Angular Routing

Routing Basics

- Setting up Basic Routing
- HTML5 vs Hash-based Urls

3-1-1

Basics Checklist: setting Up

- Define the base path
- Import RouterModule, RouterModule.forRoot([])
  - Use RouterModule.forRoot() for app routes
  - Use RouterModule.forChild() for features

3-1-1

Basics Checklist: Configuring Routes

- Path: Url segment(s) for the route
  - No leading slash
  - ‘’ - for default route
  - ‘**’ - for wildcard route
  - Casing matters
  -

Component

- Not string name, not enclosed in quotes
- Component must be imported

Order matters!

```
{ path: 'welcome', component: WelcomeComponent },
{ path: '', redirectTo: 'welcome', pathMatch: 'full' },
{ path: '**', component: PageNotFoundComponent }
```

Basics Checklist: Placing the Template

- Add the RouterOutlet directive
  - Identifies where to display the routed component's template
  - Primary RouterOutlet normally specified in the App component's template

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

3-1-3

Basics Checklist: Activating a Route

- Add the RouterLink directive as an attribute
  - Clickable element
  - Enclose in square brackets

```
<a [routerLink]="['/welcome']">Home</a>
```

- Bind to a link parameters array
  - First element is the Url segment
  - All other elements are route parameters or additional Url segments

Routing to Features

- Setting up for Feature Routing
- Route Path Naming Strategies
- Activating a Route with Code
- Accessing Feature Routes
- Defining a Routing Module

4-1-1

Routing to Features Checklist: Configuration

- Import RouterModule
  - Be sure to use RouterModule.forChild()
- Config the routes
- Order matters!

```
RouterModule.forChild([
      {
        path: '',
        component: ProductListComponent
      }
]);
```

4-1-2

Routing to Features Checklist: Naming Routes

- Use a common root path name for related feature routes
  - products
  - products:id
  - products:id/edit

Routing to Features Checklist: Activate with Code

- Import the Router
- Add a dependency on the Router service
  - As a constructor parameter
- Use the Router service **navigate** method
- Pass in a link parameters array
  - First element is the root Url segment
  - All other elements are route parameters or additional Url segments

```
this.router.navigate(['/welcome]);

this.router.navigateByUrl('/welcome');
```

4-1-4

Routing to Features Checklist: Routing Modules

- Separate out routes to their own routing module
- Keep route path order in mind

Route Parameters

- Required Parameters - Configure, Populate, Read
- Optional Parameters
- Query Parameters

5-1-1

Route Parameters Checklist: Required Parameters

- Pass needed data on a route
- Ex: DetailComponent requires an id

Configure

```
{ path: 'products/:id', component: ProductDetailComponent }
```

Populate

```
<a [routerLink]="['/products', product.id]">...</a>
```

```
.....
this.router.navigate(['/products', this.product.id]);
```

Read

```
this.route.snapshot.params['id'];
```

Or Observable

5-1-2

Route Parameters Checklist: Optional Parameters

- Pass optional or complex information to a route
- Ex: Search component passes search criteria to the ListComponent to filter data

Not Configured

```
{ path: 'products', component: ProductListComponent }
```

Populate

```
<a [routerLink]="['/products',{start: startDate, end: endDate}">...</a>
```

.....

```
this.router.navigate(['/products', this.product.id]);
```

Read

```
this.route.snapshot.params['start'];
```

Or Observable

Route Parameters Checklist: Query Parameters

- Pass optional or complex information to a route that is optionally retained across routes
- Ex: ListComponent passes its current user selections to the Detail component which passes them back

Not Configured

```
{ path: 'products', component: ProductListComponent }
```

Populate

```
<a [routerLink]="['/products']
[queryParams]="{filterBy: 'x', showImage: true}">...</a>
.....
this.router.navigate(['/products', queryParams: { filterBy: 'x',
showImage: true}]);
```

Read

```
this.route.snapshot.queryParams['filterBy'];
```

Or Observable

Prefetching Data Using Route Resolvers

- Prevents display of a partial page
- Reuses code
- Improves flow when an error occurs

- Providing Data with a Route
- Using a Route Resolver
- Building a Route Resolver Service
- Adding a Resolver to a Route Configuration
- Reading Resolver Data - Snapshot
- Reading Resolver Data - Observable

