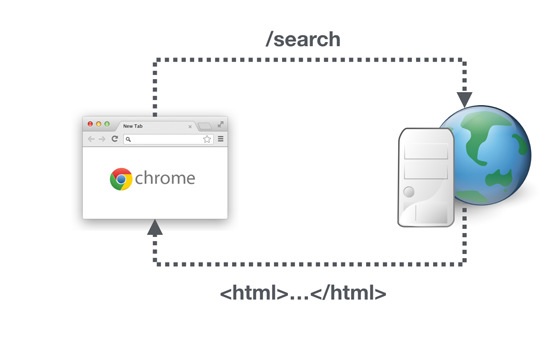
Route or Component Navigation

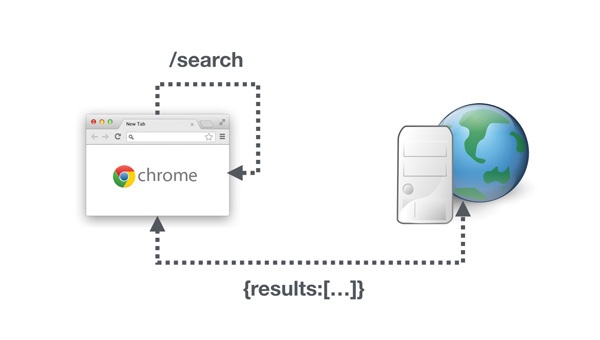
In the basic concept, routing is a process through which we can transfer packets or data from one host to another through a network. In web application development, app routing is mainly used to map any specific URL with any specific objects. It is mainly used to access any specific UI in the application. In the past, the basic web application always used a different approach for routing in comparison to the modern one. In the older approach, we normally use the server-side routing approach. In this approach, the server normally generates the HTML related markup and sending to the client or browser. Users may fill some data into the form and again those data send to the server as a response. But, nowadays JavaScript framework or client-side framework becomes much more advanced compared previously. So, for generating HTML markup data, currently, we are not dependable on the server only. We can build that on the client-side and send it back to the browser. We just need to communicate with the server for the data part only. This type of routing concept is normally known as Client-Side routing which we will discuss in brief in this article.

## **Introduction to Client-Side Routing**

Each and every web application always contains a URL that is normally shown in the address bar of the browser. Basically, this URL defines the current state of the application. As we known state means “all the stored information, at a given instant in time, to which program or application can access”. So in simple words, the state of an application is the current value of all the variables in the application. The Address in the URL can’t store a large volume of information, but using the URL we can bring the same state in the browser for the multiple users. As discussed in the previous section, in the traditional applications we normally user Server-Side Routing. In this process, the browser raises a request to the server to return some HTML markup data which it will display. The below image clearly has shown the process.



But we want to implement Client-Side Routing in our applications. In this process, when the URL changes in the browser address bar, our local application which is running in the browser (the client) needs to handle the changes. We do not want to send the request to the server. When we will navigate to a new site, the server returns the HTML, JavaScript, and CSS needed to render that application or page. After that, all the further changes to the URL will be handled locally by the client application. For retrieving the information related to the next UI which need to be shown, the client application will call one or more API request in the client application. At that time, a single HTML page will be returned from the server, then onwards all further modification of the page will be handled by the client and that’s why it is called a Single Page Application (SPA).



The main advantage of the SPA based applications are,

1. It is much faster, since instead of making a time-consuming request to a long-distance server every time the URL changes, the client application updates the page much faster.
2. Less bandwidth or network traffic. We didn’t need to send a large HTML page for every URL change request. Instead of that we just need to call a smaller API which returns just enough data to render the change in the page.
3. It is much more convenient. Since a single developer now can build most of the functionality of a site instead of splitting the effort between a front-end and a back-end or server-side developer.

Using the different modules in the Angular Framework, we can implement our web application as a SPA. For this purpose, Angular always followed a basic concept called Component Router.

