How and where to use ::ng-deep?

Asked 1 year, 11 months ago Active 17 days ago Viewed 50k times

35

I'm new to Angular 4, so could anyone please explain how and where to use `::ng-deep` in Angular 4?

Actually I want to overwrite some of the CSS properties of the child components from the parent components. Moreover is it supported on IE11?

css angular angular-template

10

edited May 9 at 9:26

Kamil Naja

2,519 3 14 27

asked Oct 17 '17 at 9:34

Jeyabalan Thavamani

469 2 9 23

Since `/deep/` and `::ng-deep` are both deprecated, I suggest you to take alook to this answer stackoverflow.com/a/49308475/2275011 and comments for more details and solutions. – Ferie May 21 at 10:32

5 Answers

44

Usually **`/deep/ "shadow-piercing"`** combinator can be used to force a style down to **`child components`** . This selector had an alias >>> and now has another one called ::ng-deep.

since **`/deep/ combinator`** has been deprecated, it is recommended to use `::ng-deep`

For example:

```
<div class="overview tab-pane" id="overview" role="tabpanel" [innerHTML]="project?.getContent( 'DETAILS')"></div>
```

and **`css`**

```
            &:last-child {
                margin-bottom: 0;
            }
        }
    }
}
```

it will be applied to child components

answered Oct 17 '17 at 9:42

Sajeetharan
143k 34 214 269

Is it support for IE11? – Jeyabalan Thavamani Oct 17 '17 at 9:48

1 I tried it on IE11 and is working – Umpa Apr 26 '18 at 8:11

2 Angular does the parsing of it - so you don't need to worry about compatibility. – Simon_Weaver Jun 14 '18 at 4:36

35

USAGE

`::ng-deep` , `>>>` and `/deep/` disable view encapsulation for specific CSS rules, in other words, it gives you access to DOM elements, which are not in your component's HTML. For example, if you're using Angular Material (or any other third-party library like this), some generated elements are outside of your component's area (such as **dialog**) and you can't access those elements directly or using a regular CSS way. If you want to change the styles of those elements, you can use one of those three things, for example:

```
::ng-deep .mat-dialog {
  /* styles here */
}
```

For now Angular team recommends making *"deep"* manipulations only with **EMULATED** view encapsulation.

DEPRECATION

"deep" manipulations are actually **deprecated** too, **BUT** it stills working for now, because Angular does pre-processing support (don't

Anyway, before following this way, I recommend you to take a look at *disabling view encapsulation* approach (which is not ideal too, it allows your styles to leak into other components), but in some cases, it's a better way. If you decided to disable view encapsulation, it's strongly recommended to use specific classes to avoid CSS rules intersection, and finally, avoid a mess in your stylesheets. It's really easy to disable right in the component's `.ts` file:

```
@Component({
  selector: '',
  template: '',
  styles: [''],
  encapsulation: ViewEncapsulation.None  // Use to disable CSS Encapsulation for this
component
})
```

You can find more info about the view encapsulation in **this** article.

edited Sep 22 at 13:57

answered Oct 17 '17 at 9:43

Commercial Suicide

10.8k 11 35 58

2 Disabling view encapsulation applies all CSS in your component globally. – Vedran Mar 25 at 16:04

5 Don't use `ViewEncapsulation.None` ! It will make a lot of damage by making those styles possible to leak into other components. – Alex Klaus May 4 at 12:12

@AlexKlaus, agree, that's why I mentioned in the answer, that it's not ideal. Actually, I used it just one time to apply shared repeatable styles to Angular Material components. If you try to disable encapsulation, you possibly will get a mess at some point. It's good to know about this option, but don't use it while you're not absolutely sure you need this. – Commercial Suicide May 4 at 14:58

14

Make sure not to miss the explanation of `:host-context` which is directly above `::ng-deep` in the angular guide : https://angular.io/guide/component-styles. Disclaimer: I missed it up until now and wish I'd seen it sooner.

`::ng-deep` is often necessary when you didn't write the component and don't have access to its source, but `:host-context` can be a very useful option when you do.

For example I have a black `<h1>` header inside a component I designed, and I want the ability to change it to white when it's displayed on a dark themed background.

```
.theme-dark widget-box ::ng-deep h1 { color: white; }
```

But instead with `:host-context` you can do this *inside* the component.

```
h1
{
    color: black;       // default color

    :host-context(.theme-dark) &
    {
        color: white;   // color for dark-theme
    }

    // OR set an attribute 'outside' with [attr.theme]="'dark'"

    :host-context([theme='dark']) &
    {
        color: white;   // color for dark-theme
    }
}
```

This will look anywhere in the component chain for `.theme-dark` and apply the css to the h1 if found. This is a good alternative to relying too much on `::ng-deep` which while often necessary is somewhat of an anti-pattern.

In this case the `&` is replaced by the `h1` (that's how sass/scss works) so you can define your 'normal' and themed/alternative css right next to each other which is very handy.

Be careful to get the correct number of `:` . For `::ng-deep` there are two and for `:host-context` only one.

edited Sep 10 '18 at 20:40

answered Jun 14 '18 at 5:08

Simon_Weaver
80.5k 68 497 552

You can also use `:host(.theme-dark)` if you don't want to inherit `theme-dark` from any parent components. This will entirely depend on your site css design. Also attributes can be very useful and can be combined in sophisticated ways in css alone `:host([theme='dark']:not([dayofweek='tuesday'))` – Simon_Weaver Aug 7 '18 at 23:22

Also note that this follows normal css rules, so if you have a component as described above (with host-context css) inside a container which has a `.theme-light` class this is in turn nested inside a container with `.theme-dark` it will still pick up the `theme-dark` and apply the css. But this is a great

3

I would emphasize the importance of limiting the `::ng-deep` to only children of a component by requiring the parent to be an encapsulated css class.

For this to work it's important to use the `::ng-deep` after the parent, not before otherwise it would apply to all the classes with the same name the moment the component is loaded.

Component css:

```
.my-component ::ng-deep .mat-checkbox-layout {
    background-color: aqua;
}
```

Component template:

```
<h1 class="my-component">
    <mat-checkbox ....></mat-checkbox>
</h1>
```

Resulting css:

```
.my-component[_ngcontent-c1] .mat-checkbox-layout {
    background-color: aqua;
}
```

edited Mar 26 at 12:30

answered Mar 26 at 11:56

Vedran

6,484 3 35 53

1

Just an update:

You should use `::ng-deep` instead of `/deep/` which seems to be deprecated.

Per documentation:

> The shadow-piercing descendant combinator is deprecated and support is being removed from major browsers and tools. As such we plan to drop support in Angular (for all 3 of /deep/, >>> and ::ng-deep). Until then ::ng-deep should be preferred for a broader compatibility with the tools.

You can find it here

edited Sep 21 '18 at 18:36

Dmitriy
5,121 11 20 35

answered Sep 21 '18 at 18:18

B

Balázs Takács
673 3 16

5 In this text it clearly says ::ng-deep is also deprecated: "we plan to drop support in Angular (for all 3 of /deep/, >>> and ::ng-deep)". – adripanico Dec 13 '18 at 8:40