6-1-1

Route Resolvers Checklist: Building

- Create an Angular service
- Implement the Resolve<> interface

```
export class ProductResolver implements Resolve<IProduct> {}
```

```
resolve(route: ActivatedRouteSnapshot,
            state: RouterStateSnapshot): Observable<IProduct> { }
```

6-1-2

Route Resolvers Checklist: Configuring

- Configure using resolve
- Give each type of data a logical name
- Specify a reference to the route resolver

```
{       path: 'products:id',
        component: ProductDetailComponent,
        resolve: { product: ProductResolver }
},
```

6-1-3

Route Resolvers Checklist: Reading

- Read the data from the route
  - Snapshot
  - Data Observable

```
this.product = this.route.snapshot.data['product'];

….

this.route.data.subscribe(

     data => this.product = data['product'];

}
```

Child Routes

- Using Child Routes
- Configuring Child Routes
- Placing the Child View
- Activating Child Routes
- Obtaining Data for Child Routes
- Validating Across Child Routes

7-1-1

Child Routes: Configuring

- Add a children array to the parent route
- Define the child routes within that array
- Child paths extends the parent route

```
{        path: ':id/edit',
        component: ProductEditComponent,
        resolve: { product: ProductResolver },
        children: [
          { path: 'info', component: ProductEditInfoComponent },
          { path: 'tags', component: ProductEditTagsComponent }
    ]   }
```

Child Routes: Placing

- Place the child view by defining a RouterOutlet directive in the parent component's template

```
<div class="panel-body">
        <div class="wizard">
            <a [routerLink]="['info']">Info</a>
            <a [routerLink]="['tags']">Tags</a>
        </div>
        <router-outlet></router-outlet>
```

Child Routes: Activating

- Using an absolute path
  - Start with a slash
  - Define each Url segment
- Using a relative path
  - No staring slash
  - Only the child Url segment
-

```
<a [routerLink]="['/products', product.id, 'edit', 'info']">Info</a>

<a [routerLink]="['info']">Info</a>

this.router.navigate(['/products', this.product.id, 'edit', 'info']);

this.router.navigate(['info'], { relativeTo: this.route});
```

7-1-3

Child Routes: Obtaining Data

- Read the data from the route
  - Snapshot
  - Data Observable

```
this.product = this.route.snapshot.data['product'];

this.route.data.subscribe(

    data => this.product = data['product'];

);
```

Demo: Edit

7-1-3

Child Routes: Obtaining Data

- Read the data from the route
  - Snapshot
  - Data Observable

```
this.product = this.route.snapshot.data['product'];

this.route.data.subscribe(

    data => this.product = data['product'];

);
```

Demo: Edit

Grouping and Component-less Routes

- Better organization
- Share resolvers and guards
- Lazy loading

Demo: products

8-1-1

Checklist: Grouping

- Define routes as children of one parent route
- Specify relative paths

```
RouterModule.forChild([{
       { path: '', component: ProductListComponent },
        children: [{
       path: ':id/edit', component: ProductEditComponent,
       children: [{ path: '', redirectTo: 'info', pathMatch: 'full' },
         { path: 'info', component: ProductEditInfoComponent },
         { path: 'tags',  component: ProductEditTagsComponent }
       ]}]
   ])
```

Demo: Edit

Styling, Animating, and Watching Routes

- Styling the Selected Route
- Animating Route Transitions
- Watching Routing Events
- Reacting to Routing Events

9-1-1

Routes: Styling

- Style the active route using the routerLinkActive directive
- Style the correct element
- For an exact path match use routerLinkActiveOptions

```
<a routerLinkActive="active" [routerLink]="['info]">Basic Information</a>
<li routerLinkActive="active" [routerLinkActiveOptions]="{exact:true}">...</li>
```

Demo: Edit

9-1-2

Routes: Animating

- Use CSS animation
- Use Angular animation

Demo: Edit

9-1-3

Routes: Watching Events

- Enable tracing to watch routing events in the console
- Add enableTracing to the route configuration

```
 RouterModule.forRoot([
            { path: 'welcome', component: WelcomeComponent },
…], { enableTracing: true })
   ],
```

9-1-4

Routes: Reacting to Events

- Subscribe to the Router’s events observable
- Check the event type as needed (ex: useful for displaying loading spinner)

```
this.router.events.subscribe((routerEvent: Event) => {
    if (routerEvent instanceof NavigationStart) {
        ...
    }
});
```

