Immersive Web F2F Notes

July 24, 2018 - Seattle

WiFi is "Guest"
Join the Chime call for IM https://chime.aws/6007892569

Agenda:

8:30: Coffee and breakfast, AV equipment futzing

9:30 - 9:45
- Agenda
- Introductions
- Situational info

Speed round: *15 minutes*
- Animation timestamp #347
- XRRay
- Ada's doc

Working group logistics: *30 minutes*
- Recap relationship between CG and WG
- Charter IP scope
- Chairs & editors
- Testability

Input: *1 hour*
- Dependence on OpenXR design
- Timing of advanced controller input

11:30: Bathroom & coffee break

Render path and XRPresentationContext: *30 minutes*
- Unified render path

12:15 1:30: Lunch

Navigation: *30 minutes*

Sessions I: *1 hour*

3: Snack and coffee break

Sessions II: *1 hour*

Bathroom break & voting on breakout sessions

Breakout sessions: *2 x ½ hour*
- (**4**) Performance and timing
- (**6**) Multiview rendering and texture arrays
- (**8**) AR topics (i.e. capabilities of the APIs)
- (3) AR privacy for the APIs
- (**4**) Testability
- (2) UA strings for headset browsers

- (3) (continue) Session creation / capabilities
- (0) (continue) Navigation
- (2) (continue) Inputs
- (3) No more no more please stop
- More suggestions
- Even more suggestions
- (7) Soft whimpering in a corner

End at 5:30
Dinner

Session Notes

Working Group Relationship

Brandon loves everyone, but is turning over chairpersonship to Ada, Chris, Trevor.

Trevor: The CG will continue with the same membership; it will act as the incubator for new ideas and will use the same functional process.

The WG will take ideas out of the CG and will run them through the W3C process to become recommendations. This will include getting feedback from TAG.

Note that variants and adjustments to things already in flight in the WG will stay within the WG (changes don't require moving things back to the CG). However things that are open-ended or require design work will fall into the CG.

Chris Wilson (cwilso): The goal of the WG is to get things to actual specifications and it's important to do this to get the spec. to market.

The CG model is similar to WICG which allows ideas to informally progress and then graduate to workgroups; in general each WG requires a charter.

The biggest difference between a CG and a WG is intellectual property (IP) coverage. Contributions of ideas to the WG are contributions of the IP behind the ideas to the WG. This isn't true of the CG.

CG and WG should work closely together, the goal is to create a waterfall model where ideas can be public early (to get as much feedback as possible) and create a transparent specification process that is open to the public.

Things should move towards the WG when they're relatively stable in terms of discussion.

Trevor: There is a draft charter available. (https://immersive-web.github.io/webxr/charter/)

Chris: Terminology of charter could use some more crispiness (includes the word 'magic' twice)

Responsible for signing charter w/in each company (*apologies for miisspellingz)*

Nell, Amazon
Trevor, Mozilla
Chris Wilson, Google
Yashar, Microsoft
Frank Olivier, Oculus
Rick, Magic Leap
Ada, Samsung
Bryan, Intel

Chris: At present the charter does not include AR-related features such as lighting estimation. Would everyone prefer a broader charter scope, or a narrower one?

Justin : AR would open a can of worms. Trying to land something in the next 3-6 months and expanding the scope from an IP perspective and would require inclusion from a different team.

- Passthrough, ray casting are OK from a VR perspective

Blair: Basic AR stuff is already in the charter; there's 2 levels. Deeper AR is world knowledge, tracking thiings, etc. etc. However, see-through displays, optical items (foundational work to set up AR) is in the charter, as are a few other things (e.g. hit testing, anchors) without the other things; would like to see them in the charter. Will want to set things up so that things can expand in the future.

Chris: CG work doesn't prevent implementation, but things in the WG are more stable. Sounds like recommendation is to put a narrow section of AR into the charter (i.e. discussion around anchors)

Blair: All WebXR traffic #s are likely to be driven by smartphone AR - not VR headsets in the short term - so it's important to include AR as part of the WG charter.

Nell: We'll rework the charter to include some of the AR features.

(moving on to ownership)
Nell: Clearer ownership and roles will give us better ability to execute on the specifications, and clear delineation between categories of work will help as well.

