# Spring 2023 Cross-Root ARIA F2F

https://github.com/WICG/webcomponents/issues/1005

## Attendees

- Anne van Kesteren (webkit @ apple)
- Benny Powers (design systems @ redhat, open-wc)
- Nolan Lawson (lightning web components @ salesforce, AOM)
- Brian Kardell (Igalia)
- Chris Holt (FAST @ microsoft, openui)
- Ryosuke Niwa (webkit @ apple)
- Westbrook Johnson (spectrum design system @ adobe, wccg, open-wc)

## Agenda

- Can we use CSS ::part ?
- What about encapsulation-preserving element references and semantic delegate?
- What about a global ID?
- Preventing "friendly fire" vs strong encapsulation – are they distinct use cases?
- Exporting a proxy/wrapper instead of the element itself
- Spillover topics from F2F:
    - Custom delegation attributes leobalter/cross-root-aria-delegation#5
    - Generic proposal (Potentially expand beyond aria?) leobalter/cross-root-aria-delegation#13
    - Reflection as a counterpart to delegation leobalter/cross-root-aria-delegation#14
    - Delegate something that isn't provided leobalter/cross-root-aria-delegation#21
    - Delegation proposal syntax leobalter/cross-root-aria-delegation#22
    - Reflection proposal: Expose internal elements to be referenced from the outside leobalter/cross-root-aria-delegation#23
    - Delegation attributes modification timing leobalter/cross-root-aria-delegation#24
- There is no way to navigate to a fragment inside a shadow tree #924
- Add delegatesLabel and a content attribute to specify the label element within a shadow tree #916
- Supporting <noscript> when FACE is in play #1004

## Can we use CSS ::part

Brian: Open UI were looking at a native element using parts.
Benny: Why not parts?

Chris: Open UI was looking at using parts in native elements, which may be a different consideration.
Anne: Would leveraging part give an aria value that would need special semantics
Anne: A part should have no special meaning
Anne: The value is completely in use land. Claiming that back is weird for such an API.
Benny: Imperatively listing parts for applying aria might not have that issue?
Anne: depends on what you're exposing. Wouldn't want to expose elements.
Benny: but parts expose this
Anne: but not the elements, just placing them for applying styles
Nolan: the subject comes up as the application of styling and the application of aria around the same time.
Benny: not all aria targetted items should be stylable…
Ryosuke: Expands on this in the context of a calendar element
Nolan: Points to a quality response that they are similar and some times overlap, but shouldn't be the right answer because they are not always the same.
**Using CSS Parts for applying Aria:** consensus these are not 100% the same use case, let's not conflate.

## What about encapsulation-preserving element references and semantic delegate?

Nolan: "semantic delegate" already in experimentation in Chromium
Nolan: explains the larger API proposal
Nolan: a subset of cross-root aria, in a way
Nolan: proposed to break through the boilerplate of the cross-root aria proposal and focus on the use case where the custom element has a one to one relationship with a child in its shadow root.
Nolan: shares example code
Ryosuke: does the focus here make it not workable?
Chris: sees it as an initial offering, specifically for teams focusing on design systems
Ryosuke: should these be separate?
Chris: How can we delegate everything? Reduce pains for devs.
Benny: If we relax the rule that focusable elements may not be aria-hidden, then there is a path to doing some of this imperatively with Element Internals. Advantage of not adding new things and being able to continue to iterate of CrossRoot aria.
Brian: Focuses on a large numer of element that have this focus.
Nolan: Wants to know more about Benny's point.
Benny: Input in Element that is form associated. Even with aria-hidden=true/role=presentation the browser will assume the input is required to be part of the accessibility tree. This would work because the host is form associated and the input in the shadow doesn't need to be related to the form/logo around element.
Anne: not sure how we'd be able to get there. It needs to focus.
Benny: The host _might_ take this role/respoonsibility rather than shipping a role=input inside a role=input.

