Reliably Measuring Responsiveness in the Wild

Shubhie Panicker

@shubhie

Nic Jansma

@nicj

When is **load**?

Search Facebook
Home
Edit Profile
FAVORITES
News Feed
Messages
Ads Manager
Events
PAGES
Pages Feed
Like Pages
Create Ad
Create Page
APPS
Live Video
Games
On This Day
Photos
FRIENDS
Close Friends
Family
GROUPS
Photo/Video
Photo Album
Blah blah blah
Friends
Post
Mike Posthumus
Like
Comment
Share
Write a comment...
See More
English (US) · Español · Português (Brasil) · Français (France) · Deutsch
Privacy · Terms · Advertising · Ad Choices · Cookies · More
Facebook © 2016
Search

Nic works at Akamai, develops high performance websites, apps and open-source tools, and is a Microsoft MVP for IE.

email [illegible] rss facebook google+ github

Founder of Wolverine Digital LLC

Recent Posts

- Measuring Real User Performance in the Browser
- AMP: Does it Really Make Your Site Faster?
- Measuring Continuity
- Particle Photon/Electron Remote Temperature and Humidity Logger
- Compressing UserTiming
- Forensic Tools for In-Depth Performance Investigations
- Measuring the Performance of Single Page Applications
- UserTiming in Practice
- ResourceTiming in Practice
- NavigationTiming in Practice

Copyright © 2004-2017 NicJ.net

```
| 13.json | 39  | 46          | 118%         |
+---------+-----+-------------+--------------+
| 14.json | 145 | 112         | 77%          |
+---------+-----+-------------+--------------+
| 15.json | 48  | 45          | 94%          |
+---------+-----+-------------+--------------+
| 16.json | 53  | 70          | 132%         |
+---------+-----+-------------+--------------+
| 17.json | 18  | 37          | 206%         |
+---------+-----+-------------+--------------+
| 18.json | 9   | [illegible] | [illegible]  |
+---------+-----+-------------+--------------+
| 19.json | 32  | 53          | 166%         |
+---------+-----+-------------+--------------+
| 20.json | 38  | 51          | 134%         |
+---------+-----+-------------+--------------+
| Total   | 811 | 968         | 119%         |
+---------+-----+-------------+--------------+

Key:
* UTC     = Applying full UserTimingCompression (bytes)
* TS.gz   = gzip(UTC timestamp compression).length
* TS.gz % = TS.gz bytes / UTC bytes
```

Even with pre-applying the timestamp compression and gzipping the result, gzip doesn't beat the full UserTimingCompression techniques. Here, in general, gzip is 19% larger than UserTimingCompression. There are a few cases where gzip is better, notably in test cases with a lot of repeating strings.

Additionally, applying gzip requires your app include a JavaScript gzip library, like pako — whose deflate code is currently around 26.3 KB minified. usertiming-compression.js is much smaller, at only 3.9 KB minified.

Finally, if you're using gzip compression, you can't just stick the gzip data into a Query String, as URL encoding will increase its size tremendously.
]

Old load metrics don’t capture user experience.

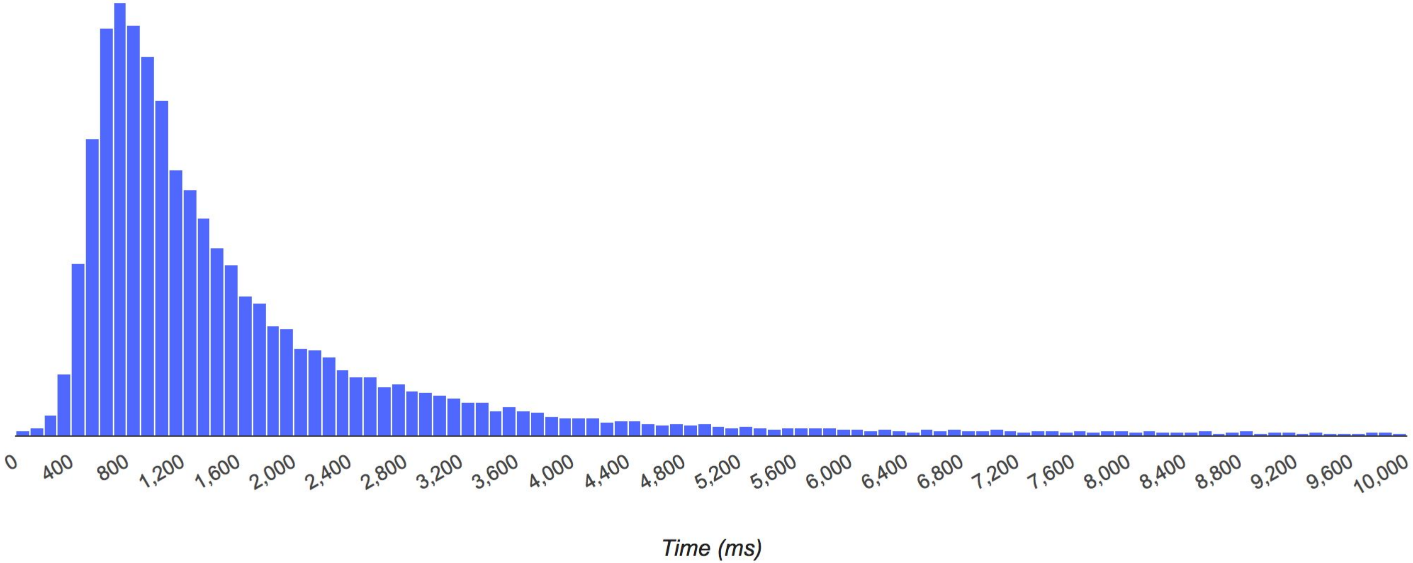
We need to rethink our metrics and **focus on what matters**.

