Reducing input latency on the web

bit.ly/reduce-input-latency

W3C Games Workshop - June 2019

Navid Zolghadr (nzolghadr@chromium.org)

Chromium Input Dev

- Chromium input dev
- Goals
  - Smooth user interaction and scrolling across the web
  - Introducing new input capabilities to the web
  - Reducing the developer pain points

Latency sources

- Developers
  - Running long running js tasks
- User Agents
  - Detecting Browser actions
  - Optimizing document life cycle
  - ...

Avoid long running tasks on critical path

- Existing methods
  - Using requestIdleCallback for low priority tasks
  - Transfer the task to another thread using workers
- There is no existing way to check for pending user input
  - Check isInputPending (in incubation)

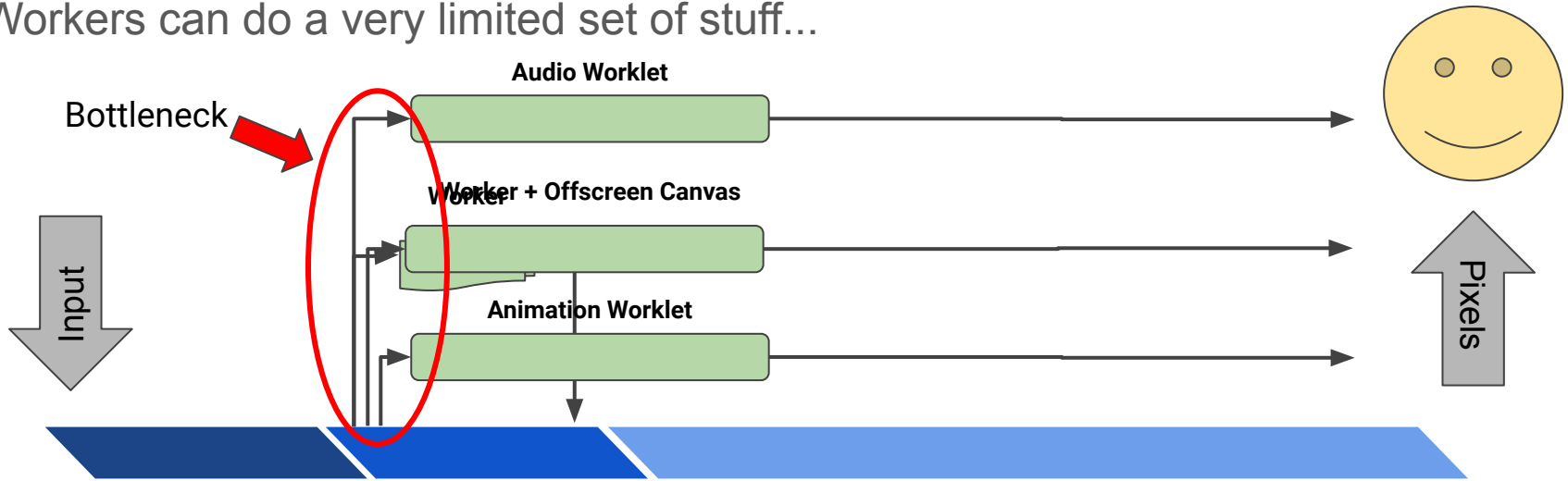
Browser induced latency

- Browser actions
  - Gesture detection (e.g. not sending touchmove events in slop region)
- Optimizing events with respect to document life cycle
  - Aligning user input with rAF to avoid excessive processing
- Processing on the main thread
  - Style, Layout, Paint, ….
- ...

Input for workers/worklets

The web of today

Workers can do a very limited set of stuff...



