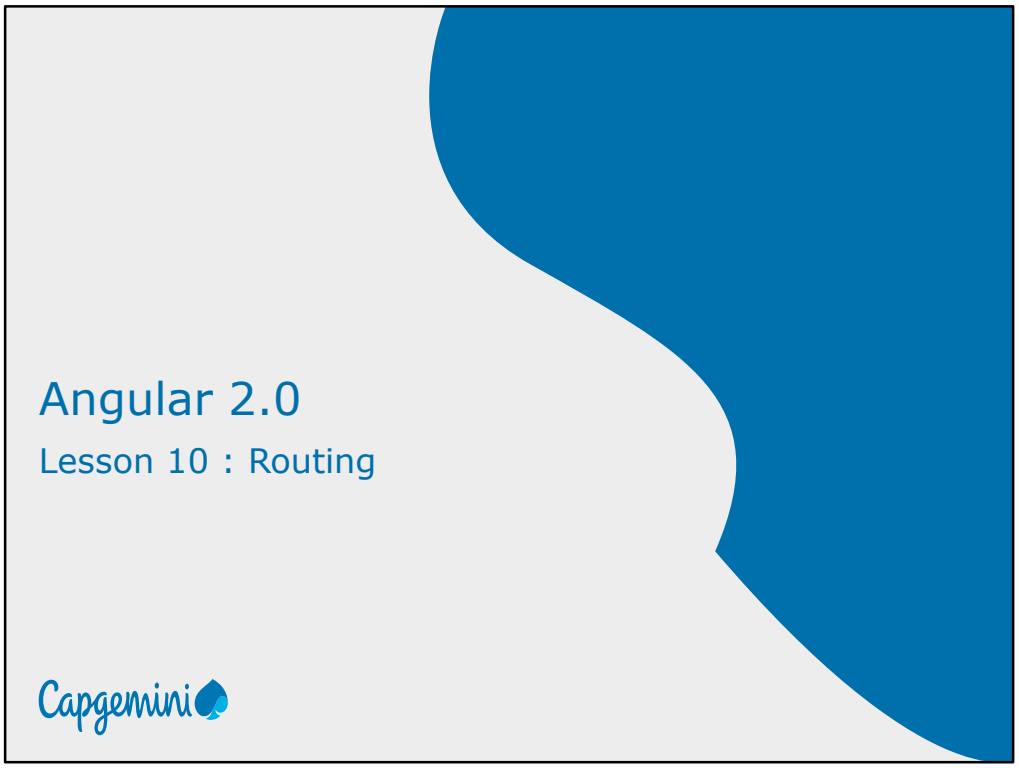
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.

Lesson Objectives

- Why use routing?
- Defining a route table
- Navigation using hyperlink & code
- Supplying parameters to a route URL

Instructor Notes:

Add instructor notes here.

Routing

- Routing means loading sub-templates depending upon the URL of the page.
- We can break out the view into a layout and template views and only show the view which we want to show based upon the URL the user is accessing.
- Routes are a way for multiple views to be used within a single HTML page. This enables you page to look more "app-like" because users are not seeing page reloads happen within the browser.
- Defining routes in application can:
  - Separate different areas of the app
  - Maintain the state in the app
  - Protect areas of the app based on certain rules

The browser is a familiar model of application navigation:
Enter a URL in the address bar and the browser navigates to a corresponding page.
Click links on the page and the browser navigates to a new page.
Click the browser's back and forward buttons and the browser navigates backward and forward through the history of pages you've seen.
The Angular Router ("the router") borrows from this model. It can interpret a browser URL as an instruction to navigate to a client-generated view. It can pass optional parameters along to the supporting view component that help it decide what specific content to present.
import { RouterModule, Routes } from '@angular/router';

Instructor Notes:

Add instructor notes here.

AngularJS Routes

- AngularJS routes enable us to create different URLs for different content in our application.
- Having different URLs for different content enables the user to bookmark URLs to specific content.
- In Angular 2 routes are configured by mapping paths to the component that will handle them.
- For instance, let consider an application with 2 routes:
  - A main page route, using the /#/home path;
  - An about page, using the /#/about path;
  - And when the user visits the root path (/#/), it will redirect to the home path.

import { RouterModule, Routes } from '@angular/router';
A router has no routes until you configure it. The following example creates four route definitions, configures the router via the RouterModule.forRoot method, and adds the result to the AppModule's imports array.

Instructor Notes:

Add instructor notes here.

