Angular HTML binding

Asked 4 years ago Active 11 days ago Viewed 412k times

686

I am writing an Angular application and I have an HTML response I want to display.

How do I do that? If I simply use the binding syntax `{{myVal}}` it encodes all HTML characters (of course).

I need somehow to bind the `innerHTML` of a `div` to the variable value.

100

angular angular2-template angular2-databinding

edited Jun 9 at 17:45

Mark Amery
70.4k 33 273 319

asked Jul 21 '15 at 19:48

Aviad P.
19.3k 8 81 100

Related post for getting CSS defined in a component to work right in the HTML binding stackoverflow.com/questions/36265026/… – y3sh Sep 11 '17 at 18:32

I put together a video response to explain the solution and give an example: youtube.com/watch?v=Pem2UXp7TXA – Caleb Grams Jan 30 at 0:24

what if variable holds angular tag or user defined tag like <span [routerLink]=['some-router']>link – G. Muqtada Apr 27 at 19:57

17 Answers

1126

The correct syntax is the following:

```
<div [innerHTML]="theHtmlString"></div>
```

Works on `8.1.0`

Documentation Reference

edited Jul 17 at 14:03

answered Dec 22 '15 at 21:02

prolink007
22.9k 19 102 165

4 This is helpful: victorsavkin.com/post/119943127151/angular-2-template-syntax – malix Apr 4 '16 at 14:33

12 Is there any way I can force angular to run its binding on the elements of that innerHTML? I need to use an <a [router-link]="...">, and want to provide that from external html. – thouliha Apr 29 '16 at 22:04

4 @thouliha I would recommend starting a new post regarding your question. – prolink007 Jun 20 '16 at 20:35

4 It renders the string in my case, but does something to the markup. Seems to have stripped out attributes on markup . I'm on 2.4.6 – paqogomez Mar 20 '17 at 23:10

2 @paqogomez Yes, it strips out anything it deems unsafe – Juan Mendes Jan 17 '18 at 16:30

268

Angular 2.0.0 and Angular 4.0.0 final

For safe content just

```
<div [innerHTML]="myVal"></div>
```

DOMSanitizer

Potential unsafe HTML needs to be explicitly marked as trusted using Angulars DOM sanitizer so doesn't strip potentially unsafe parts of the content

```
<div [innerHTML]="myVal | safeHtml"></div>
```

with a pipe like

```
@Pipe({name: 'safeHtml'})
export class Safe {
  constructor(private sanitizer:DomSanitizer){}

  transform(style) {
    return this.sanitizer.bypassSecurityTrustHtml(style);
    //return this.sanitizer.bypassSecurityTrustStyle(style);
    // return this.sanitizer.bypassSecurityTrustXxx(style); - see docs
  }
}
```

See also In RC.1 some styles can't be added using binding syntax

And docs: https://angular.io/api/platform-browser/DomSanitizer

Security warning

Trusting user added HTML may pose a security risk. The before mentioned docs state:

> Calling any of the `bypassSecurityTrust...` APIs disables Angular's built-in sanitization for the value passed in. Carefully check and audit all values and code paths going into this call. Make sure any user data is appropriately escaped for this security context. For more detail, see the Security Guide.

Angular markup

Something like

```
class FooComponent {
  bar = 'bar';
  foo = `<div>{{bar}}</div>
    <my-comp></my-comp>
    <input [(ngModel)]="bar">`;
```

with

```
<div [innerHTML]="foo"></div>
```

won't cause Angular to process anything Angular-specific in `foo` . Angular replaces Angular specific markup at build time with generated code. Markup added at runtime **won't be processed by Angular**.

