A Tale of Animation Performance

css-tricks.com/tale-of-animation-performance/

Published December 20, 2012 by Chris Coyier

It's all started with a tweet:

> *Hey @chriscoyier, do you have any source confirming translate() is better than top/right/bottom/left for moving objects please? :)— Hugo Giraudel (@HugoGiraudel) December 10, 2012*

Context

So we're all on the same page, what Hugo means is, there are two different ways you can "move" elements.

- Give the element `relative`, `absolute`, or `fixed` positioning. Then you can use the `top`, `right`, `left`, `bottom` (or any combination therein) to move the objects around.
- Ensure the element had a `display` value of `block` or `inline-block` and then use the transform value `translate()`, `translateX()`, or `translateY()` to move the element.

"Better", v1

My first thought on what "better" meant in this context is which one is more appropriate to use under different circumstances. Which leads me to say: **"don't confuse positioning with design-y motion."**

Case in point. You have a button. You want to apply an effect to that button so that in it's `:active` state it nudges down 2 pixels to mimic a "pushed" effect. That is a design-y motion that should be done with `translate()`. You could do it with `top` or `bottom` and relative positioning, but then you are confusing the concepts of positioning and design-y motion.

Let's say somewhere else in the app you absolutely position a button (perfectly legit). Now when that `top: 2px;` gets applied to the button in it's :active state, that button will likely go zooming off someplace you didn't expect, possibly making the button unclickable.

Using `translate()` will always "nudge" the element from it's current position which is perfect for an effect like this, or really any design-specific motion.

"Better", v2

What Hugo was more likely getting at is **performance**. It has become common generic advice that using `translate()` to move elements has better performance than using top/right/bottom/left. But does the data hold up? Let's take a look.

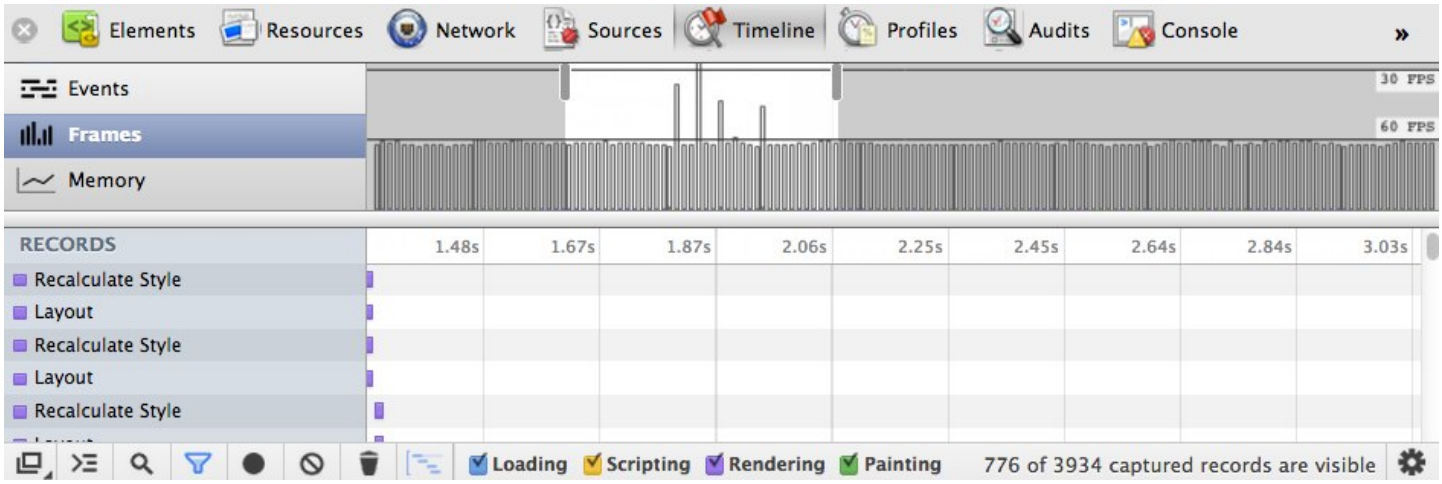
To start, I made a very simple animation of a red ball animating back and forth.

