

Introduction to Web Development

Part 4: Demo Angular – SPA

arnaud.nauwynck@gmail.com

Course Esilv 2023

This document:

<https://github.com/Arnaud-Nauwynck/presentations/web/angular-demos-4-angular-spa.pdf>

Demo Outline

Reminder : http client/server, SPA: Single Page Application

Classical Web app tutorial: « Todo web application »

Rest Api Backend Server – NodeJS, Express, Swagger

Angular + ng-bootstrap setup

Component Pages (offline / mock data)

- List Page
- Duplicate/Delete button
- Detail Page
- New Page
- links between pages

Service

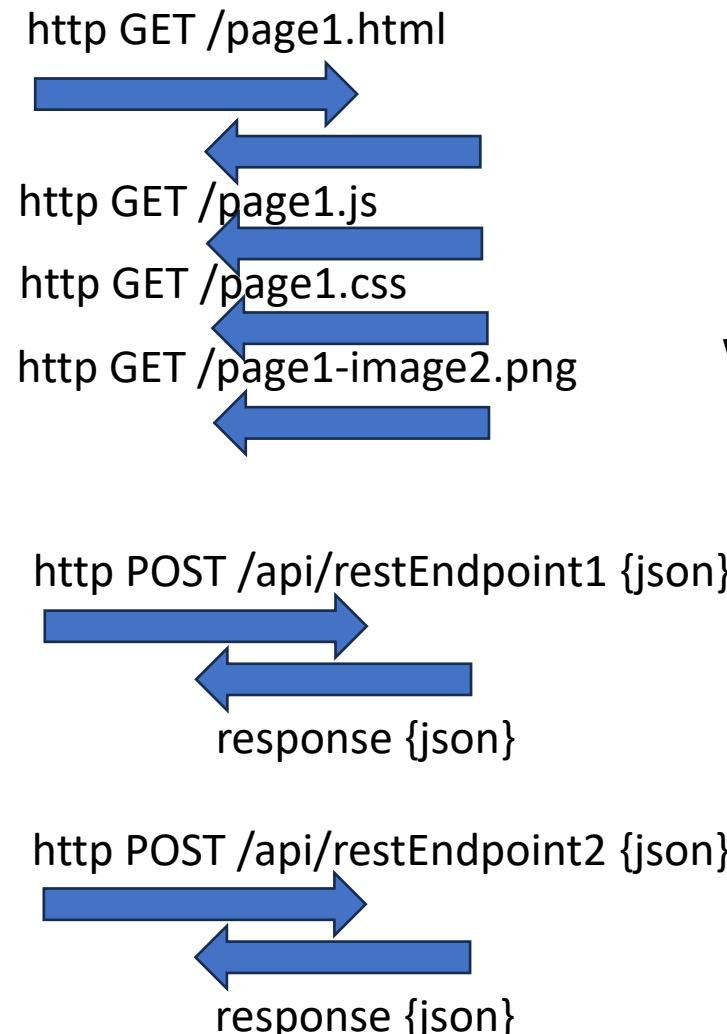
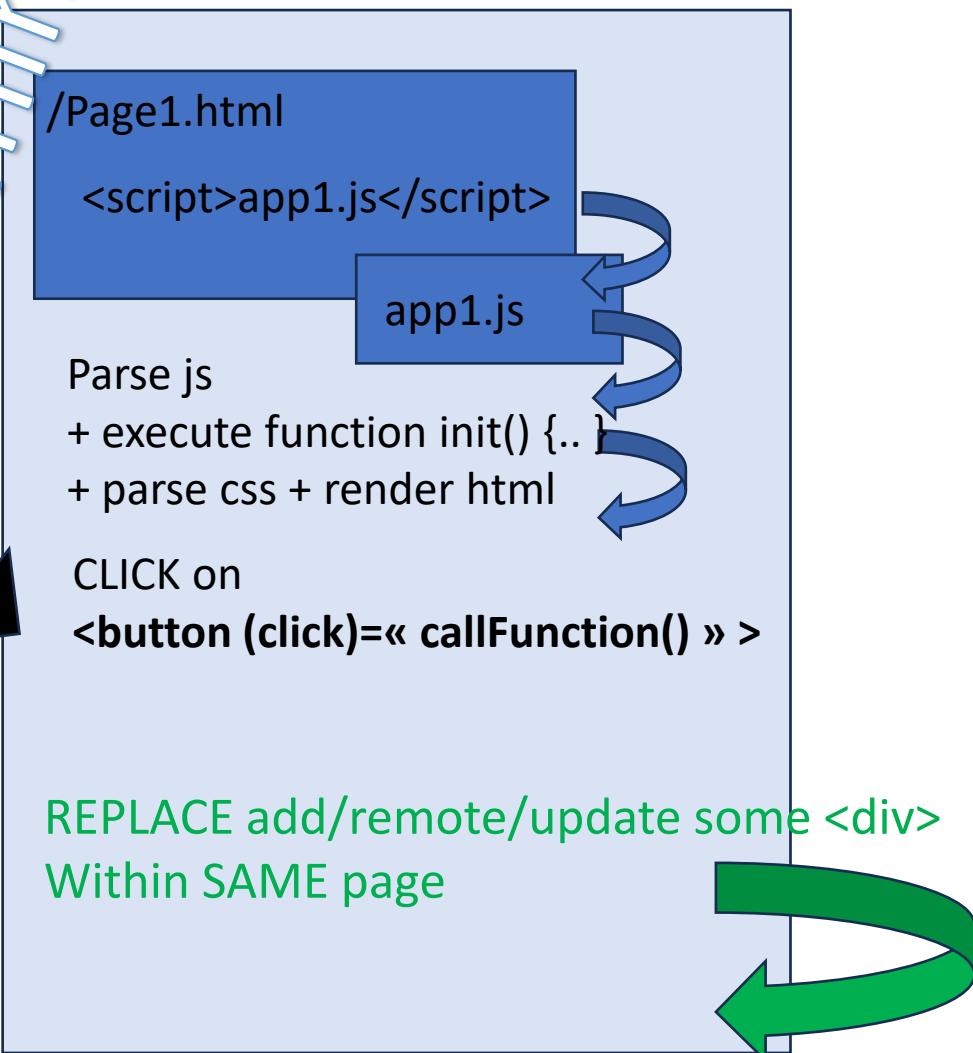
HttpClient (call to API backend)

Advanced (ag-grid, ...)

SPA = Single Page Application

single GET {html|js}, Multiple Rest Json requests

Reminder



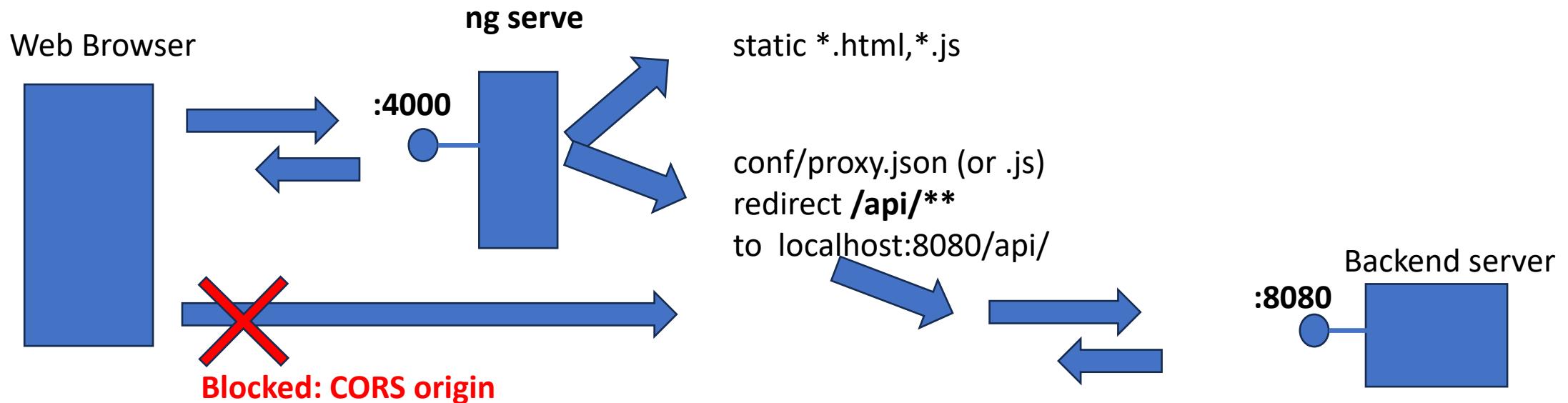
Web Server

Reminder FrontEnd / Rest API Backend CORS Origin problem !

