

webpack

babel内部是如何转的

与vite的比较

Loader和Plugin的区别

构建流程

<https://juejin.cn/post/6844904094281236487#heading-2>

是一个打包模块化的JS工具

利用webpack优化前端性能

webpack的打包文件类型

工作流程

- bundle
  - 是webpack打包出来的文件
- trunk
  - 是代码块，一个trunk由多个模块组合而成，用于代码的合并和分割
  - 对于打包产物bundle，有些情况下，我们觉得太大了。为了优化性能，比如快速打开首屏，利用缓存等，我们需要对bundle进行以下拆分，对于拆分出来的东西，我们叫它chunk。
- module
  - 是开发中的单个模块

- 使用CDN
  - 使用CDN
  - 将引用的静态资源路径修改为CDN路径
  - 可以通过output参数和各个loader的publicPath来修改路径
- 提取公共代码
  - 提取公共代码
  - 通过externals配置来提取常用库
- 压缩代码并删除多余代码，简化代码写法
  - 压缩js文件
    - uglify.js
    - paralleluglify-plugin
  - 压缩css文件
    - cssnano

- 分包SplitChunk
  - 使用插件：split-chunks-plugin
  - webpack 将根据以下条件自动拆分 chunks：
    - 新的 chunk 可以被共享，或者模块来自于 node\_modules 文件夹
    - 新的 chunk 体积大于 20kb（在进行 min+gz 之前的体积）
    - 当按需加载 chunks 时，并行请求的最大数量小于或等于 30
    - 当加载初始化页面时，并发请求的最大数量小于或等于 30

vite

比较

vite缺点

是一个打包模块化的JS工具

服务器快速启动

更新速度快

比较

vite缺点

是一个打包模块化的JS工具

原生ESM：浏览器自身对模块的支持

Vite 以原生ESM方式提供源码。这实际上是让浏览器接管了打包程序的部分工作

比较

vite缺点

是一个打包模块化的JS工具

依赖 大多为在开发时不会变动的纯 JavaScript。比如npm安装的库

源码 通常包含一些并非直接是 JavaScript 的文件，需要转换（例如 JSX，CSS 或者 Vue/Svelte 组件），时常会被编辑

并不是所有的源码都需要同时被加载（例如基于路由拆分的代码模块）。

Vite 将会使用 esbuild 预构建依赖。Esbuild 使用 Go 编写，并且比以 JavaScript 编写的打包器预构建依赖快 10-100 倍。

原生ESM：浏览器自身对模块的支持

Vite 以原生ESM方式提供源码。这实际上是让浏览器接管了打包程序的部分工作

根据情景动态导入代码，即只在当前屏幕上实际使用时才会被处理。

当编辑一个文件时，Vite 只需要精确地使已编辑的模块与其最近的 HMR 边界之间的链失活

由于现代浏览器本身就支持ES Module，会自动向依赖的Module发出请求

Rollup 在应用打包方面更加成熟和灵活。使用Rollup打包

对 TypeScript、JSX、CSS 等支持开箱即用。

webpack会先打包，然后启动开发服务器

在HMR（热更新）方面，当改动了一个模块后，vite仅需让浏览器重新请求该模块即可，不像webpack那样需要把该模块的相关依赖模块全部编译一次，效率更高。

vite的主要优势在开发阶段

生态不及webpack，加载器、插件不够丰富

生产环境esbuild构建对于css和代码分割不够友好，没被大规模重度使用，会隐藏一些问题

在webpack中，一切文件皆模块

通过loader转换文件，通过plugin注入钩子

最后输出有多个模块组合成的文件

webpack所做的事情就是分析项目结构，找到JS模块以及其他浏览器不能直接运行的拓展语言，例如sass,TS等，并将其打包成合适的格式，以供浏览器使用

loader

plugin

初始化参数 从配置文件和 Shell 语句中读取与合并参数，得出最终的参数

加载loader,plugin,并执行run方法 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

loader编译模块 从入口文件出发，调用配置的 Loader 对模块进行翻译，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