To add HTML that contains Angular-specific markup (property or value binding, components, directives, pipes, ...) it is required to add the dynamic module and compile components at runtime. This answer provides more details How can I use/create dynamic template to compile dynamic Component with Angular 2.0?

edited Feb 7 '18 at 9:52

answered Dec 11 '16 at 17:42

Günter Zöchbauer

358k 82 1135 1035

11 This should be the answer. Pay attention to the two lines that are commented out. It is actually the second one that handles HTML. – paqogomez Mar

20 '17 at 23:28

7 be sure to `import { BrowserModule, DomSanitizer } from '@angular/platform-browser'` – paqogomez Mar 20 '17 at 23:29

4 Also `import { Pipe } from '@angular/core'` – Appulus Mar 30 '17 at 9:37

1 This is the answer, right here! Was looking for the details about what in NG2 replaced $SCE of NG1. ;) – jrista Apr 4 '17 at 22:34

2 Great answer. Solved my issue. Thanks a lot. In case someone's not sure how to use the pipe in a component (like I was): angular.io/guide/pipes Just add it to your declarations in the corresponding module and voilá! – Alejandro Nagy Jun 23 '17 at 19:47

160

`[innerHtml]` is great option in most cases, but it fails with really large strings or when you need hard-coded styling in html.

I would like to share other approach:

All you need to do, is to create a div in your html file and give it some id:

```
<div #dataContainer></div>
```

Then, in your Angular 2 component, create reference to this object (TypeScript here):

```
import { Component, ViewChild, ElementRef } from '@angular/core';

@Component({
    templateUrl: "some html file"
})
export class MainPageComponent {

    @ViewChild('dataContainer') dataContainer: ElementRef;

    loadData(data) {
        this.dataContainer.nativeElement.innerHTML = data;
    }
}
```

Then simply use `loadData` function to append some text to html element.

It's just a way that you would do it using native javascript, but in Angular environment. I don't recommend it, because makes code more messy, but sometimes there is no other option.

See also Angular 2 - innerHTML styling

edited Jul 19 '17 at 8:17

Günter Zöchbauer
358k 82 1135 1035

answered Aug 8 '16 at 10:10

Piotrek
5,202 5 52 97

1 I don't see a difference to the other solutions except that yours accesses properties of `nativeElement` directly which is considered bad practice. I'm sure `[innerHTML]="..."` does the same under the hood but in good Angular2 practice way. – Günter Zöchbauer Aug 8 '16 at 10:15

1 That's not how Angular2 works. The HTML you add to templates of Angular2 components is first processed by Angular and only afterwards added to the DOM. Did you actually experience issues with `[innerHTML]` and large strings in Angular2? – Günter Zöchbauer Aug 8 '16 at 10:42

24 `[innerHtml]` removes styling hard-coded in the Html. In order to integrate a wysiwyg editor, I had to use the approach listed here. – Jony Adamit Aug 14 '16 at 6:11

1 This is useful for generating content that will go into an HTML email where inline styling is unfortunately still necessary. Other methods using interpolation removed the inline styles. – Frank Aug 31 '16 at 17:35

2 For me this solution worked for including an inline SVG document, while the `[innerHTML]` approach didn't. – Jared Phelps Nov 4 '16 at 7:32

44

On angular2@2.0.0-alpha.44:

Html-Binding will not work when using an `{{interpolation}}` , use an "Expression" instead:

invalid

```
<p [innerHTML]="{{item.anleser}}"></p>
```

-> throws an error (Interpolation instead of expected Expression)

correct

```
<p [innerHTML]="item.anleser"></p>
```

-> this is the correct way.

you may add additional elements to the expression, like:

```
<p [innerHTML]="'<b>'+item.anleser+'</b>'"></p>
```

hint

HTML added using `[innerHTML]` (or added dynamically by other means like `element.appenChild()` or similar) won't be processed by Angular in any way except sanitization for security purposed.
Such things work only when the HTML is added statically to a components template. If you need this, you can create a component at runtime like explained in [How can I use/create dynamic template to compile dynamic Component with Angular 2.0?]

edited Nov 27 '18 at 13:36

answered Dec 13 '15 at 23:39

jvoigt
753 8 15

1 Edited after trying out again. Solution found :) – jvoigt Dec 14 '15 at 8:01

2 The third example not working. The expression is not evaluate. The output is simply string... Any other way to combile trustedHTML with another tags elements ? – Kévin Vilela Pinto Dec 6 '16 at 8:41

