Introduction to TypeScript

Estimated Effort: 5 mins

Learning Objectives

After reading this article, you will be able to:

- Differentiate between JavaScript, JSX, and TypeScript
- Describe the advantages of using TypeScript for React applications rather than JavaScript

Now that you have been introduced to JSX, letâ€™s discuss TypeScript. TypeScript is a superset of JavaScript, meaning all JavaScript is TypeScript, but not all TypeScript is JavaScript. React allows you to choose whether to use either JavaScript or TypeScript. Before we discuss why you may want to choose TypeScript over JavaScript, letâ€™s learn a little more about TypeScript.

TypeScript is a compiled language that supports type-checking. TypeScript is statically typed. This means that variables are static. Once they are defined, a variable's type, such as â€œnumâ€ or â€œstring,â€ cannot be changed. A variable that is declared a number cannot later take a string value. Using TypeScript can save a lot of headaches later to help avoid â€œrun-time errorsâ€ when the code is being run or avoid hard-to-identify bugs during testing. With TypeScript many of these errors are identified as â€œtype errorsâ€ during compilation rather than â€œundefinedâ€ bugs at run-time. During compilation of the Typescript code, this is called â€œtype checking,â€ where the compiler ensures that once a variable is declared, it is not reassigned to another data type.

Now letâ€™s explore the relationship between JSX and TypeScript. Recall that JSX provides additional syntax to JavaScript, allowing you to write HTML-like code for JavaScript. JSX requires a compiler to translate the JSX into JavaScript. Babel is a popular compiler for JSX.

TypeScript supports embedding. This means you can embed HTML directly into TypeScript, and the compiler will translate the HTML and the TypeScript into JavaScript at compile time, similarly to how the Babel compiler translates JSX into JavaScript.

There are a couple of compilation differences, though, if you choose to compile your JSX or JavaScript with Babel vs. the TypeScript compiler. First of all, Babel does not support type-checking. Secondly, the TypeScript compiler compiles the entire project simultaneously rather than one file at a time. This means that type errors are caught amongst different files rather than only within a single file.

There are several advantages to using TypeScript rather than JavaScript in your React application and possibly a few drawbacks. Regarding advantages:

- TypeScript makes it easy to define Prop types in React. This makes writing code with an IDE that supports code completion a breeze since the IDE automatically populates the Prop type.
- Most common third-party libraries support TypeScript definitions.
- As mentioned, TypeScript has static type-checking, which enables you to help identify errors earlier in the development process.
- Refactoring code becomes easier since types are known.
- There will be fewer â€œundefinedâ€ errors at run-time due to type-checking at compile time.
- Code is easier to read and maintain than JavaScript due to typed variables.
- You can still write JavaScript and use the TypeScript compiler.

There are also possibly a few challenges with using TypeScript rather than JavaScript for React applications:

- There is a learning curve
- Compilation takes a little longer than using Babel with JSX
- Third-party libraries could be missing Type definitions, though this is rare because TypeScript has grown very popular

Next Steps

You can find more information about TypeScript in this link.

Author(s)

Upkar Lidder