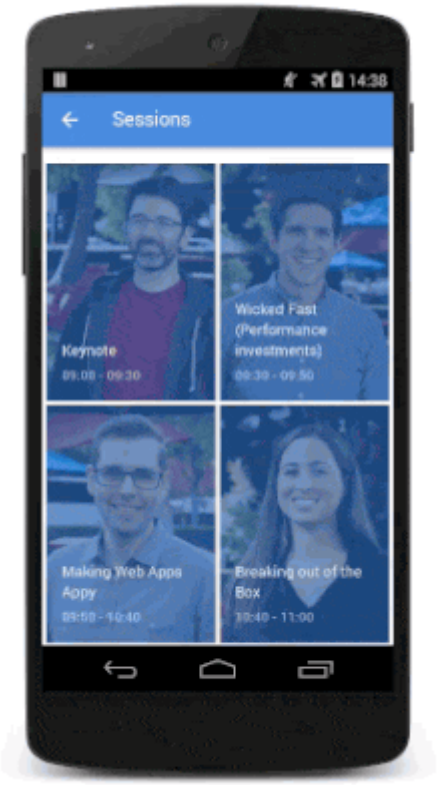
Chrome Dev Summit 2014: The Applied Science of Runtime Performance

By Paul Lewis
Paul is a Design and Perf Advocate



Late last year I built the Chrome Dev Summit site. I wanted it to have a Material Design look and feel, since it's a great design language, and I felt it would be a great fit for the kind of site I wanted to create. But, as with any new design language, there are questions, challenges, and decisions to take, and especially so when dealing with the conventions of the web.

One aspect of the site that was particularly challenging to create was the "takeover" effect when you click on a card.

Getting an effect like this to run at 60fps took some thinking, prototyping, and a few interesting compromises. At Chrome Dev Summit, I spoke about this effect and explained in

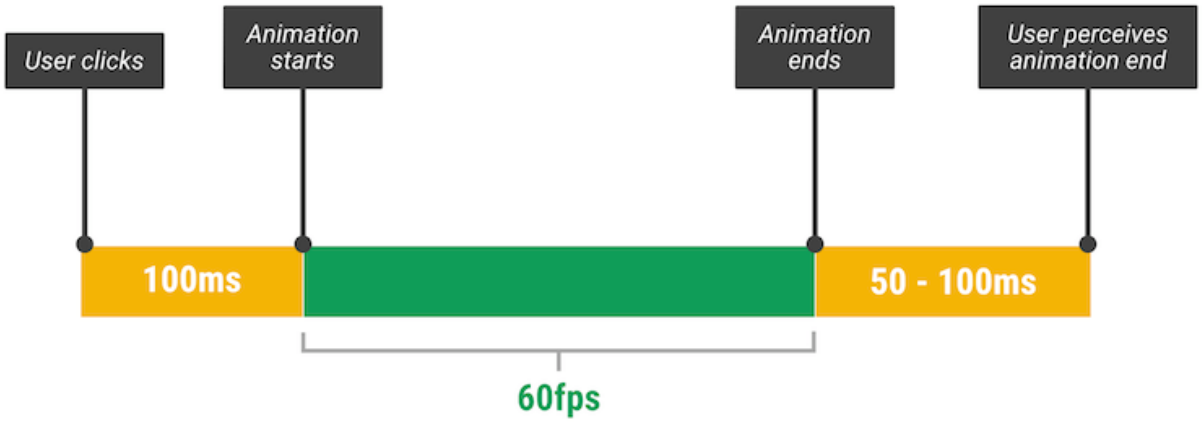
gory detail how I went about building it.

Building a high performance animation

High performance animations, today at least, are those that favor the browser's compositor. If you're able to stick to changing transform and opacity properties, typically you'll see great performance. The general approach I took to the card animation does just that:

1. Measure the position of all the elements in the card when the card is collapsed.
2. Toggle the card's classes to expand it (without animating).
3. Remeasure the position of the elements in the card now the card is expanded.
4. Calculate the differences.
5. Apply negative transforms to move the elements back to the start positions.
6. Switch on animations.
7. Remove the negative transforms and watch the elements whizz out to their final locations on screen.

All of this might sound expensive, but there is a window of 100ms from the moment a user interacts before the animation must start. Any more than this and the user will perceive a delay. You can use this time to do expensive preparatory work so that you can run more cheaply during the animation itself. There is also a window at the end of about 50-100ms to do tidy up work, which may be handy depending on what you're trying to do.



The expensive work to do the animation is done inside that first 100ms and, on a Nexus 5 the work takes something in the region of 70ms, so there's room to spare.

Get the code

If you're interested in looking at the site in more detail you'll be pleased to hear that the code has been released on GitHub, so go and take a look!

Last updated July 2, 2018.