ionic

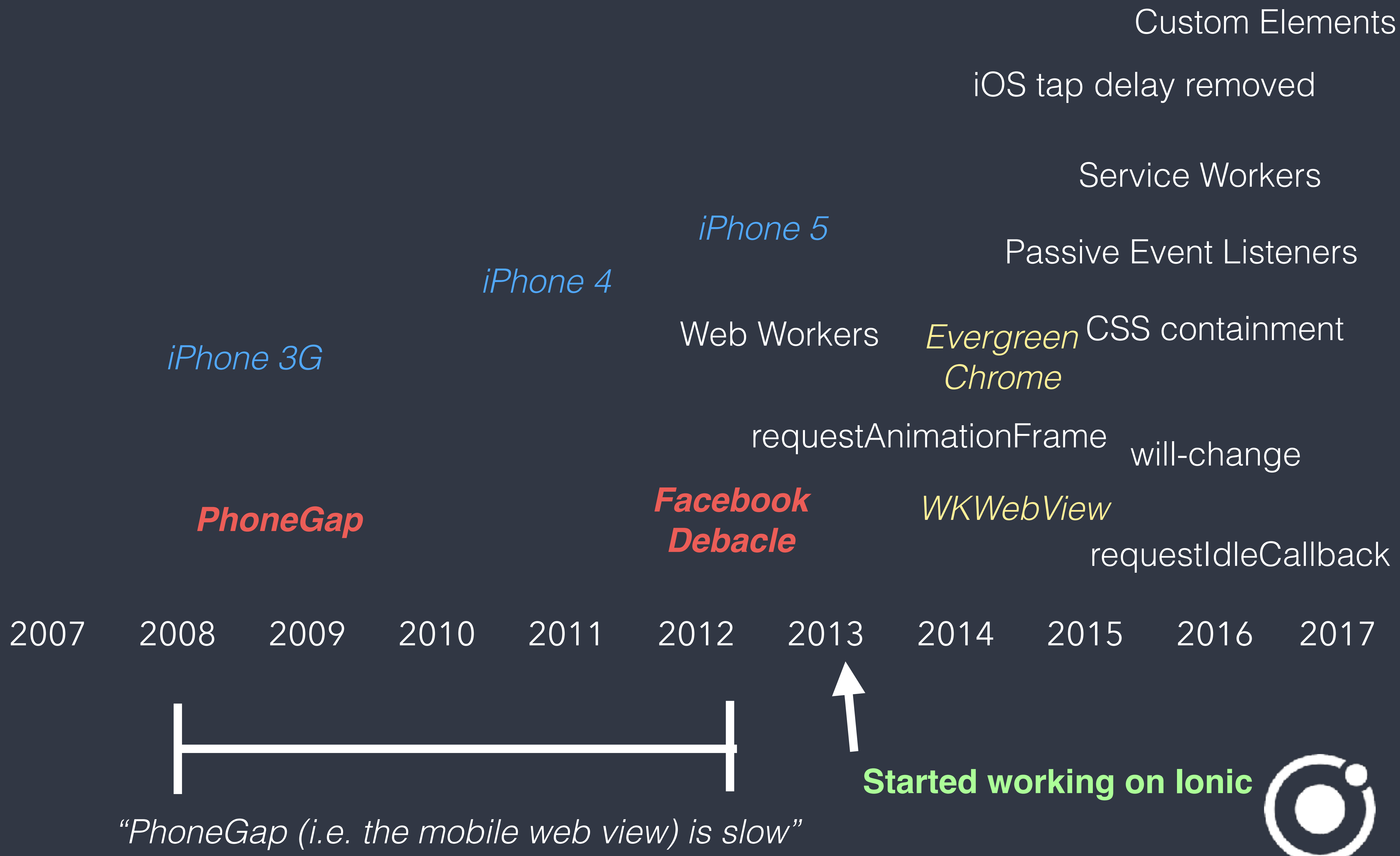
Performance Techniques in 2017

Getting native performance with new Web APIs

Max Lynch
@ionicframework
@maxlynch

Yearly check-in

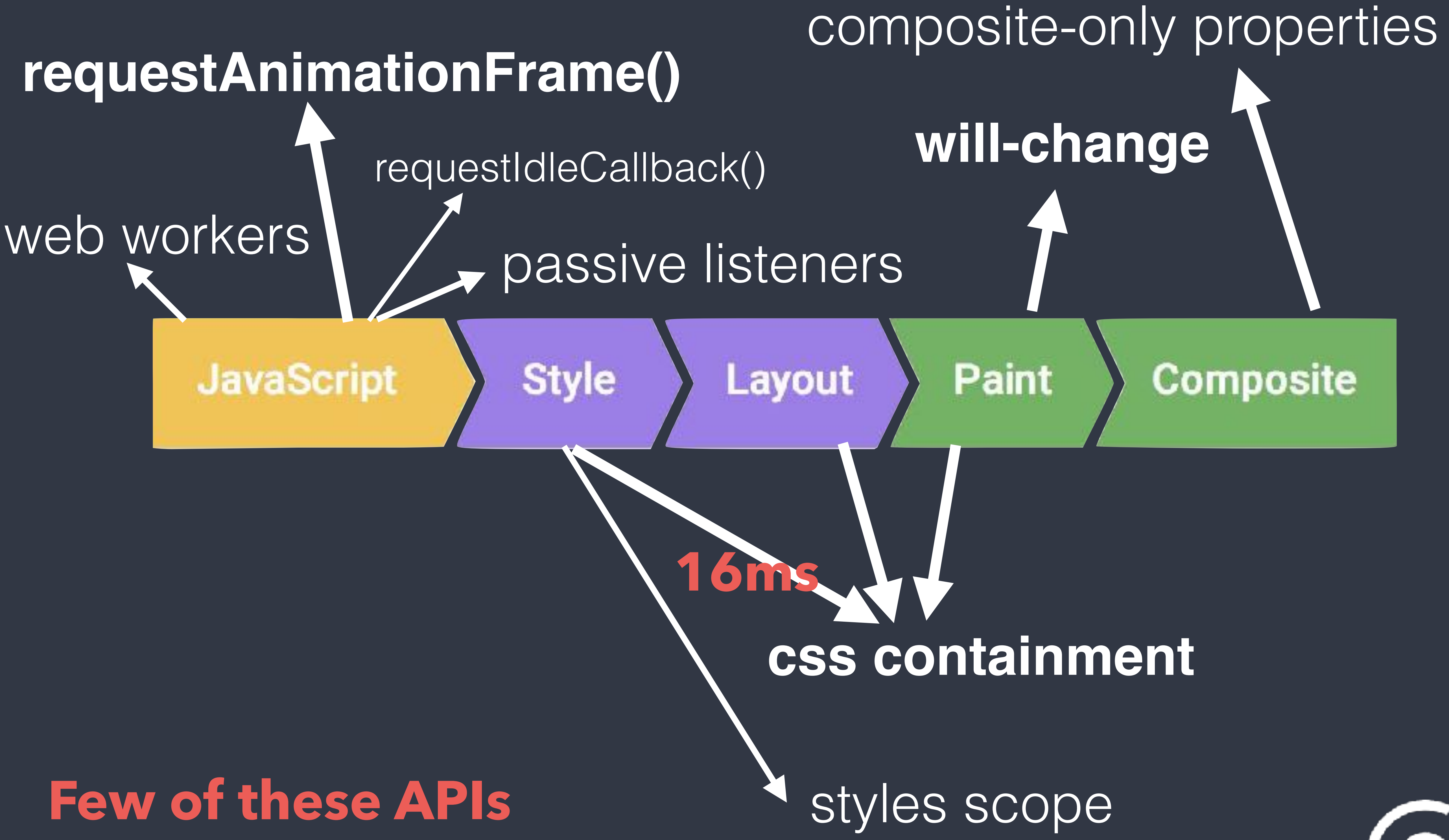
API/Perf Timeline (mobile)



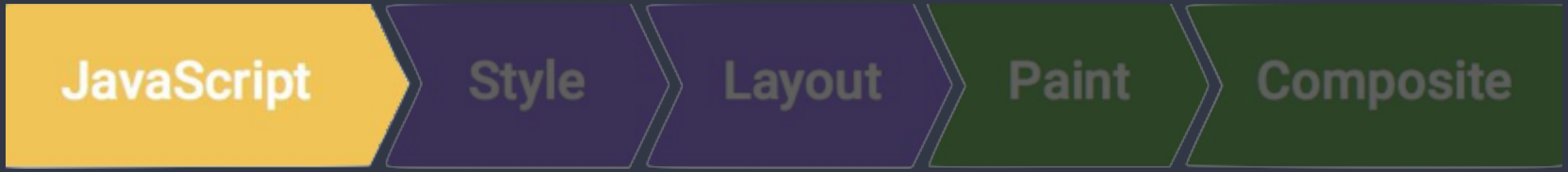
Making Fast Apps

“Rendering Performance”, Paul Lewis, Google
https://developers.google.com/web/fundamentals/performance/rendering/

The life of a frame



Few of these APIs available in 2013!


JavaScript
Style
Layout
Paint
Composite

requestAnimationFrame()

Request that your function be called before next paint

↓

Function called ~60 times/sec or throttled (background)

