What forces layout/reflow. The comprehensive list.

what-forces-layout.md

What forces layout / reflow

All of the below properties or methods, when requested/called in JavaScript, will trigger the browser to synchronously calculate the style and layout*. This is also called reflow or layout thrashing, and is common performance bottleneck.

Element

Box metrics

- `elem.offsetLeft`, `elem.offsetTop`, `elem.offsetWidth`, `elem.offsetHeight`, `elem.offsetParent`
- `elem.clientLeft`, `elem.clientTop`, `elem.clientWidth`, `elem.clientHeight`
- `elem.getClientRects()`, `elem.getBoundingClientRect()`

Scroll stuff

- `elem.scrollBy()`, `elem.scrollTo()`
- `elem.scrollIntoView()`, `elem.scrollIntoViewIfNeeded()`
- `elem.scrollWidth`, `elem.scrollHeight`
- `elem.scrollLeft`, `elem.scrollTop` also, setting them

Focus

- `elem.focus()` can trigger a *double* forced layout (source&l=2923)

Also...

- `elem.computedRole`, `elem.computedName`
- `elem.innerText` (source&l=3440))

getComputedStyle

`window.getComputedStyle()` will typically force style recalc

`window.getComputedStyle()` will force layout, as well, if any of the following is true:

1. The element is in a shadow tree
2. There are media queries (viewport-related ones). Specifically, one of the following: (source)

- `min-width`, `min-height`, `max-width`, `max-height`, `width`, `height`
- `aspect-ratio`, `min-aspect-ratio`, `max-aspect-ratio`
- `device-pixel-ratio`, `resolution`, `orientation`, `min-device-pixel-ratio`, `max-device-pixel-ratio`

1. The property requested is one of the following: (source)

- `height`, `width`
- `top`, `right`, `bottom`, `left`
- `margin` [`-top`, `-right`, `-bottom`, `-left`, or *shorthand*] only if the margin is fixed.
- `padding` [`-top`, `-right`, `-bottom`, `-left`, or *shorthand*] only if the padding is fixed.
- `transform`, `transform-origin`, `perspective-origin`
- `translate`, `rotate`, `scale`
- `grid`, `grid-template`, `grid-template-columns`, `grid-template-rows`

- `perspective-origin`
- These items were previously in the list but appear to not be any longer (as of Feb 2018): `motion-path`, `motion-offset`, `motion-rotation`, `x`, `y`, `rx`, `ry`

window

- `window.scrollX`, `window.scrollY`
- `window.innerHeight`, `window.innerWidth`
- `window.getMatchedCSSRules()` only forces style

Forms

- `inputElem.focus()`
- `inputElem.select()`, `textareaElem.select()`

Mouse events

- `mouseEvt.layerX`, `mouseEvt.layerY`, `mouseEvt.offsetX`, `mouseEvt.offsetY` (source)

