무료 전자 책

배우기

Angular 2

Free unaffiliated eBook created from
Stack Overflow contributors.

#angular2

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: angular-2

It is an unofficial and free Angular 2 ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Angular 2.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1: Angular 2

Angular2 + Angular-Cli IDE .

Angular AngularJS Angular 1 . .

4.3.3	2017-08-02
4.3.2	2017-07-26
4.3.1	2017-07-19
4.3.0	2017-07-14
4.2.0	2017-06-08
4.1.0	2017-04-26
4.0.0	2017-03-23
2.3.0	2016-12-08
2.2.0	2016-11-14
2.1.0	2016-10-13
2.0.2	2016-10-05
2.0.1	2016-09-23
2.0.0	2016-09-14
2.0.0-rc.7	2016-09-13
2.0.0-rc.6	2016-08-31
2.0.0-rc.5	2016-08-09
2.0.0-rc.4	2016-06-30
2.0.0-rc.3	2016-06-21
2.0.0-rc.2	2016-06-15
2.0.0-rc.1	2016-05-03
2.0.0-rc.0	2016-05-02

Examples

angular-cli angular2

Angular 2 .

:

- Node.js v4 .
- npm v3 .

.

```
npm install -g @angular/cli
```

```
yarn global add @angular/cli
```

.

, **/ CLI** @ `ng` PATH .

.

.

```
ng new PROJECT_NAME
cd PROJECT_NAME
ng serve
```

, Angular 2 . .

.

.

```
ng init
```

. .

.

```
ng serve
```

. .

```
http://localhost:4200
```

Hot Module Reloading HTML, CSS ( ).

, ,

`ng generate <scaffold-type> <name>` (`ng g <scaffold-type> <name>`) .

```
# The command below will generate a component in the folder you are currently at
ng generate component my-generated-component
# Using the alias (same outcome as above)
ng g component my-generated-component
```

angular-cli .

	`ng g module my-new-module`
	`ng g component my-new-component`
	`ng g directive my-new-directive`
	`ng g pipe my-new-pipe`
	`ng g service my-new-service`
	`ng g class my-new-class`
	`ng g interface my-new-interface`
	`ng g enum my-new-enum`

. :

`ng gm my-new-module` `ng gm my-new-module` `ng gc my-new-component` `ng gc my-new-component` .

/

Angular 2 Apache Tomcat . .

```
ng build
```

```
ng build --prod
```

`/dist` .

Ahead-of-Time . .

```
ng build --prod --aot
```

Angular 2 Angular-Cli . jasmine . .

```
ng test
```

.

angular-cli github .

angular-cli Angular 2 .

2.0.0-rc.4

"Hello World!" (`AppComponent`) `AppComponent` .

:

- Node.js v5
- npm v3

  : / `node -v npm -v` .

1

. `angular2-example` `angular2-example` .

```
mkdir angular2-example
cd angular2-example
```

2

`package.json` , `tsconfig.json` , `typings.json` `systemjs.config.js` 4 .

: Official 5 Minute Quickstart .

`package.json` - npm . (Gulp .)

```
{
  "name": "angular2-example",
  "version": "1.0.0",
  "scripts": {
    "start": "tsc && concurrently \"npm run tsc:w\" \"npm run lite\" ",
    "lite": "lite-server",
    "postinstall": "typings install",
    "tsc": "tsc",
    "tsc:w": "tsc -w",
    "typings": "typings"
  },
  "license": "ISC",
```

```
  "dependencies": {
    "@angular/common": "2.0.0-rc.4",
    "@angular/compiler": "2.0.0-rc.4",
    "@angular/core": "2.0.0-rc.4",
    "@angular/forms": "0.2.0",
    "@angular/http": "2.0.0-rc.4",
    "@angular/platform-browser": "2.0.0-rc.4",
    "@angular/platform-browser-dynamic": "2.0.0-rc.4",
    "@angular/router": "3.0.0-beta.1",
    "@angular/router-deprecated": "2.0.0-rc.2",
    "@angular/upgrade": "2.0.0-rc.4",
    "systemjs": "0.19.27",
    "core-js": "^2.4.0",
    "reflect-metadata": "^0.1.3",
    "rxjs": "5.0.0-beta.6",
    "zone.js": "^0.6.12",
    "angular2-in-memory-web-api": "0.0.14",
    "bootstrap": "^3.3.6"
  },
  "devDependencies": {
    "concurrently": "^2.0.0",
    "lite-server": "^2.2.0",
    "typescript": "^1.8.10",
    "typings":"^1.0.4"
  }
}
```

`tsconfig.json` - TypeScript .

```
{
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "moduleResolution": "node",
    "sourceMap": true,
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "removeComments": false,
    "noImplicitAny": false
  }
}
```

`typings.json` - TypeScript .

```
{
  "globalDependencies": {
    "core-js": "registry:dt/core-js#0.0.0+20160602141332",
    "jasmine": "registry:dt/jasmine#2.2.0+20160621224255",
    "node": "registry:dt/node#6.0.0+20160621231320"
  }
}
```

`systemjs.config.js` - `systemjs.config.js` (webpack).

```
/**
 * System configuration for Angular 2 samples
 * Adjust as necessary for your application's needs.
```

```
 */
(function(global) {
  // map tells the System loader where to look for things
  var map = {
    'app':                        'app', // 'dist',
    '@angular':                   'node_modules/@angular',
    'angular2-in-memory-web-api': 'node_modules/angular2-in-memory-web-api',
    'rxjs':                       'node_modules/rxjs'
  };
  // packages tells the System loader how to load when no filename and/or no extension
  var packages = {
    'app':                        { main: 'main.js',  defaultExtension: 'js' },
    'rxjs':                       { defaultExtension: 'js' },
    'angular2-in-memory-web-api': { main: 'index.js', defaultExtension: 'js' },
  };
  var ngPackageNames = [
    'common',
    'compiler',
    'core',
    'forms',
    'http',
    'platform-browser',
    'platform-browser-dynamic',
    'router',
    'router-deprecated',
    'upgrade',
  ];
  // Individual files (~300 requests):
  function packIndex(pkgName) {
    packages['@angular/'+pkgName] = { main: 'index.js', defaultExtension: 'js' };
  }
  // Bundled (~40 requests):
  function packUmd(pkgName) {
    packages['@angular/'+pkgName] = { main: '/bundles/' + pkgName + '.umd.js',
defaultExtension: 'js' };
  }
  // Most environments should use UMD; some (Karma) need the individual index files
  var setPackageConfig = System.packageWithIndex ? packIndex : packUmd;
  // Add package entries for angular packages
  ngPackageNames.forEach(setPackageConfig);
  var config = {
    map: map,
    packages: packages
  };
  System.config(config);
})(this);
```

3

.

```
npm install
```

/ .

4

`angular2-example`

index.html .

```
<html>
  <head>
    <title>Angular2 example</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- 1. Load libraries -->
    <!-- Polyfill(s) for older browsers -->
    <script src="node_modules/core-js/client/shim.min.js"></script>
    <script src="node_modules/zone.js/dist/zone.js"></script>
    <script src="node_modules/reflect-metadata/Reflect.js"></script>
    <script src="node_modules/systemjs/dist/system.src.js"></script>
    <!-- 2. Configure SystemJS -->
    <script src="systemjs.config.js"></script>
    <script>
      System.import('app').catch(function(err){ console.error(err); });
    </script>
  </head>
  <!-- 3. Display the application -->
  <body>
    <my-app></my-app>
  </body>
</html>
```

my-app .

Angular . AppComponent AppComponent .

5

app app . (AppComponent main.ts .)

```
mkdir app
```

6

app/app.component.ts .

```
import { Component } from '@angular/core';

@Component({
  selector: 'my-app',
  template: `
    <h1>{{title}}</h1>
    <ul>
        <li *ngFor="let message of messages">
            {{message}}
        </li>
    </ul>
  `
})
export class AppComponent {
    title = "Angular2 example";
```

```
    messages = [
        "Hello World!",
        "Another string",
        "Another one"
    ];
}
```

? , Angular HTML `@Component` `@Component` . `title` `messages` `AppComponent` `AppComponent` .

.

```
<h1>{{title}}</h1>
<ul>
    <li *ngFor="let message of messages">
        {{message}}
    </li>
</ul>
```

`h1` `title` `*ngFor` `messages` . `*ngFor` `li` `message` . .

```
<h1>Angular 2 example</h1>
<ul>
    <li>Hello World!</li>
    <li>Another string</li>
    <li>Another one</li>
</ul>
```

7

Angular `main.ts` .

`app/main.ts` .

```
import { bootstrap }    from '@angular/platform-browser-dynamic';
import { AppComponent } from './app.component';

bootstrap(AppComponent);
```

`bootstrap` `AppComponent` `bootstrap` Angular .

8

.

```
npm start
```

/ . lite-server TypeScript `package.json` (`.ts`) .

?

Angular 2 StackOverflow .

`AppComponent` , / . .

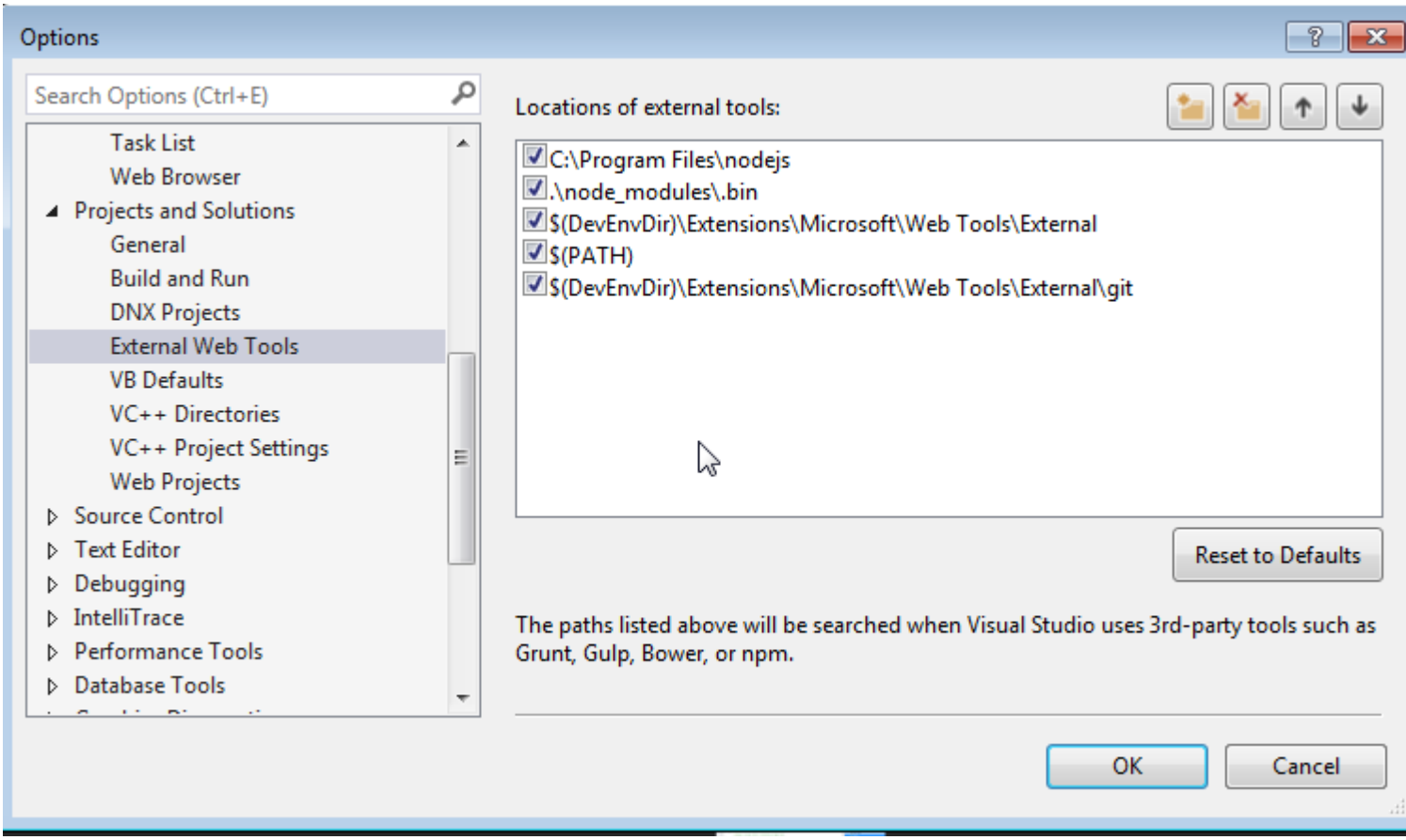
Visual Studio NPM NODE

1 : Node.js . C : / program files / nodejs .

2 : Visual Studio "> " .

3 : " > " .

4 : Node.js (C : / program files / nodejs) .



5 : Visual Studio npm npm .

.

XYZ MegaCorp Windows Angular2 .

() .

1. NPM
2.

NPM `.npmrc` .

```
proxy=http://[DOMAIN]%5C[USER]:[PASS]@[PROXY]:[PROXYPORT]/
https-proxy=http://[DOMAIN]%5C[USER]:[PASS]@[PROXY]:[PROXYPORT]/
```

`.typingsrc` .

```
proxy=http://[DOMAIN]%5C[USER]:[PASS]@[PROXY]:[PROXYPORT]/
https-proxy=http://[DOMAIN]%5C[USER]:[PASS]@[PROXY]:[PROXYPORT]/
rejectUnauthorized=false
```

. (`package.json` `%HOMEPATH%`).

`\` URL `%5C` . Steve Roberts . .pac npm

node.js / expressjs Angular 2 (http)

"Hello World!" . node.js (expressjs) Angular2 2.4.1 (`@NgModule`) .

- Node.js v4.xx
- npm v3.xx

`npm install -g typescript` `yarn global add typescript` typescript `yarn global add typescript` `yarn global add typescript` .

1

( ) . `Angular2-express` `Angular2-express` .

:

```
mkdir Angular2-express
cd Angular2-express
```

2

`node.js` `package.json` ( ) `app.js` ( ) .

package.json :

```
{
  "name": "Angular2-express",
  "version": "1.0.0",
  "description": "",
  "scripts": {
    "start": "node app.js"
  },
  "author": "",
  "license": "ISC",
```

```
  "dependencies": {
    "body-parser": "^1.13.3",
    "express": "^4.13.3"
  }
}
```

app.js :

```
var express = require('express');
var app = express();
var server = require('http').Server(app);
var bodyParser = require('body-parser');

server.listen(process.env.PORT || 9999, function(){
    console.log("Server connected. Listening on port: " + (process.env.PORT || 9999));
});

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: true}) );

app.use( express.static(__dirname + '/front' ) );

app.get('/test', function(req,res){ //example http request receiver
  return res.send(myTestVar);
});

//send the index.html on every page refresh and let angular handle the routing
app.get('/*',  function(req, res, next) {
    console.log("Reloading");
    res.sendFile('index.html', { root: __dirname });
});
```

npm install yarn .

. .

3

front Angular2-express .

:

```
mkdir front
cd front
```

. . package.json , systemjs.config.js , tsconfig.json

package.json :

```
{
  "name": "Angular2-express",
  "version": "1.0.0",
  "scripts": {
    "tsc": "tsc",
```

```
    "tsc:w": "tsc -w"
  },
  "licenses": [
    {
      "type": "MIT",
      "url": "https://github.com/angular/angular.io/blob/master/LICENSE"
    }
  ],
  "dependencies": {
    "@angular/common": "~2.4.1",
    "@angular/compiler": "~2.4.1",
    "@angular/compiler-cli": "^2.4.1",
    "@angular/core": "~2.4.1",
    "@angular/forms": "~2.4.1",
    "@angular/http": "~2.4.1",
    "@angular/platform-browser": "~2.4.1",
    "@angular/platform-browser-dynamic": "~2.4.1",
    "@angular/platform-server": "^2.4.1",
    "@angular/router": "~3.4.0",
    "core-js": "^2.4.1",
    "reflect-metadata": "^0.1.8",
    "rxjs": "^5.0.2",
    "systemjs": "0.19.40",
    "zone.js": "^0.7.4"
  },
  "devDependencies": {
    "@types/core-js": "^0.9.34",
    "@types/node": "^6.0.45",
    "typescript": "2.0.2"
  }
}
```

systemjs.config.js :

```
/**
 * System configuration for Angular samples
 * Adjust as necessary for your application needs.
 */
(function (global) {
  System.config({
    defaultJSExtensions:true,
    paths: {
      // paths serve as alias
      'npm:': 'node_modules/'
    },
    // map tells the System loader where to look for things
    map: {
      // our app is within the app folder
      app: 'app',
      // angular bundles
      '@angular/core': 'npm:@angular/core/bundles/core.umd.js',
      '@angular/common': 'npm:@angular/common/bundles/common.umd.js',
      '@angular/compiler': 'npm:@angular/compiler/bundles/compiler.umd.js',
      '@angular/platform-browser': 'npm:@angular/platform-browser/bundles/platform-
browser.umd.js',
      '@angular/platform-browser-dynamic': 'npm:@angular/platform-browser-
dynamic/bundles/platform-browser-dynamic.umd.js',
      '@angular/http': 'npm:@angular/http/bundles/http.umd.js',
      '@angular/router': 'npm:@angular/router/bundles/router.umd.js',
      '@angular/forms': 'npm:@angular/forms/bundles/forms.umd.js',
```

```
      // other libraries
      'rxjs':                      'npm:rxjs',
      'angular-in-memory-web-api': 'npm:angular-in-memory-web-api',
    },
    // packages tells the System loader how to load when no filename and/or no extension
    packages: {
      app: {
        main: './main.js',
        defaultExtension: 'js'
      },
      rxjs: {
        defaultExtension: 'js'
      }
    }
  });
})(this);
```

tsconfig.json :

```
{
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "moduleResolution": "node",
    "sourceMap": true,
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "removeComments": false,
    "noImplicitAny": false
  },
  "compileOnSave": true,
  "exclude": [
    "node_modules/*"
  ]
}
```

`npm install yarn  .`

`. index.html  :`

index.html :

```
<html>
  <head>
    <base href="/">
    <title>Angular2-express</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- 1. Load libraries -->
     <!-- Polyfill(s) for older browsers -->
    <script src="node_modules/core-js/client/shim.min.js"></script>
    <script src="node_modules/zone.js/dist/zone.js"></script>
    <script src="node_modules/reflect-metadata/Reflect.js"></script>
    <script src="node_modules/systemjs/dist/system.src.js"></script>
    <!-- 2. Configure SystemJS -->
    <script src="systemjs.config.js"></script>
    <script>
      System.import('app').catch(function(err){ console.error(err); });
```

```
    </script>

  </head>
  <!-- 3. Display the application -->
  <body>
    <my-app>Loading...</my-app>
  </body>
</html>
```

. front app .

:

```
mkdir app
cd app
```

main.ts , app.module.ts , app.component.ts

main.ts :

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app.module';

const platform = platformBrowserDynamic();
platform.bootstrapModule(AppModule);
```

app.module.ts :

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HttpModule } from "@angular/http";

import { AppComponent }   from './app.component';

@NgModule({
  imports:      [
    BrowserModule,
    HttpModule
  ],
  declarations: [
    AppComponent
  ],
  providers:[ ],
  bootstrap:    [ AppComponent ]
})
export class AppModule {}
```

app.component.ts :

```
import { Component } from '@angular/core';
import { Http } from '@angular/http';

@Component({
  selector: 'my-app',
```

```
  template: 'Hello World!',
  providers: []
})
export class AppComponent {
  constructor(private http: Http){
    //http get example
    this.http.get('/test')
     .subscribe((res)=>{
       console.log(res);
     });
  }
}
```

typescript javascript . dir (Angular2-express) 2 .

:

```
cd ..
cd ..
tsc -p front
```

.

```
Angular2-express
├── app.js
├── node_modules
├── package.json
├── front
|   ├── package.json
|   ├── index.html
|   ├── node_modules
|   ├── systemjs.config.js
|   ├── tsconfig.json
|   ├── app
|   |   ├── app.component.ts
|   |   ├── app.component.js.map
|   |   ├── app.component.js
|   |   ├── app.module.ts
|   |   ├── app.module.js.map
|   |   ├── app.module.js
|   |   ├── main.ts
|   |   ├── main.js.map
|   |   ├── main.js
```

, Angular2-express `node app.js` `node app.js` . `localhost:9999` .

4 !

4 ! Angular Angular 2 semver . . Angular Angular 4 . 3 Angular 3 .

Angular Angular 4 . @ angular / core . . if / else . Angular 4 Typescript 2.1 2.2 . Angular 4 .

4 .

.

Angular-CLI (Command Line Interface) .

- Angular 2 Angular 4 .
- github Angular4 . (.)
- Angular-CLI ( ) .

Angular-CLI . , v7.8.0 . Angular-CLI .

```
npm install -g @angular/cli
```

```
yarn global add @angular/cli
```

.

Angular-CLI Angular 4 .

```
ng new Angular4-boilerplate
```

cd Angular4-boilerplate 4 . .

2 4

. Angular 2 Angular 4 . Angular 2 Angular 4 Dependency Angular 2 . :

```
"dependencies": {
    "@angular/animations": "^4.1.0",
    "@angular/common": "4.0.2",
    "@angular/compiler": "4.0.2",
    "@angular/core": "^4.0.1",
    "@angular/forms": "4.0.2",
    "@angular/http": "4.0.2",
    "@angular/material": "^2.0.0-beta.3",
    "@angular/platform-browser": "4.0.2",
    "@angular/platform-browser-dynamic": "4.0.2",
    "@angular/router": "4.0.2",
    "typescript": "2.2.2"
   }
```

Angular 4 . npm npm . package.json .

github

git . angular4-boilerplate .

```
git@github.com:CypherTree/angular4-boilerplate.git
```

.

```
npm install

npm start
```

Angular 4 . .

angular4 -

```
Angular4-boilerplate
-karma
-node_modules
-src
   -mocks
   -models
      -loginform.ts
      -index.ts
   -modules
      -app
        -app.component.ts
     -app.component.html
      -login
    -login.component.ts
    -login.component.html
    -login.component.css
      -widget
     -widget.component.ts
     -widget.component.html
     -widget.component.css
    ........
   -services
       -login.service.ts
    -rest.service.ts
   -app.routing.module.ts
   -app.module.ts
   -bootstrap.ts
   -index.html
   -vendor.ts
-typings
-webpack
-package.json
-tsconfig.json
-tslint.json
-typings.json
```

:

src .

mocks .

model .

modules app, login, widget . typescript, html css . index.ts .

services . . http . .

app.routing.ts .

app.module.ts app .

bootstrap.ts .

webpack webpack .

package.json .

.

node_modules .

. login.component.html

```
<form>Dreamfactory – Addressbook 2.0
 <label>Email</label> <input id="email" form="" name="email" type="email" />
 <label>Password</label> <input id="password" form="" name="password"
 type="password" />
 <button form="">Login</button>
</form>
```

login.component.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { Form, FormGroup } from '@angular/forms';
import { LoginForm } from '../../models';
import { LoginService } from '../../services/login.service';

@Component({
    selector: 'login',
    template: require('./login.component.html'),
    styles: [require('./login.component.css')]
})
export class LoginComponent {

    constructor(private loginService: LoginService, private router: Router, form: LoginForm) {
}

    getLogin(form: LoginForm): void {
      let username = form.email;
      let password = form.password;
      this.loginService.getAuthenticate(form).subscribe(() => {
          this.router.navigate(['/calender']);
      });
    }
}
```

index.ts .

```
export * from './login/login.component';
```

app.routes.ts .

```
const appRoutes: Routes = [
```

```
    {
        path: 'login',
        component: LoginComponent
    },
    ........
    {
        path: '',
        pathMatch: 'full',
        redirectTo: '/login'
    }
];
```

app.module.ts .

```
.....
import { LoginComponent } from './modules';
......
@NgModule({
    bootstrap: [AppComponent],
    declarations: [
       LoginComponent
       .....
       .....
       ]
      .....
  })
  export class AppModule { }
```

npm install npm . ,! . , angular4 - .

Angular 4 . Angular 2 .

Angular 2 : https://riptutorial.com/ko/angular2/topic/789/angular-2-

2: @HostBinding .

Examples

@HostBinding

@HostBinding . . . , . .

```
import { Directive, HostBinding, HostListener } from '@angular/core';

@Directive({
  selector: '[appButtonPress]'
})
export class ButtonPressDirective {
  @HostBinding('attr.role') role = 'button';
  @HostBinding('class.pressed') isPressed: boolean;

  @HostListener('mousedown') hasPressed() {
    this.isPressed = true;
  }
  @HostListener('mouseup') hasReleased() {
    this.isPressed = false;
  }
}
```

@HostBinding . . @HostBinding role . isPressed true .

@HostBinding . : https://riptutorial.com/ko/angular2/topic/9455/-hostbinding----------

3: Angular 2+ NPM

npm .

npm .

.

Examples

Angular 2+ npm .

`git` , `npm` , `gulp` , `typescript` .

.gitignore

`.gitignore` . .

```
npm-debug.log
node_modules
jspm_packages
.idea
build
```

.npmignore

`.npmignore` . .

```
examples
node_modules
src
```

gulpfile.js

Gulp `gulpfile.js` . . .

```
var gulp = require('gulp');
var embedTemplates = require('gulp-angular-embed-templates');
var inlineNg2Styles = require('gulp-inline-ng2-styles');

gulp.task('js:build', function () {
    gulp.src('src/*.ts') // also can use *.js files
        .pipe(embedTemplates({sourceType:'ts'}))
        .pipe(inlineNg2Styles({ base: '/src' }))
        .pipe(gulp.dest('./dist'));
});
```

index.d.ts

`index.d.ts` typescript . .

```
export * from './lib';
```

index.js

. NPM .

```
exports.AngularXMinimalNpmPackageModule = require('./lib').AngularXMinimalNpmPackageModule;
```

`lib` , `/lib` .

package.json

npm .

```
{
  "name": "angular-x-minimal-npm-package",
  "version": "0.0.18",
  "description": "An Angular 2+ Data Table that uses HTTP to create, read, update and delete
data from an external API such REST.",
  "main": "index.js",
  "scripts": {
    "watch": "tsc -p src -w",
    "build": "gulp js:build && rm -rf lib && tsc -p dist"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/vinagreti/angular-x-minimal-npm-package.git"
  },
  "keywords": [
    "Angular",
    "Angular2",
    "Datatable",
    "Rest"
  ],
  "author": "bruno@tzadi.com",
  "license": "MIT",
  "bugs": {
    "url": "https://github.com/vinagreti/angular-x-minimal-npm-package/issues"
  },
  "homepage": "https://github.com/vinagreti/angular-x-minimal-npm-package#readme",
  "devDependencies": {
    "gulp": "3.9.1",
    "gulp-angular-embed-templates": "2.3.0",
    "gulp-inline-ng2-styles": "0.0.1",
    "typescript": "2.0.0"
  },
  "dependencies": {
    "@angular/common": "2.4.1",
    "@angular/compiler": "2.4.1",
    "@angular/core": "2.4.1",
    "@angular/http": "2.4.1",
```

```
    "@angular/platform-browser": "2.4.1",
    "@angular/platform-browser-dynamic": "2.4.1",
    "rxjs": "5.0.2",
    "zone.js": "0.7.4"
  }
}
```

dist / tsconfig.json

dist . Typescript . typescript .

```
{
  "compilerOptions": {
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "mapRoot": "",
    "rootDir": ".",
    "target": "es5",
    "lib": ["es6", "es2015", "dom"],
    "inlineSources": true,
    "stripInternal": true,
    "module": "commonjs",
    "moduleResolution": "node",
    "removeComments": true,
    "sourceMap": true,
    "outDir": "../lib",
    "declaration": true
  }
}
```

. . HTML `<angular-x-minimal-npm-package></angular-x-minimal-npm-package>` . npm .

src / angular-x-minimal-npm-package.component.ts

```
import {Component} from '@angular/core';
@Component({
    selector: 'angular-x-minimal-npm-package',
    styleUrls: ['./angular-x-minimal-npm-package.component.scss'],
    templateUrl: './angular-x-minimal-npm-package.component.html'
})
export class AngularXMinimalNpmPackageComponent {
    message = "Click Me ...";
    onClick() {
        this.message = "Angular 2+ Minimal NPM Package. With external scss and html!";
    }
}
```

src / angular-x-minimal-npm-package.component.html

```
<div>
  <h1 (click)="onClick()">{{message}}</h1>
</div>
```

src / angular-x-data-table.component.css

```
h1{
    color: red;
}
```

src / angular-x-minimal-npm-package.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule  } from '@angular/common';

import { AngularXMinimalNpmPackageComponent } from './angular-x-minimal-npm-
package.component';

@NgModule({
  imports: [ CommonModule ],
  declarations: [ AngularXMinimalNpmPackageComponent ],
  exports:  [ AngularXMinimalNpmPackageComponent ],
  entryComponents: [ AngularXMinimalNpmPackageComponent ],
})
export class AngularXMinimalNpmPackageModule {}
```

, .

`gulp tsc` `tsc` . `scripts.build` package.json . `gulp js:build && rm -rf lib && tsc -p dist` `gulp js:build && rm -rf lib && tsc -p dist` . .

.

```
npm run build
```

`/lib` in `/dist` . `index.js` `/lib` `/src` .

---

npm . .

```
npm publish
```

!!!

Angular 2+ NPM : https://riptutorial.com/ko/angular2/topic/8790/angular-2plus-npm--

4: Angular 2 jQuery

Angular 2.x jQuery, Google Analytics, JavaScript API .

Examples

CLI

NPM

`jQuery` NPM

```
npm install --save jquery
```

`angular-cli.json`

```
"scripts": [
    "../node_modules/jquery/dist/jquery.js"
]
```

`assets/js` `angular-cli.json` `angular-cli.json`

```
"scripts": [
    "assets/js/jquery.js"
]
```

`jquery` `jquery-cycle-plugin` assets `angular-cli.json` . .

Angular 2.x jQuery

Angular 2.x `jquery` .

jQuery `$`

```
declare var $: any;
```

`jQuery` for jQuery

```
declare var jQuery: any
```

Angular 2.x `$` `jQuery` .

Angular 2 jQuery : https://riptutorial.com/ko/angular2/topic/9285/angular-2-jquery----

5: Angular2 CanActivate

Examples

Angular2 CanActivate

:

```
export const MainRoutes: Route[] = [{
   path: '',
   children: [ {
      path: 'main',
      component: MainComponent ,
      canActivate : [CanActivateRoute]
   }]
}];
```

canActivateRoute :

```
@Injectable()
  export class  CanActivateRoute implements CanActivate{
  constructor(){}
  canActivate(next: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {
     return true;
  }
}
```

Angular2 CanActivate : https://riptutorial.com/ko/angular2/topic/8899/angular2-canactivate

6: Angular2 Databinding

Examples

@()

: .

```
@Component({
  selector: 'parent-component',
  template: '<div>
               <child-component [users]="users"></child-component>
             </div>'
})
export class ParentComponent implements OnInit{
  let users : List<User> = null;

  ngOnInit() {
    users.push(new User('A', 'A', 'A@gmail.com');
    users.push(new User('B', 'B', 'B@gmail.com');
    users.push(new User('C', 'C', 'C@gmail.com');
  }
}
```

Input () .

```
@Component({
selector: 'child-component',
  template: '<div>
                  <table *ngIf="users !== null">
                    <thead>
                         <th>Name</th>
                         <th>FName</th>
                         <th>Email</th>
                    </thead>
                    <tbody>
                        <tr *ngFor="let user of users">
                            <td>{{user.name}}</td>
                            <td>{{user.fname}}</td>
                            <td>{{user.email}}</td>
                        </tr>
                    </tbody>
                  </table>

             </div>',
})
export class ChildComponent {
  @Input() users : List<User> = null;
}


export class User {
  name : string;
```

```
  fname : string;
  email : string;

  constructor(_name : string, _fname : string, _email : string){
     this.name = _name;
     this.fname = _fname;
     this.email = _email;
  }
}
```

Angular2 Databinding : https://riptutorial.com/ko/angular2/topic/9036/angular2-databinding

7: Angular2 In Memory API

Angular2-In-Memory-Web-Api API . .