Because ...

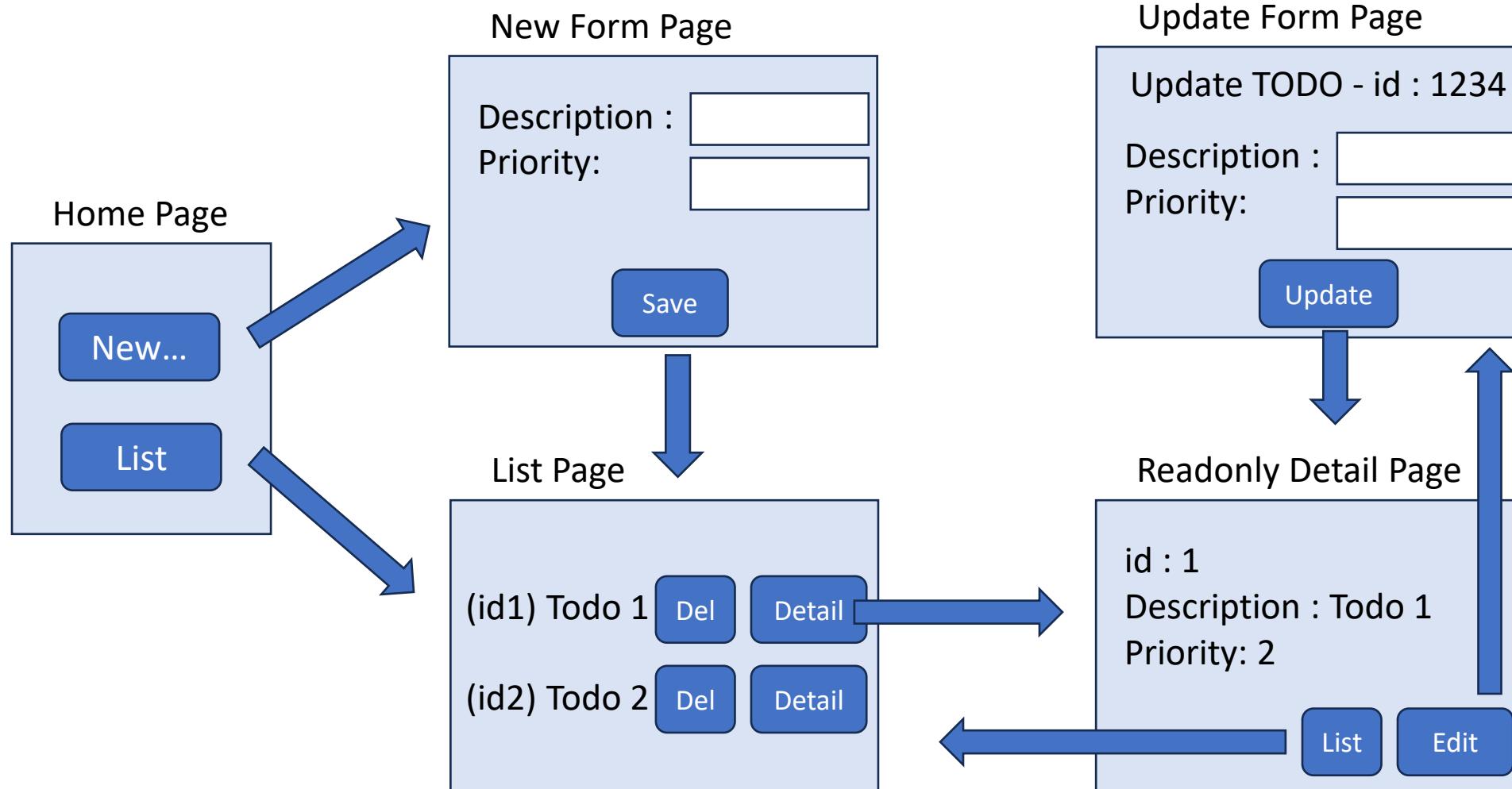
Web Browser blocks calls to « http://localhost:**8080**/api/** »

not comming from same « origin domain » as « http://localhost:**4000**/index.html » «app.js» !



Reminder

Reminder : Todo App Web Site Sketch



Todo Web App Features Description ... CRUD

- C **Create** → Page 1: Simple Html Form with input fields
Click on « Save » button => save « Todo » object on server
- R **Read All** → Page 2: Show list of « Todos »
- Read By Id** → page 3 for url « /todo/{id} »
Show detailed « Todo» page
- U **Update** → Page 4: Editable page + « update » button
- D **Delete** → « delete » button

Rest using NodeJs - Express

```
const express = require('express');
const app = express();

app.get('/', function (req, res) {
  res.send({ some: 'Hello Json Express' })
});

app.listen(3000);
```

Reminder

Swagger-UI : <http://localhost:3000/api-docs/>

The screenshot shows the Swagger-UI interface for a Todo Rest API. At the top, there's a header bar with a back arrow, a refresh icon, a search icon, a star icon, a puzzle piece icon, a square icon, a purple circular icon with a white letter 'A', and a three-dot menu icon. Below the header is a dark navigation bar with the 'Swagger' logo and 'Supported by SMARTBEAR'. The main content area has a title 'Todo Rest API'. On the left, there's a 'Servers' dropdown set to 'http://localhost:3000'. The main area displays a 'default' endpoint with five operations: GET /todo (blue), POST /todo (green), PUT /todo (orange), GET /todo/:id (blue), and DELETE /todo/:id (red). Each operation has a collapse/expand arrow on the right.

localhost:3000/api-docs/

Swagger
Supported by SMARTBEAR

Todo Rest API

Servers

http://localhost:3000

default

GET /todo

POST /todo

PUT /todo

GET /todo/:id

DELETE /todo/:id

Demo Outline

Reminder : http client/server, SPA: Single Page Application

Classical Web app tutorial: « Todo web application »

Rest Api Backend Server – NodeJS, Express, Swagger



Angular + ng-bootstrap setup

Component Pages (offline / mock data)

- List Page
- Duplicate/Delete button
- Detail Page
- New Page
- links between pages

Service

HttpClient (call to API backend)

Advanced (ag-grid, ...)

