

Angular Routing

In a single-page app, you change what the user sees by showing or hiding portions of the display that correspond to particular components, rather than going out to the server to get a new page. As users perform application tasks, they need to move between the different [views](#) that you have defined.

To handle the navigation from one [view](#) to the next, you use the Angular `Router`. The `Router` enables navigation by interpreting a browser URL as an instruction to change the view.

To explore a sample app featuring the router's primary features, see the [Tour of Heroes](#).

Prerequisites

Before creating a route, you should be familiar with the following:

- [Basics of components](#)
- [Basics of templates](#)
- An Angular app—you can generate a basic Angular application using the [Angular CLI](#).

Learn about Angular routing

[Common routing tasks](#)

[Learn how to implement many of the common tasks associated with Angular routing.](#)

[Common routing tasks](#)

[Single-page applications \(SPAs\) routing tutorial](#)

[A tutorial that covers patterns associated with Angular routing.](#)

[Routing SPA tutorial](#)

[Tour of Heroes expanded routing tutorial](#)

[Add more routing features to the Tour of Heroes tutorial.](#)

[Routing Tour of Heroes](#)

[Creating custom route matches tutorial](#)

[A tutorial that covers how to use custom matching strategy patterns with Angular routing.](#)

[Custom route matches tutorial](#)

[Router reference](#)

[Describes some core router API concepts.](#)

[Router reference](#)