Examples

mock-data.ts

API

```
export class MockData {
  createDb() {
    let mock = [
      { id: '1', name: 'Object A' },
      { id: '2', name: 'Object B' },
      { id: '3', name: 'Object C' },
      { id: '4', name: 'Object D' }
    ];

    return {mock};
  }
}
```

main.ts

XHRBackend InMemoryBackendService . .

```
createDb()
```

(MockData) SEED_DATA API .

```
import { XHRBackend, HTTP_PROVIDERS } from '@angular/http';
import { InMemoryBackendService, SEED_DATA } from 'angular2-in-memory-web-api';
import { MockData } from './mock-data';
import { bootstrap } from '@angular/platform-browser-dynamic';

import { AppComponent } from './app.component';

bootstrap(AppComponent, [
    HTTP_PROVIDERS,
    { provide: XHRBackend, useClass: InMemoryBackendService },
    { provide: SEED_DATA,  useClass: MockData }
]);
```

mock.service.ts

API get

```
import { Injectable }     from '@angular/core';
import { Http, Response } from '@angular/http';
import { Mock } from './mock';
```

```
@Injectable()
export class MockService {
  // URL to web api
  private mockUrl = 'app/mock';

  constructor (private http: Http) {}

  getData(): Promise<Mock[]> {
    return this.http.get(this.mockUrl)
                    .toPromise()
                    .then(this.extractData)
                    .catch(this.handleError);
  }

  private extractData(res: Response) {
    let body = res.json();
    return body.data || { };
  }

  private handleError (error: any) {
    let errMsg = (error.message) ? error.message :
      error.status ? `${error.status} - ${error.statusText}` : 'Server error';
    console.error(errMsg);
    return Promise.reject(errMsg);
  }
}
```

Test API

mock-data.ts

```
export class MockData {
    createDb() {
        let mock = [
            { id: '1', name: 'Object A' },
            { id: '2', name: 'Object B' },
            { id: '3', name: 'Object C' }
        ];

        let data = [
            { id: '1', name: 'Data A' },
            { id: '2', name: 'Data B' },
            { id: '3', name: 'Data C' }
        ];

        return { mock, data };
    }
}
```

.

```
app/mock
```

```
app/data
```

.

Angular2 In Memory API : https://riptutorial.com/ko/angular2/topic/6576/angular2-in-memory--api

8: Angular2

. control.value .

Examples

:

Angular 2 . ( ). .

```
export class CustomValidators {

static cannotContainSpace(control: Control) {
    if (control.value.indexOf(' ') >= 0)
        return { cannotContainSpace: true };

    return null;
}

static shouldBeUnique(control: Control) {
    return new Promise((resolve, reject) => {
        // Fake a remote validator.
        setTimeout(function () {
            if (control.value == "exisitingUser")
                resolve({ shouldBeUnique: true });
            else
                resolve(null);
        }, 1000);
    });
}}
```

null . .

Formbuilder

```
   constructor(fb: FormBuilder) {
    this.form = fb.group({
        firstInput: ['', Validators.compose([Validators.required,
CustomValidators.cannotContainSpace]), CustomValidators.shouldBeUnique],
        secondInput: ['', Validators.required]
    });
}
```

FormBuilder . FromBuilder .

1. .
2. . Validators.compose ([arrayOfValidators]) .
3. .

get / set formBuilder

formBuilder .

1. :

```
exampleForm : FormGroup;
constructor(fb: FormBuilder){
  this.exampleForm = fb.group({
      name : new FormControl({value: 'default name'}, Validators.compose([Validators.required,
Validators.maxLength(15)]))
   });
}
```

2.After :

```
this.exampleForm.controls['name'].setValue('default name');
```

formBuilder :

```
let name = this.exampleForm.controls['name'].value();
```

Angular2 : https://riptutorial.com/ko/angular2/topic/6284/angular2----

9: Angular2

Angular CSS . .

Examples

- .

app.component.html

```
<div>
  <div>
    <div *ngFor="let user of users">
      <button
        class="btn"
        [@buttonState]="user.active"
        (click)="user.changeButtonState()">{{user.firstName}}</button>
    </div>
  </div>
</div>
```

app.component.ts

```
import {Component, trigger, state, transition, animate, style} from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styles: [`
     .btn {
      height: 30px;
      width: 100px;
      border: 1px solid rgba(0, 0, 0, 0.33);
      border-radius: 3px;
      margin-bottom: 5px;
     }

  `],
  animations: [
    trigger('buttonState', [
      state('true', style({
        background: '#04b104',
        transform: 'scale(1)'
      })),
      state('false',   style({
        background: '#e40202',
        transform: 'scale(1.1)'
      })),
      transition('true => false', animate('100ms ease-in')),
      transition('false => true', animate('100ms ease-out'))
    ])
  ]
})
```

```
export class AppComponent {
  users : Array<User> = [];
  constructor(){
    this.users.push(new User('Narco', false));
    this.users.push(new User('Bombasto',false));
    this.users.push(new User('Celeritas', false));
    this.users.push(new User('Magneta', false));
  }
}


export class User {
  firstName : string;
  active : boolean;

  changeButtonState(){
    this.active = !this.active;
  }
  constructor(_firstName :string, _active : boolean){
    this.firstName = _firstName;
    this.active = _active;
  }

}
```

Angular2 : https://riptutorial.com/ko/angular2/topic/8970/angular2-

10: Angular2 () ()

Examples

()

: .

```
@Component({
  selector: 'parent-component',
  template: '<div>
               <child-component [users]="users"></child-component>
             </div>'
})
export class ParentComponent implements OnInit{
  let users : List<User> = null;

  ngOnInit() {
    users.push(new User('A', 'A', 'A@gmail.com');
    users.push(new User('B', 'B', 'B@gmail.com');
    users.push(new User('C', 'C', 'C@gmail.com');
  }
}
```

Input () .

```
@Component({
selector: 'child-component',
  template: '<div>
                  <table *ngIf="users !== null">
                    <thead>
                         <th>Name</th>
                         <th>FName</th>
                         <th>Email</th>
                    </thead>
                    <tbody>
                        <tr *ngFor="let user of users">
                            <td>{{user.name}}</td>
                            <td>{{user.fname}}</td>
                            <td>{{user.email}}</td>
                        </tr>
                    </tbody>
                  </table>

             </div>',
})
export class ChildComponent {
  @Input() users : List<User> = null;
}


export class User {
  name : string;
```

```
  fname : string;
  email : string;

  constructor(_name : string, _fname : string, _email : string){
     this.name = _name;
     this.fname = _fname;
     this.email = _email;
  }
}
```

html

```
<child-component [isSelected]="inputPropValue"></child-component>
```

TS

```
export class AppComponent {
     inputPropValue: true
}
```

ts :

```
export class ChildComponent {
    @Input() inputPropValue = false;
}
```

html :

```
<div [class.simpleCssClass]="inputPropValue"></div>
```

inputPropValue , . false. .

Angular2 () () : https://riptutorial.com/ko/angular2/topic/8943/angular2--------

11: Angular2 .

. , , .

Examples

Angular . .

```
import { platformBrowserDynamic } from "@angular/platform-browser-dynamic";
import { AppModule } from "./src/app";
export function runAngular2App(legacyModel: any) {
       platformBrowserDynamic([
           { provide: "legacyModel", useValue: model }
       ]).bootstrapModule(AppModule)
       .then(success => console.log("Ng2 Bootstrap success"))
       .catch(err => console.error(err));
}
```

" " .

```
import { Injectable } from "@angular/core";
@Injectable()
export class MyService {
   constructor(@Inject("legacyModel") private legacyModel) {
      console.log("Legacy data – ", legacyModel);
    }
 }
```

.

```
require(["myAngular2App"], function(app) {
   app.runAngular2App(legacyModel); // Input to your APP
});
```

Angular2 . : https://riptutorial.com/ko/angular2/topic/9203/angular2--------

12: angular-cli@1.0.0-β10 3

, .

```
'lodash': {
  format: 'cjs',
  defaultExtension: 'js',
  main: 'index.js'
}
```

```
'moment': {
  main: 'moment.js'
}
```

Examples

angular-cli jquery

1. npm jquery .

```
npm install jquery --save
```

 .

.

```
typings install jquery --global --save
```

---

2. angular-cli-build.js jquery vendorNpmFiles .

 . angular-cli-build.js .

 node_modules .

```
var Angular2App = require('angular-cli/lib/broccoli/angular2-app');

module.exports = function(defaults) {
  return new Angular2App(defaults, {
    vendorNpmFiles: [
      // ...
      'jquery/dist/*.js'


    ]
  });
};
```

---

3. jquery SystemJS .

SystemJS system-config.ts .

```
/** Map relative paths to URLs. */
const map: any = {
  'jquery': 'vendor/jquery'
};

/** User packages configuration. */
const packages: any = {

// no need to add anything here for jquery

};
```

---

4. src / index.html

```
<script src="vendor/jquery/dist/jquery.min.js" type="text/javascript"></script>
```

.

```
<script src="vendor/jquery/dist/jquery.js" type="text/javascript"></script>
```

```
<script src="/vendor/jquery/dist/jquery.slim.js" type="text/javascript"></script>
```

```
<script src="/vendor/jquery/dist/jquery.slim.min.js" type="text/javascript"></script>
```

---

5. jquery :

.ts jquery .

```
declare var $:any;

@Component({
})
export class YourComponent {
  ngOnInit() {
    $.("button").click(function(){
       // now you can DO, what ever you want
     });
     console.log();
  }
}
```

jquery . !

.

, 1.0.0-beta.10 -cli !

. TypeScript . .

1. (`index.html`)

```
<script src="//cdn.somewhe.re/lib.min.js" type="text/javascript"></script>
<script src="/local/path/to/lib.min.js" type="text/javascript"></script>
```

   - (: `THREE` , `mapbox` , `$`)

2. lib `declare` . TypeScript . [1]

```
declare var <globalname>: any;
```

   libs `window` .

```
interface WindowIntercom extends Window { Intercom: any; }
declare var window: WindowIntercom;
```

3. lib .

```
@Component { ... }
export class AppComponent implements AfterViewInit {
    ...
    ngAfterViewInit() {
        var geometry = new THREE.BoxGeometry( 1, 1, 1 );
        window.Intercom('boot', { ... }
    }
}
```

   - : libs DOM .

angular-cli@1.0.0-β10 3 : https://riptutorial.com/ko/angular2/topic/2328/angular-cli-1-0-0-β10--3---

13: API RXJS Observables

Angular 2 Http RxJS API Angular 1.x .

Http . Http HTTP `GET` , `POST` , `PUT` , `DELETE` , `PATCH` , `HEAD` . HTTP `request` .

`Http` `Observable<Response>` RxJS . `.subscribe()` .

`Response` HTTP . / .

, `Http` *(cold)*. , . . GET PUT POST .

Examples

HTTP GET . `http.get()` subscribe Observable . `posts` .

```
var posts = []

getPosts(http: Http):void {
    this.http.get(`https://jsonplaceholder.typicode.com/posts`)
        .map(response => response.json())
        .subscribe(post => posts.push(post));
}
```

API

HTTP . . `http.get()` Observable `.map` Response Post .

```
import {Injectable} from "@angular/core";
import {Http, Response} from "@angular/http";

@Injectable()
export class BlogApi {

  constructor(private http: Http) {
  }

  getPost(id: number): Observable<Post> {
    return this.http.get(`https://jsonplaceholder.typicode.com/posts/${id}`)
      .map((response: Response) => {
        const srcData = response.json();
        return new Post(srcData)
      });
  }
}
```

`Post` .

```
export class Post {
  userId: number;
  id: number;
  title: string;
```

```
  body: string;

  constructor(src: any) {
      this.userId = src && src.userId;
      this.id = src && src.id;
      this.title = src && src.title;
      this.body = src && src.body;
  }
}
```

BlogApi Http Post .

.

. forkJoin .

forkJoin Observables . Observable .subscribe . .subscribe .forkJoin . posts tags .

```
loadData() : void {
    Observable.forkJoin(
        this.blogApi.getPosts(),
        this.blogApi.getTags()
    ).subscribe((([posts, tags]: [Post[], Tag[]]) => {
        this.posts = posts;
        this.tags = tags;
    }));
}
```

API RXJS Observables : https://riptutorial.com/ko/angular2/topic/3577/api---rxjs---observables

14: ASP.net Angular 2 TypeScript

: ASP.NET Asp.net 2 2

2 Asp.Net . MVC Angular 2 Angular 2 MVC .

Examples

Asp.Net Core + Angular2 + Gulp

Startup.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using CoreAngular000.Data;
using CoreAngular000.Models;
using CoreAngular000.Services;
using Microsoft.Extensions.FileProviders;
using System.IO;

namespace CoreAngular000
{
    public class Startup
    {
        public Startup(IHostingEnvironment env)
        {
        var builder = new ConfigurationBuilder()
            .SetBasePath(env.ContentRootPath)
            .AddJsonFile("appsettings.json", optional: false, reloadOnChange:
true)
            .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional:
true);

        if (env.IsDevelopment())
        {

            builder.AddUserSecrets<Startup>();
        }

        builder.AddEnvironmentVariables();
        Configuration = builder.Build();
    }

    public IConfigurationRoot Configuration { get; }
```

```
    public void ConfigureServices(IServiceCollection services)
    {
        // Add framework services.
        services.AddDbContext<ApplicationDbContext>(options =>
            options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

        services.AddIdentity<ApplicationUser, IdentityRole>()
            .AddEntityFrameworkStores<ApplicationDbContext>()
            .AddDefaultTokenProviders();

        services.AddMvc();

        // Add application services.
        services.AddTransient<IEmailSender, AuthMessageSender>();
        services.AddTransient<ISmsSender, AuthMessageSender>();
    }

   public void Configure(IApplicationBuilder app, IHostingEnvironment env,
ILoggerFactory loggerFactory)
    {
        loggerFactory.AddConsole(Configuration.GetSection("Logging"));
        loggerFactory.AddDebug();

        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
            app.UseDatabaseErrorPage();
            app.UseBrowserLink();
        }
        else
        {
            app.UseExceptionHandler("/Home/Error");
        }

        app.UseDefaultFiles();
        app.UseStaticFiles();
        app.UseStaticFiles(new StaticFileOptions
        {
            FileProvider = new
PhysicalFileProvider(Path.Combine(env.ContentRootPath, "node_modules")),
            RequestPath = "/node_modules"
        });

        app.UseMvc(routes =>
        {
            routes.MapRoute(
                name: "default",
                template: "{controller=Home}/{action=Index}/{id?}");
        });
    }
}
}
```

tsConfig.json

```
    {
  "compilerOptions": {
    "diagnostics": true,
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
```

```
    "lib": [ "es2015", "dom" ],
    "listFiles": true,
    "module": "commonjs",
    "moduleResolution": "node",
    "noImplicitAny": true,
    "outDir": "wwwroot",
    "removeComments": false,
   "rootDir": "wwwroot",
    "sourceMap": true,
    "suppressImplicitAnyIndexErrors": true,
    "target": "es5"
  },
  "exclude": [
    "node_modules",
    "wwwroot/lib/"
  ]
}
```

Package.json

```
   {
 "name": "angular dependencies and web dev package",
 "version": "1.0.0",
 "description": "Angular 2 MVC. Samuel Maícas Template",
 "scripts": {},
 "dependencies": {
   "@angular/common": "~2.4.0",
   "@angular/compiler": "~2.4.0",
   "@angular/core": "~2.4.0",
   "@angular/forms": "~2.4.0",
   "@angular/http": "~2.4.0",
   "@angular/platform-browser": "~2.4.0",
   "@angular/platform-browser-dynamic": "~2.4.0",
   "@angular/router": "~3.4.0",
  "angular-in-memory-web-api": "~0.2.4",
   "systemjs": "0.19.40",
   "core-js": "^2.4.1",
   "rxjs": "5.0.1",
   "zone.js": "^0.7.4"
 },
 "devDependencies": {
   "del": "^2.2.2",
   "gulp": "^3.9.1",
   "gulp-concat": "^2.6.1",
   "gulp-cssmin": "^0.1.7",
   "gulp-htmlmin": "^3.0.0",
   "gulp-uglify": "^2.1.2",
   "merge-stream": "^1.0.1",
   "tslint": "^3.15.1",
   "typescript": "~2.0.10"
  },
  "repository": {}
}
```

bundleconfig.json

```
    [
  {
    "outputFileName": "wwwroot/css/site.min.css",
```

```
    "inputFiles": [
      "wwwroot/css/site.css"
    ]
  },
  {
    "outputFileName": "wwwroot/js/site.min.js",
    "inputFiles": [
      "wwwroot/js/site.js"
    ],
    "minify": {
      "enabled": true,
      "renameLocals": true
    },
    "sourceMap": false
  }
]
```

bundleconfig.json gulpfile (RightClick bundleconfig.json, Bundler & Minifier> Convert to Gulp) .

/ / Index.cshtml

```
    @{
    ViewData["Title"] = "Home Page";
}
<div>{{ nombre }}</div>
```

wwwroot https://github.com/angular/quickstart seed . : **index.html main.ts, systemjs-angular-loader.js, systemjs.config.js, tsconfig.json app**

wwwroot / Index.html

```
    <html>
  <head>
    <title>SMTemplate Angular2 & ASP.NET Core</title>
    <base href="/">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">


    <script src="node_modules/core-js/client/shim.min.js"></script>

    <script src="node_modules/zone.js/dist/zone.js"></script>
    <script src="node_modules/systemjs/dist/system.src.js"></script>

    <script src="systemjs.config.js"></script>
    <script>
      System.import('main.js').catch(function(err){ console.error(err); });
    </script>
  </head>

  <body>
    <my-app>Loading AppComponent here ...</my-app>
  </body>
</html>
```

templateUrl Controllers . wwwroot / app / app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'my-app',
  templateUrl: '/home/index',
 })
export class AppComponent  { nombre = 'Samuel Maícas'; }
```

[] Asp.Net Core + Visual Studio 2017 Angular2 + Gulp

1.
2. dotnet
3. npm install .

. .

https://github.com/SamML/CoreAngular000

MVC <-> 2

: ASP.NET HTML 2 / JS ANNUAL CALL CONTROLLER :

HTML View () .

```
 return File("~/html/About.html", "text/html");
```

HTML . . .

wwwroot / html / About.html

```
    <!DOCTYPE html>
<html>
  <head>
    <title>About Page</title>
    <base href="/">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
      <link href="../css/site.min.css" rel="stylesheet" type="text/css"/>

    <script src="../node_modules/core-js/client/shim.min.js"></script>

    <script src="../node_modules/zone.js/dist/zone.js"></script>
    <script src="../node_modules/systemjs/dist/system.src.js"></script>

    <script src="../systemjs.config.js"></script>
    <script>
      System.import('../main.js').catch(function(err){ console.error(err); });
    </script>
  </head>

  <body>
    <aboutpage>Loading AppComponent here ...</aboutpage>
  </body>
</html>
```

(*)

: ASP.NET Core Controller Angular2 MVC .

```
import { Component } from '@angular/core';

@Component({
  selector: 'aboutpage',
  templateUrl: '/home/about',
})
export class AboutComponent  {

}
```

ASP.net Angular 2 TypeScript : https://riptutorial.com/ko/angular2/topic/9543/asp-net----angular-2--typescript---

15: ChangeDetectionStrategy

Examples

OnPush

myInput someInternalValue . .

```
import {Component, Input} from '@angular/core';

@Component({
  template:`
  <div>
    <p>{{myInput}}</p>
    <p>{{someInternalValue}}</p>
  </div>
  `
})
class MyComponent {
  @Input() myInput: any;

  someInternalValue: any;

  // ...
}
```

changeDetection: ChangeDetectionStrategy.Default . . MyComponent .,myInput someInternalValue 2 someInternalValue .

. changeDetection: .ChangeDetectionStrategy.OnPush changeDetection:

```
import {Component, ChangeDetectionStrategy, Input} from '@angular/core';

@Component({
  changeDetection: ChangeDetectionStrategy.OnPush
  template:`
  <div>
    <p>{{myInput}}</p>
    <p>{{someInternalValue}}</p>
  </div>
  `
})
class MyComponent {
  @Input() myInput: any;

  someInternalValue: any;

  // ...
}
```

changeDetection: ChangeDetectionStrategy.OnPush MyComponent . myInput .

ChangeDetectionStrategy :

https://riptutorial.com/ko/angular2/topic/2644/changedetectionstrategy---

16: Dropzone 2

Examples

Dropzone 2 .

npm angular2-dropzone-wrapper . --save-dev

```
import { DropzoneModule } from 'angular2-dropzone-wrapper';
import { DropzoneConfigInterface } from 'angular2-dropzone-wrapper';

const DROPZONE_CONFIG: DropzoneConfigInterface = {
  // Change this to your upload POST address:
  server: 'https://example.com/post',
  maxFilesize: 10,
  acceptedFiles: 'image/*'
};

@NgModule({
  ...
  imports: [
    ...
    DropzoneModule.forRoot(DROPZONE_CONFIG)
  ]
})
```

Dropzone dropzone .

```
<dropzone [config]="config" [message]="'Click or drag images here to upload'"
(error)="onUploadError($event)" (success)="onUploadSuccess($event)"></dropzone>
```

```
import {Component} from '@angular/core';
@Component({
    selector: 'app-new-media',
    templateUrl: './dropzone.component.html',
    styleUrls: ['./dropzone.component.scss']
})
export class DropZoneComponent {


    onUploadError(args: any) {
        console.log('onUploadError:', args);
    }

    onUploadSuccess(args: any) {
        console.log('onUploadSuccess:', args);
    }
}
```

Dropzone 2 : https://riptutorial.com/ko/angular2/topic/10010/dropzone--2

17: EventEmitter

Examples

```
class EventEmitter extends Subject {
    constructor(isAsync?: boolean)
    emit(value?: T)
    subscribe(generatorOrNext?: any, error?: any, complete?: any) : any
}
```

```
@Component({
  selector: 'zippy',
  template: `
  <div class="zippy">
    <div (click)="toggle()">Toggle</div>
    <div [hidden]="!visible">
      <ng-content></ng-content>
    </div>
 </div>`})
export class Zippy {
  visible: boolean = true;
  @Output() open: EventEmitter<any> = new EventEmitter();
  @Output() close: EventEmitter<any> = new EventEmitter();
  toggle() {
    this.visible = !this.visible;
    if (this.visible) {
      this.open.emit(null);
    } else {
      this.close.emit(null);
    }
  }
}
```

Emmiting Events

```
<zippy (open)="onOpen($event)" (close)="onClose($event)"></zippy>
```

```
import {EventEmitter} from 'angular2/core';
export class NavService {
    navchange: EventEmitter<number> = new EventEmitter();
    constructor() {}
    emitNavChangeEvent(number) {
        this.navchange.emit(number);
    }
    getNavChangeEmitter() {
        return this.navchange;
    }
}
```

-

```
import {Component} from 'angular2/core';
```

```
import {NavService} from '../services/NavService';

@Component({
    selector: 'obs-comp',
    template: `obs component, item: {{item}}`
    })
    export class ObservingComponent {
    item: number = 0;
    subscription: any;
    constructor(private navService:NavService) {}
    ngOnInit() {
        this.subscription = this.navService.getNavChangeEmitter()
        .subscribe(item => this.selectedNavItem(item));
    }
    selectedNavItem(item: number) {
        this.item = item;
    }
    ngOnDestroy() {
        this.subscription.unsubscribe();
    }
}

@Component({
    selector: 'my-nav',
    template:`
        <div class="nav-item" (click)="selectedNavItem(1)">nav 1 (click me)</div>
        <div class="nav-item" (click)="selectedNavItem(2)">nav 2 (click me)</div>
    `,
})
export class Navigation {
    item = 1;
    constructor(private navService:NavService) {}
    selectedNavItem(item: number) {
        console.log('selected nav item ' + item);
        this.navService.emitNavChangeEvent(item);
    }
}
```

.

EventEmitter : https://riptutorial.com/ko/angular2/topic/9159/eventemitter-

18: HTTP

HttpServiceLayer Http .

HttpServiceLayer Http .

.

```
import { Http } from '@angular/http';
```

.

Examples

Http

```
import { Http, Request, RequestOptionsArgs, Response, RequestOptions, ConnectionBackend,
Headers } from '@angular/http';
import { Router } from '@angular/router';
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/observable/empty';
import 'rxjs/add/observable/throw';
import 'rxjs/add/operator/catch';
import { ApplicationConfiguration } from '../application-configuration/application-
configuration';

/**
* This class extends the Http class from angular and adds automaticaly the server URL(if in
development mode) and 2 headers by default:
* Headers added: 'Content-Type' and 'X-AUTH-TOKEN'.
* 'Content-Type' can be set in any othe service, and if set, it will NOT be overwritten in
this class any more.
*/
export class HttpServiceLayer extends Http {

  constructor(backend: ConnectionBackend, defaultOptions: RequestOptions, private _router:
Router, private appConfig: ApplicationConfiguration) {
    super(backend, defaultOptions);
  }

  request(url: string | Request, options?: RequestOptionsArgs): Observable<Response> {
    this.getRequestOptionArgs(options);
    return this.intercept(super.request(this.appConfig.getServerAdress() + url, options));
  }

  /**
  * This method checks if there are any headers added and if not created the headers map and
ads 'Content-Type' and 'X-AUTH-TOKEN'
  * 'Content-Type' is not overwritten if it is allready available in the headers map
  */
  getRequestOptionArgs(options?: RequestOptionsArgs): RequestOptionsArgs {
    if (options == null) {
      options = new RequestOptions();
    }
```

```
    if (options.headers == null) {
      options.headers = new Headers();
    }

    if (!options.headers.get('Content-Type')) {
      options.headers.append('Content-Type', 'application/json');
    }

    if (this.appConfig.getAuthToken() != null) {
      options.headers.append('X-AUTH-TOKEN', this.appConfig.getAuthToken());
    }

    return options;
  }

  /**
  * This method as the name sugests intercepts the request and checks if there are any errors.
  * If an error is present it will be checked what error there is and if it is a general one
then it will be handled here, otherwise, will be
  * thrown up in the service layers
  */
  intercept(observable: Observable<Response>): Observable<Response> {

    //  return observable;
    return observable.catch((err, source) => {
      if (err.status == 401) {
        this._router.navigate(['/login']);
        //return observable;
        return Observable.empty();
      } else {
        //return observable;
        return Observable.throw(err);
      }
    });
  }
}
```

Angular 's Http

Http Http .

, (,), :

```
export function httpServiceFactory(xhrBackend: XHRBackend, requestOptions: RequestOptions,
router: Router, appConfig: ApplicationConfiguration) {
  return  new HttpServiceLayer(xhrBackend, requestOptions, router, appConfig);
}

import { HttpModule, Http, Request, RequestOptionsArgs, Response, XHRBackend, RequestOptions,
ConnectionBackend, Headers } from '@angular/http';
import { Router } from '@angular/router';

@NgModule({
  declarations: [ ... ],
  imports: [ ... ],
  exports: [ ... ],
  providers: [
    ApplicationConfiguration,
    {
```

```
      provide: Http,
      useFactory: httpServiceFactory,
      deps: [XHRBackend, RequestOptions, Router, ApplicationConfiguration]
}
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

: ApplicationConfiguration .

HttpClient AuthToken (4.3)

```
import { Injectable } from '@angular/core';
import { HttpEvent, HttpHandler, HttpInterceptor, HttpRequest } from '@angular/common/http';
import { UserService } from '../services/user.service';
import { Observable } from 'rxjs/Observable';

@Injectable()
export class AuthHeaderInterceptor implements HttpInterceptor {

  constructor(private userService: UserService) {
  }

  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    if (this.userService.isAuthenticated()) {
      req = req.clone({
        setHeaders: {
          Authorization:  `Bearer ${this.userService.token}`
        }
      });
    }
    return next.handle(req);
  }
}
```

(some-module.module.ts)

```
{provide: HTTP_INTERCEPTORS, useClass: AuthHeaderInterceptor, multi: true},
```

HTTP : https://riptutorial.com/ko/angular2/topic/1413/http-

19: ngfor

`ngFor` Angular2 . iterable DOM iterable DOM .

Examples

```
<ul>
  <li *ngFor="let item of items">{{item.name}}</li>
</ul>
```

```
<div *ngFor="let item of items">
  <p>{{item.name}}</p>
  <p>{{item.price}}</p>
  <p>{{item.description}}</p>
</div>
```

```
<div *ngFor="let item of items; let i = index">
  <p>Item number: {{i}}</p>
</div>
```

. .

Angular2

Angular2 . :

-
-
-
-
-

`index` `Boolean` `Boolean` . .

```
<div *ngFor="let item of items; let firstItem = first; let lastItem = last">
  <p *ngIf="firstItem">I am the first item and I am gonna be showed</p>
  <p *ngIf="firstItem">I am not the first item and I will not show up :(</p>
  <p *ngIf="lastItem">But I'm gonna be showed as I am the last item :)</p>
</div>
```

*

```
import { Pipe, PipeTransform } from '@angular/core';
@Pipe({
  name: 'even'
})

export class EvenPipe implements PipeTransform {
    transform(value: string): string {
```

```
        if(value && value %2 === 0){
          return value;
        }
    }
}

@Component({
      selector: 'example-component',
      template: '<div>
                      <div *ngFor="let number of numbers | even">
                          {{number}}
                      </div>
                </div>'
})

export class exampleComponent {
    let numbers : List<number> = Array.apply(null, {length: 10}).map(Number.call, Number)
}
```

ngfor : https://riptutorial.com/ko/angular2/topic/8051/ngfor-

20: ngif

* **NgIf** : DOM . , DOM .

- <div * ngIf = "false"> </ div> <! - false ->
- <div * ngIf = "undefined"> </ div> <! - false ->
- <div * ngIf = "null"> </ div> <! - false . ->
- <div * ngIf = "0"> </ div> <! - false . ->
- <div * ngIf = "NaN"> </ div> <! - false ->
- <div * ngIf = ""> </ div> <! - false . ->
- true .

Examples

* ngIf . :

boolean .

```
loading: boolean = false;
```

`ngOnInit` .

```
ngOnInit() {
   this.loading = true;
}
```

boolean false .

```
 this.loading=false;
```

html `loading` * ngIf .

```
<div *ngIf="loading" class="progress">
   <div class="progress-bar info" style="width: 125%;"></div>
</div>
```

```
<p class="alert alert-success" *ngIf="names.length > 2">Currently there are more than 2
names!</p>
```

* ngFor * ngIf

NgFor .

- **index** - 0 iterable ()
- **first** - (boolean) iterable true
- **last** - (boolean) iterable true

- **even** - (boolean) true
- **odd** - (boolean) true

```
<div *ngFor="let note of csvdata; let i=index; let lastcall=last">
      <h3>{{i}}</h3> <-- to show index position
      <h3>{{note}}</h3>
      <span *ngIf="lastcall">{{anyfunction()}} </span><-- this lastcall boolean value will be
true only if this is last in loop
      // anyfunction() will run at the end of loop same way we can do at start
    </div>
```

* ngIf * ngFor .

*ngIf *ngFor ( ) *ngIf *ngFor .

1 :

```
<div *ngFor="let item of items; let i = index">
  <div *ngIf="<your condition here>">

   <!-- Execute code here if statement true -->

  </div>
</div>
```

2 :

```
<div *ngFor="let item of items; let i = index">
  <div *ngIf="i % 2 == 0">
    {{ item }}
  </div>
</div>
```

div .

div * ngFor use * ngIf div . * ngFor template .