Chris: In a WG there is the design work (deep, broad knowledge of the space); editor work (writing stuff down, which is separate from design but can convert design into spec); flow control/administrative work (chair) to prove that the group is operating on consensus. CGs don't have to operate on consensus, but a WG does. The *objective* flow control piece is the most important to get right, but it's also separate from the design work; *objectivity* is important for the chair roles.

Nell: It's probably reasonable to split these roles apart, they're a fair amount of work. After discussions w/ Brandon it makes sense for Nell and Brandon to step away from chairpersonship and move towards spec designing and editing (due to a preference for being opinionated and not objective :))
Nell: Would like to propose Trevor as the chair for the CG for flow control. For the WG, propose Ada and Chris co-chair. This gives representation from Mozilla (Trevor), Samsung (Ada), Google (Brandon, Chris), Amazon (Nell) (including both editor and chair roles) as well as geographic diversity.

Chris: Chair appointments are approved by W3C team; they want the WG to be happy with it, but they have ability to appoint or remove chairs. Chairs control who the editors are.

Trevor, Ada: Looking forward to working w/ everyone
Chris: Excited that Ada has direct connection w/ developer community particularly for XR

Input

(scribe Trevor)

Brandon: This is an ongoing, somewhat contentious topic. The goals is not to solve the entire problem, but to reach consensus about the level of input that is required for V1.
We started with a single button input. It has been implemented in Chrome, but I've heard from developers that it's a step backwards from WebVR, please don't do this. It would be unfortunate to break React or Sumerian or other projects by taking away capabilities.
So, let's try to get a consensus for a direction.

Question: do you mean the gamepad extensions?

Brandon: Yes! The WebVR spec doesn't address input but we piled on extensions to the gamepad API for poses and haptics. The gamepad API is not the most user friendly API on the web, loosely tied to what the buttons are for, for example. Apps are sniffing the name of the gamepad, like looking for "Vive" and giving up if it's not there.
So a more robust system would get away from user agent or gamepad name sniffing, instead using feature detection.
So, if it works with Vive controllers then it should work with Oculus Touch, for example.
Some people in the W3C do not want us to hang non-gamepad data off of the gamepad API.
Some of the gamepad group folks are open to changes.

Q: Should we make a higher level … (missed it) probably about more abstract input types

Brandon: That’s a good idea! We should talk about that.
Some history, Nell and Alex ands I started an input API and then the OpenXR input model came along and it was incompatible. It is based around declaring high level actions and then suggest input mappings for that. The intention is that the runtimes will expose a remapping feature where users can change the mappings. This is a great thing for accessibility, but it comes with problems for how it maps to browsers and the web. There is explicitly no mechanism in OpenXR for iteration through inputs, so a gamepad like API isn’t possible.
Our initial idea in reaction was to have just a primary action and some method of pointing, whether it’s your hand, a controllers, or tapping on the screen. The app receives notice that the user activated and deactivated the primary action and a ray.
Most applications on the web won’t need that, like video players and slideshows. This also allows us to be a user-activation event.
So that brings us to today and this explainer. (proposals/Issues/17) Because developers are regularly expressing concern, the question is whether we want to do more.
Opens the floor: how critical is exposing more?

Nell: The concerned I’ve raised is with WebVR today we can render the controller and the state of the thumbstick, but the existing proposal is a regression so devs need to consider whether WebXR is ready, should they stick with the gamepad API?
If the idea is to deprecate WebVR, then we can’t leave behind devs using more complex VR controllers.

Artem: I agree. If we don’t match current gamepad API they’ll still use gamepad API unless you aggressively cut it off. I mean, identifying the controller and button states.

Nell: We run the risk of developer confusion if we do this half-way model where the gamepad API is for some things and the WebXR input is for others. We would lose credibility as a spec.

Justin: We have specific VR controllers, but there is the existing gamepad API that serves different needs. Devs will still have to bridge the gaps in libraries.

Alex T: I don’t see a world where gamepad API completely disappears for XBox gamepads but the question is whether VR controllers are exposed in two APIs. I believe the intention is that VR controllers aren’t exposed until the session starts.

