⚠ IMPORTANT INFORMATION ⚠

Technical, Best Practices & Etiquette

**Pro Tip: Please check your Network Connectivity Settings & browser compatibility prior to your talk*

- Please review Best Practices for Speakers
- Preferable to use a hard line connection over wireless & headset
- Find a quiet location with a background that's not too distracting
- Have a light source in front of you to help bring the focus on you
- Put your devices in "Do Not Disturb" mode to silence notifications on your computer & other nearby devices
- Mute your mic when you are not speaking
- Look directly into the camera as much as possible to establish eye contact with your viewers
- sign up for a Hopin Demo or watch a short orientation video for attendees

IMPORTANT INFORMATION

Breakout Session Instructions

To present your slides live or pre-recorded talk

1. When it’s time for you to start your session, click on your session you have been assigned in your calendar invite
    - Choose how you would like to capture captions for your presentation:
        - [Syncwords Integration]
        - [Accessibility: Tips on Using Captions and Translations]
        - [Creating Captions with Google Slides]
        - [Creating Captions with Google Meet and Zoom]
        - [Webcaptioner]
    - Click share audio and video
    - Once you are on screen, you are live to anyone viewing the session
2. As the moderator, you will be able to bring people onto the screen from the moderation panel in your bottom left corner.
3. Click the “Return” button to go back to the “Main Event page” when your session is complete

IMPORTANT INFORMATION

Live Captions on Google Chrome

Follow these steps to enable Google Chrome's live captioning tool if you want to see captions anytime audio is detected through your browser:

1. Copy and paste this link into Chrome: **chrome://settings/accessibility**
2. Toggle the “*Live Caption*” control on. When **Live Caption** is on, the toggle turns blue
3. Quit and relaunch Chrome.
    a. The next time Google Chrome detects audio (whether you're watching a video, chatting on Google Meet or attending an event on Hopin), the **Live Caption** window will automatically appear.

IMPORTANT INFORMATION

Hopin Resources

Hopin Resources

**Pro Tip: Don't have many apps or tabs open while using hopin for best performance*

Tech Troubleshooting and Speaker Resources

- Speaker Instructions
- Organizers, Moderators, and Speakers
- Speakers vs. Moderators
- Attendee Troubleshooting Steps
- Troubleshooting Steps
- Hopin Knowledge Base
- Troubleshooting Tips

Measuring Dropped Frames and Animation Smoothness

Michal Mocny (mmocny@), Chrome Speed Metrics

In this session:

- Introduction to the “Overall Animation Smoothness” project
- Visual Completeness vs Smoothness vs Latency
- Where we are, where we’re going
- How to play with it

What is Overall Animation Smoothness?

Overall Animation Smoothness

Goal: Label visual completeness of animation frames, **during animations**.

Goal: Identify which frame updates **matter the most to users**.

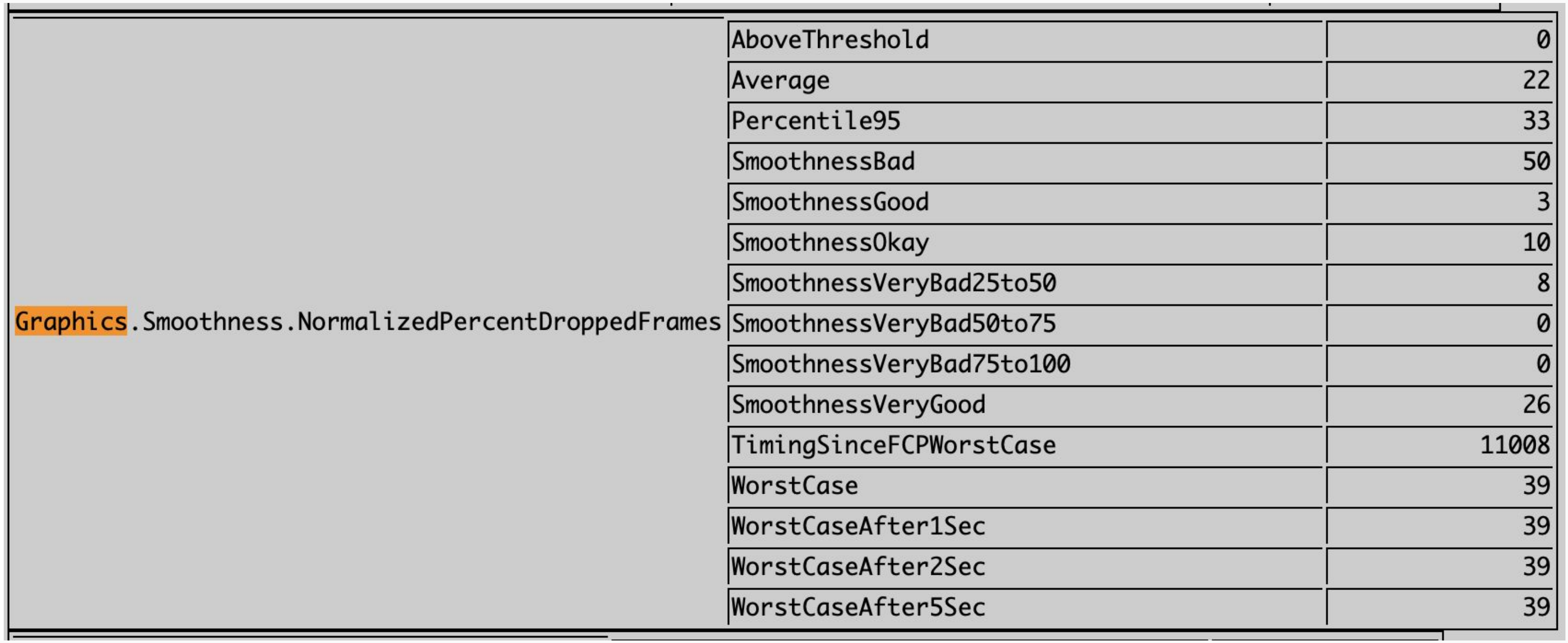
Goal: A metric (web vital) that measures a **full page lifecycle** experience.

web.dev/smoothness

Overall Animation Smoothness

Why is it different than existing metrics?

- Captures visual smoothness during **active animations**
- Accounts for **all animations** types within one metric
- Measures renderer, meaning it includes **jank caused by developers**
- Tries to match up with **user perception**: was it actually painful?
  - **Precision** vs Recall



Graphics.Smoothness.NormalizedPercentDroppedFrames	AboveThreshold	0
	Average	22
	Percentile95	33
	SmoothnessBad	50
	SmoothnessGood	3
	SmoothnessOkay	10
	SmoothnessVeryBad25to50	8
	SmoothnessVeryBad50to75	0
	SmoothnessVeryBad75to100	0
	SmoothnessVeryGood	26
	TimingSinceFCPWorstCase	11008
	WorstCase	39
	WorstCaseAfter1Sec	39
	WorstCaseAfter2Sec	39
	WorstCaseAfter5Sec	39

Why work on this?

First, we want a better guardrail metric.

