

Improving Notification Handling with AriaNotify

Alison Maher / Microsoft Edge
(almaher@microsoft.com)

What is an AT?

- AT = Assistive Technology
- “Products, equipment, and systems that enhance learning, working, and daily living for persons with disabilities.”
- Examples:
 - Screen readers (JAWs, NVDA, etc)
 - Braille displays
 - Voice driven ATs (Voice Access, Voice Control, etc)
 - Magnifier
 - High Contrast Mode
 - And many others

ATs and Notification Use Cases

- Difficult to convey to blind or low vision users:
 - Confirmation of action (ex: “bold on”, “bold off”)
 - Current presence (ex: “John Doe entered the meeting”, “Present state set to do not disturb”)
 - Reactions (ex: Raised hands, applause or thumbs up reactions in video meetings)
 - Error messages
 - Etc

Available solutions: ARIA Live Regions

- Set `aria-live` on an element that has dynamically changing content
- Will notify the user when the content changes
- Values
 - `off` - changes should not be announced
 - `polite` - important information for user, but not urgent
 - `assertive` - immediate attention required

- Example:

```
<form name="signup" id="signup">  
  <p id="errors" aria-live="assertive"></p>  
  ... <Other form elements> ...  
</form>
```

ARIA Live Regions Pain Points

- Can only be used with dynamically changed content
- Not author ergonomic for other types of notifications like confirmation of action
 - Authors mimic dynamic updates offscreen to accomplish these notifications today
 - This means this text can be separately discoverable by screen reader users with their virtual cursor
 - Site-wide live region systems often manage and insert live regions in a specific area of the DOM (often at the end), which can fail when users open a modal dialog.
- Live regions are inconsistently implemented
- Live regions lack richness (no context and primitive prioritization)

Other Problems with Notifications

- Screen reader users are frustrated with irrelevant interruptions and want more control
- Notifications are often overused and noisy
- Some ATs will filter these out, which leads to missed information

Solution Goals

1. Improve end user experience that is consistent and predictable, where users have more control over how they consume notifications
2. Empower screen readers developers by providing clear expectations and context
3. Empower web developers with improved notification handling ergonomics that is consistent and allows for more fine-tuning that improves end user experiences

Solution: ariaNotify

- [Explainer](#)
- New ariaNotify() JS API that allows authors to directly tell screen reader what to read
- In its most basic form, an author provides a string to be read to the AT

```
document.ariaNotify("bold on");
```

Notification source

- Notifications are sent on behalf of a specific source (document / element)

```
document.ariaNotify("bold on");
```

```
document.querySelector("#foo").ariaNotify("bold on");
```

(In scope for v1)

Notification priority

- Each notification can have a `priority` of `high` or `normal`
- Default is `normal`
- Determines where in the output queue the string should be added

```
document.ariaNotify( "Error connecting to server.",  
    { "priority": "high" } );
```

(In scope for v1)

Interrupt property

- Each notification can have an `interrupt` of `none`, `all`, or `pending`.
- Determines what should be interrupted from previous notifications sent to the AT
- ***Out of scope for V1 of API***

```
document.ariaNotify( "Uploading file untitled-1 to the  
cloud.", { "interrupt": "pending" });
```

Interrupt property

- **none**
 - Add the current notification, allowing all items to be spoken
- **all**
 - Add the current notification
 - If currently speaking a notification from same source, and priority is of equal or lesser importance, silence it
 - Flush any pending notifications of the same source and equal or lesser priority
- **pending**
 - Add the current notification
 - Allow any currently speaking notification to complete
 - Flush any pending notifications of the same source and equal or lesser priority

(*Out of scope for v1*)

Notification Type

- Each notification can have a non-localized `type`
- Provides context to the screen reader, allowing users flexibility on how the notification is handled (filter / speech vs. braille / sound effects, etc.)
- ***Out of scope for V1 of API***

```
document.ariaNotify( "Audience applause",  
  { "type": "reaction" });
```

Fallback mechanism

- If the platform doesn't support ariaNotify, the browser will fallback to live regions
 - `priority` – maps to polite / assertive
 - `interrupt` – ignored
 - `type` – ignored
- Won't apply to IA2 on Windows (just UIA)

