



# Angular 18

**ANGULAR 18:  
YENİ ÖZELLİKLER VE  
GELİŞTİRMELER**

Geleceğe Adım Atın:  
Daha Hızlı, Daha Esnek, Daha  
Güçlü!

*Egemen Agostos*

# NG CONTENT : DEFAULT CONTENT

## APP COMPONENT

```
... Angular  
  
<app-second>  
  <header>new header</header>  
  
  new content  
  
  <footer>new footer</footer>  
</app-second>
```

## SECOND COMPONENT

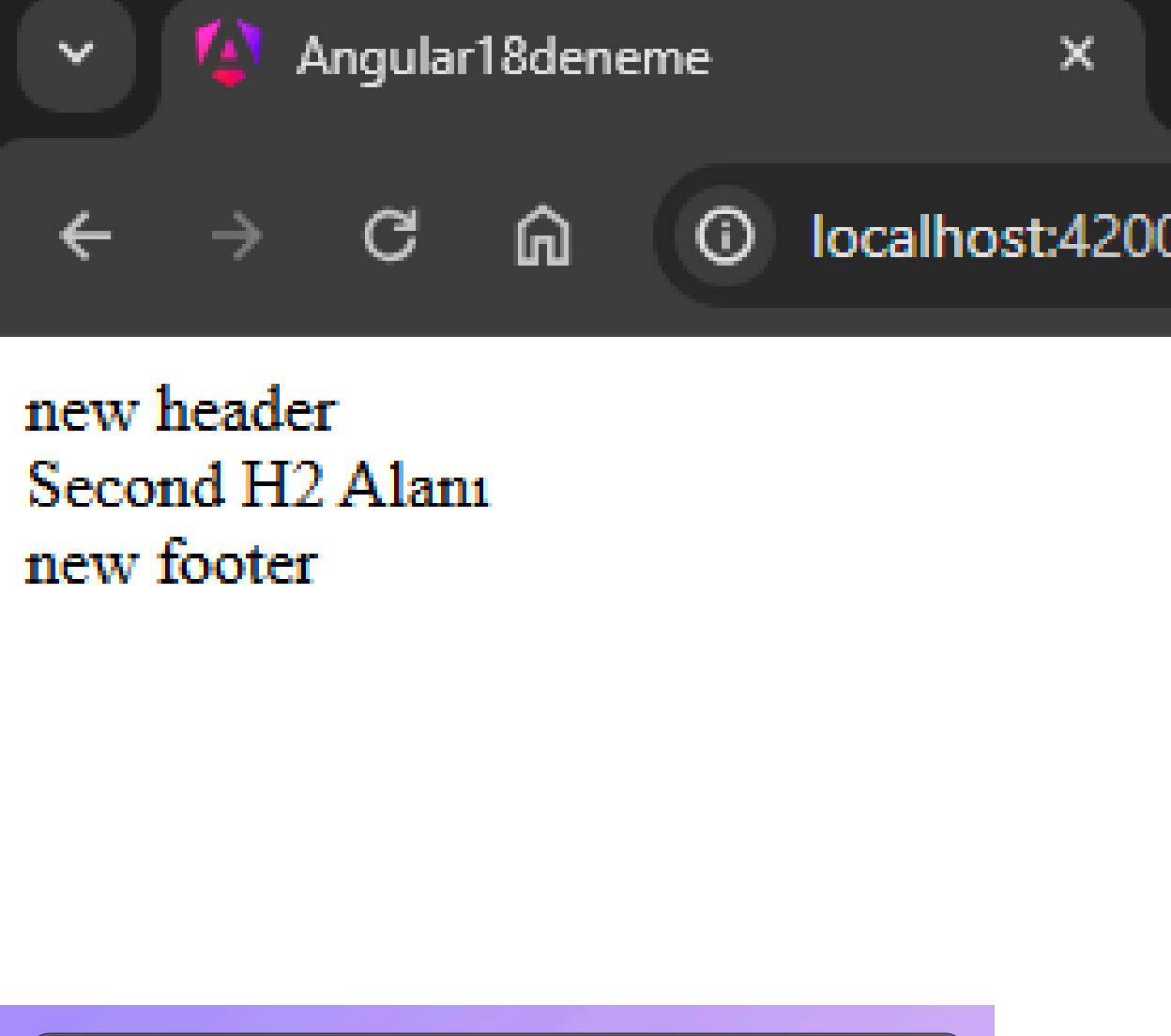
```
... Angular  
  
<ng-content select="header">  
  <div>Second Header Alanı</div>  
</ng-content>  
  
<ng-content select="h2">  
  <div>Second H2 Alanı</div>  
</ng-content>  
  
<ng-content select="footer">  
  <div>Second Footer Alanı</div>  
</ng-content>
```

Angular 18 ile ng-template API'sine yeni özellikler ekleyerek daha güçlü ve yeniden kullanılabilir şablonlar oluşturmayı kolaylaştırıyor.

