Angular Routing

## What is Client Side Routing

Routing is an important feature for any frontend framework just like in traditional backend frameworks. It allows developers to build modern Single Page Applications (SPAs) that can be loaded once by the browser and provide multiple views that the user can navigate between them using the browser’s address bar and navigation buttons or hyperlinks in the application interface.

*Routing is the process of driving the UI of an application using URLs.*

Client side routing is similar in concept to server side routing but uses different mechanisms for implementation. In a traditional web application, you will mostly have multiple pages. Each page maps to a different URL which you can use to access it either directly by entering the URL in the browser’s address bar or via clicking a navigation link or hyperlink in the application UI which has the URL in its href attribute (e.g<a href="/page1">Page 1</a>).

In server side routing, when the user navigates to a specific page, the browser sends a request to get the page from the server and display it. In client side routing, the browser has already retrieved all the application in the first place and the logic to show the requested view will be done in the browser with JavaScript.

### What is a Client Side Router

Here comes the role of the client side (or JavaScript) router.

A client side router is an essential entity in Angular and most frontend frameworks and libraries. It is the code responsible for organizing the application into multiple views (also called screens or states) and navigating between them on user request.

For example, the router will display the home view when the application is first loaded, and then allow the user to navigate to other views like contact or about screens via a navigation menu.

### What is a URL

URL stands for **Uniform Resource Locator** and in its simplest definition it’s a web address that references a web page.

## Introducing the Angular 8 Router

The Angular 8 router is an essential element of the Angular platform. It allows developers to build **Single Page Applications** with multiple states and views using routes and components and allows client side navigation and routing between the various components. It’s built and maintained by the core team behind Angular development and it’s contained in the @angular/router package.

You can use the browser's URL to navigate between Angular components in the same way you can use the usual server side navigation.

Angular Router has a plethora of features such as:

* The support for multiple Router outlets which helps you easily add complex routing scenario like nested routing,
* Various path matching strategies ( prefix and full) to tell the Router how to match a specific path to a component,
* Easy access to route parameters and query parameters,
* Resolvers,
* Lazy loading of modules,
* Route guards for adding client side protection and allow or disallow access to components or modules, etc.

Angular 8 provides a powerful router that allows you to map browser routes to components.

various concepts related to Angular routing such as:

* The Components, routes and paths,
* The router outlet,
* The route matching strategies,
* Route parameters,
* Query parameters,
* Route guards,
* Route resolvers,
* The routerLink directive (replaces the href attribute),
* Auxiliary routes,
* Primary and secondary router outlets.

### Routes and Paths

In Angular, a **route** is an object (instance of [Route](https://angular.io/api/router/Route)) that provides information about which component maps to a specific path. A **path** is the fragment of a URL that determines where exactly is located the resource (or page) you want to access. You can get the path by taking off the domain name from the URL.

In Angular you can define a route using route configurations or instances of the [Route](https://angular.io/api/router/Route) interface.

A collection of routes defines the router configuration which is an instance of [Routes](https://angular.io/api/router/Routes).

Each route can have the following properties:

* path is a string that specifies the path of the route.
* [pathMatch](https://angular.io/api/router/Route#pathMatch) is a string that specifies the matching strategy. It can take prefix (default) or [full](https://angular.io/api/core/Version#full).
* component is a component type that specifies the component that should be mapped to the route.
* [redirectTo](https://angular.io/api/router/Route#redirectTo) is the URL fragment which you will be redirected to if a route is matched.