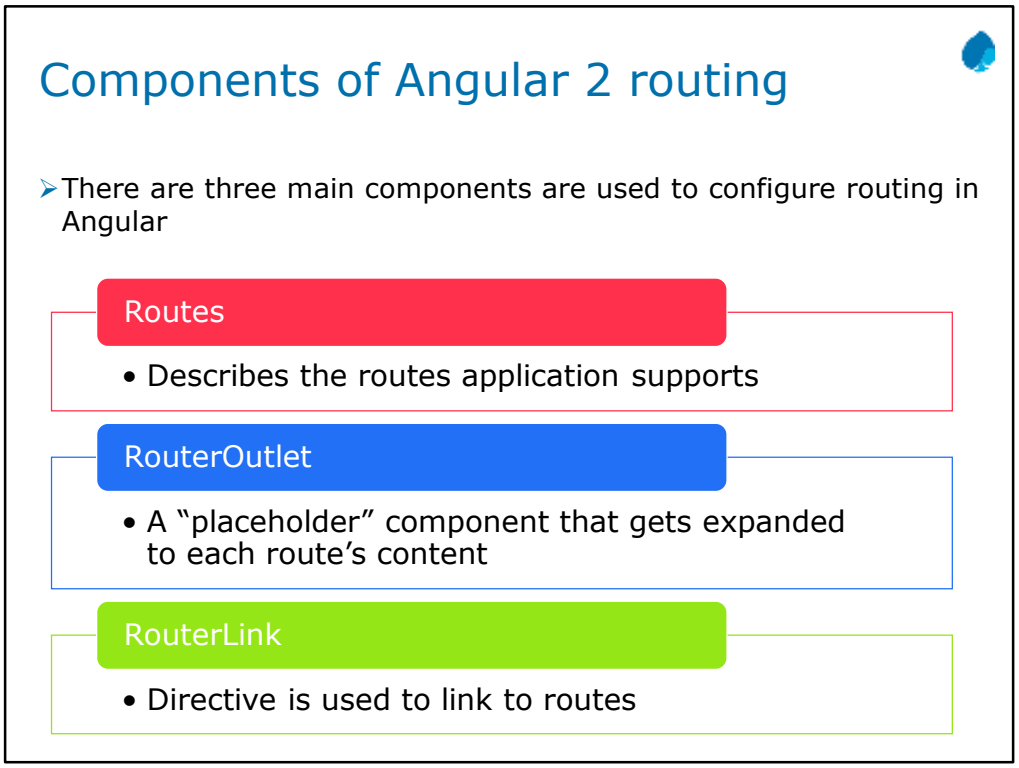
Routing Setup

- To implement Routing to Angular Application
- Import RouterModule and Routes from '@angular/router'
  - import { RouterModule, Routes} from '@angular/router';
- Define routes for application
  - const routes: Routes = [{ path: 'home', component: HomeComponent }];
- Install the routes using RouterModule.forRoot(routes) in the imports of NgModule
  - imports: [BrowserModule, RouterModule.forRoot(routes)]

A router has no routes until you configure it. The following example creates four route definitions, configures the router via the RouterModule.forRoot method, and adds the result to the AppModule's imports array.
The appRoutes array of *routes* describes how to navigate. Pass it to the RouterModule.forRoot method in the module imports to configure the router.
Each Route maps a URL path to a component. There are *no leading slashes* in the *path*. The router parses and builds the final URL for you, allowing you to use both relative and absolute paths when navigating between application views

Instructor Notes:

Add instructor notes here.



Basic Routing Steps

Set <base href="/"> tag
Use the RouterConfig on the root component
Use the RouterOutlet Component as placeholder
Use the RouterLink directive for Link
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 *after* a RouterOutlet that you've placed in the host view's HTML.
COPY CODE<router-outlet></router-outlet> <!-- Routed views go here -->
Router links
Now you have routes configured and a place to render them, but how do you navigate? The URL could arrive directly from the browser address bar.
The RouterLink directives on the anchor tags give the router control over those elements. The navigation paths are fixed, so you can assign a string to the routerLink (a "one-time" binding).
Had the navigation path been more dynamic, you could have bound to a template expression that returned an array of route link parameters (the *link parameters array*). The router resolves that array into a complete URL.
The **RouterLinkActive** directive on each anchor tag helps visually distinguish the anchor for the currently selected "active" route. The router adds the active CSS class to the element when the associated *RouterLink* becomes active. You can add this directive to the anchor or to its parent element.

Instructor Notes:

Router

Router Part	Meaning
Router	Displays the application component for the active URL. Manages navigation from one component to the next.
RouterModule	A separate Angular module that provides the necessary service providers and directives for navigating through application views.
Routes	Defines an array of Routes, each mapping a URL path to a component.
Route	Defines how the router should navigate to a component based on a URL pattern. Most routes consist of a path and a component type.
RouterOutlet	The directive (<router-outlet>) that marks where the router displays a view.

Instructor Notes:

Router (Contd…)

Router Part	Meaning
RouterLink	The directive for binding a clickable HTML element to a route. Clicking an element with a routerLinkdirective that is bound to a string or a link parameters array triggers a navigation.
RouterLinkActive	The directive for adding/removing classes from an HTML element when an associated routerLink contained on or inside the element becomes active/inactive.
ActivatedRoute	A service that is provided to each route component that contains route specific information such as route parameters, static data, resolve data, global query params, and the global fragment.
RouterState	The current state of the router including a tree of the currently activated routes together with convenience methods for traversing the route tree.

