Gone in 60 frames per second

Rendering Performance Case Studies
Addy Osmani - @addyosmani
Chrome Developer Relations
Google
3 Pillars Of Performance

Network  Compute  Render
Our agenda for today...

1. DOM to pixels on the screen
2. Rendering performance tooling
3. Real-world case studies

- Flickr
- Bootstrap
- Pitchfork
- Pinterest
- Google+
What we build for the web is **evolving**.
Users ❤️ snappy experiences.
High performance web apps have:

Silky smooth scrolling
Buttery animation
Great performance everywhere.
Speed must be treated as an essential design feature.
The fastest web page is..
Everything we add increases the work the browser has to do to put pixels on the screen.
Addy Osmani originally shared:
DevTools snippet: save an object to a JSON file with console.save

http://bgrins.github.io/devtools-snippets/#console-save
Mobile web performance goals

1. Connectivity
   - Show above the fold content in < 1s
   - Serve critical path CSS in first 14KB

2. Max of 200ms server response time

3. 60fps scrolling, 60fps transitions

4. Speed index under 1000*

* average time visual parts of the page display per WebPageTest
Today we’ll focus on this.

1. Connectivity
   - Show above the fold content in < 1s
   - Serve critical path CSS in first 14KB
2. Max of 200ms server response time
3. 60fps scrolling, 60fps transitions
4. Speed index under 1000
Rendering performance impacts user experience.
In an A/B test, we slowed down scrolling from 60fps down to 30fps. Engagement collapsed

~ Shane O'Sullivan

* in their native app, fluctuating between 30 to 45fps.
* Consistent 30fps performed second best
"We tested pre-fetching JS in search results, which caused jank in our pages. All business metrics got worse"

~ Jonathan Klein
Response rates matter.
See the flow of how Chrome renders pages
30fps = 33ms per frame, 60fps = 16ms per frame
what do these records mean?
Let’s dive in!
Make a request

GET / HTTP /1.1
host: www.example.com
<!DOCTYPE html>
<html class="no-js">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>MA RESPONSE</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="css/main.css">
  </head>
  <body>
    <section>
      <h1>HTML wizaaaaaard</h1>
      <p>I am teh HTML masterz.</p>
    </section>
  </body>
</html>
Get a response

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<title>MA RESPONSE</title>
<meta name="description" content="">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/main.css">
</head>
<body>
<section>
<h1>HTML wizaaaaard</h1>
<p>I am teh HTML masterz.</p>
</section>
</body>
</html>
Get a response
Recalculate Style
Styling the DOM

DOM + CSS

Recalculate Style
Styling the DOM

Render tree
Styling the DOM
section p {
    display: none;
}
section h1:after {
    content: "<333 pseudo elemz"
}
Layout
Laying out the document

html, body {
    margin: 0;
    width: 300px;
    height: 700px;
    background: white;
    color: white;
}

body {
    background: #888;
}

section {
    display: block;
    margin-top:30%;
    padding-top:60px;
    width:100%;
    background:#444;
}

section h1:after{
    content: '<3 pseudo';
    height: 40px;
    margin-top: 10px;
    display: block;
}

img {
    margin: 30px;
    border-radius: 4px;
    border: 3px solid white;
    box-shadow: 0 2px 2px rgba(0,0,0,3);
}
<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>drawPoints</td>
<td>drawArc</td>
</tr>
<tr>
<td>drawOval</td>
<td>clear</td>
</tr>
<tr>
<td>drawRectangle</td>
<td>translate</td>
</tr>
<tr>
<td>drawRoundedRect</td>
<td>save</td>
</tr>
<tr>
<td>drawPath</td>
<td>restore</td>
</tr>
<tr>
<td>drawBitmap</td>
<td>clipPath</td>
</tr>
<tr>
<td>drawText</td>
<td>moveTo</td>
</tr>
<tr>
<td>drawLine</td>
<td>lineTo</td>
</tr>
</tbody>
</table>
Paint
10. drawRoundedRectangle
11. restore
12. drawPath
13. save
14. clipRoundedRectangle
15. drawBitmap
16. restore
17. translate
18. restore

Welcome to ma web paaaage!
<3 pseudo
Instant replay

10. drawRoundedRectangle
11. restore
12. drawPath
13. save
14. clipRoundedRectangle
15. drawBitmap
16. restore
17. translate
18. restore
Image Resize
Draw Bitmap

