Migrating TypeScript to Modules

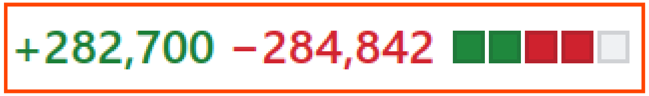
The Fine Details

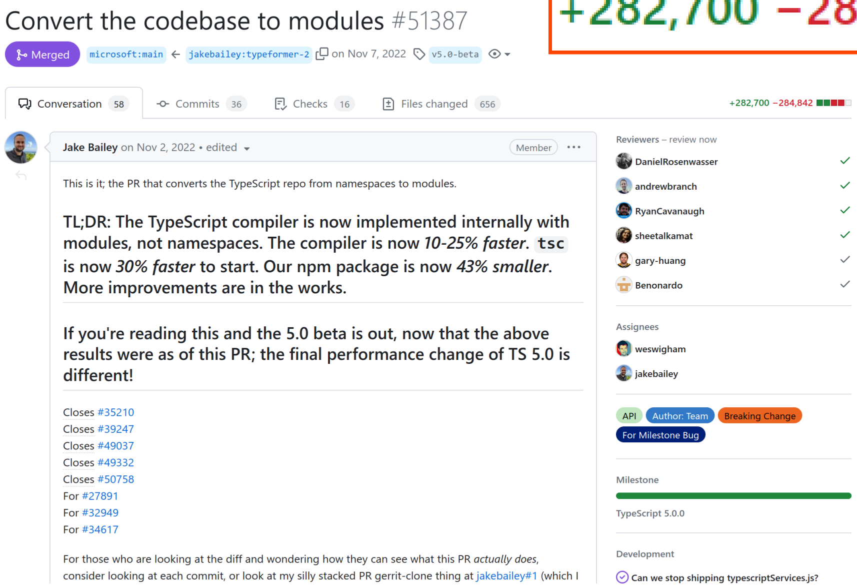
Jake Bailey

Senior Software Engineer, TypeScript @ Microsoft

jakebailey.dev/talk-ts-congress-2023

What are we talking about?


+282,700 −284,842 🟩🟩🟥🟥⬜


Convert the codebase to modules #51387

Merged microsoft:main ← jakebailey:typeformer-2 on Nov 7, 2022 v5.0-beta

Jake Bailey on Nov 2, 2022 • edited Member

This is it; the PR that converts the TypeScript repo from namespaces to modules.

TL;DR: The TypeScript compiler is now implemented internally with modules, not namespaces. The compiler is now *10-25% faster*. `tsc` is now *30% faster* to start. Our npm package is now *43% smaller*. More improvements are in the works.

If you're reading this and the 5.0 beta is out, now that the above results were as of this PR; the final performance change of TS 5.0 is different!

Closes #35210
Closes #39247
Closes #49037
Closes #49332
Closes #50758
For #27891
For #32949
For #34617

For those who are looking at the diff and wondering how they can see what this PR *actually does*, consider looking at each commit, or look at my silly stacked PR gerrit-clone thing at jakebailey#1 (which I

Reviewers – review now
DanielRosenwasser ✓
andrewbranch ✓
RyanCavanaugh ✓
sheetalkamat ✓
gary-huang ✓
Benonardo ✓

Assignees
weswigham
jakebailey

Milestone
TypeScript 5.0.0

Development
Can we stop shipping typescriptServices.js?

More details at jakebailey.dev/go/module-migration-blog

An outline

- What even is a "migration to modules"?
- Why was it so challenging?
- How did I make it less painful?
- How did the migration *actually* work under the hood?
- How did it go and what’s next?

What even *are* modules?

A few different definitions... two most critical are:

- Modules are a *syntax* (``import``, ``export``)
- Modules are an *output format* (ESM, CommonJS, SystemJS, AMD, UMD, IIFE, ...)

```ts
// @filename: src/someFile.ts
export function sayHello(name: string) { // Export from one file...
    console.log(`Hello, ${name}!`);
}

// @filename: src/index.ts
import { sayHello } from "./someFile"; // ... import it in another.

sayHello("TypeScript Congress");
```

TypeScript pre-modules

The opposite of modules is... scripts 😱 Everything is placed within *global* namespaces.

```ts
// @filename: src/compiler/parser.ts
namespace ts {
    export function createSourceFile(sourceText: string): SourceFile {/* ... */}
}

// @filename: src/compiler/program.ts
namespace ts {
    export function createProgram(): Program {
        const sourceFile = createSourceFile(text);
}
```

Fun fact: namespaces were originally called "internal modules".

Emitting namespaces

Namespaces turn into plain objects and functions.

```
// was: src/compiler/parser.ts
var ts;
(function(ts) {
    function createSourceFile(sourceText) {/* ... */}
    ts.createSourceFile = createSourceFile;
})(ts || (ts = {}));

// was: src/compiler/program.ts
var ts;
(function(ts) {
    function createProgram() {
        const sourceFile = ts.createSourceFile(text);
    }
    ts.createProgram = createProgram;
})(ts || (ts = {}));
```

"Bundling" with `prepend`

```
// @filename: src/tsc/tsconfig.json
{
    "compilerOptions": { "outFile": "../../built/local/tsc.js" },
    "references": [
        { "path": "../compiler", "prepend": true },
        { "path": "../executeCommandLine", "prepend": true }
    ]
}
```

Makes `tsc` emit:

```
var ts;
// Cram all of src/compiler/**/*.ts and src/executeCommandLine/**/*.ts on top.
(function(ts) {/*...*/})(ts || (ts = {}));
// ...
```

What if someone wants to import us?

