# What is the cleanest way to disable CSS transition effects temporarily?

Asked 12 years, 6 months ago  Modified 1 year, 2 months ago  Viewed 311k times

266

I have a DOM element with this effect applied:

```
#elem {
  transition: height 0.4s ease;
}
```

I am writing a jQuery plugin that is resizing this element, I need to disable these effects temporarily so I can resize it smoothly.

What is the most elegant way of disabling these effects temporarily (and then re-enabling them), given they may be applied from parents or may not be applied at all.

javascript jquery css

Share
Improve this question
Follow

edited Jan 13, 2023 at 20:26

isherwood
60.9k ● 16 ● 120 ● 168

asked Jun 21, 2012 at 5:16

Sam Saffron
131k ● 81 ● 333 ● 510

## 12 Answers

Sorted by: Highest score (default)

596

## Short Answer

**Use this CSS:**

```
.notransition {
  -webkit-transition: none !important;
  -moz-transition: none !important;
  -o-transition: none !important;
  transition: none !important;
}
```

**Plus either this JS (without jQuery)...**

```
someElement.classList.add('notransition'); // Disable transitions
doWhateverCssChangesYouWant(someElement);
```

```
someElement.offsetHeight; // Trigger a reflow, flushing the CSS changes
someElement.classList.remove('notransition'); // Re-enable transitions
```

**Or this JS with jQuery...**

```
$someElement.addClass('notransition'); // Disable transitions
doWhateverCssChangesYouWant($someElement);
$someElement[0].offsetHeight; // Trigger a reflow, flushing the CSS changes
$someElement.removeClass('notransition'); // Re-enable transitions
```

... or equivalent code using whatever other library or framework you're working with.

## Explanation

This is actually a fairly subtle problem.

First up, you probably want to create a 'notransition' class that you can apply to elements to set their `*-transition` CSS attributes to `none`. For instance:

```
.notransition {
  -webkit-transition: none !important;
  -moz-transition: none !important;
  -o-transition: none !important;
  transition: none !important;
}
```

Some minor remarks on the CSS before moving on:

- These days you may not want to bother with the vendor-prefixed properties like `-webkit-transition`, or may have a CSS preprocessor that will add them for you. Specifying them manually was the right thing to do for most webapps when I first posted this answer in 2013, but as of 2023, per https://caniuse.com/mdn-css_properties_transition, only about 0.4% of users in the world are still using a browser that supports only a vendor-prefixed version of `transition`.
- There's no such thing as `-ms-transition`. The first version of Internet Explorer to support transitions *at all* was IE 10, which supported them unprefixed.
- This answer assumes that `!important` is enough to let this rule override your existing styles. But if you're *already* using `!important` on some of your `transition` rules, that might not work. In that case, you might need to instead do `someElement.style.setProperty("transition", "none", "important")` to disable the transitions (and figure out yourself how to revert that change).

Anyway, when you come to try and use this class, you'll run into a trap. The trap is that code like this won't work the way you might naively expect:

```
// Don't do things this way! It doesn't work!
someElement.classList.add('notransition')
someElement.style.height = '50px' // just an example; could be any CSS change
someElement.classList.remove('notransition')
```

Naively, you might think that the change in height won't be animated, because it happens while the 'notransition' class is applied. In reality, though, it *will* be animated, at least in all modern browsers I've tried. The problem is that the browser is buffering the styling changes that it needs to make until the JavaScript has finished executing, and then making all the changes in a single "reflow". As a result, it does a reflow where there is no net change to whether or not transitions are enabled, but there is a net change to the height. Consequently, it animates the height change.

You might think a reasonable and clean way to get around this would be to wrap the removal of the 'notransition' class in a 1ms timeout, like this:

```
// Don't do things this way! It STILL doesn't work!
someElement.classList.add('notransition')
someElement.style.height = '50px' // just an example; could be any CSS change
setTimeout(function () {someElement.classList.remove('notransition')}, 1);
```

but this doesn't reliably work either. I wasn't able to make the above code break in WebKit browsers, but on Firefox (on both slow and fast machines) you'll sometimes (seemingly at random) get the same behaviour as using the naive approach. I guess the reason for this is that it's possible for the JavaScript execution to be slow enough that the timeout function is waiting to execute by the time the browser is idle and would otherwise be thinking about doing an opportunistic reflow, and if that scenario happens, Firefox executes the queued function before the reflow.

