USTREAM

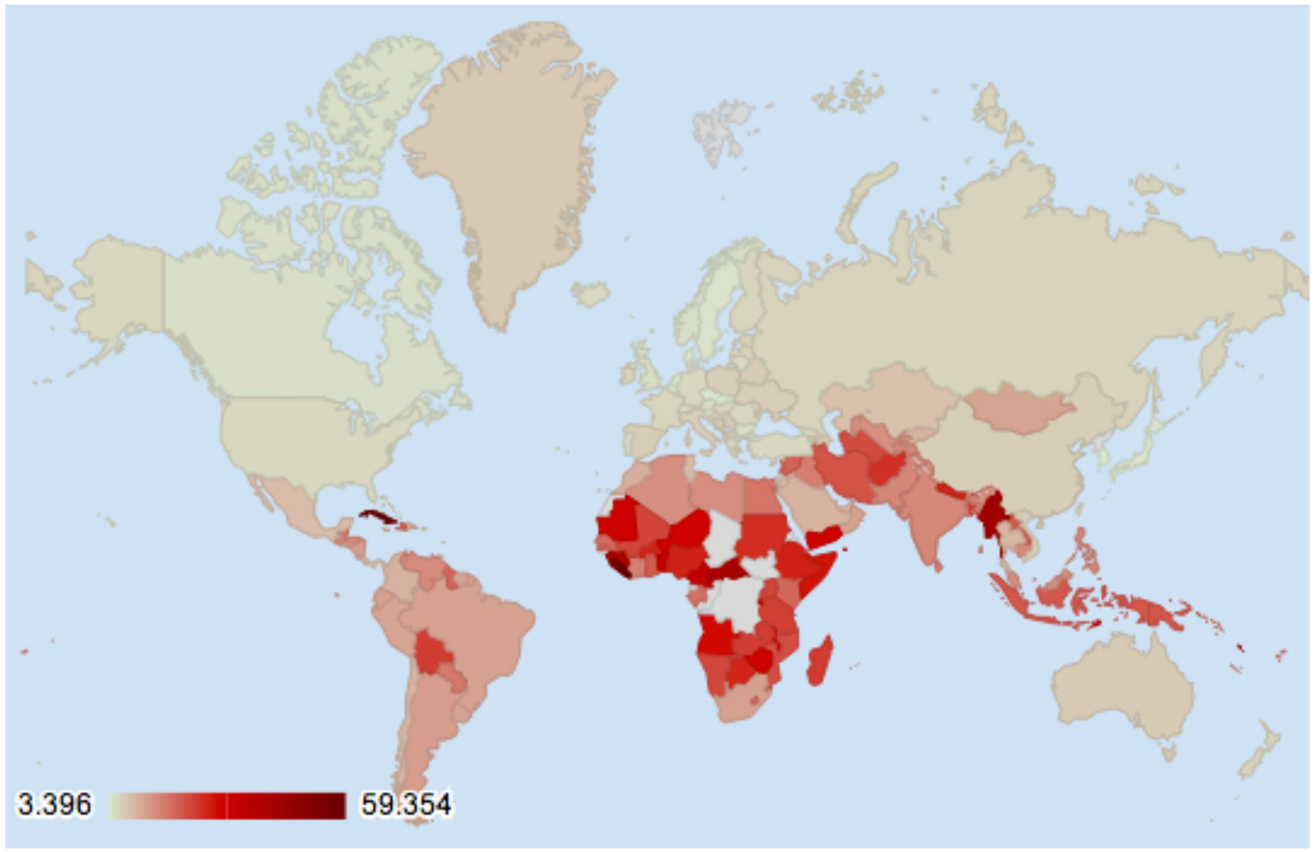
Optimizing Browser experience focusing on Mobile

MATE NADASDI

@matenadasdi

Perf matters

- Every second = 0.65% increase in bounce rate
- Facebook 60FPS - 30FPS timeline experiment (lower engagement)
- 86% of the user waiting time spent on client side



Page load speed on the globe

Some facts about Mobile vs Desktop perf

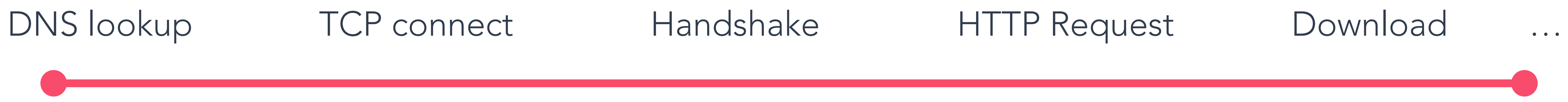
- 3G/4G vs Cable/Fiber
- **Latency is higher** (18ms - 26ms - 43ms - 150ms - 400ms)
- Radio Resource Controller is in the game
- Touch events - Software & Hardware input latency
- **Users expects the same speed as desktop**

