# Tree Shaking

### **一、背景介绍**

Webpack 是一个前端资源加载/打包工具。它将根据模块的依赖关系进行静态分析，然后将这些模块按照指定的规则生成对应的静态资源。([原理详细解说](https://segmentfault.com/a/1190000017890529?utm_source=tag-newest))

在性能优化方面，它主要做两件事：

1、加快打包速度:

优化loader文件(配置test/include/exclude)、cache-loader、插件HappyPack处理loader多线程等。

2、减小打包后的文件体积:

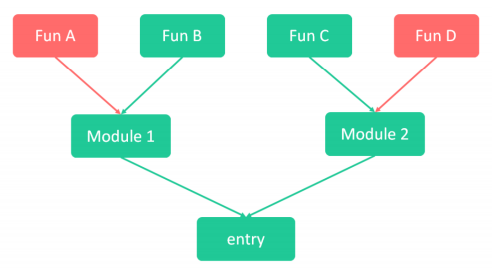
代码压缩([TerserWebpackPlugin](https://webpack.docschina.org/plugins/terser-webpack-plugin))、代码分割([optimazation.splitChunks](https://webpack.docschina.org/plugins/split-chunks-plugin#optimizationsplitchunks))、外部扩展[externals](https://webpack.docschina.org/configuration/externals#root)(cdn静态资源)、**Tree Shaking**。

### **二、什么是Tree Shaking？**

[Tree Shaking](https://webpack.docschina.org/guides/tree-shaking/#root) 的字面意思是「树摇」，是将项目中一些没有用到的模块「摇」掉。

本质上为了**消除未引用的代码**（dead code），减少加载文件体积的方式，使其整体执行时间更短。

例如：下图中的 函数 A 和 函数 D 是未引用的代码。Webpack 打包时，会将其“摇”掉。



* Dead Code：

1. 程序中没有执行的代码 (如return 之后的语句等)

2. 只声明，没有使用的代码

3. 只引入，没有使用的代码

* AST：

对 JS 代码进行语法分析后得出的抽象语法树 ([Abstract Syntax Tree](https://en.wikipedia.org/wiki/Abstract_syntax_tree))。AST语法树可以把一段 JS 代码的每一个语句都转化为树中的一个节点。

(在线可视化网站：[https://astexplorer.net/](https://link.zhihu.com/?target=https%3A//astexplorer.net/)，利用这个网站我们可以很清晰的看到各种语言的 AST 结构)

像 Uglify 或 Terser 这样的 JS 压缩工具也能来做死代码消除(Dead Code Elimination)，为什么还需要 Tree Shaking？

原因是 Uglify 目前不能跨文件去做死代码消除，只会对单个文件的 AST语法树 进行分析，无法得知 export 的模块是否会被其他文件使用。

### **三、Tree Shaking 原理**

简单来说，基于 ES6 Module 的静态引入，对 import 的代码进行静态分析，如果发现没有被用到的部分就不再 export。

webpack 的实现是把所有 import 标记为有使用/无使用两种，Tree Shaking 不会直接把没有用到的代码删除，真正删除代码的是 Uglify 或 Terser 这样的 JS 压缩工具。

