


Rendering Performance

Render your web app at 60fps

Đinh Quang Trung
24 October 2015

A decorative light blue triangle is located in the bottom right corner of the slide, pointing towards the top right.

About me



Đinh Quang Trung
Front-end Developer



fb.com/trungdq88



[@trungdq88](https://twitter.com/trungdq88)



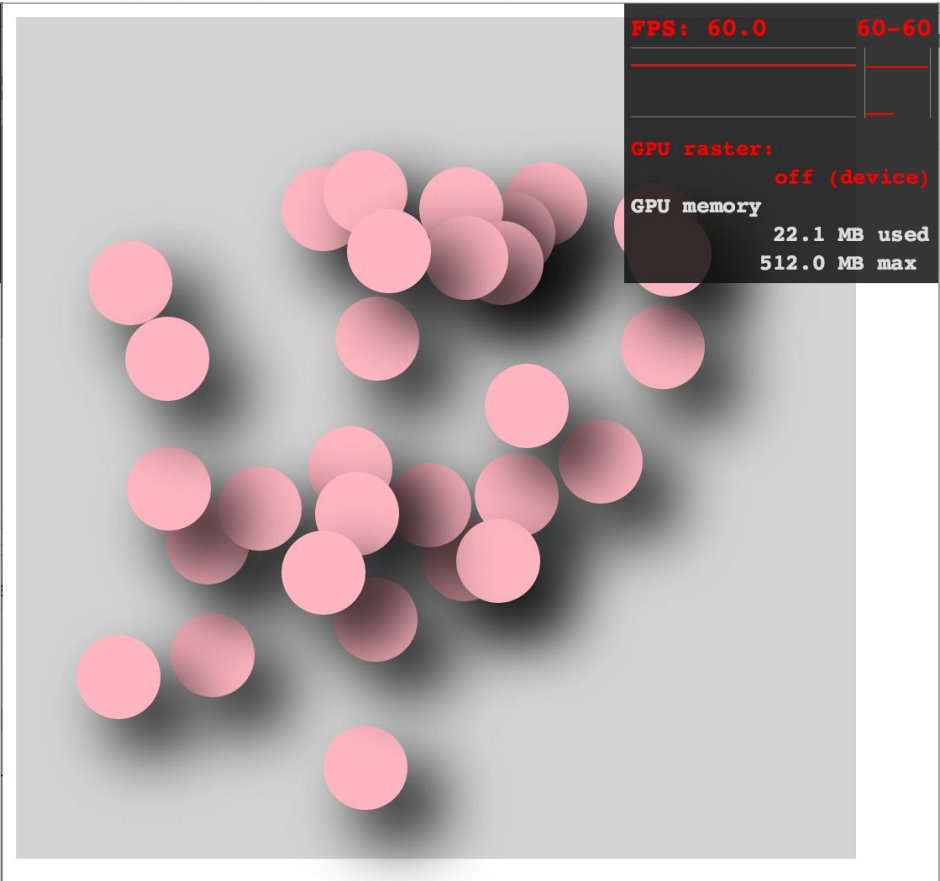
github.com/trungdq88

Rendering Performance

Overview

1. Understand how browser render web pages
2. Optimize the render process
3. Useful resources for Web Performance

60fps vs 30fps



16ms

16ms

60 frames / second

16ms

60 frames / second

1/60 second / frame

16ms

60 frames / second

$1/60$ second / frame

16ms / frame

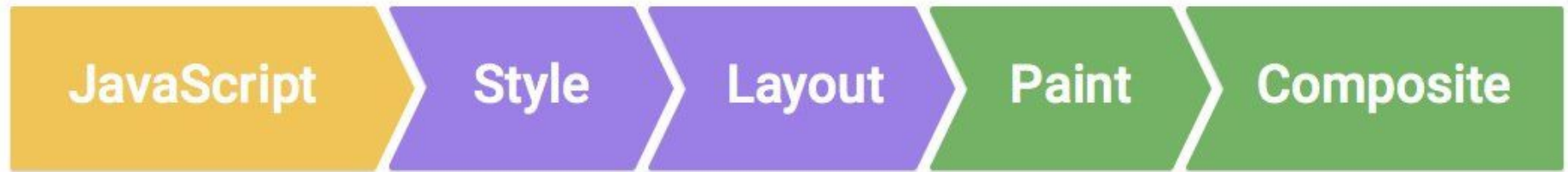
16ms

60 frames / second

1/60 second / frame

16ms / frame

The Pixel Pipeline



The Pixel Pipeline

```
box.style.width = '300px';
```



JavaScript

Style

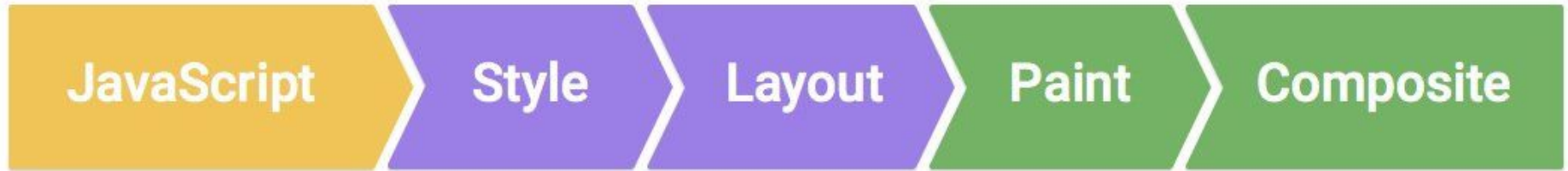
Layout

Paint

Composite

The Pixel Pipeline

```
box.style.width = '300px';
```

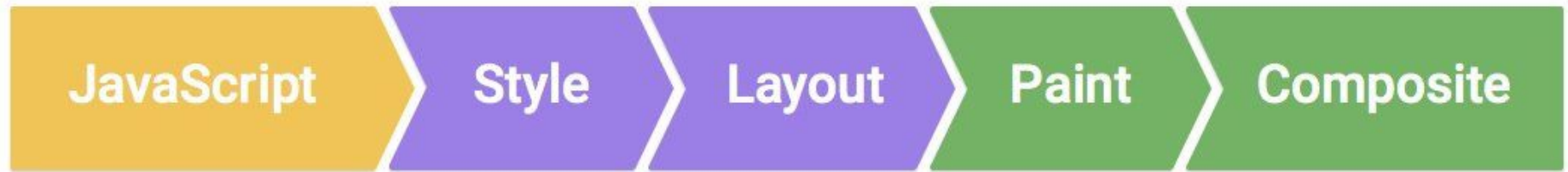


```
<div style="300px">
```

The Pixel Pipeline

```
box.style.width = '300px';
```

Recalculate element size, and
other **affected elements** size.

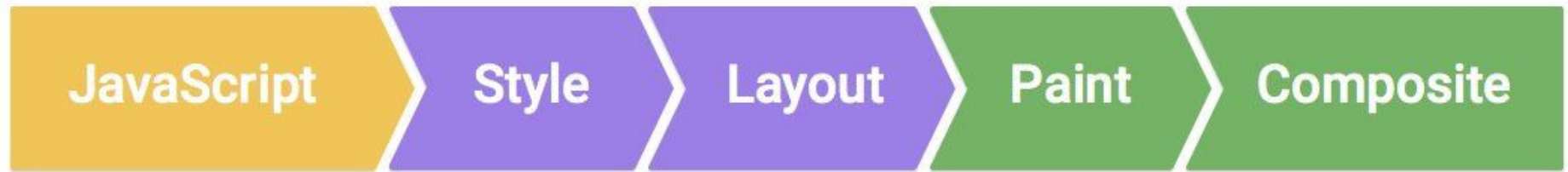


```
<div style="300px">
```

The Pixel Pipeline

```
box.style.width = '300px';
```

Recalculate element size, and other **affected elements** size.



```
<div style="300px">
```

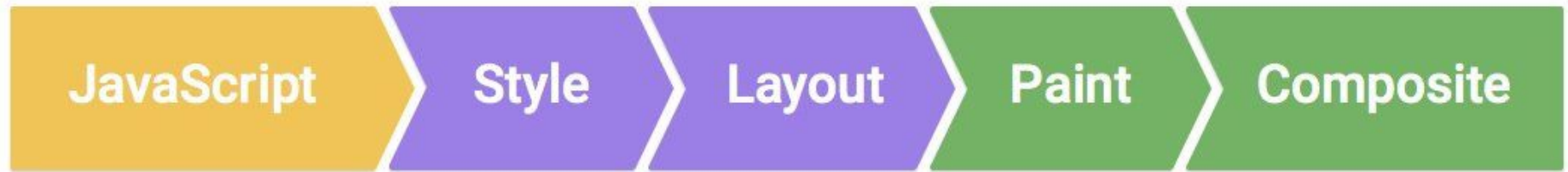
Filling pixels, render fonts, colors, borders, box shadows...

The Pixel Pipeline

```
box.style.width = '300px';
```

Recalculate element size, and other **affected elements** size.