```
<template ngFor let-item [ngForOf]="items">
    <div *ngIf="item.price > 100">
    </div>
</template>
```

div <template> DOM . DOM iterated div .

: Angular v4 <template> <ng-template> v5 . Angular v2.x <template> .

ngif : https://riptutorial.com/ko/angular2/topic/8346/ngif--

21: ngModel

ngModel Angular2 .

Examples

```
import { BrowserModule } from '@angular/platform-browser';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';

import { Component, DebugElement } from '@angular/core';
import { dispatchEvent } from "@angular/platform-browser/testing/browser_util";
import { TestBed, ComponentFixture} from '@angular/core/testing';
import {By} from "@angular/platform-browser";

import { MyComponentModule } from 'ng2-my-component';
import { MyComponent } from './my-component';

describe('MyComponent:',()=> {

  const template = `
    <div>
       <my-component type="text" [(ngModel)]="value" name="TestName" size="9" min="3" max="8"
placeholder="testPlaceholder" disabled=false required=false></my-component>
    </div>
  `;

  let fixture:any;
  let element:any;
  let context:any;

  beforeEach(() => {

      TestBed.configureTestingModule({
          declarations: [InlineEditorComponent],
          imports: [
            FormsModule,
            InlineEditorModule]
      });
      fixture = TestBed.overrideComponent(InlineEditorComponent, {
      set: {
        selector:"inline-editor-test",
        template: template
      }})
      .createComponent(InlineEditorComponent);
      context = fixture.componentInstance;
      fixture.detectChanges();
  });

  it('should change value of the component', () => {
        let input = fixture.nativeElement.querySelector("input");
        input.value = "Username";
        dispatchEvent(input, 'input');
        fixture.detectChanges();

        fixture.whenStable().then(() => {
```

```
            //this button dispatch event for save the text in component.value
            fixture.nativeElement.querySelectorAll('button')[0].click();
            expect(context.value).toBe("Username");
        });
    });
});
```

ngModel : https://riptutorial.com/ko/angular2/topic/8693/ngmodel-

22: ngrx

Ngrx Angular2 . . - [Redux] [1] - [RxJs] [2] : - Angular2 . (!) [1] : http://redux.js.org [2] : http : // reactivex. io / rxjs

Examples

: /

Redux / Ngrx .

- Redux .
- RxJs Observable .

, .

.

1. `User` `IUser` .
2. `Store` `User` .
3. `UserReducer`
4. `UserReducer`
5. `UserReducer` `UserReducer` `Store` .
6. `Store` .

Spoiler : Plunkr ( ) .

1) `IUser`

.

-
- UI ( )

`IUser` .

`user.interface.ts`

```
export interface IUser {
  // from server
  username: string;
  email: string;

  // for UI
  isConnecting: boolean;
  isConnected: boolean;
};
```

---

2) User

.
:

user.actions.ts

```
export const UserActions = {
  // when the user clicks on login button, before we launch the HTTP request
  // this will allow us to disable the login button during the request
  USR_IS_CONNECTING: 'USR_IS_CONNECTING',
  // this allows us to save the username and email of the user
  // we assume those data were fetched in the previous request
  USR_IS_CONNECTED: 'USR_IS_CONNECTED',

  // same pattern for disconnecting the user
  USR_IS_DISCONNECTING: 'USR_IS_DISCONNECTING',
  USR_IS_DISCONNECTED: 'USR_IS_DISCONNECTED'
};
```

, .

. .

- .
- catch `userService.login` .
- `userService.login` `dispatch` store `dispatch` . `user.isConnecting`
- HTTP (`setTimeout`)
- `HTTP` .

user.service.ts

```
@Injectable()
export class UserService {
  constructor(public store$: Store<AppState>) { }

  login(username: string) {
    // first, dispatch an action saying that the user's tyring to connect
    // so we can lock the button until the HTTP request finish
    this.store$.dispatch({ type: UserActions.USR_IS_CONNECTING });

    // simulate some delay like we would have with an HTTP request
    // by using a timeout
    setTimeout(() => {
      // some email (or data) that you'd have get as HTTP response
      let email = `${username}@email.com`;

      this.store$.dispatch({ type: UserActions.USR_IS_CONNECTED, payload: { username, email }
});
    }, 2000);
  }

  logout() {
    // first, dispatch an action saying that the user's tyring to connect
    // so we can lock the button until the HTTP request finish
```

```
    this.store$.dispatch({ type: UserActions.USR_IS_DISCONNECTING });

    // simulate some delay like we would have with an HTTP request
    // by using a timeout
    setTimeout(() => {
      this.store$.dispatch({ type: UserActions.USR_IS_DISCONNECTED });
    }, 2000);
  }
}
```

---

3) UserReducer

user.state.ts

```
export const UserFactory: IUser = () => {
  return {
    // from server
    username: null,
    email: null,

    // for UI
    isConnecting: false,
    isConnected: false,
    isDisconnecting: false
  };
};
```

---

4) UserReducer

2 .

-
- Action Action<{type: string, payload: any}>

: .

3 .

user.reducer.ts

```
export const UserReducer: ActionReducer<IUser> = (user: IUser, action: Action) => {
  if (user === null) {
    return userFactory();
  }

  // ...
}
```

factory (ES6) .

```
export const UserReducer: ActionReducer<IUser> = (user: IUser = UserFactory(), action: Action)
=> {
  // ...
}
```

. : ES6 Object.assign .

```
export const UserReducer: ActionReducer<IUser> = (user: IUser = UserFactory(), action: Action)
=> {
  switch (action.type) {
    case UserActions.USR_IS_CONNECTING:
      return Object.assign({}, user, { isConnecting: true });

    case UserActions.USR_IS_CONNECTED:
      return Object.assign({}, user, { isConnecting: false, isConnected: true, username:
action.payload.username });

    case UserActions.USR_IS_DISCONNECTING:
      return Object.assign({}, user, { isDisconnecting: true });

    case UserActions.USR_IS_DISCONNECTED:
      return Object.assign({}, user, { isDisconnecting: false, isConnected: false });

    default:
      return user;
  }
};
```

---

5) UserReducer UserReducer Store

app.module.ts

```
@NgModule({
    declarations: [
    AppComponent
    ],
    imports: [
    // angular modules
    // ...

    // declare your store by providing your reducers
    // (every reducer should return a default state)
    StoreModule.provideStore({
        user: UserReducer,
        // of course, you can put as many reducers here as you want
        // ...
    }),

    // other modules to import
    // ...
    ]
});
```

6) Store ▪

.

- UserComponent : **[ ]** @Input async . (user Observable<IUser> IUser).
- LoginComponent **[Smart component]** Store Observable user .

user.component.ts

```
@Component({
  selector: 'user',
  styles: [
    '.table { max-width: 250px; }',
    '.truthy { color: green; font-weight: bold; }',
    '.falsy { color: red; }'
  ],
  template: `
    <h2>User information :</h2>

    <table class="table">
      <tr>
        <th>Property</th>
        <th>Value</th>
      </tr>

      <tr>
        <td>username</td>
        <td [class.truthy]="user.username" [class.falsy]="!user.username">
          {{ user.username ? user.username : 'null' }}
        </td>
      </tr>

      <tr>
        <td>email</td>
        <td [class.truthy]="user.email" [class.falsy]="!user.email">
          {{ user.email ? user.email : 'null' }}
        </td>
      </tr>

      <tr>
        <td>isConnecting</td>
        <td [class.truthy]="user.isConnecting" [class.falsy]="!user.isConnecting">
          {{ user.isConnecting }}
        </td>
      </tr>

      <tr>
        <td>isConnected</td>
        <td [class.truthy]="user.isConnected" [class.falsy]="!user.isConnected">
          {{ user.isConnected }}
        </td>
      </tr>

      <tr>
        <td>isDisconnecting</td>
        <td [class.truthy]="user.isDisconnecting" [class.falsy]="!user.isDisconnecting">
          {{ user.isDisconnecting }}
        </td>
```

```
      </tr>
    </table>
  `
})
export class UserComponent {
  @Input() user;

  constructor() { }
}
```

login.component.ts

```
@Component({
  selector: 'login',
  template: `
    <form
      *ngIf="!(user | async).isConnected"
      #loginForm="ngForm"
      (ngSubmit)="login(loginForm.value.username)"
    >
      <input
        type="text"
        name="username"
        placeholder="Username"
        [disabled]="(user | async).isConnecting"
        ngModel
      >

      <button
        type="submit"
        [disabled]="(user | async).isConnecting || (user | async).isConnected"
      >Log me in</button>
    </form>

    <button
      *ngIf="(user | async).isConnected"
      (click)="logout()"
      [disabled]="(user | async).isDisconnecting"
    >Log me out</button>
  `
})
export class LoginComponent {
  public user: Observable<IUser>;

  constructor(public store$: Store<AppState>, private userService: UserService) {
      this.user = store$.select('user');
  }

  login(username: string) {
    this.userService.login(username);
  }

  logout() {
    this.userService.logout();
  }
}
```

Ngrx Redux RxJs Ngrx Ngrx RxJs . . Plunkr !

tho , !

ngrx : https://riptutorial.com/ko/angular2/topic/8086/ngrx

23: OrderBy

.

Examples

```
import {Pipe, PipeTransform} from '@angular/core';

@Pipe({
  name: 'orderBy',
  pure: false
})
export class OrderBy implements PipeTransform {

  value:string[] =[];

  static _orderByComparator(a:any, b:any):number{

    if(a === null || typeof a === 'undefined') a = 0;
    if(b === null || typeof b === 'undefined') b = 0;

    if((isNaN(parseFloat(a)) || !isFinite(a)) || (isNaN(parseFloat(b)) || !isFinite(b))){
      //Isn't a number so lowercase the string to properly compare
      if(a.toLowerCase() < b.toLowerCase()) return -1;
      if(a.toLowerCase() > b.toLowerCase()) return 1;
    }else{
      //Parse strings as numbers to compare properly
      if(parseFloat(a) < parseFloat(b)) return -1;
      if(parseFloat(a) > parseFloat(b)) return 1;
    }

    return 0; //equal each other
  }

  transform(input:any, config:string = '+'): any{

    //make a copy of the input's reference
    this.value = [...input];
    let value = this.value;

    if(!Array.isArray(value)) return value;

    if(!Array.isArray(config) || (Array.isArray(config) && config.length === 1)){
      let propertyToCheck:string = !Array.isArray(config) ? config : config[0];
      let desc = propertyToCheck.substr(0, 1) === '-';

      //Basic array
      if(!propertyToCheck || propertyToCheck === '-' || propertyToCheck === '+'){
        return !desc ? value.sort() : value.sort().reverse();
      }else {
        let property:string = propertyToCheck.substr(0, 1) === '+' ||
propertyToCheck.substr(0, 1) === '-'
          ? propertyToCheck.substr(1)
          : propertyToCheck;

        return value.sort(function(a:any,b:any){
```

```
          return !desc
            ? OrderBy._orderByComparator(a[property], b[property])
            : -OrderBy._orderByComparator(a[property], b[property]);
        });
      }
    } else {
      //Loop over property of the array in order and sort
      return value.sort(function(a:any,b:any){
        for(let i:number = 0; i < config.length; i++){
          let desc = config[i].substr(0, 1) === '-';
          let property = config[i].substr(0, 1) === '+' || config[i].substr(0, 1) === '-'
            ? config[i].substr(1)
            : config[i];

          let comparison = !desc
            ? OrderBy._orderByComparator(a[property], b[property])
            : -OrderBy._orderByComparator(a[property], b[property]);

          //Don't return 0 yet in case of needing to sort by next property
          if(comparison !== 0) return comparison;
        }

        return 0; //equal each other
      });
    }
  }
}
```

HTML -

```
<table>
   <thead>
    <tr>
     <th>First Name</th>
     <th>Last Name</th>
    <th>Age</th>
    </tr>
</thead>
<tbody>
   <tr *ngFor="let user of users | orderBy : ['firstName']>
       <td>{{user.firstName}}</td>
       <td>{{user.lastName}}</td>
       <td>{{user.age}}</td>
   </tr>
</tbody>
</table>
```

HTML -

```
<table>
   <thead>
    <tr>
     <th>First Name</th>
     <th>Last Name</th>
    <th>Age</th>
    </tr>
</thead>
```

```
<tbody>
   <tr *ngFor="let user of users | orderBy : ['-firstName']>
       <td>{{user.firstName}}</td>
       <td>{{user.lastName}}</td>
       <td>{{user.age}}</td>
   </tr>
</tbody>
</table>
```

OrderBy : https://riptutorial.com/ko/angular2/topic/8969/orderby-

24: ViewContainerRef.createComponent

Examples

. .

```
@Component({
  selector: 'dcl-wrapper',
  template: `<div #target></div>`
})
export class DclWrapper {
  @ViewChild('target', {
    read: ViewContainerRef
  }) target;
  @Input() type;
  cmpRef: ComponentRef;
  private isViewInitialized: boolean = false;

  constructor(private resolver: ComponentResolver) {}

  updateComponent() {
    if (!this.isViewInitialized) {
      return;
    }
    if (this.cmpRef) {
      this.cmpRef.destroy();
    }
    this.resolver.resolveComponent(this.type).then((factory: ComponentFactory < any > ) => {
      this.cmpRef = this.target.createComponent(factory)
        // to access the created instance use
        // this.cmpRef.instance.someProperty = 'someValue';
        // this.cmpRef.instance.someOutput.subscribe(val => doSomething());
    });
  }

  ngOnChanges() {
    this.updateComponent();
  }

  ngAfterViewInit() {
    this.isViewInitialized = true;
    this.updateComponent();
  }

  ngOnDestroy() {
    if (this.cmpRef) {
      this.cmpRef.destroy();
    }
  }
}
```

.

```
<dcl-wrapper [type]="someComponentType"></dcl-wrapper>
```

()

:

```
//our root app component
import {Component, NgModule, ViewChild, ViewContainerRef, ComponentFactoryResolver,
ComponentRef} from '@angular/core'
import {BrowserModule} from '@angular/platform-browser'
import {ChildComponent} from './childComp.ts'

@Component({
  selector: 'my-app',
  template: `
    <div>
      <h2>Hello {{name}}</h2>
      <input type="button" value="Click me to add element" (click) = addElement()> // call the
function on click of the button
      <div #parent> </div> // Dynamic component will be loaded here
    </div>
  `,
})
export class App {
  name:string;

  @ViewChild('parent', {read: ViewContainerRef}) target: ViewContainerRef;
  private componentRef: ComponentRef<any>;

  constructor(private componentFactoryResolver: ComponentFactoryResolver) {
    this.name = 'Angular2'
  }

  addElement(){
    let childComponent =
this.componentFactoryResolver.resolveComponentFactory(ChildComponent);
    this.componentRef = this.target.createComponent(childComponent);
  }
}
```

childComp.ts :

```
import{Component} from '@angular/core';

@Component({
  selector: 'child',
  template: `
    <p>This is Child</p>
  `,
})
export class ChildComponent {
  constructor(){

  }
}
```

app.module.ts :

```
@NgModule({
```

```
  imports: [ BrowserModule ],
  declarations: [ App, ChildComponent ],
  bootstrap: [ App ],
  entryComponents: [ChildComponent] // define the dynamic component here in module.ts
})
export class AppModule {}
```

Angular2 html

.

WidgetComponent , ChartWidget PatientWidget .

ChartWidget.ts

```
@Component({
selector: 'chart-widget',
templateUrl: 'chart-widget.component.html',
providers: [{provide: WidgetComponent, useExisting: forwardRef(() => ChartWidget) }]
})

export class ChartWidget extends WidgetComponent implements OnInit {
       constructor(ngEl: ElementRef, renderer: Renderer) {
    super(ngEl, renderer);
    }
    ngOnInit() {}
     close(){
      console.log('close');
    }
    refresh(){
      console.log('refresh');
    }
    ...
}
```

chart-widget.compoment.html (primeng Panel)

```
<p-panel [style]="{'margin-bottom':'20px'}">
    <p-header>
        <div class="ui-helper-clearfix">
           <span class="ui-panel-title" style="font-size:14px;display:inline-block;margin-
top:2px">Chart Widget</span>
            <div class="ui-toolbar-group-right">
               <button pButton type="button" icon="fa-window-minimize"
(click)="minimize()"</button>
              <button pButton type="button" icon="fa-refresh" (click)="refresh()"></button>
              <button pButton type="button"  icon="fa-expand" (click)="expand()" ></button>
             <button pButton type="button" (click)="close()" icon="fa-window-close"></button>
                   </div>
               </div>
    </p-header>
      some data
</p-panel>
```

DataWidget.ts

```
@Component({
    selector: 'data-widget',
    templateUrl: 'data-widget.component.html',
    providers: [{provide: WidgetComponent, useExisting: forwardRef(() =>DataWidget) }]
    })

export class DataWidget extends WidgetComponent implements OnInit {
       constructor(ngEl: ElementRef, renderer: Renderer) {
    super(ngEl, renderer);
    }
    ngOnInit() {}
    close(){
      console.log('close');
    }
    refresh(){
      console.log('refresh');
    }
    ...
}
```

data-widget.compoment.html (primeng Panel)

WidgetComponent.ts

```
@Component({
  selector: 'widget',
  template: '<ng-content></ng-content>'
})
export  class WidgetComponent{
}
```

. ,

```
@Component({

    selector: 'dynamic-component',
    template: `<div #container><ng-content></ng-content></div>`

})
export class DynamicComponent {
@ViewChild('container', {read: ViewContainerRef}) container: ViewContainerRef;

    public addComponent(ngItem: Type<WidgetComponent>): WidgetComponent {
    let factory = this.compFactoryResolver.resolveComponentFactory(ngItem);
    const ref = this.container.createComponent(factory);
    const newItem: WidgetComponent = ref.instance;
    this._elements.push(newItem);
    return newItem;
  }
}
```

. app.component.ts

```
@Component({
  selector: 'app-root',
  templateUrl: './app/app.component.html',
  styleUrls: ['./app/app.component.css'],
```

```
  entryComponents: [ChartWidget,  DataWidget],
})

export class AppComponent {
   private elements: Array<WidgetComponent>=[];
   private WidgetClasses = {
      'ChartWidget': ChartWidget,
      'DataWidget': DataWidget
  }
  @ViewChild(DynamicComponent) dynamicComponent:DynamicComponent;

   addComponent(widget: string ): void{
     let ref= this.dynamicComponent.addComponent(this.WidgetClasses[widget]);
     this.elements.push(ref);
     console.log(this.elements);

     this.dynamicComponent.resetContainer();
  }
}
```

app.component.html

```
<button (click)="addComponent('ChartWidget')">Add ChartWidget</button>
<button (click)="addComponent('DataWidget')">Add DataWidget</button>

<dynamic-component [hidden]="true" ></dynamic-component>

<hr>
Dynamic Components
<hr>
<widget *ngFor="let item of elements">
    <div>{{item}}</div>
   <div [innerHTML]="item._ngEl.nativeElement.innerHTML | sanitizeHtml">
   </div>
</widget>
```

https://plnkr.co/edit/lugU2pPsSBd3XhPHiUP1?p=preview

@yurzui .

view.directive.ts

'@ angular / core' {ViewRef, Directive, Input, ViewContainerRef} ;

```
@Directive({
    selector: '[view]'
})
export class ViewDirective {
  constructor(private vcRef: ViewContainerRef) {}

  @Input()
  set view(view: ViewRef) {
    this.vcRef.clear();
    this.vcRef.insert(view);
  }

  ngOnDestroy() {
    this.vcRef.clear()
```

```
  }
}
```

app.component.ts

```
private elements: Array<{ view: ViewRef, component: WidgetComponent}> = [];

...
addComponent(widget: string ): void{
  let component = this.dynamicComponent.addComponent(this.WidgetClasses[widget]);
  let view: ViewRef = this.dynamicComponent.container.detach(0);
  this.elements.push({view,component});

  this.dynamicComponent.resetContainer();
}
```

app.component.html

```
<widget *ngFor="let item of elements">
  <ng-container *view="item.view"></ng-container>
</widget>
```

https://plnkr.co/edit/JHpIHR43SvJd0OxJVMfV?p=preview

ViewContainerRef.createComponent : https://riptutorial.com/ko/angular2/topic/831/viewcontainerref-createcomponent-----

25: Visual Studio Angular2

Examples

Launch.json

1. ->
2. launch.json . launch.json . launch.json // new launch.json
   .

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Launch Extension",
            "type": "extensionHost",
            "request": "launch",
            "runtimeExecutable": "${execPath}",
            "args": [
                "--extensionDevelopmentPath=${workspaceRoot}"
            ],
            "stopOnEntry": false,
            "sourceMaps": true,
            "outDir": "${workspaceRoot}/out",
            "preLaunchTask": "npm"
        }
    ]
}
```

launch.json .
launch.json
** // main.js **

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Launch",
            "type": "node",
            "request": "launch",
            "program": "${workspaceRoot}/app/main.js", // put your main.js path
            "stopOnEntry": false,
            "args": [],
            "cwd": "${workspaceRoot}",
            "preLaunchTask": null,
            "runtimeExecutable": null,
            "runtimeArgs": [
                "--nolazy"
            ],
            "env": {
                "NODE_ENV": "development"
            },
```

```
            "console": "internalConsole",
            "sourceMaps": false,
            "outDir": null
        },
        {
            "name": "Attach",
            "type": "node",
            "request": "attach",
            "port": 5858,
            "address": "localhost",
            "restart": false,
            "sourceMaps": false,
            "outDir": null,
            "localRoot": "${workspaceRoot}",
            "remoteRoot": null
        },
        {
            "name": "Attach to Process",
            "type": "node",
            "request": "attach",
            "processId": "${command.PickProcess}",
            "port": 5858,
            "sourceMaps": false,
            "outDir": null
        }
    ]
}
```

3. .

Visual Studio Angular2 : https://riptutorial.com/ko/angular2/topic/7139/visual-studio---angular2-----

26: Webpack Angular2

Examples

2

webpack.config.js

```
const webpack = require("webpack")
const helpers = require('./helpers')
const path = require("path")
const WebpackNotifierPlugin = require('webpack-notifier');

module.exports = {

    // set entry point for your app module
    "entry": {
        "app": helpers.root("app/main.module.ts"),
    },

    // output files to dist folder
    "output": {
        "filename": "[name].js",
        "path": helpers.root("dist"),
        "publicPath": "/",
    },

    "resolve": {
        "extensions": ['.ts', '.js'],
    },

    "module": {
        "rules": [
            {
                "test": /\.ts$/,
                "loaders": [
                    {
                        "loader": 'awesome-typescript-loader',
                        "options": {
                            "configFileName": helpers.root("./tsconfig.json")
                        }
                    },
                    "angular2-template-loader"
                ]
            },
        ],
    },

    "plugins": [

        // notify when build is complete
        new WebpackNotifierPlugin({title: "build complete"}),

        // get reference for shims
        new webpack.DllReferencePlugin({
            "context": helpers.root("src/app"),
```

```
            "manifest": helpers.root("config/polyfills-manifest.json")
        }),

        // get reference of vendor DLL
        new webpack.DllReferencePlugin({
            "context": helpers.root("src/app"),
            "manifest": helpers.root("config/vendor-manifest.json")
        }),

        // minify compiled js
        new webpack.optimize.UglifyJsPlugin(),
    ],
}
```

vendor.config.js

```
const webpack = require("webpack")
const helpers = require('./helpers')
const path = require("path")

module.exports = {
    // specify vendor file where all vendors are imported
    "entry": {
        // optionally add your shims as well
        "polyfills": [helpers.root("src/app/shims.ts")],
        "vendor": [helpers.root("src/app/vendor.ts")],
    },

    // output vendor to dist
    "output": {
        "filename": "[name].js",
        "path": helpers.root("dist"),
        "publicPath": "/",
        "library": "[name]"
    },

    "resolve": {
        "extensions": ['.ts', '.js'],
    },

    "module": {
        "rules": [
            {
                "test": /\.ts$/,
                "loaders": [
                    {
                        "loader": 'awesome-typescript-loader',
                        "options": {
                            "configFileName": helpers.root("./tsconfig.json")
                        }
                    },
                ]
            },
        ],
    },

    "plugins": [

        // create DLL for entries
        new webpack.DllPlugin({
```

```
            "name": "[name]",
            "context": helpers.root("src/app"),
            "path": helpers.root("config/[name]-manifest.json")
        }),

        // minify generated js
        new webpack.optimize.UglifyJsPlugin(),
    ],
}
```

helpers.js

```
var path = require('path');

var _root = path.resolve(__dirname, '..');

function root(args) {
  args = Array.prototype.slice.call(arguments, 0);
  return path.join.apply(path, [_root].concat(args));
}

exports.root = root;
```

vendor.ts

```
import "@angular/platform-browser"
import "@angular/platform-browser-dynamic"
import "@angular/core"
import "@angular/common"
import "@angular/http"
import "@angular/router"
import "@angular/forms"
import "rxjs"
```

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Angular 2 webpack</title>

    <script src="/dist/vendor.js" type="text/javascript"></script>
    <script src="/dist/app.js" type="text/javascript"></script>
</head>
<body>
    <app>loading...</app>
</body>
</html>
```

package.json

```
{
  "name": "webpack example",
  "version": "0.0.0",
  "description": "webpack",
  "scripts": {
```

```
    "build:webpack": "webpack --config config/webpack.config.js",
    "build:vendor": "webpack --config config/vendor.config.js",
    "watch": "webpack --config config/webpack.config.js --watch"
  },
  "devDependencies": {
    "@angular/common": "2.4.7",
    "@angular/compiler": "2.4.7",
    "@angular/core": "2.4.7",
    "@angular/forms": "2.4.7",
    "@angular/http": "2.4.7",
    "@angular/platform-browser": "2.4.7",
    "@angular/platform-browser-dynamic": "2.4.7",
    "@angular/router": "3.4.7",
    "webpack": "^2.2.1",
    "awesome-typescript-loader": "^3.1.2",
  },
  "dependencies": {
  }
}
```

Webpack Angular2 : https://riptutorial.com/ko/angular2/topic/9554/webpack--angular2

27: Zone.js

Examples

NgZone

`NgZone` DI (Dependency Injection) .

my.component.ts

```
import { Component, NgOnInit, NgZone } from '@angular/core';

@Component({...})
export class Mycomponent implements NgOnInit {
  constructor(private _ngZone: NgZone) { }
  ngOnInit() {
    this._ngZone.runOutsideAngular(() => {
      // Do something outside Angular so it won't get noticed
    });
  }
}
```

NgZone HTTP

`runOutsideAngular` 2 . , HTTP . Angular 2 `NgZone run NgZone` .

my.component.ts

```
import { Component, OnInit, NgZone } from '@angular/core';
import { Http } from '@angular/http';

@Component({...})
export class Mycomponent implements OnInit {
  private data: any[];
  constructor(private http: Http, private _ngZone: NgZone) { }
  ngOnInit() {
    this._ngZone.runOutsideAngular(() => {
      this.http.get('resource1').subscribe((data1:any) => {
        // First response came back, so its data can be used in consecutive request
        this.http.get(`resource2?id=${data1['id']}`).subscribe((data2:any) => {
          this.http.get(`resource3?id1=${data1['id']}&id2=${data2}`).subscribe((data3:any) =>
{
            this._ngZone.run(() => {
              this.data = [data1, data2, data3];
            });
          });
        });
      });
    });
  }
}
```

Zone.js : https://riptutorial.com/ko/angular2/topic/4184/zone-js

28: - ForLoop

1. <div * ngFor = " ; let i = index"> {{i}} {{item}} </ div>

`*ngFor` html .

`@View` . `@Component template` 'templateUrl' .

Examples

2 for-loop

plnkr ...

```
<!doctype html>
<html>
<head>
    <title>ng for loop in angular 2 with ES5.</title>
    <script type="text/javascript" src="https://code.angularjs.org/2.0.0-
alpha.28/angular2.sfx.dev.js"></script>
    <script>
        var ngForLoop = function () {
            this.msg = "ng for loop in angular 2 with ES5.";
            this.users = ["Anil Singh", "Sunil Singh", "Sushil Singh", "Aradhya", 'Reena'];
        };

        ngForLoop.annotations = [
                new angular.Component({
                    selector: 'ngforloop'
                }),
                new angular.View({
                    template: '<H1>{{msg}}</H1>' +
                            '<p> User List : </p>' +
                            '<ul>' +
                            '<li *ng-for="let user of users">' +
                            '{{user}}' +
                            '</li>' +
                            '</ul>',
                    directives: [angular.NgFor]
                })
        ];

        document.addEventListener("DOMContentLoaded", function () {
            angular.bootstrap(ngForLoop);
        });
    </script>
</head>
<body>
    <ngforloop></ngforloop>
    <h2>
      <a href="http://www.code-sample.com/" target="_blank">For more detail...</a>
    </h2>
</body>
</html>
```

NgFor -

NgFor . .

NgFor **trackBy** . **trackBy** index item . **trackBy** .

```
<li *ngFor="let item of items; let i = index; trackBy: trackByFn">
    {{i}} - {{item.name}}
</li>
```

: NgFor .

- .
- .
- **last** .
- .
- .

* ngFor

```
<table>
    <thead>
        <th>Name</th>
        <th>Index</th>
    </thead>
    <tbody>
        <tr *ngFor="let hero of heroes">
            <td>{{hero.name}}</td>
        </tr>
    </tbody>
</table>
```

* ngFor

```
   @Component({
     selector: 'main-component',
     template: '<example-component
                   *ngFor="let hero of heroes"
                   [hero]="hero"></example-component>'
   })


   @Component({
      selector: 'example-component',
      template: '<div>{{hero?.name}}</div>'
   })

   export class ExampleComponent {
     @Input() hero : Hero = null;
   }
```

* ng X

: 5 :

```
<div *ngFor="let item of items; let i = index">
  <div *ngIf="i % 5 == 0" class="row">
    {{ item }}
    <div *ngIf="i + 1 < items.length">{{ items[i + 1] }}</div>
    <div *ngIf="i + 2 < items.length">{{ items[i + 2] }}</div>
    <div *ngIf="i + 3 < items.length">{{ items[i + 3] }}</div>
    <div *ngIf="i + 4 < items.length">{{ items[i + 4] }}</div>
  </div>
</div>
```

- ForLoop : https://riptutorial.com/ko/angular2/topic/6543/---forloop

29: 2 -

Examples

Navbar

navbar.html 3 . (, ,)

```
<nav class="navbar navbar-default" role="navigation">
<ul class="nav navbar-nav">
  <li>
    <a id="home-navbar" routerLink="/home">Home</a>
  </li>
  <li>
    <a id="list-navbar" routerLink="/create" >List</a>
  </li>
  <li>
    <a id="create-navbar" routerLink="/create">Create</a>
  </li>
</ul>
```

navbar.e2e-spec.ts .

```
describe('Navbar', () => {

  beforeEach(() => {
    browser.get('home'); // before each test navigate to home page.
  });

  it('testing Navbar', () => {
    browser.sleep(2000).then(function(){
      checkNavbarTexts();
      navigateToListPage();
    });
  });

  function checkNavbarTexts(){
    element(by.id('home-navbar')).getText().then(function(text){ // Promise
      expect(text).toEqual('Home');
    });

    element(by.id('list-navbar')).getText().then(function(text){ // Promise
      expect(text).toEqual('List');
    });

    element(by.id('create-navbar')).getText().then(function(text){ // Promise
      expect(text).toEqual('Create');
    });
  }

  function navigateToListPage(){
    element(by.id('list-home')).click().then(function(){ // first find list-home a tag and
than click
        browser.sleep(2000).then(function(){
          browser.getCurrentUrl().then(function(actualUrl){ // promise
```

```
            expect(actualUrl.indexOf('list') !== -1).toBeTruthy(); // check the current url is
list
          });
        });

    });
  }
});
```

Angular2 -

cmd follows .