Brandon: Let’s get into the weeds of API shape. Just yet.

Ada: For the browser people in the room, is the intention to keep the VR extensions to the gamepad API present until the WebXR input API is fully implemented as a replacement?

Reading chat window…
<Leonard Daly›
Would it be possible to define an extension mechanism that allows developers to create a spec-compliant extension that isn't explicitly in the spec for access to features that may be limited to specific hardware/hardware capabilities?
Brandon: that's API shape which we should defer.

John @ Google: I think nobody wants the WebXR input API to grow to encompass all possible input types like keyboard and mice. Is seems like tracked input controllers is a different problem, and how much of this input proposal is addressing the latter problem?

Brandon:

Kip: My 2 cents: the new proposal ads a lot of value but I also don't want to see the regression. The extensions to the gamepad API are also useful outside of VR. So, what would be the state of the extensions if WebXR moves away from it and how have we interacted with the gamepad API community?

Brandon: The gamepad API hasn't had a lot of attention, and I'm not sure what their status is. There's some of it that has been merged back into the gamepad API.

Kip: How many would be considering keeping the gamepad extensions and the WebXR input API?

Brandon: We would be very reluctant to keep dual modality. I'm a little surprised that we don't want gamepad extension to go away. I think there are opportunities for polyfills to continue to support gamepad extensions.

Alex T: We took pains to expose things exactly once. The cross-exposure makes it hard to prevent double events. If we still want to light up gamepad for XR controllers, the current exact way is a mess with the indices not being consistent, forcing users to check for names. So, we'd need to change the gamepad APIs anyway.

Justin: There is a standard button mapping and the intent was there in the original gamepad spec. We haven't talked about the other things it does well, like press-and-hold, long-press, etc. The question of would we support both, the answer is yes. As soon as they have overlapping feature-set, it becomes a problem.

Brandon: There is pseudo-mapping that is an attempt to match whatever I did the first time I implemented it, which is somewhat random and we would need a better mapping.

Lewis: A standardized mapping will leave gaps in the array, which can bee more problematic. Indexed based APIs make it really hard to standardize a mapping.

Josh from Oculus Browser: There seem like two very different cases: small experiences and complex experiences, so why not have two APIs.

Brandon: Yes, it might make sense to have the simple activation WebXR API and the more complex WebXR input API.

Justin: They always want more!

Brandon: The intention was never to forever have this simple input, but to wait for changing things like OpenXR to settle. On the Chrome side, we want to deliver something before OpenXR is a viable thing so we need to resolve before that.

Nell: In response to Kip, about who maintains gampead API extensions that work outside of VR: Pose data would need to change because it’s not integrated with XRFrameOfReference.
We need to clarify what are the problems with gamepad, called out separately from the WebXR input API.
I think if we go down the list of things that need fixing in gamepad, is there an actual person who edits that spec?

Brandon: There are a couple of people working on it, including Matt Reynolds, but he’s not an editor. There is an editor whose name escapes me. (Chris say “Steve” from Sony)

Nell: The expectation used to be that changes to gamepad was a non-starter.
I don’t think we have clarity around form factor for XR as we do for things like the gamepad API. Knuckles is different than Touch is different than hand gestures.
I can see why OpenXR is moving to mapping high level actions instead of listening for low level input. As new form factors are introduced, how do we bring existing web content along for the ride in an unstable market?
If we want to do an OpenXR style input-action mapping, are there things that are unique to the web environment that makes an OpenXR design problematic?

Brandon: The sentiment that I’m hearing is it would be a significant regression to not bring along at least a comparable method of input as the gamepad extensions provide and that might hurt the WebXR Device API.

David D: Chrome has an origin-trial for the gamepad extensions, so you could have an XR input and get a gamepad for it, but the extension is duplication to what’s in XRInput. So don’t get stuck on the extensions.

Alex T: Your proposal is about polling, so let’s make sure that we’re talking about the same thing.