The only solution I've found to the problem is to *force* a reflow of the element, flushing the CSS changes made to it, before removing the 'notransition' class. There are various ways to do this - see here for some. The closest thing there is to a 'standard' way of doing this is to read the `offsetHeight` property of the element.

One solution that actually works, then, is

```
someElement.classList.add('notransition'); // Disable transitions
doWhateverCssChangesYouWant(someElement);
someElement.offsetHeight; // Trigger a reflow, flushing the CSS changes
someElement.classList.remove('notransition'); // Re-enable transitions
```

Here's a JS fiddle that illustrates the three possible approaches I've described here (both the one successful approach and the two unsuccessful ones): http://jsfiddle.net/2uVAA/131/

Share

Improve this answer

Follow

edited Jan 22, 2023 at 8:27

answered May 15, 2013 at 22:08

Mark Amery

**154k** ● 89 ● 426 ● 469

11 Excelente answer. Good job, and appreciated since I came up with the same problem. What if I wanted to remove only one type of transition? – rafaelmorais Dec 16, 2014 at 11:44

5 Your solution is right, but for those looking for more background information: stackoverflow.com/a/31862081/1026 – Nickolay Aug 8, 2015 at 17:36

3 *Minor* minor aside, IE10 was the first *stable* version of IE to ship with unprefixed transitions. The pre-release versions, excluding the Release Preview, required the -ms- prefix for transitions, along with a slew of other things. That, I suspect, is one of the reasons the -ms- prefix appears today. The other, much more likely of course, reason is the usual cargo cult that we've all come to know and love about vendor prefixes. – BoltClock Feb 6, 2017 at 17:15

2 Interesting that simply checking the offset height of an element triggers a reflow. Wish I knew this a year ago when I was banging my head against the wall with this same problem. Is this still considered the most ideal way of handling this? – HaulinOats Apr 20, 2017 at 18:21

3 The reflow trick doesn't seem to work when animating SVG elements which don't have an offsetHeight attribute; setTimout works. – piotr_cz Aug 16, 2017 at 11:14

**26**

I would advocate disabling animation as suggested by DaneSoul, but making the switch global:

```
/*kill the transitions on any descendant elements of .notransition*/
.notransition * {
  transition: none !important;
}
```

`.notransition` can be then applied to the `body` element, effectively overriding any transition animation on the page:

```
$('body').toggleClass('notransition');
```

Share

Improve this answer

Follow

edited Jan 13, 2023 at 20:39

isherwood

**60.9k** ● 16 ● 120 ● 168

answered Jun 21, 2012 at 6:45

Oleg

**25k** ● 8 ● 64 ● 94

1 This is the right answer for situations where you just want to do all at once. For instance, I'm wanting to disable transitions while rotating on mobile and this is perfect for that. – Ben Lachman Feb 15, 2013 at 5:09

1 I ended up using this in angularjs, in a somewhat complicated scenario, but by god this was so helpful after days of banging my head. – Aditya M P Jul 25, 2013 at 20:15

3 Performance wise I cannot expect this to be a good idea at all. "Oh, just apply this to EVERYTHING on the page, that makes things easier!" – Lodewijk Aug 3, 2014 at 13:19

3 Should probably select both ".notransition" and ".notransition *" to be fully effective. – Nathan Dec 27, 2014 at 6:56

2 I recommend not using !important unless you absolutely have to. it can cause A LOT OF headaches later. I've tried this without it and as long as you follow CSS's cascade rules it still works fine. – Jordan Jul 19, 2021 at 13:18

---

22

Add an additional CSS class that blocks the transition, and then remove it to return to the previous state. This make both CSS and JQuery code short, simple and well understandable.

