**2. Angular CLI**

1. Создание проекта:

create-nx-workspace angular-core-workshop

Смена названия проекта:

nx.json ->"npmScope": "workshop"

1. Установка препроцессора по умолчанию:

ng config schematics.@nrwl/schematics:component.styleext scss

1. Создание приложения dashboard:

ng g app dashboard --routings -p=app --style=scss

1. Добавляем Angular.material:

ng add @angular/material

1. Создаем библиотеку для Angular.material:

ng g lib material

1. Меняем значение export в файле index.ts библиотеки material:

C

export \* from './lib/material.module';

на

export {MaterialModule} from './lib/material.module';

для точности

1. Добавляем в @NgModule imports и exports файла material.module.ts компоненты которые нам понадобятся.

Таким образом создадим библиотеку компонентов, чтобы не вызывать их из родной библиотеки загромождая проект

1. В файле app.module.ts ветки dashboard добавляем в imports MaterialModule
2. Создание библиотеки работы с данными core-data

ng g lib core-data

1. Создание библиотеки интерфейса входа ui-login

ng g lib ui-login -p=ui

Библиотека имеет префикс=ui.

1. Создание библиотеки интерфейса панели инструментов ui-toolbar

ng g lib ui-toolbar -p=ui

Библиотека имеет префикс=ui.

1. Создание моделя home

ng g m home –routing

1. Создание компонента home

ng g c home

1. Создание модуля projects

ng g m projects –routing

1. Создание компонента projects

ng g c projects

1. Создание модуля customers

ng g m customers –routing

1. Создание компонента customers

ng g c customers

Добавляем в файл app.module.ts приложения dashboard в *imports созданные модули* HomeModule,ProjectsModule, CustomersModule

1. В файл app.component.html удалем все и добавляем

<**app-home**></**app-home**>  
<**router-outlet**></**router-outlet**>

1. В файл [home.module.ts](http://home.module.ts) добавляем значение в exports

**exports**:[HomeComponent]

Для отображения созданного компонента home

1. В файл [projects.module.ts](http://home.module.ts) добавляем значение в exports

**exports**:[ ProjectsComponent]

Для отображения созданного компонента projects

1. В файле app.component.html меняем <**app-home**></**app-home**>

На <**app-projects**></**app-projects**>

1. В файле projects.component.html добавляем код страницы

Projects Component

1. В файле projects.module.ts добавляем в imports MaterialModule
2. В файле projects.component.html добавляем код страницы

Projects Component

1. В файл styles.scss добляем следующие стили:

.container {

display: flex;

margin: 10px;

flex-wrap: wrap;

}

.container [class\*='col'] {

padding: 10px;

flex: 1;

}

app-root {

display: flex;

flex-direction: column;

flex: 1;

}

// стили для для записи на карточке

mat-list-item:not(:first-of-type) {

border-top: 1px solid #efefef;

}

mat-list-item:hover {

cursor: pointer;

background: whitesmoke;

}

mat-list-item:not(:first-of-type) {

border-top: 1px solid #efefef;

}

1. В файле app.component.html добавляем код страницы

App Component

1. В файл styles.scss добляем следующие стили:

// стили для бокового меню

mat-sidenav {

box-shadow: 3px 0 6px rgba(0, 0, 0, 0.24);

width: 200px;

}

.mat-sidenav-container {

background: #f5f5f5;

flex: 1;

}

.nav-link {

color: rgba(0, 0, 0, 0.54);

display: flex !important;

align-items: center;

padding-top: 5px;

padding-bottom: 5px;

}

// UI TOOLBAR

mat-toolbar {

box-shadow: 0 3px 5px -1px rgba(0, 0, 0, 0.2),

0 6px 10px 0 rgba(0, 0, 0, 0.14), 0 1px 18px 0 rgba(0, 0, 0, 0.12);

z-index: 900;

> .mat-mini-fab {

margin-right: 10px;

}

.spacer {

flex: 1 1 auto;

}

.title {

vertical-align: middle;

}

.logo {

margin-left: 20px;

img {

vertical-align: middle;

width: 100px;

height: auto;

