# Future course revisions

[Future course revisions 1](#_Toc1504387016)

[An effort flag precedes the task title (1 least, 3 most) 1](#_Toc1517669345)

[1 - Add a link to post a Github issue 1](#_Toc2060118721)

[1 - Remove excess left margin from left shoulder menu 1](#_Toc1621823960)

[2 - Review the JS in the course to see where we can replace it with :has() 2](#_Toc183845817)

[1 or 3 - Add "separator" role to hidden begins/ends 3](#_Toc1034832203)

[1 - Move Expand All/Collapse to below the TOC on All Best Practices 4](#_Toc113943366)

[1 - Update Mod 2 > Landmarks with links to mod 12 "application" and "document" roles 4](#_Toc1423032817)

[1 – Update Mod 2 > Landmarks > aside 4](#_Toc722981686)

[2 – Update Mod 2 > Headings with <hgroup> element 5](#_Toc431791135)

[3 - Update Mod 2 > Content structure with details/summary, barriers and button solution 7](#_Toc2126854500)

[2 - Expand mod 2 > Content structure > Article 7](#_Toc614301507)

[2 - Expand mod 3 > Navigation > Character key shortcuts 9](#_Toc1446813299)

[1 – Update Mod 3 > Navigation > Focus order 9](#_Toc1314490302)

[2 - Update Mod 3 > Changing context > On input with examples 10](#_Toc1376644488)

[1 - Fix phrasing in mod 3 > Visible focus indicator > Best practice (and All BP) 11](#_Toc626644225)

[1 – Add CSS style rules to mod 3 > Links > Good example: Warns users link opens 11](#_Toc1232196632)

[1 – Mod 4: Fix id/headers example and add method 11](#_Toc1855989298)

[1 – Mod 4: fix placement of <colgroup> 12](#_Toc1499123193)

[3 - Update mod 5 or 7 to address SVG icons 12](#_Toc1084589734)

[2 - Expand mod 6 > labelling controls to include nesting a control in a label 13](#_Toc1376787049)

[2 - Expand mod 6 > User notifications to include aria-live 13](#_Toc53270599)

[3 - Update mod 6 > Identify input purpose with autocomplete=”on|off” method 13](#_Toc1836361290)

[2 - Expand mod 6 > Multi-step forms with example of setting focus 13](#_Toc713575286)

[3 - Update Mod 7 > colour with three examples 13](#_Toc1341943488)

[3 - Update mod 7 to address "High contrast mode" (or throughout course) 15](#_Toc2015816044)

[3 - Update mod 9 > Multi-step form > Approach 3 with WET progress meter 16](#_Toc638936578)

[1 - Rephrase mod 11 > ARIA live explanation 16](#_Toc1803085613)

[1 - Test mod 11 > ARIA live example 16](#_Toc1876627877)

[1 - Update mod 11 > ARIA live links from ARIA 1.1 to ARIA 1.3 16](#_Toc578171726)

[2 - Update mod 11 > ARIA live to use WET "Alert" pattern 16](#_Toc2055940284)

[2 - Update mod 11 > ARIA live to mention and link to mod 12 > role="alertdialog" 16](#_Toc1203329167)

[1 - Course intro and presentation invitations mention that knowledge of HTML is required 17](#_Toc875838947)

[2 - Using redundant ARIA attributes to restore semantics removed by CSS 17](#_Toc732499931)

[3 - Update WCAG View with links to term definition from the English success criteria 17](#_Toc1106445584)

[3 - Add <mark> elements to our codeblocks to highlight the syntax of interest, like Deque 17](#_Toc1929653227)

[3 - The course could benefit from a lot more internal linking of concepts 18](#_Toc1655984789)

[NO: Update mod 5 definition of "pure decoration" 18](#_Toc2051186286)

[Done 19](#_Toc1174488886)

[Done: Add Chrome emulation to example in mod9 animation-motion 19](#_Toc1596021755)

[Done: Report the site wrapper’s mobile menu markup issues 19](#_Toc823463924)

[Revisions grouped by effort 19](#_Toc1312538591)

## An effort flag precedes the task title (1 least, 3 most)

1 = minimim (up to an hour)

2 = moderate (up to a day)

3 = considerable (at least a day)

If you make changes, update the list of revisions grouped by effort at the end of the document.

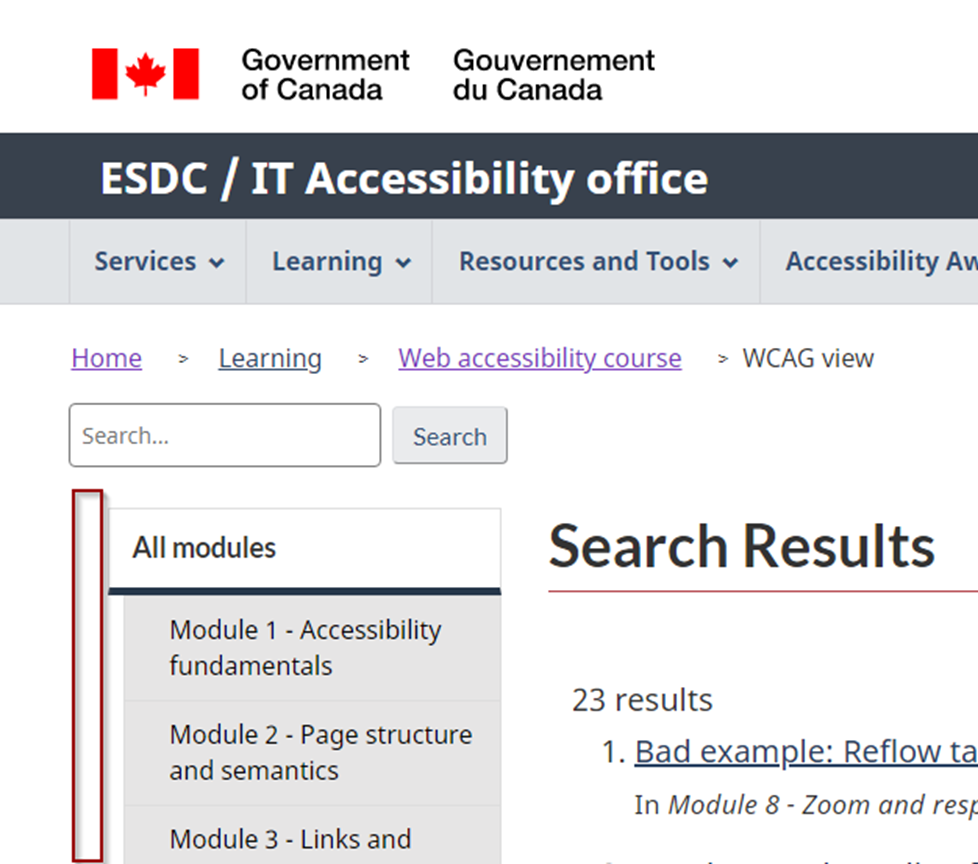
## 1 - Add a link to post a Github issue

