link: null
title: 珠峰架构师成长计划
description: 插件向第三方开发者提供了 webpack 引擎中完整的能力。使用阶段式的构建回调，开发者可以引入它们自己的行为到 webpack 构建流程中。创建插件比创建 loader 更加高级，因为你将需要理解一些 webpack 底层的内部特性来做相应的钩子
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=156 sentences=353, words=2701

---

1. plugin

插件向第三方开发者提供了 webpack 引擎中完整的能力。使用阶段式的构建回调，开发者可以引入它们自己的行为到 webpack 构建流程中。创建插件比创建 loader 更加高级，因为你将需要理解一些 webpack 底层的内部特性来做相应的钩子

1.1 为什么需要一个插件

- webpack 基础配置无法满足需求
- 插件几乎能够任意更改 webpack 编译结果
- webpack 内部也是通过大量内部插件实现的

1.2 可以加载插件的常用对象

对象 钩子
Compiler (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compiler.js)

run,compile,compilation,make,emit,done
Compilation (https://github.com/webpack/webpack/blob/v4.39.3/lib/Compilation.js)

buildModule,normalModuleLoader,succeedModule,finishModules,seal,optimize,after-seal
Module Factory (https://github.com/webpack/webpack/blob/master/lib/ModuleFactory.js)

beforeResolver,afterResolver,module,parser Module
Parser (https://github.com/webpack/webpack/blob/master/lib/Parser.js)

program,statement,call,expression
Template (https://github.com/webpack/webpack/blob/master/lib/Template.js)

hash,bootstrap,localVars,render

2. 创建插件

- 插件是一个类
- 类上有一个apply的实例方法
- apply的参数是compiler

```
class DonePlugin {
    constructor(options) {
        this.options = options;
    }
    apply(compiler) {

    }
}
module.exports = DonePlugin;
```

3. Compiler 和 Compilation

在插件开发中最重要的两个资源就是 compiler和 compilation对象。理解它们的角色是扩展 webpack 引擎重要的第一步。

- compiler 对象代表了完整的 webpack 环境配置。这个对象在启动 webpack 时被一次性建立，并配置好所有可操作的设置，包括 options，loader 和 plugin。当在 webpack 环境中应用一个插件时，插件将收到此 compiler 对象的引用。可以使用它来访问 webpack 的主环境。
- compilation 对象代表了一次资源版本构建。当运行 webpack 开发环境中间件时，每当检测到一个文件变化，就会创建一个新的 compilation，从而生成一组新的编译资源。一个 compilation 对象表现了当前的模块资源、编译生成资源、变化的文件、以及被跟踪依赖的状态信息。compilation 对象也提供了很多关键时机的回调，以供插件做自定义处理时选择使用。

4. 基本插件架构

- 插件是由「具有 apply 方法的 prototype 对象」所实例化出来的
- 这个 apply 方法在安装插件时，会被 webpack compiler 调用一次
- apply 方法可以接收一个 webpack compiler 对象的引用，从而可以在回调函数中访问到 compiler 对象

4.1 使用插件代码

- [使用插件] https://github.com/webpack/webpack/blob/master/lib/webpack.js#L60-L69 (https://github.com/webpack/webpack/blob/master/lib/webpack.js#L60-L69))

```
if (options.plugins && Array.isArray(options.plugins)) {
  for (const plugin of options.plugins) {
    plugin.apply(compiler);
  }
}
```

4.2 Compiler 插件

- done: new AsyncSeriesHook(["stats"]) (https://github.com/webpack/webpack/blob/master/lib/Compiler.js#L105)

4.2.1 同步

```
class DonePlugin {
  constructor(options) {
    this.options = options;
  }
  apply(compiler) {
    compiler.hooks.done.tap("DonePlugin", (stats) => {
      console.log("Hello ", this.options.name);
    });
  }
}
module.exports = DonePlugin;
```