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

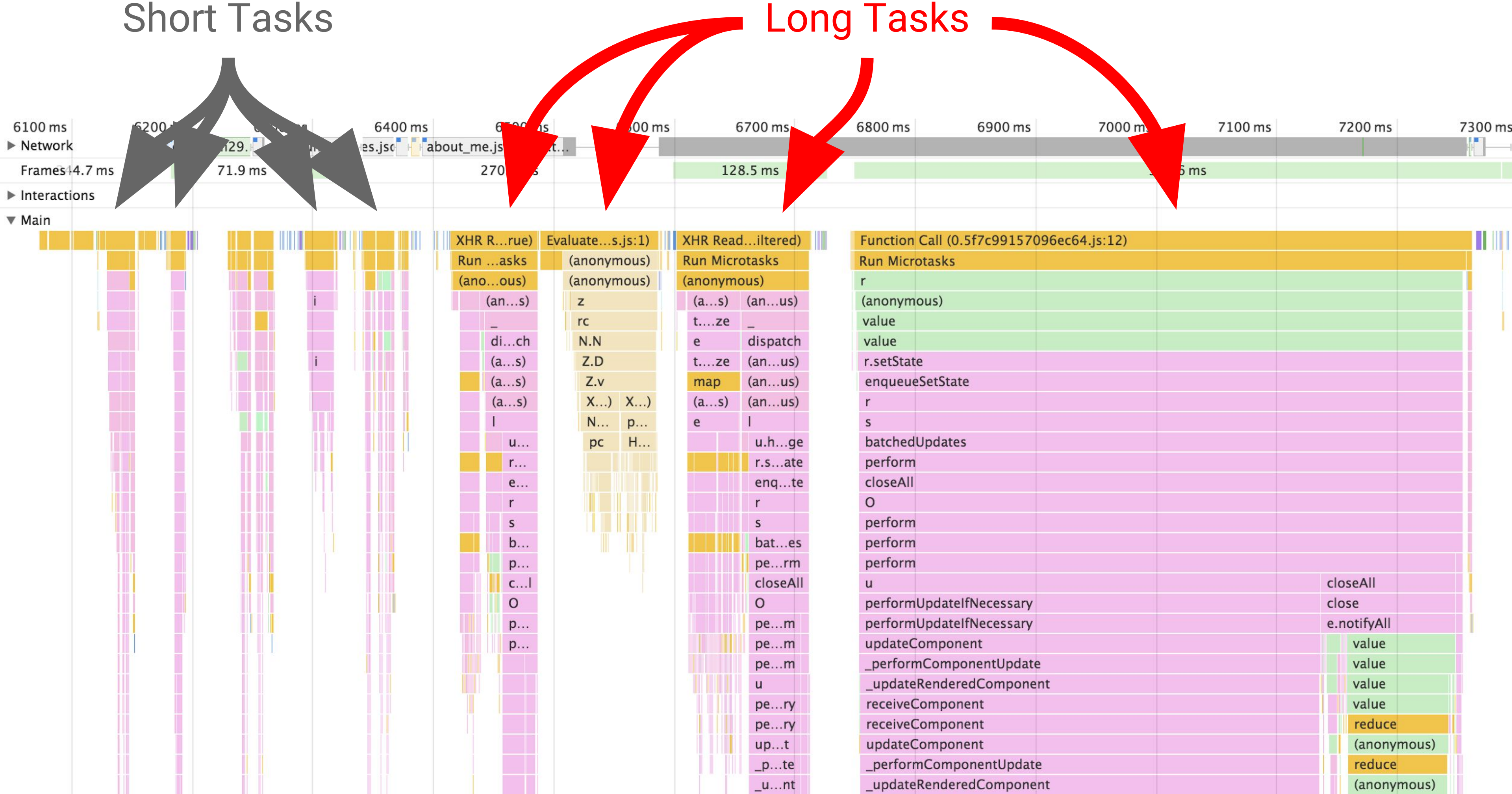
- What metrics accurately capture responsiveness?
- How to measure these on real users?
- How to analyze data and find areas to improve?

Responsiveness Metrics

Queueing Time

Short Tasks

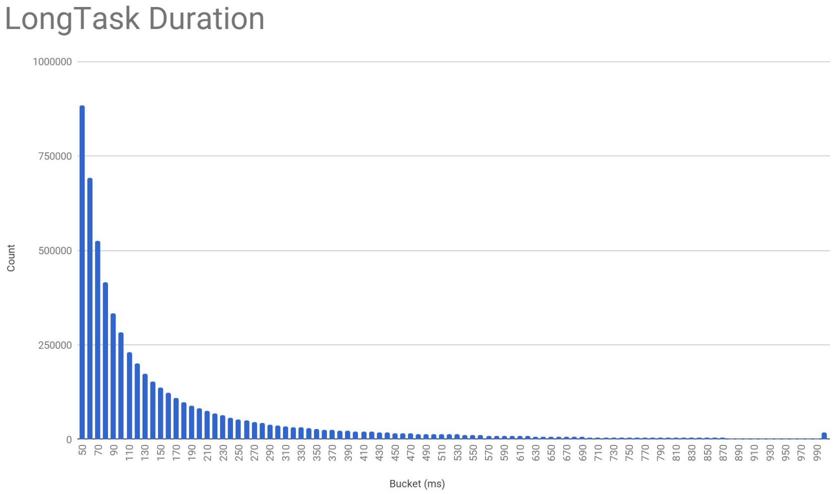
Long Tasks



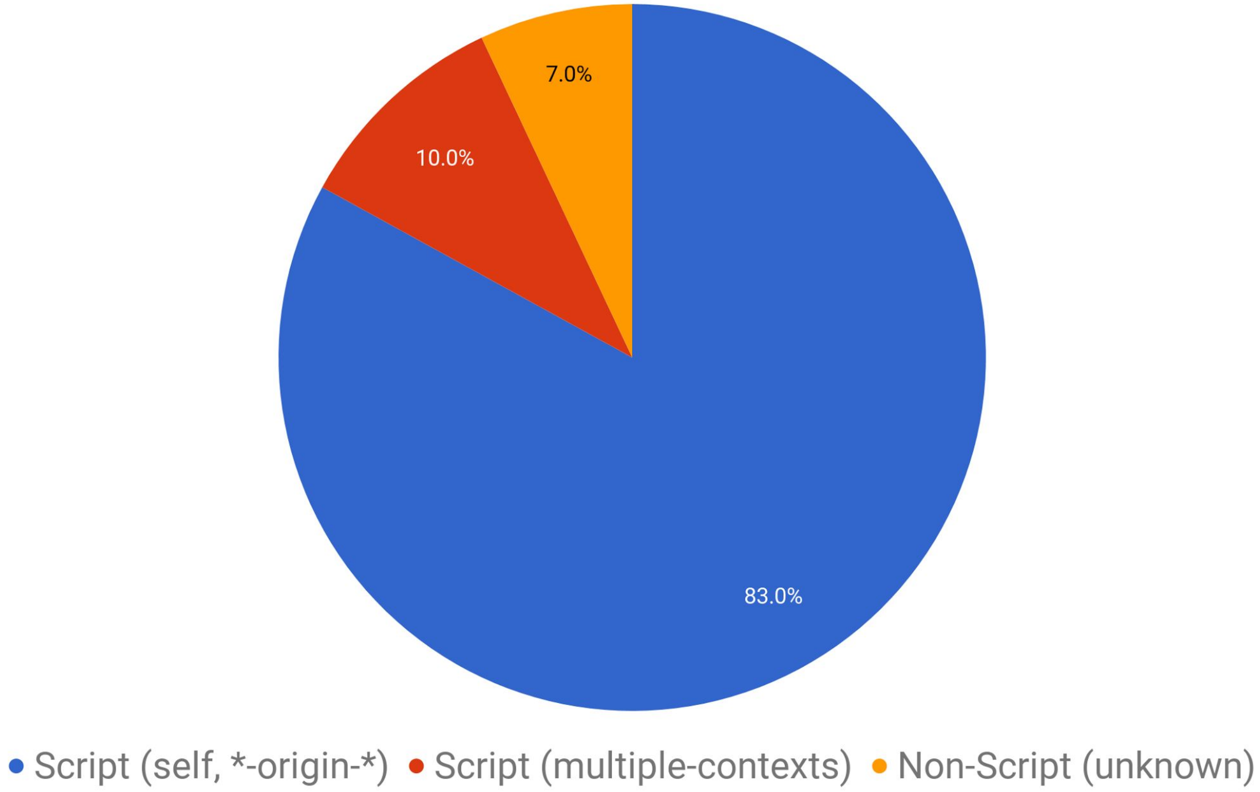
Millions of Long Tasks

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?



