Cheatography

# Angular 2 Router Cheat Sheet

by Nathan (Nathane2005) via cheatography.com/28056/cs/8478/

## Getting started

```
In index.html
<base href="/ ">
```

## Router Imports

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

## Sample Router

```
RouterModule.forRoot([
          { path: 'hero/ :id',
component: HeroDe tai lCo -
mponent },
          { path: 'crisi s-c -
enter', component: Crisis Lis -
tCo mponent },
          {
               path: 'heroes',
               com ponent:
HeroLi stC omp onent,
               data: {
                   title:
'Heroes List'
               }
          },
          { path: '',
component: HomeCo mponent },
          { path: '**',
component: PageNo tFo und Com -
ponent }
       ])
   ]
```

There is an array of routes defined for the specific section of the application, depending on the embedded nature of the application there will be more child routes defined

## General

1. The order of the routes matter
2. There is usually a default route with an empty path

## Router Outlet

```
<!-- Routed views go here -->
<ro ute r-o utl et> </r out -
er- out let>
```

## Router Link

| | |
|---|---|
| RouterLink | The directive for binding a clickable HTML element to a route. Clicking an anchor tag with a routerLink directive 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. |

## Router Link (cont)

| | |
|---|---|
| Router State | The current state of the router including a tree of the currently activated routes in our application along convenience methods for traversing the route tree. |
| Router | Displays the application component for the active URL. Manages navigation from one component to the next. |
| Router Module | 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. |

Cheatography

# Angular 2 Router Cheat Sheet

by Nathan (Nathane2005) via cheatography.com/28056/cs/8478/

## Router Controls

| | |
|---|---|
| CanActivate | Checking route access |
| CanActivateChild | Checking child route access |
| CanDeactivate | Ask permission to discard unsaved changes |
| CanLoad | Check before loading feature module assets |

## Child Routes Definition

```
RouterModule.forChild([
            { path: 'heroes',
component: HeroLi stC omp onent
},
            { path: 'hero/ :id',
component: HeroDe tai lCo -
mponent }
        ])
```

In the parent definition include the loadChildren attributes

## Lazy Loading Of Routes

```
export function
loadCompensationModule() {
    return requir e(' es6 -pr -
omi se!../ cf/ she ll/ she -
ll.m od ule ')( 'Sh ell Mod -
ule');
}
export const routes: Routes = [
   { path: '', redire ctTo:
'home', pathMatch: 'full'},
   { path: 'cf', loadCh ildren:
loadCo mpe nsa tio nMo dule},
   { path: 'home', component:
MainCo mpo nent, canAct ivate:
[AuthG uard]},
```

## Lazy Loading Of Routes (cont)

```
>  { path:'**', redirectTo: '' }
];
export const appRoutingProviders: any[] = [
];
export const routing: ModuleWithProviders
= RouterModule.forRoot(routes, { useHash:
false});
```

This is a sample. The children definition would be defined in the children array namely:

```
export const routes: Routes = [
{
path: '', component: ShellComponent,
children: [
{path: 'home', component: ShellComp-
onent},
{path: 'one', loadChildren: loadOneModule},
{path: 'two', loadChildren: loadTwoModule},
{path: 'three', loadChildren: loadThreeMod-
ule},
{path: 'four', loadChildren: loadFourModule}
]
},
{path: '**', redirectTo: 'home'}
];
```

## Route Parameters - Mandatory

| | |
|---|---|
| Route Parameter | xxx/:{type} |

To Specify the route information
1. Construct the URL
2. Pass in this.router.navigate(['/hero', hero.id]);

## Route Paramters - Optional

```
<a [routerLink]="['/crisis-
center', { foo: 'foo' }]">Crisis
Center</a>
```

## Location Stategy

| | |
|---|---|
| PathLocation-Strategy | the default "HTML 5 pushState" style. |
| HashLocation-Strategy | the "hash URL" style. |

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

## CanActivate Guard

```
import { Injectable } from
'@angular/core';
import {
    Can Act ivate, Router,
    Act iva ted Rou teS nap -
shot,
    Rou ter Sta teS napshot
} from '@angu lar /ro uter';
import { AuthSe rvice } from
'./aut h.s erv ice';
@Injec table()
export class AuthGuard
implements CanAct ivate {
    con str uct or( private
authSe rvice: AuthSe rvice,
private router: Router) {}
    can Act iva te( route:
Activa ted Rou teS nap shot,
state: Router Sta teS nap shot):
boolean {
        let url: string =
state.url;
         return this.c hec kLo -
gin (url);
    }
    che ckL ogi n(url: string):
boolean {
        if (this.a ut hSe rvi -
ce.i sL ogg edIn) { return true;
}
        // Store the attempted
URL for redire cting
         thi s.a uth Ser vic -
e.r edi rectUrl = url;
```

## CanActivate Guard (cont)

```
>    // Navigate to the login page with extras
   this.router.navigate(['/login']);
   return false;
 }
}
```

To reference the guard:

canActivate: [AuthGuard],

## Route Animations

```
Import
import { Component, OnInit,
HostBi nding,
                  tri gger,
transi tion, animate,
                  style, state }
from '@angu lar /core';
animat ions: [
        tri gge r(' rou teA -
nim ation', [
            sta te('*',
                style({
                    opa city: 1,
                    tra nsform:
'trans lat eX(0)'
                })
           ),
            tra nsi tio n('void
=> *', [
                style({
                    opa city: 0,
                    tra nsform:
'trans lat eX( -100%)'
                }),
                ani mat e('0.2s
ease-in')
           ]),
            tra nsi tion('* =>
void', [
```

## Route Animations (cont)

```
>        animate('0.5s ease-out', style({
          opacity: 0,
          transform: 'translateY(100%)'
        }))
      ])
    ])
  ]
  @HostBinding('@routeAnimation') get
routeAnimation() {
    return true;
  }
```

the host binding is for linking to the transition effect.