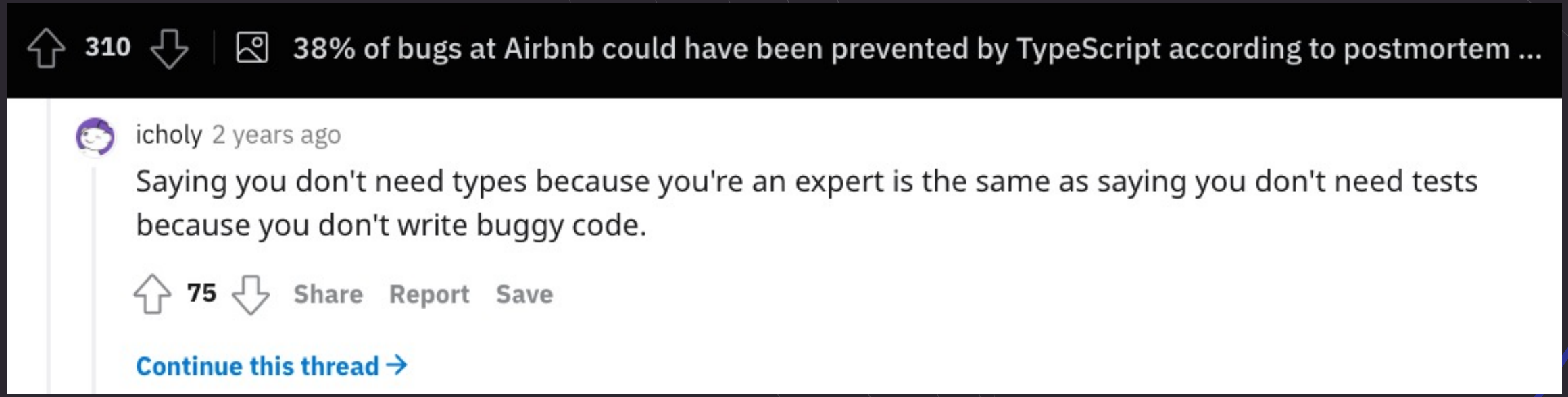
“

tsc --strict@vue

Daniel Danielecki



TypeScript helps preventing bugs

That’s not only what Reddit says…

To Type or Not to Type: Quantifying Detectable Bugs in JavaScript

Zheng Gao
University College London
London, UK
z.gao.12@ucl.ac.uk

Christian Bird
Microsoft Research
Redmond, USA
cbird@microsoft.com

Earl T. Barr
University College London
London, UK
e.barr@ucl.ac.uk

search/completion and serving as documentation. Despite this uneven playing field, our central finding is that both static type systems find an important percentage of public bugs: both Flow 0.30 and TypeScript 2.0 successfully detect 15%!

TypeScript helps preventing bugs

It's a scientific fact

TypeScript Compiler’s Rules

Compiler Options

Top Level	files, extends, include, exclude and references
"compilerOptions"	
Type Checking	allowUnreachableCode, allowUnusedLabels, alwaysStrict, exactOptionalPropertyTypes, noFallthroughCasesInSwitch, noImplicitAny, noImplicitOverride, noImplicitReturns, noImplicitThis, noPropertyAccessFromIndexSignature, noUncheckedIndexedAccess, noUnusedLocals, noUnusedParameters, strict, strictBindCallApply, strictFunctionTypes, strictNullChecks, strictPropertyInitialization and useUnknownInCatchVariables
Modules	allowUmdGlobalAccess, baseUrl, module, moduleResolution, noResolve, paths, resolveJsonModule, rootDir, rootDirs, typeRoots and types
Emit	declaration, declarationDir, declarationMap, downlevelIteration, emitBOM, emitDeclarationOnly, importHelpers, importsNotUsedAsValues, inlineSourceMap, inlineSources, mapRoot, newLine, noEmit, noEmitHelpers, noEmitOnError, outDir, outFile, preserveConstEnums, removeComments, sourceMap, sourceRoot and stripInternal
JavaScript Support	allowJs, checkJs and maxNodeModuleJsDepth
Editor Support	disableSizeLimit and plugins
Interop Constraints	allowSyntheticDefaultImports, esModuleInterop, forceConsistentCasingInFileNames, isolatedModules and preserveSymlinks
Backwards Compatibility	charset, keyofStringsOnly, noImplicitUseStrict, noStrictGenericChecks, out, suppressExcessPropertyErrors and suppressImplicitAnyIndexErrors

