Web Accessibility

> ***“The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.”***

Tim Berners-Lee

w3.org/WAI/fundamentals/accessibility-intro/

Web a11y has a hard time in being a high priority matter.

Web a11y is **not sexy**, because it is an **old** topic which deals with **disabled** people.

Web a11y **myths** make it even worse.

Myth #1: There’s only a small audience for web a11y

Myth #2: Accessible websites are ugly and boring

Myth #3: Web a11y is the sole responsibility of developers

Myth #4: A11y is expensive

Goals of this talk:

1. Tackle the misconceptions of web a11y

2. Provide you with the essential tools to build accessible web sites

Who’s the audience?

Who’s the audience?

- **People with sensory disabilities:** Colour deficiency, blindness, deafness etc.

Who’s the audience?

- **People with physical disabilities:** Contergan, motor disabilities etc.

Who’s the audience?

- **People with learning disabilities:** Dyslexia, lassasthenia etc.

Who’s the audience?

- **People with cognitive disabilities:** Down syndrome, autism etc.

Who’s the audience?

- **People with chronic illness:** Cerebral palsy, epilepsy, chronic fatigue syndrome…

Who’s the audience?

- People with other disabilities

Who’s the audience?

- **People without disabilities**
    - Busy people or people who are in a hurry
    - People who have their hands full (think of baby, ski lift) etc.
    - People with super slow network connectivity
    - People with sun shining on their smart phones
    - People reading you site in a language with incredibly long words
    - People who read captions for language related reasons
    - People who simply use your website differently than others/you
    - Fatigued people
    - Search engines
    - etc.

Accessible websites are not necessarily ugly and boring.

Here is an example as a counter document: *mozilla.org/en-US/firefox*

A team as a whole is responsible for making a website accessible.

If the designer doesn’t take a11y into account, **shall the dev fix the design?**

What's the point, if one dev incorporates a11y into **"his" components** but the next dude uses a for a submit button?

A11y is not necessarily expensive.

If you know how a11y works and do things the right way from the beginning, the costs involved are reasonable.

How do I make my website accessible?

W3C® WCAG 2.1

- Principles based on
  - **P**erceivable (1.x.x)
  - **O**perable (2.x.x)
  - **U**nderstandable (3.x.x)
  - **R**obust (4.x.x)
- Three levels of compliance: **A, AA, AAA**
- Finalized 5. June 2018

w3.org/TR/WCAG21

w3.org/WAI/standards-guidelines/wcag/new-in-21

Perceivable

Web content is made available to the senses - sight, hearing and/or touch

Perceivable

1.1 Text Alternatives: Provide text alternatives for any non-text content (images, buttons, form inputs, embedded multimedia, frames etc.)

1.2 Time-based Media: Provide alternatives for time-based media (audio, video)

1.3 Adaptable: Create content that can be presented in different ways without losing information or structure (meaningful relationships, sequence and sensory characteristics)

1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background (color, typography, audio controls etc.)

Perceivable

1.4.3 Contrast (Minimum) (AA)

- Text and images of text have a contrast ratio of at least 4.5:1.
- Large text - at least 18 point (typically 24px) or 14 point (typically 18.66px) bold has a contrast ratio of at least 3:1.

https://webaim.org/resources/contrastchecker/

Perceivable

1.4.13 Content on Hover or Focus (AA)

Where receiving and then removing pointer hover or keyboard focus triggers additional content to become visible and then hidden, the following are true:

- **Dismissable:** A mechanism is available to dismiss the additional content without moving pointer hover or keyboard focus, unless the additional content communicates an input error or does not obscure or replace other content;
- **Hoverable:** If pointer hover can trigger the additional content, then the pointer can be moved over that content without the additional content disappearing;
- **Persistent:** The additional content remains visible until the hover or focus trigger is removed, the user dismisses it, or its information is no longer valid.

Exception: The visual presentation of the additional content is controlled by the user agent and is not modified by the author.

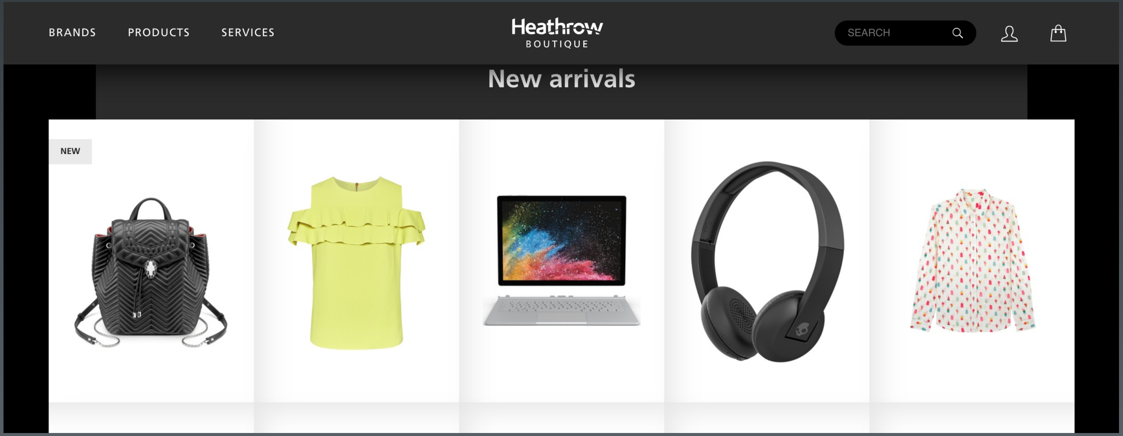
Perceivable

Scenario: A person with low vision who uses screen magnification software.

