What is TypeScript?

In this lesson, we will get a high level understanding of the programming language TypeScript.

TypeScript is a language on top of JavaScript. Everything possible in JavaScript is available in TypeScript – it is a superset of JavaScript. It provides two strong advantages. The first one is that it transpiles TypeScript into JavaScript so the advanced ECMAScript features not available to all web browsers can be used by providing a polyfill. It acts as a combination of other static checkers combined with Babel. The second advantage of TypeScipt is that it can enforce static typing to catch potential issues earlier in the lifecycle of development. It reduces the need for some unit tests and can analyze the code to find runtime errors at design time.

TypeScript was made public since 1 October 2012, but its inception started at Redmond two years prior as an internal product (2010) at Microsoft. The project is open-source, hosted on Github.com under an Apache2 license.

TypeScript's popularity increases year after year. It has 317 branches, 228 contributors, 48 releases, 188 pull requests monthly created with over 3 600 forks and 25 000 stars. The consistent cadence of a new version releases every two months and the great ecosystem of people contributing make it a stellar

open source project. It evolves rapidly to sustain developers' needs, and it follows the rapid pace of web technologies.

TypeScript is far from being used only by its creator, Microsoft. While it's true that many Microsoft products use TypeScript, such as Microsoft Teams, Visual Studio Team Services (VSTS), the online version of Office, and Visual Studio Code (VSCode), to say the least, other technology giants have been borrowing the advantages of static typing. Google has been using TypeScript since Angular 2, Slack has migrated their JavasScript code base to TypeScript as well, and other companies like Ubisoft, Asana, Ericsson, Upwork, Bloomberg, and Lyft are following suit. The reason I mention these systems that use TypeScript is to highlight the investment of thousands of developers and the millions of lines in these technologies. Most of them come from different backgrounds. Aside from the tremendous amount of work and knowledge poured across the industry, it illustrates that many people before you had to decide whether to go with TypeScript or not—and chose to do so. While most of the examples provided are big projects, they all do have one thing in common, i.e. some people must write and maintain the code. Static typing shines in these areas, from small to big projects.

Here is a quick note about who is leading TypeScript. Anders Hejlsberg manages TypeScript. He is the creator of Turbo Pascal, Delphi, C# and now TypeScript. Hejlsberg graduated in the '80s, has received many awards in his career, and is the person who made the most commits on GitHub on that project. Having such a prolific engineer at the head of TypeScript boosts the maturity of the language.

The TypeScript community is beyond the repository where it resides. StackOverflow's members are strong, with more than 35 000 questions. There are plenty of examples with TypeScript code with many different frameworks. It's not difficult to find Angular or React examples written in TypeScript or popular projects like the famous TodoMVC.

TypeScript is all about having a strongly typed language at design time, but the code produced doesn't contain any type. The type gets erased because Typescript removes types, interfaces, aliases at transpilation and ends up with a common JavaScript file. The removal may sound logical but the lack of type at runtime results in a design that must not rely on the type dynamically at runtime. We will see examples of this when working on type comparisons to

figure out the interface used. Keep in mind that interface and type are not

available at runtime, hence how can this check be performed when the code executed is a challenge?