Appendix F. TSC Compiler Flags for Safety

TIP

For a complete list of available compiler flags, head over to the TypeScript Handbook website.

Each TypeScript release introduces new checks that you can enable to squeeze even more safety out of your code. Some of these flags—prefixed with `strict`—are included as part of the `strict` flag; or, you can opt into `strict` flags one at a time. Table F-1 lists the compiler flags related to safety that are available at the time of writing.

Table F-1. TSC safety flags

Flag	Description
`alwaysStrict`	Emit `'use strict'`.
`noEmitOnError`	Don’t emit JavaScript when your code has type errors.
`noFallthroughCasesInSwitch`	Make sure that every `switch` case either returns a value or breaks.
`noImplicitAny`	Error when a variable’s type is inferred as `any`.
`noImplicitReturns`	Make sure that every code path in every function explicitly returns. See “Totality”.
`noImplicitThis`	Error when you use `this` in a function without explicitly annotating the `this` type. See “Typing this”.
`noUnusedLocals`	Warn about unused local variables.
`noUnusedParameters`	Warn about unused function parameters. Prefix your parameter name with `_` to ignore this error.
`strictBindCallApply`	Enforce type safety for `bind`, `call`, and `apply`. See “call, apply, and bind”.
`strictFunctionTypes`	Enforce that functions are contravariant in their

Flag	Description
	parameter and `this` types. See “Function variance”.
`strictNullChecks`	Promote `null` to a type. See “null, undefined, void, and never”.
`strictPropertyInitialization`	Enforce that class properties are either nullable or initialized. See Chapter 5.