変わりゆく TypeScript
6.0 と 7.0 のバージョン移行の全体像

somnicat

2026-04-13

自己紹介

名前: **somnicat** (@somnicattus)

職業: Web アプリケーションエンジニア

使う技術: フルスタック TypeScript (Vue/Node.js)

最近 VRChat で技術系イベントの共同主催を始めた

早速ですが、

今、TypeScript が大きく変わろうとしている

ことを知っていますか？

現在までのタイムライン

2025/3 に Go 移植が発表され、近日中に 7.0 としてリリース予定

時期	出来事
2025/3	A 10x Faster TypeScript - Go 移植とロードマップを発表
2025/5	`@typescript/native-preview` として npm 公開開始
2025/12	Progress on TypeScript 7 - 開発ツール向け API を含む変更内容を予告
2026/2	TypeScript 6.0 Beta - 表面上の変更がほぼ確定
2026/3	**TypeScript 6.0** - 7.0 への橋渡しとなる最後の JS ベース
数か月後	**TypeScript 7.0** リリース予定 - Go 実装・LSP 移行・新しい API

今日話すこと

大体は公式のブログやドキュメントに書かれていることの焼き直しです
日本語で簡潔にまとめて、移植の背景についての考察も含めてお伝えします

- **A 10x Faster TypeScript** - Go で実装される TypeScript
- **TypeScript 6.0** - 最後の JavaScript ベースのリリース
- **TypeScript 7.0** - Go 実装への転換
- **今できること** - TypeScript 7.0 に向けて始められること

A 10x Faster TypeScript

Go で実装される TypeScript

A 10x Faster TypeScript

> *The native implementation will **drastically improve editor startup, reduce most build times by 10x, and substantially reduce memory usage.***
>
> （訳）ネイティブ実装により、**エディターの起動が劇的に改善され、ほとんどのビルド時間が 10 倍短縮され、メモリ使用量が大幅に削減されます。**
>
> — *A 10x Faster TypeScript (2025/3/11)*

なぜ Go で再実装するのか（推測）

- **もとの TS 実装ではパフォーマンスに限界があった**
  - tsc や TSServer は TS で実装されている → つまり **JS で動作する**
  - 大規模プロジェクトでビルド速度やエディタ起動速度が問題に
- **代替の並行メンテナンスは非現実的だった**
  - 型チェッカー実装 `checker.ts` は **3 MB を超える単一ファイル**
  - このファイルの変更に代替実装が追従するのはかなり非現実的
- **パフォーマンスと移植性で Go が選ばれた**
  - 並列処理が得意で高速に動作し TS と同じく GC があるため移植が容易

TypeScript 6.0

最後の JavaScript ベースのリリース

6.0 の位置づけ

> *TypeScript 6.0 is a unique release in that we intend for it to be the last release based on the current JavaScript codebase.*
>
> *（訳）TypeScript 6.0 は、現在の JavaScript コードベースを基盤とした最後のリリースとなることを意図した、特別なリリースです。*
>
> — *Announcing TypeScript 6.0 (2026/3/23)*

6.0 は 7.0 への移行準備

Progress on TypeScript 7 (2025/12/2) より:

- 6.0 は 5.9 から 7.0 への橋渡し
  - 7.0 で実装自体が削除されるオプションが 6.0 で非推奨に
  - モダンな環境に合わせてオプションのデフォルト値を変更
- 6.0 は最後の JS ベースの TypeScript
  - **6.1 はない**
  - パッチリリース (6.0.x) のみ

6.0 の変更点

Announcing TypeScript 6.0 (2026/3) より:

- 多くの設定を非推奨化
  - `target: es5`, `moduleResolution: node`, `amd` / `umd`, `outFile` 等
  - `"ignoreDeprecations": "6.0"` で一時的に抑制可能。7.0 で完全削除
- デフォルト値を大胆に変更
  - `strict: true`, `target: es2025`, `module: esnext`, `types: []` 等
- 新しい ECMAScript をサポート
  - `Temporal` API, `Map.getOrInsert`, `RegExp.escape` 等

TypeScript 7.0

Go 実装への転換

7.0 で変わること・変わらないこと

Progress on TypeScript 7 (2025/12/2) より:

- **変わらない**: TypeScript の言語仕様・型システム・チェック結果
  - 約 20,000 テストケースでほぼ完全互換
- **変わる**: コンパイラーの実装言語
  - `tsc` (JS) → `tsgo` (Go ネイティブ) へ
  - 独自の TSServer プロトコル → 標準的な LSP へ移行
  - 互換性を捨てて新しい TypeScript API に移行
    - Strada API → Corsa API

7.0 への移行で特に注意したいこと

- **非互換な新しい TypeScript API** (API を使う開発ツールの開発者)
  - リンター等のツール連携に影響がある
- **古い出力方法の互換切り** (`tsc` でトランスパイルしていた場合)
  - ダウンレベル出力は `es2015` まで、 `amd` / `umd` の廃止など
- **JSDoc 型チェック** (`.js` ファイルで JSDoc を使っていた場合)
  - `@enum` , `@constructor` タグなど一部が非対応に
- **並列型チェックによるユニオン型の順序問題** (高度な型操作)
  - 詳細は Appendix に記載