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

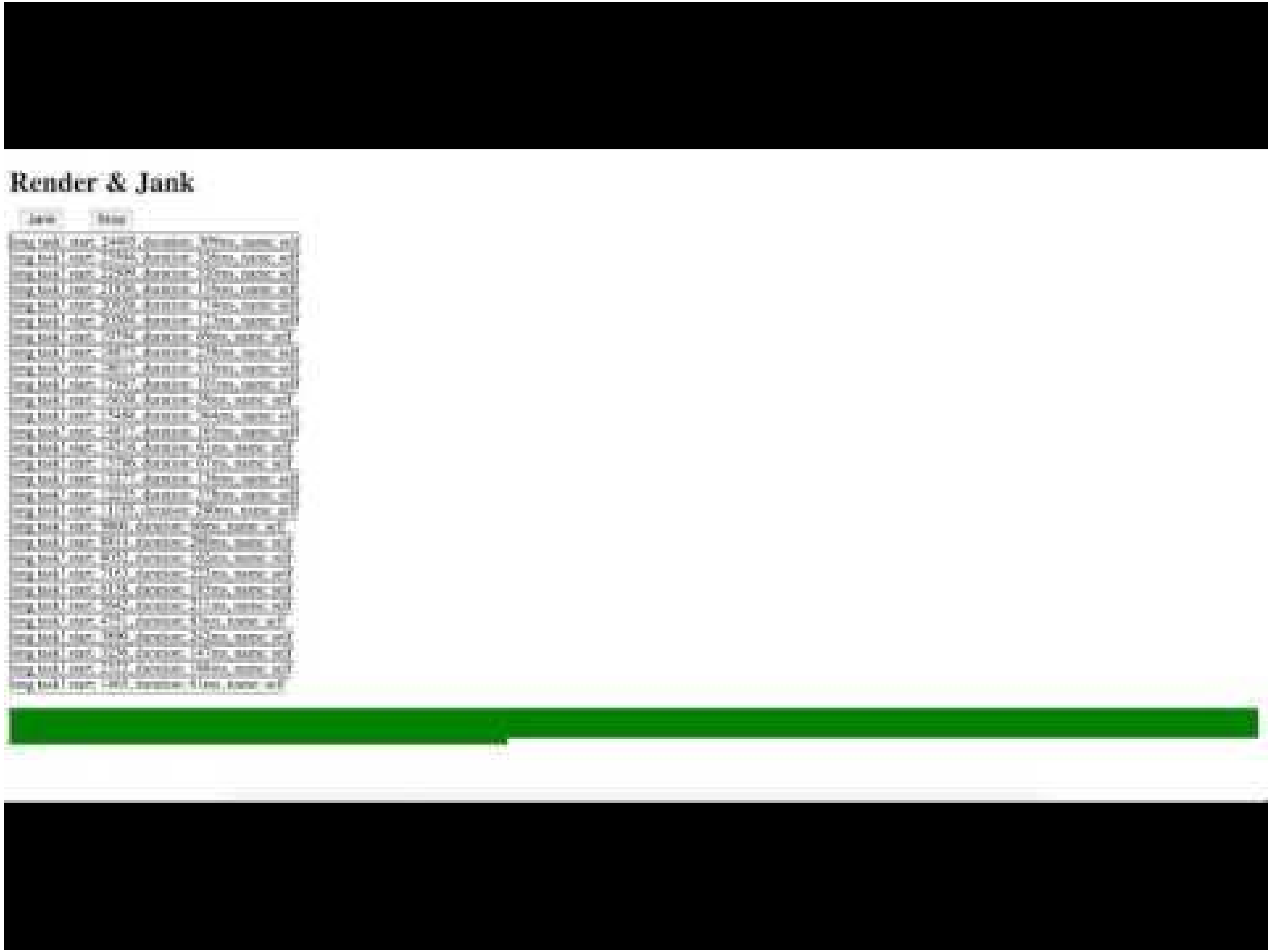
Issues

- Performance overhead
- Battery drain
- Precludes rIC
- No attribution

LongTasks via PerformanceObserver

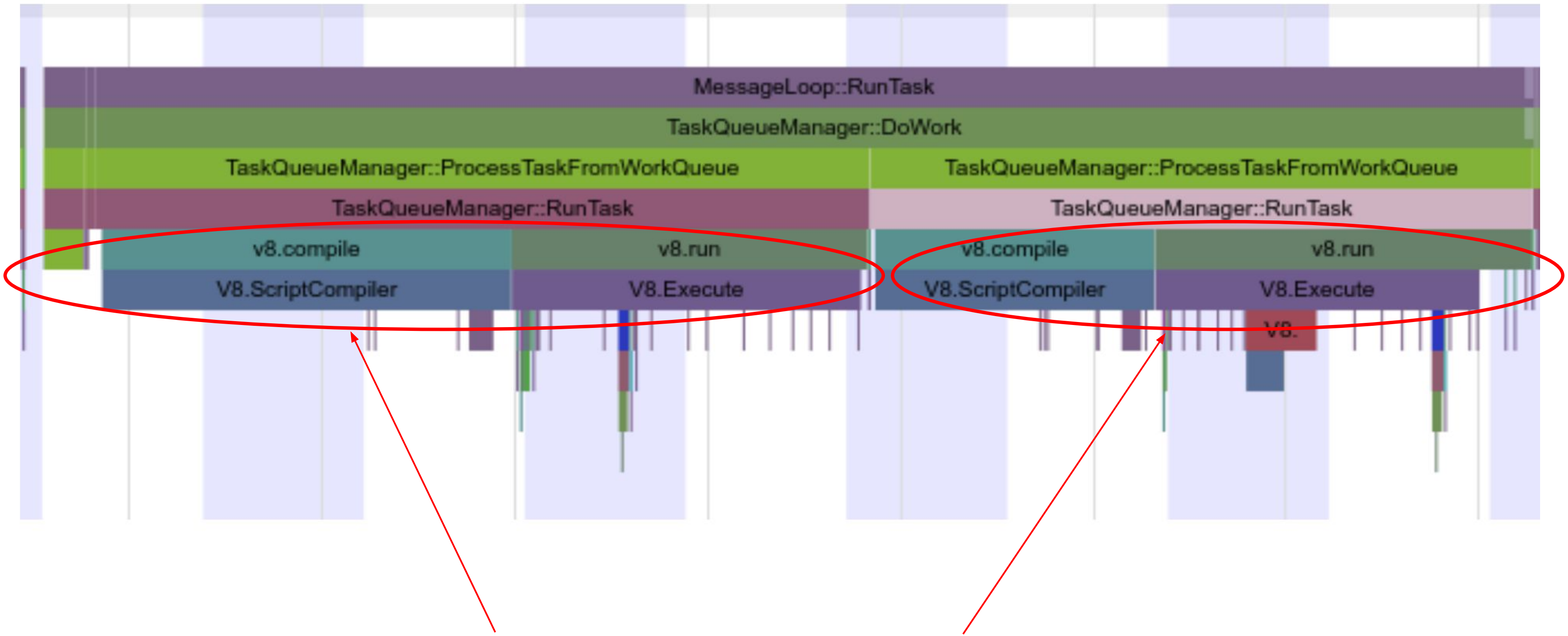
```
const observer = new PerformanceObserver((list) => {
  for (const entry of list.getEntries()) {
    sendDataToAnalytics('Long Task', {
      time: entry.startTime + entry.duration,
      attribution: JSON.stringify(entry.attribution),
    });
  }
});

observer.observe({entryTypes: ['longtask']});
```