Memorize 2 numbers

1000ms - Show usable content to the user

16.6ms - Deliver a frame to go for 60FPS

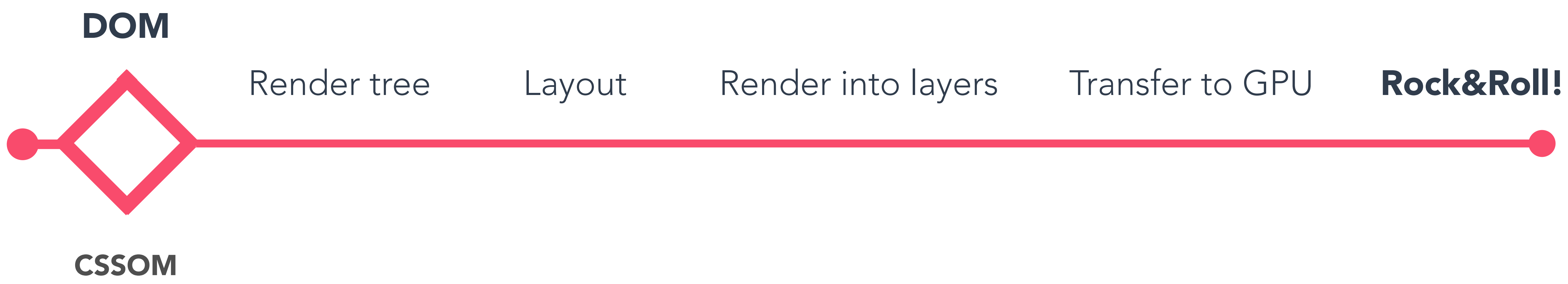
Networking



Networking

- **69,5% of time block on networking** (Top 1 Million Alexa sites)
- DNS lookups and TCP connects are expensive
- DNS prefetch, Prefetch, Prerender
- Compress, Sprite images, **count on TCP Slow Start**
- **Mobile radio** is one of the most battery killer resources.

