**Route Guards in Angular**

Route guards are a powerful mechanism in Angular that allow you to control access to specific routes within your application.1 They act as gatekeepers, determining whether a user is allowed to navigate to a particular route2 or not.3

**Types of Route Guards**

* **CanActivate:** Determines if a route can be activated.4 It checks if the user is authorized to access the route before allowing navigation.5
* **CanDeactivate:** Determines if a user can leave a route.6 This is useful for preventing accidental data loss, such as when a user navigates away from a form with unsaved changes.7
* **CanLoad:** Determines if a route can be loaded.8 This is useful for lazy-loaded modules, allowing you to control whether a module should be loaded at all.9
* **CanMatch:** Determines if a URL matches the route configuration.10

**Example: CanActivate Guard (Authentication)**

TypeScript

import { Injectable } from '@angular/core';

import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree, Router } from '@angular/router';

import { Observable } from 'rxjs';

import { AuthService } from './auth.service';

@Injectable({ providedIn: 'root' })

export class AuthGuard implements CanActivate {

constructor(private authService: AuthService, private router: Router) {}

canActivate(

route: ActivatedRouteSnapshot,

state: RouterStateSnapshot

): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {

return this.authService.isLoggedIn().pipe(

map(isLoggedIn => {

if (isLoggedIn) {

return true;

} else {

this.router.navigate(['/login']);

return false;

}

})

);

}

}

* This guard checks if the user is logged in using the AuthService.
* If the user is logged in, it returns true to allow navigation.
* If the user is not logged in, it redirects the user to the login page and returns false.

**Using the Guard in Routing Configuration:**

TypeScript

const routes: Routes = [

{ path: 'admin', component: AdminComponent, canActivate: [AuthGuard] }

];

* The canActivate property in the route configuration specifies that the AuthGuard must be passed before the AdminComponent can be activated.

**Other Guard Examples:**

* **CanDeactivate Guard (Unsaved Changes):**
  + Implement a guard that prompts the user to confirm if they want to navigate away from a component with unsaved changes.11
* **CanLoad Guard (Lazy Loading):**
  + Control whether a lazy-loaded module should be loaded based on certain conditions.

By effectively using route guards, you can enhance the security, user experience, and overall functionality of your Angular applications.12