document

- `doc.scrollingElement` only forces style

Range

- `range.getClientRects()`, `range.getBoundingClientRect()`

SVG

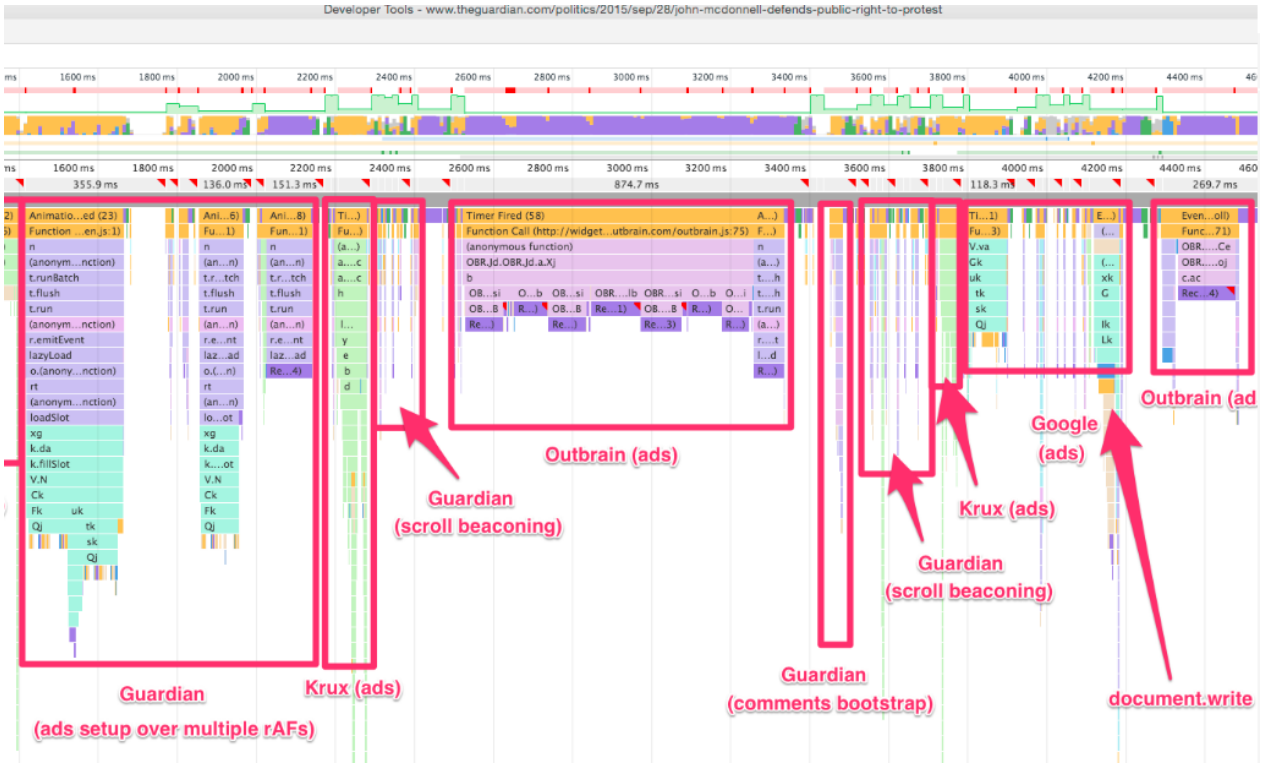
- Quite a lot; haven't made an exhaustive list, but Tony Gentilcore's 2011 Layout Triggering List pointed to a few.

contenteditable

- Lots & lots of stuff, ...including copying an image to clipboard (source)

*Appendix

- Reflow only has a cost if the document has changed and invalidated the style or layout. Typically, this is because the DOM was changed (classes modified, nodes added/removed, even adding a psuedo-class like :focus).
- If layout is forced, style must be recalculated first. So forced layout triggers both operations. Their costs are very dependent on the content/situation, but typically both operations are similar in cost.
- What should you do about all this? Well, the `More on forced layout` section below covers everything in more detail, but the short version is:
  i. `for` loops that force layout & change the DOM are the worst, avoid them.
  ii. Use DevTools Timeline to see where this happens. You may be surprised to see how often your app code and library code hits this.
  iii. Batch your writes & reads to the DOM (via FastDOM or a virtual DOM implementation). Read your metrics at the begininng of the frame (very very start of `rAF`, scroll handler, etc), when the numbers are still identical to the last time layout was done.



Timeline trace of The Guardian. Outbrain is forcing layout repeatedly, probably in a loop.

Cross-browser

- The above data was built by reading the Blink source, so it's true for Chrome, Opera, and most android browsers.
- Tony Gentilcore's Layout Triggering List was for 2011 WebKit and generally aligns with the above.
- Modern WebKit's instances of forced layout are mostly consistent: `updateLayoutIgnorePendingStylesheets` - GitHub search - WebKit/WebKit
- Gecko's reflow appears to be requested via FrameNeedsReflow. Results: `FrameNeedsReflow` - mozilla-central search
- No concrete data on Edge/IE, but it should fall roughly in line, as the return values for these properties are spec'd. What would differ is the amount of clever optimization.

Browsing the Chromium source:

- forced layout (and style recalc): `UpdateStyleAndLayoutIgnorePendingStylesheets` - Chromium Code Search
- forced style recalc: `UpdateStyleAndLayoutTreeIgnorePendingStylesheets` - Chromium Code Search

CSS Triggers

CSS Triggers is a related resource and all about what operations are required to happen in the browser lifecycle as a result of setting/changing a given CSS value. It's a great resource. The above list, however, are all about what forces the purple/green/darkgreen circles synchronously from JavaScript.

