webpack

mode	production	development	none
process.env.NODE_ENV	'production'	'development'	❌
devtool （控制是否生成，以及如何生成 source map）	❌	'eval' 📉 📉 📉 打包更慢，包体积更大； 📈 📈 📈 更好的调试体验；	❌
cache （缓存模块，避免在未更改时重新构建它们，改善构建速度，只在 watch 模式下有用）	❌	✅ 📉 内存占用更多； 📈 📈 📈 更快的增量打包；	❌
output.pathinfo （输出包中是否包含模块注释信息）	❌	✅ 📉 包更大，并且泄露路径信息； 📈 提高了包代码的可读性；	❌
performance （性能设置）	✅ 📉 算法成本； 📈 包过大时，会警告；	❌	❌
optimization.removeAvailableModules （删除已可用模块） 📉 📉 算法成本； 📈 📈 📈 减小包体积；	✅	✅	✅
optimization.removeEmptyChunks （删除空模块） 📉 算法成本； 📈 📈 📈 减小包体积；	✅	✅	✅

mode	production	development	none
optimization.mergeDuplicateChunks （合并相等块） 📉 算法成本； 📈 📈 📈 更少的请求与下载；	✅	✅	✅
optimization.flagIncludedChunks （标记块是否是其它块的子集， 控制加载块的大小，当加载较大块时，不加载其子集）	✅ 📉 算法成本； 📈 📈 更少的请求与下载；	❌	❌
optimization.occurrenceOrder （标记模块的加载顺序，使初始包更小）	✅ 📉 算法成本； 📈 更小的包体积；	❌	❌
optimization.providedExports （尽可能确定每个模块的导出信息） 📉 算法成本； 📈 包体积及其它优化的需求；	✅	✅	✅
optimization.usedExports （不会为未使用的导出生成导出，最小化的消除死代码，可被其他优化或代码生成所使用）	✅ 📉 📉 算法成本； 📈 📈 更小的包体积；	❌	❌
optimization.sideEffects （识别 package.json 或者 module.rules 的 sideEffects 标志（纯的 ES2015 模块），安全地删除未用到的 export 导出。 这取决于 optimization.providedExports和 optimization.usedExports。 这些依赖性有一定的成本，但是由于减少了代码生成，因此消除模块会对性能产生积极影响）	✅ 📉 算法成本； 📈 📈 📈 更小的包体积，更小的代码生成；		❌
optimization.concatenateModules （查找模块图中可以安全的连接到其它模块的片段，取决于optimization.providedExports和 optimization.usedExports）	✅ 📉 📉 📉 额外的解析，范围分析和标识符重命名（性能）； 📈 📈 📈 提升运行时性能，减小包大小；	❌	❌

mode	production	development	none
optimization.splitChunks （拆分块，默认只针对异步块进行拆分） 📉 算法成本，额外的请求； 📈 📈 📈 更少的代码生成，更好的缓存，更少的下载；	✅	✅	✅
optimization.runtimeChunk （为webpack运行时代码和块清单创建一个单独的块。该块应内联到HTML中） 📉 📉 📉 更大的HTML文件； 📈 📈 📈 更好的缓存；	✅	✅	✅
optimization.noEmitOnErrors （不输出编译错误的包）	✅ 📉 无法使用应用程序的工作部分； 📈 没有坏包；	❌	❌
optimization.nameModules （以名称固化 module id）	❌	✅ 📉 更大包体积； 📈 更好的错误报告和调试；	❌
optimization.namedChunks （以名称固化 chunk id）	❌	✅ 📉 更大包体积； 📈 更好的错误报告和调试；	❌
optimization.nodeEnv （设置 process.env.NODE_ENV）	'production' 📉 区别开发环境与生产环境； 📈 📈 包大小，运行时性能；	'development' 📉 区别开发环境与生产环境； 📈 📈 包大小，运行时性能；	❌
optimization.minimize （使用 optimization.minimizer || TerserPlugin 来最小化包）	✅ 📉 📉 📉 更慢； 📈 📈 📈 包体积；	❌	❌
optimize.ModuleConcatenationPlugin （预编译所有模块到一个闭包中，提升代码在浏览器中的执行速度）	✅	❌	❌