Routing

Setup for router: add import to the top of the file, add a constant for you route ex: const appRoutes: Routes = []; inside of the Routes the notation is path: ‘’, component than in the imports section add RouterModuel.forRoot(nameOfRoutesConst) and in the html add router-outlet to where you want the pages to load.

When routing to a new page make sure to have a / in front of it

Absolute paths: /name

Relative path: name

Relative path will append to the end of the link.

../ go back one path and go to the name

routerLinkActive directive will let the active tab on a router link similar to class active in html. This can be added to the wrapping or the link itself. If you are using bootstrap you have to add it to the wrapping to get they styling added with it.

routnerLinkActiveOptions will tell angular on the home page to only add the active tag if a certain criteria is met. Ex: [routerLinkActiveOptions]=”{exact: true}” will only use router link active if the string for the link exactly matches instead of making home active the entire time.

Add a link programmatically: You can link a button or whatever you want to link in the front end to the ts file of that component in the component you have to import Router from @angular/router and initialize it in the constructor ex: private router: Router. Now in the method you created to link or do whatever calculations you need you would call this initialized variable in the constructor and tell it to navigate to wherever you want ex: this.router.naviate([‘/anotherTab’]);

ActivatedRoute is used to tell what the current route is and what it is called. Ex: private route: ActivatedRoute.

:id in path allows you to dynamically create pages based upon variables passed.

Params vs snapshot: params is a observable are objects that work with asynchronous task. If you know that none of your information is going to change use snapshot if your information can change use params with subscribe.

If you have your own observables you have to unsubscribe from them on your own by implementing ondestroy and adding a subscription variable to store the data.

Query params allows you to add variables to either allow you to enter certain areas. Ex: [queryParams]=”{allowEdit: ‘1’}”

To add a fragment: fragment=”whateverYouWantInURL” in the html file

You can setup the query params and the fragment in typescript by doing the router method programmatically and adding to it: , {queryParams: {parameter: ‘value’}, fragment: ‘value’}

Adding a plus in front of something will make sure that it is an number when it comes out instead of a string. Ex: +variableName

To add nested routes add to the original route a variable called children and put your children routes in an array inside of the route and in the class put a router outlet where you would want to access the data.

To configure navigation use queryParamsHandling. Preserve will keep no matter what even if you add new ones and merge will merge new and old parameters. By default it will drop all of the parameters. Ex: this.router.navigate([‘navigationLocation’], {relativeTo: this.route, queryParamsHandling: ‘merge’});

Relative to will allow you to create a relative path appending whatever name you have added in the navigate to the end of the path. Ex: this.router.naviate([‘navigationLocation’], {relativeTo: this.route});

Redirecting: If you don’t want to load a specific component you can add the redirectTo function in the module.ts file. Ex: { path: ‘something’, redirectTo: ‘/not-found’ } this will redirect the page something to the not-found f page instead. To catch all routes that do not exist use \*\* and make sure it is the final path.

Wildcard route: \*\* means catch all

To redirect to a blank path use pathMatch: ‘full’ which basically means that the path has to match in every character allowing you to match to an empty path.

If you have more than two or three routes add a new file called app-routing.module.ts. In the class you have to have @NgModule at the top. Add the routes array to the new file and import all of the components. In NgModule add RouterModule.forRoot(appRoutes) to imports and add exports array with RouterModule in it. Than in the old file delete the routes array, remove the routes, and routerModule from the imports and import AppRoutingModule.

Route guards: functionality which is executed once a route is loaded or when you leave a route.

In the route guard you have to implement the CanActivate method which in turn needs an ActivatedRouteSnapshot and RouterStateSnapshot this method should return an observable that is a Boolean or a Promise that is a Boolean or just a boolean. Then inject the Router method so that you can send them to another url if needed. To use the guard in the module.ts file add canActivate[AuthGaurd] to the paths you want to be guarded. Remember to add the service to the module.ts file so that it is accessible by the app.

To guard children add canActivateChild. You have to implement it just like canActivate and has the same signature as canActivate.

To control navigation with canDeactivate make sure you create an interface which uses the same return type as canActivate than in the class itself implement the interface you just created in the same file and have it take in the interface type you just created, activated route snapshot, router state snapshot, and another router state snapshot that will be the next state and have it return the same type as canActivate.

You can pass static data using data in the path. Ex: data: {‘Page not fount!’} and you have to inject your active route into the class.

In the resolve when you go to access the data it is the same thing you named it in the path on the module.ts.

To configure urls to use hash. This allows the url to be parsed before and after the hash symbol. Do not use this unless needed.