RxJS Installation or Environment Setup

There are different ways to install RxJS but before installing RxJS, we need the following setup to work efficiently with RxJS.

You have to install-

* js
* Npm
* RxJS package installation

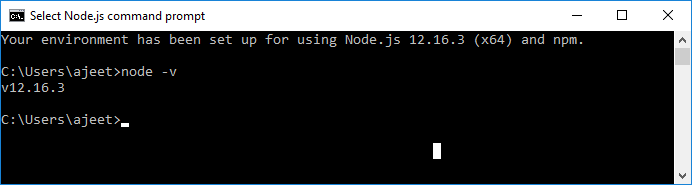
Installing Node.js and npm

If you are already familiar with [Node.js](https://www.javatpoint.com/nodejs-tutorial) then you must have installed Node.js on your computer system. You should check whether you have installed Node.js on your system or not. In the case if you have not installed Node.js then it is very easy to install [RxJS](https://www.javatpoint.com/rxjs) using npm.

First, let's verify if Node.js and npm are installed on your system. Open your Node.js command prompt and execute the following command.

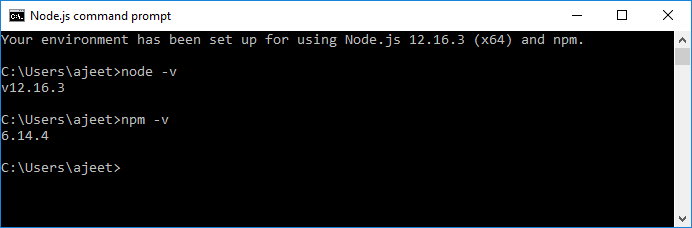
**Run the following command to check Node's version:**

1. node -v



Run the following command to check npm version:

1. npm -v



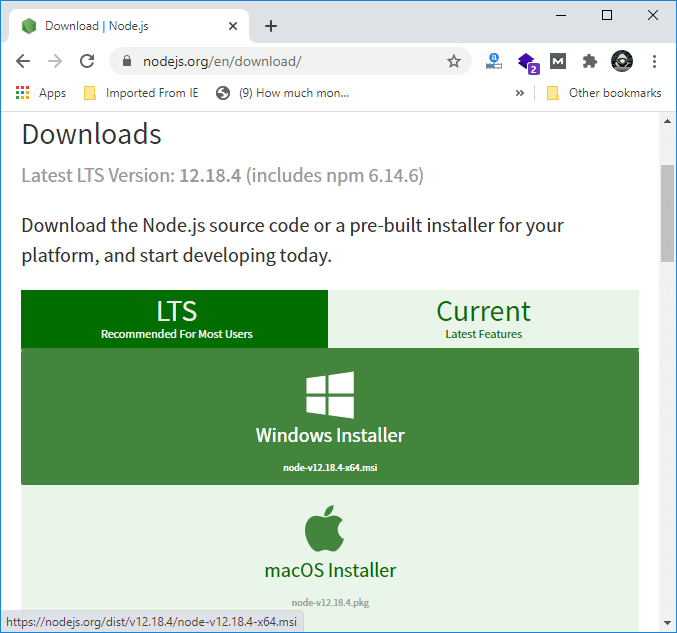
After running the above commands, you can see that we have got a result. It means the Node.js and npm are installed on our system and the version is 12.16.3 and 6.14.4 respectively. So, we can proceed further.

If your command prompt does not print anything, it means Node.js and npm are not installed on your system

How to install Node.js?

Installing Node.js is not a big task. To install Node.js, first go to the download page of the official website of the Node.js <https://nodejs.org/en/download/> and then install the package according to your operating system.

You will see that the download page of Node.js would look like this:



Install the required package according to your operating system. After installing the Node.js, npm will also be installed along with it. Now, check if npm is installed or not by typing npm -v command in the terminal and you will see the version of the npm.

You can read in details: How to install Node.js? <https://www.javatpoint.com/install-nodejs>

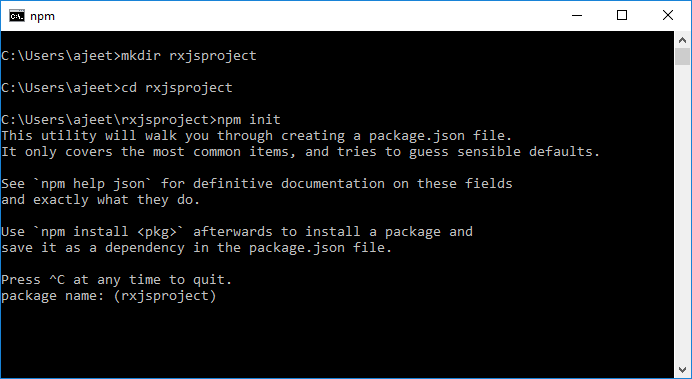
Install RxJS Package

Before going to start RxJS installation, we have to first create a folder named "rxjsproject" where we will practice all RxJS examples.

