Optimizing Interaction to Next Paint

Jeremy Wagner — jlwagner.net — @malchata — Armada JS

What is responsiveness?

gShoe product Q&A:

- What is gShoe?
- What technology does gShoe use?
- How much does gShoe cost?

Poor responsiveness

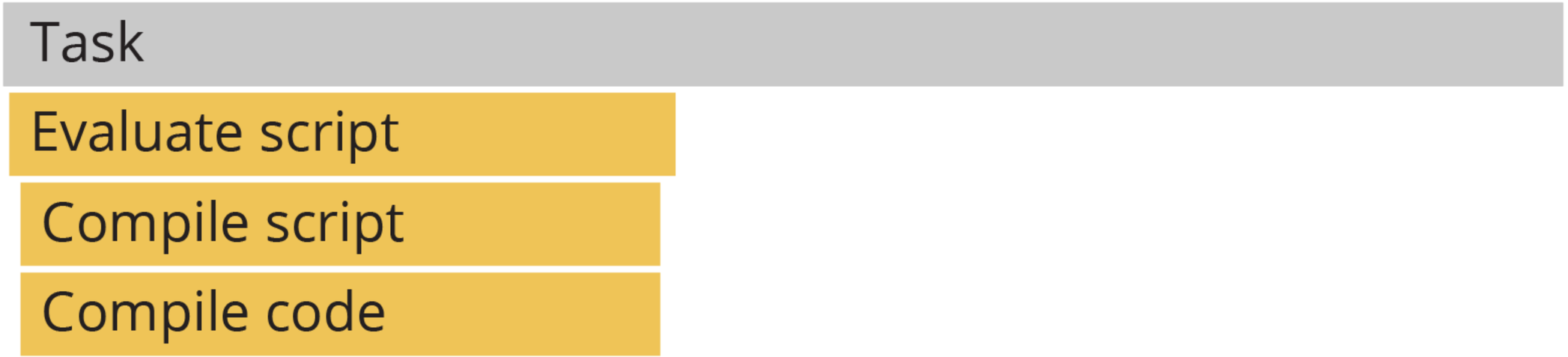
gShoe product Q&A:

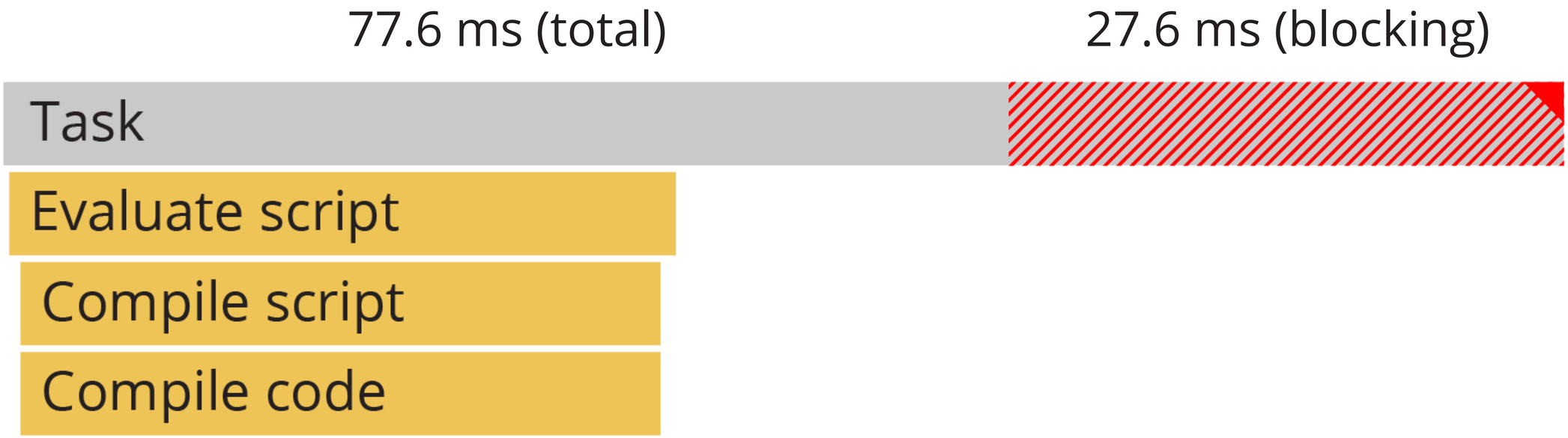
- What is gShoe?
- What technology does gShoe use?
- How much does gShoe cost?

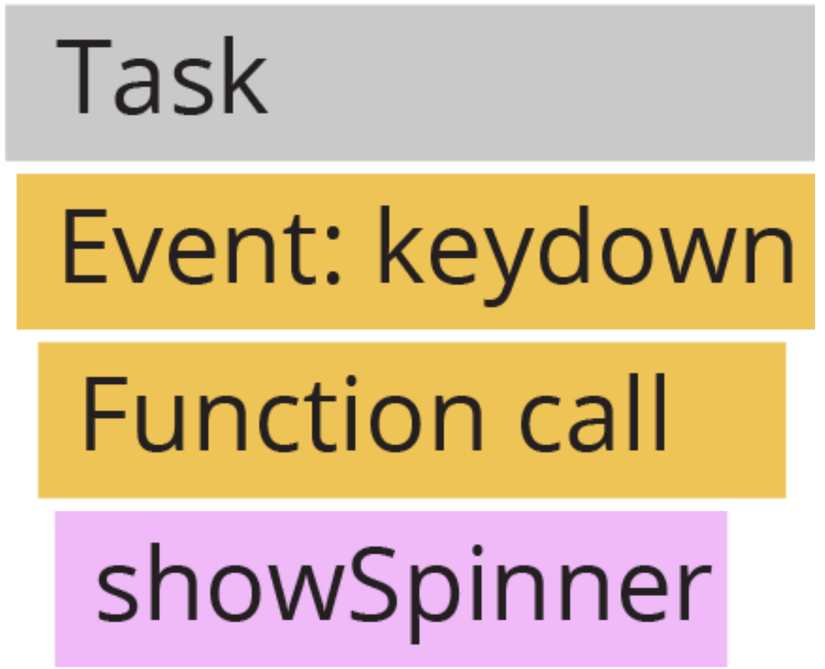
Good responsiveness

What causes poor responsiveness?

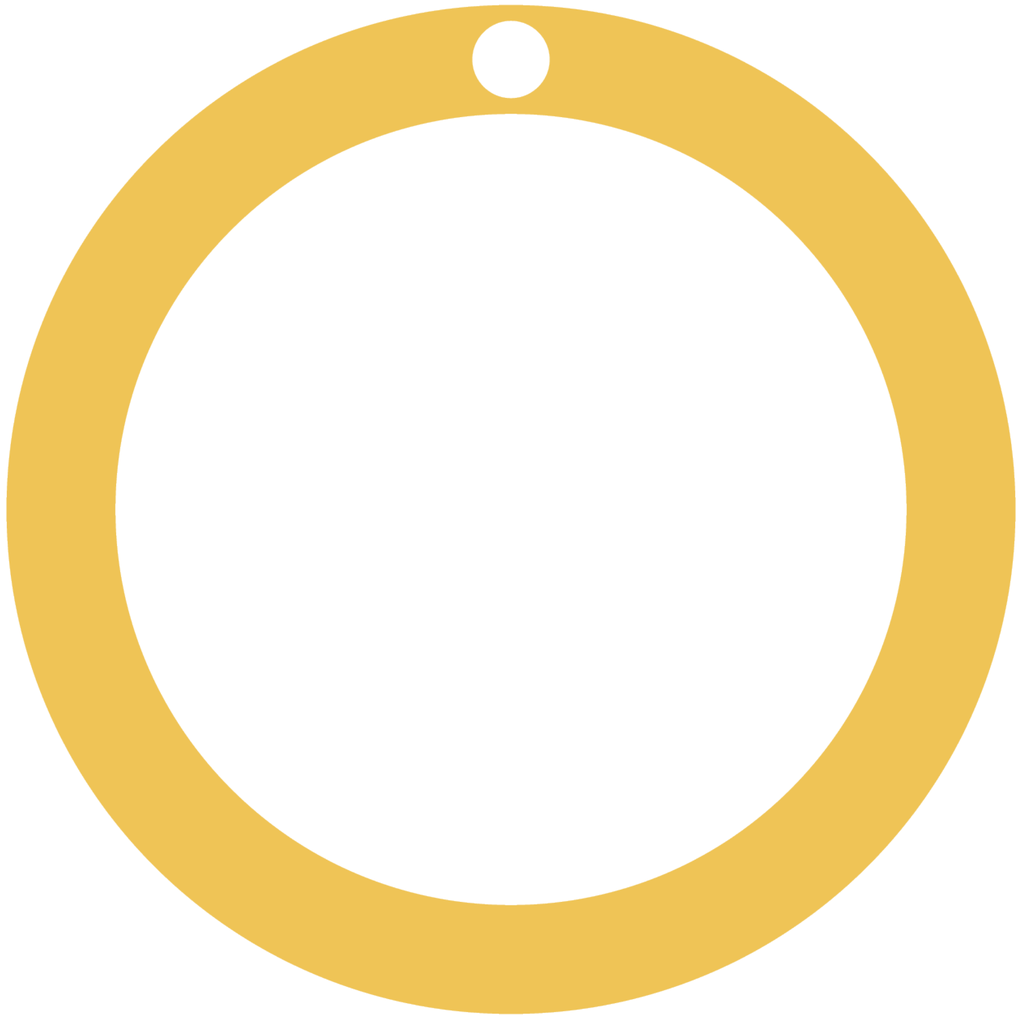

Main thread
Task
Evaluate script
Compile script
Compile code
Task
Event: keyup
Function call
fetchWrapper
sendFetch

