tsx Run TypeScript directly

START HERE

`npx tsx <file.ts>`
Run a TS file, no setup

`node --import tsx <file.ts>`
Plain node with TS support

`tsx <file.ts> <args>`
Run with arguments

`tsc --noEmit`
Type-check (tsx will not)

`tsx watch <file.ts>`
Rerun on change

`tsx`
Open a TS-aware REPL

`npm i -D tsx`
Add to a project

`tsx --tsconfig <path> <file.ts>`
Use another tsconfig

file.ts your source → `esbuild strips types` → **tsx loader** on top of node → `JS, no type check` → **node** runs the code → `tsc --noEmit` → **Type errors** found by tsc only

CONCEPTS

tsx is node with a TypeScript loader: it strips types with esbuild and never type-checks, so run tsc --noEmit for errors.

It handles ESM and CommonJS, .ts/.tsx/.mts/.cts files and tsconfig paths in one go, with no build step.

Use tsx for scripts and dev runs; for production, compile with tsc or bundle with esbuild.

INSTALL

`npx tsx <file.ts>`
Run once without installing

`npm install -D tsx`
Add to the project
Then call tsx from package.json scripts or via npx tsx.

`npm install -g tsx`
Install globally

INSTALL (cont.)

`tsx --version`
Show tsx and node versions

RUN

`tsx <file.ts>`
Run a TypeScript file

`tsx <file.ts> <args>`
Pass arguments to the script
Read them from process.argv.slice(2).

`tsx -e "<code>"`
Evaluate a TS snippet

`tsx -p "<expr>"`
Evaluate and print the result

`tsx --import <module> <file.ts>`
Preload a module first

`tsx --env-file=<file> <file.ts>`
Load env vars from a file
Node flag passed through; needs node 20.6+.

`tsx --no-cache <file.ts>`
Skip the compile cache

RUN (cont.)

`tsx --enable-source-maps <file.ts>`
Source-mapped stack traces

`./script.ts`
Run as an executable script
First line: #!/usr/bin/env tsx, then chmod +x.

Any node flag goes before the script name and is passed to node.

WATCH

`tsx watch <file.ts>`
Rerun when imported files change

`tsx watch --include "<glob>" <file.ts>`
Also watch extra paths
Quote globs so the shell does not expand them.

`tsx watch --exclude "<glob>" <file.ts>`
Do not watch these paths

WATCH (cont.)

`tsx watch --clear-screen=false <file.ts>`
Keep earlier output

`Enter (in watch mode)`
Rerun by hand

Watch ignores node_modules, dist, vendor and hidden folders by default; add them with --include.

CONFIG

`tsx --tsconfig <path> <file.ts>`
Use a specific tsconfig

`TSX_TSCONFIG_PATH=<path> tsx <file.ts>`
Same, set via the environment

`TSX_DISABLE_CACHE=1 tsx <file.ts>`
Disable the cache via the environment

CONFIG (cont.)

tsx reads paths, jsx and target from tsconfig.json, but ignores type options such as strict.

USE WITH NODE

`node --import tsx <file.ts>`
Run TS with plain node
Also works in NODE_OPTIONS or a shebang.

`node --import tsx --test`
Run the node test runner on TS

`tsx --test`
Same, shorter

`node --require tsx/cjs <file.ts>`
Register for CommonJS only

`import { tsImport } from 'tsx/esm/api'`
Import one TS file from JS
await tsImport('./file.ts', import.meta.url); not cached.

USE WITH NODE (cont.)

`tsx --inspect <file.ts>`
Attach a debugger

`tsx -i`
Always enter the REPL

RECIPES

package.json scripts

```
"dev": "tsx watch src/
index.ts",
"start": "tsx src/index.ts",
"typecheck": "tsc --noEmit"
```

Dev run with env file

```
tsx watch --env-file=.env --
exclude "./logs/**" src/
server.ts
```

Type-check and run

```
tsc --noEmit && tsx src/job.ts
<arg>
```

Tests with node runner

```
node --import tsx --test "src/
**/*.test.ts"
```