# Creating a Type Declaration File in TypeScript

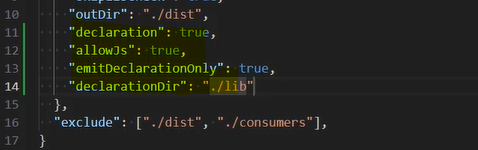
Declaration files supports all the generally used module formats like CommonJS, AMD and ESM.

Declaration files are typescript files with a d.ts extension and used by the editor to provide code-completion. They are also used by the compiler to enforce correct usage.

Types – regular files that use globals (for older code) or modules that use export (ambient).

The types for the private members do not get generated as they are supposed to be used by the consuming classes.

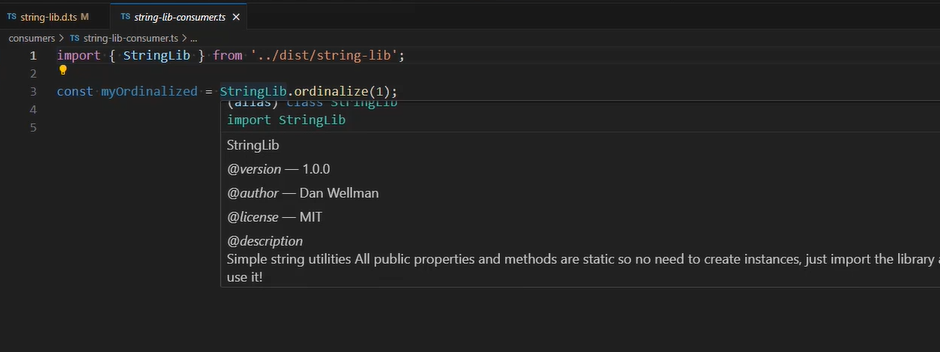
To generate the base version of the declaration file from the JS code, we can set the below config in the tsconfig.json file then we can manually enhance it further –



Using JSDoc with declaration file –

A screenshot of a computer program

Description automatically generated

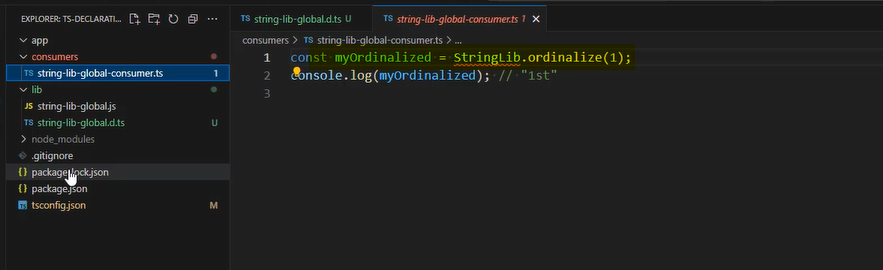


Originally there were two kinds of JS library – libraries that added a global object, e.g. jQuery, another one is libraries that extended built-in objects, e.g. Prototype. Both approaches have significant downsides as extending built-in objects in unwise in case the object changes. Global objects are convenient, but can easily be overwritten.

Instead of these both approach, we should use modular-based coding which is much safer paradigm that eliminates the worst aspects of working with globals and brings many other benefits.

Creating a declaration file for a global library – for global libraries we need to use ‘namespace’ keyword. It is not an ambient module but just an ambient declaration –

In this case the consumer directly use the Global object name which is in different folder and doesn’t import anything. To handle this, we can create the @types folder in the node\_modules folder to let the IDE recognize this global type. This @types folder automatically picked up by the IDE for any third party types installation.



A screen shot of a computer

Description automatically generated

Or, we can use the triple slash directive approach by providing the relative path, but it is an older approach and not recommended one, the earlier one is recommended –

A screen shot of a computer

Description automatically generated

Creating a declaration file for object-extending library –

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

For the declaration file of this approach is a bit different as TS already have a type for String, we just need to extend it by providing definition for custom method which we added in the actual source file –

A screenshot of a computer

Description automatically generated

Creating a declaration file for an ESM library –

For the default export we need to write the ‘declare’ instead of ‘export’ keyword and in the last need to export the library name as default.

A screen shot of a computer program

Description automatically generated

A computer screen shot of a code

Description automatically generated

Creating a declaration file for CommonJS library –

A screen shot of a computer program

Description automatically generated

A computer screen with text

Description automatically generated

By default the TS doesn’t come with the types for Node, we need to install it explicitly by using npm install @types/node. Then only the ‘require’ will show it types otherwise it was ‘any’ –

A computer screen with text

Description automatically generated

In the d.ts file we need to declare the module which is also implicitly exported –

A screen shot of a computer program

Description automatically generated

Declaration file for an AMD file –

A screen shot of a computer program

Description automatically generated

3 ways to use this AMD module –

A computer screen shot of a program code

Description automatically generated

TypeScript cannot extract any type information when using the define or requires functions to load the library as a dependency. So, way 2 and way 3 won’t show type information until we use a workaround for it.

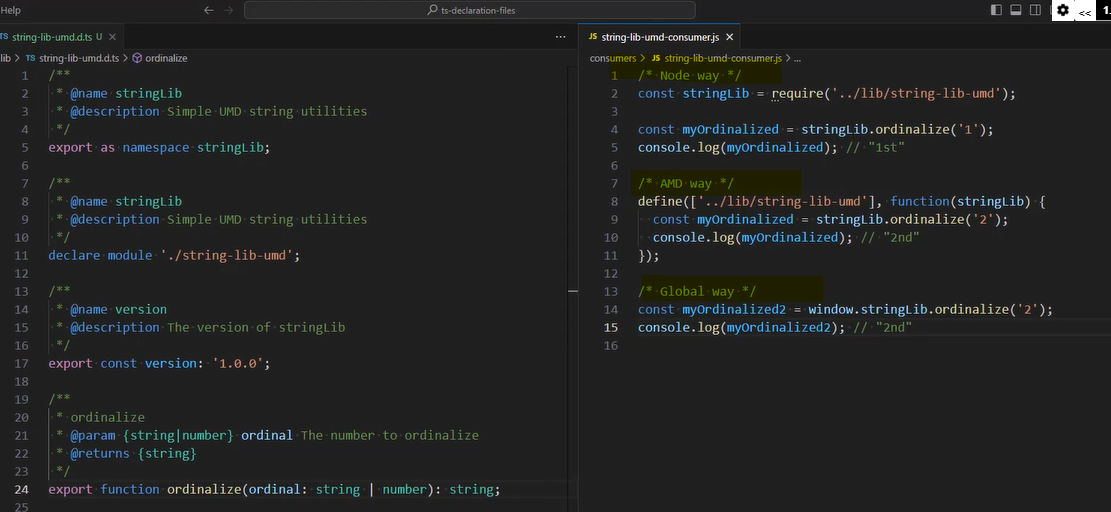
Creating a declaration file for an UMD library – this UMD format allows to write module once and able to run it in browser and Node JS both.

The first ‘if’ to check if it is AMD module, the next ‘if else’ block check if is running in node js, the ‘else’ block means the code is running in the browser. So, this code library can run on node.js, in browsers and even web workers without any problem.

A screen shot of a computer code

Description automatically generated

Declaration file for this format – way to consume the library –



# Angular Misc.

Change detection is a sequence of steps that ensures the UI reflects the current state of the application.

Zone.js patches browser apis and tells angular when anything occurs that 'could' impac the view. It does not tell angular what or where an event occurred. but the new signal where the event has been raised and which component and dependencies need to be changed.