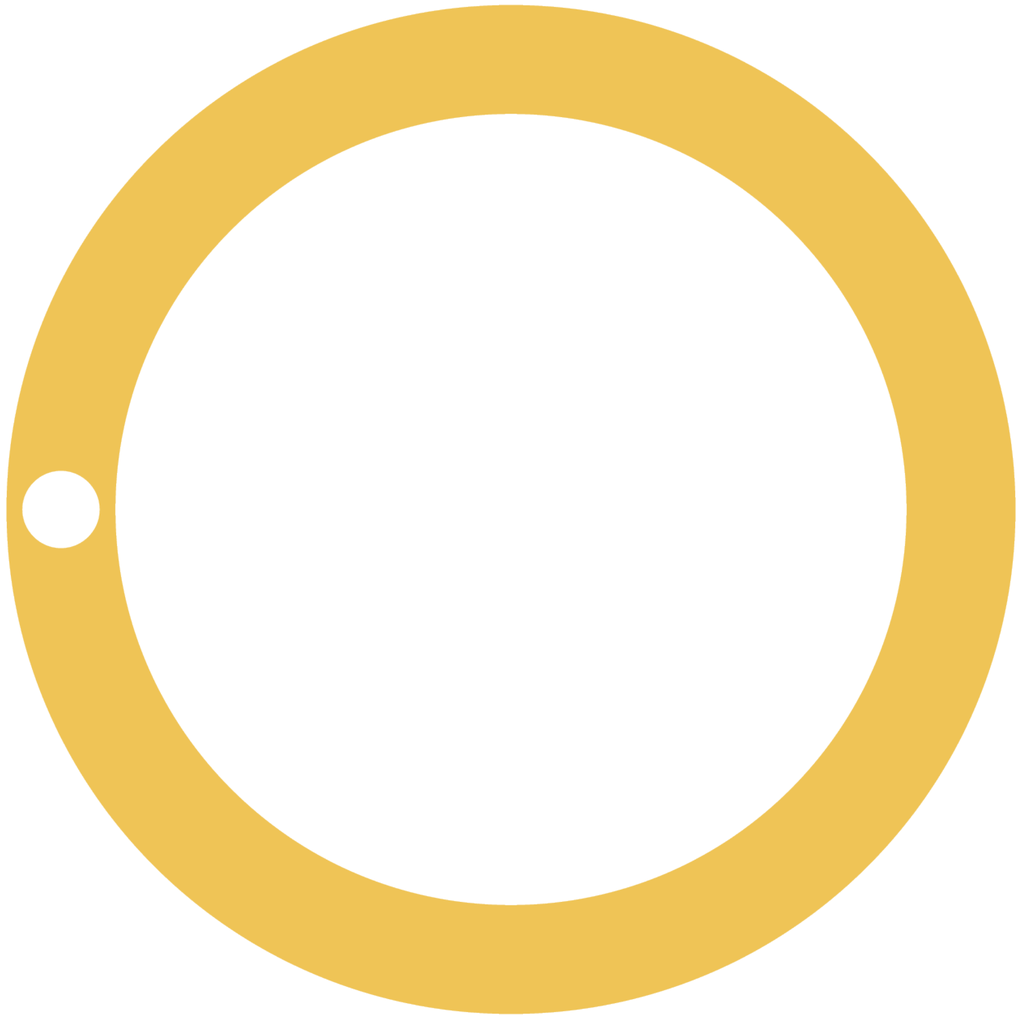

Task
Evaluate script
Compile script
Compile code


77.6 ms (total)
27.6 ms (blocking)
Task
Evaluate script
Compile script
Compile code

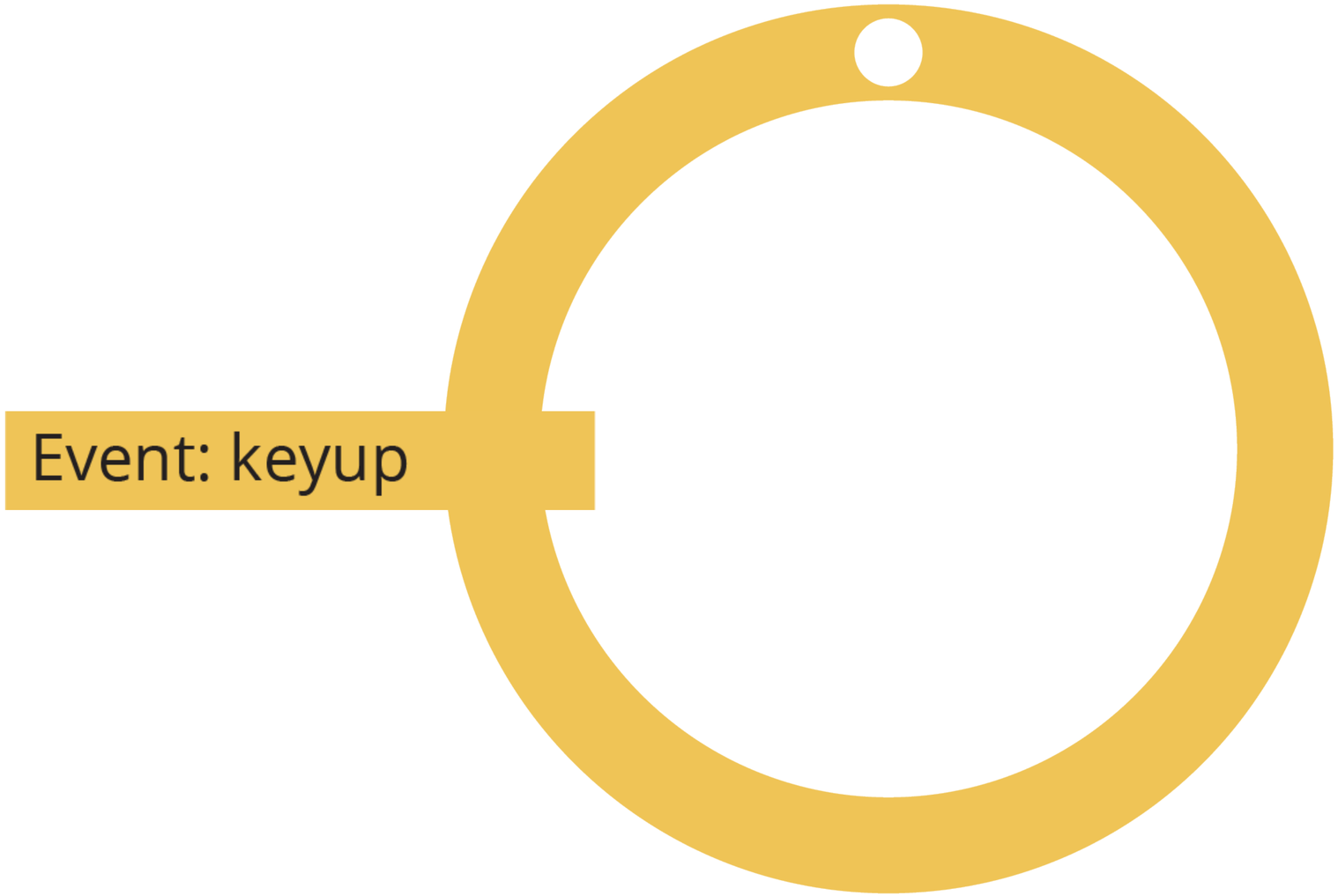

Task
Event: keydown
Function call
showSpinner

Why so much focus on tasks?