More on forced layout

- Avoiding layout thrashing — Web Fundamentals
- Fixing Layout thrashing in the real world | Matt Andrews
- Timeline demo: Diagnosing forced synchronous layouts - Google Chrome
- Preventing 'layout thrashing' | Wilson Page
- wilsonpage/fastdom
- Rendering: repaint, reflow/relayout, restyle / Stoyan
- We spent a week making Trello boards load extremely fast. Here's how we did it. - Fog Creek Blog
- Minimizing browser reflow | PageSpeed Insights | Google Developers
- Optimizing Web Content in UIWebViews and Websites on iOS
- Accelerated Rendering in Chrome
- web performance for the curious
- Jank Free

Updated slightly Feb 2018. Codesearch links and a few changes to relevant element properties.

Theodeus commented on 19 Sep 2015

Mouse event offsets. Ouch.

pikharov commented on 19 Sep 2015

Looks like it can be easier to write down a list of stuff that wouldn't trigger a reflow. Like, you know, safe colors.

tobireif commented on 19 Sep 2015

I hope Chrome can be improved so that the list becomes shorter.

d0mme commented on 19 Sep 2015

👍

kutec commented on 19 Sep 2015

Oh longer but awesome! 👍

Thanks for creating.

webdesignberlin commented on 19 Sep 2015

Pretty long. Thanks for creating.

jakearchibald commented on 19 Sep 2015

@tobireif which of these do you consider to be bugs?

mohsen1 commented on 19 Sep 2015

What about CSS transition and animations?

kuka commented on 20 Sep 2015

Thanks for the list Paul!

jlukic commented on 20 Sep 2015

Will keep in mind, thank you.

unbug commented on 20 Sep 2015

@paulirish I added some links to my fork,can you review them?

t0lkman commented on 20 Sep 2015

great work **@paulirish**

rafaeleyng commented on 21 Sep 2015

Very nice list, thanks.

glenn-allen commented on 21 Sep 2015

@paulirish, This is great, thanks!
Regarding focus: Given it's an important part of accessibility but it appears to be performance bottleneck (double forced layout) do you have any recommendations on how to avoid or reduce its impact while maintaining accessibility (beyond simply avoiding layout thrashing)?
On dynamic pages with transitions it seems incredibly difficult & problematic, however there doesn't seem to be much written about it in the community.

jt3k commented on 21 Sep 2015

ok

jenshedqvist commented on 21 Sep 2015

Ok, so basically what the browser and DOM tells us is "**don't touch me!**". And it's our job to respect that and optimize where we hurt it and the user attached to it. Prioritize, cache/store, debounce etc.

danburzo commented on 21 Sep 2015

You know what would be cool? A Sublime Text plugin to highlight all these in JavaScript files.
I'm looking into how this might be done.

njoyard commented on 21 Sep 2015

@paulirish unfortunately, this is only a list of what *may* cause a reflow. In my case (preparing a complex print layout using CSS columns), I actually *need* reflows sometimes (mainly after splitting table/ul elements) and the list you present above is not accurate in that perspective. Some research should be done as to whether those *always* cause a reflow, or if they need specific conditions.

akost commented on 22 Sep 2015

Thanks!

simevidas commented on 22 Sep 2015

@paulirish If I change the DOM (e.g. set a class on an element), but *don't* force a reflow in my code, when will the reflow happen? Before or after the `requestAnimationFrame` callback? This demo suggests before. I thought the whole purpose of rAF is to update the DOM, so why did the browser reflow before, when it is expected that the rAF callback will make another reflow necessary. What am I missing?

Update: Nevermind. `getComputedStyle` forced the reflow. My bad. But if I didn't force it within the rAF callback, would it have happened by then? (Not sure if onto something, or asking a stupid question.)

jakearchibald commented on 22 Sep 2015

@simevidas it depends on when you changed the dom.