Compiler Options

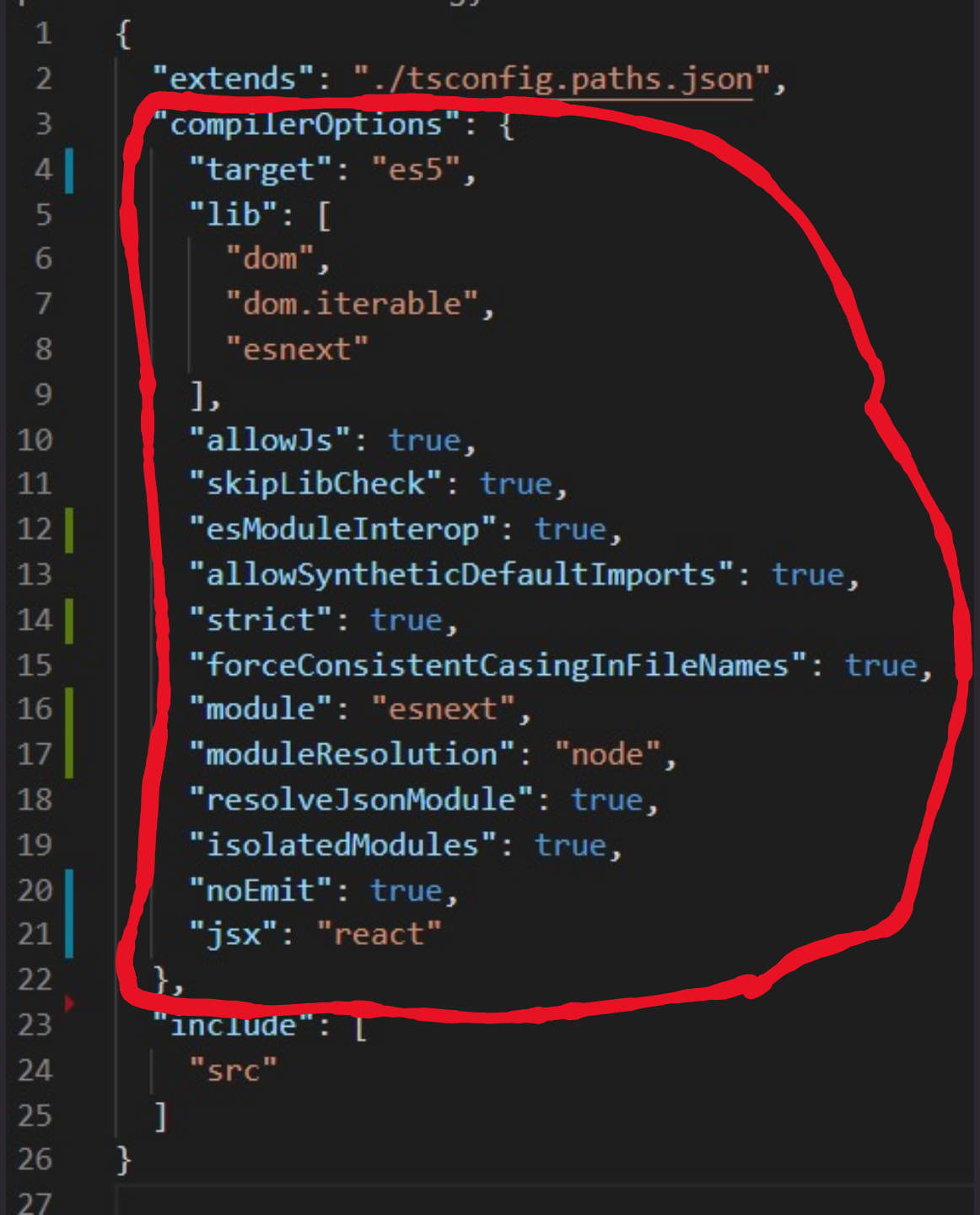
Top Level	files, extends, include, exclude and references
"compilerOptions"	
Type Checking	allowUnreachableCode, allowUnusedLabels, alwaysStrict, exactOptionalPropertyTypes, noFallthroughCasesInSwitch, noImplicitAny, noImplicitOverride, noImplicitReturns, noImplicitThis, noPropertyAccessFromIndexSignature, noUncheckedIndexedAccess, noUnusedLocals, noUnusedParameters, strict, strictBindCallApply, strictFunctionTypes, strictNullChecks, strictPropertyInitialization and useUnknownInCatchVariables
Modules	allowUmdGlobalAccess, baseUrl, module, moduleResolution, noResolve, paths, resolveJsonModule, rootDir, rootDirs, typeRoots and types
Emit	declaration, declarationDir, declarationMap, downlevelIteration, emitBOM, emitDeclarationOnly, importHelpers, importsNotUsedAsValues, inlineSourceMap, inlineSources, mapRoot, newLine, noEmit, noEmitHelpers, noEmitOnError, outDir, outFile, preserveConstEnums, removeComments, sourceMap, sourceRoot and stripInternal
JavaScript Support	allowJs, checkJs and maxNodeModuleJsDepth
Editor Support	disableSizeLimit and plugins
Interop Constraints	allowSyntheticDefaultImports, esModuleInterop, forceConsistentCasingInFileNames, isolatedModules and preserveSymlinks
Backwards Compatibility	charset, keyofStringsOnly, noImplicitUseStrict, noStrictGenericChecks, out, suppressExcessPropertyErrors and suppressImplicitAnyIndexErrors