Brain: Is there a bug for this?
Benny: Not yet, but it's a feature request. (editor: bug filed on WICG/webcomponents #1014)
Benny: browser doesn't allow this because the input _should_ be surfaced, but the custom element does surface it instead.
Westbrook: brings back to "semantic delegate"
Westbrook: outlines a little clearer that "semantic delegate" builds up to "cross-root aria" so that the two APIs don't diverge
Ryosuke: yes, but we'd need to start from the complex place and work back, rather than starting from the simple place and working up
Nolan: we're getting close to an agreement of starting from the complex API spec and the taking the simplified path to get started
Anne: unless there is some other form of ElementInternals usage we could unlock here
Nolan: that makes sense, Ben (Chromium implementor) started with the simple because it seemed to unlock some discussion and capability sooner
Chris: yes, that's the basis of his approach.
Anne: FACE can be target of label, already?
Chris: yes, but it's the internals that cause issue
Anne: presumable the host can delegate to active a child?
Nolan: seems like FACE and ElementInternals are specifically target towards owning and building everything and the APIs are really already there.
Nolan: If I don't want to rebuild the world, that's where we really run into the needs for the aria passing work.
Nolan: the mix is difficult but often comes to brear in actual development
Chris: agrees
Anne: does it need to be FACEsemeanticDelegate then semanticDelegate is very generic
Brian: I'll take the feedback to Alice
Chris: Focus here makes sense, would that allow for it to get there faster?
Chris: what non-FACE elements would this happen with?
Anne: possibly with composite controls
Brian: composite controls are why things get scary and are interested in something simple
Westbrook: composite controls will come, do we want to over scope this API so that it can't build up to something.
Brian: no API answers all the problems, to date.
Anne: we do have the face API already, so it might align, but we have yet to lock down the scope
Brian: Good feedback about the scope to FACE in some way for continuing exploration.
Chris: The only thing with FACE is that I can't delegate to an <a> which would seems like it would also have the same considerations as the other elements which can be Form Associated
Ryosuke: super simple versions of this are solved with `is=`
Benny: `is=` doesn't give us shadow roots on all the things
Anne: so no facy
Brian: so maybe it's time to get back to 2014 and mixins!
**Table "semantic delegate"**

## encapsulation-preserving element references

Nolan: overview of the proposal
Nolan: setting `aria…Element` with an element in a deeper shadow root
Nolan: setting "works" but getting returns the "retargetted" element (in shared document)
RN: so this is totally imperative
Nolan: declarative syntax may be tbd
Ryosuke: status quo? Element Internals?
Nolan: EI is for defaults, this allows for consumers to set values on top of that. - given two references across different roots this allows me to imperatively hook them up
RN: we need a declarative syntax to solve cross root aria in the long term, let's not invest in a script-only solution when other proposals may address html as well
NL: long term yes, but short term: we rely on javascript 100% (spa). If we had this imperative access it would solve all our problems today and allow us to drop our bespoke ShadyDOM-like polyfill. We would like to solve a declarative syntax eventually. Both alternatives suffer from the bottleneck effect, whereas this proposal isn't limited to the shadow host's mediating across roots.
AVK: similar to part, in cross-root aria we had named retargetting. A CE could be the target for multiple aria props, but b/c they were named, the delegations can be targetted at fine grain. RN and i discussed this some months back. (in other words, bottle neck problem is overstated). As well, this proposal does not allow you to refactor encapsulated content because highly coupled external dependencies are at play. We also have these newly-valueless attrs like `aria-activedescendant` which might break pages backwards [sic]
WJ: doesn' existing use of ariaFooElement and other IDL APIs leave dangling attrs? True we can't yet cross boundaries. If we set one of those with the element IDL, it will reflect with a valueless attr, no?
AVK: correct we have no html syntax reflection yet, but on the API side it works fine
NL: most recent HTML spec: when you set an IDL prop, the element sprouts an attr, but without trying to infer the ID - empty string. This was a compromise. Alice proposed this so that selectors can target it, but there's no id mapping to the declarative syntax.
AVK: i was talking about the IDL attribute hanging, not the content attr
WJ: on line 2 of the second snippet in the proposal, it's returns the currently assigned element
AVK: ok this isn't as bad as i initially felt, but it's not great as it doesn't allow you to change the shadow tree without breaking consumers. You need a global view to make shadow changes. This breaks encapsulation.
WJ: ok, so 2nd question: wrt to the custom option list, where values will be changing, what way does this empower the consumer of custom-option-list to make the connections with this api?
NL: currently our shadow dom polyfill lets users pass ids around. Outside of web components, in framework land, this is how web developers typically do these things. We'd probably want to set up a mediator that handles the relationships between shadow dom and light dom
AVK: so then let's standardize the mediator in the platform
WJ: agreed, rather than running a bunch of imperative IDL at each `change` event. We'd rather the browser do those mappings. The "implied simplicity" here is tempting, but i'm not convinced this fully "makes native" the things we want to make native here.