Event Handling

Dispatch input events and run event handlers

Animation Frame

Run user rAF callbacks

Style, Layout | Paint, Raster | Compositing

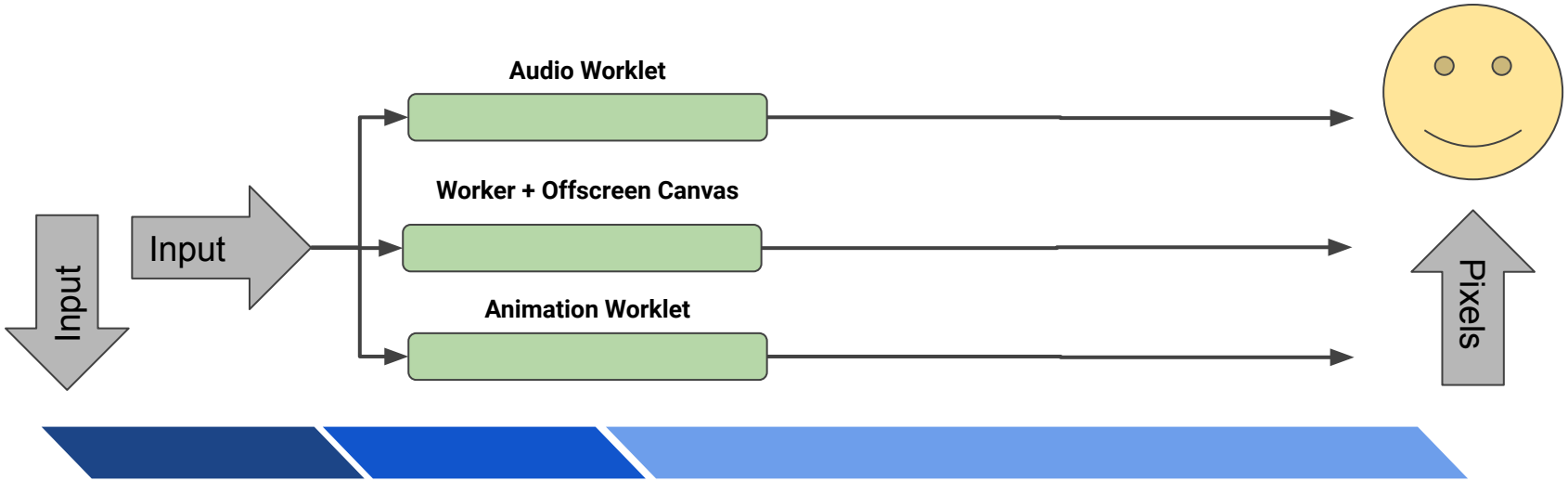
Compute new style based on scripted changes and other animations, Position all element, and paint them appropriately, Raster painted content and composite (Parallelize and use GPU as much as possible)

Potential Use cases

- Gaming & XR:
  - Worker + Offscreen Canvas + Gamepad + Mouse + Keyboard
  - Worker + Pointer Events + Network streaming
- Low-Latency Drawing
  - Worker + Offscreen Canvas + Pointer Events
- Low-Latency Interactive Audio
  - Audio Worklet + Pointer Events
- Interactive Animations
  - Animation Worklet + Pointer Events


Not Secure
mustaqahmed.github.io/web/input-for-worker/events-in-of...
Apps
jank
Main thread jank (0-500ms):
Drag mouse on the first canvas object above.

The goal



Event Handling

Dispatch input events and run event handlers

Animation Frame

Run user rAF callbacks

Style, Layout | Paint, Raster | Compositing

Compute new style based on scripted changes and other animations, Position all element, and paint them appropriately, Raster painted content and composite (Parallelize and use GPU as much as possible)

Proposed API

Main

```
var t1 =
 document.getElementById("target");

var worker = new Worker("worker.js");
worker.addEventTarget(t1);
```

Worker

```
self.addEventListener("eventtargetadded
",(event) => {
    // target is t1
    event.target.addEventListener(
           "pointermove",
           (e) => {
               // Handle event e
           }, {capture: true}
});
```