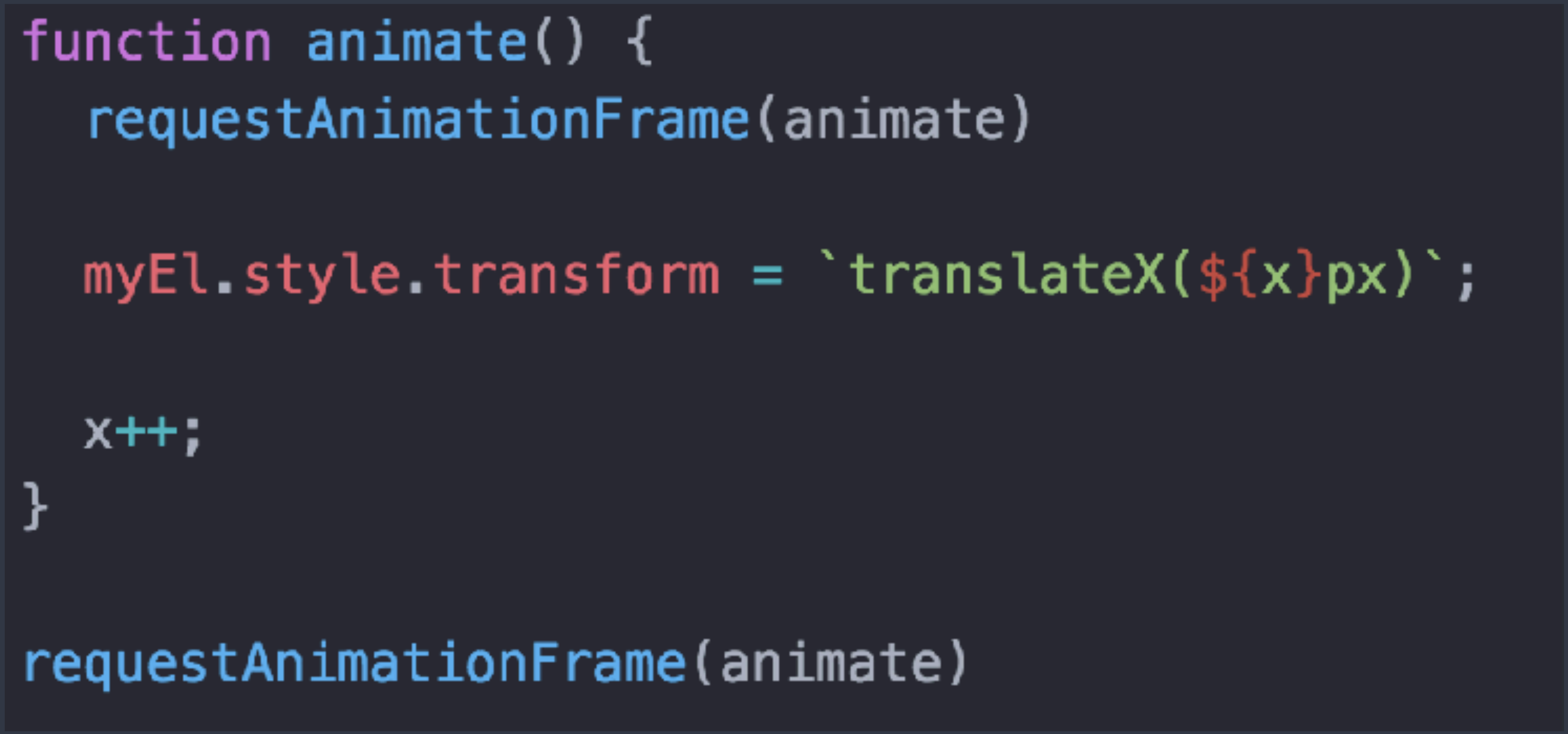
↓

Animations optimized into single reflow/repaint

↓

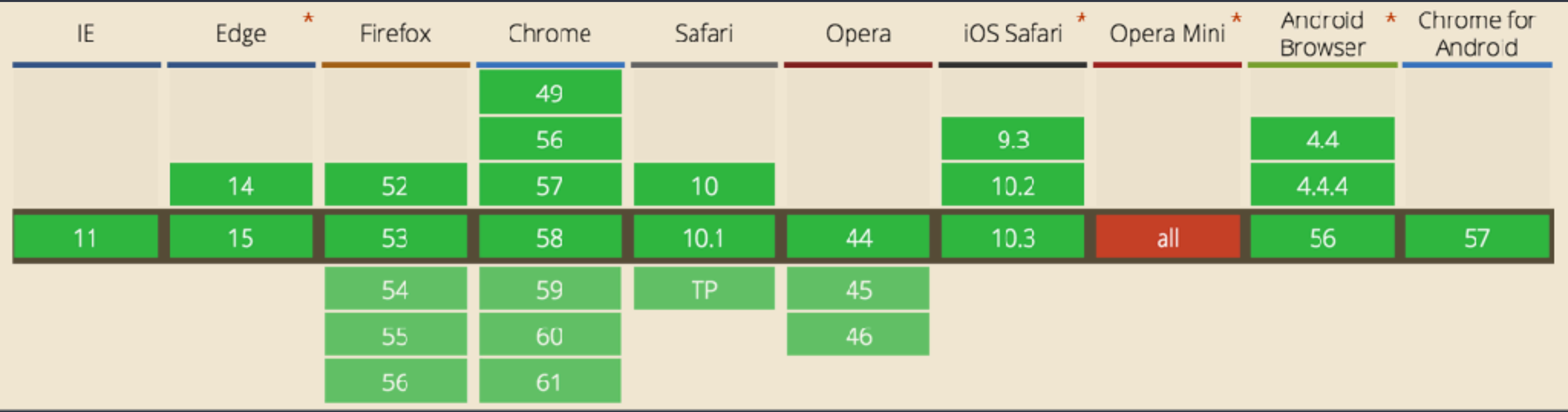
Smooth animations w/o jank

requestAnimationFrame() (example)



```
function animate() {
  requestAnimationFrame(animate)

  myEl.style.transform = `translateX(${x}px)`;

  x++;
}

requestAnimationFrame(animate)
```