23

Using [innerHTML] directly without using Angular's DOM sanitizer is not an option if it contains user-created content. The safeHtml pipe suggested by @GünterZöchbauer [in his answer] is one way of sanitizing the content. The following directive is another one:

```
import { Directive, ElementRef, Input, OnChanges, Sanitizer, SecurityContext,
  SimpleChanges } from '@angular/core';

// Sets the element's innerHTML to a sanitized version of [safeHtml]
@Directive({ selector: '[safeHtml]' })
export class HtmlDirective implements OnChanges {
  @Input() safeHtml: string;

  constructor(private elementRef: ElementRef, private sanitizer: Sanitizer) {}

  ngOnChanges(changes: SimpleChanges): any {
    if ('safeHtml' in changes) {
      this.elementRef.nativeElement.innerHTML =
        this.sanitizer.sanitize(SecurityContext.HTML, this.safeHtml);
    }
  }
}
```

To be used

```
<div [safeHtml]="myVal"></div>
```

edited May 23 '17 at 12:10

Community ♦
1 1

answered Dec 13 '16 at 12:20

Rene Hamburger
1,206 8 8

I tried to use this but am getting the following error `Can't bind to 'safeHtml' since it isn't a known property of 'div'.` ng-version 2.4.4 – LearnToday Feb 1 '17 at 17:08

@ObasiObenyOj you can still do that without the using of a separate directive if is a limited case, `constructor( private sanitizer: Sanitizer) {}` and bind the result into whatever you need, also the usage of ElementRef is strongly unsuggested. – Vale Steve Feb 3 '17 at 11:58

18

This works for me: `<div innerHTML = "{{ myVal }}"></div>` (Angular2, Alpha 33)

According to another SO: Inserting HTML from server into DOM with angular2 (general DOM manipulation in Angular2), "inner-html" is equivalent to "ng-bind-html" in Angular 1.X

edited May 23 '17 at 11:55

Community ♦
1 1

answered Aug 19 '15 at 20:10

Fan Li
441 3 10

1 This didn't work for me – Shervin Asgari Jun 14 '16 at 13:49

The correct way is without the {{ }}: – Christian Benseler Oct 5 '17 at 13:48

1 Use the [property] binding syntax instead of the {{interpolation}} – superluminary Oct 27 '17 at 9:29

11

Just to make for a complete answer, if your html content is in a component variable, you could also use:

```
<div [innerHTML]=componementVariableThatHasTheHtml></div>
```

answered Dec 31 '15 at 5:43

Serj Sagan
17.1k 10 112 130

9

I apologize if I am missing the point here, but I would like to recommend a different approach:

I think it's better to return raw data from your server side application and bind it to a template on the client side. This makes for more nimble requests since you're only returning json from your server.

To me it doesn't seem like it makes sense to use Angular if all you're doing is fetching html from the server and injecting it "as is" into the DOM.

I know Angular 1.x has an html binding, but I have not seen a counterpart in Angular 2.0 yet. They might add it later though. Anyway, I would still consider a data api for your Angular 2.0 app.

I have a few samples here with some simple data binding if you are interested: http://www.syntaxsuccess.com/viewarticle/angular-2.0-examples

answered Jul 21 '15 at 22:17

TGH
34.4k 8 82 119

After spending a few hours last night on this, I reached the same conclusion. – Aviad P. Jul 22 '15 at 5:33

25 There's definitely use cases where you'd want to fetch and display raw html. E.g. fetching a formatted piece of article from remote. – Alexander Chen Dec 24 '15 at 2:17

2 Another often-ignored scenario is protecting the business logic in the template, you sometimes don't want unauthorized users to see the logic you are using to display information, so you would rather prepare the view on server side – Ayyash Feb 20 '16 at 11:45

2 Also, displaying an HTML email, for example - fair point/question though! – a darren Feb 6 '17 at 9:00

1 If you are missing the point (which you seem to be by your own admission), then why post a response? Obviously the point of Angular is to use its view engine to bind and render the data. But considering the fact that there are countless applications where an Angular app might be used, it is actually feasible that one or two of them might have the requirement that some of the data that needs to be displayed in your application may already be formatted HTML, and it might just happen to be the case where the developer does not have control over that content. In other words... relevant question. – Gregor Nov 1 '17 at 5:50

4

Just simply use `[innerHTML]` attribute in your **HTML**, something like this below:

```
<div [innerHTML]="myVal"></div>
```

> Ever had properties in your component that contain some html markup or entities that you need to display in your template? The traditional interpolation won't work, but the innerHTML property binding comes to the rescue.

