**Explanation:**

1. **Define Route with Query Parameters:**
   * In app-routing.module.ts, the route path: 'search' does not include any path parameters.
   * Query parameters are added to the URL after the question mark (e.g., /search?term=angular&other=param).
2. **Access Query Parameters:**
   * In SearchComponent, the ActivatedRoute service is used to access the route parameters.
   * route.queryParams: An Observable that emits the query parameters as a Params object.
   * params['term']: Access the value of the term query parameter.
   * params['other']: Access the value of the other query parameter.
3. **Navigate with Query Parameters:**
   * In your application, you can navigate to the search route with query parameters as follows:

HTML

<a [routerLink]="['/search']" [queryParams]="{ term: 'angular', other: 'value' }">Search</a>

**Key Points:**

* Query parameters are optional and do not affect the component that is displayed.
* They are useful for passing additional information to the component without changing the URL path.
* You can use multiple query parameters by separating them with & (e.g., /search?term=angular&other=value).

This example demonstrates how to use query parameters in Angular routing to pass additional information to your components. You can adapt this approach to various scenarios, such as filtering data, customizing component behavior, or passing state information.

TypeScript

import { NgModule } from '@angular/core';

import { RouterModule, Routes } from '@angular/router';

import { SearchComponent } from './search/search.component';

const routes: Routes = [

{ path: 'search', component: SearchComponent }

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule]

})

export class AppRoutingModule { }

TypeScript

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute } from '@angular/router';

@Component({

selector: 'app-search',

template: `

<h2>Search Results</h2>

<p>Search Term: {{ searchTerm }}</p>

<p>Other Param: {{ otherParam }}</p>

`

})

export class SearchComponent implements OnInit {

searchTerm: string = '';

otherParam: string = '';

constructor(private route: ActivatedRoute) {}

ngOnInit() {

this.route.queryParams.subscribe(params => {

this.searchTerm = params['term'] || '';

this.otherParam = params['other'] || '';

});

}

}

**Explanation:**

1. **Define Routes:**
   * In app-routing.module.ts, define a route for the SearchComponent at the /search path.
2. **Access Query Params:**
   * In SearchComponent, inject the ActivatedRoute service.
   * Use route.queryParams.subscribe() to listen for changes to the query parameters.
   * Access the query parameters using params['term'] and params['other'].
   * Provide default values (empty strings) for the parameters in case they are not present.

**Navigating to the Search Component with Query Params:**

* In your application, create a link to the SearchComponent with query parameters:

HTML

<a [routerLink]="['/search']" [queryParams]="{ term: 'keyword', other: 'value' }">Search</a>

This link will navigate to /search?term=keyword&other=value.

**Key Points:**

* Query parameters are part of the URL after the question mark (?).
* They are used to pass additional information to the component without changing the main URL path.
* You can use multiple query parameters by separating them with ampersands (&).
* The ActivatedRoute service provides access to route parameters and query parameters.

This example demonstrates how to use query parameters in Angular routing to pass data to your components and create more dynamic and interactive user experiences.