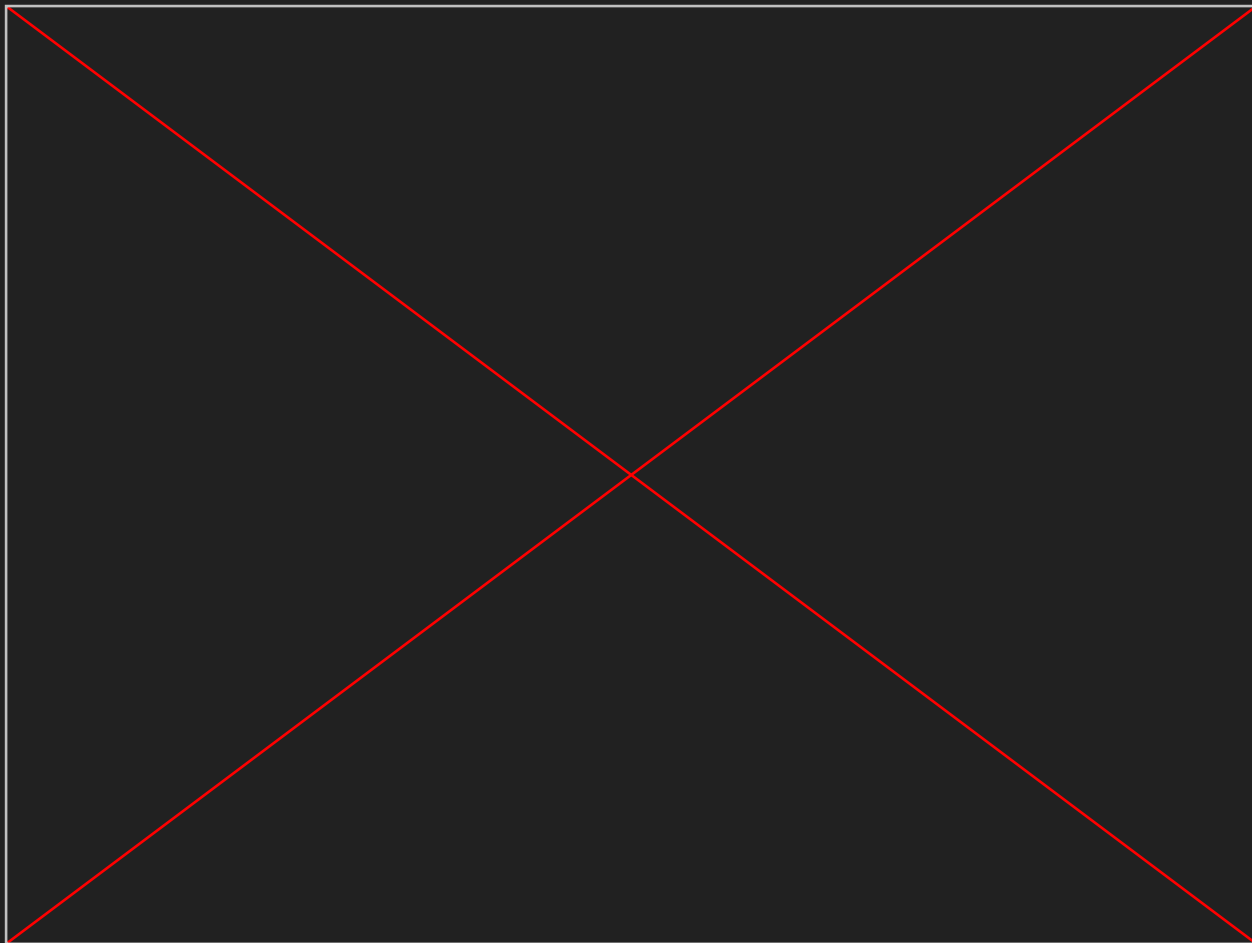
AriaNotify in iFrames

- Create new `Permissions-Policy` called `aria-notify`.
- By default `aria-notify` is allowed in iframes.
- Authors can opt iframes out of `aria-notify` by
 - `<iframe allow="aria-notify 'none'">`
 - `Permissions-Policy: aria-notify=(self)`
- Open WHATWG issue: <https://github.com/whatwg/html/issues/11004>

Current Status

- Prototyped in Chromium behind feature flag on Windows, MacOS, Linux and Android
- Test out v1 with the following flags:
 - `--enable-blink-features=AriaNotify`
 - `--enable-features=UIAProvider`
- Test out v2 with the following flags:
 - `--enable-blink-features=AriaNotifyV2`
 - `--enable-features=UIAProvider`
- [Draft spec](#) in progress by Github team
- [Polyfill](#) published by Github team

Demo Video



Next Steps

- HTML spec change needed for new iFrame permission policy
- Work to publish Aria and HTML spec updates
- Work with ATs on consistent support of ariaNotify (and live regions)
- Ship UIA in Chromium
- Start Origin Trial in Edge/Chromium
 - Goal: April/May timeframe

Thank you!

- Thank you to the following contributors
 - Evelynn Kaplan, Ethan Jimenez, Doug Geoffray, Keith Cirkel, Clay Miller, Andy Luhrs, Sara Tang
- Thank you to the following for reviewing the API and for their advice and guidance throughout the project thus far
 - Sarah Higley, Scott O'Hara, Aaron Leventhal, Jamie Teh, James Craig, Brett Humphrey, Benjamin Beaudry, and more!
- Thank you to the original explainer authors
 - Travis Leithead, Daniel Libby

Questions/Feedback?