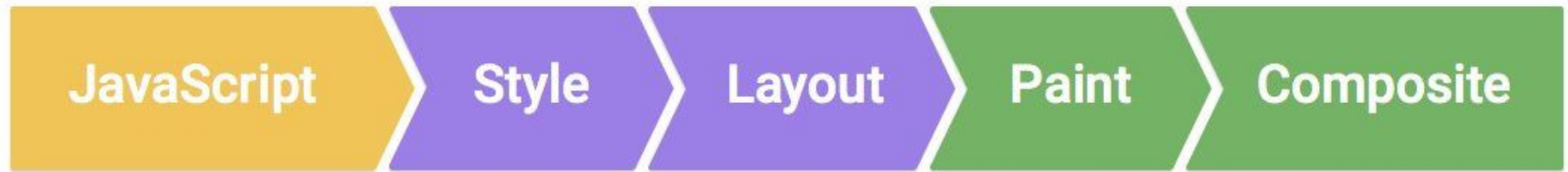
Composite the **printed layers**



```
<div style="300px">
```

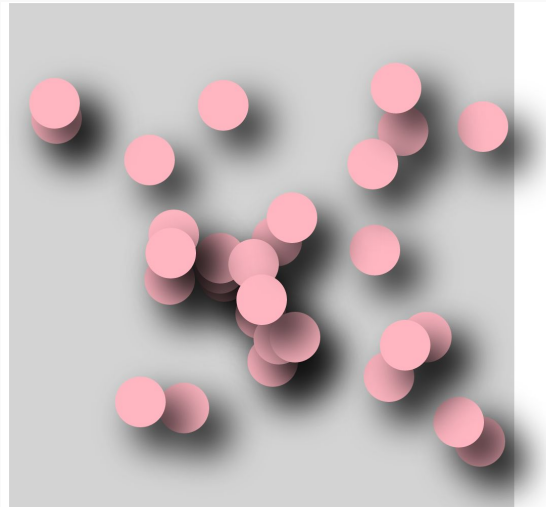
Filling pixels, render fonts, colors, borders, box shadows...

The Pixel Pipeline

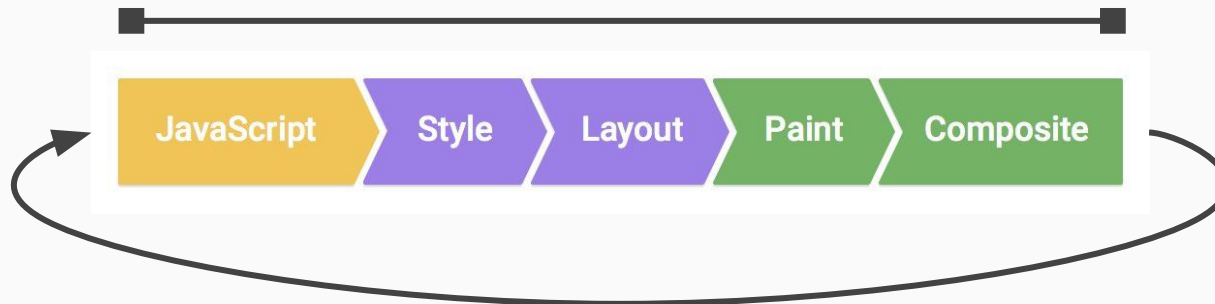


1 frame is created!

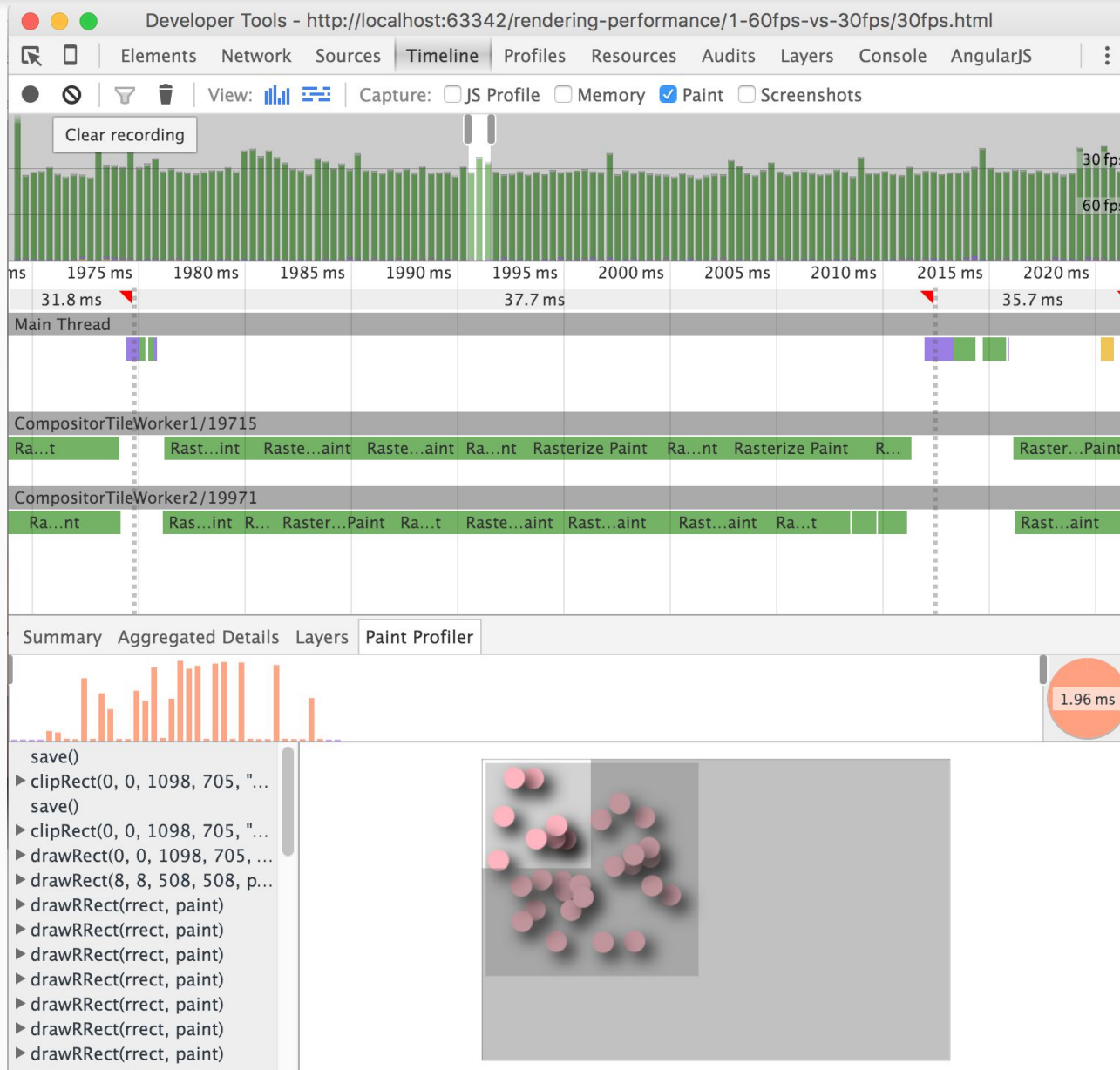
Every frame



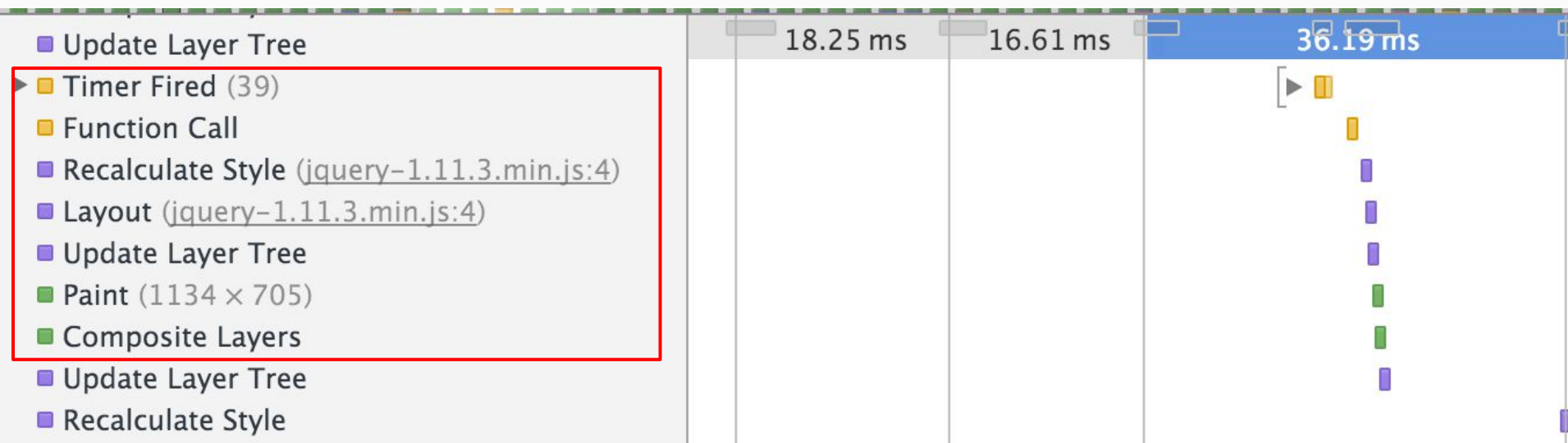
16ms



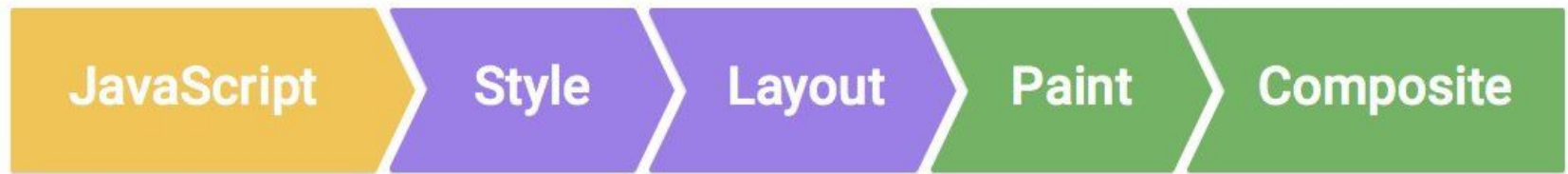
Understand the report



Use Chrome DevTools to inspect the steps



How to optimize?