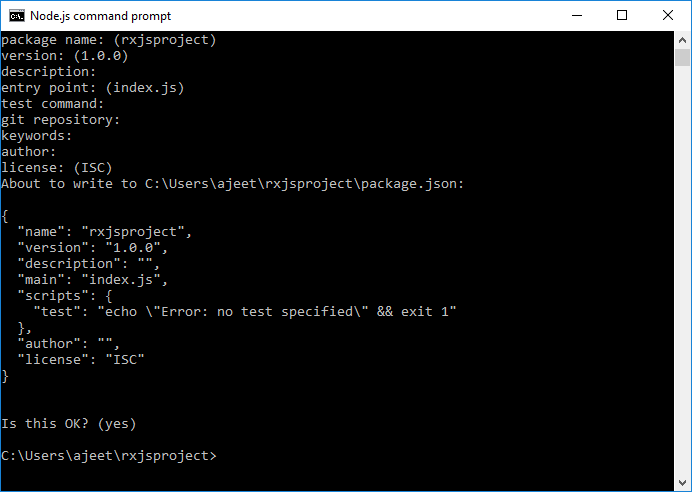
After creating the folder rxjsproject, we have to run the command npm init, to setup the project.

**Execute the command given below:**

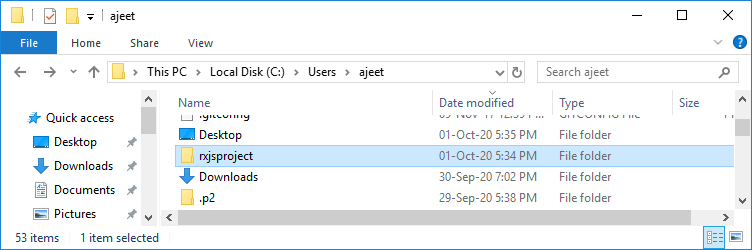
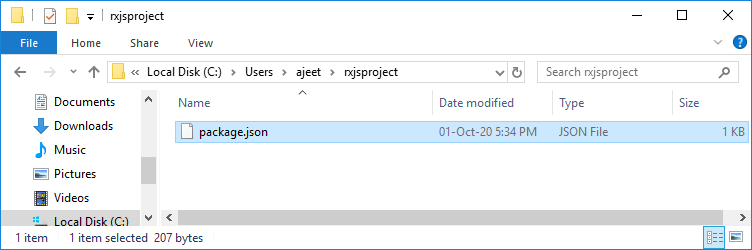
1. >mkdir rxjsproject
2. >cd rxjsproject
3. \rxjsproject>npm init



When you run the **npm init** command, it will ask few questions during execution. You have to just press enter and proceed.



When the execution of the **npm init** is completed, it will create a file **package.json** inside **rxjsproject/** as following:

The File **package.json** contains the following code:

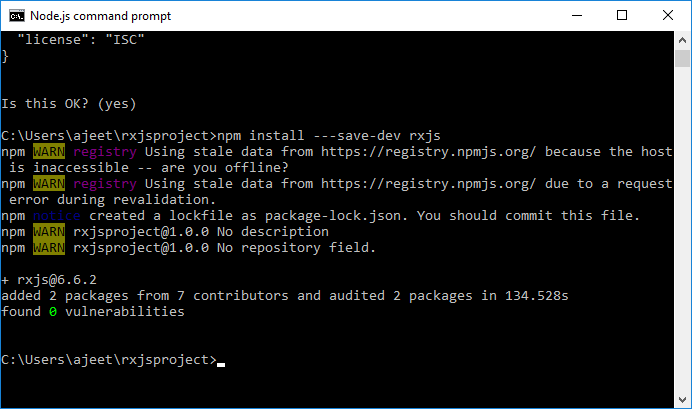
1. {
2. "name": "rxjsproject",
3. "version": "1.0.0",
4. "description": "",
5. "main": "index.js",
6. "scripts": {
7. "test": "echo \"Error: no test specified\" && exit 1"
8. },
9. "author": "",
10. "license": "ISC"
11. }

Install RxJS

Now, all the prerequisites are done and it's time to install RxJS. Use the following command to install RxJS:

1. npm install ---save-dev rxjs

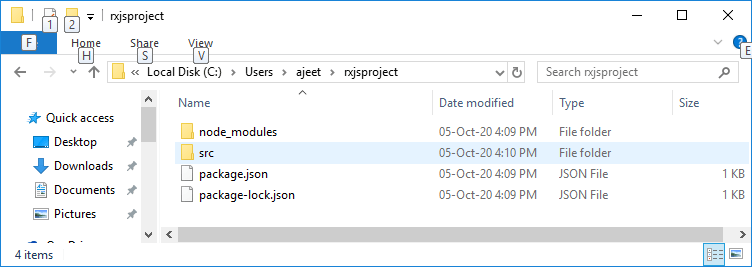
After running the installation code, you will see that the npm has created a lockfile as **package-lock.json.** You should commit this file.



In the above image, you can see that the installation of RxJS is completed.

Now, let us see how to use RxJS.

