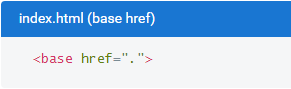
***<base href>***

Most routing applications should add a <base> element to the **index.html** as the first child in the <head> tag to tell the router how to compose navigation URLs.

If the **app** folder is the application root, as it is for our sample application, set the **href** value exactly as shown here.

**

**Router imports**

The Angular Component Router is an optional service that presents a particular component view for a given URL. It is not part of the Angular2 core. It is in its own library package, **@angular/router** . We import what we need from it as we would from any other Angular package.

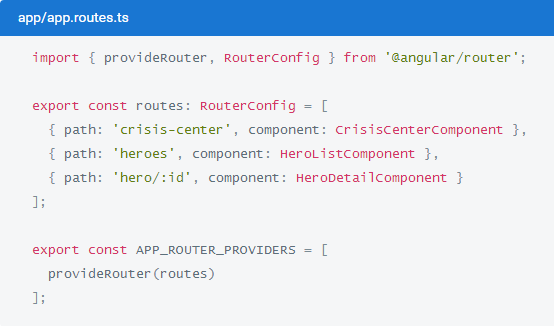
**

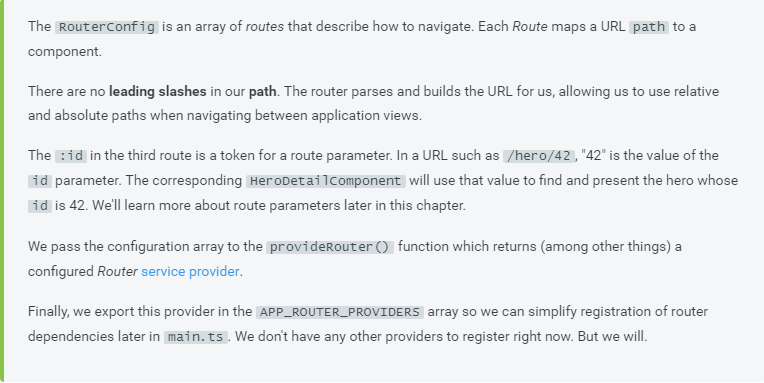
**Configuration**

The application will have one **router**. When the browser’s URL changes, the router looks for a corresponding **Route** from which it can determine the component to display.

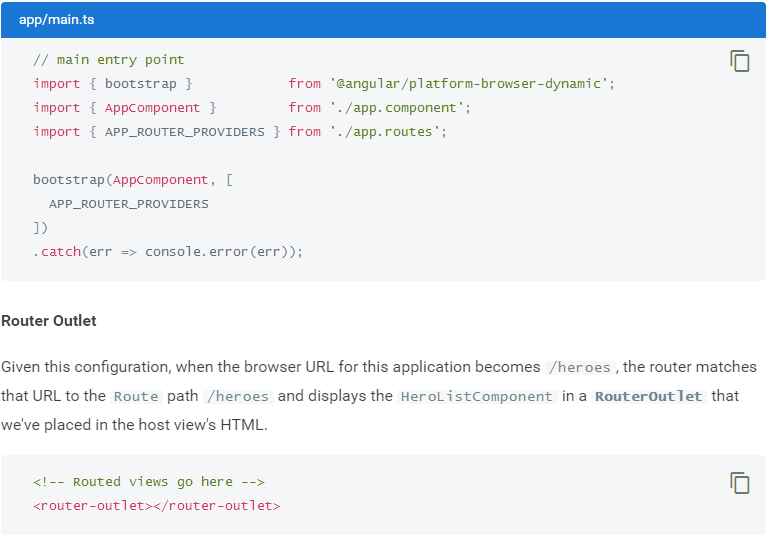
A router has no routes until we configure it. The preferred way is to bootstrap our application with an array of routes using the **provideRouter** function.

In the following example, we configure our application with three route definitions.

**

**

Next we open **main.ts** where we must register our router providers in the **bootstrap** method.

**

**Router Outlet**

Given this configuration, when the browser URL for this application becomes /heroes, the router matches that URL to the Route path /heroes and displays the HeroListComponent in a RouterOutlet that we’ve placed in the host view’s HTML.

<!—Routed views go here -->

<router-outlet></router-outlet>

**Router Links**

Now we have routes configured and a place to render them, but how do we navigate? The URL could arrive directly from the browser addressbar. But most of the time we navigate as a result of some user action such as the click of an anchor tag.

We add a **RouterLink** directive to the anchor tag and bind it to a template expression that returns an array of route link parameters (the link parameters array). The router ultimately resolves that array into a URL and a component view.

We see such bindings in the following **AppComponent** template:

**