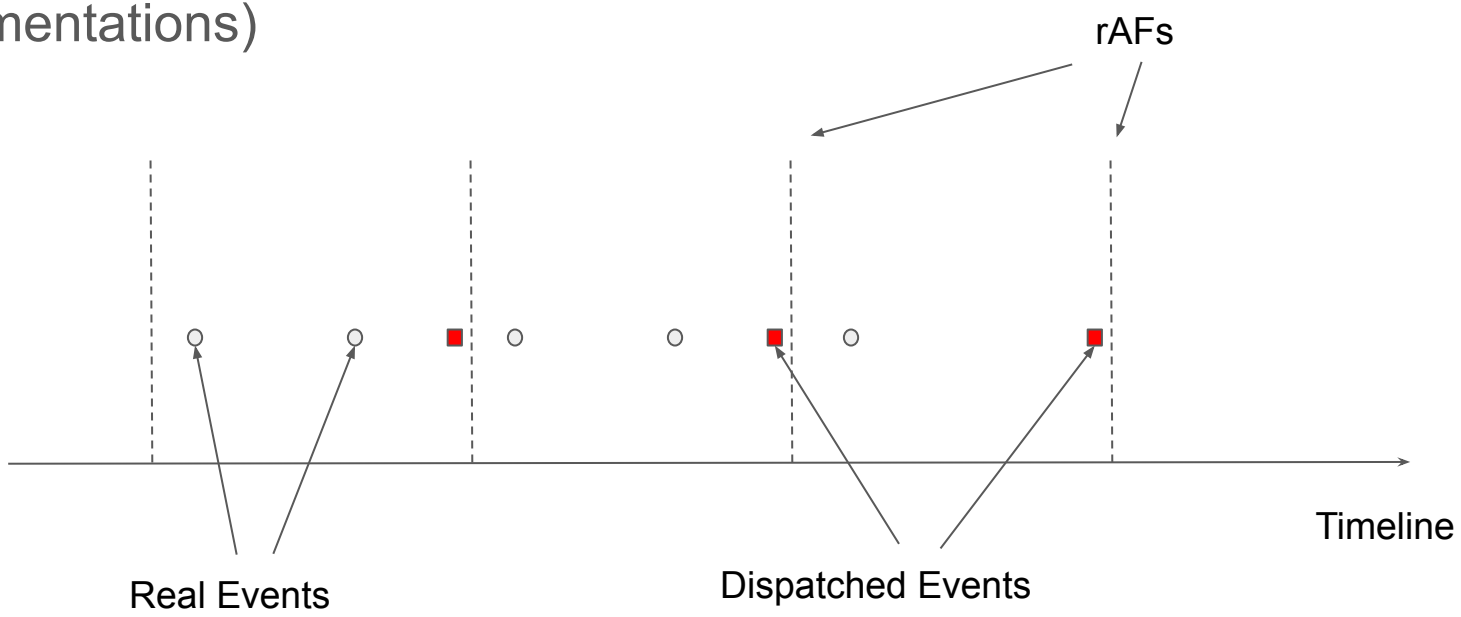
Practical considerations

- No need to postMessage the serialized input from the main thread
- No need to block on the main thread at all to process input
  - Can be sent over the network completely off the main thread
- Fully polyfillable! (at least in terms of the functionality)
- We are in early stages of the API design (explainer)
  - We need feedback to make sure this indeed does address related use cases out there.

Non-raf aligned events

The web of today

Delaying events until the next rAF (some heuristics involved in different implementations)



Pointer raw update

- High frequency events
- Gives an experience that the current eventing system doesn't provide.
- Demo (behind #enable-experimental-web-platform-features on Chrome)

	Count	Average Latency in ms
pointermove	299	10.78
pointerrawupdate	596	2.19

- Caution: it is potentially a performance footgun
- Early adopters and feedback are much appreciated (Pointer Events spec)

Discussion

New input capabilities on the web

- Looking forward to bring more input capabilities to the web
  - Pointer Events V3 features (better drawing features for gaming related use cases)
  - Unaccelerated mouse events during PointerLock
  - Support gaming mouse buttons and controls
- We need developer requests for the features to be able to justify them. Don’t be shy!

Reducing developer pain points

- We work towards interoperability across browsers around APIs
  - User activation inconsistencies for requestPointerLock and requestFullScreen APIs at the same time.
  - PointerLock movementX/Y coordinate space inconsistencies.
  - ...
- Feel free to reach out to me directly or file bugs in crbug against Chromium with Blink>input component.