Basic Angular Routing Setup

Lara Newsom

Software Engineer | Speaker | Angular GDE

@laranerdsom

Bootstrapping Routing with Modules

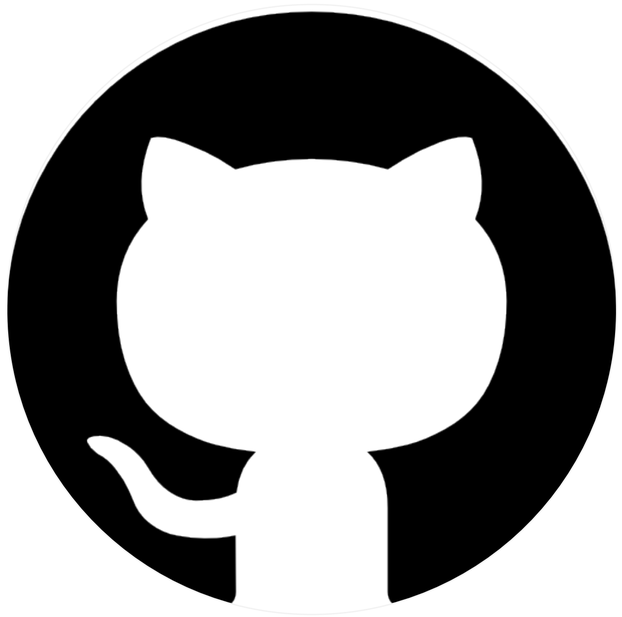
Typically have an app.module.ts file

The current default bootstrapping

The alternative is standalone component bootstrapping

What We Will Cover in This Module

- Bootstrap routing in an Angular application
- Add a route outlet to the DOM
- Declare routes
- Enable router tracing
- Hook into router lifecycle events



Repository for Module Three

github.com/lara-newsom/angular-routing-course

- module-three-start
- module-three-end

The primary entry point export for Angular routing is the RouterModule

Treeshaking

A term used to describe the process of removing unused javascript code from the bundle at compile time.

The RouterModule must be imported and bootstrapped for routing to work

Angular RouterModule Exports

Classes

Functions

Structures

Directives

Types

Placeholder slide: Show Angular Documentation

Importing Angular Router Module

Can be imported to any module

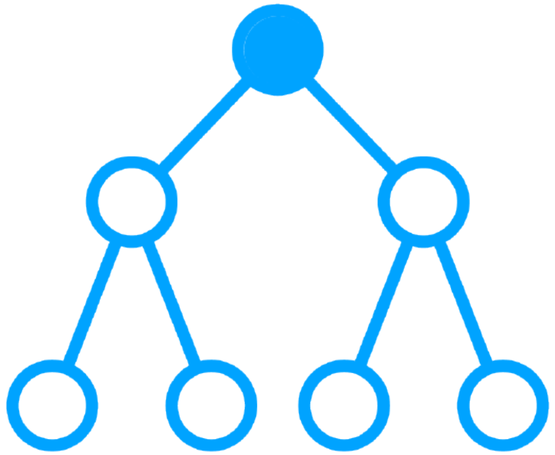
Can be imported many times

The first import must be at the root of the application

Bootstrapping

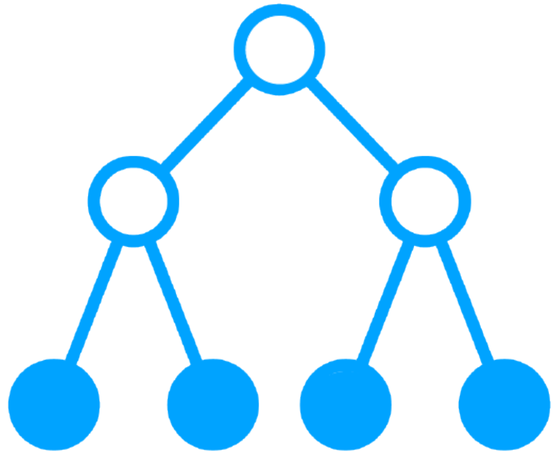
Running the code that initializes a piece of software, in this case the Angular Router

Angular RouterModule Methods



forRoot()

- Requires route declarations
- Instantiates RouterService
- Configures RouterModule
- Invoked only one time

