

# Routing

---



**John Papa**

PRINCIPAL ARCHITECT

@john\_papa   [www.johnpapa.net](http://www.johnpapa.net)



# Overview



**Defining Routes**

**Routing Modules**

**Route Parameters**

**Guards**



# Routing Basics

Routing allows our application to navigate between different Components, passing parameters where needed



# Routing Then and Now

**Angular 1**

**Angular 2**

index.html

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

Define the `<base>` element

Required for routing to work properly



1

2

3

4

5

# Routing, Step by Step



1

# Routing, Step by Step

2

1

Import  
RouterModule

3

4

5



1

# Routing, Step by Step

2

1

Import  
RouterModule

2

Import  
@angular/router

3

4

5





1

# Routing, Step by Step

2

1

Import  
RouterModule

2

Import  
`@angular/router`

3

Define the routes

3

4

5



1

# Routing, Step by Step

2

1

Import  
RouterModule

2

Import  
`@angular/router`

3

Define the routes

3

4

4

Declare a  
`<router-outlet>`

5



1

# Routing, Step by Step

2

1

Import  
RouterModule

2

Import  
@angular/router

3

Define the routes

3

4

4

Declare a  
<router-outlet>

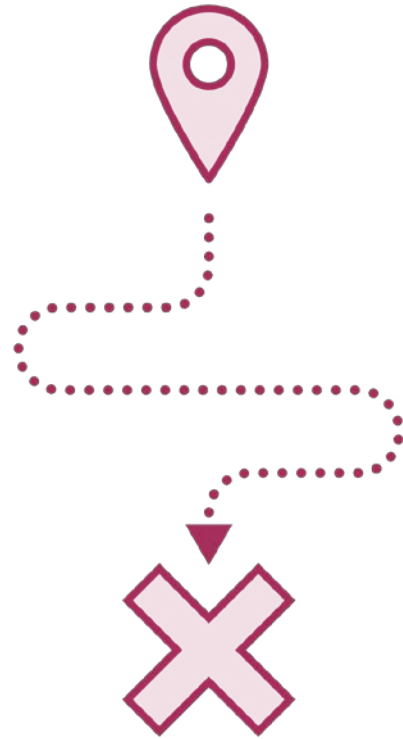
5

Add  
[routerLink]  
bindings

5



# Routing



```
import { NgModule } from '@angular/core';  
import { Routes, RouterModule } from '@angular/router';
```

Import routing features

Import **RouterModule**

**RouterModule** gives us access to routing features

**Routes** help us declare or route definitions



```
const routes: Routes = [  
  { path: '', pathMatch: 'full', redirectTo: 'characters', },  
  { path: 'characters', component: CharacterListComponent },  
  { path: 'character/:id', component: CharacterComponent },  
  { path: '**', pathMatch: 'full', component: PageNotFoundComponent }  
];
```

## Defining Routes

Define the route's **path**

Indicate parameters with **:**

Set the **component** that we'll route to



## app-routing.module.ts

```
@NgModule({  
  imports: [RouterModule.forRoot(routes)],  
  exports: [RouterModule]  
})  
export class AppRoutingModule { }
```

Only use `forRoot()` for the root module's routes

Define a **Module**

Create a routing module using our routes, and import it

Export our new **AppRoutingModule**



## app-routing.module.ts

Configure routes

```
const routes: Routes = [  
  { path: '', pathMatch: 'full', redirectTo: 'characters', },  
  { path: 'characters', component: CharacterListComponent },  
  { path: 'character/:id', component: CharacterComponent },  
  { path: '**', pathMatch: 'full', component: PageNotFoundComponent }  
];
```

Use route to make a  
NgModule

```
@NgModule({  
  imports: [RouterModule.forRoot(routes)],  
  exports: [RouterModule]  
})
```

Create and export an  
explicitly named  
NgModule

```
export class AppRoutingModule { }
```

Export components

```
export const routableComponents = [  
  CharacterListComponent,  
  CharacterComponent,  
  PageNotFoundComponent  
];
```

