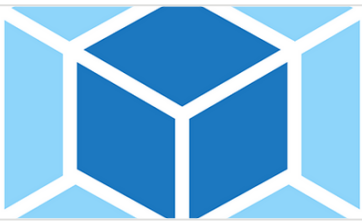
工程化押题

常见 loader 和 plugin 有哪些？二者区别是什么？

常见 loader

在 webpack 文档里写了：

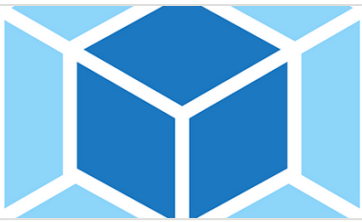


你可以记住：

1. `babel-loader` 把 JS/TS 变成 JS
2. `ts-loader` 把 TS 变成 JS，**并提示类型错误**
3. `markdown-loader` 把 markdown 变成 html
4. `html-loader` 把 html 变成 JS 字符串
5. `sass-loader` 把 SASS/SCSS 变成 CSS
6. `css-loader` 把 CSS 变成 JS 字符串
7. `style-loader` 把 JS 字符串变成 style 标签
8. `postcss-loader` 把 CSS 变成更优化的 CSS
9. `vue-loader` 把单文件组件（SFC）变成 JS 模块
10. `thread-loader` 用于多进程打包

常见 plugin

也在 webpack 文档里写了：



你可以记住这些：

1. `html-webpack-plugin` 用于创建 HTML 页面并自动引入 JS 和 CSS
2. `clean-webpack-plugin` 用于清理之前打包的残余文件
3. `mini-css-extract-plugin` 用于将 JS 中的 CSS 抽离成单独的 CSS 文件
4. `SplitChunksPlugin` 用于代码分包（Code Split）
5. `DllPlugin` + `DllReferencePlugin` 用于避免大依赖被频繁重新打包，大幅降低打包时间

6. `eslint-webpack-plugin` 用于检查代码中的错误
7. `DefinePlugin` 用于在 webpack config 里添加全局变量
8. `copy-webpack-plugin` 用于拷贝静态文件到 dist

二者的区别

- loader 是文件加载器（这句废话很重要）
  - 功能：能够对文件进行编译、优化、混淆（压缩）等，比如 babel-loader / vue-loader
  - 运行时机：在创建最终产物之前运行
- plugin 是 webpack 插件（这句废话也很重要）
  - 功能：能实现更多功能，比如定义全局变量、Code Split、加速编译等
  - 运行时机：在整个打包过程（以及前后）都能运行

webpack 如何解决开发时的跨域问题?

在开发时，我们的页面在 `localhost:8080`，JS 直接访问后端接口（如 `https://xiedaimala.com` 或 `http://localhost:3000`）会报跨域错误。

为了解决这个问题，可以在 webpack.config.js 中添加如下配置：

```
module.exports = {
  //...
  devServer: {
    proxy: {
      '/api': {
        target: 'http://xiedaimala.com',
        changeOrigin: true,
      },
    },
  },
};
```

此时，在 JS 中请求 `/api/users` 就会自动被代理到 `http://xiedaimala.com/api/users`。

如果希望请求中的 Origin 从 8080 修改为 xiedaimala.com，可以添加 `changeOrigin: true` 。

如果要访问的是 HTTPS API，那么就需要配置 HTTPS 证书，否则会报错。

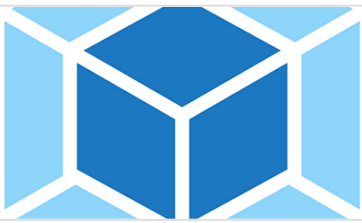
不过，如果在 target 下面添加 `secure: false` ，就可以不配置证书且忽略 HTTPS 报错。

总之，记住常用选项就行了。

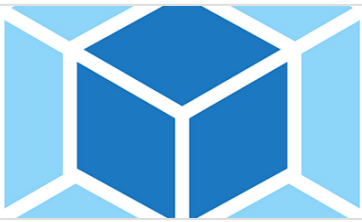
如何实现 tree-shaking?

这题属于拿着文档问面试者，欺负那些背不下文档的人。

> Tree Shaking | webpack
>
> webpack is a module bundler. Its main purpose is to bundle JavaScript files for usage in a browser, yet it is also capable of
>
> https://webpack.js.org/guides/tree-shaking/#conclusion



> Tree Shaking | webpack 中文文档
>
> tree shaking 是一个术语，通常用于描述移除 JavaScript 上下文中的未引用代码(dead-code)。它依赖于 ES2015 模块语法的 静态结
>
> https://webpack.docschina.org/guides/tree-shaking/#conclusion