Our outputs are constructed global scripts, but we can be clever.

```
namespace ts {
    if (typeof module !== "undefined" && module.exports) module.exports = ts;
}
```

Emits like:

```
var ts;
(function(ts) {/* ... */})(ts || (ts = {}));
// ...
(function(ts) {
    if (typeof module !== "undefined" && module.exports) module.exports = ts;
})(ts || (ts = {}));
```

Namespaces have some upsides

With namespaces, we don’t have to write imports, ever!

- When adding code, no new imports
- When moving code, no changed imports
- `` `tsc` `` "bundles" our code using `` `prepend` ``

But…

Nobody writes code like this anymore!

- We don't get to dogfood modules
- We can't use external tools
- We have to maintain `prepend`... but nobody uses it *except us* 🥴

What we want:

```
// @filename: src/compiler/parser.ts
export function createSourceFile(sourceText: string): SourceFile {/* ... */}

// @filename: src/compiler/program.ts
import { createSourceFile } from "./parser";

export function createProgram(): Program {
```

We know what we want; let’s do it

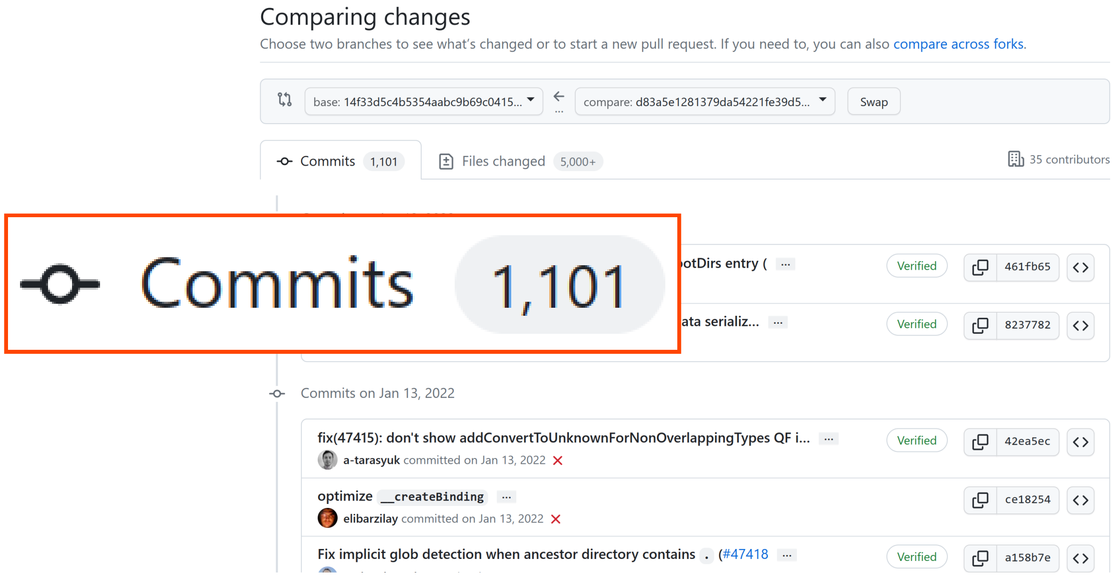
The question is… how can we:

- Actually make the switch …
- … while maintaining the same behavior …
- … and preserving a compatible API?

The challenge

TypeScript is huge!

TypeScript changes often!



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base: 14f33d5c4b5354aabc9b69c0415... ← compare: d83a5e1281379da54221fe39d5... Swap

Commits 1,101 Files changed 5,000+ 35 contributors

Commits 1,101

otDirs entry (… Verified 461fb65

ata serializ… … Verified 8237782

Commits on Jan 13, 2022

fix(47415): don't show addConvertToUnknownForNonOverlappingTypes QF i… … Verified 42ea5ec
a-tarasyuk committed on Jan 13, 2022 ✗

optimize `__createBinding` … ce18254
elibarzilay committed on Jan 13, 2022 ✗

Fix implicit glob detection when ancestor directory contains `.` (#47418 … Verified a158b7e

How can we change a huge, moving project?

Certainly not by hand! Automate *everything*.

- Code transformation where possible
- ``git`` patches to store manual changes
- Done stepwise, for debugging, review, ``git blame`` preservation

What does the migration tool look like?

- Code transformation is performed with `ts-morph`
  - An extremely helpful TypeScript API wrapper by David Sherret ❤ (ts-morph.com)
- Manual changes are managed by `git` with `.patch` files!
  - `git format-patch` dumps commits to disk
  - `git am` applies the patches during the migration
  - If a patch fails to apply, `git` pauses for us!

Code transformation