TypeScript

TypeScript is an open-source programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript and adds optional static typing to the language. TypeScript is designed for development of large applications and transcompiles to JavaScript.[4] As TypeScript is a superset of JavaScript, existing JavaScript programs are also valid TypeScript programs.

TypeScript may be used to develop JavaScript applications for both client-side and server-side execution (as with Node.js or Deno). There are multiple options available for transcompilation. Either the default TypeScript Checker can be used,[5] or the Babel compiler can be invoked to convert TypeScript to JavaScript.

TypeScript supports definition files that can contain type information of existing JavaScript libraries, much like C++ header files can describe the structure of existing object files. This enables other programs to use the values defined in the files as if they were statically typed TypeScript entities. There are third-party header files for popular libraries such as jQuery, MongoDB, and D3.js. TypeScript headers for the Node.js basic modules are also available, allowing development of Node.js programs within TypeScript.[6]

The TypeScript compiler is itself written in TypeScript and compiled to JavaScript. It is licensed under the Apache License 2.0. TypeScript is included as a first-class programming language in Microsoft Visual Studio 2013 Update 2 and later, beside C# and other Microsoft languages.[7] An official extension allows Visual Studio 2012 to support TypeScript as well.[8] Anders Hejlsberg, lead architect of C# and creator of Delphi and Turbo Pascal, has worked on the development of TypeScript.[9][10][11][12]

TypeScript

Paradigm	Multi-paradigm: functional, generic, imperative, object-oriented
Designed by	Microsoft
Developer	Microsoft
First appeared	1 October 2012[1]
Stable release	4.0.3 / 18 September 2020[2]
Preview release	4.0 Beta / 7 July 2020[2]
Typing discipline	Duck, gradual, structural[3]
License	Apache License 2.0
Filename extensions	.ts, .tsx
Website	www.typescriptlang.org (https://www.typescriptlang.org)
Influenced by	
C#, Java, JavaScript	
Influenced	
AtScript, AssemblyScript	

Contents

History

TypeScript was first made public in October 2012 (at version 0.8), after two years of internal development at Microsoft.[13][14] Soon after the announcement, Miguel de Icaza praised the language itself, but criticized the lack of mature IDE support apart from Microsoft Visual Studio, which was not available on Linux and OS X at that time.[15][16] Today there is support in other IDEs, particularly in Eclipse, via a plug-in contributed by Palantir Technologies.[17][18] Various text editors, including Emacs, Vim, Webstorm, Atom[19] and Microsoft's own Visual Studio Code also support TypeScript.[20]

TypeScript 0.9, released in 2013, added support for generics.[21] TypeScript 1.0 was released at Microsoft's Build developer conference in 2014.[22] Visual Studio 2013 Update 2 provides built-in support for TypeScript.[23]

In July 2014, the development team announced a new TypeScript compiler, claiming 5× performance gains. Simultaneously, the source code, which was initially hosted on CodePlex, was moved to GitHub.[24]

On 22 September 2016, TypeScript 2.0 was released; it introduced several features, including the ability for programmers to optionally prevent variables from being assigned `null` values,[25] sometimes referred to as the billion-dollar mistake.

TypeScript 3.0 was released on 30 July 2018,[26] bringing many language additions like tuples in rest parameters and spread expressions, rest parameters with tuple types, generic rest parameters and so on.[27]

Design

TypeScript originated from the shortcomings of JavaScript for the development of large-scale applications both at Microsoft and among their external customers.[28] Challenges with dealing with complex JavaScript code led to demand for custom tooling to ease developing of components in the language.[29]

TypeScript developers sought a solution that would not break compatibility with the standard and its cross-platform support. Knowing that the current ECMAScript standard proposal promised future support for class-based programming, TypeScript was based on that proposal. That led to a JavaScript compiler with a set of syntactical language extensions, a superset based on the proposal, that transforms the extensions into regular JavaScript. In this sense TypeScript was a preview of what to expect of ECMAScript 2015. A unique aspect not in the proposal, but added to TypeScript, is optional static typing[30] that enables static language analysis, which facilitates tooling and IDE support.

ECMAScript 2015 support

TypeScript adds support for features such as classes, modules, and an arrow function syntax as defined in the ECMAScript 2015 standard.