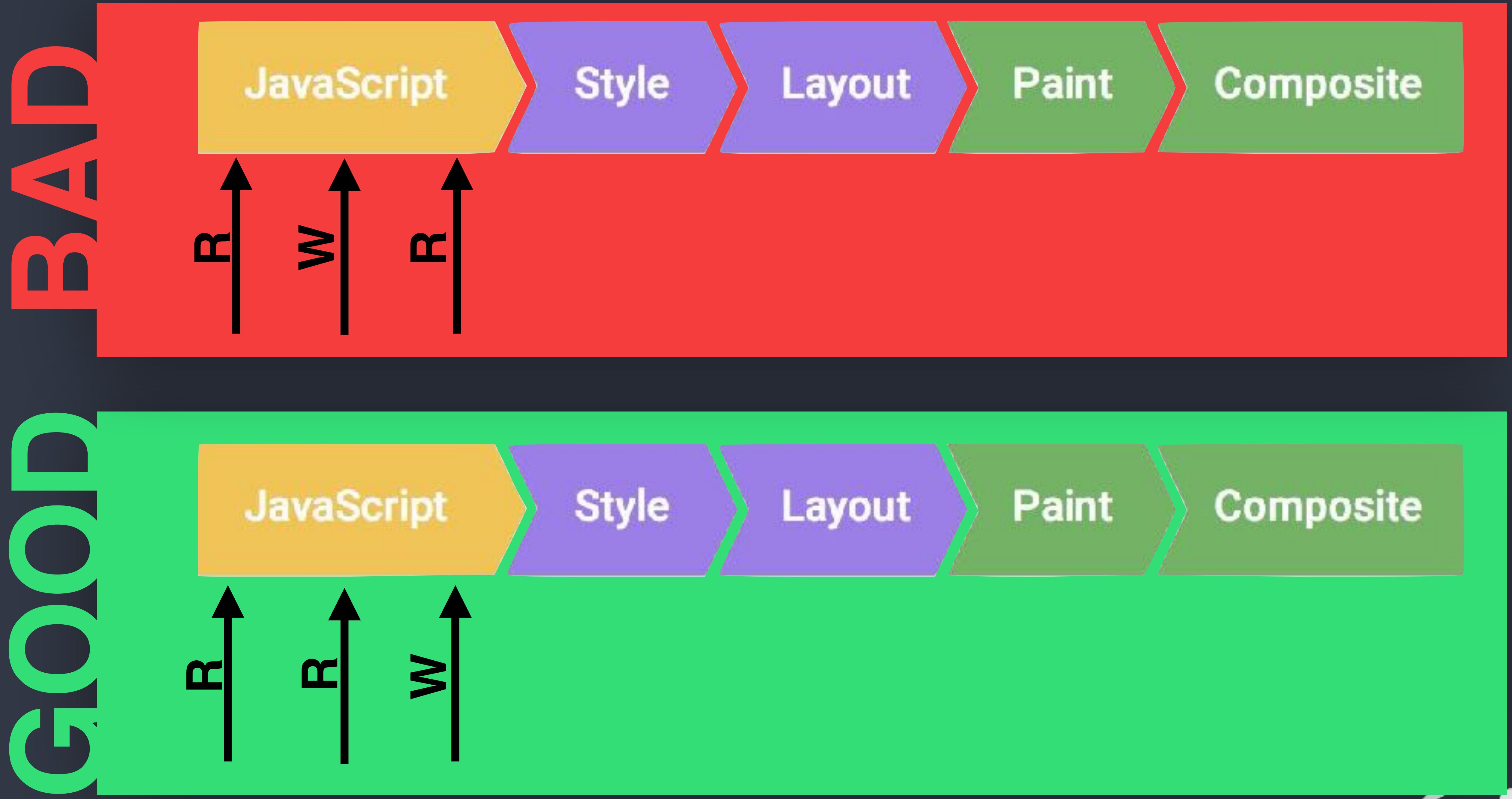
requestAnimationFrame() - availability



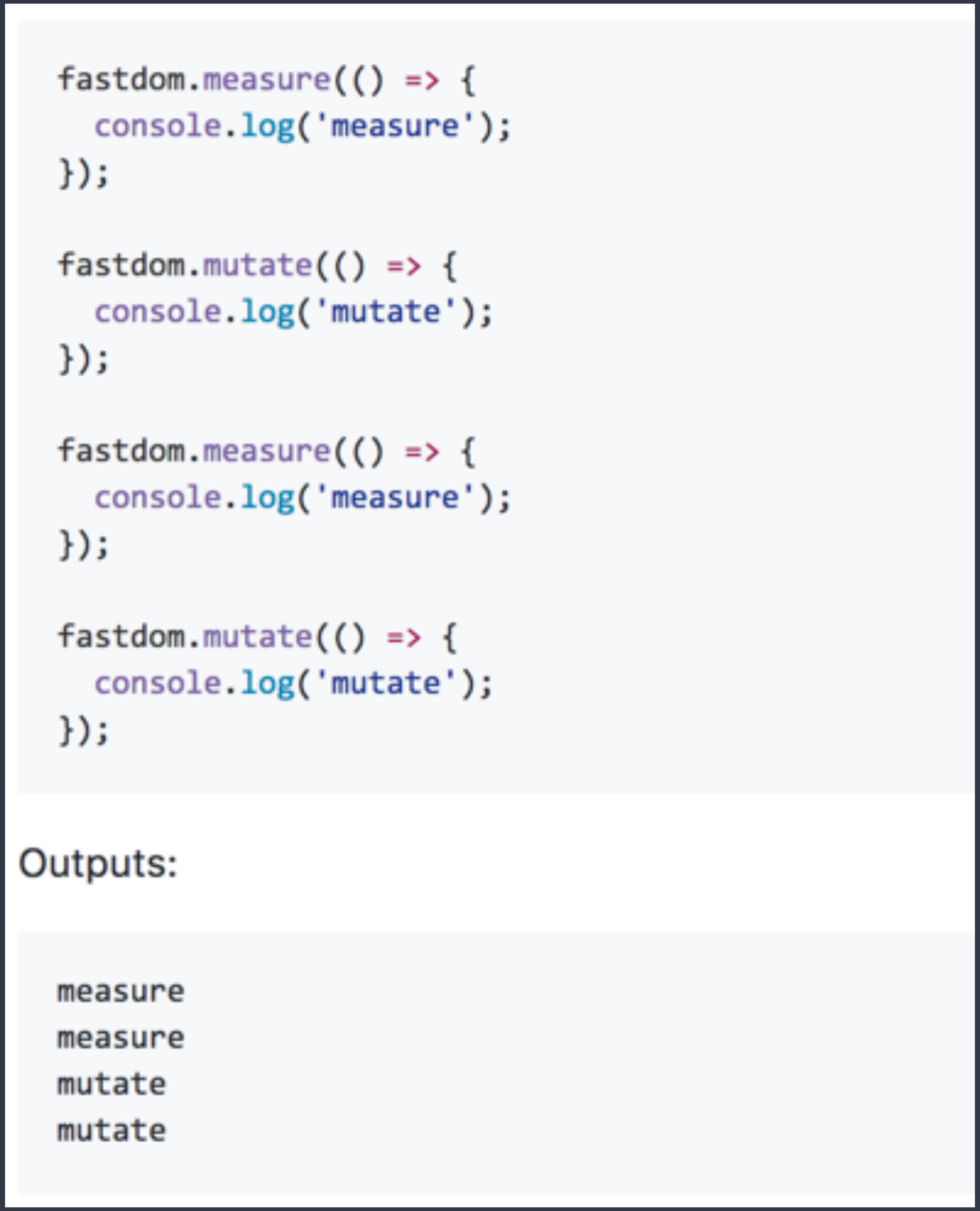
IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49						
			56			9.3		4.4	
	14	52	57	10		10.2		4.4.4	
11	15	53	58	10.1	44	10.3	all	56	57
		54	59	TP	45				
		55	60		46				
		56	61						

