**Angular**

Graphical user interface, application

Description automatically generated

# What are Route Guards?

Angular’s route guards are interfaces which can tell the router whether or not it should allow navigation to a requested route. They make this decision by looking for a true or false return value from a class which implements the given guard interface.

There are five different types of guards and each of them is called in a particular sequence. The router’s behavior is modified differently depending on which guard is used. The guards are:

* CanActivate
* CanActivateChild
* CanDeactivate
* CanLoad
* Resolve

**https://www.concretepage.com/angular-2/angular-2-routing-and-navigation-example**

[**https://www.concretepage.com/angular/angular-select-option-selected**](https://www.concretepage.com/angular/angular-select-option-selected)

**https://www.concretepage.com/angular/**

[**https://www.tektutorialshub.com/angular-tutorial/**](https://www.tektutorialshub.com/angular-tutorial/)

[**https://www.freecodecamp.org/news/angular-8-tutorial-in-easy-steps/**](https://www.freecodecamp.org/news/angular-8-tutorial-in-easy-steps/)

[**https://www.concretepage.com/angular-2/angular-2-formgroup-example**](https://www.concretepage.com/angular-2/angular-2-formgroup-example)

This example covers the Angular example in typescript.

* Declare and initialize an array of objects with values.
* Array of Type
* Using an interface to hold an array of objects.

**https://www.cloudhadoop.com/angular-declare-array-objects/**

**https://www.bezkoder.com/angular-12-jwt-auth/**

## What is Angular?

The Angular is a development platform for building a Single Page Application for mobile and desktop. It uses Typescript & HTML to build Apps. The Angular itself is written using the Typescript. It now comes with every feature you need to build a complex and sophisticated web or mobile application. It comes with features like [component](https://www.tektutorialshub.com/angular/angular-component/), [Directives](https://www.tektutorialshub.com/angular/angular-directives/), [Forms](https://www.tektutorialshub.com/angular/angular-forms-fundamentals/), [Pipes](https://www.tektutorialshub.com/angular/angular-pipes/), [HTTP Services](https://www.tektutorialshub.com/angular/angular-httpclient/), [Dependency Injection](https://www.tektutorialshub.com/angular/angular-dependency-injection/), etc

### Key features of Angular

* [**MVC architecture**](https://www.educative.io/blog/mvc-tutorial) (Model-View-Controller): Separates program components into buckets of Model, View, and Controller to separate the presentation layer from the business logic.
* **Two-way Data Binding**: Angular automatically synchronizes your HTML view with your code, allowing you to watch your view update to changes in real-time.
* **Routing Support**: Easily build single-page applications (SPAs) that provide a desktop experience when navigating across views.
* **Dependency Injection**: Allows code dependencies to be automatically injected with a dependency container, so no main() method is required.
* **Form Validation**: Improve user experience in CRUD applications with Angular’s easy-to-implement form validation.

# Node Package Manager

* **Npm** − This is known as the node package manager that is used to work with the open source repositories. Angular JS as a framework has dependencies on other components. And **npm** can be used to download these dependencies and attach them to your project.

# Modules

Modules are used in Angular JS to put logical boundaries in your application. Hence, instead of coding everything into one application, you can instead build everything into separate modules to separate the functionality of your application. Let’s inspect the code which gets added to the demo application.

[**https://stackblitz.com/edit/angular-tour-of-heroes-example?file=app%2Fhero-detail.component.html**](https://stackblitz.com/edit/angular-tour-of-heroes-example?file=app%2Fhero-detail.component.html)

**Examples**[**https://stackblitz.com/angular/jxbjbvaedvj?file=src%2Fapp%2Fapp.component.ts**](https://stackblitz.com/angular/jxbjbvaedvj?file=src%2Fapp%2Fapp.component.ts)

[**https://www.digitalocean.com/community/tutorials/angular-query-parameters**](https://www.digitalocean.com/community/tutorials/angular-query-parameters)

And in our example, if you want to 'preserve' or 'merge' query parameters on subsequent navigation you would use queryParamsHandling like this:

<a [routerLink]="['/users']"

[queryParams]="{ filter: 'new' }"

queryParamsHandling="merge">

Users

</a>

**Key features of Angular**

* [**MVC architecture**](https://www.educative.io/blog/mvc-tutorial) (Model-View-Controller): Separates program components into buckets of Model, View, and Controller to separate the presentation layer from the business logic.
* **Two-way Data Binding**: Angular automatically synchronizes your HTML view with your code, allowing you to watch your view update to changes in real-time.
* **Routing Support**: Easily build single-page applications (SPAs) that provide a desktop experience when navigating across views.
* **Dependency Injection**: Allows code dependencies to be automatically injected with a dependency container, so no main() method is required.
* **Form Validation**: Improve user experience in CRUD applications with Angular’s easy-to-implement form validation.

**Angular**

Angular is built with the opposite of React: maximize control at the cost of beginner-friendliness.

Once you master Angular, you have the tools to control and fine-tune every aspect of your app in a way you couldn’t with React.

**Advantages**

* Angular is supported by Google, with detailed documentation and a large community. There are numerous high-quality resources provided to help you learn quicker.
* Angular-language-service speeds up development with advanced coding features like autocomplete for external HTML template files.
* Advanced MVC architecture for better division of responsibilities and code organization.
* Supports test-driven development with instant code-to-view updates and tools for both end-to-end and unit testing.

**Limitations**

* Harder to learn than React because Angular offers a variety of different structures like Injectables, Components, Pipes, Modules, and more. It takes time to learn the place for each of these structures rather than learning just components to React.
* Slower performance by default because it works with the real DOM. Requires additional work to perform as fast as React, such as manual control of the rendering process.