Features

TypeScript is a language extension that adds features to ECMAScript 6. Additional features include:

- Type annotations and compile-time type checking
- Type inference
- Type erasure
- Interfaces
- Enumerated types
- Generics
- Namespaces
- Tuples
- Async/await

The following features are backported from ECMAScript 2015:

- Classes
- Modules[31]
- Abbreviated "arrow" syntax for anonymous functions
- Optional parameters and default parameters

Syntactically, TypeScript is very similar to JScript .NET, another Microsoft implementation of the ECMA-262 language standard that added support for static typing and classical object-oriented language features such as classes, inheritance, interfaces, and namespaces.

Compatibility with JavaScript

TypeScript is a strict superset of ECMAScript 2015, which is itself a superset of ECMAScript 5, commonly referred to as JavaScript.[32] As such, a JavaScript program is also a valid TypeScript program, and a TypeScript program can seamlessly consume JavaScript. By default the compiler targets ECMAScript 5, the current prevailing standard, but is also able to generate constructs used in ECMAScript 3 or 2015.

With TypeScript, it is possible to use existing JavaScript code, incorporate popular JavaScript libraries, and call TypeScript-generated code from other JavaScript.[33] Type declarations for these libraries are provided with the source code.

Type annotations

TypeScript provides static typing through type annotations to enable type checking at compile time. This is optional and can be ignored to use the regular dynamic typing of JavaScript.

```
function add(left: number, right: number): number {
    return left + right;
}
```

The annotations for the primitive types are `number`, `boolean` and `string`. Weakly- or dynamically-typed structures are of type `any`.

Type annotations can be exported to a separate *declarations file* to make type information available for TypeScript scripts using types already compiled into JavaScript. Annotations can be declared for an existing JavaScript library, as has been done for Node.js and jQuery.

The TypeScript compiler makes use of type inference to infer types when types are not given. For example, the `add` method in the code above would be inferred as returning a `number` even if no return type annotation had been provided. This is based on the static types of `left` and `right` being `numbers`, and the compiler's knowledge that the result of adding two `numbers` is always a `number`. However, explicitly declaring the return type allows the compiler to verify correctness.

If no type can be inferred because of lack of declarations, then it defaults to the dynamic `any` type. A value of the `any` type supports the same operations as a value in JavaScript and minimal static type checking is performed for operations on `any` values.[34]

Declaration files

When a TypeScript script gets compiled there is an option to generate a *declaration file* (with the extension `.d.ts`) that functions as an interface to the components in the compiled JavaScript. In the process the compiler strips away all function and method bodies and preserves only the signatures of the types that are exported. The resulting declaration file can then be used to describe the exported virtual TypeScript types of a JavaScript library or module when a third-party developer consumes it from TypeScript.

The concept of declaration files is analogous to the concept of header file found in C/C++.

```
declare namespace arithmetics {
    add(left: number, right: number): number;
    subtract(left: number, right: number): number;
    multiply(left: number, right: number): number;
    divide(left: number, right: number): number;
}
```

Type declaration files can be written by hand for existing JavaScript libraries, as has been done for jQuery and Node.js.

Large collections of declaration files for popular JavaScript libraries are hosted on GitHub in DefinitelyTyped (https://github.com/DefinitelyTyped/DefinitelyTyped).

Classes

TypeScript supports ECMAScript 2015 classes that integrate the optional type annotations support.

```
class Person {
    private name: string;
    private age: number;
    private salary: number;

    constructor(name: string, age: number, salary: number) {
        this.name = name;
        this.age = age;
        this.salary = salary;
    }
```

```
    toString(): string {
        return `${this.name} (${this.age}) (${this.salary})`; // As of version 1.4
    }
}
```

Generics

TypeScript supports generic programming.[35] The following is an example of the identity function.[36]

```
function doSomething<T>(arg: T): T {
    return arg;
}
```

Modules and namespaces

TypeScript distinguishes between modules and namespaces. Both features in TypeScript support encapsulation of classes, interfaces, functions and variables into containers. Namespaces (formerly internal modules) utilize immediately-invoked function expression of JavaScript to encapsulate code, whereas modules (formerly external modules) leverage JavaScript library patterns to do so (AMD or CommonJS).[37]

Development tools

Compiler

The TypeScript compiler, named `tsc`, is written in TypeScript. As a result, it can be compiled into regular JavaScript and can then be executed in any JavaScript engine (e.g. a browser). The compiler package comes bundled with a script host that can execute the compiler. It is also available as a Node.js package that uses Node.js as a host.