Render & Jank

https://w3c.github.io/longtasks/render-jank-demo.html


MessageLoop::RunTask
TaskQueueManager::DoWork
TaskQueueManager::ProcessTaskFromWorkQueue
TaskQueueManager::ProcessTaskFromWorkQueue
TaskQueueManager::RunTask
TaskQueueManager::RunTask
v8.compile
v8.run
v8.compile
v8.run
V8.ScriptCompiler
V8.Execute
V8.ScriptCompiler
V8.Execute
V8.

script 1

script 2

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, objec`
  - `containerSrc: <iframe src="http://`
  - `containerId: <iframe id="ad" />`
  - `containerName: <iframe name="ad-unit-1" /`
  - **`scriptUrl:`** `https://connect.facebook.net/en_US/fbevents.js:97`

Long Tasks V2!

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`.

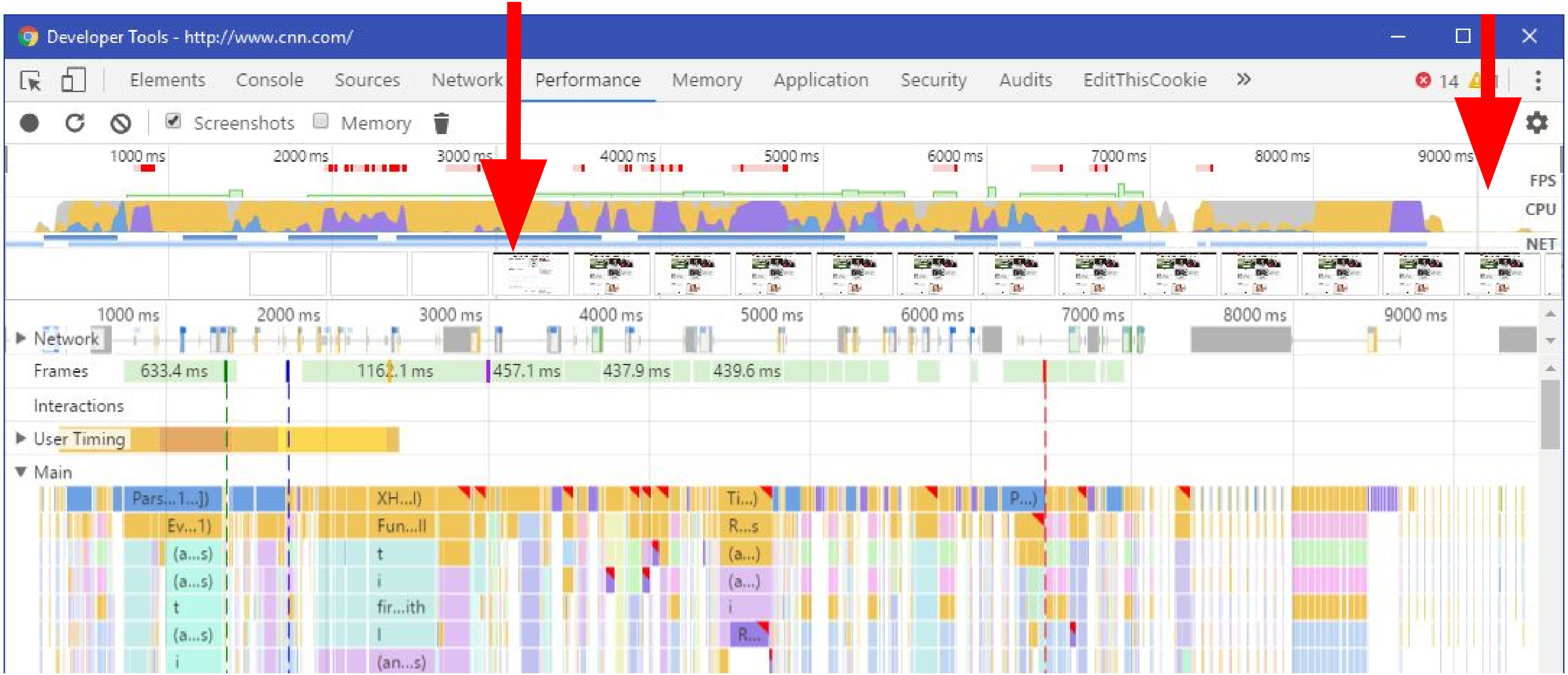
Time to Interactive

Is it Usable?