Using `{{myVal}}` **Does NOT** work as expected! This **won't** pick up the HTML tags like `<p>` , `<strong>` etc and pass it only as strings...

Imagine you have this code in your component:

```
const myVal:string ='<strong>Stackoverflow</strong> is <em>helpful!</em>'
```

If you use `{{myVal}}` , you will get this in the view:

```
<strong>Stackoverflow</strong> is <em>helpful!</em>
```

but using `[innerHTML]="myVal"` makes the result as expected like this:

Stackoverflow is *helpful!*

edited Oct 10 '17 at 12:31

answered Jun 20 '17 at 15:21

Alireza
59.2k 14 195 127

In **Angular 2** you can do 3 types of bindings:

1

- `[property]="expression"` -> Any html property can link to an expression. In this case, if expression changes property will update, but this doesn't work the other way.
- `(event)="expression"` -> When event activates execute expression.
- `[(ngModel)]="property"` -> Binds the property from js (or ts) to html. Any update on this property will be noticeable everywhere.

An expression can be a value, an attribute or a method. For example: '4', 'controller.var', 'getValue()'

Example here

edited May 23 '17 at 12:26

Community ♦
1 1

answered Feb 17 '17 at 11:13

adrisons
607 1 7 7

1

We can always pass html content to `innerHTML` property to render html dynamic content but that dynamic html content can be infected or malicious also. So before passing dynamic content to `innerHTML` we should always make sure the content is sanitized (using `DOMSanitizer`) so that we can escaped all malicious content.

Try below pipe:

```
import { Pipe, PipeTransform } from "@angular/core";
import { DomSanitizer } from "@angular/platform-browser";

@Pipe({name: 'safeHtml'})
export class SafeHtmlPipe implements PipeTransform {
    constructor(private sanitized: DomSanitizer) {
    }
    transform(value: string) {
        return this.sanitized.bypassSecurityTrustHtml(value);
    }
}

Usage:
<div [innerHTML]="content | safeHtml"></div>
```

answered Mar 4 at 9:58

Suneet Bansal
2,020 1 8 14

This is necessary even when you may think it isn't. For instance `style: background-color` of all things can get stripped out and so it's best to just start using this from the start or you'll get very confused later. – Simon_Weaver Jun 7 at 0:38

0

Working in AngularJS v2.1.1

```
<div [innerHTML]="variable or htmlString">
</div>
```

answered Nov 23 '16 at 7:05

FACode
389 5 17

2 This produces: `<div _ngcontent-luf-0=""></div>` for me. The `div` is empty. – Scott Marcus Nov 23 '16 at 15:54

0

The way to dynamically add elements to DOM, as explained on Angular 2 doc, is by using ViewContainerRef class from @Angular/core.

What you have to do is to declare a directive that will implement ViewContainerRef and act like a placeholder on your DOM.

Directive

```
import { Directive, ViewContainerRef } from '@angular/core';

@Directive({
  selector: '[appInject]'
})
export class InjectDirective {

  constructor(public viewContainerRef: ViewContainerRef) { }

}
```

Then, in the template where you want to inject the component:

HTML

```
<div class="where_you_want_to_inject">
  <ng-template appInject></ng-template>
</div>
```

Then, from the injected component code, you will inject the component containing the HTML you want:

```
import { Component, OnInit, ViewChild, ComponentFactoryResolver } from '@angular/core';
import { InjectDirective } from '../inject.directive';
import { InjectedComponent } from '../injected/injected.component';

@Component({
  selector: 'app-parent',
  templateUrl: './parent.component.html',
  styleUrls: ['./parent.component.css']
})
export class ParentComponent implements OnInit {

  @ViewChild(InjectDirective) injectComp: InjectDirective;
```

```
  constructor(private _componentFactoryResolver: ComponentFactoryResolver) {
  }

  ngOnInit() {
  }

  public addComp() {
    const componentFactory =
this._componentFactoryResolver.resolveComponentFactory(InjectedComponent);
    const viewContainerRef = this.injectComp.viewContainerRef;
    const componentRef = viewContainerRef.createComponent(componentFactory);
  }

  public removeComp() {
    const componentFactory =
this._componentFactoryResolver.resolveComponentFactory(InjectedComponent);
    const viewContainerRef = this.injectComp.viewContainerRef;
    const componentRef = viewContainerRef.remove();
  }

}
```

