1.2 Migration: JavaScript to TypeScript



This section will guide you to:

* Convert JavaScript to TypeScript

This guide has four subsections, namely:

1.2.1 Initializing the tsconfig.json file

1.2.2 Setting up the TypeScript environment

1.2.3 Migrating JavaScript to TypeScript

1.2.4 Pushing the code to GitHub repositories

**Note:** You will be able to proceed with this demonstration only if you have completed all the steps mentioned in the document **TypeScript with Grunt.**

**Step 1.2.1:** Initializing the tsconfig.json file

* Create a folder **JavaScript to TypeScript** and import the project in Visual Studio Code.
* Create two folders, **src** and **build** in your project directory
* Open the terminal navigating to **view** > **Terminal**
* Run the following to initialize tsconfig.json file:

*tsc –init*

* Replace the source code with the following source code in the tsconfig.json file:

{

"compilerOptions": {

"module": "commonjs",

"sourceMap": true,

"jsx": "react"

},

"exclude": [

"node\_modules"

]

}

* Run the following command to initialize the package.json file

*npm init*

**Step 1.2.2:** Setting up the TypeScript environment

You will be prompted to enter the package name, version number, description, entry point, test command, git repository, keywords, author, and license. You can set it to default by pressing [enter] 9 times.

You will be prompted to check if the configuration you suggested is correct. Press [enter] again.

* Run the following command to set up TypeScript:

*npm install --save-dev typescript*

*npm install --save-dev @types/node*

*npm install --save-dev @types/react*

* Add the following command to the package.json file:

"tsc:w": "tsc -w"

**Step 1.2.3:** Migrating JavaScript to TypeScript

* Create a folder **src** in the project’s root directory. We will add the following code to generate its equivalent JavaScript code. We are reverse engineering to understand how JavaScript code can run as TypeScript. TypeScript is a superset of JavaScript. Therefore, the program written in JavaScript should run in the TypeScript environment without changing the source code and just by changing .js to .ts.
* Create a file **serve.ts** in the **src** folder and add the following source code:

class Name {

first\_name: any;

last\_name: any;

constructor(fname: any, lname: any)

{

this.first\_name = fname;

this.last\_name = lname;

}

getName(): any

{

var fullname: any = this.first\_name + this.last\_name;

return fullname;

}

}

var author\_name: Name;

* Run the following command in the terminal to automatically detect any errors raised in the TypeScript file:

tsc -w

* This will generate two files, serve.js and serve.js.map.
* The JavaScript code should look as the source code mentioned below:

var Name = /\*\* @class \*/ (function () {

function Name(fname, lname) {

this.first\_name = fname;

this.last\_name = lname;

}

Name.prototype.getName = function () {

var fullname = this.first\_name + this.last\_name;

return fullname;

};

return Name;

}());

var author\_name;

//# sourceMappingURL=serve.js.map

* You can delete the serve.ts file and the terminal will prompt with error(s) due to absence of the .ts file in the project.
* You can rename **serve.js** to **serve.ts,** which will resolve the errors. You are running a JavaScript source code just by changing the .js file extension to .ts.

Note: You can confirm again by comparing the auto-generated JavaScript file with the renamed TypeScript file. The source code matches.

**Step 1.2.4:** Pushing the codes to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

Initialize your repository using the following command:

**git init**

Add all the files to your git repository using the following command:

**git add .**

Commit the changes using the following command:

**git commit . -m “Changes have been committed.”**

Push the files to the folder you initially created using the following command:

**git push -u origin master**