Cheat Sheet – Routing

Learning Resources

Official Guide (constantly updated):
https://angular.io/docs/ts/latest/guide/router.html
Blog Post by Angular 2 Team Member:
http://victorsavkin.com/post/145672529346/angular-router

Route Setup

Setting up Routes is a two-step process:

1. Create the Routes (e.g. in a separate constant / file)
2. Provide the Routes to the Application

Routes are created like this:

```
const ROUTES: Routes = [
        { path: 'my-path', component: MyComponent }
];
```

Regarding additional configuration etc., consult the above links or the videos in the course.
Routes are then provided via the RouterModule which has to be imported to the routes file.

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

const ROUTES: Routes = [ … ];

export const routing = RouterModule.forRoot(ROUTES);
```

Finally, the exported constant needs to be added to the imports[] in your AppModule:

```
import { routing } from 'path/to/routing/file';

@NgModule({
   declarations: […],
   imports: […, routing],
   …
})
…
```

Of course, make sure to add the right imports to the file.

Router Outlet & Links

RouterOutlets mark the places where Angular 2 should render components loaded through routing. In order to use `<router-outlet></router-outlet>`, you have to add the ROUTER_DIRECTIVES to your Component.

RouterLinks are then used to provide links for the User to navigate through your Application. They are placed on Anchor tags like this:

```
<a [routerLink]="['/my-path']">Link</a>
```

You may use absolute ('/path') and relative ('path') paths in a RouterLink.

Imperative Routing

You can also navigate to a route imperatively, which means: In your code. For that, you need to inject the Router. Thereafter, you may navigate like this:

```
this.router.navigate(['my-path']);
```

Route Parameters

You can also pass route parameters by setting the Routes up like this:

```
const ROUTES: Routes = [
    { path: 'my-path/:param', component: MyComponent }
];
```

A parameter could then be passed like this:

```
<a [routerLink]="['/my-path', 'param value']">Link</a>
```

In order to retrieve the parameter in the target component, you can either access the snapshot of the ActivatedRoute or subscribe to the params Observable on the ActivatedRoute. The latter approach guarantees that the parameter will be updated if it changes whilst the component isn't recreated. You have to `unsubscribe()` from the Observable in the ngOnDestroy method/ lifecycle hook when using this approach.

Guards

Guards are useful helpers which allow you to control access to and from a Route / Component. The CanActivate Guard does the first, the CanDeactivate Guard the latter.

Guards are added as classes and then have to be provided in the providers[] array in the AppModule method (like Services). Thereafter, you may attach them to routes like this:

```
const ROUTES: Routes = [
    { path: 'my-path', component: MyComponent, canActivate:
    [MyGuard], canDeactivate: [MyOtherGuard] }
];
```