JavaScript

TypeScript vs JavaScript: Which One You Should Use, and Why

Tim Mouskhelichvili
August 11, 2022

Share

This article will look at TypeScript vs JavaScript: how they compare, and when to use TypeScript vs JavaScript. We'll explore what TypeScript actually is, why it exists, and what its advantages and disadvantages are.

What is TypeScript?

TypeScript is an open-source programming language developed and maintained by Microsoft. It's a superset of JavaScript: any JavaScript code runs in a TypeScript environment.

TypeScript was created to address problems coming with developing large-scale applications in JavaScript. Those kinds of applications contain hundreds of different files. Making one change can affect the behavior of multiple files. Since JavaScript can't validate the connection between those files, developers have a lot of room to make mistakes that cause bugs in production. That's why there was a need for a frontend language like TypeScript that included static type checking.

TypeScript is a compiled language, unlike JavaScript, which is an interpreted language. You need to compile TypeScript to JavaScript for it to work in a browser. To compile TypeScript to JavaScript, you can use the TypeScript npm package.

If you come from an object-oriented language like C# or Java, you'll find many familiar features in the list that TypeScript adds. My favorite ones include the addition of enums, interfaces, access modifiers, namespaces, generics, and many more.

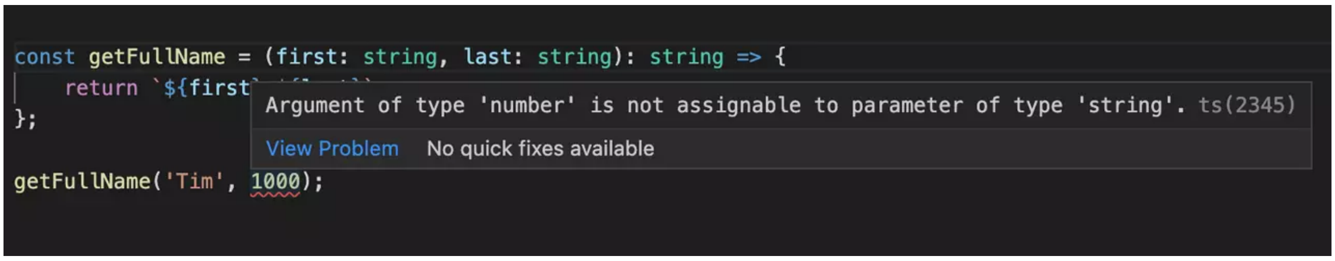
Since many external npm libraries are written in JavaScript, TypeScript needed a way to type-check them. This is where type declaration files (`.d.ts` files) come into play. Those type declarations provide type definitions with information about the code. The compiler uses them to type-check the TypeScript code. Nowadays, most JavaScript libraries have TypeScript definitions written for them. Even if a library lacks those definitions, TypeScript can sometimes infer the types. Alternatively, you can quickly write definitions yourself.

TypeScript is great for large-scale web applications, as you can write frontend and backend code with it. On the backend, you need to install a library called **ts-node** to execute TypeScript code in a **Node.js environment**.

What are the Advantages of TypeScript vs JavaScript?

In 2022, TypeScript joined the world's **top 5 most used languages**. This is unsurprising, because TypeScript constantly develops, adds new features, and has a fantastic developer community that TypeScript's team listens to.

TypeScript's most significant advantage over regular JavaScript is its compiler, which provides type and error checking. It checks for type validity and shows errors in real time. It's particularly useful when refactoring your code, because the compiler shows you things you may have missed.



```
const getFullName = (first: string, last: string): string => {
    return `${first
};

getFullName('Tim', 1000);
```

```
Argument of type 'number' is not assignable to parameter of type 'string'. ts(2345)
```

When used in pairs with a newer IDE like Visual Studio Code, TypeScript offers great Intellisense features such as code hinting and code completion. Those features help increase the speed of the development of the program.

Using the `tsconfig` file, a developer can configure TypeScript's behavior. For example, TypeScript provides an option to transpile the code to a previous version of JavaScript, so the code runs on all browsers. Another option a developer has is TypeScript's strict mode, which adds many more checks to ensure the code's correctness.