Secondary Routes

- Using Secondary Routes
- Defining a Named RouterOutlet
- Configuring Secondary Routes
- Activating Secondary Routes

Secondary Routes: Named RouterOutlet

- Add another RouterOutlet within a template
- Set its name attribute to a unique name

```
<div class="container">

    <router-outlet></router-outlet>

    <router-outlet name="popup"></router-outlet>

</div>
```

Secondary Routes: Configuring

- Add the outlet property
- Set it to the name of the associated RouterOutlet

```
RouterModule.forChild([
            {
                path: 'messages',
                component: MessageComponent,
                outlet: 'popup'
            }
        ])
```

Secondary Routes: Activating

- Activate a secondary route using an object and setting its outlets property
  - Key: Outlet name
  - Value: Link parameters array

```
<a [routerLink]="[{ outlets: { popup:
['messages']}}]">Messages</a>


this.router.navigate([{ outlets: { popup: ['messages']} }]);
```

Secondary Routes: Clearing

- Clear a secondary route using an object and setting its outlets property
  - Key: Outlet name
  - Value: null

```
<a [routerLink]="[{ outlets: { popup: null }}]">Messages</a>


this.router.navigate([{ outlets: { popup: null} }]);
```

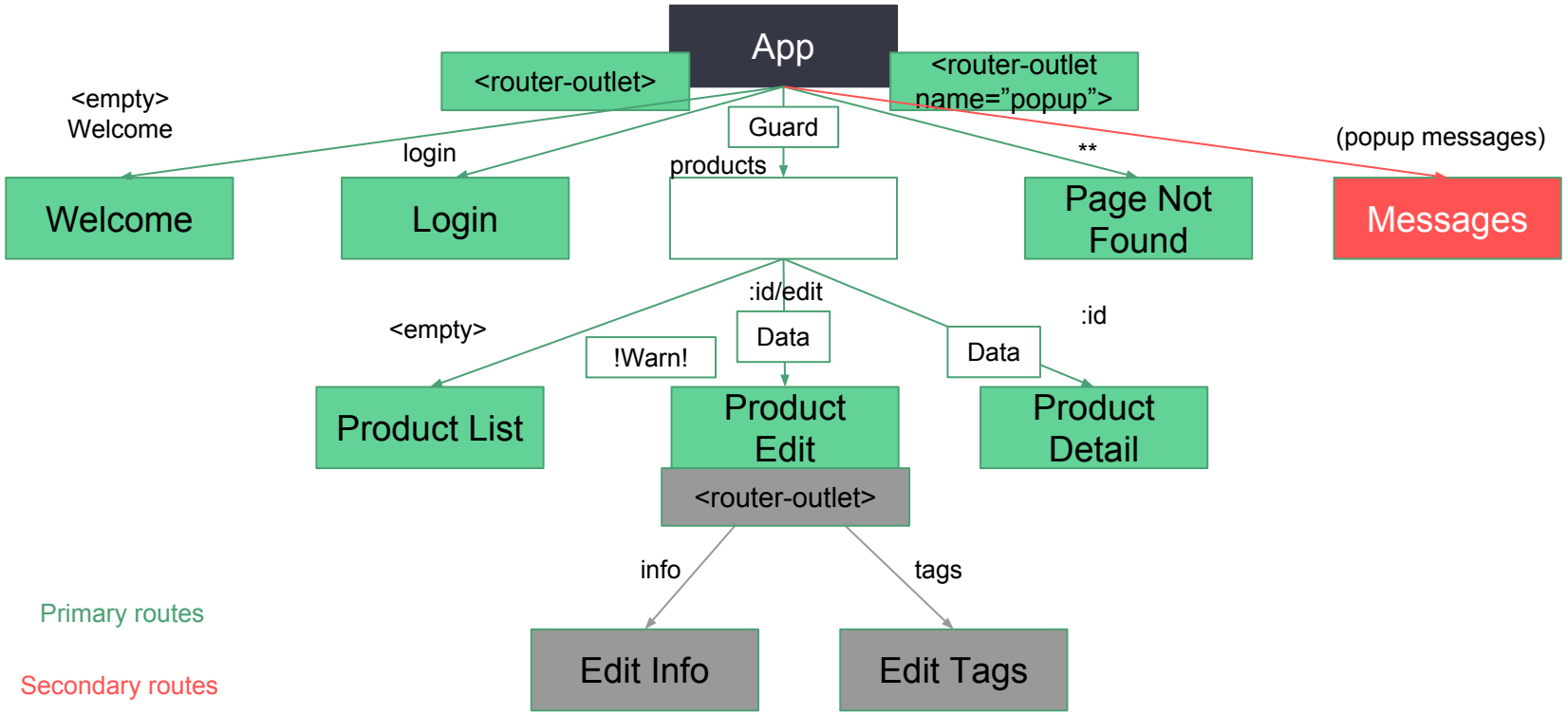
Demo: messages

Route Guards - Protecting Routes

