What is the equivalent of ngShow and ngHide in Angular 2+?

Asked 3 years, 5 months ago Active 1 month ago Viewed 455k times

463

I have a number of elements that I want to be visible under certain conditions.

In AngularJS I would write

```
<div ng-show="myVar">stuff</div>
```

How can I do this in Angular 2+?

62

angular angular-components angular-template

edited May 27 at 9:08

Kamil Naja
2,120 3 12 24

asked Feb 23 '16 at 12:50

Mihai Răducanu
3,904 3 17 26

15 Answers

795

Just bind to the `hidden` property

```
[hidden]="!myVar"
```

See also

- https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/hidden

issues

`hidden` has some issues though because it can conflict with CSS for the `display` property.

See how `some` in **Plunker example** doesn't get hidden because it has a style

```
:host {display: block;}
```

set. (This might behave differently in other browsers - I tested with Chrome 50)

workaround

You can fix it by adding

```
[hidden] { display: none !important;}
```

To a global style in `index.html` .

another pitfall

```
hidden="false"
hidden="{{false}}"
hidden="{{isHidden}}" // isHidden = false;
```

are the same as

```
hidden="true"
```

and will not show the element.

`hidden="false"` will assign the string `"false"` which is considered truthy.
Only the value `false` or removing the attribute will actually make the element visible.

Using `{{}}` also converts the expression to a string and won't work as expected.

Only binding with `[]` will work as expected because this `false` is assigned as `false` instead of `"false"` .

`*ngIf` vs `[hidden]`

`*ngIf` effectively removes its content from the DOM while `[hidden]` modifies the `display` property and only instructs the browser to not show the content but the DOM still contains it.

edited Feb 12 '17 at 10:35

answered Feb 23 '16 at 12:51

Günter Zöchbauer
358k 82 1137 1037

19 Using hidden is actually not recommended. angularjs.blogspot.com/2016/04/… – Sam Sep 3 '16 at 5:02

11 I mentioned that in my answer already. – Günter Zöchbauer Sep 3 '16 at 16:31

7 `*ngIf` may be the correct way in most cases, but sometimes you actually want an element to be there, visually hidden. A CSS style with `[hidden] {display:none!important}` helps. That's, for example, how Bootstrap makes sure `[hidden]` elements are actually hidden. See GitHub – sb. Mar 1 '17 at 13:42

You may encounter some issue when you use (myStream | async) pipe inside of *ngIf that also uses (myStream | async) pipe – Pavel Blagodov Sep 26 '17 at 12:03

1 you are my saviour! using *ngIf will reset the DOM position to the top but [hidden] solved my problem and preserved the position. – Santosh Dec 8 '17 at 15:39

129

Use the `[hidden]` attribute:

```
[hidden]="!myVar"
```

Or you can use `*ngIf`

```
*ngIf="myVar"
```

These are two ways to show/hide an element. The only difference is: `*ngIf` will remove the element from DOM while `[hidden]` will tell the browser to show/hide an element using CSS `display` property by keeping the element in DOM.

edited May 15 at 5:17

answered Feb 23 '16 at 12:56

Ali Shahzad
2,770 5 19 45

3 [hidden] is adding conditionnaly an attribute "hidden" to the element. It also could be [whatever] or [ali]. The important thing here is to load a CSS rule who mention "hidden" attributes has to be display:none – Gabriel Aug 5 '16 at 7:14

4 Bear in mind: *ngIf and [hidden] are fundamentalyl different. ngIf will not evaluate the content inside the *ngIf block until the condition is true. This is especially important if you use the `async` pipe, as the subscription to the observable will only be added after the condition becomes true! – Dynalon Sep 9 '16 at 7:59

1 One more thing to take into consideration is that *ngIf destroys the component and it has to be re-created, while [hidden] keeps it alive and in memory. If you have a resource-intense component it may be preferable to hide it instead of destroy it – Michael Kork. Nov 16 '17 at 15:33

1 they are not same thing. – Kamuran Sönecek Aug 18 '18 at 10:37

27

I find myself in the same situation with the difference than in my case the element was a flex container.If is not your case an easy work around could be

```
[style.display]="!isLoading ? 'block' : 'none'"
```

in my case due to the fact that a lot of browsers that we support still need the vendor prefix to avoid problems i went for another easy solution

```
[class.is-loading]="isLoading"
```

where then the CSS is simple as

```
&.is-loading { display: none }
```

to leave then the displayed state handled by the default class.