4.2.2 异步

```
class DonePlugin {
  constructor(options) {
    this.options = options;
  }
  apply(compiler) {
    compiler.hooks.done.tapAsync("DonePlugin", (stats, callback) => {
      console.log("Hello ", this.options.name);
      callback();
    });
  }
}
module.exports = DonePlugin;
```

4.3 使用插件

- 要安装这个插件，只需要在你的 webpack 配置的 plugin 数组中添加一个实例

```
const DonePlugin = require("./plugins/DonePlugin");
module.exports = {
  entry: "./src/index.js",
  output: {
    path: path.resolve("build"),
    filename: "bundle.js",
  },
  plugins: [new DonePlugin({ name: "zhufeng" })],
};
```

5. compilation 插件

- 使用 compiler 对象时，你可以绑定提供了编译 compilation 引用的回调函数，然后拿到每次新的 compilation 对象。这些 compilation 对象提供了一些钩子函数，来钩入到构建流程的很多步骤中

5.1 webpack-assets-plugin.js

plugins\webpack-assets-plugin.js

```

class WebpackAssetsPlugin {
  constructor(options) {
    this.options = options;
  }
  apply(compiler) {

    compiler.hooks.compilation.tap('WebpackAssetsPlugin', (compilation) => {

      compilation.hooks.chunkAsset.tap('WebpackAssetsPlugin', (chunk, filename) => {
        console.log(chunk.name || chunk.id, filename);
      });
    });
  }
}
module.exports = WebpackAssetsPlugin;
```

6. 打包 zip

- webpack-sources (https://www.npmjs.com/package/webpack-sources)

6.1 webpack-archive-plugin.js

plugins\webpack-archive-plugin.js

```
const jszip = require('jszip');
const { RawSource } = require('webpack-sources');
const { Compilation } = require('webpack');

class WebpackArchivePlugin {
  constructor(options) {
    this.options = options;
  }
  apply(compiler) {
    compiler.hooks.compilation.tap('WebpackAssetsPlugin', (compilation) => {

      compilation.hooks.processAssets.tapPromise({ name: 'WebpackArchivePlugin' }, (assets) => {
        const zip = new jszip();
        for (const filename in assets) {
          const sourceObj = assets[filename];
          const sourceCode = sourceObj.source();
          zip.file(filename, sourceCode);
        }
        return zip.generateAsync({ type: 'nodebuffer' }).then(zipContent => {
          assets[`archive_${Date.now()}。zip`] = new RawSource(zipContent);
        });
      });
    });
  }
}
module.exports = WebpackArchivePlugin;
```

6.2 webpack.config.js

webpack.config.js

```
const WebpackArchivePlugin = require('./plugins/webpack-archive-plugin');
  plugins: [
+   new WebpackArchivePlugin({
+     filename:'[timestamp].zip'
+   })
]
```

7.自动外链

7.自动外链

7.1 使用外部类库

- 手动指定 external
- 手动引入 script

> 能否检测代码中的 import 自动处理这个步骤?

```
{
  externals:{

  'jquery':'{{content}}#x27;
},
  module:{}
}
```

7.2 思路

- 解决 import 自动处理 external和 script的问题，需要怎么实现，该从哪方面开始考虑
- 依赖 当检测到有 import该 library时，将其设置为不打包类似 exteral,并在指定模版中加入 script,那么如何检测 import？这里就用 Parser
- external依赖 需要了解 external 是如何实现的，webpack 的 external 是通过插件 ExternalsPlugin实现的，ExternalsPlugin 通过 tap NormalModuleFactory 在每次创建 Module 的时候判断是否是 ExternalModule
- webpack4 加入了模块类型之后， Parser获取需要指定类型 moduleType,一般使用 javascript/auto即可

7.3 使用 plugins

```
plugins: [
  new HtmlWebpackPlugin({
          template:'./src/index.html'
  }),
  new AutoExternalPlugin({
      jquery:{
          variable:'jQuery',
          url:'https://cdn.bootcss.com/jquery/3.1.0/jquery.js'
      },
      lodash:{
          variable:'_',
          url:'https://cdn.bootcdn.net/ajax/libs/lodash.js/4.17.21/lodash.js'
      }
  })
];
```

