Re-rastering Composited Layers on Scale Change

By Chris Harrelson
Chris is a contributor to Web**Fundamentals**

TL;DR

Starting in Chrome 53, all content is re-rastered when its transform scale changes, if it does not have the `will-change: transform` CSS property. In other words, `will-change: transform` means "please animate it fast".

This only applies to transforms scales that happen via script manipulation, and **does not apply to CSS animations or Web Animations**.

This means your site will likely get better-looking content, but it may also be slower without some simple changes outlined below.

Implications for web developers

Under this change, `will-change: transform` can be thought of as forcing the content to be rastered into a fixed bitmap, which subsequently never changes under transform updates. This allows developers to increase the speed of transform animations on that bitmap, such as moving it around, rotating or scaling it.

We do not distinguish between scale and translation transforms.

Recommended actions

Put `will-change: transform` on elements when you need very fast (in other words, 60fps) transform animation speeds, *and* it is expected that rastering the element at high quality on every frame is not fast enough. Otherwise, avoid `will-change: transform`.

To optimize the performance-quality tradeoff, you may want to add `will-change: transform` when animations begin and remove it when they end. Be aware, however, that there is often a large one-time performance cost to adding or removing `will-change: transform`.

Additional implementation considerations

Removing `will-change: transform` causes content to be re-rastered at a crisp scale, but only on the next animation frame (via `requestAnimationFrame`). Thus if you have a layer with `will-change: transform` on it and simply wish to trigger a re-raster but then continue animating, you must remove will-change: transform, then re-add it in a `requestAnimationFrame()` callback.

If at any time during an animation, you want to raster at the current scale, apply the above technique to remove in one frame, the re-add `will-change: transform` in a subsequent frame.

This may have the side-effect of the content losing its composited layer, causing the above recommendation to be somewhat expensive. If that is a problem, we recommend adding `transform: translateZ(0)` to the content as well to ensure it remains in a composited layer during this operation.

Summary of impact

This change has implications for rendered content quality, performance, and developer control.

- **Rendered content quality**: rendered output of elements which animate transform scale will always be crisp by default.
- **Performance**: animating transform when `will-change: transform` is present will be fast.
- **Developer control**: developers can choose between quality and speed, on a per-element and per-animation frame basis by adding and removing `will-change: transform`.

See the referenced design doc above for much more detail.

Examples

In this example, the element with the `remainsBlurry` ID will stay blurry after this change, but the element with the `noLongerBlurry` ID will become crisp. That is because the former has a `will- change: transform` CSS property on it.

Examples of transform scale animations from real applications

- From this bug: Zooming tiger
- Google Maps on mobile (maps.google.com) - zoom the map
- Google Maps Lite on desktop
- Ticketmaster seat map