Second, we want to develop an Animation Smoothness Web Vital.

Finally, ship a web performance API. Common feedback from developers:

- Replacement for rAF polling
- Understand the performance of scrolling
- Confusion about “checkerboarding”
- UX: Site redesign added a bunch of animations… measure effects on performance

Visual Completeness
Smoothness
Latency

Visual Completeness

- Each frame opportunity, how much work is included and presented?
- Four possible states for an animation frame:
    - **No Update Desired**
    - **Fully Presented**
    - **Partially Presented**
    - **Dropped**

Life of a Frame

BlinkOn: “Tracing Frames 101” breakout tomorrow

Beyond Visual Completeness...

- However, there are other factors, beyond just if frames updates are presented:

- Missing content/paint updates (i.e. checkerboarding)
- Quality vs. quantity (i.e. video bitrate)
- **Detecting active Animations**

Animation Smoothness

- “frame sequences” become really important during active animations…
    - No active animations means dropped frames are (less) visually detectable

- Some animations only require a partially presented (compositor) update
- Some animations can mask other animations (i.e. scrolling)

- Need to classify whether a given frame update factors into smoothness.
    - Main or compositor thread update may *affect smoothness*

Examples of animation updates include:

- Declarative Animations (e.g. CSS animations)
- Videos
- Javascript updating a CSS property of an element regularly
- Scrolling
- Canvas + rAF

Examples of non-animation updates:

- Page loading
- Clicking on a button takes a long time to produce view in response
- Button appearance responding to mousedown.
- Background loading / insertion of new content

When does Compositor thread update affect smoothness?

1. Threaded Scrolling
2. Compositor driven Animations
3. Pinch Zoom
4. Video, using Accelerated Rendering
5. OffscreenCanvas

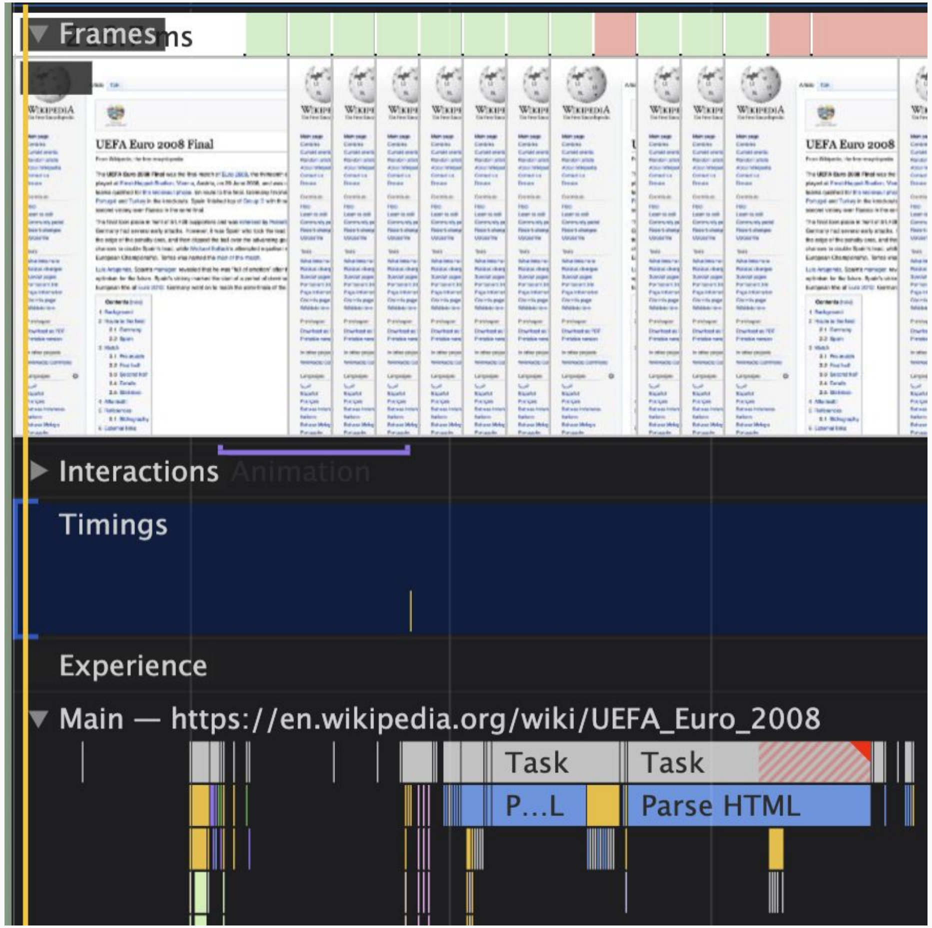
...

When does Main thread update affect smoothness?

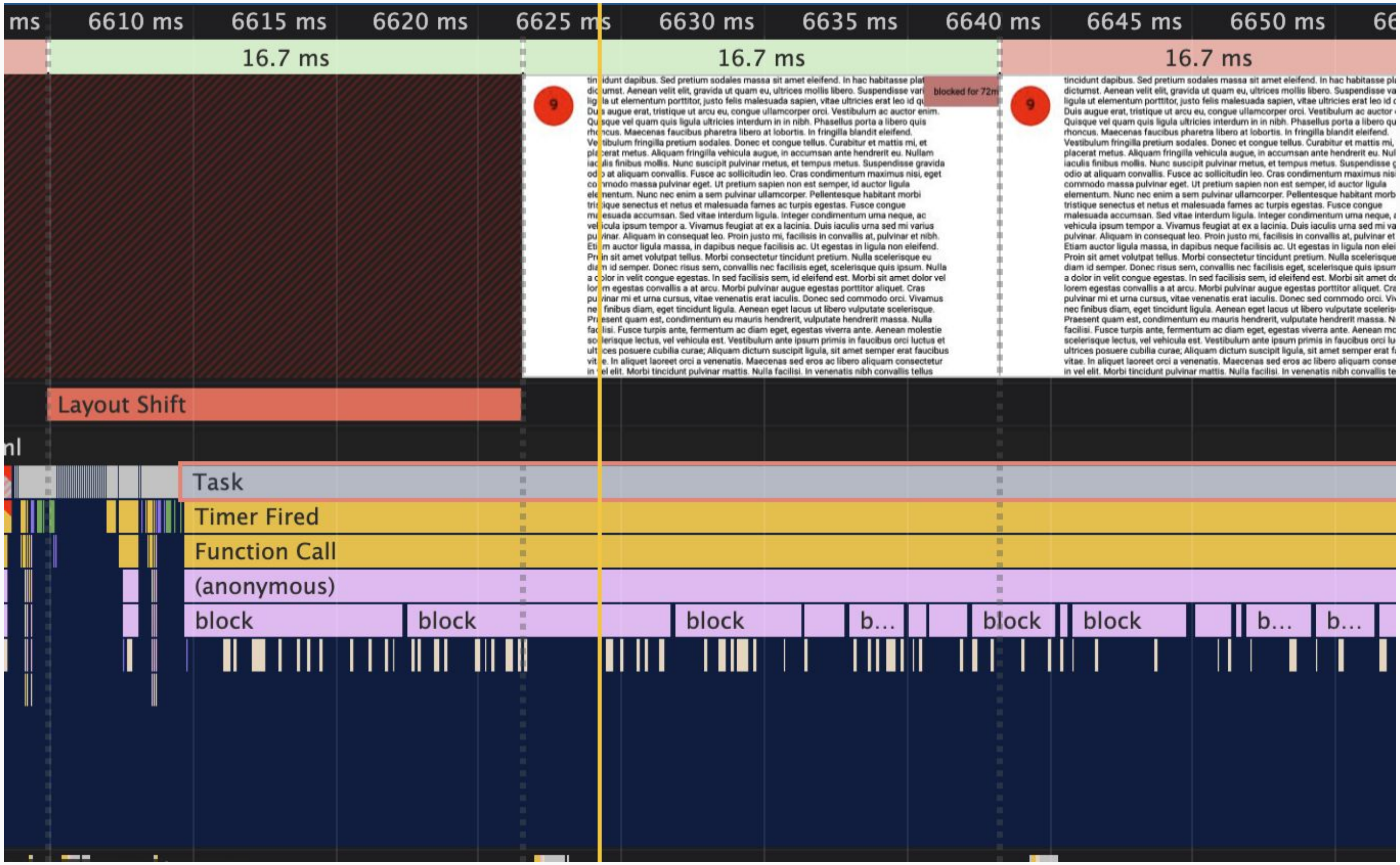
1. has a CSS/web animation update not reflected on the compositor (i.e. main thread animation), or
2. has an active scroll which is blocked on the main thread, or
3. has a playing video which does not support accelerated rendering, or
4. has a canvas invalidation on a canvas element which **was visible** prior to the current frame, or
5. has a **direct style** update of an **animatable property** (see Animating properties) on an element which **was visible** prior to the current frame.

