Reliably Measuring Responsiveness in the Wild

Shubhie Panicker
@shubhie

Nic Jansma
@nicj
When is load?
## Recent Posts

1. Measuring Real User Performance in the Browser
2. AMP: Does it Really Make Your Site Faster?
3. Measuring Continuity
4. Particle Photon/Fluxon Remote Temperature and Humidity Logger
5. Compressing User Timing
6. Domain-Local Tools for In-Depth Performance Investigations
7. Measuring the Performance of Single Page Applications
8. User Timing in Practice
9. Resource Timing in Practice
10. Navigation Timing in Practice

### Table

<table>
<thead>
<tr>
<th>Date</th>
<th>Before</th>
<th>After</th>
<th>Save</th>
</tr>
</thead>
<tbody>
<tr>
<td>13-Jan</td>
<td>158</td>
<td>153</td>
<td>3</td>
</tr>
<tr>
<td>14-Jan</td>
<td>271</td>
<td>251</td>
<td>20</td>
</tr>
<tr>
<td>15-Jan</td>
<td>98</td>
<td>80</td>
<td>18</td>
</tr>
<tr>
<td>20-Jan</td>
<td>169</td>
<td>139</td>
<td>30</td>
</tr>
<tr>
<td>21-Jan</td>
<td>70</td>
<td>56</td>
<td>14</td>
</tr>
<tr>
<td>17-Jan</td>
<td>124</td>
<td>99</td>
<td>25</td>
</tr>
<tr>
<td>22-Jan</td>
<td>209</td>
<td>189</td>
<td>20</td>
</tr>
<tr>
<td>Total</td>
<td>616</td>
<td>458</td>
<td>158</td>
</tr>
</tbody>
</table>

### Key
- `$T$` = Applying Full UserTimingCompress Compiler Optimizer
- `$50$-ps` = minifying timings compression length
- `$58$-ps` = 58-psi length / 100 bytes

Even with pre-applying the 10-byte compression and garbling the result, gap doesn't beat the full UserTimingCompress techniques. Here, in general, gap is 15% larger than UserTimingCompress. There is a fine trade where get is better, mostly in real cases with a lot of repeating strings.

Additionally, applying gaps requires your app include a wall-clock gap library, side --- whereas before code is currently around 20.3 KB modified. UserTiming compression is much smaller, at only 3.9 KB modified.

Finally, if you're using gap compression, you can't just start the gap data into a QueryString, as is, essentially HTML instead of lose tremendously.
Old load metrics don’t capture user experience.

We need to rethink our metrics and **focus on what matters**.
Performance only matters at Load time
My app loads in
X.X seconds
Load metrics are NOT a single number
Performance in the **Lab**

Performance in the **Real World**
Key questions

- What metrics accurately capture responsiveness?
- How to measure these on real users?
- How to analyze data and find areas to improve?
Queueing Time
Millions of Long Tasks

Long Tasks on 3 customer sites (daily average)

- Site 1 (Travel): 276,000
- Site 2 (Gaming): 200,000
- Site 3 (Retail): 593,000
What Are LongTasks?

- Script (self, *-origin-*) - 83.0%
- Script (multiple-contexts) - 10.0%
- Non-Script (unknown) - 7.0%
60 fps: An Elusive Dream
Real User Measurement (RUM)
Real world measurement with Web Performance APIs
New Performance APIs and Metrics

- Performance Observer
- LongTasks
- Time to Interactive
- Input Latency
Performance Building Blocks

- Performance Observer
- High Resolution Time
- Performance Entry
PerformanceTimeline vs PerformanceObserver

// PerformanceTimeline
var entries = performance.getEntriesByType("resource");

// PerformanceObserver
var entries = [];
const observer = new PerformanceObserver((list) => {
    for (const entry of list.getEntries()) {
        entries.push(entry);
    }
});
observer.observe({entryTypes: ['resource']});
LongTasks

https://github.com/w3c/longtasks
Bad Workarounds

- Timeout polling
- rAF loop

Issues

- Performance overhead
- Battery drain
- Precludes rIC
- No attribution
Long Tasks via PerformanceObserver

```javascript
const observer = new PerformanceObserver((list) => {
  for (const entry of list.getEntries()) {
    sendDataToAnalytics('Long Task', {
      time: entry.startTime + entry.duration,
      attribution: JSON.stringify(entry.attribution),
    });
  }
});

observer.observe({entryTypes: ['longtask']});
```
## Render & Jank

<table>
<thead>
<tr>
<th>Jank</th>
<th>Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>456</td>
</tr>
<tr>
<td>789</td>
<td>012</td>
</tr>
<tr>
<td>345</td>
<td>678</td>
</tr>
<tr>
<td>901</td>
<td>234</td>
</tr>
<tr>
<td>567</td>
<td>890</td>
</tr>
<tr>
<td>012</td>
<td>345</td>
</tr>
<tr>
<td>678</td>
<td>901</td>
</tr>
<tr>
<td>234</td>
<td>567</td>
</tr>
<tr>
<td>456</td>
<td>789</td>
</tr>
</tbody>
</table>

[https://w3c.github.io/longtasks/render-jank-demo.html](https://w3c.github.io/longtasks/render-jank-demo.html)
Multiple sub-tasks (scripts) within a long task
Attribution: Who?