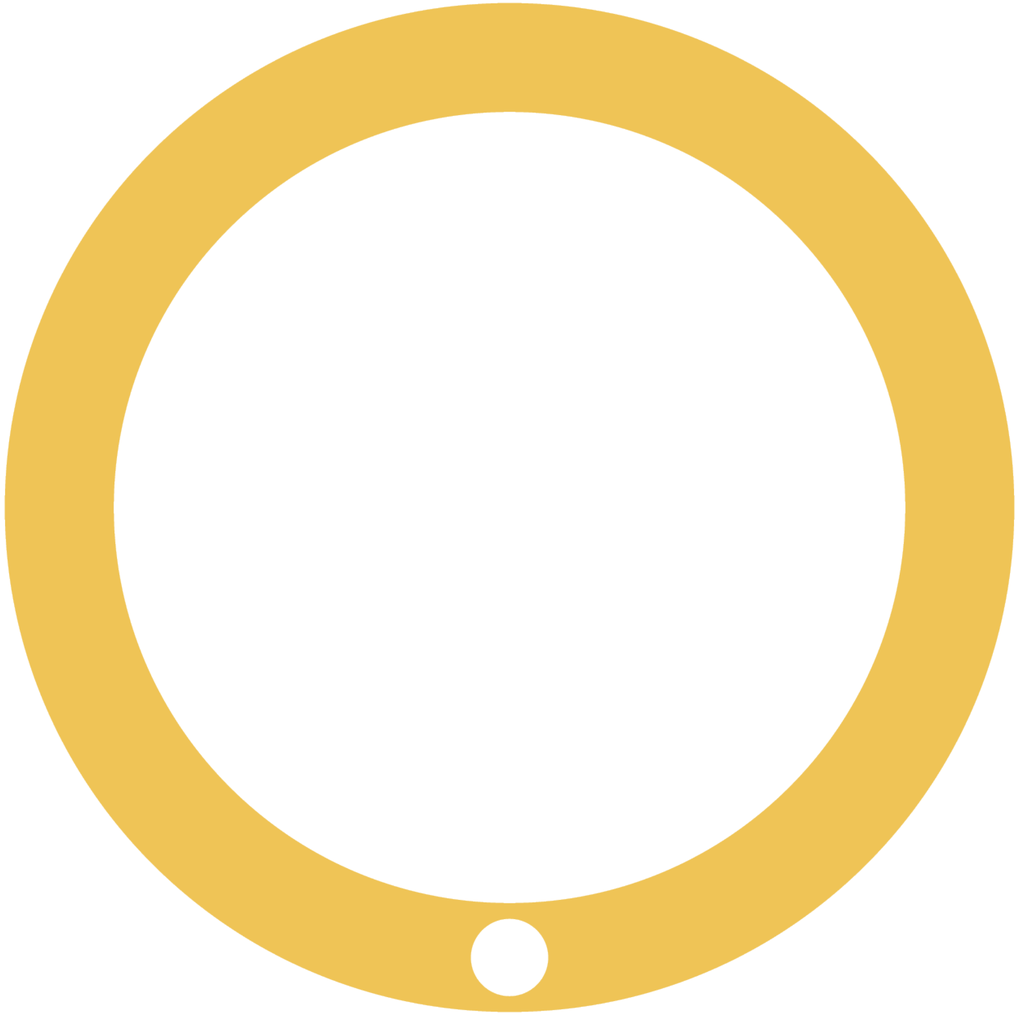



4GIFs.com

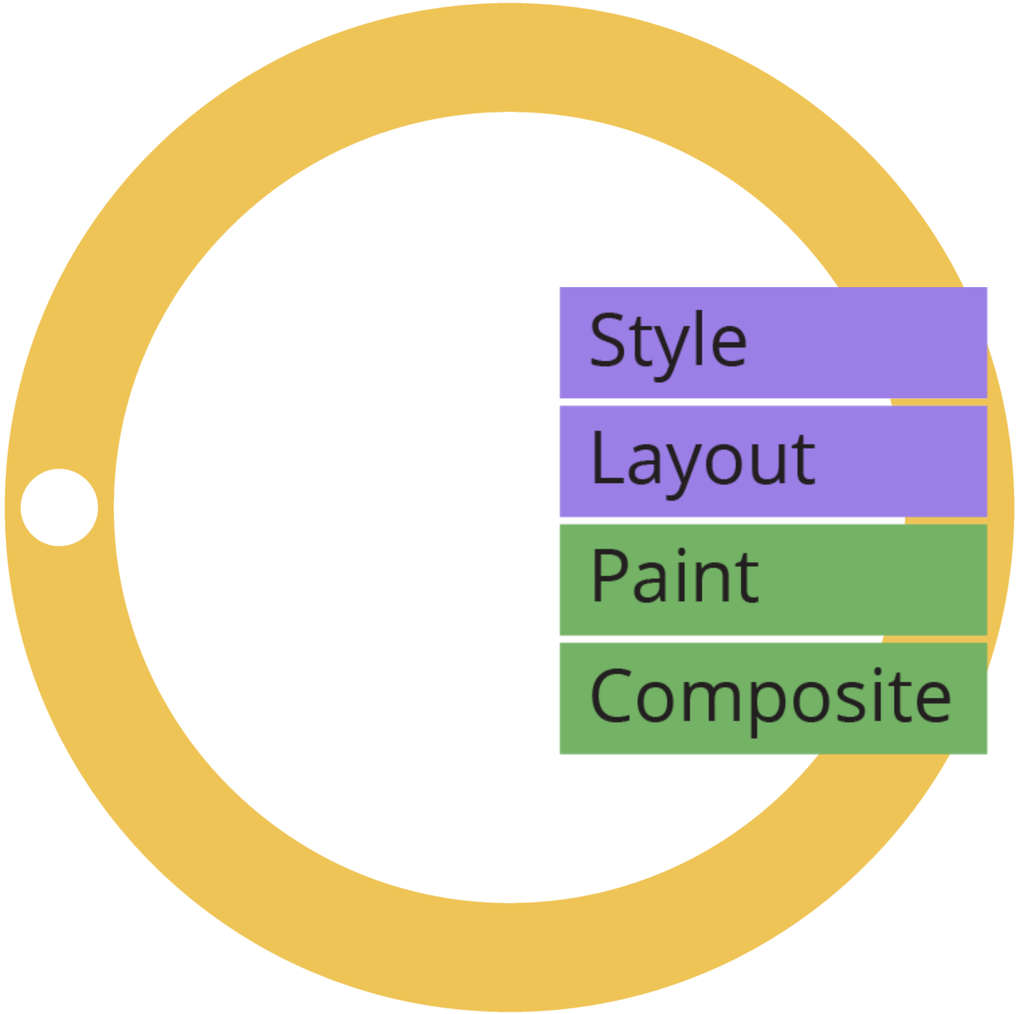

Event: keyup

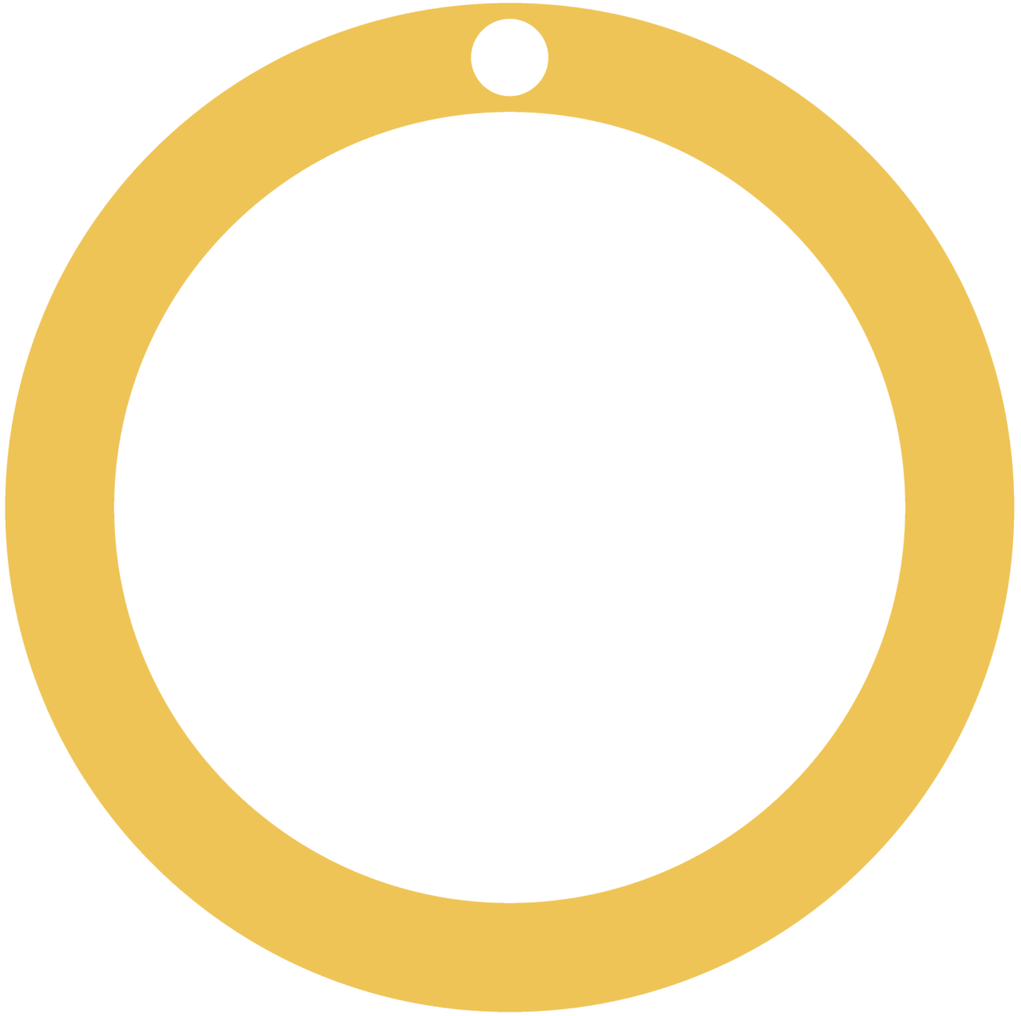
Event: keyup

Event: keyup



Event: keyup





Style

Layout

Paint

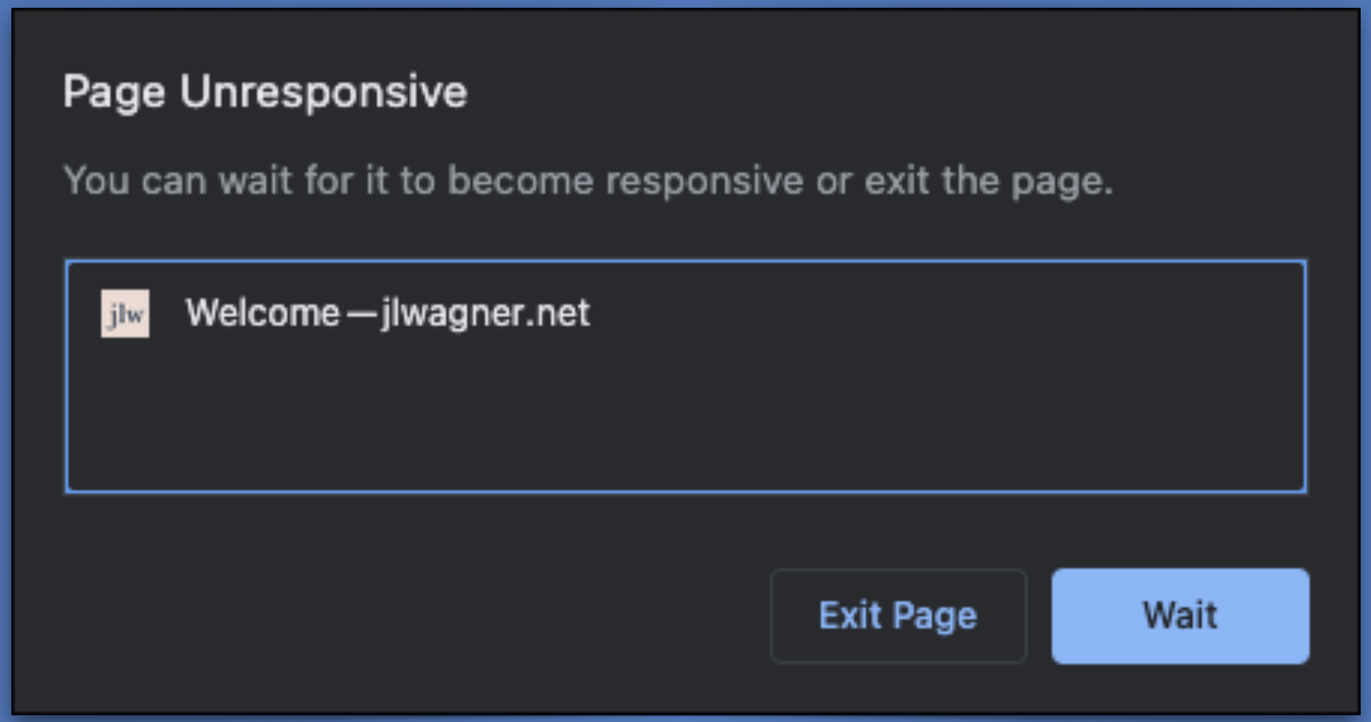
Composite

Style

Layout

Paint

Composite


Page Unresponsive
You can wait for it to become responsive or exit the page.
jlw
Welcome — jlwagner.net
Exit Page
Wait

