Definition Files and Global Definition Files

In this lesson, you see what a global definition file is.

TypeScript works seamlessly with JavaScript code because of the concept of definition files. Definition files contain the type of each public function or variable that an external module not written in Typescript offers. The most popular definition file is probably lib.d.ts which contains thousands of lines of definitions about the core functionalities of JavaScript. It comes with TypeScript, and you shouldn't have to care about much else that it provides, except for Intellisense, because it contains the types of the ECMAScript core language functions and variables; for example, the definition of the function `parseFloat`.

TypeScript's configuration allows adding other `system` libraries. It works as a supplement to the core features of a specific ECMAScript target; for example, the `promise` library, which allows for developing with a library not yet incorporated inside any core ECMAScript target. Adding a system library requires changing the tsconfig.json under `lib`. Projects automatically add libraries from the lib's files. Unlike modules, lib doesn't need the keyword `import`. You can dive into the lib.d.ts file by going to the definition (F12 in most IDEs). It opens the definition file at the variables, functions, or types that has focus. Exploring libraries under `lib` is a great way to learn what ECMAScript and ECMAScript's neighbor libraries contain. The lib.d.ts changes depending on the selected target defined in the TypeScript configuration.

Another common definition file is globals.d.ts. It's an extension to lib.d.ts. Any addition to this file is globally available. However, it is better to have a custom definition file per module and not to mix different non-related definitions. The globals.d.ts is an option for cases where you need to change something from the JavaScript library. Since JavaScript lets you modify, add, or remove a declaration to the core library, TypeScript must also support a way to type any change to the core library. The use of global.d.ts brings all the modifications in

a single file. However, globals.d.ts is not the only place that can affect the

global score. In fact, the use of `declare global` and the interface within the curly bracket sets the definition in the global scope.

In the section about the declaration, you were briefly introduced the use of the keyword `declare`. Definition files often use `declare` to mention that a variable exists, not declared in this file necessarily, but existing as a specific type. That's is right, since many variables, e.g. `window` or `Math`, do exist beyond Typescript, in JavaScript. Most of the types in lib.d.ts use interface as well. The justification to use interface is because of its property of being extensional. The extension feature goes along with globals.d.ts, which allows modifying an already defined core interface. It leverages the interface feature that merges multiple definitions with lib.d.ts.