Response parsing



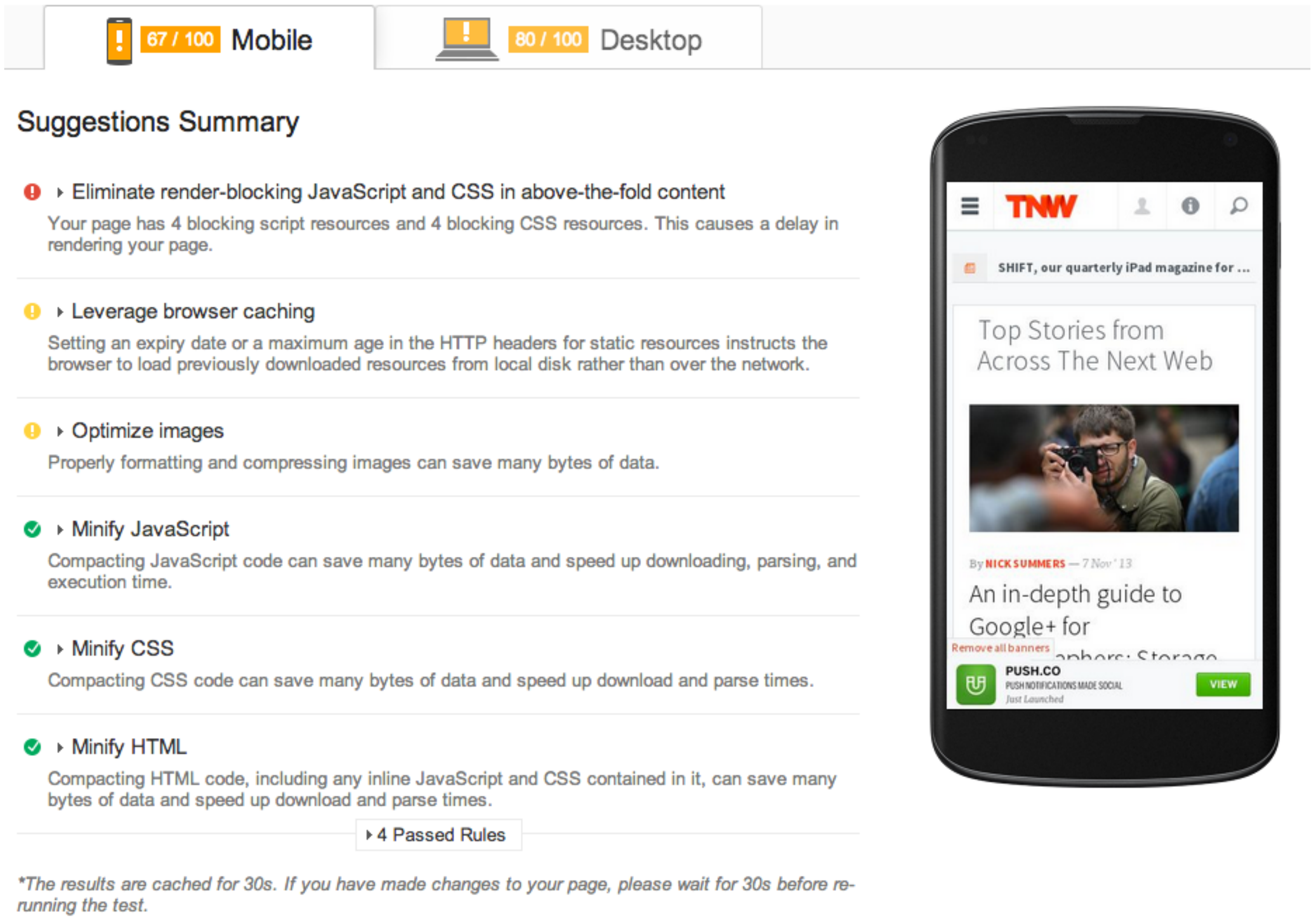
Initial rendering tips

- Inline critical JS/CSS, lazy load others
- **Do not load** resources required for **below the fold** experience
- Use **deferred, async** JS to save page load time
- Remember! **CSS is not incremental.**

Measure, Analyze, Optimize!

Pagespeed insights

(https://developers.google.com/speed/pagespeed/insights/)

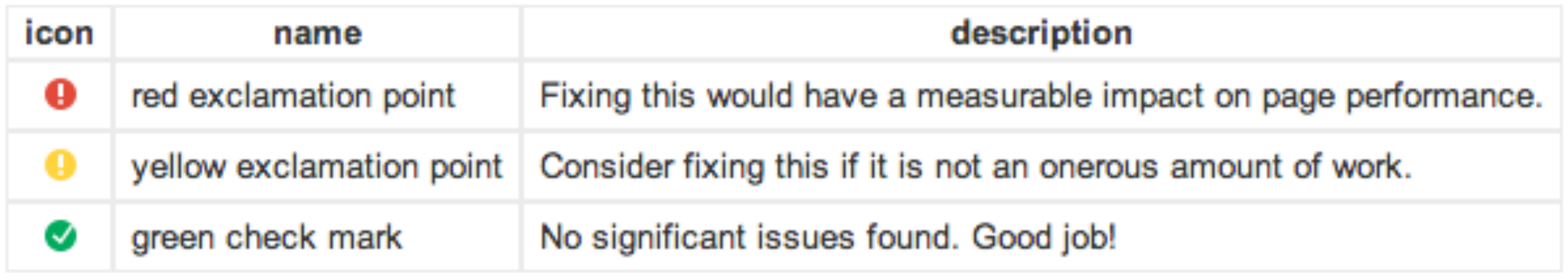

67 / 100 Mobile
80 / 100 Desktop
TNW
SHIFT, our quarterly iPad magazine for ...
Top Stories from Across The Next Web
By NICK SUMMERS — 7 Nov '13
An in-depth guide to Google+ for
Remove all banners
PUSH.CO
PUSH NOTIFICATIONS MADE SOCIAL
Just Launched
VIEW

Suggestions Summary

- Eliminate render-blocking JavaScript and CSS in above-the-fold content
  Your page has 4 blocking script resources and 4 blocking CSS resources. This causes a delay in rendering your page.
- Leverage browser caching
  Setting an expiry date or a maximum age in the HTTP headers for static resources instructs the browser to load previously downloaded resources from local disk rather than over the network.
- Optimize images
  Properly formatting and compressing images can save many bytes of data.
- Minify JavaScript
  Compacting JavaScript code can save many bytes of data and speed up downloading, parsing, and execution time.
- Minify CSS
  Compacting CSS code can save many bytes of data and speed up download and parse times.
- Minify HTML
  Compacting HTML code, including any inline JavaScript and CSS contained in it, can save many bytes of data and speed up download and parse times.

4 Passed Rules

The results are cached for 30s. If you have made changes to your page, please wait for 30s before re-running the test.

