Summary

Get a small set of browser/framework engineers who are interested in input together for a face-to-face brainstorming session. Where can we best collaborate to make input on the web awesome? What new APIs should we pursue standardizing at the W3C (and via which working groups)? Are there other opportunities for collaboration where our priorities overlap, and how can we make this collaboration most effective?

Add notes/comments here with any thoughts you have.

When

Monday June 23rd, 2014

Sorry for the short notice - we only just realized we'd have many Chrome input engineers together in California the previous week.

Where

Microsoft Redmond offices, meet in building 37 lobby

Agenda

Time	Topic
9:30 – 10:00	Introductions, Agenda Bashing
10:00—10:30	Pointer / Touch Events Sync
10:30—11:00	CSS :active, :hover, :focus with touch
11:00 – 11:30	Viewports: review the status quo Get everyone on the same page as to what each browser currently does
11:30—12:30	Viewports++: where do we want to go? Open discussion on the good parts of each implementation, problems we still need to solve, top priorities for the coming year
12:30-1:30	Lunch
1:30-2:30	Zoom Related APIs Device-Fixed, zoomTo(), Coordinates (conversion APIs, floating point precision, etc), sub zoomers
2:30-3:30	Scroll Customization APIs Snap Points, Scroll Blocking behavior, Customizing Physics,

3:30—4:00	**Break**
4:00 – 5:00	**Scroll Response APIs** **Events during/after scroll, scroll event triggers, scroll synchronized effects, fractional offsets, input driven animations**
5:00 – 5:30	**On-Screen Keyboards**
5:30-6:30	**Travel**
6:30 – 8:30	**Social Event (Lucky Strike)**

People interested in attending / related areas of expertise

- Rick Byers, Google
  - Input tech-lead / blink engineer, Pointer Events, touch-action, TouchEvents API / standardization, touch scrolling behavior, etc.
- Jacob Rossi, Microsoft
  - Pointer Events spec editor, input PM wizard
- Tab Atkins, Google
  - Spec wrangler extraordinaire
- Matt Brubeck, Mozilla
  - Touch events spec editor
- Max Heinritz, Google
  - Input and web-animations PM, blink feature process owner
- Tim Dresser, Google
  - Chrome engineer, gesture recognition, pull-to-refresh
- Zeeshan Qureshi, Google
  - Blink engineer, hit testing, OSK focus behavior
- Alexandre Elias, Google
  - Chrome engineer, compositor, touch, viewport, keyboard, etc.
- Gary Kačmarčík, Google
  - Chromoting engineer, OSK behavior
- David Bokan, Google
  - Blink engineer, pinch-zoom, OSK viewport behavior
- Yufeng Shen, Google
  - Chrome engineer, input latency tooling, perf benchmarks, touch screen drivers

Chrome priorities summary
Almost all effort in chrome input is focused on making the web more competitive with native mobile platforms in three broad areas:

- performance (consistently low input->paint latency)
- richness (apps can build complicated effects without re-implementing browser behavior)
- rationality (simple/obvious things tend to be correct, interoperability, layered abstractions)

See this presentation for an overview of the toughest ongoing problems we see here. But there are also many smaller issues which should be more tractable for concrete collaboration.

IE priorities summary
cross-browser pointer events, snap points, … ? TODO(jacob)

Confirmed topics
Topics we agree are worth discussing, grouped by area / working-group. Should form the basis of an agenda.

1. Viewports and zooming functionality
    a. Redefining the single viewport as two viewports to account for zooming
        - should we allow websites to opt-in to reasoning about the pinch viewport and adapting their UI for it's current position/scale?
    b. Device-fixed
    c. zoomTo()
    d. Coordinate space conversion APIs (zoomed to unzoomed, origin shifts, etc.)
    e. Sub-zoomers (nested zoomable regions)

2. Customizing manipulation behavior
    a. APIs for scroll customization
        - transitioning between browser-driven threaded scrolling and JS-driven effects
        - customizing fling physics
        - snap-points
        - control over scroll blocking behavior (eg. should it be possible for apps to choose to block scrolling instead of showing checkerboarding / unrastered content?)
    b. Input-driven animations
        - eg. declaratively connecting a WebAnimations timeline to input positions?
    c. Eventing during scrolling / manipulation
        - Enabling effects like pull to refresh and snap points to be built in JS
        - Gesture info regarding the manipulation particulars (2-finger pan vs. 1-finger, how much rotation, etc.)
        - Boundary eventing (e.g. for pull-to-refresh)

- Mouse events when clicking on scrollbar, e.g. for image carousel scenarios
- Inertia destination APIs
- Scroll-synchronized effects (scheduling of 'scroll' events)
- Scroll-position triggered events (e.g. fire an event when passing 1000px, rather than listening to every scroll event)

Other input items of interest

Here's a list of input-related issues that we on the Chrome team are working on or at least thinking about, in roughly decreasing priority. How does this map to the interests of the other browsers?

