RxJS 工作环境配置

1. 需要安装 node.js , 使用 npm 安装各种模块
2. 初始化项目： npm init 生成 package.json 文件

* npm init

Package.json:

{

"name": "myrx1",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC"

}

1. 安装 Typescript : npm install typescript

* npm install typescript

1. 创建 typescript config file: tsconfig.json

Tsconfig.json:

{

"compilerOptions": {

"noImplicitAny": true,

"noEmitOnError": true,

"removeComments": true,

"module": "commonjs",

"sourceMap": true,

"target": "es5",

"allowSyntheticDefaultImports": true,

"experimentalDecorators": true,

//"moduleResolution": "node",

//"typeRoots" : ["./typings"],

//"types": ["jquery", "angular"]

"outDir": "dst", -- 创建编译后的目录

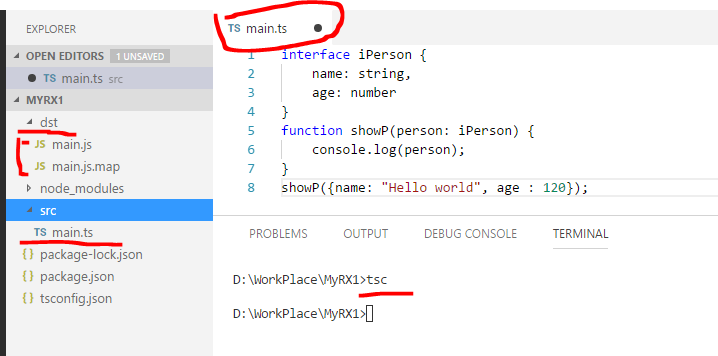
},

"include":["src"], -- 创建原代码目录

"exclude": ["dst", "node\_modules"]

}

1. 测试Typescript 编译器正常工作：



1. 安装webpack 打包工具和测试环境

* npm install webpack
* npm install webpack-cli
* npm install webpack-dev-server
* 安装必要的加载包：ts-loader, babel-loader, style-loader,css-loader
* npm install ts-loader babel-loader style-loader css-loader

1. webpack 配置文件：webpack.config.js

webpack.config.js 各参数的具体用法请参考 webpack 文档

var path = require("path");

module.exports = {

context: \_\_dirname,

entry: {

main: ["./src/main.ts"]

},

output: {

path: path.resolve(\_\_dirname, "js"),

publicPath: "jstest",

filename: "[name].js"

//chunkFilename: 'js/jquery.chunk.js'

},

module: {

rules: [

{ test: /\.ts$/, use: ["ts-loader"], exclude: /node\_modules/ },

{ test: /\.js$/, use: ["babel-loader"], exclude: /node\_modules/ },

{ test: /\.css$/, use:["style-loader", "css-loader"], exclude: /node\_modules/ }

]

},

devServer: {

contentBase: path.resolve(\_\_dirname, "test"),

compress: true,

port: 8018,

host: "localhost",

inline: true

},

resolve: {

extensions: [".ts", ".js", ".css"] //注意不能包括含： ".d.ts"

}

}

**注意**： resolve -> extensions:[] 注意不能包括含： ".d.ts"

否则会出现下面的错误信息：

@ ./node\_modules/rxjs-compat/Rx.js 147:0-32

@ ./node\_modules/rxjs/Rx.d.ts

@ ./src/main.ts

@ multi ./src/main.ts

-----------------------------------------------------------------------------------

webpack.config.js 的另外一种配置：

var path = require("path");

module.exports = {

context: path.resolve(\_\_dirname, "src"), //把工作目录改成 “src”s

watch: false, //使用devServer, 建议不使用 watch 模式

//生成 .js.map sourceMap文件

devtool: 'source-map', // source-map; inline-source-map; cheap-module-eval-source-map

entry: {

main: ["./main.ts"] //直接使用文件，但是需要写上相对路径 “./”

},

output: {

path: path.resolve(\_\_dirname, "js"),

publicPath: "jstest",

filename: "[name].js"

//chunkFilename: 'js/jquery.chunk.js'

},

module: {

rules: [

{ test: /\.ts$/, use: ["ts-loader"], exclude: /node\_modules/ },

{ test: /\.js$/, use: ["babel-loader"], exclude: /node\_modules/ },

{ test: /\.css$/, use:["style-loader", "css-loader"], exclude: /node\_modules/ }

]