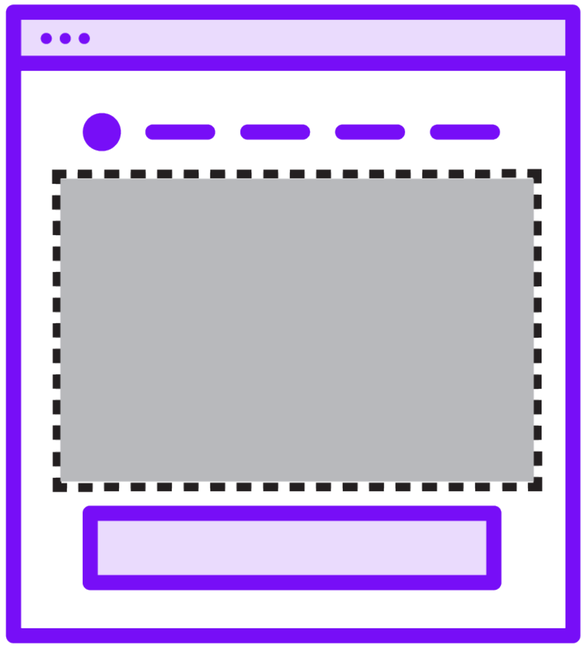


forChild()

