Welcome to the Apprenticeship Program – Session 29

Agenda

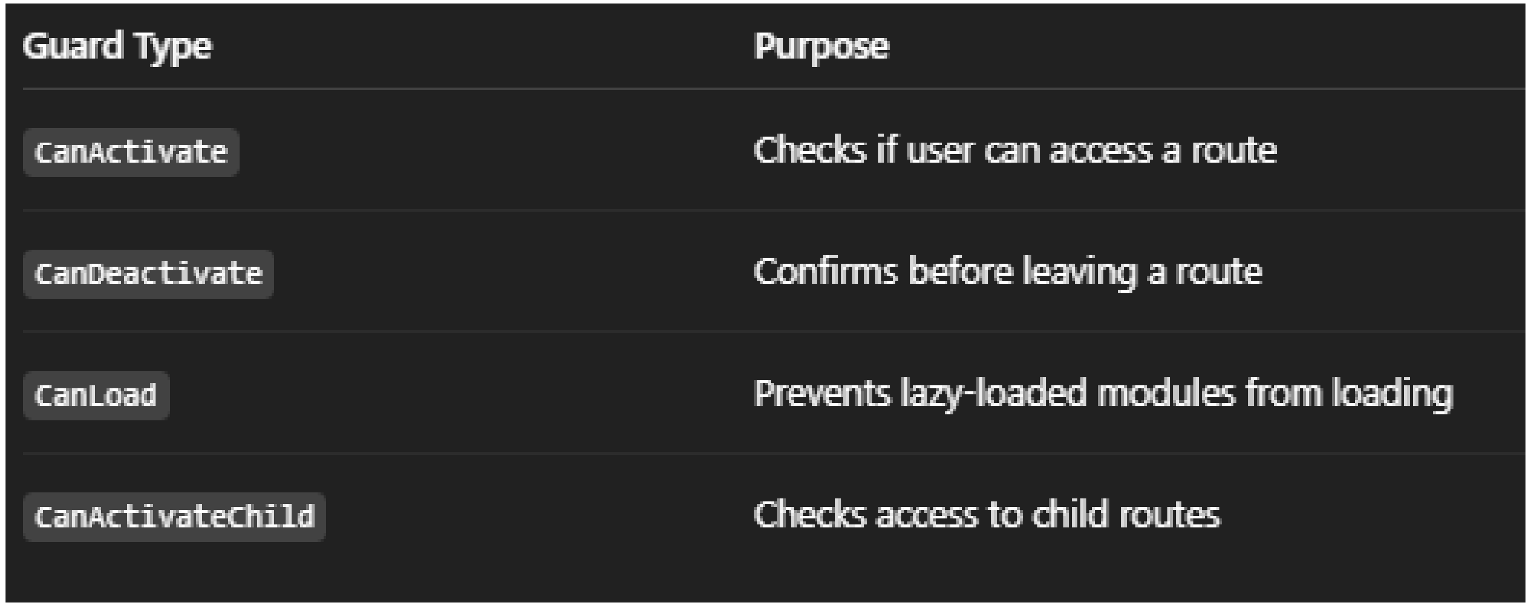
- What Are Route Guards?
- Types of Route Guards
- Creating a CanActivate Guard
- Creating a CanDeactivate Guard
- Optional Guards: CanLoad, CanActivateChild
- Real-World Use Cases

What Are Route Guards?

- Guards control access to routes based on logic (auth, roles, forms, etc.)
- Triggered before navigation occurs
- Return true (allow) or false (block), or an Observable or Promise
- 🧱 Acts as a middleware for route navigation.

Step8Up Academy

Types of Route Guards in Angular



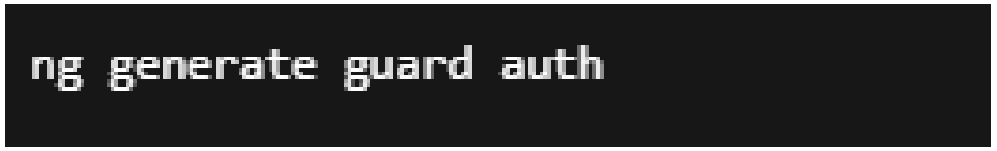
Guard Type	Purpose
`CanActivate`	Checks if user can access a route
`CanDeactivate`	Confirms before leaving a route
`CanLoad`	Prevents lazy-loaded modules from loading
`CanActivateChild`	Checks access to child routes