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

就是插件，基于事件流框架 插件可以扩展 Webpack 的功能 在 Webpack 运行的生命周期中会广播出许多事件，Plugin 可以监听这些事件，在合适的时机通过 Webpack 提供的 API 改变输出结果。

用来自定义webpack打包过程的插件，插件含有apply方法，可以参与到webpack打包的各个流程

在 plugins 中单独配置 每一项是一个plugin实例，参数通过构造函数传入

常用的plugin

- html-webpack-plugin 简化 HTML 文件创建 (依赖于 html-loader)
- HotModuleReplacementPlugin 模块热替换
- compression-webpack-plugin 把生成的文件进行压缩
- vue-skeleton-webpack-plugin 骨架屏

常用的loader

- babel-loader 吧ES6转成ES5
- vue-loader 找到.vue文件，将template、css、js提取出来交给相应的loader进行处理
- sass-loader
- css-loader 加载CSS，支持模块化，压缩，文件导入
- style-loader 把css代码注入到js中，通过DOM操作去加载css
- eslint-loader 通过ESlint检查JS代码
- file-loader 把文件输出到一个文件夹中，在代码中通过相对URL的方式引入文件
- source-map-loader 生成sourcemap，方便debug
- image-loader 加载并压缩文件

告诉webpack如何如何转化处理某一类型的文件，并引入到打包文件当中

在 module.rules 中配置

- 作为模块的解析规则
- 类型为数组 每一项都是一个object 描述了对于什么类型的文件，使用什么加载，使用什么参数

因为 Webpack 只认识 JavaScript，所以 Loader 就成了翻译官，对其他类型的资源进行转译的预处理工作。

本质就是一个函数，在该函数中对接收到的内容进行转换，返回转换后的结果

告诉webpack如何如何转化处理某一类型的文件，并引入到打包文件当中

在 module.rules 中配置

作为模块的解析规则

类型为数组

每一项都是一个object

描述了对于什么类型的文件，使用什么加载，使用什么参数

吧ES6转成ES5

找到.vue文件，将template、css、js提取出来交给相应的loader进行处理

加载CSS，支持模块化，压缩，文件导入

把css代码注入到js中，通过DOM操作去加载css

通过ESlint检查JS代码

把文件输出到一个文件夹中，在代码中通过相对URL的方式引入文件

生成sourcemap，方便debug

加载并压缩文件

插件可以扩展 Webpack 的功能

在 Webpack 运行的生命周期中会广播出许多事件，Plugin 可以监听这些事件，在合适的时机通过 Webpack 提供的 API 改变输出结果。

用来自定义webpack打包过程的插件，插件含有apply方法，可以参与到webpack打包的各个流程

每一项是一个plugin实例，参数通过构造函数传入

简化 HTML 文件创建 (依赖于 html-loader)

模块热替换

把生成的文件进行压缩

骨架屏

吧ES6转成ES5

找到.vue文件，将template、css、js提取出来交给相应的loader进行处理

加载CSS，支持模块化，压缩，文件导入

把css代码注入到js中，通过DOM操作去加载css

通过ESlint检查JS代码

把文件输出到一个文件夹中，在代码中通过相对URL的方式引入文件

生成sourcemap，方便debug

加载并压缩文件

插件可以扩展 Webpack 的功能

在 Webpack 运行的生命周期中会广播出许多事件，Plugin 可以监听这些事件，在合适的时机通过 Webpack 提供的 API 改变输出结果。

用来自定义webpack打包过程的插件，插件含有apply方法，可以参与到webpack打包的各个流程

每一项是一个plugin实例，参数通过构造函数传入

简化 HTML 文件创建 (依赖于 html-loader)