When do we know?

- For main updates in particular, we don’t necessarily know if there will be a visual update **until the frame is complete**.
- Don’t know if the frame will ultimately be Fully Presented or Partial.
- Don’t know if there will be an animation update “affecting smoothness”.
- Have to wait for long tasks to complete and “re-write history”.



Completeness vs Smoothness



Smoothness vs Latency

Possible states for a single Animation Frame

Possible states for a single Animation Frame

Conceptually, every frame is not **boolean.** It has is a **fractional** value.

It may even be worthwhile to consider it as a **probability**.

No Update Desired

Idle time, repeat of the previous frame.

Fully presented

The main thread update was either committed within deadline, or no main thread update was desired.

Partially presented

Compositor only; the delayed main thread update had no visual change.

Partially presented

Compositor only; the main thread had a visual update, but that update did not include an animation that affects smoothness.

Partially presented

Compositor only; the main thread had a visual update that affects smoothness, but a previously stale frame arrived and was used instead.

Partially presented

Compositor only; without the desired main update, and the compositor update has an animation that affects smoothness.

Partially presented

Compositor only but the compositor update does not have an animation that affects smoothness.

Dropped frame

No update. There was no compositor update desired, and main was delayed.

Dropped frame

A compositor update was desired, but it was delayed.

Stale frame

An update was desired, it was produced by the renderer, but the GPU still did not present it before the vsync deadline.

States of a single Animation Frame

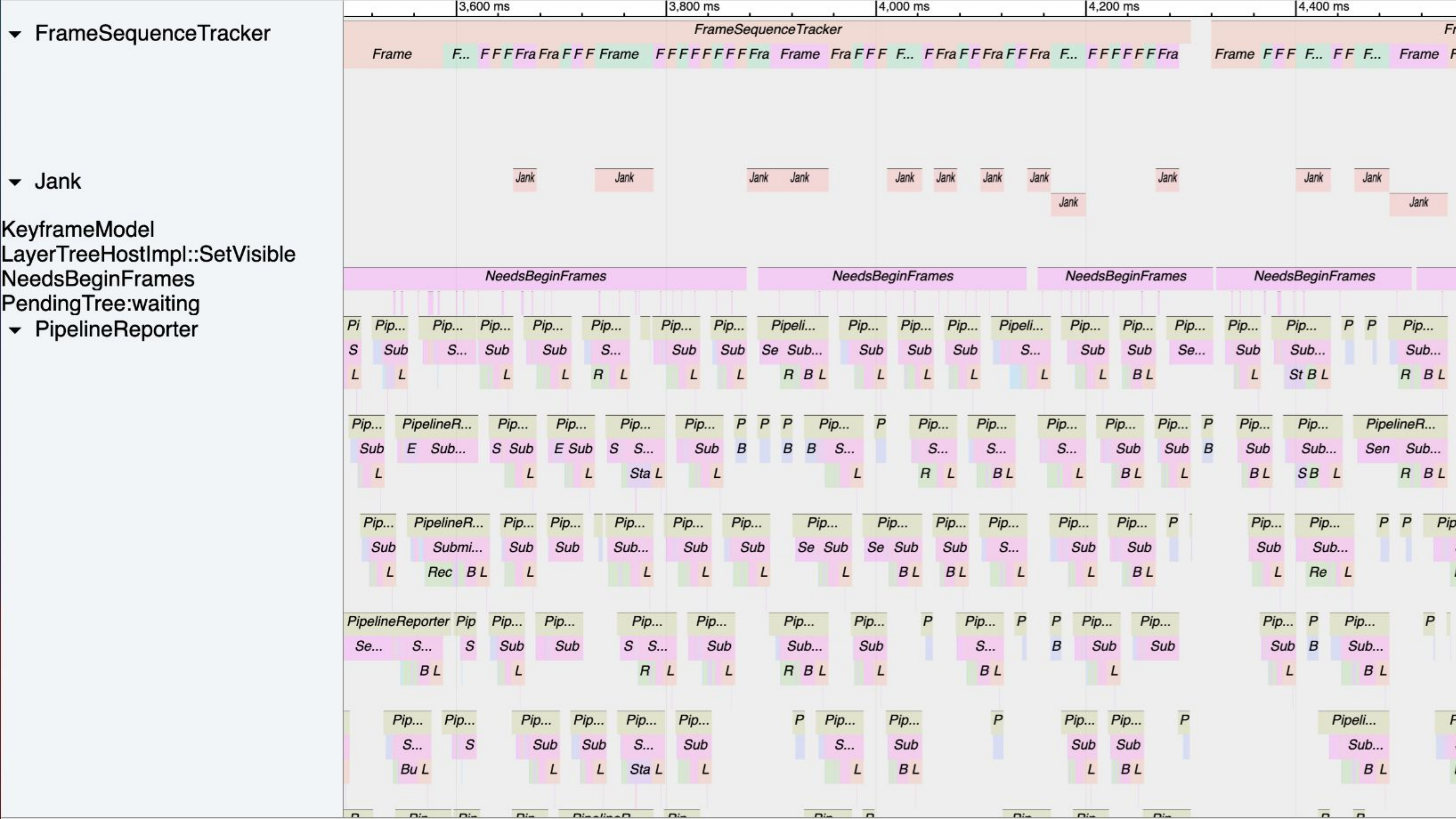
And there’s more...