Layout Thrashing

R (read) **W (write)**



Avoiding Layout Thrashing: DOM batching



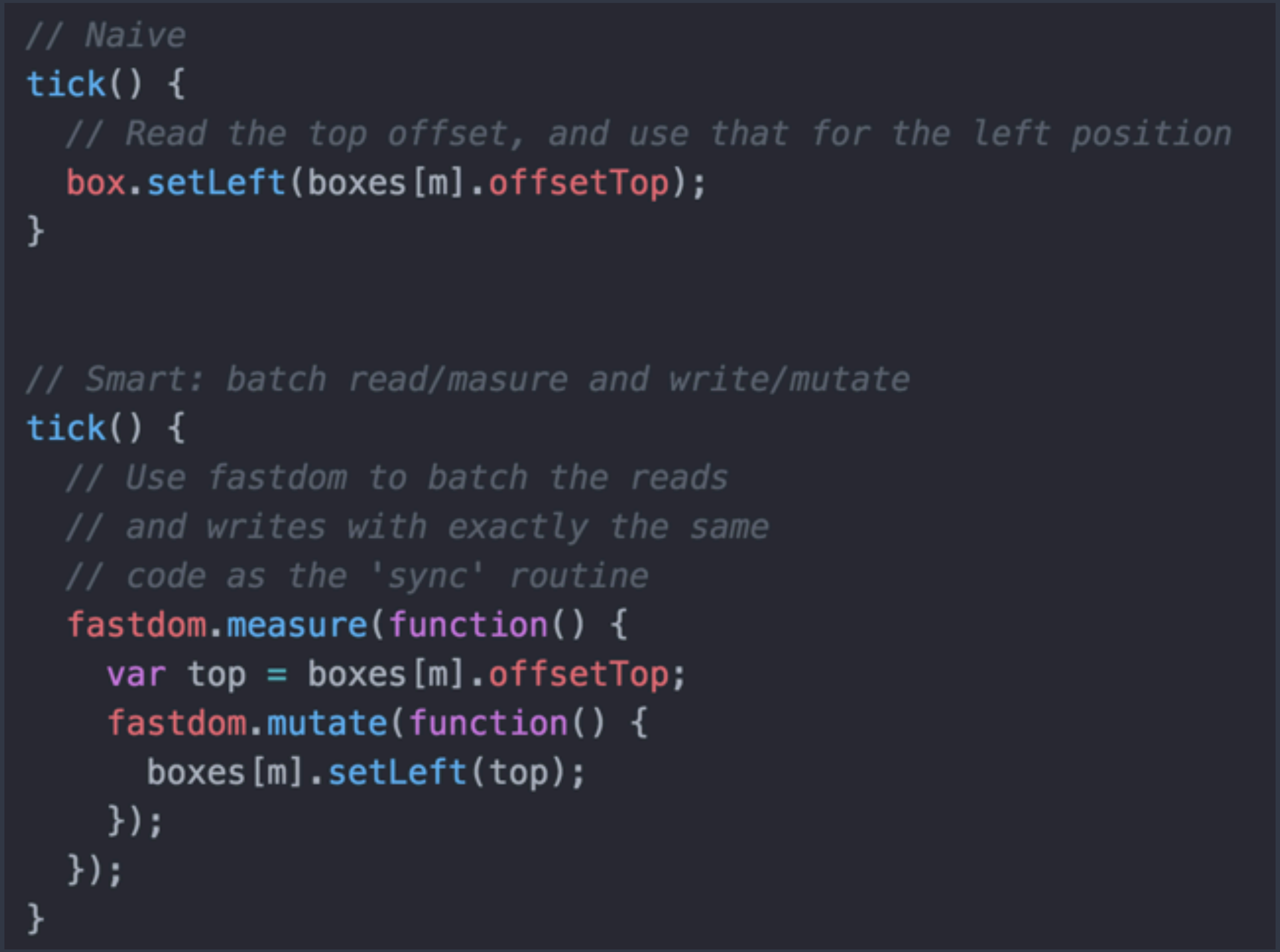
```
fastdom.measure(() => {
  console.log('measure');
});

fastdom.mutate(() => {
  console.log('mutate');
});

fastdom.measure(() => {
  console.log('measure');
});

fastdom.mutate(() => {
  console.log('mutate');
});
```

Outputs:

```
measure
measure
mutate
mutate
```

https://github.com/wilsonpage/fastdom

Avoiding Layout Thrashing: DOM batching

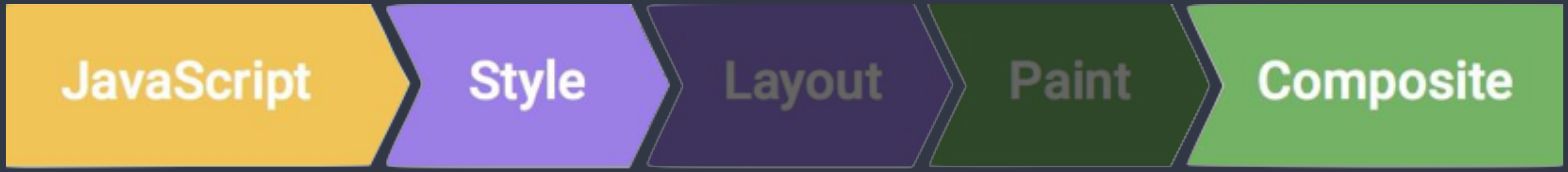


```
// Naive
tick() {
  // Read the top offset, and use that for the left position
  box.setLeft(boxes[m].offsetTop);
}


// Smart: batch read/masure and write/mutate
tick() {
  // Use fastdom to batch the reads
  // and writes with exactly the same
  // code as the 'sync' routine
  fastdom.measure(function() {
    var top = boxes[m].offsetTop;
    fastdom.mutate(function() {
      boxes[m].setLeft(top);
    });
  });
}
```