\$ ng new

```
$ ng new demo4-todo-app
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE demo4-todo-app/angular.json (2740 bytes)
CREATE demo4-todo-app/package.json (1045 bytes)
CREATE demo4-todo-app/README.md (1066 bytes)
CREATE demo4-todo-app/tsconfig.json (901 bytes)
CREATE demo4-todo-app/.editorconfig (274 bytes)
CREATE demo4-todo-app/.gitignore (548 bytes)
CREATE demo4-todo-app/tsconfig.app.json (263 bytes)
CREATE demo4-todo-app/tsconfig.spec.json (273 bytes)
CREATE demo4-todo-app/.vscode/extensions.json (130 bytes)
CREATE demo4-todo-app/.vscode/launch.json (470 bytes)
CREATE demo4-todo-app/.vscode/tasks.json (938 bytes)
CREATE demo4-todo-app/src/main.ts (214 bytes)
CREATE demo4-todo-app/src/favicon.ico (948 bytes)
CREATE demo4-todo-app/src/index.html (298 bytes)
CREATE demo4-todo-app/src/styles.css (80 bytes)
CREATE demo4-todo-app/src/app/app-routing.module.ts (245 bytes)
CREATE demo4-todo-app/src/app/app.module.ts (393 bytes)
CREATE demo4-todo-app/src/app/app.component.html (23115 bytes)
CREATE demo4-todo-app/src/app/app.component.spec.ts (1015 bytes)
CREATE demo4-todo-app/src/app/app.component.ts (218 bytes)
CREATE demo4-todo-app/src/app/app.component.css (0 bytes)
CREATE demo4-todo-app/src/assets/.gitkeep (0 bytes)
/ Installing packages (npm)...
```

\$ git add . ; git ci -m « init »

```
| $ git st
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
(use "git restore --staged <file>..." to unstage)
  new file:  .editorconfig
  new file:  .gitignore
  new file:  .vscode/extensions.json
  new file:  .vscode/launch.json
  new file:  .vscode/tasks.json
  new file:  README.md
  new file:  angular.json
  new file:  package-lock.json
  new file:  package.json
  new file:  src/app/app-routing.module.ts
  new file:  src/app/app.component.css
  new file:  src/app/app.component.html
  new file:  src/app/app.component.spec.ts
  new file:  src/app/app.component.ts
  new file:  src/app/app.module.ts
  new file:  src/assets/.gitkeep
  new file:  src/favicon.ico
  new file:  src/index.html
  new file:  src/main.ts
  new file:  src/styles.css
  new file:  tsconfig.app.json
  new file:  tsconfig.json
  new file:  tsconfig.spec.json

arnaud@DesktopArnaud /cygdrive/c/arn/perso/cours/esilv/web/angular-demos/demo4-todo-app
| $ git ci -m "init"
[main 0feeb6c] init
 23 files changed, 20784 insertions(+)
 create mode 100755 demo4-todo-app/.editorconfig
```

\$ ng add @ng-bootstrap/ng-bootstrap

```
$ ng add @ng-bootstrap/ng-bootstrap
i  Using package manager: npm
✓  Found compatible package version: @ng-bootstrap/ng-bootstrap@15.1.0.
✓  Package information loaded.

The package @ng-bootstrap/ng-bootstrap@15.1.0 will be installed and executed.
Would you like to proceed? Yes
✓  Packages successfully installed.
UPDATE package.json (1185 bytes)
✓  Packages installed successfully.
UPDATE src/app/app.module.ts (464 bytes)
UPDATE angular.json (2807 bytes)
UPDATE src/main.ts (259 bytes)
UPDATE tsconfig.app.json (294 bytes)
UPDATE tsconfig.spec.json (300 bytes)
```

```
$ git add .; git ci -m "add @ng-bootstrap";
[main 3186a7f] add @ng-bootstrap
 7 files changed, 206 insertions(+), 269 deletions(-)
```

Demo Outline

Reminder : http client/server, SPA: Single Page Application

Classical Web app tutorial: « Todo web application »

Rest Api Backend Server – NodeJS, Express, Swagger

Angular + ng-bootstrap setup

Component Pages (offline / mock data)



- List Page
- Duplicate/Delete button
- Detail Page
- New Page
- links between pages

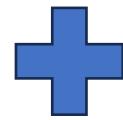
Service

HttpClient (call to API backend)

Advanced (ag-grid, ...)

\$ ng g c todo-list-page

```
$ ng g c todo-list-page
CREATE src/app/todo-list-page/todo-list-page.component.html (29 bytes)
CREATE src/app/todo-list-page/todo-list-page.component.spec.ts (603 bytes)
CREATE src/app/todo-list-page/todo-list-page.component.ts (232 bytes)
CREATE src/app/todo-list-page/todo-list-page.component.css (0 bytes)
UPDATE src/app/app.module.ts (574 bytes)
```



Basic HTML : <li *ngFor=<< let todo of todos >>>

Hardcoded Todo objects in *.ts

Defining model with Interface or Class

```
2  /**
3   * structural interface for Todo
4   * (example: for exchanging via json)
5   */
6  export interface Todo {
7    id: number;
8    description: string;
9    priority: number;
10 }
```

```
12 /**
13  * model class for Todo
14 */
15 export class TodoModel {
16   id: number;
17   description: string;
18   priority: number;
19
20   constructor(id: number, description: string, priority: number) {
21     this.id = id;
22     this.description = description;
23     this.priority = priority;
24   }
25 }
```

Todo list page

```
$ ng g c TodoListPage
```

todo-list-page.component.html

```
1 <H1>Todo list</H1>
2
3 <ul>
4   <li *ngFor="let todo of todos; index as idx">
5     {{todo.id}}) {{todo.description}} {{todo.priority}}
6   </li>
7 </ul>
```

todo-list-page.component.ts

```
1 import { Component } from '@angular/core';
2 import { Todo, TodoModel } from '../model/todo';
3
4 @Component({
5   selector: 'app-todo-list-page',
6   templateUrl: './todo-list-page.component.html'
7 })
8 export class TodoListPageComponent {
9
10   todos: TodoModel[] = [
11     new TodoModel(1, "learn angular", 1),
12     new TodoModel(2, "learn typescript", 1),
13     new TodoModel(3, "learn http", 1),
14   ];
15
16   todoInterfaces: Todo[] = [
17     {id: 1, description: "learn angular", priority: 1},
18     {id: 2, description: "learn typescript", priority: 1},
19     {id: 3, description: "learn http", priority: 1},
20   ];
21
22 }
```

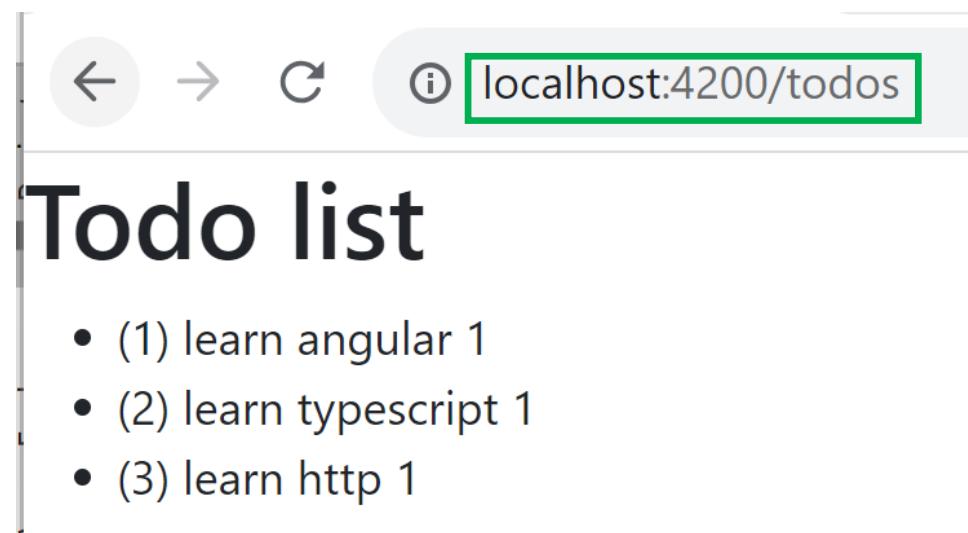
Todo router page « /todos »

Add to router url: « /todos »

```
app-routing.module.ts
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { TodoListPageComponent } from './todo-list-page/todo-list-page.component';
4
5 const routes: Routes = [
6   { path: 'todos', component: TodoListPageComponent },
7 ];
```

Use «router-outlet» in main page

```
app.component.html
1 <router-outlet></router-outlet>
2
```