Export the Module and the Components



We define a Routing  
Module, and import it into  
the App Root or Feature  
Module



## app.module.ts

```
import { AppComponent } from './app.component';
import { AppRoutingModuleModule, routableComponents } from './app-routing.module';

@NgModule({
  imports: [BrowserModule, AppRoutingModuleModule],
  declarations: [AppComponent, routableComponents]
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Import the module

Declare the  
components

## Using Our Routing Module

Import our routing module

Declare the components



# Routing in a Template



EXPLORER

OPEN EDITORS

A2-FIRST-LOOK

router-guard

router-lazy

api

app

characters

vehicles

app.component.html

TS app.component.ts

TS app.module.ts

TS app.routing.ts

TS config.ts

TS page-not-found.component.ts

TS rxjs-extensions.ts

node\_modules

typings

.editorconfig

example-config.json

index.html

JS karma-test-shim.js

K karma.conf.js

TS main.ts

app.component.html x

```
1 <div>
2   <header>
3     <h1>Storyline Tracker</h1>
4     <h3>Lazily Loaded Router Demo</h3>
5     <nav>
6       <ul>
7         <li><a [routerLink]="['/characters']" href="">Characters</a></li>
8         <li><a [routerLink]="['/vehicles']" href="">Vehicles</a></li>
9       </ul>
10    </nav>
11  </header>
12  <main>
13    <section>
14      <router-outlet></router-outlet>
15    </section>
16  </main>
17 </div>
18
```

# RouterLink

The **RouterLink** directive navigates to a route path

We define the link parameters array

```
<nav>
  <ul>
    <li><a [routerLink]="['/characters']" href="">Characters</a></li>
    <li><a [routerLink]="['/vehicles']" href="">Vehicles</a></li>
  </ul>
</nav>
```

Define the routerLink

Set the link parameters

app.component.html

app.component.html

```
<router-outlet></router-outlet>
```

## Using the RouterOutlet

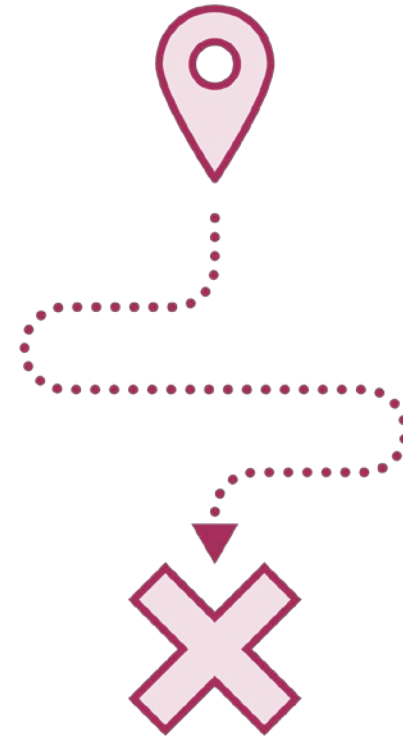
Defines where to put the components when we route

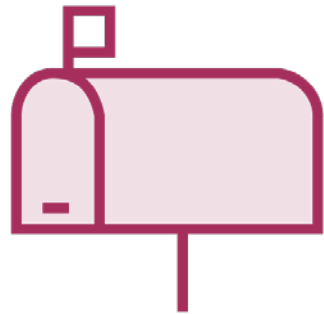


Demo



Routing





# Routing Parameters



```
const routes: Routes = [  
  { path: '', pathMatch: 'full', redirectTo: 'characters', },  
  { path: 'characters', component: CharacterListComponent },  
  { path: 'character/:id', component: CharacterComponent },  
  { path: '**', pathMatch: 'full', component: PageNotFoundComponent }  
];
```

Define the id parameter

## Defining Parameters

Indicate parameters with :





# Passing Data Through Routing

## Snapshot

Easiest, as long as parameter values do not change

## Observable

Gets new parameter values when component is re-used

## Resolvers

Gets data before a component loads



## session.component.ts