Compiler Options

Top Level	files, extends, include, exclude and references
"compilerOptions"	
Type Checking	allowUnreachableCode, allowUnusedLabels, alwaysStrict, exactOptionalPropertyTypes, noFallthroughCasesInSwitch, noImplicitAny, noImplicitOverride, noImplicitReturns, noImplicitThis, noPropertyAccessFromIndexSignature, noUncheckedIndexedAccess, noUnusedLocals, noUnusedParameters, strict, strictBindCallApply, strictFunctionTypes, strictNullChecks, strictPropertyInitialization and useUnknownInCatchVariables
Modules	allowUmdGlobalAccess, baseUrl, module, moduleResolution, noResolve, paths, resolveJsonModule, rootDir, rootDirs, typeRoots and types
Emit	declaration, declarationDir, declarationMap, downlevelIteration, emitBOM, emitDeclarationOnly, importHelpers, importsNotUsedAsValues, inlineSourceMap, inlineSources, mapRoot, newLine, noEmit, noEmitHelpers, noEmitOnError, outDir, outFile, preserveConstEnums, removeComments, sourceMap, sourceRoot and stripInternal
JavaScript Support	allowJs, checkJs and maxNodeModuleJsDepth
Editor Support	disableSizeLimit and plugins
Interop Constraints	allowSyntheticDefaultImports, esModuleInterop, forceConsistentCasingInFileNames, isolatedModules and preserveSymlinks
Backwards Compatibility	charset, keyofStringsOnly, noImplicitUseStrict, noStrictGenericChecks, out, suppressExcessPropertyErrors and suppressImplicitAnyIndexErrors

tsconfig

```jsonc
{
  "extends": "./tsconfig.paths.json",
  "compilerOptions": {
    "target": "es5",
    "lib": [
      "dom",
      "dom.iterable",
      "esnext"
    ],
    "allowJs": true,
    "skipLibCheck": true,
    "esModuleInterop": true,
    "allowSyntheticDefaultImports": true,
    "strict": true,
    "forceConsistentCasingInFileNames": true,
    "module": "esnext",
    "moduleResolution": "node",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "noEmit": true,
    "jsx": "react"
  },
  "include": [
    "src"
  ]
}
```



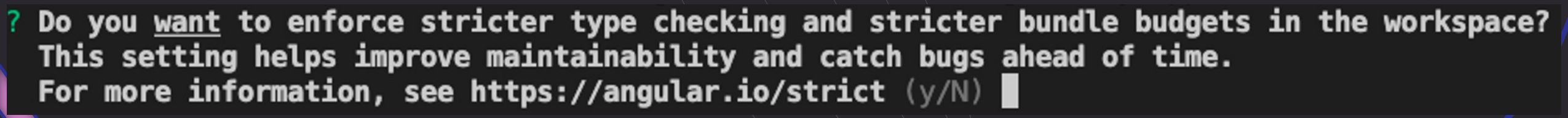
```
{
  "extends": "./tsconfig.paths.json",
  "compilerOptions": {
    "target": "es5",
    "lib": [
      "dom",
      "dom.iterable",
      "esnext"
    ],
    "allowJs": true,
    "skipLibCheck": true,
    "esModuleInterop": true,
    "allowSyntheticDefaultImports": true,
    "strict": true,
    "forceConsistentCasingInFileNames": true,
    "module": "esnext",
    "moduleResolution": "node",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "noEmit": true,
    "jsx": "react"
  },
  "include": [
    "src"
  ]
}
```

