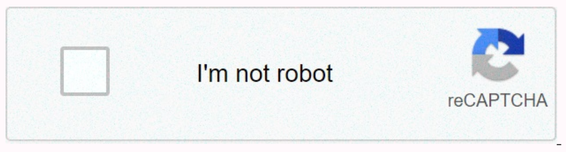

I'm not robot!

Angular template if else if

Angular ng template if else example. Angular if else example. Angular template ng if else if.

Contents More Related Answers ngclass else ngif else ng-container if condition in class angular 8 ionic ngif else example ngif using multiple elseif ngif else if angular ngif ngif else angular angular nested if else angular js if ng if ng if ng elseif ngif else ngif else ngif else ng-if ng else ng-template if else njk else if A quick tutorial on how to use the *ngIf else directive in Angular to perform comparisons in your Angular apps. What is Angular? This post aims to solve common questions about using *ngIf while developing web apps with Angular. For those new to it, Angular, Google's JavaScript (TypeScript) framework for building web applications, mobile or desktop, has over 56,000 stars on GitHub. It's maintained by the Angular team at Google and a host of community members and organizations. Before You Start To be able to follow through in this article's demonstration you should have: A integrated development environment like VS Code Node version 11.0 installed on your machine Node Package Manager version 6.7 (usually ships with Node installation) Angular CLI version 9.0 The latest version of Angular (version 9) // run the command in a terminal ng version Confirm that you are using version 9, and update if you are not. Download this tutorial's starter project here to follow through the demonstrations Unzip the project and initialize the node modules in your terminal with this command: npm install Other nice-to-haves are: A working knowledge of the Angular framework at a beginner level Comparisons in Programming Logic When building your Angular application or any other application, there is always a time when you have to compare between two entities. These entities can be variables or array items. The concept of comparison has helped break down a lot of complex logic into simple terms. These comparisons can be done with conditionals. Conditionals in Angular 9 For every comparison, there is a condition — like, if today is Friday, then display "Happy Friday!" Just like most programming languages, Angular has directives like if, for and switch for handling comparisons. In this post, you will learn how to use the if directive to handle comparisons in Angular.
The ngIf Directive + else According to the Angular API, the ngIf directive is a structural directive that conditionally includes a template based on the value of an expression coerced to Boolean. When the expression evaluates to true, Angular renders the template provided in a then clause, and when false or null, Angular renders the template provided in an optional else clause. ngIf Syntax The ngIf directive syntax looks like this:
Content to render when condition is true.
An extended version of this would look like this:
Content to render when condition is true.
Finally, you can add an outcome for when your conditions were not met. This is where the else clause comes into an if statement. The idea is that your logic should be constructed like this: If condition is met, do this, else do something new. Adding the else clause, the syntax looks like this:
Content to render when condition is true.
Content to render when condition is false. Demo Let's build a simple toggle Angular component to illustrate the ngIf directive.
Open up your ng canvas project you already unzipped in VS Code, and inside your src directory you will find an app folder. The app.component.ts file should look like this: import { Component } from '@angular/core'; @Component({ selector: 'app-root', templateUrl: './app.component.html', styleUrls: ['./app.component.css'] }) export class AppComponent { title = 'ngcanvas'; } To start off, we will generate a new component to work with. Open the terminal inside VS Code and run the command below inside it: ng generate component comparisons You should have a new comparisons folder inside the app directory. Clean up your app.component.html file and paste this code block inside it:

Welcome to ngIf app

For the toggle logic, open your comparisons.component.html file and replace the paragraph code (comparisons work!) with this code block below:

Toggle Greetings

Hello

Good morning to you

Today is Friday and this is a dummy text element to make you feel better

Understanding the ngIf directive with the else clause

Here, we first created a button and assigned it to a toggleOn event which turns either true or false as you click. Then, we have a div tag that contains the greetings. On the div tag we added the ngIf directive that would be displayed if the value of toggleOn is false. After that, we added some dummy paragraphs. This is how to use the ngIf directive. It can be used in all types of use cases and comparisons you can think of in your component template. The else Clause There are some scenarios where you want to display or return something else if the condition you set does not work out. Like heading to the store to get Snickers but they are out of stock, so you get Bounty instead. Angular provides an additional enhancement to the ngIf directive by adding an else clause. This also has a very simple logic: If the condition you specified is not fulfilled, do this instead. In our demo above, you see that if the toggleOn value is false, nothing is displayed; but if it is on, then the greetings will be displayed. With the else clause, we can now display something when the toggleOn value is false. Modify your your comparisons.component.html file with this newer version below:

Toggle Greetings

Hello

Good morning to you

No greeting, Lagbaja nothing for you.