```
export class SessionComponent implements OnInit {  
  private id: any;  
  
  constructor(private route: ActivatedRoute) { }  
  
  ngOnInit() {  
    this.id = parseInt(this.route.snapshot.params['id']);  
    this.getSession();  
  }  
}
```

Inject the ActivatedRoute

Grab the parameter

## Snapshots

Grab the **ActivatedRoute**

Access the snapshot parameter by name

Easiest, if component is not reused



# ActivatedRoute and Observables

**route.params** is an Observable

Map the response, perform side effects, and subscribe

```
export class SessionComponent implements OnInit {  
  private id: any;  
  
  constructor(private route: ActivatedRoute) { }  
  
  ngOnInit() {  
    this.route.params.map(params => params['id'])  
      .do(id => this.id = parseInt(id))  
      .subscribe(id => this.getSession());  
  }
```

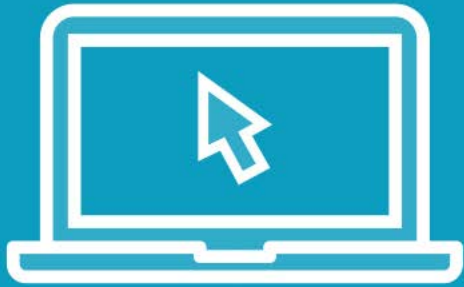
Map each response

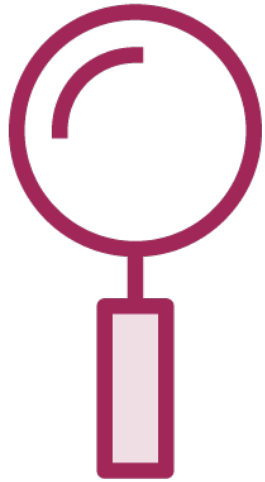
Subscribe to the stream

**session.component.ts**

# Routing Parameters

Demo





# Routing Resolvers



# Resolvers

Get data prior to traversing the route



## vehicle-resolver.service.ts

```
@Injectable()
export class VehicleResolver implements Resolve<Vehicle> {
  constructor(
    private vehicleService: VehicleService,
    private router: Router) { }

  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {
    let id = +route.params['id'];
    return this.vehicleService.getVehicle(id)
      .map(vehicle => vehicle ? vehicle : new Vehicle())
      .catch((error: any) => {
        console.log(`${error}. Heading back to vehicle list`);
        this.router.navigate(['/vehicles']);
        return Observable.of(null);
      });
  }
}
```

Grab the route and its parameter

Return the vehicle

Service that operates in the midst of a routing action

```
{  
  path: 'vehicles:id',  
  component: VehicleComponent,  
  resolve: {  
    vehicle: VehicleResolver  
  }  
},
```

Define one or more resolvers

## Resolvers

Defined within the route configuration

Remember to provide the resolver service





vehicle.component.ts

Subscribe to the data