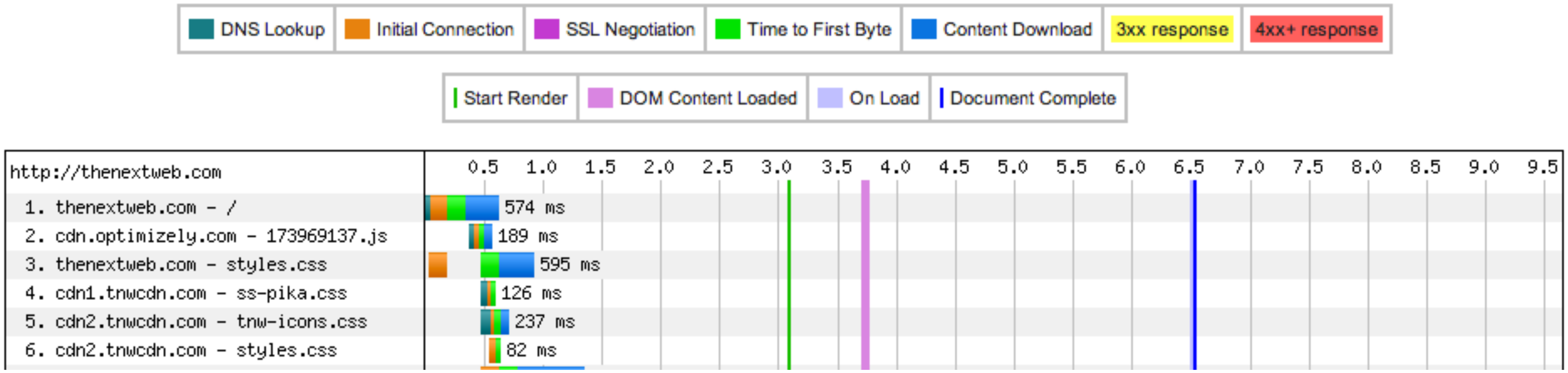


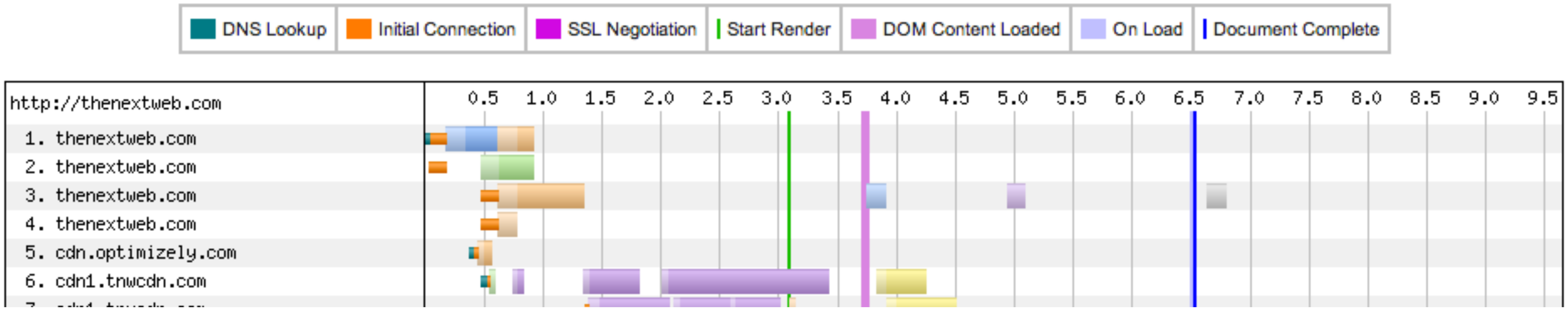
icon	name	description
❗	red exclamation point	Fixing this would have a measurable impact on page performance.
!	yellow exclamation point	Consider fixing this if it is not an onerous amount of work.
✓	green check mark	No significant issues found. Good job!

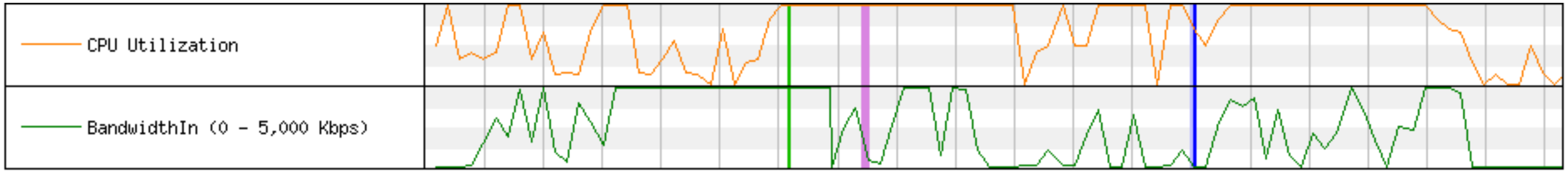
Measure, Analyze, Optimize!

