Spinning ARIA Around

Eric Eggert, Knowbility · AccessU · May 2019

Eric Eggert

Eric Eggert is a Web Developer and Teacher who works with Knowbility and the W3C on improving the Web for People with Disabilities, and everyone else.

Knowbility

W3C®

Important Note:

This presentation contains ARIA examples that are preventing websites and applications from being accessible.

Don’t copy & paste mindlessly.

WAI-ARIA:
Accessible Rich Internet Applications

Internet Applications

Internet Applications

ARIA ≠ 🧙‍♀️ || 🧙‍♂️

No

```
aria-make-accessible="true"
```

5 Rules of ARIA[1]

[1] *Actually not harmful, but really, what is the point?!

1

If you can use a native HTML element or attribute with the semantics and behavior you require **already built in**, *instead of re-purposing an element and adding an ARIA role, state or property to make it accessible,* **then do so**.

Do not change native semantics, unless you really have to.

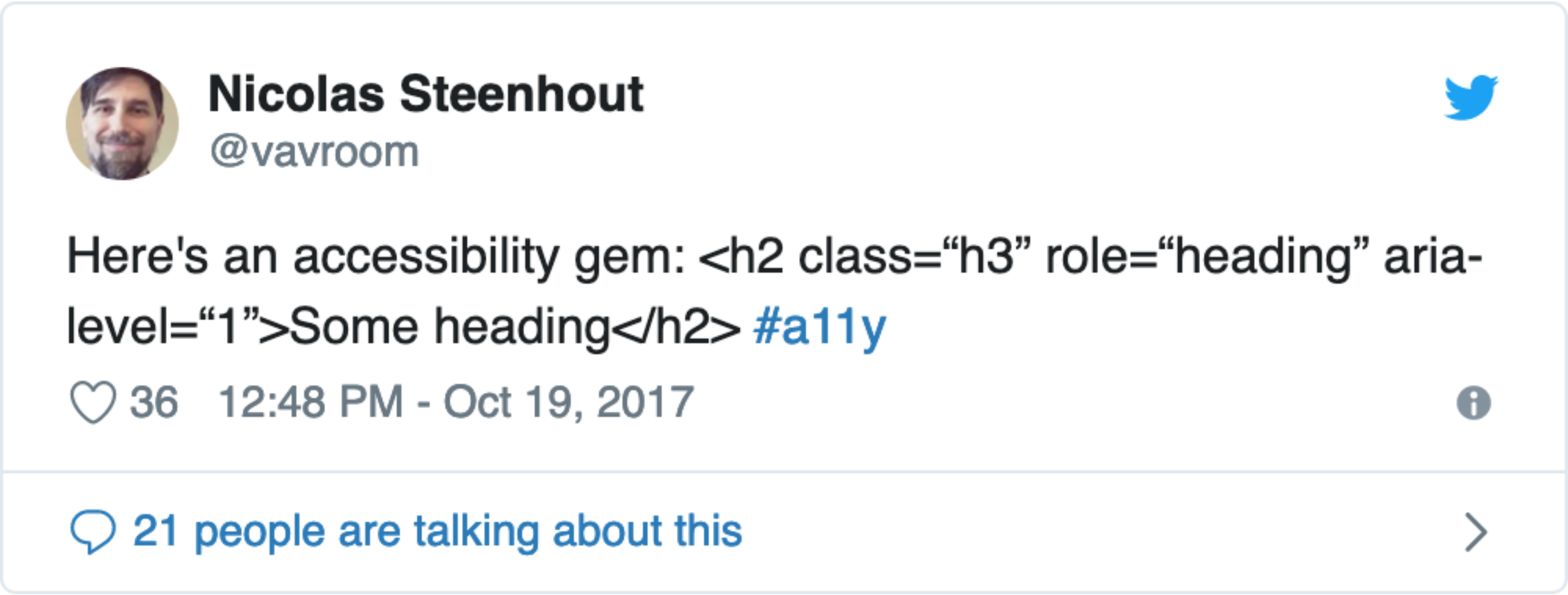
2

Bad Example:

```
<h1 role="button">heading button</h1>
```

Good Example:

```
<h1><button>heading button</button></h1>
```



Nicolas Steenhout
@vavroom

Here's an accessibility gem: <h2 class="h3" role="heading" aria-level="1">Some heading</h2> #a11y

36 12:48 PM - Oct 19, 2017

21 people are talking about this

All interactive ARIA controls must be usable with the keyboard.