I added a fully working demo app on [Angular 2 dynamically add component to DOM demo]

edited Jun 29 '17 at 13:02

answered Jun 26 '17 at 20:13

BogdanC

1,667 2 16 15

0

You can use several approaches to achieve the solution. As already said in the approved answer, you can use:

```
<div [innerHTML]="myVal"></div>
```

depending on what you are trying to achieve, you can also try other things like javascript DOM (not recommended, DOM operations are slow):

Presentation

```
<div id="test"></test>
```

Component

```
var p = document.getElementsById("test");
p.outerHTML = myVal;
```

Property Binding

Javascript DOM Outer HTML

answered Feb 26 at 11:47

João Beirão
366 1 3 10

Regardless of whether DOM operations are slower than angular or not, doing it by using `getElementsById` or any other selection method is bad because it might capture elements belonging to completely different components if they contain elements with the same id (or other criteria). – Aviad P. Feb 26 at 12:57

Plus it completely performs outside of any angular-zone so changes will not be picked up. – Philipp Meissner Jun 13 at 10:05

0

If you want that in Angular 2 or Angular 4 and also want to keep inline CSS then you can use

```
<div [innerHTML]="theHtmlString | keepHtml"></div>
```

answered May 25 at 9:23

Jay Momaya
500 4 15

-2

If you have templates in your angular (or whatever framework) application, and you return HTML templates from your backend through a HTTP request/response, you are mixing up templates between the frontend and the backend.

Why not just leave the templating stuff either in the frontend (i would suggest that), or in the backend (pretty intransparent imo)?

And if you keep templates in the frontend, why not just respond with JSON for requests to the backend. You do not even have to implement a RESTful structure, but keeping templates on one side makes your code more transparent.

This will pay back when someone else has to cope with your code (or even you yourself are re-entering your own code after a while)!

If you do it right, you will have small components with small templates, and best of all, if your code is imba, someone who doesn't know coding languages will be able to understand your templates and your logic! So additionally, keep your functions/methods as small you can. You will eventually find out that maintaining, refactoring, reviewing, and adding features will be much easier compared to large functions/methods/classes and mixing up templating and logic between the frontend and the backend - and keep as much of the logic in the backend if your frontend needs to be more flexible (e.g. writing an android frontend or switching to a different frontend framework).

Philosophy, man :)

p.s.: you do not have to implement 100% clean code, because it is very expensive - especially if you have to motivate team members ;) but: you should find a good balance between an approach to cleaner code and what you have (maybe it is already pretty clean)

check the book if you can and let it enter your soul: https://de.wikipedia.org/wiki/Clean_Code

edited Aug 30 '17 at 23:40

answered Aug 30 '17 at 23:25

Guntram

659 10 14

yes, i saw @TGHs answer :) – Guntram Aug 30 '17 at 23:34

Sometimes it is necessary to get HTML from the server side when you are dealing with the old API like in SOAP. I was working on one my project with BSC (Bharat Stock Exchange) and they return the HTML code of the bank page while making payments. So you cannot change their API you have update your code accordingly. – Mahendra Waykos Apr 4 '18 at 10:59

You could write a middleware which frequently queries the soap api, and provide the extracted results in a socket for example. Consuming and extracting information via soap can be a pain. – Guntram Apr 12 '18 at 14:11

The really big obvious use case for raw markup is when your markup comes from a CMS and is written in a WYSIWYG editor. You might be serving multiple endpoints from a headless CMS, for example. This kind of thing is why every templating engine has an option for raw markup. – gburton Oct 16 '18 at 9:59

-3

The below code will help you

myval you can replace with the desired html

```
<div [innerHTML]="myVal"></div>
```

answered Jul 6 '17 at 10:16

Rahul Cv
440 3 9

4 This is the same as the accepted answer from over a year earlier. – Michael May 8 '18 at 14:22

protected by Community ♦ Oct 10 '16 at 18:09

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?