webpagetest.org

API/Docs: https://sites.google.com/a/webpagetest.org/docs/

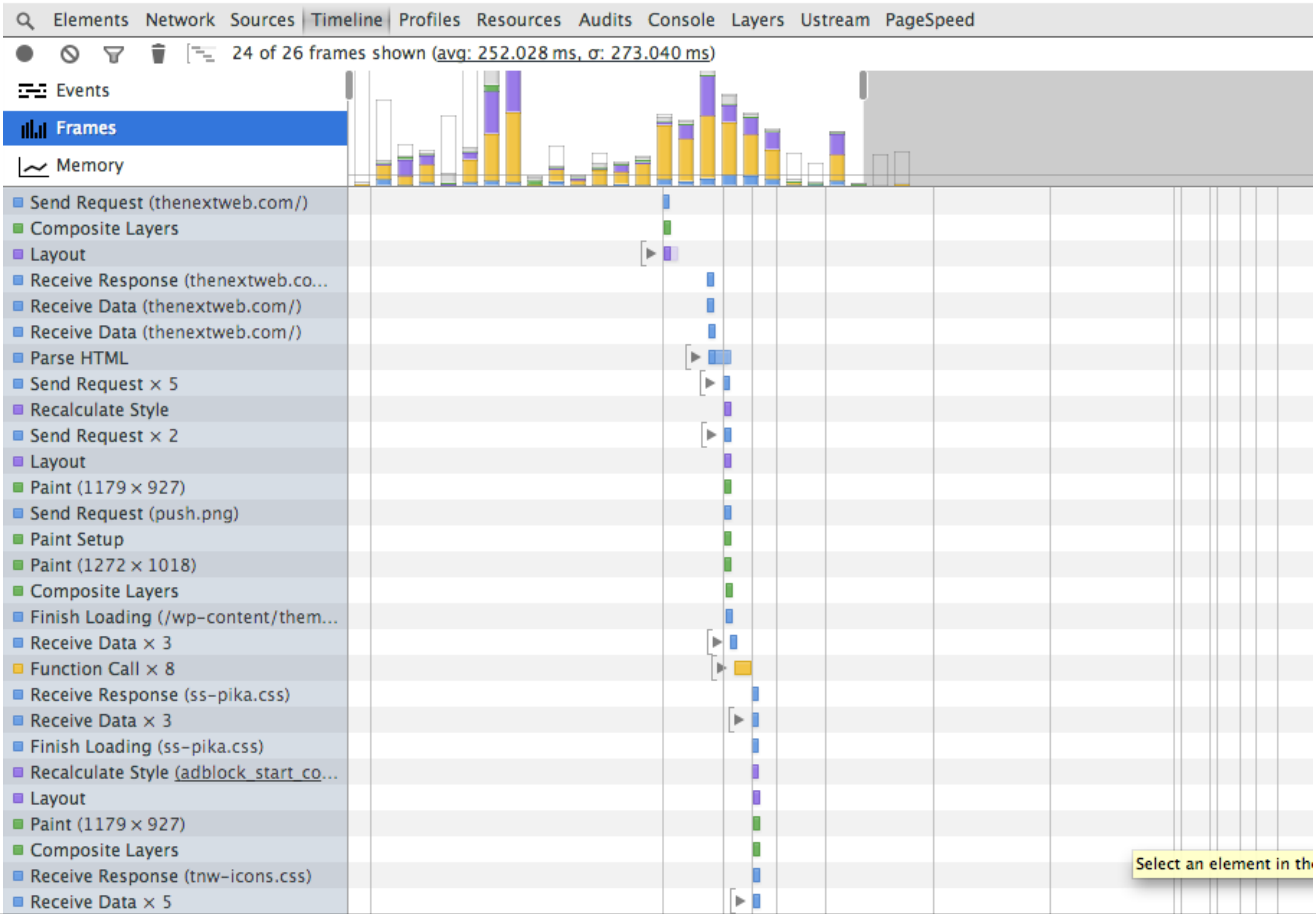






Measure, Analyze, Optimize!

