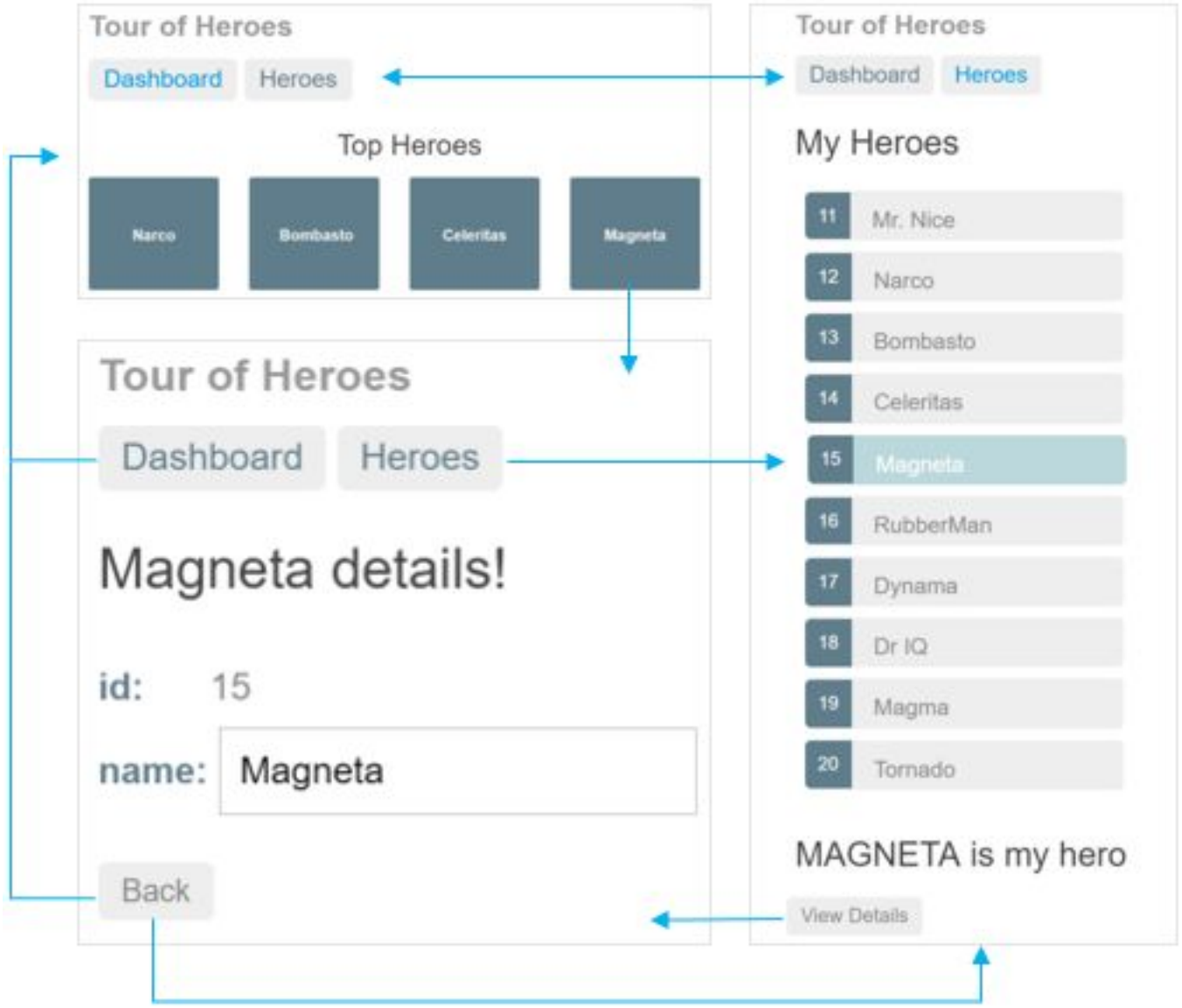
ANGULAR 2

ROUTING

LUXOFT

LXFT LISTED NYSE

ROUTING: COMPONENTS AND PAGE LAYOUT



ROUTES

Angular routes enables you to show different content depending on what route is chosen. A route is specified in the URL. Thus, the following URL's all point to the same AngularJS application, but each point to different route:

http://localhost:3000**/books**	books-> BookComponent
http://localhost:3000**/albums**	albums->AlbumsComponent
http://localhost:3000**/games**	Games->GamesComponent
http://localhost:3000**/apps**	Apps->AppsComponent

route: loads mapped component in placehoder in index.html

Advantages: better app structure; forward/backward in browser; app state can be shared as URL (to search engine, social framework, favorites, messengers, etc…)

BASE HREF

The HTML <base> tag is used to specify a base URI, or URL, for relative links.