Vue.js

```
Vue CLI v4.5.15
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, TS, PWA, Router, Vuex, CSS Pre-processors, Linter, Unit, E2E
? Choose a version of Vue.js that you want to start the project with 3.x
? Use class-style component syntax? No
? Use Babel alongside TypeScript (required for modern mode, auto-detected polyfills, transpiling JSX)? Yes
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): Sass/SCSS (with dart-sass)
? Pick a linter / formatter config: Prettier
? Pick additional lint features: Lint on save
? Pick a unit testing solution: Jest
? Pick an E2E testing solution: Cypress
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated config files
? Save this as a preset for future projects? (y/N)
```

vue create

No question regarding --strict



Angular

Forgot about –strict?

```
"compilerOptions": {
  "baseUrl": "./",
  "outDir": "./dist/out-tsc",
  "sourceMap": true,
  "declaration": false,
  "downlevelIteration": true,
  "experimentalDecorators": true,
  "moduleResolution": "node",
  "importHelpers": true,
  "target": "es2015",
  "module": "es2020",
  "lib": [
    "es2018",
    "dom"
```

```
4  "compilerOptions": {
5    "baseUrl": "./",
6    "outDir": "./dist/out-tsc",
7    "forceConsistentCasingInFileNames": true,
8    "strict": true,
9    "noImplicitReturns": true,
10   "noFallthroughCasesInSwitch": true,
11   "sourceMap": true,
12   "declaration": false,
13   "downlevelIteration": true,
14   "experimentalDecorators": true,
15   "moduleResolution": "node",
16   "importHelpers": true,
17   "target": "es2015",
```

Angular

no strict vs --strict

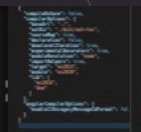
```
"compilerOptions": {
  "baseUrl": "./",
  "outDir": "./dist/out-tsc",
  "sourceMap": true,
  "declaration": false,
  "downlevelIteration": true,
  "experimentalDecorators": true,
  "moduleResolution": "node",
  "importHelpers": true,
  "target": "es2015",
  "module": "es2020",
  "lib": [
    "es2018",
    "dom"
```

```
4  "compilerOptions": {
5    "baseUrl": "./",
6    "outDir": "./dist/out-tsc",
7    "forceConsistentCasingInFileNames": true,
8    "strict": true,
9    "noImplicitReturns": true,
10   "noFallthroughCasesInSwitch": true,
11   "sourceMap": true,
12   "declaration": false,
13   "downlevelIteration": true,
14   "experimentalDecorators": true,
15   "moduleResolution": "node",
16   "importHelpers": true,
17   "target": "es2015",
```

Angular

no strict vs --strict

```
20    "angularCompilerOptions": {
21      "enableI18nLegacyMessageIdFormat"
22    }
23  }
24
```



```
24    "angularCompilerOptions": {
25      "enableI18nLegacyMessageIdFormat": false,
26      "strictInjectionParameters": true,
27      "strictInputAccessModifiers": true,
28      "strictTemplates": true
29    }
30  }
```

Angular

no strict vs --strict

```
added 119 packages from 101 contributors, updated 1 package and audited 1864 packages in 31.107s

108 packages are looking for funding
  run `npm fund` for details

found 38 vulnerabilities (22 moderate, 16 high)
  run `npm audit fix` to fix them, or `npm audit` for details
⚓  Running completion hooks...

📄  Generating README.md...

🎉  Successfully created project hello-world.
👉  Get started with the following commands:

 $ cd hello-world
 $ npm run serve
```

Project generated

Let’s see what’s inside…

```
"strict": true,
"jsx": "preserve",
"importHelpers": true,
"moduleResolution": "node",
"experimentalDecorators": true,
"skipLibCheck": true,
"esModuleInterop": true,
"allowSyntheticDefaultImports": true,
"sourceMap": true,
"baseUrl": ".",
```