Add comments if you'd like any context on any of these ideas. For the most part there are public Chromium bugs (and sometimes even design docs) discussing them which I would link to if I wasn't lazy.

1. Default on screen keyboard behavior [Chrome P1]
    a. the extent to which it should be exposed to the platform or hidden from it by default (eg. window size, fixed position elements)
    b. when it should be shown/hidden (when focus invokes the keyboard).
2. New APIs for understanding and controlling on-screen keyboard behavior [Chrome P1]
    a. Explicitly showing/hiding the OSK
    b. Customizing keyboard layout / labels
    c. media queries for desking OSK vs. physical keyboards
    d. Attaching custom UI (shortcut buttons) to the top of the keyboard (including during animation in/out)
3. Exposing fractional event co-ordinates and scroll offsets to the web [Chrome P1]
4. Performance tooling [Chrome P1]
    a. eg. measuring input->paint latency (both mean and variability)
    b. enabling web developers to better reason about their latency
    c. exposing more performance APIs to JavaScript?
    d. effective input-latency cross-browser benchmarks for driving competition in this space
5. Rationalizing/specifying :active/:hover behavior for touch [Chrome P2]
    a. Chrome leaves :hover set after a tap, IE doesn't. We think we can match IE for mobile viewport sites, but not legacy sites.
    b. Can we standardize this?
6. touch-action use cases / improvements [Chrome P2]
    a. eg. is the precise hit testing behavior important / something to worry about interoperability on?
    b. are IE and Chrome otherwise completely interoperable today?
    c. pan-y-positive etc.
    d. getting Apple on board with touch-action

    e. relying on pointercancel for permitted pans - a valuable property to make side-scrollers simpler to implement?
7. PointerEvents in blink [Chrome P2]
    a. Can anything be done to alleviate the fundamental concerns here? Seems unlikely.
    b. Strategies for improving interoperability even without support for pointer events in blink. eg. could we incrementally add enough features that we could trivially map between the pointer API and mouse/touch API with a simple JS polyfill?
    c. Tradeoffs relative to evolving touch events to be device agnostic?
    d. Addressing the hit-testing concern, better supporting the Polymer PointerEvents polyfill
    e. Defining Pointer events and touch events interaction (perhaps mainly of interest only if we can find a path forward for blink)
8. TouchEvents in IE [Chrome P2]
    a. What would it look like if IE decided to implement TouchEvents?
    b. Could we make it fully compatible with Chrome/Safari/Firefox/Android without breaking IE's PointerEvents compat?
    c. What could we add to TouchEvents to make it more appealing for IE?
9. Eliminating the click delay from the web [Chrome P2]
10. Input prediction [Chrome P2]
    a. Should we try to move away from a world where movement is represented as points at some discrete time point subject to error and instead towards a notion of a 'motion vector' where motion is plumbed through the system as a function of time (with that function being updated)
11. Task scheduling [Chrome P2]
    a. to what extent are we free to re-order JS tasks (eg. input events) for better scheduling wrt. to vsync and the rendering pipeline
12. Improving Chrome's touchscreen support on Windows [Chrome P2]
    a. Should we abandon touch support on Win7 and move only to pointer events on Win8+?
    b. Enabling radius support (correcting windows radius scale problems)
    c. Touchscreen detection and conditional enabling of touch events
13. APIs for describing the input hardware available [Chrome P3]
    a. pointer/hover media queries
    b. more?
14. Threaded input handling [Chrome P3 - important, but not terrible tractable yet]
    a. Eg. should we allow apps to handle input events or at least track scroll offsets in a webworker thread, unaffected by blocking on the main thread
    b. "Performance modularity" - enabling some effects (eg. a particular touch animation) to be isolated from a perf perspective from other portions of a large application

Social event

Lucky Strike for Bowling, food and drinks
6:30-8:30 after the event