Another great TypeScript feature is its ability to run parallel with JavaScript code, making it easier to implement in legacy projects with many JavaScript files.

Also, TypeScript allows using some ES6 and ES7 candidate features not yet supported by major browsers. For example, the optional chaining operator and **class constants** were available in TypeScript long before ECMAScript officially supported them.

Finally, TypeScript provides many features that JavaScript lacks, which makes coding more fun and enjoyable. Features such as interfaces, enums, and generics, to name a few, significantly improve your code readability.

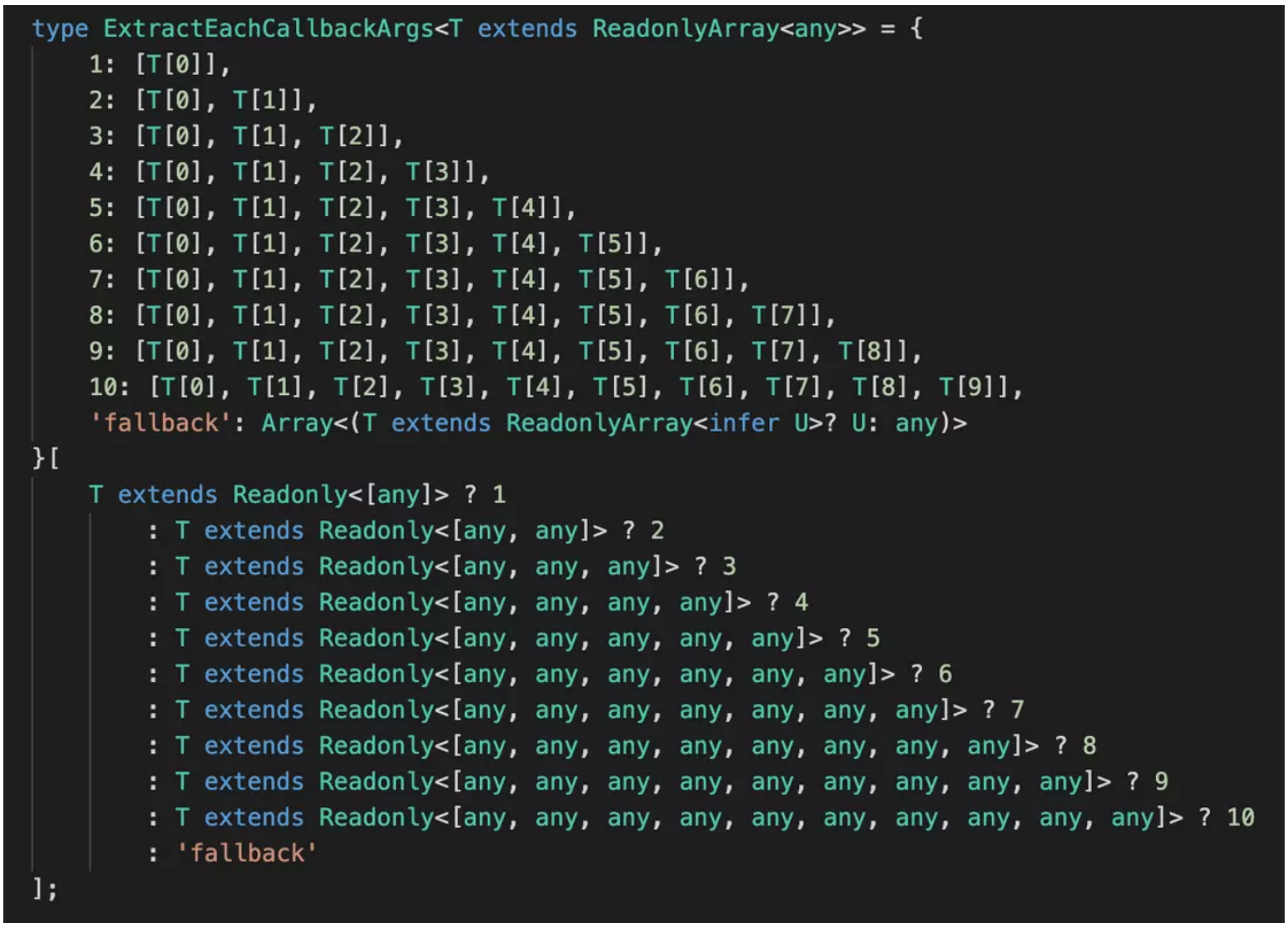
What are the Disadvantages of TypeScript vs JavaScript?