Time to Interactive

User Sees Content

Not Interactive Until Here

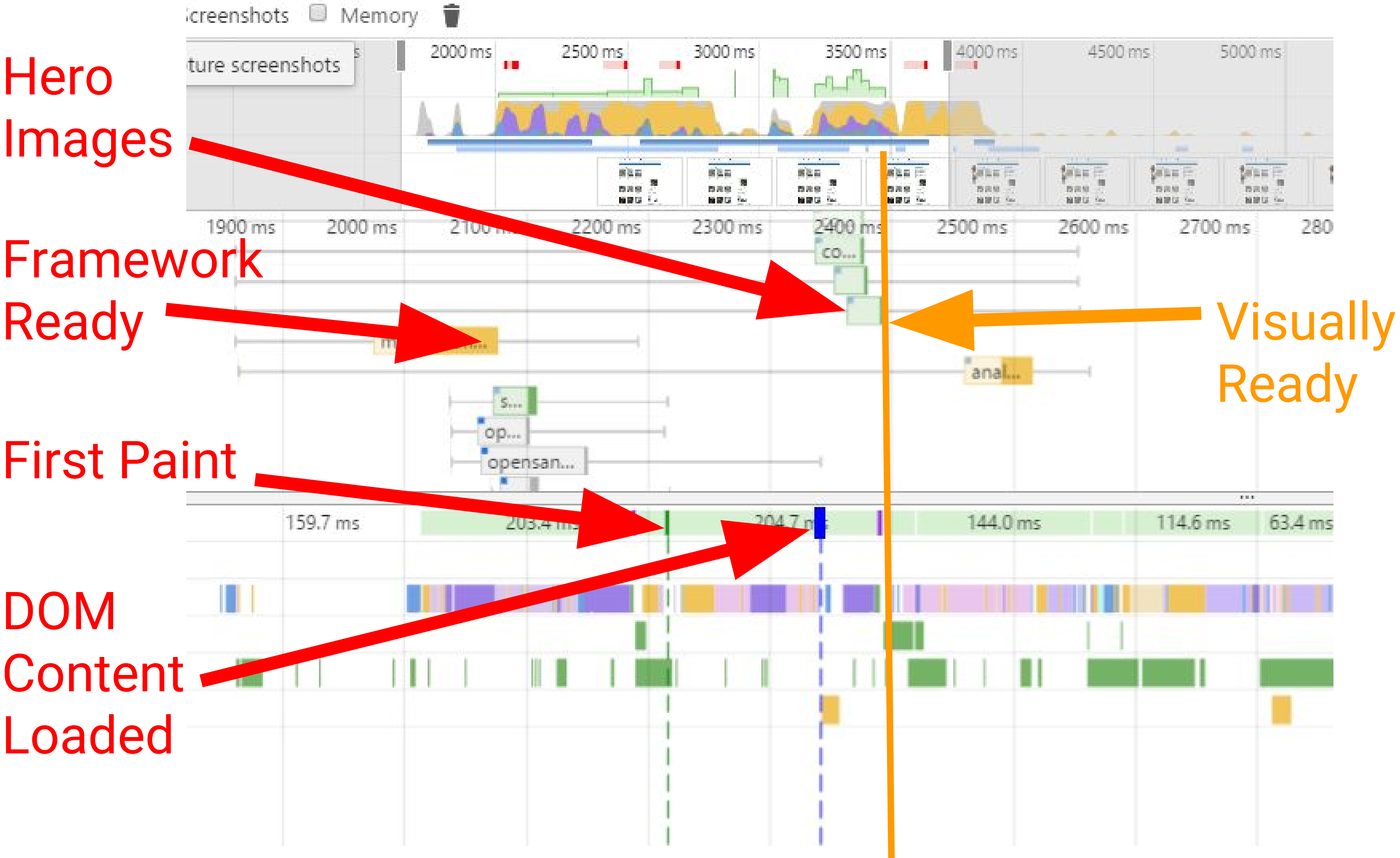


Time to Interactive: Lower Bound

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)



Hero Images

Framework Ready

First Paint

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 (`performance.now() - event.timeStamp`)
- Latency can be attributed via LongTasks

