Setting up VSCode for TypeScript

In this section we are going to set up Visual Studio Code for TypeScript. The project files for this chapter are in `angular2-intro/project-files/vscode-demo`. You can either follow along or check out the folder to see the final result.

Installing TypeScript

Before anything, we need to install the TypeScript compiler. You can install the TypeScript compiler with npm:

```
npm i typescript -g
```

Then to verify that it is installed, run `tsc -v` to see the version of the compiler. You will get an output like this:

```
message TS6029: Version 1.7.5
```

In addition to the compiler, we also need to install the TypeScript Definition manager for DefinitelyTyped (tsd). You can install tsd with:

```
npm i tsd -g
```

Using TSD, you can search and install TypeScript definition files directly from the community driven DefinitelyTyped repository. To verify that tsd is installed, run tsd with the `version` flag:

```
tsd --version
```

You should get an output like this:

```
>> tsd 0.6.5
```

After `tsd` and `tsc` are installed, we can compile a hello world program:

make a file called `hello.ts` on your desktop:

```
touch ~/Desktop/hello.ts
```

Then, put some TypeScript code in the file:

```
echo "const adder = (a: number, b: number): number => a + b;" > ~/De
```

Then you can compile the file to JavaScript:

```
tsc ~/Desktop/hello.ts
```

It should output a file in `Desktop/hello.js` :

```
var adder = function (a, b) { return a + b; };
```

Now that your TypeScript compiler setup, we can move on to configuring Visual Studio Code.

Add VSCode Configurations

- First download and install Visual Studio Code from the VSCode Website
- After installing VSCode, open it and then make a new window: `File > New Window`
- Then, make a folder on your desktop for a new project: `mkdir ~/Desktop/vscode-demo`
- After that, open the folder in VSCode: `File > open` and select the `vscode-demo` folder on your desktop.

- Now we need to make three configuration files:

    1. `tsconfig.json` : configuration for the TypeScript compiler
    2. `tasks.json` : Task configuration for VSCode to watch and compile files
    3. `launch.json` : Configuration for the debugger

The `tsconfig.json` file should be in the root of the project. Let's make the file and put the following in it:

```
{
  "compilerOptions": {
    "experimentalDecorators": true,
    "emitDecoratorMetadata": true,
    "module": "commonjs",
    "target": "es5",
    "sourceMap": true,
    "outDir": "output",
    "watch": true
  }
}
```

Now to make the `tasks.json` file. Open the prompt with `command + shift + p` and type:

```
> configure task runner
```

Then put the following in the file and save the file:

```
{
  "version": "0.1.0",
  "command": "tsc",
  "showOutput": "silent",
  "isShellCommand": true,
  "problemMatcher": "$tsc"
}
```

The last thing that we need to set up is the debugger, i.e. the `launch.json` file. Right click on the `.vscode` folder in the file navigator and make a new file called `launch.json` and put in the following:

```
{
  "version": "0.1.0",
  "configurations": [
    {
      "name": "TS Debugger",
      "type": "node",
      "program": "main.ts",
      "stopOnEntry": false,
      "sourceMaps": true,
      "outDir": "output"
    }
  ]
}
```

After you save the file, you should be able to see the debugger in the debugger dropdown options.

Now, we are ready to make the `main.ts` file in the root of the project:

`main.ts`

```
const sum = (a: number, b: number): number => a + b;
const r = sum(1,2);
console.log(r);
```

Now you can start the task to watch the files and compile as you work. Open the prompt with `command + shift + p` and type:

```
> run build tasks
```

you can also use the `command + shift + b` keyboard shortcut instead. This will start the debugger and watch the files. After making a change to `main.ts`, you should be able to see the output in the `output` folder.

After the build task is running, we can put a breakpoint anywhere in our TypeScript code. Let's put a breakpoint on line 2 by clicking on the margin. Then start the debugger by going to the debugger tab and clicking the green play icon.

Now you should see that the program will stop at the breakpoint and you should be able to step over or into your program.

To stop the task you can terminate it. Open the prompt and type:

```
> terminate running task
```

Running VSCode from the Terminal

If you want to run VSCode from the terminal, you can follow the guide on VSCode's website. Below is the summary of the guide:

MAC

Add the following to your "bash" file:

```
function code () { VSCODE_CWD="$PWD" open -n -b "com.microsoft.VSCode" --c
```

Linux

```
sudo ln -s /path/to/vscode/Code /usr/local/bin/code
```

Windows

You might need to log off after the installation for the change to the PATH environmental variable to take effect.

Debugging App from VSCode

The "vscode-chrome-debug" extension allows you to attach VSCode to a running instance of chrome. This makes it very convenient because you can put breakpoints in your TypeScript code and run the debugger to debug your app. Let's get started.

In order to install the extension open the prompt in VSCode with `command + shift + p` and type:

```
> install extension
```

hit enter and then type:

```
debugger for chrome
```

Then just click on the result to install the extension. Restart VSCode when you are prompted.

After installing the extension, we need to update or create a `launch.json` file for debugging. You can create one in the `.vscode` folder. After you created the file, put in the following:

```
{
  "version": "0.1.0",
  "configurations": [
    {
      "name": "Launch Chrome Debugger",
      "type": "chrome",
      "request": "launch",
      "url": "http://localhost:8080",
      "sourceMaps": true,
      "webRoot": ".",
      "runtimeExecutable": "/Applications/Google Chrome.app/Contents/MacOS
      "runtimeArgs": ["--remote-debugging-port=9222", "--incognito"]
    }
  ]
}
```

Notes:

- Depending on your platform you need to change the `runtimeExecutable` path to Chrome's executable path. After configuring the debugger you need to have a server running serving the app. You can change the `url` value accordingly. Also make sure that the `webRoot` path is set to the root of your web server.
- After that it is a good idea to close all the instances of chrome. Then, put a breakpoint in your code and run the debugger. If everything is set up correctly, you should see an instance of chrome running in incognito mode. To trigger the breakpoint, just reload the page and you should be able to see the debugger paused at the breakpoint.

- Also make sure that you have the compiler running so that you can use the JavaScript output and the sourcemaps to use the debugger. See the TypeScript and VSCode set up for more details.