Varsayılm ki 2 tane componentimiz var. Biri default olarak gelen app.component diğer ise second component olsun. App component içinde second componenti çağrırdığımızı düşünelim.

Second component de görüldüğü üzere select ile alanlarımızı seçebiliriyoruz.

Yani şöyle ki select=h2 ile app component içinde second component'i çağrırdığımız alan için de h2 etikteki ile ilgili bir alan olmadığı için bizim second component de yazdığımız default Second H2 alanı yazısı devreye girecektir. Ama header ve footer olduğu için onlar devreye giremeyecektir.



Kafanızda tam canlamamış olabilir. O yüzden tarayıcıda bir görüntüleyelim. Görüğünüz üzere ben eğer ki app second componentimi çağrırdığım yerde h2 ile bir alan oluşturmadığım için default olarak Second H2 Alanı geldi. Şimdi h2 tanımlayarak deneyelim.

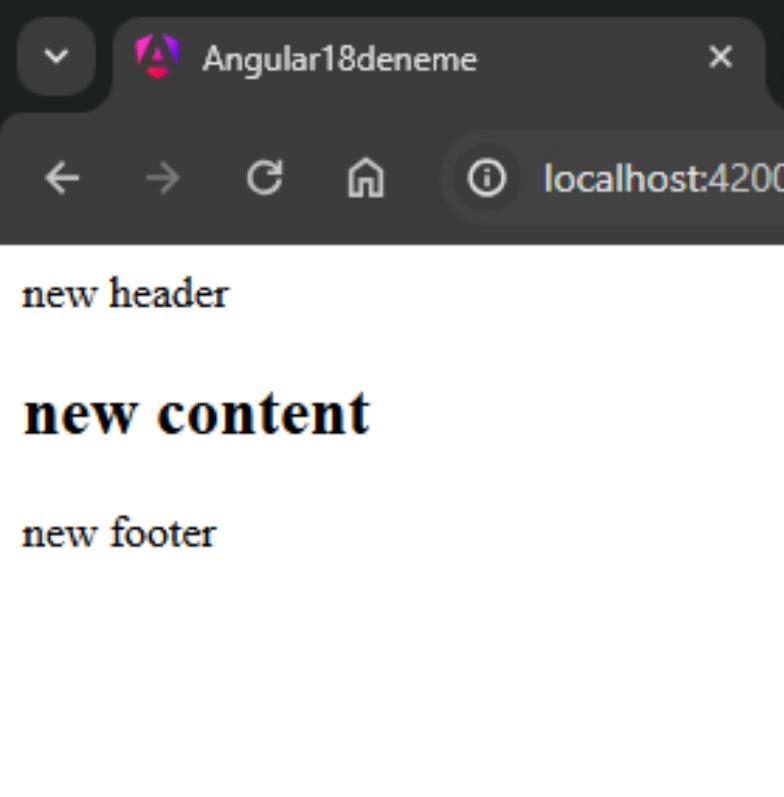
```
Angular

<app-second>
  <header>new header</header>

  <h2>new content </h2>

  <footer>new footer</footer>
</app-second>
```

Componentimi çağrırdım alan içinde gelip h2 alanımızı oluşturup tekrardan tarayıcıya gidelim.



Tarayıcıya geldiğimiz zaman new content yazısını görmüş olduk. Bu sayede default content'i ezmiş olduk. Bu gelen özelliğimiz bu şekildeydi.

# IMPROVED PERFORMANCE WITH IVY

Sırada ki geliştirme ise Angular 18, Ivy derleyicisi için optimizasyonlar sunarak daha hızlı başlangıç süreleri ve daha küçük paket boyutları sağlıyor.



Angular

```
// angular.json
{
  "projects": {
    "my-app": {
      "architect": {
        "build": {
          "options": {
            "optimization": true,
            "aot": true,
            "buildOptimizer": true,
            "sourceMap": false
          }
        }
      }
    }
  }
}
```

Burada ki yapılandırma dosyamız, uygulamanın üretim yapısında optimize edilmesini ve performans iyileştirmelerinin etkinleştirilmesini sağlar.

