**Diving Deeper Into .d.ts Files**

This course focuses on using TypeScript in your own **applications**.

But another area where you can leverage TypeScript is when building **libraries** (i.e., packages used by other projects and developers).

That's where .d.ts files (as shown in the previous lecture) become more relevant.

.d.ts stands for "**declaration file**" or "**type declaration file**" and those files contain the type declarations needed to bridge the gap between the JavaScript code in which your library may be shipped and the TypeScript code that may be used in the codebase that wants to use your library.

Just to be clear: When creating and sharing a library, you have three main options when it comes to "helping" TypeScript users:

1) Ship your library code in .ts files (i.e., do not compile it to JavaScript)

2) Ship your library code in .js files (i.e., compiled or natively written in JavaScript) AND also ship .d.ts files to provide a "bridge" from JS to TS

3) Ship your library code in .js files (i.e., compiled or natively written in JavaScript) AND rely on other people to create .d.ts files that are installable via @types/<your-library> (or share those .d.ts files in that repository yourself)

Typically, 2) or 3) are chosen - and in both cases, .d.ts files must be created.

If that's something you're interested in you can dive deeper and learn more about writing .d.ts files [here](https://www.typescriptlang.org/docs/handbook/declaration-files/introduction.html).