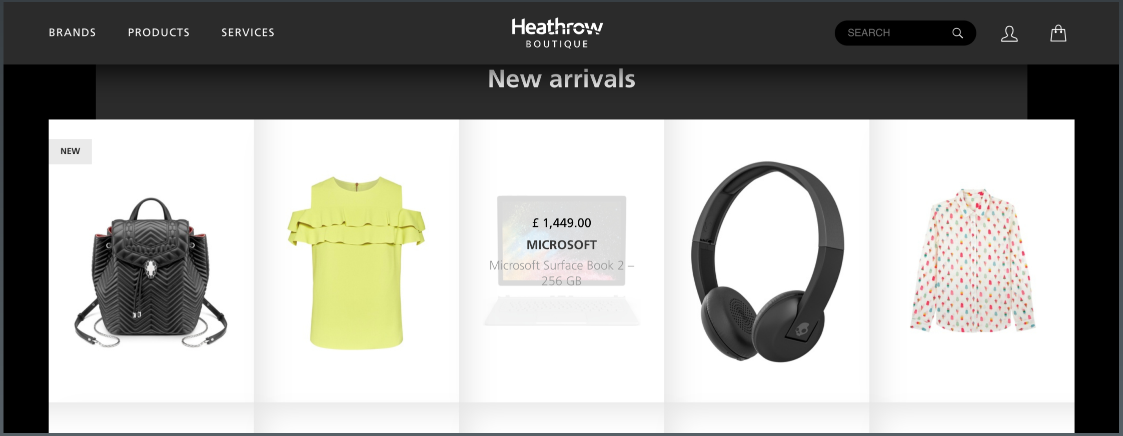
Problem: *“I was moving my mouse around to track what I was looking at on a web page. It helps me keep focused. Then -boom- this little box popped up. It covered what I was trying to read and I couldn't get it to go away.”*

Works well: *“I hovered over a word and a box popped up with the definition, but it was mostly off the screen with my magnification. I moved my mouse pointer to the definition box and scrolled the magnified area over to the definition box and it stayed popped up so I could read it.”*

Perceivable



Perceivable



Operable

Interface forms, controls, and navigation are operable

Operable

2.1 Keyboard Accessible: Make all functionality available from a keyboard

2.2 Enough Time: Provide users enough time to read and use content

2.3 Seizures: Do not design content in a way that is known to cause seizures

2.4 Navigable: Provide ways to help users navigate, find content, and determine where they are

Operable

2.1.2 No Keyboard Trap (Level A)

- Keyboard focus is never locked or trapped at one particular page element. The user can navigate to and from all navigable page elements using only a keyboard.

Operable

2.1.2 No Keyboard Trap (Level A)

- Keyboard focus is never locked or trapped at one particular page element. The user can navigate to and from all navigable page elements using only a keyboard.

Q: *“But what about modal layers?”*

Operable

2.1.2 No Keyboard Trap (Level A)

- Keyboard focus is never locked or trapped at one particular page element. The user can navigate to and from all navigable page elements using only a keyboard.