Relies on requestAnimationFrame()

Efficient style modifications

Skip layout and paint by only modifying composite-only properties.
Those are: **transform**, and **opacity**



https://developers.google.com/web/fundamentals/performance/rendering/stick-to-compositor-only-properties-and-manage-layer-count

Passive Event Listeners

Indicate touch events won’t block scrolling

↓

Run event listener in separate thread w/o blocking

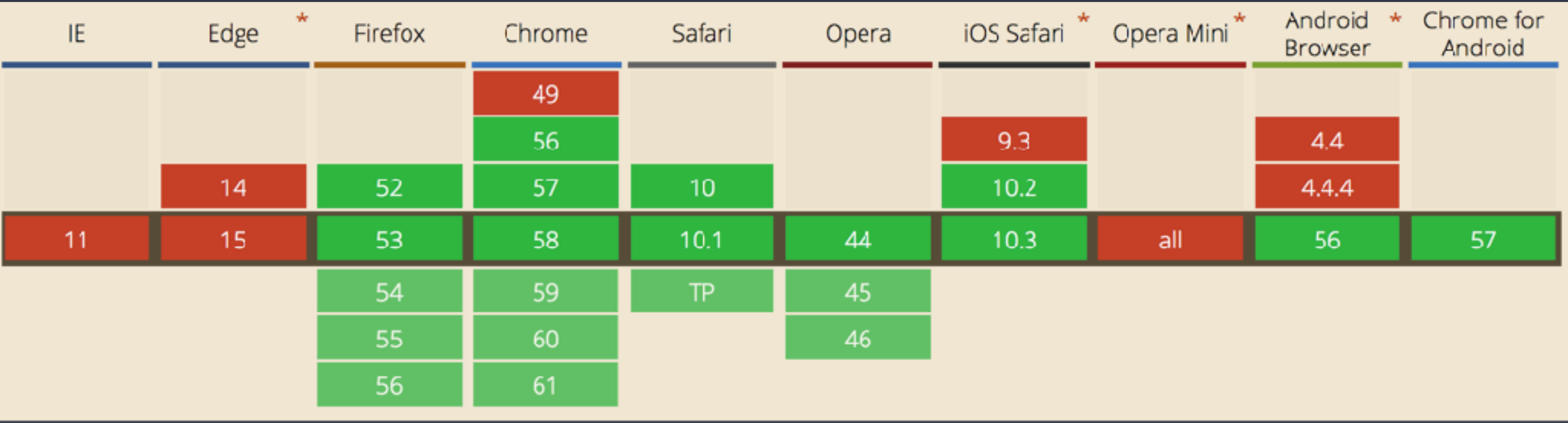
↓

Smooth touch and scroll animations and gestures

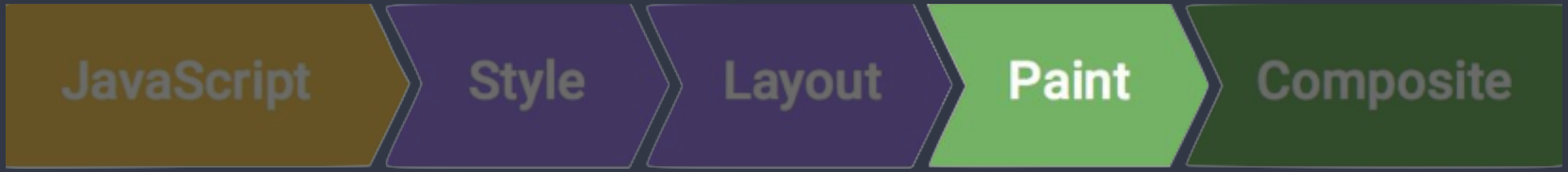
Passive Event Listeners (example)

```
addEventListener(document, "touchstart", function(e) {
  // e.preventDefault(); -> Can't! It's passive
}, { passive: true });
```

Passive Event Listeners - availability



IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49						
			56			9.3		4.4	
	14	52	57	10		10.2		4.4.4	
11	15	53	58	10.1	44	10.3	all	56	57
		54	59	TP	45				
		55	60		46				
		56	61						


JavaScript
Style
Layout
Paint
Composite

will-change

Indicates to the browser certain properties will **change frequently** (ex: scrolling, animations, gestures)

↓

Browser promotes element to own layer

↓

Smoother animations with less CPU usage (though possibly higher RAM usage)

Use sparingly If everything is optimized, nothing is

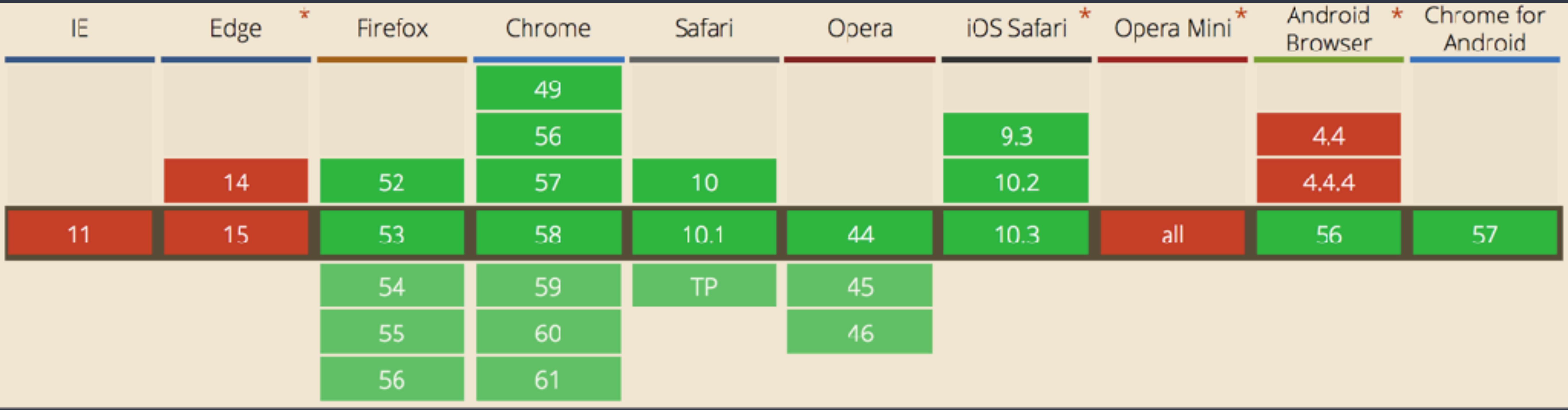
will-change (example)

```
will-change: auto;
will-change: scroll-position;
will-change: contents;
will-change: transform;
will-change: opacity;
will-change: left, top;
```