- **Can be reused across multiple routes or modules.**

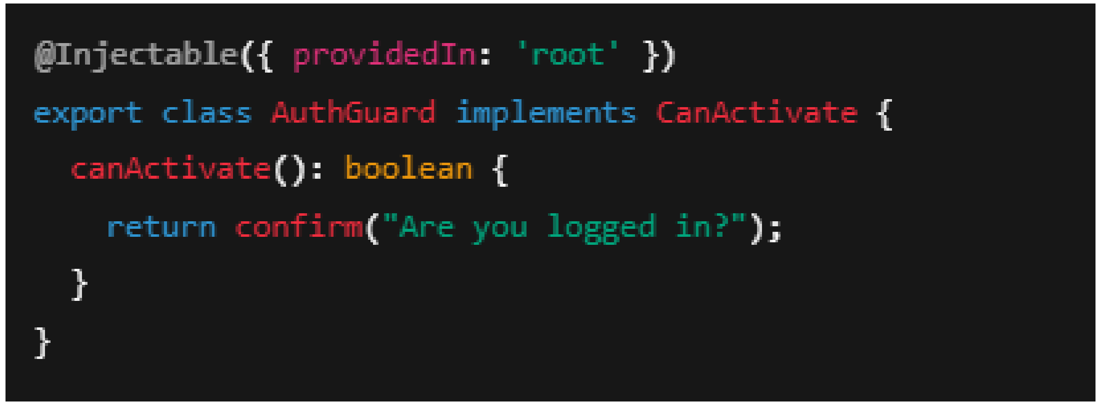
Create a CanActivate Auth Guard

- Step 1: Generate Guard.
- Step 2: Add logic

✅ Returns true to allow, or false to block navigation.

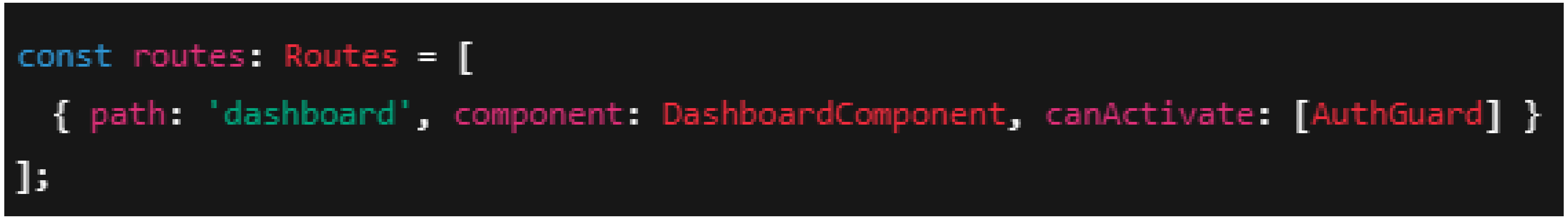


```
ng generate guard auth
```



```
@Injectable({ providedIn: 'root' })
export class AuthGuard implements CanActivate {
  canActivate(): boolean {
    return confirm("Are you logged in?");
  }
}
```

Apply CanActivate in Routes



```
const routes: Routes = [
  { path: 'dashboard', component: DashboardComponent, canActivate: [AuthGuard] }
];
```

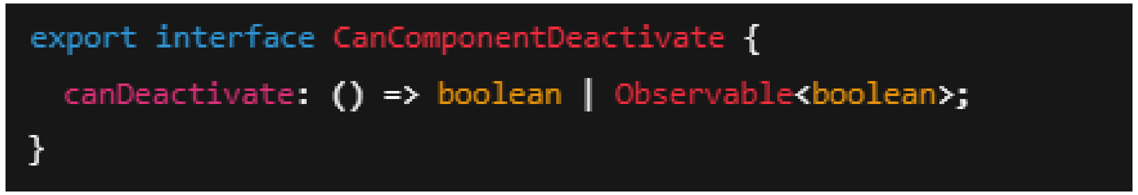
- Now /dashboard is protected by the guard
- If user says "No", they stay on the current route.