DevTools timeline panel - https://developers.google.com/chrome-developer-tools/docs/timeline



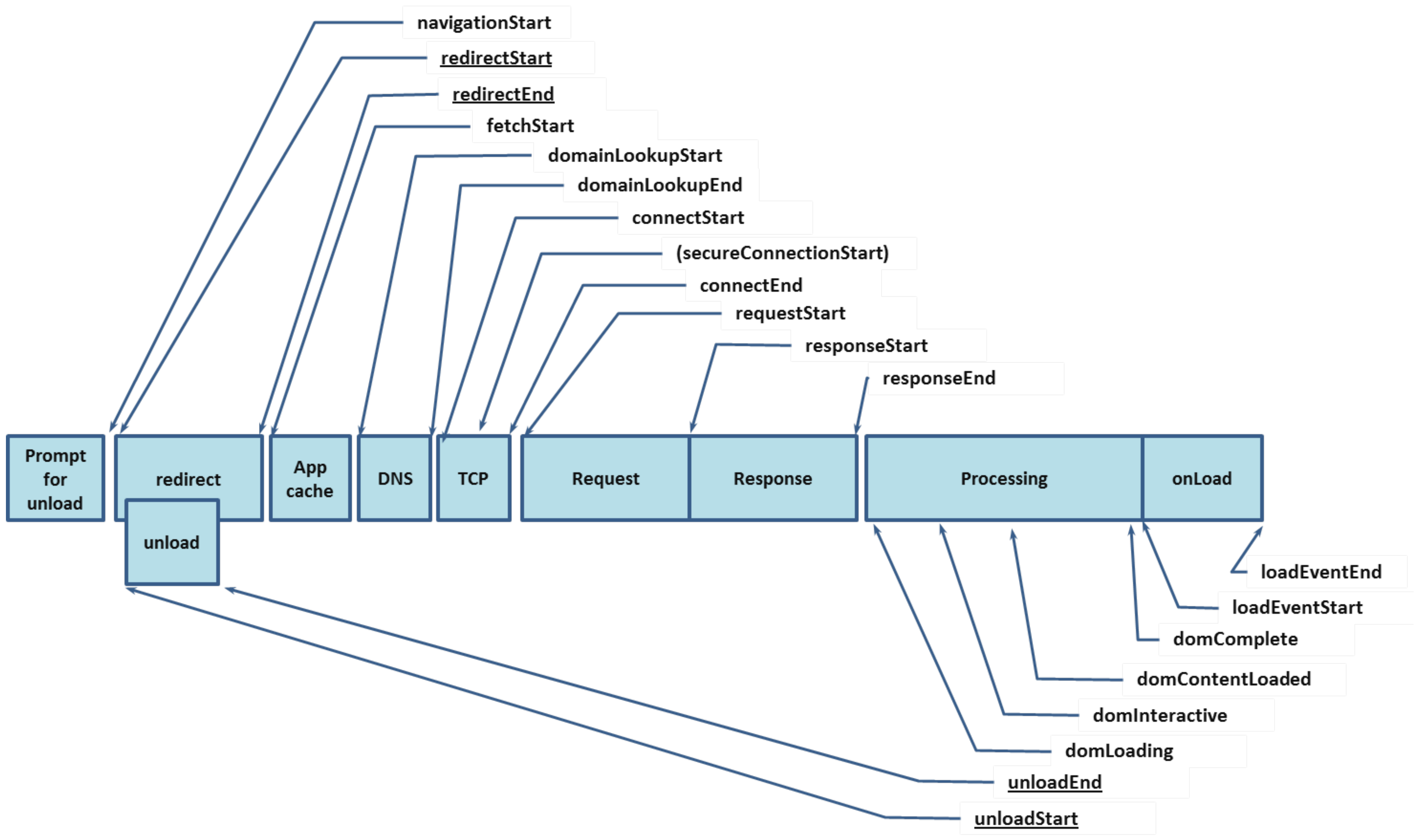
24 of 26 frames shown (avg: 252.028 ms, σ: 273.040 ms)

```
Send Request (thenextweb.com/)
Composite Layers
Layout
Receive Response (thenextweb.co...
Receive Data (thenextweb.com/)
Receive Data (thenextweb.com/)
Parse HTML
Send Request × 5
Recalculate Style
Send Request × 2
Layout
Paint (1179 × 927)
Send Request (push.png)
Paint Setup
Paint (1272 × 1018)
Composite Layers
Finish Loading (/wp-content/them...
Receive Data × 3
Function Call × 8
Receive Response (ss-pika.css)
Receive Data × 3
Finish Loading (ss-pika.css)
Recalculate Style (adblock_start_co...
Layout
Paint (1179 × 927)
Composite Layers
Receive Response (tnw-icons.css)
Receive Data × 5
```

Measure, Analyze, Optimize!