Brandon: Let’s talk about form factor. Do people in the room feel that the OpenXR model is workable or do we need something more specialized?
On the Issue, John S from Mozilla pointed out that they’d like something like the OpenXR mapping.
My responses are: (not retyping, see proposals/Issues/17)
So, if we do go with an action mapping system, to make that work against the OpenXR proposal you’d have to create a 1 to 1 mapping from OpenXR semantic path to a WebXR path, and we’d be always going back to add more mappings for new hardware.
So, our choice is fully OpenXR or is only different in cosmetic ways. That means that OpenXR would be *the* canonical back end.
We’re not there, yet.
Discusses the API proposal in Issue 17
Has the ability to be run and remapped on top of OpenXR.
If everyone decided for a good use of the pinky, it wouldn’t be supported by this proposed API.

Nell: We’re coming toward the end of the time for this session.

Brandon: Another alternative would be to continue using the gamepad API proper without the poses to fix the discontinuation with the WebXR input API. We could add a gamepad attribute to the XRInputSource, which has some issues but is possible.

Justin: We could use a derivation syntax instead, class extending the gamepad class as we need to. So, XRInputSource could extend gamepad and we expose them through the WebXR input API.

Brandon: We’d have both the array of gamepads and also the XRControllerState?

Justin: TrackedInput would be the primary thing and gamepad would do what it’s good for with buttons, etc.

Nell: I seem to remember that we were working on teleportation and there was confusion around different controller mapping. We got some feedback from someone (maybe Fernando at Mozilla) that they were looking at action maps for their input model.

Trevor explains action-input work at Mozilla

Justin: as long as there’s a connection between the gamepad instances and the XRInputSource that prevents the problem of duplicating information and events.

Alex T: One problem is disambiguating between different types of devices that are exposed in the same APIs. So, how do devs make design decisions for input?

Brandon: For things like bluetooth clickers, the XRInputSource wouldn't have an XRControllerState.

Alex T: That feels find if the state comes through the WebXR API. If it's also available through a different front door and only controller-type sources show up there, it's hard to figure out how to use each data type.

Nell: We're at time. What do we want to do?

Brandon: If there are people who are highly invested, we can continue in the breakout session.

general chatting about timing, end with continuing with input until lunch

Brandon: Ok, we now know that we sholuldn't ship without a better input plan. Does anyone strongly advocate for an in-built input-action mapping system instead of a library on top?

Kip: It should not be the only system.

Brandon: Would you be OK with shipping with the proposed API and then add a mapping system down the road.

Kip: I don't see a problem with adding features down the road.

Jokes about naming: PointingThing, GestureThing

Alex T: We have the tiny beachhead with the select event and exploring with libs on top make sense.
The OpenXR action-mapping system exist and hard-core? It has a title mentality instead of an app mentality. Title mentality: The problem is forward-compat problem to solve when adding new controllers for old games. The app mentality: The problem is forward-compat core that the UA is responsible for making it work in the future, nobody will rewrite the web app.
But, if you want to do the JS lib for action-mapping, that's possible with JS (e.g. fetching new data from CDN) and we're not restricted because it's more dynamic than changing the native runtime.

Kip: In 2D games, you need basics maps and game specific maps. Maybe the UA offers the basics and up in JS one could use game specific maps.

Brandon: yes.

Justin: A particular audience expects the action-mapping system, but it's not something that users who aren't developers won't use. They'll leave the app before they set up a new action map.

Brandon: Even if we have a data structure state (like gamepad) we can take advantage of input mapping in the JS libs. It doesn't give you per-page control and how practical would that be, anyway.

Alex T: The title approach only works because there are a small number of titles with a community of end-users who might create action-maps, which doesn't work on the web. Or '*Trevor style*'™. (DANG! -T)

Trevor: Action-input is a group project, not a Trevor library.

Josh from oculus: …. Per-page doesn't work.

Nell and Brandon: So many complexities of sharing action-maps.
(sorry, lost track a bit - Trevor)

Brandon: So, yes we need the more complete API, no it won't look like action-mapping, yes XRInputSource will probably expose the relationship with a gamepad.
That's good for us to work on going forward.

Alex T: Plus, we need clarity on whether we're iterating through an array of gamepads that include XRInputSources.

Brandon: Should VR controllers show up in that array?

Justin: Yes.

Josh from Oculu: Yes, it's a breaking change.