- `npm install -g protractor`
- `webdriver-manager update`
- `webdriver-manager start`

** protractor.conf.js .

useAllAngular2AppRoots : true .

```
  const config = {
  baseUrl: 'http://localhost:3000/',

  specs: [
      './dev/**/*.e2e-spec.js'
  ],

  exclude: [],
  framework: 'jasmine',

  jasmineNodeOpts: {
    showColors: true,
    isVerbose: false,
    includeStackTrace: false
  },

  directConnect: true,

  capabilities: {
    browserName: 'chrome',
    shardTestFiles: false,
    chromeOptions: {
      'args': ['--disable-web-security ','--no-sandbox', 'disable-extensions', 'start-
maximized', 'enable-crash-reporter-for-testing']
    }
  },

  onPrepare: function() {
    const SpecReporter = require('jasmine-spec-reporter');
    // add jasmine spec reporter
    jasmine.getEnv().addReporter(new SpecReporter({ displayStacktrace: true }));

    browser.ignoreSynchronization = false;
  },
  useAllAngular2AppRoots: true
};
```

```
if (process.env.TRAVIS) {
  config.capabilities = {
    browserName: 'firefox'
  };
}

exports.config = config;
```

dev .

```
describe('basic test', () => {

  beforeEach(() => {
    browser.get('http://google.com');
  });

  it('testing basic test', () => {
    browser.sleep(2000).then(function(){
      browser.getCurrentUrl().then(function(actualUrl){
        expect(actualUrl.indexOf('google') !== -1).toBeTruthy();
      });
    });
  });
});
```

cmd

```
protractor conf.js
```

2 - : https://riptutorial.com/ko/angular2/topic/8900/-2---

30: 2

`this.myForm = this.formBuilder.group`

this.myForm .

`'loginCredentials': this.formBuilder.group`

formControlName . `login` value `['', Validators.required],` secons `'email': ['', [Validators.required, customValidator]],` `'email': ['', [Validators.required, customValidator]],`

```
'hobbies': this.formBuilder.array
```

formGroupName .

```
<div *ngFor="let hobby of myForm.find('hobbies').controls; let i = index">
   <div formGroupName="{{i}}">...</div>
</div>
```

```
onAddHobby() {
    (<FormArray>this.myForm.find('hobbies')).push(new FormGroup({
      'hobby': new FormControl('', Validators.required)
    }))
  }
```

formGroup . . `<FormArray>`

```
removeHobby(index: number){
    (<FormArray>this.myForm.find('hobbies')).removeAt(index);
  }
```

.

Examples

```
    import {Component, OnInit} from '@angular/core';
import {
    FormGroup,
    FormControl,
    FORM_DIRECTIVES,
    REACTIVE_FORM_DIRECTIVES,
    Validators,
    FormBuilder,
    FormArray
} from "@angular/forms";
import {Control} from "@angular/common";
```

```
@Component({
  moduleId: module.id,
  selector: 'app-data-driven-form',
  templateUrl: 'data-driven-form.component.html',
  styleUrls: ['data-driven-form.component.css'],
  directives: [FORM_DIRECTIVES, REACTIVE_FORM_DIRECTIVES]
})
export class DataDrivenFormComponent  implements OnInit {
  myForm: FormGroup;

  constructor(private formBuilder: FormBuilder) {}

  ngOnInit() {
    this.myForm = this.formBuilder.group({
      'loginCredentials': this.formBuilder.group({
        'login': ['', Validators.required],
        'email': ['',  [Validators.required, customValidator]],
        'password': ['',  Validators.required]
      }),
      'hobbies': this.formBuilder.array([
        this.formBuilder.group({
          'hobby': ['', Validators.required]
        })
      ])
    });
  }

  removeHobby(index: number){
    (<FormArray>this.myForm.find('hobbies')).removeAt(index);
  }

  onAddHobby() {
    (<FormArray>this.myForm.find('hobbies')).push(new FormGroup({
      'hobby': new FormControl('', Validators.required)
    }))
  }

  onSubmit() {
    console.log(this.myForm.value);
  }
}

function customValidator(control: Control): {[s: string]: boolean} {
  if(!control.value.match("[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-
]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?")) {
    return {error: true}
  }
}
```

HTML

```
<h3>Register page</h3>
<form [formGroup]="myForm" (ngSubmit)="onSubmit()">
  <div formGroupName="loginCredentials">
    <div class="form-group">
      <div>
        <label for="login">Login</label>
        <input  id="login" type="text" class="form-control" formControlName="login">
      </div>
      <div>
```

```
        <label for="email">Email</label>
        <input  id="email" type="text" class="form-control"  formControlName="email">
      </div>
      <div>
        <label for="password">Password</label>
        <input  id="password" type="text" class="form-control"  formControlName="password">
      </div>
    </div>
  </div>
  <div class="row" >
    <div  formGroupName="hobbies">
      <div class="form-group">
        <label>Hobbies array:</label>
        <div  *ngFor="let hobby of myForm.find('hobbies').controls; let i = index">
          <div formGroupName="{{i}}">
            <input id="hobby_{{i}}" type="text" class="form-control"  formControlName="hobby">
            <button *ngIf="myForm.find('hobbies').length > 1"
(click)="removeHobby(i)">x</button>
          </div>
        </div>
        <button (click)="onAddHobby()">Add hobby</button>
      </div>
    </div>
  </div>
  <button type="submit" [disabled]="!myForm.valid">Submit</button>
</form>
```

2 : https://riptutorial.com/ko/angular2/topic/6463/-2---

31: 2

Examples

:

```
import {Component} from '@angular/core';

@Component({
  selector: 'parent-component',
  templateUrl: './parent-component.html'
})
export class ParentComponent {
  users : Array<User> = [];
  changeUsersActivation(user : User){
    user.changeButtonState();
  }
  constructor(){
    this.users.push(new User('Narco', false));
    this.users.push(new User('Bombasto',false));
    this.users.push(new User('Celeritas', false));
    this.users.push(new User('Magneta', false));
  }
}


export class User {
  firstName : string;
  active : boolean;

  changeButtonState(){
    this.active = !this.active;
  }
  constructor(_firstName :string, _active : boolean){
    this.firstName = _firstName;
    this.active = _active;
  }

}
```

HTML :

```
<div>
  <child-component [usersDetails]="users"
                   (changeUsersActivation)="changeUsersActivation($event)">
   </child-component>
</div>
```

:

```
import {Component, Input, EventEmitter, Output} from '@angular/core';
import {User} from "./parent.component";
```

```
@Component({
  selector: 'child-component',
  templateUrl: './child-component.html',
  styles: [`
     .btn {
      height: 30px;
      width: 100px;
      border: 1px solid rgba(0, 0, 0, 0.33);
      border-radius: 3px;
      margin-bottom: 5px;
     }

  `]
})
export class ChildComponent{
  @Input() usersDetails : Array<User> = null;
  @Output() changeUsersActivation = new EventEmitter();

  triggerEvent(user : User){
    this.changeUsersActivation.emit(user);
  }
}
```

HTML :

```
<div>
  <div>
    <table>
      <thead>
        <tr>
          <th>Name</th>
          <th></th>
        </tr>
      </thead>
      <tbody *ngIf="user !== null">
        <tr *ngFor="let user of usersDetails">
          <td>{{user.firstName}}</td>
          <td><button class="btn" (click)="triggerEvent(user)">{{user.active}}</button></td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

2 : https://riptutorial.com/ko/angular2/topic/8971/-2-----

32: 2

Examples

Angular 2 Jasmine . Jasmine .

`jasmine-core` (`jasmine`) .

```
npm install jasmine-core --save-dev --save-exact
```

Jasmine `./src/unit-tests.html` .

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="content-type" content="text/html;charset=utf-8">
  <title>Ng App Unit Tests</title>
  <link rel="stylesheet" href="../node_modules/jasmine-core/lib/jasmine-core/jasmine.css">
  <script src="../node_modules/jasmine-core/lib/jasmine-core/jasmine.js"></script>
  <script src="../node_modules/jasmine-core/lib/jasmine-core/jasmine-html.js"></script>
  <script src="../node_modules/jasmine-core/lib/jasmine-core/boot.js"></script>
</head>
<body>
   <!-- Unit Testing Chapter #1: Proof of life.  -->
   <script>
     it('true is true', function () {
       expect(true).toEqual(true);
     });
   </script>
</body>
</html>
```

Gulp, Webpack, Karma Jasmine

Webpack Webpack . ES6 (bavelle) . Typescript . Pug (Jade) .

.

webpack :

```
const webpack = require("webpack");
let packConfig = {
    entry: {},
    output: {},
    plugins:[
        new webpack.DefinePlugin({
            ENVIRONMENT: JSON.stringify('test')
        })
    ],
```

```
    module: {
       loaders: [
        {
            test: /\.js$/,
            exclude:/(node_modules|bower_components)/,
            loader: "babel",
            query:{
                presets:["es2015", "angular2"]
            }
        },
        {
            test: /\.woff2?$|\.ttf$|\.eot$|\.svg$/,
            loader: "file"
        },
        {
            test: /\.scss$/,
            loaders: ["style", "css", "sass"]
        },
        {
            test: /\.pug$/,
            loader: 'pug-html-loader'
        },
        ]
    },
    devtool : 'inline-cheap-source-map'
};
module.exports = packConfig;
```

webpack config karma.config.js .

```
const packConfig = require("./webpack.config.js");
module.exports = function (config) {
    config.set({
    basePath: '',
    frameworks: ['jasmine'],
    exclude:[],
    files: [
        {pattern: './karma.shim.js', watched: false}
    ],

    preprocessors: {
        "./karma.shim.js":["webpack"]
    },
    webpack: packConfig,

    webpackServer: {noInfo: true},

    port: 9876,

    colors: true,

    logLevel: config.LOG_INFO,

    browsers: ['PhantomJS'],

    concurrency: Infinity,

    autoWatch: false,
    singleRun: true
});
```

```
};
```

, , **karma.shim.js** . webpack . webpack **import require** .

karma.shim.js .

```
// Start of ES6 Specific stuff
import "es6-shim";
import "es6-promise";
import "reflect-metadata";
// End of ES6 Specific stuff

import "zone.js/dist/zone";
import "zone.js/dist/long-stack-trace-zone";
import "zone.js/dist/jasmine-patch";
import "zone.js/dist/async-test";
import "zone.js/dist/fake-async-test";
import "zone.js/dist/sync-test";
import "zone.js/dist/proxy-zone";

import 'rxjs/add/operator/map';
import 'rxjs/add/observable/of';

Error.stackTraceLimit = Infinity;

import {TestBed} from "@angular/core/testing";
import { BrowserDynamicTestingModule, platformBrowserDynamicTesting} from "@angular/platform-
browser-dynamic/testing";

TestBed.initTestEnvironment(
     BrowserDynamicTestingModule,
     platformBrowserDynamicTesting());

let testContext = require.context('../src/app', true, /\.spec\.js/);
testContext.keys().forEach(testContext);
```

TestBed . **src / app** **.spec.js** testContext .

. Personat . **src / test** . karma.shim.js .

. ? , karma.config.js :

```
gulp.task("karmaTests",function(done){
    var Server = require("karma").Server;
    new Server({
        configFile : "./karma.config.js",
        singleRun: true,
        autoWatch: false
    }, function(result){
        return result ? done(new Error(`Karma failed with error code ${result}`)):done();
    }).start();
});
```

config , . . .

Angular 2 "Tour of Heroes" .

```
import {
    TestBed,
    ComponentFixture,
    async
} from "@angular/core/testing";

import {AppComponent} from "./app.component";
import {AppModule} from "./app.module";
import Hero from "./hero/hero";

describe("App Component", function () {

    beforeEach(()=> {
        TestBed.configureTestingModule({
            imports: [AppModule]
        });

        this.fixture = TestBed.createComponent(AppComponent);
        this.fixture.detectChanges();
    });

    it("Should have a title", async(()=> {
        this.fixture.whenStable().then(()=> {
            expect(this.fixture.componentInstance.title).toEqual("Tour of Heros");
        });
    }));

    it("Should have a hero", async(()=> {
        this.fixture.whenStable().then(()=> {
            expect(this.fixture.componentInstance.selectedHero).toBeNull();
        });
    }));

    it("Should have an array of heros", async(()=>
        this.fixture.whenStable().then(()=> {
            const cmp = this.fixture.componentInstance;
            expect(cmp.heroes).toBeDefined("component should have a list of heroes");
            expect(cmp.heroes.length).toEqual(10, "heroes list should have 10 members");
            cmp.heroes.map((h, i)=> {
                expect(h instanceof Hero).toBeTruthy(`member ${i} is not a Hero instance.
${h}`)
            });
        })));

        it("Should have one list item per hero", async(()=>
        this.fixture.whenStable().then(()=> {
            const ul = this.fixture.nativeElement.querySelector("ul.heroes");
            const li = Array.prototype.slice.call(
                this.fixture.nativeElement.querySelectorAll("ul.heroes>li"));
            const cmp = this.fixture.componentInstance;
            expect(ul).toBeTruthy("There should be an unnumbered list for heroes");
            expect(li.length).toEqual(cmp.heroes.length, "there should be one li for each
hero");
            li.forEach((li, i)=> {
                expect(li.querySelector("span.badge"))
                    .toBeTruthy(`hero ${i} has to have a span for id`);
                expect(li.querySelector("span.badge").textContent.trim())
                    .toEqual(cmp.heroes[i].id.toString(), `hero ${i} had wrong id displayed`);
                expect(li.textContent)
                    .toMatch(cmp.heroes[i].name, `hero ${i} has wrong name displayed`);
            });
```

```
        }));

    it("should have correct styling of hero items", async(()=>
        this.fixture.whenStable().then(()=> {
            const hero = this.fixture.nativeElement.querySelector("ul.heroes>li");
            const win = hero.ownerDocument.defaultView ||hero.ownerDocument.parentWindow;
            const styles = win.getComputedStyle(hero);
            expect(styles["cursor"]).toEqual("pointer", "cursor should be pointer on hero");
            expect(styles["borderRadius"]).toEqual("4px", "borderRadius should be 4px");
        }));

    it("should have a click handler for hero items",async(()=>
        this.fixture.whenStable().then(()=>{
            const cmp = this.fixture.componentInstance;
            expect(cmp.onSelect)
                .toBeDefined("should have a click handler for heros");
            expect(this.fixture.nativeElement.querySelector("input.heroName"))
                .toBeNull("should not show the hero details when no hero has been selected");
            expect(this.fixture.nativeElement.querySelector("ul.heroes li.selected"))
                .toBeNull("Should not have any selected heroes at start");

            spyOn(cmp,"onSelect").and.callThrough();
            this.fixture.nativeElement.querySelectorAll("ul.heroes li")[5].click();

            expect(cmp.onSelect)
                .toHaveBeenCalledWith(cmp.heroes[5]);
            expect(cmp.selectedHero)
                .toEqual(cmp.heroes[5], "click on hero should change hero");
        })
    ));
});
```

beforeEach () **detectChanges ()** all .

async () fixture **StStable** . componentInstance **nativeElement** .

. . **getComputedStyle ()** Window . .

HTTP

API / . . Angular `XHRBackend` HTTP . . http `MockBackend MockConnection` .

`posts.service.ts` API `posts.service.ts` .

```
import { Http } from '@angular/http';
import { Injectable } from '@angular/core';
import { Observable }     from 'rxjs/rx';

import 'rxjs/add/operator/map';

export interface IPost {
    userId: number;
    id: number;
    title: string;
    body: string;
}

@Injectable()
```

```
export class PostsService {
    posts: IPost[];

    private postsUri = 'http://jsonplaceholder.typicode.com/posts';

    constructor(private http: Http) {
    }

    get(): Observable<IPost[]> {
        return this.http.get(this.postsUri)
                .map((response) => response.json());
    }
}
```

`posts.service.spec.ts` http API .

```
import { TestBed, inject, fakeAsync } from '@angular/core/testing';
import {
    HttpModule,
    XHRBackend,
    ResponseOptions,
    Response,
    RequestMethod
} from '@angular/http';
import {
    MockBackend,
    MockConnection
} from '@angular/http/testing';

import { PostsService } from './posts.service';

describe('PostsService', () => {
    // Mock http response
    const mockResponse = [
        {
            'userId': 1,
            'id': 1,
            'title': 'sunt aut facere repellat provident occaecati excepturi optio
reprehenderit',
            'body': 'quia et suscipit\nsuscipit recusandae consequuntur expedita et
cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet
architecto'
        },
        {
            'userId': 1,
            'id': 2,
            'title': 'qui est esse',
            'body': 'est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea
dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\nqui
aperiam non debitis possimus qui neque nisi nulla'
        },
        {
            'userId': 1,
            'id': 3,
            'title': 'ea molestias quasi exercitationem repellat qui ipsa sit aut',
            'body': 'et iusto sed quo iure\nvoluptatem occaecati omnis eligendi aut
ad\nvoluptatem doloribus vel accusantium quis pariatur\nmolestiae porro eius odio et labore et
velit aut'
        },
        {
```

```
            'userId': 1,
            'id': 4,
            'title': 'eum et est occaecati',
            'body': 'ullam et saepe reiciendis voluptatem adipisci\nsit amet autem assumenda
provident rerum culpa\nquis hic commodi nesciunt rem tenetur doloremque ipsam iure\nquis sunt
voluptatem rerum illo velit'
        }
    ];

    beforeEach(() => {
        TestBed.configureTestingModule({
            imports: [HttpModule],
            providers: [
                {
                    provide: XHRBackend,
                    // This provides mocked XHR backend
                    useClass: MockBackend
                },
                PostsService
            ]
        });
    });

    it('should return posts retrieved from Api', fakeAsync(
        inject([XHRBackend, PostsService],
            (mockBackend, postsService) => {
                mockBackend.connections.subscribe(
                    (connection: MockConnection) => {
                        // Assert that service has requested correct url with expected method
                        expect(connection.request.method).toBe(RequestMethod.Get);

expect(connection.request.url).toBe('http://jsonplaceholder.typicode.com/posts');
                        // Send mock response
                        connection.mockRespond(new Response(new ResponseOptions({
                            body: mockResponse
                        })));
                    });

                postsService.get()
                    .subscribe((posts) => {
                        expect(posts).toBe(mockResponse);
                    });

            })));
});
```

-

.

```
import { Component } from '@angular/core';

@Component({
  selector: 'my-app',
  template: '<h1>{{title}}</h1>'
})
export class MyAppComponent{
  title = 'welcome';
}
```

angle . @angular/core/testing .

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { MyAppComponent } from './banner-inline.component';

describe('Tests for MyAppComponent', () => {

  let fixture: ComponentFixture<MyAppComponent>;
  let comp: MyAppComponent;

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [
        MyAppComponent
      ]
    });
  });

  beforeEach(() => {

    fixture = TestBed.createComponent(MyAppComponent);
    comp = fixture.componentInstance;

  });

  it('should create the MyAppComponent', () => {

      expect(comp).toBeTruthy();

  });

});
```

. TestBed ComponentFixture .

TestBed configureTestingModule configureTestingModule . beforeEach .

TestBed createComponent . createComponent ComponentFixture . Fixture .

2 : https://riptutorial.com/ko/angular2/topic/2329/-2--

33: 2

2 ngForm . Angular 2 exportAs ngForm . ngForm (, ,) .

```
#f = ngForm (creates local template instance "f")
```

ngForm "ngSubmit" (@Output)

```
(ngSubmit)= "login(f.value,f.submitted)"
```

"ngModel" "name" .

```
<input type="text" [(ngModel)]="username" placeholder="enter username" required>
```

f.value JSON .

{username : 'Sachin', : 'Welcome1'}

Examples

RC3 API .

pw-change.template.html

```
<form class="container" [formGroup]="pwChangeForm">
    <label for="current">Current Password</label>
    <input id="current" formControlName="current" type="password" required><br />

    <label for="newPW">New Password</label>
    <input id="newPW" formControlName="newPW" type="password" required><br/>
    <div *ngIf="newPW.touched && newPW.newIsNotOld">
        New password can't be the same as current password.
    </div>

    <label for="confirm">Confirm new password</label>
    <input id="confirm" formControlName="confirm" type="password" required><br />
    <div *ngIf="confirm.touched && confirm.errors.newMatchesConfirm">
        The confirmation does not match.
    </div>

    <button type="submit">Submit</button>
</form>
```

pw-change.component.ts

```
import {Component} from '@angular/core'
import {REACTIVE_FORM_DIRECTIVES, FormBuilder, AbstractControl, FormGroup,
    Validators} from '@angular/forms'
```

```
import {PWChangeValidators} from './pw-validators'

@Component({
    moduleId: module.id
    selector: 'pw-change-form',
    templateUrl: `./pw-change.template.html`,
    directives: [REACTIVE_FORM_DIRECTIVES]
})

export class PWChangeFormComponent {
    pwChangeForm: FormGroup;

    // Properties that store paths to FormControls makes our template less verbose
    current: AbstractControl;
    newPW: AbstractControl;
    confirm: AbstractControl;

    constructor(private fb: FormBuilder) { }
    ngOnInit() {
        this.pwChangeForm = this.fb.group({
            current: ['', Validators.required],
            newPW: ['', Validators.required],
            confirm: ['', Validators.required]
        }, {
            // Here we create validators to be used for the group as a whole
            validator: Validators.compose([
                PWChangeValidators.newIsNotOld,
                PWChangeValidators.newMatchesConfirm
            ])
        );
        this.current = this.pwChangeForm.controls['current'];
        this.newPW = this.pwChangeForm.controls['newPW'];
        this.confirm = this.pwChangeForm.controls['confirm'];
    }
}
```

pw-validators.ts

```
import {FormControl, FormGroup} from '@angular/forms'
export class PWChangeValidators {

    static OldPasswordMustBeCorrect(control: FormControl) {
        var invalid = false;
        if (control.value != PWChangeValidators.oldPW)
            return { oldPasswordMustBeCorrect: true }
        return null;
    }

    // Our cross control validators are below
    // NOTE: They take in type FormGroup rather than FormControl
    static newIsNotOld(group: FormGroup){
        var newPW = group.controls['newPW'];
        if(group.controls['current'].value == newPW.value)
            newPW.setErrors({ newIsNotOld: true });
        return null;
    }

    static newMatchesConfirm(group: FormGroup){
        var confirm = group.controls['confirm'];
```

```
        if(group.controls['newPW'].value !== confirm.value)
            confirm.setErrors({ newMatchesConfirm: true });
        return null;
    }
}
```

.

2 :

```
import { Component } from '@angular/core';
import { Router , ROUTER_DIRECTIVES} from '@angular/router';
import { NgForm }    from '@angular/forms';

@Component({
    selector: 'login',
    template: `
<h2>Login</h2>
<form #f="ngForm" (ngSubmit)="login(f.value,f.valid)" novalidate>
    <div>
        <label>Username</label>
        <input type="text" [(ngModel)]="username" placeholder="enter username" required>
    </div>
    <div>
        <label>Password</label>
        <input type="password" name="password" [(ngModel)]="password" placeholder="enter
password" required>
    </div>
    <input class="btn-primary" type="submit" value="Login">
</form>`
   //For long form we can use **templateUrl** instead of template
})

export class LoginComponent{

    constructor(private router : Router){ }

    login (formValue: any, valid: boolean){
        console.log(formValue);

        if(valid){
            console.log(valid);
        }
    }
}
```

Angular 2 Form - /

..

ts

```
import {bootstrap} from '@angular/platform-browser-dynamic';
import {MyForm} from './my-form.component.ts';

bootstrap(MyForm);
```

```
import {Control} from @'angular/common';

export class CustomValidators {
  static emailFormat(control: Control): [[key: string]: boolean] {
    let pattern:RegExp = /\S+@\S+\.\S+/;
    return pattern.test(control.value) ? null : {"emailFormat": true};
  }
}
```

ts

```
import {Component} from '@angular/core';
import {FORM_DIRECTIVES, NgForm, FormBuilder, Control, ControlGroup, Validators} from
'@angular/common';
import {CustomValidators} from './custom-validators';

@Component({
  selector: 'my-form',
  templateUrl: 'app/my-form.component.html',
  directives: [FORM_DIRECTIVES],
  styleUrls: ['styles.css']
})
export class MyForm {
  email: Control;
  password: Control;
  group: ControlGroup;

  constructor(builder: FormBuilder) {
    this.email = new Control('',
      Validators.compose([Validators.required, CustomValidators.emailFormat])
    );

    this.password = new Control('',
      Validators.compose([Validators.required, Validators.minLength(4)])
    );

    this.group = builder.group({
      email: this.email,
      password: this.password
    });
  }

  onSubmit() {
    console.log(this.group.value);
  }
}
```

HTML

```
<form [ngFormModel]="group" (ngSubmit)="onSubmit()" novalidate>

  <div>
    <label for="email">Email:</label>
    <input type="email" id="email" [ngFormControl]="email">

    <ul *ngIf="email.dirty && !email.valid">
      <li *ngIf="email.hasError('required')">An email is required</li>
    </ul>
```

```
  </div>

  <div>
    <label for="password">Password:</label>
    <input type="password" id="password" [ngFormControl]="password">

    <ul *ngIf="password.dirty && !password.valid">
      <li *ngIf="password.hasError('required')">A password is required</li>
      <li *ngIf="password.hasError('minlength')">A password needs to have at least 4
characterss</li>
    </ul>
  </div>

  <button type="submit">Register</button>

</form>
```

2 : ( )

Angular 2.0.0 .

registration-form.component.ts

```
import { FormGroup,
    FormControl,
    FormBuilder,
    Validators }              from '@angular/forms';

@Component({
    templateUrl: "./registration-form.html"
})
export class ExampleComponent {
    constructor(private _fb: FormBuilder) { }

exampleForm = this._fb.group({
    name: ['DefaultValue', [<any>Validators.required, <any>Validators.minLength(2)]],
    email: ['default@defa.ult', [<any>Validators.required, <any>Validators.minLength(2)]]
})
```

registration-form.html

```
<form [formGroup]="exampleForm" novalidate (ngSubmit)="submit(exampleForm)">
    <label>Name: </label>
    <input type="text" formControlName="name"/>
    <label>Email: </label>
    <input type="email" formControlName="email"/>
    <button type="submit">Submit</button>
</form>
```

Angular 2 Forms (Reactive Forms)

app.module.ts

app.module.ts .

```
import { NgModule } from '@angular/core';
    import { BrowserModule } from '@angular/platform-browser';
    import { FormsModule, ReactiveFormsModule } from '@angular/forms';
    import { AppComponent } from './app.component';
    @NgModule({
      imports: [
        BrowserModule,
        FormsModule,
        ReactiveFormsModule,
     ],
    declarations: [
    AppComponent
     ]
  providers: [],
  bootstrap: [
    AppComponent
   ]
  })
export class AppModule {}
```

app.component.ts

```
import { Component,OnInit } from '@angular/core';
import template from './add.component.html';
import { FormGroup,FormBuilder,Validators } from '@angular/forms';
import { matchingPasswords } from './validators';
@Component({
    selector: 'app',
    template
})
export class AppComponent implements OnInit {
    addForm: FormGroup;
    constructor(private formBuilder: FormBuilder) {
    }
    ngOnInit() {

    this.addForm = this.formBuilder.group({
            username: ['', Validators.required],
            email: ['', Validators.required],
            role: ['', Validators.required],
            password: ['', Validators.required],
            password2: ['', Validators.required] },
          { validator: matchingPasswords('password', 'password2')
        })
    };

addUser() {
        if (this.addForm.valid) {
            var adduser = {
                username: this.addForm.controls['username'].value,
                email: this.addForm.controls['email'].value,
                password: this.addForm.controls['password'].value,
                profile: {
                    role: this.addForm.controls['role'].value,
                    name: this.addForm.controls['username'].value,
                    email: this.addForm.controls['email'].value
```

```
                }
            };
          console.log(adduser);// adduser var contains all our form values. store it where you
want
            this.addForm.reset();// this will reset our form values to null
        }
    }
}
```

app.component.html

```
<div>
  <form [formGroup]="addForm">
   <input type="text" placeholder="Enter username" formControlName="username" />
   <input type="text" placeholder="Enter Email Address" formControlName="email"/>
   <input type="password" placeholder="Enter Password" formControlName="password" />
   <input type="password" placeholder="Confirm Password" name="password2"
formControlName="password2"/>
   <div class='error' *ngIf="addForm.controls.password2.touched">
    <div class="alert-danger errormessageadduser"
*ngIf="addForm.hasError('mismatchedPasswords')">                                    Passwords do
not match
  </div>
</div>
<select name="Role" formControlName="role">
    <option value="admin" >Admin</option>
    <option value="Accounts">Accounts</option>
    <option value="guest">Guest</option>
</select>
<br/>
<br/>
<button type="submit" (click)="addUser()"><span><i class="fa fa-user-plus" aria-
hidden="true"></i></span> Add User </button>
</form>
</div>
```

validators.ts

```
export function matchingPasswords(passwordKey: string, confirmPasswordKey: string) {
    return (group: ControlGroup): {
        [key: string]: any
    } => {
        let password = group.controls[passwordKey];
        let confirmPassword = group.controls[confirmPasswordKey];

        if (password.value !== confirmPassword.value) {
            return {
                mismatchedPasswords: true
            };
        }
    }
}
```

Angular2 -

FormComponent.ts

```
import {Component} from "@angular/core";
import {FormBuilder} from "@angular/forms";

@Component({
 selector: 'app-form',
 templateUrl: './form.component.html',
 styleUrls: ['./form.component.scss'],
 providers : [FormBuilder]
})

export class FormComponent{
  form : FormGroup;
  emailRegex = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;

  constructor(fb: FormBuilder) {

    this.form = fb.group({
      FirstName : new FormControl({value: null}, Validators.compose([Validators.required,
Validators.maxLength(15)])),
      LastName : new FormControl({value: null}, Validators.compose([Validators.required,
Validators.maxLength(15)])),
      Email : new FormControl({value: null}, Validators.compose([
        Validators.required,
        Validators.maxLength(15),
        Validators.pattern(this.emailRegex)]))
    });
  }
}
```

form.component.html

```
<form class="form-details" role="form" [formGroup]="form">
    <div class="row input-label">
      <label class="form-label" for="FirstName">First name</label>
      <input
        [formControl]="form.controls['FirstName']"
        type="text"
        class="form-control"
        id="FirstName"
        name="FirstName">
    </div>
    <div class="row input-label">
      <label class="form-label" for="LastName">Last name</label>
      <input
        [formControl]="form.controls['LastName']"
        type="text"
        class="form-control"
        id="LastName"
        name="LastName">
    </div>
    <div class="row">
      <label class="form-label" for="Email">Email</label>
      <input
        [formControl]="form.controls['Email']"
        type="email"
        class="form-control"
        id="Email"
```

```
        name="Email">
    </div>
    <div class="row">
      <button
        (click)="submit()"
        role="button"
        class="btn btn-primary submit-btn"
        type="button"
        [disabled]="!form.valid">Submit</button>
    </div>
  </div>
</form>
```

2 : https://riptutorial.com/ko/angular2/topic/4607/-2--

34: 2 Ahead (Ahead-of-time)

Examples

1. Angular 2 .

: Angular-CLI .

```
npm install angular/{core,common,compiler,platform-browser,platform-browser-
dynamic,http,router,forms,compiler-cli,tsc-wrapped,platform-server}
```

2 `.compiler` .

2.`tsconfig.json` `angularCompilerOptions` .

```
...
"angularCompilerOptions": {
    "genDir": "./ngfactory"
}
...
```

.

3. ngc .

`./node_modules/.bin/ngc -p src src` 2 . `ngfactory ngfactory` .

`"node_modules/.bin/ngc" -p src` for windows

4. NgFactory `main.ts` .

```
// this is the static platform browser, the usual counterpart is @angular/platform-browser-
dynamic.
import { platformBrowser } from '@angular/platform-browser';

// this is generated by the angular compiler
import { AppModuleNgFactory } from './ngfactory/app/app.module.ngfactory';

// note the use of `bootstrapModuleFactory`, as opposed to `bootstrapModule`.
platformBrowser().bootstrapModuleFactory(AppModuleNgFactory);
```

. Angular-CLI .

