To install -

npm install --global typescript

The typescript packages installed will include a compiler - 'tsc', which compiles the typescript code to pure javascript code.

To define the configuration for the typescript compiler, create a file - tsconfig.json with the following content

{

"compilerOptions": {

"target": "es2018",

"outDir": "./dist",

"rootDir": "./src",

"module": "CommonJS"

}

}

These will tell the compiler that I want to use the latest javascript, that the project's typescript file will in the 'src' folder, and output it produces will be in the 'dist' folder and the commonjs standard should be used for loading code from separate files.

Syntax:

defining a class:

export class ClassName{

public porperty\_name string; //string is a variable type

constructor(variable\_name: string){ //string is a variable type // it can have access modifier - e.g. public, private

}

}

Access keywords are provided by the Typescript, not JS.

Support for static datatypes - e.g public variable\_name string; // making it acceppt only string datatype.

Typescript assumes that all method are public if there is no access modifier mentioned.

Map in JS - Key Value Dictionary Pair

JS spread operator - ... to create array

**Difference between export and public ->**

*https://stackoverflow.com/questions/15760462/why-does-typescript-use-the-keyword-export-to-make-classes-and-interfaces-publ*

The primary reason is that export matches the plans for ECMAScript. You could argue that "they should have used "export" instead of "public", but asides from "export/private/protected" being a poorly matched set of access modifiers, I believe there is a subtle difference between the two that explains this.

In TypeScript, marking a class member as public or private has no effect on the generated JavaScript. It is simply a design / compile time tool that you can use to stop your TypeScript code accessing things it shouldn't.

With the export keyword, the JavaScript adds a line to add the exported item to the module. In your example: here.SomeClass = SomeClass;.

So conceptually, visibility as controlled by public and private is just for tooling, whereas the export keyword changes the output.