**27. Adding Offline Capabilities with Service Workers**

415. Adding Service Workers

**Angular Offline Service Workers 1.1:**

ჯავასკრიპტი სტანდარტულად ერთ მწკრივზე ეშვება.

ჯავასკრიპტი და ბრაუზერი ასევე გვთავაზობს, რომ სხვა ნაკადი გამოვიყენოთ მთავარი **js**კოდისგან, იგივე **Service Worker**.

იგი მთავარი **js**-დან გამოთიშულია და ცალკე იკითხება.

მას შეუძლია მაგალითად **http**რექვესტები დაიჭიროს და თუ ინტერნეტი არ გვაქვს, დაქეშოს ისინი და ინტერნეტის მოსვლისას აამუშავოს.

ანგულარი გვთავაზობს მსგავს **service worker**-ს, ამისთვის სპეციალური პაკეტი უნდა დავამატოთ **CLI**-დან:

ng add @angular/pwa

იგი ჩაამატებს რამდენიმე დანამატს, ასევე ფაილს**manifest.json**, სადაც გარკვეული კონფიგურირებაა შესაძლებელი.

იგი ასევე **appModule**-ში დაამატებს ახალ იმპორტს, რომელიც მოიცავს**ngsw-worker.js** ფაილს, რომელიც არ არსებობს ჯერ, თუმცა ამომავლოდ ავტომატურად დაგენერირდება*(ბილდისას, dist ფოლდერში)*.

შემდეგ შეგვიძლია დავბილდოთ და **dist**-ფოლდერშ ვნახოთ ახლად შექმნილი **js**ფაილი (ng build --prod).

**27. Adding Offline Capabilities with Service Workers**

415. Adding Service Workers

**Angular Offline Service Workers 1.2:**

დაბილდული აპლიკაციის გასატესტად დაგვჭრდება სერვერი. ანგულარი თავადვე გვაწვდის აღნიშნულ ფიჩურს, უბრალოდ მისი გაშვება მოხდება კონკრეტულად პროექტის ფოლდერში, თუმცა ჯერ მანამდე უნდა დავაინსტალიროთ: npm install -g http-server

შემდეგ გადავდივართ კონკრეტულ ფოლდერში: http-server -p 8081

შემდეგ შეგვიძლია რომელიმე მოცემულ **IP-**ზე გადავიდეთ და ვნახოთ პროექტუ. თუ **inspect/application/offline**-ს გავააქტიურებთ, ინტერნეტის არყოფნას არ გვიჩვენებს, უბრალოდ ბლანკ გვერდი იქნება.

ამის გამოსასწორებლად, უბრალოდ **app.component.html**-ფაილში უნდა გვეწეროს რაიმე კოდი, მაგალითად **h1**-ში რამე. შემდეგ თავიდან დავბილდავთ და შედეგს ვნახავთ.

ხოლო მთლიანი ფაილის დასაქეშად, უნდა შევიდეთ **ngsw-config.json**-ში და...

**27. Adding Offline Capabilities with Service Workers**

416. Caching Assets for Offline Use

**Angular Offline Service Workers 1.3:**

**ngsw-config.json** ფაილში არის გაწერილი რომელი სტატიკური ინფორმაცია უნდა დაქეშდეს (**assetGroups**-ში).

**"installMode"="prefetch"** - იგი მაინც წამოიღებს მის ქვემოთ მოცემულ ასეტებს, რომც არ გჭირდებოდეს, და დაქეშავს, მისი ალტერნატივაა **lazy**, რომელიც მხოლოდ მაშინ დაქეშავდა, თუ მომხამრებელი გადავიდოდა იქ.

**"updateMode": "prefetch"**- იგი უკვე დააფდეითებულ დასაქეშ ასეტებზე აგებს პასუხს.

მაგალითად, თუ ფონტს ვიყენებთ, რამდენადაც აღნიშნული რექვესტს წარმოადგენს, უნდა ვუთხრათ ანგულარს რომ ეგ რექვესტი დაქეშოს. ამისთვის აღნიშნულ ლინკს წამოვიღებთ და

თავად **resources**-ის ობიექტში ჩავამატებთ ახალ ფროფერთის, რომელიც ასევე მასივი იქნება : **"urls": []**:

1. "resources": {
2. "urls":[
3. "https://fonts.googleapis.com/css2?family=Rampart+One&display=swap"
4. ]
5. }

**27. Adding Offline Capabilities with Service Workers**

417. Caching Dynamic Assets & URLs

**Angular Offline დინამიურად სხვა ფაილების და რექვესტების ატვირთვა 1.1:**

მის გასააქტიურებლად, **"assetGroups"**-ის ქვევით უნდა დავამატოთ **"dataGroups"**-ის მასივი (**ngsw-config.json**-ში), სადაც ობიექტების სახით მიდის მონაცემთა ჯგუფები. მაგალითად, ჩვენს შემთხვევაში გვსურს **http**რექვესტის ლინკის შენახვა, ამისთვის ჩავამატებთ ობიექტს და შიგნით იქნება **name**და **urls**ფროფერთიები, ასევე შეგვეშლო **versions**ფროფერთის გამოყენებაც, რომელიც დააკვირდებოდა სხვადასხვა **API** ლინკის ვერსიებს, თუმცა ახლა არ გვჭირდება.

რაც გვჭირდება არიან:

**27. Adding Offline Capabilities with Service Workers**

417. Caching Dynamic Assets & URLs

**Angular Offline დინამიურად სხვა ფაილების და რექვესტების ატვირთვა 1.2:**

შეგვიძლია ჩავამატოთ **"cacheConfig"**, რომელიც დასაქეშ მოცულობას განსაზღვრავს (ანუ რამდენი რესპონსი), და ასევე ქეშის მაქსიმალური ხანგრძლივობა, ანუ რამდენი ხნის მერე უნდა განახლდეს, **"maxAge": "6h ან 50m და ა.შ"**, და ასევე შეგვიძლია ჩავამატოთ**"timeout":"10s"**, რომელშიც განისაზღვრება რესპონსზე ლოდინის დრო. ასევე შეგვიძლია დავამატოთ **"strategy": "freshness ან performance"**, იგი მიიღებს **freshness**, რომელიც ყოველთვის მიმართავს ბაქს პირველად და ყოველთვის გამოიყენებს ქაშს თუ ოფლაინაა, ან **performance**- რომელიც პირდაპირ ეკრანზე გამოიტანს პასუხს და არ დაელოდება **timeout**-ს. ჩვენ **freshness**გამოვიყენებთ. *კოდი შემდეგში...*

**27. Adding Offline Capabilities with Service Workers**

417. Caching Dynamic Assets & URLs

**Angular Offline დინამიურად სხვა ფაილების და რექვესტების ატვირთვა 1.3:**

**ngsw-config.json** ფაილში დამატება:

1. "dataGroups": [
2. {
3. "name": "posts",
4. "urls":[
5. "https://jsonplaceholder.typicode.com/posts"
6. ],
7. "cacheConfig": {
8. "maxSize": 5,
9. "maxAge": "6h"
10. }
11. }
12. ]

* **Official Angular Service Worker Docs:** https://angular.io/guide/service-worker-intro
* **Academind Resources on PWAs:** https://academind.com/learn/progressive-web-apps