```
> ng serve
```

, ?

Q. ? Ans. Angular .

,

```
// ...
compile: function (el, scope) {
  var dirs = this._getElDirectives(el);
  var dir;
  var scopeCreated;
  dirs.forEach(function (d) {
    dir = Provider.get(d.name + Provider.DIRECTIVES_SUFFIX);
    if (dir.scope && !scopeCreated) {
      scope = scope.$new();
      scopeCreated = true;
    }
    dir.link(el, scope, d.value);
  });
  Array.prototype.slice.call(el.children).forEach(function (c) {
    this.compile(c, scope);
  }, this);
},
// ...
```

,

```
<ul>
  <li *ngFor="let name of names"></li>
</ul>
```

.

```
// ...
this._text_9 = this.renderer.createText(this._el_3, '\n', null);
this._text_10 = this.renderer.createText(parentRenderNode, '\n\n', null);
this._el_11 = this.renderer.createElement(parentRenderNode, 'ul', null);
this._text_12 = this.renderer.createText(this._el_11, '\n  ', null);
this._anchor_13 = this.renderer.createTemplateAnchor(this._el_11, null);
this._appEl_13 = new import2.AppElement(13, 11, this, this._anchor_13);
this._TemplateRef_13_5 = new import17.TemplateRef_(this._appEl_13,
viewFactory_HomeComponent1);
this._NgFor_13_6 = new import15.NgFor(this._appEl_13.vcRef, this._TemplateRef_13_5,
this.parentInjector.get(import18.IterableDiffers), this.ref);
// ...
```

AoT .

1. TypeScript Angular 2 .
2. ngc .
3. TypeScript .
4. TypeScript JavaScript .
5. .
6. Minification.
7. .

.

1. .
2. .
3. .

. UX angle2-seed angular-cli .

! !

CLI AoT

Angular CLI 17 AoT .

AoT .

```
ng build --prod --aot
```

2 Ahead (Ahead-of-time) : https://riptutorial.com/ko/angular2/topic/6634/-2-ahead--ahead-of-time--

35: 2

Angular 2 angle . .

" " . .

Examples

```
import { NgModule, CUSTOM_ELEMENTS_SCHEMA } from '@angular/core';
import { CommonModule } from '@angular/common';
import { AboutComponent } from './about.component';

@NgModule({
    imports: [ CommonModule ],
    declarations: [ AboutComponent ],
    exports: [ AboutComponent ],
    schemas: [ CUSTOM_ELEMENTS_SCHEMA ]
})

export class AboutModule { }
```

```
import { Component } from '@angular/core';

@Component({
  selector: 'myapp-about',
  template: `<my-webcomponent></my-webcomponent>`
})
export class AboutComponent { }
```

2 : https://riptutorial.com/ko/angular2/topic/7414/-2-----

36: CLI

.

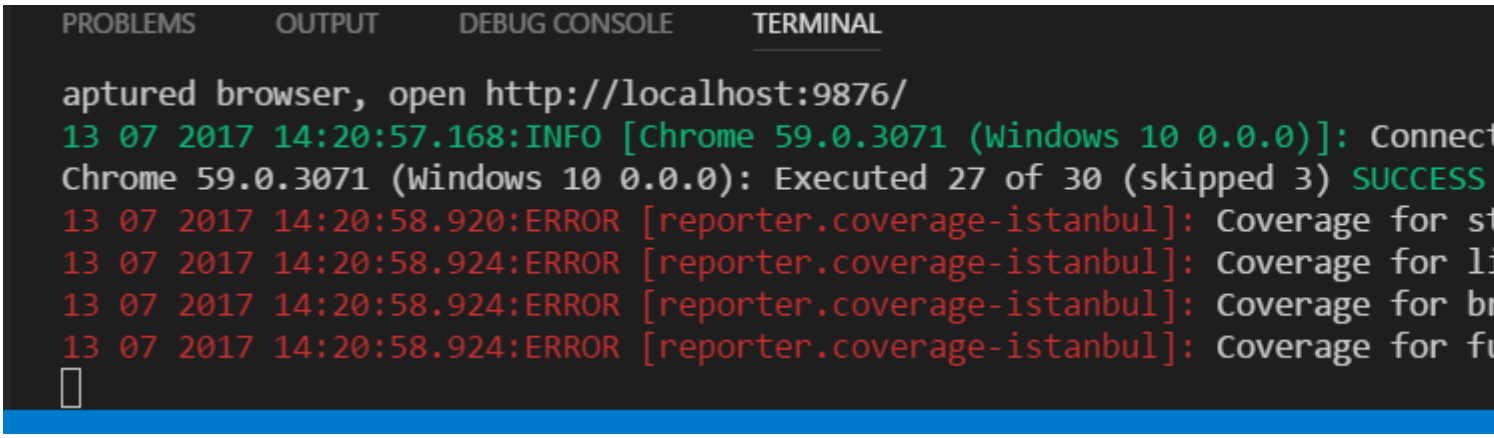
Angular CLI `ng test --cc` .

Examples

CLI

Angular CLI .

```
ng test --cc // or --code-coverage
```

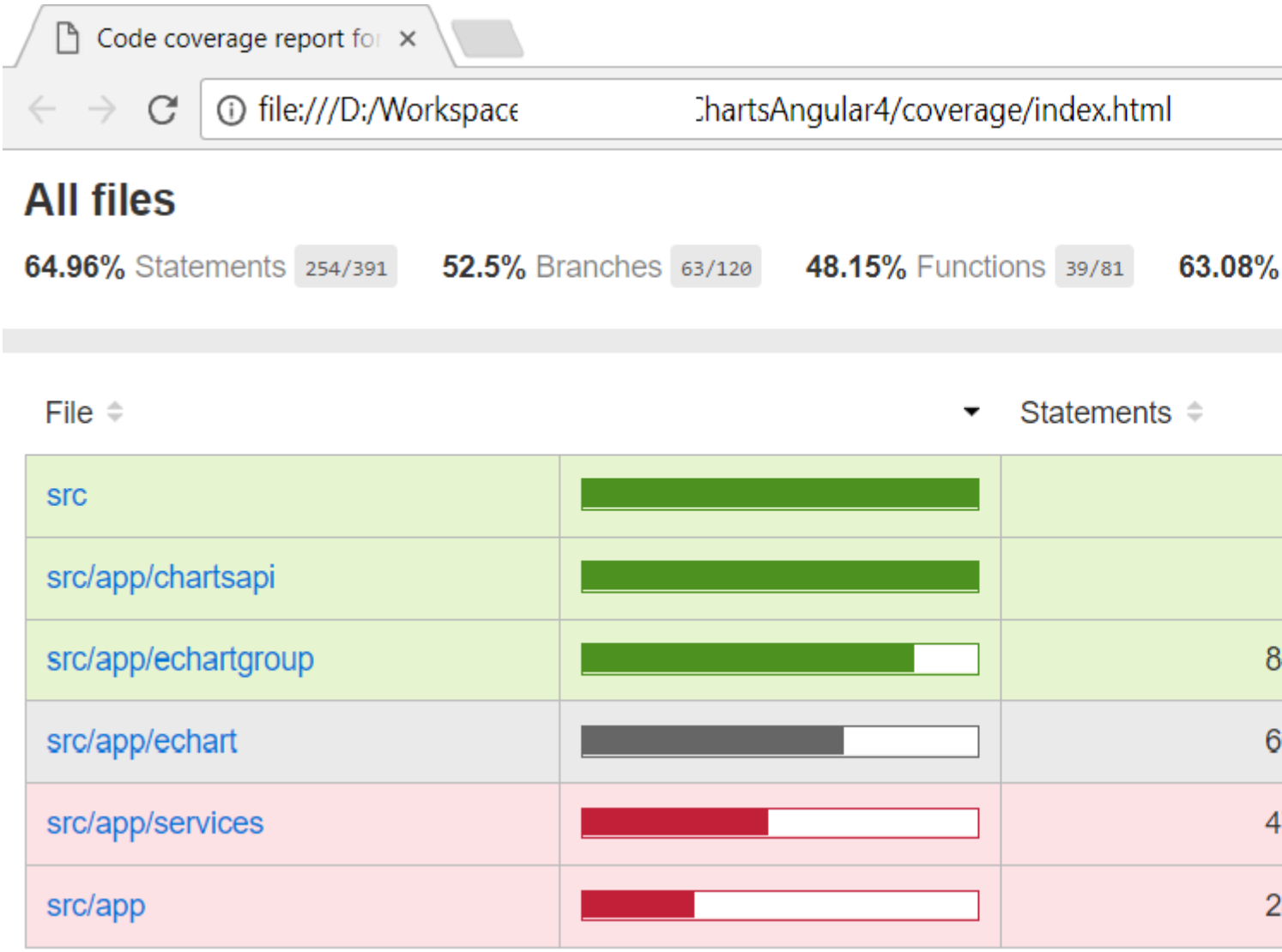


.

1. `npm install --save-dev karma-teamcity-reporter`
2. ``Add `require('karma-teamcity-reporter')` to list of plugins in karma.conf.js``
3. `ng test --code-coverage --reporters=teamcity,coverage-istanbul`

. .

dir `coverage` `index.html` .



`karma.conf.js` .

```
coverageIstanbulReporter: {
      reports: ['html', 'lcovonly'],
      fixWebpackSourcePaths: true,
      thresholds: {
        statements: 90,
        lines: 90,
        branches: 90,
        functions: 90
      }
    },
```

CLI : https://riptutorial.com/ko/angular2/topic/10764/-cli---

37: npm

NgModule npm TypeScript . npm , , .

Examples

```
/
    -src/
        awesome.service.ts
        another-awesome.service.ts
        awesome.module.ts
    -index.ts
    -tsconfig.json
    -package.json
    -rollup.config.js
    -.npmignore
```

.

src / awesome.service.ts :

```
export class AwesomeService {
    public doSomethingAwesome(): void {
        console.log("I am so awesome!");
    }
}
```

src / awesome.module.ts :

```
import { NgModule } from '@angular/core'
import { AwesomeService } from './awesome.service';
import { AnotherAwesomeService } from './another-awesome.service';

@NgModule({
    providers: [AwesomeService, AnotherAwesomeService]
})
export class AwesomeModule {}
```

.

/index.ts :

```
export { AwesomeService } from './src/awesome.service';
export { AnotherAwesomeService } from './src/another-awesome.service';
export { AwesomeModule } from './src/awesome.module';
```

compilerOptions.paths .

/tsconfig.json

```
{
  "compilerOptions": {
    "baseUrl": ".",
    "declaration": true,
    "stripInternal": true,
    "experimentalDecorators": true,
    "strictNullChecks": false,
    "noImplicitAny": true,
    "module": "es2015",
    "moduleResolution": "node",
    "paths": {
      "@angular/core": ["node_modules/@angular/core"],
      "rxjs/*": ["node_modules/rxjs/*"]
    },
    "rootDir": ".",
    "outDir": "dist",
    "sourceMap": true,
    "inlineSources": true,
    "target": "es5",
    "skipLibCheck": true,
    "lib": [
      "es2015",
      "dom"
    ]
  },
  "files": [
    "index.ts"
  ],
  "angularCompilerOptions": {
    "strictMetadataEmit": true
  }
}
```

.

/rollup.config.js

```
export default {
    entry: 'dist/index.js',
    dest: 'dist/bundles/awesome.module.umd.js',
    sourceMap: false,
    format: 'umd',
    moduleName: 'ng.awesome.module',
    globals: {
        '@angular/core': 'ng.core',
        'rxjs': 'Rx',
        'rxjs/Observable': 'Rx',
        'rxjs/ReplaySubject': 'Rx',
        'rxjs/add/operator/map': 'Rx.Observable.prototype',
        'rxjs/add/operator/mergeMap': 'Rx.Observable.prototype',
        'rxjs/add/observable/fromEvent': 'Rx.Observable',
        'rxjs/add/observable/of': 'Rx.Observable'
    },
    external: ['@angular/core', 'rxjs']
}
```

NPM

npm .

/package.json

```
{
  "name": "awesome-angular-module",
  "version": "1.0.4",
  "description": "Awesome angular module",
  "main": "dist/bundles/awesome.module.umd.min.js",
  "module": "dist/index.js",
  "typings": "dist/index.d.ts",
  "scripts": {
    "test": "",
    "transpile": "ngc",
    "package": "rollup -c",
    "minify": "uglifyjs dist/bundles/awesome.module.umd.js --screw-ie8 --compress --mangle --
comments --output dist/bundles/awesome.module.umd.min.js",
    "build": "rimraf dist && npm run transpile && npm run package && npm run minify",
    "prepublishOnly": "npm run build"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/maciejtreder/awesome-angular-module.git"
  },
  "keywords": [
    "awesome",
    "angular",
    "module",
    "minimal"
  ],
  "author": "Maciej Treder <contact@maciejtreder.com>",
  "license": "MIT",
  "bugs": {
    "url": "https://github.com/maciejtreder/awesome-angular-module/issues"
  },
  "homepage": "https://github.com/maciejtreder/awesome-angular-module#readme",
  "devDependencies": {
    "@angular/compiler": "^4.0.0",
    "@angular/compiler-cli": "^4.0.0",
    "rimraf": "^2.6.1",
    "rollup": "^0.43.0",
    "typescript": "^2.3.4",
    "uglify-js": "^3.0.21"
  },
  "dependencies": {
    "@angular/core": "^4.0.0",
    "rxjs": "^5.3.0"
  }
}
```

npm .

/.npmignore

```
node_modules
npm-debug.log
Thumbs.db
.DS_Store
src
!dist/src
plugin
!dist/plugin
*.ngsummary.json
*.iml
rollup.config.js
tsconfig.json
*.ts
!*.d.ts
.idea
```

.

.travis.yml

```
language: node_js
node_js:
- node

deploy:
  provider: npm
  email: contact@maciejtreder.com
  api_key:
    secure: <your api key>
  on:
    tags: true
    repo: maciejtreder/awesome-angular-module
```

https://github.com/maciejtreder/awesome-angular-module .

npm : https://riptutorial.com/ko/angular2/topic/10704/-npm--

38:

Examples

app.module.ts

```
import {appStoreProviders} from "./app.store";
providers : [
  ...
  appStoreProviders,
  ...
]
```

app.store.ts

```
import {InjectionToken} from '@angular/core';
import {createStore, Store, compose, StoreEnhancer} from 'redux';
import {AppState, default as reducer} from "../app.reducer";



export const AppStore = new InjectionToken('App.store');

const devtools: StoreEnhancer<AppState> =
    window['devToolsExtension'] ?
    window['devToolsExtension']() : f => f;

export function createAppStore(): Store<AppState> {
  return createStore<AppState>(
    reducer,
    compose(devtools)
  );
}

export const appStoreProviders = [
  {provide: AppStore, useFactory: createAppStore}
];
```

app.reducer.ts

```
export interface AppState {
  example : string
}

const rootReducer: Reducer<AppState> = combineReducers<AppState>({
 example : string
});

export default rootReducer;
```

store.ts

```
export interface IAppState {
```

```
  example?: string;
}

export const INITIAL_STATE: IAppState = {
  example: null,
};

export function rootReducer(state: IAppState = INITIAL_STATE, action: Action): IAppState {
  switch (action.type) {
    case EXAMPLE_CHANGED:
      return Object.assign(state, state, (<UpdateAction>action));
    default:
      return state;
  }
}
```

actions.ts

```
import {Action} from "redux";
export const EXAMPLE_CHANGED = 'EXAMPLE CHANGED';

export interface UpdateAction extends Action {
  example: string;
}
```

```
import * as Redux from 'redux';
import {Inject, Injectable} from '@angular/core';

@Injectable()
export class exampleService {
    constructor(@Inject(AppStore) private store: Redux.Store<AppState>) {}
    getExampleState(){
      console.log(this.store.getState().example);
   }
}
```

```
import * as Redux from 'redux';
import {Inject, Injectable} from '@angular/core';

@Injectable()
export class exampleService {
    constructor(@Inject(AppStore) private store: Redux.Store<AppState>) {}
    setExampleState(){
       this.store.dispatch(updateExample("new value"));
   }
}
```

actions.ts

```
export interface UpdateExapleAction extends Action {
  example?: string;
}

export const updateExample: ActionCreator<UpdateExapleAction> =
   (newVal) => ({
     type: EXAMPLE_CHANGED,
```

```
     example: newVal
});
```

redux

app.store.ts

```
 import {InjectionToken} from '@angular/core';
 import {createStore, Store, compose, StoreEnhancer} from 'redux';
 import {AppState, default as reducer} from "../app.reducer";


 export const AppStore = new InjectionToken('App.store');

const devtools: StoreEnhancer<AppState> =
     window['devToolsExtension'] ?
     window['devToolsExtension']() : f => f;

export function createAppStore(): Store<AppState> {
   return createStore<AppState>(
     reducer,
     compose(devtools)
   );
}

    export const appStoreProviders = [
      {provide: AppStore, useFactory: createAppStore}
    ];
```

Redux DevTools

: https://riptutorial.com/ko/angular2/topic/10652/-

39:

cli , cli / / / , 3 , .

Examples

angle-cli Angular2

:

- NodeJS :
- npm

---

cmd .

1. `npm install -g @angular/cli yarn global add @angular/cli`
2. `ng new PROJECT_NAME`
3. `cd PROJECT_NAME`
4. `ng serve`

---

localhost : 4200

,,

cmd . ng generate (ng g) Angular .

- : `ng g component my-new-component`
- : `ng g directive my-new-directive`
- : `ng g pipe my-new-pipe`
- : `ng g service my-new-service`
- : `ng g class my-new-classt`
- : `ng g interface my-new-interface`
- `ng g enum my-new-enum` : `ng g enum my-new-enum`
- : `ng g module my-module`

angular-cli.json .

ng2-bootstrap .

1. `npm install ng2-bootstrap --save yarn add ng2-bootstrap`
2. angular-cli.json node-modules .

```
"scripts": [
    "../node_modules/jquery/dist/jquery.js",
```

```
    "../node_modules/bootstrap/dist/js/bootstrap.js"
]
```

cli

outDir angular-cli.json .

```
ng build --target=production --environment=prod
ng build --prod --env=prod
ng build --prod
```

.

```
ng build --target=development --environment=dev
ng build --dev --e=dev
ng build --dev
ng build
```

--base-href your-url index.html () .

index.html href / myUrl / .

```
ng build --base-href /myUrl/
ng build --bh /myUrl/
```

scss / sass

`@angular/cli` **css** .

scss .

```
ng new project_name --style=scss
```

sass .

```
ng new project_name --style=sass
```

@ angular / cli .

Yarn @ angular / cli npm . @ angular / cli .

- (`npm install --global yarn`)
- @ angular / cli (`npm install -g @angular/cli yarn global add @angular/cli`)

@ angular / cli ,

```
ng set --global packageManager=yarn
```

npm @ angular / cli .

```
ng set --global packageManager=npm
```

: https://riptutorial.com/ko/angular2/topic/8956/-

40:

2 . . , , sidenav .

Examples

. . .

image-preview.html

html .

```
<!-- Icon as placeholder when no file picked -->
<i class="material-icons">cloud_upload</i>

<!-- file input, accepts images only. Detect when file has been picked/changed with Angular's
native (change) event listener -->
<input type="file" accept="image/*" (change)="updateSource($event)">

<!-- img placeholder when a file has been picked. shows only when 'source' is not empty -->
<img *ngIf="source" [src]="source" src="">
```

.ts

`<image-preview>` .

```
import {
    Component,
    Output,
    EventEmitter,
} from '@angular/core';

@Component({
    selector: 'image-preview',
    styleUrls: [ './image-preview.css' ],
    templateUrl: './image-preview.html'
})
export class MtImagePreviewComponent {

    // Emit an event when a file has been picked. Here we return the file itself
    @Output() onChange: EventEmitter<File> = new EventEmitter<File>();

    constructor() {}

    // If the input has changed(file picked) we project the file into the img previewer
    updateSource($event: Event) {
        // We access he file with $event.target['files'][0]
        this.projectImage($event.target['files'][0]);
    }

    // Uses FileReader to read the file from the input
    source:string = '';
    projectImage(file: File) {
        let reader = new FileReader;
```

```
        // TODO: Define type of 'e'
        reader.onload = (e: any) => {
            // Simply set e.target.result as our <img> src in the layout
            this.source = e.target.result;
            this.onChange.emit(file);
        };
        // This will process our file and get it's attributes/data
        reader.readAsDataURL(file);
    }
}
```

another.component.html

```
<form (ngSubmit)="submitPhoto()">
    <image-preview (onChange)="getFile($event)"></image-preview>
    <button type="submit">Upload</button>
</form>
```

. AngularJS 1.x . AngularJS 1.5.5 .

`ReactiveFormsModule`

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { FormControl } from '@angular/forms';
import { Subscription } from 'rxjs';

@Component({
  selector: 'component',
  template: `
    <input [formControl]="control" />
    <div *ngFor="let item of content">
      {{item.id}} - {{item.name}}
    </div>
  `
})
export class MyComponent implements OnInit, OnDestroy {

  public control = new FormControl('');

  public content: { id: number; name: string; }[];

  private originalContent = [
    { id: 1, name: 'abc' },
    { id: 2, name: 'abce' },
    { id: 3, name: 'ced' }
  ];

  private subscription: Subscription;

  public ngOnInit() {
    this.subscription = this.control.valueChanges.subscribe(value => {
      this.content = this.originalContent.filter(item => item.name.startsWith(value));
    });
  }

  public ngOnDestroy() {
    this.subscription.unsubscribe();
  }
```

```
}
```

: https://riptutorial.com/ko/angular2/topic/5597/---

41:

.

Examples

`@Component` .

- `providers` :
- `selector` : HTML
- `styles` : . : require . .
- `styleUrls` :
- `template` : HTML
- `templateUrl` : HTML

.

:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-required',
  styleUrls: ['required.component.scss'],
  // template: `This field is required.`,
  templateUrl: 'required.component.html',
})
export class RequiredComponent { }
```

HTML .

.

```
@Component({
  templateUrl: 'hero.component.html',
})
```

```
@Component({
  template: `<div>My template here</div>`,
})
```

. `@Component` . .

```
div { background: red; }
```

`div` HTML div .

.

```
<style>div[_ngcontent-c1] { background: red; }</style>
```

.

```
@Component({
  styleUrls: ['hero.component.css'],
})
```

```
styles: [ `div { background: lime; }` ]
```

require styles .

hero.component.html

```
<form (ngSubmit)="submit($event)" [formGroup]="form" novalidate>
  <input type="text" formControlName="name" />
  <button type="submit">Show hero name</button>
</form>
```

hero.component.ts

```
import { FormControl, FormGroup, Validators } from '@angular/forms';

import { Component } from '@angular/core';

@Component({
  selector: 'app-hero',
  templateUrl: 'hero.component.html',
})
export class HeroComponent {
  public form = new FormGroup({
    name: new FormControl('', Validators.required),
  });

  submit(event) {
    console.log(event);
    console.log(this.form.controls.name.value);
  }
}
```

hero.component.spec.ts

```
import { ComponentFixture, TestBed, async } from '@angular/core/testing';

import { HeroComponent } from './hero.component';
import { ReactiveFormsModule } from '@angular/forms';

describe('HeroComponent', () => {
  let component: HeroComponent;
  let fixture: ComponentFixture<HeroComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [HeroComponent],
      imports: [ReactiveFormsModule],
```

```
    }).compileComponents();

    fixture = TestBed.createComponent(HeroComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  }));

  it('should be created', () => {
    expect(component).toBeTruthy();
  });

  it('should log hero name in the console when user submit form', async(() => {
    const heroName = 'Saitama';
    const element = <HTMLFormElement>fixture.debugElement.nativeElement.querySelector('form');

    spyOn(console, 'log').and.callThrough();

    component.form.controls['name'].setValue(heroName);

    element.querySelector('button').click();

    fixture.whenStable().then(() => {
      fixture.detectChanges();
      expect(console.log).toHaveBeenCalledWith(heroName);
    });
  }));

  it('should validate name field as required', () => {
    component.form.controls['name'].setValue('');
    expect(component.form.invalid).toBeTruthy();
  });
});
```

`selector` .

:

```
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-required',
  template: `{{name}} is required.`
})
export class RequiredComponent {
  @Input()
  public name: String = '';
}
```

`app-required` ( ) .

```
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-sample',
  template: `
    <input type="text" name="heroName" />
    <app-required name="Hero Name"></app-required>
  `
```

```
})
export class RequiredComponent {
  @Input()
  public name: String = '';
}
```

: https://riptutorial.com/ko/angular2/topic/10838/-

42:

- `<element [variableName]="value"></element> //Declaring input to child when using @Input() method.`
- `<element (childOutput)="parentFunction($event)"></element> //Declaring output from child when using @Output() method.`
- `@Output() pageNumberClicked = new EventEmitter(); //Used for sending output data from child component when using @Output() method.`
- `this.pageNumberClicked.emit(pageNum); //Used to trigger data output from child component. when using @Output() method.`
- `@ViewChild(ComponentClass) //Property decorator is required when using ViewChild.`

pageCount	.
pageNumberClicked	.
pageChanged	.

Examples

- @Input @Output

DataListComponent . DataListComponent PagerComponent .

PagerComponent DataListComponent . PagerComponent Output DataListComponent .

```
import { Component, NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { DataListService } from './dataList.service';
import { PagerComponent } from './pager.component';

@Component({
    selector: 'datalist',
    template: `
        <table>
        <tr *ngFor="let person of personsData">
            <td>{{person.name}}</td>
            <td>{{person.surname}}</td>
        </tr>
        </table>

        <pager [pageCount]="pageCount" (pageNumberClicked)="pageChanged($event)"></pager>
    `
})
export class DataListComponent {
    private personsData = null;
    private pageCount: number;

    constructor(private dataListService: DataListService) {
        var response = this.dataListService.getData(1); //Request first page from the service
        this.personsData = response.persons;
        this.pageCount = response.totalCount / 10;//We will show 10 records per page.
    }
```

```
    pageChanged(pageNumber: number){
        var response = this.dataListService.getData(pageNumber); //Request data from the
service with new page number
        this.personsData = response.persons;
    }
}

@NgModule({
    imports: [CommonModule],
    exports: [],
    declarations: [DataListComponent, PagerComponent],
    providers: [DataListService],
})
export class DataListModule { }
```

PagerComponent . .

```
import { Component, Input, Output, EventEmitter } from '@angular/core';

@Component({
    selector: 'pager',
    template: `
    <div id="pager-wrapper">
        <span *ngFor="#page of pageCount" (click)="pageClicked(page)">{{page}}</span>
    </div>
    `
})
export class PagerComponent {
    @Input() pageCount: number;
    @Output() pageNumberClicked = new EventEmitter();
    constructor() { }

    pageClicked(pageNum){
        this.pageNumberClicked.emit(pageNum); //Send clicked page number as output
    }
}
```

- ViewChild

Viewchild . ViewChild .

DataListComponent . DataListComponent PagerComponent. DataListComponent PagerComponent .

```
import { Component, NgModule, ViewChild } from '@angular/core';
import { CommonModule } from '@angular/common';
import { DataListService } from './dataList.service';
import { PagerComponent } from './pager.component';

@Component({
    selector: 'datalist',
    template: `<input type='text' [(ngModel)]="searchText" />
               <button (click)="getData()">Search</button>
        <table>
        <tr *ngFor="let person of personsData">
            <td>{{person.name}}</td>
```

```
            <td>{{person.surname}}</td>
        </tr>
        </table>

        <pager></pager>
    `
})
export class DataListComponent {
    private personsData = null;
    private searchText: string;

    @ViewChild(PagerComponent)
    private pagerComponent: PagerComponent;

    constructor(private dataListService: DataListService) {}

    getData(){
        var response = this.dataListService.getData(this.searchText);
        this.personsData = response.data;
        this.pagerComponent.setPaging(this.personsData / 10); //Show 10 records per page
    }
}

@NgModule({
    imports: [CommonModule],
    exports: [],
    declarations: [DataListComponent, PagerComponent],
    providers: [DataListService],
})
export class DataListModule { }
```

.

. AfterViewInit AfterViewInit .

-

:

```
import { Injectable } from '@angular/core';
import { Subject } from 'rxjs/Subject';

@Injectable()
export class ComponentCommunicationService {

    private componentChangeSource = new Subject();
    private newDateCreationSource = new Subject<Date>();

    componentChanged$ = this.componentChangeSource.asObservable();
    dateCreated$ = this.newDateCreationSource.asObservable();

    refresh() {
        this.componentChangeSource.next();
    }

    broadcastDate(date: Date) {
        this.newDateCreationSource.next(date);
    }
```

```
}
```

:

```
import { Component, Inject } from '@angular/core';
import { ComponentCommunicationService } from './component-refresh.service';

@Component({
    selector: 'parent',
    template: `
    <button (click)="refreshSubsribed()">Refresh</button>
    <h1>Last date from child received: {{lastDate}}</h1>
    <child-component></child-component>
    `
})
export class ParentComponent implements OnInit {

    lastDate: Date;
    constructor(private communicationService: ComponentCommunicationService) { }

    ngOnInit() {
        this.communicationService.dateCreated$.subscribe(newDate => {
            this.lastDate = newDate;
        });
    }

    refreshSubsribed() {
        this.communicationService.refresh();
    }
}
```

:

```
import { Component, OnInit, Inject } from '@angular/core';
import { ComponentCommunicationService } from './component-refresh.service';

@Component({
    selector: 'child-component',
    template: `
    <h1>Last refresh from parent: {{lastRefreshed}}</h1>
    <button (click)="sendNewDate()">Send new date</button>
    `
})
export class ChildComponent implements OnInit {

    lastRefreshed: Date;
    constructor(private communicationService: ComponentCommunicationService) { }

    ngOnInit() {
        this.communicationService.componentChanged$.subscribe(event => {
            this.onRefresh();
        });
    }

    sendNewDate() {
        this.communicationService.broadcastDate(new Date());
    }

    onRefresh() {
```

```
        this.lastRefreshed = new Date();
    }
}
```

AppModule :

```
@NgModule({
    declarations: [
        ParentComponent,
        ChildComponent
    ],
    providers: [ComponentCommunicationService],
    bootstrap: [AppComponent] // not included in the example
})
export class AppModule {}
```

: https://riptutorial.com/ko/angular2/topic/7400/---

43:

.

Examples

HeroChildComponent @Input .

```
import { Component, Input } from '@angular/core';
import { Hero } from './hero';
@Component({
  selector: 'hero-child',
  template: `
    <h3>{{hero.name}} says:</h3>
    <p>I, {{hero.name}}, am at your service, {{masterName}}.</p>
  `
})
export class HeroChildComponent {
  @Input() hero: Hero;
  @Input('master') masterName: string;
}
```

setter

setter .

NameChildComponent .

```
import { Component, Input } from '@angular/core';
@Component({
  selector: 'name-child',
  template: '<h3>"{{name}}"</h3>'
})
export class NameChildComponent {
  private _name = '';
  @Input()
  set name(name: string) {
    this._name = (name && name.trim()) || '<no name set>';
  }
  get name(): string { return this._name; }
}
```

NameParentComponent.

```
import { Component } from '@angular/core';
@Component({
  selector: 'name-parent',
  template: `
  <h2>Master controls {{names.length}} names</h2>
  <name-child *ngFor="let name of names" [name]="name"></name-child>
  `
})
export class NameParentComponent {
```

```
  // Displays 'Mr. IQ', '<no name set>', 'Bombasto'
  names = ['Mr. IQ', '   ', '  Bombasto  '];
}
```

.

EventEmitter . .

EventEmitter . VoterComponent @Output .

```
import { Component, EventEmitter, Input, Output } from '@angular/core';
@Component({
  selector: 'my-voter',
  template: `
    <h4>{{name}}</h4>
    <button (click)="vote(true)"  [disabled]="voted">Agree</button>
    <button (click)="vote(false)" [disabled]="voted">Disagree</button>
  `
})
export class VoterComponent {
  @Input()  name: string;
  @Output() onVoted = new EventEmitter<boolean>();
  voted = false;
  vote(agreed: boolean) {
    this.onVoted.emit(agreed);
    this.voted = true;
  }
}
```

true false ( ) .

VoteTakerComponent ($ event) (onVoted) .

