipdata\\vipproject\\webpack-source\\src\\index.js",
          "module": "./src/index.js",
          "moduleName": "./src/index.js",
          "type": "import()",
          "userRequest": "./lazy",
          "loc": "2:0-46"
        }
      ],
      "providedExports": null,
      "optimizationBailout": [],
      "depth": 1,
      "source": "module.exports = 'lazy';"
    }
  ],
  "filteredModules": 0,
  "logging": {
    "webpack.buildChunkGraph.visitModules": {
      "entries": [],
      "filteredEntries": 5,
      "debug": false
    }
  },
  "children": []
}
```

6. 主要工作流程

- Webpack 的运行流程是一个串行的过程，从启动到结束会依次执行以下流程：
- &#x521D;&#x59CB;&#x5316;&#x53C2;&#x6570;：从配置文件和 Shell 语句中读取与合并参数，得出最终的参数；
- 开始编译：用上一步得到的参数初始化 Compiler 对象，加载所有配置的插件，执行对象的run方法开始执行编译； 确定入口：根据配置中的 entry 找出所有的入口文件
- 编译模块：从入口文件出发，调用所有配置的 Loader 对模块进行编译，再找出该模块依赖的模块，再递归本步骤直到所有入口依赖的文件都经过了本步骤的处理；
- 完成模块编译：在经过第4步使用 Loader 翻译完所有模块后，得到了每个模块被翻译后的最终内容以及它们之间的依赖关系；
- 输出资源：根据入口和模块之间的依赖关系，组装成一个个包含多个模块的 Chunk，再把每个 Chunk 转换成一个单独的文件加入到输出列表，这步是可以修改输出内容的最后机会；
- 输出完成：在确定好输出内容后，根据配置确定输出的路径和文件名，把文件内容写入到文件系统。
- 在以上过程中，Webpack 会在特定的时间点广播出特定的事件，插件在监听到感兴趣的事件后会执行特定的逻辑，并且插件可以调用 Webpack 提供的 API 改变 Webpack 的运行结果。


发射事件
webpack.config.js
shell options
webpack-cli/bin/webpack.js:217
1.解析shell和webpack的配置项
2.激活webpack插件的加载
用yargs参数解析
compiler = webpack(options); webpack初始化
webpack-cli/bin/webpack.js:436
compiler = webpack(options)
1. 创建Compiler对象
2. 注册NodeEnvironmentPlugin插件
3. 挂载options中的plugins插件
4. 使用WebpackOptionsApply初始化基础插件
WebpackOptionsApply:entryOption
run
开始编译
webpack-cli/bin/webpack.js:507
compiler.run(compilerCallback)
1. 调用compile开启编译:Compiler第200行
2.创建Compilation对象:Compiler第457行
1. 负责整个编译过程
2. 内部保留对compiler对象引用-this.compiler
3. this.entries 入口
3. this.modules 所有模块
4. this.chunks 代码块
5. this.assets 所有的资源
6. template
1. this.mainTemplate
2. this.chunkTemplate
3. this.hotUpdateChunkTemplate
4. this.runtimeTemplate
5. this.moduleTemplates
Compiler:beforeRun
Compiler:beforeCompile
Compiler:compile
Compiler:thisCompilation
Compiler:compilation
make
分析入口文件
创建模块对象
webpack/lib/SingleEntryPlugin.js 32行
compilation.addEntry
const moduleFactory= Compilation : 639行
callback 749行->Compiler:this.hooks.make.callAsync
_addModuleChain
this.buildModule : 702行
buildModule
build NormalModule:322行
module.build
1. make事件触发SingleEntryPlugin插件执行
2. 根据入口文件通过对应的工厂方法创建模块，保存到Compilation上。
3. 对module进行build(Compilation:346行)
1. 调用loader进行处理 runLoaders NormalModule:219行 LoadRunner.runLoaders 242行
2. 使用acorn生成AST遍历AST NormalModule:258行
3.如果遇到require依赖，创建Dependency加入依赖数组。
4. module处理完毕后处理依赖的module: Compilation 686行
5. 异步对依赖的模块进行处理build,如果依赖还有依赖，则递归处理。
Compilation:buildModule
Compilation:succeedModule
Compilation:finishModules
processModuleDependencies
Compilation.js:686行
递归处理依赖的模块
compilation:seal
1. 调用seal方法封装，逐次对每个module和chunk进行整理，生成编译后的源码，合并，拆分。
Compilation:seal
Compilation:optimize
Compilation:optimizeTree
Compilation:afterOptimizeTree
Compilation:optimizeChunkModulesBasic
Compilation:reviveModules
Compilation:reviveChunks
Compilation:beforeChunkIds
Compilation:beforeHash
Compilation:afterHash
Compilation:beforeModuleAssets
Compilation:chunkAsset
this.createChunkAssets(); : Compilation:933
this.hooks.afterSeal.callAsync(callback);>this.hooks.afterCompile:467
this.createChunkAssets();
template.getRenderManifest.render()
JavascriptModulesPlugin
MainTemplate.js
ChunkTemplate.js
1. 通过模版把chunk生成_webpack_require()的格式
2.MainTemplate: 处理入口文件的module
3. ChunkTemplate:处理非首屏，需要异步加载的module
4.生成的资源保存在compilation.assets中
Compiler:emit
Compiler:afterEmit
Compiler:done
Compilation:afterSeal
this.emitAssets:Compiler 157
emitAssets
1.把Assets输出到output的path中

6.1 初始化阶段

事件名 解释 代码位置 读取命令行参数 从命令行中读取用户输入的参数
require("./convert-argv")(argv) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/webpack-cli/bin/cli.js#L241)

实例化 Compiler 1.用上一步得到的参数初始化 Compiler 实例

2.Compiler 负责文件监听和启动编译

3.Compiler 实例中包含了完整的 Webpack 配置，全局只有一个 Compiler 实例。

compiler = webpack(options); (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/webpack-cli/bin/cli.js#L443)

