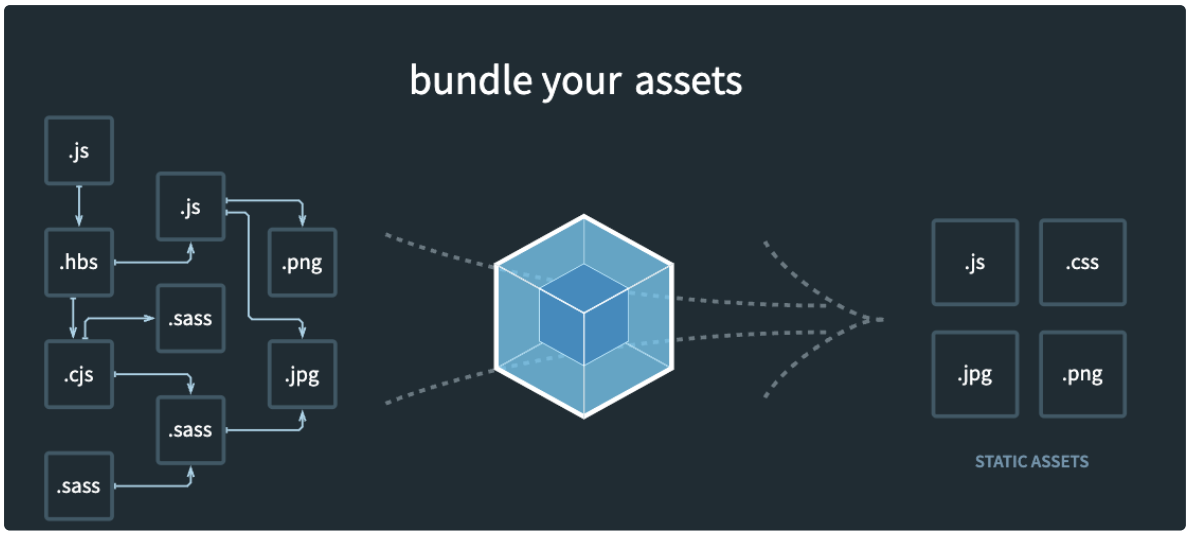
页面性能： Treeshaking

Tree Shaking 概念

Tree-shaking (摇树) 是一个术语，通常指通过打包工具"摇"我们的代码，将未引用代码 (Dead Code) "摇" 掉。本质是消除无用的 js 代码。无用代码消除在广泛存在于传统的编程语言编译器中，编译器可以判断出某些代码根本不影响输出，然后消除这些代码，这个称之为 DCE（dead code elimination）

在 Webpack 项目中，有一个入口文件，相当于一棵树的主干，入口文件有很多依赖的模块，相当于树枝，虽然依赖了某些模块，但其实只使用其中的某些方法，通过 Tree Shaking ，将没有使用的方法摇掉，这样来达到删除无用代码的目的。



Tree Shaking 的原理

Tree Shaking 在去除代码冗余的过程中，程序会从入口文件出发，扫描所有的模块依赖，以及模块的子依赖，然后将它们链接起来形成一个“抽象语法树”(AST)。随后，运行所有代码，查看哪些代码是用到过的，做好标记。最后，再将“抽象语法树”中没有用到的代码“摇落”。经历这样一个过程后，就去除了没有用到的代码。

> 可以参考原理篇： 手写 Rollup

为什么 CommonJS 模块化无法 Treeshaking

CommonJS 与 ES6 Module 模块的依赖的区别在于，CommonJS 是**动态的**，ES6 Module 是**静态的**。

CommonJS 导入时，`require` 的路径参数是支持表达式的，路径在代码执行时是可以动态改变的，所以如果在代码编译阶段就建立各个模块的依赖关系，那么一定是不准确的，只有在代码运行了以后，才可以真正确认模块的依赖关系，因此说 CommonJS 是动态的。

JavaScript

```
// CommonJS
if(Math.random() > 0.5) {
  cosnt a = require('./module.js')
}
```

Webpack 精准使用 TreeShaking 的条件

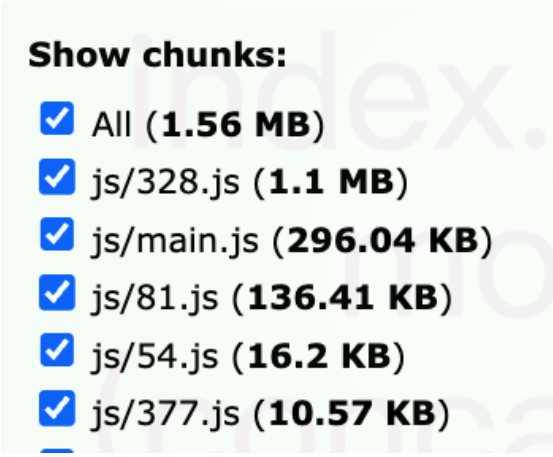
- 使用 ES6 Module 语法（即 `import` 和 `export`）。
- 确保没有 `@babel/preset-env` 等工具将 ES6Module 语法转换为 CommonJS 模块。`module : false`
- 使用支持 Tree Shaking 的第三方库。

开启 Treeshaking

webpack.config.js

```JavaScript
//
optimization: {
  usedExports: true
 }
```

优化前



优化后