3

If you create a widget that a user can click or tap or drag or drop or slide or scroll, a user must also be able to navigate to the widget and perform an equivalent action using the keyboard.

All interactive widgets must be scripted to respond to standard keystrokes or keystroke combinations where applicable.

3

Example

If using role=button the element must be able to receive focus and a user must be able to activate the action associated with the element using both the enter (on WIN OS) or return (MAC OS) and the space key.

— **ARIA Practices Guide**

DEMO

Do not use `role="presentation"` *or* `aria-hidden="true"` *on a focusable element.*

Using either of these on a focusable element will result in some users focusing on 'nothing'.

4

All interactive elements must have an accessible name.

5

Some Roles

```html
<div role="main"> <!-- better: <main> -->

<form role="search">

<div role="tooltip">

<output role="alert">
```

Some Properties and States

```
<input aria-required="true">

<input aria-labelledby="label" aria-describedby="errormsg">

<button aria-expanded="false">

<div aria-hidden="true">

<div aria-live="assertive">
```

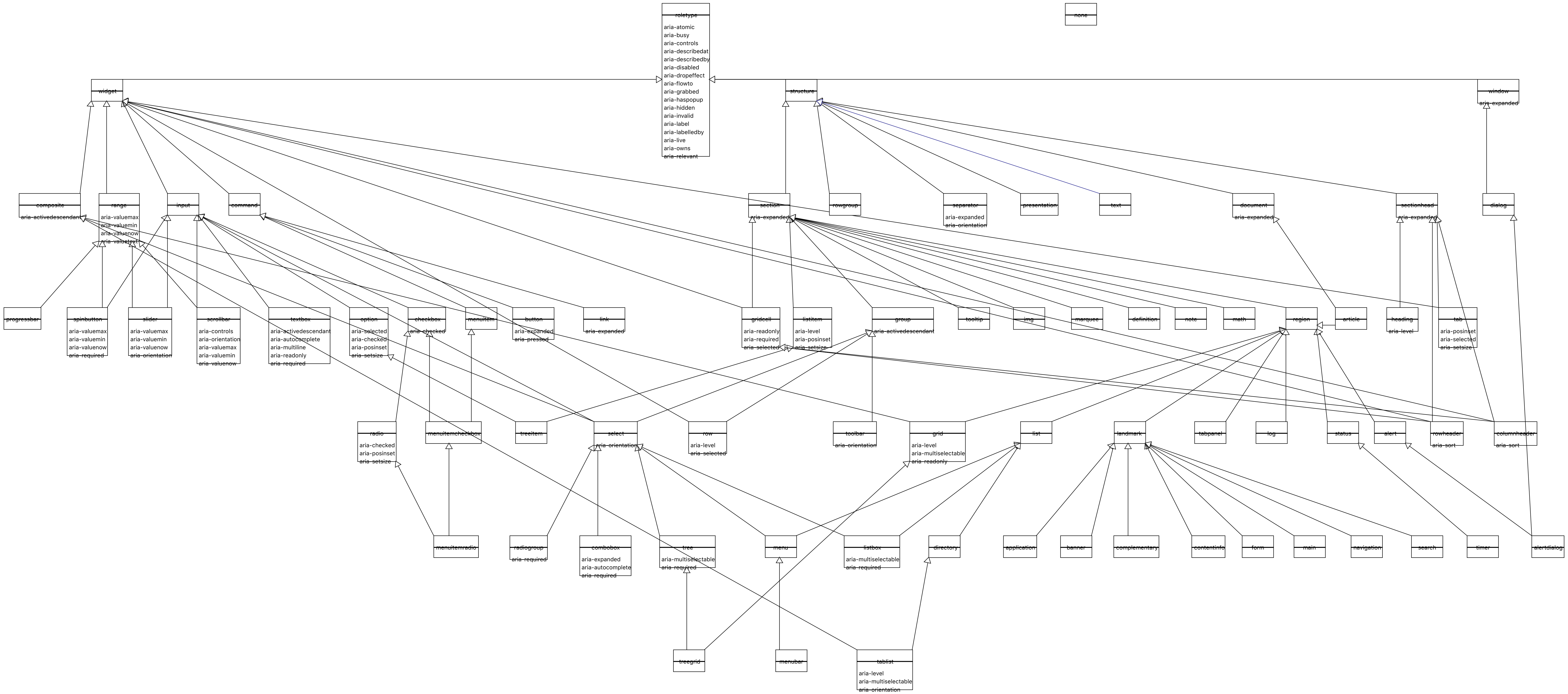
Live Regions

```
<div aria-live="off">

<div aria-live="polite">

<div aria-live="assertive">
```

