STOP READING IMMEDIATELY

THIS PAGE PROBABLY DOES NOT PERTAIN TO YOU.

These are Coding Guidelines for ***Contributors to TypeScript***. This is ***NOT*** a prescriptive guideline for the TypeScript community. These guidelines are meant for **contributors to the TypeScript project's codebase**. We have chosen many of them for team consistency. Feel free to adopt them for your own team.

AGAIN: This is ***NOT*** a prescriptive guideline for the TypeScript community

---

Please do not file issues about these guidelines.

Names

1. Use PascalCase for type names.
2. Do not use `I` as a prefix for interface names.
3. Use PascalCase for enum values.
4. Use camelCase for function names.
5. Use camelCase for property names and local variables.
6. Do not use `_` as a prefix for private properties.
7. Use whole words in names when possible.

Components

1. 1 file per logical component (e.g. parser, scanner, emitter, checker).
2. Do not add new files. :)
3. files with `.generated.*` suffix are auto-generated, do not hand-edit them.

Types

1. Do not export types/functions unless you need to share it across multiple components.
2. Do not introduce new types/values to the global namespace.
3. Shared types should be defined in `types.ts`.
4. Within a file, type definitions should come first.

`null` and `undefined`

1. Use `undefined`. Do not use null.

General Assumptions

1. Consider objects like Nodes, Symbols, etc. as immutable outside the component that created them. Do not change them.
2. Consider arrays as immutable by default after creation.

Classes

1. For consistency, do not use classes in the core compiler pipeline. Use function closures instead.

Flags

1. More than 2 related Boolean properties on a type should be turned into a flag.

Comments

1. Use JSDoc style comments for functions, interfaces, enums, and classes.

Strings

1. Use double quotes for strings.
2. All strings visible to the user need to be localized (make an entry in diagnosticMessages.json).

Diagnostic Messages

1. Use a period at the end of a sentence.
2. Use indefinite articles for indefinite entities.
3. Definite entities should be named (this is for a variable name, type name, etc..).
4. When stating a rule, the subject should be in the singular (e.g. "An external module cannot..." instead of "External modules cannot...").
5. Use present tense.

Diagnostic Message Codes

Diagnostics are categorized into general ranges. If adding a new diagnostic message, use the first integral number greater than the last used number in the appropriate range. * 1000 range for syntactic messages * 2000 for semantic messages * 4000 for declaration emit messages * 5000 for compiler options messages * 6000 for command line compiler messages * 7000 for noImplicitAny messages

General Constructs

For a variety of reasons, we avoid certain constructs, and use some of our own. Among them:

1. Do not use `for..in` statements; instead, use `ts.forEach`, `ts.forEachKey` and `ts.forEachValue`. Be aware of their slightly different semantics.
2. Try to use `ts.forEach`, `ts.map`, and `ts.filter` instead of loops when it is not strongly inconvenient.

Style

1. Use arrow functions over anonymous function expressions.
2. Only surround arrow function parameters when necessary. For example, `(x) => x + x` is wrong but the following are correct:
   - `x => x + x`
   - `(x,y) => x + y`
   - `<T>(x: T, y: T) => x === y`
3. Always surround loop and conditional bodies with curly braces. Statements on the same line are allowed to omit braces.
4. Open curly braces always go on the same line as whatever necessitates them.
5. Parenthesized constructs should have no surrounding whitespace. A single space follows commas, colons, and semicolons in those constructs. For example:
   - `for (var i = 0, n = str.length; i < 10; i++) { }`
   - `if (x < 10) { }`
   - `function f(x: number, y: string): void { }`
6. Use a single declaration per variable statement (i.e. use `var x = 1; var y = 2;` over `var x = 1, y = 2;`).
7. `else` goes on a separate line from the closing curly brace.
8. Use 4 spaces per indentation.