There is also an alpha version of a client-side compiler in JavaScript, which executes TypeScript code on the fly, upon page load.[38]

The current version of the compiler supports ECMAScript 5 by default. An option is allowed to target ECMAScript 2015 to make use of language features exclusive to that version (e.g. generators). Classes, despite being part of the ECMAScript 2015 standard, are available in both modes.

IDE and editor support

- Microsoft provides a plug-in for Visual Studio 2012 and WebMatrix, full integrated support in Visual Studio 2013, Visual Studio 2015, and basic text editor support for Emacs and Vim.[39]
- Visual Studio Code is a (mostly) open-source, cross-platform source code editor developed by Microsoft based on Electron. It supports TypeScript in addition to several other languages, and offers features like debugging and intelligent code completion.
- alm.tools is an open source cloud IDE for TypeScript built using TypeScript, ReactJS and TypeStyle.
- JetBrains supports TypeScript with code completion, refactoring and debugging in its IDEs built on IntelliJ platform, such as PhpStorm 6, WebStorm 6, and IntelliJ IDEA,[40] as well as their Visual Studio Add-in and extension, ReSharper 8.1.[41][42]

- Atom has a TypeScript Plugin by Basarat with support for code completion, navigation, formatting, and fast compilation.[43]
- The online Cloud9 IDE and Codenvy support TypeScript.
- A plugin is available for the NetBeans IDE.
- A plugin is available for the Eclipse IDE (version Kepler)
- TypEcs is available for the Eclipse IDE.
- The Cross Platform Cloud IDE Codeanywhere supports TypeScript.
- Webclipse An Eclipse plugin designed to develop TypeScript and Angular 2.
- Angular IDE A standalone IDE available via npm to develop TypeScript and Angular 2 applications, with integrated terminal support.
- Tide – TypeScript Interactive Development Environment for Emacs.

Integration with build automation tools

Using plug-ins, TypeScript can be integrated with build automation tools, including Grunt (grunt-ts[44]), Apache Maven (TypeScript Maven Plugin[45]), Gulp (gulp-typescript[46]) and Gradle (TypeScript Gradle Plugin[47]).

Linting tools

TSLint[48] scans TypeScript code for conformance to a set of standards and guidelines. ESLint, a standard JavaScript linter, also provided some support for TypeScript via community plugins. However, ESLint's inability to leverage TypeScript's language services precluded certain forms of semantic linting and program-wide analysis.[49] In early 2019, the TSLint team announced the linter's deprecation in favor of `typescript-eslint`, a joint effort of the TSLint, ESLint and TypeScript teams to consolidate linting under the ESLint umbrella for improved performance, community unity and developer accessibility.[50] For using TypeScript with ESLint, you need to add the `@typescript-eslint/eslint-plugin`and `@typescript-eslint/parser`.

Release history

Version number	Release date	Significant changes
0.8	1 October 2012	
0.9	18 June 2013	
1.0	12 April 2014	
1.1	6 October 2014	performance improvements
1.3	12 November 2014	`protected` modifier, tuple types
1.4	20 January 2015	union types, `let` and `const` declarations, template strings, type guards, type aliases
1.5	20 July 2015	ES6 modules, `namespace` keyword, `for..of` support, decorators
1.6	16 September 2015	JSX support, intersection types, local type declarations, abstract classes and methods, user-defined type guard functions
1.7	30 November 2015	`async` and `await` support,
1.8	22 February 2016	constraints generics, control flow analysis errors, string literal types, `allowJs`
2.0	22 September 2016	null- and undefined-aware types, control flow based type analysis, discriminated union types, `never` type, `readonly` keyword, type of `this` for functions
2.1	8 November 2016	`keyof` and lookup types, mapped types, object spread and rest,
2.2	22 February 2017	mix-in classes, `object` type,
2.3	27 April 2017	`async` iteration, generic parameter defaults, strict option
2.4	27 June 2017	dynamic import expressions, string enums, improved inference for generics, strict contravariance for callback parameters
2.5	31 August 2017	optional catch clause variables
2.6	31 October 2017	strict function types
2.7	31 January 2018	constant-named properties, fixed length tuples
2.8	27 March 2018	conditional types, improved `keyof` with intersection types
2.9	14 May 2018	support for symbols and numeric literals in keyof and mapped object types
3.0	30 July 2018	project references, extracting and spreading parameter lists with tuples
3.1	27 September 2018	mappable tuple and array types
3.2	30 November 2018	stricter checking for bind, call, and apply
3.3	31 January 2019	relaxed rules on methods of union types, incremental builds for composite projects
3.4	29 March 2019	faster incremental builds, type inference from generic functions, readonly modifier for arrays, const assertions, type-checking globalThis
3.5	29 May 2019	faster incremental builds, omit helper type, improved excess property checks in union types, smarter union type checking
3.6	28 August 2019	Stricter generators, more accurate array spread, better unicode support for identifiers
3.7	5 November 2019	Optional Chaining, Nullish Coalescing
3.8	20 February 2020	Type-only imports and exports, ECMAScript private fields, top-level await
3.9	12 May 2020	
4.0	20 August 2020	