First, create a folder **src/** inside the **rxjsproject/**. See the following image:



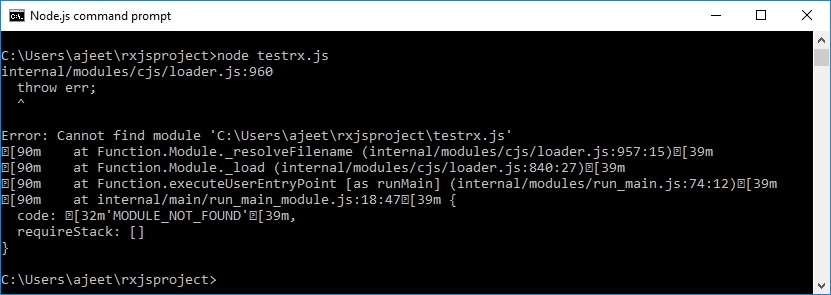
Now, you will see that the folder structure would look like this:

1. rxjsproject/
2. node\_modules/
3. src/
4. **package**.json

Now, create a file **testrx.js** inside the **src/ folder** and write the following code:

1. **import** { of } from 'rxjs';
2. **import** { map } from 'rxjs/operators';
3. map(x => x \* x)(of(1, 2, 3, 4, 5)).subscribe((v) => console.log(`Output is: ${v}`));

If you execute the above code in command prompt using the command- **node testrx.js**, it will display an error for import. The error occurs because Node.js does not know what to do with import.

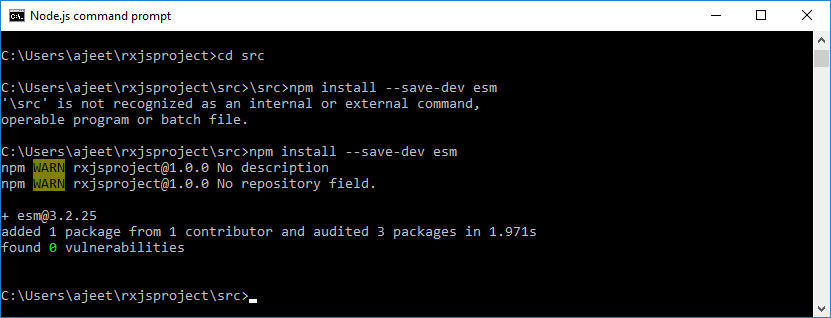


To resolve this issue, we have to make it possible that it can import with Node.js.

Install ES6 Modules Package

For this purpose, we have to install [ES6](https://www.javatpoint.com/es6) modules package using npm. Use the following command to install ES6 module package.

1. \src>npm install --save-dev esm



Now, the package has been installed and you can execute the **testrx.js file** by using the following command:

1. \src>node -r esm testrx.js

**Output:**

Output is: 1

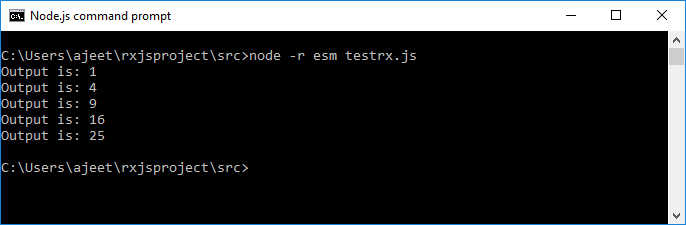
Output is: 4

Output is: 9

Output is: 16

Output is: 25

See the following image:



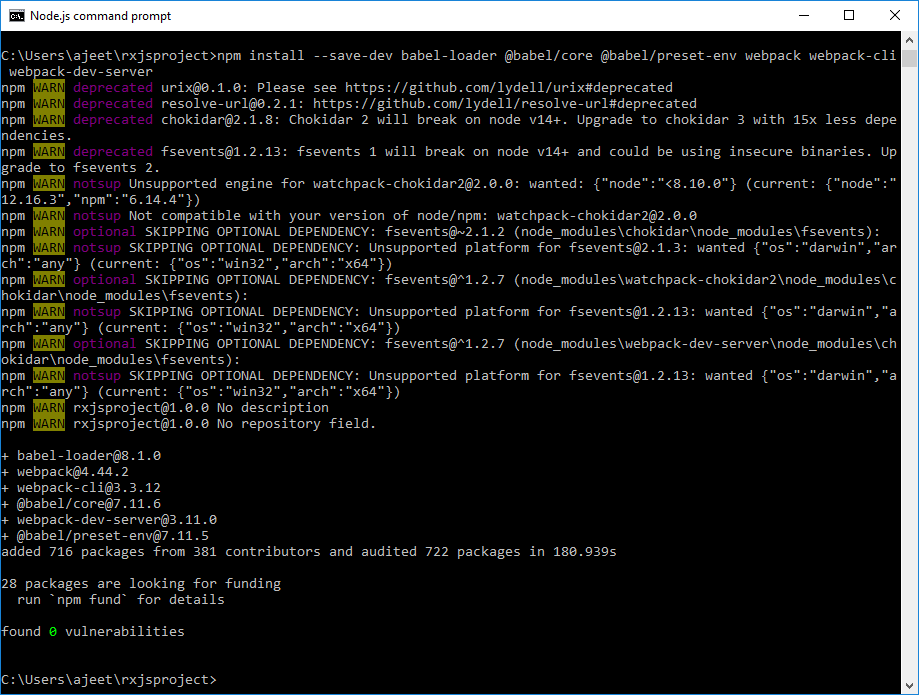
In the above image, you can see the output on the console. You can see that RxJS is installed and ready to use but the above method will help us to test RxJS in the command line only. If you want to test RxJS in the browser, you have to install some additional packages.

Test RxJS in Browser

We have to install the following packages inside rxjsproj/ folder to test RxJS in browser.