ARIA **Support**

Heydon Pickering:

`aria-controls` is 💩


roletype
aria-atomic
aria-busy
aria-controls
aria-describedat
aria-describedby
aria-disabled
aria-dropeffect
aria-flowto
aria-grabbed
aria-haspopup
aria-hidden
aria-invalid
aria-label
aria-labelledby
aria-live
aria-owns
aria-relevant
none
widget
structure
window
aria-expanded
composite
aria-activedescendant
range
aria-valuemax
aria-valuemin
aria-valuenow
aria-valuetext
input
command
section
aria-expanded
rowgroup
separator
aria-expanded
aria-orientation
presentation
text
document
aria-expanded
sectionhead
aria-expanded
dialog
progressbar
spinbutton
aria-valuemax
aria-valuemin
aria-valuenow
aria-required
slider
aria-valuemax
aria-valuemin
aria-valuenow
aria-orientation
scrollbar
aria-controls
aria-orientation
aria-valuemax
aria-valuemin
aria-valuenow
textbox
aria-activedescendant
aria-autocomplete
aria-multiline
aria-readonly
aria-required
option
aria-selected
aria-checked
aria-posinset
aria-setsize
checkbox
aria-checked
menuitem
button
aria-expanded
aria-pressed
link
aria-expanded
gridcell
aria-readonly
aria-required
aria-selected
listitem
aria-level
aria-posinset
aria-setsize
group
aria-activedescendant
tooltip
img
marquee
definition
note
math
region
article
heading
aria-level
tab
aria-posinset
aria-selected
aria-setsize
radio
aria-checked
aria-posinset
aria-setsize
menuitemcheckbox
treeitem
select
aria-orientation
row
aria-level
aria-selected
toolbar
aria-orientation
grid
aria-level
aria-multiselectable
aria-readonly
list
landmark
tabpanel
log
status
alert
rowheader
aria-sort
columnheader
aria-sort
menuitemradio
radiogroup
aria-required
combobox
aria-expanded
aria-autocomplete
aria-required
tree
aria-multiselectable
aria-required
menu
listbox
aria-multiselectable
aria-required
directory
application
banner
complementary
contentinfo
form
main
navigation
search
timer
alertdialog
treegrid
menubar
tablist
aria-level
aria-multiselectable
aria-orientation

WAI ARIA is **really** Complicated!

Makes it also complicated for Web Developers!

Exhibit 1 – **Bad!**

```html
<a id="airline-logo" href="…"
   class="logo"
   aria-label="Airline Name">
   &nbsp;
</a>
```

Exhibit 1 – **Better!**

```
<a id="logo" href="…">
   <img src="…" alt="Airline Name">
</a>
```

or:

```
<a id="logo" href="…">
   <svg>
     <title>Airline Name</title>
     …
   </svg>
</a>
```

Exhibit 2 – **Bad!**

```html
<div class="nav">
  <a href="javascript:void(0);" class="navInactive" role="button">
    <span class="hiddenText">Slide 1</span>
  </a>
  <a href="javascript:void(0);" class="navActive" role="button">
    <span class="hiddenText">Slide 2</span>
  </a>
  <a href="javascript:void(0);" class="navInactive" role="button">
    <span class="hiddenText">Slide 3</span>
  </a>
</div>
```