Brandon: That's a good question and there are ways to polyfill that by changing the gamepad API for reasons of backward compatibility.

Nell: We do that across the board for WebXR in the existing polyfill.

Brandon: The polyfill goes in the other direction today, but could change.

David D: For XBox controllers, those would not be exposed by WebXR so how does the dev know which to use?

Kip: Let's not enumerate in the gamepad API things that are not gamepads, but if a VR controller is a gamepad it should be in that array.

Lewis: Something like XBox controller would be an XRInputSource and voice but ephemeral, like only during a session. So, they don't need to be in the input array.

David D: Immersive things could be in WebXR input and non-immersive controls would be in gamepad or other APIs. We could break getGamepads if the session ends.

Nell: Ok, we're at lunch in 4 minutes!

Artem: How do we distinguish controllers and gamepads

Brandon: Sessions would have an array of XRInputSources. There is no way to know that you're using a Touch or Knuckles and rendering a model as such.

Everyone worries about fingerprinting.

Brandon: Avoid name sniffing!

David D: What if there are five gamepads returned, what should the app do?

Brandon: This is a part of what the original input API was supposed to solve, but would go away if we just share an array of data structures.
You'd iterate through, render them as specified, respond to events from them, and that's a forward compatible.
So, use the simple select/activate API for simple things and the complex iteration / data structure API for more complex situations.

Summary:
We should consider iterating on Issue 17.
We can talk about bringing existing gamepad structures into it.
Down the road we should discuss how to render.
We definitely should figure this out before shipping.

Navigation

David Dorwin: (ran through explainer, please read for more detail, a few salient points)

- Potential Threats are non-exhaustive, but include some of the challenges related to this being an immersive nature.
    - E.g. It's possible for an experience to imitate the browser and this could be a way of spoofing passwords.
    - E.g. We don't want users to be surprised or presented with content they didn't ask for.
- Cross-origin linking might break the 'metaverse' notion where you can browse through the web in an immersive way

Q: Why is the metaverse scenario different than just browsing the web?
(bajones) A: It's harder to guarantee in an immersive scenario that you know what URL you're on and get the same signals on security, etc.
Nell: Note that in immersive mode every pixel is drawn on the page. In specs we avoid specifying what a browser would do around the page (e.g. displaying a URL, HTTP vs HTTPS warnings are not part of specs, they are best practices).

David: The goal of the explainer is to highlight considerations for browsers during implementation. Only the normative sections are required for implementation, but listing the considerations helps browsers fulfill their user commitments.

Fullscreen is some of the best prior art for fully immersive scenarios; if you navigate today away from fullscreen it kicks you out (otherwise sites could navigate and then prompt you for your password and pretend to be the user agent).

Nell: One key point is that users generally won't close their eyes to avoid bad experiences

David: (ran through normative concerns and possible mitigations), and possible normative text.

Q: For second normative text suggestion, does this mean the browser would have to keep track of sessions on previous pages and future pages if it wanted to re-allow creation of immersive session when, for example, the user goes 'back'? (A: Yes, they'd have to do this)

Q: Why does same-origin navigation require a user gesture during each navigation?
David: This might be affected by page architecture, i.e. if pages are in iframes then feature policy might be required.
Nell: This would depend upon what types of sessions require what types of permissions. A question when we discuss sessions is this particular question.

Q: But a lot of 2D pages compose content from other pages. The VR space doesn't do this now, but can we get to this eventually?
David: This proposal doesn't preclude permissions, which might require a user activation. For example, GetUserMedia doesn't require a user gesture but *does* require a permission.

Switched to Brandon discussion the PR:
- Seemed like navigator.xr was the appropriate place to request a NavigationSession because it has the right context
  - Could require that it's called before a window 'load' event to allow safe switching out; might address concerns about hanging in the mode for too long. (Brandon isn't concerned about having a spinner and then getting a 2D page, but willing to discuss perspectives)
- Key differences from previous conversations:

- Not reusing regular requestSession (questions about it being confusing, arguments). This means the API call implies you're *only* getting an immersive session.

Q: Question on design - requires that pages all be 'navigatible' - is that intentional?
Brandon: Hard to imagine otherwise. In a more declarative world maybe pages don't need to opt-in, but as a core API then page would need to create session and voluntarily opt in to this. May want to have restrictions requiring a query for support.