## **Route Definition Objects**

So, before going to discuss the Angular Router Modules, we first need to discuss route definition objects. For defining the router definition objects, we need to define the route type which is basically is an array of routes that defines the routing for the client application. Within this array, we can mention setup related to the expected paths, the related components which we want to use or open using the route. Each route can be defined with different attributes. Some of the common and most used attributes are,

* path – This attribute is used to mention the URL which will be shown in the browser when the application is redirecting on the specific route.
* component – This attribute is used to mention the name of the component which will be rendered when the application is on the specific route
* redirectTo – This attribute is an optional attribute. It needs to mention when we need to redirect any specific route if our main route is missing. The value of this attribute will be either component name or redirect attribute defined in the route.
* pathMatch – It is also an optional attribute property that defaults to 'prefix'. It determines whether to match full URLs or just the beginning. When defining a route with empty path string set pathMatch to 'full', otherwise it will match all paths.
* children – This attribute contains an array of route definitions objects representing the child routes of this route.

To use Routes, we need to define an array of route configurations as below,

1. **const** routes: Routes = [
2. { path: 'component-one', component: ComponentOne },
3. { path: 'component-two', component: ComponentTwo }
4. ];

## **Concept of Router Modules**

To implement routes in any web application, we need to import RouterModule in our application with the help of NgModules. RouterModule.forRoot takes the Routes array as an argument and returns a configured router module. The following sample shows how we import this module in an app.routes.ts file.

1. **import** { RouterModule, Routes } from '@angular/router';
3. **const** routes: Routes = [
4. { path: 'component-one', component: ComponentOne },
5. { path: 'component-two', component: ComponentTwo }
6. ];
8. **export** **const** routing = RouterModule.forRoot(routes);

After defining the router configuration, we need to import the router configuration into our app module files as below,

1. **import** { routing } from './app.routes';
3. @NgModule({
4. imports: [ BrowserModule,routing],
5. declarations: [AppComponent,ComponentOne,ComponentTwo],
6. bootstrap: [ AppComponent ]
7. })
9. **export** **class** AppModule {
10. }

When our application starts, it navigates to the empty route by default. So, we can configure the router in such a way that the router will be redirected to the mentioned route by default,

1. **export** **const** routes: Routes = [
2. { path: '', redirectTo: 'component-one', pathMatch: 'full' },
3. { path: 'component-one', component: ComponentOne },
4. { path: 'component-two', component: ComponentTwo }
5. ];

The pathMatch property, which is mainly required for redirects, instructs the router how it should match the URL provided in order to redirect to the specified route. Since pathMatch: full is mentioned, the router will be redirected to component-one when the entire URL matches ('').

In this way, when the application starts, the router system automatically loads the component-one component at the beginning as the default route.

## **Router Link**

We can also control the navigation by using the ruterLink directive in the HTML template like as below –

1. <nav **class**="navbar navbar-light bg-faded">
2. <a **class**="navbar-brand" [routerLink]="['component-one']">Component 1</a>
3. <ul **class**="nav navbar-nav">
4. <li **class**="nav-item active">
5. <a **class**="nav-link" [routerLink]="['']">Home</a>
6. </li>
7. <li **class**="nav-item">
8. <a **class**="nav-link" [routerLink]="['component-two']">Component 2</a>
9. </li>
10. </ul>
11. </nav>

Alternatively, we can navigate to a route by calling the navigate function on the router:

1. **this**.router.navigate(['/component-one']);

An important feature of any navigation component is giving the user feedback about which link item they are currently viewing. Another way to describe this is by giving the user feedback about which route is currently active. To help in adding and removing classes depending on the currently active route Angular provides another directive called routerLinkActive. A routerLinkActive directive is associated with a route through a routerLink directive. It takes as input an array of classes which it will add to the element it’s attached to if it’s route is currently active, like so,

1. <a **class**="nav-link"
2. [routerLink]="['home']"
3. [routerLinkActive]="['active']">Home</a>

## **Add Route Component**

In spite of defining each route component separately, we can use RouterOutlet directives which basically acts as a component placeholder in our application. Angular Framework dynamically injects the components for the active route within the <router-outlet></router-outlet> element.