Exhibit 2 – **Better!**

```
<nav>
  <button aria-pressed="false" aria-label="Slide 1">
     &nbsp;
  </button>
  <button aria-pressed="true" aria-label="Slide 2">
     &nbsp;
  </button>
  <button aria-pressed="false" aria-label="Slide 3">
     &nbsp;
  </button>
</nav>
```

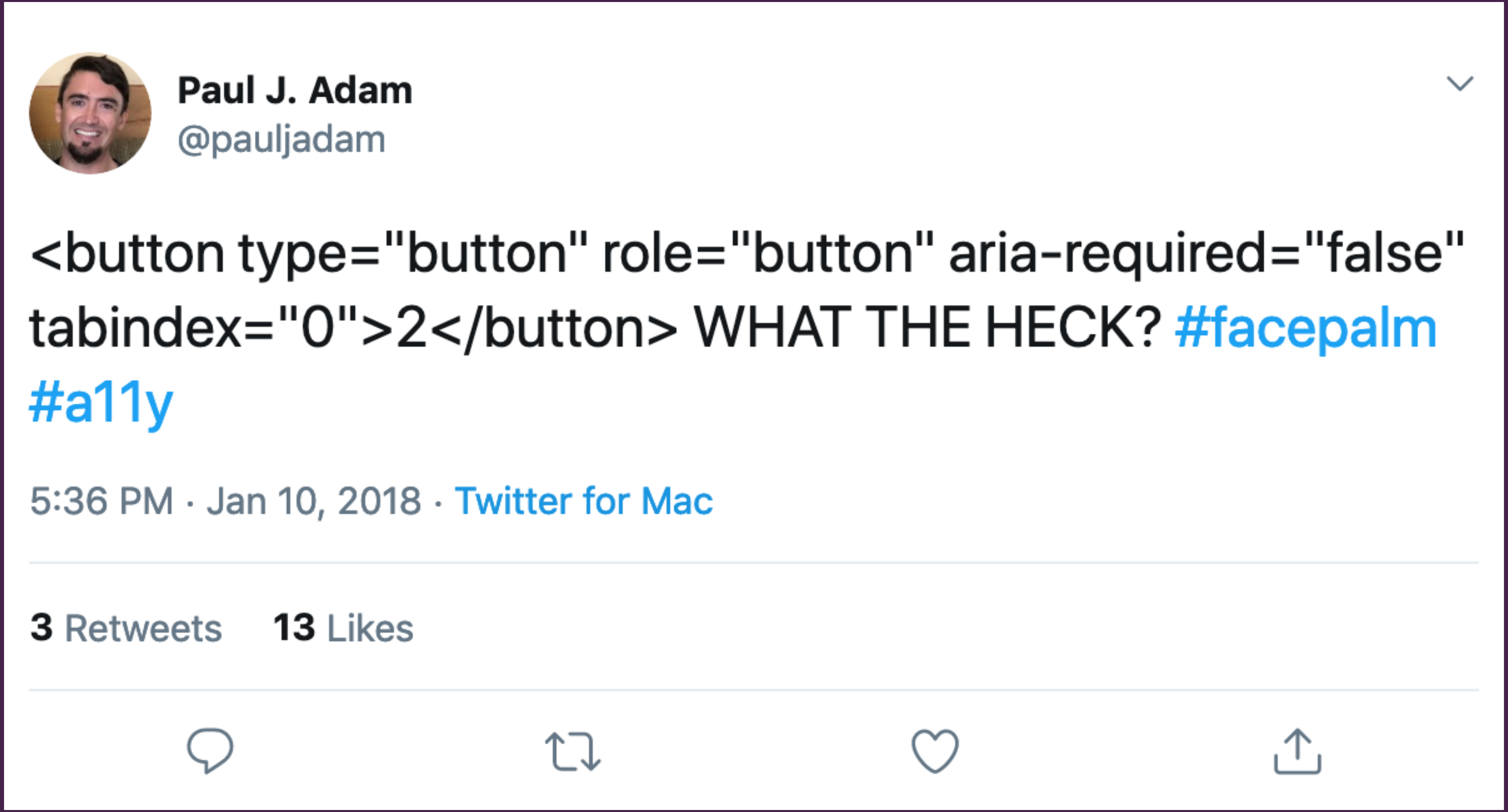
Exhibit 2 – **Even Better!**

```html
<nav>
  <button aria-pressed="false">
   <img src="…" alt="Slide 1">
  </button>
  <button aria-pressed="true">
   <img src="…" alt="Slide 2">
  </button>
  <button aria-pressed="false">
   <img src="…" alt="Slide 3">
  </button>
</nav>
```