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.

index.html:

```
<base href="/" />
```

ROUTING

```
@NgModule({
 imports: [
   BrowserModule, FormsModule,
   RouterModule.forRoot([
    { path: 'hero/:id', component: HeroDetailComponent },
    { path: 'crisis-center', component: CrisisListComponent },
    { path: 'heroes', component: HeroListComponent,
      data: { title: 'Heroes List' }
    },
    { path: '', component: HomeComponent },
    { path: '**', component: PageNotFoundComponent }
   ])
 ],

<!-- Routed views go here -->
<router-outlet></router-outlet>
```

ROUTING: DEFINING IN MAIN APP COMPONENT

app.component.ts

```
import { RouterModule }   from '@angular/router';

@Component({
   selector: 'my-app',
   template: `
      <nav>
         <a routerLink="/crisis-center" routerLinkActive="active">
                  Crisis Center</a>
         <a routerLink="/heroes" routerLinkActive="active">
                  Heroes</a>
      </nav>
      <router-outlet></router-outlet>`
})
export class AppComponent {}
```

ROUTING: CHANGING URL PROGRAMMATICALLY

```
@Component({
   selector: 'my-heroes',
   templateUrl: 'app/heroes.component.html',
   styleUrls:  ['app/heroes.component.css’]
})
export class HeroesComponent implements OnInit {
   heroes: Hero[];
   selectedHero: Hero;

   constructor( private router: Router, private heroService: HeroService) {}
   getHeroes() {
      this.heroService.getHeroes().then(heroes => this.heroes = heroes);
   }
   ngOnInit() { this.getHeroes(); }
   onSelect(hero: Hero) { this.selectedHero = hero; }
   gotoDetail() {
      this.router.navigate([’/hero', { id: this.selectedHero.id }]);
   }
}
```


goes to localhost:3000/hero/15

ROUTING: HEROES TEMPLATE

```
<h2>My Heroes</h2>
<ul class="heroes">
   <li *ngFor="let hero of heroes"
      [class.selected]="hero === selectedHero"
      (click)="onSelect(hero)">
      <span class="badge">{{hero.id}}</span> {{hero.name}}
   </li>
</ul>

<div *ngIf="selectedHero">
   <h2>
      {{selectedHero.name | uppercase}} is my hero
   </h2>
   <button (click)="gotoDetail()">View Details</button>
</div>
```

ROUTING: HERO DETAILS COMPONENT

app/hero-detail.component.html

```
@Component({
  selector: 'my-hero-detail',
  templateUrl: 'app/hero-detail.component.html',
})
export class HeroDetailComponent implements OnInit {
  hero: Hero;
  constructor(private heroService: HeroService,
              private route: ActivatedRoute, private location: Location) { }
    goBack(): void { this.location.back(); }

    ngOnInit(): void {
        this.route.params
         .switchMap((params: Params) =>
              this.heroService.getHero(params['id']))
        .subscribe(hero => this.hero = hero);
    }
}
```

ROUTING: HERO DETAILS TEMPLATE

hero-detail.component.html

```
<div *ngIf="hero">
  <h2>{{hero.name}} details!</h2>
  <div>
    <label>id: </label>{{hero.id}}</div>
  <div>
    <label>name: </label>
    <input [(ngModel)]="hero.name" placeholder="name" />
  </div>
  <button (click)="goBack()">Back</button>
</div>
```

CANDEACTIVATE

app/services/CanDeactivateTeam.ts

```
@Injectable()
class CanDeactivateTeam implements CanDeactivate<TeamComponent> {
  constructor(private permissions: Permissions, private currentUser: UserToken) {}

  canDeactivate(component: TeamComponent, route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean>|Promise<boolean>|boolean {
        return this.permissions.canDeactivate(this.currentUser, route.params.id);
  }
}
```

app/AppModule.ts

```
@NgModule({
  imports: [
    RouterModule.forRoot([ {
        path: 'team/:id', component: TeamComponent,
        canDeactivate: ['canDeactivateTeam']
    }])
  ],
```

REDIRECTING ROUTES

The preferred solution is to add a redirect route that translates the initial relative URL ('') to the desired default path (/heroes). The browser address bar shows .../heroes as if you'd navigated there directly.

```
const appRoutes: Routes = [
        { path: 'crisis-center', component: CrisisListComponent },
        { path: 'heroes',       component: HeroListComponent },
        { path: '',   redirectTo: '/heroes', pathMatch: 'full' },
        { path: '**', component: PageNotFoundComponent }
];
```

HASH STRATEGY

Here's the *Crisis Center* URL in this "HTML 5 pushState" style:

localhost:3002/crisis-center/

Here's a "hash URL" that routes to the *Crisis Center*

localhost:3002/src/#/crisis-center/

The router supports both styles with two LocationStrategy providers:

- **PathLocationStrategy** - the **default** "HTML 5 pushState" style.
- **HashLocationStrategy** - the "hash URL" style.

You can go old-school with the **HashLocationStrategy** by providing the **useHash: true** in an object as the second argument of the **RouterModule.forRoot** in the **AppModule**:

```
RouterModule.forRoot(routes, { useHash: true }) // .../#/crisis-center/
```