模块热替换

把生成的文件进行压缩

骨架屏

吧ES6转成ES5

找到.vue文件，将template、css、js提取出来交给相应的loader进行处理

加载CSS，支持模块化，压缩，文件导入

把css代码注入到js中，通过DOM操作去加载css

通过ESlint检查JS代码

把文件输出到一个文件夹中，在代码中通过相对URL的方式引入文件

生成sourcemap，方便debug

加载并压缩文件

插件可以扩展 Webpack 的功能

在 Webpack 运行的生命周期中会广播出许多事件，Plugin 可以监听这些事件，在合适的时机通过 Webpack 提供的 API 改变输出结果。

用来自定义webpack打包过程的插件，插件含有apply方法，可以参与到webpack打包的各个流程

每一项是一个plugin实例，参数通过构造函数传入

简化 HTML 文件创建 (依赖于 html-loader)

模块热替换

把生成的文件进行压缩

骨架屏

吧ES6转成ES5

找到.vue文件，将template、css、js提取出来交给相应的loader进行处理

加载CSS，支持模块化，压缩，文件导入

把css代码注入到js中，通过DOM操作去加载css

通过ESlint检查JS代码

把文件输出到一个文件夹中，在代码中通过相对URL的方式引入文件

生成sourcemap，方便debug

加载并压缩文件

插件可以扩展 Webpack 的功能

在 Webpack 运行的生命周期中会广播出许多事件，Plugin 可以监听这些事件，在合适的时机通过 Webpack 提供的 API 改变输出结果。

用来自定义webpack打包过程的插件，插件含有apply方法，可以参与到webpack打包的各个流程

每一项是一个plugin实例，参数通过构造函数传入

简化 HTML 文件创建 (依赖于 html-loader)

模块热替换

把生成的文件进行压缩

骨架屏

吧ES6转成ES5

找到.vue文件，将template、css、js提取出来交给相应的loader进行处理

加载CSS，支持模块化，压缩，文件导入

把css代码注入到js中，通过DOM操作去加载css

通过ESlint检查JS代码

把文件输出到一个文件夹中，在代码中通过相对URL的方式引入文件

生成sourcemap，方便debug

加载并压缩文件

插件可以扩展 Webpack 的功能

在 Webpack 运行的生命周期中会广播出许多事件，Plugin 可以监听这些事件，在合适的时机通过 Webpack 提供的 API 改变输出结果。

用来自定义webpack打包过程的插件，插件含有apply方法，可以参与到webpack打包的各个流程

每一项是一个plugin实例，参数通过构造函数传入

简化 HTML 文件创建 (依赖于 html-loader)

模块热替换

把生成的文件进行压缩

骨架屏

吧ES6转成ES5

找到.vue文件，将template、css、js提取出来交给相应的loader进行处理

加载CSS，支持模块化，压缩，文件导入

把css代码注入到js中，通过DOM操作去加载css

通过ESlint检查JS代码

把文件输出到一个文件夹中，在代码中通过相对URL的方式引入文件

生成sourcemap，方便debug

加载并压缩文件

插件可以扩展 Webpack 的功能

在 Webpack 运行的生命周期中会广播出许多事件，Plugin 可以监听这些事件，在合适的时机通过 Webpack 提供的 API 改变输出结果。

用来自定义webpack打包过程的插件，插件含有apply方法，可以参与到webpack打包的各个流程

每一项是一个plugin实例，参数通过构造函数传入

简化 HTML 文件创建 (依赖于 html-loader)

模块热替换

把生成的文件进行压缩

骨架屏

吧ES6转成ES5

找到.vue文件，将template、css、js提取出来交给相应的loader进行处理

加载CSS，支持模块化，压缩，文件导入

把css代码注入到js中，通过DOM操作去加载css

通过ESlint检查JS代码

把文件输出到一个文件夹中，在代码中通过相对URL的方式引入文件

生成sourcemap，方便debug

加载并压缩文件

插件可以扩展 Webpack 的功能

在 Webpack 运行的生命周期中会广播出许多事件，Plugin 可以监听这些事件，在合适的时机通过 Webpack 提供的 API 改变输出结果。

用来自定义webpack打包过程的插件，插件含有apply方法，可以参与到webpack打包的各个流程

每一项是一个plugin实例，参数通过构造函数传入

简化 HTML 文件创建 (依赖于 html-loader)