**Run the following npm command:**

1. npm install --save-dev babel-loader @babel/core @babel/preset-env webpack webpack-cli webpack-dev-server
2. >npm install --save-dev babel-loader @babel/core @babel/preset-env webpack webpack-cli webpack-dev-server
3. npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
4. npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
5. npm WARN deprecated chokidar@2.1.8: Chokidar 2 will **break** on node v14+. Upgrade to chokidar 3 with 15x less dependencies.
6. npm WARN deprecated fsevents@1.2.13: fsevents 1 will **break** on node v14+ and could be using insecure binaries. Upgrade to fsevents 2.
7. npm WARN notsup Unsupported engine **for** watchpack-chokidar2@2.0.0: wanted: {"node":"<8.10.0"} (current: {"node":"12.16.3","npm":"6.14.4"})
8. npm WARN notsup Not compatible with your version of node/npm: watchpack-chokidar2@2.0.0
9. npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.1.2 (node\_modules\chokidar\node\_modules\fsevents):
10. npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform **for** fsevents@2.1.3: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
11. npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@^1.2.7 (node\_modules\watchpack-chokidar2\node\_modules\chokidar\node\_modules\fsevents):
12. npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform **for** fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
13. npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@^1.2.7 (node\_modules\webpack-dev-server\node\_modules\chokidar\node\_modules\fsevents):
14. npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform **for** fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
15. npm WARN rxjsproject@1.0.0 No description
16. npm WARN rxjsproject@1.0.0 No repository field.
18. + babel-loader@8.1.0
19. + webpack@4.44.2
20. + webpack-cli@3.3.12
21. + @babel/core@7.11.6
22. + webpack-dev-server@3.11.0
23. + @babel/preset-env@7.11.5
24. added 716 packages from 381 contributors and audited 722 packages in 180.939s
26. 28 packages are looking **for** funding
27. run `npm fund` **for** details
29. found 0 vulnerabilities



Here, we have installed webpack-server to start the server and execute our Html file. The command "publish" in package.json is used to start as well as pack all the js files using webpack server.

To use webpack, we have to run **npm run publish** command. You can see that the publish command is added in package.json as following:

Update the package.json file with the following code:

**Package.json:**

1. {
2. "name": "rxjsproject",
3. "version": "1.0.0",
4. "description": "",
5. "main": "index.js",
6. "scripts": {
7. "publish":"webpack && webpack-dev-server --output-public=/dev/",
8. "test": "echo \"Error: no test specified\" && exit 1"
9. },
10. "author": "",
11. "license": "ISC",
12. "devDependencies": {
13. "@babel/core": "^7.11.6",
14. "@babel/preset-env": "^7.11.5",
15. "babel-loader": "^8.1.0",
16. "esm": "^3.2.25",
17. "rxjs": "^6.6.3",
18. "webpack": "^4.44.2",
19. "webpack-cli": "^3.3.12",
20. "webpack-dev-server": "^3.11.0"
21. }
22. }

To work with webpack, create a file called **webpack.config.js** that has the following code:

1. var path = require('path');
2. module.exports = {
3. entry: {
4. app: './src/testrx.js'
5. },
6. output: {
7. path: path.resolve(\_\_dirname, 'dev'),
8. filename: 'main\_bundle.js'
9. },
10. mode:'development',
11. module: {
12. rules: [
13. {
14. test:/\.(js)$/,
15. include: path.resolve(\_\_dirname, 'src'),
16. loader: 'babel-loader',
17. query: {
18. presets: ['@babel/preset-env']
19. }
20. }
21. ]
22. }
23. };

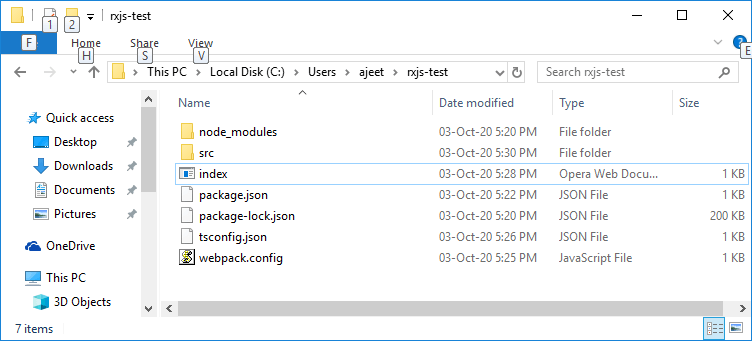
Now, create an index.html file within the root project. This file will contain what you want to show on your browser.

**Index.html:**

1. <!DOCTYPE html**>**
2. **<html** lang="en"**>**
3. **<head>**
4. **<style>**
5. body { font-family: 'Arial'; background: lightpink }
6. **</style>**
7. **</head>**
8. **<body>**
9. **<h2>**Hello World!**</h2>**
10. **<p>**You are Welcome**<p>**
11. **</body>**
12. **</html>**