Measure input latency: `event.timeStamp` and `performance.now()`

```javascript
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

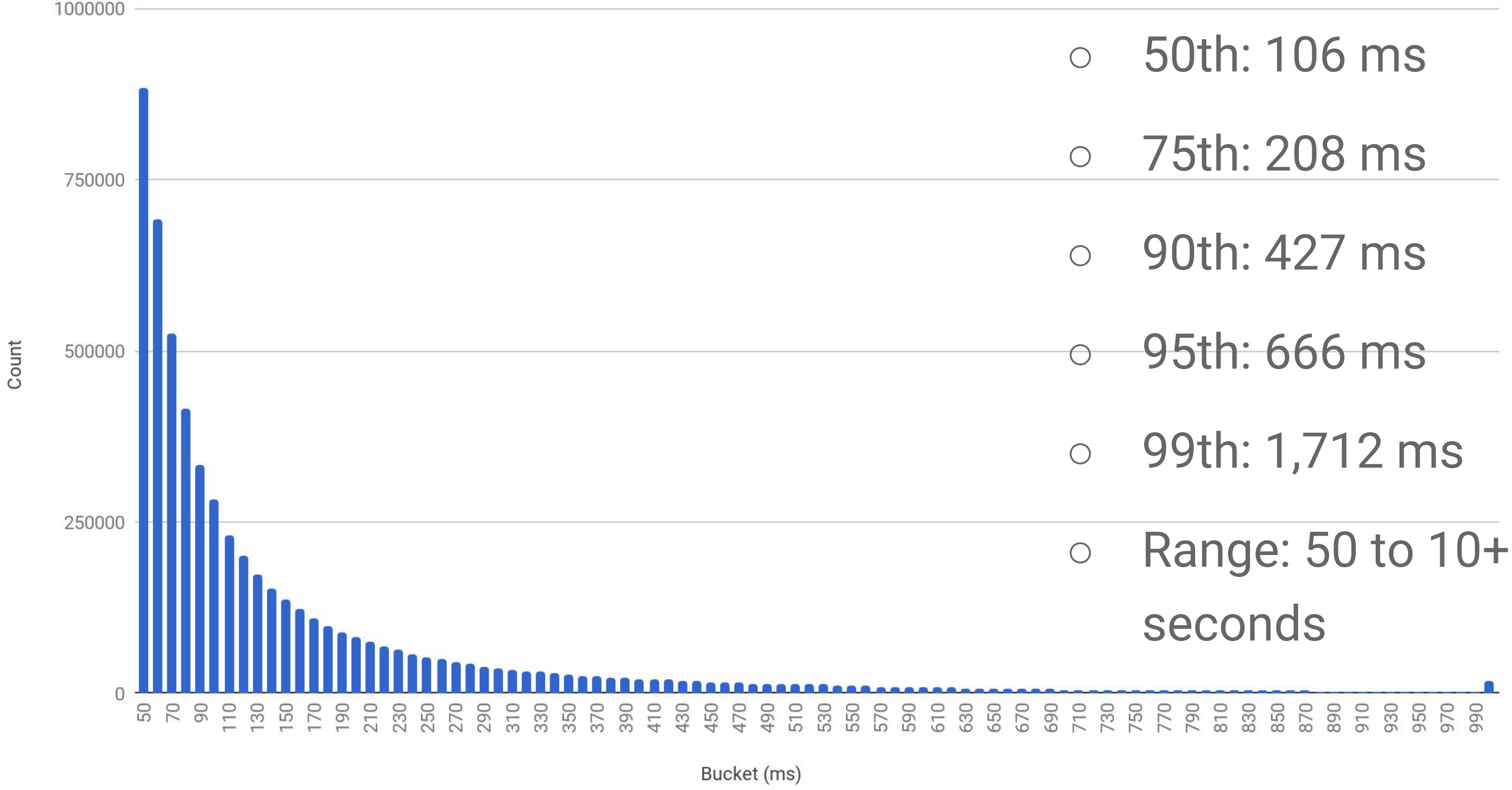
Case Studies

3 sites over 1 month

- Site 1: Travel (ads, social)
- Site 2: Gaming (ads, social)
- Site 3: Retail (social, 3p, spa)

18+ million LongTasks

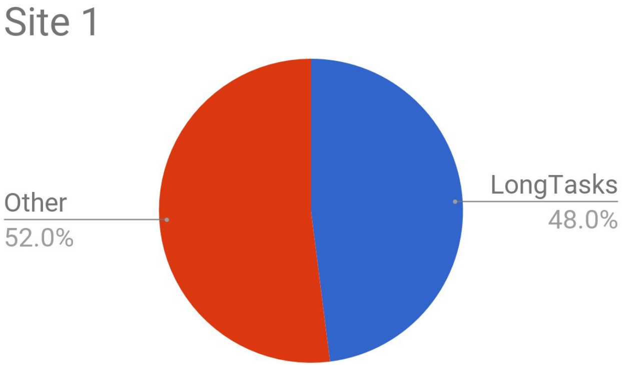
LongTask Duration

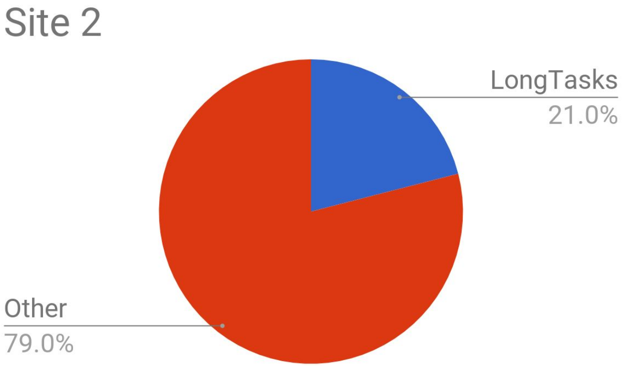


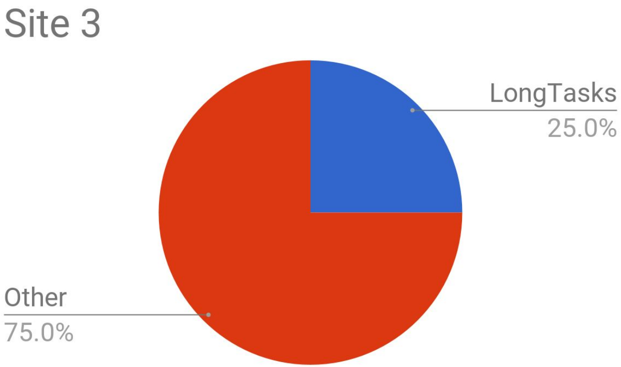
Duration Percentiles:

- ○ 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

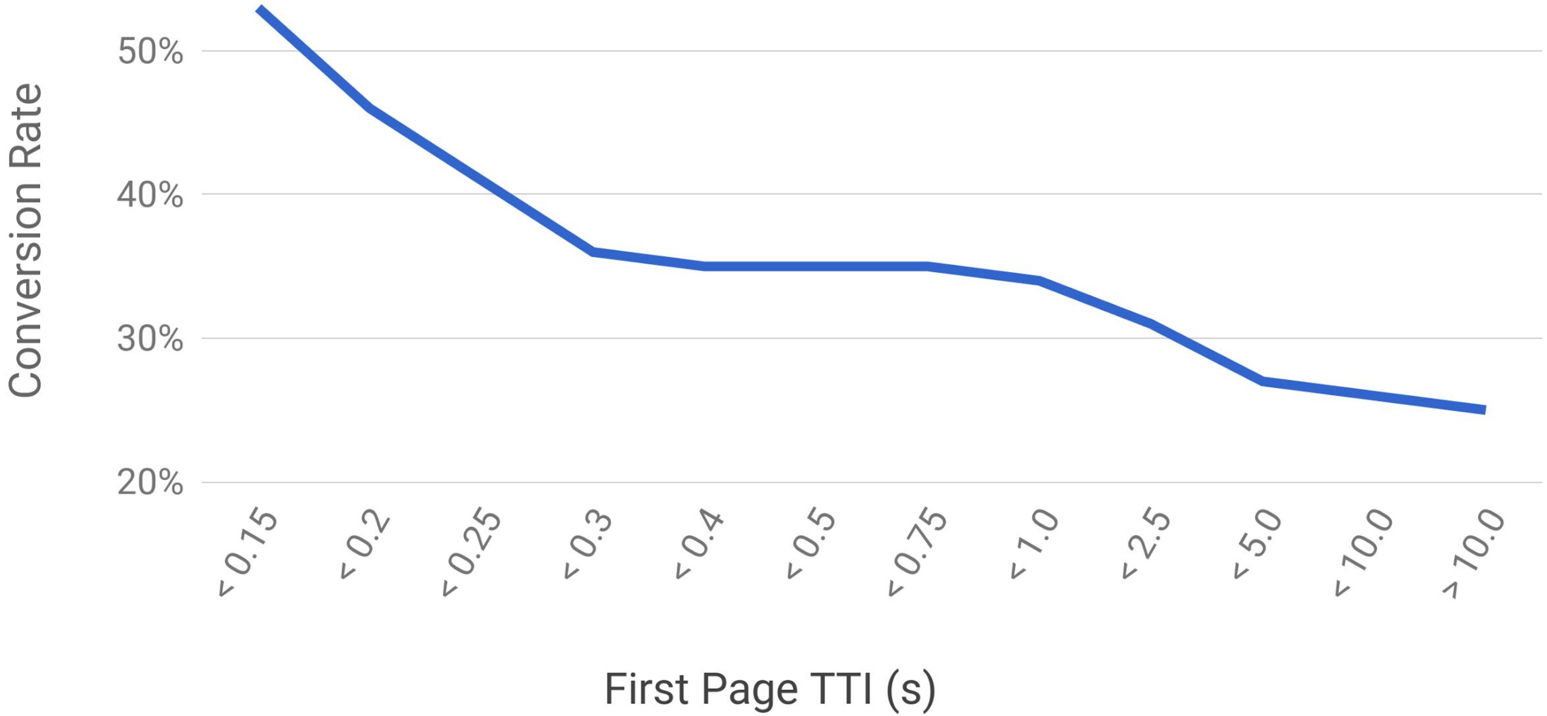




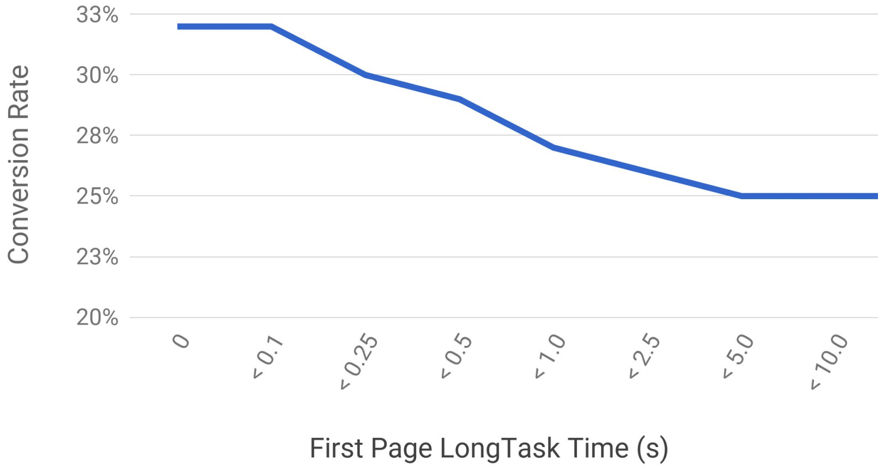