加载插件 1.依次调用插件的 apply 方法，让插件可以监听后续的所有事件节点。

同时给插件传入 compiler 实例的引用，以方便插件通过 compiler 调用 Webpack 提供的 API。
plugin.apply(compiler) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/webpack.js#L42-L50)

处理入口 读取配置的 Entrys，为每个 Entry 实例化一个对应的 EntryPlugin，为后面该 Entry 的递归解析工作做准备
new EntryOptionPlugin().apply(compiler) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/WebpackOptionsApply.js#L306)
new SingleEntryPlugin(context, item, name) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/EntryOptionPlugin.js#L24)
compiler.hooks.make.tapAsync (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/SingleEntryPlugin.js#L40-L48)

6.2 编译阶段

事件名 解释 代码位置 run 启动一次新的编译
this.hooks.run.callAsync (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js#L263-L271)

compile 该事件是为了告诉插件一次新的编译将要启动，同时会给插件传入compiler 对象。
compile(callback) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js#L529-L555)

compilation 当 Webpack 以开发模式运行时，每当检测到文件变化，一次新的 Compilation 将被创建。

一个 Compilation 对象包含了当前的模块资源、编译生成资源、变化的文件等。

Compilation 对象也提供了很多事件回调供插件做扩展。
newCompilation(params) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js#L491-L501)

make 一个新的 Compilation 创建完毕主开始编译
this.hooks.make.callAsync (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js#L544)

addEntry 即将从 Entry 开始读取文件
compilation.addEntry (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compilation.js#L1027)
this._addModuleChain (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compilation.js#L1047)

moduleFactory 创建模块工厂
const moduleFactory = this.dependencyFactories.get(Dep) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compilation.js#L933)

create 创建模块
moduleFactory.create (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModuleFactory.js#L369-L409)

factory 开始创建模块
factory(result, (err, module) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModuleFactory.js#L396-L406)
resolver(result (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModuleFactory.js#L129)
this.hooks.resolver.tap("NormalModuleFactory" (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModuleFactory.js#L159)

resolveRequestArray 解析loader路径
resolveRequestArray (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModuleFactory.js#L411)

resolve 解析资源文件路径
resolve (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_enhanced-resolve%404.1.0%40enhanced-resolve/lib/Resolver.js#L136)

userRequest 得到包括loader在内的资源文件的绝对路径用!拼起来的字符串
userRequest (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModuleFactory.js#L254-L259)

ruleSet.exec 它可以根据模块路径名，匹配出模块所需的loader
this.ruleSet.exec (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModuleFactory.js#L270-L279)

_run 它可以根据模块路径名，匹配出模块所需的loader
_run (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/RuleSet.js#L485-L558)

loaders 得到所有的loader数组
results[0].concat(loaders, results[1], results[2]) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModuleFactory.js#L338)

getParser 获取AST解析器
this.getParser(type, settings.parser) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModuleFactory.js#L357)

buildModule 开始编译模块
this.buildModule(module (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compilation.js#L996-L1009)
buildModule(module, optional, origin, dependencies, thisCallback) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compilation.js#L602-L656)

build 开始真正编译入口模块
build(options (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModule.js#L396-L469)

doBuild 开始真正编译入口模块
doBuild (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModule.js#L257-L330)

执行loader 使用loader进行转换
runLoaders (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModule.js#L265)
runLoaders (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_loader-runner%402.3.1%40loader-runner/lib/LoaderRunner.js#L242)

iteratePitchingLoaders 开始递归执行pitch loader
iteratePitchingLoaders (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_loader-runner%402.3.1%40loader-runner/lib/LoaderRunner.js#L362)

loadLoader 加载loader
loadLoader (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_loader-runner%402.3.1%40loader-runner/lib/loadLoader.js#L13)