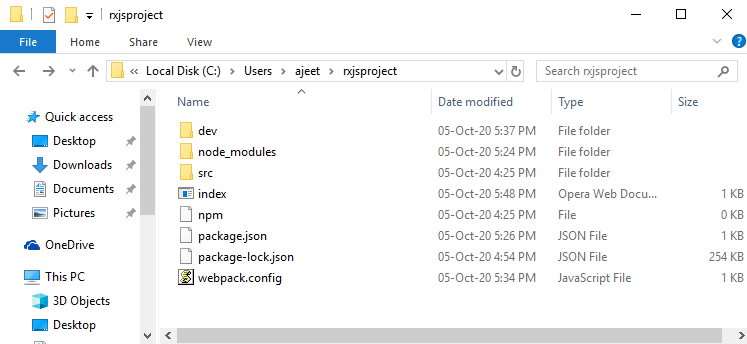
The final folder structure would look like following:

1. rxjsproject/
2. node\_modules/
3. src/
4. testrx.js
5. index.html
6. package.json
7. webpack.config.js

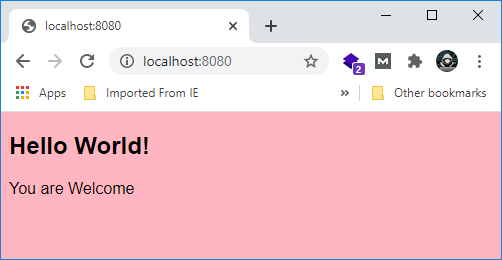


Run the Project

Now, execute the run command **npm run publish**. It will create a folder named dev with main\_bundle.js file in it. The server will be started and you can test your index.html in the browser.



Open the browser and goto the local host url: http://localhost:8080/ and you will see the output:



# RxJS First Example

Let's get started with the RxJS tutorial with the first example of RxJS. Here, we will see how to set up the development environment and get started with the concept of observables and create the first example of [RxJS](https://www.javatpoint.com/rxjs).

### Set up and install the dependencies

To get started with our project, first, we have to create a new project folder by using the following command:

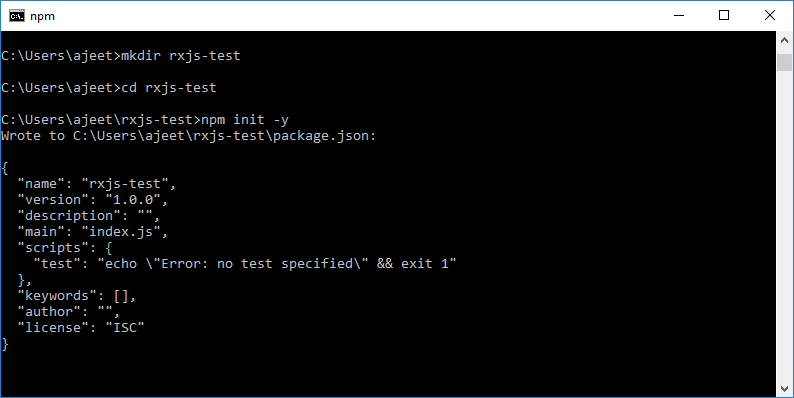
1. mkdir rxjs-test

Now, go into the newly created folder by executing the following command:

1. cd rxjs-test

Now, we will create a new package.json file inside of our project folder by using the following npm (Node.js Package Manager) command:

1. npm init -y



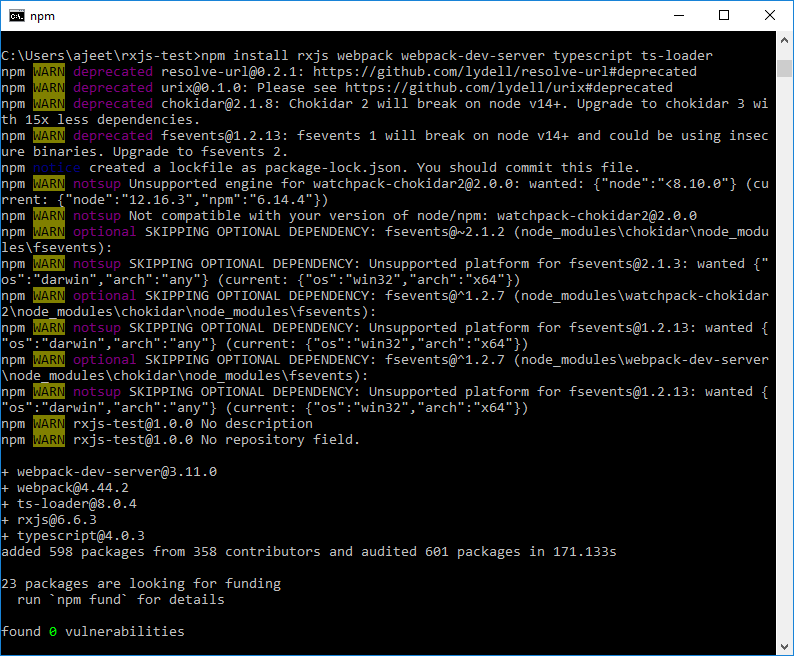
In the above image, you can see that the package.json file has been created. You can also see the default code in that image.

#### Note: If you have not installed Node.js, then you can easily install it by using npm. First, to install Node.js, go to the download page of the official website of the Node.js https://nodejs.org/en/download/ and then install the package according to your operating system.

### Install Webpack, TypeScript, and TypeScript loader for Webpack

Here, we have to add and install a few dependencies such as Webpack, TypeScript, the corresponding TypeScript loader for Webpack, and the Webpack development web server by using NPM again by using the following command:

1. npm install rxjs webpack webpack-dev-server typescript ts-loader



**What is Webpack?**

Webpack is a module bundler for modern [JavaScript](https://www.javatpoint.com/javascript-tutorial) applications. It is used to process our application. When the Webpack processes our application, it internally builds a dependency graph that maps every module our project needs and generates one or more bundles.