Today is Friday and this is a dummy text element to make you feel better

Understanding the ngIf directive with the else clause

Now the app shows the predefined content for the else side of things when you make your comparisons. Other Perks The ngIf directive also has some more perks you might want to know about, one of which is local variable assignment. Sometimes, the result of the ngIf directive is not a boolean. Angular allows you to save variables locally and access them with your ngIf directive. Using the snacks analogy I introduced during the else section, open your comparisons.component.ts file and replace the content with the code block below: import { Component, OnInit } from '@angular/core'; @Component({ selector: 'app-comparisons', templateUrl: './comparisons.component.html', styleUrls: ['./comparisons.component.css'] }) export class ComparisonsComponent implements OnInit { Snacks = { chocolate: 'Snickers' }; constructor() { } ngOnInit() {} } Now we have declared a snacks object and saved Snickers as the only type of chocolate in it. Update your comparisons.component.html file with the code block below:

Toggle Greetings

Hello

Good morning to you

No greeting, Lagbaja nothing for you.

Today is Friday and this is a dummy text element to make you feel better

Understanding the ngIf directive with the else clause

Nice {{ chocolate }}!
Get bounty instead. If you serve the application, you will see that you can indeed access Snickers from your directive and even the else block in it. The applications of these ngIf directive concepts are endless and are only limited to your creativity and the use cases you can imagine. Conclusion You have gone through the Angular ngIf directive and how it is used to make handling comparisons easy. You were also introduced to additional concepts provided by the directive, like the else clause and the local variable assignment, and how you can start using them in your apps today. Happy coding! Let’s look at the NgIf directive in this tutorial and uncover how to use it. We’ll also explore using NgIf with the “Else” statement and “Then”, to give you a full guide on how to use it. You’ll learn how to show and hide DOM content based on your data, which we can then let NgIf handle and render updates to the DOM for us! What is NgIf? Before we dive in too deep, let’s learn the concepts behind NgIf and why it exists. NgIf is a behavioral directive that allows us to toggle part of a template based on a conditional value. This conditional would be evaluated similarly to how JavaScript would evaluate an if (condition) {} statement, converting the value you supply to a truthy or falsy value and progressing accordingly. Let’s explore the ins and outs of ngIf, and how we can utilise (the right way) in our Angular apps. Using Angular’s NgIf The syntax for NgIf is nice and simple, we can declare it on an element, or component, and let it work its magic. Placing the ngIf directive on a component, or element, will in fact hide or show that element based on the expression you pass it to be evaluated. Angular will simply add or remove your DOM nodes, mount or remount your components as the expression changes (if it ever does, that’s up to you). We’ll also cover why we use the asterisk syntax, shortly.
Standard *ngIf in Angular There are four main ways we can use ngIf, so let’s start by exploring the most basic use case. Let’s take an empty component and a simple Boolean value of true: @Component({ selector: 'app-component', template: `
Welcome back!
`, }) export class AppComponent { isLoggedIn = true; } We can also use JavaScript-like expressions to achieve a final truthy/falsy value to supply to ngIf - as well as composing through multiple variables through various operators. The basic syntax of the ngIf directive is simple and effective, all we need to do is prefix the directive name with an asterisk (*) and add it anywhere inside our template:
Please login, friend.
Welcome back, friend.
Welcome!
Just a few examples, but I’m sure you can see how easy and clean it is to use ngIf. Note that the ngIf used is a lowercase “n” when declared on an element or component. Let’s move onto some more interesting examples! *ngIf and Else One fantastic addition in Angular is the “else” statement. It behaves very similar to a JavaScript if (condition) { } else { } statement. Nice and simple! Here’s how we can use the “else” statement, to control the render flow inside a component’s template:
Welcome back, friend.
Please friend, login. Woah, what’s this whole #loggedOut syntax? That’s a template variable. You can name template variables as you wish. Using a template variable means that we can create a reference to a specific template part and then use it elsewhere - in this example we’re supplying it as an “else” value to ngIf. We use the because much like it’s HTML5 counterpart