DEMO

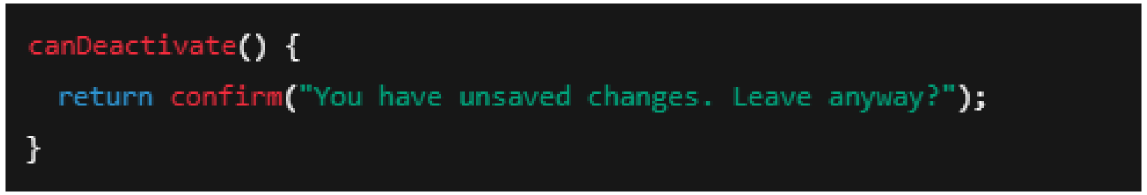
Create a CanDeactivate Auth Guard

- Step 1: Create Interface.
- Step 2: Implement in Component

✅ Used to warn users before navigating away (e.g., unsaved form).



```
export interface CanComponentDeactivate {
  canDeactivate: () => boolean | Observable<boolean>;
}
```



```
canDeactivate() {
  return confirm("You have unsaved changes. Leave anyway?");
}
```

Use CanDeactivate Guard in Route

- Guard Code:

```
@Injectable({ providedIn: 'root' })
export class DeactivateGuard implements CanDeactivate<CanComponentDeactivate> {
  canDeactivate(component: CanComponentDeactivate): boolean {
    return component.canDeactivate();
  }
}
```

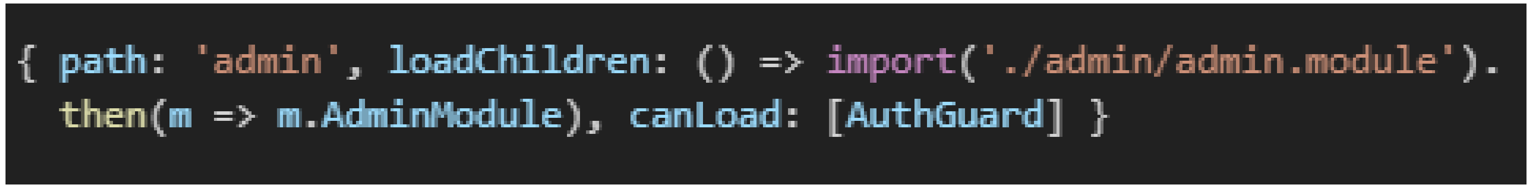
- Apply in Routes:

```
{ path: 'form', component: MyFormComponent, canDeactivate: [DeactivateGuard] }
```

✅ Prevents data loss on accidental navigation.

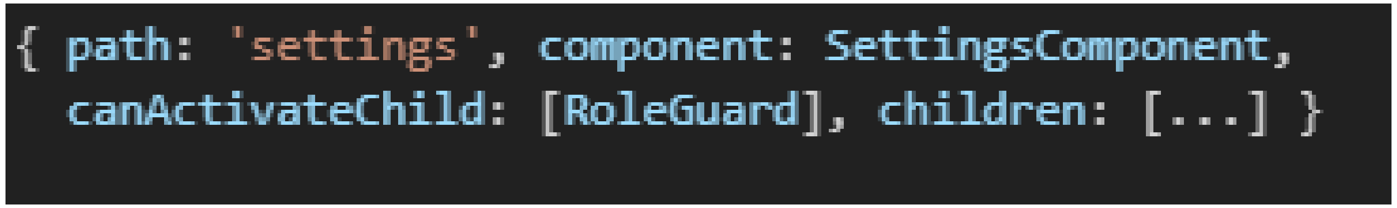
Optional Guards – Overview

- CanLoad: Prevents loading lazy-loaded modules unless condition is met.



```
{ path: 'admin', loadChildren: () => import('./admin/admin.module').
  then(m => m.AdminModule), canLoad: [AuthGuard] }
```

- CanActivateChild: Applies guard to child routes of a parent route.



```
{ path: 'settings', component: SettingsComponent,
  canActivateChild: [RoleGuard], children: [...] }
```

Step8Up Academy

Conclusion and Q&A

- **Key Takeaways:**
  - Guards are middleware for route navigation.
  - Add reusable logic to protect or restrict routes.
  - Helps enforce security, roles, and UX flow.
  - Easy to scale and manage via services and observables.

Questions? 👀