- Requires route declarations
- Can be invoked many times

Bootstrap RouterModule where the application is bootstrapped

Routing in a Single Page Application



RouterOutlet

An Angular routing directive

A container for routed content

Placed in the flow of the DOM

Demo

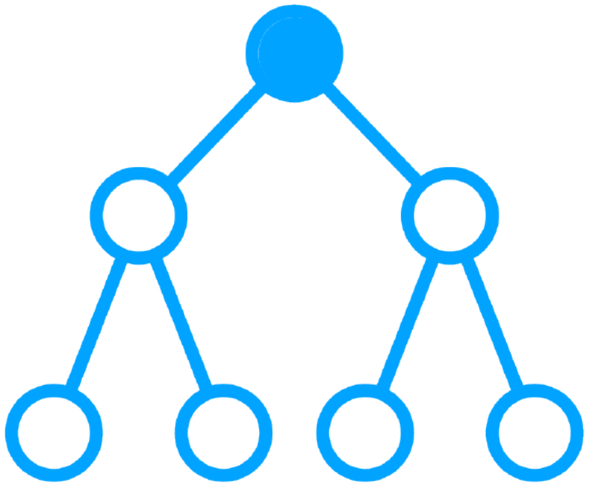
Bootstrap Angular RouterModule

Placeholder slide: Code Demonstration

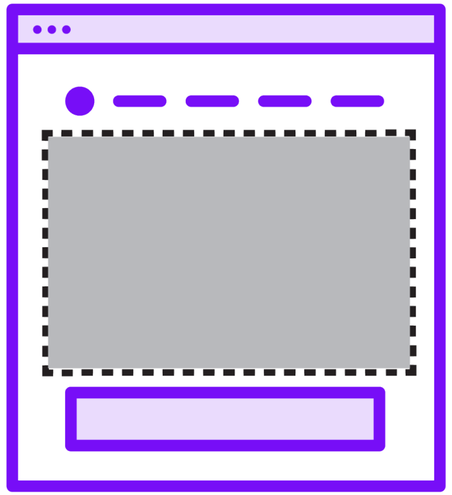
Importing RouterModule

Import in the root of the application

The root of the example app is app.modue.ts



RouterModule



RouterOutlet

Routes

Route Interface

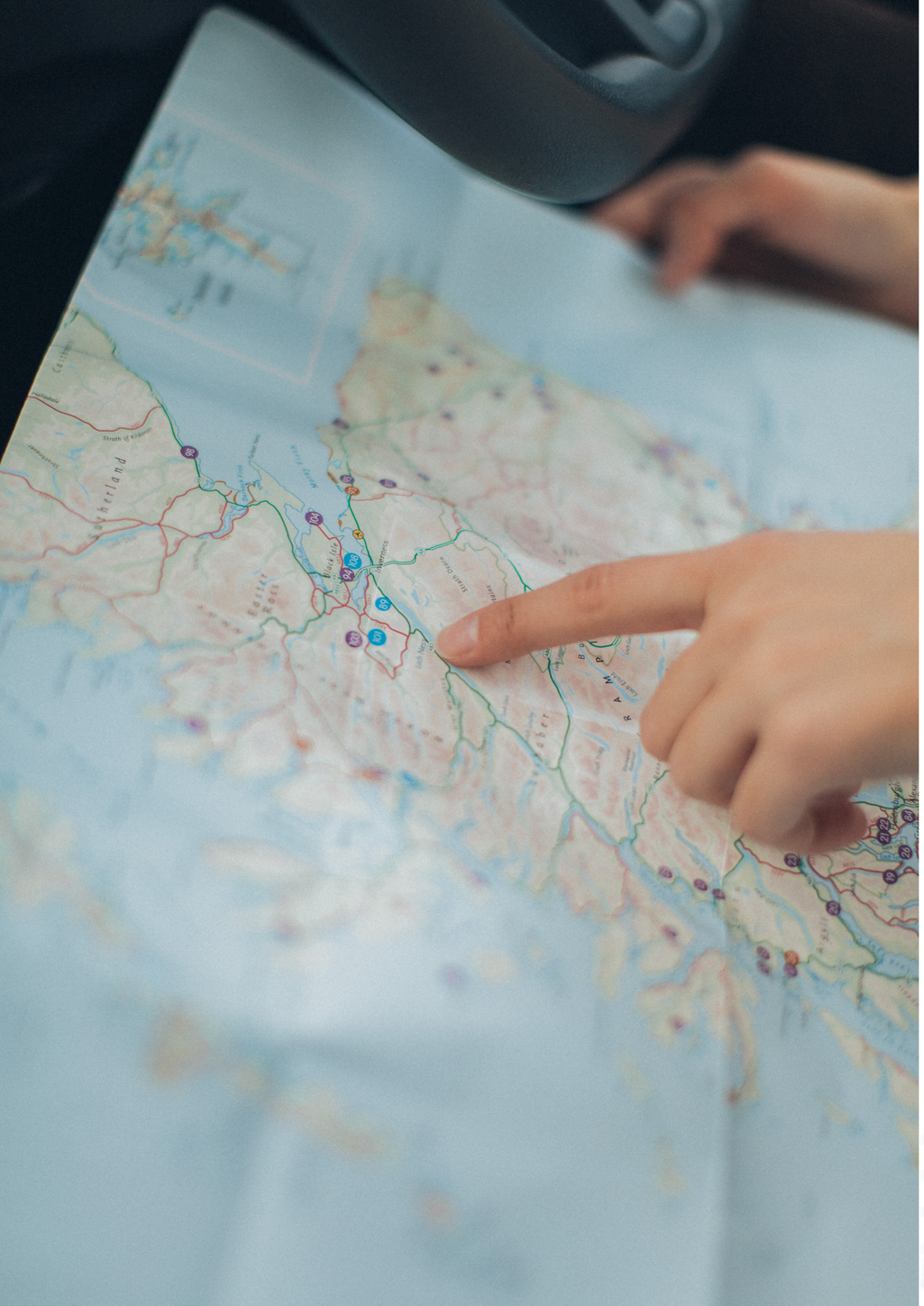
- Paths
- Route Guards
- Route Resolvers
- Child Routes
- Auxiliary Routes

