**25. Bonus: Angular Universal**

388. Module Introduction

**Angular Universal Intro:**

იგი საშუალებას გვაძლევს წინასწარ დარენდერდეს ის გვერდები, რომლებზეც მომხმარებელი შედის (ანუ ინიციალ რენდერი აღარაა საჭირო).

მაგალითად, თუ მომხმარებელს ნელი ინტერნეტი აქვს, სანამ**js** ფაილი არ ჩამოიტვირთება, ისინი ვერაფერს ვერ დაინახავენ ეკრანზე, ან მეორე მაგალითია, **Search Engine**, რომელიც ინდექსირებას აკეთებს საიტების, იგი არ ელოდება ბოლომდე **js**-ის ჩამოტვირთვას და ანგულარის შემთხვევაში, მხოლოდ მცირე კონტენტის დანახვას ასწრებს, ამიტომაც აღნიშნული ცუდად აისახება ჩვენი საიტის ძებნადობაზე.

ანგულარ უნივერსალმა, ყველა js-ის ჩამოტვირთვამდე უნდა მოგვაწოდოს **html**ფაილი, რომელიც ასევე შეიცავს ყველა**js** იმპორტს, თუმცა ყველაზე ადრე ჩაიტვირთება და ვიზუალს შექმნის.

**25. Bonus: Angular Universal**

390. Adding Angular Universal

**Angular Universal Adding 1.1:**

საწყისში **CLI**-ში უნდა გავუშვათ შემდეგი ბრძანება:

ng add @nguniversal/express-engine --client-project shopping-list *(ეს უკანასკნელი დამოკიდებულია****angular.json****-ში****projects****-ის ქვემოთ რაც წერია, მაგ ფროფერთიზე).*

აღნიშნული შექმნის ახალ მოდულს (მაგ. **app.server.module.t**s) სადაც დიდი ალბათობით არ იქნება დაიმპორტებული **ModuleMapLoaderModule**, რის გამოსასწორებლადაც **CLI**-ში უნდა გავუშვათ შემდეგი ბრძანება:

npm install --save @nguniversal/module-map-ngfactory-loader

ხოლო მოდულში დავაიმპორტოთ **ModuleMapLoaderModule**, **@nguniversal/module-map-ngfactory-loader**-დან.

აღნიშნული დამატება ლეიზილოადინგისთვისაა, რათა იმუშავოს.

*ამის დაყენების შემდეგ უნდა ვიცოდეთ, რომ ჯერ ერთი გვერდი დარენდერდება სერვერზე და მას შემდეგ დაიწყებს მთელი აპლიკაცია დარენდერებას.*

**25. Bonus: Angular Universal**

390. Adding Angular Universal

**Angular Universal Adding 1.2:**

ასევე ერთი რამ შეიცვლება, რამდენადაც ჯერ სერვერზე რენდერდება კოდი და არა ბრაუზერში **localStorage**აღარ იმუშავებს.

ამის გამოსასწორებლად, **app.component**-ში უნდა გადავიდეთ და უნდა დავრწმუნდეთ, რომ აღნიშნული ფაილი, სადაც **localStorage**მუშავდება,  საწყისში არ დარენდერდება.

ამისთვის **app.componenti** უნდა დვაინჯექტოთ კონსტრუქტორში შემდეგნაირად (**@angular/core**-დან შემოდის): @Inject(PLATFORM\_ID) private platformId.

შემდეგ კი **ngOnInit**-ში ქონდიშენს ვამოწმებთ რომელ პლატფორმაზე იკითხება კოდი (უნდა დაიმპორტდეს ჯერ **isPlatformBrowser @angular/common**-იდან) და აღნიშნული მეთოდი ჩაჯდეს ქონდიშენში და მას არგუმენტად ჩაეწოდოს platformId და შიგნით ჩაჯდეს უკვე ის კომპონენტი, სადაც **localStorage**იკითხება (**dispatch()** ჩვენს შემთხვევაში).

