Contents

Task 01. Import Modules

1. Добавьте в файле **app.module.ts** следующий фрагмент кода:

```
import { FormsModule } from '@angular/forms';
import { HttpModule, JsonpModule } from '@angular/http';

imports: [
    BrowserModule,
    CommonModule,
    FormsModule,
    TasksModule,
    UsersModule,
    HttpModule,
    JsonpModule,
    routing
]
```

Task 02. Simulating Web API

1. npm install -g json-server
2. create file \db\db.json следующего содержания

```
{
      "tasks": [
            { "id": 1, "action": "Estimate", "priority": 1, "estHours": 8},
            { "id": 2, "action": "Create", "priority": 2, "estHours": 8},
            { "id": 3, "action": "Edit", "priority": 3, "estHours": 4},
            { "id": 4, "action": "Delete", "priority": 3, "estHours": 2},
            { "id": 5, "action": "Build", "priority": 1, "estHours": 4},
            { "id": 6, "action": "Deploy", "priority": 2, "estHours": 8}
      ]
}
```

3. run json-server from cmd: json-server --watch db\db.json

Task 03. Task Promise Service

1. Создайте файл **tasks/task-promise.service.ts** используя следующий фрагмент кода

```
import { Injectable } from '@angular/core';
import { Http } from '@angular/http';
import 'rxjs/add/operator/toPromise';

import { Task } from './../models/task';

@Injectable()
export class TaskPromiseService {
  private tasksUrl: string = 'http://localhost:3000/tasks';

  constructor(
    private http: Http
  ) {}

  getTasks(): Promise<Task[]> {
    return this.http.get(this.tasksUrl)
            .toPromise()
            .then( response => response.json() as Task[] )
            .catch(this.handleError);
  }

  private handleError(error: any): Promise<any> {
    console.error('An error occurred', error);
    return Promise.reject(error.message || error);
  }
}
```

2. Внесите изменения в файл **tasks/tasks.module.ts**

```
import { TaskArrayService } from './task-array-service/task-array.service';
import { TaskPromiseService } from './task-promise.service';

providers: [
    TaskArrayService,
    TaskPromiseService
]
```

3. Внесите изменения в компонент TaskListComponent

```
import { TaskArrayService } from './../task-array-service/task-array.service';
import { TaskPromiseService } from './../task-promise.service';

constructor(
    private tasksService: TaskArrayService,
    private taskPromiseService: TaskPromiseService
) { }

ngOnInit() {
    this.tasksService.getTasks()
```

```
  this.taskPromiseService.getTasks()
    .then(tasks => this.tasks = tasks);
}
```

Task 04. GetTask

1. Добавьте метод getTask в сервис TaskPromiseService используя следующий фрагмент кода

```
getTask(id: number): Promise<Task> {
    return this.http.get(`this.tasksUrl/${id}`)
            .toPromise()
            .then( response => response.json() as Task )
            .catch( this.handleError );
  }
```

2. Внесите изменения в компонент TaskFormComponent

```
import { TaskArrayService } from './../task-array-service/task-array.service';
import { TaskPromiseService } from './../task-promise.service';

constructor(
    private tasksService: TaskArrayService,
    private tasksPromiseService: TaskPromiseService,
    private router: Router,
    private route: ActivatedRoute
  ) { }
```

3. Внесите изменения в метод ngOnInit компонента TaskFormComponent

```
if (id) {
        this.tasksService.getTask(id)
        this.tasksPromiseService.getTask(id)
          .then(task => this.task = Object.assign({}, task));
}
```

Task 05. UpdateTask

1. Внесите изменения в TaskPromiseService

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

2. Добавьте метод updateTask в сервис TaskPromiseService используя следующий фрагмент кода

```
updateTask(task: Task): Promise<Task> {
    let url = `${this.tasksUrl}/${task.id}`,
        body = JSON.stringify(task),
        headers = new Headers({'Content-Type': 'application/json'}),
        options = new RequestOptions();

    options.headers = headers;

    return this.http.put(url, body, options)
            .toPromise()
            .then( response => response.json() as Task )
            .catch( this.handleError );
  }
```

