Flexbox layout isn't slow

By Paul Irish

Paul is a contributor to Web**Fundamentals**

Success: Old flexbox (`display: box`) is 2.3x **slower** than new flexbox (`display: flex`).

A bit ago, Wilson Page wrote a great article for Smashing Magazine digging into how they brought the Financial Times web app to life. In the article, Wilson notes:

> As the app began to grow, we found performance was getting worse and worse.
>
> We spent a good few hours in Chrome Developers Tools' timeline and found the culprit: Shock, horror! — it was our new best friend, flexbox. The timeline showed that some layouts were taking close to 100 milliseconds; reworking our layouts without flexbox reduced this to 10 milliseconds!

Wilson's comments were about the original (legacy) flexbox that used `display: box;`. Unfortunately they never got a chance to answer if the newer flexbox (`display: flex;`) was faster, but over on CSS Tricks, Chris Coyier opened that question.

We asked Ojan Vafai, who wrote much of the implementation in WebKit & Blink, about the newer flexbox model and implementation.

> The new flexbox code has a lot fewer multi-pass layout codepaths. You can still hit multi-pass codepaths pretty easily though (e.g. `flex-align: stretch` is often 2-pass). In general, it should be much faster in the common case, but you can construct a case where it's equally as slow.
>
> That said, if you can get away with it, regular block layout (non-float), will usually be as fast or faster than new flexbox since it's always single-pass. But new flexbox should be faster than using tables or writing custom JS-base layout code.

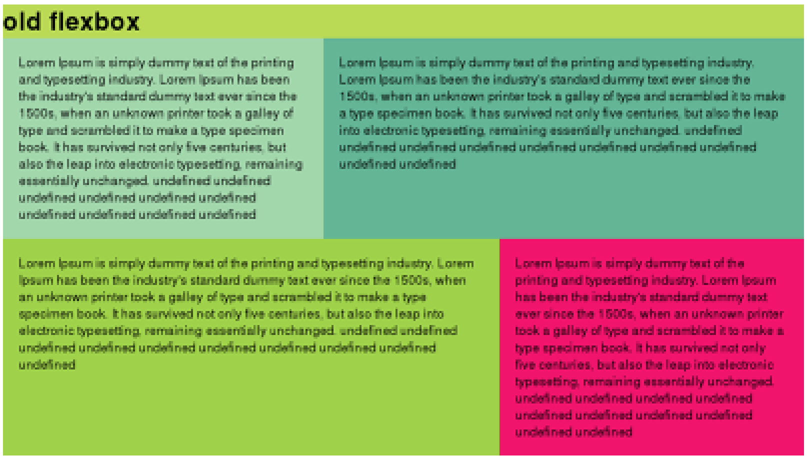
To see the difference in numbers, I made a head-to-head comparison of old v new syntax.

Old v New Flexbox Benchmark

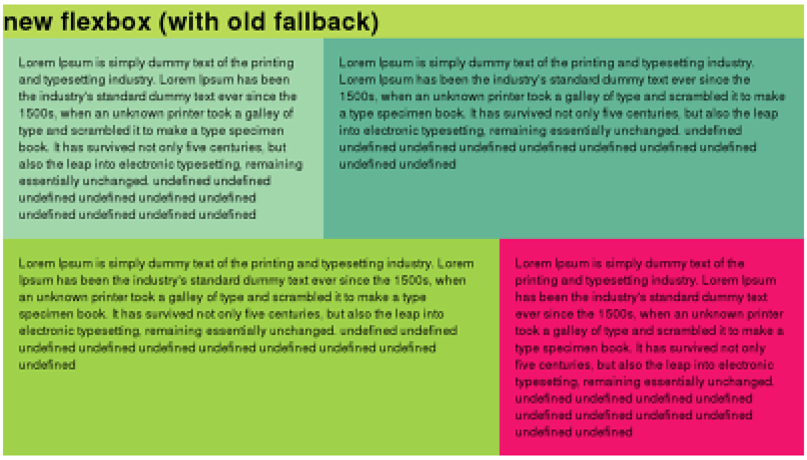
- old flexbox vs new flexbox (with fallback)

- 500 elements per page
- evaluating page load cost to lay out the elements
- averaged across 3 runs each
- measured on desktop. (mobile will be ~10x slower)

Old flexbox: layout costs of *~43.5ms*



New flexbox: layout costs of *~18.2ms*



Summary: Old is 2.3x slower than new.

What should you do?

When using flexbox, always author for the new stuff: the IE10 tweener syntax and the new updated flexbox that's in Chrome 21+, Safari 7+, Firefox 22+, Opera (& Opera Mobile) 12.1+,

IE 11+, and Blackberry 10+. In many cases you can do a fallback to the legacy flexbox to pick up some older mobile browsers.

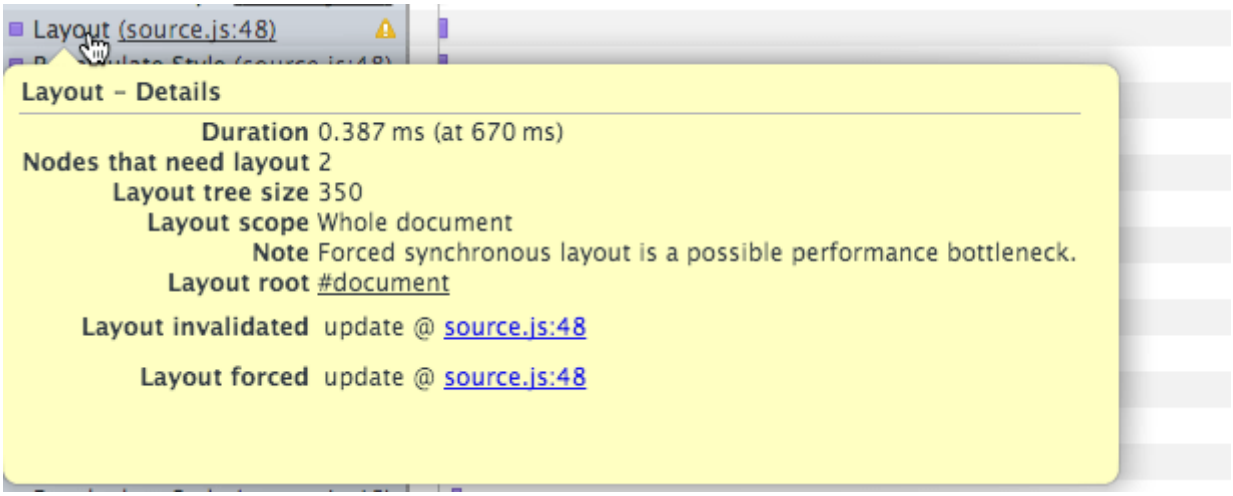
Notes:

- I also ran the benchmark using `display:table-cell` and it hit 30ms, right between the two flexbox implementations.
- The benchmarks above only represent the Blink & WebKit side of things. Due to the time of implementation, flexbox is nearly identical across Safari, Chrome & Android.

Remember: Tools, not rules

What's more important is optimizing what matters. Always use the timeline to identify your bottlenecks before spending time optimizing one sort of operation.

In fact, we've connected with Wilson and the Financial Times Labs team and, as a result, improved the Chrome DevTools coverage of layout performance tooling. We'll soon be adding the ability to view the relayout boundary of an element, and Layout events in the timeline are loaded with details of the scope, root, and cost of each layout:



Last updated July 2, 2018.