Demo Outline

Reminder : http client/server, SPA: Single Page Application

Classical Web app tutorial: « Todo web application »

Rest Api Backend Server – NodeJS, Express, Swagger

Angular + ng-bootstrap setup

Component Pages (offline / mock data)



- List Page
- Duplicate/Delete button
- Detail Page
- New Page
- links between pages

Service

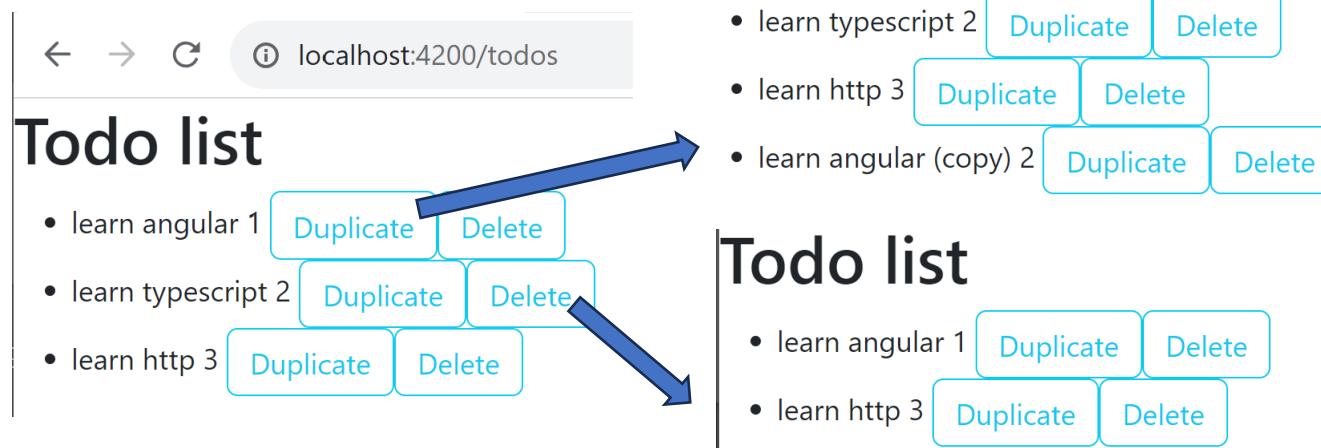
HttpClient (call to API backend)

Advanced (ag-grid, ...)

Button to duplicate / delete Todo

```
7 <button type="button" class="btn btn-outline-info"
8   (click)="duplicateTodo(todo, idx)">Duplicate</button>
9
10 <button type="button" class="btn btn-outline-info"
11   (click)="deleteTodo(todo, idx)">Delete</button>
```

```
19 duplicateTodo(item: TodoModel, idx: number) {
20   let newItem = this.todos.length + 1;
21   let dupItem = new TodoModel(newId, item.description + ' (copy)',
22                             item.priority+1);
23   this.todos.push(dupItem);
24 }
25
26 deleteTodo(item: TodoModel, idx: number) {
27   this.todos.splice(idx, 1);
28 }
```



Demo Outline

Reminder : http client/server, SPA: Single Page Application

Classical Web app tutorial: « Todo web application »

Rest Api Backend Server – NodeJS, Express, Swagger

Angular + ng-bootstrap setup

Component Pages (offline / mock data)

- List Page
- Duplicate/Delete button
- Detail Page
- New Page
- links between pages



Service

HttpClient (call to API backend)

Advanced (ag-grid, ...)

New Todo basic « Form »

```
$ ng g c TodoCreateSimple
CREATE src/app/todo-create-simple/todo-create-simple.component.html (33 bytes)
CREATE src/app/todo-create-simple/todo-create-simple.component.spec.ts (631 bytes)
CREATE src/app/todo-create-simple/todo-create-simple.component.ts (248 bytes)
CREATE src/app/todo-create-simple/todo-create-simple.component.css (0 bytes)
UPDATE src/app/app.module.ts (700 bytes)
```

The image shows a code editor with two tabs: `todo-create-simple.component.html` and `todo-create-simple.component.ts`.

todo-create-simple.component.html

```
<H1>Create TODO</H1>
<div class="container">
  <div class="row">
    <label for="description" class="col-md-3">Description</label>
    <input id="description" class="col-md-5" type="text" [(ngModel)]="todo.description">
  </div>
  <div class="row">
    <label for="priority" class="col-md-3">Priority</label>
    <input id="priority" class="col-md-5" type="number" [(ngModel)]="todo.priority">
  </div>
  <div class="row">
    <div class="col-md-3"></div>
    <div class="col-md-3">
      <button type="button" class="btn btn-outline-info"
        (click)="onClickNew()">New</button>
    </div>
  </div>
</div>
```

todo-create-simple.component.ts

```
import { Component } from '@angular/core';
import { TodoModel } from '../model/todo';
@Component({
  selector: 'app-todo-create-simple',
  templateUrl: './todo-create-simple.component.html',
})
export class TodoCreateSimpleComponent {
  todo: TodoModel = new TodoModel(0, "", 1);
  onClickNew() {
    console.log('todo to add', this.todo);
  }
}
```

Add router « new-todo »

```
app-routing.module.ts ×
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { TodoListPageComponent } from './todo-list-page/todo-list-page.component';
4 import { TodoCreateSimpleComponent } from './todo-create-simple/todo-create-simple.component';
5
6 const routes: Routes = [
7   { path: 'todos', component: TodoListPageComponent },
8   { path: 'new-todo', component: TodoCreateSimpleComponent }, // New route
9 ];
```

Create TODO

Description

learn rxjs

Priority

2

New

Basic Form ... [(ngModel)]=..

Need FormsModule no validation yet

```
3 import { FormsModule } from '@angular/forms';  
  
12 @NgModule({  
13   declarations: [  
14     AppComponent,  
15     TodoListPageComponent,  
16     TodoCreateSimpleComponent,  
17     TodoUpdateSimpleComponent  
18   ],  
19   imports: [  
20     BrowserModule,  
21     AppRoutingModule,  
22     NgbModule,  
23     FormsModule  
24   ],
```

Create TODO

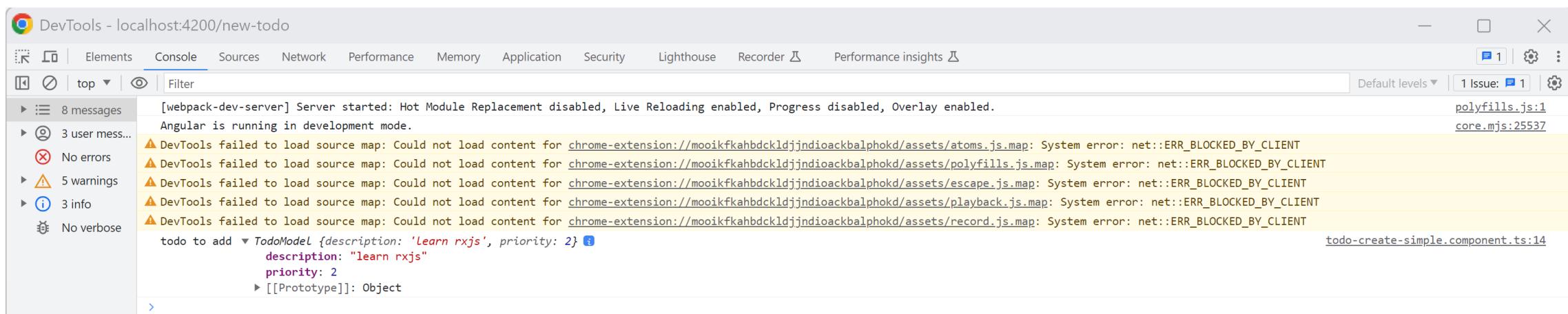
Description

Priority

New



Click + F12 Debugger Console



Demo Outline

Reminder : http client/server, SPA: Single Page Application

Classical Web app tutorial: « Todo web application »

Rest Api Backend Server – NodeJS, Express, Swagger

Angular + ng-bootstrap setup

Component Pages (offline / mock data)

- List Page
- Duplicate/Delete button
- Detail Page
- New Page
- links between pages



Service

HttpClient (call to API backend)

Advanced (ag-grid, ...)

add <A> or <button> « routerLink »

```
16  <div class="container">
17    <div class="row">
18      <label class="col-md-4">New using link or button</label>
19      <div class="col-md-2">
20        <A routerLink="/new-todo">New ...</A>
21      </div>
22      <div class="col-md-2">
23        <button type="button" class="btn btn-outline-info" routerLink="/new-todo">New ...</button>
24      </div>
25    </div>
26  </div>
```

url: /todos

New using link or button

New ...