Image Decode + Resize
Move Elements
Composite Layers
Layers

Composite Layers
Layer creation criteria

3D or perspective transform
HW-accelerated `<video>` element
HW-accelerated `<canvas>` element (2D or 3D)
Contains a composited plugin
CSS animation on opacity
Animated webkit transform
Accelerated CSS filters
Rendered on top of another layer
To the screen

CPU

GPU

Composite Layers
All together, you should expect to see..
DOM to pixels on the screen

**Recalc styles**
Calc styles that apply to elements

**Layout**
Generate geometry for each element

**Paint**
Fill pixels for each element into layers (Paint)

**Composite layers**
Draw layers out to the screen
What about frame rate?
Frame rate

Rate at which a device produces consecutive images to the screen
To see what’s impacting rendering, look at FPS
A consistent frame rate is our ideal.
Why target 60fps?
Match the **refresh rate** of the devices you are targeting.

<table>
<thead>
<tr>
<th>Device</th>
<th>Lower-end refresh rate</th>
<th>Average refresh rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phones</td>
<td>55hz</td>
<td>60hz</td>
</tr>
<tr>
<td>Laptops</td>
<td>58hz</td>
<td>60hz</td>
</tr>
<tr>
<td>Monitors</td>
<td>50hz</td>
<td>62hz</td>
</tr>
</tbody>
</table>
Jank

Disruption in consistent frame rate that manifests itself visually
< 15fps

Your users will perceive **jerkiness** and jank

15 fps
30fps

Smooth as long as it’s a constant 30fps
60fps
Smooth animations and transitions with **no stutter**
Frame budget

At 60fps, you have 16.7ms budget for

Logic processing
Compute processing
Rendering

✓ Scripting ✓ Rendering ✓ Painting ✓ Misch
Frame budget

It’s more like **8-10ms** budget because

- Browser, JS engine, renderer processes
- Margin for slower devices like mobile.

✅ Scripting ✅ Rendering ✅ Painting ✅ Misch
What causes jank?
Properties that trigger layout (reflow)

- `clientHeight`
- `clientLeft`
- `clientTop`
- `clientWidth`
- `focus()`
- `getBoundingClientRect()`
- `getClientRects()`
- `innerText`
- `offsetHeight`
- `offsetLeft`
- `offsetParent`
- `offsetTop`
- `offsetWidth`
- `outerText`
- `scrollByLines()`
- `scrollByPages()`
- `scrollHeight`
- `scrollIntoView()`
- `scrollIntoViewIfNeeded()`
- `scrollLeft`
- `scrollTop`
- `scrollWidth`
Reading offsetTop in a loop
Cache that read outside loop or don't use offsetTop

Layout determines how to render the HTML elements in appropriate styles and sizes.
Heavy styles can cause jank*

Drop-shadows
Blurs
Linear-gradients
Fixed background images

*Correct as of November, 2013. Likely to change!
CSS Styles that affect paint, layout
Performance playground

Resize this:

Then:

Set block width  Go slow

Gradient  Border Radius  Box Shadow  Blur Filter  Background  Alt-BG

diam elementum suspendisse adipiscing fermentum lorem ultrices molestie sed sagittis
evulputate in lobortis lacus, sodales eget ut facilisi fringilla est
egestas eu ornare arcu vel, at neque auctor ullamcorper lectus
ut placerat libero vitae donec tempus in purus nisl venenatis
eleifend vivamus ut nam aenean nulla eros praesent urna, dolor
sollicitudin sagittis tortor sit tincidunt amet quis magna sclerisque adipiscing
quam purus tortor aliquet amet tincidunt etiam sit vel metus
adipiscing quis ut libero vitae sapien a commodo egestas suspendisse
viverra sit facilisis blandit enim, sapien sociis ut ipsum varius
voluptat fusce aenean vestibulum dolor volutpat, felis enim senectus nam
gravida rutrum in iaculis lectus nec nec tempor non vehicula
sem diam, morbi euismod enim euismod tincidunt ac, aliquam aliquet
viverra ultricies felis sed eget suscipit nibh sed eros donec
elit, magna gravida adipiscing mi diam aliquam sit rice orci
vitae, varius non auctor purus turpis morbi tellus viverra massa
pellentesque commodo, eget nibh sit ultricies id condimentum ultrices a...
Recalculate style triggered when styles are computed or changed.