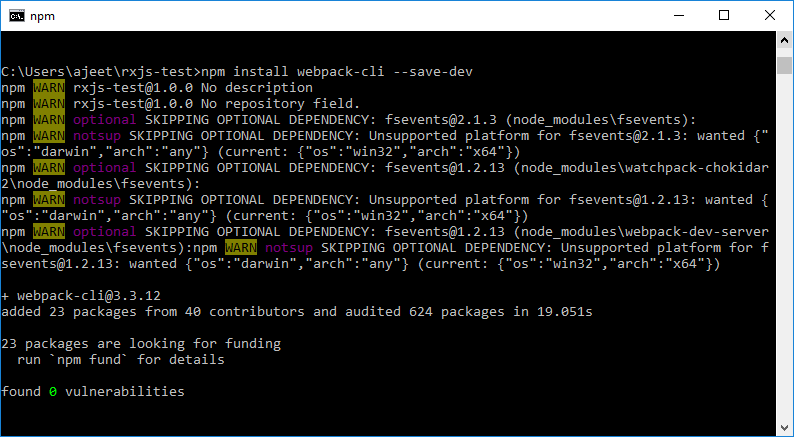
**What is TypeScript?**

[TypeScript](https://www.javatpoint.com/typescript-tutorial) is an open-source programming language, developed and maintained by Microsoft. It is used to add optional static typing to JavaScript. It is a strict syntactical superset of JavaScript language.

### Install the Webpack CLI

We must ensure that we have installed the Webpack CLI (Command Line Interface) as development dependencies. Install the Webpack CLI by using the following command:

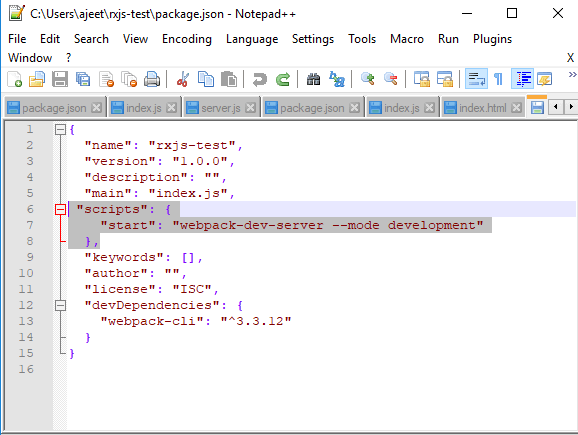
1. npm install webpack-cli --save-dev



### Add Script to Package.JSON File

Add the following script with name start to the scripts section in package.json file:

1. "scripts": {
2. "start": "webpack-dev-server --mode development"
3. },



This script is used to execute the webpack-dev-server command in development mode when we start up the Webpack development web server when the project is completed.

### Set Up Webpack

Now, we have to add the Webpack configuration to our project. So, create a new file named webpack.config.js in the root project folder and use the following configuration code:

1. **const** path = require('path');
2. module.exports = {
3. entry: './src/index.ts',
4. devtool: 'inline-source-map',
5. module: {
6. rules: [
7. {
8. test: /\.tsx?$/,
9. use: 'ts-loader',
10. exclude: /node\_modules/
11. }
12. ]
13. },
14. resolve: {
15. extensions: [ '.tsx', '.ts', '.js' ]
16. },
17. output: {
18. filename: 'bundle.js',
19. path: path.resolve(\_\_dirname, 'dist')
20. }
21. };

### Configure TypeScript

We also have to add the TypeScript compiler configuration in the project. to do so create a new file named tsconfig.json and use the following [JSON](https://www.javatpoint.com/json-tutorial) code which contains configuration properties for the TypeScript compiler:

1. {
2. "compilerOptions": {
3. "outDir": "./dist/",
4. "sourceMap": **true**,
5. "noImplicitAny": **true**,
6. "module": "es6",
7. "moduleResolution": "node",
8. "target": "es6",
9. "allowJs": **true**,
10. "lib": [
11. "es2017",
12. "dom"
13. ]
14. }
15. }

### Create the home page Index.html File

Now, create the home page named index.html file for the browser inside the project folder:

**Index.html file:**

1. <!DOCTYPE html**>**
2. **<html** lang="en"**>**
3. **<head>**
4. **<meta** charset="UTF-8"**>**
5. **<meta** name="viewport" content="width=device-width, initial-scale=1.0"**>**
6. **<meta** http-equiv="X-UA-Compatible" content="ie=edge"**>**
7. **<title>**RxJS Demo**</title>**
8. **<style>**
9. body { font-family: 'Arial'; background: lightgray }
10. ul { list-style-type: none; padding: 20px; }
11. li { padding: 15px; background: lightcoral; margin-bottom: 10px; }
12. **</style>**
13. **</head>**
14. **<body>**
15. **<h1>**RxJS Demo**</h1>**
16. **<div>**
17. **<ul** id="list"**></ul>**
18. **</div>**
19. **<script** src="/bundle.js"**></script>**
20. **</body>**
21. **</html>**