Navigation Timing API

https://dvcs.w3.org/hg/webperf/raw-file/tip/specs/NavigationTiming/Overview.html



In-App rendering

- We need **60fps** for **jank** free rendering
- **16.6ms** is not so much time for layout/paint/JS/GC
- Touch handlers can block the GPU Compositing on mobile
- Scroll handler functions have to finish in this range too

Rendering Tips

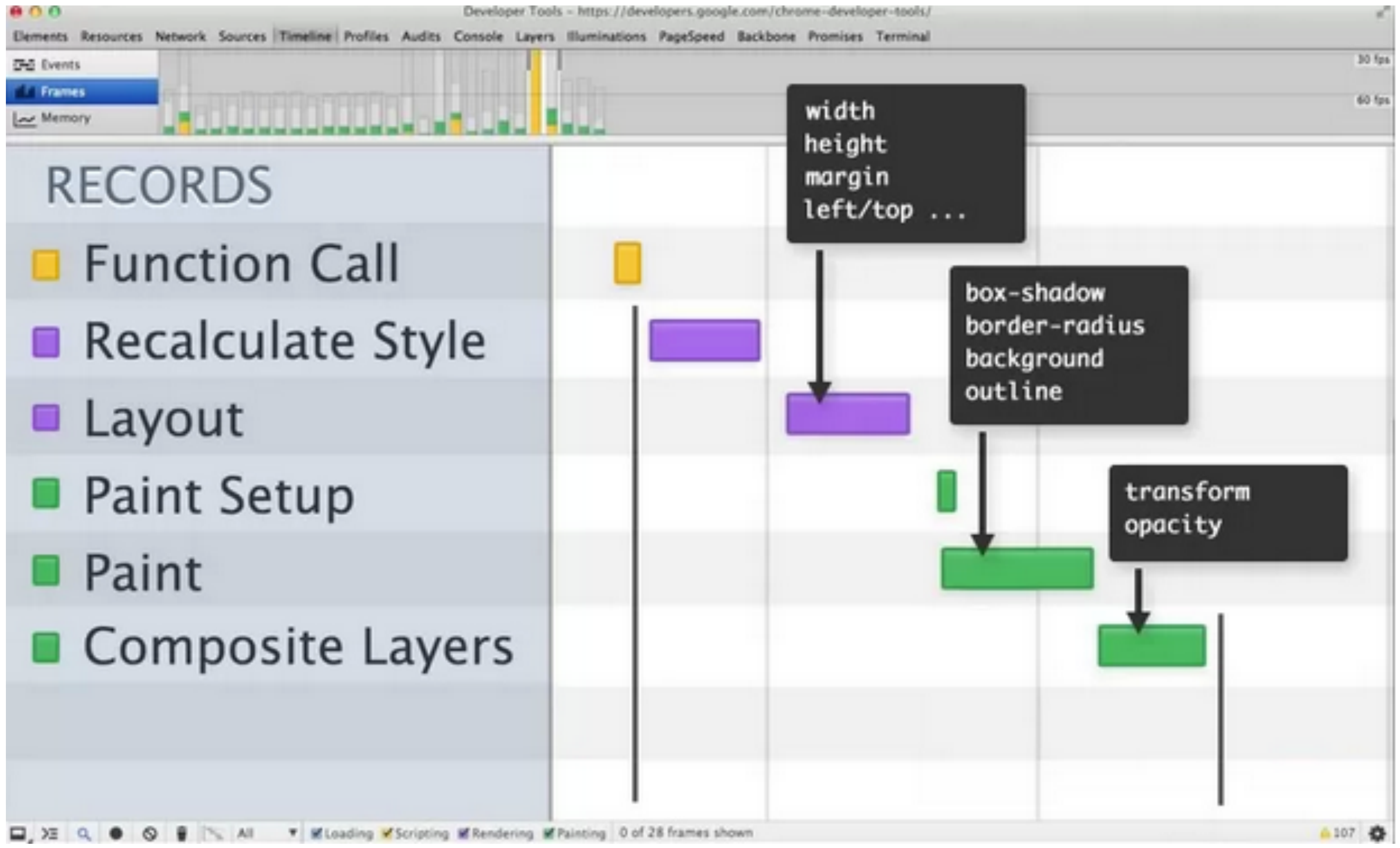
- Try to modify **smaller subtrees** in the Render Tree
- Animate props which affect compositing only (transform/opacity)
- Avoid setTimeout, use **requestAnimationFrame**
- Bind handlers close to the target
- Image resizing in the browser is **evil**!
- Dedicate layers for the most expensive parts.