runSyncOrAsync 执行pitchLoader
runSyncOrAsync (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_loader-runner%402.3.1%40loader-runner/lib/LoaderRunner.js#L175-L188)

processResource 开始处理资源
processResource (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_loader-runner%402.3.1%40loader-runner/lib/LoaderRunner.js#L192)
options.readResource (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_loader-runner%402.3.1%40loader-runner/lib/LoaderRunner.js#L199)
iterateNormalLoaders (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_loader-runner%402.3.1%40loader-runner/lib/LoaderRunner.js#L202)
iterateNormalLoaders (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_loader-runner%402.3.1%40loader-runner/lib/LoaderRunner.js#L209-L235)

createSource 创建源代码对象
this.createSource (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModule.js#L316)

parse 使用parser转换抽象语法树
this.parser.parse (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModule.js#L445-L467)

parse 解析抽象语法树
parse(source, initialState) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Parser.js#2022)

acorn.parse 解析语法树
acorn.parse(code, parserOptions) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Parser.js#L2158)

ImportDependency 遍历并添加添加依赖
parser.state.module.addDependency(clearDep) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/dependencies/HarmonyImportDependencyParserPlugin.js#L28)

succeedModule 生成语法树后就表示一个模块编译完成
this.hooks.succeedModule.call(module) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compilation.js#L652)

processModuleDependencies 递归编译依赖的模块
this.processModuleDependencies(module (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compilation.js#L980)
processModuleDependencies(module, callback) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compilation.js#L663)
this.addModuleDependencies (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compilation.js#L716)
buildModule (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compilation.js#L859)

make后 结束make
this.hooks.make.callAsync(compilation, err => {} (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js#L545)

finish 编译完成
compilation.finish(); (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js#L547)

6.3 结束阶段

事件名 解释 代码位置 seal 封装
compilation.seal (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js#L549)
seal(callback) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compilation.js#L1159-L1301)

addChunk 生成资源
addChunk(name) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compilation.js#L1400)

createChunkAssets 创建资源
this.createChunkAssets() (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compilation.js#L1270)

getRenderManifest 获得要渲染的描述文件
getRenderManifest(options) (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/MainTemplate.js#L355-L360)

render 渲染源码
source = fileManifest.render(); (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compilation.js#L2369)

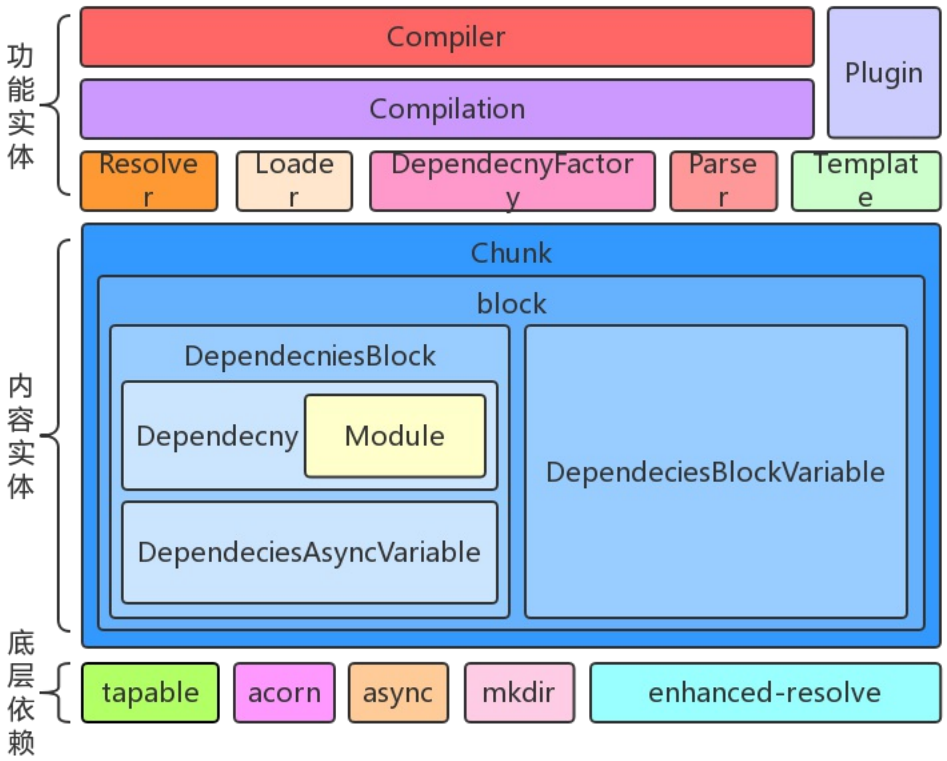
afterCompile 编译结束
this.hooks.afterCompile (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js##L552)

shouldEmit 所有需要输出的文件已经生成好，询问插件哪些文件需要输出，哪些不需要。
this.hooks.shouldEmit (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js##L215)

emit 确定好要输出哪些文件后，执行文件输出，可以在这里获取和修改输出内容。
this.emitAssets(compilation (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js##L228)
this.hooks.emit.callAsync (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js##L363-L367)
const emitFiles = err (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js##L308-L361)
this.outputFileSystem.writeFile (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js##L338)

this.emitRecords 写入记录
this.emitRecords (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js##L249)

done 全部完成
this.hooks.done.callAsync (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/Compiler.js##L255)



7. 参考

- webpack-analysis (https://github.com/zhufengnodejs/webpack-analysis)
- webpack-internal-plugin-relation (https://alienzhou.github.io/webpack-internal-plugin-relation/)