Babel JavaScript compiler CLI

CONCEPTS

Install @babel/cli and @babel/core in the project. Without them npx may fetch an ancient Babel 6 'babel'.

Source modern JS, JSX, TS
↓ parse
Plugins transform syntax
↓ grouped by
Presets env, react, typescript
↓ generate
Output plus source map

Babel only transforms syntax. It does not type-check TypeScript and does not bundle modules.

Pass plugin and preset names without the prefix: env means @babel/preset-env.

INPUT AND OUTPUT

`babel <file>`
Compile one file to stdout

`babel <file> -o <out>`
Compile one file into a file

`babel <dir> -d <outdir>`
Compile a directory tree

`babel <a> <b> -d <outdir>`
Compile several files or dirs

`babel <dir> -o <out>`
Concatenate a dir into one file

`babel -f <name> < <in> > <out>`
Compile stdin

`babel <dir> -d <outdir> --delete-dir-on-start`
Clean the output dir first

`babel <dir> -d <outdir> -D`
Copy non-compilable files

SELECTING FILES

`babel <dir> -d <outdir> -x .ts,.tsx,.js`
Compile these extensions

`babel <dir> -d <outdir> --ignore "<glob>"`
Skip files matching a glob

`babel <dir> -d <outdir> --ignore "<dir>/**/*.test.js"`
Skip test files

SELECTING FILES (cont.)

`babel <dir> -d <outdir> --keep-file-extension`
Keep input extensions

`babel <dir> -d <outdir> --out-file-extension .mjs`
Set the output extension

PRESETS AND PLUGINS

`babel <file> --presets @babel/preset-env`
Use a preset

`babel <file> --presets env,react`
Several presets, comma-separated

`babel <file> --plugins <plugin>`
Use a plugin

`babel <file> --presets @babel/preset-typescript -x .ts`
Strip TypeScript types

`babel <file> --presets @babel/preset-react`
Compile JSX

`babel <file> --env-name <name>`
Pick the env block from the config

`BABEL_ENV=production babel <dir> -d <outdir>`
Select the production env

CONFIG FILES

`babel <file> --config-file <path>`
Use a specific config file

`babel <file> --no-babelrc`
Ignore .babelrc files

`babel <file> --root-mode upward`
Find babel.config.* in a parent dir

Config file kinds

File	Scope
`babel.config.json`	Project-wide; also covers node_modules
`.babelrc.json`	Per package; applies to files below it
`package.json 'babel'`	Same as .babelrc

CONFIG FILES (cont.)

babel.config.json

```
{
  "presets": [
    ["@babel/preset-env",
{ "targets": "defaults" }]
  ]
}
```

Targets decide how much preset-env rewrites. Set them in the config or a browserslist file.

WATCH AND OUTPUT CONTROL

`babel <dir> -d <outdir> -w`
Recompile on change

`babel <dir> -d <outdir> -w --skip-initial-build`
Watch without the first build

`babel <dir> -d <outdir> --verbose`
Log every file

`babel <dir> -d <outdir> --quiet`
Print nothing

`babel <file> --no-comments`
Strip comments

SOURCE MAPS

`babel <dir> -d <outdir> -s`
Write .map files

`babel <dir> -d <outdir> -s inline`
Embed the map in the output

PACKAGE SCRIPTS

package.json scripts

```
"scripts": {
  "build": "babel src -d lib -s",
  "dev": "babel src -d lib -w"
}
```

`npx babel --version`
Show CLI and core versions

`npx babel --help`
List all options

RECIPES

Watch and copy assets

```
npx babel src -d dist -D -w
```

Per-env build

```
BABEL_ENV=production npx babel
src -d dist --delete-dir-on-start
--no-comments
```