```
import { Component }      from '@angular/core';
@Component({
  selector: 'vote-taker',
  template: `
    <h2>Should mankind colonize the Universe?</h2>
    <h3>Agree: {{agreed}}, Disagree: {{disagreed}}</h3>
    <my-voter *ngFor="let voter of voters"
      [name]="voter"
      (onVoted)="onVoted($event)">
    </my-voter>
  `
})
export class VoteTakerComponent {
  agreed = 0;
  disagreed = 0;
  voters = ['Mr. IQ', 'Ms. Universe', 'Bombasto'];
  onVoted(agreed: boolean) {
    agreed ? this.agreed++ : this.disagreed++;
  }
}
```

.

. .

0 CountdownTimerComponent . .

```
import { Component, OnDestroy, OnInit } from '@angular/core';
@Component({
  selector: 'countdown-timer',
  template: '<p>{{message}}</p>'
})
export class CountdownTimerComponent implements OnInit, OnDestroy {
  intervalId = 0;
  message = '';
  seconds = 11;
  clearTimer() { clearInterval(this.intervalId); }
  ngOnInit()    { this.start(); }
  ngOnDestroy() { this.clearTimer(); }
  start() { this.countDown(); }
  stop()  {
    this.clearTimer();
    this.message = `Holding at T-${this.seconds} seconds`;
  }
  private countDown() {
    this.clearTimer();
    this.intervalId = window.setInterval(() => {
      this.seconds -= 1;
      if (this.seconds === 0) {
        this.message = 'Blast off!';
      } else {
        if (this.seconds < 0) { this.seconds = 10; } // reset
        this.message = `T-${this.seconds} seconds and counting`;
      }
    }, 1000);
  }
}
```

CountdownLocalVarParentComponent .

```
import { Component }                from '@angular/core';
import { CountdownTimerComponent }  from './countdown-timer.component';
@Component({
  selector: 'countdown-parent-lv',
  template: `
  <h3>Countdown to Liftoff (via local variable)</h3>
  <button (click)="timer.start()">Start</button>
  <button (click)="timer.stop()">Stop</button>
  <div class="seconds">{{timer.seconds}}</div>
  <countdown-timer #timer></countdown-timer>
  `,
  styleUrls: ['demo.css']
})
export class CountdownLocalVarParentComponent { }
```

.

() (#timer) . .

start seconds .

.

ViewChild .

. - . .

.

, ViewChild .

. . CountdownTimerComponent .

ViewChild . CountdownViewChildParentComponent.

```
import { AfterViewInit, ViewChild } from '@angular/core';
import { Component }                from '@angular/core';
import { CountdownTimerComponent }  from './countdown-timer.component';
@Component({
  selector: 'countdown-parent-vc',
  template: `
  <h3>Countdown to Liftoff (via ViewChild)</h3>
  <button (click)="start()">Start</button>
  <button (click)="stop()">Stop</button>
  <div class="seconds">{{ seconds() }}</div>
  <countdown-timer></countdown-timer>
  `,
  styleUrls: ['demo.css']
})
export class CountdownViewChildParentComponent implements AfterViewInit {
  @ViewChild(CountdownTimerComponent)
  private timerComponent: CountdownTimerComponent;
  seconds() { return 0; }
  ngAfterViewInit() {
    // Redefine `seconds()` to get from the `CountdownTimerComponent.seconds` ...
    // but wait a tick first to avoid one-time devMode
    // unidirectional-data-flow-violation error
    setTimeout(() => this.seconds = () => this.timerComponent.seconds, 0);
  }
  start() { this.timerComponent.start(); }
  stop() { this.timerComponent.stop(); }
}
```

.

ViewChild AfterViewInit .

@ViewChild timerInputComponent timerComponent .

#timer . .

.

ngAfterViewInit . Angular . 0 .

Angular ngAfterViewInit . Angular . .

setTimeout 1 tick seconds .

.

.

. .

MissionService MissionControlComponent AstronautComponent .

```
import { Injectable } from '@angular/core';
import { Subject }    from 'rxjs/Subject';
@Injectable()
export class MissionService {
  // Observable string sources
  private missionAnnouncedSource = new Subject<string>();
  private missionConfirmedSource = new Subject<string>();
  // Observable string streams
  missionAnnounced$ = this.missionAnnouncedSource.asObservable();
  missionConfirmed$ = this.missionConfirmedSource.asObservable();
  // Service message commands
  announceMission(mission: string) {
    this.missionAnnouncedSource.next(mission);
  }
  confirmMission(astronaut: string) {
    this.missionConfirmedSource.next(astronaut);
  }
}
```

MissionControlComponent .

```
import { Component }          from '@angular/core';
import { MissionService }     from './mission.service';
@Component({
  selector: 'mission-control',
  template: `
    <h2>Mission Control</h2>
    <button (click)="announce()">Announce mission</button>
    <my-astronaut *ngFor="let astronaut of astronauts"
      [astronaut]="astronaut">
    </my-astronaut>
    <h3>History</h3>
    <ul>
      <li *ngFor="let event of history">{{event}}</li>
    </ul>
  `,
  providers: [MissionService]
})
export class MissionControlComponent {
  astronauts = ['Lovell', 'Swigert', 'Haise'];
  history: string[] = [];
  missions = ['Fly to the moon!',
              'Fly to mars!',
              'Fly to Vegas!'];
  nextMission = 0;
  constructor(private missionService: MissionService) {
    missionService.missionConfirmed$.subscribe(
      astronaut => {
        this.history.push(`${astronaut} confirmed the mission`);
      });
```

```
  }
  announce() {
    let mission = this.missions[this.nextMission++];
    this.missionService.announceMission(mission);
    this.history.push(`Mission "${mission}" announced`);
    if (this.nextMission >= this.missions.length) { this.nextMission = 0; }
  }
}
```

AstronautComponent . AstronautComponent MissionControlComponent .

```
import { Component, Input, OnDestroy } from '@angular/core';
import { MissionService } from './mission.service';
import { Subscription }   from 'rxjs/Subscription';
@Component({
  selector: 'my-astronaut',
  template: `
    <p>
      {{astronaut}}: <strong>{{mission}}</strong>
      <button
        (click)="confirm()"
        [disabled]="!announced || confirmed">
        Confirm
      </button>
    </p>
  `
})
export class AstronautComponent implements OnDestroy {
  @Input() astronaut: string;
  mission = '<no mission announced>';
  confirmed = false;
  announced = false;
  subscription: Subscription;
  constructor(private missionService: MissionService) {
    this.subscription = missionService.missionAnnounced$.subscribe(
      mission => {
        this.mission = mission;
        this.announced = true;
        this.confirmed = false;
    });
  }
  confirm() {
    this.confirmed = true;
    this.missionService.confirmMission(this.astronaut);
  }
  ngOnDestroy() {
    // prevent memory leak when component destroyed
    this.subscription.unsubscribe();
  }
}
```

AstronautComponent . . AstronautComponent . .

MissionService MissionControlComponent . MissionControlComponent AstronautComponent .

: https://riptutorial.com/ko/angular2/topic/9454/---

44:

Examples

```
// my-feature.module.ts
import { CommonModule } from '@angular/common';
import { NgModule }     from '@angular/core';

import { MyComponent } from './my.component';
import { MyDirective } from './my.directive';
import { MyPipe }      from './my.pipe';
import { MyService }   from './my.service';

@NgModule({
  imports:      [ CommonModule ],
  declarations: [ MyComponent, MyDirective, MyPipe ],
  exports:      [ MyComponent ],
  providers:    [ MyService ]
})
export class MyFeatureModule { }
```

, (`app.module.ts`) :

```
// app.module.ts
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppComponent }    from './app.component';
import { MyFeatureModule } from './my-feature.module';

@NgModule({
  // import MyFeatureModule in root module
  imports:      [ BrowserModule, MyFeatureModule ],
  declarations: [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

: https://riptutorial.com/ko/angular2/topic/6551/-

45:

Examples

```
@Component({
  selector: 'example-test-compnent',
  template: '<div>
                  <div>{{user.name}}</div>
                  <div>{{user.fname}}</div>
                  <div>{{user.email}}</div>
             </div>'
})

export class ExampleTestComponent implements OnInit{

    let user :User = null;
    ngOnInit(): void {
       this.user.name = 'name';
       this.user.fname= 'fname';
       this.user.email= 'email';
    }

}
```

```
describe('Example unit test component', () => {
  let component: ExampleTestComponent ;
  let fixture: ComponentFixture<ExampleTestComponent >;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ExampleTestComponent]
    }).compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(ExampleTestComponent );
    component = fixture.componentInstance;
    fixture.detectChanges();
  });


  it('ngOnInit should change user object values', () => {
    expect(component.user).toBeNull(); // check that user is null on initialize
    component.ngOnInit(); // run ngOnInit

    expect(component.user.name).toEqual('name');
    expect(component.user.fname).toEqual('fname');
    expect(component.user.email).toEqual('email');
  });
});
```

: https://riptutorial.com/ko/angular2/topic/8955/-

46:

Examples

.

Angular 2 .

: .

```
    <base href='/'>
```

index.html head . Angular 2 .

package.json / .

```
"dependencies": {
  ......
  "@angular/router": "3.0.0-beta.1",
  ......
}
```

.

```
class Route {
  path : string
  pathMatch : 'full'|'prefix'
  component : Type|string
  .........
}
```

(`route/routes.ts`) . . ":" .

. ProviderRouter RouterConfig .

```
import { provideRouter, RouterConfig } from '@angular/router';
import { BarDetailComponent } from '../components/bar-detail.component';
import { DashboardComponent } from '../components/dashboard.component';
import { LoginComponent } from '../components/login.component';
import { SignupComponent } from '../components/signup.component';

export const appRoutes: RouterConfig = [
  { path: '', pathMatch: 'full', redirectTo: 'login' },
  { path: 'dashboard', component: DashboardComponent },
  { path: 'bars/:id', component: BarDetailComponent },
  { path: 'login', component: LoginComponent },
  { path: 'signup',   component: SignupComponent }
];

export const APP_ROUTER_PROVIDER = [provideRouter(appRoutes)];
```

.

`main.ts` (. systemjs.config)

```
import { bootstrap } from '@angular/platform-browser-dynamic';
import { AppComponent } from './components/app.component';
import { APP_ROUTER_PROVIDER } from "./routes/routes";

bootstrap(AppComponent, [ APP_ROUTER_PROVIDER ]).catch(err => console.error(err));
```

/ . . ROUTER_DIRECTIVES .

```
import { ROUTER_DIRECTIVES } from '@angular/router';

@Component({
  selector: 'demo-app',
  template: `
    ..................................
    <div>
      <router-outlet></router-outlet>
    </div>
    ..................................
  `,
  // Add our router directives we will be using
  directives: [ROUTER_DIRECTIVES]
})
```

. RouterOutlet RouterConfig . html . RouterLink href . :

```
import { Component } from '@angular/core';
import { ROUTER_DIRECTIVES } from '@angular/router';

@Component({
  selector: 'demo-app',
  template: `
    <a [routerLink]="['/login']">Login</a>
    <a [routerLink]="['/signup']">Signup</a>
    <a [routerLink]="['/dashboard']">Dashboard</a>
    <div>
      <router-outlet></router-outlet>
    </div>
  `,
  // Add our router directives we will be using
  directives: [ROUTER_DIRECTIVES]
})
export class AppComponent { }
```

. RouterLink .

{ } '@ / ' . '@ angular / router' {ROUTER_DIRECTIVES} () .

```
@Component({
  selector: 'demo-app',
  template: `
        <ul>
          <li *ngFor="let bar of bars | async">
```

```
            <a [routerLink]="['/bars', bar.id]">
              {{bar.name}}
            </a>
          </li>
        </ul>
    <div>
      <router-outlet></router-outlet>
    </div>
  `,
  // Add our router directives we will be using
  directives: [ROUTER_DIRECTIVES]
})
export class AppComponent { }
```

RouterLink .

. , , . .

RouterConfig .

```
import { ProjectsComponent } from '../components/projects.component';
import { MessagesComponent} from '../components/messages.component';

export const appRoutes: RouterConfig = [
  { path: '', pathMatch: 'full', redirectTo: 'login' },
  { path: 'dashboard', component: DashboardComponent,
    children: [
      { path: '', redirectTo: 'projects', pathMatch: 'full' },
      { path: 'projects', component: 'ProjectsComponent' },
      { path: 'messages', component: 'MessagesComponent' }
    ] },
  { path: 'bars/:id', component: BarDetailComponent },
  { path: 'login', component: LoginComponent },
  { path: 'signup',   component: SignupComponent }
];
```

DashboardComponent . DashboardComponent . <router-outlet></router-outlet> .
DashboardComponent RouterOutlet .

```
import { Component } from '@angular/core';

@Component({
  selector: 'dashboard',
  template: `
    <a [routerLink]="['projects']">Projects</a>
    <a [routerLink]="['messages']">Messages</a>
    <div>
      <router-outlet></router-outlet>
    </div>
  `
})
export class DashboardComponent { }
```

, RouterOutlet . dashboard projects ., . .

```
Cannot match any routes: 'dashboard'
```

.

ResolveData

.

/ 3.0.0-beta.2

users.service.ts

```
...
import { Http, Response } from '@angular/http';
import { Observable } from 'rxjs/Rx';
import { User } from './user.ts';



@Injectable()
export class UsersService {

    constructor(public http:Http) {}

    /**
     * Returns all users
     * @returns {Observable<User[]>}
     */
    index():Observable<User[]> {

        return this.http.get('http://mywebsite.com/api/v1/users')
            .map((res:Response) => res.json());
    }

    /**
     * Returns a user by ID
     * @param id
     * @returns {Observable<User>}
     */
    get(id:number|string):Observable<User> {

        return this.http.get('http://mywebsite.com/api/v1/users/' + id)
            .map((res:Response) => res.json());
    }
}
```

users.resolver.ts

```
...
import { UsersService } from './users.service.ts';
import { Observable } from 'rxjs/Rx';
import {
    Resolve,
    ActivatedRouteSnapshot,
    RouterStateSnapshot
} from "@angular/router";
```

```
@Injectable()
export class UsersResolver implements Resolve<User[] | User> {

    // Inject UsersService into the resolver
    constructor(private service:UsersService) {}

    resolve(route:ActivatedRouteSnapshot, state:RouterStateSnapshot):Observable<User[] | User>
{
        // If userId param exists in current URL, return a single user, else return all users
        // Uses brackets notation to access `id` to suppress editor warning, may use dot
notation if you create an interface extending ActivatedRoute with an optional id? attribute
        if (route.params['id']) return this.service.get(route.params['id']);
        return this.service.index();
    }
}
```

users.component.ts

. `data.users` `data.user` *app.routes.ts* ( ) .

```
...
import { ActivatedRoute} from "@angular/router";



@Component(...)
export class UsersComponent {

    users:User[];


    constructor(route: ActivatedRoute) {
        route.data.subscribe(data => {
            // data['Match key defined in RouterConfig, see below']
            this.users = data.users;
        });
    }

    /**
     * It is not required to unsubscribe from the resolver as Angular's HTTP
     * automatically completes the subscription when data is received from the server
     */
}
```

app.routes.ts

```
...
import { UsersResolver } from './resolvers/users.resolver';



export const routes:RouterConfig = <RouterConfig>[
    ...
    {
        path: 'user/:id',
        component: UserComponent,
        resolve: {
```

```
            // hence data.user in UserComponent
            user: UsersResolver
        }
    },
    {
        path: 'users',
        component: UsersComponent,
        resolve: {
            // hence data.users in UsersComponent, note the pluralisation
            users: UsersResolver
        }
    },
    ...
]
...
```

app.resolver.ts

.

: . ' '. *providers* . .

```
...
import { UsersService } from './users.service';
import { UsersResolver } from './users.resolver';

export const ROUTE_RESOLVERS = [
    ...,
    UsersService,
    UsersResolver
]
```

main.browser.ts

.

```
...
import {bootstrap} from '@angular/platform-browser-dynamic';
import { ROUTE_RESOLVERS } from './app.resolver';

bootstrap(<Type>App, [
    ...
    ...ROUTE_RESOLVERS
])
.catch(err => console.error(err));
```

, app.routing.ts app.module.ts ( ) . WebPack SystemJS .

home, home / counter home / counter / fetch-data . . Route . . app.module.ts

Angular . URL URL.

```
import { NgModule } from "@angular/core";
import { RouterModule, Routes } from "@angular/router";
```

```
import { HomeComponent } from "./components/home/home.component";
import { FetchDataComponent } from "./components/fetchdata/fetchdata.component";
import { CounterComponent } from "./components/counter/counter.component";

const appRoutes: Routes = [
    {
        path: "",
        redirectTo: "home",
        pathMatch: "full"
    },
    {
        path: "home",
        children: [
            {
                path: "",
                component: HomeComponent
            },
            {
                path: "counter",
                children: [
                    {
                        path: "",
                        component: CounterComponent
                    },
                    {
                        path: "fetch-data",
                        component: FetchDataComponent
                    }
                ]
            }
        ]
    },
    {
        path: "**",
        redirectTo: "home"
    }
];

@NgModule({
    imports: [
        RouterModule.forRoot(appRoutes)
    ],
    exports: [
        RouterModule
    ]
})
export class AppRoutingModule { }
```

Siraj

: https://riptutorial.com/ko/angular2/topic/2334/

47: (3.0.0)

( ) .

`app.routes.ts` .

1.
2. `routes` . `path` `component` .

Examples

. .

(`main.ts`).

```
//main.ts

import {bootstrap} from '@angular/platform-browser-dynamic';

//Import the App component (root component)
import { App } from './app/app';

//Also import the app routes
import { APP_ROUTES_PROVIDER } from './app/app.routes';

bootstrap(App, [
  APP_ROUTES_PROVIDER,
])
.catch(err => console.error(err));
```

.

(app) HTML / .

```
//app.ts

import {Component} from '@angular/core';
import {Router, ROUTER_DIRECTIVES} from '@angular/router';

@Component({
    selector: 'app',
    templateUrl: 'app.html',
    styleUrls: ['app.css'],
    directives: [
        ROUTER_DIRECTIVES,
    ]
})
export class App {
    constructor() {
    }
}

<!-- app.html -->
```

```
<!-- All of your 'views' will go here -->
<router-outlet></router-outlet>
```

`<router-outlet></router-outlet>` . HTML .

: . `<router-outlet>` .

( )

.

TypeScript .

().

```
<a routerLink="/home">Home</a>
```

`/home` . `/home` Home .

.

```
<a *ngFor="let link of links" [routerLink]="link">{{link}}</a>
```

`links` , `app.ts` :

```
public links[] = [
    'home',
    'login'
]
```

`routerLink` = `<a>` `routerLink` .

```
 <a routerLink="home">home</a>
 <a routerLink="login">login</a>
```

. Angular .

`links[]` .

: @ angle / router 3.0.0.-beta.2 . .

TypeScript

```
//app.routes.ts

import {provideRouter} from '@angular/router';

import {Home} from './routes/home/home';
import {Profile} from './routes/profile/profile';

export const routes = [
```

```
    {path: '', redirectTo: 'home'},
    {path: 'home', component: Home},
    {path: 'login', component: Login},
];

export const APP_ROUTES_PROVIDER = provideRouter(routes);
```

`provideRouter` .

`Home Profile` . `Component` .

.

`path` : . **'/ .....' .** Angular .

`component` :

`redirectTo` : . .

. `provideRouter` boostrap .

. .

.

(Route Guard)

- .
- .
- .
- .

Route Guards . *true* . *false* .

.

- *CanActivate* : .
- *CanActivateChild* : .
- *CanDeactivate* : .
- *CanLoad* : .
- : .

.

Route Guards .

.

```
import { Injectable }     from '@angular/core';
import { CanActivate }    from '@angular/router';

@Injectable()
export class SynchronousGuard implements CanActivate {
  canActivate() {
    console.log('SynchronousGuard#canActivate called');
    return true;
  }
}
```

. Observable Promise .

, , .

```
import { Injectable }     from '@angular/core';
import { CanActivate, Router, ActivatedRouteSnapshot, RouterStateSnapshot } from
'@angular/router';
import { Observable }     from 'rxjs/Rx';
import { MockAuthenticationService } from './authentication/authentication.service';

@Injectable()
export class AsynchronousGuard implements CanActivate {
    constructor(private router: Router, private auth: MockAuthenticationService) {}

    canActivate(route:ActivatedRouteSnapshot,
state:RouterStateSnapshot):Observable<boolean>|boolean {
        this.auth.subscribe((authenticated) => {
            if (authenticated) {
                return true;
            }
            this.router.navigateByUrl('/login');
            return false;
        });
    }
}
```

app.routes

canActivate Guard

```
import { provideRouter, Router, RouterConfig, CanActivate } from '@angular/router';

//components
import { LoginComponent } from './login/login.component';
import { DashboardComponent } from './dashboard/dashboard.component';

export const routes: RouterConfig = [
    { path: 'login', component: LoginComponent },
    { path: 'dashboard', component: DashboardComponent, canActivate: [AuthGuard] }
```

```
}
```

APP_ROUTER_PROVIDERS

```
export const APP_ROUTER_PROVIDERS = [
    AuthGuard,
    provideRouter(routes)
];
```

main.ts (*boot.ts*)

.

1. (Guard)
2. **Guard** (Guard **APP_ROUTER_PROVIDERS**).
   Guard .

```
import { bootstrap } from '@angular/platform-browser-dynamic';
import { provide } from '@angular/core';

import { APP_ROUTER_PROVIDERS } from './app.routes';
import { AppComponent } from './app.component';

bootstrap(AppComponent, [
    APP_ROUTER_PROVIDERS
])
.then(success => console.log(`Bootstrap success`))
.catch(error => console.log(error));
```

route config `currentUser` .

.

.

```
export const routes = [
{
  path: 'Dash',
  pathMatch : 'prefix',
  component: DashCmp,
  canActivate: [AuthGuard],
  resolve: {
      currentUser: CurrentUserResolver
  },
  children: [...[
    path: '',
    component: ProfileCmp,
    resolve: {
        currentUser: currentUser
    }
  ]]
  }
];
```

`AuthService`

```
import { Injectable } from '@angular/core';
import { Http, Headers, RequestOptions } from '@angular/http';
import { Observable } from 'rxjs/Rx';
import 'rxjs/add/operator/do';

@Injectable()
export class AuthService {
  constructor(http: Http) {
    this.http = http;

    let headers = new Headers({ 'Content-Type': 'application/json' });
    this.options = new RequestOptions({ headers: headers });
  }
  fetchCurrentUser() {
    return this.http.get('/api/users/me')
      .map(res => res.json())
      .do(val => this.currentUser = val);
 }
}
```

`AuthGuard` .

```
import { Injectable } from '@angular/core';
import { CanActivate } from "@angular/router";
import { Observable } from 'rxjs/Rx';

import { AuthService } from '../services/AuthService';

@Injectable()
export class AuthGuard implements CanActivate {
  constructor(auth: AuthService) {
    this.auth = auth;
  }
  canActivate(route, state) {
    return Observable
      .merge(this.auth.fetchCurrentUser(), Observable.of(true))
      .filter(x => x == true);
  }
}
```

`CurrentUserResolver` .

```
import { Injectable } from '@angular/core';
import { Resolve } from "@angular/router";
import { Observable } from 'rxjs/Rx';

import { AuthService } from '../services/AuthService';

@Injectable()
export class CurrentUserResolver implements Resolve {
  constructor(auth: AuthService) {
    this.auth = auth;
  }
  resolve(route, state) {
    return this.auth.currentUser;
  }
}
```

(3.0.0) : https://riptutorial.com/ko/angular2/topic/1208/--3-0-0--

48:

`AfterViewInit` `AfterViewChecked` .

- `OnChanges` ( )
- `OnInit` ( )
- `DoCheck` ( )
- `AfterContentInit` ( )
- `AfterContentChecked` ( )
- `AfterViewInit` ( ) ( )
- `AfterViewChecked` ( ) ( )
- `OnDestroy` ( )

- -

Examples

OnInit

.

( )

```
import { Component, OnInit } from '@angular/core';

@Component({
    selector: 'so-oninit-component',
    templateUrl: 'oninit-component.html',
    styleUrls: ['oninit-component.']
})
class OnInitComponent implements OnInit {

    ngOnInit(): void {
        console.log('Component is ready !');
    }
}
```

OnDestroy

.

```
import { Component, OnDestroy } from '@angular/core';

@Component({
    selector: 'so-ondestroy-component',
```

```
    templateUrl: 'ondestroy-component.html',
    styleUrls: ['ondestroy-component.']
})
class OnDestroyComponent implements OnDestroy {

    ngOnDestroy(): void {
        console.log('Component was destroyed !');
    }
}
```

OnChanges

.

```
import { Component, OnChanges, Input } from '@angular/core';

@Component({
    selector: 'so-onchanges-component',
    templateUrl: 'onchanges-component.html',
    styleUrls: ['onchanges-component.']
})
class OnChangesComponent implements OnChanges {
    @Input() name: string;
    message: string;

    ngOnChanges(changes: SimpleChanges): void {
        console.log(changes);
    }
}
```

.

```
name: {
    currentValue: 'new name value',
    previousValue: 'old name value'
},
message: {
    currentValue: 'new message value',
    previousValue: 'old message value'
}
```

AfterContentInit

.

(OnInit)

```
import { Component, AfterContentInit } from '@angular/core';

@Component({
    selector: 'so-aftercontentinit-component',
    templateUrl: 'aftercontentinit-component.html',
    styleUrls: ['aftercontentinit-component.']
})
class AfterContentInitComponent implements AfterContentInit {
```

```
    ngAfterContentInit(): void {
        console.log('Component content have been loaded!');
    }
}
```

AfterContentChecked

.

( )

```
import { Component, AfterContentChecked } from '@angular/core';

@Component({
    selector: 'so-aftercontentchecked-component',
    templateUrl: 'aftercontentchecked-component.html',
    styleUrls: ['aftercontentchecked-component.']
})
class AfterContentCheckedComponent implements AfterContentChecked {

    ngAfterContentChecked(): void {
        console.log('Component content have been checked!');
    }
}
```

AfterViewInit

. Angular 2 . Angular 2 jQuery .

```
import { Component, AfterViewInit } from '@angular/core';

@Component({
    selector: 'so-afterviewinit-component',
    templateUrl: 'afterviewinit-component.html',
    styleUrls: ['afterviewinit-component.']
})
class AfterViewInitComponent implements AfterViewInit {

    ngAfterViewInit(): void {
        console.log('This event fire after the content init have been loaded!');
    }
}
```

AfterViewChecked

.

( )

```
import { Component, AfterViewChecked } from '@angular/core';

@Component({
    selector: 'so-afterviewchecked-component',
```

```
    templateUrl: 'afterviewchecked-component.html',
    styleUrls: ['afterviewchecked-component.']
})
class AfterViewCheckedComponent implements AfterViewChecked {

    ngAfterViewChecked(): void {
        console.log('This event fire after the content have been checked!');
    }
}
```

DoCheck

```
import { Component, DoCheck, Input } from '@angular/core';

@Component({
    selector: 'so-docheck-component',
    templateUrl: 'docheck-component.html',
    styleUrls: ['docheck-component.']
})
class DoCheckComponent implements DoCheck {
    @Input() elements: string[];
    differ: any;
    ngDoCheck(): void {
        // get value for elements property
        const changes = this.differ.diff(this.elements);

        if (changes) {
            changes.forEachAddedItem(res => console.log('Added', r.item));
            changes.forEachRemovedItem(r => console.log('Removed', r.item));
        }
    }
}
```

: https://riptutorial.com/ko/angular2/topic/2935/--

49:

.

. app.module BrowserModule RouterModule .

Examples

@NgModule . @NgModule @NgModule .

- bootstrap : . .
- declarations : . (ng generate component)
- exports :
- imports : ( )
- providers : (di)

:

```
import { AppComponent } from './app.component';
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

@NgModule({
  bootstrap: [AppComponent]
  declarations: [AppComponent],
  exports: [],
  imports: [BrowserModule],
  providers: [],
})
export class AppModule { }
```

@NgModule imports .

core.module , ReservePipe ( ) :

```
import { CommonModule } from '@angular/common';
import { NgModule } from '@angular/core';
import { ReversePipe } from '../reverse.pipe';

@NgModule({
  imports: [
    CommonModule
  ],
  exports: [ReversePipe], // export things to be imported in another module
  declarations: [ReversePipe],
})
export class CoreModule { }
```

app.module :

```
import { CoreModule } from 'app/core/core.module';
```

```
@NgModule({
  declarations: [...], // ReversePipe is available without declaring here
                       // because CoreModule exports it
  imports: [
    CoreModule,        // import things from CoreModule
    ...
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

: https://riptutorial.com/ko/angular2/topic/10840/

50:

Examples

. , . .

app / app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule  } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { EagerComponent } from './eager.component';
import { routing } from './app.routing';
@NgModule({
  imports: [
    BrowserModule,
    routing
  ],
  declarations: [
    AppComponent,
    EagerComponent
  ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

app / app.component.ts

```
import { Component } from '@angular/core';
@Component({
  selector: 'my-app',
  template: `<h1>My App</h1>    <nav>
      <a routerLink="eager">Eager</a>
      <a routerLink="lazy">Lazy</a>
    </nav>
    <router-outlet></router-outlet>
  `
})
export class AppComponent {}
```

app / app.routing.ts

```
import { ModuleWithProviders } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { EagerComponent } from './eager.component';
const routes: Routes = [
  { path: '', redirectTo: 'eager', pathMatch: 'full' },
  { path: 'eager', component: EagerComponent },
  { path: 'lazy', loadChildren: './lazy.module' }
];
export const routing: ModuleWithProviders = RouterModule.forRoot(routes);
```

app / eager.component.ts

```
import { Component } from '@angular/core';
@Component({
  template: '`<p>Eager Component</p>`'
})
export class EagerComponent {}
```

LazyModule LazyComponent (simliar) LazyModule .

app / lazy.module.ts

```
import { NgModule } from '@angular/core';
import { LazyComponent }   from './lazy.component';
import { routing } from './lazy.routing';
@NgModule({
  imports: [routing],
  declarations: [LazyComponent]
})
export class LazyModule {}
```

app / lazy.routing.ts

```
import { ModuleWithProviders } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { LazyComponent } from './lazy.component';
const routes: Routes = [
  { path: '', component: LazyComponent }
];
export const routing: ModuleWithProviders = RouterModule.forChild(routes);
```

app / lazy.component.ts

```
import { Component } from '@angular/core';
@Component({
  template: `<p>Lazy Component</p>`
})
export class LazyComponent {}
```

: https://riptutorial.com/ko/angular2/topic/7751/--

51:

Angular CLI Angular CLI . Angular CLI .

Angular CLI *Angular CLI* .

Angular 2 , , , NgModule .

Examples

CLI

```
ng new NewProject
```

```
ng init NewProject
```

: https://riptutorial.com/ko/angular2/topic/9152/--

52:

.

. / . HTTP .

Examples

mobile.angular.io --mobile .

cli .