New ...



To url: /new-todo

Add Button with contextual id « /todo/id »

```
router.navigate(['todo', id]);
```

13 |

```
<button type="button" class="btn btn-outline-info"  
     (click)="onClickOpenDetail(todo)">Detail...</button>
```

14 |

2

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

13

```
constructor(private router: Router) {}
```

35

```
onClickOpenDetail(item: TodoModel) {
```

36

```
  this.router.navigate(['/todo', item.id]);
```

37

```
}
```

Declare url « /todo/:id » with path parameter

```
app-routing.module.ts
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { TodoListPageComponent } from './todo-list-page/todo-list-page.component';
4 import { TodoCreateSimpleComponent } from './todo-create-simple/todo-create-simple.component';
5 import { TodoDetailPageComponent } from './todo-detail-page/todo-detail-page.component';
6
7 const routes: Routes = [
8   { path:'todos', component: TodoListPageComponent },
9   { path:'new-todo', component: TodoCreateSimpleComponent },
10  { path:'todo/:id', component: TodoDetailPageComponent },
11];
```

Demo Outline

Reminder : http client/server, SPA: Single Page Application

Classical Web app tutorial: « Todo web application »

Rest Api Backend Server – NodeJS, Express, Swagger

Angular + ng-bootstrap setup

Component Pages (offline / mock data)

- List Page
- Duplicate/Delete button
- Detail Page
- New Page
- links between pages



Service

HttpClient (call to API backend)

Advanced (ag-grid, ...)

Todo Detail page for « /todo/:id »

```
todo-detail-page.component.html ×
1  <H1>Todo {{id}} </H1>
2
3  <div *ngIf="todo" class="container">
4    <div class="row">
5      <label class="col-md-3">Description</label>
6      <span class="col-md-5">{{todo.description}}</span>
7    </div>
8    <div class="row">
9      <label class="col-md-3">Priority</label>
10     <span class="col-md-5">{{todo.priority}}</span>
11   </div>
12 </div>
13
14 <div *ngIf="!todo">
15   Content not loaded yet for id: {{id}}
16 </div>
```

```
todo-detail-page.component.ts ×
1  import { Component } from '@angular/core';
2  import { ActivatedRoute } from '@angular/router';
3
4  import { TodoModel } from '../model/todo';
5
6  @Component({
7    selector: 'app-todo-detail-page',
8    templateUrl: './todo-detail-page.component.html'
9  })
10 export class TodoDetailPageComponent {
11
12   id: number = 0;
13   todo: TodoModel|undefined = undefined;
14
15   constructor(private activatedRoute: ActivatedRoute) {
16     this.id = +activatedRoute.snapshot.params['id'];
17     // cf next: load from id ... this.todo = data;
18   }
19 }
```

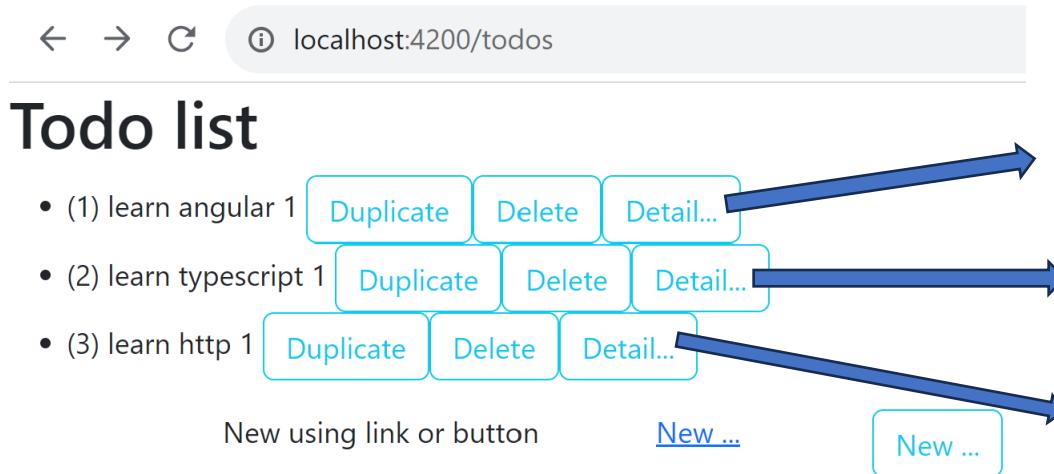
Todo Detail Page

← → ⌂ ⓘ localhost:4200/todos

Todo list

- (1) learn angular 1 [Duplicate](#) [Delete](#) [Detail...](#)
- (2) learn typescript 1 [Duplicate](#) [Delete](#) [Detail...](#)
- (3) learn http 1 [Duplicate](#) [Delete](#) [Detail...](#)

New using link or button [New ...](#) [New ...](#)



← → ⌂ ⓘ localhost:4200/todo/1

← → ⌂ ⓘ localhost:4200/todo/2

← → ⌂ ⓘ localhost:4200/todo/3

← → ⌂ ⓘ localhost:4200/todo/1

Todo (1)

Content not loaded yet for id: 1

Back to Todo List button

18

```
<A routerLink="/todos">Todo List page</A>
```

localhost:4200/todos

Todo list

- (1) learn angular 1 [Duplicate](#) [Delete](#) [Detail...](#)
- (2) learn typescript 1 [Duplicate](#) [Delete](#) [Detail...](#)
- (3) learn http 1 [Duplicate](#) [Delete](#) [Detail...](#)

New using link or button [New ...](#)

localhost:4200/todo/1

Todo (1)

Content not loaded yet for id: 1

[Todo List page](#)

Demo Outline

Reminder : http client/server, SPA: Single Page Application

Classical Web app tutorial: « Todo web application »

Rest Api Backend Server – NodeJS, Express, Swagger

Angular + ng-bootstrap setup

Component Pages (offline / mock data)

- List Page
- Duplicate/Delete button
- Detail Page
- New Page
- links between pages



Service

HttpClient (call to API backend)

Advanced (ag-grid, ...)

Storing model in Service class (lifecycle != Component)

```
| $ ng g s Todo
| CREATE src/app/todo.service.spec.ts (347 bytes)
| CREATE src/app/todo.service.ts (133 bytes)
```



A screenshot of a code editor showing the file `todo.service.ts`. The code defines a service named `TodoService` using the `@Injectable` decorator with `providedIn: 'root'`. The service has an empty constructor.