- Red ball animation: using top/left
- Red ball animation: using translate()

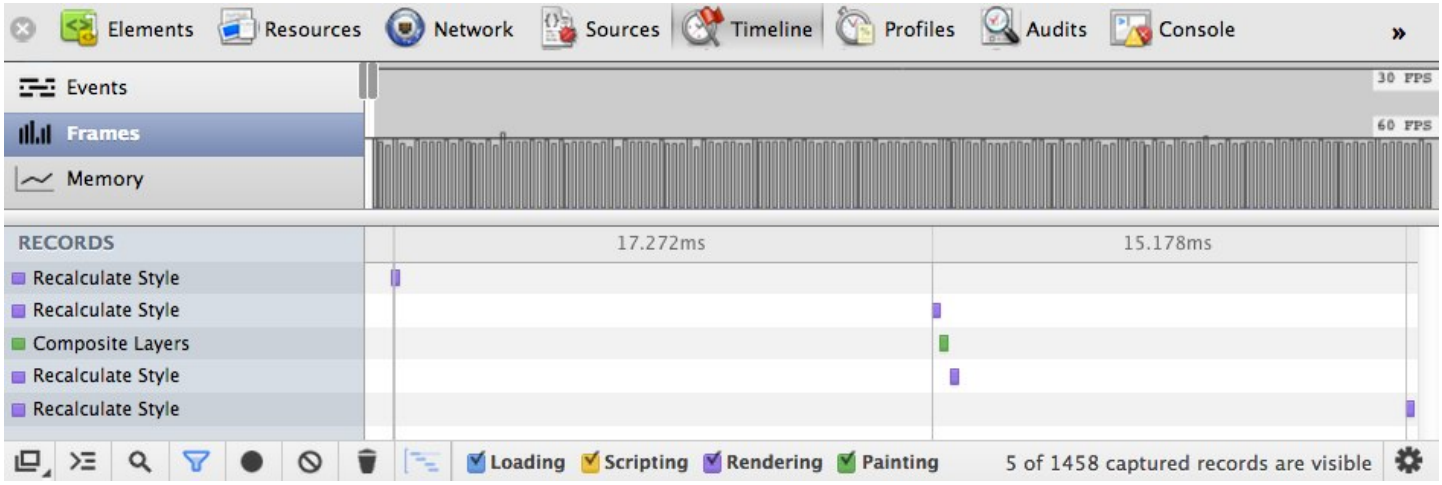
To my naked eye, in the latest stable Chrome (23) and a Retina MacBook Pro loaded with RAM, the `translate()` version does perform slightly better. I can see very slightly choppiness on the top/left version. Perception is important, but let's look at real data.

In the Chrome dev tools, you can click over to the "Timeline" tab, then onto "Frames", and then press the circle "Record" button along the bottom. With the animation running, start the recording for a little bit then stop it. The data is best if no other web sites are visible particularly if they are doing any animation.

In the top/left version, you can see occasional spikes above the line marked "60 FPS", or sixty frames per second. That's the line we try and stay under as that equates to perfectly smooth perceived motion.



Spikes above line == choppiness.



Smooth

But wait

This gets more complicated. I started talking with Paul Irish who works on the Google Chrome team. Paul thought that because these "paints" are so simple, it's probably not the best true performance indicator. Paul took my demos and upped the complexity. Instead of a red ball, Josh Hibberts MacBook Pro. Instead of a white background, Nate Eagle's upholstery.

- Heavy paint animation: using top/left
- Heavy paint animation: using translate()

Things got weird when **Paul's examples showed the reverse of what we expected.** The top/left version looks smoother than the translate() version. And surprisingly, the Frames timeline didn't show any big differences.

This is where Paul started to dig a lot deeper into this weird world. He came up with lots of interesting stuff! Paul talked to another Paul on the Chrome Team, Paul Lewis, who agreed that it is "smarter to use translate() for design-y motion." But went on to say that there is more to this than frame rate. With translate(), you get sub-pixel animation which is a kind of blurring between pixels that usually leads to smoother animation.

> *With subpixel [animations] you lose clarity, which is correct. The key is to hold for longer on rounded values at the extremes so your eyes get the satisfaction of clarity but the smoothness of motion.*

Paul Irish Digs Way Deeper

I'm gonna just pass this over to Paul. He did a 13-minute video you should watch. It explains all this fancy GPU stuff and subpixel rendering and all that.