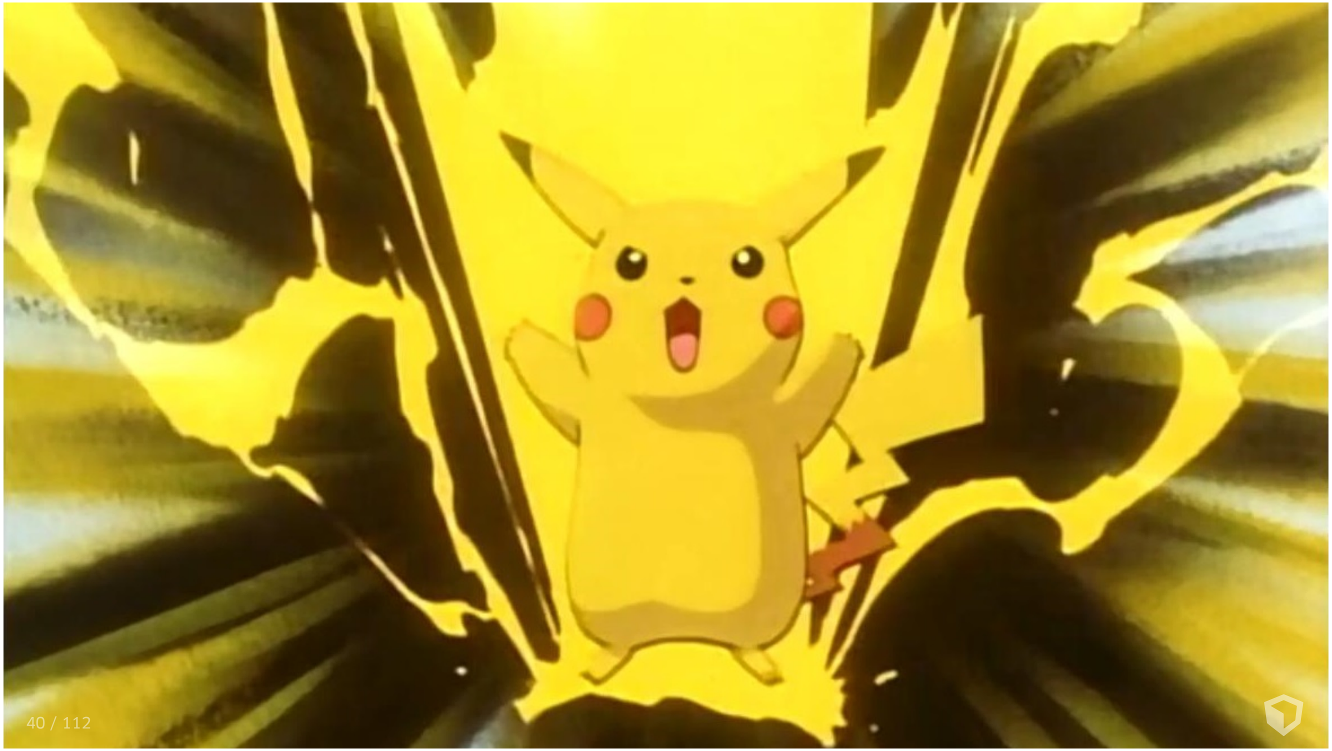
Q: *“But what about modal layers?”*

A: *“If you have a trapped mouse user, you can as well have a trapped keyboard user.”*

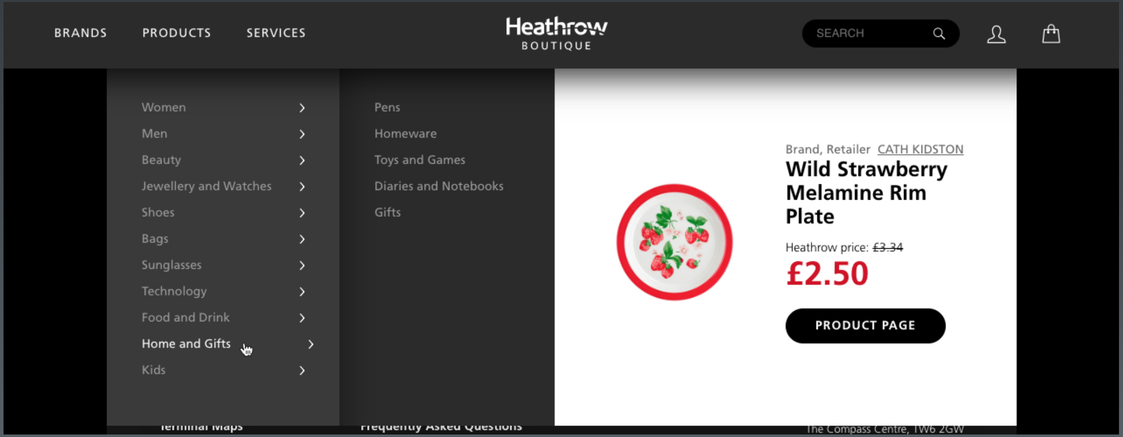
Operable

2.3.1 Three Flashes or Below Threshold (Level A)

- No page content flashes more than 3 times per second unless that flashing content is sufficiently small and the flashes are of low contrast and do not contain too much red.



Operable


BRANDS
PRODUCTS
SERVICES
Heathrow
BOUTIQUE
SEARCH
Women
Men
Beauty
Jewellery and Watches
Shoes
Bags
Sunglasses
Technology
Food and Drink
Home and Gifts
Kids
Pens
Homeware
Toys and Games
Diaries and Notebooks
Gifts
Brand, Retailer CATH KIDSTON
Wild Strawberry Melamine Rim Plate
Heathrow price: ~~£3.34~~
£2.50
PRODUCT PAGE
Terminal Maps
Frequently Asked Questions
The Compass Centre, TW6 2GW

Understandable

3.1 Readable: Make text content readable and understandable

3.2 Predictable: Make Web pages appear and operate in predictable ways

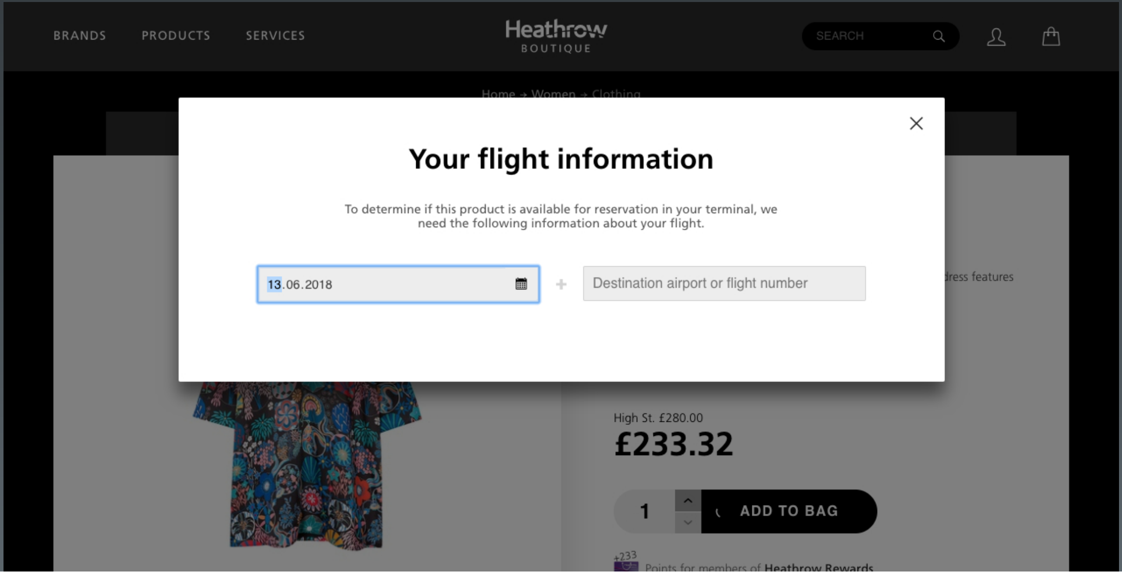
3.3 Input Assistance: Help users avoid and correct mistakes

Understandable

3.2.2 On Input (Level A)

- When a user inputs information or interacts with a control, it does not result in a substantial change to the page, the spawning of a pop-up window, an additional change of keyboard focus, or any other change that could confuse or disorient the user unless the user is informed of the change ahead of time.