See https://jakearchibald.com/2015/tasks-microtasks-queues-and-schedules/ for how the event loop works - rendering is part of this, and raf is called just before a natural style calc, layout, and paint.

So, if you make dom changes as part of a task, then read layout/style values in another task, you *may* trigger a sync calc/layout, because the browser doesn't always render in between tasks, as the loop can run faster than 60hz.

Similarly, if you make a dom change in a task, then read in a raf, you'll get a sync layout.

Ideally, treat your tasks as layout/style read only, and do your writes within raf.

jpmedley commented on 22 Sep 2015

Nice work, Paul.

mayankchd commented on 23 Sep 2015

In the case of reflow does the whole dom is recalculated or only the element that is changed ?

adardesign commented on 24 Sep 2015

Very helpful and a step closer to a smoother and jank freeer experience
Thanks!

ephraimtaback... commented on 24 Sep 2015

computeRelativePosition() is only called if it is dirty. If the value has already been computed it will not be called again. Instead, it will be read from the cached m_layerLocation.

mayrop commented on 25 Sep 2015

Very helpful, thank you!

SpoBo commented on 28 Sep 2015

Can someone make a linter for this?

paulirish commented on 28 Sep 2015 Author Owner

> **@paulirish** I added some links to my fork,can you review them?

@unbug merged those links back in. thank you!

> Regarding focus: Given it's an important part of accessibility but it appears to be performance bottleneck (double forced layout) do you have any recommendations on how to avoid or reduce its impact while maintaining accessibility (beyond simply avoiding layout thrashing)?

@glenn-allen, TBH this `focus` discovery above was unexpected and new to me. It requires some more investigation. The most concrete thing I can offer right now is... If you need to set focus(), do it at the start of the event handler, before any changes to the DOM.

> unfortunately, this is only a list of what *may* cause a reflow.

@njoyard, as long as the DOM is dirty, what I listed above will def cause a reflow. Whether the browser paints immediately after isn't guaranteed, however.

> In the case of reflow does the whole dom is recalculated or only the element that is changed ?

@mayankchd browsers try to be smart about it, but typically (probably ~75% of the time) it's the whole DOM. A "layout boundary" or "relayout root" can help to contain the layout: http://wilsonpage.co.uk/introducing-layout-boundaries/ It's effective, but requires a fixed width and height.

> Very helpful, thank you!

@mayrop my pleasure. :)

Lewiscowles1986 commented on 29 Sep 2015

@paulirish this is just JS triggered changes or would CSS be the same / worse?

tigt commented on 29 Sep 2015

@paulirish So if we need to scroll to a particular element, is setting the hash the most performant option?

JoeMilsom commented on 29 Sep 2015

Think window.pageXOffset/window.pageYOffset causes a relayout as well. At least in Chrome.

glenn-allen commented on 29 Sep 2015

@paulirish - Thanks for the response, greatly appreciated!

fregante commented on 29 Sep 2015

Can somebody explain to me why `window.innerHeight` and `window.innerWidth` would force a layout? They don't event depend on the DOM.

aFarkas commented on 30 Sep 2015

Please developers, don't be scared of those methods and properties. In fact those work extremely fast and can be called/accessed a "million time" without any harm. The problem is not whether you use them, but when/how.

> Ideally, treat your tasks as layout/style read only, and do your writes within raf.

The following little function might help to organize your code, while doing this read/write separation. It's similar to `Function.prototype.bind`:

```
/*
* returns a function, that is wrapped in a requestAnimationFrame. Useful to separate read from write.
* While all reads can happen where needed (but never inside a rAF) any DOM writes should happen inside a rAF.
* @param {function} fn - function that should be wrapped in a rAF
* @param {object} [options] - options
*   @param {object} [options.that] - the this context in which the function should be invoked (binds that to the function)
*   @param {object} [options.batch] - whether multiple calls to the function during one frame should be let through or sho
*
*   example usage:
 class MyWidget {
     constructor(element){
         this.element = element;
         this.changeWidget = writeFn(this.changeWidget);

         this.checkWidget();
         //add resize events and do more useful things....
     }

     changeWidget(add){
        this.element.classList[add ? 'add' : 'remove']('is-large-widget');
     }

     checkWidget(){
        this.changeWidget(this.element.offsetWidth > 600);
     }
 }
*/

var writeFn = function(fn, options){
    var running, args, that;
    var batchStack = [];
    var run = function(){
        running = false;
        if(options.batch){
            while(batchStack.length){
                args = batchStack.shift();
                fn.apply(args[0], args[1]);
```

```
            }
        } else {
            fn.apply(that, args);
        }
    };

    if(!options){
        options = {};
    }

    return function(){
        args = arguments;
        that = options.that || this;
        if(options.batch){
            batchStack.push([that, args]);
        }

        if(!running){
            running = true;
            requestAnimationFrame(run);
        }
    };
};
```

