Building Application using Angular

Routing

Routing

- Routing is a kind of process in which we can divide the UI of a web application with the help of a URL. With the help of Routing, we can build modern single-page applications in which the entire page is loaded only once, but with multiple views with the same page.
- In Single Page Applications, once the web page has been loaded, the web application will interact with the users dynamically without loading the entire page every time when a user requests a specified data element.
- using the Routing we can navigate from one component to another component within the same application. When an application has been loaded for the first time, the homepage or index page will be displayed first on the view, then based on the request from the user the page will be navigated from one to another.

Route and the path

- The Route is like an object in which the information about the component is to be mapped with a specific path.
- An example of routing with representing with the ID parameter

```
{path: ‘/:id’, component: ComponentName}
```

Routing

- Create new application with Routing Option in VS Code
- app-routing.module.ts will be created in our project

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
//here we need to define all the routes of our application
const routes= [{ path: '', redirectTo: 'home', },
 { path: 'home', component: HomeComponent },
];

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

forRoot creates a module that contains all the directives, the given routes, and the router service itself.

Router-outlet

- The router-outlet is a directive that's available from the @angular/router package and is used by the router to mark where in a template, a matched component should be inserted.
-

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

Example

- https://stackblitz.com/edit/routingsample

Life Cycle Hooks

- Every component in Angular has its own lifecycle events that occurs as the component gets created, renders, changes it's property values or gets destroyed. Angular invokes certain set of methods or we call them hooks, that gets executed as soon as those lifecycle events gets fired.
- Lifecycle hooks are wrapped in certain interfaces which are included in the angular core **'@angular/core'** library.

Life cycle events

- One thing to note that each of these interfaces includes one method whose name is same as of the interface name but it is just prefixed by **"ng".** For example **OnInit** interface has one method **ngOnInit().** The following lifecycle hooks are exposed by Angular corresponding to different lifecycle events.
  - ngOnChanges()
  - ngOnInit()
  - ngDoCheck()
  - ngAfterContentInit()
  - ngAfterContentChecked()
  - ngAfterViewInit()
  - ngAfterViewChecked()
  - ngOnDestroy()

Usefull link

- Dependency Injection
  - https://angular.io/guide/dependency-injection
  - https://medium.com/razroo/dependency-injection-in-angular-c265043883f8
- Routing
  - https://angular.io/guide/router
  - https://medium.com/@jinalshah999/all-about-routing-in-angular-application-23f469a33f89