* ES6 Module

静态引入 (import)、编译时输出接口、输出的是模块的引用、import 或 export 的模块名只能是字符串常量。

* CommonJS

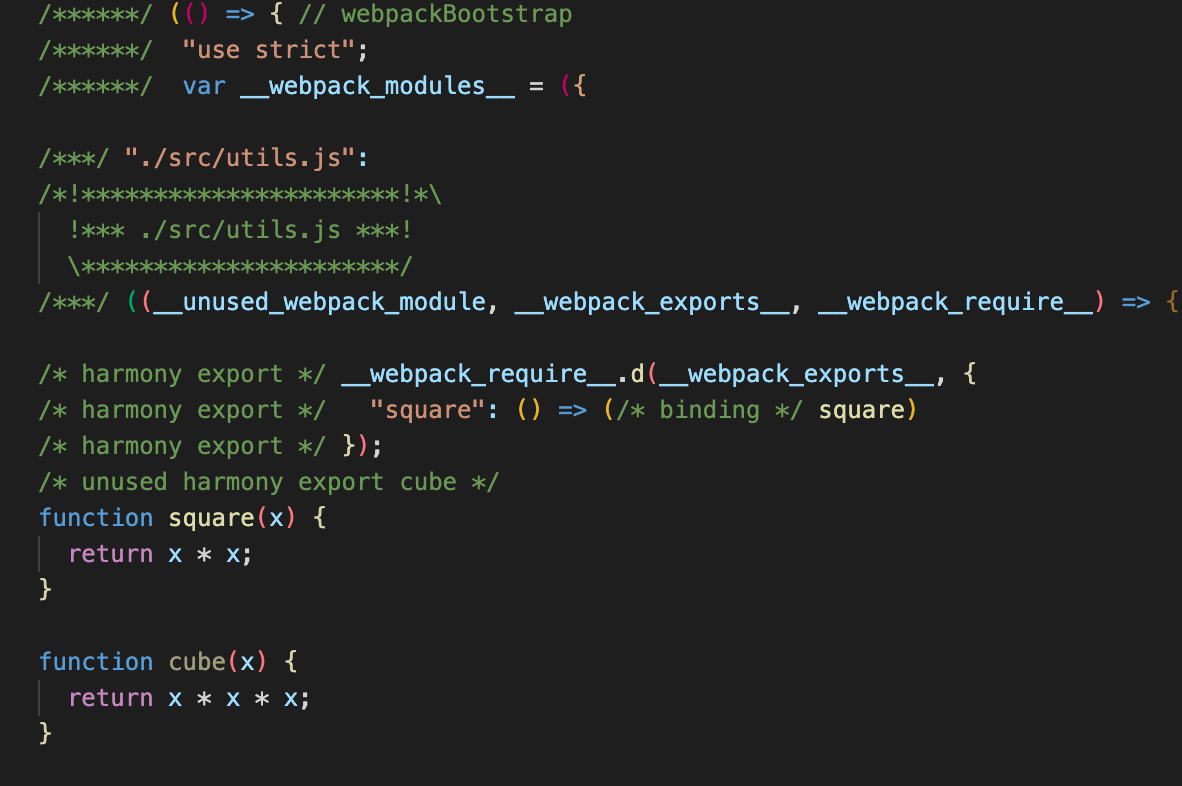
动态引入 (require())、运行时加载、输出是值的拷贝。

Tree-Shaking 实现分两步走：

Step 1：标记出模块导出值哪些没有被用过；

webpack 负责对代码进行标记，把 import & export 标记为3类：

* 所有 import 标记为： /\* harmony import \*/
* 被使用过的 export 标记为：/\* harmony export \*/
* 未被使用过的 import 标记为: /\* unused harmony export [FuncName] \*/，其中[FuncName] 为 export 的方法名称。



这里去 require 其他模块的函数主要是 \_\_webpack\_require\_\_, 加载对应的模块

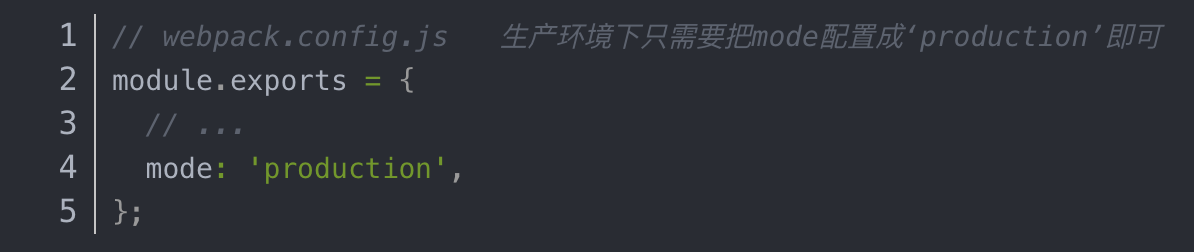
Step 2：使用 Terser (或者其他类似的工具) 进行代码精简，删除掉没有被用到的导出语句。

### **四、如何使用 Tree Shaking**

* 开发模式：usedExports 、sideEffects



生产模式：Tree Shaking 自动开启



PS：特别注意 devtool 的配置，想要使用 Tree Shaking，devtool 只能使用以下几种模式：

source-map | inline-source-map | hidden-source-map | nosources-source-map

eval 模式，将 JS 输出为字符串（不是 ES Modules 规范），导致 Tree Shaking 失效



* 使用 ES6 模块语法（即 import 和 export）
* 确保没有编译器将 ES6模块语法 转换为 CommonJS模块。比如在使用babel-loader编译代码时，设置modules为fasle，表示不对ES6模块进行处理。
* { // .babelrc配置
* "presets": [
* [
* "@babel/preset-env", // ES2015、ES2016、ES2017的语法插件集
* {
* "modules": false
* }
* ],
* ],

### **五、副作用 sideEffects**

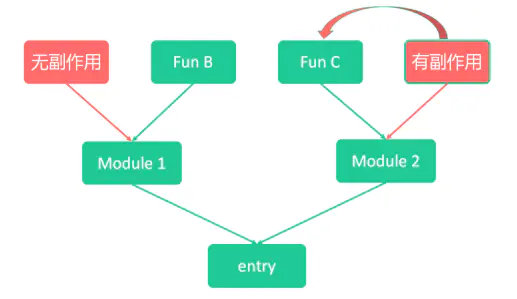
副作用是函数式编程的一个概念，是指当调用函数时，除了返回函数值之外，还会对调用函数产生附加的影响。

* 无副作用：如果一个模块单纯的导入导出变量，是无副作用；
* 有副作用：如果一个模块还修改其他模块或者全局的一些东西，就有副作用；

修改全局变量、CSS 的引入等。

**sideEffects的作用：把未使用但无副作用的模块一并删除。**

对于没有副作用的模块，未使用时代码不会被打包 (相当于压缩了输出内容)；如果模块或函数有副作用，不能随便删除。如下图，删除有副作用的函数，可能影响正在使用的 Fun C：

****

标识代码是否有副作用, 通过 **package.json** 的 "**sideEffects**" 属性来实现：

* **false**：所有代码都没有副作用（告诉 webpack 可以安全地删除未用的 exports）
* **true**：所有代码都有副作用
* **数组**：告诉 webpack 哪些模块有副作用，不进行删除。例如：['./src/wp.js', '\*.css']，数组中可以使用相对路径，绝对路径或正则。

**Tip：**

注意，所有导入文件都会受到 tree shaking 的影响。这意味着，如果在项目中使用类似 css-loader 并 import 一个 CSS 文件，则需要将其添加到 side effect 列表中，以免在生产模式中无意中将它删除。

{

"name": "tree-shaking-project",

"sideEffects": [

"\*.css",

"\*.less"

],

}