LongTasks directly delay Time to Interactive.

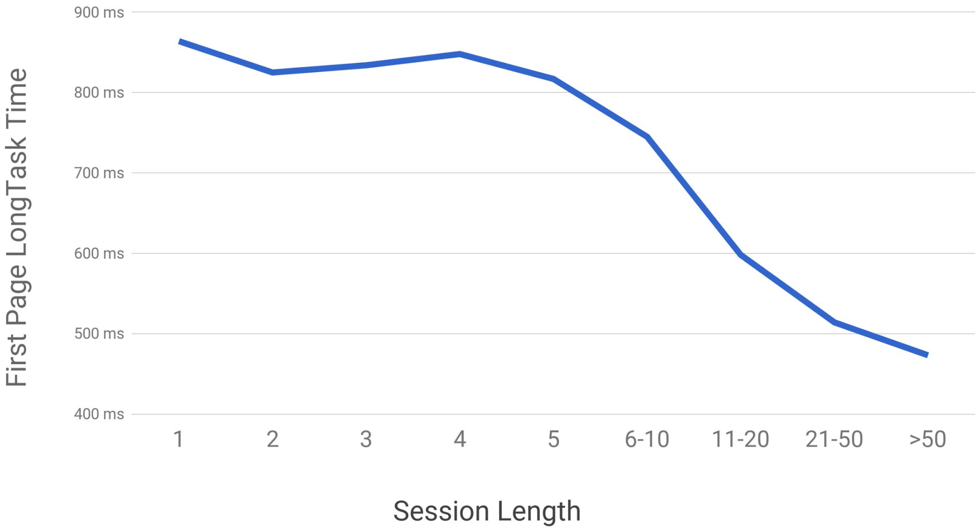
First Page TTI vs Conversion Rate



Time to Interactive has high correlation with overall conversion rate.



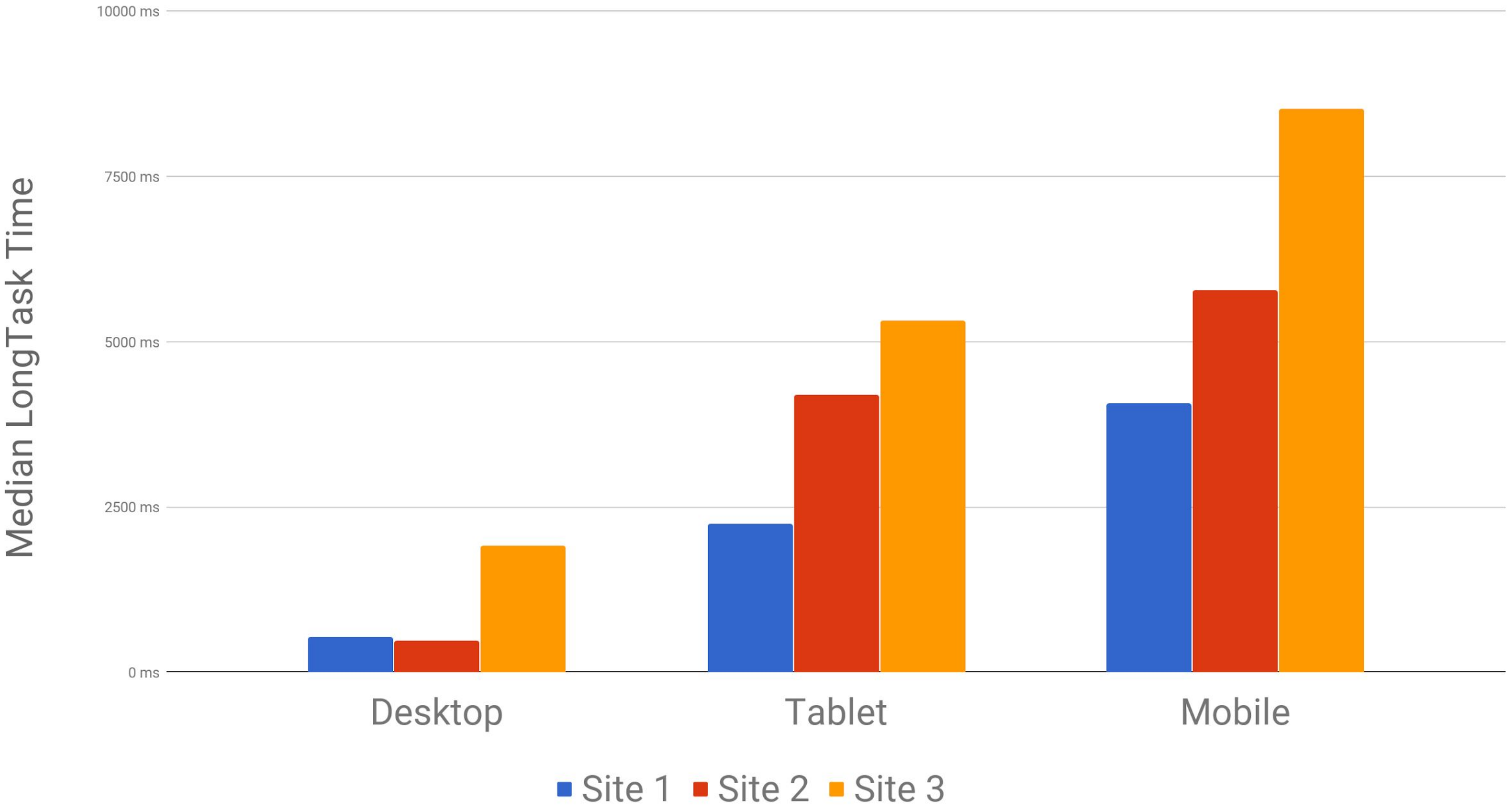
First Page LongTask Time vs. Conversion Rate



First Page LongTask Time vs. Session Length

First impressions matter: as first-page LongTask time increased, overall Conversion Rate decreased

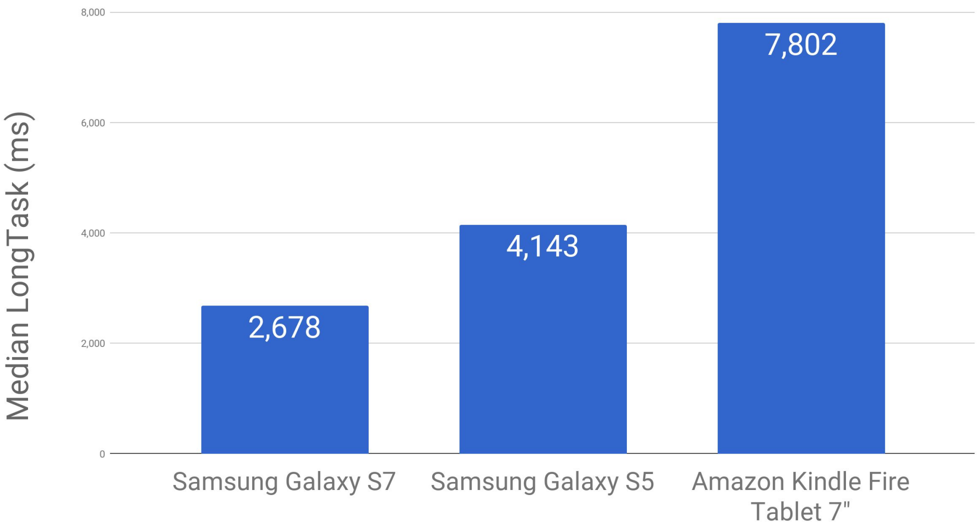
Desktop vs. Tablet vs. Mobile



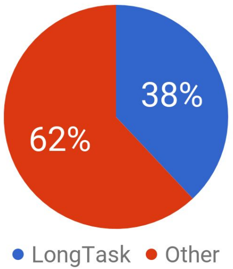
Mobile devices could see 12x LongTask time as Desktop.

LongTask as