napengam commented on 30 Sep 2015

Good to know, thanks !

However I don't care because systems are getting faster and faster all the time.
As long as the responds time is acceptable for and by the user , it is all fine.

BTW. what about

obj.style.display
obj.style.visibility
obj.style.position

pvolyntsev commented on 7 Oct 2015

The changes that achieve silky smooth animations

As quote from the article http://www.html5rocks.com/en/tutorials/speed/high-performance-animations/

> Today transforms are the best properties to animate because the GPU can assist with the heavy lifting, so where you can limit your animations to these, do so.
>
> *opacity* opacity: 0...1 ;
> *position* transform: translate(_n_px, _n_px) ;
> *rotate* transform: rotate(_n_deg) ;
> *scale* transform: scale(*n*) ;

P.S. I see, it's yours :)

aFarkas commented on 30 Nov 2015

@paulirish

The cause of the forced layout of `focus` is, that focus first invalidates layout (by applying `:focus` styles) and then invoking the `scrollIntoView` algorithm.

The later is also a pain if you want to create accessible and animated UI components (see: http://allyjs.io/api/when/focusable.html). Maybe a new API that sets `focus` without invoking the `scrollIntoView` algorithm would be great.

Cristy94 commented on 3 Dec 2015

> What about CSS transition and animations?

+1

Why do CSS transitions on the **transform** property trigger Layout? Makes no sense.
http://jsfiddle.net/cygvaubo/1/

Without the transition everything seems fine.

LE: Adding **will-change** seems to fix this issue: http://jsfiddle.net/cygvaubo/3/ but text quality drops.

dmnd commented on 9 Dec 2015

> Can somebody explain to me why window.innerHeight and window.innerWidth would force a layout? They don't event depend on the DOM.

@bfred-it `window.innerHeight` and `window.innerWidth` depend on whether or not a scrollbar is visible, so they do depend on the DOM. (That's an assumption — someone more knowledgable please confirm or contradict)

pygy commented on 10 Feb 2016

@paulirish What do you mean by *"The element is in a shadow tree"*?

Edit: As in shadow DOM?

riskers commented on 6 Apr 2016

how to use timeline check dom is repint or reflow?

yardfarmer commented on 19 Apr 2016

√

bvaughn commented on 20 Jun 2016

> Regarding focus: Given it's an important part of accessibility but it appears to be performance bottleneck (double forced layout) do you have any recommendations on how to avoid or reduce its impact while maintaining accessibility (beyond simply avoiding layout thrashing)?
>
> **@glenn-allen**, TBH this focus discovery above was unexpected and new to me. It requires some more investigation. The most concrete thing I can offer right now is... If you need to set focus(), do it at the start of the event handler, before any changes to the DOM.

@paulirish: It's been a few months since you left this comment. Don't suppose you have more insight or suggestions for dealing with it? Recently ran into this snag when trying to optimize scrolling frame rate for react-virtualized. A focused grid repaints entirely on-scroll whereas an unfocused one only paints newly-added rows. Unfortunately focus is a necessary part of a11y for the library.

Just thought I'd ask in case you had any pointers! Thanks for putting together this gist! :)

trusktr commented on 5 Jul 2016

> Mouse event offsets. Ouch.