See also

- Dart
- Kotlin
- JS++

References

Citations

1. "TypeScript" (https://typescript.codeplex.com/releases/view/95554). *CodePlex*. Retrieved 26 April 2015.
2. "Announcing TypeScript 4.0" (https://devblogs.microsoft.com/typescript/announcing-typescript-4-0/). *TypeScript*. 22 September 2020.
3. "Type Compatibility" (https://www.typescriptlang.org/docs/handbook/type-compatibility.html). *TypeScript*. Retrieved 21 March 2018.
4. Bright, Peter (3 October 2012). "Microsoft TypeScript: the JavaScript we need, or a solution looking for a problem?" (https://arstechnica.com/information-technology/2012/10/microsoft-typescript-the-javascript-we-need-or-a-solution-looking-for-a-problem/). *Ars Technica*. Condé Nast. Retrieved 26 April 2015.
5. "TypeScript Programming with Visual Studio Code" (https://code.visualstudio.com/docs/languages/typescript). *code.visualstudio.com*. Retrieved 12 February 2019.
6. "borisyankov/DefinitelyTyped" (https://github.com/borisyankov/DefinitelyTyped). *GitHub*. Retrieved 26 April 2015.
7. TypeScript Homepage (http://www.typescriptlang.org/), *"Visual Studio includes TypeScript in the box, starting with Visual Studio 2013 Update 2"*
8. TypeScript 1.0 Tools for Visual Studio 2012 (https://visualstudiogallery.msdn.microsoft.com/fa041d2d-5d77-494b-b0ba-8b4550792b4d)
9. Foley, Mary Jo (1 October 2012). "Microsoft takes the wraps off TypeScript, a superset of JavaScript" (https://www.zdnet.com/microsoft-takes-the-wraps-off-typescript-a-superset-of-javascript-7000004993/). *ZDNet*. CBS Interactive. Retrieved 26 April 2015.
10. Somasegar, S. (1 October 2012). "Somasegar's blog" (http://blogs.msdn.com/b/somasegar/archive/2012/10/01/typescript-javascript-development-at-application-scale.aspx). Microsoft. Retrieved 26 April 2015.
11. Baxter-Reynolds, Matt (1 October 2012). "Microsoft TypeScript: Can the father of C# save us from the tyranny of JavaScript?" (https://www.zdnet.com/microsoft-typescript-can-the-father-of-c-save-us-from-the-tyranny-of-javascript-7000005054/). *ZDNet*. Retrieved 26 April 2015.
12. Jackson, Joab (1 October 2012). "Microsoft Augments Javascript for Large-scale Development" (http://www.cio.com/article/717679/Microsoft_Augments_Javascript_for_Large_scale_Development). *CIO*. IDG Enterprise. Retrieved 26 April 2015.
13. "Microsoft augments JavaScript for large-scale development" (http://www.infoworld.com/d/application-development/microsoft-augments-javascript-large-scale-development-203737). *InfoWorld*. IDG. 1 October 2012. Retrieved 26 April 2015.
14. Turner, Jonathan (2 April 2014). "Announcing TypeScript 1.0" (http://blogs.msdn.com/b/typescript/archive/2014/04/02/announcing-typescript-1-0.aspx). *TypeScript Language team blog*. Microsoft. Retrieved 26 April 2015.

