Rollup JS module bundler

START HERE

`rollup -c`
Build with the config file

`rollup -c --environment BUILD:production`
Pass settings in

`rollup -i <entry> -o <file> -f es`
Bundle one file

`rollup -c -p <plugin>`
Add a plugin

`rollup -i <entry> -d <dir>`
Bundle to a folder

`rollup -i <entry> -o <file> -m`
Add a sourcemap

`rollup -c -w`
Rebuild on change

`rollup --help`
List all flags

Entry input → `resolve + load` → **Module graph** plugins transform → `tree-shake` → **Chunks** only used code → `render format` → **Output** file or dir

CONCEPTS

Rollup bundles ES modules; resolving npm packages or CommonJS needs plugins (node-resolve, commonjs).

Tree-shaking works on static import/export; mark pure calls with /*#__PURE__*/ and set sideEffects in package.json.

CLI flags override the same options in the config file.

BUILD

`rollup -c`
Build using rollup.config.(mjs/cjs/js)

`rollup -c <file>`
Build using a named config

`rollup -i <entry> -o <file>`
Bundle entry into one file

`rollup <entry> -o <file>`
Same, entry as plain argument

`rollup -i <entry> -d <dir>`
Write chunks to a directory

`rollup -i <entry> -o <file> -f <fmt>`
Pick output format
es, cjs, iife, umd, amd or system.

`rollup -c -w`
Watch and rebuild

`rollup -c --silent`
Hide warnings

`rollup -c --failAfterWarnings`
Fail the build on warnings

`rollup -v`
Show version

OUTPUT OPTIONS

`-f iife -n <Name>`
Browser global with name

`-f umd -n <Name>`
Universal module, needs a name

`-g <id>:<Global>`
Map external to global (iife/umd)

`-m`
Write sourcemap file

OUTPUT OPTIONS (cont.)

`-m inline`
Embed the sourcemap in the bundle

`--sourcemapExcludeSources`
Leave source text out of maps

`--sourcemapBaseUrl <url>`
Emit absolute sourcemap URLs

`--exports <mode>`
Set export mode
auto, default, named or none. Mixed default and named exports in cjs need named.

`--entryFileNames <pattern>`
Name entry chunks
For example [name]-[hash].js

`--chunkFileNames <pattern>`
Name shared chunks

`--assetFileNames <pattern>`
Name emitted assets

`--banner <text>`
Text at the top of the bundle

`--footer <text>`
Text at the end of the bundle

`--compact`
Minify wrapper code only

`--no-esModule`
Skip the __esModule flag

`--generatedCode <preset>`
Choose es5 or es2015 syntax

DEPENDENCIES

`-e <ids>`
Keep modules out of the bundle
Comma-separated, e.g. -e react,react-dom

`--no-externalLiveBindings`
Simplify generated code for externals

`--preserveSymlinks`
Do not follow symlinks

`--shimMissingExports`
Replace missing exports with a shim

`--inlineDynamicImports`
Fold dynamic imports into one file

`--preserveModules`
One output file per module

`--preserveModulesRoot <path>`
Strip this prefix from module paths

PLUGINS AND CONFIG

`-p <plugin>`
Use a plugin by name
Repeat or comma-separate; -p node-resolve loads @rollup/plugin-node-resolve.

`-p '{name: ..}'`
Define a small inline plugin

`--environment <KEY[:val]>`
Set process.env values for the config
BUILD sets "true"; BUILD:production sets a string.

`--configPlugin <plugin>`
Transpile the config file (TypeScript)

`--bundleConfigAsCjs`
Allow __dirname and require in config

`-c node:<name>`
Load shared config from node_modules

`--config<Name>`
Custom flag read by a config function
Must start with config; the function gets commandLineArgs.

`echo '<code>' | rollup -f cjs -o <file>`
Bundle code from stdin

TREE-SHAKING

`--no-treeshake`
Turn tree-shaking off

`--no-treeshake.annotations`
Ignore /*#__PURE__*/ comments

`--treeshake.manualPureFunctions <names>`
Declare functions as pure

`--no-treeshake.moduleSideEffects`
Assume imports have no side effects

`--no-treeshake.propertyReadSideEffects`
Ignore side effects of property reads

`--no-preserveEntrySignatures`
Allow entries to lose exports

`--no-hoistTransitiveImports`
Do not hoist imports into entries

WATCH MODE

`-w`
Watch files, rebuild on change

WATCH MODE (cont.)

`--watch.include <files>`
Only watch these files

`--watch.exclude <files>`
Do not watch these files

`--watch.buildDelay <ms>`
Throttle rebuilds

`--no-watch.clearScreen`
Keep the terminal on rebuild

`--watch.onStart <cmd>`
Run a shell command on START

`--watch.onBundleEnd <cmd>`
Run a command after each bundle

LOGGING AND DEBUGGING

`--logLevel <level>`
Choose which logs show
debug, info, warn or silent.

`--filterLogs <filter>`
Show only matching logs
e.g. code:CIRCULAR_DEPENDENCY; ! negates.

`--perf`
Print timing per phase

RECIPES

Library with ES and CJS builds

```
# rollup.config.mjs
import { defineConfig } from 'rollup';
export default defineConfig({
  input: 'src/index.js',
  external: ['react'],
  output: [
    { file: 'dist/index.mjs', format: 'es', sourcemap: true },
    { file: 'dist/index.cjs', format: 'cjs', sourcemap: true }
  ]
});
```

Browser bundle with resolve and commonjs

```
npm i -D rollup @rollup/plugin-node-resolve @rollup/plugin-commonjs
# in the config: plugins: [resolve(), commonjs()]
rollup -c --environment BUILD:production
```

TypeScript config file

```
npm i -D @rollup/plugin-typescript
rollup -c rollup.config.ts --configPlugin typescript
```

Quick IIFE for a script tag

```
rollup -i src/main.js -o dist/app.js -f iife -n App \
  -g jquery:jQuery -e jquery -m
```