Exhibit 3 – **Bad!**

```
<th
 tabindex="0"
 role="button"
 aria-label="Sort column"
>Name</th>
```

Exhibit 3a – **Bad!**



Paul J. Adam
@pauljadam

<button type="button" role="button" aria-required="false" tabindex="0">2</button> WHAT THE HECK? #facepalm #a11y

5:36 PM · Jan 10, 2018 · Twitter for Mac

3 Retweets **13** Likes

Exhibit 4 – **Bad!**[1]

```
<span
  aria-hidden="true"
  role="img"
  class="icon">
</span>
```

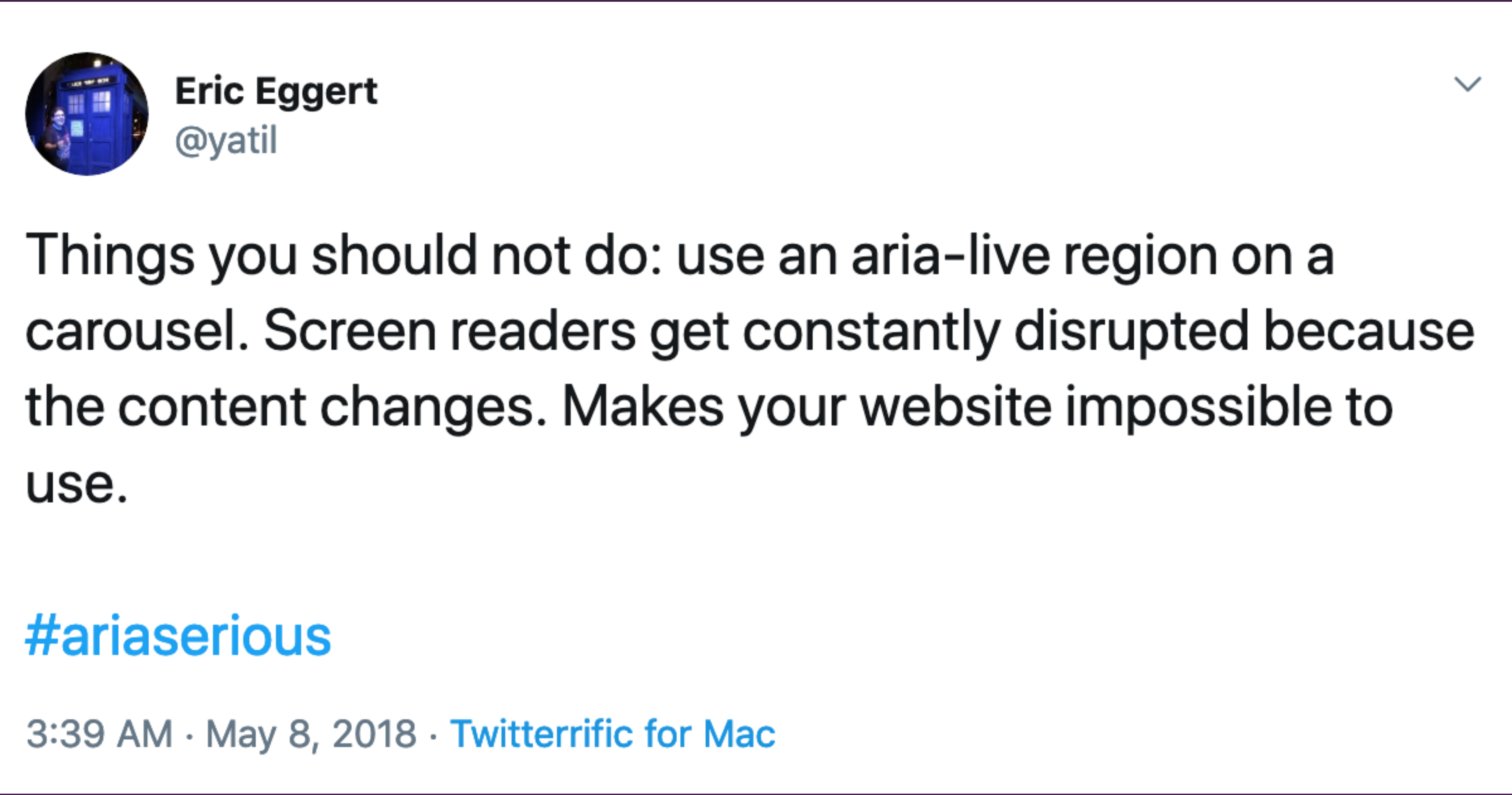
[1] *Actually not harmful, but really, what is the point?!