模块热替换

把生成的文件进行压缩

骨架屏

吧ES6转成ES5

找到.vue文件，将template、css、js提取出来交给相应的loader进行处理

加载CSS，支持模块化，压缩，文件导入

把css代码注入到js中，通过DOM操作去加载css

通过ESlint检查JS代码

把文件输出到一个文件夹中，在代码中通过相对URL的方式引入文件

生成sourcemap，方便debug

加载并压缩文件

插件可以扩展 Webpack 的功能

在 Webpack 运行的生命周期中会广播出许多事件，Plugin 可以监听这些事件，在合适的时机通过 Webpack 提供的 API 改变输出结果。

用来自定义webpack打包过程的插件，插件含有apply方法，可以参与到webpack打包的各个流程

每一项是一个plugin实例，参数通过构造函数传入

简化 HTML 文件创建 (依赖于 html-loader)

模块热替换

把生成的文件进行压缩

骨架屏

吧ES6转成ES5

找到.vue文件，将template、css、js提取出来交给相应的loader进行处理

加载CSS，支持模块化，压缩，文件导入

把css代码注入到js中，通过DOM操作去加载css

通过ESlint检查JS代码

把文件输出到一个文件夹中，在代码中通过相对URL的方式引入文件

生成sourcemap，方便debug

加载并压缩文件

插件可以扩展 Webpack 的功能

在 Webpack 运行的生命周期中会广播出许多事件，Plugin 可以监听这些事件，在合适的时机通过 Webpack 提供的 API 改变输出结果。

用来自定义webpack打包过程的插件，插件含有apply方法，可以参与到webpack打包的各个流程

每一项是一个plugin实例，参数通过构造函数传入

简化 HTML 文件创建 (依赖于 html-loader)

模块热替换

把生成的文件进行压缩

骨架屏

吧ES6转成ES5

找到.vue文件，将template、css、js提取出来交给相应的loader进行处理

加载CSS，支持模块化，压缩，文件导入

把css代码注入到js中，通过DOM操作去加载css

通过ESlint检查JS代码

把文件输出到一个文件夹中，在代码中通过相对URL的方式引入文件

生成sourcemap，方便debug

加载并压缩文件

插件可以扩展 Webpack 的功能

在 Webpack 运行的生命周期中会广播出许多事件，Plugin 可以监听这些事件，在合适的时机通过 Webpack 提供的 API 改变输出结果。

用来自定义webpack打包过程的插件，插件含有apply方法，可以参与到webpack打包的各个流程

每一项是一个plugin实例，参数通过构造函数传入

简化 HTML 文件创建 (依赖于 html-loader)

模块热替换

把生成的文件进行压缩

骨架屏

吧ES6转成ES5

找到.vue文件，将template、css、js提取出来交给相应的loader进行处理

加载CSS，支持模块化，压缩，文件导入

把css代码注入到js中，通过DOM操作去加载css

通过ESlint检查JS代码

把文件输出到一个文件夹中，在代码中通过相对URL的方式引入文件

生成sourcemap，方便debug

加载并压缩文件

插件可以扩展 Webpack 的功能

在 Webpack 运行的生命周期中会广播出许多事件，Plugin 可以监听这些事件，在合适的时机通过 Webpack 提供的 API 改变输出结果。

用来自定义webpack打包过程的插件，插件含有apply方法，可以参与到webpack打包的各个流程

每一项是一个plugin实例，参数通过构造函数传入

简化 HTML 文件创建 (依赖于 html-loader)

模块热替换

把生成的文件进行压缩

骨架屏

吧ES6转成ES5

找到.vue文件，将template、css、js提取出来交给相应的loader进行处理

加载CSS，支持模块化，压缩，文件导入

把css代码注入到js中，通过DOM操作去加载css

通过ESlint检查JS代码

把文件输出到一个文件夹中，在代码中通过相对URL的方式引入文件

生成sourcemap，方便debug

加载并压缩文件

插件可以扩展 Webpack 的功能

在 Webpack 运行的生命周期