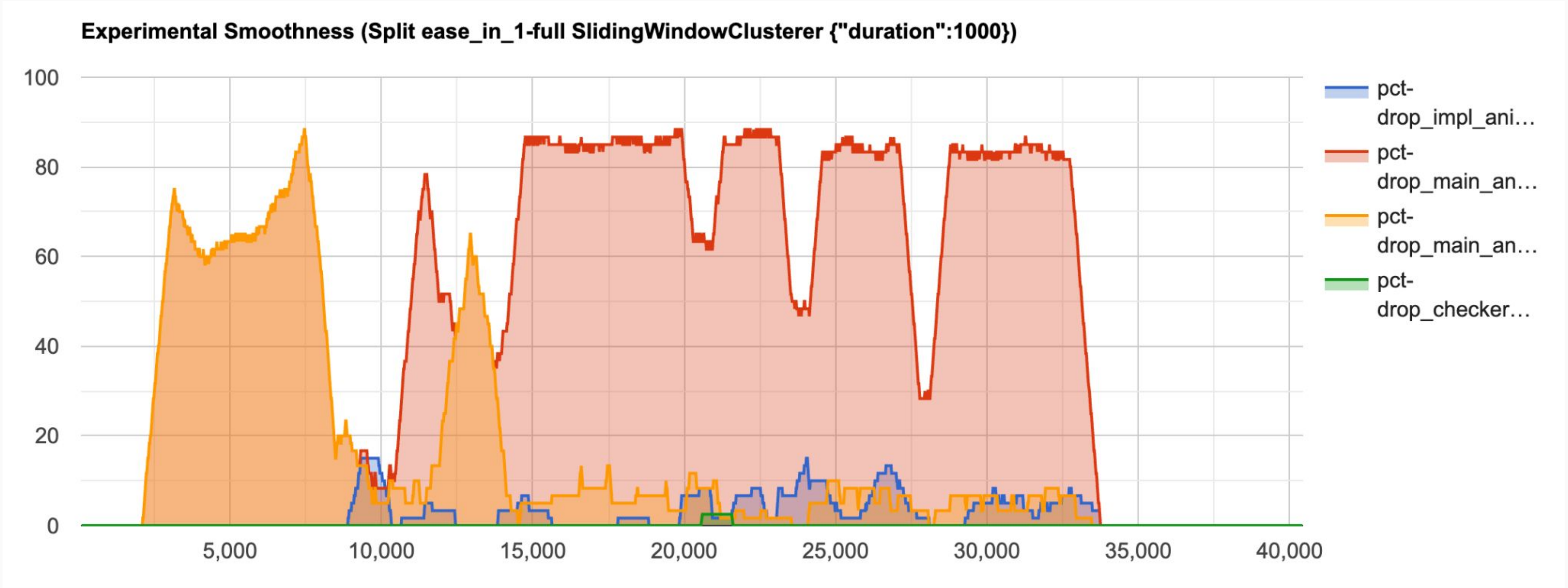
- Layered on top of each of those states are other factors:
  - Missing/stale frame content (checkerboarding, late raster...)
  - Quantity of animations (#elements? Variety of animation types?)
  - Quality of the animation itself (i.e. video, canvas games)
- Long-term aspirations:
  - Visual size of the animation updates (fraction of viewport?)
  - Semantic value of the animation

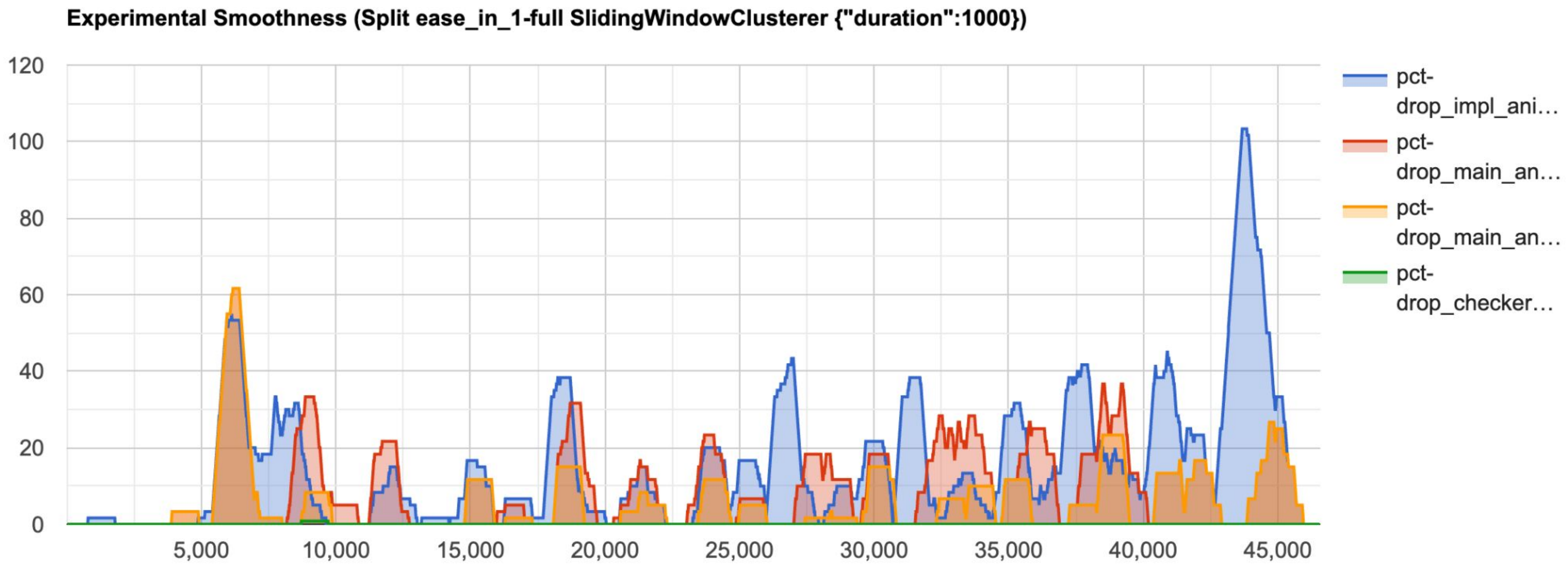
Putting it all together: metric definition