1. <router-outlet></router-outlet>

In the above example, the component related to the route specified will be activated after the <router-outlet></router-outlet> element when the router link is clicked.

## **Demo 1 - Route Demo**

For the demonstration of the route example in Angular, we first need to develop 3 components that are independent and can be used as a route component. For that purpose, we first create the below components – MobileComponent, TVComponent and ComputerComponent, and HomeComponent.

**app.component.tv.ts**

1. **import** { Component, OnInit } from '@angular/core';
3. @Component({
4. selector: 'app-tv',
5. templateUrl: 'app.component.tv.html'
6. })
8. **export** **class** TvComponent **implements** OnInit {
10. **private** data: Array<any> = [];
12. constructor() {
13. **this**.data = [{ name: 'LED TV 20"', company: 'Samsung', quantity: '10', price: '11000.00' },
14. { name: 'LED TV 24"', company: 'Samsung', quantity: '50', price: '15000.00' },
15. { name: 'LED TV 32"', company: 'LG', quantity: '10', price: '32000.00' },
16. { name: 'LED TV 48"', company: 'SONY', quantity: '25', price: '28000.00' }];
17. }
19. ngOnInit(): **void** {
20. }
21. }

**app.component.tv.html**

1. <div **class**="panel-body">
2. <table **class**="table table-striped table-bordered">
3. <thead>
4. <tr>
5. <th>Name</th>
6. <th>Company</th>
7. <th **class**="text-right">Quantity</th>
8. <th **class**="text-right">Price</th>
9. </tr>
10. </thead>
11. <tbody>
12. <tr \*ngFor="let item of data">
13. <td>{{item.name}}</td>
14. <td>{{item.company}}</td>
15. <td **class**="text-right">{{item.quantity}}</td>
16. <td **class**="text-right">{{item.price | currency}}</td>
17. </tr>
18. </tbody>
19. </table>
20. </div>

**app.component.mobile.ts**

1. **import** { Component, OnInit } from '@angular/core';
3. @Component({
4. selector: 'app-mobile',
5. templateUrl: 'app.component.mobile.html'
6. })
8. **export** **class** MobileComponent **implements** OnInit {
10. **private** data: Array<any> = [];
12. constructor() {
13. **this**.data = [{ name: 'Galaxy Tab 3', company: 'Samsung', quantity: '10', price: '25000.00' },
14. { name: 'Galaxy Tab 5', company: 'Samsung', quantity: '50', price: '55000.00' },
15. { name: 'G4', company: 'LG', quantity: '10', price: '40000.00' },
16. { name: 'Canvas 3', company: 'Micromax', quantity: '25', price: '18000.00' }];
17. }
19. ngOnInit(): **void** {
20. }
21. }

**app.component.mobile.html**

1. <div **class**="panel-body">
2. <table **class**="table table-striped table-bordered">
3. <thead>
4. <tr>
5. <th>Name</th>
6. <th>Company</th>
7. <th **class**="text-right">Quantity</th>
8. <th **class**="text-right">Price</th>
9. </tr>
10. </thead>
11. <tbody>
12. <tr \*ngFor="let item of data">
13. <td>{{item.name}}</td>
14. <td>{{item.company}}</td>
15. <td **class**="text-right">{{item.quantity}}</td>
16. <td **class**="text-right">{{item.price |currency:'INR':**true**}}</td>
17. </tr>
18. </tbody>
19. </table>
20. </div>