Q: Sure, but if you request 'present' the actual callback could be the same regardless of the mechanism for requesting a session
Brandon: One difference is that in the navigation scenario, you might not be querying the XRDevice beforehand, but it'd be trivial to structure page so that both a raw start, and a navigation start, could funnel to the same sessionStart function

Feedback: Challenge with the promise is that it could be fulfilled much, much later
Feedback: Browser should fulfill promise *after* onLoad to make sure all the content, images are present (brandon: agreed)

Q: Could you specify a timeout for when the session creation must return / expire?
Brandon: Probably a good argument for doing this as part of implementation, might not need to be a firm requirement
Q: But if the load event finishes and nobody calls ImmersiveNavigationSession? (Brandon: You're stuck in 2D)

Justin: Delaying for an onLoad event could be a challenge if the content is loaded afterwards; the timeout would still work, but the onLoad event might not

Feedback: As soon as you expand this to more complex pages does this break down?

Brandon: I don't think this approach breaks down on things like the Facebook feed, or pages where you have places you want to navigate to.

David: This is likely restricted to the top-level document; x-origin iframes would not have access to this capability

Nell: One of the concerns previously brought up was the timing - what if a page can't create something meaningful quickly? Nominal language was discussed allowing the UA to give users a method for exiting the session, but would need to be really clear on what best practice guidance would look like. E.g. If page takes 5 minutes to load it'd be good to immediately get a session ended event

Brandon: In fact there’s nothing preventing the UA from giving you an ended session when the promise is fulfilled

Kip: Use case - how do you fulfill a splash screen during load (to allow more time)? How would this transition work?

Brandon: Backing up - in a 2D UX would you want to do this?
Kip: Generally developers may want to display something during loading, that will be a common question, how would we answer?

Brandon: In this scenario there isn’t a transition point to a splash screen and back. The ultimate answer is probably something along the lines of an image layer; doesn’t require RAFs - but not sure if that’s something to pursue right now, defining more than one layer will take some time. There’s a reasonable expectation that even if you aren’t getting full frame rate, you could put in black frames or possibly the tech stack (e.g. THREE.js) could do a basic image rendering.

Kip: Is there any benefit to having a declarative ‘I am finished’ state for the browser to know?

Brandon: Not sure what browser would do with that

Feedback: Seems like it’s one of those transition information pieces where the UA might show some safety information during load times - possibly as a parameter?

Brandon: Could be something where a URL / favicon was sitting in a metaworld while page was processing in the background, but this is probably UA behavior

Leonard: Would "z-index" be available? If so, a higher index could be displayed over the canvas div.
Brandon: Not sure that’d apply in this case

Feedback: User agents would probably need to show some safety information during transition, what can we hang off that?
Brandon: Probably depends upon whether not its same origin, but this might just be something the UA requires
Trevor: This is something that Firefox Reality will do - the user and the browser will have a unique, visual thing that the browser can NOT spoof on a per-session basis, and the browser will definitely show an interstitial

Feedback: If we do a normal requestSession at the ‘right time’ - what makes the requestImmersiveNavigationSession different?

Brandon: Because it has a purpose we can have better errors, more explicit documentation around what is (and isn’t) valid. Trying to avoid having requestSession work ‘this way, unless it’s called in this alternative context, in which case it's entirely different' - might as well break it out separately. Prefer to have a separate call.

Justin: Did similar things with send beacon and fetch, but took ~2 years.

Q: If there are new arguments, e.g. AR features, how will this extend?

Brandon: In that case it gets into the session conversation

David: There some interesting properties of the navigation session which make it important. You may be constrained. For example, certain headsets may prevent it; these constraints would need to consider privacy as well. May need to restart runtime in some situations (tear down the old one before starting up the new one for privacy reasons)

Summary: Yes, we would like the metaverse, but there are many considerations. We should pick a basic solution (e.g. what Brandon proposed) as a starting point with narrow cases, figure out what the mitigations and best practices are, and then grow from that point.