Heavy use of JS to rearrange the page (e.g. onscroll) is bad.
Heavy `onscroll()` handlers

Doing shit inside of scroll is terrible
Heavy onScroll  ✓ Costly effects
Scrolling
Watch out for:

Unnecessary paints:
- position:fixed
- overflow:scroll
- hover effects
- touch listeners

Long paints:
- Complex CSS
- Image decodes
- Large empty layers

Correct as of November, 2013.
Long image decodes and resizes

Remember to pre-scale your images!
THE PYRAMID BLOWOUT WAS AKIN to your typical rock star extravaganza in scale and scope, but also laced with the more inclusive and diffusive aspects of traditional DJ gigs, where everyone's the star. It put Daft Punk in a unique position within contemporary music's personality-driven ecosystem: legitimately famous and faceless. To this point, Bangalter compares their situation to Batman ("we feel that the pyramid was like our Batmobile"), Cinderella ("after the show is over, we go back to anonymity and normality"), the Wizard of Oz ("we're just guys behind a curtain pushing the knobs and creating the spectacle"), and a dude in a Mickey Mouse costume at Disney World ("if you have 100 kids around you all day long, are you not becoming big-headed?"). Their mechanized identities also act as a buffer for the out-of-control egomania that could result from a sea of people losing their shit in your general direction as you stand over them from the apex of a million-watt triangle.

"Looking at robots is not like looking at an idol," contends de Homem-Christo. "It's not a human being, so it's more like a mirror—the energy people send to the stage bounces back and everybody has a good time together rather than focusing on us." Also, it turns out those helmets make it pretty hard to, you know, see. "The visors are very, very tinted, and I'm shortsighted, anyway," says Bangalter. "I could hear the clamor, but I have hardly any visual memory of the tour aside from looking at our controllers."

Just as their costumes put up a physical boundary between themselves and their audience, Daft Punk enjoy a "total separation" between their private and public lives, which is crucial to their success. "It's all very much about being anonymous," says de Homem-Christo. "The fact that we're not famous and faceless is the thing that people think we are, is part of our charm. It's our entire thing."

Should be pre-scaled. Instead using overly high-res sources with more costly resizes.
DOM elements with measurably high paint cost

Measure what elements might be slowing you down.
Too much inside your event handlers
On some operating systems (not OSX) input events can arrive as often as every 2ms!
Heavy animation or data processing

Where possible rely on rAF or Web Workers
4 things a browser can animate cheaply

<table>
<thead>
<tr>
<th></th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td><code>transform: translate(npx, npx);</code></td>
</tr>
<tr>
<td>Scale</td>
<td><code>transform: scale(n);</code></td>
</tr>
<tr>
<td>Rotation</td>
<td><code>transform: rotate(ndeg);</code></td>
</tr>
<tr>
<td>Opacity</td>
<td><code>opacity: 0...1;</code></td>
</tr>
</tbody>
</table>

Move all your visual effects to these things. Transition at your own risk.

*translateZ(0)* or *translate3D()* may be required
@keyframes animation of top, left

@keyframes animation of transform: translate()

captured at 240 FPS
We're going to cut straight to the chase. Modern browsers can animate four things really cheaply: position, scale, rotation and opacity. If you animate anything else, it's at your own risk, and the chances are you're not going to hit a silky smooth 60fps.

4 things a browser can animate cheaply

**Position**

```
transform: translate(npx, npx);
```

**Scale**

```
transform: scale(n);
```

**Rotation**

```
transform: rotate(ndeg);
```
Hardware acceleration
GPU compositing
Old-skool painting

All your elements get painted into one big bitmap
Old-skool painting

What if we had separate bitmaps, or "layers"? Click.
Old-skool painting
What if we had separate bitmaps, or "layers"? Click.
Layers & Compositing

Hardware compositing uses the GPU to help build the page. Elements are broken out to a bunch of layers. Those layers are uploaded to the GPU as textures. The GPU composites those textures together.
More: DevTools Settings