Interaction to Next Paint (INP)

May 6, 2022 — Updated Jul 18, 2022

Appears in: Metrics

Jeremy Wagner

Twitter GitHub Homepage

On this page

web.dev/inp

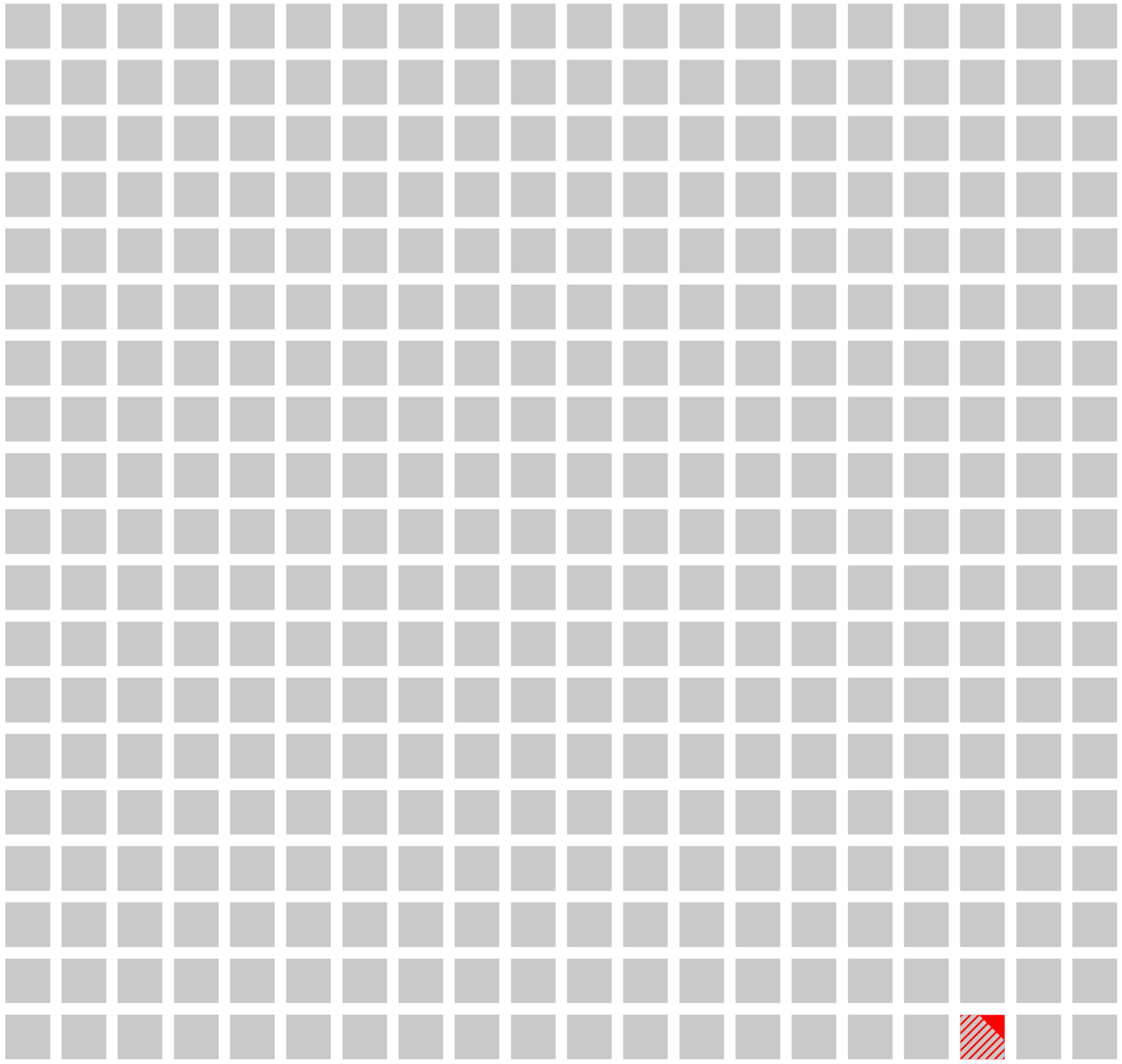
- Observes certain interactions using the Event Timing API.
- Unlike First Input Delay (FID), measures more than just the input delay of the first interaction.
- Chooses the single worst* interaction latency (in milliseconds), which is representative of the page's overall responsiveness.

INP is all about the next frame.

- Keyboard interactions (including physical and onscreen).
- Mouse clicks.
- Taps on touchscreens.

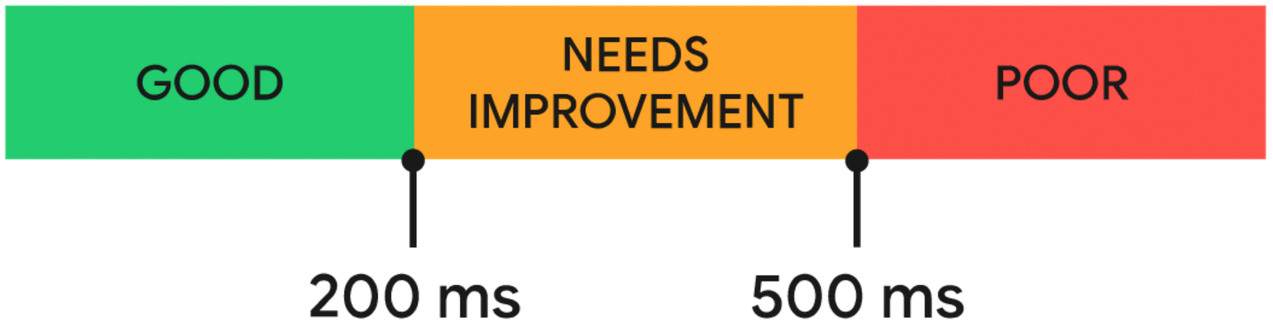
- Hovering.
- Network activity.
- Scrolling*.

