**In Angular 19, routing is a fundamental concept that allows you to navigate between different views within your single-page application.**1

**Key Concepts:**

* **Routes:** Define the URL paths and the components that should be displayed for each path.2
* **Router Outlet:** A placeholder in your template where the routed components are displayed.3
* **Navigation:** Users navigate between routes by clicking links or interacting with other UI elements.4

**Setting Up Routing:**

1. **Import necessary modules:**

TypeScript

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

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

1. **Define routes:**

TypeScript

const routes: Routes = [

{ path: '', component: HomeComponent },

{ path: 'about', component: AboutComponent },

{ path: 'contact', component: ContactComponent },

// ... more routes

];

1. **Create components for each route:**

TypeScript

// home.component.ts

// about.component.ts

// contact.component.ts

1. **Use the router-outlet directive:**

HTML

<router-outlet></router-outlet>

1. **Create navigation links:**

HTML

<a routerLink="/">Home</a>

<a routerLink="/about">About</a>

<a routerLink="/contact">Contact</a>

**Advanced Routing Concepts:**

* **Route Parameters:** Pass data to components using route parameters (e.g., /user/:id).
* **Child Routes:** Define nested routes for more complex navigation structures.5
* **Route Guards:** Control access to routes using guards (e.g., canActivate, canDeactivate).
* **Lazy Loading:** Load components only when they are needed, improving initial load time.
* **Server-Side Rendering (SSR):** Enhance SEO and improve initial load times by rendering routes on the server.6

**Example:**

TypeScript

// app-routing.module.ts

const routes: Routes = [

{ path: '', component: HomeComponent },

{ path: 'user/:id', component: UserDetailComponent }

];

// user-detail.component.ts

constructor(private route: ActivatedRoute) {

this.route.params.subscribe(params => {

const userId = +params['id'];

// ...

});

}

**In Angular 19, routing continues to be a fundamental aspect of building single-page applications. By effectively utilizing routing, you can create user-friendly, well-structured, and performant applications.**7

**Note:** This information is based on current understanding and may evolve with future updates to Angular. Always refer to the official Angular documentation for the most up-to-date information and best practices.