- Show paint rectangles
- Force accelerated compositing
- Show composited layer borders
- Show FPS meter
- Enable continuous page repainting
- Show potential scroll bottlenecks
- Search in content scripts
- Enable JS source maps
- Enable CSS source maps
Useful Settings

- Red shaded rectangles around repainted regions
- Orange borders around composited layers
- Yellow border around touch handler listeners
Layer promotion hacks

Blink/Webkit

-webkit-transform: translateZ(0);
-webkit-transform: translate3d(0,0,0);

iOS

-webkit-perspective: 1000;
-webkit-backface-visibility: hidden;

Use with caution!!
Since the first LS was launched in 1989, Lexus has never slowed in its relentless pursuit of perfection. Each LS generation set new industry standards and showcased class-leading excellence.
New! Layers panel

Visualize elements promoted to a layer with the new layers panel (experimental)
then used to rotate each ring, and to spin the containing element around too.

Note that you can still select the numbers on the ring; everything remains clickable.
Layout thrashing
When JS violently writes, then reads from the DOM repeatedly causing reflow
Performance playground

Resize this:

Then:

Set block width  Go slow

- Gradient
- Border Radius
- Box Shadow
- Blur Filter
- Background
- Alt-BG

blandit purus dui ipsum sagittis augue rutrum enim nunc, aliquet
platea ornare erat at condimentum egestas quis commodo, sit justo,
habitasse lorem lectus vestibulum interdum magna mauris tristique donec feugiat
congue arcu pellentesque nisl, malesuada ultrices massa feugiat nullam turpis

proin cursus proin tellus nisi, dui egestas gravida elementum eget
quisque turpis hac sed arcu quam sagittis donec amet bibendum
sagittis tristique eu volutpat, turpis fringilla elementum nullam cursus nisl
vel auctor faucibus odio volutpat aenean purus feugiat sed amet

elementum tempus, ut nunc, porta quisque scelerisque commodo augue cursus
neque potenti enim faucibus velit proin sed volutpat at ut
nulla vitae pretium, iaculis feugiat in cras dui non phasellus
odio et non, elementum tincidunt bibendum mauris tempor et, sapien,

mauris cursus tincidunt
est eget dictum sed
lacus, id dolor nisl nunc
egest nibh dignissim
The slow way

while (i--) {
    var greenBlockWidth = sizer.offsetWidth;
    ps[i].style.width = greenBlockWidth + 'px';
}
var greenBlockWidth = sizer.offsetWidth;
while (i--) {
    ps[i].style.width = greenBlockWidth + 'px';
}
Writes to the DOM

invalidate layout

Browser wants to wait until the end of the current frame to reflow.
Forced synchronous layout

Querying geometric values before a frame completes, forces early layout.
Duration 471.000ms (at 12.87s)

Note Forced synchronous layout is a possible performance bottleneck.
It’s a performance killer.
FastDOM

Batches DOM reads/writes on the next frame using rAF for scheduling.
Prevent layout thrashing with requestAnimationFrame

In this example we are carrying out the task of setting heights of four divs on the page based on their width. We've solved the problem in two ways:

The first is less performant because it causes the DOM to reflow 4 times by reading and writing in quick succession.

The second makes use of requestAnimationFrame to schedule all DOM writes for the next frame. This means that all the reads happen together, followed by all the writes, and our layout doesn't get 'thrashed' ;)

Run timeline in DevTools and observe the number of layout events that occur for each solution.

With layout thrash | Without layout thrash

24.61400000235755ms
Prevent layout thrashing with requestAnimationFrame

In this example we are carrying out the task of setting heights of four divs on the page based on their width. We've solved the problem in two ways:

The first is less performant because it causes the DOM to reflow 4 times by reading and writing in quick succession.

The second makes use of requestAnimationFrame to schedule all DOM writes for the next frame. This means that all the reads happen together, followed by all the writes, and our layout doesn't get 'thashed' :)

Run timeline in DevTools and observe the number of layouts event that occur for each solution.