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class TodoService {
7
8   constructor() { }
9 }
```

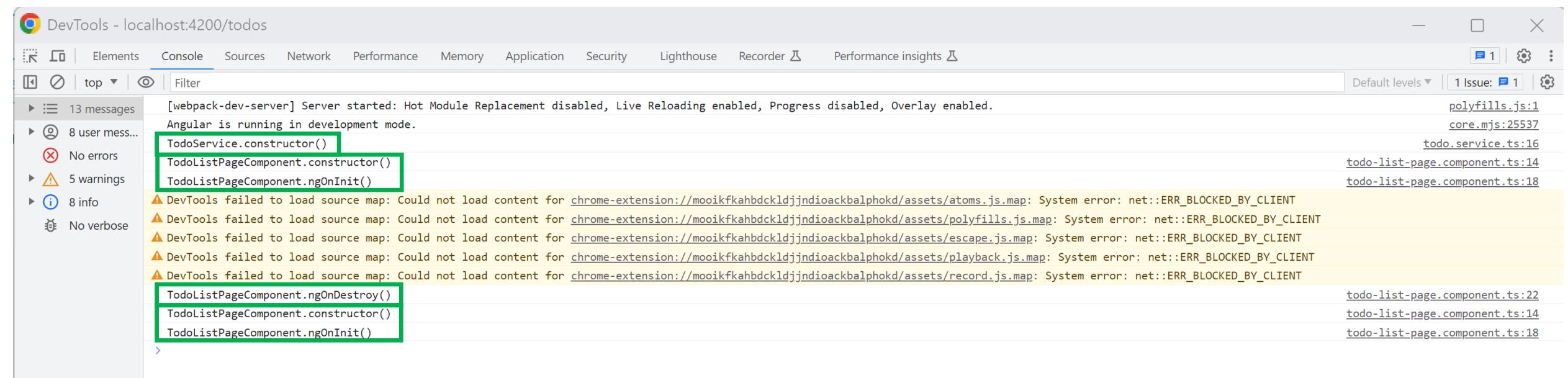
inject Service in Component constructor

```
todo-list-page.component.ts
1 import { Component } from '@angular/core';
2 import { Router } from '@angular/router';
3 import { TodoModel } from '../model/todo';
4 import { TodoService } from '../todo.service';
5
6 @Component({
7   selector: 'app-todo-list-page',
8   templateUrl: './todo-list-page.component.html'
9 })
10 export class TodoListPageComponent {
11
12   constructor(private todoService: TodoService,
13             private router: Router) {
14 }
```

Showing LifeCycle: console.log() in constructor(), ngOnInit(), ngOnDestroy()

```
todo-list-page.component.ts
1 import { Component, OnInit, OnDestroy } from '@angular/core';
2 import { Router } from '@angular/router';
3 import { TodoModel } from '../model/todo';
4 import { TodoService } from '../todo.service';
5
6 @Component({
7   selector: 'app-todo-list-page',
8   templateUrl: './todo-list-page.component.html'
9 })
10 export class TodoListPageComponent implements OnInit, OnDestroy {
11
12   constructor(private todoService: TodoService, private router: Router) {
13     console.log('TodoListPageComponent.constructor()');
14   }
15
16   ngOnInit() {
17     console.log('TodoListPageComponent.ngOnInit()');
18   }
19
20   ngOnDestroy() {
21     console.log('TodoListPageComponent.ngOnDestroy()');
22   }
}
```

Showing Life Cycle



The screenshot shows the Google DevTools Console tab for a browser window titled "DevTools - localhost:4200/todos". The console output displays several messages related to the Angular application's life cycle:

- [webpack-dev-server] Server started: Hot Module Replacement disabled, Live Reloading enabled, Progress disabled, Overlay enabled.
- Angular is running in development mode.
- TodoService.constructor()
- TodoListPageComponent.constructor()
- TodoListPageComponent.ngOnInit()
- DevTools failed to load source map: Could not load content for chrome-extension://mooikfkahbdckldjjndioackbalphokd/assets/atoms.js.map: System error: net::ERR_BLOCKED_BY_CLIENT
- DevTools failed to load source map: Could not load content for chrome-extension://mooikfkahbdckldjjndioackbalphokd/assets/polyfills.js.map: System error: net::ERR_BLOCKED_BY_CLIENT
- DevTools failed to load source map: Could not load content for chrome-extension://mooikfkahbdckldjjndioackbalphokd/assets/escape.js.map: System error: net::ERR_BLOCKED_BY_CLIENT
- DevTools failed to load source map: Could not load content for chrome-extension://mooikfkahbdckldjjndioackbalphokd/assets/playback.js.map: System error: net::ERR_BLOCKED_BY_CLIENT
- DevTools failed to load source map: Could not load content for chrome-extension://mooikfkahbdckldjjndioackbalphokd/assets/record.js.map: System error: net::ERR_BLOCKED_BY_CLIENT
- TodoListPageComponent.ngOnDestroy()
- TodoListPageComponent.constructor()
- TodoListPageComponent.ngOnInit()

The constructor and ngOnInit events for TodoListPageComponent are highlighted with green boxes, indicating they are being tracked or analyzed.

Refactor: move « todos » code in service

```
todo.service.ts ×
1 import { Injectable } from '@angular/core';
2 import { TodoModel } from './model/todo';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class TodoService {
8
9   todos: TodoModel[] = [
10     new TodoModel(1, "learn angular", 1),
11     new TodoModel(2, "learn typescript", 1),
12     new TodoModel(3, "learn http", 1),
13   ];
14
15   constructor() { }
16
17   duplicateTodo(item: TodoModel, idx: number) {
18     let newId = this.todos.length + 1;
19     let dupItem = new TodoModel(newId, item.description + ' (copy)',
20                               item.priority+1);
21     this.todos.push(dupItem);
22   }
23
24   deleteTodo(item: TodoModel, idx: number) {
25     this.todos.splice(idx, 1);
26   }
27
28 }
```

Refactor Component: call service methods

