Accessibility in Adapt

A technical perspective

This document is an analysis of the approach taken to implement accessibility in Adapt. It has been written as a technical breakdown of the code, but there is also an emphasis on describing the qualitative aspects of the approach.

The material here can be used to compliment the official documentation here.

Two files handle the majority of the responsibilities:

- `accessibility.js` handles initialisation and configuration of accessibility within the course and marshalls the setup of accessibility in menus and pages as they are created. This singleton uses `_isEnabled` to indicate if accessibility *can* be used (i.e. the course is accessible) and `_isActive` to indicate whether accessibility *is* being used (i.e. the user has requested it).
- `jquery.a11y.js` is the library responsible for making content accessible and acts as a programming API to enable plugins to maintain accessibility.

The following material covers these two files and their salient features. Function listings are given with explanations of behaviour and any important features. Listings for trivial or self-explanatory functions are omitted.

`accessibility.js`

`initialize`

Registers various listeners and Handlebars helpers and restores `_isActive` state from tracking data if present. If `_isEnabled` is false the accessibility configuration is setup so that most calls from plugins or the Adapt core result in no-ops. However, some functions such as disabling scrolling when popups are shown are available.

`checkTabCapture`

Developer note: has no apparent function other than debugging.

`setupLegacy`

Legacy support pertains only to toggling a `focused` class on focusable elements in IE8.

`focusInitial`

Called each time a page or menu has rendered. If accessibility `_isActive` then one of two things will happen: if the usage instruction has not been read then it is given focus. Otherwise, focus is given to the first navigable element in the document (unless the user has begun to scroll or tab through the content).

`onKeyUp`

If the tab key is pressed, focus is given to the accessibility toggle button. Subsequent tabbing continues to focus this button until either accessibility is activated or a native tab element is focussed[3].

`jquery.a11y.js`

This library makes elements accessible or inaccessible by manipulating a small number of their properties. Generally, it is just the `aria-hidden`, `role` and `tabindex` attributes that are changed. The `disabled` attribute is also commonly toggled and an `aria-hidden` CSS class is added or removed accordingly. The `aria-label` attribute is also manipulated, but typically only during initialisation routines.

`preventScroll(event)`

Given an `event`, determine if the `event` took place in an area where scrolling should be disabled and if so prevent that area from scrolling.

`preventScrollKeys(event)`

Given a key `event`, determine if the `event` took place in an area where scrolling should be disabled. If so, and the `event` originated from a cursor key, prevent that area from scrolling.

`getScrollingParent`

When a vertical scroll event occurs on an element, this function can be used to determine the actual element doing the scrolling – either the document `body` or an element with `overflow-y` set to `scroll/auto`.

`makeHTMLOrTextAccessible(text)`

Converts a given a piece of `text` (plain or HTML) to accessible markup. Operates recursively on HTML, breaking it into individually navigable parts.

`onFocusCapture`

Determines `$activeElement` – the element which has focus. Begins with event target and looks for nearest valid element, potentially traversing entire DOM. N.B. does not perform focus or scroll operations.

`onFocus`

The handler invoked when an element receives focus. The `$activeElement` is set and the element is brought into view if necessary. N.B. a screen reader will read applicable content from this element.

`a11y_triggerReadEvent`

It is assumed that this is a debugging function used to check screen readers are speaking the correct lines.

`$.fn.limitedScrollTo`

Scroll matched element into (vertically central) view if currently off screen. Does not operate on fixed-position elements.

`$.fn.focusOrNext`

Gives focus to matched element if possible. If matched element cannot be given focus the function looks for the nearest valid element, potentially traversing the entire DOM. Does not perform scroll operation.

`$.fn.a11y_update`

Called on significant DOM changes, such as when a page has rendered or when a popup has been opened/closed. The most significant operations that may be performed by this function are to parse the DOM for any `aria-label` attributes that need to be converted to tabbable text and to ensure the focusguard is correctly placed.

`$.a11y_on`

Matches elements against a selector and determines if they are accessible or not. N.B. this is a special function used internally during initialisation of pages/menus.

`$.fn.a11y_on`

Determines whether all descendents of the matched elements are accessible or not.

`$.fn.a11y_cntrl`

For each matched element, determines whether the element is accessible or not. Optionally toggles the `disabled` property of DOM elements and a `disabled` CSS class.

N.B. normally this function operates on all elements. The exception is when tabbable text is disabled – in this case it will operate on every element except those that *have a tabindex defined yet are not part of the tab order by default*[2].