% of Front End Load Time

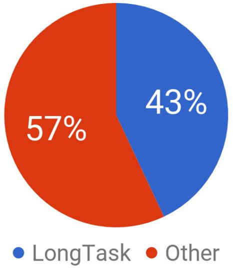
Sample Devices



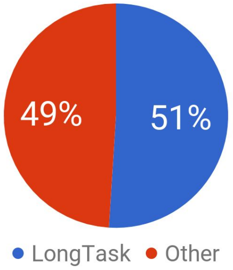
Samsung Galaxy S7



Samsung Galaxy S5

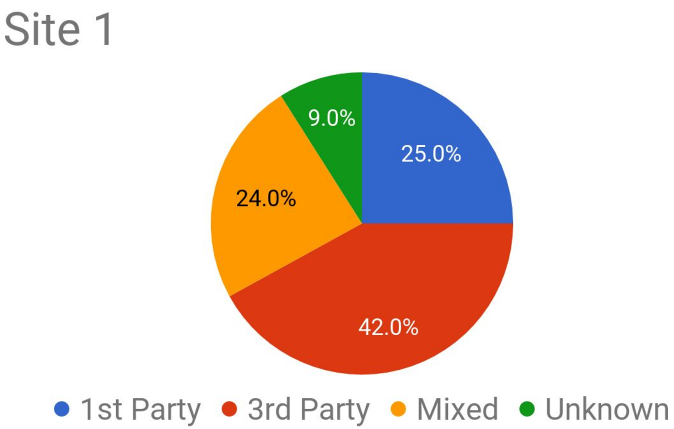


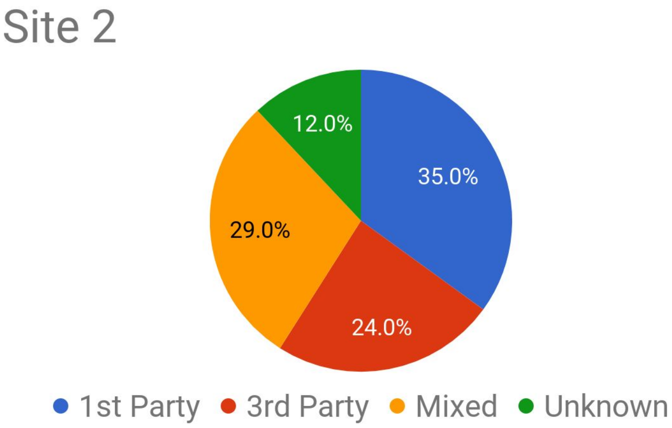
Amazon Kindle Fire Tablet 7"

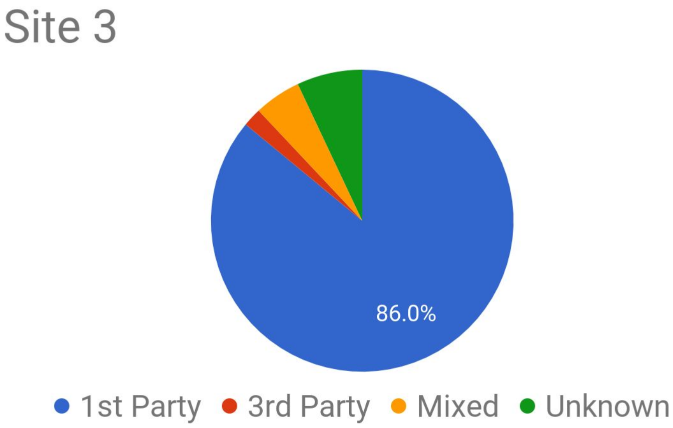


Older devices could be spending half of their load time on LongTasks.

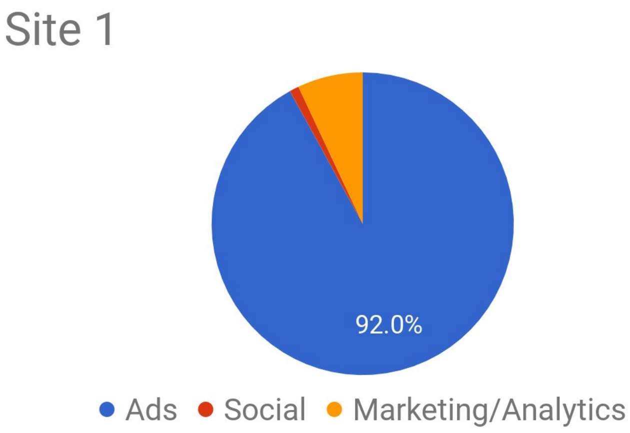
1st vs. 3rd Party

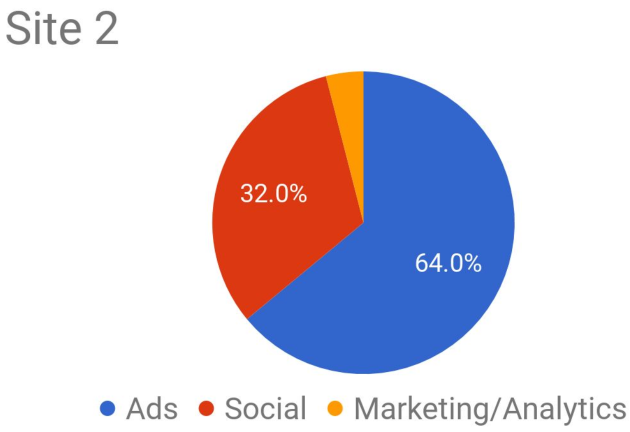

Site 1

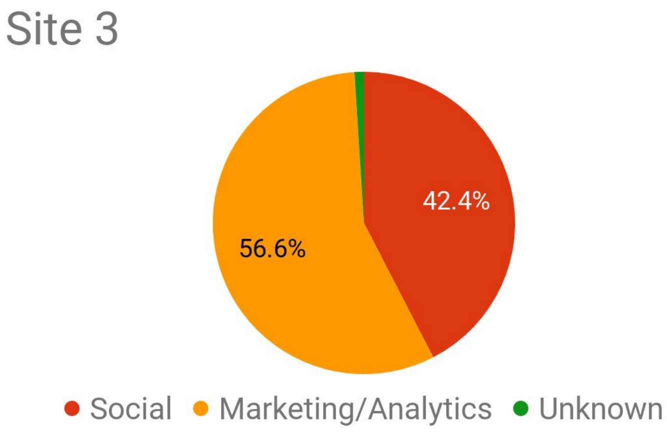

Site 2


Site 3

3rd Party Types







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