```
this.route.data.subscribe((data: { vehicle: Vehicle }) => this.vehicle = data.vehicle);
```

Grab the resolved vehicle

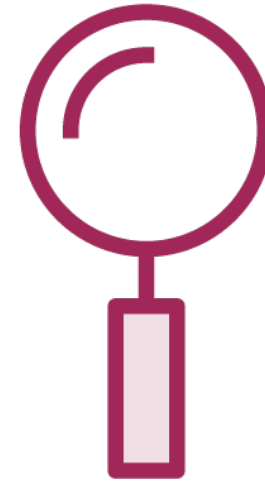
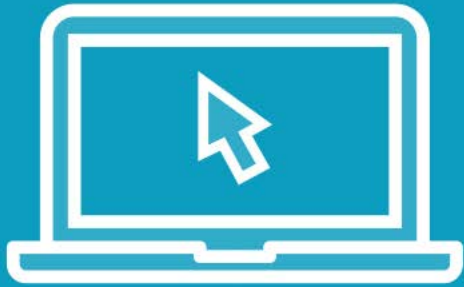
## Getting Resolver Data

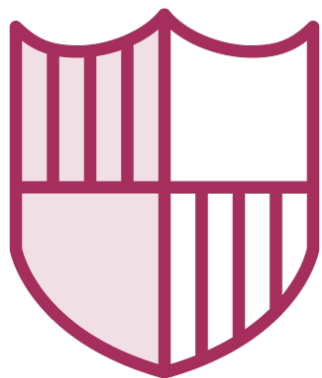
Subscribe to the **data** property on the **ActivatedRoute**



# Routing Resolvers

Demo





# Routing Guards



# Guards

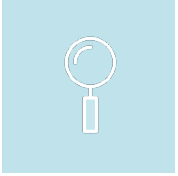
Guards allow us to make a decision at key points in the routing lifecycle and either continue, abort or take a new direction





# Types of Router Guards



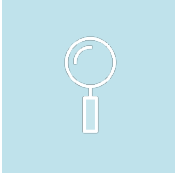


# Types of Router Guards



Resolve





# Types of Router Guards

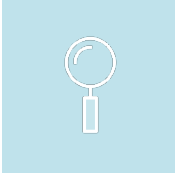


Resolve



CanActivate





# Types of Router Guards



Resolve



CanActivate



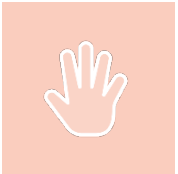
CanActivateChild







# Types of Router Guards



Resolve



CanActivate



CanActivateChild

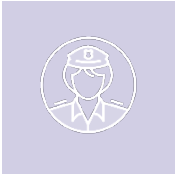


CanDeactivate





# Types of Router Guards



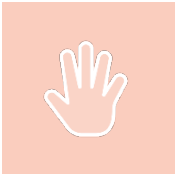
Resolve



CanActivate



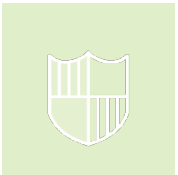
CanActivateChild



CanDeactivate



CanLoad



## auth-guard.service.ts

Implement the interface

```
@Injectable()
export class AuthGuard implements CanActivate {
  constructor(private userProfileService: UserProfileService, private router: Router) { }

  canActivate(next: ActivatedRouteSnapshot, state: RouterStateSnapshot) {
    if (this.userProfileService.isLoggedIn) {
      return true;
    }
    this.router.navigate(['/login'], { queryParams: { redirectTo: state.url } });
    return false;
  }
}
```

Return true or false

## CanActivate Guard

Implement the **CanActivate** interface

Make a determination if the route should be activated

Can re-navigate elsewhere



```
{  
  path: 'dashboard',  
  component: DashboardComponent,  
  canActivate: [AuthGuard]  
},
```

Apply the guard(s)

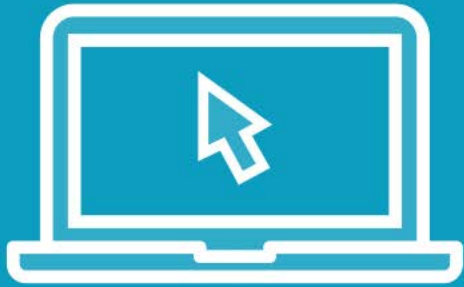
## Applying a Guard

### Apply to the route



# Routing Guards

Demo



# Child Routes

A Component may define routes for other Components. This creates a series of hierarchical child routes.



## app-routing.module.ts

