There are 2 ways to get the parameter from the route.

**1. Snapshot (route.snapshot.paramMap.get). Read it during init.**

Use the Snapshot if you only need the initial value of the parameter once during the component's initialization, and don't expect the URL to change while the user is still on that same component.

* I.e. if on a product/2 route, and the only way they'd get to product/3 is by going back to the product search screen and then clicking a product detail (leaving the detail component, then re-opening it with a new route param)

class MyComponent

{

constructor(route: [ActivatedRoute](https://angular.io/api/router/ActivatedRoute))

{

const id: string = route.snapshot.params.id;

const url: string = route.snapshot.url.join('');

const user = route.snapshot.data.user;

}

}

**2. Observable (route.paramMap.subscribe). Subscribe to it during init.**

Use the Observable if it's possible for the route to change while the user is still on the same component, and hence the Component's initialization would not be called again, but the observable would call your subscribed logic when the URL changed.

* I.e. if on your product detail page you have a "next" button to go to the next id, hence the user did not leave/re-open the component but the URL did receive a new param.

Generally speaking, subscribing is the safest route if you're unsure.

ngOnInit() {

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

this.products.forEach((p: Product) => {

if (p.id == params.id) {

this.product = p;

}

});

});

}

### Route Guards

Guarding routes means whether we can visit the route or not. For example in login authentication based application, a user has to login first to enter into the application. If there is no route guard then anyone can access any link but using route guard we restrict the access of links. To achieve route guards, Angular provides following interfaces that are contained in @angular/router package.   
1. CanActivate   
2. CanActivateChild   
3. CanDeactivate   
4. Resolve   
5. CanLoad

**A. Using CanActivate**  
  
**1.** CanActivate is an Angular interface. It is used to force user to login into application before navigating to the route. Find the declaration of CanActivate interface from Angular doc.

interface CanActivate {

canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean>|Promise<boolean>|boolean

}

CanActivate interface has a method named as canActivate() which has following arguments.   
**ActivatedRouteSnapshot**: Contains the information about a route associated with component loaded in outlet in particular time. It can traverse router state tree.   
**RouterStateSnapshot**: It is a tree of activated route snapshots. It has url property that gives the URL from which this snapshot was created.   
  
canActivate() returns boolean value or Observable or Promise of boolean value.   
**2.** We need to create a service implementing CanActivate interface and override canActivate() method.

@Injectable()

export class AuthGuardService implements CanActivate {

canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {

return true;

}

}

**3.** Use canActivate property of Route interface to guard the route and assign service class implementing CanActivate interface, for example AuthGuardService. Now find the canActivate property used in route declarations.

{

path: 'home',

component: DashboardLayoutComponent,

canActivate: [ AuthGuardService ]

}

If canActivate() method from AuthGuardService returns true only when route can be navigated. In case of falsevalue, navigation can be redirected to login page.   
  
**B. Using CanActivateChild**  
  
**1.** CanActivateChild is an Angular interface to guard child routes. Suppose a user has been authenticated but not authorized to visit the child routes, so child routes can be guarded using CanActivateChild. Find its declaration from the Angular doc.

interface CanActivateChild {

canActivateChild(childRoute: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable|Promise|boolean

}

Method signature of canActivateChild() is the same as canActivate() .   
**2.** We need to create a service implementing CanActivateChild and override canActivateChild() method.

@Injectable()

export class AuthGuardService implements CanActivate, CanActivateChild {

canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {

return true;

}

canActivateChild(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {

return true;

}

}