Although TypeScript offers multiple advantages, it also brings some disadvantages. None of those disadvantages are deal-breakers, but a new developer needs to be aware of them and consider them.

The most significant disadvantage of TypeScript is the false sense of security it brings to its users. Indeed, when developers come to TypeScript, they often rely too much on its compiler, hoping it will flag every type error. You shouldn't expect your TypeScript code to be 100% bulletproof. That's a choice that TypeScript's development team made: wanting TypeScript to be a balance between flexibility and correctness. Although TypeScript isn't bulletproof, it's still better than plain JavaScript code, in my opinion, because its compiler will find bugs you wouldn't have otherwise. Indeed, it's **estimated** that TypeScript helps find 15% more bugs than JavaScript.

Another disadvantage of TypeScript is the required extra compilation step. This step can slow down the build time and complicate the bundler setup.

Since TypeScript adds many new features that can be unknown to a frontend developer, it increases the learning curve of a codebase. Indeed, when used to its full potential, TypeScript can be challenging to understand for untrained developers and requires a lot of googling or mentorship from elder teammates.



```
type ExtractEachCallbackArgs<T extends ReadonlyArray<any>> = {
    1: [T[0]],
    2: [T[0], T[1]],
    3: [T[0], T[1], T[2]],
    4: [T[0], T[1], T[2], T[3]],
    5: [T[0], T[1], T[2], T[3], T[4]],
    6: [T[0], T[1], T[2], T[3], T[4], T[5]],
    7: [T[0], T[1], T[2], T[3], T[4], T[5], T[6]],
    8: [T[0], T[1], T[2], T[3], T[4], T[5], T[6], T[7]],
    9: [T[0], T[1], T[2], T[3], T[4], T[5], T[6], T[7], T[8]],
    10: [T[0], T[1], T[2], T[3], T[4], T[5], T[6], T[7], T[8], T[9]],
    'fallback': Array<(T extends ReadonlyArray<infer U>? U: any)>
}[
    T extends Readonly<[any]> ? 1
        : T extends Readonly<[any, any]> ? 2
        : T extends Readonly<[any, any, any]> ? 3
        : T extends Readonly<[any, any, any, any]> ? 4
        : T extends Readonly<[any, any, any, any, any]> ? 5
        : T extends Readonly<[any, any, any, any, any, any]> ? 6
        : T extends Readonly<[any, any, any, any, any, any, any]> ? 7
        : T extends Readonly<[any, any, any, any, any, any, any, any]> ? 8
        : T extends Readonly<[any, any, any, any, any, any, any, any, any]> ? 9
        : T extends Readonly<[any, any, any, any, any, any, any, any, any, any]> ? 10
        : 'fallback'
];
```

Lastly, some developers complain that using TypeScript requires adding a lot of extra code to define the types. Although true, this saves time in the long run because it is simpler to onboard new project members. It also makes refactoring of the codebase easier.

When Should You Use TypeScript for a New Project?

Adopting a new technology you haven't had any experience working with can be daunting. However, with TypeScript, it's worth the shot, because its advantages outweigh its disadvantages.

If you're working on a project alone and aren't planning to bring more help from outside, I recommend choosing TypeScript for your next web application project.

Although the setup and initial few hours of development can be tricky, you'll quickly fall in love with TypeScript and never want to return to regular JavaScript. Initially, you'll write your code slower, but TypeScript will save you debugging and refactoring time in the long run.