}

}

}

**3. Components**

**3.1. Binding Component Data to Templates**

1. В файле projects.component.ts добавляем массив объектов:

projects = [

{

id: '1',

title: 'Project One',

details: 'This is a sample project',

percentComplete: 20,

approved: false,

},

{

id: '2',

title: 'Project Two',

details: 'This is a sample project',

percentComplete: 40,

approved: false,

},

{

id: '3',

title: 'Project Three',

details: 'This is a sample project',

percentComplete: 100,

approved: true,

}

];

1. В файле projects.component.html добвляем запись <pre>{{projects|json}}</pre>

Данная запись предоставляет механизм каналов при помощи символа |

<pre> - форматирование текста

1. Удаляем пункт 2. В тег mat-list-item добавляем оператор цикла

\*ngFor="let project of projects"

Далее выводим название и детали проектов

<h3 mat-line>

{{project.title}}

</h3>

<p mat-line>

{{project.details}}

</p>

**3.2. Event Handling**

1. В конструкцию

<mat-list-item (click)="selectproject(project \*ngFor="let project of projects">

Добавляем обработку события щелкок мыши. Получаем:

<mat-list-item (click)="selectproject(project)" \*ngFor="let project of projects">

**3.3 Inspect Raw DOM Event**

1. Добавляем вывод деталей о проекте во второе окно

<!-- PROJECT FORM HERE -->

<pre> {{selectedProject|json}}</pre>

**3.4 Unselecting an Element**

1. Добавляем отмену выбора проекта component.ts

cancel()

{

this.selectProject(null);

}

В projects.component.html :

<button type="button" mat-button (click)="cancel()">Cancel</button>

**3.5 ng-if & else**

1. Добавляем вывод заголовка и проверку его наличия

<h1>

<!-- PROJECT TITLE HERE -->

<!—main variant -->

<span \*ngIf="selectedProject; then thenBlock;else elseBlock"></span>

<ng-template #thenBlock>{{selectedProject.title}}</ng-template>

<ng-template #elseBlock>Select a Project</ng-template>

<!—alternative -->

{{selectedProject?selectedProject.title:'Select a Project'}}

</h1>

**3.6 Sidebar Toggle**

1. Добавляем сворачивание бокового меню по нажатия кнопки в файл app.component.html добавляем :

**<button mat-mini-fab (click)="sidenav.toggle()">**

<mat-icon>menu</mat-icon>

</button>

1. Добавляем заголовок приложению в файл app.component.html

<span class="title">

<!-- TITLE -->

**{{title}}**

</span>

В файле app.component.ts значение загловка:

title = 'Angular core workshop';

**3.7 Binding Styles**

1. Добавляем привязку свойства класса на примере привязки стиля. В файле projects.component.ts добавляем переменную:

primaryColor='red';

Применяем переменную для назначения стиля. В файле projects.component.html добавляем:

<h3 **[style.color]="primaryColor"** mat-line>

{{project.title}}

</h3>

# в html шаблоне для создания локальной ссылки на объект

Чтобы получить ссылку на объект из компонента необходимо создать ссылку в компоненте следующим образом:

@ViewChild (‘<имя ссылки компонента в шаблоне>’)<локальное имя>;

**4. Routing**

**4.1 Creating App Routing Module**

1. Создаем файл app-routing.module.ts на уровне app\*. Вставляем туда следующий текст:

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

import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

1. В файле app.module.ts заменяем:

RouterModule.forRoot([], { initialNavigation: 'enabled' }),

На созданный файл маршрутизации:

AppRoutingModule

**4.2 Login Component Routing**

1. В файле app-routing.module.ts редактируем массив маршрутов

const routes: Routes = [

{path:'',component:LoginComponent},

{path:'\*\*',redirectTo:'',pathMatch:'full'}

];

1. Создаем компонент входа пользователя LoginComponent

ng g c login --project=ui-login

1. В файле добавляем экспорт созданного компонента

exports:[LoginComponent]

1. В файле index.ts библиотеки ui-login меняем

export \* from './lib/ui-login.module';

На

export {UiLoginModule} from './lib/ui-login.module';

и добавляем запись

export {LoginComponent} from "./lib/login/login.component";

1. Добавляем созданный в модуль в главный модуль приложения app.module.ts в раздел imports:

UiLoginModule

1. В файле app.component.html из раздела <div class="app-content"> удаляем

<app-projects></app-projects>, чтобы передать управление системе маршрутизации, которую мы создали

**4.3 Lazy Loading Modules**

1. В файл home-routing.module.ts добавляем переменную

const routes: Routes = [

{path:'', component: HomeComponent}

];