Rendering perf tools

DevTools timeline frames panel

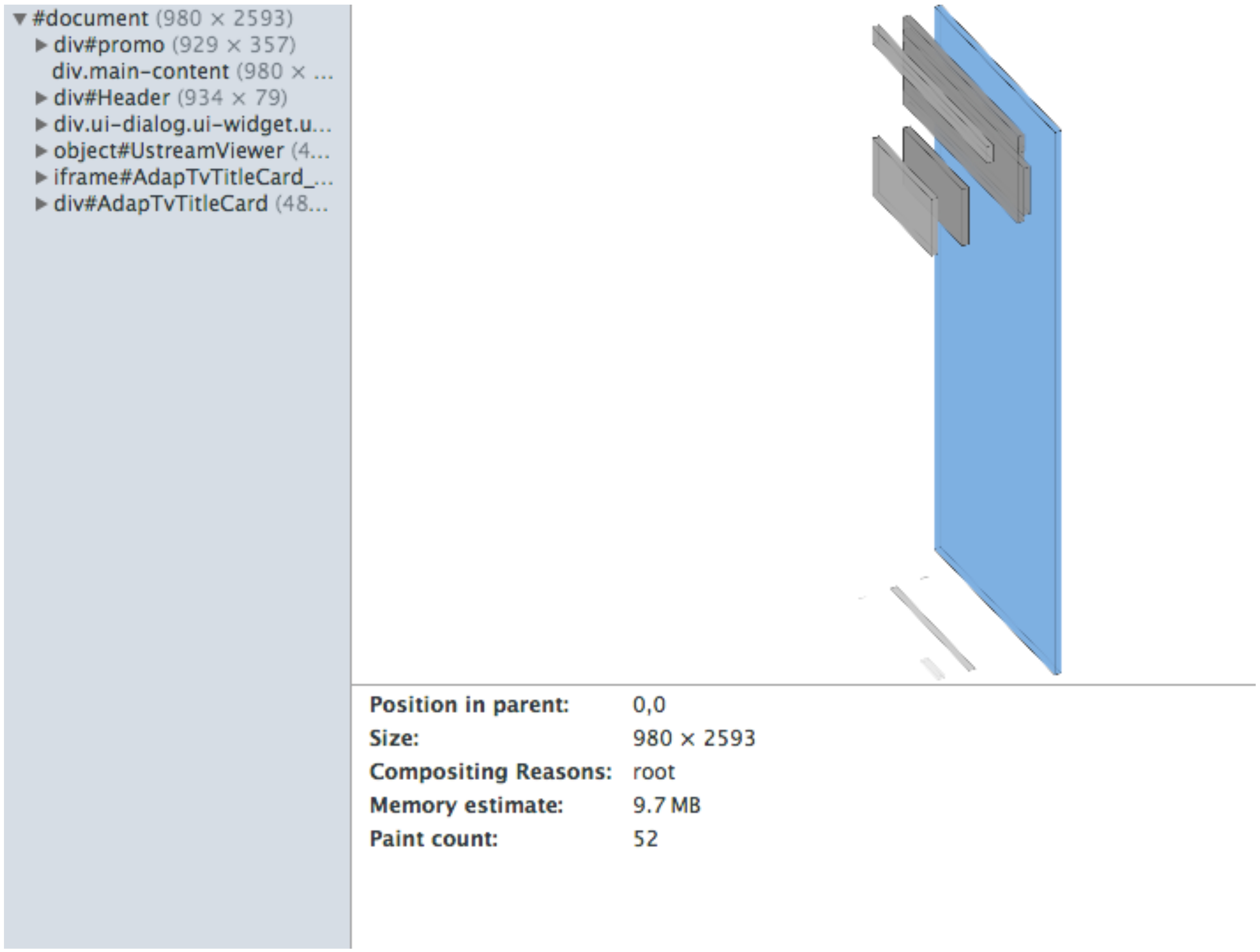
http://www.html5rocks.com/en/tutorials/speed/high-performance-animations/

https://developers.google.com/chrome-developer-tools/docs/timeline



Rendering perf tools

DevTools layers (experimental - Canary)



```
▼ #document (980 × 2593)
  ▶ div#promo (929 × 357)
    div.main-content (980 × ...
  ▶ div#Header (934 × 79)
  ▶ div.ui-dialog.ui-widget.u...
  ▶ object#UstreamViewer (4...
  ▶ iframe#AdapTvTitleCard_...
  ▶ div#AdapTvTitleCard (48...
```

Position in parent:	0,0
Size:	980 × 2593
Compositing Reasons:	root
Memory estimate:	9.7 MB
Paint count:	52

Chrome DevTools frames panel & layers panel

DEMO!

http://slideshare.net/matenadasdi1/optimizing-browser-experience-hwsw-app-conf

USTREAM

Thank you!

Q&A?

Props to **Ilya Grigorik** for the review

@matenadasdi