@Theodus This is why I've taken the stance at starting new projects with all events disabled, then enabling them only as needed, and also using only transforms as much as possible with constant sizing.

trusktr commented on 5 Jul 2016

@paulirish, you have unanswered questions that need your respected and appreciated attention.

grese commented on 25 Aug 2016

Nice :) Thanks for taking the time to put this together!

przeor commented on 1 Sep 2016

Hi I see that you use React, so I am sure that you will find interesting the https://reactjs.co - this is the free online convention and tutorial book for React.JS Developers. React is not only the View (in MVC) anymore. ReactJS For Dummies: Why & How to Learn React Redux, the Right Way.

ckomop0x commented on 20 Jan 2017

Nice =) Thanx!

hereisfun commented on 1 Mar 2017

I'm suprised that according to CSS Triggers, visibility and opacity will also trigger reflow in WebKit. It makes me confused. Aren't they only visual properties?

stevemao commented on 28 Mar 2017

@hereisfun Probably because WebKit is not optimised for them

kaseopea commented on 11 May 2017 • edited

@njoyard I have faced with the same - complex layout using CSS columns. And I have a problem on Ipad. When I change orientation from landscape to portrait css columns doesn't repaint for the first time. Did you experience this issue on Ipad? If so, how did you solve this?

tarekahf commented on 21 Jun 2017 • edited

I am using `ngProgress` in my project which has large number of directives with `$compile` service. It seems that the progress bar won't show unless all elements are compiled. I tried to use several methods as mentioned in this post, but none of them worked. I am using `ngProgress.start()` as in the beginning of the `ng-controller`, and yet, the progress bar won't show until almost everything is done.

Appreciate your help to solve this problem.

More details here.

Tarek

jeremychone commented on 22 Jun 2017

Thanks for spending the time to share this.

Gibolt commented on 2 Jul 2017

http://csstriggers.com/ is dead :(

trusktr commented on 7 Jul 2017

Does adding an element with `display:none` cause any layout (without reading those above listed properties)? f.e., if I add a bunch of SVG elements to the DOM, and the `<svg>` is `display:none`, can I avoid any overhead?

prchando commented on 22 Jul 2017

Is there a way to test what causes reflow/layout/repaint and what does not

PraveenVignesh commented on 13 Oct 2017

@paulirish **@SpoBo** did you find any linter for Layout thrashing? If so please share the code.
Thanks.

weo3dev commented on 22 Oct 2017

@Gibolt http://csstriggers.com is back

holmberd commented on 10 Dec 2017

Using `document.getSelection().removeAllRanges();` does in some cases cause this when toggling elements visibility with `display: none` .

sambgordon commented on 16 Jan 2018

Awesome, thank you!

fmortens commented on 30 Jan 2018

Just curious. Could the elem.blur() also trigger a repaint?

Cerbrus commented on 1 Feb 2018 • edited

In your Appendix, the `<center>` tags around the image break GitHub's markdown rendering for that image.

See my fork, compared to yours ;-)

Or, for your convenience, a rich diff

paulirish commented on 6 Feb 2018 Author Owner

@Cerbrus thanks!! fixed.

paulirish commented on 6 Feb 2018 • edited Author Owner

@fmortens yes i've seen `blur()` DEFINITELY force layout before. good question as to why its not on my list. i'll take another look.

update: nevermind, i was thinking of `focus()`

ExtAnimal commented on 14 Feb 2018

@paulirish pretty important addition:

`elem.classList.remove('this-class-is-NOT-in-the-classlist')` invalidates the layout. This is poor behaviour from the browser vendors!

paulirish commented on 23 Feb 2018 Author Owner

@fmortens blur() doesn't force a layout. You can verify if you profile this:

```
  document.body.style.background = '';

  document.body.onclick = e => {
   document.body.style.background = `hsl(${Math.floor(Math.random()*360)}, 50%, 50%)`;
    now = Date.now();
    document.querySelector('#i').blur();
    while (Date.now() < now + 10) { }
  }
```

@ExtAnimal yes technically invalidation is an entirely different thing. It's a shame `classList.remove(fakeclassname)` does an invalidation; you can file it on crbug.com and there's a good change an engineer will take it on.