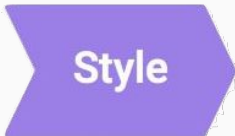



"Performance is the art of avoiding work"

1. Try to optimize in every single step
2. Avoid going through all the steps

Step by step:

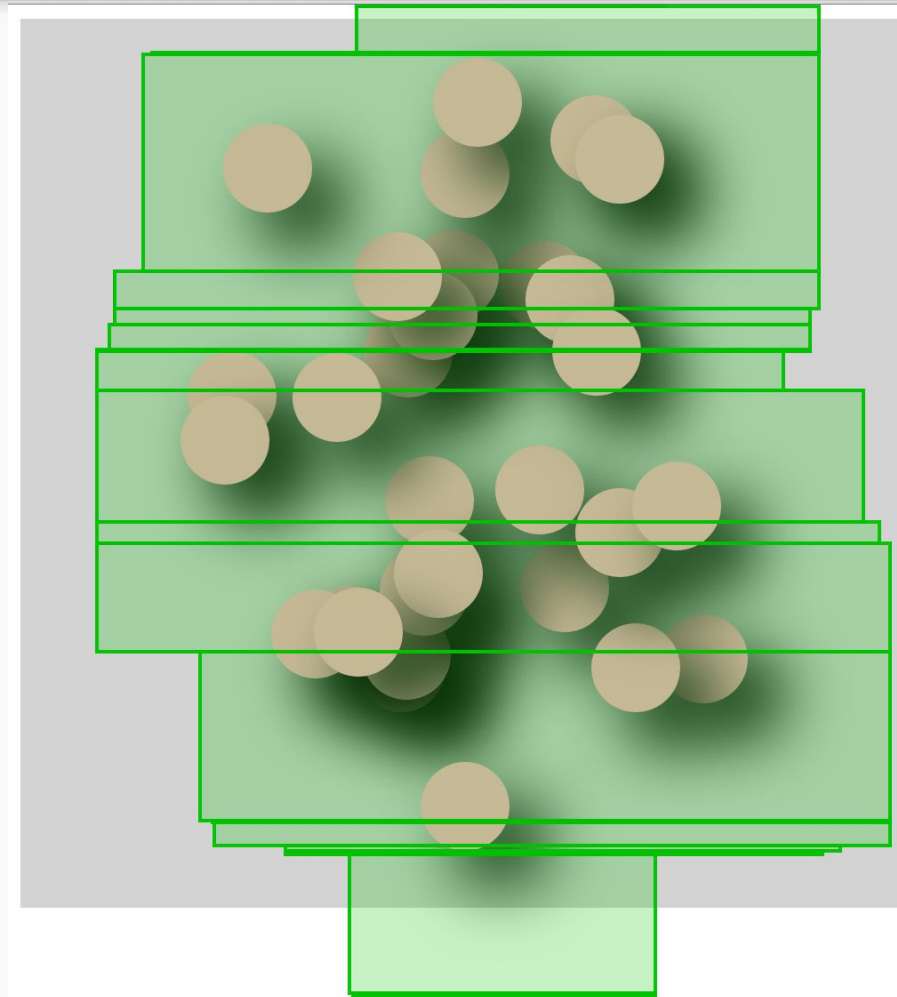
Optimize rendering performance

- 1  Paint
Paint is the most expensive work, avoid it as much as possible
- 2  Layout
Unnecessary layout triggers might cause performance issue
- 3  JavaScript
Don't do hard works on UI thread
- 4  Style
Avoid complex selectors
- 5  Composite
Reduce unnecessary steps in the pixel pipeline

Paint

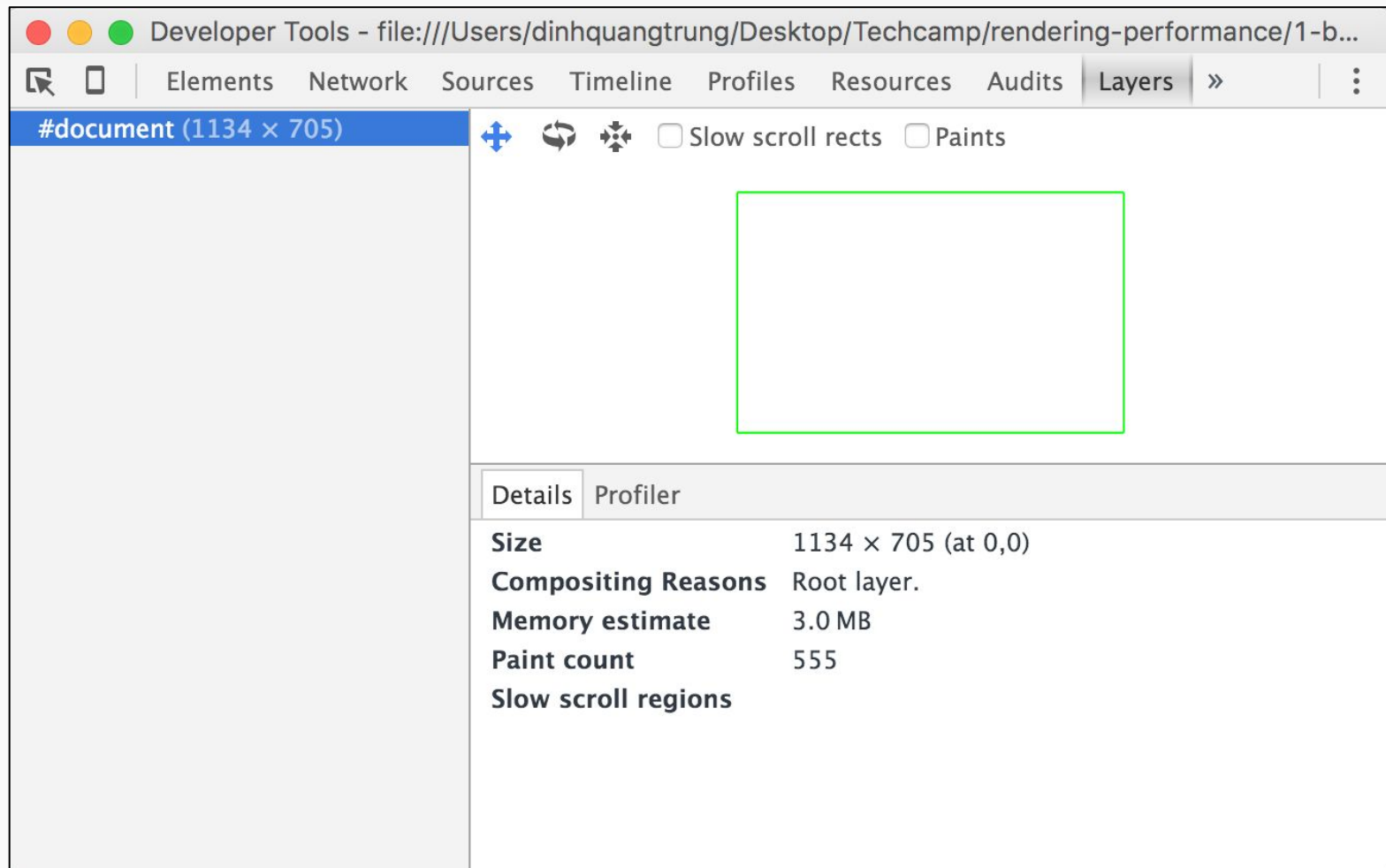
...is a hard work

Chrome DevTools



Use Chrome DevTools to see how browser draws

Chrome DevTools



Use Chrome DevTools to see how browser draws

How to create layers?