1. В файле app-routing.module.ts ts заменяем

{path:'',component:LoginComponent},

На

{ path: '', loadChildren: './home/home.module#HomeModule' },

{path:'login',component:LoginComponent}

Чтобы по умолчанию приложение ссылалось на HomeComponent

1. В файл customers-routing.module.ts добавляем переменную

const routes: Routes = [

{path:'', component: CustomersComponent}

];

Тоже самое в файле projects-routing.module.ts

const routes: Routes = [

{ path: '', component: ProjectsComponent }

];

1. В файле app.component.ts добавляем массив:

links=[

{ path:'/',icon:'home',titele:'Home' },

{ path:'/customers',icon:'face',titele:'Customer' },

{ path:'/projects',icon:'work',title:'Projects' }

]

**4.4 Adding Navigation**

1. Заполняем боковую панель нашими проектами. В файле app.component.html

Заменяем

<a mat-button class="nav-link">

<mat-icon>

<!-- LINK ICON -->

</mat-icon>

<!-- LINK LABEL -->

</a>

На

<a mat-button class="nav-link" **\*ngFor="let link of links"**>

<mat-icon>

<!-- LINK ICON -->

**{{link.icon}}**

</mat-icon>

<!-- LINK LABEL -->

**{{link.title}}**

</a>

1. Добавляем переход по путям при нажатии на кнопку проекта для этого
2. В файле app.component.html

Заменяем

<a mat-button class="nav-link" \*ngFor="let link of links">

На

<a mat-button class="nav-link" \*ngFor="let link of links"

**[routerLink]="link.path" routerLinkActive="active"** >

1. В файле app.module.ts из секции imports удаляем:

HomeModule,

ProjectsModule,

CustomersModule.

Для того чтобы работала медленная загрузка

**5. Services**

**5.1 Creating a Service**

1. Создаем сервис projects в библиотеке core-data:

ng g s projects/projects --project=core-data

1. Создаем интерфейс projects в службе projects в библиотеке core-data:

ng g i projects/project --project=core-data

1. Переносим массив projects проектов из projects.component.ts в projects.service.ts и делаем его private
2. В файл projects.service.ts создаем метод all() возвращающий массив проектов

all()

{

return this.projects;

}

1. В интерфейс Project в файле project.ts добаляем структуру записи project

export interface Project {

id: string,

title: string,

details: string,

percentComplete: number,

approved: boolean

}

1. В файл projects.service.ts в массив добавляем объявленный интерфейс в качестве типа. Также делаем в методе all() добавляем тип возвращаемого значения:

all():Project[]

{

return this.projects;

}

1. В файле index.ts библиотеки core-data меняем:

export \* from './lib/core-data.module';

На

export {CoreDataModule} from './lib/core-data.module';

И добавляем экспорт :

export {Project} from "./lib/projects/project";

export {ProjectsService} from "./lib/projects/projects.service";

1. В файле core-data.module.ts в разделе NgModules добляем созданный сервис:

providers:[

ProjectsService

]

1. В файле projects.component.ts внедряем нам сервис :

constructor(**private projectsService:ProjectsService**) {

}

И создаем метод получения проектов:

getProjects()

{

this.projects=this.projectsService.all();

}

1. Добавляем инициализацию на данных на этапе инициализации компонента:

ngOnInit() {

this.getProjects();

}

И добляем типизацию данным типа Project

projects: Project[];

selectedProject: Project;

**5.2 HttpClient**

1. В файле core-data.module.ts и app.module.ts в раздел imports добавляем HttpClientModule
2. Добляем json-server в проект:

npm i json-server

1. В файле верхнего уровня package.json в scripts добавляем:

"server": "json-server server/db.json",

1. Создаем папку верхнего уровня server и в ней файл db.json в файл заносим наши проекты

