# From JavaScript to TypeScript

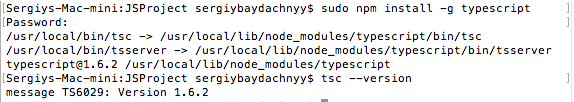
Today, JavaScript is so popular and it’s possible to find lots of projects that contain hundreds thousands lines of JavaScript code. Using JavaScript you can create web sites, Windows applications and server side services. But in case of really large projects it’s not easy to support the existing code. Compared to C#, C++ and Java, JavaScript lacks static typing, classes and interfaces that don’t allow common IDEs to provide richer environment.

That’s why many developers prefer to use supersets of JavaScript like TypeScript, CoffeeScript or Dart (not superset but supports compilation to JS).

I would recommend to use TypeScript which is “native” for Visual Studio Code and is supported by Visual Studio, Eclipse, WebStorm and so on. As I mentioned earlier, TypeScript is a superset of JavaScript that brings classes, generics, interfaces and allows to combine TypeScript code and JavaScript code together. Thanks to TypeScript compiler you can easily compile TypeScript code to JavaScript, so you can use it anywhere you can use JavaScript.

In order to start working with TypeScript you can visit http://www.typescriptlang.org and find there some tutorials and TypeScript compiler itself. So, if you want to install TypeScript compiler, you can use node.js package manager (npm) and use the following command:

**sudo npm install –g typescript**



Once the module is installed you can use tsc tool and try to check a version of TypeScript using:

**tsc --version**

If everything is OK, you can open Visual Studio Code and start working with TypeScript.

Pay special attention that we used node.js package manager. In order to have it you need to visit <https://nodejs.org/en/> site and install node.js from there.

From Visual Studio Code Editor perspective, you have access to the same features as in JavaScript – you can peek definitions, refactor code, use IntelliSense and so on. So, let’s start with configuring TypeScript project. First of all you need to create **tsconfig.json**. This file contains information related to output JavaScript code and it’s similar to **jsconfig.json**. So, the basic configuration file can look like:

{

"compilerOptions": {

"target": "ES5",

"module": "amd",

"sourceMap": true

}

}

It’s enough to start working with TypeScript code using all features of VS Code. But it’s not enough to compile your code to JavaScript. In order to do it we need to configure a task. We have discussed the tasks earlier and in case of TypeScript you can create the following **task.json**:

{

"version": "0.1.0",

"command": "tsc",

"isShellCommand": true,

"showOutput": "silent",

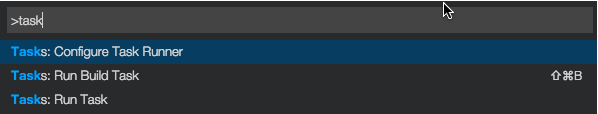
"args": ["index.ts"],

"problemMatcher": "$tsc"

}

You can see that the task calls tsc compiler and passes array of files as a parameter. Visual Studio Code supports build-in problem matcher that integrates all messages with Code interface.

If you still didn’t use tasks in Code, you can create a simple **task.json** and past all code there. Pay attention that **task.json** should be located in **.vscode** folder. So, in order to avoid any problems you can configure the folder structure using Command Palette and run Configure Task Runner:



To test how it works I propose to visit <http://www.typescriptlang.org> and use tutorial there – just copy the following code from the tutorial and create your first TypeScript file in Code:

class Student {

fullname : string;

constructor(public firstname, public middleinitial, public lastname) {

this.fullname = firstname + " " + middleinitial + " " + lastname;

}

}

interface Person {

firstname: string;

lastname: string;

}

function greeter(person : Person) {

return "Hello, " + person.firstname + " " + person.lastname;

}

var user = new Student("Jane", "M.", "User");

document.body.innerHTML = greeter(user);

Once you prepared the code, you can start the task clicking ⇧⌘B or use Command Palette. As a result you can find output JavaScript file.

Finally, create HTML file to test the JavaScript code:

<!DOCTYPE html>

<html>

<head><title>TypeScript Greeter</title></head>

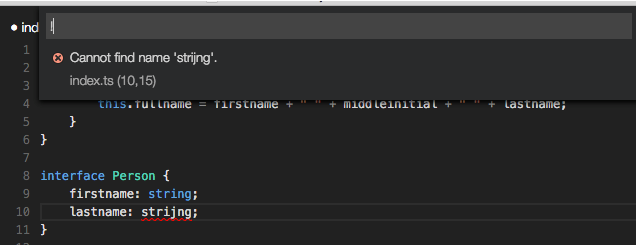
<body>

<script src="index.js"></script>

</body>

</html>

At the end I would recommend to spend some time with the Editor to check all IntelliSense features:

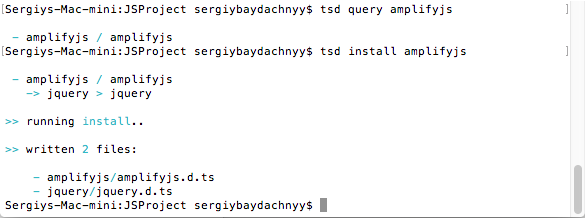


Ok, we know how to create and compile TypeScript files but what about popular libraries like jquery, angular and so on. In order to use these libraries in native for TypeScript way you can create a special TypeScript definition files that provide metadata for JavaScript code. Of course, for popular libraries you should not create these files yourself. All that you need is to install TypeScript Definition manager using the following command:

sudo npm install -g tsd

Once the manager is installed you can run tsd tool in context of your working folder. For example **tsd query** allows to check availability of definition files based on search term and **tsd install** allows to install definition packages:

tsd install amplifyjs



Visual Studio Code can automatically identify all **.d.ts** files and include them to IntelliSense.