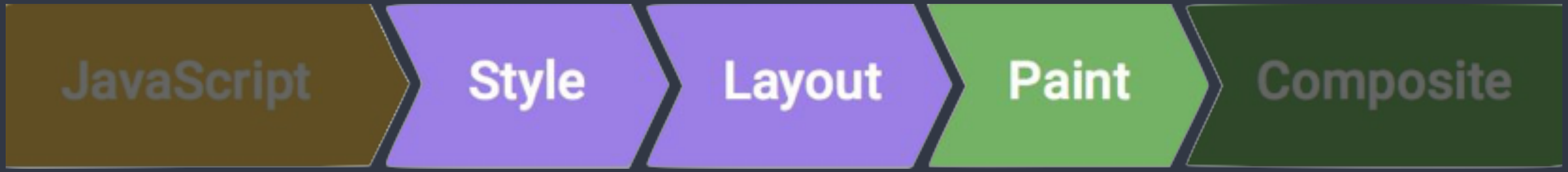
Fallback:

```
transform: translateZ(0)
```

will-change - availability



IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			🟩 49						
			🟩 56			🟩 9.3		🟥 4.4	
	🟥 14	🟩 52	🟩 57	🟩 10		🟩 10.2		🟥 4.4.4	
🟥 11	🟥 15	🟩 53	🟩 58	🟩 10.1	🟩 44	🟩 10.3	🟥 all	🟩 56	🟩 57
		🟩 54	🟩 59	🟩 TP	🟩 45				
		🟩 55	🟩 60		🟩 46				
		🟩 56	🟩 61						


JavaScript
Style
Layout
Paint
Composite

CSS containment

Indicate isolated elements

↓

Browser optimizes, limiting recalc paint/layout/size/style to sub-tree

↓

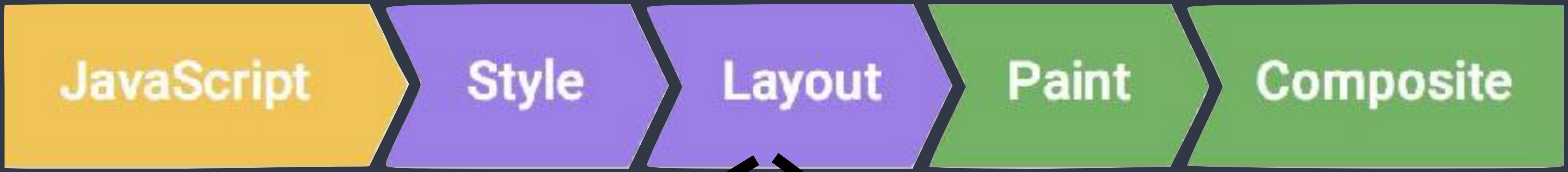
Fast component updates

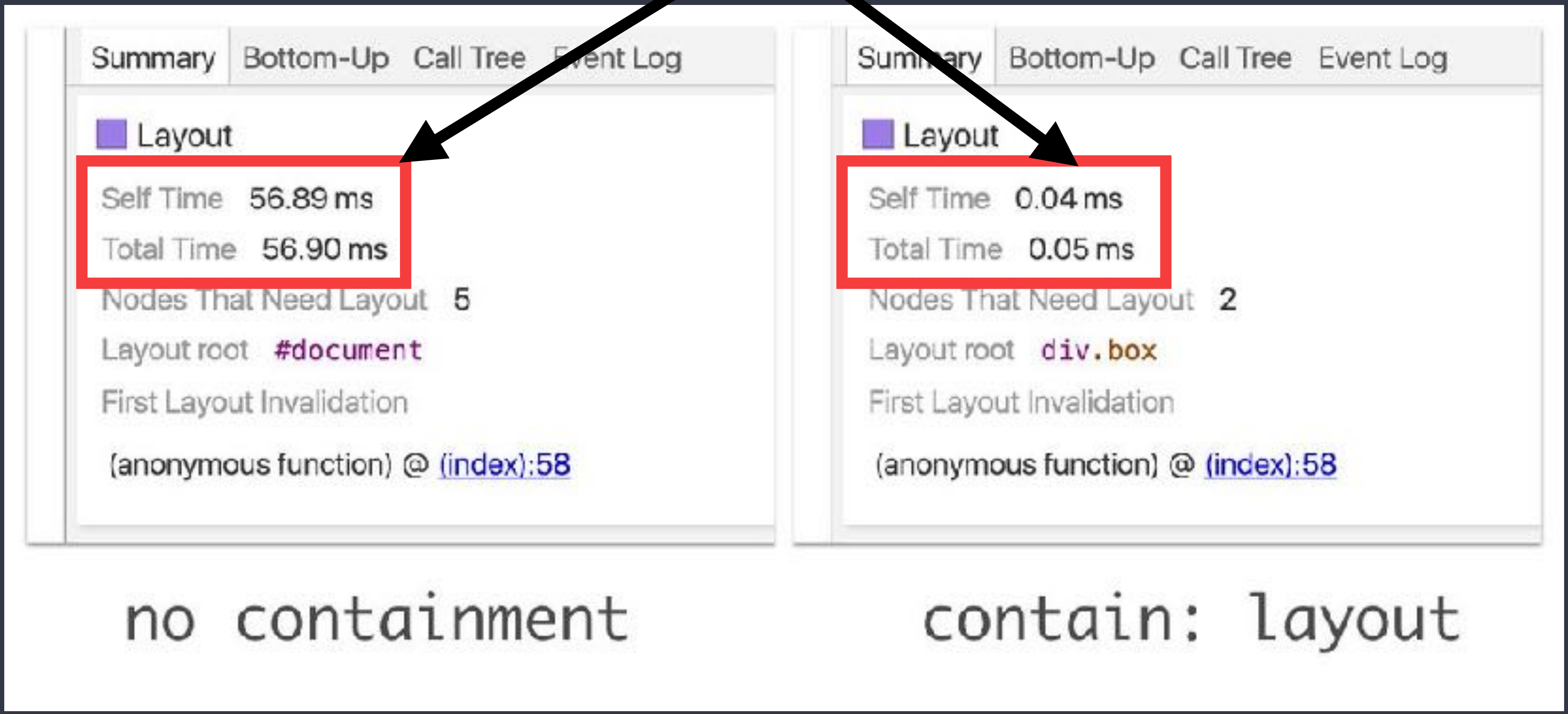
CSS containment (example)

```
ion-modal {
  position: absolute;
  top: 0;
  left: 0;

  display: block;
  visibility: inherit !important;

  width: 100%;
  height: 100%;

  contain: strict;
}
```

```
contain: none | strict | content | [ size || layout || style || paint ]
```