{

"customers": [

{

"id": "1",

"firstName": "Jonathan",

"lastName": "Garvey",

"email": "jon@gmail.coms",

"phone": "1234567890",

"title": "Technical Lead",

"status": 1

},

{

"id": "2",

"firstName": "Bob",

"lastName": "Jones",

"email": "bob@bobjones.com",

"phone": "9372932391",

"title": "CEO",

"status": 2

},

{

"id": "3",

"firstName": "Lukas",

"lastName": "Ruebbelke",

"email": "hey.player@gmail.com",

"phone": "9876543212",

"title": "CEO",

"status": 1

},

{

"id": "4",

"firstName": "Sally",

"lastName": "McBride",

"email": "silly.sally@hotmail.com",

"phone": "3252352531",

"title": "Receptionist",

"status": 3

},

{

"id": "5",

"firstName": "Donna",

"lastName": "Barkley",

"email": "donnab@aol.com",

"phone": "0192837475",

"title": "Student",

"status": 4

},

{

"id": "6",

"firstName": "Larry",

"lastName": "Fitzgerald",

"email": "super.secret@gmail.com",

"phone": "1234432122",

"title": "Quarterback",

"status": 4

},

{

"id": "7",

"firstName": "Victor",

"lastName": "Wooten",

"email": "w00t@victor.com",

"phone": "1888333992",

"title": "Bassist",

"status": 1

},

{

"id": "8",

"firstName": "Chris",

"lastName": "Take",

"email": "crispy@gmail.com",

"phone": "9927381924",

"title": "Developer",

"status": 1

},

{

"id": "9",

"firstName": "Christian",

"lastName": "Peterson",

"email": "cp3@lol.com",

"phone": "0099831234",

"title": "Handyman",

"status": 3

},

{

"id": "10",

"firstName": "RIP",

"lastName": "Stick",

"email": "r.i.p@deadly.com",

"phone": "6123512311",

"title": "Grim Reaper",

"status": 2

}

],

"projects": [

{

"id": "1",

"title": "Project One",

"details": "This is a sample project",

"percentComplete": 20,

"approved": false,

"startDate": null,

"targetDate": null,

"completionDate": null,

"customerId": "1"

},

{

"id": "2",

"title": "Project Two",

"details": "This is a sample project",

"percentComplete": 40,

"approved": false,

"startDate": null,

"targetDate": null,

"completionDate": null,

"customerId": "3"

},

{

"id": "3",

"title": "Project Three",

"details": "This is a sample project",

"percentComplete": 60,

"approved": false,

"startDate": null,

"targetDate": null,

"completionDate": null,

"customerId": "7"

},

{

"id": "5",

"title": "Project Five",

"details": "This is a sample project",

"percentComplete": 100,

"approved": true,

"startDate": null,

"targetDate": "2018-10-26T07:00:00.000Z",

"completionDate": null,

"customerId": "1"

}

]

}

1. В файле projects.service.ts создаем переменные:

const BASE\_URL='http://localhost:3000/';

model='projects';

Удаляем массив проектов и меняем метод all () следующим образом:

all()

{

return **this.httpClient.get(`${BASE\_URL}${this.model}`)**

}

**5.3 Handling Observable with async Filter**

1. В файле projects.component.ts меняем объявление массива projects на:

projects$;

И метод getProjects() на:

this.projects$=this.projectsService.all();

$ - означает поток

1. В файле projects.component.html изменяем цикл обработки проектов на:

\*ngFor="let project of projects$|async">

**5.4 Create, Update & Delete**

1. Создаем Create, Update & Delete методы в сервисе проектов в файле projects.service.ts:

create(project)

{

return this.httpClient.post(this.getUrl(),project);

}

update(project)

{

return this.httpClient.patch(this.getUrlforId(project.id),project);

}

delete(projectId)

{

return this.httpClient.delete(this.getUrlforId(projectId));

}

И вспомогательные методы для построения адреса команд:

getUrl()

{

return `${BASE\_URL}${this.model}`

}

getUrlforId(id)

{

return `${this.getUrl()}/${id}`;

}

1. В файле projects.component.ts добавляем методы для Create, Update & Delete

deleteProject(project)

{

this.projectsService.delete(project.id).subscribe(resulet=>this.getProjects());

}

createProject(project)

{

this.projectsService.create(project).subscribe(result=>this.getProjects());

}

updateProject(project)

{

this.projectsService.update(project).subscribe(result=>this.getProjects());

}

1. В файле projects.component.html добавим удаление по нажатии кнопки крест:

<button mat-icon-button (click)="deleteProject(project); $event.stopImmediatePropagation()">

<mat-icon>close</mat-icon>

</button>

stopImmediatePropagation() – останавливает все остальные события вызванные нажатием на кнопку после отработки события deleteProject(project)