```
24   get todos(): TodoModel[] { return this.todoService.todos; }

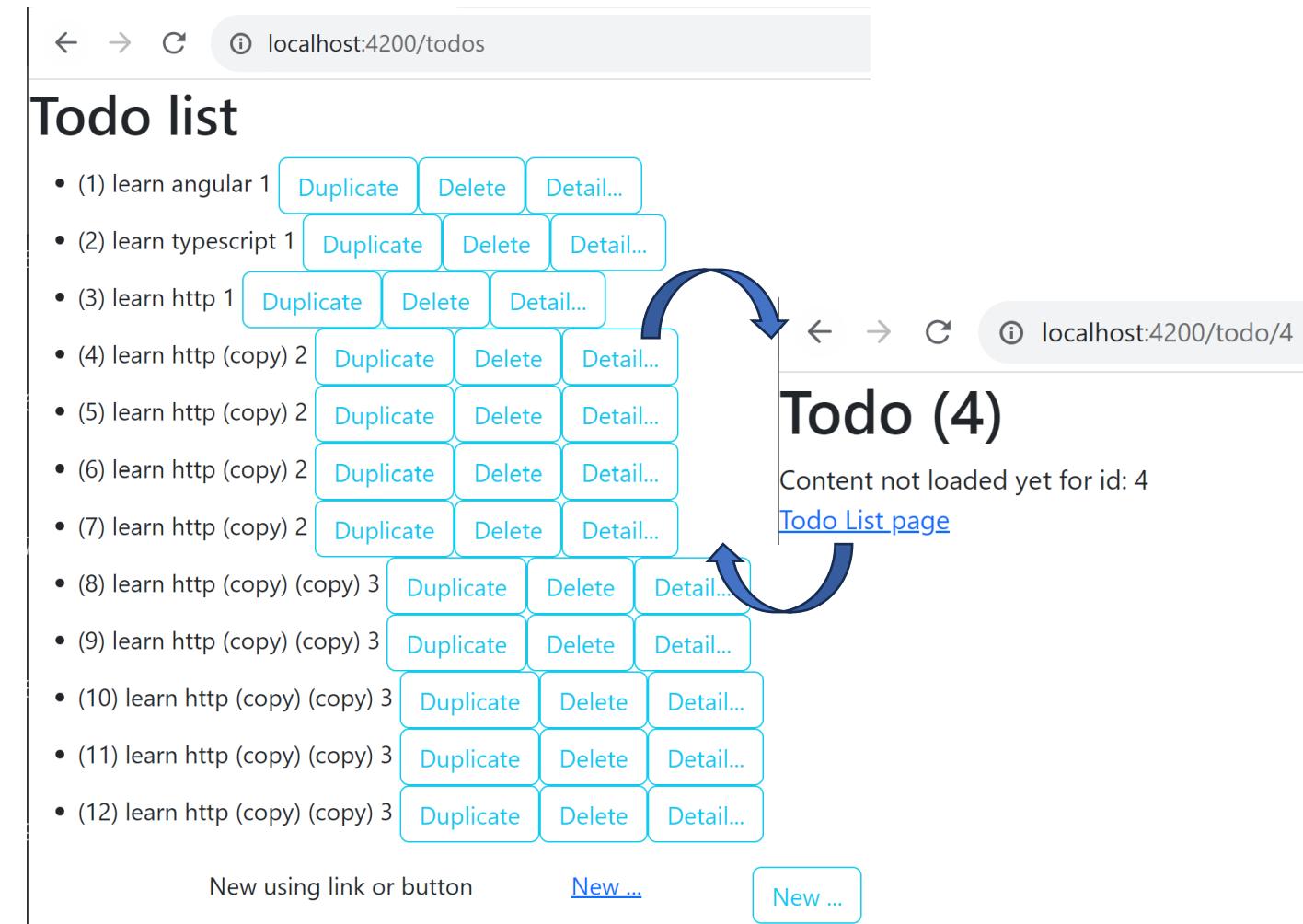
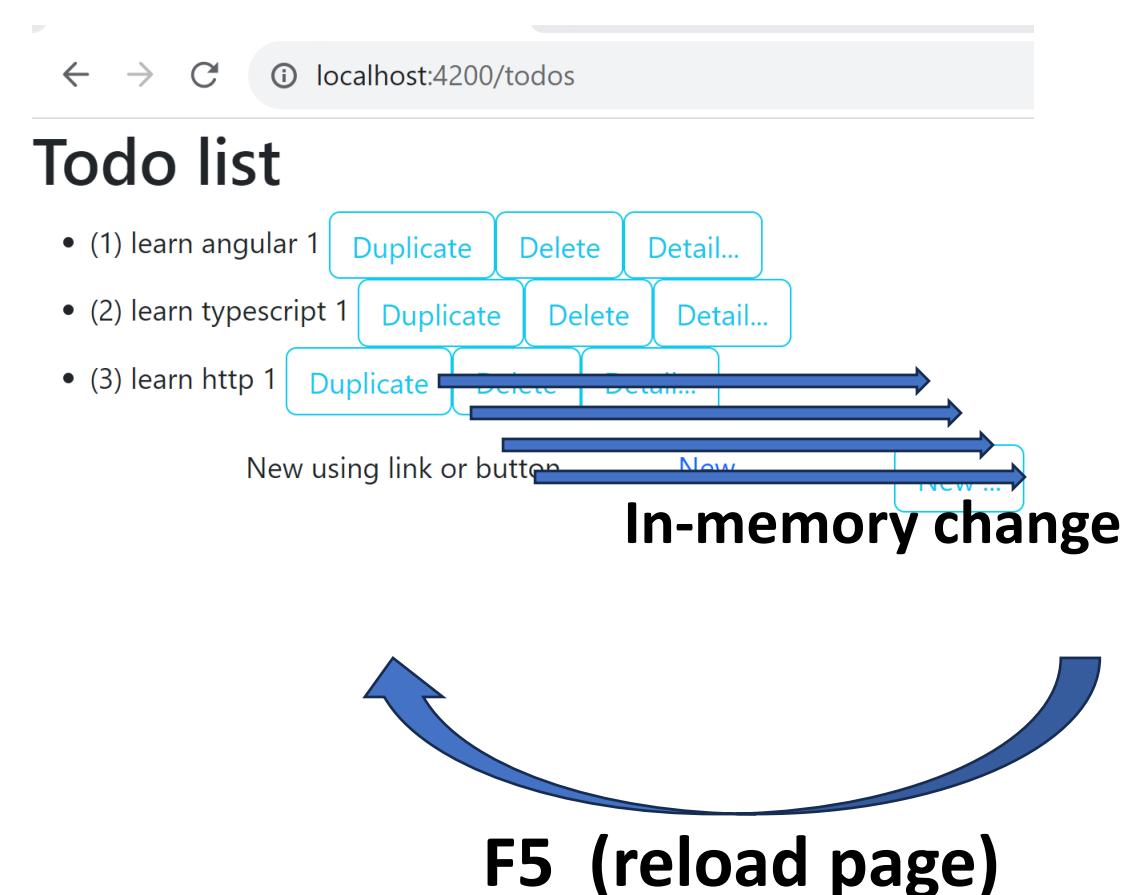
25

26   duplicateTodo(item: TodoModel, idx: number) {
27     this.todoService.duplicateTodo(item, idx);
28   }

29

30   deleteTodo(item: TodoModel, idx: number) {
31     this.todoService.deleteTodo(item, idx);
32   }
```

Navigate pages => Todos in-memory stored
Refresh page => Todos reset



Using Service for detailed page « /todo/:id » « todoService.findById(id) »

```
todo-detail-page.component.ts ×
1 import { Component } from '@angular/core';
2 import { ActivatedRoute } from '@angular/router';
3
4 import { TodoModel } from '../model/todo';
5 import { TodoService } from '../todo.service';

6
7 @Component({
8   selector: 'app-todo-detail-page',
9   templateUrl: './todo-detail-page.component.html'
10 })
11 export class TodoDetailPageComponent {
12
13   id: number = 0;
14   todo: TodoModel|undefined = undefined;
15
16   constructor(private todoService: TodoService,
17             private activatedRoute: ActivatedRoute) {
18     this.id = +activatedRoute.snapshot.params['id'];
19     this.todo = this.todoService.findById(this.id);
20   }
21 }
```

```
todo.service.ts ×
19
20   findById(id: number): TodoModel|undefined {
21     return this.todos.find(x => x.id === id);
}
```

← → ⏪ ⓘ localhost:4200/todo/2

Todo (2)

Description	Priority
learn typescript	1

[Todo List page](#)

← → ⏪ ⓘ localhost:4200/todo/9999999

Todo (9999999)

Content not loaded yet for id: 9999999

[Todo List page](#)

Using Service for « new » Form page

```
todo-create-simple.component.ts
1 import { Component } from '@angular/core';
2 import { Router } from '@angular/router';
3
4 import { TodoModel } from './model/todo';
5 import { TodoService } from './todo.service';
6
7 @Component({
8   selector: 'app-todo-create-simple',
9   templateUrl: './todo-create-simple.component.html',
10 })
11 export class TodoCreateSimpleComponent {
12
13   todo: TodoModel = new TodoModel(0, "", 1);
14
15   constructor(private todoService: TodoService,
16             private router: Router) {}
17
18   onClickNew() {
19     let newObj = this.todoService.saveNew(this.todo);
20     this.router.navigate([ 'todo', newObj.id ]);
21   }
22
23 }
```