3. Внесите изменения в метод saveTask компонента TaskFormComponent

```
if (task.id) {
    this.tasksService.updateTask(task);
    this.tasksPromiseService.updateTask(task)
        .then( () => this.goBack() );
  }
  else {
    this.tasksService.addTask(task);
    this. this.goBack();
  }

  this.router.navigate(["home"]);
```

4. Внесите изменения в метод completeTask компонента TaskListComponent

```
completeTask(task: Task): void {
    task.done = true;
    this.taskPromiseService.updateTask(task);
}
```

Task 06. CreateTask

1. Внесите изменения в темплейт компонента TaskListComponent использу следующий фрагмент разметки

```
<div>
      <button class="btn btn-primary"
            (click)="createTask()">New Task</button>
      <br><br>
      <task
        *ngFor='let task of tasks'
        [task]="task"
        (onComplete)="completeTask($event)">
      </task>
</div>
```

2. Внесите изменения в компонент TaskListComponent

```
import { Router } from '@angular/router';

constructor(
    private taskPromiseService: TaskPromiseService,
    private router: Router
  ) { }
```

3. Добавьте метод createTask в компонент TaskListComponent используя следующий фрагмент кода

```
createTask() {
  let link = ['add'];
  this.router.navigate(link);
 }
```

4. Внесите изменения в файл tasks/tasks.routing.ts

```
const tasksRoutes: Routes = [
  {
    path: 'home',
    component: TaskListComponent
  },
  {
    path: 'add',
    component: TaskFormComponent
  },
  {
    path: 'edit/:id',
    component: TaskFormComponent
  }
];
```

5. Добавьте метод createTask в сервис TaskPromiseService используя следующий фрагмент кода

```
createTask(task: Task): Promise<Task> {
    let url = this.tasksUrl,
```

```
        body = JSON.stringify(task),
        headers = new Headers({'Content-Type': 'application/json'}),
        options = new RequestOptions({headers: headers});

    return this.http.post(url, body, options)
            .toPromise()
            .then( response => response.json() as Task )
            .catch( this.handleError );
  }
```

6. Внесите изменения в метод saveTask компонента TaskFormComponent используя следующий фрагмент кода

```
if (task.id) {
    this.tasksPromiseService.updateTask(task)
        .then( () => this.goBack() );
  }
  else {
    this.tasksService.addTask(task);
    this.goBack();
  }
let method = task.id ? "updateTask" : "createTask";
  this.tasksPromiseService[method](task)
    .then( () => this.goBack() )
```

Task 07. DeleteTask

1. Внесите изменения в темплейт компонента TaskComponent используя следующий фрагмент разметки

```
<div class="panel panel-default">
      <div class="panel-heading">Task</div>
      <div class="panel-body">
            <ul>
                  <li>Action: {{task.action}}</li>
                  <li>Priority: {{task.priority}}</li>
                  <li>Estimate Hours: {{task.estHours}}</li>
                  <li>Actual Hours: {{task.actHours}}</li>
                  <li>Done: {{task.done}}</li>
            </ul>
            <button class="btn btn-primary btn-sm"
                  (click)="completeTask($event)">
                  Done
            </button>
            <button class="btn btn-warning btn-sm"
                  (click)="editTask(task)">
                  Edit
            </button>
            <button class="btn btn-danger btn-sm"
                  (click)="deleteTask(task)">
                  Delete
            </button>
      </div>
</div>
```

2. Внесите изменения в компонент TaskComponent используя следующий фрагмент кода:

```
@Output() onComplete = new EventEmitter<Task>();
@Output() onDelete = new EventEmitter<Task>();

deleteTask(task: Task) {
    this.onDelete.emit(task);
}
```

3. Внесите изменения в темплейт компонента TaskListComponent используя следующий фрагмент разметки

```
<task
        *ngFor='let task of tasks'
        [task]="task"
        (onComplete)="completeTask($event)"
        (onDelete)="deleteTask($event)">
</task>
```

