JavaScript Development Environments

## JavaScript

### JavaScript and React (CRA)

#### Run all tests with Filewatch

If you create you react app using npx create-react-app my-react-app then this all tests with filewatch is the default for the npm test script. From the terminal just enter the following command

npm test

#### Debug All Tests

Add the following code to your launch.config

{

"name": "Debug CRA Tests",

"type": "node",

"request": "launch",

"runtimeExecutable": "${workspaceRoot}/node\_modules/.bin/react-scripts",

"args": ["test", "--runInBand", "--no-cache", "--watchAll=false"],

"cwd": "${workspaceRoot}",

"protocol": "inspector",

"console": "integratedTerminal",

"internalConsoleOptions": "neverOpen",

"env": { "CI": "true" },

"disableOptimisticBPs": true

}

#### Debug Single Test File

## Typescript

### Setup

#### JavaScript/Typescript Project Structure

|  |  |
| --- | --- |
| Command | Details |
| package.json | Describes a project’s top-level dependencies. These are packages that have been added to a project using npm install |
| package-lock.json | All package dependencies for the project |
| tsconfig.json | TypeScript compiler configuration |

#### Node Packages

|  |  |
| --- | --- |
| Package | Description |
| typescript | The typescript compiler |
| jest | JavaScript testing framework |
| tsc-watch | Run a script specified in package.json |
| ts-jest | Enable us to use Jest with TypeScript |

#### TypeScript Compiler Options (TSCONFIG.JSON)

|  |  |
| --- | --- |
| Package | Description |
| target | The version of JavaScript to transpile to |
| module | The JavaScript module format. Some environments such as node do not support ES2015 modules so specifying commonjs tells the compiler to generate older module code |
| lib | Tell TypeScript that certain APIs will be available in the environment in which the code will run. Examples such as DOM’s document.querySelector |
| outDir | The directory into which we will put the transpiled JavaScript. |
| include | The folders to look in for TypeScript files |

The following listing shows an example tsconfig.json file.

Listing 1 tsconfig.json

{

    "compilerOptions": {

        "target": "ES2018",

        "outDir": "./dist",

        "noEmitOnError": true,

        "sourceMap": true,

        "module": "commonjs"

    }

}

### Development Template

This section shows how to setup a TypeScript development environment with the following benefits.

* Automatically transpile TypeScript files as they change.
* Unit testing with Jest.
* Debugging individual unit tests with Jest Visual Studio Code Plugin.

#### Package.json

{

  "name": "tools",

  "version": "1.0.0",

  "description": "",

  "main": "index.js",

  "scripts": {

**"test": "npx jest --watchAll",**

**"start": "tsc-watch --onsuccess \" node dist/index.js\""**

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "devDependencies": {

    "tsc-watch": "^4.2.3",

    "typescript": "^3.8.3"

  }

}

The bold lines specify scripts that can be run by npm. We have added a script called start that monitors files for change and executes the index.js when changed files have been compiled

## Typescript – Review to delete

### Debugging

If we want to debug in VSCode we need to add a folder called .vscode into which we add a file called launch.json

{

    // Use IntelliSense to learn about possible attributes.

    // Hover to view descriptions of existing attributes.

    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387

    "version": "0.2.0",

    "configurations": [

        {

            "type": "node",

            "request": "launch",

            "name": "Launch Program",

            "program": "${workspaceFolder}\\dist\\index.js",

        }

    ]

}

We can then run our debugger using F5 in visual studio code

### Unit Testing

Unit testing with Jest consists of two parts. The first part is to setup a configuration file called jest.config.js at the root level of our project. The following is a good example.

module.exports = {

    "roots": ["src"],

    "transform":{"^.+\\.tsx?$": "ts-jest"}

}

Then we simply add tests in our source code folder. If we have a module called adder.ts as follows

export function add(a: number, b: number): number {

    return a+b;

}

We can create a test called adder.test.ts as follows

import {add} from "./adder";

test("do a test", () => {

    let result = add(10,5);

    expect(result).toBe(15);