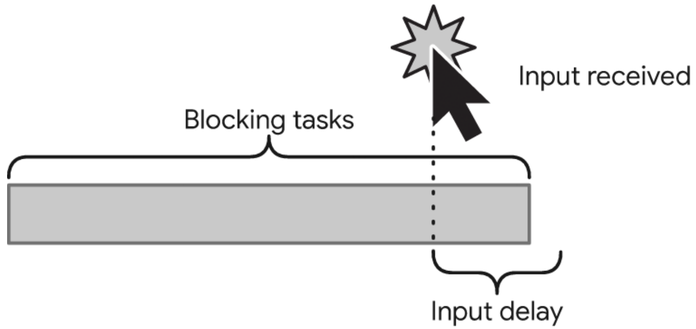

One interaction
INP

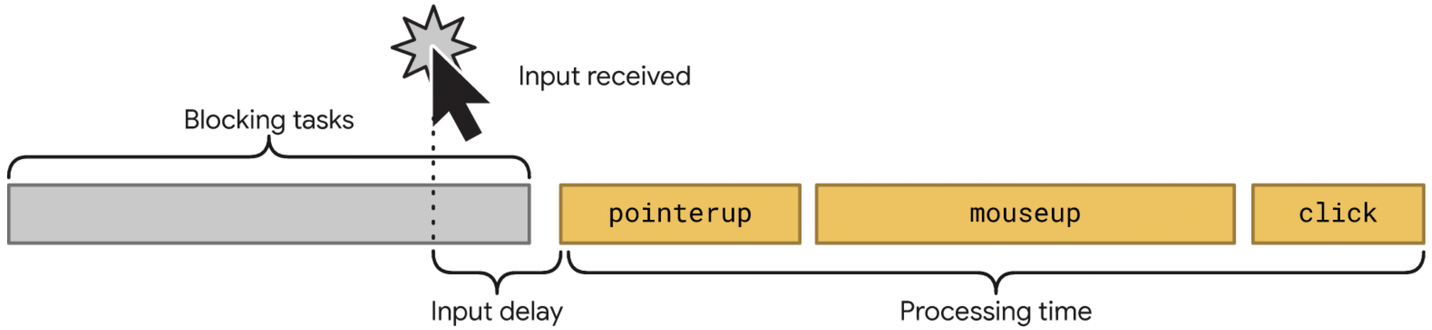


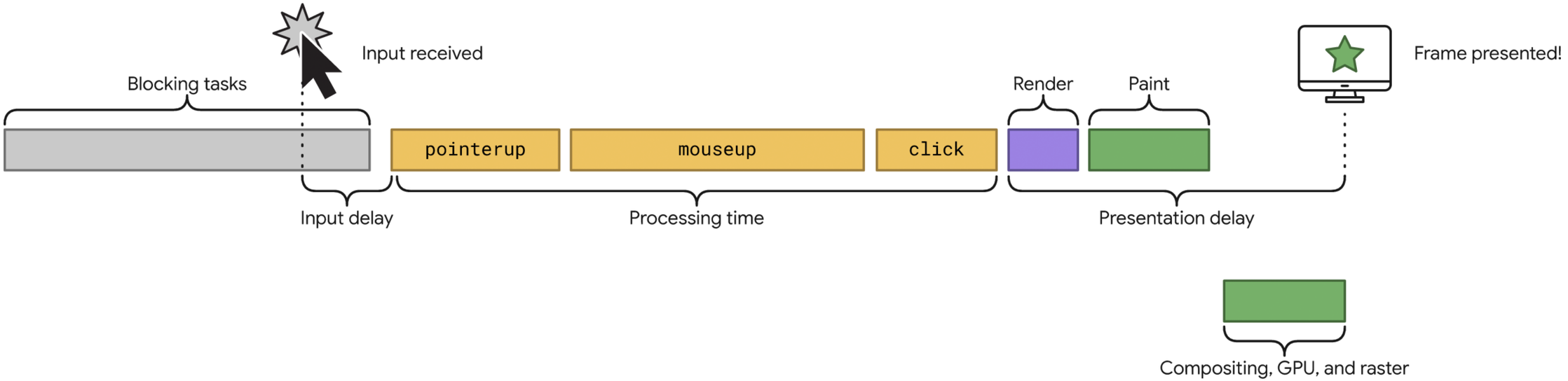
INP

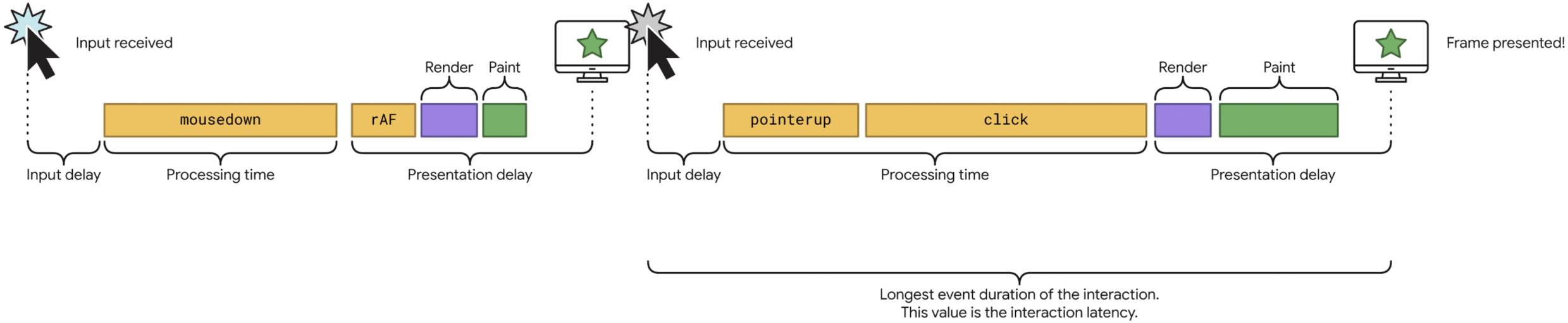
Interaction to Next Paint




Input received
Blocking tasks
Input delay


Input received
Blocking tasks
pointerup
mouseup
click
Input delay
Processing time


Input received
Blocking tasks
Render
Paint
Frame presented!
pointerup
mouseup
click
Input delay
Processing time
Presentation delay
Compositing, GPU, and raster

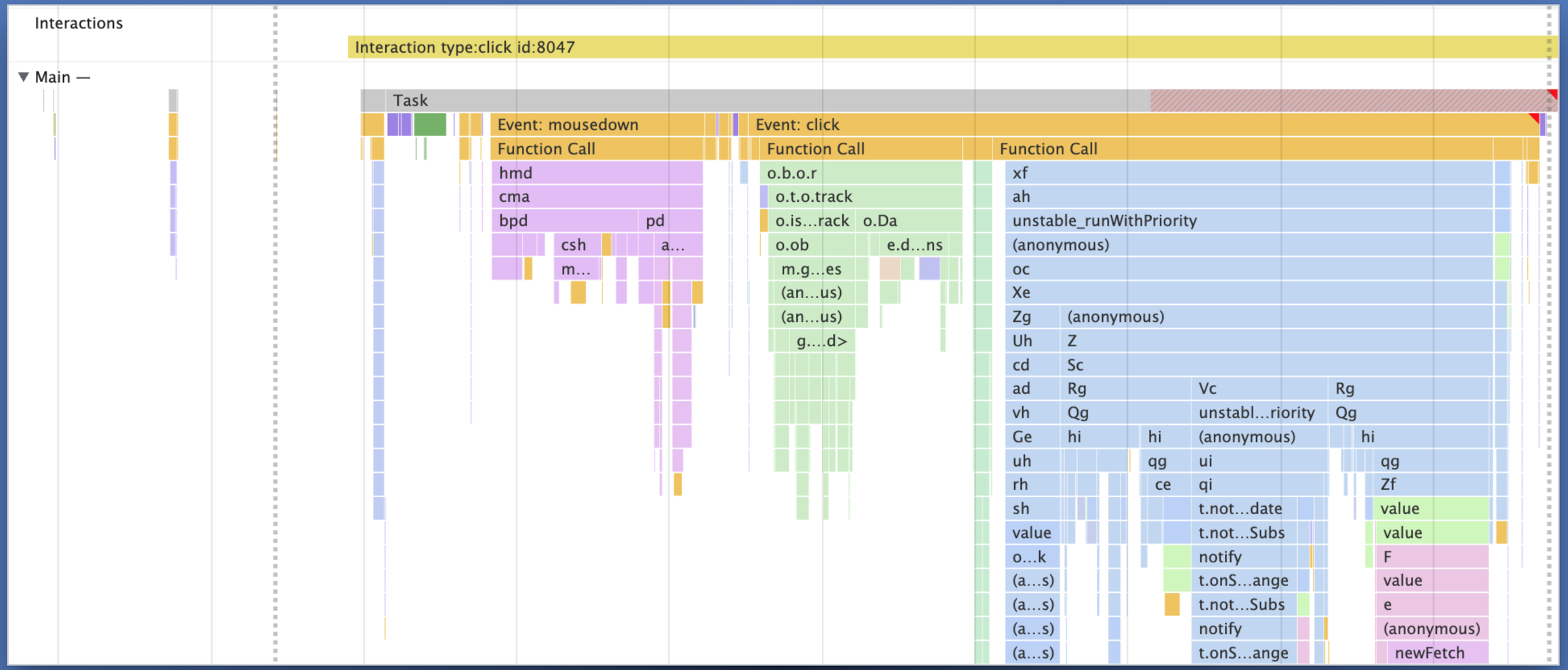

Input received
mousedown
rAF
Render
Paint
Input delay
Processing time
Presentation delay
Input received
pointerup
click
Render
Paint
Frame presented!
Input delay
Processing time
Presentation delay
Longest event duration of the interaction.
This value is the interaction latency.

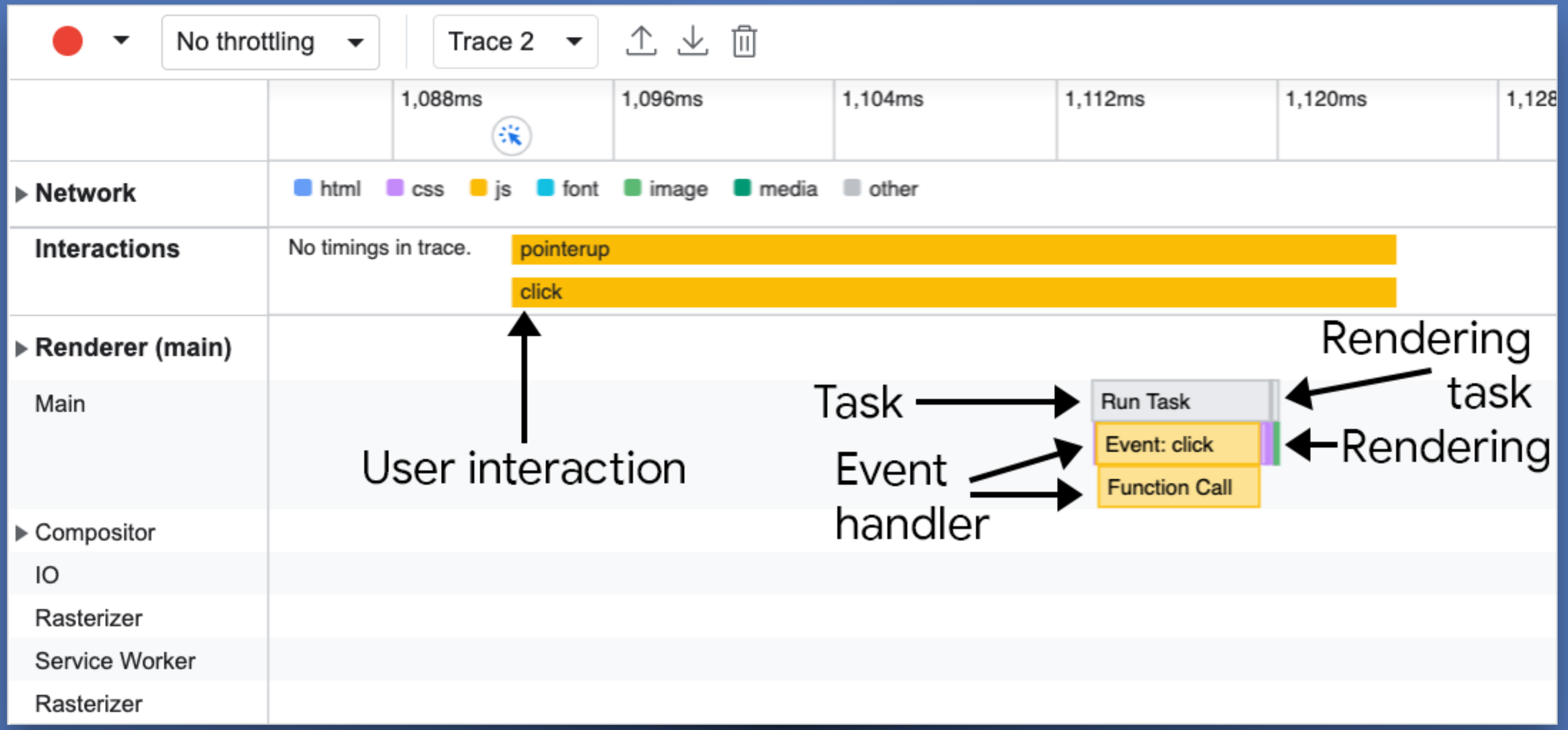
Optimizing INP

```
// Vendors
import { onINP } from "web-vitals";

// https://gist.github.com/malchata/56e92306acb0c5b9fd3541ed16ce2d23
import { getSelector } from "./utils.js";

function sendMetrics({ metric, value, entries }) {
  fetch("https://mycoolwebsite.net/collect", {
    body: JSON.stringify({
      metric: value,
      element: getSelector(entries[0].target)
    }),
    method: "POST",
    keepalive: true,
    headers: {
      "Content-Type": "application/json",
    }
  });
}

onINP(sendMetrics);
```