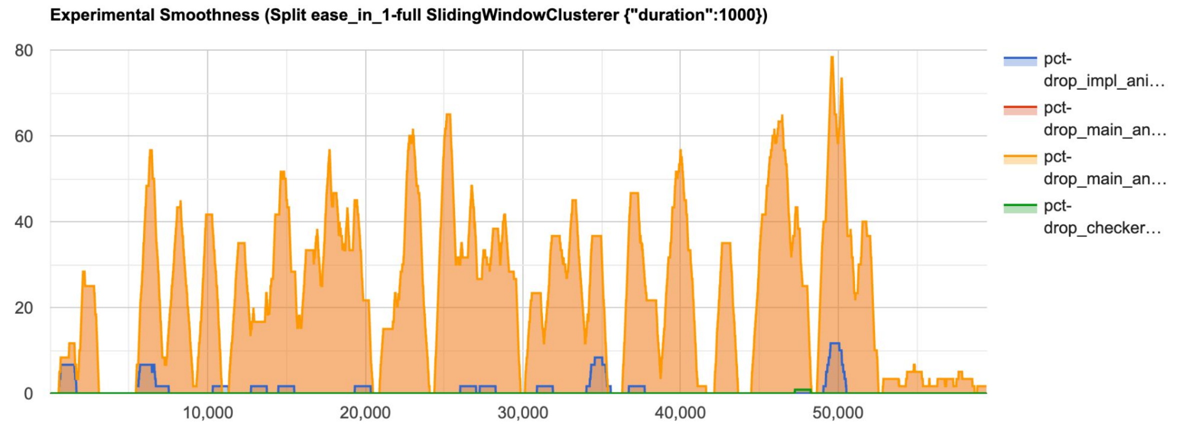
- What matches the natural way users experience smoothness?
  - The proportion of time waiting for important updates.

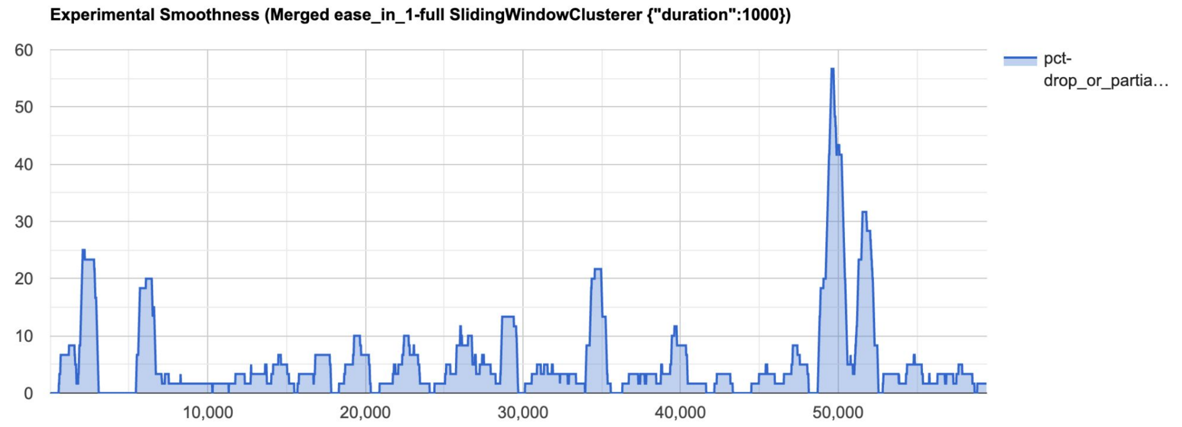
- For **each** animation frame, label smoothness deficits.
- For a specific **series** of animation frames, can try to convert these labels into a running metric.
- For a **whole timeline**, convert everything into an overall score.


3,600 ms
3,800 ms
4,000 ms
4,200 ms
4,400 ms
FrameSequenceTracker
Frame
Jank
KeyframeModel
LayerTreeHostImpl::SetVisible
NeedsBeginFrames
PendingTree:waiting
PipelineReporter
Sub
Submi...
Rec
Sta
Bu
Sen


Experimental Smoothness (Split ease_in_1-full SlidingWindowClusterer {"duration":1000})
100
80
60
40
20
0
5,000
10,000
15,000
20,000
25,000
30,000
35,000
40,000
pct-drop_impl_ani…
pct-drop_main_an…
pct-drop_main_an…
pct-drop_checker…

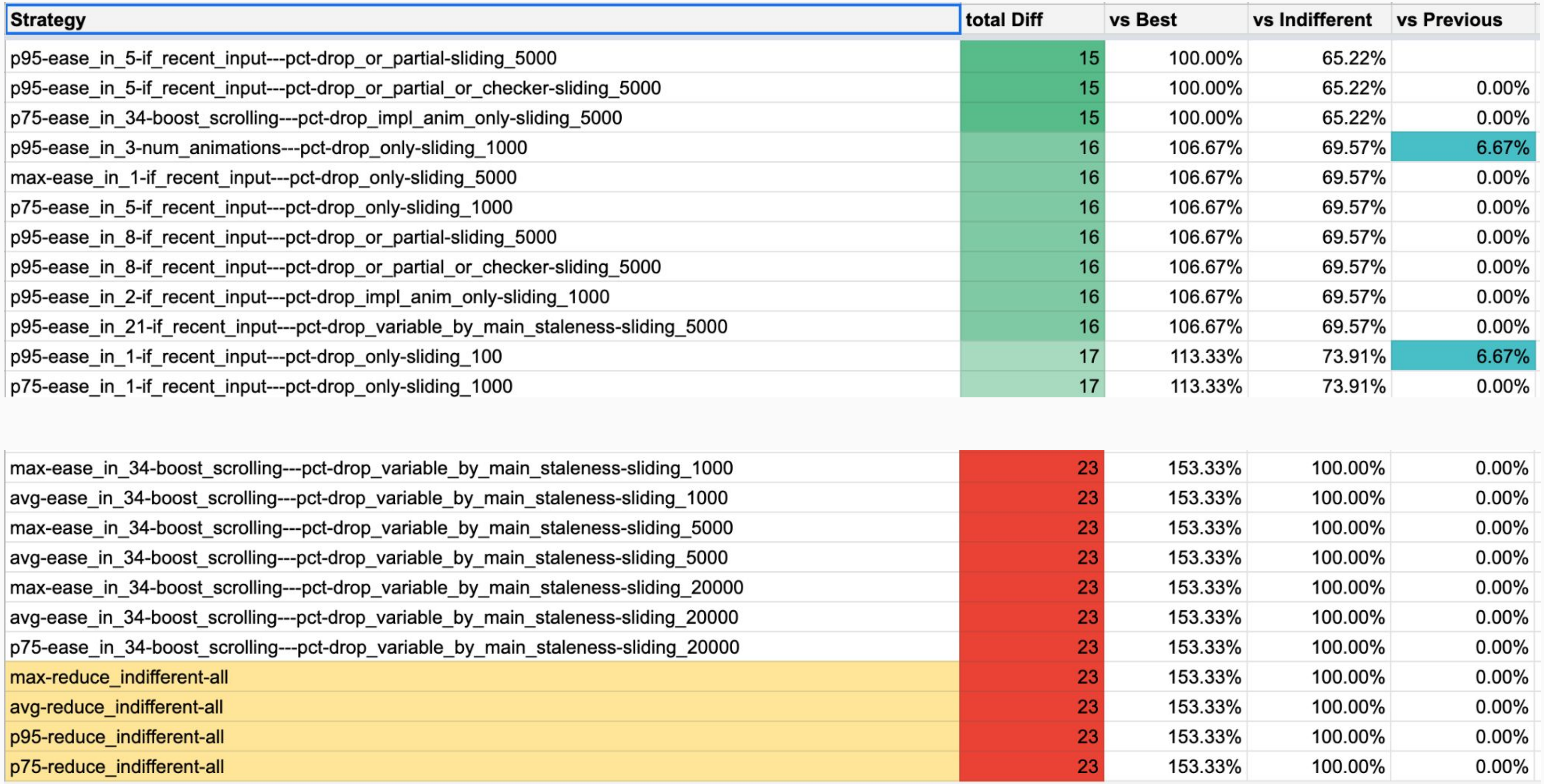

Experimental Smoothness (Split ease_in_1-full SlidingWindowClusterer {"duration":1000})
120
100
80
60
40
20
0
5,000
10,000
15,000
20,000
25,000
30,000
35,000
40,000
45,000
pct-drop_impl_ani…
pct-drop_main_an…
pct-drop_main_an…
pct-drop_checker…