Understandable



Your flight information

To determine if this product is available for reservation in your terminal, we need the following information about your flight.

13.06.2018 + Destination airport or flight number

High St. £280.00
£233.32

1

+233 Points for members of Heathrow Rewards

Robust

4.1 Compatible: Maximize compatibility with current and future user agents, including assistive technologies

Robust

Common screen reader combinations

- JAWS + IE (24.7%)
- NVDA + Firefox (23.6%)
- JAWS with Firefox (15.1%)
- VoiceOver + (Mobile) Safari (10.0%)

ChromeVox and Windows Narrator are very rarely used, so be cautious in using them exclusively for testing.

webaim.org/projects/screenreadersurvey7/#browsercombos

Evaluating a11y

Evaluating a11y

- Doesn’t really make sense at the end of the project
- Integrate a11y into the project as a continuous process

Evaluating a11y

Methodologies

- Automated tools
- Checklists
- Usability testing
- Self-testing

Evaluating a11y

Automated tools

- Chrome a11y audit
  - Accessibility Developer Tools
- aXe
  - aXe Chrome extension
  - aXe Firefox extension
  - aXe-core
- Pa11y

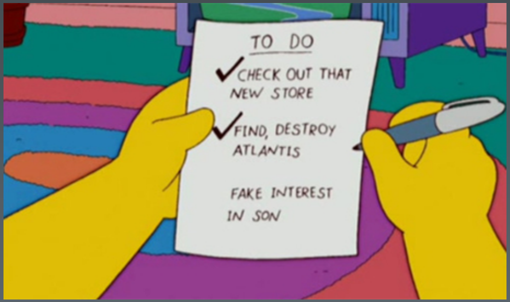
Evaluating a11y

Use automated tools

Evaluating a11y

Use check lists

- www.w3.org/WAI/WCAG21/quickref
- webaim.org/standards/wcag/checklist

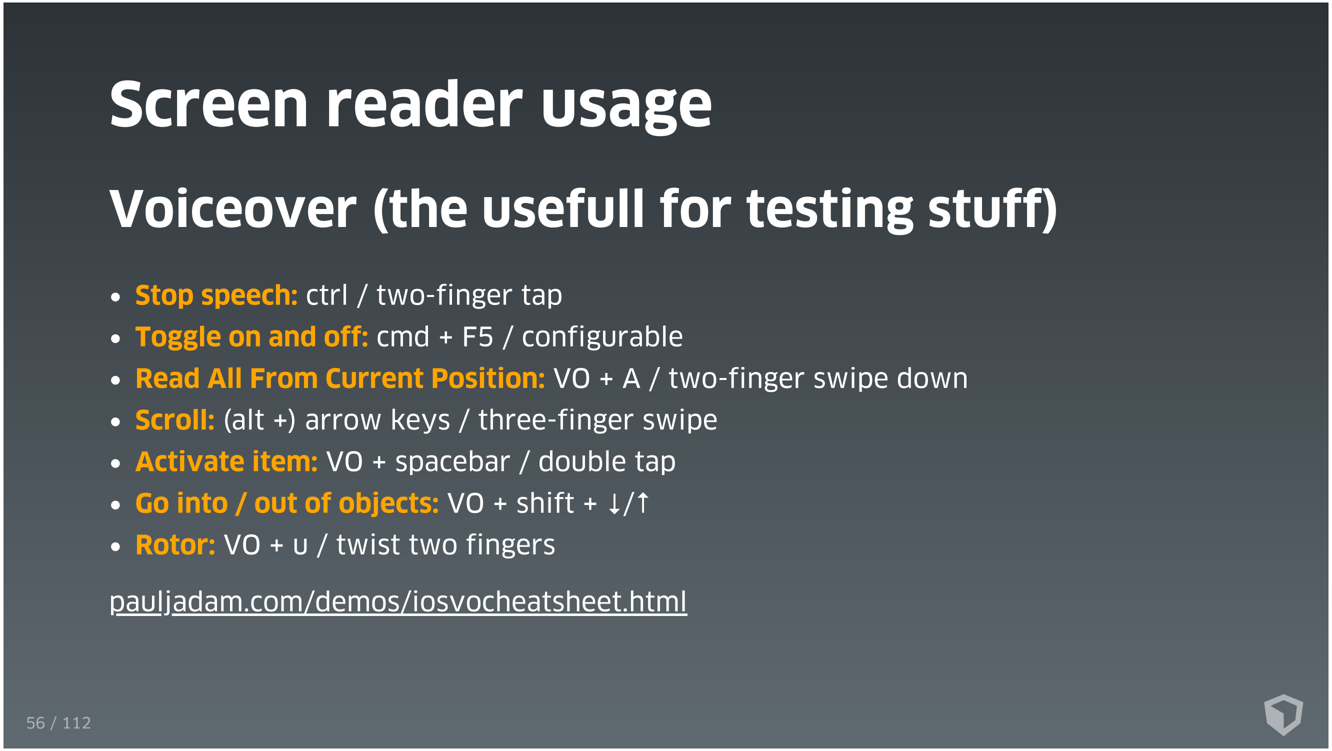


Evaluating a11y

Conduct usability testing

- Do not conduct **accessibility testing** with users with disabilities
- Conduct **usability testing** and **include users with disabilities**