Unified render path

[scribe:cwilso]

Brandon: [summarizes https://github.com/immersive-web/webxr/issues/367. Responsive design incorporating XR is hard.] Jordan's suggestion: "poseless session", return a session with the identity matrix as pose, maybe UA adds mouse nav, etc.   If you had the polyfill on your page, you'd always get at least that device back. You'd use the magic window session rendering, but without device sensors driving the pose.

Nell: key things of note: 1) we've sung the praises of progressive enhancement before, but we've left this first step off. This would step up to magic window. 2) window raf vs session raf might be simplified in this case. The concern, on the other hand, is that you might never go into a "real XR" session (no hooks in to device sensors), and we're getting close to reinventing GL context. (also, we'd want to "promote" sessions from poseless magic window to real device)

Leonard Daly asks:
1) What kinds of devices would not support at least magic window?
2) Would this cause a problem for an app that was trying to deal with disability access when the app wished to define the appropriate capability down-grade? for an xR app that is designed for use by a disabled person. If there is a pre-defined fallback, is that fallback suitable for disability access OR can the app dictate certain features that should/should not be enabled?

1) desktop - no accelerometer driving pose

David - if you do have touch or mouse you can override with, someone may implement for 0-DOF, but not 3DOF.

Justin - seems like a good idea, we could then expose XR controllers as well (in poseless). You could also have multiple Sessions going.

Brandon - there's a possibility you'd want this.

Nell - probably worth calling out that poseless doesn't work for AR. I'm wondering what the value of the XRDevice object is at all. What does it mean to progressively enhance if we have to rebond sessions and xxx I don't have answers, just questions. What does it mean to return an XRDevice when a PC doesn't have a headset plugged in but then one gets plugged in and it's not the same type? Brandon - I feel like it's good to be able to get to before getting to a session, but maybe not.

Nell: why would a developer want a poseless session?

Justin - anything the developer can get back, testability. David - couldn't we solve with browser APIs?

Brandon- maybe we should ALLOW poseless sessions - like say "it's okay" or "it's not okay to give me a poseless session". Testability probably should be solved in a different manner. Should probably be able to force hardware use.

Alex - in code forking section of this issue, the use cases are pretty compelling.

Nell - the key thing was the switching-back-and-forth cost.

Chris- "poseless" is, as Jordan said, a crap name. (Artem suggests "sensorless") And

Nell - testability doesn't need to come through this path.

Daniel - in this mode, would it be possible to accidentally select a different GPU via compatible device, and then it doesn't upgrade properly?

Brandon - if we handle poseless at the device level, we can handle this. If you've legitimately got your headset plugged into an older GPU and you promote a poseless, it might be an issue.

Nell - maybe this should be on a per-session basis not a per-device basis. Desktop implementers - do you see this demand?

Kip - not really against this, but not seeing the demand - seems like most people handle this (in polyfill, framework, etc). Would we stop at a pose, or would we want content creating arbitrary devices...

Rafael - is the purpose is to simulate a magic window session - yes, “should we create non-immersive sessions with no sensors.”

Nell - the reasoning to include magic window can be extended to apply to this too.
Kip - I really like one way that works, with and without hardware. I’m kind of on the side of “be able to request a session and see if it’s backed by hardware”? Need to be careful that fallbacks don’t detect this falsely, etc…

Blair - having run into a lot of crappy vr sites that don’t fall back on desktop, it would be great if they DID work on desktop by default...

David - laptops do sometimes have accelerometers. Navigator.XR would always return something.

Nell: 1) can you get a session when there’s no sensor data to update its position. 2) can that sensorless session have UA affordances to physically navigate/provide DOFs in the session.

Blair - as long as you can detect what situation the system is in, … but there might be very different experiences, tailored to each of those scenarios. (e.g., John points out that sometimes you’d want the mouse to move around a scene, sometimes you’d want it to pivot around an object)

Kip - [points out this isn’t a “you can’t do this some other way” problem]

David - yep, if we had layers this would be challenging. Based on history, we’ve assumed non-immersive means 3DOF, but it could mean 6DOF.

Nell - plenty of discussion to continue on this topic. Still some confusion about the usefulness, and the difference between this and magic window. Still open question about whether adding the XR attribute to navigator means you’d always provide a Device or not.

