**TypeScript**

1. What is TypeScript?
   1. A Programming Language
   2. Free and open source
   3. Developed and maintained by Microsoft
2. Why TypeScript?
   1. Problem with JavaScript Variables as they don’t have type checks.



* 1. Problem with JavaScript functions where it can accept any number of argument.



* 1. Problem with JavaScript Objects as they are loosely structured.



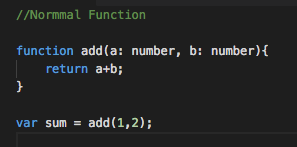
1. How TypeScript works?
2. Code is written into TypeScript which is compiled and converted into JavaScript.
3. The main purpose of TypeScript is to transpile the code which is checked during compile time and not during runtime. This helps a develop to catch the bug before it goes to the browser.



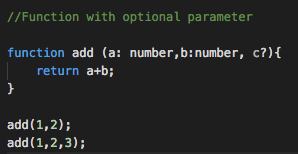
1. Setting up TypeScript
   1. Install VS Code
   2. Install Node.js (platform where TypeScript runs)
   3. Install TypeScript Compiler (npm install typescript –g)
2. Introducing Type Declaration
   1. Create a TypeScript file in VS Code.
   2. To run, type **tsc** which will generate a JavaScript file.
   3. To run js file use node.
   4. Run: hello-world.ts && node hello-world.js



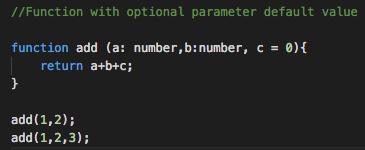
1. Typing with functions
   1. Normal function.



* 1. Function with optional parameter.



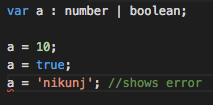
* 1. Function with optional parameter having default value.



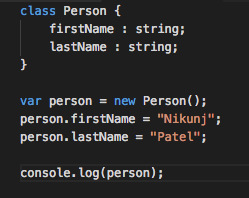
1. Implicit Typing works only when the variable is assigned a value in the same assignment. If the variable is just defined, it can take any datatype.



1. Union types
   1. When a variable is supposed to have multiple datatypes, we can use union data types which is separated by pipe symbol.

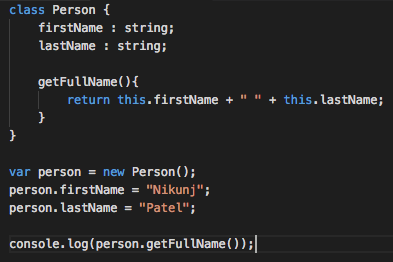


1. Classes

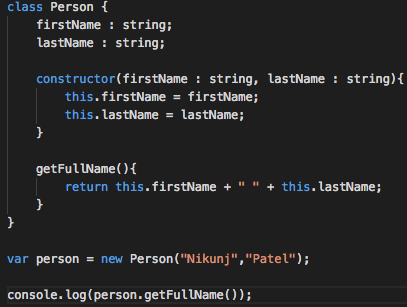


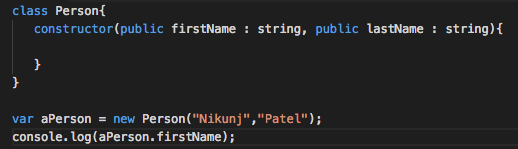
Screen%20Shot%202018-07-04%20at%208.50.01%20PM.png