Exhibit 5 – **Really Bad!**

```
<body aria-hidden="true">
```

Exhibit 6 – **Bad!**

```
<a
  href="…"
  target="_blank"
  title="Click here to
         view the video."
  tabindex="-1"
  role="button"
  aria-label="External link"
></a>
```

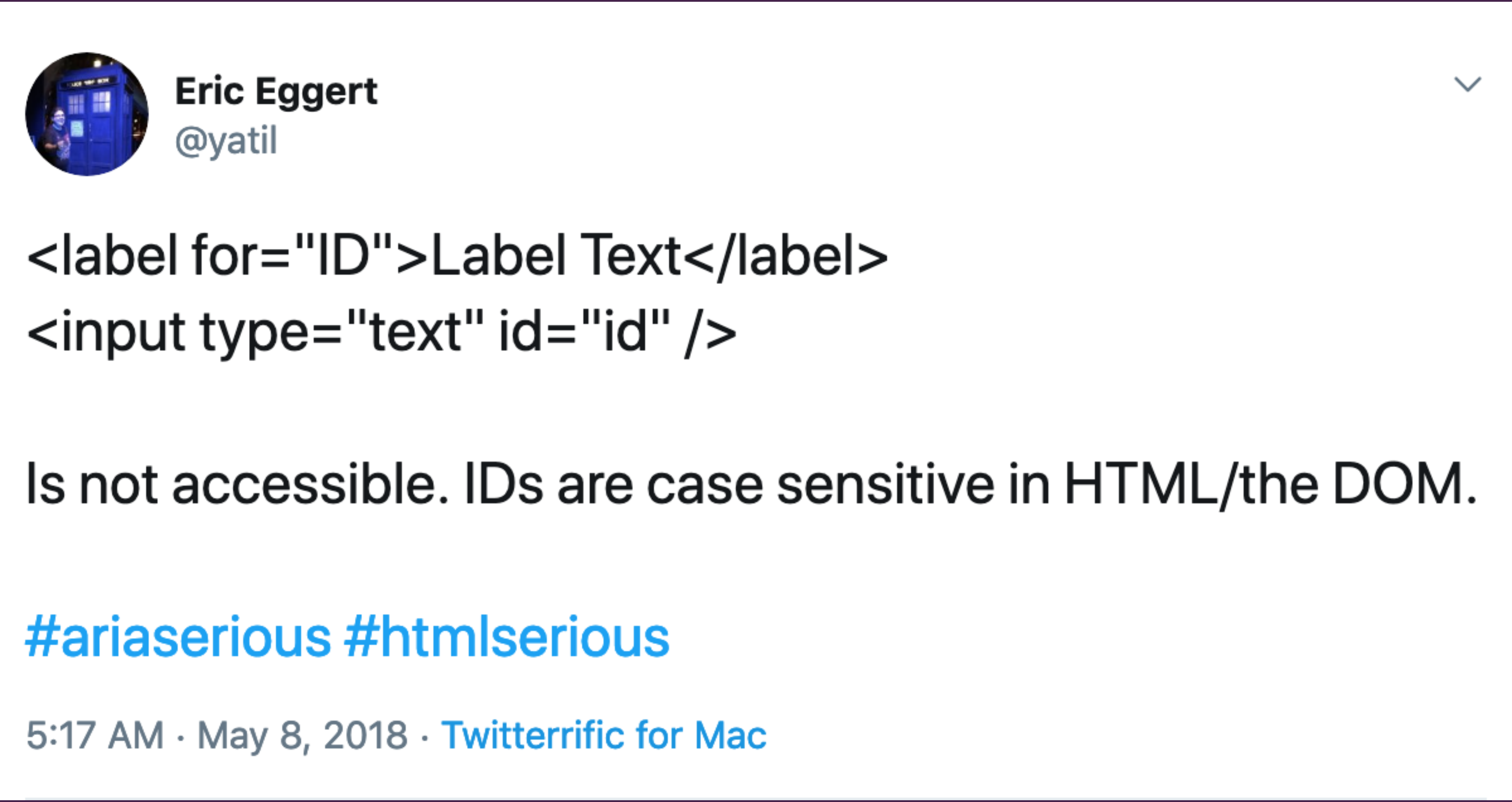


Eric Eggert
@yatil

Things you should not do: use an aria-live region on a carousel. Screen readers get constantly disrupted because the content changes. Makes your website impossible to use.

#ariaserious

3:39 AM · May 8, 2018 · Twitterrific for Mac



Eric Eggert
@yatil

```
<label for="ID">Label Text</label>
<input type="text" id="id" />
```

Is not accessible. IDs are case sensitive in HTML/the DOM.

#ariaserious #htmlserious

5:17 AM · May 8, 2018 · Twitterrific for Mac

W3C Working Group Note

WAI-ARIA Authoring Practices 1.1

W3C Working Group Note 07 February 2019

This version:
https://www.w3.org/TR/2019/NOTE-wai-aria-practices-1.1-20190207/
Latest published version:
https://www.w3.org/TR/wai-aria-practices-1.1/
Latest editor's draft:
https://w3c.github.io/aria-practices/
Previous version:
https://www.w3.org/TR/2018/NOTE-wai-aria-practices-1.1-20180726/
Editors:
Matt King (Facebook)
JaEun Jemma Ku (University of Illinois)
James Nurthen (Adobe)
Michiel Bijl (Invited Expert)
Michael Cooper (W3C)
Former editors:
Joseph Scheuhammer (Inclusive Design Research Centre, OCAD University) (Editor until October 2014)
Lisa Pappas (SAS) (Editor until October 2009)
Rich Schwerdtfeger (IBM Corporation) (Editor until October 2014)

Abstract