## **Available local variables**

You can also set local variables for the following exported values: index, first, last, even and odd. index will return the current loop index, and the other values with provide a boolean indicating if the value is true or false. For example:

<ul>

<li \*ngFor="let user of users; let i = index; let odd = odd"

[class.odd]="odd">

{{i + 1}}. {{ user.name }}

</li>

</ul>

# Scoping rules

Main difference is scoping rules. Variables declared by var keyword are scoped to the immediate function body (**hence the function scope**) while let variables are scoped to the immediate enclosing block denoted by { } (**hence the block scope**).

function run() {

var foo = "Foo";

let bar = "Bar";

console.log(foo, bar); // Foo Bar

{

var moo = "Mooo"

let baz = "Bazz";

console.log(moo, baz); // Mooo Bazz

}

console.log(moo); // Mooo

console.log(baz); // ReferenceError

}

Variable declaration :

export class className {

property: propertytype = value;

}

appTitle: string = 'Welcome';

### Example: Variables in TypeScript

var name:string = "John";

var score1:number = 50;

var score2:number = 42.50

var sum = score1 + score2

console.log("name"+name)

console.log("first score: "+score1)

console.log("second score: "+score2)

console.log("sum of the scores: "+sum)

## **TypeScript Variable Scope**

The scope of a variable specifies where the variable is defined. The availability of a variable within a program is determined by its scope. TypeScript variables can be of the following scopes −

* **Global Scope** − Global variables are declared outside the programming constructs. These variables can be accessed from anywhere within your code.
* **Class Scope** − These variables are also called **fields**. Fields or class variables are declared within the class but outside the methods. These variables can be accessed using the object of the class. Fields can also be static. Static fields can be accessed using the class name.
* **Local Scope** − Local variables, as the name suggests, are declared within the constructs like methods, loops etc. Local variables are accessible only within the construct where they are declared.

The following example illustrates variable scopes in TypeScript.

### Example: Variable Scope

var global\_num = 12 //global variable

class Numbers {

num\_val = 13; //class variable

static sval = 10; //static field

storeNum():void {

var local\_num = 14; //local variable

}

}

console.log("Global num: "+global\_num)

console.log(Numbers.sval) //static variable

var obj = new Numbers();

console.log("Global num: "+obj.num\_val)

## **Declaring and Initializing Arrays**

To declare an initialize an array in Typescript use the following syntax −

### Syntax

var array\_name[:datatype]; //declaration

array\_name = [val1,val2,valn..] //initialization

As of TypeScript 2.7.2, you are required to initialise a property in the constructor if it was not assigned to at the point of declaratio

Solution -

Added "**strictPropertyInitialization":** false in 'compilerOptions' of **tsconfig.json**.

services and dependency injection

To define a class as a service in Angular, use the @[Injectable](https://angular.io/api/core/Injectable)() decorator to provide the metadata that allows Angular to inject it into a component as a dependency. Similarly, use the @[Injectable](https://angular.io/api/core/Injectable)() decorator to indicate that a component or other class (such as another service, a pipe, or an NgModule) has a dependency.

* The injector is the main mechanism. Angular creates an application-wide injector for you during the bootstrap process, and additional injectors as needed. You don't have to create injectors.
* An injector creates dependencies, and maintains a container of dependency instances that it reuses if possible.
* A provider is an object that tells an injector how to obtain or create a dependency.

Module can be register in NGMoudle or component level.

@[NgModule](https://angular.io/api/core/NgModule)({ providers: [ BackendService, Logger ], ... })

@[Component](https://angular.io/api/core/Component)({ selector: 'app-hero-list', templateUrl: './hero-list.component.html', providers: [ HeroService ] })

When Angular creates a new instance of a component class, it determines which services or other dependencies that component needs by looking at the constructor parameter types. For example, the constructor of HeroListComponent needs HeroService.

src/app/hero-list.component.ts (constructor)

content\_copyconstructor(private service: HeroService) { }

# ReactiveForm validation:

<https://www.tektutorialshub.com/angular/angular-reactive-forms-validation/#how-to-add-a-validator-to-reactive-forms>

<https://blog.jscrambler.com/validations-in-angular-reactive-forms>

https://www.techiediaries.com/angular/angular-9-reactive-forms-validation-tutorial-example/

ngOnInit() {

this.userForm = this.formBuilder.group({

firstName: ['', [Validators.required, Validators.pattern('^[a-zA-Z]+$'), Validators.maxLength(10)]],

lastName: ['',[Validators.required, Validators.pattern('^[a-zA-Z]+$'), Validators.maxLength(10)]],

email: ['', [Validators.required, Validators.email, this.duplicateEmailValidator.bind(this)]]

});

}

https://jasonwatmore.com/search?q=angular

# Routing

https://www.tektutorialshub.com/angular/angular-routing-navigation/

# What is Typescript

The [Typescript](https://www.tektutorialshub.com/typescript-tutorial/) is a superset of Javascript. The Code written in Typescript cannot be used directly in the web browser. It must be compiled to Javascript before running in the web browser. This process is known as Transpiling.

<https://www.tutorialkart.com/typescript/typescript-class/>

<https://www.javatpoint.com/angular-8-ngfor-directive>

# Forms :

<https://www.concretepage.com/angular-2/angular-2-formgroup-example>

<https://appdividend.com/2019/12/16/angular-form-control-example/>

https://www.tektutorialshub.com/angular/select-options-example-in-angular/

https://blog.angular-university.io/angular-form-array/

# Spring boot:

https://www.javainuse.com/spring/cloud-gateway