},

devServer: {

contentBase: path.resolve(\_\_dirname, "test"),

**historyApiFallback**: {

index: 'goodmain.html' //当找不到index 文件，则使用这里指定的文件作为默认文件

},

compress: true,

port: 8018,

host: "localhost",

inline: true

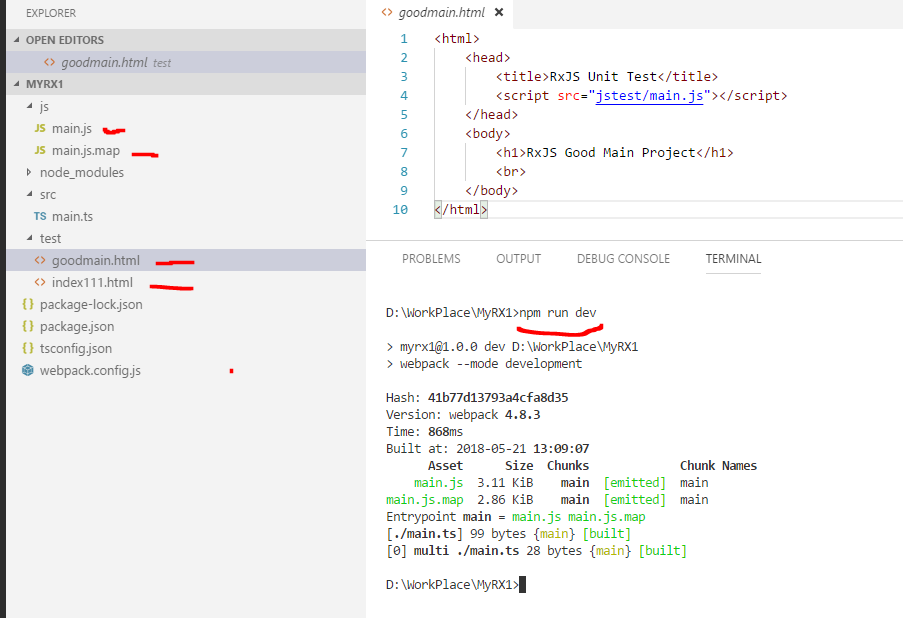
},

resolve: {

extensions: [".ts", ".js", ".css"]

}

}



1. 设置运行环境： package.json

package.json

{

"name": "myrx1",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"**start**": "webpack-dev-server --mode development --open",

"**dev**": "webpack --mode development",

"**build**": "webpack --mode production",

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC",

"dependencies": {

"babel-loader": "^7.1.4",

"css-loader": "^0.28.11",

"style-loader": "^0.21.0",

"ts-loader": "^4.3.0",

"typescript": "^2.8.3",

"webpack": "^4.8.3",

"webpack-cli": "^2.1.3",

"webpack-dev-server": "^3.1.4"

}

}

1. 安装常用的JQuery 和 RxJS

* npm install @types/jquery
* npm install jquery --必须安装 @types/jquery 和 jquery
* npm install rxjs-compat -- 必须先安装依赖包
* npm install rxjs – 然后安装rxjs
* npm install ts-promise – 安装 Typescript 的 Promise

为什么要先安装 rxjs-compat?

请查看 node\_modules/rxjs/Rx.d.ts 文件

Rx.d.ts

export \* from 'rxjs-compat';

1. 程序中引用 jquery 和 rxjs 模块

import \* as $ from "jquery";

import \* as Rx from "rxjs/Rx";

import Promise from "ts-promise"

$(function(){

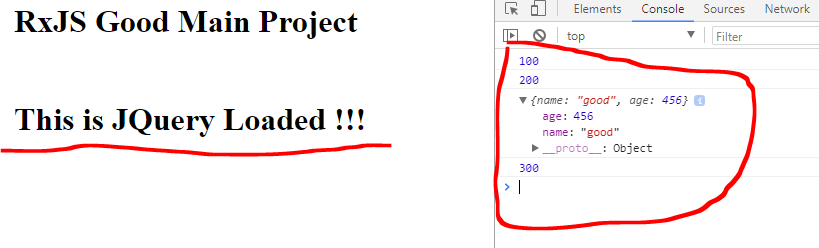
$("body").append("<h1>This is JQuery Loaded !!!</h1>");

})

Rx.Observable.from([100, 200, {name:"good", age:456}, 300 ])

.subscribe(a=>console.log(a));

运行结果：

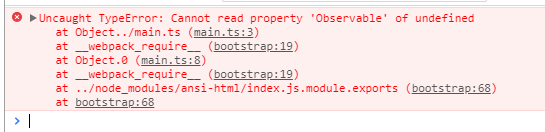


注意：网上有的例子

import $ from "jquery"; --错误：由于没有default export



import Rx from "rxjs/Rx"; --错误：由于没有default export



总结：完整的案例配置文件：

psackage.json

{

"name": "myrx1",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"start": "webpack-dev-server --mode development --open",

"dev": "webpack --mode development",

"build": "webpack --mode production",

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC",

"dependencies": {

"@types/jquery": "^3.3.2",

"babel-loader": "^7.1.4",

"css-loader": "^0.28.11",

"jquery": "^3.3.1",

"rxjs": "^6.1.0",

"rxjs-compat": "^6.1.0",

"style-loader": "^0.21.0",

"ts-loader": "^4.3.0",

"typescript": "^2.8.3",

"webpack": "^4.8.3",

"webpack-cli": "^2.1.3",

"webpack-dev-server": "^3.1.4"