Routes to Define

Home

Default

Wildcard



Properties to Define a Simple Route

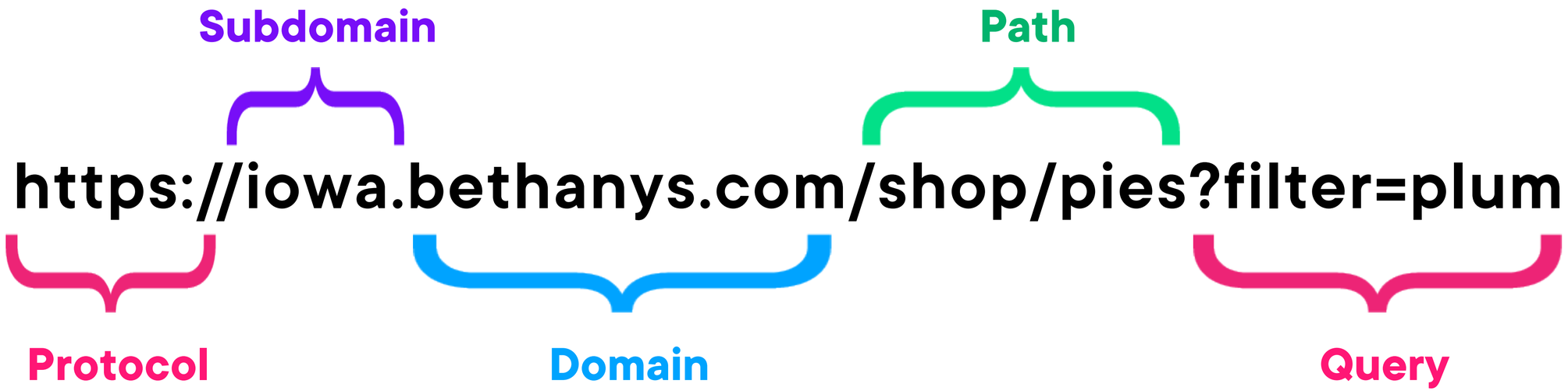
path

component

redirectTo

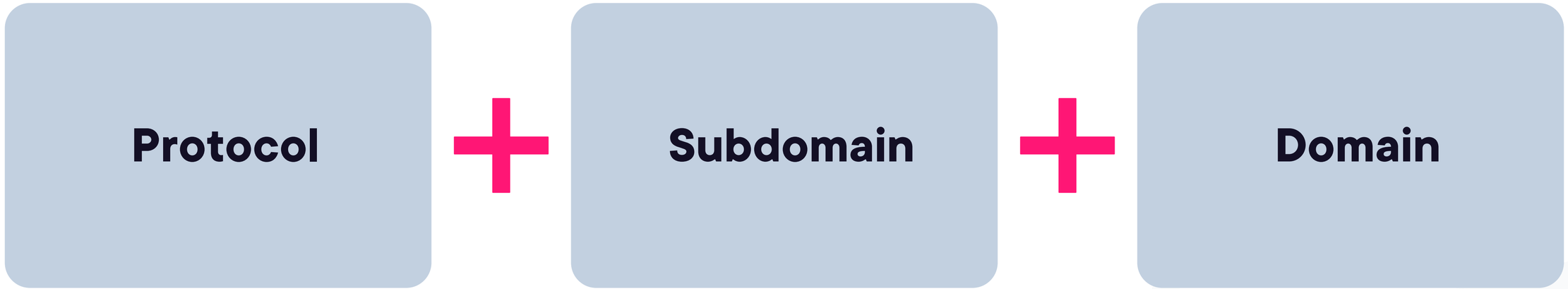
pathMethod

Parts of a URL



Angular Base URL

https://iowa.bethanys.com



Angular Path Segments

/shop /pies /fruit

Valid Path Declarations

```
path: 'shop'

path: ''

path: 'shop/pie'

path: '**'
```

First Match Wins Hierarchy

More specific route

Less specific route

Wildcard route

Angular Route PathMatch Strategies

prefix	VS	full
Default strategy **Partial segment matches** **Remaining segments will try to match a child route**		**All segments must match the path** **Remaining segments automatically fail to match** **Often used with default routes**

Full Config Match

All path segments have been consumed and the router has matched the route.

Additional Route Properties

component

redirectTo

Demo

Adding routes to the application

Placeholder slide for route demo

Declaring Child Routes with forChild( )

Takes an array of routes

Reuses the existing singleton router service instance

Moves route declaration closer to the code

The Angular Cli follows a similar pattern

Demo looking through generated code

Create child router modules to put route declaration closer to routed components

Route Array Concatenation with Modules

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

Route Array Concatenation with Modules

```
const routes: Routes = [
  {
    path: 'home/child-one',
    component: ChildOneComponent
  },{
    path: 'home/child-two',
    component: ChildTwoComponent
  },
];
```

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

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

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

Route Array Concatenation with Modules

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

Router Module Routes Array

```
[
  {
    path: 'home/child-one',
    component: ChildOneComponent
  },
  {
    path: 'home/child-two',
    component: ChildTwoComponent
  },
  {
    path: 'home',
    component: HomeComponent
  },
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full',
  }
];
```

We can follow the same route declaration strategy used in standalone applications in module based applications

Route Array Concatenation with Routes

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    RouterModule.forRoot(appRoutes),
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Route Array Concatenation with Routes

```
const appRoutes: Routes = [
 {
   path: ‘home’,
   component: HomeComponent,
   children: childRoutes,
 },{
   path: '',
   redirectTo: 'home',
   pathMatch: 'full',
 }
];
```

```
const childRoutes: Routes = [
 {
   path: 'child-one',
   component: ChildOneComponent
 },{
   path: 'child-two',
   component: ChildTwoComponent
 },
];
```

Route Array Concatenation with Routes

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    RouterModule.forRoot(appRoutes),
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