Right after the link to the feedback form. Separate the two links with a vertical border using CSS.

## 1 - Remove excess left margin from left shoulder menu

Two elements need their left margin or padding reduced, but it's complicated by class names shared by other component on the page:

* The <nav> element shares a padding-left rule with other elements. Set padding-left to zero on the <nav> **only**.
* The child <section> element shares a margin-left rule with other elements. Set margin-left to zero on the child <section> **only**.



## 2 - Review the JS in the course to see where we can replace it with :has()

The CSS :has() selector has implications we gotta consider, especially wrt states and hiding content.

The amazing CSS :has() selector allows you to style an element based on its descendant elements or their properties.

All the major browser but Firefox support it, but Firefox seems close. The ticket for the issue is 15 years old but they’ve been busy recently and have an experimental implementation.

<https://bugzilla.mozilla.org/show_bug.cgi?id=418039>

You enable it in Firefox by typing about:config in the address bar and searching for "has".

Somebody posted a link in Nov 2022 in that Firefox github thread “there's a lot of buzz around this selector now, especially after this article: <https://webkit.org/blog/13096/css-has-pseudo-class/> “

Check it out. Setting the whole page to high contrast mode with a single checkbox anywhere on the page blew my mind.

That article ends calling :has() a revolution and linking to a tweet asking designers how they might use :has(). Mostly it’s a lot of cool styling use cases, but I found these a11y-relevant examples …

**Twitter:**

‘li:has(ul)’ to append a chevron to items that have a submenu in the classical CSS dropdown menu.

:has(:focus) {visibility: visible;} to create non-terrible pure CSS menus

JavaScript-less hamburger menus

label:has( input:invalid ) { color: red } instead of various div wrapping class adding or non semantic structure methods used commonly

div:has(input:invalid) to expose the input’s error description. Also triggers a live region if used. The HTML5 input types do their own validation, so there’s no JS needed for their error messages.

body:has(.settings-view-minimal:checked) { } to adjust the page to minimal view settings. Like high contrast, could be a checkbox anywhere on the page.

hide the table heading rows if there are no body rows. Useful when rows are added and removed dynamically.

**MDN:**

The new :modal pseudo-class is triggered when a dialog is in the open state. With :has(:modal) you can style anything in the DOM based on whether the dialog is open or closed.

**Adrian Roselli (guru):**

This is for a disclosure widget, where the toggled panel follows the heading/button. From a useful blog post on CSS and a11y that mostly covers strategies for other selectors: <https://adrianroselli.com/2021/06/using-css-to-enforce-accessibility.html>

h2:has(> button[aria-expanded="false"]) + div {

display: none;

}

The only thing JavaScript has to change is the aria-expanded attribute value! No class names needed, either. That’s definitely changing the Disclosure widget in the mod 12 example.

## 1 or 3 - Add "separator" role to hidden begins/ends

The appropriate semantics for the hidden course text marking boundaries is the role="separator" attribute.

We don't need to name it since it's static, non-focusable separator. (A separator can also be a widget users drag to widen an area, in which case it needs to be focusable, have a name, aria-valuemin, etc).

I'm pretty sure I made all the boundary declarations consistent. They should just have one of these values:

<p class="wb-inv">Example begins</p>  
<p class="wb-inv">L'exemple commence</p>

<p class="wb-inv">Example ends</p>  
<p class="wb-inv">L'exemple finit</p>

<p class="wb-inv">Code begins</p>  
<p class="wb-inv">Début du code</p>

<p class="wb-inv">Code ends</p>  
<p class="wb-inv">Fin du code</p>

All we need to do is add role="separator" to each.