A11y is about **people** –
only **people** can evaluate true a11y

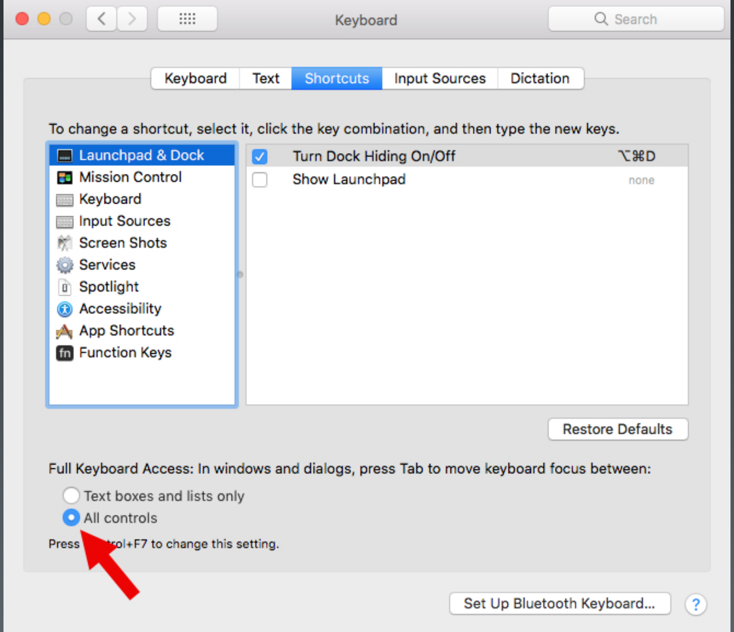


Screen reader usage

Voiceover (the usefull for testing stuff)

- **Stop speech:** ctrl / two-finger tap
- **Toggle on and off:** cmd + F5 / configurable
- **Read All From Current Position:** VO + A / two-finger swipe down
- **Scroll:** (alt +) arrow keys / three-finger swipe
- **Activate item:** VO + spacebar / double tap
- **Go into / out of objects:** VO + shift + ↓/↑
- **Rotor:** VO + u / twist two fingers

pauljadam.com/demos/iosvocheatsheet.html



To change a shortcut, select it, click the key combination, and then type the new keys.

☑	Turn Dock Hiding On/Off	⌥⌘D
□	Show Launchpad	none

Full Keyboard Access: In windows and dialogs, press Tab to move keyboard focus between:

○ Text boxes and lists only

● All controls

Press trol+F7 to change this setting.

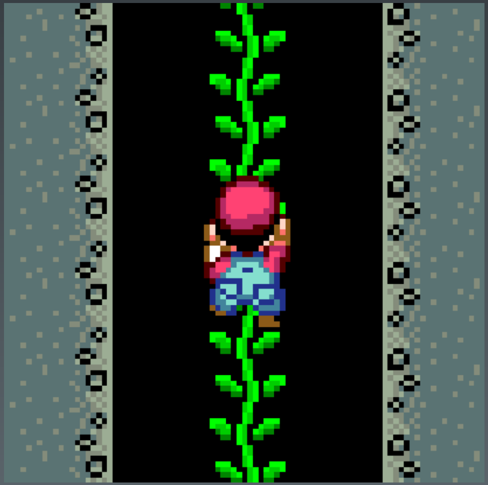
Screen reader usage

Jaws (the usefull for testing stuff)

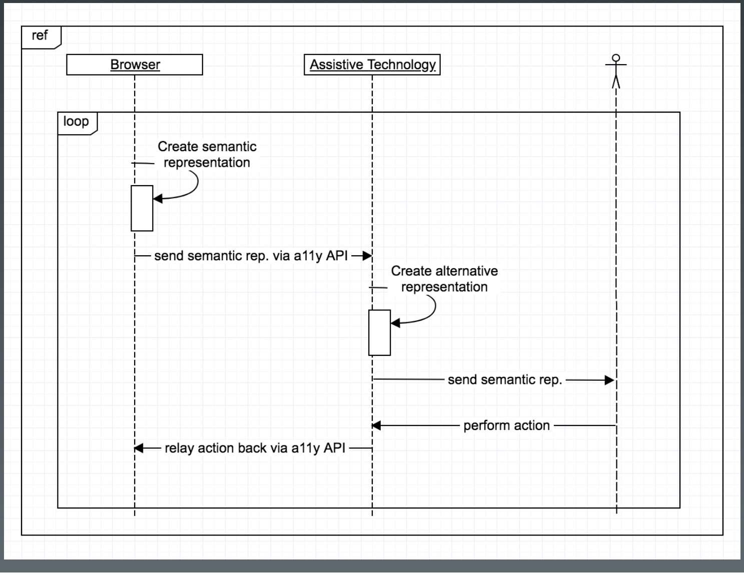
- **Stop speech:** ctrl
- **Toggle on and off:** insert + spacebar, s
- **Read All From Current Position:** insert + ↓
- **Activate item:** ↵ or spacebar
- **Enter forms mode:** ↵ (in a form element)
- **Exit forms mode:** +
- **Go to next heading:** H
- **Go to next landmark/region:** R