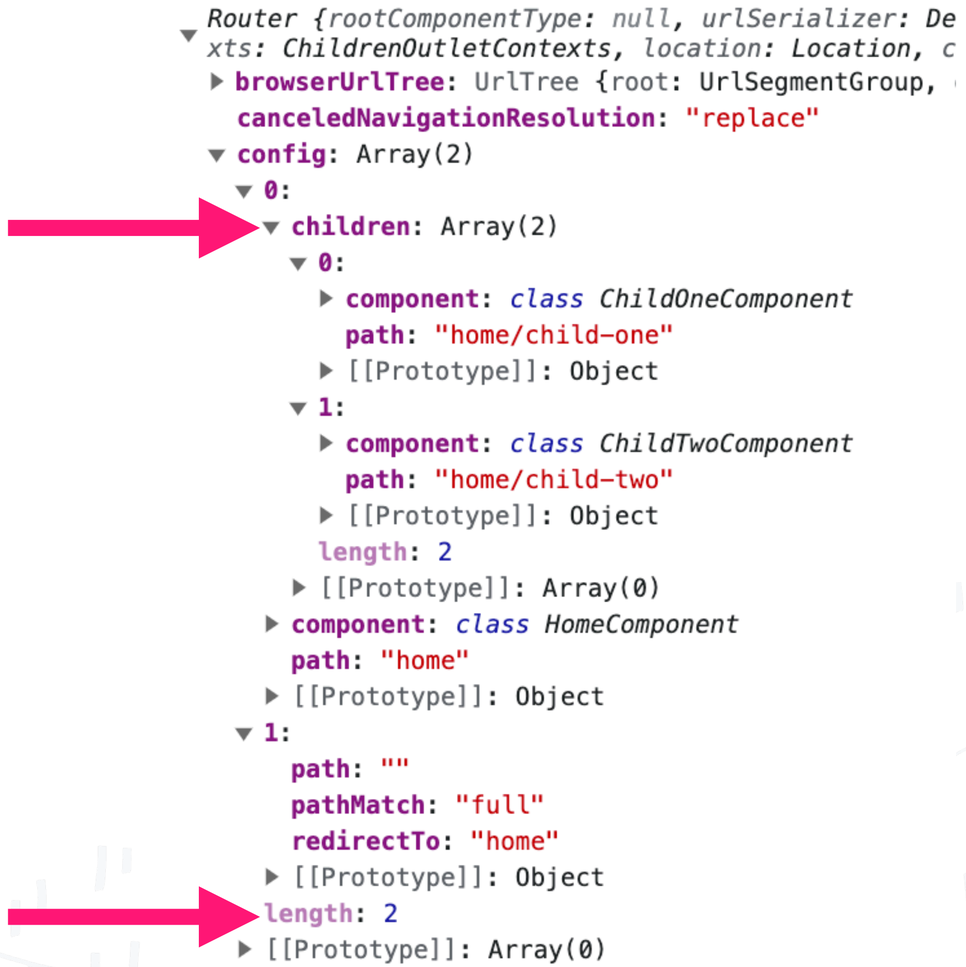
Router Module Routes Array

```
[{
    path: 'home',
    component: HomeComponent.
   children: [
     {
       path: 'home/child-one',
       component: ChildOneComponent
     },{
       path: 'home/child-two',
       component: ChildTwoComponent
     },
   ],
  },{
    path: '',
    redirectTo: 'home',
    pathMatch: 'full',
  }
];
```

Console Log Output for Router Service

Using forChild( )

Using children



The wildcard route is the last chance the router has to make a match.

Demo

Adding a wildcard route

Placeholder for adding a wildcard route and tracing

When tracing is enabled all internal navigation events are logged to the console.

Angular Navigation Lifecycle Events

- NavigationStart
- RouteConfigLoadStart
- RouteConfigLoadEnd
- RoutesRecognized
- GuardsCheckStart
- ChildActivationStart
- ActivationStart
- GuardsCheckEnd
- ResolveStart
- ResolveEnd
- ChildActivationEnd
- ActivationEnd
- NavigationEnd
- NavigationCancel
- NavigationError
- Scroll

Placeholder show tracing in the console

Demo

Comparing component lifecycle events to router lifecycle events

Demo

Hooking into router events

Hooking Into Router Events

Animations between views

Logging user flows

Run Custom logic

Summary

- Bootstrapped Angular routing
- Added a RouterOutlet
- Declared three different routes
- Enabled router tracing
- Hooked into router events with the RouterService
- Hooked into dispatched RouterOutlet events

Up Next:

Routing in Standalone Component Applications