# IMPROVED FORMS API

```
Angular

import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, FormsModule, PristineChangeEvent, ReactiveFormsModule,
StatusChangeEvent, TouchedChangeEvent, ValueChangeEvent } from '@angular/forms';

@Component({
  selector: 'app-form-example',
  standalone: true,
  imports: [FormsModule, ReactiveFormsModule],
  templateUrl: './form-example.component.html',
  styleUrls: ['./form-example.component.css'
})
export class FormExampleComponent implements OnInit {

  myForm = new FormGroup({
    title : new FormControl('My App'),
    version : new FormControl('1.1')
  });

  ngOnInit(): void {
    this.myForm.events.subscribe((event) => {

      //İlgili inputa dokunulduğunda tetiklenir.
      if(event instanceof TouchedChangeEvent){
        console.log('Touched' + event.touched)
      }

      //Başlangıç durumu değiştiğinde tetiklenir.
      if(event instanceof PristineChangeEvent){
        console.log('Pristine' + event.pristine)
      }

      //Değer her değiştiğinde tetiklenir.
      if(event instanceof ValueChangeEvent){
        console.log('valueChange' + event.value);
        console.log(event.value.version);
      }

      //Değer her değiştiğinde tetiklenir.
      if(event instanceof StatusChangeEvent){
        console.log('Status' + event.status)
      }
    });
  }
}
```

Angular 18 ile forms API'sine birçok yenilik eklendi. Yeni olay sınıfları (ControlEvent, PristineChangeEvent, StatusChangeEvent, TouchedChangeEvent, ValueChangeEvent) ile form durumlarını ve değer değişikliklerini daha iyi izleyebiliyoruz.

Burada ki örneğimiz de FormControl'deki değer değişiklikleri ValueChangeEvent ile izleniyor ve bir değişiklik olduğunda bileşen durumu güncelleniyor.

```
<form [formGroup]="myForm">

  <label for="title">
    Title:
  </label>
  <input id="title" type="text" formControlName="title">

  <label for="version">
    Version:
  </label>
  <input id="version" type="text" formControlName="version">
</form>
```

Form sayfamızın html tarafında ilgili inputları oluşturup. Bunları form içine alalım ve formumuza [formGroup]= " myform" olduğunu bildirelim.

The screenshot shows a browser window with the URL `localhost:4200/form-example`. The developer tools' Console tab is active, displaying the following logs:

- `Angular is running in development mode.` (`core.mjs:30820`)
- `Angular hydrated 2 component(s) and 11 node(s), 0 component(s) were skipped. Learn more at https://angular.io/guide/hydration.` (`core.mjs:30820`)

To the right of the console, there is a form with two inputs:

- `Title:`
- `Version:`

Burada sayfa ilk açıldığında görüldüğü üzere bir tetikleme olmamakta şimdi gelip title input alanına 1 kere tıklayalım ve console tekrardan göz atalım.

The screenshot shows the same browser setup as before, but now the title input has been clicked. The developer tools' Console tab shows the following additional log entry:

- `Touchedtrue` (`form-example.component.ts:18`)

The form inputs remain the same as in the previous screenshot.

The screenshot shows the Chrome DevTools Console tab for the URL `localhost:4200/form-example`. The console output is as follows:

```
Angular is running in development mode. core.mjs:30820
Angular hydrated 2 component(s) and 11 node(s), 0 component(s) were core.mjs:30820
skipped. Learn more at https://angular.io/guide/hydration.
Touchedtrue form-example.component.ts:18
Pristinefalse form-example.component.ts:23
valueChange[object Object] form-example.component.ts:28
1.1 form-example.component.ts:29
StatusVALID form-example.component.ts:34
```

Title inputumuzda ki değerimizi değiştirdiğimiz de ise gördüğünüz gibi değeri ilk defa değiştiği için PristineChangeEvent tetiklendi. Ardından ValueChangeEvent ve StatusChangeEvent tetiklenmiş oldu.

# ROUTE REDIRECTS WITH FUNCTIONS

Bir diğer özellikimiz ise sayfalar arası yönlendirmeler ile ilgili. Gelen bu özelliğin test etmek için öncelikle 2 tane componentimiz olmalı ardından. Route konfigürasyonlarınımızı yaptığımız app.routes.ts dosyamızı açalım.