<break>

Session topic - David to review design in
https://github.com/immersive-web/webxr/blob/session-creation/session-creation-explainer.md

Proposal is to separate coarse capability detection from creation.

Nell: in researching similar constructs, I looked at https://developer.mozilla.org/en-US/docs/Web/API/Media_Streams_API/Constraints. Seems like a similar case. The ordering of constraints is important here, and it has preferences vs exact needs. It's promise-based, and can fail out. This is part of a candidate rec, so it's probably worth digging in. There are numeric values and enum-like values. Not saying how user interaction or permission will play into this system.

Leonard Daly - I've just worked with this API for XSeen. The choice doesn't happen until applyConstraints. User approval is required to query the device. Once permission is granted (at least for Media) AND the page is HTTPS, then subsequent requests are auto-approved.

[point is made that fingerprinting isn't a concern here because it's post-permission.]

John - canPlayType is a similar call, but is pre-permission. There's such a dense matrix of support here, not sure if this matches the XR case (if 80% of the type devs are asking for the same type of constraints).

Rafael - we do have concerns about fingerprinting as well.

David - we should look more closely at this.
Kip - I like following patterns. I'm not sure if this model is the right one - returning a constrainable set might require firing up a runtime (which would be costly/bad).

John - some of these might be dynamic, or already in use.

Chris - points out this api doesn't do pre-creation capability testing, just if a given constraint type is something the system understands.

Nell: this is talking about the algorithm for selection, not just capability testing.

Kip - this has applyConstraints, which also has to happen in a user gesture, which helps prevent high-bandwidth fingerprinting. We might have a constraint that says "no user gesture required" e.g. for poseless.

Blair - I feel like user gesture is faux security. John: we'll talk over wine tonight.

Blair - I feel like lots of things choose device 1 and never offer device 2. Might be nice if there were explicit focus on enabling selection for the user.

John - the good thing is this can find the best device.

Blair - user can pick.

Chris - user picking by name is harsh (based on WebMIDI experience)

Nell - this should really be developer choice

David - one of the issues where this may cause issues is that ARKit and ARCore let you select plane detection when you create the session, but you can't turn it on and off - and it uses more power.

Nell - maybe we would want an applyConstraint like step.

Justin - would be nice if we could turn features on/off. sparse/dense point cloud, even AR vs VR.

Blair - when we were experimenting, we did actually sometimes do this - changing a property would cause a large-scale "reset" of the underlying ARKit system.

John - "can this device do this thing ever" "can this device do this thing right now" "can you change this thing on the device right now" are different questions.

Justin - Session was really to map to adapter startup/shutdown, but we've already kinda broken that. Sessions can mutate/morph to do what they need to do.

Nell - isn't Exclusive vs not or immersive vs not just an escalation of the session?

Blair - i kinda like "this is just session escalation" not sure what to do for things that do require a shutdown all the way back to zero, but maybe we're not building toward that.

Chris - getusermedia doesn't latch the camera - the API isn't necessarily exclusive mode.

Justin - many of those shutdown/startup or surprise "steam started up" things are mistakes that we've then backed away from.

David - the thing we have to solve for here is the structure. Does changing plane detection cause you to lose your anchors, etc.

Timing breakout

Brandon: The timestamp that's passed into the rAF is pretty well set, so the question now is what additional, XR specific values should we feed into the API that benefit the developer in runtime.

Lewis: Most important runtime value is how many times the frame has been reprojected and how much you've overrun your budget.

Artem: Seems hard to make use of. Only think Oculus API can report is how much GPU time it's internal presentation took.

Is the missed frame count a reasonable place to start? It's a simple value (just an int)

Lewis/Rafael: Easier than returning timing information, which may require fuzzing for Spectre-proofing.

Hard to use and CPU-based timing for decision making because, because you don't know how much time the UA is spending.

Lewis: Maybe there should be text in the explainer that says "don't rely on CPU timing for performance scaling"

Brandon: At least a missed frame count gives a rough value to track. Web Devs have been asking for a proxy for detecting refresh rate for a while.

Arterm: There's no way to detect that in WebGL?

Brandon: Nope. :P