Instructor Notes:

Router (Contd…)

Router Part	Meaning
Link parameters array	An array that the router interprets as a routing instruction. You can bind that array to a RouterLink or pass the array as an argument to the Router.navigate method.
Routing component	An Angular component with a RouterOutlet that displays views based on router navigations.

Instructor Notes:

Add instructor notes here.

Routes

- To define routes for application, create a Routes configuration and then use RouterModule.forRoot(routes) to provide application with the dependencies necessary to use the router.
  - path specifies the URL this route will handle
  - component maps to the Component and its template
  - optional redirectTo is used to redirect a given path to an existing route

```
const routes: Routes = [
        { path: '', redirectTo: 'home', pathMatch: 'full' },
        { path: 'home', component: HomeComponent },
        { path: 'about', component: AboutComponent },
        { path: 'contact', component: ContactComponent },
        { path: 'contactus', redirectTo: 'contact' },
];
```

Instructor Notes:

Add instructor notes here.

RouterOutlet

- The router-outlet element indicates where the contents of each route component will be rendered.
- RouterOutlet directive is used to describe to Angular where in our page we want to render the contents for each route

```
@Component({
selector: 'my-app',
template:`<div class="container">
      <router-outlet></router-outlet>
      </div>`
})
```

Instructor Notes:

Add instructor notes here.

RouterLink

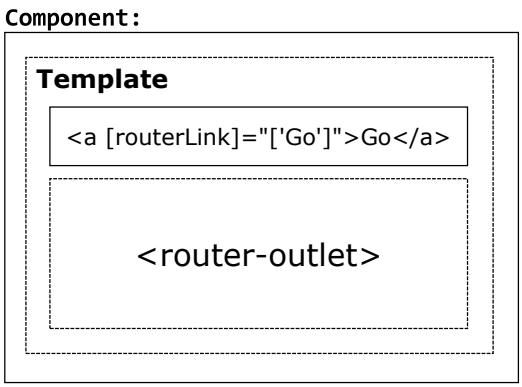
- It generates link based on the route path.
- routerLink navigates to a route

```
<div>
   <a [routerLink]="['Home']">Home</a>
   <a [routerLink]="[‘About']">About Us</a>
</div>
```

Instructor Notes:

Add instructor notes here.

RouterOutlet & RouterLink



HTML : <a [routerLink]="['Go']">Go

Code : router.navigate(['Go']);

Instructor Notes:

Add instructor notes here.

Routing Strategies

- The way the Angular application parses and creates paths from and to route definitions is now location strategy.
- HashLocationStrategy ('#/')
- PathLocationStrategy (HTML 5 Mode Default)

```
//import LocationStrategy and HashLocationStrategy
import {LocationStrategy, HashLocationStrategy} from
'@angular/common';

//add that location strategy to the providers of NgModule
providers: [
  { provide: LocationStrategy, useClass: HashLocationStrategy }
]
```

HTML5 client-side routing

With the introduction of HTML5, browsers acquired the ability to programmatically create new browser history entries that change the displayed URL without the need for a new request. This is achieved using the history.pushState method that exposes the browser's navigational history to JavaScript.

So now, instead of relying on the anchor hack to navigate routes, modern frameworks can rely on pushState to perform history manipulation without reloads.

This way of routing already works in Angular 1, but it needs to be explicitly enabled using $locationProvider.html5Mode(true).

In Angular 2, the HTML5 is the default mode.

Instructor Notes:

Add instructor notes here.

Route Parameters

- Route Parametes helps to navigate to a specific resource. For instance product with id 3
  - /products/3
- route takes a parameter by putting a colon : in front of the path segment
  - /route/:param
- To add a parameter to router configuration and to access the value refer the code given below

```
const routes: Routes =([
     { path:'/products/:id', name:‘Product', component:ProductComponent }
])

/*To access the parameter value */
routeParams.get('id')
```

Instructor Notes:

Add instructor notes here.

ActivatedRoute

- In order to access route parameter value in Components, we need to import ActivatedRoute

```
import { ActivatedRoute } from '@angular/router'
```

- inject the ActivatedRoute into the constructor of our component

```
export class ProductComponent {
  id: string;

  constructor(private route: ActivatedRoute) {
    route.params.subscribe(params => { this.id = params['id']; });
  }
}
```

Notice that route.params is an observable. We can extract the value of the param into a hard value by using .subscribe.

Instructor Notes:

Add instructor notes here.

Demo

- Demo Router
- Demo Router Passing Parameter

Add the notes here.

Instructor Notes:

Add instructor notes here.

Summary

- Routing means loading sub-templates depending upon the URL of the page.
- To implement Routing to Angular Application
- Import RouterModule and Routes from '@angular/router'
  - import { RouterModule, Routes} from '@angular/router';