|  |
| --- |
| J176873 |
| Typescript Research Paper |
| Matthew taggart |

|  |
| --- |
|  |

Over 20 years after its introduction to the programming community, JavaScript is now one of the most widespread cross-platform languages ever created. Starting as a small scripting language for adding trivial interactivity to webpages, JavaScript has grown to be a language of choice for both frontend and backend applications of every size. While the size, scope, and complexity of programs written in JavaScript has grown, it's ability to express the relationships between different units of code has not. Combined with JavaScript’s rather peculiar runtime semantics, this mismatch between language and program complexity has made JavaScript development a difficult task to manage at scale.

But it has some issues.

The most common kinds of errors that programmers write can be described as type errors: a certain kind of value was used where a different kind of value was expected. This could be due to simple typos, a failure to understand the API surface of a library, incorrect assumptions about runtime behavior, or other errors. The goal of TypeScript is to be a static typechecker for JavaScript programs - in other words, a tool that runs before your code runs (static) and ensures that the types of the program are correct (typechecked).

TypeScript is a free and open-source programming language developed and maintained by Microsoft.

It is designed for the development of large applications and transpiles to JavaScript. Which basically means it compiles in JavaScript!

JavaScript code is valid TypeScript code; TypeScript is a superset of JavaScript. You can almost rename your .js files to .ts files and start using TypeScript. TypeScript files are compiled to readable JavaScript, so that migration back is possible and understanding the compiled TypeScript is not hard at all. TypeScript builds on the successes of JavaScript while improving on its weaknesses.

Microsoft's TypeScript programming language over the past few years has been steadily climbing the popularity rankings in Stack Overflow's huge annual developer survey, this year knocking off Java to crack the top five.

Since 2019, that top five has been dominated by JavaScript, HTML/CSS, SQL, Python and Java, with some juxtaposition in slots 3-5 after the cemented-in-stone No. 1 and No. 2 rankings for JavaScript and HTML/CSS.

Since 2018, TypeScript has climbed steadily, this year jumping two spots from 7th to 5th:

2022 - 5  
2021 - 7  
2020 - 9  
2019 - 10  
2018 - 12

TypeScript jumped a whopping 4.64 percent points (from being listed by 30.19 percent of respondents in 2021 to 34.83 percent in 2022) to overtake Java (which fell slightly from 33.91 percent in 2021 to 33.27 percent in 2022). That's by far the biggest jump among the top 20 languages.

1. Compile-Time Bug Detection

Like JS, TS detects program errors by highlighting unexpected behaviors. This bug detection feature happens during compilation as opposed to execution. Through this, TS can generate more efficient code.

2. Static Typing

TypeScript is a syntactic superset of JavaScript which adds static typing.

This basically means that TypeScript adds syntax on top of JavaScript, allowing developers to add types.

TypeScript being a "Syntactic Superset" means that it shares the same base syntax as JavaScript, but adds something to it.

3. Offers Large-Scale Solution Support

The majority of websites operate in complex systems. However, JavaScript was not developed for this. TS uses object-oriented features such as interfaces, modules, and generics to circumvent the aforementioned issue. In addition, top JS libraries and frameworks support TS. And it compiles into readable and standard-based JavaScript.

4. Easy to Learn for JS Developers

TypeScript uses JS syntax and other features. For these reasons, it is easier to learn for JS developers. These features make TS a popular programming language. According to Stack Overflow’s 2021 Developer Survey, it is the third most loved language. It's likely that this trend will continue.

UPDATE:

In Backend Web development I learned to use the Windows Subsystem for Linux (WSL) as a development environment, and I continued to use it here in conjunction with Typescript.

I followed these steps to build it:

1. Install Node.js and npm (the Node.js package manager) in your WSL environment, if they are not already installed. You can do this by running the following commands:

Update the package manager

sudo apt update

Install Node.js and npm

sudo apt install nodejs npm

1. Install the TypeScript compiler globally using npm:

sudo npm install -g typescript

1. Navigate to the directory containing your TypeScript file using the cd command. For example:

cd path/to/directory

1. Compile the TypeScript file to a JavaScript file using the tsc command. For example, if your TypeScript file is named index.ts, you can compile it to a JavaScript file named script.js using the following command:

tsc index.ts

This also validates the javascript file for errors!