7.4 AutoExternalPlugin

- ExternalsPlugin.js (https://github.com/webpack/webpack/blob/0d4607c68e04a659fa58499e1332c97d5376368a/lib/ExternalsPlugin.js)
- ExternalModuleFactoryPlugin (https://github.com/webpack/webpack/blob/eeafeee32ad5a1469e39ce66df671e3710332608/lib/ExternalModuleFactoryPlugin.js)
- ExternalModule.js (https://github.com/webpack/webpack/blob/eeafeee32ad5a1469e39ce66df671e3710332608/lib/ExternalModule.js)
- parser (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModuleFactory.js#L87)
- factory (https://github.com/zhufengnodejs/webpack-analysis/blob/master/node_modules/_webpack%404.20.2%40webpack/lib/NormalModuleFactory.js#L66)
- htmlWebpackPluginAlterAssetTags (https://github.com/jantimon/html-webpack-plugin/blob/v3.2.0/index.js#L62)

AsyncSeriesBailHook factorize

```
let { AsyncSeriesBailHook } = require("tapable");
let factorize = new AsyncSeriesBailHook(['resolveData']);

factorize.tapAsync('factory1', (resolveData, callback) => {
  if (resolveData === 'jquery') {
    callback(null, {
      id: resolveData,
      type: '外部模块',
      source: 'window.jQuery'
    });
  } else {
    callback(null);
  }
});

factorize.tapAsync('factory2', (resolveData, callback) => {
  callback(null, { id: resolveData, type: '正常模块', source: 'webpack打包后的内容' });
});

factorize.callAsync('jquery', (err, module) => {
  console.log(module);
});
factorize.callAsync('lodash', (err, module) => {
  console.log(module);
});
```

plugins\auto-external-plugin.js

```
const { ExternalModule } = require("webpack");
const HtmlWebpackPlugin = require('html-webpack-plugin');

class AutoExternalPlugin{
    constructor(options){
        this.options = options;
        this.externalModules = Object.keys(this.options);
        this.importedModules = new Set();
    }
    apply(compiler){

        compiler.hooks.normalModuleFactory.tap('AutoExternalPlugin',(normalModuleFactory)=>{

            normalModuleFactory.hooks.parser
            .for('javascript/auto')
            .tap('AutoExternalPlugin',parser=>{

                parser.hooks.import.tap('AutoExternalPlugin',(statement,source)=>{
                    if(this.externalModules.includes(source)){
                        this.importedModules.add(source);
                    }
                });

                parser.hooks.call.for('require').tap('AutoExternalPlugin',(expression)=>{
                    let value = expression.arguments[0].value;
                    if(this.externalModules.includes(value)){
                        this.importedModules.add(value);
                    }
                });
            })

            normalModuleFactory.hooks.factorize.tapAsync('AutoExternalPlugin',(resolveData,callback)=>{
                let {request} = resolveData;
                if(this.externalModules.includes(request)){
                    let {variable} = this.options[request];

                    callback(null,new ExternalModule(variable,'window',request));
                }else{
                    callback(null);
                }
            });
        });

        compiler.hooks.compilation.tap('AutoExternalPlugin',(compilation)=>{
            HtmlWebpackPlugin.getHooks(compilation).alterAssetTags.tapAsync('AutoExternalPlugin',(htmlData,callback)=>{

                Reflect.ownKeys(this.options).filter(key=>this.importedModules.has(key)).forEach(key=>{

                    htmlData.assetTags.scripts.unshift({
                        tagName:'script',
                        voidTag:false,
                        attributes:{
                            defer:false,
                            src:this.options[key].url
                        }
                    });
                })
                callback(null,htmlData);
            });
        });

    }
}
module.exports = AutoExternalPlugin;
```