edited Aug 9 '18 at 7:00

answered Feb 13 '17 at 10:06

Vale Steve
703 6 12

1 This works well with bootstrap 4 `invalid-feedback` class. – Jess Oct 25 '18 at 14:35

25

Sorry, I have to disagree with binding to hidden which is considered to be unsafe when using Angular 2. This is because the hidden style could be overwritten easily for example using

```
display: flex;
```

The recommended approach is to use *ngIf which is safer. For more details, please refer to the official Angular blog. 5 Rookie Mistakes to Avoid with Angular 2

```
<div *ngIf="showGreeting">
   Hello, there!
</div>
```

edited Dec 11 '16 at 23:38

answered Oct 28 '16 at 21:28

Tim Hong
2,216 16 22

10 I think it's a rookie mistake to say something is bad before knowing the exact requirements. If one doesn't want an element to be removed and destroyed and added and recreated, `*ngIf` is a poor choice. But you are right that consequences need to be considered and pointing out pitfalls is always a good idea. – Günter Zöchbauer Nov 25 '16 at 5:43

2 I know what you mean. It is not my word about it is a novice mistake, it is taken from Angular 2 official blog. I don't mean to offend anyone. Thanks for pointing out, though. – Tim Hong Nov 25 '16 at 21:22

8 Yeah, I don't think `ngIf` exactly answers what this question is asking. I want to hide some content on a page that includes a `<router-outlet>`. If I use `ngIf`, I get an error that it can't find the outlet. I need the outlet to be *hidden* until my data loads, not *absent* until my data loads. – Jason Swett Nov 27 '16 at 23:56

I agree with you, but the problem that I have is I want to show a form and put values in it if I use the *ngIf I will have the error that it is not defined and with the hidden property it is working well – Hazem HASAN Mar 5 at 8:49

@HazemHASAN, sure. I understand. The solution is always conditional. In your case, not sure if it is possible to just check if the form is there before you run any other code against it. It is all about the trade-off. Do you want a safer way to hide the form which will not be offset by another styling in future accidentally? Or do you prefer to have the convenience not to check if the form exists? – Tim Hong Mar 5 at 21:08

4

According to Angular 1 documentation of ngShow and ngHide, both of these directive adds the css style `display: none !important;`, to the element according to the condition of that directive (for ngShow adds the css on false value, and for ngHide adds the css for true value).

We can achieve this behavior using Angular 2 directive ngClass:

```
/* style.css */
.hide
{
    display: none !important;
}

<!-- old angular1 ngShow -->
<div ng-show="ngShowVal"> I'm Angular1 ngShow... </div>
```

```
<!-- become new angular2 ngClass -->
<div [ngClass]="{ 'hide': !ngShowVal }"> I'm Angular2 ngShow... </div>

<!-- old angular2 ngHide -->
<div ng-hide="ngHideVal"> I'm Angular1 ngHide... </div>

<!-- become new angular2 ngClass -->
<div [ngClass]="{ 'hide': ngHideVal }"> I'm Angular2 ngHide... </div>
```

Notice that for `show` behavior in Angular2 we need to add `!` (not) before the ngShowVal, and for `hide` behavior in Angular2 we **don't** need to add `!` (not) before the ngHideVal.

edited Dec 1 '16 at 11:57

answered Dec 1 '16 at 10:10

Gil Epshtain

2,342 4 25 47

3

If your case is that the style is display none you can also use the ngStyle directive and modify the display directly, I did that for a bootstrap DropDown the UL on it is set to display none.

So I created a click event for "manually" toggling the UL to display

```
<div class="dropdown">
    <button class="btn btn-default" (click)="manualtoggle()"  id="dropdownMenu1" >
    Seleccione una Ubicación
    <span class="caret"></span>
    </button>
    <ul class="dropdown-menu" [ngStyle]="{display:displayddl}">
        <li *ngFor="let object of Array" (click)="selectLocation(location)">
{{object.Value}}</li>
     </ul>
 </div>
```

Then on the component I have showDropDown:bool attribute that I toggle every time, and based on int, set the displayDDL for the style as follows

```
showDropDown:boolean;
displayddl:string;
manualtoggle(){
    this.showDropDown = !this.showDropDown;
    this.displayddl = this.showDropDown ? "inline" : "none";
}
```

answered Aug 8 '16 at 15:22

Gary
364 4 7

If you are using Bootstrap is as simple as this:

3

```
<div [class.hidden]="myBooleanValue"></div>
```

answered Mar 22 '17 at 17:24

Gian Marco Gherardi
15.8k 4 40 40

3 In bootstrap 4 using `[hidden]` does the same so I recommend `[hidden]` – Vahid Mar 23 '18 at 11:42

---

in bootstrap 4.0 the class "d-none" = "display: none!important;"

3

```
<div [ngClass]="{'d-none': exp}"> </div>
```

answered Mar 7 '18 at 17:36

63RMAN
31 2

For anybody else stumbling across this issue, this is how I accomplished it.

3

```
import {Directive, ElementRef, Input, OnChanges, Renderer2} from "@angular/core";

@Directive({
  selector: '[hide]'
})
export class HideDirective implements OnChanges {
  @Input() hide: boolean;
```

```
  constructor(private renderer: Renderer2, private elRef: ElementRef) {}

  ngOnChanges() {
    if (this.hide) {
      this.renderer.setStyle(this.elRef.nativeElement, 'visibility', 'hidden');
    } else {
      this.renderer.setStyle(this.elRef.nativeElement, 'visibility', 'visible');
    }
  }
}
```

I used `'visibility'` because I wanted to preserve the space occupied by the element. If you did not wish to do so, you could just use `'display'` and set it to `'none'` ;

You can bind it to your html element, dynamically or not.

```
<span hide="true"></span>
```

or

```
<span [hide]="anyBooleanExpression"></span>
```

answered Apr 25 '18 at 23:06

Anjil Dhamala
756 2 10 22

2

Use **hidden** like you bind any **model** with control and specify **css** for it:

HTML:

```
<input type="button" class="view form-control" value="View" [hidden]="true" />
```

CSS:

```
[hidden] {
   display: none;
}
```

answered Sep 28 '16 at 9:11

Sandip - Full Stack Developer
10.2k 4 20 49

1

```
<div [hidden]="flagValue">
---content---
</div>
```

answered Apr 15 at 9:00

Chirag
291 2 11

1

```
<div [hidden]="myExpression">
```

myExpression may be set to true or false

edited Jun 18 at 14:10

masterxilo
1,071 1 13 23

answered Sep 28 '16 at 9:19

Niyaz
733 2 9 31

2 `<div hidden="{{ myExpression }}">` This won't work, as "myExpression" will get converted to a string to be rendered in the html. Both the string "true" and "false" are truthy, so it will always be hidden – Viprus Sep 6 '17 at 13:21

0

There are two examples on Angular documents https://angular.io/guide/structural-directives#why-remove-rather-than-hide

> A directive could hide the unwanted paragraph instead by setting its display style to none.

```
<p [style.display]="'block'">
  Expression sets display to "block".
```

```
  This paragraph is visible.
</p>

<p [style.display]="'none'">
  Expression sets display to "none".
  This paragraph is hidden but still in the DOM.
</p>
```

You can use [style.display]="'block'" to replace ngShow and [style.display]="'none'" to replace ngHide.

answered May 10 at 3:19

koo

55 1 7

-1

To hide and show div on button click in angular 6.

Html Code

```
<button (click)=" isShow=!isShow">FormatCell</button>
<div class="ruleOptionsPanel" *ngIf=" isShow">
<table>
<tr>
<td>Name</td>
<td>Ram</td>
</tr>
</table>
</div>
```

Component .ts Code

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent{
 isShow=false;
  }
```

this works for me and it is way to replace ng-hide and ng-show in angular6.

enjoy...

Thanks

answered Aug 30 '18 at 6:49

Manoj Gupta
138 1 6

You are using ngIf - which is different than ngShow. NgIf will remove/add the element form the DOM. This is not the same as ngShow/ngHide which will only add/remove Css styles to the Element. – Gil Epshtain Jan 28 at 17:19

The example is too long and too specific. – masterxilo Jun 18 at 11:06

-1

for me, `[hidden]=!var` has never worked.

So, `<div *ngIf="expression" style="display:none;">`

And, `<div *ngIf="expression">` Always give correct results.

edited Jun 21 at 11:59

ganesh045
1,755 2 6 17

answered Jun 21 at 5:15

Priyanka Arora
48 5

protected by Günter Zöchbauer Aug 14 '18 at 17:19

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?