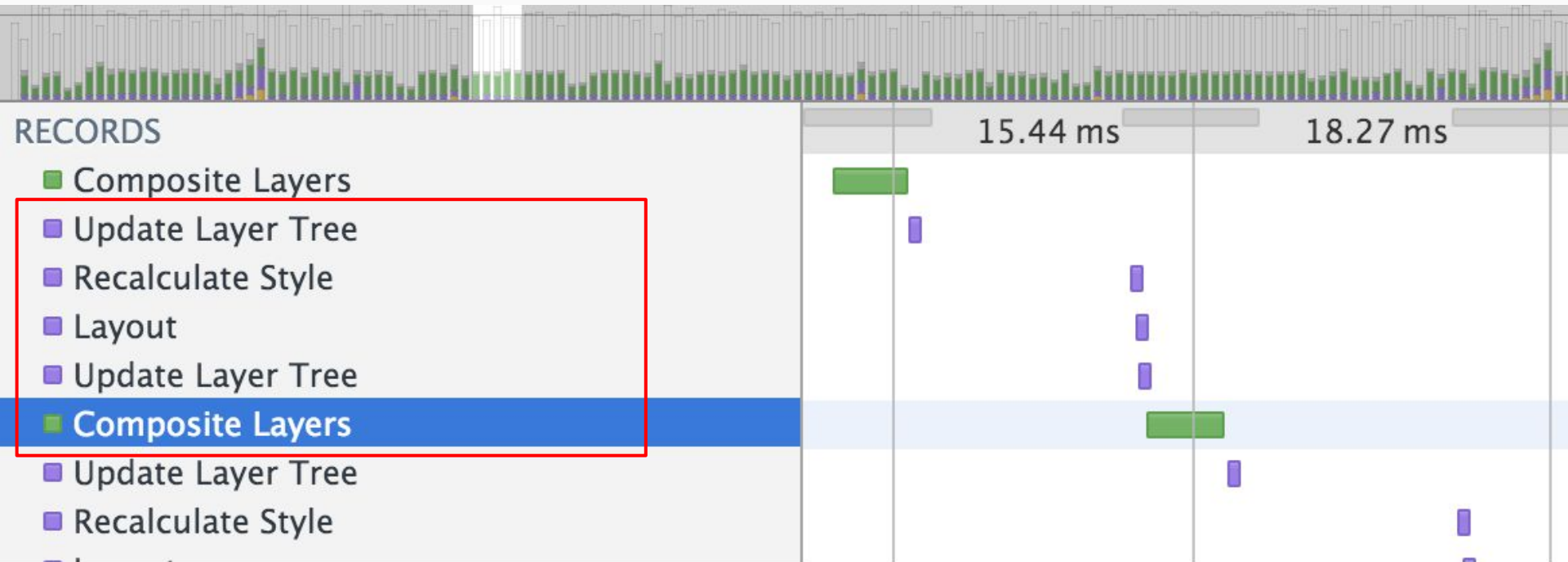
```
.bird {  
    will-change: transform;  
    transform: translateZ(0); // Old browsers  
}
```

Summary	Aggregated Details	Layers
▼ #document (1098 × 705)		<input type="checkbox"/> Slow scroll rects <input type="checkbox"/> Paints
.bird (170 × 170)		
.bird (170 × 170)		
.bird (170 × 170)		
.bird (170 × 170)		
.bird (170 × 170)		
.bird (170 × 170)		
.bird (170 × 170)		
.bird (170 × 170)		
.bird (170 × 170)		
.bird (170 × 170)		
.bird (170 × 170)		
.bird (170 × 170)		
.bird (170 × 170)		
.bird (170 × 170)		
.bird (170 × 170)		

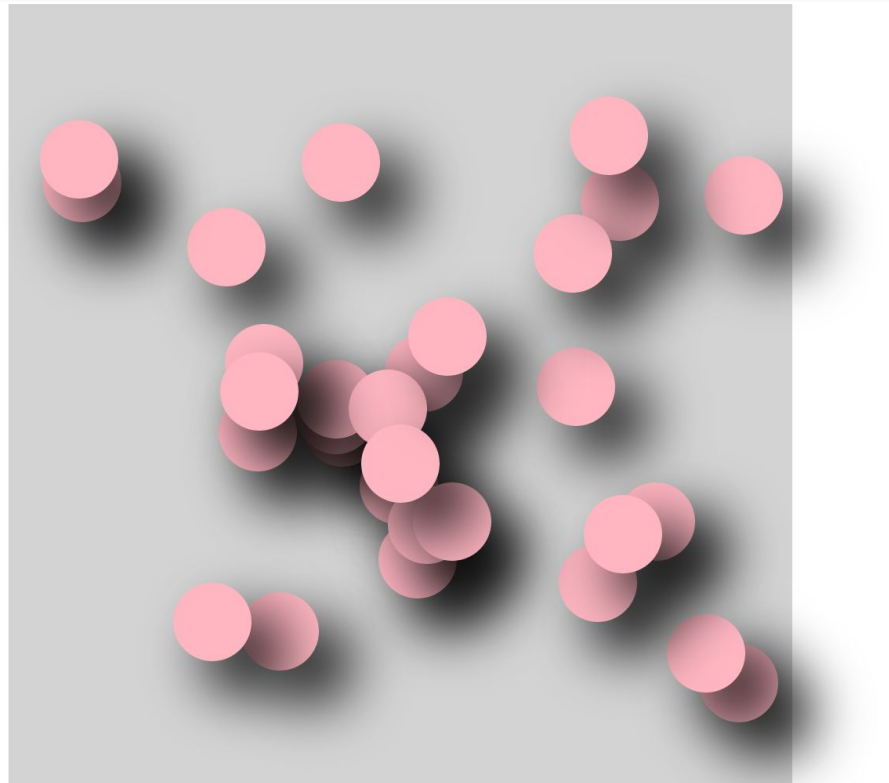
Size	170 × 170 (at -32,273)
Compositing Reasons	Has a will-change compositing hint.
Memory estimate	576 KB
Slow scroll regions	

60fps

No more paint



Let's see some examples



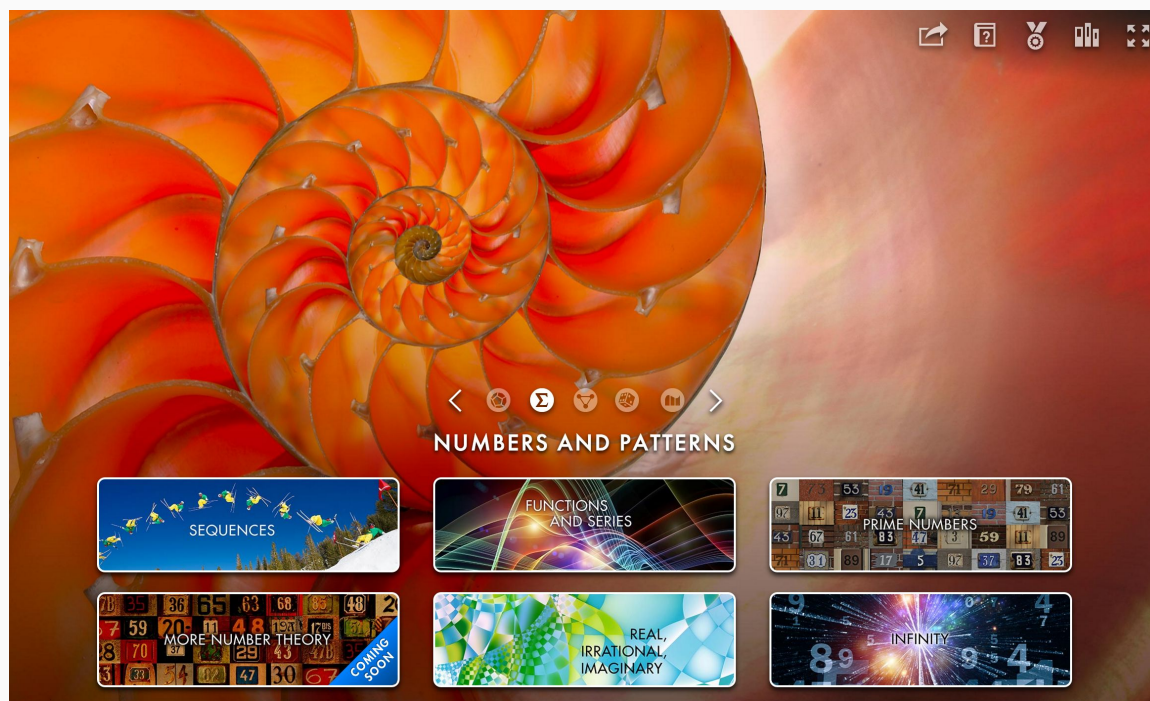
<http://trungdq88.github.io/render-performance-demo/2-composite-layers.html>

Let's see some examples



<http://matthew.wagerfield.com/parallax/>

Let's see some examples



<http://world.mathigon.org/>

Becareful!

- Composite layers require memory and management
- Create layer efficiently
- Avoid layer explosions

DON'T:

```
* {  
    will-change: transform;  
    transform: translateZ(0);  
}
```

- Don't create layers without profiling

Layout

Avoid layout whenever possible

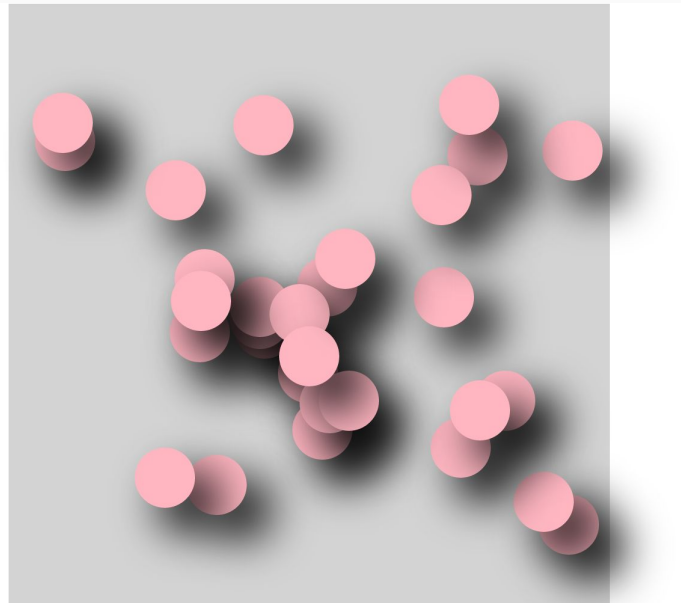
```
.box {  
  top: 20px;  
  left: 20px;  
}  
  
/**  
 * Changing top and left  
 * triggers layout.  
 */  
.box--moved {  
  top: 200px;  
  left: 350px;  
}
```

- Layout is normally scoped to the whole document.
- Changes to “geometric properties”, such as widths, heights, left, or top all require layout.