- **canActivate** - Guard navigation to a route
- **canActivateChild** - Guard navigation to a child route
- **canDeactivate** - Guard navigation away from a route
- **canLoad** - Prevent asynchronous routing
- **resolve** - Prefetch data before activating a route

Route Guards

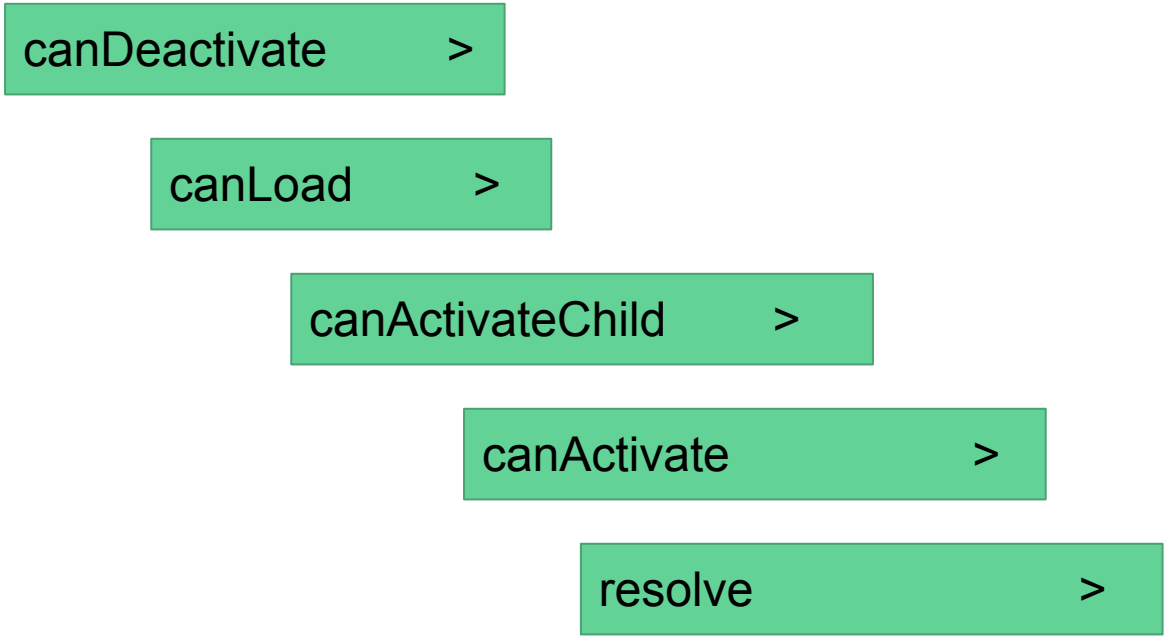
- Using Route Guards
- canActivate Guard
- Sharing Data with a Guard
- canActivateChild Guard
- canDeactivate Guard


App
<router-outlet>
<router-outlet name="popup">
<empty>
Welcome
login
Guard
products
**
(popup messages)
Welcome
Login
Page Not Found
Messages
:id/edit
<empty>
!Warn!
Data
Data
:id
Product List
Product Edit
Product Detail
<router-outlet>
info
tags
Edit Info
Edit Tags
Primary routes
Secondary routes
Child routes

Using Route Guards

- Limit access to a route
- Warn before leaving a route
- Retrieve data before accessing a route

Guard Processing



Building a Guard Service

auth-guard.service.ts

```
import { Injectable } from '@angular/core';
import { CanActivate } from '@angular/router';
@Injectable()
export class AuthGuard implements CanActivate {
    canActivate(): boolean {
    }
}
```

Registering a Guard

Angular / Feature Module

```
import { AuthGuard } from './auth-guard.service';

NgModule({

    providers: [ AuthGuard ]

})

export class UserModule { }
```