```
const routes: Routes = [  
  { path: '', pathMatch: 'full', redirectTo: 'characters', },  
  { path: 'login', component: LoginComponent },  
  {  
    path: 'characters',  
    component: CharactersComponent,  
    canActivate: [CanActivateAuthGuard],  
    children: [  
      { path: '', component: CharacterListComponent },  
      { path: ':id', component: CharacterComponent },  
    ],  
  },  
],
```

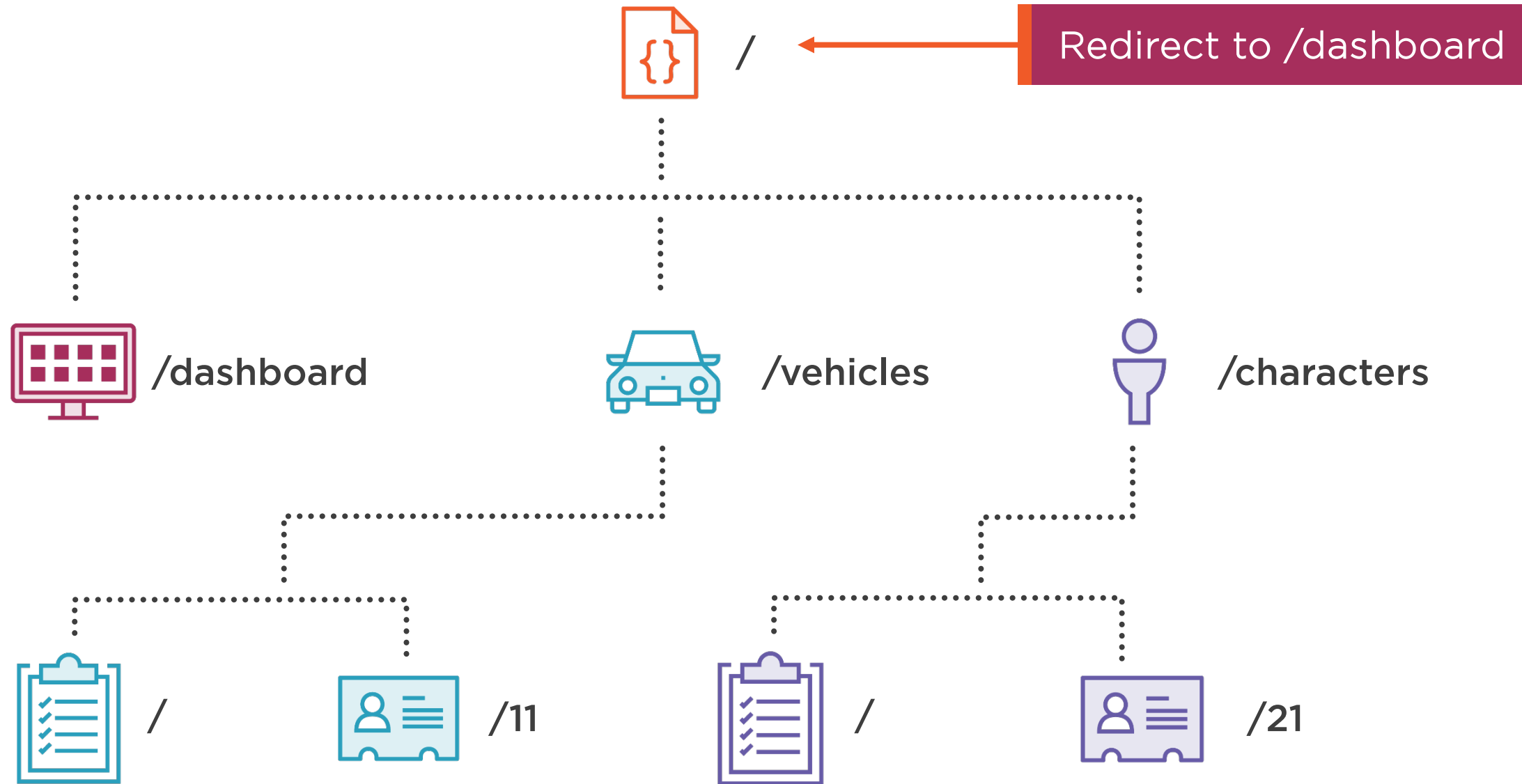
Child routes

## Child Routes

URLs for parent child

e.g. **characters/** and **characters/72**







Notice where we provided?



# Module Providers

Provided to the root injector

Available everywhere