15. Miguel de Icaza (1 October 2012). "TypeScript: First Impressions" (http://tirania.org/blog/archive/2012/Oct-01.html). Retrieved 12 October 2012. "*But TypeScript only delivers half of the value in using a strongly typed language to Unix developers: strong typing. Intellisense, code completion and refactoring are tools that are only available to Visual Studio Professional users on Windows. There is no Eclipse, MonoDevelop or Emacs support for any of the language features*"
16. "Microsoft TypeScript: Can the father of C# save us from the tyranny of JavaScript?" (https://www.zdnet.com/microsoft-typescript-can-the-father-of-c-save-us-from-the-tyranny-of-javascript-7000005054/). ZDNet. 1 October 2012. Retrieved 12 October 2012. "*And I think this is a pretty big misstep. If you're building web apps that run on anything other than Windows, you're likely using a Mac and most likely not using Visual Studio. You need the Visual Studio plug-in to get the IntelliSense. All you get without Visual Studio is the strong-typing. You don't get the productivity benefits you get from IntelliSense.*."
17. "TypeScript-Unterstützung für Eclipse" (http://www.heise.de/developer/meldung/TypeScript-Unterstuetzung-fuer-Eclipse-1930408.html). *heise Developer*. 6 August 2013. Retrieved 26 April 2015.
18. "TypeScript" (http://marketplace.eclipse.org/content/typescript#.VAmSNvm1bYg). *Eclipse Marketplace*. Eclipse Foundation. Retrieved 26 April 2015.
19. "TypeStrong: The only TypeScript package you will ever need" (https://github.com/TypeStrong/atom-typescript). Retrieved 21 July 2016.
20. Hillar, Gastón (14 May 2013). "Working with TypeScript in Visual Studio 2012" (http://www.drdobbs.com/windows/working-with-typescript-in-visual-studio/240154792). *Dr. Dobb's Journal*. Retrieved 26 April 2015.
21. "TypeScript 0.9 arrives with new compiler, support for generics" (https://www.theregister.co.uk/2013/06/18/typescript_update_0_9/). *The Register*. 18 June 2013. Retrieved 26 April 2015.
22. Hejlsberg, Anders (2 April 2014). "TypeScript" (http://channel9.msdn.com/Events/Build/2014/3-576). *Channel 9*. Microsoft. Retrieved 26 April 2015.
23. Jackson, Joab (25 February 2014). "Microsoft TypeScript graduates to Visual Studio" (http://www.pcworld.com/article/2101920/microsoft-typescript-graduates-to-visual-studio.html). *PC World*. IDG. Retrieved 26 April 2015.
24. Turner, Jonathan (21 July 2014). "New Compiler and Moving to GitHub" (http://blogs.msdn.com/b/typescript/archive/2014/07/21/new-compiler-and-moving-to-github.aspx). *TypeScript Language team blog*. Microsoft. Retrieved 26 April 2015.
25. Bright, Peter (22 September 2016). "TypeScript, Microsoft's JavaScript for big applications, reaches version 2.0" (https://arstechnica.com/information-technology/2016/09/typescript-microsofts-javascript-for-big-applications-reaches-version-2-0/). *Ars Technica*. Condé Nast. Retrieved 22 September 2016.
26. "Announcing TypeScript 3.0" (https://devblogs.microsoft.com/typescript/announcing-typescript-3-0/). 30 July 2018. Retrieved 16 March 2020.
27. "TypeScript 3.0" (https://www.typescriptlang.org/docs/handbook/release-notes/typescript-3-0.html). 30 July 2018. Retrieved 16 March 2020.
28. Anders Hejlsberg (5 October 2012). "What is TypeScript and why with Anders Hejlsberg" (http://www.hanselminutes.com/340/what-is-typescript-and-why-with-anders-hejlsberg). www.hanselminutes.com. Retrieved 15 January 2014.
29. S. Somasegar (1 October 2012). "TypeScript: JavaScript Development at Application Scale" (http://blogs.msdn.com/b/somasegar/archive/2012/10/01/typescript-javascript-development-at-application-scale.aspx). msdn.com. Retrieved 27 November 2013.
30. optional static typing is called gradual typing
31. Klint Finley (1 October 2012). "Microsoft Previews New JavaScript-Like Programming Language TypeScript" (https://techcrunch.com/2012/10/01/microsoft-previews-new-javascript-like-programming-language-typescript/). TechCrunch. Retrieved 27 November 2013.