## 1 - Move Expand All/Collapse to below the TOC on All Best Practices

## 1 - Update Mod 2 > Landmarks with links to mod 12 "application" and "document" roles

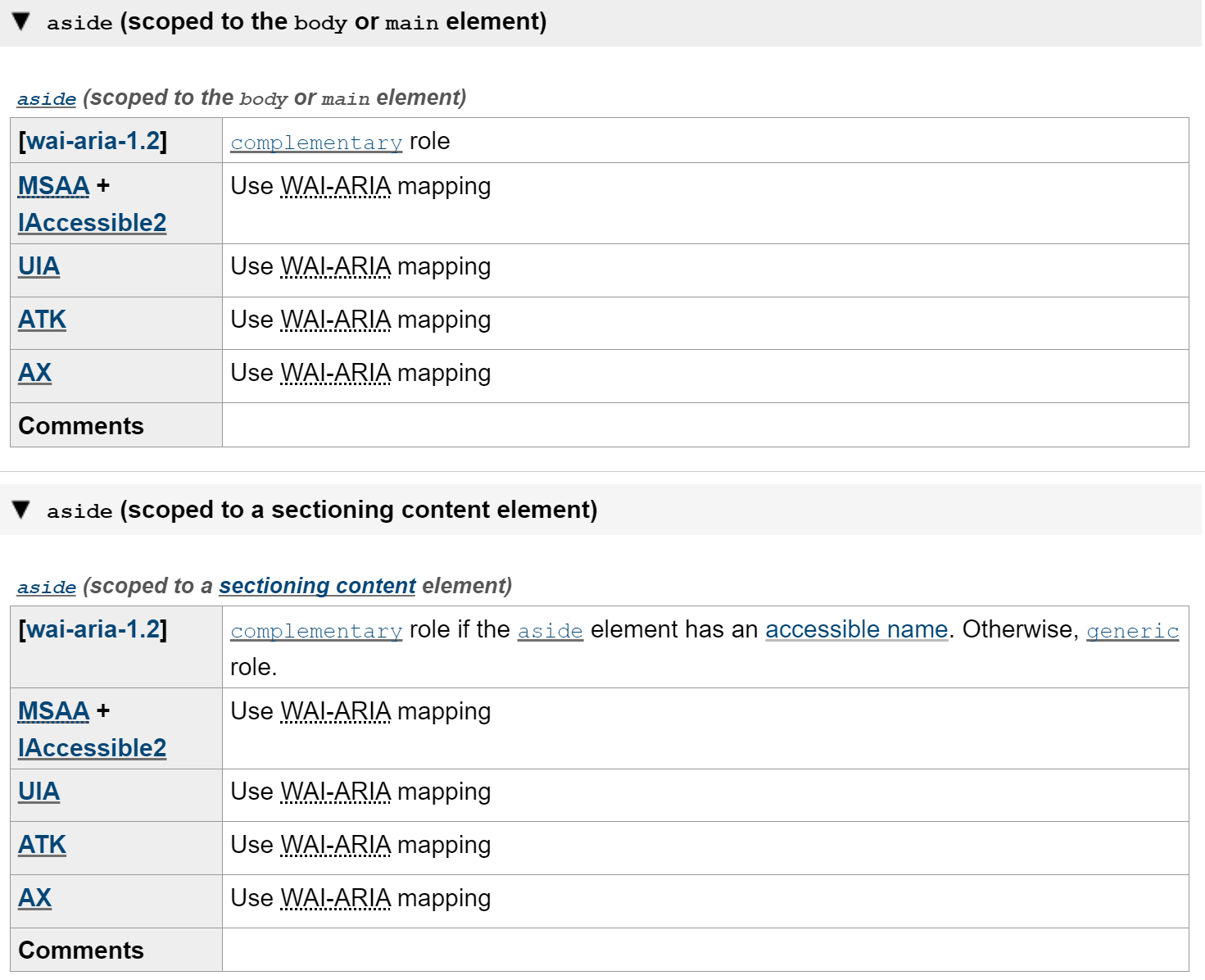
Even though "document" is easily explained in mod 2 since it's the default container, its principal use case is as a child of "application", holding screen reader navigable instructions. And "application" only comes into play with ARIA widgets.

Implement after mode 12 is written.

## 1 – Update Mod 2 > Landmarks > aside

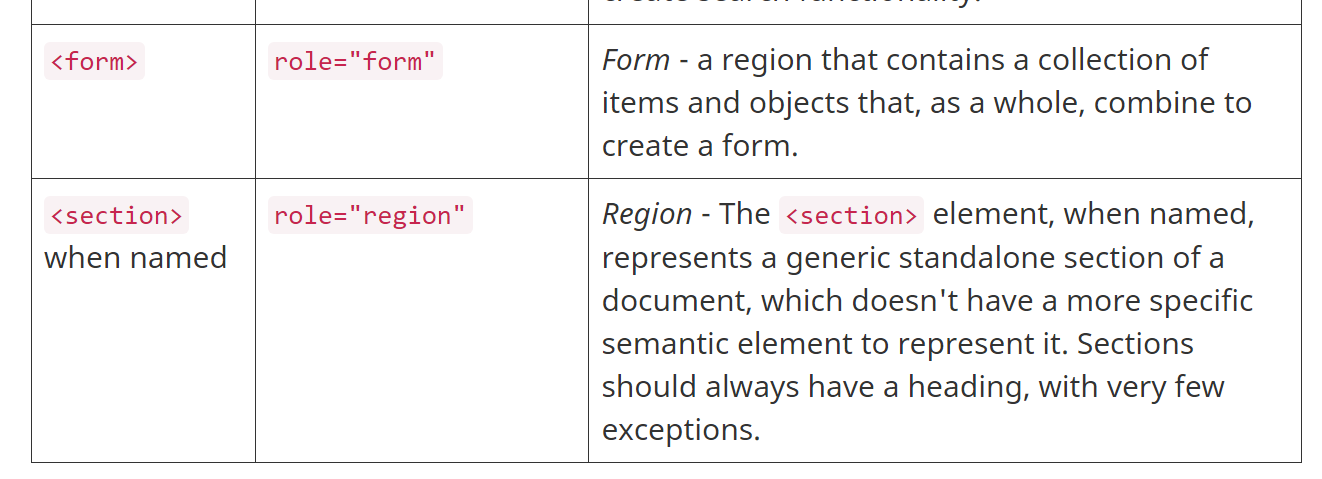
If scoped to a sectioning element, the aside must be named to be considered a complementary role.

<https://w3c.github.io/html-aam/#html-element-role-mappings>



## 1 – Update Mod 2 > Landmarks > form

<form> is a landmark only when named. We mention that in the text but not the table



## 2 – Update Mod 2 > Headings with <hgroup> element

The spec entry: <https://html.spec.whatwg.org/multipage/sections.html#the-hgroup-element>

[Content model](https://html.spec.whatwg.org/multipage/dom.html#concept-element-content-model):

Zero or more [p](https://html.spec.whatwg.org/multipage/grouping-content.html#the-p-element) elements, followed by one [h1](https://html.spec.whatwg.org/multipage/sections.html#the-h1,-h2,-h3,-h4,-h5,-and-h6-elements), [h2](https://html.spec.whatwg.org/multipage/sections.html#the-h1,-h2,-h3,-h4,-h5,-and-h6-elements), [h3](https://html.spec.whatwg.org/multipage/sections.html#the-h1,-h2,-h3,-h4,-h5,-and-h6-elements), [h4](https://html.spec.whatwg.org/multipage/sections.html#the-h1,-h2,-h3,-h4,-h5,-and-h6-elements), [h5](https://html.spec.whatwg.org/multipage/sections.html#the-h1,-h2,-h3,-h4,-h5,-and-h6-elements), or [h6](https://html.spec.whatwg.org/multipage/sections.html#the-h1,-h2,-h3,-h4,-h5,-and-h6-elements) element, followed by zero or more [p](https://html.spec.whatwg.org/multipage/grouping-content.html#the-p-element) elements, optionally intermixed with [script-supporting elements](https://html.spec.whatwg.org/multipage/dom.html#script-supporting-elements-2)

Incidentally, script-supporting elements are <script> and <template>.