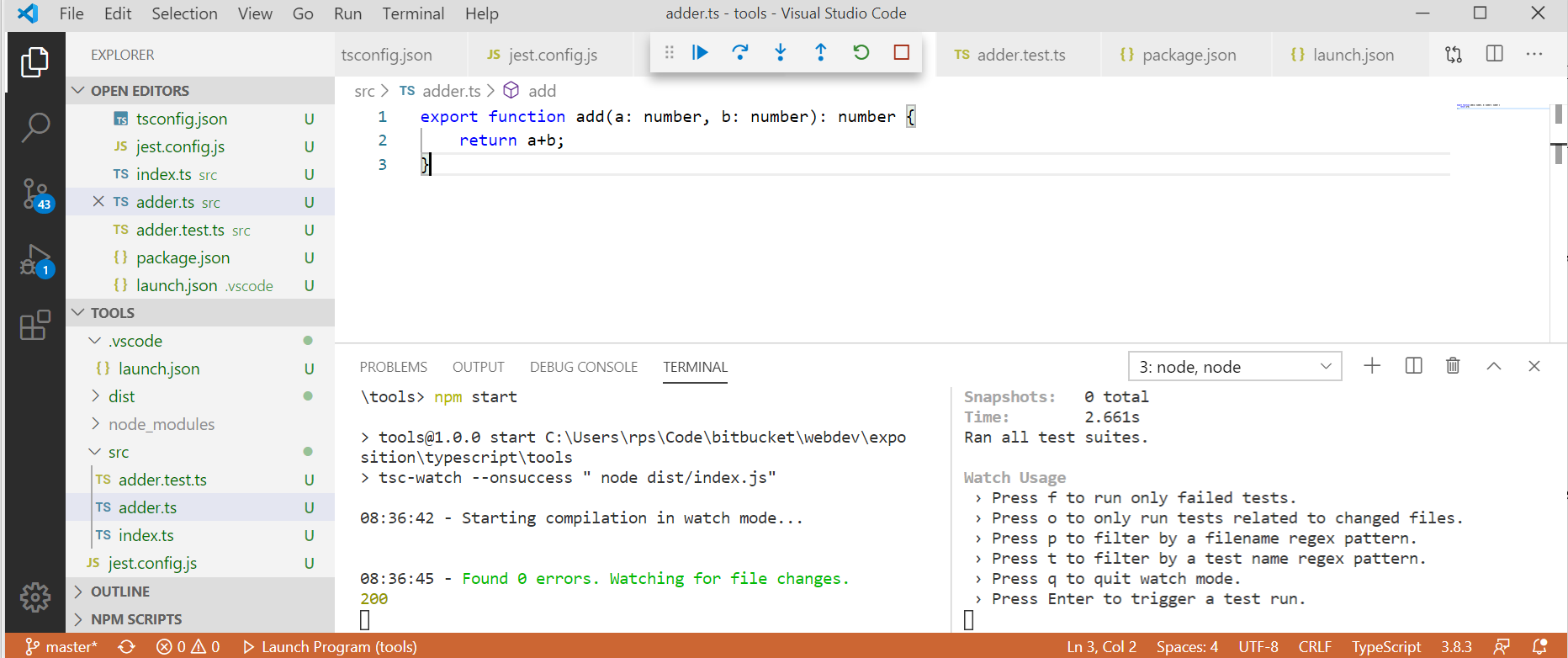
})

### Putting it together

Often it is useful to have two terminal windows: one with a file watcher compiling and running our application and one running the tests.

npm start

npm test



### Specified Single File

We need to make sure we have the typescript compiler installed.

npm init –yes

npm install –-save-dev typescript

We need to create a typescript compiler configuration file

Listing 2 tsconfig.json

{

    "compilerOptions": {

        "target": "ES2018", ➋

        "outDir": "./dist",

        "rootDir": "./src",

        "noEmitOnError": true,

        "sourceMap": true,

        "module": "commonjs" ➊

    }

}

|  |  |
| --- | --- |
| ➊ module format | Some environments such as node do not support ES2015 modules so specifying commonjs tells the compiler to generate older module code |
| ➋ target | The version of JavaScript to target |

#### Compile

When we run tsc from the command line with no arguments it will compile TypeScript source files in the rootDir to JavaScript files in the outDir

npx tsc

#### Run specified single File

We run JavaScript and not TypeScript, so the command is then

node dist/hello.js

#### Run specified single File with watch

To run typescript in watch mode we need an extra package called tsc-watch

npm install –save-dev tsc-watch

We then need to add a line to the scripts section in our package.json

"scripts": {

"test": "npx jest --watchAll",

**"start": "tsc-watch --onsuccess \" node dist/hello.js\""**

  },

### Selected File

#### Run/Debug Specified File No Watch

Setup the lauch.json with a configuration as follows.