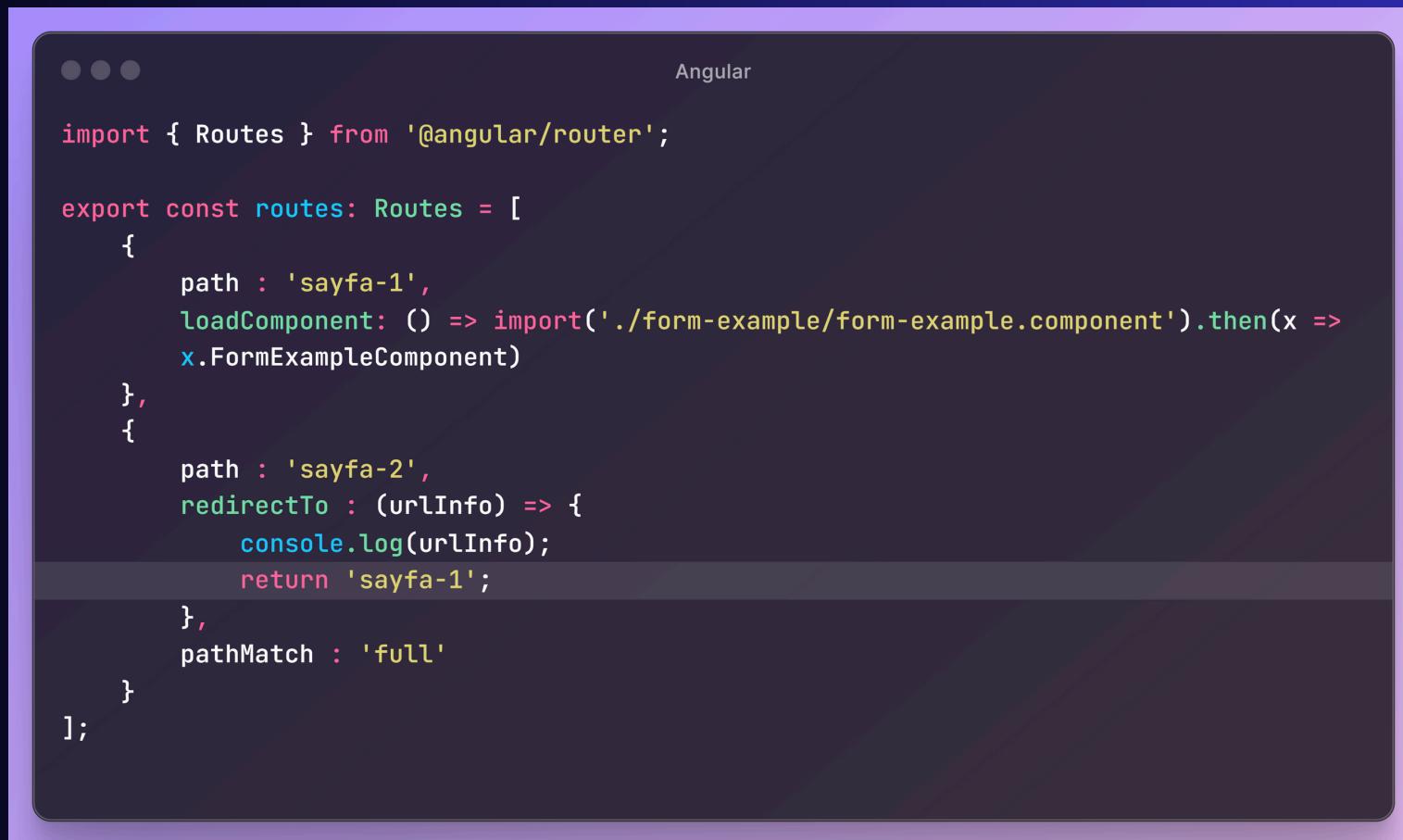
```
Angular

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

export const routes: Routes = [
  {
    path : 'sayfa-1',
    loadComponent: () => import('./form-example/form-example.component').then(x =>
      x.FormExampleComponent)
  },
  {
    path : 'sayfa-2',
    redirectTo : () => {
      return 'yonlenecek-sayfamizin-pathi';
    },
    pathMatch : 'full'
  }
]
```

Burada sayfa-2 url'ine istekte bulunduğuuzda bir fonksiyon devreye giriyor ve yönlenecek sayfamızın pathini buraya girdiğimize direkt ilgili sayfaya bizi yönlendiriyor.

