Chrome Dev Summit: Performance Summary

By Paul Lewis

Paul is a Design and Perf Advocate

#perfmatters: Tooling techniques for the performance ninja

Knowing your way around your development tools is key to becoming a performance Grand Master. Colt stepped through the three pillars of performance: network, compute and render, providing a tour of the key problem in each area and the tools available for finding and eradicating them.

Slides

- You can now profile Chrome on Android with the DevTools you know and love from desktop.
- The iteration loop for performance work is: gather data, achieve insight, take action.
- Prioritize assets that are on the critical rendering path for your pages.
- Avoid painting; it's super expensive.
- Avoid memory churn and executing code during critical times in your app.

#perfmatters: Optimizing network performance

Network and latency typically accounts for 70% of a site's total page load time. That's a large percentage, but it also means that any improvements you make there will reap huge benefits for your users. In this talk Ilya stepped through recent changes in Chrome that will improve loading time, as well as a few changes you can make in your environment to help keep network load to an absolute minimum.

Slides

- Chrome M27 has a new and improved resource scheduler.
- Chrome M28 has made SPDY sites (even) faster.
- Chrome's simple cache has received an overhaul.
- SPDY / HTTP/2.0 offer huge transfer speed improvements. There are mature SPDY modules available for nginx, Apache and Jetty (to name just three).
- QUIC is a new and experimental protocol built on top of UDP; it's early days but however it works out users will win.

#perfmatters: 60fps layout and rendering

Hitting 60fps in your projects directly correlates to user engagement and is crucial to its success. In this talk Nat and Tom talked about Chrome's rendering pipeline, some common causes of dropped frames and how to avoid them.

Slides

- A frame is 16ms long. It contains JavaScript, style calculations, painting and compositing.

- Painting is *extremely* expensive. A Paint Storm is where you unnecessarily repeat expensive paint work.
- Layers are used to cache painted elements.
- Input handlers (touch and mousewheel listeners) can kill responsiveness; avoid them if you can. Where you can't keep them to a minimum.

#perfmatters: Instant mobile web apps

The Critical Rendering Path refers to anything (JavaScript, HTML, CSS, images) that the browser requires before it is able to begin painting the page. Prioritizing the delivery of assets on the critical rendering path is a must, particularly for users on network-constrained devices such as smartphones on cellular networks. Bryan talked through how the team at Google went through the process of identifying and prioritizing the assets for the PageSpeed Insights website, taking it from a 20 second load time to just over 1 second!

Slides

- Eliminate render-blocking JavaScript and CSS.
- Prioritize visible content.
- Load scripts asynchronously.
- Render the initial view server-side as HTML and augment with JavaScript.
- Minimize render-blocking CSS; deliver only the styles needed to display the initial viewport, then deliver the rest.
- Large data URIs inlined in render-blocking CSS are harmful for render performance; they are blocking resources where image URLs are non-blocking.

Last updated July 2, 2018.