TypeScript compiler (tsc) Typed JS compiler

START HERE

`tsc --init`
Create a tsconfig.json

`tsc -p <path>`
Use a specific config

`tsc`
Compile the project

`tsc --showConfig`
Print the resolved config

`tsc --noEmit`
Type-check only

`tsc --build`
Build project references

`tsc --watch`
Recompile on change

`tsc --ignoreConfig <file>`
Compile files, skip config

tsconfig.json options + files → `tsc` → **Type check** errors, exit code 1 → `emit` → **JavaScript** outDir

CONCEPTS

tsc finds tsconfig.json by walking up from the current directory. Naming files on the command line makes it refuse to run unless you add --ignoreConfig.

Type errors do not stop emit unless noEmitOnError is set. A non-zero exit code only tells you there were errors.

These entries describe TypeScript 7 (native compiler). 7 removed baseUrl, target es5 and moduleResolution node10, and made strict the default. On 5.x, check with tsc --version.

PROJECT SETUP

`tsc --init`
Create a tsconfig.json
Refuses if one already exists in the directory.

`tsc`
Compile using the nearest tsconfig.json

`tsc -p <path>`
Compile a config file or directory
Also written --project.

`tsc --noEmit`
Type-check without writing files

`tsc --watch`
Recompile when files change
Add --preserveWatchOutput to keep earlier output on screen.

`tsc --ignoreConfig <file>`
Compile given files, ignore tsconfig

`tsc --version`
Show the compiler version

`tsc --help --all`
List every compiler option

TSCONFIG KEYS

`"extends": "<file>"`
Inherit another config

`"include": ["src"]`
Files to compile

`"exclude": ["dist"]`
Files to skip

`"files": ["a.ts"]`
Explicit file list

`"references": [{"path": "<dir>"}]`
Link another project

`"compilerOptions": {}`
Where all options below go

LANGUAGE & MODULES

`--target esnext`
Syntax level of emitted JS
Default is es2025 in 7; es5 is no longer allowed.

`--module nodenext`
Module format for Node projects

`--module preserve`
Keep imports as written (bundlers)

`--moduleResolution bundler`
Resolution for bundler setups

`--lib <list>`
Built-in type libraries to include
Add dom for browser code, e.g. --lib esnext,dom.

`--types <list>`
Which @types packages to load
An empty list loads none; add node explicitly.

`--jsx react-jsx`
How JSX is compiled

`--resolveJsonModule`
Allow importing .json files

`--allowJs`
Include .js files in the project

`--checkJs`
Report errors in .js files

`--moduleDetection force`
Treat every file as a module

`--allowImportingTsExtensions`
Allow imports ending in .ts
Needs --noEmit or --emitDeclarationOnly.

`--rewriteRelativeImportExtensions`
Rewrite .ts imports to .js on emit

`--rootDirs <list>`
Merge directories for resolution

`"paths": {"@/*": ["./src/*"]}`
Alias import prefixes
Only affects type checking; the runtime or bundler must resolve aliases too.

TYPE CHECKING

`--strict`
Enable all strict checks
On by default in 7; set false to opt out.

`--noUncheckedIndexedAccess`
Indexed access may be undefined

`--exactOptionalPropertyTypes`
Optional means absent, not undefined

TYPE CHECKING (cont.)

`--noImplicitReturns`
Every code path must return

`--noImplicitOverride`
Require override on subclass methods

`--noUnusedLocals`
Error on unused variables

`--noUnusedParameters`
Error on unused parameters

`--noFallthroughCasesInSwitch`
Error on switch fallthrough

`--skipLibCheck`
Skip checking .d.ts files
Speeds up checks and hides errors inside dependencies.

`--noCheck`
Skip type checking, still emit

EMIT

`--outDir <dir>`
Write output files here

`--rootDir <dir>`
Root of the input tree
Decides how outDir mirrors the source layout.

`--declaration`
Emit .d.ts files

`--emitDeclarationOnly`
Emit only .d.ts files

`--sourceMap`
Emit .js.map files

`--declarationMap`
Emit .d.ts.map files

`--noEmitOnError`
Emit nothing if there are errors

`--verbatimModuleSyntax`
Keep imports/exports as written
Requires import type for type-only imports.

`--isolatedModules`
Require per-file safe code

`--esModuleInterop`
Smoother CommonJS default imports
On by default in 7.

`--erasableSyntaxOnly`
Forbid enums and namespaces
Keeps code compatible with Node type stripping.

PROJECTS & BUILDS

`tsc --build`
Build projects whose inputs changed
Short form -b; follows references.

`tsc -b --watch`
Rebuild referenced projects on change

PROJECTS & BUILDS (cont.)

`tsc -b --clean`
Delete build outputs

`tsc -b --force`
Rebuild even if up to date

`tsc -b --dry`
Show what would be built

`tsc -b --verbose`
Explain why projects rebuild

`--composite`
Make the project referenceable
Implies declaration and incremental.

`--incremental`
Cache state in .tsbuildinfo

DEBUGGING

`tsc --showConfig`
Print the final merged config
Shows what extends and defaults resolve to.

`tsc --listFiles`
List every file in the program

`tsc --listFilesOnly`
List files and stop

`tsc --explainFiles`
Why each file was included

`tsc --traceResolution`
Log module lookup steps
Pipe through grep for one specifier.

`tsc --listEmittedFiles`
Print files written

`tsc --diagnostics`
Show timing and memory use

RECIPES

Minimal Node tsconfig

```
{
  "compilerOptions": {
    "module": "nodenext",
    "target": "esnext",
    "outDir": "dist",
    "types": ["node"],
    "skipLibCheck": true
  },
  "include": ["src"]
}
```

Type-check in CI, build with a bundler

```
# package.json scripts
"typecheck": "tsc --noEmit",
"build": "tsc"
```

Why is this file in my build?

```
tsc --explainFiles | grep -A3
<name>
tsc --traceResolution | grep
<module>
```

Monorepo with references

```
# root tsconfig.json
{ "files": [],
  "references": [
    {"path": "packages/core"},
    {"path": "packages/
app"} ] }
tsc -b
```