是什么

tree-shaking 就是让没有用到的 JS 代码不打包，以减小包的体积。

怎么做

背下文档说的这几点：

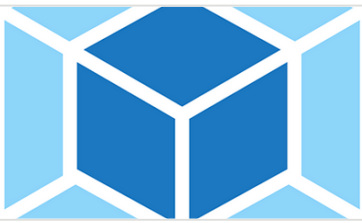
1. 怎么删
    a. 使用 ES Modules 语法（即 ES6 的 import 和 export 关键字）
    b. CommonJS 语法无法 tree-shaking（即 require 和 exports 语法）
    c. 引入的时候只引用需要的模块
        i. 要写 `import {cloneDeep} from 'lodash-es'` 因为方便 tree-shaking
        ii. 不要写 `import _ from 'lodash'` 因为会导致无法 tree-shaking 无用模块
2. 怎么不删：在 package.json 中配置 sideEffects，防止某些文件被删掉
    a. 比如我 import 了 x.js，而 x.js 只是添加了 window.x 属性，那么 x.js 就要放到 sideEffects 里
    b. 比如所有被 import 的 CSS 都要放在 sideEffects 里

3. 怎么开启：在 webpack config 中将 mode 设置为 production（开发环境没必要 tree-shaking）
    a. `mode: production` 给 webpack 加了非常多优化。

如何提高 webpack 构建速度?

webpack 文档写着呢：

> 构建性能 | webpack 中文文档
>
> 本指南包含一些改进构建/编译性能的实用技巧。 无论你是在 开发环境 还是在 生产环境 下运行构建脚本，以下最佳实践都会有
>
> https://webpack.docschina.org/guides/build-performance/



1. 使用 DllPlugin 将不常变化的代码提前打包，并复用，如 vue、react
2. 使用 thread-loader 或 HappyPack（过时）进行多线程打包
3. 处于开发环境时，在 webpack config 中将 cache 设为 true，也可用 cache-loader（过时）
4. 处于生产环境时，关闭不必要的环节，比如可以关闭 source map
5. 网传的 HardSourceWebpackPlugin 已经一年多没更新了，谨慎使用

webpack 与 vite 的区别是什么?

1. 开发环境区别
    a. vite 自己实现 server，不对代码打包，充分利用浏览器对 `<script type=module>` 的支持
        i. 假设 main.js 引入了 vue
        ii. 该 server 会把 `import { createApp } from 'vue'` 改为 `import { createApp } from "/node_modules/.vite/vue.js"` 这样浏览器就知道去哪里找 vue.js 了
    b. webpack-dev-server 常使用 babel-loader 基于内存打包，比 vite 慢很多很多很多
        i. 该 server 会把 vue.js 的代码（递归地）打包进 main.js

2. 生产环境区别
    a. vite 使用 rollup + esbuild 来打包 JS 代码
    b. webpack 使用 babel 来打包 JS 代码，比 esbuild 慢很多很多很多
        i. webpack 能使用 esbuild 吗？可以，你要自己配置（很麻烦）。
3. 文件处理时机
    a. vite 只会在你请求某个文件的时候处理该文件
    b. webpack 会提前打包好 main.js，等你请求的时候直接输出打包好的 JS 给你

目前已知 vite 的缺点有：

1. 热更新常常失败，原因不清楚
2. 有些功能 rollup 不支持，需要自己写 rollup 插件
3. 不支持非现代浏览器

webpack 怎么配置多页应用？

这是对应的 webpack config：

```
const HtmlWebpackPlugin = require('html-webpack-plugin');

module.exports = {
  entry: {
    app: './src/app.js',
    admin: './src/admin.js',
  },
  plugins: [
    new HtmlWebpackPlugin({
      filename: 'index.html',
      chunks: ['app']
    }),
    new HtmlWebpackPlugin({
      filename: 'admin.html',
      chunks: ['admin']
    })
  ],
};
```

但是，这样配置会有一个「重复打包」的问题：假设 app.js 和 admin.js 都引入了 vue.js，那么 vue.js 的代码既会打包进 app.js，也会打包进 admin.js。我们需要使用 `optimization.splitChunks` 将共同依赖单独打包成 common.js（HtmlWebpackPlugin 会自动引入 common.js）。

如何支持无限多页面呢?

写点 Node.js 代码不就实现了么?

```
const HtmlWebpackPlugin = require('html-webpack-plugin');
const fs = require('fs')
const path = require('path')

const filenames = fs.readdirSync('./src/pages')
  .filter(file => file.endsWith('.js'))
  .map(file => path.basename(file, '.js'))

const entries = filenames.reduce((result, name) => (
  { ...result, [name]: `./src/pages/${name}.js` }
), {})
const plugins = filenames.map((name) =>
  new HtmlWebpackPlugin({
    filename: name + '.html',
    chunks: [name]
  })
)

module.exports = {
  entry: {
    ...entries
  },
  plugins: [
    ...plugins
  ],
};
```

swc、esbuild 是什么?

swc

实现语言：Rust

功能：编译 JS/TS、打包 JS/TS

优势：比 babel 快很多很多很多（20倍以上）

能否集成进 webpack：能

使用者：Next.js、Parcel、Deno、Vercel、ByteDance、Tencent、Shopify……

做不到：

1. 对 TS 代码进行类型检查（用 tsc 可以）
2. 打包 CSS、SVG

esbuild

实现语言：Go

功能：同上

优势：比 babel 快很多很多很多很多很多很多（10~100倍）

能否集成进 webpack：能

使用者：vite、vuepress、snowpack、umijs、blitz.js 等

做不到：

1. 对 TS 代码进行类型检查
2. 打包 CSS、SVG