{

    "version": "0.2.0",

    "configurations": [

        {

            "type": "node",

            "request": "launch",

            "name": "Run/Debug Open File",

            "skipFiles": [

                "<node\_internals>/\*\*"

            ],

            "program": "${file}",

            "preLaunchTask": "tsc: build - tsconfig.json",

            "outFiles": [

                "${workspaceFolder}/\*\*/\*.js"

            ]

        }

    ]

}

You can then run/debug the current file using Ctrl-F5 or F5

### Jest

#### run all tests no watch

To use jest with typescript we need the following

npm install --save-dev jest

npm install --save-dev @types/jest

npm install --save-dev @babel/preset-typescript

We also need a file called babel.config.js

module.exports = {

    presets: [

      ['@babel/preset-env', {targets: {node: 'current'}}],

  +    '@babel/preset-typescript',

    ],

  };

Finally, we run the tests as follows in the terminal

npx jest

#### run all tests With watch

npx jest --watchAll

#### Run Single File Test No Watch

Run the test in hello2.test.ts Note we miss off the .ts from the filename

npx jest hello2.test

#### Run Single File Test Watch

npx jest hello2.test--watch

#### Debug/Run Single Test File No Watch

Add the following configuration to launch.json

        {

            "name": "Run/Debug Open Test",

            "type": "node",

            "request": "launch",

            "runtimeExecutable": "${workspaceRoot}/node\_modules/.bin/jest.cmd",

            "args": [

                "--runInBand",

                "--watchAll=false",

                "${fileBasenameNoExtension}"

            ],

            "cwd" : "${workspaceFolder}",

            "protocol": "inspector",

            "console": "integratedTerminal",

            "internalConsoleOptions": "neverOpen"

        }

#### Debug/Run Single Test File With Watch

Add the following configuration to launch.json

        {

            "name": "Run/Debug Open Test",

            "type": "node",

            "request": "launch",

            "runtimeExecutable": "${workspaceRoot}/node\_modules/.bin/jest.cmd",

            "args": [

                "--runInBand",

                "--watchAll=true",

                "${fileBasenameNoExtension}"

            ],

            "cwd" : "${workspaceFolder}",

            "protocol": "inspector",

            "console": "integratedTerminal",

            "internalConsoleOptions": "neverOpen"

        }

#### CodeCoverage

npx jest --coverage

### React

The following sections assume the project was setup using

npx create-react-app my-app --template typescript

#### run all tests no watch

To use jest with typescript we need the following

npm install --save-dev jest

npm install --save-dev @types/jest

npm install --save-dev @babel/preset-typescript

We also need a file called babel.config.js

module.exports = {

    presets: [

      ['@babel/preset-env', {targets: {node: 'current'}}],

  +    '@babel/preset-typescript',

    ],

  };

Finally, we run the tests as follows in the terminal

npx jest

#### run all tests With watch

npx jest --watchAll

#### Run Single File Test No Watch

Run the test in hello2.test.ts Note we miss off the .ts from the filename

npx jest hello2.test

#### Run Single File Test Watch

npx jest hello2.test--watch

#### Debug/Run Single Test File No Watch

Add the following configuration to launch.json

        {

            "name": "Run/Debug Open Test",

            "type": "node",

            "request": "launch",

            "runtimeExecutable": "${workspaceRoot}/node\_modules/.bin/jest.cmd",

            "args": [

                "--runInBand",

                "--watchAll=false",

                "${fileBasenameNoExtension}"

            ],

            "cwd" : "${workspaceFolder}",

            "protocol": "inspector",

            "console": "integratedTerminal",

            "internalConsoleOptions": "neverOpen"

        }

#### Debug/Run Single Test File With Watch

Add the following configuration to launch.json

        {

            "name": "Run/Debug Open Test",

            "type": "node",

            "request": "launch",

            "runtimeExecutable": "${workspaceRoot}/node\_modules/.bin/jest.cmd",

            "args": [

                "--runInBand",

                "--watchAll=true",

                "${fileBasenameNoExtension}"

            ],

            "cwd" : "${workspaceFolder}",

            "protocol": "inspector",

            "console": "integratedTerminal",

            "internalConsoleOptions": "neverOpen"

        }

#### Code Coverage

npx jest --coverage

### React

This section assumes react with typescript was installed with

npx create-react-app my-app --template typescript

#### Run all tests with watch

Simply run the following from the VSCode terminal

npm run tests

#### Run Single test with watch

If the test file is called maths.test.js we enter the command on the terminal

npm run test maths.test

#### Debug single test