8.AsyncQueue

8.1 AsyncQueue

```
let AsyncQueue = require('webpack/lib/util/AsyncQueue');
let AsyncQueue = require('./AsyncQueue');
function processor(item, callback) {
    setTimeout(() => {
        console.log('process',item);
        callback(null, item);
    }, 3000);
}
const getKey = (item) => {
    return item.key;
}
let queue  = new AsyncQueue({
    name:'createModule',parallelism:3,processor,getKey
});
const start = Date.now();
let item1 = {key:'module1'};
queue.add(item1,(err,result)=>{
    console.log(err,result);
    console.log(Date.now() - start);
});
queue.add(item1,(err,result)=>{
    console.log(err,result);
    console.log(Date.now() - start);
});
queue.add({key:'module2'},(err,result)=>{
    console.log(err,result);
    console.log(Date.now() - start);
});
queue.add({key:'module3'},(err,result)=>{
    console.log(err,result);
    console.log(Date.now() - start);
});
queue.add({key:'module4'},(err,result)=>{
    console.log(err,result);
    console.log(Date.now() - start);
});
```

8.2 use.js

use.js

```
const QUEUED_STATE = 0;
const PROCESSING_STATE = 1;
const DONE_STATE = 2;
class ArrayQueue {
    constructor() {
        this._list = [];
    }
    enqueue(item) {
        this._list.push(item);
    }
    dequeue() {
        return this._list.shift();
    }
}
class AsyncQueueEntry {
    constructor(item, callback) {
        this.item = item;
        this.state = QUEUED_STATE;
        this.callback = callback;
    }
}
class AsyncQueue {
    constructor({ name, parallelism, processor, getKey }) {
        this._name = name;
        this._parallelism = parallelism;
        this._processor = processor;
        this._getKey = getKey;
        this._entries = new Map();
        this._queued = new ArrayQueue();
        this._activeTasks = 0;
        this._willEnsureProcessing = false;
    }
    add = (item, callback) => {
        const key = this._getKey(item);
        const entry = this._entries.get(key);
        if (entry !== undefined) {
            if (entry.state === DONE_STATE) {
                process.nextTick(() => callback(entry.error, entry.result));
            } else if (entry.callbacks === undefined) {
                entry.callbacks = [callback];
            } else {
                entry.callbacks.push(callback);
            }
            return;
        }
        const newEntry = new AsyncQueueEntry(item, callback);
        this._entries.set(key, newEntry);
        this._queued.enqueue(newEntry);
        if (this._willEnsureProcessing === false) {
            this._willEnsureProcessing = true;
            setImmediate(this._ensureProcessing);
        }
    }
    _ensureProcessing = () => {

        while (this._activeTasks < this._parallelism) {
            const entry = this._queued.dequeue();
            if (entry === undefined) break;
            this._activeTasks++;
            entry.state = PROCESSING_STATE;
            this._startProcessing(entry);
        }
        this._willEnsureProcessing = false;
    }
    _startProcessing = (entry) => {
        this._processor(entry.item, (e, r) => {
            this._handleResult(entry, e, r);
        });
    }
    _handleResult = (entry, error, result) => {
        const callback = entry.callback;
        const callbacks = entry.callbacks;
        entry.state = DONE_STATE;
        entry.callback = undefined;
        entry.callbacks = undefined;
        entry.result = result;
        entry.error = error;
        callback(error, result);
        if (callbacks !== undefined) {
            for (const callback of callbacks) {
                callback(error, result);
            }
        }
        this._activeTasks--;
        if (this._willEnsureProcessing === false) {
            this._willEnsureProcessing = true;
            setImmediate(this._ensureProcessing);
        }
    }
}
module.exports = AsyncQueue;
```

9. 参考

- Node.js SDK (https://developer.qiniu.com/kodo/sdk/1289/nodejs)
- writing-a-plugin (https://webpack.js.org/contribute/writing-a-plugin/)
- api/plugins (https://webpack.js.org/api/plugins/)

9.1 钩子集合

- wepback-plugin-visualizer (https://www.npmjs.com/package/wepback-plugin-visualizer)