doccenter.freedomscientific.com/doccenter/archives/training/jawskeystrokes.htm

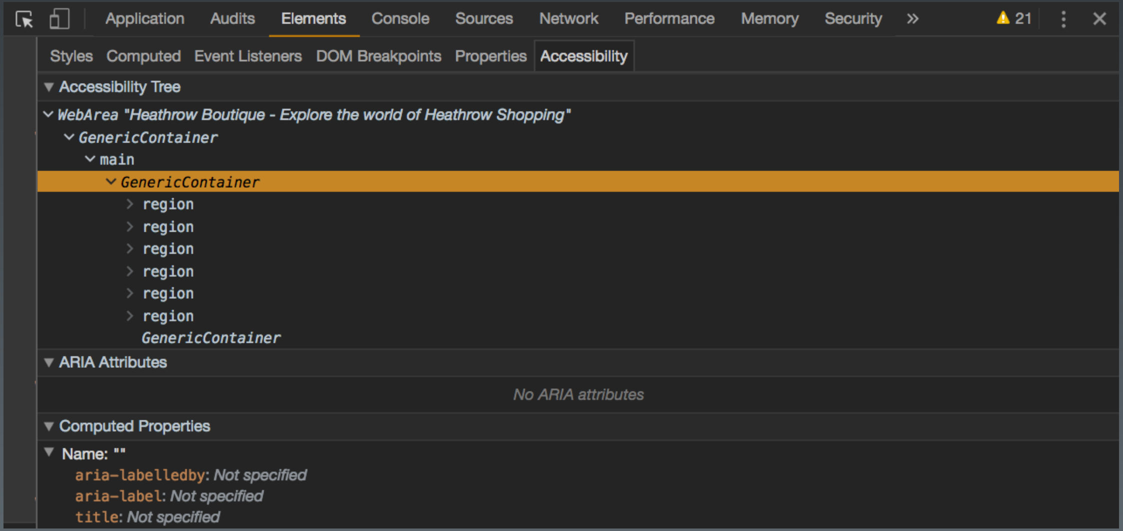
The a11y tree



The a11y tree


ref
Browser
Assistive Technology
loop
Create semantic representation
send semantic rep. via a11y API
Create alternative representation
send semantic rep.
perform action
relay action back via a11y API

The a11y tree



```
▼ Accessibility Tree
⌄ WebArea "Heathrow Boutique - Explore the world of Heathrow Shopping"
  ⌄ GenericContainer
    ⌄ main
      ⌄ GenericContainer
        › region
        › region
        › region
        › region
        › region
        › region
          GenericContainer
▼ ARIA Attributes
                    No ARIA attributes
▼ Computed Properties
▼ Name: ""
    aria-labelledby: Not specified
    aria-label: Not specified
    title: Not specified
```

Focus

Focusable stuff

1. **I’m a button**
2. I’m a focusable link
3. I’m a not-focusable link
4. I’m another focusable link
5. I’m a focusable link with a weird tab order
6. I’m a focusable link as well
7. I’m a stupid span
8. I’m a focusable span
9. I’m a programatically focusable span

Operable

2.4.3 Focus Order (A)

- The navigation order of links, form elements, etc. is logical and intuitive.

2.4.7 Focus Visible (AA)

- It is visually apparent which page element has the current keyboard focus (i.e., as you tab through the page, you can see where you are).

Focus within

I’m a button

- Simple polyfill: https://gist.github.com/aFarkas/a7e0d85450f323d5e164
- There are also PostCSS plugins out there

“Freak out” mode

(When focused elements disappear)

Focus traps

Note: The user can always escape such traps (for instance using the VoiceOver rotor).

Focus

- Don’t use tabindex > 0 unless you really know what you’re doing
- Beware of the **“freak out mode”**, circumvent it by setting focus
- Be carefull with [redacted], your focus ring might get hidden
- Check out github.com/NV/flying-focus
- Use focus traps wisely
- Hash tag links do not set focus reliably (see axesslab.com/skip-links)

Hiding content

Hiding content from everyone

- [redacted]
- [redacted] (allows applying delays by using CSS transitions)
- HTML [redacted] attribute

Hiding content visually

Use judiciously!

Hiding content from screen reader

The first rule of ARIA: Don’t use ARIA

(unless really necessary)

Inert

> *“When a node is inert, then the user agent must act **as if** the node was absent for the purposes of targeting user interaction events, may ignore the node for the purposes of text search user interfaces (commonly known as “find in page”), and may prevent the user from selecting text in that node.”*

Furthermore, a node which is **inert** should also be **hidden from assistive technology**.

html.spec.whatwg.org/multipage/interaction.html#inert

Polyfill: github.com/WICG/inert

- Sets [redacted] on everything focusable in the inert subtree
- Sets [redacted] on the inert subtree
- Sets [redacted] on the inert subtree

Hiding content

- Do **not** use [redacted], use [redacted] instead
  (A screen reader will “see” an element with [redacted] even when it is inside a container with [redacted])
- Note that setting [redacted] on a container alone does **not** remove its content from the tab-order
- Note that [redacted] is different from [redacted]. It is used to **remove semantic meaning** from an element and any of its related child elements. **The content of elements with [redacted] will still be picked up by assistive technology!**

Labeling content

Labeling content

1. Foo
2. Foo
3. Foo
4. Foo
5. Foo
6. Foo
7. Foo

Labeling content

I18n

1. It is verboten to write bad code.
2. It is verboten to write bad code.

Labeling content

Abbreviations

That joke made me LOL big time.

Labeling content

CSS effects

1. ADD TO CART
2. ADD TO CART
3. ADD TO CART

Labeling content

CSS pseudo elements

[redacted] / [redacted] are **not** a11y supported in IE11. So do not rely on them.