4. Добавьте метод deleteTask в компонент TaskListComponent используя следующий фрагмент разметки

```
deleteTask(task: Task) {
```

```
    this.taskPromiseService.deleteTask(task)
      .then( () => {
        this.tasks = this.tasks.filter(t => t !== task);
      });
}
```

5. Добавьте метод deleteTask в сервис TaskPromiseService используя следующий фрагмент разметки

```
deleteTask(task: Task): Promise<Task> {
    let url = `${this.tasksUrl}/${task.id}`;

    return this.http.delete(url)
            .toPromise()
            .then( response => response.json() as Task )
            .catch( this.handleError );
}
```

Task 08. SearchTask

1. Создайте сервис TaskSearchService используя следующий фрагмент кода

```
import { Injectable } from '@angular/core';
import { Http, Response, RequestOptions, URLSearchParams } from '@angular/http';

import { Observable } from 'rxjs/Observable';
import 'rxjs/add/operator/map';

import { Task } from './../models/task';

@Injectable()
export class TaskSearchService {
  private tasksUrl: string = 'http://localhost:3000/tasks';

  constructor(
    private http: Http
  ) {}

  search(term: string): Observable<Task[]> {
    let url = this.tasksUrl,
      urlParams: URLSearchParams = new URLSearchParams(),
      options: RequestOptions = new RequestOptions();

    urlParams.set('action_like', term);
    options.search = urlParams;

    return this.http.get(url, options)
            .map((r: Response) => r.json() as Task[]);
  }
}
```

2. Внесите изменения в tasks/tasks.module.ts

```
import { TaskArrayService } from './task-array-service/task-array.service';
import { TaskPromiseService } from './task-promise.service';
import { TaskSearchService } from './task-search.service';

providers: [
    TaskArrayService,
    TaskPromiseService,
    TaskSearchService
]
```

3. Внесите изменения в темплейт компонента TaskListComponent используя следующий фрагмент разметки

```
<button class="btn btn-primary"
            (click)="createTask()">New Task</button>
<button class="btn btn-primary"
      (click)="searchTask()">Search Task</button>
```

4. Внесите изменения в компонент TaskListComponent

```
searchTask() {
    let link = ['search'];
    this.router.navigate(link);
}
```

5. Внесите изменения в tasks/tasks.routing.ts

```
{
    path: 'add',
    component: TaskFormComponent
  },
  {
    path: 'search',
    component: TaskSearchComponent
  },
```

6. Внесите изменения в разметку компонента TaskSearchComponent

```
Search by Action: <input #searchBox id="search-box"
(keyup)="search(searchBox.value)">

<br><br>

<task

  *ngFor='let task of tasks | async'

  [task]="task">

</task>
```

7. Создайте файл rxjs-extensions.ts используя следующий фрагмент кода

```
// Observable class extensions
import 'rxjs/add/observable/of';
```

```
import 'rxjs/add/observable/throw';

// Observable operators
import 'rxjs/add/operator/catch';
import 'rxjs/add/operator/debounceTime';
import 'rxjs/add/operator/distinctUntilChanged';
import 'rxjs/add/operator/do';
import 'rxjs/add/operator/filter';
import 'rxjs/add/operator/map';
import 'rxjs/add/operator/switchMap';
```

8. Внесите изменения в компонент TaskSearchComponent

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

import { Observable } from 'rxjs/Observable';
import { Subject } from 'rxjs/Subject';

import './../../rxjs-extensions';

import { Task } from './../../models/task';
import { TaskSearchService } from './../task-search.service';

@Component({
  selector: 'task-search',
  templateUrl: 'task-search.component.html',
  styleUrls: ['task-search.component.css']
})
export class TaskSearchComponent implements OnInit {
  tasks: Observable<Task[]>;
  private searchTerms = new Subject<string>();

  constructor(
    private taskSearchService: TaskSearchService
  ) { }

  ngOnInit() {
    this.tasks = this.searchTerms
      .debounceTime(300)        // wait for 300ms pause in events
      .distinctUntilChanged()   // ignore if next search term is same as
previous
      .switchMap(term => term   // switch to new observable each time
        ? this.taskSearchService.search(term)
        : Observable.of<Task[]>([]))
      .catch(error => {
        console.log(error);
        return Observable.of<Task[]>([]);
      });
  }

  search(term: string): void {
    this.searchTerms.next(term);
```

```
    }
}
```

Task 09. Task Observable Service

1. Создайте сервис TaskObservableService в файле **tasks/task-observable.service.ts** используя следующий фрагмент кода