9.1.1 收集

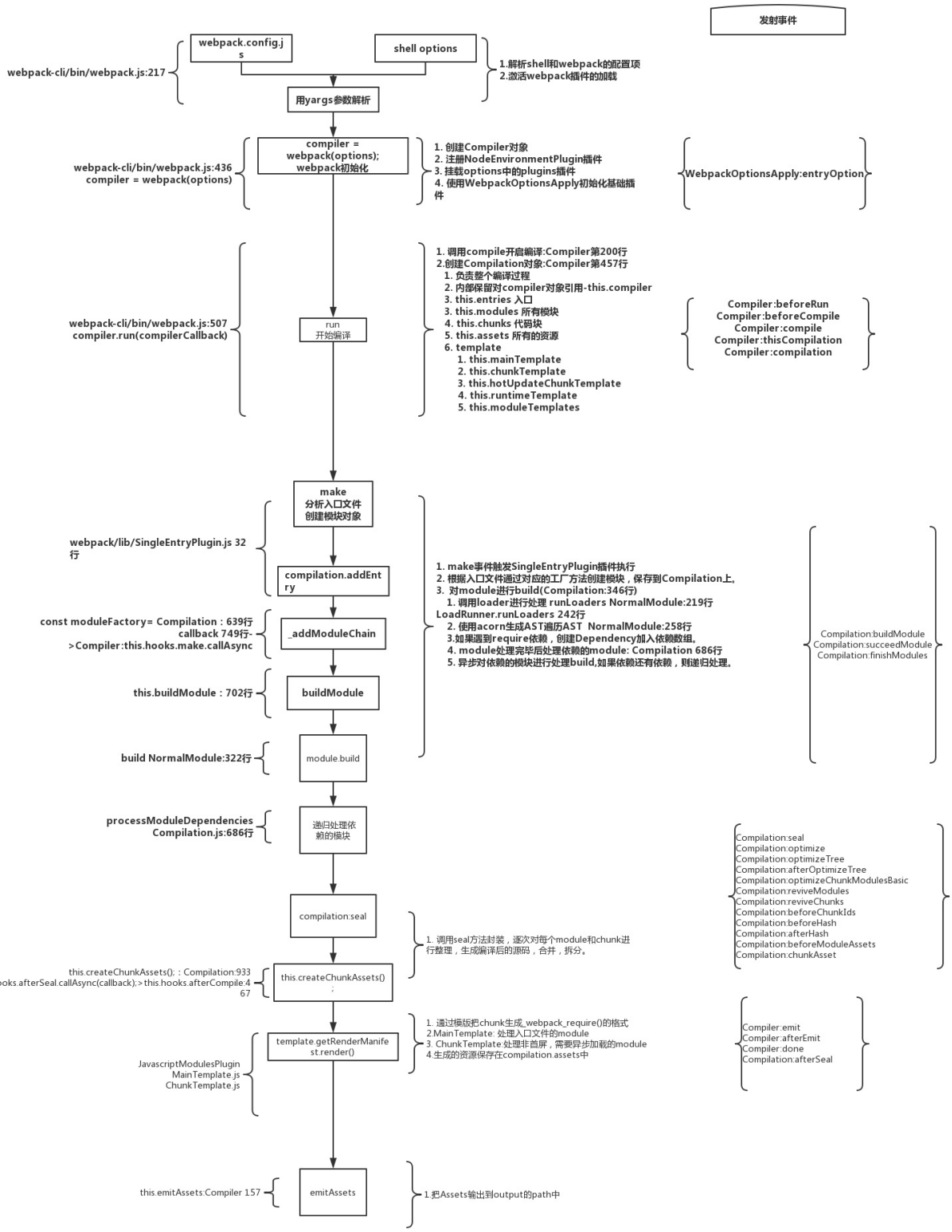
```
        Object.keys(this.hooks).forEach(hookName => {
            const hook = this.hooks[hookName];
            if (hook instanceof HookMap) {
                for (let key of hook._map.keys()) {
                    hook.for(key).tap('flow', () => {
                        console.log(`|JavascriptParser|${hookName}|${hook.for(key).constructor.name}|${hook._args}|`);
                    });
                }
            } else {
                hook.tap('flow', () => {
                    console.log(`|JavascriptParser|${hookName}|${hook.constructor.name}|${hook._args}|`);
                });
            }
        });
```