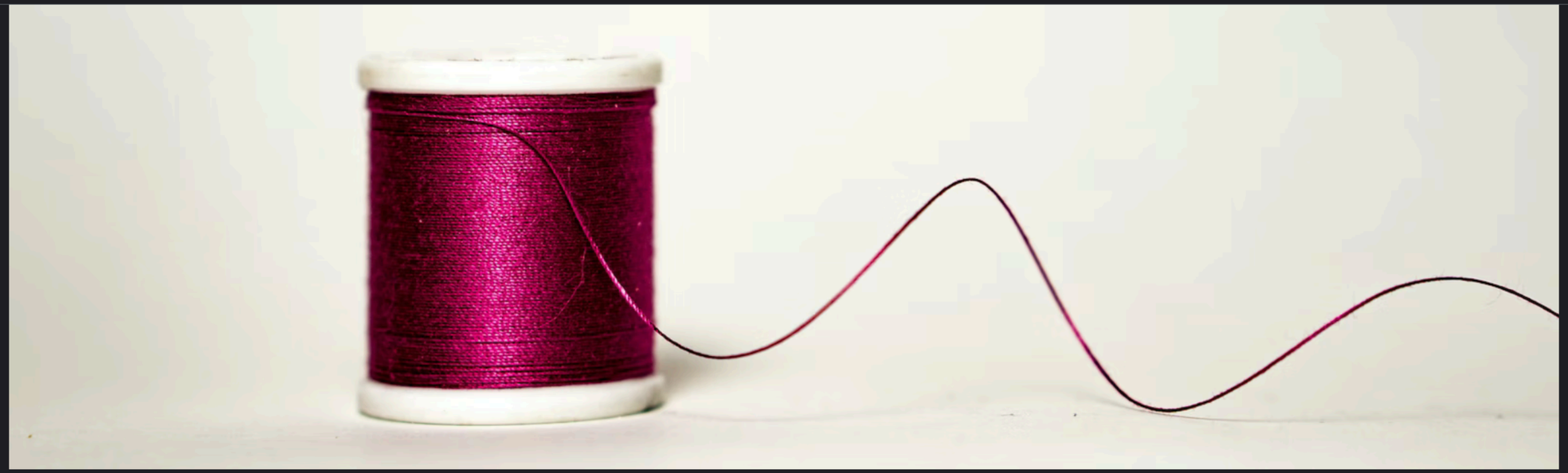
github.com/GoogleChrome/web-vitals

Lab tools


Interactions
Interaction type:click id:8047
Main —
Task
Event: mousedown
Event: click
Function Call
Function Call
Function Call
hmd
cma
bpd
pd
csh
a…
m…
o.b.o.r
o.t.o.track
o.is…rack
o.Da
o.ob
e.d…ns
m.g…es
(an…us)
(an…us)
g….d>
xf
ah
unstable_runWithPriority
(anonymous)
oc
Xe
Zg
(anonymous)
Uh
Z
cd
Sc
ad
Rg
Vc
Rg
vh
Qg
unstabl…riority
Qg
Ge
hi
hi
(anonymous)
hi
uh
qg
ui
qg
rh
ce
qi
Zf
sh
t.not…date
value
value
t.not…Subs
value
o…k
notify
F
(a…s)
t.onS…ange
value
(a…s)
t.not…Subs
e
(a…s)
notify
(anonymous)
(a…s)
t.onS…ange
newFetch


No throttling
Trace 2
1,088ms
1,096ms
1,104ms
1,112ms
1,120ms
1,128
Network
html
css
js
font
image
media
other
Interactions
No timings in trace.
pointerup
click
Renderer (main)
Main
Compositor
IO
Rasterizer
Service Worker
Rasterizer
User interaction
Task
Run Task
Event: click
Function Call
Event handler
Rendering task
Rendering

Optimize!



Optimize long tasks

You've been told: "don't block the main thread" and "break up your long tasks", but what does it mean to do those things? Find out in this article.

Sep 30, 2022

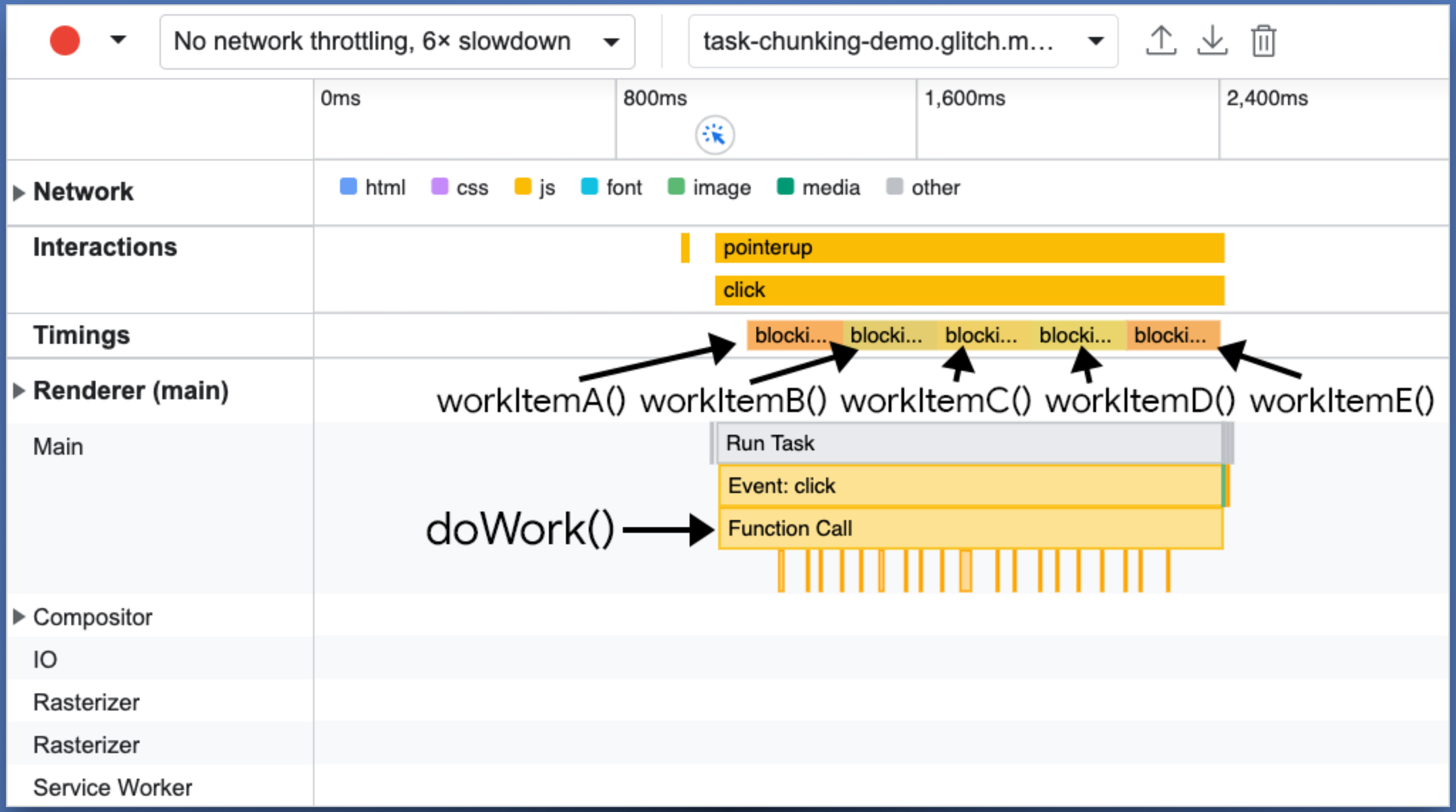
Jeremy Wagner

Twitter GitHub Homepage

web.dev/optimize-long-tasks

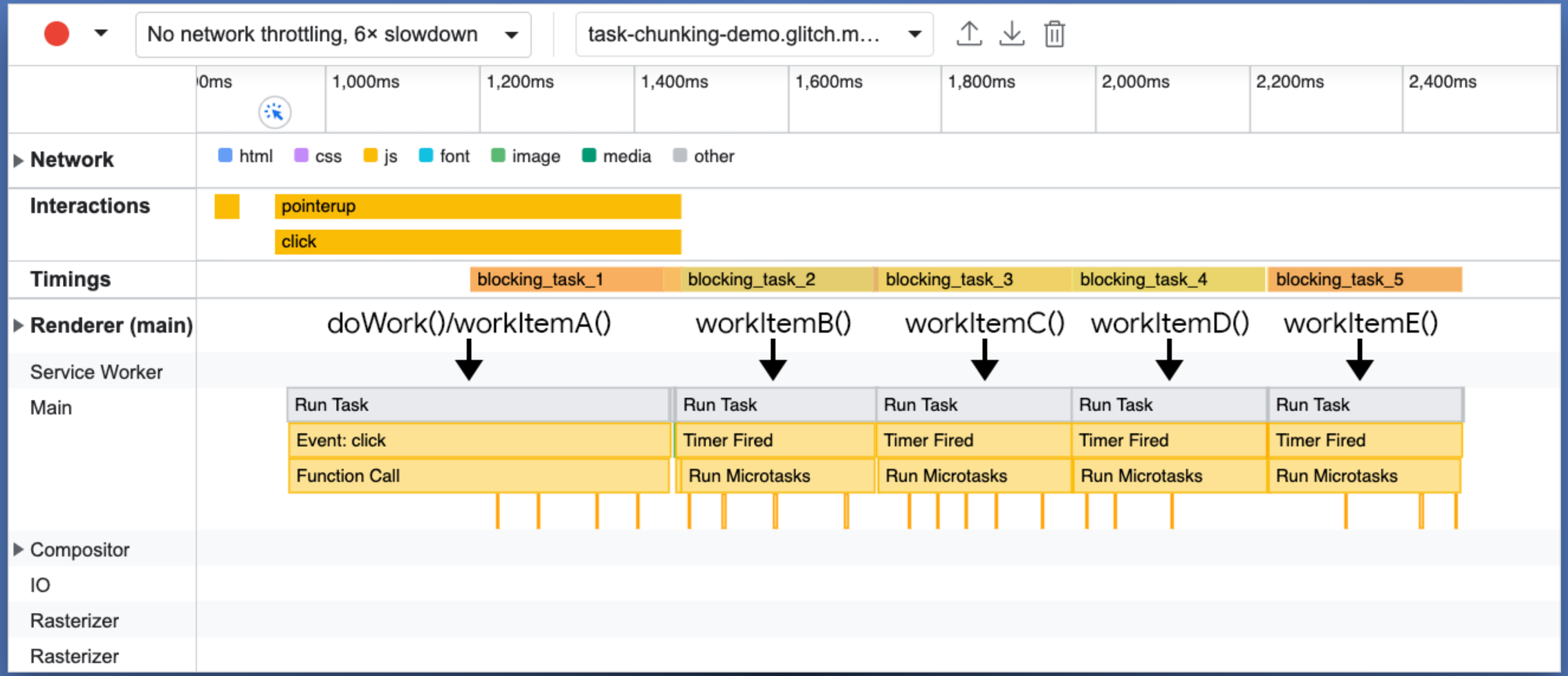
If you read lots of stuff about web performance, the advice for keeping your JavaScript

```
function doWork () {
  workItemA();
  workItemB();
  workItemC();
  workItemD();
  workItemE();
}
```