```
todo.service.ts
23
24
25
26
27
28
29
30
31
32
33
34
35
saveNew(src: TodoModel): TodoModel {
  let newId = this.generateNewId();
  let obj = new TodoModel(newId, src.description, src.priority)
  this.todos.push(obj);
  return obj;
}

generateNewId(): number {
  return 1 + this.todos.reduce(
    (acc, val) => [ Math.max(acc[0], val.id) ],
    [0]
  )[0];
}
```

Demo Outline

Reminder : http client/server, SPA: Single Page Application

Classical Web app tutorial: « Todo web application »

Rest Api Backend Server – NodeJS, Express, Swagger

Angular + ng-bootstrap setup

Component Pages (offline / mock data)

- List Page
- Duplicate/Delete button
- Detail Page
- New Page
- links between pages

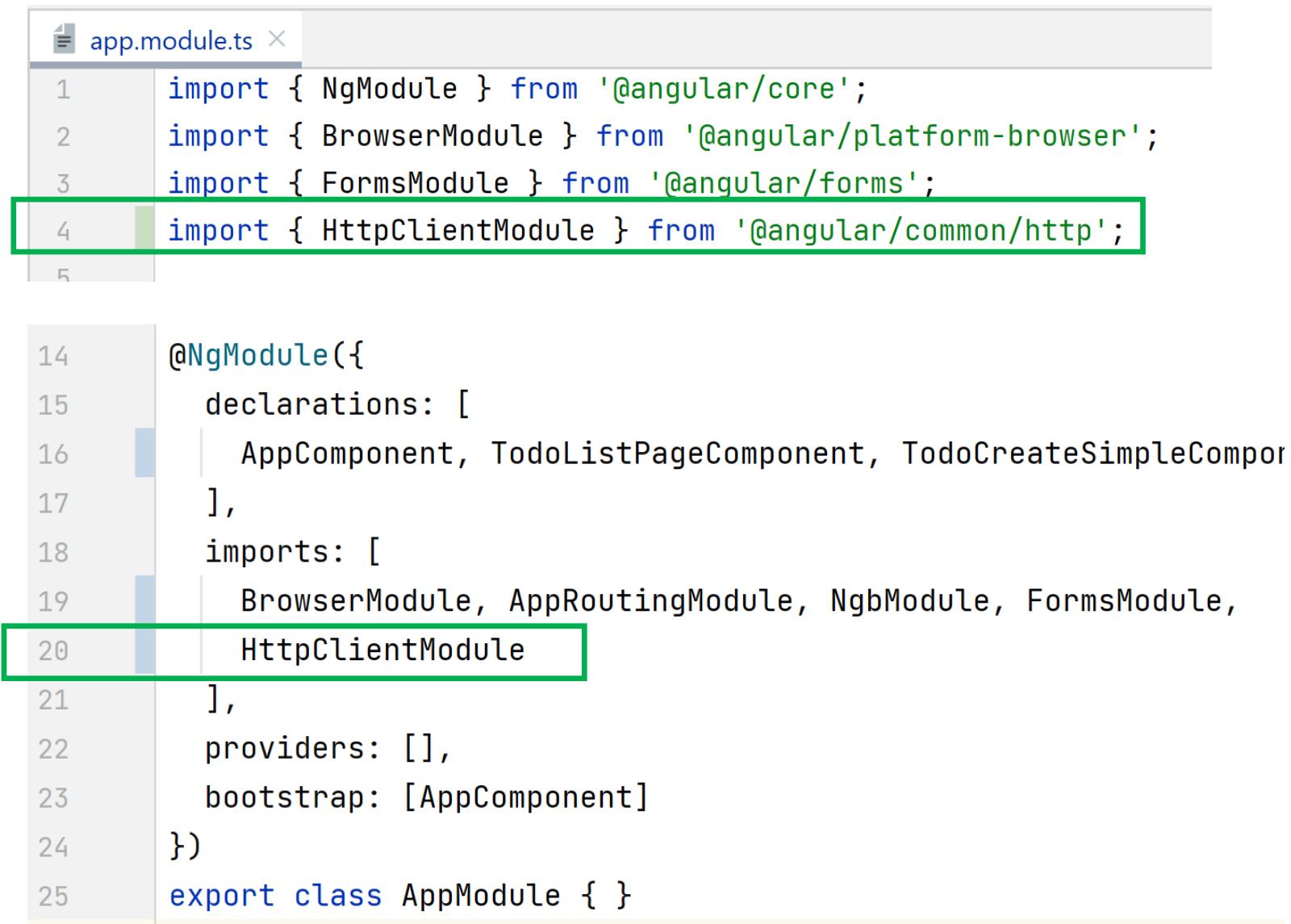


Service

HttpClient (call to API backend)

Advanced (ag-grid, ...)

Using HttpClient Module



The image shows a code editor window with the file 'app.module.ts' open. The code defines an Angular module with imports for core, browser, forms, and http modules.

```
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { FormsModule } from '@angular/forms';
4 import { HttpClientModule } from '@angular/common/http';
5
14 @NgModule({
15   declarations: [
16     AppComponent, TodoListPageComponent, TodoCreateSimpleCompor
17   ],
18   imports: [
19     BrowserModule, AppRoutingModule, NgbModule, FormsModule,
20     HttpClientModule
21   ],
22   providers: [],
23   bootstrap: [AppComponent]
24 })
25 export class AppModule { }
```

The line 'import { HttpClientModule } from '@angular/common/http';' is highlighted with a green box. The line 'HttpClientModule' in the @NgModule imports section is also highlighted with a green box.

Calling http from Service

```
todo.service.ts ×

1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';

17 constructor(private httpClient: HttpClient) {
18   console.log('TodoService.constructor()');
19
20   console.log('Calling http GET /api/todo ... subscribe to async response');
21   this.httpClient.get('/api/todo').subscribe(resp => {
22     console.log('Finished http GET /api/todo', resp);
23   }, err => {
24     console.log('Failed http GET /api/todo', err);
25   });
26 }
```

Configuring angular proxy.json

The screenshot shows a web browser displaying the Angular documentation at angular.io/guide/build. The page title is "Proxying to a backend server". The left sidebar contains a navigation menu with items like "Intercept requests and responses", "Interceptor use-cases", "Pass metadata to interceptors", etc. The main content area has a heading "Proxying to a backend server" and a paragraph explaining how to use webpack's proxying support to divert URLs to a backend server by creating a proxy configuration file. It provides steps to create the file and add content. A code block shows the proxy configuration JSON:

```
{  
  "/api": {  
    "target": "http://localhost:3000",  
    "secure": false  
  }  
}
```

The right sidebar lists other topics under "Building and serving Angular apps", including "Configuring application environments", "Configure environment-specific defaults", "Using environment-specific variables in your app", "Configure target-specific file replacements", "Configuring size budgets", "Configuring CommonJS dependencies", and "Configuring browser compatibility". At the bottom right, there is a link to "Proxying to a backend server".

proxy.json



A screenshot of a code editor window titled "proxy.conf.json". The code is a JSON configuration for a proxy. It defines a single entry point for the "/api" path, which points to "http://localhost:3000". The "pathRewrite" field is set to an empty string, indicating that requests to "/api" should be proxied directly to the target without changing the URL.

```
1  {
2    "/api": {
3      "target": "http://localhost:3000",
4      "secure": false,
5      "pathRewrite": {
6        "^/api": ""
7      }
8    }
9  }
```

Test http GET <http://localhost:4200/api/todo> => http GET <http://localhost:3000/todo>

The screenshot shows the Network tab in Google DevTools for the URL `localhost:4200/todos`. The timeline at the top shows several requests and responses. A specific request to `/api/todo` is selected, and its response is displayed in the main pane. The response is a JSON array of three todo items:

```
[{"id": 1, "description": "learn Angular", "priority": 1}, {"id": 2, "description": "learn Typescript", "priority": 1}, {"id": 3, "description": "learn http", "priority": 1}]
```

Below the Network tab, the Console tab shows the following log output:

```
TodoService.constructor()
Calling http GET /api/todo ... subscribe to async response
TodoListPageComponent.constructor()
TodoListPageComponent.ngOnInit()
Finished http GET /api/todo [3] [{}]
  ▶ 0: {id: 1, description: 'learn Angular', priority: 1}
  ▶ 1: {id: 2, description: 'learn Typescript', priority: 1}
  ▶ 2: {id: 3, description: 'learn http', priority: 1}
  length: 3
  [[Prototype]]: Array(0)
```

Take Away

Questions

Next Steps ...