```
@NgModule({  
  imports: [...  
  ],  
  declarations: [AppComponent, routableComponents],  
  providers: [  
    CanActivateAuthGuard, CharacterService, UserProfileService, VehicleService  
  ],  
  bootstrap: [AppComponent]  
})  
export class AppModule { }
```

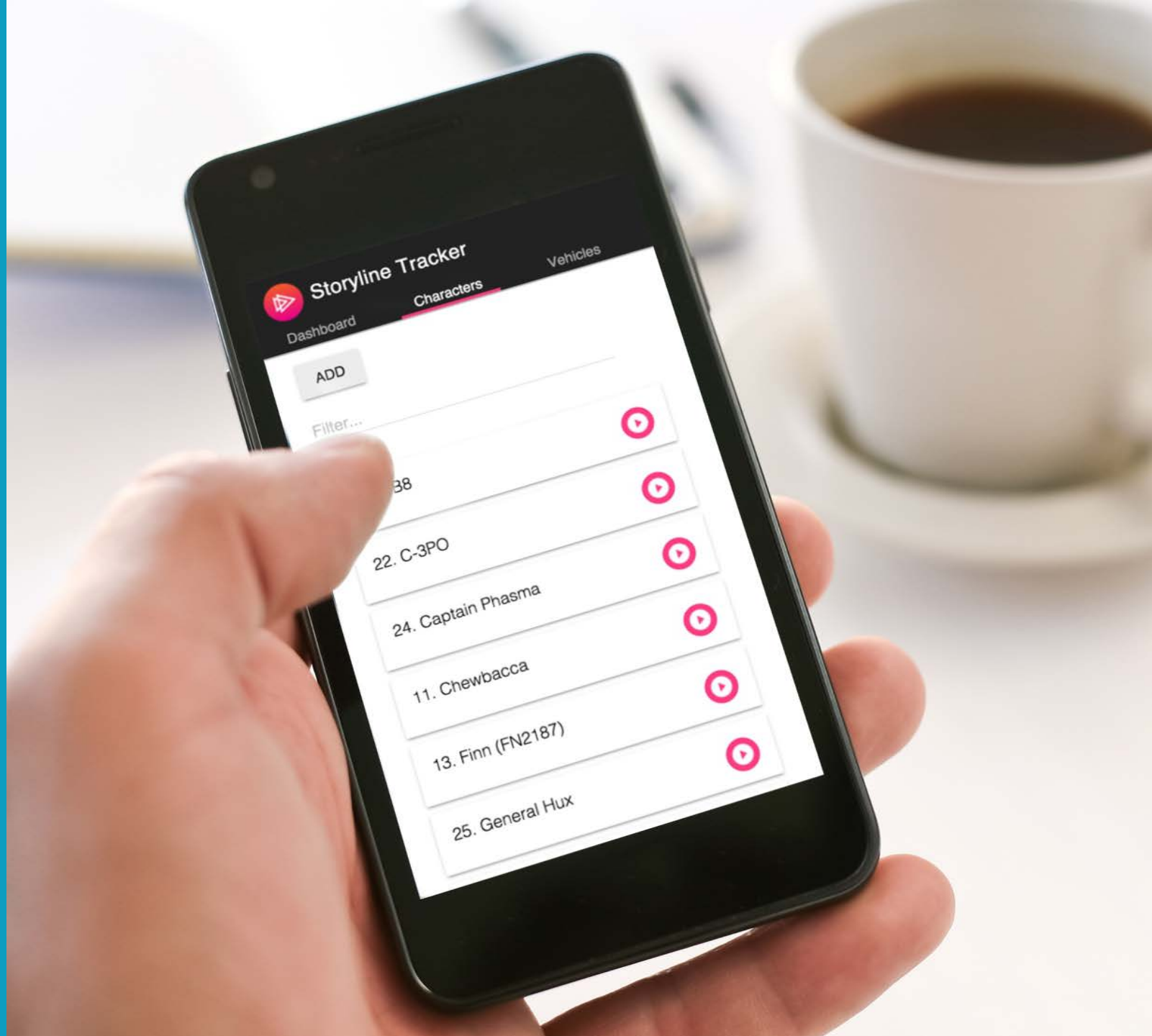
Provide to the root injector

**app.module.ts**

# Demo



## Putting It All Together



# Routing



**Defining Routes**

**Routing Modules**

**Route Parameters**

**Guards**

**Child Routes**