İlgili fonksiyonumuz da parametre olarak urlInfo alabiliriz. Bu parametre bizlere adından da anlaşılacağı gibi url ile ilgili bilgileri getirmekte.

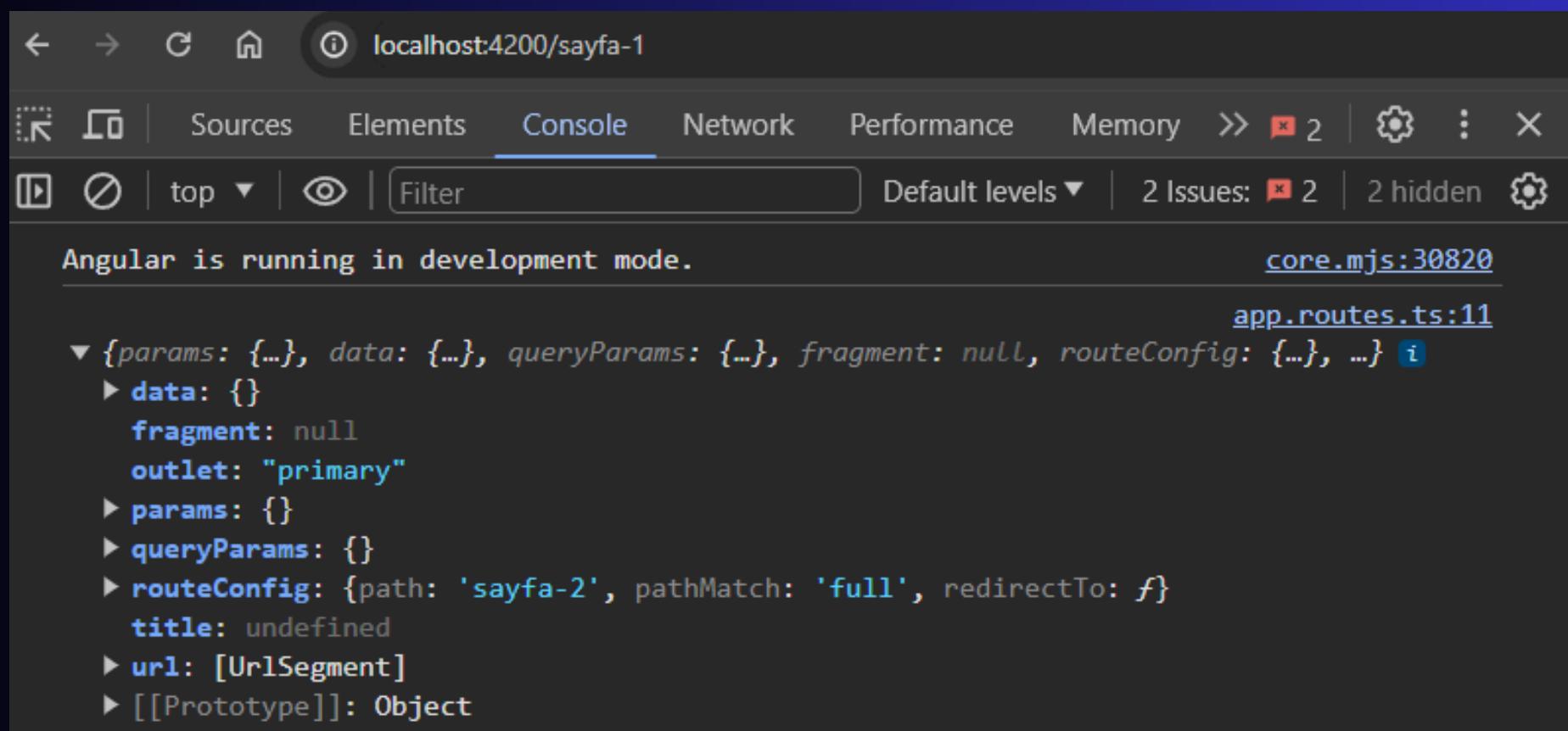


```
Angular

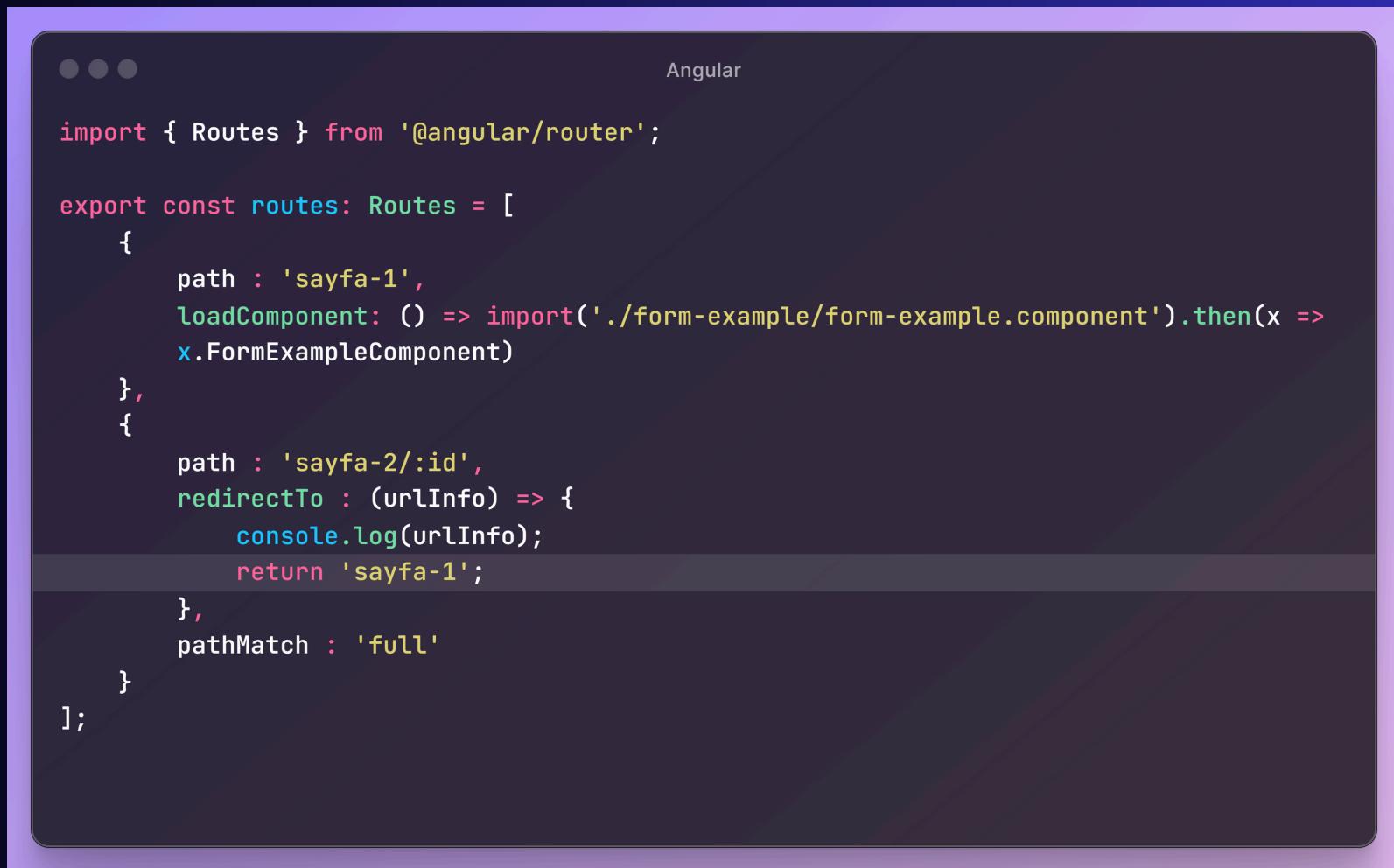
import { Routes } from '@angular/router';

export const routes: Routes = [
  {
    path : 'sayfa-1',
    loadComponent: () => import('./form-example/form-example.component').then(x =>
      x.FormExampleComponent)
  },
  {
    path : 'sayfa-2',
    redirectTo : (urlInfo) => {
      console.log(urlInfo);
      return 'sayfa-1';
    },
    pathMatch : 'full'
  }
];
```

İlgili konfigürasyonu yaptıktan sonra tarayıcımız da sayfa-2 url'ine bir istekte bulunduğumda beni otomatik olarak sayfa-1'e yollayacak ve console'a bilgilerimizi yazdıracak.



Son olarak ilgili url'de varsayıyalım ki bir id yolluyoruz bu değeri de yakalayabiliriz.