“Minimal Frame Attribution” with **name**

- self, same-origin-ancestor, same-origin-descendant, cross-origin-ancestor, cross-origin-descendant, multiple-contexts, unknown etc.
Attribution: Who And Why?

Detailed attribution with TaskAttributionTiming

- attribution[]
  - containerType: iframe, embed, object
  - containerSrc: <iframe src="http://..." />
  - containerId: <iframe id="ad" />
  - containerName: <iframe name="ad-unit-1" />
More Attribution: Coming soon!

Detailed attribution with TaskAttributionTiming

- attribution[]
  - containerType: iframe, embed, object
  - containerSrc: `<iframe src="http://..."/>
  - containerId: `<iframe id="ad"/>
  - containerName: `<iframe name="ad-unit-1"/>
LongTasks: Usage Tips

● Measuring during page load: Turn it on as early as possible (e.g. `<head>`)  
● Measuring during interactions with a circular buffer  
● First-party (“my frame”) LongTasks give only duration  
● Third-party other-frames provide attribution if the IFRAME itself is annotated via `id`, `name` or `src`. 
Time to Interactive

Is it Usable?
Time to Interactive

User Sees Content

Not Interactive Until Here
Time to Interactive: Lower Bound

*When does the page appear to the visitor to be interactable?*

Start from the latest *Visually Ready* timestamp:

- DOMContentLoaded (document loaded + parsed, without CSS, IMG, IFRAME)
- First Paint, First Contentful Paint
- Hero Images (if defined by the site, important images)
- Framework Ready (if defined by the site, when core frameworks have all loaded)
First Paint
Framework Ready
Hero Images
DOM Content Loaded
Visually Ready
Time to Interactive: Measuring

What's the first time a user could interact with the page and have a good experience?

Starting from the lower bound (Visually Ready) measure to Ready for Interaction where none of the following occur for your defined period (e.g. 500ms):

● No Long Tasks
● No long frames (FPS >= 20)
● Page Busy is less than 10% (setTimeout polling)
● Low network activity (<= 2 outstanding)

github.com/GoogleChrome/tti-polyfill
github.com/SOASTA/boomerang/tree/continuity
Input Latency

Measuring bad user experiences

- Interactions (scrolls, clicks, keys) may be delayed by script, layout and other browser work
- Latency can be measured \( \text{performance.now() - event.timeStamp} \)
- Latency can be attributed via LongTasks
Measure input latency: event.timeStamp and performance.now()

const subscribeBtn = document.querySelector('#subscribe');

subscribeBtn.addEventListener('click', (event) => {
  // Event listener logic goes here...

  const lag = performance.now() - event.timeStamp;
  if (lag > 100) {
    sendDataToAnalytics('Input latency', lag);
  }
});
Input Latency

Determining the cause via LongTasks:

1. Turn on PerformanceObserver
2. Watch for input delays
3. Find LongTasks that ended between event.timeStamp and performance.now()

Sample code:

github.com/nicjansma/reliably-measuring-responsiveness-in-the-wild/
Real World Data
Case Studies

3 sites over 1 month

- Site 1: Travel (ads, social)
- Site 2: Gaming (ads, social)
- Site 3: Retail (social, 3p, spa)
18+ million LongTasks
Duration Percentiles:

- 50th: 106 ms
- 75th: 208 ms
- 90th: 427 ms
- 95th: 666 ms
- 99th: 1,712 ms
- Range: 50 to 10+ seconds
LongTasks as % of Front End Load Time

Site 1

- Other: 52.0%
- LongTasks: 48.0%

Site 2

- Other: 79.0%
- LongTasks: 21.0%

Site 3

- Other: 75.0%
- LongTasks: 25.0%
LongTasks directly delay Time to Interactive.
Time to Interactive has high correlation with overall conversion rate.
First impressions matter: as first-page LongTask time increased, overall Conversion Rate decreased.
Mobile devices could see 12x LongTask time as Desktop.
LongTask as % of Front End Load Time

Sample Devices

Median LongTask (ms)

- Samsung Galaxy S7
  - LongTask: 62%
  - Other: 38%
- Samsung Galaxy S5
  - LongTask: 57%
  - Other: 43%
- Amazon Kindle Fire Tablet 7"
  - LongTask: 49%
  - Other: 51%

- Samsung Galaxy S7
- Samsung Galaxy S5
- Amazon Kindle Fire Tablet 7"
Older devices could be spending half of their load time on LongTasks.
Optimizing Performance
Every site is different.

Identify your core metrics.
Minimize the time to TTI

Consider Mobile traffic

Ship less JS

Break up existing JS with Code Splitting
Reduce Long Tasks

Mobile is especially hurting

Break up JS

Move intensive work off the main thread to workers
Hold Third Parties Accountable

Identify the worst offenders
Evaluate impact on TTI & business metrics
Looking Ahead

- Long Tasks V2
- Input Latency + Slow Frames
- Long Tasks is not Panacea
Thank You

Shubhie Panicker
@shubhie

Nic Jansma
@nicj