9.1.2 触发时机

- hooks (https://webpack.docschina.org/api/compiler-hooks/#environment)

对象 钩子名称 类型 参数 含义 Compiler environment SyncHook 在编译器准备环境时调用，时机就在配置文件中初始化插件之后 Compiler afterEnvironment SyncHook Compiler entryOption SyncBailHook context,entry Compiler afterPlugins SyncHook compiler Compiler afterResolvers SyncHook compiler Compiler initialize SyncHook Compiler beforeRun AsyncSeriesHook compiler Compiler run AsyncSeriesHook compiler Compiler infrastructureLog SyncBailHook origin,type,args Compiler readRecords AsyncSeriesHook 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,options NormalModuleFactory beforeResolve AsyncSeriesBailHook resolveData NormalModuleFactory factorize AsyncSeriesBailHook resolveData NormalModuleFactory resolve AsyncSeriesBailHook resolveData NormalModuleFactory afterResolve AsyncSeriesBailHook resolveData NormalModuleFactory createModule AsyncSeriesBailHook createData,resolveData NormalModuleFactory module SyncWaterfallHook module,createData,resolveData Compilation buildModule SyncHook module Compilation normalModuleLoader SyncHook loaderContext,module JavascriptParser program SyncBailHook ast,comments JavascriptParser preStatement SyncBailHook statement JavascriptParser preStatement SyncBailHook statement JavascriptParser blockPreStatement SyncBailHook declaration JavascriptParser preDeclarator SyncBailHook declarator,statement JavascriptParser blockPreStatement SyncBailHook declaration JavascriptParser statement SyncBailHook statement JavascriptParser declarator SyncBailHook declarator,statement JavascriptParser statement SyncBailHook statement JavascriptParser finish SyncBailHook ast,comments Compilation succeedModule SyncHook module NormalModuleFactory beforeResolve AsyncSeriesBailHook resolveData NormalModuleFactory factorize AsyncSeriesBailHook resolveData NormalModuleFactory resolve AsyncSeriesBailHook resolveData NormalModuleFactory afterResolve AsyncSeriesBailHook resolveData NormalModuleFactory createModule AsyncSeriesBailHook createData,resolveData NormalModuleFactory module SyncWaterfallHook module,createData,resolveData Compilation buildModule SyncHook module Compilation normalModuleLoader SyncHook loaderContext,module JavascriptParser program SyncBailHook ast,comments JavascriptParser preStatement SyncBailHook statement JavascriptParser blockPreStatement SyncBailHook declaration JavascriptParser statement SyncBailHook statement JavascriptParser finish SyncBailHook ast,comments Compilation succeedModule SyncHook module Compilation succeedEntry SyncHook entry,options,module Compilation log SyncBailHook origin,logEntry Compiler finishMake AsyncSeriesHook compilation Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation finishModules AsyncSeriesHook modules Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation afterOptimizeDependencies SyncHook modules Compilation log SyncBailHook origin,logEntry Compilation beforeChunks SyncHook Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation afterChunks SyncHook chunks Compilation log SyncBailHook origin,logEntry Compilation optimize SyncHook Compilation optimizeModules SyncBailHook modules Compilation afterOptimizeModules SyncHook modules Compilation optimizeChunks SyncBailHook chunks,chunkGroups Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation afterOptimizeChunks SyncHook chunks,chunkGroups Compilation optimizeTree AsyncSeriesHook chunks,modules Compilation afterOptimizeTree SyncHook chunks,modules Compilation optimizeChunkModules AsyncSeriesBailHook chunks,modules Compilation afterOptimizeChunkModules SyncHook chunks,modules Compilation shouldRecord SyncBailHook Compilation reviveModules SyncHook modules,records Compilation beforeModuleIds SyncHook modules Compilation moduleIds SyncHook modules Compilation optimizeModuleIds SyncHook modules Compilation afterOptimizeModuleIds SyncHook modules Compilation reviveChunks SyncHook chunks,records Compilation beforeChunkIds SyncHook chunks Compilation chunkIds SyncHook chunks Compilation