The content model is different from the original <hgroup>. The original let you nest two headings. Now it's just paragraph(s) and heading.

*The* [*hgroup*](https://html.spec.whatwg.org/multipage/sections.html#the-hgroup-element) *element* [*represents*](https://html.spec.whatwg.org/multipage/dom.html#represents) *a heading and related content. The element may be used to group an* [*h1*](https://html.spec.whatwg.org/multipage/sections.html#the-h1,-h2,-h3,-h4,-h5,-and-h6-elements)*–*[*h6*](https://html.spec.whatwg.org/multipage/sections.html#the-h1,-h2,-h3,-h4,-h5,-and-h6-elements) *element with one or more* [*p*](https://html.spec.whatwg.org/multipage/grouping-content.html#the-p-element) *elements containing content representing a* ***subheading, alternative title, or tagline****.*

Steve Faulkner's tweet structure is a good idea:

<hgroup>

<h1>heading</h1>

<p>**subtitle/subheading/tagline**</p>

</hgroup>

Examples from the spec:

<hgroup>  
 <h1>Dr. Strangelove</h1>  
 <p>Or: How I Learned to Stop Worrying and Love the Bomb</p>  
</hgroup>

<hgroup>  
<h1>The reality dysfunction</h1>  
<p>Space is not the only void</p>  
</hgroup>

MDN's example is a little confusing since the <p> content isn't a "subheading, alternative title, or tagline".

<hgroup id="document-title">

<h1>HTML: Living Standard</h1>

<p>Last Updated 12 July 2022</p>

</hgroup>

Github issue "Add examples of hgroup":

<https://github.com/whatwg/html/issues/8615>

<hgroup><p>Tom Clancy's</p><h1>Rainbow Six</h1></hgroup>

<hgroup>

<p>Based on the Marvel comic book: </p>

<h1>Ant-Man</h1>

</hgroup>

<hgroup>

<p>Wes Craven presents:</p>

<h1>They</h1>

</hgroup>

In the Github issue, somebody asks if this structure is compliant. It's not:

<hgroup>

<p>Category: <a href="/diary">diary</a> / <a href="/diary/2022">2022</a> / <a href="/diary/2022/08">08</a></p>

<h1>I went to Niagara Falls.</h1>

<p>Created at: <time datetime="2022-08-12">2022-08-12</time></p>

</hgroup>

The reply: "Those seem like information related to the article, but not a 'subheading, alternative title, or tagline'."

## 3 - Update Mod 2 > Content structure with details/summary, barriers and button solution

Best location for a description of details/summary seems to be Mod 2. In ARIA it’s classified as a widget role (Disclosure)

Barriers:

* Content in closed details elements is hidden from browser find functionality.
* Some users with cognitive disabilities using immersive readers have their concentration broken by the distracting clicking.

To fix, either:

* Set the details to be open by default or
* Provide "open all"/"collapse all" buttons

This came to our attention as a reader-posted course github issue #1097 (closed). We may want to get more info, or the appropriate phrasing, from the poster, who's an advocate.

We can use our own buttons solution as an example.

## 2 - Expand mod 2 > Content structure > Article

With a few sentences of content paraphrased from <https://www.w3.org/TR/wai-aria-1.2/#article>

An article is not a navigational [landmark](https://www.w3.org/TR/wai-aria-1.2/#dfn-landmark), but may be nested to form a discussion where assistive technologies could pay attention to article nesting to assist the user in following the discussion. An article could be a forum post, a magazine or newspaper article, a web log entry, a user-submitted comment, or any other independent item of content. It is *independent* in that its contents could stand alone, for example in syndication. However, the [element](https://www.w3.org/TR/wai-aria-1.2/#dfn-element) is still associated with its ancestors; for instance, contact information that applies to a parent body element still covers the article as well. When nesting articles, the child articles represent content that is related to the content of the parent article. For instance, a web log entry on a site that accepts user-submitted comments could represent the comments as articles nested within the article for the web log entry. Author, heading, date, or other information associated with an article does not apply to nested articles.

In this case, we need to mention screen reader support in the top brands, with > 90% of users having a keyboard shortcut and a role declaration.

* JAWS: 53.7% of users
* NVDA: 30.7%
* VoiceOver: 6.5%

This is a July 2020 snapshot and it's out of date.

* NVDA implemented it recently, but it's off by default. Go to Preferences > Settings > Document formatting > Elements, and select "Articles".
* Searching the latests Narrator keyboard shortcuts, there's no mention of "article". But Narrator is only used by 0.5%.