Experimental Smoothness (Split ease_in_1-full SlidingWindowClusterer {"duration":1000})
80
60
40
20
0
10,000
20,000
30,000
40,000
50,000
pct-drop_impl_ani…
pct-drop_main_an…
pct-drop_main_an…
pct-drop_checker…


Experimental Smoothness (Merged ease_in_1-full SlidingWindowClusterer {"duration":1000})
60
50
40
30
20
10
0
10,000
20,000
30,000
40,000
50,000
pct-drop_or_partia…

Correlation study: what do users expect?

- We have many good theories about what *may* make a good metric.
  - Treat every animation equally VS some more important than others
  - Measure all animations always VS scrolling and non-scrolling animations
  - What was the average frame rate for the whole timeline overall VS how bad was the single worst jank?
- Let's just try them all and see what sticks…
- Record a bunch of real sites, with expert labels assigned
- See which metrics match up with expected labels

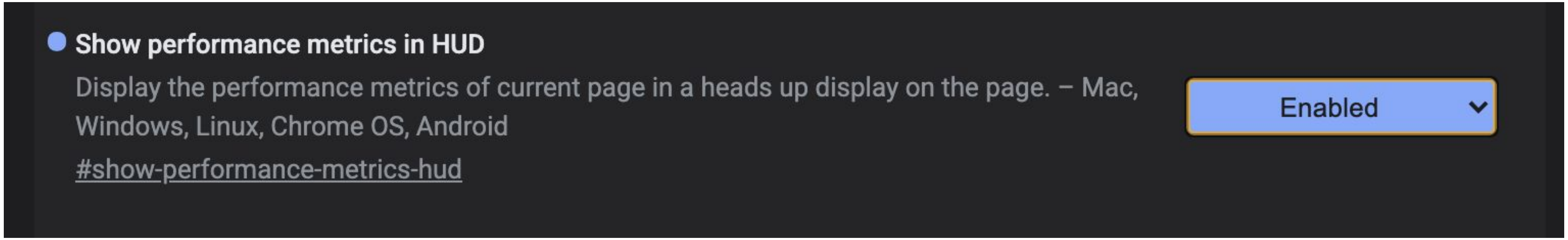


Strategy	total Diff	vs Best	vs Indifferent	vs Previous
p95-ease_in_5-if_recent_input---pct-drop_or_partial-sliding_5000	15	100.00%	65.22%	
p95-ease_in_5-if_recent_input---pct-drop_or_partial_or_checker-sliding_5000	15	100.00%	65.22%	0.00%
p75-ease_in_34-boost_scrolling---pct-drop_impl_anim_only-sliding_5000	15	100.00%	65.22%	0.00%
p95-ease_in_3-num_animations---pct-drop_only-sliding_1000	16	106.67%	69.57%	6.67%
max-ease_in_1-if_recent_input---pct-drop_only-sliding_5000	16	106.67%	69.57%	0.00%
p75-ease_in_5-if_recent_input---pct-drop_only-sliding_1000	16	106.67%	69.57%	0.00%
p95-ease_in_8-if_recent_input---pct-drop_or_partial-sliding_5000	16	106.67%	69.57%	0.00%
p95-ease_in_8-if_recent_input---pct-drop_or_partial_or_checker-sliding_5000	16	106.67%	69.57%	0.00%
p95-ease_in_2-if_recent_input---pct-drop_impl_anim_only-sliding_1000	16	106.67%	69.57%	0.00%
p95-ease_in_21-if_recent_input---pct-drop_variable_by_main_staleness-sliding_5000	16	106.67%	69.57%	0.00%
p95-ease_in_1-if_recent_input---pct-drop_only-sliding_100	17	113.33%	73.91%	6.67%
p75-ease_in_1-if_recent_input---pct-drop_only-sliding_1000	17	113.33%	73.91%	0.00%
max-ease_in_34-boost_scrolling---pct-drop_variable_by_main_staleness-sliding_1000	23	153.33%	100.00%	0.00%
avg-ease_in_34-boost_scrolling---pct-drop_variable_by_main_staleness-sliding_1000	23	153.33%	100.00%	0.00%
max-ease_in_34-boost_scrolling---pct-drop_variable_by_main_staleness-sliding_5000	23	153.33%	100.00%	0.00%
avg-ease_in_34-boost_scrolling---pct-drop_variable_by_main_staleness-sliding_5000	23	153.33%	100.00%	0.00%
max-ease_in_34-boost_scrolling---pct-drop_variable_by_main_staleness-sliding_20000	23	153.33%	100.00%	0.00%
avg-ease_in_34-boost_scrolling---pct-drop_variable_by_main_staleness-sliding_20000	23	153.33%	100.00%	0.00%
p75-ease_in_34-boost_scrolling---pct-drop_variable_by_main_staleness-sliding_20000	23	153.33%	100.00%	0.00%
max-reduce_indifferent-all	23	153.33%	100.00%	0.00%
avg-reduce_indifferent-all	23	153.33%	100.00%	0.00%
p95-reduce_indifferent-all	23	153.33%	100.00%	0.00%
p75-reduce_indifferent-all	23	153.33%	100.00%	0.00%