Labeling content

ARIA SSML (proposal)

github.com/mhakkinen/SSMLinHTMLproposal

Landmarks

Landmarks

HTML elements defining ARIA landmarks by default:

Note: When using multiple sectioning elements of the same type on the same page, consider giving each of them a title / aria-label which allows for a clear distinction.

w3.org/TR/wai-aria-practices/examples/landmarks

Live regions

Live regions

Accessible forms

Type in a number lower than 3:

Live regions

Alerting the user via JavaScript

Live regions

Giving a heads up via JavaScript

Live regions

- You should be **polite** ([redacted] is better supported than [redacted])
- **Note:** Screen readers “buffer” pages as they are loaded. Any content that is added after page load may not be picked up by the screen reader
- **Note:** Screen readers can only focus on one part of the page at a time. If something changes on another area of the page, screen readers may not pick it up
- [redacted] should not be used for non-critical purposes
- [redacted] has some support issues
- **Test, test, test!**

ARIA Roles

ARIA Roles

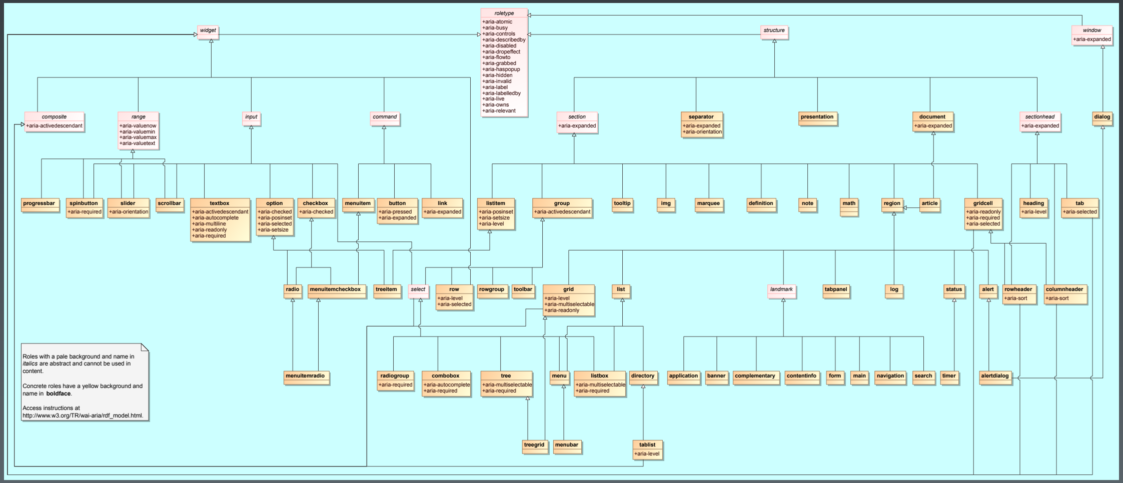
Abstract roles (**do not use!**):

Widget roles:

Document structure roles:

Landmark roles:

ARIA Roles



Roles with a pale background and name in *italics* are abstract and cannot be used in content.

Concrete roles have a yellow background and name in **boldface**.

Access instructions at http://www.w3.org/TR/wai-aria/rdf_model.html.

ARIA Roles

A role is a **promise**

[redacted]

for instance is a promise that the author of that [redacted] has also incorporated JavaScript that provides the keyboard interactions expected for a button. Unlike HTML input elements, **ARIA roles do not cause browsers to provide keyboard behaviors** or styling.

If you use an ARIA role, make sure you conform with the according **design pattern**: www.w3.org/TR/wai-aria-practices-1.1/#aria_ex

ARIA Roles

Example: Menu or menu bar

Keyboard Interaction:

The following description of keyboard behaviors assumes: A horizontal menubar containing several menuitem elements. All items in the menubar have child submenus that contain multiple vertically arranged items. Some of the menuitem elements in the submenus have child submenus with items that are also vertically arranged. When reading the following descriptions, also keep in mind that: Focusable elements, which may have role menuitem, menuitemradio, or menuitemcheckbox, are referred to as items. If a behavior applies to only certain types of items, e.g., menuitem elements, the specific role name is used. Submenus, also known as pop-up menus, are elements with role menu. Except where noted, menus opened from a menubutton behave the same as menus opened from a menubar. When a menu opens, or when a menubar receives focus, keyboard focus is placed on the first item. All items are focusable as described in 5.6 Keyboard Navigation Inside Components. Enter: When focus is on a menuitem that has a submenu, opens the submenu and places focus on its first item. Otherwise, activates the item and closes the menu. Space: (Optional): When focus is on a menuitemcheckbox, changes the state without closing the menu. (Optional): When focus is on a menuitemradio that is not checked, without closing the menu, checks the focused menuitemradio and unchecks any other checked menuitemradio element in the same group. (Optional): When focus is on a menuitem that has a submenu, opens the submenu and places focus on its first item. (Optional): When focus is on a menuitem that does not have a submenu, activates the menuitem and closes the menu. Down Arrow: When focus is on a menuitem in a menubar, opens its submenu and places focus on the first item in the submenu. When focus is in a menu, moves focus to the next item, optionally wrapping from the last to the first. Up Arrow: When focus is in a menu, moves focus to the previous item, optionally wrapping from the first to the last. (Optional): When focus is on a menuitem in a menubar, opens its submenu and places focus on the last item in the submenu. Right Arrow: When focus is in a menubar, moves focus to the next item, optionally wrapping from the last to the first. When focus is in a menu and on a menuitem that has a submenu, opens the submenu and places focus on its first item. When focus is in a menu and on an item that does not have a submenu, performs the following 3 actions: Closes the submenu and any parent menus. Moves focus to the next menuitem in the menubar. Either: (Recommended) opens the submenu of that menuitem without moving focus into the submenu, or opens the submenu of that menuitem and places focus on the first item in the submenu. Note that if the menubar were not present, e.g., the menus were opened from a menubutton, Right Arrow would not do anything when focus is on an item that does not have a submenu. Left Arrow: When focus is in a menubar, moves focus to the previous item, optionally wrapping from the last to the first. When focus is in a submenu of an item in a menu, closes the submenu and returns focus to the parent menuitem. When focus is in a submenu of an item in a menubar, performs the following 3 actions: Closes the submenu. Moves focus to the previous menuitem in the menubar. Either: (Recommended) opens the submenu of that menuitem without moving focus into the submenu, or opens the submenu of that menuitem and places focus on the first item in the submenu. Home: If arrow key wrapping is not supported, moves focus to the first item in the current menu or menubar. End: If arrow key wrapping is not supported, moves focus to the last item in the current menu or menubar. Any key that corresponds to a printable character (Optional): Move focus to the next menu item in the current menu whose label begins with that printable character. Escape: Close the menu that contains focus and return focus to the element or context, e.g., menu button or parent menuitem, from which the menu was opened. Tab: Moves focus to the next element in the tab sequence, and if the item that had focus is not in a menubar, closes its menu and all open parent menu containers. Shift + Tab: Moves focus to the previous element in the tab sequence, and if the item that had focus is not in a menubar, closes its menu and all open parent menu containers. NOTE Disabled menu items are focusable but cannot be activated. A separator in a menu is not focusable or interactive. If a menu is opened or a menubar receives focus as a result of a context action, Escape or Enter may return focus to the invoking context. For example, a rich text editor may have a menubar that receives focus when a shortcut key, e.g., alt + F10, is pressed while editing. In this case, pressing Escape or activating a command from the menu may return focus to the editor. Although it is recommended that authors avoid doing so, some implementations of navigation menubars may have menuitem elements that both perform a function and open a submenu. In such implementations, enter and Space perform a navigation function, e.g., load new content, while Down Arrow, in a horizontal menubar, opens the submenu associated with that same menuitem. When items in a menubar are arranged vertically and items in menu containers are arranged horizontally: Down Arrow performs as Right Arrow is described above, and vice versa. Up Arrow performs as Left Arrow is described above, and vice versa.

www.w3.org/TR/wai-aria-practices-1.1/#menu

No ARIA is better than bad ARIA

www.w3.org/TR/wai-aria-practices-1.1/#no_aria_better_bad_aria

Accessible SVG

Accessible SVG

Does the graphic have a function?
If so, it should be conveyed to the user.

Accessible SVG

Basic image replacement

Use plus an resp. an attribute:

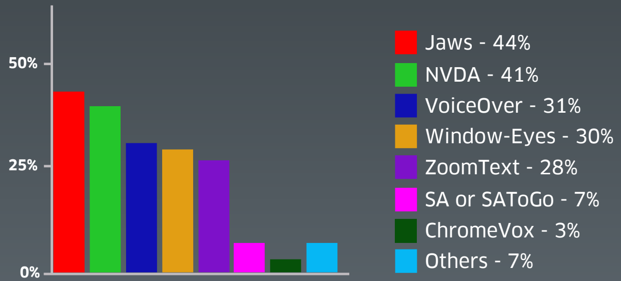
Mmmmmmm... 🍩!

Please don't...

Accessible SVG

SVG charts

Commonly Used Screen Readers



Data & original image from WebAIM's Screen Reader User Survey #6 Results

Accessible SVG

Charts

Misc

“Aria-Controls is 💩”

Or: Should or shouldn’t I stick to the specs?

> *“We need to talk about [redacted]. It’s poorly supported, does very little, and does what it does when it does badly. It is poop and we rely on it way too much. We are short-changing assistive technology users when we do.”*

heydonworks.com/article/aria-controls-is-poop

Recommendations:

- If it **serves less than it disrupts** (too much noise etc.), **don’t use it**
- Else **use it**
- **Test** with the assistive technology you want or have to support

I have a SPA. [redacted] or [redacted]?

On single page apps sometimes the view changes significantly, while the user seems to stay on the same page. WAI says:

> *“Note: If pressing the link triggers an action but does not change browser focus or page location, authors are advised to consider using the button role instead of the link role.”*

w3.org/WAI/PF/aria/roles#link

Recommendations:

- If you haven’t accounted for **history navigation**, use a button, else you **may** use a link, which promises the posibility to go back and forward in (browser) history
- Other than that: As long as the user is **clear about what will happen upfront**, you may use whatever fits best your use case

I have a SPA

More Recommendations on SPAs:

- Use structural elements ([redacted], [redacted] etc.)
- Update page title to reflect content state
- Keyboard navigation
  - Ensure only visible elements are navigable
  - Set focus when necessary
  - Use live regions for messaging, if necessary
- Read: developer.paciellogroup.com/blog/2018/01/a-tale-of-two-rooms-understanding-screen-reader-navigation/

🍞

Problem: Most of the time toasts appear at a different spot on the page, far away from where the user focus is.

Recommendations:

- Make the toast appears **at the top of the page** (DOM/a11y-tree)
- Use a titled live region (in most cases [redacted] will suffice)
- Do **not** try to trap the focus inside the toast
- Make the toast **dismissible**
- Make sure you follow the web content accessibility guidelines

Final words

A website that is not accessible is a website with disabilities.

Designing for a11y is part of making a website **awesome!**

… and not getting sued

Making a website accessible **is not that hard** if you make it part of your daily routine and make sure everyone in your team follows along.

TO START
PRESS ANY KEY