# Angular Router Demo

## 1. Setup Project

### 1.1 Install Dependencies

1. Change directory to calab:

* cd calab

1. Install dependencies by running the following command:

* npm install

## 2. Setup Application for Routing

### 2.1 Create Two New Components

1. Create a new component using CLI and name it First:

* npx -p @angular/cli ng generate component components/first

1. Create another component using CLI and name it Second:

* npx -p @angular/cli ng generate component components/second

## 3. Define and Use Your Routes

### 3.1 Define Routes In Routes Array

1. Open src/app/app.routes.ts file and do the following:
   * Import previously created components.
   * import {FirstComponent} from './components/first/first.component';  
     import {SecondComponent} from './components/second/second.component';
   * Define each route as an JavaScript object and add it to Routes array:
   * { path: 'first-component', component: FirstComponent},  
     { path: 'second-component', component: SecondComponent}

### 3.2 Use Defined Routs In an Application

1. Open src/app/app.component.ts file and do the following:
   * Import RouterLink from the @angular/router.
   * Update imports arrray with RouterLink: .js imports: [RouterOutlet, RouterLink],
2. Open src/app/app.component.html file and do the following:
   * Just below <div class="divider"... Use a routerLink attributes to add routes to selected elements:
   * <nav>  
      <ul>  
      <li><a routerLink="/first-component" >First Component</a></li>  
      <li><a routerLink="/second-component">Second Component</a></li>  
      </ul>  
     </nav>
   * Move <router-outlet /> from bottom of HTML page to just after the <nav>:

### 3.3 Review Changes

1. Start Angular Development Server if not yet started:

* npx -p @angular/cli ng serve
* *Otherwise refresh the browser tab to see updated view.*

## 4. Bind Route Info to Component Inputs

### 4.1 Enable Route Processing

1. Open src/app/app.config.ts file and do the following:
   * Import withComponentInputBinding from the @angular/router.
   * update provideRouter method with the following:
   * providers: [provideRouter(routes, withComponentInputBinding())]

### 4.2 Update the component to have an Input matching the name of the parameter

1. Open src/app/components/first/first.component.ts and add the following code:

* import { Component, Input } from '@angular/core';  
    
  @Component({  
   selector: 'app-first',  
   standalone: true,  
   imports: [],  
   templateUrl: './first.component.html',  
   styleUrl: './first.component.css'  
  })  
  export class FirstComponent {  
   @Input() name = '';  
  }

1. Open src/app/components/first/first.component.html and replace current <p> with the following:

* <p>Hello {{name}}</p>

### 4.3 Update path in router array to include path parameter or query string.

1. Open src/app/app.routes.ts file and do the following:
   * Update first-component path with the filowing:
   * { path: 'first-component/:name', component: FirstComponent},  
     { path: 'second-component', component: SecondComponent}
2. Open src/app/app.component.html file and do the following:
   * Update a First Component’s routerLink attribute to include name path parameter:
   * <li><a routerLink="/first-component/John" >First Component</a></li>

### 4.3 Review Changes

1. Start Angular Development Server if not yet started:

* npx -p @angular/cli ng serve
* *Otherwise refresh the browser tab to see updated view.*

## 5. Nesting Routes

### 5.1 Create child components

1. Create a new component using CLI and name it FirstChild:

* npx -p @angular/cli ng generate component components/first-child

1. Add Styling to first child component:
   * Open src/app/components/first-child/first-child.component.html and add replace current code with the following:
   * <div class="container">  
      <p>first-child works!</p>  
     </div>
   * Open src/app/components/first-child/first-child.component.css and add the following css:
   * .container {  
      background-color: bisque;  
      height: 100px;  
     }
2. Create another component using CLI and name it SecondChild:

* npx -p @angular/cli ng generate component components/second-child

1. Add Styling to second child component:
   * Open src/app/components/second-child/second-child.component.html and add replace current code with the following:
   * <div class="container">  
      <p>second-child works!</p>  
     </div>
   * Open src/app/components/second-child/second-child.component.css and add the following css:
   * .container {  
      background-color:cadetblue;  
      height: 100px;  
     }

### 5.2 Set Up Child Routes

1. Open src/app/app.routes.ts file and do the following:
   * Import FirstChildComponent and SecondChildComponent. .js import { FirstChildComponent } from './components/first-child/first-child.component'; import { SecondChildComponent } from './components/second-child/second-child.component';
   * Place child routes in a children array within the parent route:
   * { path: 'second-component', component: SecondComponent, children: [  
      {path: 'first-child', component: FirstChildComponent},  
      {path: 'second-child', component: SecondChildComponent}  
      ]   
     },

### 5.3 Update Parent

1. Open src/app/components/second/second.component.ts and add do the following:
   * Import RouterOutlet and RouterLink from @angular/router.
   * Update imports array with RouterOutlet and RouterLink.
   * imports: [RouterOutlet, RouterLink],
2. Open src/app/components/second/second.component.html and add do the following:
   * Use a routerLink attributes to add routes to selected elements:
   * <nav>  
      <ul>  
      <li><a routerLink="first-child">First Child</a></li>  
      <li><a routerLink="second-child">Second Child</a></li>  
      </ul>  
     </nav>  
     <router-outlet></router-outlet>

### 5.4 Review Changes

1. Start Angular Development Server if not yet started:

* npx -p @angular/cli ng serve
* *Otherwise refresh the browser tab to see updated view.*