Developer note: parentsFilter is not defined anywhere, resulting in some no-op calls.

`$.fn.a11y_cntrl_enabled`

Developer note: used by various plugins that need to manipulate the user interface on user action.

`$.fn.a11y_text`

Converts matched elements to accessible elements. Available via a Handlebars helper and used to set up tabbable text when enabled.

Developer note: used by various plugins that need to manipulate the DOM on user action.

`$.fn.a11y_selected`

Developer note: used by various plugins on user action to indicate when an item has been selected.

`$.a11y_alert`

Developer note: used by various plugins to indicate important information immediately to the user. N.B. this does not trigger scrolling.

`$.a11y_focus`

Used to return focus to the first **navigable** element in the document.

`$.fn.a11y_focus`

Gives focus to the first matched element if it is **focusable**, otherwise gives focus to the nearest **navigable** element in the DOM tree, preferring descendents of the matched element first, followed by the descendents of ancestors (beginning with immediate parent).

Developer note: used by plugins to shift focus on user action.

`$.fn.a11y_aria_label(deep)`

For each matched element find those with an `aria-label` attribute (including descendents if `deep` set to `true`) and evaluate whether each element should have the attribute converted to a DOM element. If tabbable text is enabled the new element will be navigable.

Popups

The following functions are used to control accessibility when popups are displayed to the user. Note that popups are intended to behave modally from an accessible point of view: that is, although it possible to access content outside of some popups (e.g. hot graphic popups) via the mouse, this is not considered a valid use case.

`$.fn.scrollDisable/$.fn.scrollEnable`

Add/remove matched elements to/from global list of elements for which scrolling is not permitted.

Developer note: event listener management could be improved to avoid potential duplicates.

`$.fn.a11y_popup/$.fn.a11y_popdown`

Called on elements serving content to the user in popups (notify, drawer items, hot graphic items). When the popup is closed accessibility is restored and focus given to the element that had focus prior to opening the popup.

`$.fn.a11y_only(container, storeLastTabIndex)`

Makes inaccessible all navigable controls in specified `container` except controls within matched elements[1]. When used to display popups no `container` is specified and so the function uses the entire document by default. If `storeLastTabIndex` is set to `true` the function will store a snapshot of the tab indices for all elements in `container` so that they may be restored later and also records which element currently has focus.

Special classes

- `a11y_ignore` applied to an element will exclude it from manipulation by the library. If the element is **focusable** it will remain subject to library focus operations.
- `a11y-ignore-focus` is used to denote an element that is **focusable**, but that will not be given focus programmatically, except when it is the matched element in a call to `$.fn.a11y_focus` or, in the case of special elements[1], the matched element in a call to `$.fn.focusOrNext`.

- `a11y-ignore-aria` despite its generic name is only used to prevent descendent elements (specifically `div` and `span` elements) with `aria-label` attributes from having this attribute converted into a DOM element.

Injected elements

The following are placed at the end[4] of the DOM tree and serve the various purposes:

- `#a11y-focuser` is placed in the top left of the viewport. Its main purpose is to receive focus prior to setting up accessibility within a menu/page. This ensures focus is not given unexpectedly elsewhere.
- `#a11y-focusguard` is placed variably, according to environment, and is used to prevent focus transferring outside of the document on reaching its end. N.B. the user is able to transfer focus outside of the document by tabbing backwards at the top of the document.
- `#a11y-selected` is placed in the bottom left of the viewport. It's purpose is as a marker at which to locate *alerts* (e.g. when a user selects an item). On iOS it receives focus prior to announcing the selection of an element.

Stacks

Two arrays termed 'stacks' are maintained: the `floorStack` and the `focusStack`.

The `floorStack` contains the document body as its first element and any popups *currently* open thereafter, in the order they were invoked. The focusguard special element is always present in the topmost floor stack (i.e. the last element in the array).

Each time a popup is opened the element which currently has focus is pushed onto the `focusStack`. When a popup is closed the last element in the `focusStack` is popped and it is given focus.

Notes

There is a 'hideable' feature which, according to documentation, can be used on an element "*that is not normally in the tab index but should be explicitly removed from it on popups*".

[1] with the exception of special injected elements (focuser, focusguard, selected)

[2] by default only the elements `a`, `button`, `input`, `select` and `textarea` appear in the tab order

[3] `textarea`, `input` and `select` elements

[4] opening a popup will shift the focusguard to the end of the element on which the popup is invoked.

Todo

Discuss use of constructs in Handlebars templates to facilitate accessibility.