**6 Forms**

**6.1 Wiring Up the Form**

1. В основной модуль app.module.ts и projects.module.ts импортируем модуль работы с формами:

FormsModule

1. В файле projects.component.ts добавляем методы:

resetProject() {

const emptyProject: Project = {

id: null,

title: '',

details: '',

percentComplete: 0,

approved: false,

}

this.selectProject(emptyProject);

}

И

saveProject(project)

{

console.log('Saving!');

}

1. В файле projects.component.ts добавляем метод resetProject() в метод инициализации компонента для инициализации объекта selectedProject:

ngOnInit() {

this.getProjects();

**this.resetProject();**

}

и меняем содержимое метода cancel() на:

this.resetProject();

1. В файле шаблона projects.component.html вместо:

<pre> {{selectedProject|json}}</pre>

Вставляем:

<mat-form-field class="full-width">

<input matInput placeholder="Title" [(ngModel)]="selectedProject.title" type="text" name="title">

</mat-form-field>

**6.2 Creating Form Fields**

1. В файле шаблона projects.component.html вместо:

<span \*ngIf="selectedProject; then thenBlock;else elseBlock"></span>

Вставляем:

<span \*ngIf="selectedProject**.id**; then thenBlock;else elseBlock"></span>

1. В корневом файле стилей styles.scss добавляем:

mat-card {

box-shadow: 0 3px 5px -1px rgba(0, 0, 0, 0.2),

0 6px 10px 0 rgba(0, 0, 0, 0.14), 0 1px 18px 0 rgba(0, 0, 0, 0.12);

mat-card-header {

margin-bottom: 10px;

}

mat-form-field {

width: 100%;

}

}

1. В файле шаблона projects.component.html добавляем оставшиеся поля для ввода для объекта projects:

<mat-form-field class="full-width">

<textarea matInput placeholder="Details" [(ngModel)]="selectedProject.details" type="text"

name="details"></textarea>

</mat-form-field>

<section class="full-width">

<h4>{{selectedProject.percentComplete}}% Complete</h4>

<mat-slider class="full-width" min="0" max="100" thumbLabel name="percentComplete"

[(ngModel)]="selectedProject.percentComplete">

</mat-slider>

</section>

<section>

<mat-checkbox [(ngModel)]="selectedProject.approved" name="Approve">Approved by customer</mat-checkbox>

</section>

**6.3 Saving Form Data**

1. В файле компонента projects.component.ts обновляем методы createProject и updateProject

createProject(project)

{

this.projectsService.create(project)

.subscribe(result=>

{

this.getProjects();

this.resetProject();

});

}

updateProject(project)

{

this.projectsService.update(project)

.subscribe(result=>

{

this.getProjects();

this.resetProject();

});

}

1. В файле компонента projects.component.ts обновляем метод saveProject:

saveProject(project)

{

if(project.id)

{

this.updateProject(project);

}

else

{

this.createProject(project);

}

}

**7. Component Architecture**

**7.1 Refactor App Into Components**

1. Создаем компонент projects-list:

ng g c projects/projects-list

1. Создаем компонент projects-details:

ng g c projects/projects-details

1. Переносим HTML код отвечающий за список проектов в файл projects-list.component.html

Вместо него вставляем ссылку на компонент список проектов :

<div class="col-50">

**<app-projects-list ></app-projects-list>**

</div>

1. Переносим HTML код отвечающий за детали проектов в файл

projects-details.component.html

Вместо него вставляем ссылку на компонент деталей проектов :

<div class="col-50">

**<app-projects-details></app-projects-details>**

</div>

**7.2 List Components**

1. В файле компонента списка проектов projects-list.component.ts определяем входные свойства:

@Input()projects:Project[];

@Input()readonly=false;

1. В файле шаблона HTML компонента проектов projects.component.html добавляем передаваемый параметр [projects]="projects$|async"

<div class="col-50">

<app-projects-list [projects]="projects$|async"></app-projects-list>

</div>

И меняем шаблон компонента projects-list.component.html следующим образом:

<mat-list>

<mat-list-item

\*ngFor="let project of projects">

<h3 mat-line>

{{project.title}}

</h3>

<p mat-line>

{{project.details}}

</p>

<button \*ngIf="!readonly" mat-icon-button >

<mat-icon>close</mat-icon>

</button>

</mat-list-item>

</mat-list>

**7.3 Sending Events to a Container Component**

1. Создаем выходные свойства для компонента списка проектов projects-list.component.ts для вызова методов родительского компонента:

@Output()selected=new EventEmitter();

@Output()deleted=new EventEmitter();

1. В файле шаблона projects-list.component.html вызываем родительский метод:

<div class="col-50">

<app-projects-list

[projects]="projects$|async"

**(selected)="selectProject($event)"**

**(deleted)="deleteProject($event)"**

></app-projects-list>

</div>

**7.4 Details Component**

1. В файле компонента деталей проекта projects-details.component.ts определяем входные свойства:

@Input()project:Project;

1. В файле шаблона проектов projects.component.html добавляем передаваемый параметр:

<app-projects-details

**[project]="selectedProject"**

></app-projects-details>

1. Создаем выходные свойства для компонента деталей проекта projects-details.component.ts для вызова методов родительского компонента:

@Output() saved= new EventEmitter();

@Output() cancelled = new EventEmitter();

1. В файле шаблона проектов projects.component.html вызываем родительский метод:

<mat-card-actions>

<button type="submit" mat-button color=**"primary" (click)="saved.emit(project)"**>Save</button>

<button type="button" mat-button **(click)="** **cancelled.emit()"** >Cancel</button>

</mat-card-actions>

**7.5 Fix Shared Mutable State**

1. В файле projects-details.component.ts создаем 2 переменные:

currentProject:Project;

originalTitle;

1. В том же файле меняем

@Input()project:Project;

На

@Input()set project(value)

{

this.currentProject=Object.assign({},value);

if (value)

this.originalTitle=value.title;

}

1. В файле шаблона деталей projects-details.component.html меня переменную project на currentProject, а переменную project.title на originalTitle

**8. Testing**

**8.1 Testing Overview**

1. Исправляем тестирование согласно ветки в guthub помеченной как “update: fixing tests in dashoard”
2. Запускаем тест npm test dashboard

**8.2 Creating a Component Test**

1. В файле [home.component.spec.ts](http://home.component.spec.ts) убираем все содержимое {} после describe('HomeComponent', () =>
2. Инициализация теста:

// instantiate test bed

beforeEach(() => {

fixture = TestBed.configureTestingModule(

{

declarations: [

ProjectsComponent,

ProjectsListComponent,

ProjectsDetailsComponent

],

imports: [

MaterialModule,

FormsModule,

HttpClientModule,

BrowserAnimationsModule

]

}).createComponent(ProjectsComponent);

component = fixture.componentInstance;

de = fixture.debugElement;

fixture.detectChanges();

});

1. В файл шаблона projects.component.html добавляем <h1>{{primaryColor}}</h1>
2. Добавляем первый тест, добавленной строки в [home.component.spec.ts](http://home.component.spec.ts)

it('Should have primary color of "red"',()=>{

expect(component.primaryColor).toBe('red')

})

**8.3 Unit Testing a Component Method**

1. В файле теста projects.component.spec.ts добавляем переменную:

const emptyProject: Project = {

id: null,

title: '',

details: '',

percentComplete: 0,

approved: false,

}

1. Создаем второй тест:

it('shoud select project',()=>{

component.selectProject(emptyProject);

expect(component.selectedProject).toBe(emptyProject);

});

**8.4 Debug Element**

1. Создаем третий тест, чтобы проверить состояние элемента, через поиск его на страницы при помощи css:

it('should display primaryColor',()=>{

const h1=de.query(By.css('h1'));

expect(h1.nativeElement.innerText).toBe('red');

});

1. Создаем четвертый тест, чтобы проверить состояние элемента после того как мы изменим его свойство, через поиск его на страницы при помощи css:

it('shpild update new primaryColor',()=>{

const h1=de.query(By.css('h1'));

// changing color

component.primaryColor='black';

// manually force change detection

fixture.detectChanges();

expect(h1.nativeElement.innerText).toBe('black');

})

**8.5 Services**

1. Создаем mock ProjectsService для этого объявляем переменную:

let projectService:ProjectsService;

затем:

const mockprojectService={

// mock yourself out

all() { return noop() }

}

И между declarations и imports вставляем:

providers:[

{provide: ProjectsService, useValue: mockprojectService}

],

Чтобы заставить использовать mockprojectService вместо ProjectsService:

projectService=de.injector.get(ProjectsService);