optimizeChunkIds SyncHook chunks Compilation afterOptimizeChunkIds SyncHook chunks Compilation log SyncBailHook origin,logEntry Compilation recordModules SyncHook modules,records Compilation recordChunks SyncHook chunks,records Compilation optimizeCodeGeneration SyncHook modules Compilation log SyncBailHook origin,logEntry Compilation beforeModuleHash SyncHook Compilation log SyncBailHook origin,logEntry Compilation afterModuleHash SyncHook Compilation log SyncBailHook origin,logEntry Compilation beforeCodeGeneration SyncHook Compilation log SyncBailHook origin,logEntry Compilation afterCodeGeneration SyncHook Compilation log SyncBailHook origin,logEntry Compilation beforeRuntimeRequirements SyncHook Compilation additionalModuleRuntimeRequirements SyncHook module,runtimeRequirements,context Compilation additionalModuleRuntimeRequirements SyncHook module,runtimeRequirements,context Compilation log SyncBailHook origin,logEntry Compilation additionalChunkRuntimeRequirements SyncHook chunk,runtimeRequirements,context Compilation log SyncBailHook origin,logEntry Compilation additionalTreeRuntimeRequirements SyncHook chunk,runtimeRequirements,context Compilation log SyncBailHook origin,logEntry Compilation afterRuntimeRequirements SyncHook Compilation log SyncBailHook origin,logEntry Compilation beforeHash SyncHook Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation chunkHash SyncHook chunk,chunkHash,ChunkHashContext Compilation contentHash SyncHook chunk Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation fullHash SyncHook hash Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation afterHash SyncHook Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation recordHash SyncHook records Compilation log SyncBailHook origin,logEntry Compilation beforeModuleAssets SyncHook Compilation log SyncBailHook origin,logEntry Compilation shouldGenerateChunkAssets SyncBailHook Compilation beforeChunkAssets SyncHook Compilation renderManifest SyncWaterfallHook result,options Compilation assetPath SyncWaterfallHook path,options,assetInfo Compilation chunkAsset SyncHook chunk,filename Compilation log SyncBailHook origin,logEntry Compilation additionalChunkAssets Object undefined Compilation additionalAssets Object undefined Compilation optimizeAssets AsyncSeriesHook assets Compilation processAssets AsyncSeriesHook assets Compilation optimizeChunkAssets Object undefined Compilation afterOptimizeChunkAssets Object undefined Compilation afterOptimizeAssets SyncHook assets Compilation afterProcessAssets SyncHook assets Compilation log SyncBailHook origin,logEntry Compilation record SyncHook compilation,records Compilation needAdditionalSeal SyncBailHook Compilation afterSeal AsyncSeriesHook Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compiler afterCompile AsyncSeriesHook compilation Compilation log SyncBailHook origin,logEntry Compiler shouldEmit SyncBailHook compilation Compiler emit AsyncSeriesHook compilation Compilation assetPath SyncWaterfallHook path,options,assetInfo Compiler assetEmitted AsyncSeriesHook file,info Compiler afterEmit AsyncSeriesHook compilation Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compilation needAdditionalPass SyncBailHook Compiler emitRecords AsyncSeriesHook Compilation log SyncBailHook origin,logEntry Compiler done AsyncSeriesHook stats Compilation log SyncBailHook origin,logEntry Compilation log SyncBailHook origin,logEntry Compiler shutdown AsyncSeriesHook Compilation statsNormalize SyncHook options,context Compilation statsFactory SyncHook statsFactory,options Compilation statsPrinter SyncHook statsPrinter,options Compilation processErrors SyncWaterfallHook errors Compilation processWarnings SyncWaterfallHook warnings Compiler afterDone SyncHook stats Compiler infrastructureLog SyncBailHook origin,type,args

9.1.3 工作流



9.1.3.1 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)

9.1.3.2 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)

9.1.3.3 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)