32. "Angular 2" (https://angular.io/docs/ts/latest/guide/upgrade.html). *angular.io*. Retrieved 4 May 2016.
33. "Welcome to TypeScript" (http://www.typescriptlang.org/). *typescriptlang.org*. Microsoft. Retrieved 26 April 2015.
34. TypeScript Language Specification p.24 (http://www.typescriptlang.org/Content/TypeScript%20Language%20Specification.pdf) Archived (https://web.archive.org/web/20131117065339/http://www.typescriptlang.org/Content/TypeScript%20Language%20Specification.pdf) 17 November 2013 at the Wayback Machine
35. Turner, Jonathan (18 June 2013). "Announcing TypeScript 0.9" (http://blogs.msdn.com/b/typescript/archive/2013/06/18/announcing-typescript-0-9.aspx). *TypeScript Language team blog*. Microsoft.
36. "Generics in Typescript" (https://www.typescriptlang.org/docs/handbook/generics.html#working-with-generic-type-variables). Microsoft.
37. Sönke Sothmann (31 January 2014). "An introduction to TypeScript's module system" (http://blog.oio.de/2014/01/31/an-introduction-to-typescript-module-system/). blog.oio.de. Retrieved 21 February 2014.
38. "niutech/typescript-compile" (https://github.com/niutech/typescript-compile). *GitHub*. Retrieved 26 April 2015.
39. Olivier Bloch (1 October 2012). "Sublime Text, Vi, Emacs: TypeScript enabled!" (http://blogs.msdn.com/b/interoperability/archive/2012/10/01/sublime-text-vi-emacs-typescript-enabled.aspx). Microsoft. Retrieved 28 October 2012.
40. "TypeScript support in WebStorm 6" (http://blog.jetbrains.com/webide/2013/02/typescript-support-in-webstorm-6). JetBrains.
41. "TypeScript support in ReSharper 8.1" (http://blog.jetbrains.com/dotnet/2013/10/28/typescript-support-resharper-81/). JetBrains. 28 October 2013.
42. ReSharper: The Visual Studio Extension for .NET Developers by JetBrains (https://www.jetbrains.com/resharper)
43. "atom-typescript" (https://atom.io/packages/atom-typescript). *Atom*. Retrieved 9 January 2020.
44. "TypeStrong/grunt-ts" (https://github.com/basarat/grunt-ts). *GitHub*. Retrieved 26 April 2015.
45. "ppedregal/typescript-maven-plugin" (https://github.com/ppedregal/typescript-maven-plugin). *GitHub*. Retrieved 26 April 2015.
46. "ivogabe/gulp-typescript" (https://github.com/ivogabe/gulp-typescript). *GitHub*. Retrieved 14 July 2017.
47. "sothmann/typescript-gradle-plugin" (https://github.com/sothmann/typescript-gradle-plugin). *GitHub*. Retrieved 26 April 2015.
48. "TSLint" (https://palantir.github.io/tslint/).
49. Palantir (19 February 2019). "TSLint in 2019" (https://medium.com/palantir/tslint-in-2019-1a144c2317a9). *Medium*. Retrieved 24 April 2019.
50. "TSLint Deprecated to Focus Support on typescript-eslint" (https://www.infoq.com/news/2019/02/tslint-deprecated-eslint). *InfoQ*. Retrieved 24 April 2019.

Sources

- "Webclipse : Eclipse Plugin" (https://www.genuitec.com/products/webclipse/) Genuitec. Retrieved 9 November 2016.
- "Angular IDE by Webclipse : Standalone IDE" (https://www.genuitec.com/products/angular-ide/) Genuitec. Retrieved 9 November 2016.

External links

- TypeScript (https://github.com/Microsoft/TypeScript) project at GitHub
- TypeScript Language Specification (https://github.com/Microsoft/TypeScript/blob/730f18955dc17068be33691f0fb0e0285ebbf9f5/doc/spec.md)
- CATS Cross Platform TypeScript Editor (https://github.com/jbaron/cats)