This document provides readers with an understanding of how to use *WAI-ARIA 1.1* [WAI-ARIA] to create accessible rich internet applications. It describes considerations that might not be evident to most authors from the WAI-ARIA specification alone and recommends approaches to make widgets, navigation, and behaviors accessible using WAI-ARIA roles, states, and properties. This document is directed primarily to Web application developers, but the guidance is also useful for user agent and assistive technology developers.

This document is part of the WAI-ARIA suite described in the WAI-ARIA Overview.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at https://www.w3.org/TR/.

W3C Working Draft

TABLE OF CONTENTS

Using ARIA

W3C Working Draft 27 September 2018

This version:
https://www.w3.org/TR/2018/WD-using-aria-20180927/

Latest published version:
https://www.w3.org/TR/using-aria/

Latest editor's draft:
https://w3c.github.io/using-aria/

Previous version:
https://www.w3.org/TR/2018/WD-using-aria-20180924/

Editors:
Steve Faulkner (The Paciello Group)
David MacDonald (CanAdapt Solutions Inc.)

Participate:
GitHub w3c/using-aria
File a bug
Commit history
Pull requests

Abstract

This document is a practical guide for developers on how to add accessibility information to HTML elements using the Accessible Rich Internet Applications specification [WAI-ARIA-1.1], which defines a way to make Web content and Web applications more accessible to people with disabilities. This document demonstrates how to use WAI-ARIA in [HTML51], which especially helps with dynamic content and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies.

Status of This Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in

Conclusion

✅ Landmarks

✅ States & Properties

⚠️ (Widget) Roles — Mind the Keyboard