Playing with the metric

Playing with the metric

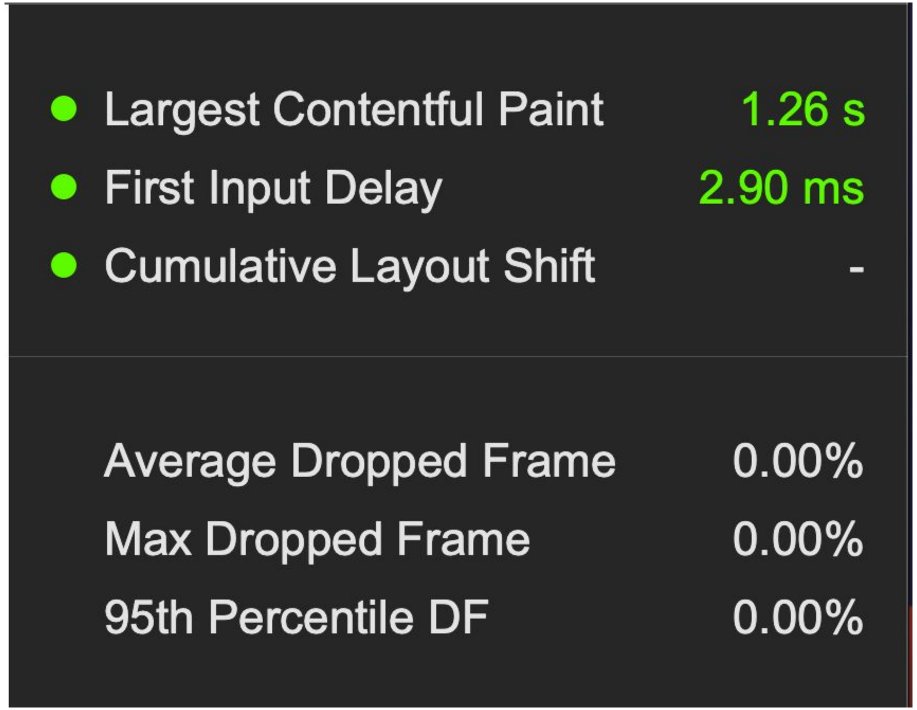
- Existing Chrome:ukm values
  - Graphics.Smoothness.NormalizedPercentDroppedFrames
- Performance HUD
- DevTools
  - Frame Rendering Stats (previously: fps meter)
  - Frame Viewer in Performance panel
- Tracing (ui.perfetto.dev, chrome:tracing)



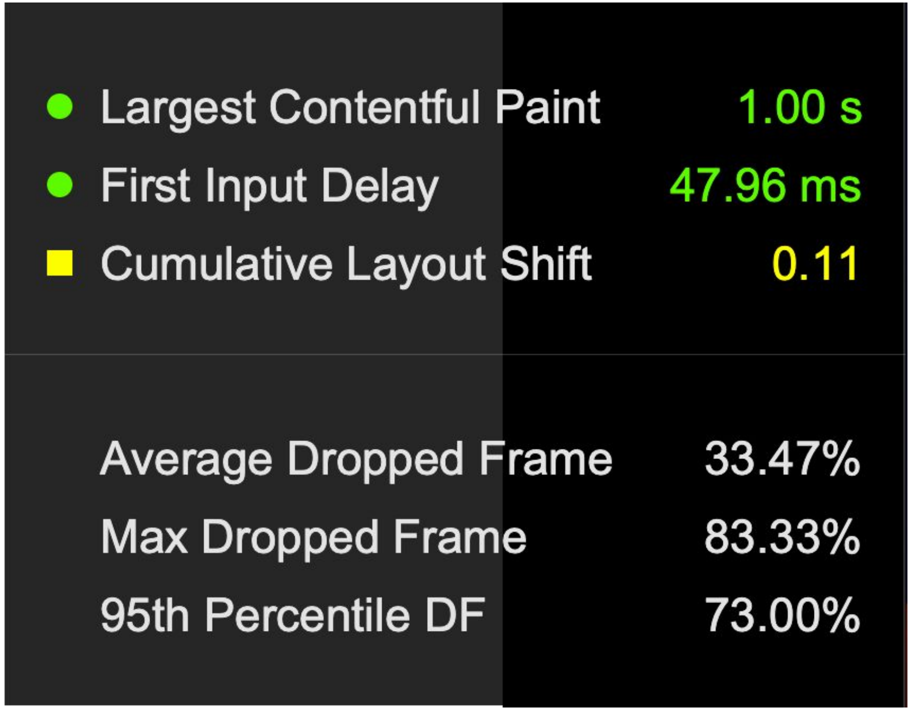
● **Show performance metrics in HUD**

Display the performance metrics of current page in a heads up display on the page. – Mac, Windows, Linux, Chrome OS, Android

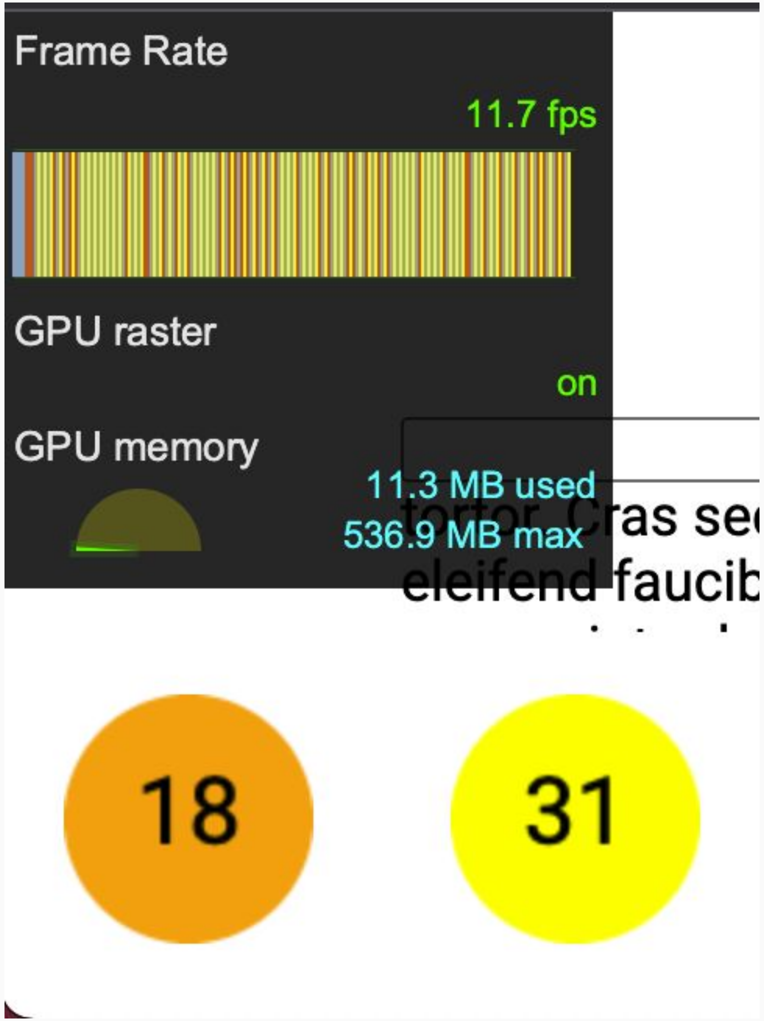
#show-performance-metrics-hud



●	Largest Contentful Paint	1.26 s
●	First Input Delay	2.90 ms
●	Cumulative Layout Shift	-
	Average Dropped Frame	0.00%
	Max Dropped Frame	0.00%
	95th Percentile DF	0.00%



●	Largest Contentful Paint	1.00 s
●	First Input Delay	47.96 ms
■	Cumulative Layout Shift	0.11
	Average Dropped Frame	33.47%
	Max Dropped Frame	83.33%
	95th Percentile DF	73.00%



Frame Rate

11.7 fps

GPU raster

on

GPU memory

11.3 MB used
536.9 MB max

Lab Tooling: how does it work?