### Add the TypeScript Code

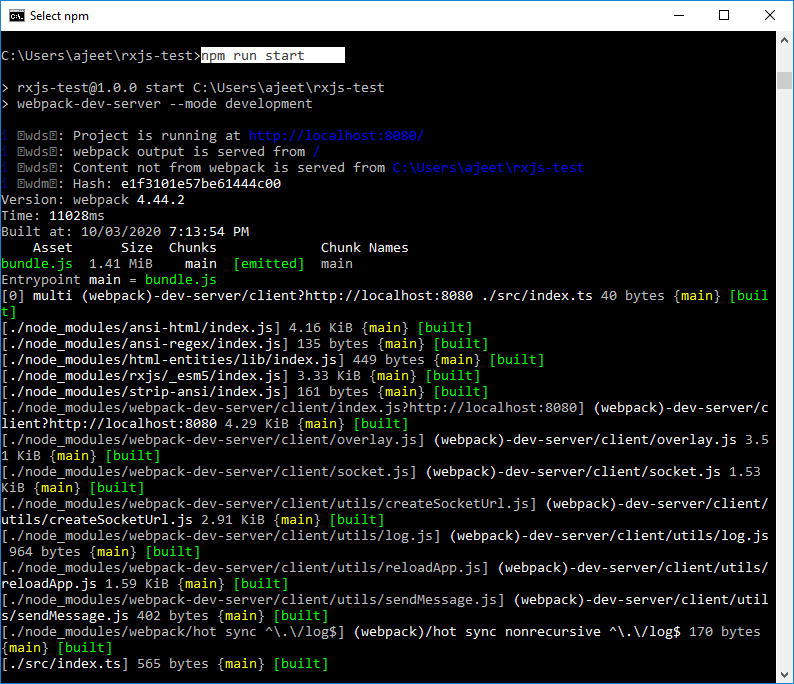
Now, the project setup is ready, the TypeScript and Webpack configuration set and the [HTML](https://www.javatpoint.com/html-tutorial) document has been created. Next, create a new subfolder named **src** inside the project folder and then also create a new file named **index.ts** within it. Insert the following code inside **index.ts** file:

1. **import** { Observable } from 'rxjs';
2. var observable = Observable.create((observer:any) => {
3. observer.next('Hello World!');
4. observer.next('Welcome to JavaTpoint!');
5. observer.complete();
6. observer.next('Bye');
7. })
8. observable.subscribe(
9. (x:any) => logItem(x),
10. (error: any) => logItem ('Error: ' + error),
11. () => logItem('This is the first Example')
12. );
13. function logItem(val:any) {
14. var node = document.createElement("li");
15. var textnode = document.createTextNode(val);
16. node.appendChild(textnode);
17. document.getElementById("list").appendChild(node);
18. }

### Run the Project

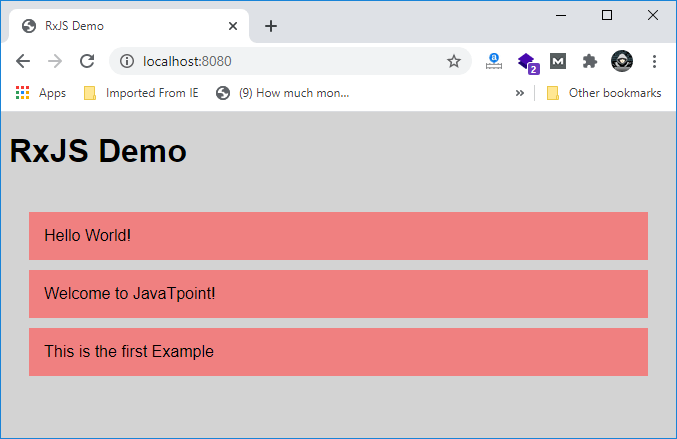
Now, the project is ready to start. Run the following command to open the development web server:

1. npm run start



Now, you can access the web application via URL http://localhost:8080. Open your web browser to see the following result:

**Output**



RxJS Latest Release Updates

The latest version of RxJS is v6. It is version 6, which is released recently, and we are doing this tutorial in RxJS version 6. We know that RxJS is used to deal with reactive programming. It is common and most often used with [Angular](https://www.javatpoint.com/angularjs-tutorial) and ReactJS. Angular 6 loads rxjs6 by default.

Why update RxJS Version 6?

[RxJS](https://www.javatpoint.com/rxjs) v6 is the major version change from RxJS 5.x to RxJS 6.x. The developers have done a lot of work to keep the hard breaking changes to a minimum as usual. In most cases, this allows application and library developers to update incrementally and use RxJS v6 without any modifications to their code. But, RxJS version 5 was handled differently in comparison to RxJS version 6. The code will break if you update your RxJS 5 to 6.

How to update RxJS v5 to RxJS v6 efficiently?

The backward-compatibility layer makes the update process easy and allows you to keep your apps working without failure while making changes in your code. This complete process is carried out in the following stages.

Here, we will see the different ways of handling the version update:

* Update to the latest version of RxJS 5.5 and fix all issues caused by bug fixes.
* Install RxJS v6 along with the backward-compatibility package, by using the command rxjs-compact.
* If your app faces problems and the breaking changes are not covered by rxjs-compat, then update the affected code according to the instructions provided below.
* Drop the compatibility layer to complete the update to RxJS v6. By doing this, it will significantly decrease the size of your apps.
* Use the rxjs-tslint command to refactor the TypeScript code so that it doesn't depend on rxjs-compat.