[HTML5 sectioning elements and screen readers - AccessibilityOz](https://www.accessibilityoz.com/2020/02/html5-sectioning-elements-and-screen-readers/)

Screen readers’ support for ARTICLE element

Narrator~~, NVDA~~ and TalkBack on Firefox do not interpret the Article tag in any noticeable manner. JAWS interprets the tag. The Article tag is supported and included within navigation by landmarks by TalkBack on Chrome and VoiceOver on Safari.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Element** | **JAWS** | | | **NVDA on Chrome** | | |
| **Navigable by a hotkey** | | **Phrase** | **Navigable by a hotkey** | | **Phrase** |
| **Chrome** | **Firefox** | **Chrome** | **Firefox** |
| **Article** | yes \* | yes \* | “article” | ~~No~~ Yes | ~~No~~ Yes | TO TEST |

\* a different hotkey than for landmarks. it is not included within navigation by landmarks. If the JAWS user wants to get to the article by using a hotkey, instead of pressing “R” (next landmark), they press “O” (next article). This separation of article and other landmarks might be done for user’s convenience.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Element** | **Narrator** | | **TalkBack** | | | **VoiceOver for iOS** | |
| **Navigable** | **Phrase** | **Navigable by a gesture** | | **Phrase (Chrome only)** | **Navigable by gesture** | **Phrase** |
| **Edge** | **Chrome** | **Firefox** | **Safari** |
| **Article** | No | N/A | Yes | No | “article” | Yes | “article, landmark” |

## 2 - Expand mod 3 > Navigation > Character key shortcuts

The APG covers when to move focus and trigger a function, and what shortcuts to avoid.

<https://www.w3.org/WAI/ARIA/apg/practices/keyboard-interface/#x6-9-keyboard-shortcuts>

## 1 – Update Mod 3 > Navigation > Focus order

You add role="lnk" to disable a link. If you remove the href attribute from a link, it no longer receives focus, but I assumed it still declared as a link. I guess it doesn't. Test!

Current text:

To receive focus, a link must have a populated href attribute.

To prevent a link from receiving focus, remove its href attribute.

From ARIA in HTML:

In situations **where ARIA and HTML have feature parity, but diverge in allowances,** it can create for a misalignment in support, if not also user experiences. In situations where ARIA allows a feature not supported by HTML, it will often be in the author's and ultimately the user's best interest to instead **implement as a fully custom ARIA widget**.

In the following example, a hyperlink needs to be communicated as being in the disabled state. HTML does not allow for the use of the disabled attribute on a hyperlink, and using aria-disabled=true would communicate the hyperlink as being disabled to assistive technologies, but would not actually disable the element. The most effective way to both communicate and actually disable a hyperlink would be to remove the href from the [a](https://html.spec.whatwg.org/multipage/text-level-semantics.html#the-a-element) element, creating a placeholder. Then, ARIA can be applied to this placeholder link to communicate the element's intended role and state.

[EXAMPLE 2](https://www.w3.org/TR/html-aria/#example-communicate-a-disabled-link-with-aria): Communicate a disabled link with ARIA

<a role=link aria-disabled=true>...</a>

## 2 - Update Mod 3 > Changing context > On input with examples

[3.2.2: On Input:](https://www.w3.org/WAI/WCAG21/Understanding/on-input) Changing the setting of any user interface component does not automatically cause a change of context unless the user has been advised of the behavior before using the component.

<https://www.w3.org/WAI/WCAG21/Techniques/general/G13:> G13: Describing what will happen before a change to a form control that causes a change of context to occur is made

From Mod 3:

On input

Changing the setting of a user interface component must not cause a change of context. Examples of changing a setting include checking a checkbox, entering text into a text field, or changing the selected option in a <select> element. Clicking on links or tabs in a tab control is activating the control, not changing the setting of that control.

Again, the design goal is predictability. There are two options:

* Give the user control over changes of context with an explicit user action that's generally understood to cause a change of context; e.g, clicking a link or pressing a submit button.
* Describe to the user how interacting with the control changes the context, before the control is encountered.

Using aria-describedby on the problem element isn't listed as a sufficient or advisory technique for [3.2.2: On Input](https://www.w3.org/WAI/WCAG21/Understanding/on-input) but technique G13 refers to it as a good idea:

The objective of this technique is to provide information to users about what will happen when a change to a form control results in a change of context. Because changing the value of a form control does not typically result in a change of context, it is important that authors provide instructions that make the user aware of the behavior in advance. Where possible, it is a good idea to programmatically associate the instructions describing the change with the form control itself.

[12:58 PM] Chan, Michael [NC] Technique G13 has some examples of how to provide instruction in different situations. We should include that in addition to your aria-describedby technique.

## 1 - Fix phrasing in mod 3 > Visible focus indicator > Best practice (and All BP)

Best practice bullet is borked. See actual page.

"Avoid styles that the visibility of keyboard focus indicators"

## 1 – Add CSS style rules to mod 3 > Links > Good example: Warns users link opens

<https://bati-itao.github.io/learning/esdc-self-paced-web-accessibility-course/module3/links.html#link-window-visually-hidden>

you style the tooltip to look like a title attribute's tooltip, with yellow background with darker yellow borders, just offset from the triggering text.

a:hover span.tooltip, a:focus span.tooltip {

display: block;

left: inherit;

z-index: 99;

color: #626562;

background-color: #ffffe7;

border: 1px solid #626562;

position: absolute;

margin-left: 20px;

text-transform: initial;

padding-left: 5px;

overflow: hidden;

}

You add a language to the class names, since the width is set in the CSS and French is longer. .

new-tab.en { width: 6.2em }

.new-tab.fr { width: 10.1rem }

You can see it in action in the very top links on bell.ca: <https://www.bell.ca/Bell_Internet>

## 1 – Mod 4: Fix id/headers example and add method

Our example has incorrect headers values. We worked on this material for the Dec 2022 tutorial.

to correctly populate the headers attributes, you first reference the adjacent solo headers, and the solo headers reference the group headers. Screen readers chain the associations together.

I updated the a course example and it worked much better in Chrome + JAWS.

<file:///F:/Tests/tables/id-headers-three-deep.html>

## 1 – Mod 4: fix placement of <colgroup>

Course says it falls before <caption>. It falls after.

## 3 - Update mod 5 or 7 to address SVG icons

Two options:

1. We already address icon fonts in Mod 7 > CSS-generated content. We could address SVG icons in Mod 5 > SVG.
2. Or, we could create a new Mod 7 section called "Icons" and move the icon fonts material to it and describe the pros and cons of font icons vs SVG icons vs sprites vs simple img elements. We'd link to the section from the CSS-generated content and SVG pages.

I think I prefer option 2.