NL: happy to talk about a mediator or proxy primitive. Instead of passing the option element, maybe the platform passes a proxy for the developer.
**RN, WJ: let's timebox and move on**

## Global ID

NL: it's easy to pass references in an imperative API. declarative needs namespaces (unpopular), but maybe we should just introduce global ids.
WJ: as Brian (the champion) is not on the call, let's move on

## Friendly Fire

NL: Brian not on call

## Proxy

WJ: let's create an issue for this subject.
NL: graciously agrees to be volunteered for this.

## Custom delegation attributes

BP: ideas from this issue have been addressed elsewhere, let's move on

## Generics Proposal

NL: spec covers role + aria*, but doesn't cover other idref attrs e.g. `for`, `popuptoggletarget`, `list` on input, svg use and href
RN: whichever solution we come up with, if we can extend it to other idrefs, that's a good idea, but we should be careful of scope creep or else we'll be unable to solve anything.
NL: what about constraining the discussion to `for`
RN: sounds like a FACE issue
AVK: yeah but if we had a generic cross-root solution, `for` as well as the rest could be brought in.
AVK: let's solve aria first and then see if that solution works here
RN: agreed
AVK: and if not we can fall-back on FACE to clean up the remaining cases. FACE is nice, except painful to set up.

## Reflection as a counterpart to delegation

AVK: only about 8 aria properties reflect to attributes. Let's determine first why those act the way they do before we move on.
NL: when this was first proposed, there were 2 separate explainers. They've now been combined into one explainer. delegate/reflect is also called import/export.

WJ: the longer we discussed it, the less realistic it felt to solve either delegation or reflection without its counterpart. (see proposal snippets). Alternatives to a proliferation of autoariafoo autoariabar could be autoaria="foo,bar"
AVK: how does this work in many-to-many relationships? You'd need something like named parts, with some kind of mapping on the boundary, perhaps in ElementInternals or as a public api on the element.
AVK: generally this example makes sense to me. I'd prefer to see this on element internals not on shadowroot
WJ: to clarify: attrs shown on #shadow-root are assumed to either be on the template[shadowrootmode] or imperatively in ElementInternals.
AVK: in other words,if we have the declarative api, what does the imperative api look like?
WJ: in attachShadow({ delegatesAriaAttributes }))
AVK: still very declarative. How do you set the internal semantics of the host element to which the shadow tree belongs? Consider ARIAMixin, which is mixed in the EI. the difference between setting stuff on EI and on the element itself is that EI is internal, but `<h1 role=heading>` (for example) is a public api. EI corresponds to "native semantics" in ARIA.
AVK: perhaps because it's 'delegated', then it's not 'native' - since they are relationships, we can go about it by focusing on what is delegated
Benny: something like `this.internals.delegatesAriaTarget = this.shadowRoot`
Nolan: not really convinced that Element Internals is the place for this. The Combobox example show a dual purpose element that might have all of these things rather than a specific role as would be applied via Element Internals. There would be internal semantics but not host semantics.
RN: let's review scenarios in which CEs participate in the AX tree

1. `fancy-button`. Same as button but fancy
2. `combo-box`. Host has a specific role, shadow root contains multiple elements within. Needs to hook up specific aria props across roots
3. `novo-element`: Novel concept where there even isn't an established role

AVK: with fancy-input and fancy-listbox, which compose as siblings to form a combobox, wouldn't we want the public api to be `for` and use an imperative api to hook up the relationships?
NL: if i was using EI, i'd set a group or region role.
AVK: ARIA shouldn't *really* be a public api, it should be a behind-the-scenes implementation detail. "Aria" is seen as a gap in html. Custom-elements should be custom, and not expose the workaround that is aria. We don't want end users to have to deal with those things. It gets ugly and unwieldy fast.

# Conclusions

RN: we need to continue at a separate session. I'll be in US in JULY, but if the rest can join in, i'll join then, or alternatively schedule with EU hours.
NL: counter proposals don't have so much interest, we conclude that cross-root aria proposal is most interesting.

AVK: yes let’s design nicer bridges on top of aria

## Not Covered:

- Delegate something that isn't provided
- Delegation proposal syntax
- Reflection proposal: Expose internal elements to be referenced from the outside
- Delegation attributes modification timing