今できること

TypeScript 7.0 に向けて始められること

今できること: 6.0 への更新

6.0 を使って 7.0 に向けて段階的に移行していく

- **tsconfig の調整**
  - `"types": ["node"]` など明示指定
  - `"rootDir": "./src"` の明示指定
  - 非推奨オプションの対応 (`moduleResolution: node` → `nodenext` 等)
    - **段階的移行**: `"ignoreDeprecations": "6.0"` で非推奨エラーを一時抑制
- **自動修正**: `ts5to6` ツールで `baseUrl` や `rootDir` を一括対応可能

今できること: 7.0 を試す

7.0 対応が必要なライブラリ開発者は Native Preview で早めに移行を開始
現状では LSP の機能が少し足りないが、ほとんど問題なく使える

- **npm パッケージ (コンパイラー)** `@typescript/native-preview`
- **VS Code 拡張 (LSP)** `TypeScriptTeam.native-preview`
- **Oxlint (リンター)**: `oxlint-tsgolint`
  - 内部で `typescript-go` を直接組み込み
- **テンプレート**: Vite+ の一部のテンプレートは 7.0 Preview を使用
  - 新規プロジェクトなら 7.0 前提のツールチェーンをすぐ体験できる

参考リンク１(公式ブログ)

- A 10x Faster TypeScript: https://devblogs.microsoft.com/typescript/typescript-native-port/
- Progress on TypeScript 7: https://devblogs.microsoft.com/typescript/progress-on-typescript-7-december-2025/
- Announcing TypeScript 6.0 Beta: https://devblogs.microsoft.com/typescript/announcing-typescript-6-0-beta/
- Announcing TypeScript 6.0: https://devblogs.microsoft.com/typescript/announcing-typescript-6-0/

参考リンク 2 (関連ツール)

- ts5to6: https://github.com/andrewbranch/ts5to6
- TypeScript (Native Preview) (npm): https://www.npmjs.com/package/@typescript/native-preview
- TypeScript (Native Preview) (Visual Studio Marketplace): https://marketplace.visualstudio.com/items?itemName=TypeScriptTeam.native-preview
- tsgolint: https://github.com/oxc-project/tsgolint
- Vite+: https://viteplus.dev/guide/create

Thank You

Appendix

- 6.0 の変更点の一覧
- `--stableTypeOrdering` について

新しいデフォルト値

6.0 で多くのコンパイラーオプションのデフォルトが変更された

オプション	旧デフォルト (〜5.9)	新デフォルト (6.0〜)
`strict`	`false`	**`true`**
`target`	`es3`	**`es2025`**（最新安定 ES）
`module`	`commonjs`	**`esnext`**
`rootDir`	ソースファイルの共通ディレクトリを推論	`.`（tsconfig.json のディレクトリ）
`types`	`node_modules/@types` を全て列挙	`[]`（空配列 ー 明示指定が必要）

大量の非推奨化 (7.0 で削除)

非推奨項目	推奨される移行先
`target: es5`	`es2015` 以上
`moduleResolution: node`	`nodenext` or `bundler`
`module: amd / umd / systemjs`	ESM ベースの設定
`baseUrl`	`paths` にプレフィックスを直接記述
`outFile`	外部バンドラー (Vite, esbuild 等)
`esModuleInterop: false`	常に `true` に (設定不可)
import `asserts` 構文	import `with` 構文

`"ignoreDeprecations": "6.0"` で一時的に抑制可能。

新しい ECMAScript サポート

機能	lib	概要
`Temporal` API	`esnext.temporal`	Stage 4 到達。`Date` に代わる日時処理の標準
`Map.getOrInsert` / `getOrInsertComputed`	`esnext`	"upsert" パターンをワンライナーに
`RegExp.escape`	`es2025`	正規表現の特殊文字を安全にエスケープ
`dom.iterable` → `dom` に統合	`dom`	`"lib": ["dom"]` だけで `NodeList` の `for...of` が可能に

並列型チェックによる型定義出力順の変化

7.0 は **並列で型チェックを行う** ため、型の順序付けアルゴリズムが変化

- 従来は「コンパイラーの訪問順」で `.d.ts` 型定義ファイルに出力
  - `100 | 500` が関係ないソースの変更で `500 | 100` になることがある
  - 同じソースコードに対しては順序が安定
- 7.0 以降は並列でコンパイルするため、訪問順が非決定的になる
  - そのままでは `.d.ts` 型定義ファイルの出力がコンパイルのたびに変わってしまう
- 7.0 以降では順序を保証するために型をソートする
  - ソースコードを変更しても常に `100 | 500` で安定
- 6.0 の `--stableTypeOrdering` フラグで事前に検証可能
  - 「暗黙的な順序によってたまたま動いていた」型推論を事前に発見
  - ただしソートのために追加の計算が発生し最大 25% 速度が低下