No network throttling, 6× slowdown
task-chunking-demo.glitch.m…
0ms
800ms
1,600ms
2,400ms
Network
html
css
js
font
image
media
other
Interactions
pointerup
click
Timings
blocki...
blocki...
blocki...
blocki...
blocki...
Renderer (main)
workItemA()
workItemB()
workItemC()
workItemD()
workItemE()
Main
Run Task
Event: click
doWork()
Function Call
Compositor
IO
Rasterizer
Rasterizer
Service Worker

```
function yield () {
  return new Promise(resolve => {
    resolve(setTimeout, 0);
  });
}
```

```
async function doWork () {
  workItemA();
  await yield();
  workItemB();
  await yield();
  workItemC();
  await yield();
  workItemD();
  await yield();
  workItemE();
}
```

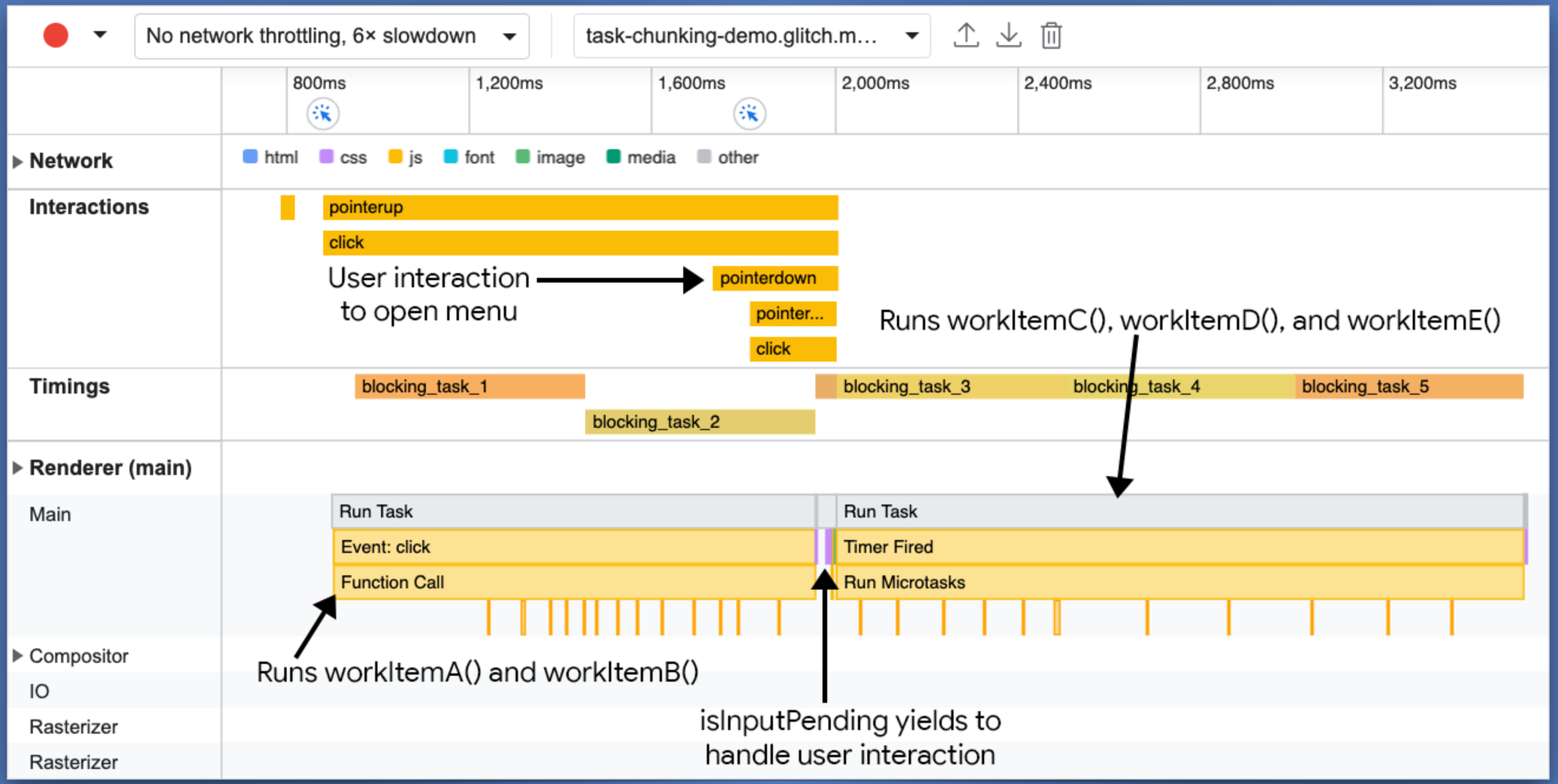

No network throttling, 6× slowdown
task-chunking-demo.glitch.m…
0ms
1,000ms
1,200ms
1,400ms
1,600ms
1,800ms
2,000ms
2,200ms
2,400ms
Network
html
css
js
font
image
media
other
Interactions
pointerup
click
Timings
blocking_task_1
blocking_task_2
blocking_task_3
blocking_task_4
blocking_task_5
Renderer (main)
doWork()/workItemA()
workItemB()
workItemC()
workItemD()
workItemE()
Service Worker
Main
Run Task
Event: click
Function Call
Timer Fired
Run Microtasks
Compositor
IO
Rasterizer
Rasterizer

```
async function doWork () {
  workItemA();
  await scheduler.yield();
  workItemB();
  await scheduler.yield();
  workItemC();
  await scheduler.yield();
  workItemD();
  await scheduler.yield();
  workItemE();
}
```

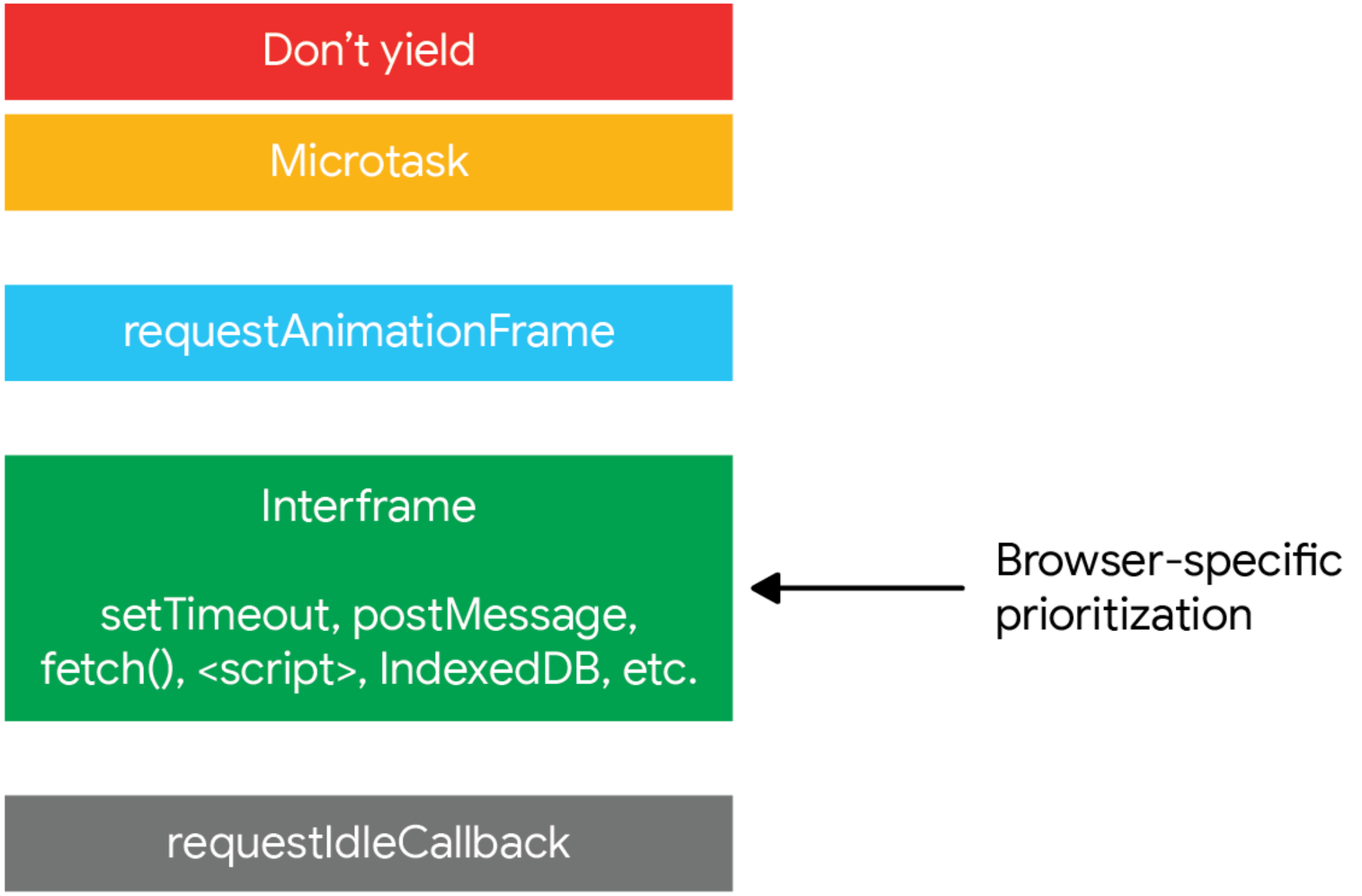
You still need to do as little work as possible.

Yield to user input.