```
ng new serviceWorking-example
cd serviceWorking-example
```

, .

ng sets apps.0.serviceWorker = true .

@ angular / service-worker .

> serviceWorker = true @ angular / service-worker . `npm install --save-dev @angular/service-worker` .angular-cli.json `ng set apps.0.serviceWorker=false` .

.angular-cli.json . "serviceWorker": true

true .

ngsw-manifest.json (ngsw-manifest.json .) dist / root . index.html .

ng build --prod

dist / .

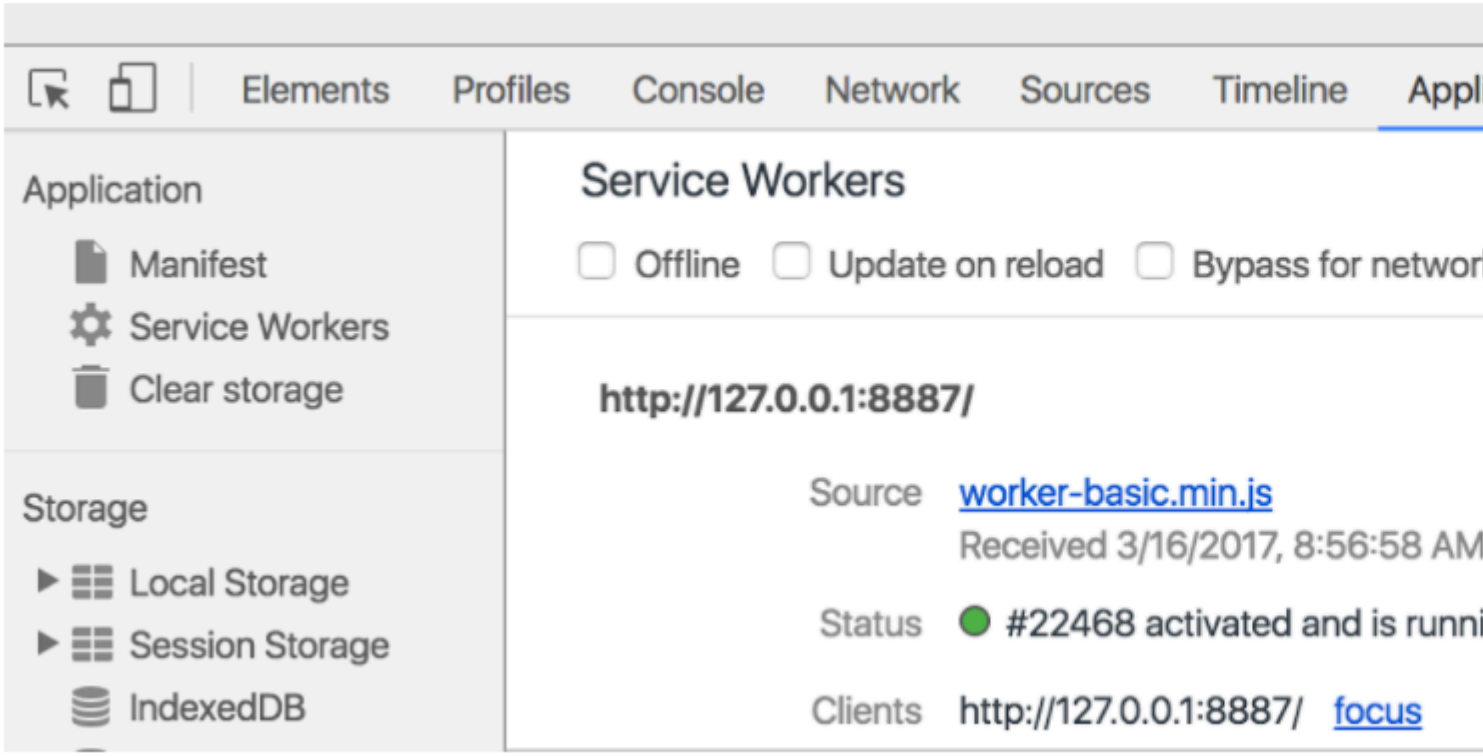
.

- worker-basic.min.js
- sw-register.HASH.bundle.js
- ngsw-manifest.json

index.html (ASW) sw-register .

(Chrome Web Server).

. -> .



.

.

http : // localhost : 4200 / index.html .

http : // localhost : 4200 / . .

/index.html URL /index.html / / some / route .

. .

(19-7-2017). , .

ngsw-manifest.json

```
{
  "routing": {
    "routes": {
      "/": {
        "prefix": false
      }
    },
    "index": "/index.html"
  }
}
```

. http : // localhost : 4200 / http : // localhost : 4200 / index.html .

.

.

https://developers.google.com/web/fundamentals/getting-started/primers/service-workers

https://docs.google.com/document/d/19S5ozevWighny788nI99worpcIMDnwWVmaJDGf_RoDY/edit#

SW precache .

https://coryrylan.com/blog/fast-offline-angular-apps-with-service-workers

: https://riptutorial.com/ko/angular2/topic/10809/-

53: 2

Examples

```
import { NgModule } from '@angular/core';

@NgModule({
  declarations: [], // components your module owns.
  imports: [], // other modules your module needs.
  providers: [], // providers available to your module.
  bootstrap: [] // bootstrap this root component.
})
export class MyModule {}
```

, , . .

.

```
// app.module.ts

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HttpModule } from '@angular/http';
import { MyRootComponent } from './app.component';

@NgModule({
  declarations: [MyRootComponent],
  imports: [BrowserModule, HttpModule],
  bootstrap: [MyRootComponent]
})
export class MyModule {}
```

`MyRootComponent` `MyModule` . Angular 2 .

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { MyModule }               from './app.module';

platformBrowserDynamic().bootstrapModule( MyModule );
```

`MyModule` . `MyModule` Angular 2 .

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppComponent }  from './app.component';

@NgModule({
  imports: [ BrowserModule ],
  declarations: [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

ES5 Javascript . .

```
import { platformBrowser } from '@angular/platform-browser';
import { AppModuleNgFactory } from './main.ngfactory';

// Launch with the app module factory.
platformBrowser().bootstrapModuleFactory(AppModuleNgFactory);
```

ngc .

2 : https://riptutorial.com/ko/angular2/topic/5508/---2--

54: ngx-bootstrap datepicker + input

Examples

ngx-bootstrap datepicker

datepicker.component.html

```
<div (clickOutside)="onClickedOutside($event)" (blur)="onClickedOutside($event)">
      <div class="input-group date" [ngClass]="{'disabled-icon': disabledDatePicker == false
}">
        <input (change)="changedDate()" type="text" [ngModel]="value" class="form-control"
id="{{id}}" (focus)="openCloseDatepicker()" disabled="{{disabledInput}}" />
        <span id="openCloseDatePicker" class="input-group-addon"
(click)="openCloseDatepicker()">
                <span class="glyphicon-calendar glyphicon"></span>
            </span>
      </div>

      <div class="dp-popup" *ngIf="showDatepicker">
        <datepicker [startingDay]="1" [startingDay]="dt" [minDate]="min"  [(ngModel)]="dt"
(selectionDone)="onSelectionDone($event)"></datepicker>
      </div>
    </div>
```

datepicker.component.ts

```
import {Component, Input, EventEmitter, Output, OnChanges, SimpleChanges, ElementRef, OnInit}
from "@angular/core";
import {DatePipe} from "@angular/common";
import {NgModel} from "@angular/forms";
import * as moment from 'moment';

@Component({
  selector: 'custom-datepicker',
  templateUrl: 'datepicker.component.html',
  providers: [DatePipe, NgModel],
  host: {
    '(document:mousedown)': 'onClick($event)',
  }
})

export class DatepickerComponent implements OnChanges , OnInit{
  ngOnInit(): void {
    this.dt = null;
  }

  inputElement : ElementRef;
  dt: Date = null;
  showDatepicker: boolean = false;

  @Input() disabledInput : boolean = false;
  @Input() disabledDatePicker: boolean = false;
  @Input() value: string = null;
  @Input() id: string;
```

```
@Input() min: Date = null;
@Input() max: Date = null;

@Output() dateModelChange = new EventEmitter();
constructor(el: ElementRef) {
  this.inputElement = el;
}

changedDate(){
  if(this.value === ''){
    this.dateModelChange.emit(null);
  }else if(this.value.split('/').length === 3){
    this.dateModelChange.emit(DatepickerComponent.convertToDate(this.value));
  }
}

clickOutSide(event : Event){
  if(this.inputElement.nativeElement !== event.target) {
    console.log('click outside', event);
  }
}

onClick(event) {
  if (!this.inputElement.nativeElement.contains(event.target)) {
    this.close();
  }
}
ngOnChanges(changes: SimpleChanges): void {
  if (this.value !== null && this.value !== undefined && this.value.length > 0) {
    this.value = null;
    this.dt  = null;
  }else {
    if(this.value !== null){
      this.dt = new Date(this.value);
      this.value = moment(this.value).format('MM/DD/YYYY');
    }
  }
}

private static transformDate(date: Date): string {
  return new DatePipe('pt-PT').transform(date, 'MM/dd/yyyy');
}

openCloseDatepicker(): void {
  if (!this.disabledDatePicker) {
    this.showDatepicker = !this.showDatepicker;
  }
}

open(): void {
  this.showDatepicker = true;
}

close(): void {
  this.showDatepicker = false;
}

private apply(): void {
  this.value = DatepickerComponent.transformDate(this.dt);
  this.dateModelChange.emit(this.dt);
}
```

```
  onSelectionDone(event: Date): void {
    this.dt = event;
    this.apply();
    this.close();
  }

  onClickedOutside(event: Date): void {
    if (this.showDatepicker) {
      this.close();
    }
  }

  static convertToDate(val : string): Date {
    return new Date(val.replace('/','-'));
  }

}
```

ngx-bootstrap datepicker + input : https://riptutorial.com/ko/angular2/topic/10549/--ngx-bootstrap-datepicker-plus-input

55:

Examples

null

```
    @Component({
        ...
        animations: [
            trigger('appear', [
                transition(':enter', [
                    style({
                      //style applied at the start of animation
                    }),
                animate('300ms ease-in', style({
                    //style applied at the end of animation
                }))
                ])
            ])
        ]
    })
    class AnimComponent {

    }
]
```

 50px 50px 100px 50px 20px .

state @Component .

state . size "small", "medium" "large" .

 @Component trigger .[@size]="size" .

```
@Component({
  template: '<div [@size]="size">Some Text</div><button
(click)="toggleSize()">TOGGLE</button>',
  animations: [
    trigger('size', [
      state('small', style({
        height: '20px'
      })),
      state('medium', style({
        height: '50px'
      })),
      state('large', style({
        height: '100px'
      })),
      transition('small => medium', animate('100ms')),
      transition('medium => large', animate('200ms')),
      transition('large => small', animate('300ms'))
    ])
  ]
})
```

```
export class TestComponent {

    size: string;

    constructor(){
        this.size = 'small';
    }
    toggleSize(){
        switch(this.size) {
            case 'small':
                this.size = 'medium';
                break;
            case 'medium':
                this.size = 'large';
                break;
            case 'large':
                this.size = 'small';
        }
    }
}
```

: https://riptutorial.com/ko/angular2/topic/8127/

56:

Examples

/ my.service.ts

```
import { Injectable } from '@angular/core';

@Injectable()
export class MyService {
  data: any = [1, 2, 3];

  getData() {
    return this.data;
  }
}
```

.

main.ts

```
import { bootstrap } from '@angular/platform-browser-dynamic';
import { AppComponent } from 'app.component.ts';
import { MyService } from 'services/my.service';

bootstrap(AppComponent, [MyService]);
```

RC5 . ngmodule . *app_module.ts*

```
import { NgModule }       from '@angular/core';
import { BrowserModule  } from '@angular/platform-browser';
import { routing, appRoutingProviders } from './app-routes/app.routes';
import { HttpModule} from '@angular/http';

import { AppComponent }   from './app.component';
import { MyService } from 'services/my.service';

import { routing } from './app-resources/app-routes/app.routes';

@NgModule({
    declarations: [ AppComponent ],
    imports:      [ BrowserModule,
                    routing,
                    RouterModule,
                    HttpModule ],
    providers: [    appRoutingProviders,
                    MyService
    ],
    bootstrap:    [AppComponent],
})
export class AppModule {}
```

`MyComponent`

components / my.component.ts

. .

```
import { Component, OnInit } from '@angular/core';
import { MyService } from '../services/my.service';

@Component({
    ...
    ...
    providers:[MyService] //
 })
export class MyComponent implements OnInit {
    data: any[];
    // Creates private variable myService to use, of type MyService
    constructor(private myService: MyService) { }

    ngOnInit() {
        this.data = this.myService.getData();
    }
}
```

Promise.resolve

/ my.service.ts

```
import { Injectable } from '@angular/core';

@Injectable()
export class MyService {
    data: any = [1, 2, 3];

    getData() {
        return Promise.resolve(this.data);
    }
}
```

`getData()` Promise REST `getData()` . `.then()` . .

components / my.component.ts

```
import { Component, OnInit } from '@angular/core';
import { MyService } from '../services/my.service';

@Component({...})
export class MyComponent implements OnInit {
    data: any[];
    // Creates private variable myService to use, of type MyService
    constructor(private myService: MyService) { }

    ngOnInit() {
        // Uses an "arrow" function to set data
        this.myService.getData().then(data => this.data = data);
    }
}
```

.

```
import 'rxjs/add/operator/toPromise';

import { Http } from '@angular/http';
import { Injectable } from '@angular/core';

interface LoginCredentials {
  password: string;
  user: string;
}

@Injectable()
export class AuthService {
  constructor(private http: Http) { }

  async signIn({ user, password }: LoginCredentials) {
    const response = await this.http.post('/login', {
      password,
      user,
    }).toPromise();

    return response.json();
  }
}
```

.

```
import { ConnectionBackend, Http, HttpModule, Response, ResponseOptions } from
'@angular/http';
import { TestBed, async, inject } from '@angular/core/testing';

import { AuthService } from './auth.service';
import { MockBackend } from '@angular/http/testing';
import { MockConnection } from '@angular/http/testing';

describe('AuthService', () => {
  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [HttpModule],
      providers: [
        AuthService,
        Http,
        { provide: ConnectionBackend, useClass: MockBackend },
      ]
    });
  });

  it('should be created', inject([AuthService], (service: AuthService) => {
    expect(service).toBeTruthy();
  }));

  // Alternative 1
  it('should login user if right credentials are passed', async(
    inject([AuthService], async (authService) => {
      const backend: MockBackend = TestBed.get(ConnectionBackend);
      const http: Http = TestBed.get(Http);

      backend.connections.subscribe((c: MockConnection) => {
```

```
      c.mockRespond(
        new Response(
          new ResponseOptions({
            body: {
              accessToken: 'abcdef',
            },
          }),
        ),
      );
    });

    const result = await authService.signIn({ password: 'ok', user: 'bruno' });

    expect(result).toEqual({
      accessToken: 'abcdef',
    });
  }))
);

// Alternative 2
it('should login user if right credentials are passed', async () => {
  const backend: MockBackend = TestBed.get(ConnectionBackend);
  const http: Http = TestBed.get(Http);

  backend.connections.subscribe((c: MockConnection) => {
    c.mockRespond(
      new Response(
        new ResponseOptions({
          body: {
            accessToken: 'abcdef',
          },
        }),
      ),
    );
  });

  const authService: AuthService = TestBed.get(AuthService);

  const result = await authService.signIn({ password: 'ok', user: 'bruno' });

  expect(result).toEqual({
    accessToken: 'abcdef',
  });
});

// Alternative 3
it('should login user if right credentials are passed', async (done) => {
  const authService: AuthService = TestBed.get(AuthService);

  const backend: MockBackend = TestBed.get(ConnectionBackend);
  const http: Http = TestBed.get(Http);

  backend.connections.subscribe((c: MockConnection) => {
    c.mockRespond(
      new Response(
        new ResponseOptions({
          body: {
            accessToken: 'abcdef',
          },
        }),
      ),
```

```
      );
    });

    try {
      const result = await authService.signIn({ password: 'ok', user: 'bruno' });

      expect(result).toEqual({
        accessToken: 'abcdef',
      });

      done();
    } catch (err) {
      fail(err);
      done();
    }
  });
});
```

: https://riptutorial.com/ko/angular2/topic/4187/---

57:

Params	
	html
(templateUrl)	`<selector>` html . templateUrl html .
	.

!

SANITIZING XSS ( ) . 100 % .

.

```
<tag [attribute]="expression or variable reference | pipeName">
```

.

```
<tag attribute="expression or variable reference | pipeName">
```

```
<tag attribute={{expression or variable reference | pipeName}}
```

Examples

( )

Angular2 .

```
RootOfProject
|
+-- app
|   |-- app.component.ts
|   |-- main.ts
|   |-- pipeUser.component.ts
|   \-- sanitize.pipe.ts
|
|-- index.html
|-- main.html
|-- pipe.html
```

main.ts

```
import { bootstrap } from '@angular/platform-browser-dynamic';
```

```
import { AppComponent } from './app.component';

bootstrap(AppComponent);
```

index.html .

app.component.ts

```
import { Component } from '@angular/core';
import { PipeUserComponent } from './pipeUser.component';

@Component({
    selector: 'main-app',
    templateUrl: 'main.html',
    directives: [PipeUserComponent]
})

export class AppComponent { }
```

.

pipeUser.component.ts

```
import { Component } from '@angular/core';
import { IgnoreSanitize } from "./sanitize.pipe";

@Component({
    selector: 'pipe-example',
    templateUrl: "pipe.html",
    pipes: [IgnoreSanitize]
})

export class PipeUserComponent{
    constructor () { }
    unsafeValue: string = "unsafe/picUrl?id=";
    docNum: string;

    getUrl(input: string): any {
        if(input !== undefined) {
            return this.unsafeValue.concat(input);
            // returns : "unsafe/picUrl?id=input"
        } else {
            return "fallback/to/something";
        }
    }
}
```

.

sanitize.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';
import { DomSanitizationService } from '@angular/platform-browser';

@Pipe({
    name: 'sanitaryPipe'
```

```
})
export class IgnoreSanitize implements PipeTransform {

   constructor(private sanitizer: DomSanitizationService){}

   transform(input: string) : any {
       return this.sanitizer.bypassSecurityTrustUrl(input);
   }

}
```

.

index.html

```
<head>
    Stuff goes here...
</head>
<body>
    <main-app>
        main.html will load inside here.
    </main-app>
</body>
```

main.html

```
<othertags>
</othertags>

<pipe-example>
    pipe.html will load inside here.
</pipe-example>

<moretags>
</moretags>
```

pipe.html

```
<img [src]="getUrl('1234') | sanitaryPipe">
<embed [src]="getUrl() | sanitaryPipe">
```

html .

```
<head>
    Stuff goes here...
</head>

<body>

    <othertags>
    </othertags>

    <img [src]="getUrl('1234') | sanitaryPipe">
    <embed [src]="getUrl() | sanitaryPipe">

    <moretags>
```

```
    </moretags>

</body>
```

: https://riptutorial.com/ko/angular2/topic/5942/------

58: API 2 CRUD

- @Injectable () // .
- request.subscribe () // . . .
- constructor (private wikiService : WikipediaService) {} // .

Examples

Angular2 API

API API . Wikipedia API Wiki .

```
    import { Http, Response } from '@angular/http';
    import { Injectable } from '@angular/core';
    import { Observable }     from 'rxjs/Observable';
    import 'rxjs/Rx';

    @Injectable()
    export class WikipediaService{
        constructor(private http: Http) {}

        getRandomArticles(numberOfArticles: number)
        {
            var request =
this.http.get("https://en.wikipedia.org/w/api.php?action=query&list=random&format=json&rnlimit="
+ numberOfArticles);
            return request.map((response: Response) => {
                return response.json();
            },(error) => {
                console.log(error);
                //your want to implement your own error handling here.
            });
        }
    }
```

API .

```
import { Component, OnInit } from '@angular/core';
import { WikipediaService } from './wikipedia.Service';

@Component({
    selector: 'wikipedia',
    templateUrl: 'wikipedia.component.html'
})
export class WikipediaComponent implements OnInit {
    constructor(private wikiService: WikipediaService) { }

    private articles: any[] = null;
    ngOnInit() {
        var request = this.wikiService.getRandomArticles(5);
        request.subscribe((res) => {
            this.articles = res.query.random;
```

```
            });
        }
}
```

API 2 CRUD : https://riptutorial.com/ko/angular2/topic/7343/-api--2-crud

59:

Examples

Md2Select

:

```
<md2-select [(ngModel)]="item" (change)="change($event)" [disabled]="disabled">
<md2-option *ngFor="let i of items" [value]="i.value" [disabled]="i.disabled">
{{i.name}}</md2-option>
</md2-select>
```

.

```
<md2-select></md2-select>
<md2-option></md2-option>
<md2-select-header></md2-select-header>
```

Md2

. .

.

```
<span tooltip-direction="left" tooltip="On the Left!">Left</span>
<button tooltip="some message"
    tooltip-position="below"
    tooltip-delay="1000">Hover Me
</button>
```

Md2Toast

Toast .

.

```
import {Md2Toast} from 'md2/toast/toast';

@Component({
 selector: "..."
})

export class ... {

...
constructor(private toast: Md2Toast) { }
toastMe() {
this.toast.show('Toast message...');
```

```
---  or  ---

this.toast.show('Toast message...', 1000);
}

...

}
```

Md2Datepicker

Datepicker .

```
<md2-datepicker [(ngModel)]="date"></md2-datepicker>
```

.

Md2Accordion Md2Collapse

Md2Collapse : Collapse .

.

```
<div [collapse]="isCollapsed">
  Lorum Ipsum Content
</div>
```

Md2Accordion : .

.

```
<md2-accordion [multiple]="multiple">
  <md2-accordion-tab *ngFor="let tab of accordions"
                     [header]="tab.title"
                     [active]="tab.active"
                     [disabled]="tab.disabled">
    {{tab.content}}
  </md2-accordion-tab>
  <md2-accordion-tab>
    <md2-accordion-header>Custom Header</md2-accordion-header>
    test content
  </md2-accordion-tab>
</md2-accordion>
```

: https://riptutorial.com/ko/angular2/topic/10005/--

60:

@ / - @ / -

Examples

URL . LocationStrategy URL URL .

Location href URL .

```
import {Component} from '@angular/core';
import {Location} from '@angular/common';

@Component({
    selector: 'app-component'
})
class AppCmp {

  constructor(_location: Location) {

    //Changes the browsers URL to the normalized version of the given URL,
    //and pushes a new item onto the platform's history.
    _location.go('/foo');

  }

  backClicked() {
    //Navigates back in the platform's history.
    this._location.back();
  }

  forwardClicked() {
    //Navigates forward in the platform's history.
    this._location.back();
  }
}
```

AsyncPipe

Observable Promise . . .

```
@Component({
  selector: 'async-observable-pipe',
  template: '<div><code>observable|async</code>: Time: {{ time | async }}</div>'
})
export class AsyncObservablePipeComponent {
  time = new Observable<string>((observer: Subscriber<string>) => {
    setInterval(() => observer.next(new Date().toString()), 1000);
  });
}
```

angular2

@ angular / core **VERSION** .

```
import { Component, VERSION } from '@angular/core';

@Component({
  selector: 'my-app',
  template: `<h1>Hello {{name}}</h1>
  <h2>Current Version: {{ver}}</h2>
  `,
})
export class AppComponent  {
   name = 'Angular2';
   ver = VERSION.full;
}
```

(,) .

```
@Component({
  selector: 'currency-pipe',
  template: `<div>
    <p>A: {{myMoney | currency:'USD':false}}</p>
    <p>B: {{yourMoney | currency:'USD':true:'4.2-2'}}</p>
  </div>`
})
export class CurrencyPipeComponent {
  myMoney: number = 100000.653;
  yourMoney: number = 5.3495;
}
```

3 .

- **currencyCode** : ISO 4217 .
- **symbolDisplay** : .
- **digitInfo** : .

: https://angular.io/docs/ts/latest/api/common/index/CurrencyPipe-pipe.html

: https://riptutorial.com/ko/angular2/topic/8252/-----

61:

Examples

: WARN

:

```
typings WARN deprecated 10/25/2016: "registry:dt/jasmine#2.5.0+20161003201800" is deprecated
(updated, replaced or removed)
```

.

```
npm run typings -- install dt~jasmine --save --global
```

[jazmine] .

: https://riptutorial.com/ko/angular2/topic/7814/--

62: URL / route / subroute

Examples

app.module.ts

```
import {routes} from "./app.routes";

@NgModule({
   declarations: [AppComponent],
   imports: [BrowserModule, mainModule.forRoot(), RouterModule.forRoot(routes)],
   providers: [],
   bootstrap: [AppComponent]
})

export class AppModule { }
```

app.routes.ts

```
import { Routes } from '@angular/router';
import {SubTreeRoutes} from "./subTree/subTreeRoutes.routes";

export const routes: Routes = [
  ...SubTreeRoutes,
  { path: '',  redirectTo: 'home', pathMatch: 'full'}
];
```

subTreeRoutes.ts

```
import {Route} from '@angular/router';
import {exampleComponent} from "./example.component";

export const SubTreeRoutes: Route[] = [
  {
    path: 'subTree',
    children: [
      {path: '',component: exampleComponent}
    ]
  }
];
```

URL / route / subroute : https://riptutorial.com/ko/angular2/topic/8910/-url----route---subroute---

63: @ ngrx /

@ ngrx / Store Angular 2 . . . . Observer Store . TestBed Store .

$	
actionReducer $	
obs $	

any . Store Observer . any .

null StoreMock .

.

Examples

```
class ObserverMock implements Observer<any> {
  closed?: boolean = false; // inherited from Observer
  nextVal: any = ''; // variable I made up

  constructor() {}

  next = (value: any): void => { this.nextVal = value; };
  error = (err: any): void => { console.error(err); };
  complete = (): void => { this.closed = true; }
}

let actionReducer$: ObserverMock = new ObserverMock();
let action$: ObserverMock = new ObserverMock();
let obs$: Observable<any> = new Observable<any>();

class StoreMock extends Store<any> {
  constructor() {
    super(action$, actionReducer$, obs$);
  }
}

describe('Component:Typeahead', () => {
    beforeEach(() => {
        TestBed.configureTestingModule({
            imports: [...],
            declarations: [Typeahead],
            providers: [
                {provide: Store, useClass: StoreMock} // NOTICE useClass instead of useValue
            ]
        }).compileComponents();
```

```
    });
});
```

Store . *MockStore* . Store .

```
import { Injectable } from '@angular/core';
import { TestBed, async} from '@angular/core/testing';
import { AppComponent } from './app.component';
import {DumbComponentComponent} from "./dumb-component/dumb-component.component";
import {SmartComponentComponent} from "./smart-component/smart-component.component";
import {mainReducer} from "./state-management/reducers/main-reducer";
import { StoreModule } from "@ngrx/store";
import { Store } from "@ngrx/store";
import {Observable} from "rxjs";


class MockStore {
  public dispatch(obj) {
    console.log('dispatching from the mock store!')
  }

  public select(obj) {
    console.log('selecting from the mock store!');

    return Observable.of({})
  }
}


describe('AppComponent', () => {
  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [
        AppComponent,
        SmartComponentComponent,
        DumbComponentComponent,
      ],
      imports: [
        StoreModule.provideStore({mainReducer})
      ],
      providers: [
        {provide: Store, useClass: MockStore}
      ]
    });
  });

  it('should create the app', async(() => {

    let fixture = TestBed.createComponent(AppComponent);
    let app = fixture.debugElement.componentInstance;
    expect(app).toBeTruthy();
  }));
```

Store . *store.dispatch ()* " " .

```
import {TestBed, async} from '@angular/core/testing';
import {AppComponent} from './app.component';
import {DumbComponentComponent} from "./dumb-component/dumb-component.component";
```

```
import {SmartComponentComponent} from "./smart-component/smart-component.component";
import {mainReducer} from "./state-management/reducers/main-reducer";
import {StoreModule} from "@ngrx/store";
import {Store} from "@ngrx/store";
import {Observable} from "rxjs";

describe('AppComponent', () => {
  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [
        AppComponent,
        SmartComponentComponent,
        DumbComponentComponent,
      ],
      imports: [
        StoreModule.provideStore({mainReducer})
      ]
    });

  });


  it('should create the app', async(() => {
    let fixture = TestBed.createComponent(AppComponent);
    let app = fixture.debugElement.componentInstance;

  var mockStore = fixture.debugElement.injector.get(Store);
  var storeSpy = spyOn(mockStore, 'dispatch').and.callFake(function () {
  console.log('dispatching from the spy!');
});


}));


});
```

2 - (+)

- postRequest post .

```
import {Injectable} from '@angular/core';
import {Http, Headers, Response} from "@angular/http";
import {PostModel} from "./PostModel";
import 'rxjs/add/operator/map';
import {Observable} from "rxjs";

@Injectable()
export class PostService {

  constructor(private _http: Http) {
  }

  postRequest(postModel: PostModel) : Observable<Response> {
      let headers = new Headers();
      headers.append('Content-Type', 'application/json');
    return this._http.post("/postUrl", postModel, {headers})
      .map(res => res.json());
  }
```

```
}
```

- postService postExample .
- postquest result 'Success'else 'Fail'

```
import {Component} from '@angular/core';
import {PostService} from "./PostService";
import {PostModel} from "./PostModel";

@Component({
  selector: 'app-post',
  templateUrl: './post.component.html',
  styleUrls: ['./post.component.scss'],
  providers : [PostService]
})
export class PostComponent{


  constructor(private _postService : PostService) {

  let postModel = new PostModel();
  result : string = null;
  postExample(){
    this._postService.postRequest(this.postModel)
      .subscribe(
        () =>  {
          this.result = 'Success';
        },
        err =>  this.result = 'Fail'
      )
  }
}
```

- http mockBackend . .
- postService .