With layout thrash  Without layout thrash

6.305999995674938ms
Timeline Action Cheat Sheet

script-bound?
- JS Profiler. Flame chart it.

layout-bound?
- Dig into JS triggering it.

paint-bound?
- Find your paints: turn on *Paint Rectangles & Layer Borders*
- Promote layers out of paint storms *(with `translateZ()`)*
- Bisect using continuous paint mode

None of this worked?
- about:tracing

~ Paul Irish
Case studies
Don’t guess it, test it!
Case study: Flickr

with thanks to Scott Schiller @ Flickr
Web seeing a trend of vertical parallax effects.
LEVEL THREE
EAMES' DREAM WITHIN ARTHUR'S
DREAM WITHIN YUSUF'S DREAM

INCEPTION

COBB AND ARIADNE USE THE SEDATIVE TO FOLLOW
FISCHER INTO LIMBO. 3 LEVELS IS AS DEEP AS YOU
CAN GO WITHOUT FALLING INTO LIMBO.

EAMES FORGER

COBB EXTRACTOR

ARIADE ARCHITECT

FISCHER (IN LIMBO)
flickr.com (optimized)
Smile.
Everyone gets a free terabyte.

Sign up for Flickr

Biggr. That's right, a terabyte.
Spectacular. Share in full resolution.
Wherevr. Available anywhere you go.
But scroll used to be significantly slower.
For each \( \gamma \) pixels of vertical axis scrolling, move an absolutely positioned image in the same direction.
For each $\gamma$ pixels of vertical axis scrolling, move an absolutely positioned image in the same direction.
window.onscroll = function(e) {

    var parallax = document.getElementById('parallax-background');
    parallax.style.marginTop = (window.scrollY/2) + 'px';

}
Neither `marginTop` or `backgroundPosition` alone perform well. They don’t use hardware compositing.
Demo (slow)
Smile.
Everyone gets a free terabyte.

Sign up for Flickr

Biggr. That's right, a terabyte.
Spectaculr. Share in full resolution.
Wherevr. Available anywhere you go.

Sign In
Timeline view
Trick

Have the GPU help with accelerating compositing of the expensive parts
Avoid expensive paints
Reminder

Elements can be promoted to a layer using `translateZ()` or `translate3D()`
window.onscroll = function(e) {

  var parallax = document.getElementById('parallax-background');
  parallax.style.transform = 'translate3d(0px,' + (window.scrollY/2) + 'px, 0px)';

}
Promoting the element to a new layer
Demo (fast)
Spectaculr
Share in full resolution.
All of your photos in their original quality, beautifully displayed.
Timeline view

Wherevr

Upload once, send to any device, any screen, any friend, and any follower.

Download on the App Store

GET IT ON Google play
Twitter Bootstrap 3
Bootstrap

Sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development.

Download Bootstrap
<table>
<thead>
<tr>
<th>Button</th>
<th>class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>btn</td>
<td>Standard gray button with gradient</td>
</tr>
<tr>
<td>Primary</td>
<td>btn btn-primary</td>
<td>Provides extra visual weight and identifies the primary action in a set of buttons</td>
</tr>
<tr>
<td>Info</td>
<td>btn btn-info</td>
<td>Used as an alternative to the default styles</td>
</tr>
<tr>
<td>Success</td>
<td>btn btn-success</td>
<td>Indicates a successful or positive action</td>
</tr>
<tr>
<td>Warning</td>
<td>btn btn-warning</td>
<td>Indicates caution should be taken with this action</td>
</tr>
<tr>
<td>Danger</td>
<td>btn btn-danger</td>
<td>Indicates a dangerous or potentially negative action</td>
</tr>
<tr>
<td>Inverse</td>
<td>btn btn-inverse</td>
<td>Alternate dark gray button, not tied to a semantic action or use</td>
</tr>
<tr>
<td>Link</td>
<td>btn btn-link</td>
<td>Deemphasize a button by making it look like a link while maintaining button behavior</td>
</tr>
<tr>
<td>Button</td>
<td>class</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td><strong>Default</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Primary</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
100% faster paint time.
BS3 much more viable for sites that need to work well on under-powered mobile and tablet devices
BS3 is much more viable for sites that need to work well on under-powered mobile and tablet devices.
Why is scrolling in this Daft Punk site so slow?
High image resize cost

Image shipped (high-res)

Size actually needed
Pre-scale images where possible to avoid resize cost.
Live case study: Pinterest
Post-optimization
Great scrolling performance
How can there possibly be anything better! There’s chocolate covered strawberries on it!❤️

