**🧩 Simple Explanation – Part 1: Introduction to Angular Routing**

**🌐 What is Routing?**

* **Routing** means **navigating between pages** in a web application.
* Just like **road signs guide drivers** to the right place, **routes guide users** to the correct part of a website.

Example:  
When you click on

* Home → you go to /home
* About → you go to /about

Routing helps the browser know **which component or page** to show for each URL.

**🏗️ What is Angular Router?**

* **Angular Router** is a **feature in Angular** that helps your app move between pages (or components) **without reloading** the whole page.
* It watches the **URL** in the browser and decides **which component** to show.

Example:  
If your app has a Home page and About page:

* When the URL changes to /home → show the Home component
* When the URL changes to /about → show the About component

**⚡ Why Angular Router is Important**

Normally, in old websites, when you click a link:

* The **entire page reloads**.
* That makes it **slow**.

In Angular (and other modern frameworks like React or Vue):

* The **page does not reload**.
* Only the **content inside** the page changes.
* This makes your app feel **fast and smooth** — like a mobile app.

**💡 Example in Real Life**

Think of your Angular app like a **big house** 🏠:

* Each **room** = a **component** (Home, About, Contact)
* The **router** = a **map or guide** that helps you move between rooms **without leaving the house** (no page reloads)

**🧩 Part 2: Example – How Angular Routing Works**

**🧱 1️ Structure of the App**

In Angular, an app is made up of **many small components**.

In the first image, we see:

* **Navbar Component** → top part (menu or header)
* **Sidebar Component** → side menu
* **App Component** → main layout that holds everything
* **FindJob, InterviewList, PostJob Components** → different pages or sections that change when we navigate.

So, the **App Component** is like a frame that stays fixed.  
Inside it, the middle area changes based on routing.

**🚗 2️ When User Navigates**

In the second image:

* When you click on **“Find Job”** from the sidebar or navbar,
* Only the **middle part** (main area) changes — it shows the **FindJob Component**.

The **Navbar** and **Sidebar** remain the same — they don’t reload.  
This is what makes Angular apps fast and smooth. ✅

**⚡ 3️ What Happens Internally**

In the third image:

* When Angular reloads a part like **FindJob**, it only **renders that one component**.
* It does **not reload** the whole page (Navbar + Sidebar are not fetched again from the server).
* So, the page feels instant, just like a mobile app.

Example:  
When you click on *Find Job*, Angular only loads job-related data like “Job1”, “Job2”, etc., inside that section.

Navbar and Sidebar stay as they are.

**🧭 Part 3: How Does Route Work? (Simple Explanation)**

Let’s imagine you are **sending a letter** to your friend 📨

**🏠 1️ Sender and Receiver**

* **You (Sender)** = your **computer or phone**.
* **Your Friend (Receiver)** = another **computer or server** on the internet.

Example:  
You want to visit a website — that means **you send a request** to that website’s **server**.

**📬 2️ Addressing**

Just like a letter needs your friend’s **house address**,  
your computer needs the **address of the website**.

That website’s address is called an **IP Address** (example: 192.168.1.1).  
It tells the internet **where to send your request**.

**🚚 3️ Router – The Postal Sorting Center**

Once you “send” your letter (your data),  
it first goes to a **router** — like a **post office**.

The **router’s job** is to look at the IP address and decide  
➡️ *“Where should this packet go next?”*

It checks a **routing table** (like a map) to find the best and fastest path.

**🛣️ 4️ Routing Decision & Transmission**

Your data might travel through **multiple routers**  
(just like a letter passes through many post offices).

Each router moves it **closer** to the destination.  
This process continues until it reaches the correct server.

**🎯 5️ Receiver**

Finally, your **data packet** reaches the **destination server** (the website).  
The server reads your request — like your friend reading your letter.

**🔁 6️ Response (Reply)**

If the website needs to send data back (like your friend replying to your letter),  
the process happens in **reverse** —  
the data travels back through routers to your computer.

**💡 In Short**

| **Real World** | **Internet World** |
| --- | --- |
| You send a letter | You send a data packet |
| Your friend’s address | IP address |
| Post office | Router |
| Postal route | Network path |
| Friend receives and replies | Server responds to your computer |

Example 1 :

app-routing.module.ts

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

* This line **imports** NgModule from Angular’s core library.
* NgModule helps us **group related code** together into one module (like a box of related things).
* Here, we are going to make a special module just for **routing**.

🧠 Think of it like:

We’re creating a “Routing Box” to store all our route information.

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

* This line brings two important things:
  1. **RouterModule** → allows our app to use routing (page navigation).
  2. **Routes** → a type that defines the **list of paths** and their matching components.

🧠 Think of RouterModule like the engine that moves your app between pages.  
And Routes is like the **map** that tells it where to go.

import { FindJobComponent } from './find-job/find-job.component';

import { InterviewListComponent } from './interview-list/interview-list.component';

import { PostJobComponent } from './post-job/post-job.component';

 These lines **import the components** that will be shown for each route (page).

 Each component represents one page or view in your app.

const routes: Routes = [

   { path: '', redirectTo: 'find-job', pathMatch: 'full' },

  { path: 'find-job', component: FindJobComponent },

  { path: 'interviews', component: InterviewListComponent },

  { path: 'post-job', component: PostJobComponent },

  { path: '\*\*', redirectTo: 'find-job' }

];

* Here we are creating a **list (array)** named routes.
* This list contains **objects**, where each object defines:
  + the **path** (URL part after the /)
  + the **component** that should appear for that path

🧠 Think of this as your **navigation map**.

* + When the path is empty (just http://localhost:4200/),
  + Redirect (automatically go) to /find-job.
* pathMatch: 'full' means the entire path must be empty — not partial.

🧠 Example: If user types only the main URL → they’ll automatically see the “Find Job” page.

* When user visits /find-job,  
  Angular will display the FindJobComponent.

🧠 Example:  
http://localhost:4200/find-job → shows Find Job page.

* \*\* means **any unknown path** (for example, typing /xyz).
* If user goes to a wrong page, Angular redirects to the **Find Job** page.

🧠 Example:  
http://localhost:4200/anything → automatically redirects to /find-job.

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

#### imports: [RouterModule.forRoot(routes)]

* RouterModule.forRoot(routes) → loads the route configuration (the list above) into the app.
* .forRoot() means this is the **main** routing setup for the whole application (used only once, in root app).

🧠 Think of it like:

“Please use this list of routes when the app starts.”

|  |
| --- |
| **<router-outlet>** |

|  |
| --- |
| Loads the routed component here (Find Job, Post Job, etc.) |

* This is the **most important line** in routing.
* It works like a **placeholder** or **screen area**.
* When you navigate (for example, go to /find-job), Angular **loads that component (FindJobComponent)** inside this spot.

👉 So this is where your **page changes dynamically** based on what route is selected.

**routerLink="/find-job" :**

 It tells Angular to **navigate to the route /find-job** when clicked.

 It does **not reload the page**, it just changes the view (SPA – Single Page Application).