Guarding a Route

Angular / Feature Module

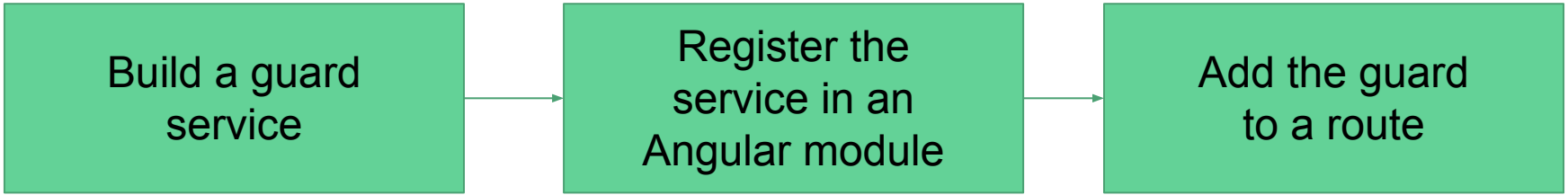
```
import { AuthGuard } from './auth-guard.service';

{
path: 'products',
canActivate: [ AuthGuard ],
data: { preload: true },
loadChildren:'app/products/product.module#ProductModule'
},
```

Sharing Guards

Angular / Feature Module

```
RouterModule.forChild([
      { path: '', component: ProductListComponent },
      { path: ':id', component: ProductDetailComponent},
      { path: ':id/edit', component: ProductEditComponent}
])
```

Guarding a Route



canActivate Guard

- Checks criteria before activating a route
- Used to:
  - Limit route access to specific users
  - Ensure prerequisites are met
- Called when the Url changes to the route

Sharing Data

Route parameters

```
canActivate(route: ActivatedRouteSnapshot,
            state: RouterStateSnapshot): boolean {
        console.log(route.params['id']);
    }
```

Service property

```
export class AuthService {
    currentUser: IUser;
    redirectUrl: string;
}
```

Demo: authguard, authservice, login component

canActivateChild Guard

- Checks criteria before activating a child route
- Used to:
  - Limit access to child routes
  - Ensure prerequisites for child routes are met
- Called when the Url changes to the child route

canDeactivate Guard

- Checks criteria before leaving a route
- Used to:
  - Check for unsaved changes
  - Confirm leaving an incomplete operation
- Called when the Url changes to a different route

Demo: product edit, product guard, product module

Route Guards: Building

- Build a service (AuthGuard)
- Implement the guard type (CanActivate)
- Create the associated method (canActivate())
- Register the service provider (providers: [AuthGuard,...])
- Add the guard to a route

```
{ path: ':id/edit',
  component: ProductEditComponent,
  canDeactivate: [ProductEditGuard] }
```

Demo: product edit, product guard, product module

Preparing for Lazy Loading

- Use a feature module
- Routes grouped under a single parent
- Not imported in another module

Lazy Loading

app-routing.module.ts

```
RouterModule.forRoot([

{

        Path: ‘products’,

        loadChildren: ‘app/products/product.module#ProductModule’

},

])
```

canLoad Guard

- Checks criteria before loading an asynchronous route
- we use canActivate for preloading lazy modules

Custom Preloading Strategy

- Build a preloading strategy service
- Register the service in an Angular module
- Set the preloading strategy routing option

Building a Preload Strategy Service

selective-strategy.service.ts

```
import { Injectable } from '@angular/core';

Import { PreloadingStrategy } from '@angular/router';

Import { Observable } from '@angular/router';

@Injectable()

Export class SelectiveStrategy implements PreloadingStrategy {

        preload(route: Rote, load: Function): Observable<any> {}

}
```

Registering and Enabling a Strategy

app-routing.module.ts

```
import { SelectiveStrategy } from './selective-strategy.service';

@NgModule({
  imports: [
    RouterModule.forRoot([
      { path: 'welcome', component: WelcomeComponent },
      { path: 'products',
        loadChildren: 'app/products/product.module#ProductModule'
      }], { preloadingStrategy: SelectiveStrategy })],
  providers: [ SelectiveStrategy ]})
```

Preload All Strategy

app-routing.module.ts

```
Import { RouterModule, PreloadAllModules } from '@angular/router';

RouterModule.forRoot([

{

        Path: 'products',

        loadChildren: 'app/products/product.module#ProductModule'

},

], { preloadingStrategy: PreloadAllModules })
```