**app.component.computer.ts**

1. **import** { Component, OnInit } from '@angular/core';
3. @Component({
4. selector: 'computer',
5. templateUrl: 'app.component.computer.html'
6. })
8. **export** **class** ComputerComponent **implements** OnInit {
10. **private** data: Array<any> = [];
12. constructor() {
13. **this**.data = [{ name: 'HP Pavilion 15"', company: 'HP', quantity: '10', price: '42000.00', specification: 'Intel Core i3 2 GB Ram 500 GB HDD with Windows 10' },
14. { name: 'Lenovo Flex 2"', company: 'Lenovo', quantity: '20', price: '32000.00', specification: 'Intel Core i3 2 GB Ram 500 GB HDD with DOS OS' },
15. { name: 'Lenovo Yova 500"', company: 'Lenovo', quantity: '20', price: '70000.00', specification: 'Intel Core i7 8 GB Ram 1TB HDD with Windows 8.1' }]
16. }
18. ngOnInit(): **void** {
19. }
20. }

**app.component.computer.html**

1. <div **class**="panel-body">
2. <table **class**="table table-striped table-bordered">
3. <thead>
4. <tr>
5. <th>Name</th>
6. <th>Company</th>
7. <th **class**="text-right">Quantity</th>
8. <th **class**="text-right">Price</th>
9. </tr>
10. </thead>
11. <tbody>
12. <tr \*ngFor="let item of data">
13. <td>{{item.name}}</td>
14. <td>{{item.company}}</td>
15. <td **class**="text-right">{{item.quantity}}</td>
16. <td **class**="text-right">{{item.price | currency}}</td>
17. </tr>
18. </tbody>
19. </table>
20. </div>

**app.component.home.ts**

1. **import** { Component, OnInit } from '@angular/core';
3. @Component({
4. selector: 'home',
5. templateUrl: 'app.component.home.html'
6. })
8. **export** **class** HomeComponent **implements** OnInit {
10. **private** message: string = '';
11. constructor() {
12. **this**.message = 'Click link to move other pages';
13. }
15. ngOnInit(): **void** {
16. }
17. }

**app.component.home.html**

1. <div **class**="row">
2. <div **class**="panel-body">
3. Home Page
4. <br />
5. <h3 **class**="panel-heading"><span>{{message}}</span></h3>
6. </div>
7. </div>

Now, we need to define the router link in the app-root component so that we can navigate between the different route components.

**app.component.ts**

1. **import** { Component } from '@angular/core';
3. @Component({
4. selector: 'app-root',
5. templateUrl: './app.component.html',
6. styleUrls: ['./app.component.css']
7. })
8. **export** **class** AppComponent {
9. title = 'Ang8RouteDemo';
10. }