**CSS**:

```
.notransition {
  transition: none !important;
}
```

Note: `!important` was added to be sure that this rule will have higher preference, because using an ID is more specific than class.

**JQuery**:

```
$('#elem').addClass('notransition'); // to remove transition
$('#elem').removeClass('notransition'); // to return to previouse transition
```

Share

Improve this answer

Follow

edited Jan 13, 2023 at 20:38

isherwood

60.9k ● 16 ● 120 ● 168

answered Jun 21, 2012 at 5:31

DaneSoul

4,511 ● 3 ● 23 ● 37

7 -1 because this wasn't sufficient to solve the problem for me in Chrome or Firefox - if I have Javascript that adds the class, makes the changes, and removes the class, sometimes the changes *still* animate. I've spent the last hour or two studying this problem in some detail and will post an answer later with fiddles showing cases where the naive approach fails, plus a neat hack that fixes the solution here by forcing a reflow between making the changes and removing the 'notransition' class. – Mark Amery May 15, 2013 at 10:26

3 I recommend not using !important unless you absolutely have to. it can cause A LOT OF headaches later. I've tried this without it and as long as you follow CSS's cascade rules it still works fine. – Jordan Jul 19, 2021 at 13:18

17

For a pure JS solution (no CSS classes), just set the `transition` to `'none'`. To restore the transition as specified in the CSS, set the `transition` to an empty string.

```
// Remove the transition
elem.style.transition = 'none';

// Restore the transition
elem.style.transition = '';
```

If you're using vendor prefixes, you'll need to set those too.

```
elem.style.webkitTransition = 'none'
```

Share Improve this answer Follow

answered May 14, 2015 at 1:49

Thomas Higginbotham
**1,782** ● 20 ● 25

1 @Siwei, this is perfectly useful on a single element. If you have a bunch of elements having transitions, as I do in an image grid, it's less useful. Hence the tendency to make it reusable with class toggling. – isherwood Jan 13, 2023 at 20:41

15

You can disable animation, transition, transforms for all of element in page with this CSS code:

```
var style = document.createElement('style');
style.type = 'text/css';
style.innerHTML = '* {' +
'  transition-property: none !important;' +
'  transform: none !important;' +
'  animation: none !important;}';

document.getElementsByTagName('head')[0].appendChild(style);
```

Share

Improve this answer

Follow

edited Jan 13, 2023 at 20:42

isherwood
**60.9k** ● 16 ● 120 ● 168

answered Oct 7, 2015 at 9:12

Ali
**3,459** ● 6 ● 43 ● 56

2 Firstly this is awesome, thank you! Secondly there are also some other properties would should be set; see github.com/japgolly/test-state/blob/master/util/shared/src/test/… – Golly Oct 15, 2018 at 2:04

@Golly Those other properties could affect the final design – jonperl Nov 27, 2018 at 14:33

I think you could create a separate CSS class that you can use in these cases:

```
.disable-transition {
  transition: none;
}
```

Then in jQuery you would toggle the class like so:

```
$('#<your-element>').addClass('disable-transition');
```

Share

Improve this answer

Follow

edited Jan 13, 2023 at 20:41

isherwood

60.9k ● 16 ● 120 ● 168

answered Jun 21, 2012 at 5:30

Cyclonecode

29.9k ● 11 ● 75 ● 96

yes I think this is the best approach, in fact the plugin can inject that css block into the page – Sam Saffron Jun 21, 2012 at 5:32

3 Are you sure without !important CLASS ruler will override ID one? – DaneSoul Jun 21, 2012 at 5:33

If you want a simple no-jquery solution to prevent all transitions:

1. Add this CSS:

```
body.no-transition * {
  transition: none !important;
}
```

2. And then in your js:

```
document.body.classList.add("no-transition");

// do your work, and then either immediately remove the class:

document.body.classList.remove("no-transition");

// or, if browser rendering takes longer and you need to wait until a paint or
two:

setTimeout(() => document.body.classList.remove("no-transition"), 1);

// (try changing 1 to a larger value if the transition is still applying)
```

Share

answered Apr 23, 2020 at 4:58

community wiki

mrm

Improve this answer

Follow

I added an `animation: none !important;` as well, but this does the trick. Very handy for managing prefers-motion-reduced. – James Tomasino Aug 17, 2021 at 20:18

2

This is the workaround that worked easily for me. It isn't direct answer to the question but still may help someone.