[CSS-generated content – Module 7 - Visual design and colours (bati-itao.github.io)](https://bati-itao.github.io/learning/esdc-self-paced-web-accessibility-course/module7/css-generated.html)

"In recent times there has been a general consensus and momentum towards SVG icons format. SVG icons system ensures better performance, higher accessibility standards, high rendering quality, unmatched flexibility, and extensive customization."

<https://www.lambdatest.com/blog/its-2019-lets-end-the-debate-on-icon-fonts-vs-svg-icons/>

Browser support has improved since that article was published

"Step 3 - Using your SVG icon system" uses an interesting technique where the icons are defined in a hidden block of SVG holding all the shape details, and then deployed with some succinct and unusual HTML. Not sure of the point of this span though:

<svg class="icon-twitter">

<use xlink:href="#icon-twitter"></use>

</svg><span class="name"> icon-twitter</span>

2012 TPGi post on accessible CSS image sprites

<https://www.tpgi.com/notes-on-accessible-css-image-sprites/>

## 2 - Expand mod 6 > labelling controls to include nesting a control in a label

## 2 - Expand mod 6 > User notifications to include aria-live

The section lists four methods but doesn’t cover In-line feedback and using aria-live. See W3C-WAI Forms Concepts tutorial.

[Approach 2: During typing](https://www.w3.org/WAI/tutorials/forms/notifications/#during-typing) uses aria-live=polite

[Approach 3: On focus change](https://www.w3.org/WAI/tutorials/forms/notifications/#on-focus-change) uses aria-live=assertive

## 3 - Update mod 6 > Identify input purpose with autocomplete=”on|off” method

See the HTML 5 spec: <https://html.spec.whatwg.org/multipage/form-control-infrastructure.html#autofill>

It frames the "on"/"off" attribute values and the token values as different "mantles", or ways the attribute is used.

According to this [blog post on autocomplete](https://cloudfour.com/thinks/autofill-what-web-devs-should-know-but-dont/), "[Y]ou’re either using on and off values OR you’re using autofill field names. You can’t use them at the same time."

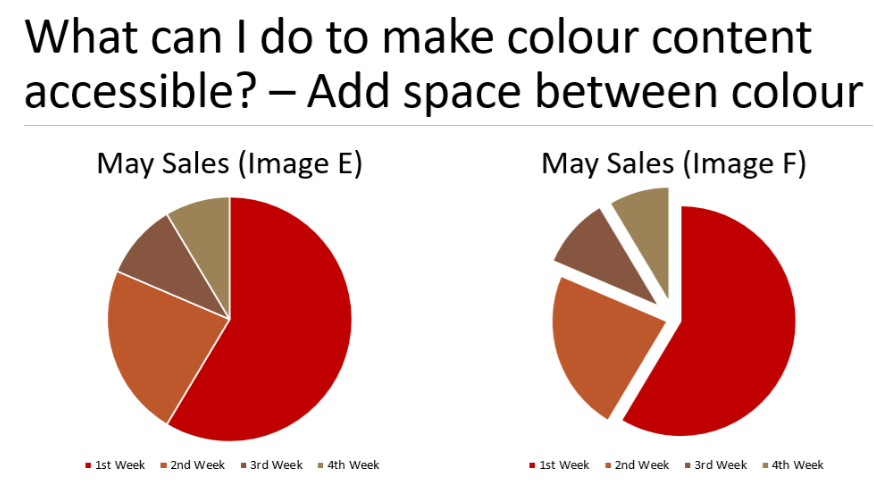
I removed the page’s mention of autocomplete being usable on the <form> element. It made no sense without the accompanying discussion of “on” and ”off” values.

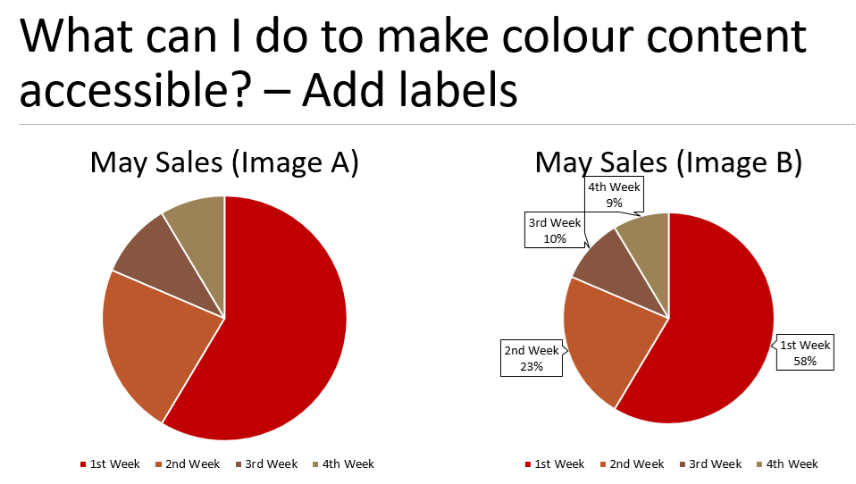
## 2 - Expand mod 6 > Multi-step forms with example of setting focus

In mod 6 > multi-step forms we list the best practices then break out the last bullet in a description of the different approaches to indicating progress. But we don't discuss how to implement this focus best practice. We could explain it here or we could link the phrase "set the focus" to the mod 12 section, when that bit is written

Ensure keyboard focus moves smoothly from step to step, backwards and forwards. At a new step, set the focus preferably on the next relevant form element, or relevant heading or container (a heading or container would need the tabindex="-1" attribute to programmatically receive focus).

## 3 - Update Mod 7 > colour with three examples





## 3 - Update mod 7 to address "High contrast mode" (or throughout course)

If we distribute high-contrast mode advice instead of creating a dedicated section, we'll need to address it in Mods 5 and 7, maybe others.

