esbuild Fast JS bundler

Entry points app.ts, ... → `resolve` → **Module graph** imports, packages → `loader + target` → **Transformed** TS/JSX to JS, minify → `format` → **Output** --outfile / --outdir

CONCEPTS

Without --bundle esbuild only transforms each file (TS/JSX to JS) and does not follow imports. With no --outfile it prints to stdout.

Flag forms: --flag, --flag=value for single values, --flag:key=value for repeatable ones (--loader:.png=file, --external:react).

Quote globs like "src/**/*.ts" so esbuild expands them instead of your shell.

BUILD

`esbuild <entry> --bundle --outfile=<file>`
Bundle one entry into a file

`esbuild <a> <b> --bundle --outdir=<dir>`
Bundle several entries into a directory

`--minify`
Minify whitespace, syntax and names

`--sourcemap`
Write a linked .map file

BUILD (cont.)

`--sourcemap=inline`
Embed the source map as a data URL

`--tsconfig=<file>`
Use this tsconfig instead of discovery

PLATFORM AND FORMAT

`--platform=browser|node|neutral`
Pick target environment (default browser)

`--format=iife|cjs|esm`
Output module format

`--target=es2020,chrome100`
Lower syntax for these engines

`--external:<pkg>`
Leave a module out of the bundle

`--packages=external`
Keep every package import external

`--splitting --format=esm --outdir=<dir>`
Share code between entries in chunks

`--global-name=<name>`
Global variable name for iife output

LOADERS AND ASSETS

`--loader:.<ext>=<loader>`
Choose how a file type is read

`--loader:.png=dataurl`
Inline images as data URLs

`--loader:.png=file`
Copy asset and import its path

`--loader:.js=jsx`
Allow JSX in .js files

`--jsx=automatic`
Use the React 17+ JSX runtime

`--entry-names=<template>`
Output name pattern for entries

OPTIMIZE

`--define:<name>=<value>`
Replace an identifier with a constant

`--define:process.env.NODE_ENV=\"production\"`
Set NODE_ENV for dead-code removal

`--drop:console`
Remove console.* calls

`--drop:debugger`
Remove debugger statements

OPTIMIZE (cont.)

`--keep-names`
Keep function and class .name

WATCH AND SERVE

`--watch`
Rebuild when inputs change

`--watch=forever`
Keep watching after stdin closes

`--servedir=<dir>`
Serve a directory plus build output

`--serve=<port>`
Serve build output on a port

ANALYZE AND LOGS

`--metafile=<file>`
Write build metadata JSON

`--analyze`
Print a size report of the bundle

`--analyze=verbose`
Detailed report with import chains

`--log-level=<level>`
verbose|debug|info|warning|error|silent

RESOLUTION

`--alias:<from>=<to>`
Rewrite an import path or package

`--conditions=<names>`
Extra package exports conditions

TRANSFORM STDIN

`esbuild --loader=ts < <in.ts>`
Transform TypeScript from stdin

`echo '<code>' | esbuild --minify`
Minify a quick snippet

RECIPES

Production browser bundle

```
esbuild src/index.ts --bundle --minify --sourcemap \
  --target=es2020 --define:process.env.NODE_ENV=\"production\" \
  --outfile=dist/app.js
```

Node CLI with dependencies external

```
esbuild src/cli.ts --bundle --platform=node --target=node20 \
  --packages=external --format=esm --outfile=dist/cli.mjs
```