Rather than creating `notransition` class which was supposed to cancel the transition

```
.notransition {
  -webkit-transition: none !important;
  -moz-transition: none !important;
  -o-transition: none !important;
  transition: none !important;
}
```

I created `moveTransition` class

```
.moveTransition {
      -webkit-transition: left 3s, top 3s;
      -moz-transition: left 3s, top 3s;
      -o-transition: left 3s, top 3s;
      transition: left 3s, top 3s;
}
```

Then I added this class to element with js

```
element.classList.add("moveTransition")
```

And later in setTimeout, I removed it

```
element.classList.remove("moveTransition")
```

I wasn't able to test it in different browsers but in chrome it works perfectly

Share Improve this answer Follow

answered Jul 26, 2020 at 18:02

Nika Tsogiaidze

1,097 ● 18 ● 19

2

Maybe not exactly what the OP wants, but if you want to kill all currently running animations programmatically, for example in some event handler (React, Angular, etc.), you could do this:

```
document.getAnimations().forEach((animation) => {
   animation.cancel();
});
```

Adding and then removing "animation suppressing" classes did not work for me. The animation started again as soon as I removed the suppressing class.

See Mozilla docs for more info

Share Improve this answer Follow

edited Oct 6, 2023 at 19:18

answered Oct 6, 2023 at 19:13

Zar Shardan

5,891 ● 2 ● 42 ● 40

1

If you want to remove CSS transitions, transformations and animations from the current webpage you can just execute this little script I wrote (inside your browsers console):

```
let filePath =
"https://dl.dropboxusercontent.com/s/ep1nzckmvgjq7jr/remove_transitions_from_pag
let html = `<link rel="stylesheet" type="text/css" href="${filePath}">`;
document.querySelector("html > head").insertAdjacentHTML("beforeend", html);
```

It uses vanillaJS to load this css-file. Heres also a github repo in case you want to use this in the context of a scraper (Ruby-Selenium): remove-CSS-animations-repo

Share Improve this answer Follow

answered Oct 25, 2019 at 13:02

dcts

1,629 ● 16 ● 34

-2

does

```
$('#elem').css('-webkit-transition','none !important');
```

in your js kill it?

obviously repeat for each.

Share Improve this answer Follow

answered Jun 21, 2012 at 5:25

Chris

716 ● 1 ● 6 ● 12

1 then you need to reset it after the fact, so you need to store it, which would lead to a fair amount of boiler plate – Sam Saffron Jun 21, 2012 at 5:28

---

-2

I'd have a class in your CSS like this:

```
.no-transition {
  -webkit-transition: none;
  -moz-transition: none;
  -o-transition: none;
  -ms-transition: none;
  transition: none;
}
```

and then in your jQuery:

```
$('#elem').addClass('no-transition'); //will disable it
$('#elem').removeClass('no-transition'); //will enable it
```

Share Improve this answer Follow

answered Jun 21, 2012 at 5:31

Moin Zaman

25.4k ● 6 ● 72 ● 74

1 Are you sure without !important CLASS ruler will override ID one? – DaneSoul Jun 21, 2012 at 5:33

1 Yes, it will because now you're targeting #elem.no-transition which is more specific than just the id – Moin Zaman Jun 21, 2012 at 13:04

4 @MoinZaman Erm, but you *haven't* targeted `#elem.no-transition` in your CSS, you've just targeted `.no-transition`. Perhaps you meant to write `#elem.no-transition` in your CSS? – Mark Amery May 15, 2013 at 9:41