```
import { Injectable } from '@angular/core';
import { Http, Headers, Response, RequestOptions } from '@angular/http';

import { Observable } from 'rxjs/Observable';
import './../rxjs-extensions';

import { Task } from './../models/task';

@Injectable()
export class TaskObservableService {
  private tasksUrl: string = 'http://localhost:3000/tasks';

  constructor(
    private http: Http
  ) {}

  getTasks(): Observable<Task[]> {
    return this.http.get(this.tasksUrl)
             .map( this.handleData )
             .catch( this.handleError );
  }

  getTask(id: number) {

  }

  updateTask(task: Task) {

  }

  createTask(task: Task) {

  }

  deleteTask(task: Task) {

  }

  private handleData(response: Response) {
    let body = response.json();
    return body || {};
  }

  private handleError(error: any) {
    let errMsg = (error.message)
                   ? error.message
                   : error.status
                       ? `${error.status} - ${error.statusText}`
```

```
                        : 'Server error';
    console.error(errMsg);
    return Observable.throw(errMsg);
  }
}
```

2. Внесите изменения в файл tasks/tasks.module.ts

```
import { TaskArrayService } from './task-array-service/task-array.service';
import { TaskPromiseService } from './task-promise.service';
import { TaskObservableService } from './task-observable.service';
import { TaskSearchService } from './task-search.service';

providers: [
    TaskArrayService,
    TaskPromiseService,
    TaskObservableService,
    TaskSearchService
  ]
```

3. Внесите изменения в TaskListComponent

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

import { Task } from './../../models/task';
import { TaskPromiseService } from './../task-promise.service';
import { TaskObservableService } from './../task-observable.service';

export class TaskListComponent implements OnInit, OnDestroy {

tasks: Array<Task>;
errorMessage: string;
private sub: Subscription[] = [];

constructor(
    private taskPromiseService: TaskPromiseService,
    private taskObservableService: TaskObservableService,
    private router: Router
  ) { }

ngOnInit() {
    this.taskPromiseService.getTasks()
       .then(tasks => this.tasks = tasks);

    let sub = this.taskObservableService.getTasks()
      .subscribe(
         tasks => this.tasks = tasks,
         error => this.errorMessage = <any>error
      );
    this.sub.push(sub);
  }
```

```
ngOnDestroy() {
    this.sub.forEach(sub => sub.unsubscribe());
}
```

Task 10. CreateTask

1. Внесите изменения в сервис TaskObservableService используя следующий фрагмент кода

```
createTask(task: Task): Observable<Task> {
    let url = this.tasksUrl,
        body = JSON.stringify(task),
        headers = new Headers({'Content-Type': 'application/json'}),
        options = new RequestOptions({headers: headers});

    return this.http.post(url, body, options)
            .map( this.handleData )
            .catch( this.handleError );
  }
```

2. Внести изменения в компонент TaskFormComponent используя следующий фрагмент кода

```
import { TaskArrayService } from './../task-array-service/task-array.service';
import { TaskPromiseService } from './../task-promise.service';
import { TaskObservableService } from './../task-observable.service';

constructor(
    private tasksService: TaskArrayService,
    private tasksPromiseService: TaskPromiseService,
    private tasksObservableService: TaskObservableService,
    private router: Router,
    private route: ActivatedRoute
  ) { }

let method = task.id ? "updateTask" : "createTask";
this.tasksPromiseService[method](task)
      .then( () => this.goBack() );
    // TODO:
    // 1. method update is not implemented in tasksObservableService
    // 2. implement unsubscribe
    if (task.id) {
      this.tasksPromiseService[method](task)
      .then( () => this.goBack() );
    }
    else {
      this.tasksObservableService[method](task)
      .subscribe( () => this.goBack() );
    }
```