MorrisJohns commented on 15 Mar 2018 • edited

@paulirish: In theory window.innerHeight and window.innerWidth should never cause a layout / reflow because the values do not include the scrollbar thickness (I tested this and also MDN agrees in documentation for window.innerHeight albeit not said well: "Height (in pixels) of the browser window viewport including, if rendered, the horizontal scrollbar"). I checked all your links and I could find nothing relevant, although I am not sure how you test for reflow (**@bfred-it** also asked about this. **@dmnd** commented but I am contradicting his assumption as he asked!).

Jab2870 commented on 18 Jun 2018

Don't know if this is the right place to ask but I'll give it a shot. Apologies if not.

I have tried to animate a the meta themeColor. I do this by calling `setAttribute` on the meta tag in a RAF callback. This absolutely kills performance. Any ideas why? I don't think it is causing a layout recalculation.

paulirish commented on 18 Jun 2018 Author Owner

@Jab2870 theme-color affects the browser chrome that's completely outside of the webpage. So you're definitely dealing with a new kind of browser jank. You can file browser bugs if you think that's appropriate. :)

Jab2870 commented on 18 Jun 2018

@paulirish
Thanks, I have done: https://bugs.chromium.org/p/chromium/issues/detail?id=853760

jasonwilliams commented on 20 Jun 2018

> elem.classList.remove('this-class-is-NOT-in-the-classlist') invalidates the layout. This is poor behaviour from the browser vendors!

@ExtAnimal unless i've got it wrong I'm not seeing that here https://codepen.io/jayflux/pen/PaRodg (testing on both Chrome latest stable and Firefox)

paulirish commented on 20 Jun 2018 Author Owner

> elem.classList.remove('this-class-is-NOT-in-the-classlist') invalidates the layout. This is poor behaviour from the browser vendors!

> **@ExtAnimal** unless i've got it wrong I'm not seeing that here https://codepen.io/jayflux/pen/PaRodg (testing on both Chrome latest stable and Firefox)

This does *invalidate* the layout, but it doesn't force the layout to be recomputed synchronously. This page details only those APIs which cause a forced synchronous layout. Detailing which ones cause invalidations would be an entirely new list (and much much larger).

jasonwilliams commented on 20 Jun 2018 • edited

> This does invalidate the layout, but it doesn't force the layout to be recomputed synchronously. This page details only those APIs which cause a forced synchronous layout. Detailing which ones cause invalidations would be an entirely new list (and much much larger).

Oh interesting, thanks.
@paulirish Is there anything in dev tools which shows this (asynchronous style calculation)? Or is this known from looking at blink source code?

nachoab commented on 27 Jun 2018

what about window.visualViewport.height / width / offsetTop / offsetLeft ?

andretessmann commented on 24 Jul 2018

Reading offsetHeight from an element between changing from display:none to display:block in Safari saved my day.

sag1v commented on 21 Aug 2018 • edited

@paulirish Thanks!
Is reading `entry.contentRect` from `ResizeObserver` will force a Reflow?
I've seen that the polyfill uses `elem.clientWidth` but is reading also triggers a reflow?

paulirish commented on 21 Aug 2018 • edited

Author Owner

@jasonwilliams there is an experiment in devtools for tracking invalidations. unfortunately no docs, but it adds extra metadata to recalcstyle/layout/paint events that you'll see in the Perf Panel.

@nachoab i can't verify in the source right now, but i suspect not. unless any of these would change depending on the presence of a scrollbar.

@sag1v nope. it was designed to not. https://wicg.github.io/ResizeObserver/#html-event-loop explains that it runs right after Layout happens, so its all freshly computed. :) However, if there are *multiple* RO callbacks, then it's possible that layout was dirtied during previous RO callback, so then contentRect *would* force layout. Tricky tricky.

sag1v commented on 22 Aug 2018

@paulirish Thanks for the clarification and link to the relevant spec! though i still struggle to understand something.
When you say *"multiple RO callbacks"* do you mean:

1. on the same target or multiple instances of RO?
2. running an async loop inside the callback, like `interval` ?

rezof commented on 22 Oct 2018 • edited

@sag1v it doesn't matter if it's on the same target or not, once the layout is dirtied (no longer valid) accessing contentRect will trigger a reflow / force layout.