**This page lists which widgets have High Contrast guidance:**

<https://raw.githack.com/w3c/aria-practices/master/coverage/index.html>

**This is the github issue:**

<https://github.com/w3c/aria-practices/issues/1460>

SVG icon solution

Example from APG checkbox-mixed

* To ensure the borders of the inline SVG checkbox graphics in the CSS have sufficient contrast with the background when high contrast settings invert colors, the color of the borders are synchronized with the color of the text content. For example, the color of the checkbox borders is set to match the foreground color of high contrast mode text by specifying the CSS currentcolor value for the stroke property of the rect and polyline elements used to draw the checkbox. To make the background of the checkbox graphics match the high contrast background color, the fill-opacity attribute of the rect element is set to zero. If specific colors were instead used to specify the stroke and fill properties, those colors would remain the same in high contrast mode, which could lead to insufficient contrast between the checkbox and the background or even make the checkbox invisible if the color matched the high contrast mode background.  
   Note: The SVG element needs to have the CSS forced-color-adjust property set to auto for the currentcolor value to be updated in high contrast mode. Some browsers do not use auto for the default value.

From <<https://www.w3.org/WAI/ARIA/apg/example-index/checkbox/checkbox-mixed.html>>

SVG icon solution is also explained in the AGP disclosure widget example:

<https://www.w3.org/WAI/ARIA/apg/example-index/disclosure/disclosure-image-description.html>

Also

<https://www.w3.org/WAI/ARIA/apg/example-index/menu-button/menu-button-actions-active-descendant>

<https://www.w3.org/WAI/ARIA/apg/example-index/radio/radio-rating>

<https://www.w3.org/WAI/ARIA/apg/example-index/menu-button/menu-button-actions>

<https://www.w3.org/WAI/ARIA/apg/example-index/slider/slider-temperature>

<https://www.w3.org/WAI/ARIA/apg/example-index/checkbox/checkbox>

<https://www.w3.org/WAI/ARIA/apg/example-index/treeview/treeview-navigation>

## 3 - Update mod 9 > Multi-step form > Approach 3 with WET progress meter

[Progress bar and meter - Design Guide - Style Guide - Web Experience Toolkit (WET) (wet-boew.github.io)](https://wet-boew.github.io/wet-boew-styleguide/design/progressmeter-en.html)

[Multi-step forms - Module 6 - Forms (bati-itao.github.io)](https://bati-itao.github.io/learning/esdc-self-paced-web-accessibility-course/module6/multi-step-forms.html#steps-form)

## 1 - Rephrase mod 11 > ARIA live explanation

The aria-live page says the "accessibility API" monitors the container and identifies injected text. That's not how the API works. Rephrase.

## 1 - Test mod 11 > ARIA live example

Test the role="alert" example. Does it work if the role element itself is hidden, or a child element?

## 1 - Update mod 11 > ARIA live links from ARIA 1.1 to ARIA 1.3

## 2 - Update mod 11 > ARIA live to use WET "Alert" pattern

[Alerts - Design Guide - Style Guide - Web Experience Toolkit (WET) (wet-boew.github.io)](https://wet-boew.github.io/wet-boew-styleguide/design/alerts-en.html)

When a live region is visible and qualifies as a WET Alert, use the WET markup.

WET Alert purpose: "Use to provide contextual feedback messages for typical user actions." Success, info, warning, danger.

## 2 - Update mod 11 > ARIA live to mention and link to mod 12 > role="alertdialog"

Its related concepts are alert and dialog. It's got a foot in each camp. Describe it briefly in mod 11 and link to mod 12. Then discuss it there along with dialog.

"A type of dialog that contains an alert message, where initial focus goes to an [element](https://dom.spec.whatwg.org/#concept-element) within the dialog.... If an author desires focus to move to a message when it is conveyed, the author *SHOULD* use [alertdialog](https://w3c.github.io/aria/#alertdialog) instead of alert."

## 1 - Course intro and presentation invitations mention that knowledge of HTML is required

For modules 2-12.

## 2 - Using redundant ARIA attributes to restore semantics removed by CSS

With lists, using CSS list-style: none removes the screen reader declaration "bullet" in some user agents. VoiceOver + Safari completely remove the semantics. Adding redundant ARIA labels restores it.

However, as we saw earlier, applying list-style: none; to the list changed the user's experience entirely. VoiceOver no longer said "list, 2 items," nor did it tell the user how far into the list they were. Instead, it just treated every item as a plain text node. It seems as though Safari's engineers decided lists without bullets or other markers aren't listy enough, and decided to instead nullify the list's semantics.

<https://benmyers.dev/blog/css-can-influence-screenreaders/>

With tables, designing for responsive by setting the CSS display: flex or grid removes the table semantics in screen readers. Adding redundant ARIA labels restore them

<https://www.tpgi.com/short-note-on-what-css-display-properties-do-to-table-semantics/>

<https://adrianroselli.com/2018/05/functions-to-add-aria-to-tables-and-lists.html>

<https://adrianroselli.com/2017/11/a-responsive-accessible-table.html>

The Bad example in mod 8 responsive tables lacks the redundant ARIA.