1. Methods



1. Constructors
   1. Constructor overloading is not supported in TypeScript

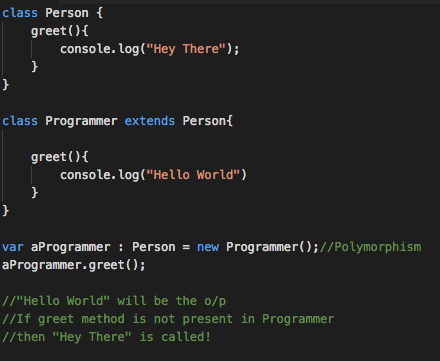




1. Inheritance



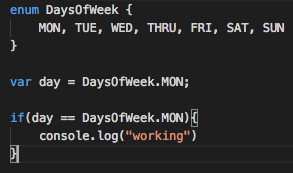
1. Polymorphism



1. Member visibility (Public, Private and Protected)
2. Read-only



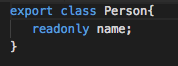
1. Enums



1. Generics
   1. sample example

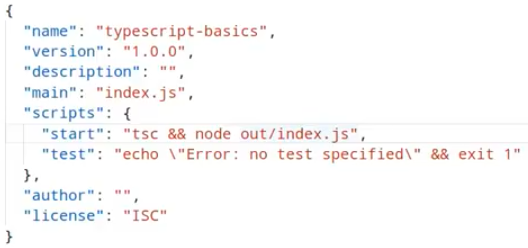


1. Modules
   1. Export the class so that it can be reused
   2. Import class so that it can be used inside the other class



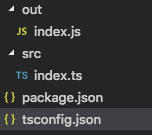
**Screen%20Shot%202018-07-05%20at%207.33.07%20AM.png**

1. TypeScript Compiler Argument
   1. to change the output js file generated to other name
      1. tsc test.js –out test-new.js (this will generate file with this name)
   2. make compilation process continuous
      1. tsc test.js –out test-new.js --watch
2. Configuration file which has all commands to run (tsconfig. json)
   1. To create type **tsc –init**
   2. There are various configuration which can be changed
   3. To run a ts file, just type tsc. Other things are handled by tsconfig. Json
   4. Some examples of tsconfig.json
      1. Strict: enables type checking
      2. outDir : all .js files are stored in the specified folder
      3. noEmitOnError : generates .js file onlyif there is no error
3. Creating an npm project
   1. initialize the node project (npm init). This will initialize the package.json

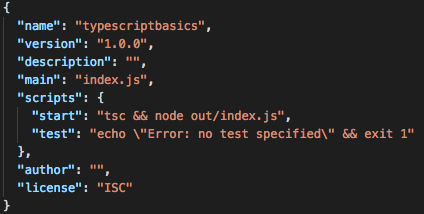


* 1. after the above configuration run the node project using npm start

1. Installing libraries using node
   1. Npm install lodash –save
2. Project Configuration (REST API)
   1. npm init
   2. tsc –init
   3. create a src and out folder where all input/output files are present and change configuration in tsconfig.json for rootDir attribute to point to this folder.
   4. Package Structure



* 1. Package.json

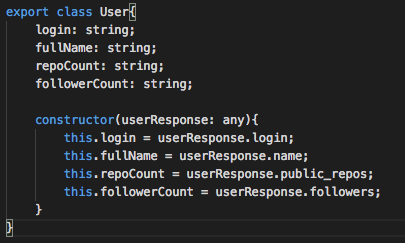


* 1. Tsconfig.json

Screen%20Shot%202018-07-05%20at%208.31.29%20AM.png

* 1. To run the project type npm start, which will load index.js and show the output

1. Installing dependencies
   1. request dependencies which allows the rest api calls and lodash for sorting
      1. npm install request lodash –save
   2. install type definition
      1. npm install @types/request @types/lodash –save-dev
2. Api’s which will be used are
   1. <https://api.github.com/users/nikunjpatel1989>
   2. <https://api.github.com/users/nikunjpatel1989/repos>
3. Model Class



1. Rest Service Class



1. Namespace or Modules: A namespace is a way to logically group related code. This is inbuilt into TypeScript unlike in JavaScript where variables declarations go into a global scope and if multiple JavaScript files are used within same project there will be possibility of overwriting or misconstruing the same variables, which will lead to the “global namespace pollution problem” in JavaScript.