Vue.js (2)

Vue.js 3 has the same settings related to strict

```
"strict": true,
"jsx": "preserve",
"importHelpers": true,
"moduleResolution": "node",
"experimentalDecorators": true,
"skipLibCheck": true,
"esModuleInterop": true,
"allowSyntheticDefaultImports": true,
"sourceMap": true,
"baseUrl": ".",
```

Vue.js (2)

Vue.js 3 has the same settings related to strict

Compiler Options

Top Level	files, extends, include, exclude and references
"compilerOptions"	
Type Checking	allowUnreachableCode, allowUnusedLabels, alwaysStrict, exactOptionalPropertyTypes, noFallthroughCasesInSwitch, noImplicitAny, noImplicitOverride, noImplicitReturns, noImplicitThis, noPropertyAccessFromIndexSignature, noUncheckedIndexedAccess, noUnusedLocals, noUnusedParameters, strict, strictBindCallApply, strictFunctionTypes, strictNullChecks, strictPropertyInitialization and useUnknownInCatchVariables
Modules	allowUmdGlobalAccess, baseUrl, module, moduleResolution, noResolve, paths, resolveJsonModule, rootDir, rootDirs, typeRoots and types
Emit	declaration, declarationDir, declarationMap, downlevelIteration, emitBOM, emitDeclarationOnly, importHelpers, importsNotUsedAsValues, inlineSourceMap, inlineSources, mapRoot, newLine, noEmit, noEmitHelpers, noEmitOnError, outDir, outFile, preserveConstEnums, removeComments, sourceMap, sourceRoot and stripInternal
JavaScript Support	allowJs, checkJs and maxNodeModuleJsDepth
Editor Support	disableSizeLimit and plugins
Interop Constraints	allowSyntheticDefaultImports, esModuleInterop, forceConsistentCasingInFileNames, isolatedModules and preserveSymlinks
Backwards Compatibility	charset, keyofStringsOnly, noImplicitUseStrict, noStrictGenericChecks, out, suppressExcessPropertyErrors and suppressImplicitAnyIndexErrors

```
"strict": true,
"jsx": "preserve",
"importHelpers": true,
"moduleResolution": "node",
"experimentalDecorators": true,
"skipLibCheck": true,
"esModuleInterop": true,
"allowSyntheticDefaultImports": true,
"sourceMap": true,
"baseUrl": ".",
```

Vue.js (2)

Vue.js 3 has the same settings related to strict

strict

Strict - `strict`

The `strict` flag enables a wide range of type checking behavior that results in stronger guarantees of program correctness. Turning this on is equivalent to enabling all of the *strict mode family* options, which are outlined below. You can then turn off individual strict mode family checks as needed.

Future versions of TypeScript may introduce additional stricter checking under this flag, so upgrades of TypeScript might result in new type errors in your program. When appropriate and possible, a corresponding flag will be added to disable that behavior.

Recommended:
True

Default:
`false`

Related:
`alwaysStrict`,
`strictNullChecks`,
`strictBindCallApply`,
`strictFunctionTypes`,
`strictPropertyInitiali`
`noImplicitAny`, `noImplic`
`useUnknownInCatchVaria`

Released:
2.3

Strict - `strict`

The `strict` flag enables a wide range of type checking behavior that results in stronger guarantees of program correctness. Turning this on is equivalent to enabling all of the *strict mode family* options, which are outlined below. You can then turn off individual strict mode family checks as needed.

Future versions of TypeScript may introduce additional stricter checking under this flag, so upgrades of TypeScript might result in new type errors in your program. When appropriate and possible, a corresponding flag will be added to disable that behavior.