**25. Bonus: Angular Universal**

390. Adding Angular Universal

**Angular Universal Adding 1.3:**

ამის შემდეგ კი გადავდივართ **package.json**-ში, სადაც იქნება**build:ssr** ბრძანება ნახსენები, რომელიც შეგვიძლია გავუშვათ: იგი **ng build --prod**-ში უშვებს:

npm run build:ssr

აღნიშნული შემდეგ ჩვენი აპლიკაცია სტატიკური აღარაა, ამიტომაც საჭიროებს ისეთ სერვერს, რომელსაც **nodeJS**-ის წაკითხვა შეუძლია.

შემდეგ კი სერვერზე გასაშვებად გამოიყენება **CLI**ბრძანება:

npm run serve:ssr

ბოლო ბრძანება ჩვენს აპლიკაციას გაუშვებს **localhost:4000**-ზე.

**25. Bonus: Angular Universal**

390. Adding Angular Universal

**Angular Universal Adding Codes 1.1:**

**app.server.module.ts:**

1. import { NgModule } from '@angular/core';
2. import { ServerModule } from '@angular/platform-server';
4. import { AppModule } from './app.module';
5. import { AppComponent } from './app.component';
6. import { ModuleMapLoaderModule } from '@nguniversal/module-map-ngfactory-loader';
8. @NgModule({
9. imports: [
10. AppModule,
11. ServerModule,
12. ModuleMapLoaderModule
13. ],
14. bootstrap: [AppComponent],
15. })
16. export class AppServerModule {}

**25. Bonus: Angular Universal**

390. Adding Angular Universal

**Angular Universal Adding Codes 1.2:**

**app.component.ts განახლებები:**

1. import { Component, Inject, OnInit, PLATFORM\_ID } from '@angular/core';
2. import { isPlatformBrowser } from '@angular/common';
4. export class AppComponent implements OnInit {
5. constructor(
6. @Inject(PLATFORM\_ID) private platformId
7. ) {}
9. ngOnInit() {
10. if(isPlatformBrowser(this.platformId)){
11. this.store.dispatch(new AuthActions.AutoLogin());
12. }
13. this.loggingService.printLog('Hello from AppComponent ngOnInit');
14. }
15. }

**მნიშვნელოვანი:!!!**

**25. Bonus: Angular Universal**

390. Adding Angular Universal

**Angular Universal 9+:**

ახალ ვერსიებში აღარაა საჭირო **module-map-ngfactory-loader**და ერორის გამოსასწორებლად:

npm uninstall @nguniversal/common

npm uninstall @nguniversal/module-map-ngfactory-loader

ასევე ვშლით **app.server.module.ts**-დან **ModuleMapLoaderModule**-ს და შემდეგ ვუშვებთ შემდეგ ბრძანებებს:

ng add @nguniversal/express-engine

npm run dev:ssr

**25. Bonus: Angular Universal**

391. Adding Angular Universal with NestJS

**Angular Universal Nest.js Setting up:**

უნივერსალ რენდერინგის დამატების მეორე გზაცაა: **Nest.js**-ის გამყენება, რომელიც სერვერსაიდ ფრეიმვორკია (**Server-side framework**). მის გასაშვებად **CLI**-ში:

ng add @nestjs/ng-universal

შემდეგ სახელში იგივენაირად **angular.json**-ის **projects**-ის ქვემოთ არსებული სახელი უნდა შევიყვანოთ.

იგი, სტანდარტული ანგულარ უნივერსალისგან განსხვავებით, ასევე შესაძლებლობას გაძლევს შენი საკუთარი **API**გამართო ბაქში.

**app.server.module.ts**-სა და **app.component.ts**-ში იგივე მოქმედებები ხდება, რაც აქამდე.

და იგივენაირად npm run build:ssr და npm run serve:ssr

**25. Bonus: Angular Universal**

400. Next Steps

**Angular Universal Gotchas:**

You typically use Angular Universal to pre-render Angular pages on the server. Once the app then runs on the client, it's a normal SPA again.

Server-side rendering (SSR) can make sense because of SEO considerations (crawler should see what your users see) or because you want to deliver a finished page to your users (rather than creating the page in the browser).

But that also has one important implication: **You MUST NOT use any browser-only APIs like document.querySelector() in your Angular code!**

Simply because it will execute on the server and there, such APIs are not available.

That's why, in this course, I recommended to **use Angular features only**: These features are safe to use because Angular will check if a certain API can be used before it uses it.