1. npm i -g rxjs-tslint
2. rxjs-5-to-6-migrate -p [path/to/tsconfig.json]

* Remove and replace all the deprecated functionality before the release of RxJS v7.

If you want to update your RxJS v5 to RxJS v6 and don't want to make the code changes, you can do it by installing the following package:

1. npm install --save-dev rxjs-compact

After installing this package, it will take care of providing backward compatibility, and your old code will work fine with RxJS version 6.

If you want to make changes in the code that it works fine with RxJS v6, you can do the following changes. The packages for operators, observables, and subject were restructured and so you have to change them how they have import for RxJS v6 in the following way:

Changes in Imports for operators

In RxJS v5, the operators and the following import statements are included as following:

1. **import** 'rxjs/add/operator/mapTo'
2. **import** 'rxjs/add/operator/take'
3. **import** 'rxjs/add/operator/tap'
4. **import** 'rxjs/add/operator/map'

In RxJS v6, these would be import as following:

1. **import** {mapTo, take, tap, map} from "rxjs/operators"

Changes in Import of Methods to create Observables

In RxJS v5, when we work with Observables, we have to import methods as following:

1. **import** "rxjs/add/observable/from";
2. **import** "rxjs/add/observable/of";
3. **import** "rxjs/add/observable/fromEvent";
4. **import** "rxjs/add/observable/interval";

In RxJS v6, this would be done as following:

1. **import** {from, of, fromEvent, interval} from 'rxjs';

Changes in Import of Observables

In RxJS v5, when we work with Observables, we have to import statements as following:

1. **import** { Observable } from 'rxjs/Observable'

In RxJS v6, this would be done as following:

1. **import** { Observable } from 'rxjs'

Changes in Import of Subject

In RxJS v5, we have to import subjects as following:

1. **import** { Subject} from 'rxjs/Subject'

In RxJS v6, this would be done as follows:

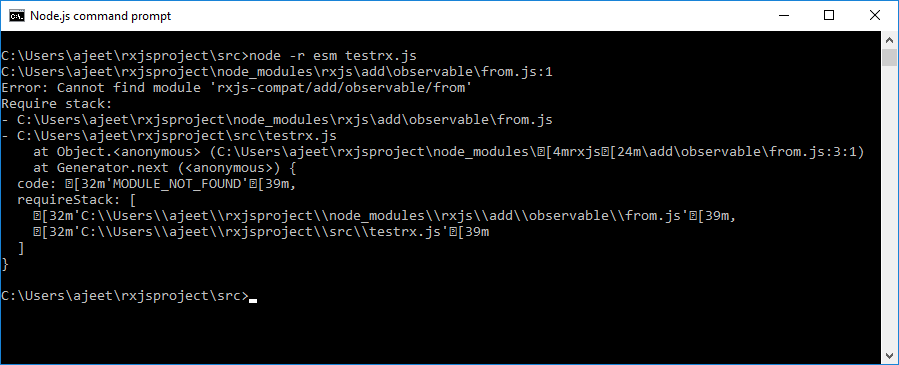
1. **import** { Subject } from 'rxjs'

This is how we use the operators in RxJS version 5. See the following example. In this example, we will find out the maximum value from a list.

**Example:**

1. **import** "rxjs/add/observable/from";
2. **import** 'rxjs/add/operator/max'
3. let list1 = [3, 7, 45, 22, 76, 11, 27];
4. from(list1).max((a,b)=>a-b).subscribe(x => console.log("The Max value is "+x));

This example is only compatible in RxJS version 5. If you run it in RxJS v6, it will give an error. See the following image:

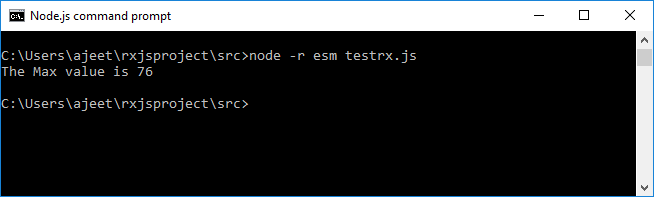


The pipe() method is available in RxJS v6 when the observable is created. It is added to RxJS from version 5.5. By using this method, you can work on multiple operators together in sequential order.

From RxJS version 5.5 onwards, we have to use pipe() method to execute the operator. See how the above example will be wrtten in rxJS v6:

1. **import** { from } from 'rxjs';
2. **import** { max } from 'rxjs/operators';
3. let list1 = [3, 7, 45, 22, 76, 11, 27];
4. from(list1).pipe(max((a,b)=>a-b)).subscribe(x => console.log(
5. "The Max value is "+x)
6. );

**Output:**



You can see that it has shown the correct output.

Operators that have been renamed

During the restructuring of the packages in RxJS v6, some of the operators are renamed because they were conflicting or matching with [JavaScript](https://www.javatpoint.com/javascript-tutorial) keywords. The following list shows those operators:

|  |  |
| --- | --- |
| **Operator** | **Renamed to** |
| do() | tap() |
| catch() | catchError() |
| switch() | switchAll() |
| finally() | finalize() |
| throw() | throwError() |