How to know which CSS properties trigger layout? Check out [CSS Triggers](http://csstriggers.com/).

<http://csstriggers.com/>

Avoid layout whenever possible



Improve the example

Avoid layout whenever possible

Self Time ▼	Total Time	Activity
534.7 ms 100.00 %	534.7 ms 100.00 %	▼ (unattributed)
226.6 ms 42.38 %	226.6 ms 42.38 %	■ Composite Layers
132.9 ms 24.86 %	132.9 ms 24.86 %	■ Update Layer Tree
114.8 ms 21.47 %	114.8 ms 21.47 %	■ Recalculate Style
48.6 ms 9.08 %	48.6 ms 9.08 %	■ Layout
11.8 ms 2.21 %	11.8 ms 2.21 %	■ Major GC

Change **top**, **left**
triggers layout

View Costly Functions ▼		Group by Domain ▼	
Self Time ▼	Total Time	Activity	
318.8 ms100.00%	318.8 ms100.00%	▼ (unattributed)	
142.9 ms 44.83%	142.9 ms 44.83%	■ Composite Layers	
122.0 ms 38.27%	122.0 ms 38.27%	■ Recalculate Style	
42.6 ms 13.36%	42.6 ms 13.36%	■ Update Layer Tree	
11.3 ms 3.55%	11.3 ms 3.55%	■ Major GC	

Change **transform**:
translate3d does not
trigger layout

JavaScript

Use requestAnimationFrame

```
/**
 * If run as a requestAnimationFrame callback, this
 * will be run at the start of the frame.
 */
function updateScreen(time) {
    // Make visual updates here.
}

requestAnimationFrame(updateScreen);
```



A horizontal timeline diagram consisting of a light blue bar with three vertical dark blue tick marks. The first tick mark is at the left end, the second is in the middle, and the third is at the right end. Below the bar, the text '16ms' is centered under the first interval, and '16ms' is centered under the second interval.

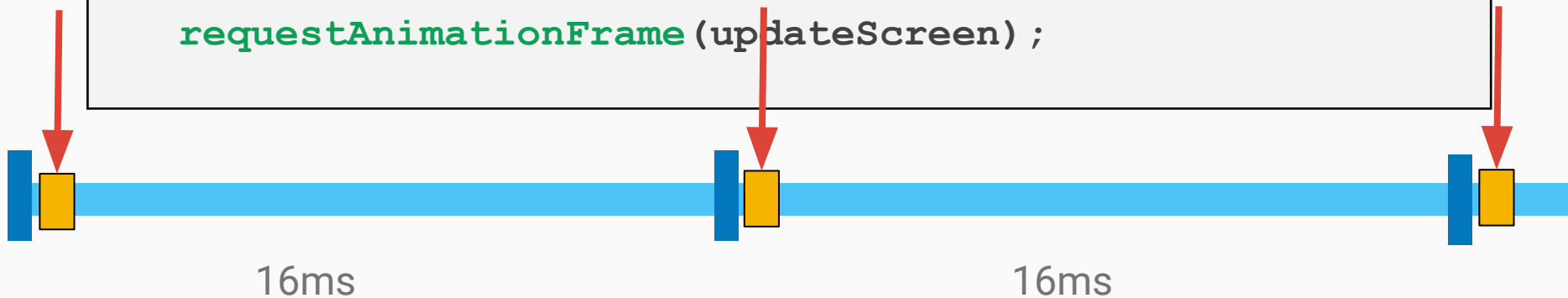
16ms

16ms

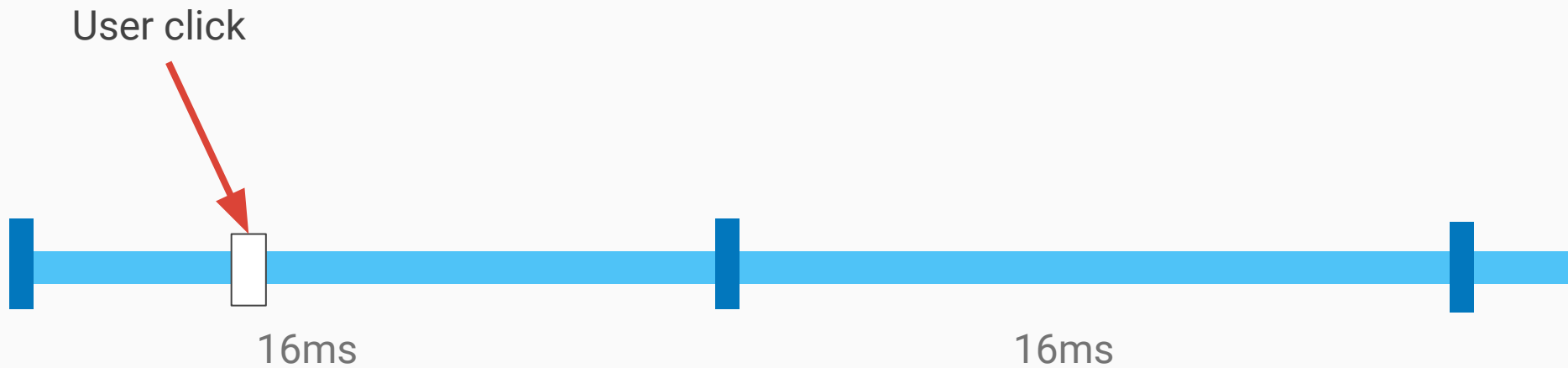
Use requestAnimationFrame

```
/**
 * If run as a requestAnimationFrame callback, this
 * will be run at the start of the frame.
 */
function updateScreen(time) {
  // Make visual updates here.
}

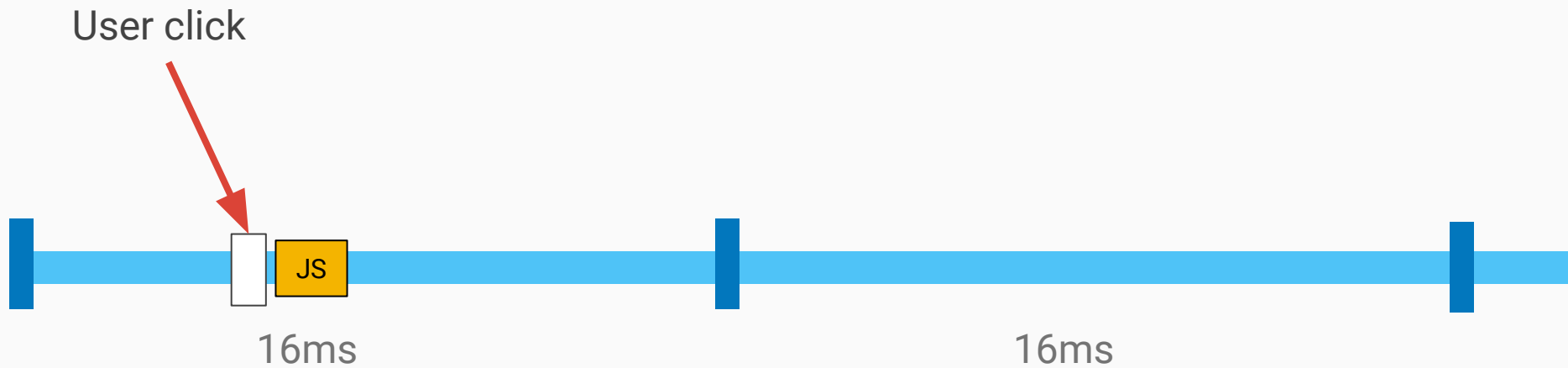
requestAnimationFrame(updateScreen);
```