If you work on a project with teammates, the decision is more complex, because it requires consent from the team and management. Although, ultimately, using TypeScript will help your team, it will slow down the development in the short run. Also, the team needs to be prepared to spend some time learning TypeScript, its features, and its **best practices**. This is where having an experienced teammate with TypeScript or another object-oriented programming language (such as C#) will help the team have a smooth transition.

If your team is prepared to sacrifice short-term performance loss and initial time to learn TypeScript, I recommend using TypeScript in your project. You won't regret it.

When Shouldn't You Use TypeScript for a New Project?

Even if TypeScript is generally fantastic, there are still reasons why I wouldn't recommend using it.

The biggest reason not to use TypeScript is if you or your team have to respect a tight deadline. Deadlines are stressful enough, and adding a new technology that you have no experience working with isn't recommended. Unfortunately, **learning TypeScript** takes time, and this time may be better spent elsewhere for projects with a deadline.

Also, TypeScript can be challenging to **configure**, especially for beginners. It may require installing multiple npm libraries and working with a bundler (like webpack). If you aren't prepared to spend the time learning all of this new information, don't use it.

Another thing to consider is that using TypeScript will increase the developers' entry threshold. If the project is open-source, this can increase developers' difficulty in contributing to it.

Also, for managers and hiring personnel, using TypeScript in a project means that the developers you hire must have experience using TypeScript or another OOP language. Adopting TypeScript raises the minimum job qualification to work on the project and can increase the project's budget, because you need to hire more experienced developers.

TypeScript vs JavaScript: What about Legacy Projects?

Migrating a legacy project written with regular JavaScript to TypeScript brings a lot of advantages to the project development life-cycle. It helps find bugs you haven't noticed and simplifies the maintenance of the project.

For smaller projects, migrating from JavaScript to TypeScript can be easily accomplished by installing the required libraries, changing the file extensions from `.js` to `.ts`, and fixing the errors outputted by the TypeScript compiler.

It can be more complicated for projects with hundreds of different JavaScript files. Luckily, you can adopt TypeScript incrementally. You can prioritize migrating some files to TypeScript and running them parallel to legacy JavaScript code.

Another option is to keep the legacy code and only write new features in TypeScript. When a feature touches a part of the legacy code, you can migrate it simultaneously while working on it.

TypeScript vs JavaScript: the Verdict

In this TypeScript vs JavaScript comparison, you've seen that TypeScript is a great object-oriented language that will help you build large-scale applications more efficiently.

Even though it has a few disadvantages, like the code complexity that it adds or the added compile time, it will save you time in the long run.

If you work alone, I strongly encourage you to build your next application using TypeScript.

If you work with a team, adopting TypeScript may require some convincing. I encourage you to **start building** a **TypeScript project** in your free time. You can then show it to your teammates and explain all of TypeScript's advantages.

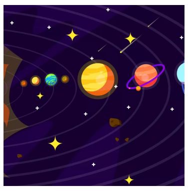
Share This Article

Tim Mouskhelichvili

Hello 👏! I am Tim Mouskhelichvili, a Freelance Developer & Consultant from Montreal, Canada CA. I specialize in React, Node.js & TypeScript application development. Also, I have a blog where I write about web technologies.

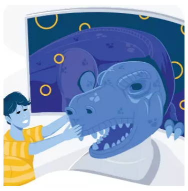
javascript **TypeScript**

Up Next



9 Best JavaScript and TypeScript ORMs for 2022

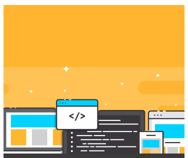
Michael Wanyoike



Learn Deno: A Secure JavaScript & TypeScript Runtime

Nilson Jacques

How TypeScript Makes You a Better JavaScript Developer



Joe Previte

Experiment with ECMAScript 6 on Babylon.js with TypeScript 1.5

David Rousset

Getting Started with Angular 2 using TypeScript

Ravi

Introducing TypeScript — JavaScript on Steroids

Craig Buckler

Stuff we do

- Premium
- Newsletters
- Forums

About

- Our story
- Terms of use
- Privacy policy
- Corporate memberships

Contact

- Contact us
- FAQ
- Publish your book with us
- Write an article for us
- Advertise

Connect