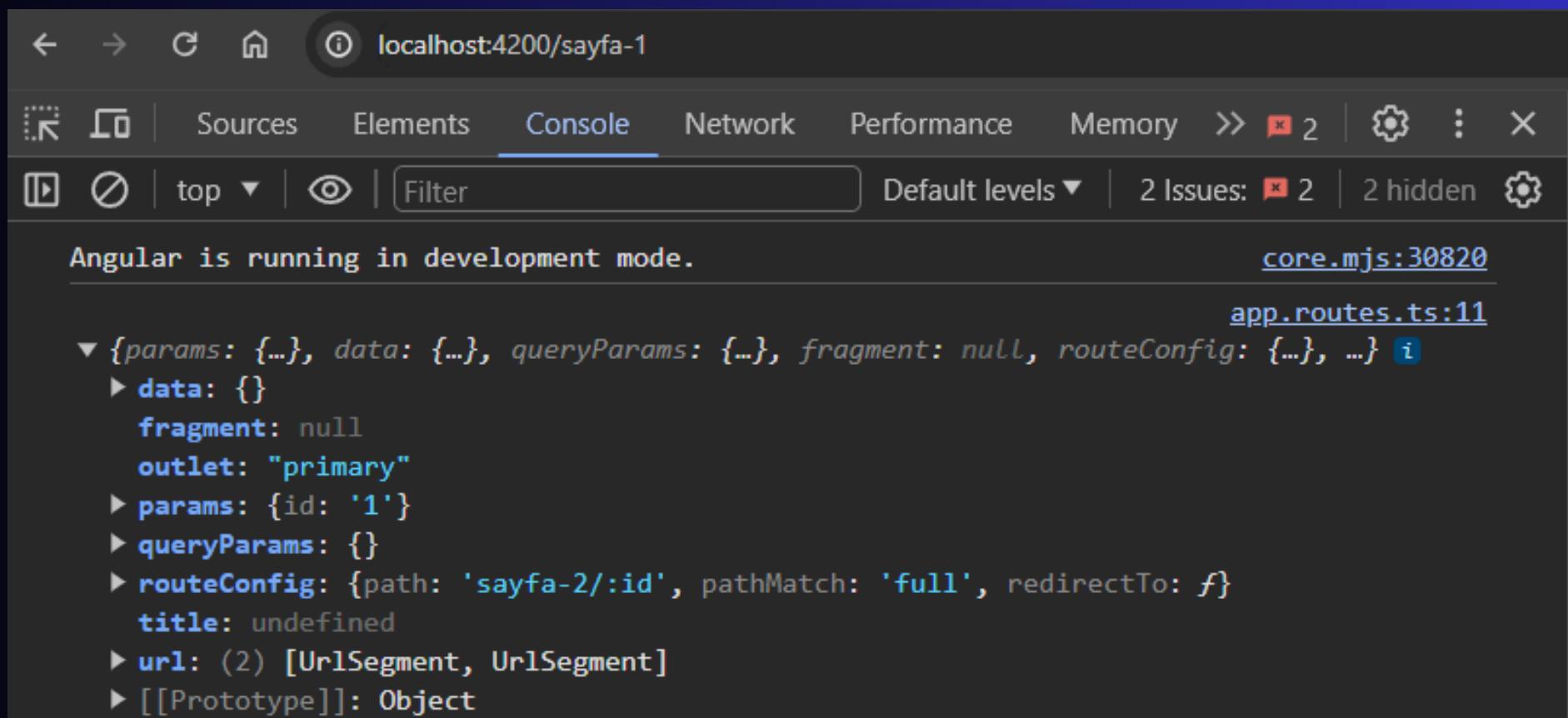


```
Angular

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

export const routes: Routes = [
  {
    path : 'sayfa-1',
    loadComponent: () => import('./form-example/form-example.component').then(x =>
      x.FormExampleComponent)
  },
  {
    path : 'sayfa-2/:id',
    redirectTo : (urlInfo) => {
      console.log(urlInfo);
      return 'sayfa-1';
    },
    pathMatch : 'full'
  }
];
```

Bu konfigürasyondan da sonra tekrar tarayıcıda sayfa-2/1 yazıp istekte bulunduğu da id : '1' olarak erişebilmekteyim.





# Angular 18

**HAZIRLAYAN :**  
**EĞEMEN**  
**AĞUSTOS**