yellow1912 commented on 8 Nov 2018

How about changing attribute using setAttribute, something like element.setAttribute('data-this', 'that'), does it force reflow?

ajaysagar commented on 13 Nov 2018 • edited

@paulirish and others, thanks a bunch for maintaining this helpful page. I went through the above comments to see if anything is mentioned about CSS Transforms invalidating styles for element subtrees, but could not find anything related. I have this fiddle: https://jsfiddle.net/58n1zg4a/86/ where an element containing a large number of child elements is transformed (along with setting "will-change: transform"). In Performance recordings I see that up to 6 ms is taken in the fiddle for Style Recalculation, I am unable to figure out why the transform on the parent element would invalidate styles on the child elements and trigger a recalc. P.S in tests of my production app that has the same characteristics as the fiddle, Firefox and Safari fared quite well with almost constant 60fps whereas Chrome was janky. Any tips on this would be greatly appreciated, thanks!

iwisunny commented on 30 Nov 2018

👍

matthew-dean commented on 4 Dec 2018

Does anyone have any data to know, what's the least impactful way (in terms of triggering repaints) to update the CSS OM?

Would it be:

1. Replacing the innerHTML of a `<style>` tag with `<style>html { --updated-variable: newvalue; }</style>`, or
2. Updating the style property of `<html>` with `<html style="--updated-variable: newvalue;">`

I'm considering building something that re-calculates CSS custom props on the fly, so I'm wondering what will have the least impact on propagating a change in styles to the rest of the document.

paulirish commented on 4 Dec 2018 — Author — Owner

@matthew-dean I think you should test it. I remember hearing that Blink had some optimizations if a stylesheet was just appended to. Like `styleElem.textContent += '.otherstyles { ... }'`. In this case it could do a smaller recalc style. But I'm not sure about that vs the two options you floated.

Best to just make a stress test page and see what the profiler says.

msujaws commented on 15 Dec 2018

@paulirish The link to the FrameNeedsReflow for Gecko is dead (that subdomain has been turned off). This link will work, https://searchfox.org/mozilla-central/search?q=FrameNeedsReflow&case=false&regexp=false&path=%5E%5B%5E%5C0%5D*

shmdhussain commented on 26 Dec 2018

If I just asks for the measurements of dom elements without changing any DOM Properties , in this case reflow/layout will be caused on each line, or after the end of the last line(that means in beginning of frame starting)

```
var currentScrollY = window.pageYOffset; //Will this line cause reflow/layout
var heightOfScreen = document.documentElement.clientHeight; //Will this line cause reflow/layout

//or reflow will be caused on the frame starting?
```

Pictor13 commented on 26 Feb

> BTW. what about
>
> obj.style.display

From my experience changing `display` triggers a reflow.
Also confirmed by CssTriggers.

gnujoow commented on 21 Mar

👍 thanks

McFarJ commented on 30 Apr • edited

Your link referring to SVG elements doesn't work (http://gent.ilcore.com/2011/03/how-not-to-trigger-layout-in-webkit.html)

oceangravity commented on 8 May

Having Chrome DevTools open cause reflow?

tomasdev commented on 14 May

does `getBBox()` trigger reflow?

Waltari10 commented on 24 May

I am confused about mixing reads and writes. So first doing all the reads and then the writes will be more efficient than mixing them up?

So is it better to:
read()
read()
write()
write()

Than:
read()
write()
read()
write()

Why is it better to id this way? Cant you batch reads and writes together anyway using FastDom?

SeonHyungJo commented on 14 Jun

👍

kirill-chirkov-at-... commented on 4 Jul • edited

What about `elementFromPoint` ? https://developer.mozilla.org/en-US/docs/Web/API/DocumentOrShadowRoot/elementFromPoint
Does it force reflow?

EECOLOR commented 21 days ago

@paulirish Thank you for the gist.

It states:

> Read your metrics at the begininng of the frame (*very very start of rAF*, scroll handler, etc),

The *very very start of rAF* seems unrealistic in real life applications, some other parts of the application (or a library) might have done a write already. Wouldn't it be better to state that in general reads should be done anywhere except for rAF (or simpler: reads should not be done in rAF)? And, to make that work: writes should only be done in rAF.