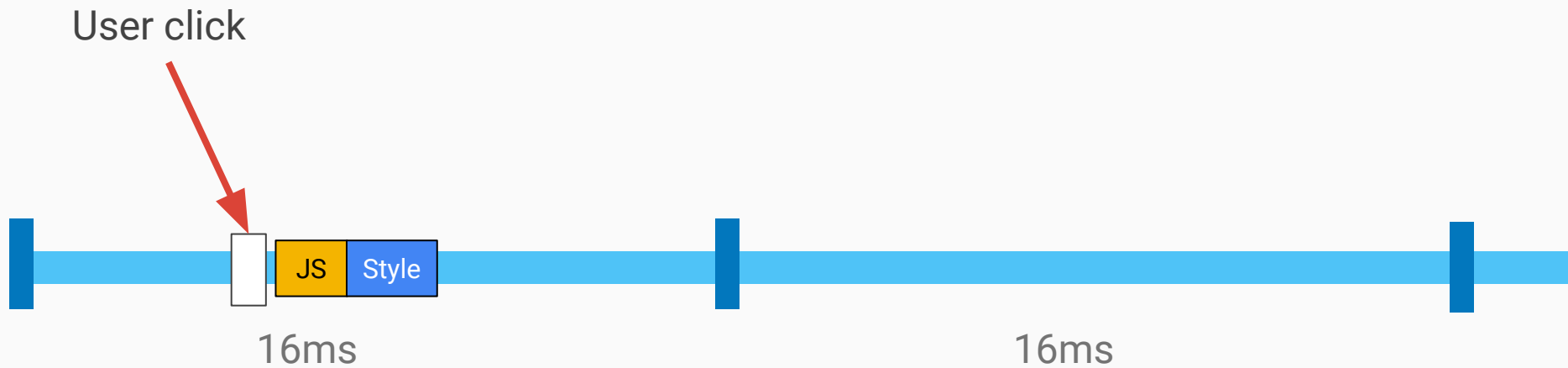
Use requestAnimationFrame



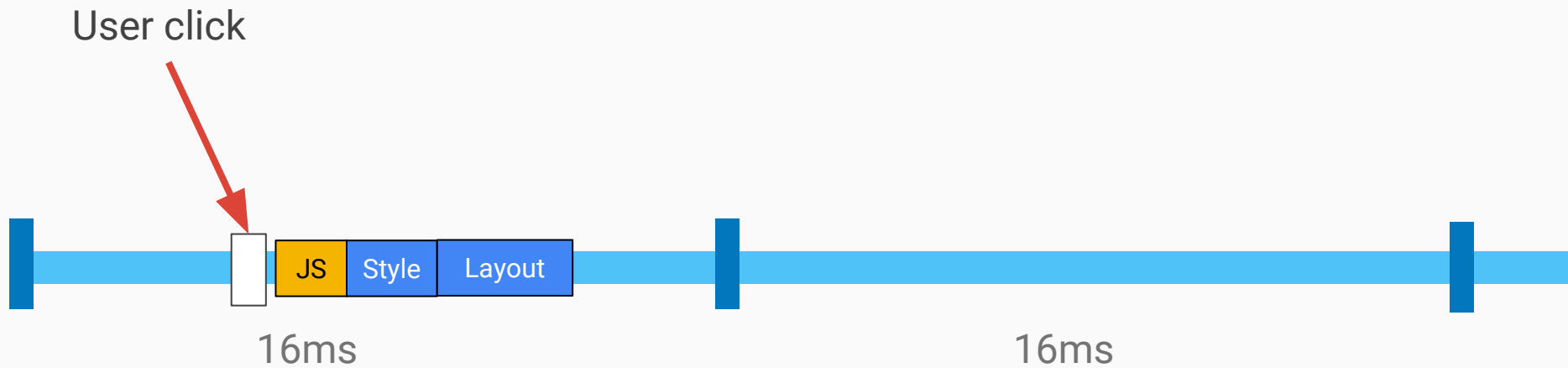
Use requestAnimationFrame



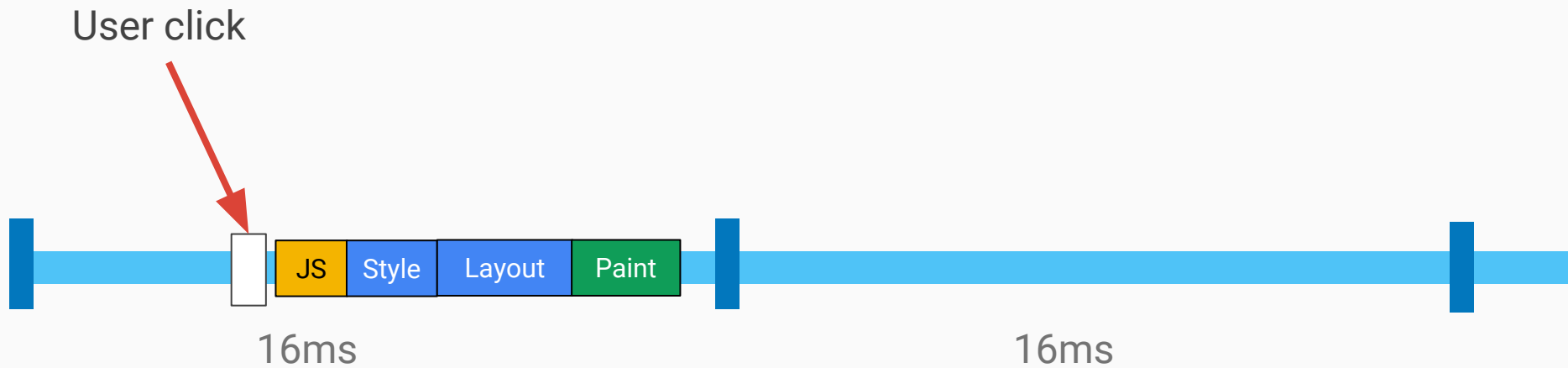
Use requestAnimationFrame



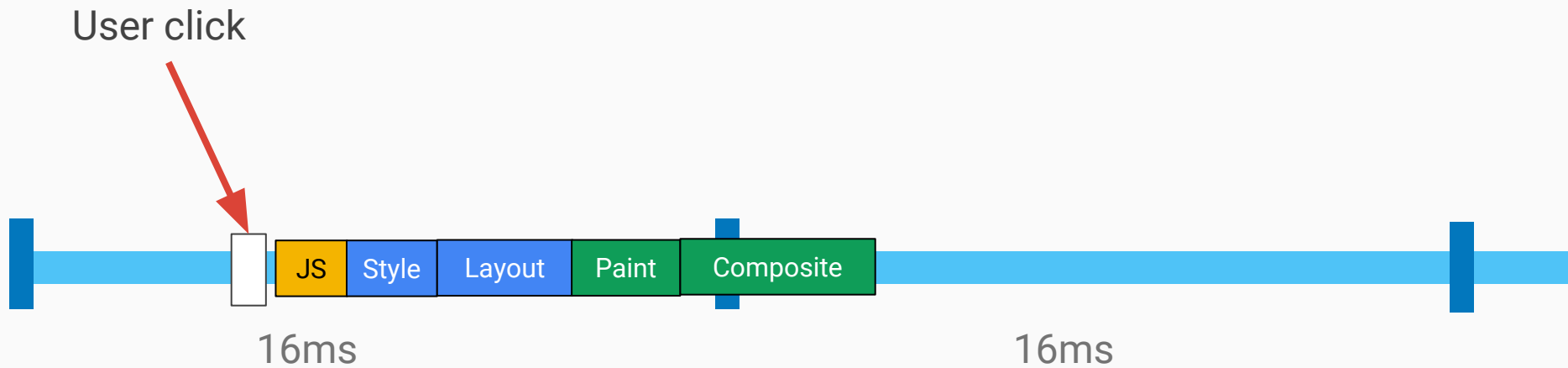
Use requestAnimationFrame



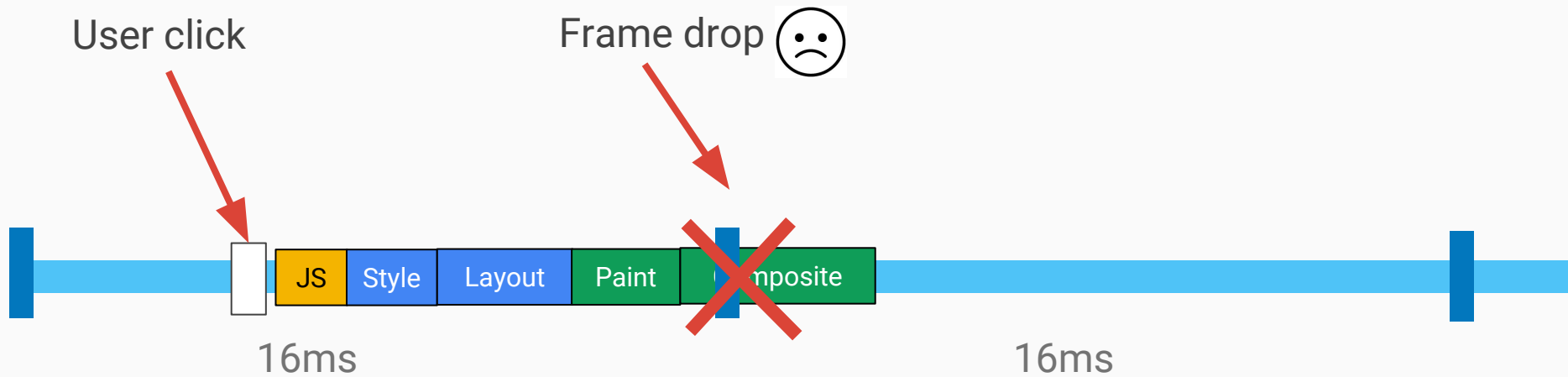
Use requestAnimationFrame



Use requestAnimationFrame



Use requestAnimationFrame



Use requestAnimationFrame

User click



16ms



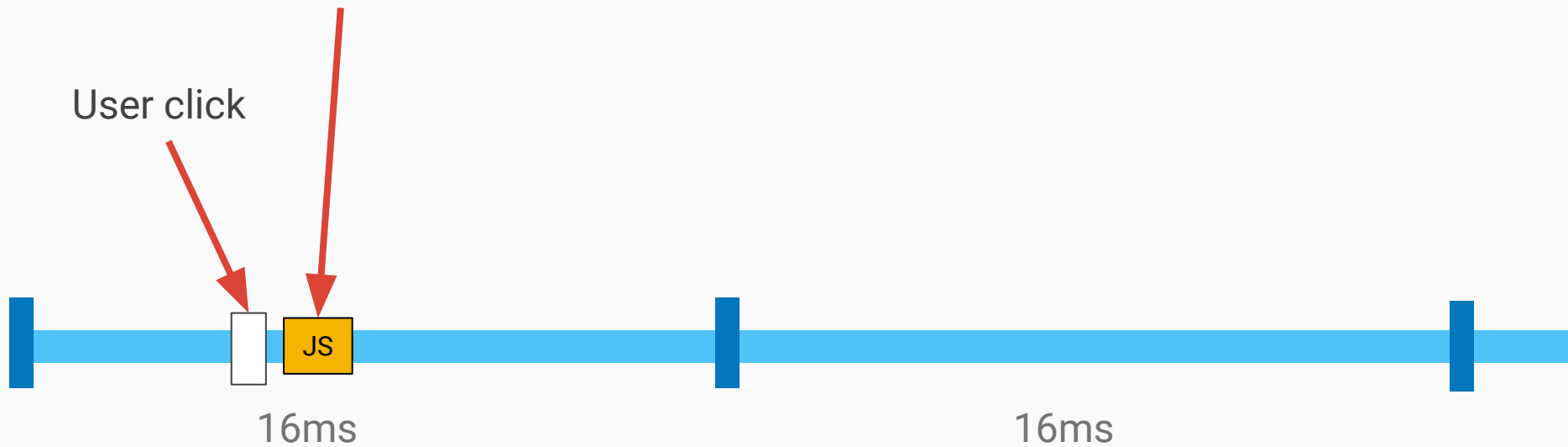
16ms



Use requestAnimationFrame

call **requestAnimationFrame(...)**

User click

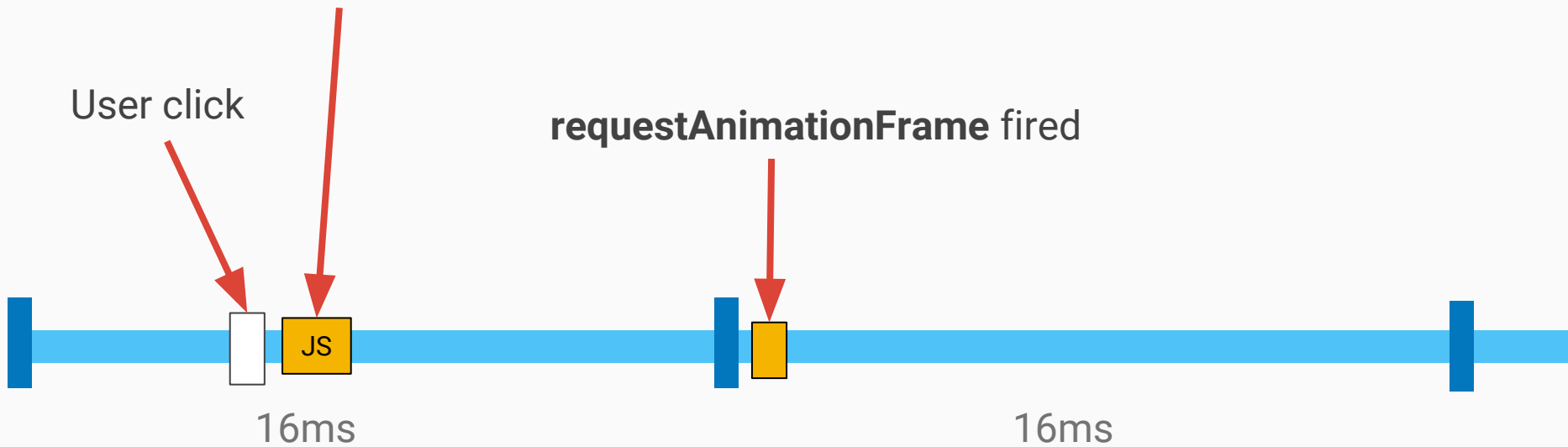


Use requestAnimationFrame

call `requestAnimationFrame(...)`

User click

`requestAnimationFrame` fired

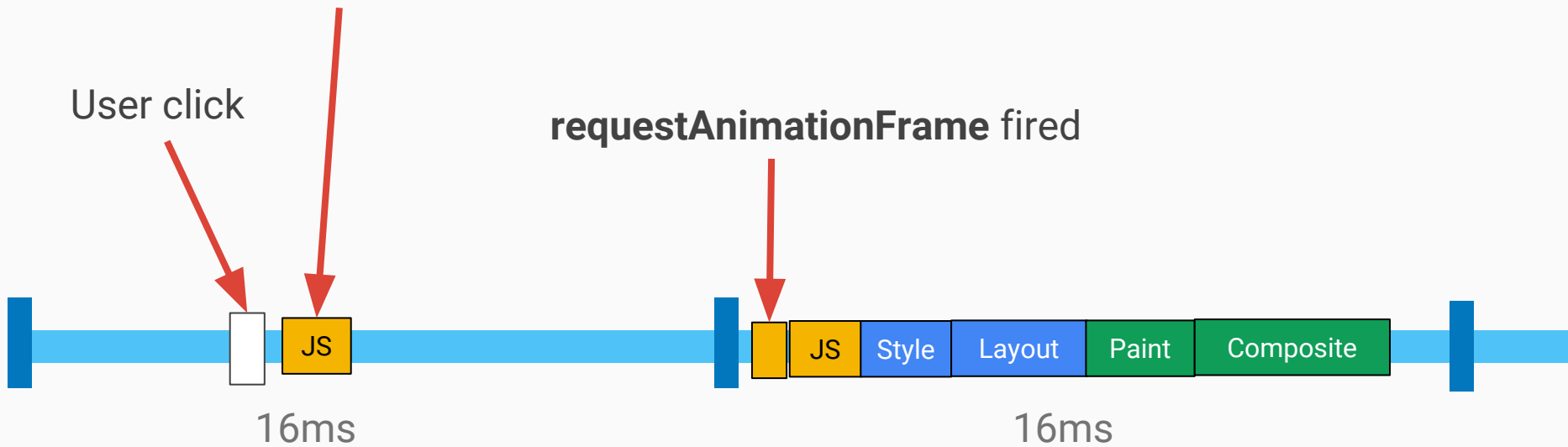


Use requestAnimationFrame

call `requestAnimationFrame(...)`

User click

`requestAnimationFrame` fired

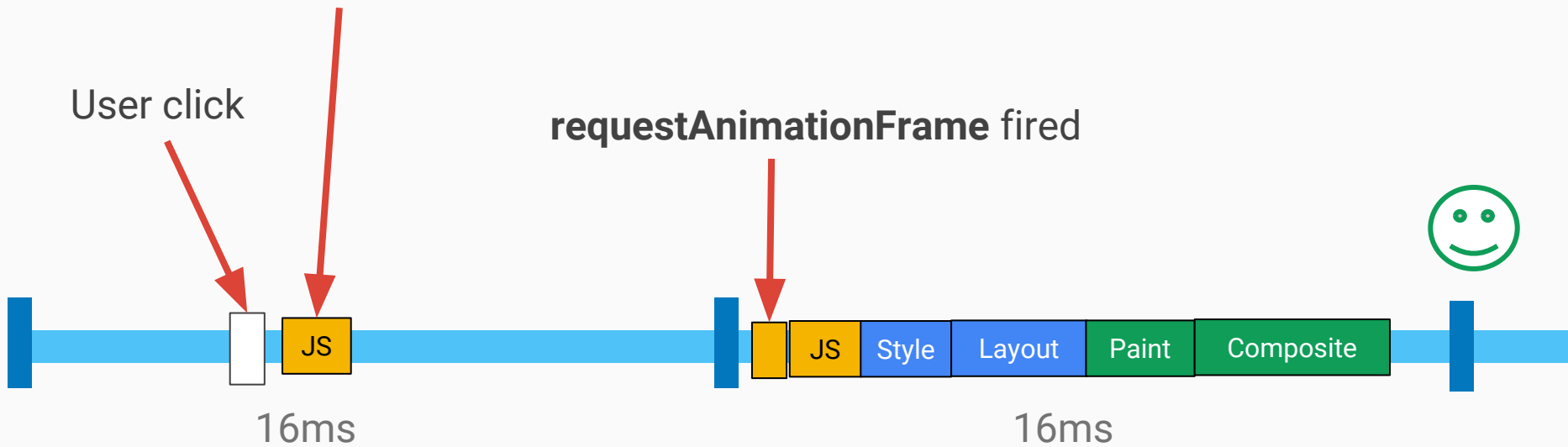


Use requestAnimationFrame

call `requestAnimationFrame(...)`

User click

`requestAnimationFrame` fired

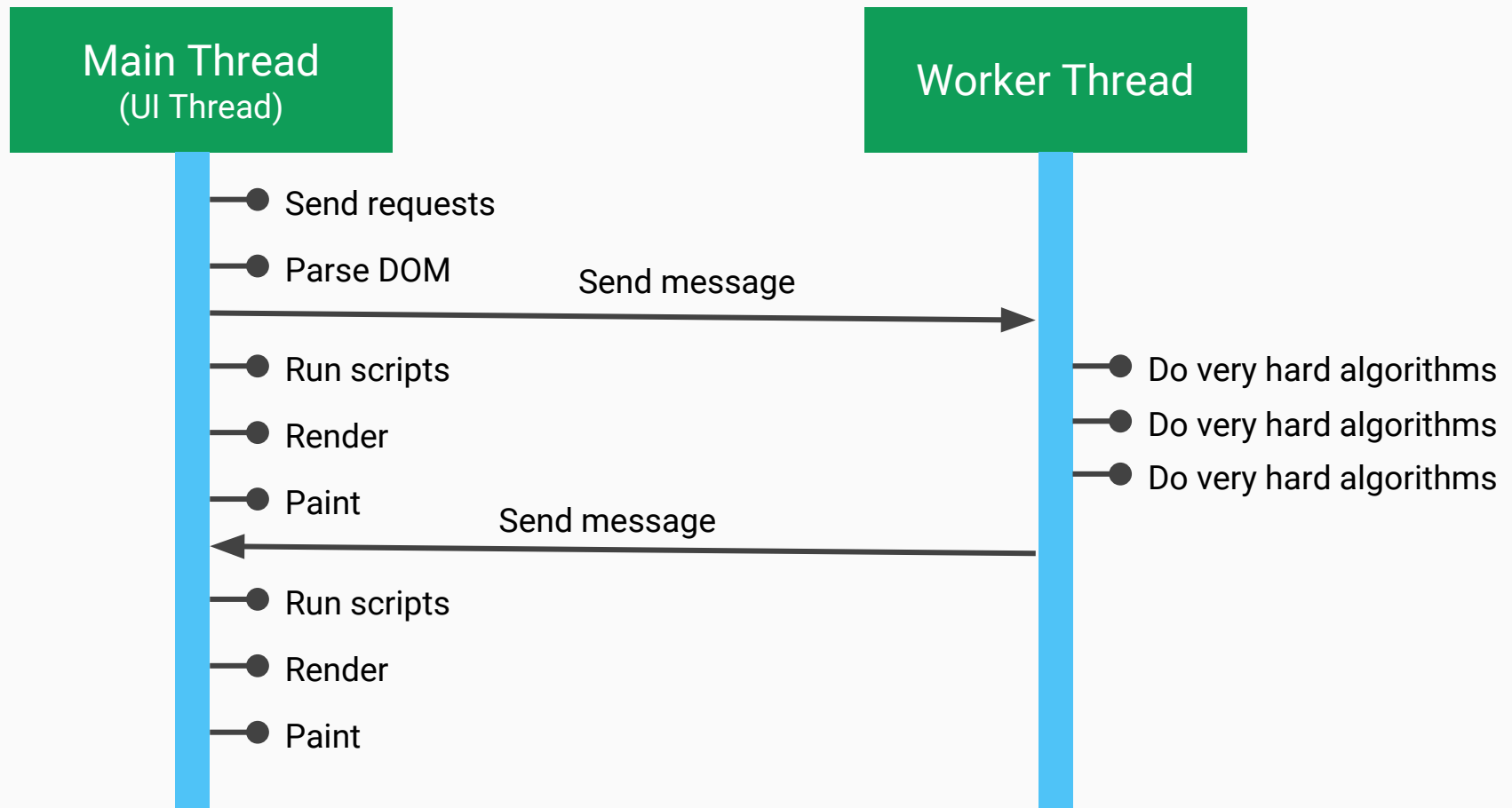


Use requestAnimationFrame



Improve the example

Use Web Workers for hard works



Use Web Workers for hard works

```
var dataSortWorker = new Worker("sort-worker.js");
dataSortWorker.postMessage(dataToSort);

// The main thread is now free to continue working on other things...

dataSortWorker.addEventListener('message', function(evt) {
    var sortedData = evt.data;
    // Update data on screen...
});
```

Web Worker Guide: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers

Other tips

Avoid complex style

```
.title {  
  /* styles */  
}
```

DO 😊

```
.box:nth-last-child(-n+1) .title {  
  /* styles */  
}
```

DON'T 😞

Manage your CSS

- Use Block, Element, Modifier
- PostCSS

```
.list { }  
.list__list-item { }
```

```
.list__list-item--last-child {}
```

Avoid going through all the steps

JavaScript

Style

Layout

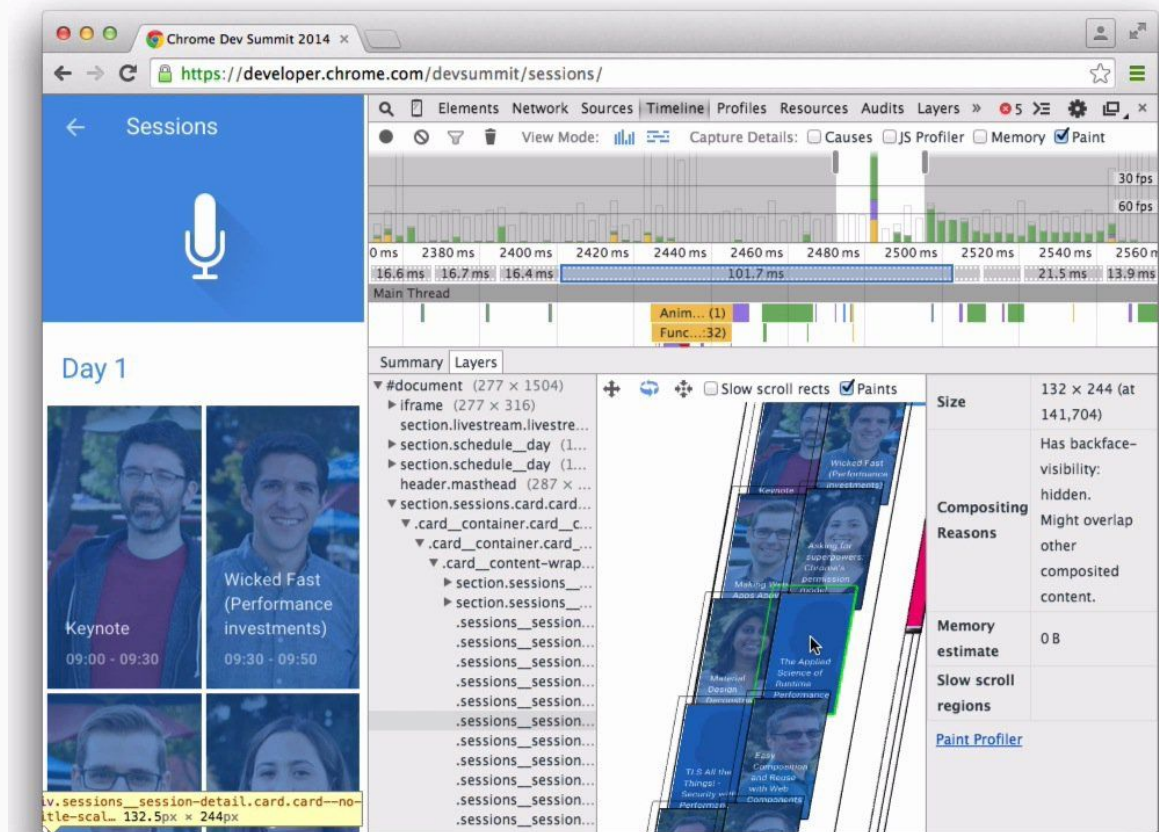
Paint

Composite

Position	<code>transform: translate(npx, npx);</code>
Scale	<code>transform: scale(n);</code>
Rotation	<code>transform: rotate(ndeg);</code>
Skew	<code>transform: skew(X Y) (ndeg);</code>
Matrix	<code>transform: matrix(3d)(...);</code>
Opacity	<code>opacity: 0...1;</code>

(The element will need to be on its own compositor layer.)

Use Chrome DevTools to profile your app




Keep your knowledge up to date!

<https://developers.google.com/web/updates/>

Updates

Be smarter, do more.

Discover the latest API's coming to the Web Platform, find out what the Chrome team are working on and check out the latest features in DevTools.



Bengaluru

City in India

Manage the triggering of Touch to Search

Weather: 30°C, Wind: NE at 12 mph (19 km/h), 38% Humidity

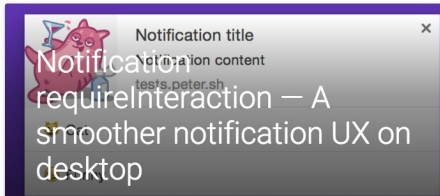
Local time: Thursday 14:05

Understanding when and how Touch to Search is triggered

Paul Kinlan

October 22, 2015

Detecting if a web app is



Notification title

Notification content

requireInteraction — A smoother notification UX on desktop

Notifications on desktop will be automatically dismissed after a short period of time.