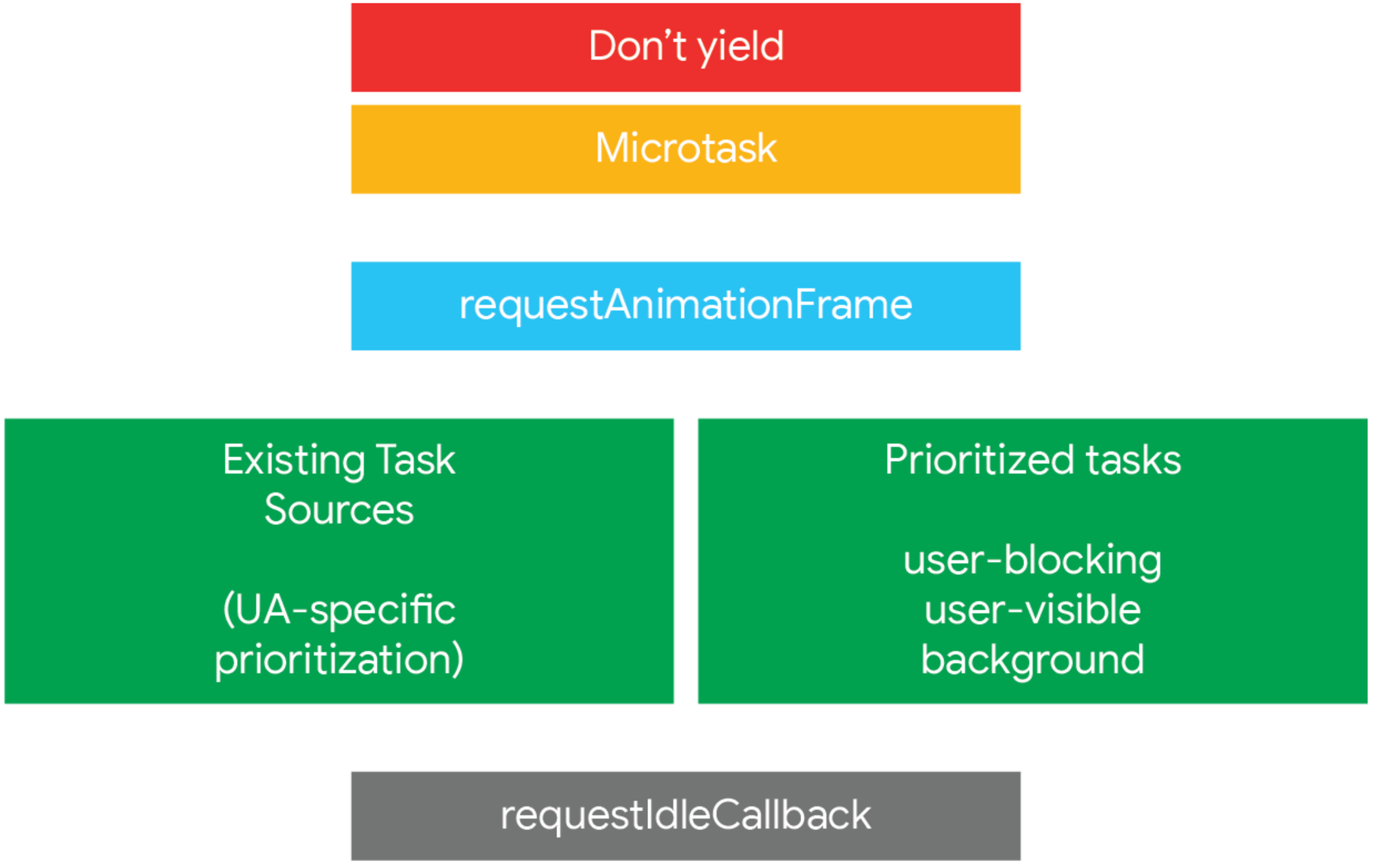
```
async function doWork () {
  // A task queue of functions
  const tasks = [
    workItemA,
    workItemB,
    workItemC,
    workItemD,
    workItemE
  ];

  while (tasks.length > 0) {
    // Optional chaining operator used here helps to avoid
    // errors in browsers that don't support `isInputPending`:
    if (navigator?.scheduling?.isInputPending()) {
      // There's a pending user input. Yield here:
      await yield();
    } else {
      // Shift the the task out of the queue:
      const task = tasks.shift();

      // Run the task:
      task();
    }
  }
}
```


No network throttling, 6× slowdown
task-chunking-demo.glitch.m…
800ms
1,200ms
1,600ms
2,000ms
2,400ms
2,800ms
3,200ms
Network
html
css
js
font
image
media
other
Interactions
pointerup
click
User interaction
to open menu
pointerdown
pointer...
click
Runs workItemC(), workItemD(), and workItemE()
Timings
blocking_task_1
blocking_task_2
blocking_task_3
blocking_task_4
blocking_task_5
Renderer (main)
Main
Run Task
Event: click
Function Call
Run Task
Timer Fired
Run Microtasks
Compositor
IO
Rasterizer
Rasterizer
Runs workItemA() and workItemB()
isInputPending yields to
handle user interaction

Prioritize work.

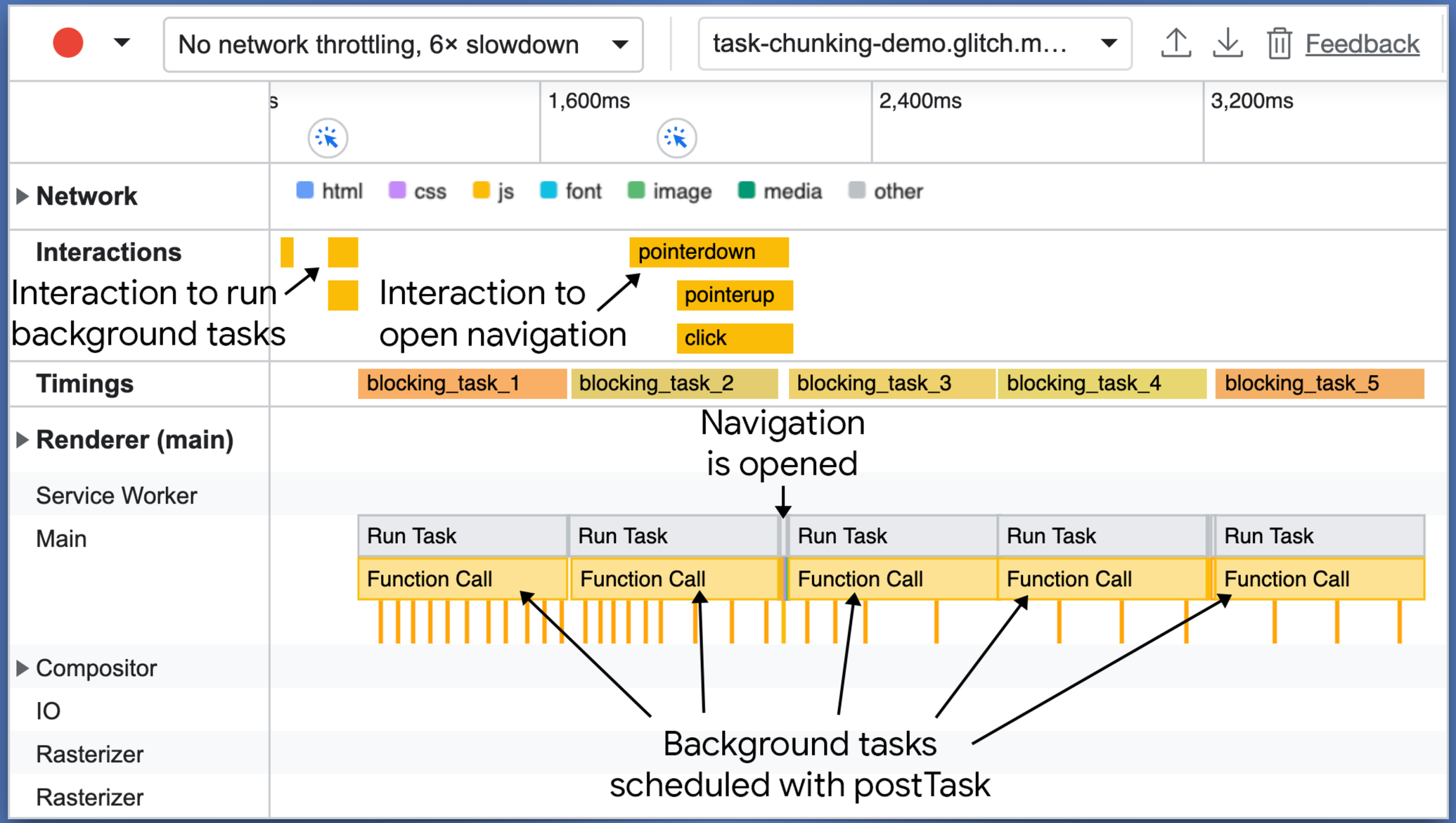


Adapted from the Scheduler API explainer at https://github.com/WICG/scheduling-apis



Adapted from the Scheduler API explainer at https://github.com/WICG/scheduling-apis

```
async function doWork () {
  // Schedule all tasks to run at low priority:
  await scheduler.postTask(workItemA, {
    priority: "background"
  });

  await scheduler.postTask(workItemB, {
    priority: "background"
  });

  await scheduler.postTask(workItemC, {
    priority: "background"
  });

  await scheduler.postTask(workItemD, {
    priority: "background"
  });

  await scheduler.postTask(workItemE, {
    priority: "background"
  });
}
```


No network throttling, 6× slowdown
task-chunking-demo.glitch.m…
Feedback
s
1,600ms
2,400ms
3,200ms
Network
html
css
js
font
image
media
other
Interactions
Interaction to run background tasks
Interaction to open navigation
pointerdown
pointerup
click
Timings
blocking_task_1
blocking_task_2
blocking_task_3
blocking_task_4
blocking_task_5
Renderer (main)
Navigation is opened
Service Worker
Main
Run Task
Function Call
Compositor
IO
Rasterizer
Rasterizer
Background tasks scheduled with postTask

INP is evolving.

Thank you for your time.

Jeremy Wagner – jlwagner.net – @malchata – Armada JS