**app.component.html**

1. <div **class**="toolbar" role="banner">
2. <img
3. width="40"
4. alt="Angular Logo"
5. src="data:image/svg+xml;base64,PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHZpZXdCb3g9IjAgMCAyNTAgMjUwIj4KICAgIDxwYXRoIGZpbGw9IiNERDAwMzEiIGQ9Ik0xMjUgMzBMMzEuOSA2My4ybDE0LjIgMTIzLjFMMTI1IDIzMGw3OC45LTQzLjcgMTQuMi0xMjMuMXoiIC8+CiAgICA8cGF0aCBmaWxsPSIjQzMwMDJGIiBkPSJNMTI1IDMwdjIyLjItLjFWMjMwbDc4LjktNDMuNyAxNC4yLTEyMy4xTDEyNSAzMHoiIC8+CiAgICA8cGF0aCAgZmlsbD0iI0ZGRkZGRiIgZD0iTTEyNSA1Mi4xTDY2LjggMTgyLjZoMjEuN2wxMS43LTI5LjJoNDkuNGwxMS43IDI5LjJIMTgzTDEyNSA1Mi4xem0xNyA4My4zaC0zNGwxNy00MC45IDE3IDQwLjl6IiAvPgogIDwvc3ZnPg=="
6. />
7. <span>Welcome !! Angular 8 Route Demo</span>
8. <div **class**="spacer"></div>
9. <a aria-label="Angular on twitter" target="\_blank" rel="noopener" href="https://twitter.com/angular" title="Twitter">
11. <svg id="twitter-logo" height="24" data-name="Logo — FIXED" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 400 400">
12. <defs>
13. <style>
14. .cls-1 {
15. fill: none;
16. }
18. .cls-2 {
19. fill: #ffffff;
20. }
21. </style>
22. </defs>
23. <rect **class**="cls-1" width="400" height="400" />
24. <path **class**="cls-2" d="M153.62,301.59c94.34,0,145.94-78.16,145.94-145.94,0-2.22,0-4.43-.15-6.63A104.36,104.36,0,0,0,325,122.47a102.38,102.38,0,0,1-29.46,8.07,51.47,51.47,0,0,0,22.55-28.37,102.79,102.79,0,0,1-32.57,12.45,51.34,51.34,0,0,0-87.41,46.78A145.62,145.62,0,0,1,92.4,107.81a51.33,51.33,0,0,0,15.88,68.47A50.91,50.91,0,0,1,85,169.86c0,.21,0,.43,0,.65a51.31,51.31,0,0,0,41.15,50.28,51.21,51.21,0,0,1-23.16.88,51.35,51.35,0,0,0,47.92,35.62,102.92,102.92,0,0,1-63.7,22A104.41,104.41,0,0,1,75,278.55a145.21,145.21,0,0,0,78.62,23"
25. />
26. </svg>
28. </a>
29. </div>
31. <div **class**="content" role="main">
32. <span><h2>{{ title }} app is running!</h2></span>
33. <table style="width:40%;">
34. <tr **class**="table-bordered">
35. <td><a routerLink="/home" **class**="btn-block" routerLinkActive="['active']">Home</a></td>
36. <td><a routerLink="/mobile">Mobile</a></td>
37. <td><a routerLink="/tv">TV</a></td>
38. <td><a routerLink="/computer">Computers</a></td>
39. </tr>
40. </table>
41. <br/>
42. <div **class**="spacer">
43. <router-outlet></router-outlet>
44. </div>
45. </div>

**app-routing.module.ts**

1. **import** { NgModule } from '@angular/core';
2. **import** { Routes, RouterModule } from '@angular/router';
3. **import** { HomeComponent } from './app.component.home';
4. **import** { MobileComponent } from './app.component.mobile';
5. **import** { TvComponent } from './app.component.tv';
6. **import** { ComputerComponent } from './app.component.computer';
8. **const** routes: Routes = [
9. { path: '', redirectTo: 'home', pathMatch: 'full' },
10. { path: 'home', component: HomeComponent },
11. { path: 'tv', component: TvComponent },
12. { path: 'mobile', component: MobileComponent },
13. { path: 'computer', component: ComputerComponent }
14. ];
16. @NgModule({
17. imports: [RouterModule.forRoot(routes)],
18. exports: [RouterModule]
19. })
20. **export** **class** AppRoutingModule { }

**app.module.ts**

1. **import** { BrowserModule } from '@angular/platform-browser';
2. **import** { NgModule } from '@angular/core';
4. **import** { AppRoutingModule } from './app-routing.module';
5. **import** { AppComponent } from './app.component';
6. **import** { HomeComponent } from './app.component.home';
7. **import** { MobileComponent } from './app.component.mobile';
8. **import** { TvComponent } from './app.component.tv';
9. **import** { ComputerComponent } from './app.component.computer';
11. @NgModule({
12. declarations: [
13. AppComponent, HomeComponent, MobileComponent, TvComponent, ComputerComponent
14. ],
15. imports: [
16. BrowserModule,
17. AppRoutingModule
18. ],
19. providers: [],
20. bootstrap: [AppComponent]
21. })
22. **export** **class** AppModule { }

Now run the application in the browser to check the output,