Paul Kinlan

October 16, 2015

Adding a Splash screen for

Recap

Paint

- Paint is the **most expensive work**
- Manage your **composite layers**

Layout

- Avoid **trigger layout**
- Use **Flexbox**
- Avoid **forced synchronous layout**

JavaScript

- Use **requestAnimationFrame**
- Use **Web workers** for hard works

Style

- Reduce selector **complexity**
- Reduce **number of element affected** by styles

Composite

- Use **compositor-only properties**

Useful resources for Web Performance

This presentation:

- This slide: <http://j.mp/web-performance-slide>

- Full article: <http://j.mp/web-performance-trungdq88>
- Example source code: <https://github.com/trungdq88/render-performance-demo>

Guides

- Google Developer: <https://developers.google.com/web/fundamentals/performance/>
- Preventing Layout Thrashing: <http://wilsonpage.co.uk/preventing-layout-thrashing/>

Course

- Udacity Course: Browser Rendering Optimization - Building 60 FPS Web Apps: <https://www.udacity.com/course/browser-rendering-optimization--ud860>

Nice webs

- <http://world.mathigon.org/>
- <http://matthew.wagerfield.com/parallax/>

Other resources

- Web Workers: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/basic_usage
- BEM: <https://bem.info/>
- PostCSS: <https://github.com/postcss/postcss>
- CSS Trigger: <http://csstriggers.com/>
- Guide to Flexbox: <http://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Thank you



fb.com/trungdq88



[@trungdq88](https://twitter.com/trungdq88)



github.com/trungdq88