Candy Crush Cake-gonna make this for my mom since she’s so addicted to the game

Alice Gilhooly cake design

Too much???? @Jess Liu-Leigh Fletcher haha

millie snell cake :P
Pre-optimization
Main issues

Sub-optimal position: fixed use causing unnecessarily large repaints
Slow scrolling due to excessive use of styles
(currently) expensive to paint
Doing a lot during hover on scroll
Case study: Google+

with thanks to Steve Kobes @ Google+
Perf went from 12fps to 60fps
http://bgrins.github.io/devtools-snippets/#console-save

Chrome Dev Summit Live Stream

📅 Wed, 20 November, 17:30
👤 Jimmy Moon + 756 others going

Are you going?
- Yes
- Maybe
- No

+50 🔄 11

👥 OPANGA ELIUĐ 7 Nov 2013
All accepted

Add a comment...

7 comments
 düd prados 8 Nov 2013
Awesome!!!!!!
Jank bustin’
Simplified G+ layout
Simplified G+ layout
Problem.
A `position:fixed` element causes a repaint when moved within its layer.
To avoid the repaint, put the element in its own layer.

```html
<div style="-webkit-transform: translateZ(0)"> Fixed element </div>
```
Simplified G+ layout

translateZ(0)

translateZ(0)
Caveats of `translateZ`

- more layers = more time compositing layers
- text anti-aliasing requires an opaque background within the layer
- triggers Safari positioning bugs inside iframes

```html
<div style="-webkit-transform: translateZ(0)"> Fixed element </div>
```
Too many layers?

Sometimes a style creates compositing layers for all positioned descendants.
Another problem.
G+ styles for card-flip animation

Paul Irish

Shared publicly - 8 Oct 2013

CSS PRETTYPRINT. YES YES Y'ALL.

François Beaufort originally shared:
Chrome DevTools now support CSS Pretty-printing in the last chromium build.
Read more
CSS PRETTYPRINT. YES YES Y'ALL.

François Beaufort originally shared:
Chrome DevTools now support CSS Pretty-printing in the last chromium build.
Read more
Causes every descendent element to get its own layer
The Fix!

Remove it, apply it through JS for duration of the card animation
one last problem
Forced synchronous layouts

Modifying the DOM invalidates layout.

element.innerHTML = '...';  // Layout invalidated.
Measuring the DOM depends on the layout

If previously invalidated, this forces **synchronous** layout
(since execution cannot continue until the correct value is obtained).

```javascript
alert(element.offsetHeight); // Layout forced.
```
They reduced them from O(n) to O(1) by refactoring a loop.
Forced Synchronous Layouts

Timeline shows where your code is causing synchronous layouts

Remember to **scroll down** for the second stack trace.
Bonus optimization

Animate with translate instead of left/top

```
.foo {
  animation: move 3s ease infinite;
}
```

```
@keyframes move {
  50% { top: 100px; }
}
```

*Heavy repaints.*

```
@keyframes move {
  50% { transform: translateY(100px); }
}
```

*GPU-accelerated, no repaints.*
Paint performance tooling in other browsers
There’s now lots of tooling to improve the responsiveness of your UIs.
IE F12 Developer Tools
UI Responsiveness Tool shows frame rate
Firefox DevTools: Paint flashing
Highlight areas being painted

The Planetarium

JUPITER is the fifth planet from the Sun and is the largest planet of the Solar System. Jupiter has 63 known satellites, the biggest of which are called the four Galilean Moons: Io, Europa, Ganymede and Callisto.
WebKit Nightlies: Paint counts of layers
WebKit Nightlies: Reasons for compositing

Element has a 3D transform
Element has "backface-visibility: hidden" style
Element overlaps other compositing element
Audience check
happy()

undefined
In summary...
Frame rate matters and can impact engagement
Don’t over-do layout.
Do reads before writes.
Be mindful of paint costs. Use the DevTools.
Check your perf on desktop and mobile.
Mobile-first performance benchmarking

Test on low-end hardware. Get 60fps there.
DevTools Remote Debugging
If you’re building something, set performance objectives up front.
Hit those objectives, add tests to measure and then never regress!
If all goes well, this could be you!
To learn more checkout jankfree.org
Use tools. not rules.
Thank you.

@addyosmani
+AddyOsmani