Recommended:
True

Default:
`false`

Related:
alwaysStrict,
strictNullChecks,
strictBindCallApply,
strictFunctionTypes,
strictPropertyInitiali
noImplicitAny, noImplic
useUnknownInCatchVaria

Released:
2.3

Compiler Options

Top Level	files, extends, include, exclude and references
"compilerOptions"	
Type Checking	allowUnreachableCode, allowUnusedLabels, alwaysStrict, exactOptionalPropertyTypes, noFallthroughCasesInSwitch, noImplicitAny, noImplicitOverride, noImplicitReturns, noImplicitThis, noPropertyAccessFromIndexSignature, noUncheckedIndexedAccess, noUnusedLocals, noUnusedParameters, strict, strictBindCallApply, strictFunctionTypes, strictNullChecks, strictPropertyInitialization and useUnknownInCatchVariables
Modules	allowUmdGlobalAccess, baseUrl, module, moduleResolution, noResolve, paths, resolveJsonModule, rootDir, rootDirs, typeRoots and types
Emit	declaration, declarationDir, declarationMap, downlevelIteration, emitBOM, emitDeclarationOnly, importHelpers, importsNotUsedAsValues, inlineSourceMap, inlineSources, mapRoot, newLine, noEmit, noEmitHelpers, noEmitOnError, outDir, outFile, preserveConstEnums, removeComments, sourceMap, sourceRoot and stripInternal
JavaScript Support	allowJs, checkJs and maxNodeModuleJsDepth
Editor Support	disableSizeLimit and plugins
Interop Constraints	allowSyntheticDefaultImports, esModuleInterop, forceConsistentCasingInFileNames, isolatedModules and preserveSymlinks
Backwards Compatibility	charset, keyofStringsOnly, noImplicitUseStrict, noStrictGenericChecks, out, suppressExcessPropertyErrors and suppressImplicitAnyIndexErrors

7 out of ~35

```
"strict": true,
"jsx": "preserve",
"importHelpers": true,
"moduleResolution": "node",
"experimentalDecorators": true,
"skipLibCheck": true,
"esModuleInterop": true,
"allowSyntheticDefaultImports": true,
"sourceMap": true,
"baseUrl": ".",
```

Vue.js (2)

Rules by default (Vue.js 3 has extra “experimentalDecorators”)

skipLibCheck
&
skipDefaultLibCheck

Completeness

Skip Default Lib Check - `skipDefaultLibCheck`

Use `skipLibCheck` instead. Skip type checking of default library declaration files.

Default:
`false`

Skip Lib Check - `skipLibCheck`

Skip type checking of declaration files.

This can save time during compilation at the expense of type-system accuracy. For example, two libraries could define two copies of the same `type` in an inconsistent way. Rather than doing a full check of all `d.ts` files, TypeScript will type check the code you specifically refer to in your app's source code.

A common case where you might think to use `skipLibCheck` is when there are two copies of a library's types in your `node_modules`. In these cases, you should consider using a feature like yarn's resolutions to ensure there is only one copy of that dependency in your tree or investigate how to ensure there is only one copy by understanding the dependency resolution to fix the issue without additional tooling.

Recommended:
True

Default:
`false`

Released:
2.0

Completeness

Skip Default Lib Check - `skipDefaultLibCheck`

Use `skipLibCheck` instead. Skip type checking of default library declaration files.

Default:
`false`

Skip Lib Check - `skipLibCheck`

Skip type checking of declaration files.

This can save time during compilation at the expense of type-system accuracy. For example, two libraries could define two copies of the same `type` in an inconsistent way. Rather than doing a full check of all `d.ts` files, TypeScript will type check the code you specifically refer to in your app's source code.

A common case where you might think to use `skipLibCheck` is when there are two copies of a library's types in your `node_modules`. In these cases, you should consider using a feature like yarn's resolutions to ensure there is only one copy of that dependency in your tree or investigate how to ensure there is only one copy by understanding the dependency resolution to fix the issue without additional tooling.

Recommended:
True

Default:
`false`

Released:
2.0

== npm overrides

allowSyntheticDefaultImports

https://www.typescriptlang.org/tsconfig#allowSyntheticDefaultImports

10 out of ~35

What about the remaining ~25?

Default tsconfig recommended options set in Vue.js	Vue.js (2/3)
Recommended false set to false	8/36
Recommended true set to true	9/36
Recommended false set to true	(1)2/36
Recommended true set to false	12/36
Other recommended settings working	5/36
Other recommended settings not working	0/36
SUM OF RECOMMENDED SETTINGS ON	**22/36**
SUM OF RECOMMENDED SETTINGS OFF	**14/36**

3 demos

Official docs

https://www.typescriptlang.org/tsconfig

Official docs

https://www.typescriptlang.org/docs/handbook/compiler-options.html

Thanks