Types of Frame Updates, in theory

- **No Update Desired** (idle time, repeat previous frame)
- **Fully presented**, main thread update is either committed within deadline, or no main update was desired
- **Partially presented** (compositor only), but the delayed main update had **no visual change**
- **Partially presented** (compositor only), and the main update had a visual update, but that update **did not include an animation which *affects smoothness***
- **Partially presented** (compositor only), and the main update had a visual update, and that update does include an animation which *affects smoothness*, but we happened to have a **new main update from a previous frame** (i.e. a previously stale main update has arrived)
- **Partially presented** (compositor only), without the desired main update, and the compositor update **has an animation which *affects smoothness***
- **Partially presented** (compositor only) but the compositor update **does not have an animation which *affects smoothness***
- **Dropped frame** (no update), there was no compositor update desired, and main was delayed
- **Dropped frame**, we desired a compositor update, but it was delayed
- **Stale frame**, we desired an update, which was produced by the renderer, but the GPU still did not present it before vsync deadline

Normalized Dropped Frame UKM (and Perf HUD)

- **No Update Desired** (idle time, repeat previous frame)
- **Fully presented** (all updates from all threads)
- **Partially presented** (compositor only), but the delayed main update had **no visual change**
- **Partially presented** (compositor only), and the main update had a visual update, but that update **did not include an animation which *affects smoothness***
- **Partially presented** (compositor only), and the main update had a visual update, and that update does include an animation which *affects smoothness*, but we happened to have a **new main update from a previous frame** (i.e. a previously stale main update has arrived)
- **Partially presented** (compositor only), without the desired main update, and the compositor update **has an animation which *affects smoothness***
- **Partially presented** (compositor only) but the compositor update **does not have an animation which *affects smoothness***
- **Dropped frame** (no update), there was no compositor update desired, and main was delayed
- **Dropped frame**, we desired a compositor update, but it was delayed
- **Stale frame**, we desired an update, which was produced by the renderer, but the GPU still did not present it before vsync deadline

Frame Rendering Stats (Live Viewer)

- **No Update Desired** (idle time, repeat previous frame)
- **Fully presented** (all updates from all threads)
- **Partially presented** (compositor only), but the delayed main update had **no visual change**
- **Partially presented** (compositor only), and the main update had a visual update, but that update **did not include an animation which *affects smoothness***
- **Partially presented** (compositor only), and the main update had a visual update, and that update does include an animation which *affects smoothness*, but we happened to have a **new main update from a previous frame** (i.e. a previously stale main update has arrived)
- **Partially presented** (compositor only), without the desired main update, and the compositor update **has an animation which *affects smoothness***
- **Partially presented** (compositor only) but the compositor update **does not have an animation which *affects smoothness***
- **Dropped frame** (no update), there was no compositor update desired, and main was delayed
- **Dropped frame**, we desired a compositor update, but it was delayed
- **Stale frame**, we desired an update, which was produced by the renderer, but the GPU still did not present it before vsync deadline

Frame Rendering Stats (3 seconds rolling FPS)

- **No Update Desired** (idle time, repeat previous frame)
- **Fully presented** (all updates from all threads)
- **Partially presented** (compositor only), but the delayed main update had **no visual change**
- **Partially presented** (compositor only), and the main update had a visual update, but that update **did not include an animation which *affects smoothness***
- **Partially presented** (compositor only), and the main update had a visual update, and that update does include an animation which *affects smoothness*, but we happened to have a **new main update from a previous frame** (i.e. a previously stale main update has arrived)
- **Partially presented** (compositor only), without the desired main update, and the compositor update **has an animation which *affects smoothness***
- **Partially presented** (compositor only) but the compositor update **does not have an animation which *affects smoothness***
- **Dropped frame** (no update), there was no compositor update desired, and main was delayed
- **Dropped frame**, we desired a compositor update, but it was delayed
- **Stale frame**, we desired an update, which was produced by the renderer, but the GPU still did not present it before vsync deadline

Frame Viewer in DevTools (Today)

- **No Update Desired** (idle time, repeat previous frame)
- **Fully presented** (all updates from all threads)
- **Partially presented** (compositor only), but the delayed main update had **no visual change**
- **Partially presented** (compositor only), and the main update had a visual update, but that update **did not include an animation which *affects smoothness***
- **Partially presented** (compositor only), and the main update had a visual update, and that update does include an animation which *affects smoothness*, but we happened to have a **new main update from a previous frame** (i.e. a previously stale main update has arrived)
- **Partially presented** (compositor only), without the desired main update, and the compositor update **has an animation which *affects smoothness***
- **Partially presented** (compositor only) but the compositor update **does not have an animation which *affects smoothness***
- **Dropped frame** (no update), there was no compositor update desired, and main was delayed
- **Dropped frame**, we desired a compositor update, but it was delayed
- **Stale frame**, we desired an update, which was produced by the renderer, but the GPU still did not present it before vsync deadline

Frame Viewer in DevTools (Soon?)

- **No Update Desired** (idle time, repeat previous frame)
- **Fully presented** (all updates from all threads)
- **Partially presented** (compositor only), but the delayed main update had **no visual change**
- **Partially presented** (compositor only), and the main update had a visual update, but that update **did not include an animation which *affects smoothness***
- **Partially presented** (compositor only), and the main update had a visual update, and that update does include an animation which *affects smoothness*, but we happened to have a **new main update from a previous frame** (i.e. a previously stale main update has arrived)
- **Partially presented** (compositor only), without the desired main update, and the compositor update **has an animation which *affects smoothness***
- **Partially presented** (compositor only) but the compositor update **does not have an animation which *affects smoothness***
- **Dropped frame** (no update), there was no compositor update desired, and main was delayed
- **Dropped frame**, we desired a compositor update, but it was delayed
- **Stale frame**, we desired an update, which was produced by the renderer, but the GPU still did not present it before vsync deadline

FPS in DevTools (based on interval of each frame update)

- **No Update Desired** (idle time, repeat previous frame)
- **Fully presented** (all updates from all threads)
- **Partially presented** (compositor only), but the delayed main update had **no visual change**
- **Partially presented** (compositor only), and the main update had a visual update, but that update **did not include an animation which *affects smoothness***
- **Partially presented** (compositor only), and the main update had a visual update, and that update does include an animation which *affects smoothness*, but we happened to have a **new main update from a previous frame** (i.e. a previously stale main update has arrived)
- **Partially presented** (compositor only), without the desired main update, and the compositor update **has an animation which *affects smoothness***
- **Partially presented** (compositor only) but the compositor update **does not have an animation which *affects smoothness***
- **Dropped frame** (no update), there was no compositor update desired, and main was delayed
- **Dropped frame**, we desired a compositor update, but it was delayed
- **Stale frame**, we desired an update, which was produced by the renderer, but the GPU still did not present it before vsync deadline

Some fun demos

Some fun demos

- https://vsynctester.com/
- https://mmocny.github.io/canvas-worker-raf-fps-meter/
- https://propjockey.github.io/DOMinion-build-demo/index.html