```
describe('Test PostService', () => {
  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [HttpModule],
      providers: [
        PostService,
        MockBackend,
        BaseRequestOptions,
        {
          provide: Http,
          deps: [MockBackend, BaseRequestOptions],
          useFactory: (backend: XHRBackend, defaultOptions: BaseRequestOptions) => {
            return new Http(backend, defaultOptions);
          }
        }
      ]
    });
  });



  it('sendPostRequest function return Observable', inject([PostService, MockBackend],
```

```
(service: PostService, mockBackend: MockBackend) => {
    let mockPostModel = PostModel();

    mockBackend.connections.subscribe((connection: MockConnection) => {
      expect(connection.request.method).toEqual(RequestMethod.Post);
      expect(connection.request.url.indexOf('postUrl')).not.toEqual(-1);
      expect(connection.request.headers.get('Content-Type')).toEqual('application/json');
    });

    service
      .postRequest(PostModel)
      .subscribe((response) => {
        expect(response).toBeDefined();
      });
  }));
});
```

```
describe('testing post component', () => {
  let component: PostComponent;
  let fixture: ComponentFixture<postComponent>;


  let mockRouter = {
    navigate: jasmine.createSpy('navigate')
  };

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [PostComponent],
      imports: [RouterTestingModule.withRoutes([]),ModalModule.forRoot() ],
      providers: [PostService ,MockBackend,BaseRequestOptions,
        {provide: Http, deps: [MockBackend, BaseRequestOptions],
          useFactory: (backend: XHRBackend, defaultOptions: BaseRequestOptions) => {
            return new Http(backend, defaultOptions);
          }
        },
        {provide: Router, useValue: mockRouter}
      ],
      schemas: [ CUSTOM_ELEMENTS_SCHEMA ]
    }).compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(PostComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });




  it('test postRequest success', inject([PostService, MockBackend], (service: PostService,
mockBackend: MockBackend) => {
    fixturePostComponent = TestBed.createComponent(PostComponent);
    componentPostComponent = fixturePostComponent.componentInstance;
    fixturePostComponent.detectChanges();


    component.postExample();
    let postModel = new PostModel();
```

```
    let response = {
      'message' : 'message',
      'ok'      : true
    };
    mockBackend.connections.subscribe((connection: MockConnection) => {
      postComponent.result = 'Success'
      connection.mockRespond(new Response(
        new ResponseOptions({
          body: response
        })
      ))
    });
    service.postRequest(postModel)
      .subscribe((data) => {
        expect(component.result).toBeDefined();
        expect(PostComponent.result).toEqual('Success');
        expect(data).toEqual(response);
      });
  }));
});
```

simple.action.ts

```
import { Action } from '@ngrx/store';

export enum simpleActionTpye {
    add = "simpleAction_Add",
    add_Success = "simpleAction_Add_Success"
}

export class simpleAction {
    type: simpleActionTpye
    constructor(public payload: number) { }
}
```

simple.efficts.ts

```
import { Effect, Actions } from '@ngrx/effects';
import { Injectable } from '@angular/core';
import { Action } from '@ngrx/store';
import { Observable } from 'rxjs';

import { simpleAction, simpleActionTpye } from './simple.action';


@Injectable()
export class simpleEffects {

    @Effect()
    addAction$: Observable<simpleAction> = this.actions$
        .ofType(simpleActionTpye.add)
        .switchMap((action: simpleAction) => {
            console.log(action);

            return Observable.of({ type: simpleActionTpye.add_Success, payload: action.payload
})
            //   if you have an api use this code
            // return this.http.post(url).catch().map(res=>{ type: simpleAction.add_Success,
```

```
payload:res})
        });
    constructor(private actions$: Actions) { }
}
```

simple.reducer.ts

```
import { Action, ActionReducer } from '@ngrx/store';

import { simpleAction, simpleActionTpye } from './simple.action';

export const simpleReducer: ActionReducer<number> = (state: number = 0, action: simpleAction):
number => {
    switch (action.type) {
        case simpleActionTpye.add_Success:
            console.log(action);
            return state + action.payload;
        default:
            return state;
    }
}
```

store / index.ts

```
import { combineReducers, ActionReducer, Action, StoreModule } from '@ngrx/store';
import { EffectsModule } from '@ngrx/effects';
import { ModuleWithProviders } from '@angular/core';
import { compose } from '@ngrx/core';

import { simpleReducer } from "./simple/simple.reducer";
import { simpleEffects } from "./simple/simple.efficts";


export interface IAppState {
    sum: number;
}

// all new reducers should be define here
const reducers = {
    sum: simpleReducer
};

export const store: ModuleWithProviders = StoreModule.forRoot(reducers);
export const effects: ModuleWithProviders[] = [
    EffectsModule.forRoot([simpleEffects])
];
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser'
import { NgModule } from '@angular/core';

import { effects, store } from "./Store/index";
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
```

```
    AppComponent
  ],
  imports: [
    BrowserModule,
    // store
    store,
    effects
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.ts

```
import { Component } from '@angular/core';

import { Store } from '@ngrx/store';
import { Observable } from 'rxjs';

import { IAppState } from './Store/index';
import { simpleActionTpye } from './Store/simple/simple.action';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app';

  constructor(private store: Store<IAppState>) {
    store.select(s => s.sum).subscribe((res) => {
      console.log(res);
    })
    this.store.dispatch({
      type: simpleActionTpye.add,
      payload: 1
    })
    this.store.dispatch({
      type: simpleActionTpye.add,
      payload: 2
    })
    this.store.dispatch({
      type: simpleActionTpye.add,
      payload: 3
    })
  }
}
```

0 1 3 6

@ ngrx / : https://riptutorial.com/ko/angular2/topic/8038/---ngrx---

64:

- `<input [value]="value">` - name .
- `<div [attr.data-note]="note">` - data-note note .
- `<p green></p>` -

Angular 2 https://angular.io/docs/ts/latest/guide/attribute-directives.html.

Examples

```
<div [class.active]="isActive"></div>

<span [style.color]="'red'"></span>

<p [attr.data-note]="'This is value for data-note attribute'">A lot of text here</p>
```

.

```
import { Component } from '@angular/core';
@Component({
  selector: 'my-app',
  template: `
    <h1>Angular 2 App</h1>
    <p>Component is directive with template</p>
  `
})
export class AppComponent {
}
```

```
<div *ngFor="let item of items">{{ item.description }}</div>

<span *ngIf="isVisible"></span>
```

```
import {Directive, ElementRef, Renderer} from '@angular/core';

@Directive({
  selector: '[green]',
})

class GreenDirective {
  constructor(private _elementRef: ElementRef,
              private _renderer: Renderer) {
    _renderer.setElementStyle(_elementRef.nativeElement, 'color', 'green');
  }
}
```

:

```
<p green>A lot of green text here</p>
```

* ngFor

form1.component.ts :

```
import { Component } from '@angular/core';

// Defines example component and associated template
@Component({
    selector: 'example',
    template: `
      <div *ngFor="let f of fruit"> {{f}} </div>
      <select required>
        <option *ngFor="let f of fruit" [value]="f"> {{f}} </option>
      </select>
    `
})

// Create a class for all functions, objects, and variables
export class ExampleComponent {
    // Array of fruit to be iterated by *ngFor
    fruit = ['Apples', 'Oranges', 'Bananas', 'Limes', 'Lemons'];
}
```

:

```
<div>Apples</div>
<div>Oranges</div>
<div>Bananas</div>
<div>Limes</div>
<div>Lemons</div>
<select required>
  <option value="Apples">Apples</option>
  <option value="Oranges">Oranges</option>
  <option value="Bananas">Bananas</option>
  <option value="Limes">Limes</option>
  <option value="Lemons">Lemons</option>
</select>
```

`*ngFor` : `let` **`variableName`** `of` **`object/array`**

`fruit = ['Apples', 'Oranges', 'Bananas', 'Limes', 'Lemons'];` ,

Apples, Oranges `fruit` .

`[value]="f" *ngFor` `fruit` (`f`) .

AngularJS Angular2 `<select>` `ng-repeat` `ng-options` .

`*ngFor` `ng-repeat` .

:

Angular2 |

Angular2 | ngFor

Angular2 |

copy-text.directive.ts

```
import {
    Directive,
    Input,
    HostListener
} from "@angular/core";

@Directive({
    selector: '[text-copy]'
})
export class TextCopyDirective {

    // Parse attribute value into a 'text' variable
    @Input('text-copy') text:string;

    constructor() {
    }


    // The HostListener will listen to click events and run the below function, the
HostListener supports other standard events such as mouseenter, mouseleave etc.
    @HostListener('click') copyText() {

        // We need to create a dummy textarea with the text to be copied in the DOM
        var textArea = document.createElement("textarea");

        // Hide the textarea from actually showing
        textArea.style.position = 'fixed';
        textArea.style.top = '-999px';
        textArea.style.left = '-999px';
        textArea.style.width = '2em';
        textArea.style.height = '2em';
        textArea.style.padding = '0';
        textArea.style.border = 'none';
        textArea.style.outline = 'none';
        textArea.style.boxShadow = 'none';
        textArea.style.background = 'transparent';

        // Set the texarea's content to our value defined in our [text-copy] attribute
        textArea.value = this.text;
        document.body.appendChild(textArea);

        // This will select the textarea
        textArea.select();

        try {
            // Most modern browsers support execCommand('copy'|'cut'|'paste'), if it doesn't
it should throw an error
            var successful = document.execCommand('copy');
            var msg = successful ? 'successful' : 'unsuccessful';
            // Let the user know the text has been copied, e.g toast, alert etc.
            console.log(msg);
        } catch (err) {
```

```
            // Tell the user copying is not supported and give alternative, e.g alert window
with the text to copy
            console.log('unable to copy');
        }

        // Finally we remove the textarea from the DOM
        document.body.removeChild(textArea);
    }
}

export const TEXT_COPY_DIRECTIVES = [TextCopyDirective];
```

some-page.component.html

TEXT_COPY_DIRECTIVES .

```
...
    <!-- Insert variable as the attribute's value, let textToBeCopied = 'http://facebook.com/'
-->
    <button [text-copy]="textToBeCopied">Copy URL</button>
    <button [text-copy]="'https://www.google.com/'">Copy URL</button>
...
```

```
import { Directive, ElementRef, HostListener, Input } from '@angular/core';

@Directive({ selector: '[appHighlight]' })
export class HighlightDirective {
  @Input('appHighlight') // tslint:disable-line no-input-rename
  highlightColor: string;

  constructor(private el: ElementRef) { }

  @HostListener('mouseenter')
  onMouseEnter() {
    this.highlight(this.highlightColor || 'red');
  }

  @HostListener('mouseleave')
  onMouseLeave() {
    this.highlight(null);
  }

  private highlight(color: string) {
    this.el.nativeElement.style.backgroundColor = color;
  }
}
```

.

```
import { ComponentFixture, ComponentFixtureAutoDetect, TestBed } from '@angular/core/testing';

import { Component } from '@angular/core';
import { HighlightDirective } from './highlight.directive';

@Component({
  selector: 'app-test-container',
  template: `
```

```
    <div>
      <span id="red" appHighlight>red text</span>
      <span id="green" [appHighlight]="'green'">green text</span>
      <span id="no">no color</span>
    </div>
  `
})
class ContainerComponent { }

const mouseEvents = {
  get enter() {
    const mouseenter = document.createEvent('MouseEvent');
    mouseenter.initEvent('mouseenter', true, true);
    return mouseenter;
  },
  get leave() {
    const mouseleave = document.createEvent('MouseEvent');
    mouseleave.initEvent('mouseleave', true, true);
    return mouseleave;
  },
};

describe('HighlightDirective', () => {
  let fixture: ComponentFixture<ContainerComponent>;
  let container: ContainerComponent;
  let element: HTMLElement;

  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [ContainerComponent, HighlightDirective],
      providers: [
        { provide: ComponentFixtureAutoDetect, useValue: true },
      ],
    });

    fixture = TestBed.createComponent(ContainerComponent);
    // fixture.detectChanges(); // without the provider
    container = fixture.componentInstance;
    element = fixture.nativeElement;
  });

  it('should set background-color to empty when mouse leaves with directive without
arguments', () => {
    const targetElement = <HTMLSpanElement>element.querySelector('#red');

    targetElement.dispatchEvent(mouseEvents.leave);
    expect(targetElement.style.backgroundColor).toEqual('');
  });

  it('should set background-color to empty when mouse leaves with directive with arguments',
() => {
    const targetElement = <HTMLSpanElement>element.querySelector('#green');

    targetElement.dispatchEvent(mouseEvents.leave);
    expect(targetElement.style.backgroundColor).toEqual('');
  });

  it('should set background-color red with no args passed', () => {
    const targetElement = <HTMLSpanElement>element.querySelector('#red');

    targetElement.dispatchEvent(mouseEvents.enter);
```

```
    expect(targetElement.style.backgroundColor).toEqual('red');
  });

  it('should set background-color green when passing green parameter', () => {
    const targetElement = <HTMLSpanElement>element.querySelector('#green');

    targetElement.dispatchEvent(mouseEvents.enter);
    expect(targetElement.style.backgroundColor).toEqual('green');
  });
});
```

: https://riptutorial.com/ko/angular2/topic/2202/

65: : @Input @Output

1. : [propertyName]
2. : (propertyName)
3. ( ) : [(propertyName)]

Examples

@input .

```
import { Input } from '@angular/core';
```

.

```
@Input() car: any;
```

' ', " .

```
<car-component [car]="car"></car-component>
```

(this.car).

:

1. car.entity.ts

```
export class CarEntity {
   constructor(public brand : string, public color : string) {
   }
}
```

2. car.component.ts

```
import { Component, Input } from '@angular/core';
import {CarEntity} from "./car.entity";

@Component({
    selector: 'car-component',
    template: require('./templates/car.html'),
})

export class CarComponent {
    @Input() car: CarEntity;

    constructor() {
        console.log('gros');
    }
}
```

3. garage.component.ts

```
import { Component } from '@angular/core';
import {CarEntity} from "./car.entity";
import {CarComponent} from "./car.component";

@Component({
    selector: 'garage',
    template: require('./templates/garage.html'),
    directives: [CarComponent]
})

export class GarageComponent {
    public cars : Array<CarEntity>;

    constructor() {
        var carOne : CarEntity = new CarEntity('renault', 'blue');
        var carTwo : CarEntity = new CarEntity('fiat', 'green');
        var carThree : CarEntity = new CarEntity('citroen', 'yellow');
        this.cars = [carOne, carTwo, carThree];
    }
}
```

4. garage.html

```
<div *ngFor="let car of cars">
<car-component [car]="car"></car-component>
</div>
```

5. car.html

```
<div>
    <span>{{ car.brand }}</span> |
    <span>{{ car.color }}</span>
</div>
```

Angular2 @Input @Output

`@Input()` Button . `@Output` .

```
import { Component, Input, Output, EventEmitter } from '@angular/core';

@Component({
    selector: 'limited-button',
    template: `<button (click)="onClick()"
                       [disabled]="disabled">
                   <ng-content></ng-content>
               </button>`,
    directives: []
})

export class LimitedButton {
    @Input() clickLimit: number;
    @Output() limitReached: EventEmitter<number> = new EventEmitter();

    disabled: boolean = false;
```

```
    private clickCount: number = 0;

    onClick() {
        this.clickCount++;
        if (this.clickCount === this.clickLimit) {
            this.disabled = true;
            this.limitReached.emit(this.clickCount);
        }
    }
}
```

:

```
import { Component } from '@angular/core';
import { LimitedButton } from './limited-button.component';

@Component({
    selector: 'my-parent-component',
    template: `<limited-button [clickLimit]="2"
                               (limitReached)="onLimitReached($event)">
                   You can only click me twice
               </limited-button>`,
    directives: [LimitedButton]
})

export class MyParentComponent {
    onLimitReached(clickCount: number) {
        alert('Button disabled after ' + clickCount + ' clicks.');
    }
}
```

Angular2 @

. . ngOnChanges @Input .

```
import { Component, OnChanges, OnInit } from '@angular/core';
import { Http, Response } from '@angular/http';
import { ChildComponent } from './child.component';

@Component ({
    selector : 'parent-component',
    template : `
        <child-component [data]="asyncData"></child-component>
    `
})
export class ParentComponent {

    asyncData : any;

    constructor(
        private _http : Http
    ){}

    ngOnInit () {
        this._http.get('some.url')
```

```
            .map(this.extractData)
            .subscribe(this.handleData)
            .catch(this.handleError);
    }

    extractData (res:Response) {
        let body = res.json();
        return body.data || { };
    }

    handleData (data:any) {
        this.asyncData = data;
    }

    handleError (error:any) {
        console.error(error);
    }
}
```

. . ngOnChanges . ngOnChanges . .

```
import { Component, OnChanges, Input } from '@angular/core';

@Component ({
    selector : 'child-component',
    template : `
        <p *ngIf="doesDataExist">Hello child</p>
    `
})
export class ChildComponent {

    doesDataExist: boolean = false;

    @Input('data') data : any;

    // Runs whenever component @Inputs change
    ngOnChanges () {
        // Check if the data exists before using it
        if (this.data) {
            this.useData(data);
        {
    }

    // contrived example to assign data to reliesOnData
    useData (data) {
        this.doesDataExist = true;
    }
}
```

: @Input @Output : https://riptutorial.com/ko/angular2/topic/3046/-------input--output

66:

Examples

.

.

```
import { Component } from '@angular/core';

@Component({
  ...
  template: `
    <div>
      <p [hidden]="!visible" (window:resize)="onResize($event)" >Now you see me...</p>
      <p>now you dont!</p>
    </div>
  `
  ...
})
export class MyComponent {
  visible: boolean = false;
  breakpoint: number = 768;

  constructor() {
  }

  onResize(event) {
    const w = event.target.innerWidth;
    if (w >= this.breakpoint) {
      this.visible = true;
    } else {
      // whenever the window is less than 768, hide this component.
      this.visible = false;
    }
  }
}
```

`visible` false `p` . `visible` `onResize` . `window:resize` `window:resize` .

: https://riptutorial.com/ko/angular2/topic/5276/---

67:

1 .

Examples

2

.

```
<div>
  Hello my name is {{name}} and I like {{thing}} quite a lot.
</div>
```

{}:

.

```
My name is {{name}}
```

"" .

[]:

[] . this.currentVolume this .

```
<video-control [volume]="currentVolume"></video-control>
(): HANDLING EVENTS
```

(): HANDLING EVENTS ()

```
<my-component (click)="onClick($event)"></my-component>
```

[()]:

[()] . .

<input [(ngModel)] = "myName"> this.myName .

*: ASTERISK

. , ngFor .

```
<my-component *ngFor="#item of items">
</my-component>
```

* ngIf * ngSwitch.

: https://riptutorial.com/ko/angular2/topic/9471/

68:

ES2015 ES2015 . ES2015 ES2015 .

Examples

import .

```
import { HeroComponent } from '../heroes/hero.component.ts';
import { Hero }          from '../heroes/hero.model.ts';
import { HeroService }   from '../heroes/hero.service.ts';
```

. " index.ts ( ).

```
export * from './hero.model.ts';   // re-export all of its exports
export * from './hero.service.ts'; // re-export all of its exports
export { HeroComponent } from './hero.component.ts'; // re-export the named thing
```

.

```
import { Hero, HeroService } from '../heroes/index';
```

. .

```
import * as h from '../heroes/index';
```

! `* as h` h .

: https://riptutorial.com/ko/angular2/topic/10717/

69:

| Angular 2 . AngularJS .

/	
@ ({, })	.
:	?
:	true, false .
(, args []?)	
:	
args : []	. args? ? transform (value, arg1, arg2?)

Angular2 HTML Angular2 Pipe .

Examples

.

```
<p>Today is {{ today | date:'fullDate' | uppercase}}.</p>
```

my.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({name: 'myPipe'})
export class MyPipe implements PipeTransform {

  transform(value:any, args?: any):string {
    let transformedValue = value; // implement your transformation logic here
    return transformedValue;
  }

}
```

my.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'my-component',
  template: `{{ value | myPipe }}`
})
export class MyComponent {
```

```
    public value:any;

}
```

my.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { MyComponent } from './my.component';
import { MyPipe } from './my.pipe';

@NgModule({
  imports: [
    BrowserModule,
  ],
  declarations: [
    MyComponent,
    MyPipe
  ],
})
export class MyModule { }
```

Angular2 .

DatePipe	date	{{ dateObj | date }} // output is 'Jun 15, 2015'
UpperCasePipe	uppercase	{{ value | uppercase }} // output is 'SOMETEXT'
LowerCasePipe	lowercase	{{ value | lowercase }} // output is 'sometext'
CurrencyPipe	currency	{{ 31.00 | currency:'USD':true }} // output is '$31'
PercentPipe	percent	{{ 0.03 | percent }} //output is %3

. .

---

hotel-reservation.component.ts

```
import { Component } from '@angular/core';

@Component({
    moduleId: module.id,
    selector: 'hotel-reservation',
    templateUrl: './hotel-reservation.template.html'
})
export class HotelReservationComponent {
    public fName: string =  'Joe';
    public lName: string = 'SCHMO';
    public reservationMade: string = '2016-06-22T07:18-08:00'
    public reservationFor: string = '2025-11-14';
    public cost: number = 99.99;
```

```
}
```

hotel-reservation.template.html

```
<div>
    <h1>Welcome back {{fName | uppercase}} {{lName | lowercase}}</h1>
    <p>
        On {reservationMade | date} at {reservationMade | date:'shortTime'} you
        reserved room 205 for {reservationDate | date} for a total cost of
        {cost | currency}.
    </p>
</div>
```

```
Welcome back JOE schmo
On Jun 26, 2016 at 7:18 you reserved room 205 for Nov 14, 2025 for a total cost of
$99.99.
```

JsonPipe

JsonPipe .

```
@Component({
  selector: 'json-example',
  template: `<div>
    <p>Without JSON pipe:</p>
    <pre>{{object}}</pre>
    <p>With JSON pipe:</p>
    <pre>{{object | json}}</pre>
  </div>`
})
export class JsonPipeExample {
  object: Object = {foo: 'bar', baz: 'qux', nested: {xyz: 3, numbers: [1, 2, 3, 4, 5]}};
}
```

```
Without JSON Pipe:
object
With JSON pipe:
{object:{foo: 'bar', baz: 'qux', nested: {xyz: 3, numbers: [1, 2, 3, 4, 5]}}
```

PLATFORM_PIPES .

```
import { bootstrap }    from '@angular/platform-browser-dynamic';
import { provide, PLATFORM_PIPES } from '@angular/core';

import { AppComponent } from './app.component';
import { MyPipe } from './my.pipe'; // your custom pipe

bootstrap(AppComponent, [
  provide(PLATFORM_PIPES, {
            useValue: [
                MyPipe
            ],
            multi: true
```

```
        })
]);
```

: https://scotch.io/tutorials/create-a-globally-available-custom-pipe-in-angular-2

app / pipes.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({name: 'truthy'})
export class Truthy implements PipeTransform {
  transform(value: any, truthy: string, falsey: string): any {
    if (typeof value === 'boolean'){return value ? truthy : falsey;}
    else return value
  }
}
```

app / my-component.component.ts

```
import { Truthy} from './pipes.pipe';

@Component({
  selector: 'my-component',
  template: `
    <p>{{value | truthy:'enabled':'disabled' }}</p>
  `,
  pipes: [Truthy]
})
export class MyComponent{ }
```

```
import { Component } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/observable/of';

@Component({
  selector: 'async-stuff',
  template: `
    <h1>Hello, {{ name | async }}</h1>
    Your Friends are:
    <ul>
      <li *ngFor="let friend of friends | async">
        {{friend}}
      </li>
    </ul>
  `
})
class AsyncStuffComponent {
  name = Promise.resolve('Misko');
  friends = Observable.of(['Igor']);
}
```

.

```
<h1>Hello, Misko</h1>
Your Friends are:
<ul>
```

```
  <li>
    Igor
  </li>
</ul>
```

```
import { Pipe, PipeTransform } from '@angular/core';
import { DatePipe } from '@angular/common'


@Pipe({name: 'ifDate'})
export class IfDate implements PipeTransform {
  private datePipe: DatePipe = new DatePipe();

  transform(value: any, pattern?:string) : any {
    if (typeof value === 'number') {return value}
    try {
      return this.datePipe.transform(value, pattern)
    } catch(err) {
      return value
    }
  }
}
```

Angular 2 ( ) . . `pure` `false` . `name` , . ( ) . 2 `AsyncPipe` .

. Angular . . Angular 2 Angular 1.x . Angular 1 . Angular 2 .

```
import {Pipe, PipeTransform, OnDestroy} from '@angular/core';

@Pipe({
  name: 'countdown',
  pure: false
})
export class CountdownPipe implements PipeTransform, OnDestroy  {
  private interval: any;
  private remainingTime: number;

  transform(value: number, interval: number = 1000): number {
    if (!parseInt(value, 10)) {
      return null;
    }

    if (typeof this.remainingTime !== 'number') {
      this.remainingTime = parseInt(value, 10);
    }

    if (!this.interval) {
      this.interval = setInterval(() => {
        this.remainingTime--;

        if (this.remainingTime <= 0) {
          this.remainingTime = 0;
          clearInterval(this.interval);
          delete this.interval;
        }
      }, interval);
    }
```

```
    return this.remainingTime;
  }

  ngOnDestroy(): void {
    if (this.interval) {
      clearInterval(this.interval);
    }
  }
}
```

.

```
{{ 1000 | countdown:50 }}
{{ 300 | countdown }}
```

OnDestroy . .

: .

table.component.ts

```
...
import { DYNAMIC_PIPES } from '../pipes/dynamic.pipe.ts';

@Component({
    ...
    pipes: [DYNAMIC_PIPES]
})
export class TableComponent {
    ...

    // pipes to be used for each column
    table.pipes = [ null, null, null, 'humanizeDate', 'statusFromBoolean' ],
    table.header = [ 'id', 'title', 'url', 'created', 'status' ],
    table.rows = [
        [ 1, 'Home', 'home', '2016-08-27T17:48:32', true ],
        [ 2, 'About Us', 'about', '2016-08-28T08:42:09', true ],
        [ 4, 'Contact Us', 'contact', '2016-08-28T13:28:18', false ],
        ...
    ]
    ...

}
```

dynamic.pipe.ts

```
import {
    Pipe,
    PipeTransform
} from '@angular/core';
// Library used to humanize a date in this example
import * as moment from 'moment';

@Pipe({name: 'dynamic'})
export class DynamicPipe implements PipeTransform {
```

```
    transform(value:string, modifier:string) {
        if (!modifier) return value;
        // Evaluate pipe string
        return eval('this.' + modifier + '(\'' + value + '\')')
    }

    // Returns 'enabled' or 'disabled' based on input value
    statusFromBoolean(value:string):string {
        switch (value) {
            case 'true':
            case '1':
                return 'enabled';
            default:
                return 'disabled';
        }
    }

    // Returns a human friendly time format e.g: '14 minutes ago', 'yesterday'
    humanizeDate(value:string):string {
        // Humanize if date difference is within a week from now else returns 'December 20,
2016' format
        if (moment().diff(moment(value), 'days') < 8) return moment(value).fromNow();
        return moment(value).format('MMMM Do YYYY');
    }
}

export const DYNAMIC_PIPES = [DynamicPipe];
```

table.component.html

```
<table>
    <thead>
        <td *ngFor="let head of data.header">{{ head }}</td>
    </thead>
    <tr *ngFor="let row of table.rows; let i = index">
        <td *ngFor="let column of row">{{ column | dynamic:table.pipes[i] }}</td>
    </tr>
</table>
```

```
| ID | Page Title     | Page URL    | Created          | Status     |
-------------------------------------------------------------------
|  1 | Home           | home        | 4 minutes ago    | Enabled    |
|  2 | About Us       | about       | Yesterday        | Enabled    |
|  4 | Contact Us     | contact     | Yesterday        | Disabled   |
-------------------------------------------------------------------
```

.

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({ name: 'reverse' })
export class ReversePipe implements PipeTransform {
  transform(value: string): string {
    return value.split('').reverse().join('');
  }
}
```

spec .

```
import { TestBed, inject } from '@angular/core/testing';

import { ReversePipe } from './reverse.pipe';

describe('ReversePipe', () => {
  beforeEach(() => {
    TestBed.configureTestingModule({
      providers: [ReversePipe],
    });
  });

  it('should be created', inject([ReversePipe], (reversePipe: ReversePipe) => {
    expect(reversePipe).toBeTruthy();
  }));

  it('should reverse a string', inject([ReversePipe], (reversePipe: ReversePipe) => {
    expect(reversePipe.transform('abc')).toEqual('cba');
  }));
});
```

: https://riptutorial.com/ko/angular2/topic/1165/

70:

?

- `setTitle(newTitle: string): void;`
- `getTitle(): string;`

Examples

1. .
2. setTitle

```
import {Title} from "@angular/platform-browser";
@Component({
  selector: 'app',
  templateUrl: './app.component.html',
  providers : [Title]
})

export class AppComponent implements {
   constructor( private title: Title) {
     this.title.setTitle('page title changed');
   }
}
```

: https://riptutorial.com/ko/angular2/topic/8954/-

S. No		Contributors
1	Angular 2	acdcjunior, Alexander Ciesielski, beagleknight, Bean0341, Bhoomi Bhalani, BogdanC, briantyler , cDecker32, Christopher Moore, Community, daniellmb, drbishop, echonax, Ekin Yücel, elliot-j , etayluz, ettanany, Everettss, H. Pauwelyn, Harry, He11ion, Janco Boscan, Jim, Kaspars Bergs, Logan H, Madhu Ranjan, michaelbahr, Michal Pietraszko, Mihai, nick, Nicolas Irisarri, Peter, QoP, rickysullivan, Shahzad, spike, theblindprophet, user6939352
2	@HostBinding .	Max Karpovets
3	Angular 2+ NPM	BogdanC, Janco Boscan, vinagreti
4	Angular 2 jQuery	Ashok Vishwakarma
5	Angular2 CanActivate	Companjo, Yoav Schniederman
6	Angular2 Databinding	Yoav Schniederman
7	Angular2 In Memory API	Jaime Still
8	Angular2	Arnold Wiersma, Norsk, Yoav Schniederman
9	Angular2	Yoav Schniederman
10	Angular2 () ()	Kaloyan, Yoav Schniederman
11	Angular2 .	Ajey
12	angular-cli@1.0.0-β10 3	Alex Morales, Daredzik, filoxo, Kaspars Bergs, pd farhad
13	API RXJS Observables	daniellmb, Maciej Treder, Ronald Zarīts, Sam Storie, Sébastien Temprado, willydee
14	ASP.net Angular 2 TypeScript	Oleksii Aza, Sam
15	ChangeDetectionStrategy	daniellmb, Eric Jimenez, Everettss
16	Dropzone 2	Ketan Akbari
17	EventEmitter	Abrar Jahin

18	HTTP	Everettss, Mihai, Mike Kovetsky, Nilz11, Paul Marshall, peeskillet, theblindprophet
19	ngfor	Jorge, Yoav Schniederman
20	ngif	Amit kumar, ob1, ppovoski, samAlvin
21	ngModel	jesussegado
22	ngrx	Maxime
23	OrderBy	Yoav Schniederman
24	ViewContainerRef.createComponent	amansoni211, daniellmb, Günter Zöchbauer, jupiter24, Khaled
25	Visual Studio Angular2	PSabuwala
26	Webpack Angular2	luukgruijs
27	Zone.js	Roope Hakulinen
28	- ForLoop	aholtry, Anil Singh, Berseker59, gerl, Johan Van de Merwe, ob1, Pujan Srivastava, Stephen Leppik, Yoav Schniederman
29	2 -	Yoav Schniederman
30	2	MatWaligora, ThunderRoid
31	2	Yoav Schniederman
32	2	Arun Redhu, michaelbahr, nick, Reza, Rumit Parakhiya
33	2	Amit kumar, Anil Singh, Christopher Taylor, Highmastdon, Johan Van de Merwe, K3v1n, Manmeet Gill, mayur, Norsk, Sachin S, victoroniibukun, vijaykumar, Yoav Schniederman
34	2 Ahead (Ahead-of-time)	Anil Singh, Eric Jimenez, Harry, Robin Dijkhof
35	2	ugreen
36	CLI	ahmadalibaloch
37	npm	Maciej Treder
38		Yoav Schniederman
39		BogdanC, Yoav Schniederman

40		borislemke, smnbbrv
41		BrunoLM
42		H. Pauwelyn, Janco Boscan, LLL, Sefa
43		AryanJ-NYC, gsc
44		Yoav Schniederman
45		aholtry, Apmis, AryanJ-NYC, borislemke, camwhite, Kaspars Bergs, LordTribual, Sachin S , theblindprophet
46	(3.0.0)	Ai_boy, Alexis Le Gal, Everettss, Gerard Simpson, Kaspars Bergs, mast3rd3mon, meorfi, rivanov, SlashTag, smnbbrv, theblindprophet, ThomasP1988, Trent
47		Alexandre Junges, daniellmb, Deen John, muetzerich, Sbats, theblindprophet
48		BrunoLM
49		Batajus, M4R1KU, Shannon Young, Syam Pradeep
50		Jim, Treveshan Naidoo
51		Roberto Fernandez
52	2	AryanJ-NYC, autoboxer, Berseker59, Eric Jimenez, Krishan, Sanket, snorkpete
53	ngx-bootstrap datepicker + input	Yoav Schniederman
54		Gaurav Mukherjee, Nate May
55		BrunoLM, Eduardo Carísio, Kaspars Bergs, Matrim, Roope Hakulinen, Syam Pradeep, theblindprophet
56		Scrambo
57	API 2 CRUD	bleakgadfly, Sefa
58		Ketan Akbari, Shailesh Ladumor
59		Jim, Sanket
60		kEpEx

61	URL / route / subroute	Yoav Schniederman
62	@ ngrx /	BrianRT, Hatem, Jim, Lucas, Yoav Schniederman
63		acdcjunior, Andrei Zhytkevich, borislemke, BrunoLM, daniellmb, Everettss, lexith, Stian Standahl, theblindprophet
64	: @Input @Output	acdcjunior, dafyddPrys, Everettss, Joel Almeida, lexith, muetzerich, theblindprophet, ThomasP1988
65		Eric Jimenez
66		Max Karpovets
67		TechJhola
68		acdcjunior, Boris, borislemke, BrunoLM, Christopher Taylor, Chybie, daniellmb, Daredzik, elliot-j, Everettss, Fredrik Lundin, Jarod Moser, Jeff Cross, Jim, Kaspars Bergs, Leon Adler, Lexi , LordTribual, michaelbahr, Philipp Kief, theblindprophet
69		Yoav Schniederman