[PUBLIC] Summary of our Web Perf WG Proposals @TPAC 2016

Google internal doc is here.

Long Task Observer

Verdict

Move forward with collecting data and feedback from canary & Origin Trials. Security review necessary to move forward with spec.

Next steps

0. Move explainer to WICG.
1. security review
2. get feedback: make V1 fully available in canary (currently available in canary without cross origin frame context, blocked on #1) and origin trial in M55
3. collect benchmark data for overhead
4. collect usage pattern data to help validate / firm up V2 attribution

Notes

Security / Privacy:

- Ryosuke / Ollie: major concern around exposing frame context URL for cross-origin iframes
- Paul: Consider mitigations: remove startTime and fuzz duration to 16ms increments
- startTime is useful for analytics to build timeline
- Ollie (Mozilla): this will likely never be enabled in Tor browser

Attribution

- much interest in attribution
- V1 is primarily useful for third parties, V2 is very useful for first parties;
- Todd: consider expanding V1 to in include V2 attribution;
- strong interest in attribution for style & layout (not viewable today), in addition to script (can be inferred with developer instrumentation today);
- and attribution *within* scripts: when script triggers style / layout or expensive things in browser eg. IPC

Shubhie: TaskAtrributionTiming is designed to be nestable for this reason
Ryosuke: could show style, layout, script as % of top level task time vs. actual duration

- strong interest in surfacing indication of "**blocked input**" - eg. show input events that were blocked by the long task
- don't show paint timing to start with; also concerns with :visited
- Nolan: handling micro-tasks, large amounts of these block input
- handling tabs: show notifications only to tabs that share process / message loop

Threshold 50ms

- 50ms is not sufficient for state of the web today: sites with third party content have copious amounts of 50ms tasks. But developers can set higher threshold on their side (eg. ignore notifications < 100ms)
- Ryosuke: platform dependent; frame rate dependent
- Ilya: 50ms is based on user responsiveness, so could be okay as default across platforms

Overhead

Shubhie: if overhead is significant then could consider sampling i.e. show attribution only 1 out of N times; primary culprit is getclocktime - platform dependent
Nate: developers can sample on their side vs. inside the platform; provide guidance on sampling
Ollie: overhead of crossing the binding layers for reporting notifications
Todd: first collect benchmark results, then continue discussion

Developer feedback

- Todd: origin trial won't work for MSN due to https, instead can work with outlook (?) site

- Nate: could try out in FB, replace their polling workaround?

First Paint & First Contentful Paint

Verdict

Move forward with specs for FP & FCP.

Next steps

1. Figure out how to spec these.
2. Collect data

Notes

First Paint
- defining First Paint in terms of colors is problematic. How can this be defined without color assumptions?
- could developers optimize FCP at the expense of FP?
- Ilya: provide pointer to Steve's research

First Contentful Paint
- use notion of initial containing box in spec?
- current definition covers 80% case - good enough?
will keep refining the definition, but okay starting point
- include background images

- gameability: not too worried; the metrics will be game-able to some extent

Processing
- before or after layout ?
ideally at swap buffer; latest timestamp you can commit to or after layout
- Ryosuke: add step #13 as checkpoint - earliest time to fire?
- Elliott: alternately, spec as "when you could have run a rAF"
- Todd: concerns with inaccuracy, not accounting for glass time
- getting full glass time is hard and platform dependant; not really feasible

Interface
- consolidate into 1 interface
- drop "paint" in interface, as we may add other things in the future (TTI?)

user timeline?
- expose with PerformanceObserver; but also want to buffer; follow up on discussion from last F2F about buffering with PerformanceObserver

Time to Interactive (TTI)

Verdict

Not ready for web perf API.
(Although very useful for lab / synthetic).
TTI can move depending on user interaction; it can be derived in JS from FCP & Long tasks.
Start with exposing as JS library and gauge developer feedback.

Notes

More detailed notes here: https://www.w3.org/2016/09/23-webperf-irc
Ilya: 10s window seems long?
Paul: based on data from gmail; first user interaction is near 3s
Ilya: interaction is safe when you have no long tasks?
Paul: ran this definition on the top 1000 sites; feels good
rniwa: users may try to interact before 10s
Nate: will push out TTI; punishing metric if user interacts early
yoav: it's possible that the page is interactive before TTI
rick: how about starting from a open source lib based on long task and collect data
Alex: amortize based on if people spend a long time on the page

Hero Element Timing

Verdict

Strong support from everyone. Move forward with spec for timing for element is painted and added to DOM (not parse time).

Next steps

1. Figure out how to spec; overlaps with FP / FCP for “paint” timing
2. Prototype & collect data

Notes

- Handling off-screen: arguments for both providing and not providing timing for off-screen (not in viewport) elements
Overlap with Intersection Observer: it doesn't provide any timing
- handling images: start or end time for image render?
Ryosuke: end time is more useful than start loading for large images

- New attack vector: leaking image decode time
for v1, don't provide information for CORS images

- Elliott: Images are bit of an exception; treat separately
Maybe start with empty div time for images, later in v2 figure out how to provide more useful image time

- discussion on parse, tokenize time
verdict: don't expose parse time; okay to expose "added to DOM"

- Kenji: FMP can be calculated based on Hero timing down the road
when all the hero elements are painted in the viewport