CSS containment





Layout
Self Time 56.89 ms
Total Time 56.90 ms
Nodes That Need Layout 5
Layout root #document
First Layout Invalidation
(anonymous function) @ (index):58

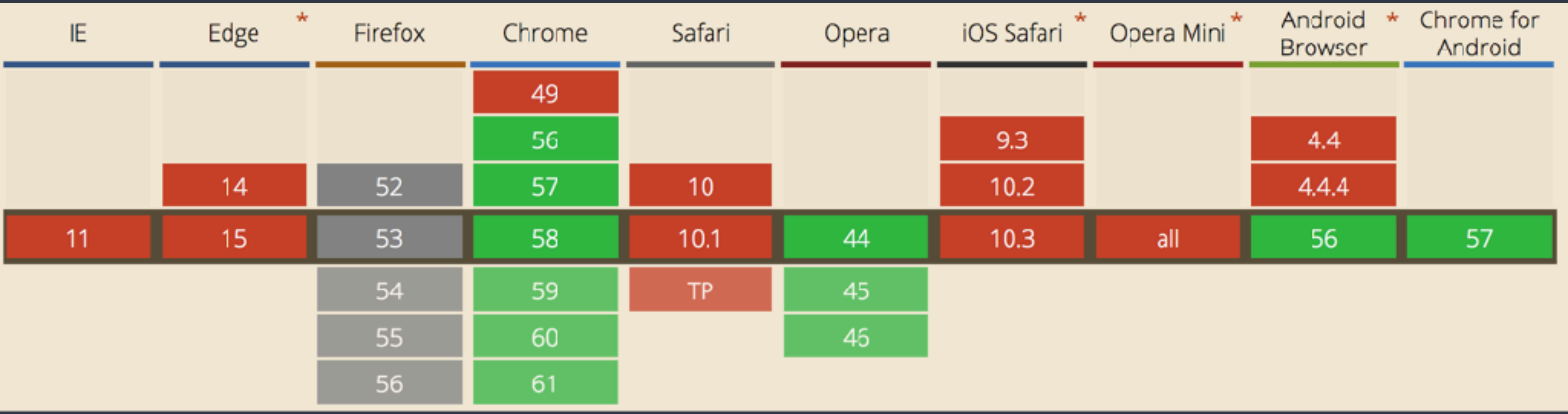
no containment

Layout
Self Time 0.04 ms
Total Time 0.05 ms
Nodes That Need Layout 2
Layout root div.box
First Layout Invalidation
(anonymous function) @ (index):58

contain: layout

layout 1425x faster!

CSS containment - availability



IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49						
			56			9.3		4.4	
	14	52	57	10		10.2		4.4.4	
11	15	53	58	10.1	44	10.3	all	56	57
		54	59	TP	45				
		55	60		46				
		56	61						

Why use frameworks?

- **Frameworks do this stuff for you**
- **Simpler way to use some APIs**
- **Avoid direct DOM manipulation**

Things I didn’t cover

- **Web Workers/SharedArrayBuffer**
- **PWA topics** (PRPL, App Shell, pre-caching, etc)
- **Optimizing CSS**
- **Debouncing input handlers**[1]
- **Bundle size concerns**
- **requestIdleCallback**
- **WebGL**
- **New JS Engine work**

[1] https://developers.google.com/web/fundamentals/performance/rendering/debounce-your-input-handlers

Further reading

- **Will-change:** https://developer.mozilla.org/en/docs/Web/CSS/will-change
- **requestAnimationFrame():** https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame
- **CSS containment:** https://developers.google.com/web/updates/2016/06/css-containment
- **Layout Thrashing:** https://developers.google.com/web/fundamentals/performance/rendering/avoid-large-complex-layouts-and-layout-thrashing
  http://wilsonpage.co.uk/preventing-layout-thrashing/
  https://github.com/wilsonpage/fastdom
- **Passive Event Listeners:** https://developers.google.com/web/updates/2016/06/passive-event-listeners

Thanks!

Presentation available online

https://github.com/mlynch/pgday-eu-2017-perf