}

}

tsconfig.json

{

"compilerOptions": {

"noImplicitAny": true,

"noEmitOnError": true,

"removeComments": true,

"module": "commonjs",

"sourceMap": true,

"target": "es5",

"allowSyntheticDefaultImports": true,

"experimentalDecorators": true,

//"moduleResolution": "node",

//"typeRoots" : ["./typings"],

//"types": ["jquery"],

"outDir": "js"

},

"include":["src"],

"exclude": ["node\_modules"]

}

Webpack.config.js

var path = require("path");

module.exports = {

context: path.resolve(\_\_dirname, "src"),

watch: false,

devtool: 'source-map', // source-map; inline-source-map; cheap-module-eval-source-map

entry: {

main: ["./main.ts"]

},

output: {

path: path.resolve(\_\_dirname, "js"),

publicPath: "jstest",

filename: "[name].js"

//chunkFilename: 'js/jquery.chunk.js'

},

module: {

rules: [

{ test: /\.ts$/, use: ["ts-loader"], exclude: /node\_modules/ },

{ test: /\.js$/, use: ["babel-loader"], exclude: /node\_modules/ },

{ test: /\.css$/, use:["style-loader", "css-loader"], exclude: /node\_modules/ }

]

},

devServer: {

contentBase: path.resolve(\_\_dirname, "test"),

historyApiFallback: {

index: 'goodmain.html'

},

compress: true,

port: 8018,

host: "localhost",

inline: true

},

resolve: {

extensions: [".ts", ".js", ".css"]

}

}

src/main.ts

import \* as $ from "jquery";

import \* as Rx from "rxjs/Rx";

//测试RxJS

Rx.Observable.from([100, 200, {name:"good", age:456}, 300 ])

.subscribe(a=>console.log(a));

//测试JQuery

$(function(){

$("body").append("<h1>This is JQuery Loaded !!!</h1>");

})

//测试 Typescript

interface iPerson {

name: string,

age: number

}

function showP(person: iPerson) {

console.log(person);

}

showP({name: "Hello world", age : 1999});

test/goodmain.html

<html>

<head>

<title>RxJS Unit Test</title>

<script src="jstest/main.js"></script>

</head>

<body>

<h1>RxJS Good Main Project</h1>

<br>

</body>

</html>

关于Typescript – Promise , 安装ts-promise 是关键

* npm install ts-promise – 安装 Typescript 的 Promise

tsconfig.json

{

"compilerOptions": {

"noImplicitAny": true,

"noEmitOnError": true,

"removeComments": true,

"module": "commonjs",

"sourceMap": true,

"target": "es2015",

"allowSyntheticDefaultImports": true,

"experimentalDecorators": true,

"moduleResolution": "node",

//"typeRoots" : ["./typings"],

//"types": ["jquery"],

"outDir": "js",

"lib": [ -- lib 是可选，可有可无的设置，不重要

"dom",

"es2015",

"scripthost",

"es2015.promise"

]

},

"include":["src"],

"exclude": ["node\_modules"]

}

测试 main.ts

import \* as $ from "jquery";

import \* as Rx from "rxjs/Rx";

import Promise from "ts-promise"

//测试RxJS - Typescript Promise

var ppp = new Promise(

(res, rej)=> {

res("Hello World and Good Victoria Day!!!");

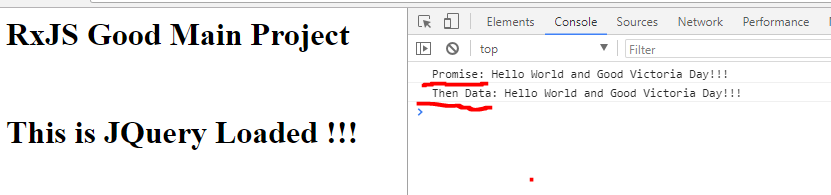
}

);

Rx.Observable.fromPromise(ppp).subscribe(a=>console.log("Promise: " + a));

ppp.then(data => console.log("Then Data: " + data));

输出结果：



如果Typescript 同时使用 Promise，async/await

则会出现名称冲突：解决的方法是对Promise 重新命名

import \* as $ from "jquery";

import \* as Rx from "rxjs/Rx";

import promSSS from "ts-promise"; //重新命名PromSSS,否则名字冲突

//测试 async - 即返回的是 promise

async function showMe() {

return "This is Async Method";

};

showMe().then( a => console.log("Async: " + a) );

Rx.Observable.fromPromise(showMe()).subscribe(a=>console.log("Rx Async FromPromise: " + a));

//测试RxJS - Typescript Promise

var ppp = new promSSS(

(res, rej)=> {

res("Hello World and Good Victoria Day!!!");

}

);

Rx.Observable.fromPromise(ppp).subscribe(a=>console.log("Promise: " + a));

ppp.then(data => console.log("Then Data: " + data));

输出结果：