## 3 - Update WCAG View with links to term definition from the English success criteria

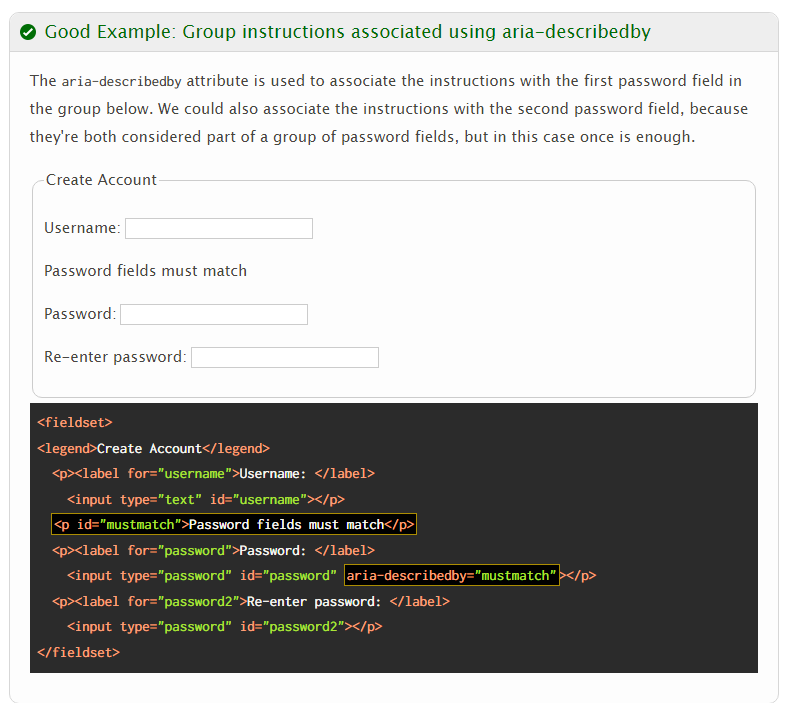
the French has them

## 3 - Add <mark> elements to our codeblocks to highlight the syntax of interest, like Deque

Unfortunately, I can't figure out how they implemented it. When I add a naked <mark> or deque's <mark class="highlight">, they get stripped out. Deque has modified the JavaScript, or they add the <mark> after the block has been rendered by highlight.js.

There’s a module that adds background yellow highlighting: <https://github.com/TRSasasusu/highlightjs-highlight-lines.js>

Example:



## 3 - The course could benefit from a lot more internal linking of concepts

For instance, this best practice bullet from mod 6 > multi-step forms could use a link to mod 11 > Time limits

Avoid a time limit to fill out the form. If a limit is required, enable the user to adjust or extend it.

## NO: Update mod 5 definition of "pure decoration"

**Decided against this one.** Three concepts now define decorative images. It's a little confusing since adjacency is a type of redundancy.

The WCAG Working Group, has allowed the interpretation of "pure decoration" to evolve to include ***redundancy***, ***adjacency*** and ***ambience*** in the new EO WCAG tutorial, and HTML5 specification without comment or objection, [and it is doubtful the WCAG will further define "pure decoration"](http://www.w3.org/2015/01/27-wai-wcag-minutes.html#item03). They probably would if the disability community raised serious concerns but that hasn't happened. Most screen reader users feel that alt on ambient images is nice to have, but not worth fighting for.

Most experts in the field identify a grey zone in the interpretation of pure decoration. Think of it as a continuum:

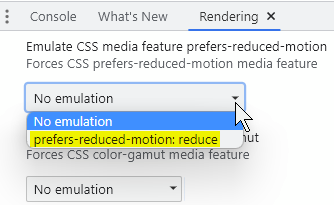
* 1. If it's a gradient image or spacer image etc., all agree it needs alt="".
  2. If it's a picture that has a purpose, information or function then it needs descriptive alt text.
  3. If it is a photo of some **thing** for ambience or eye candy, then it is in the grey zone and can have alt="". However, a short alt description is preferable to most screen reader users.

[When can a web author use null alt text alt="" to make an image conform to WCAG? - David MacDonald Web Accessibility Blog, testing, teaching, and discussion](https://www.davidmacd.com/blog/what-is-pure-decoration-alt-text-in-wcag.html)

# Done

## Done: Add Chrome emulation to example in mod9 animation-motion

Chrome > More tools > Rendering > Emulate CSS media feature prefers-reduced-motion



You can instead emulate the operating system’s reduced motion feature in a browser that offers the option. For instance, in Chrome’s Tools menu, the Rendering submenu offers “Emulate CSS media feature prefers-reduced-motion”. Activating that option is equivalent to changing the operating system’s setting.

Vous pouvez plutôt imiter la fonction de mouvement réduit du système d’exploitation dans un navigateur qui propose l’option. Par exemple, dans le menu Outils de Chrome, le sous-menu Rendu propose « Emulate CSS media feature prefers-reduced-motion » (Imiter la fonction CSS « Préférer les médias à mouvement réduit »). L’activation de cette option équivaut à modifier les paramètres du système d’exploitation.

## Done: Report the site wrapper’s mobile menu markup issues

Its nav elements are unnamed and one nav nests three child navs

Issues are posted to Github.

# Revisions grouped by effort

Keep this list up to date!

1 - Add a link to post a Github issue

1 - Remove excess left margin from left shoulder menu

1 or 3 - Add "separator" role to hidden begins/ends

1 - Move Expand All/Collapse to below the TOC on All Best Practices

1 - Update Mod 2 > Landmarks with links to mod 12 "application" and "document" roles

1 - Fix phrasing in mod 3 > Visible focus indicator > Best practice (and All BP)

1 - Course intro and presentation invitations mention knowledge of HTML is required

1 – Mod 4: Fix id/headers example and add method

1 – Mod 4: fix placement of <colgroup>

1 - Rephrase mod 11 > ARIA live explanation

1 - Test mod 11 > ARIA live example

1 - Update mod 11 > ARIA live links from ARIA 1.1 to ARIA 1.3

2 - Expand mod 2 > Content structure > Article

2 - Expand mod 3 > Navigation > Character key shortcuts

2 - Update Mod 3 > Changing context > On input with examples

2 - Expand mod 6 > labelling controls to include nesting a control in a label

2 - Expand mod 6 > User notifications to include aria-live

2 - Expand mod 6 > Multi-step forms with example of setting focus

2 - Update mod 11 > ARIA live to use WET "Alert" pattern

2 - Update mod 11 > ARIA live to mention and link to mod 12 > role="alertdialog"

2 - Using redundant ARIA attributes to restore semantics removed by CSS

3 - Update WCAG View with links to term definition from the English success criteria

3 - Add <mark> elements to our codeblocks to highlight the syntax of interest, like Deque

3 - Update Mod 2 > Content structure with details/summary, barriers and button solution

3 - Update mod 5 or 7 to address SVG icons

3 - Update mod 6 > Identify input purpose with autocomplete=”on|off” method

3 - Update Mod 7 > colour with three examples

3 - Update mod 7 to address "High contrast mode" (or throughout course)

3 - Update mod 9 > Multi-step form > Approach 3 with WET progress meter

3 - The course could benefit from a lot more internal linking of concepts