Web Accessibility

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

Tim Berners-Lee https://www.w3.org/standards/webdesign/accessibility

Preface whole thing with I am not an expert!!
In principle who would disagree but a11y is often overlooked - its not baked into our AC/requirements so it becomes an afterthought. We do it but no consistently and often by accident.

Now building public facing stuff, although is important for non-public stuff as well.

No legal obligaition in the UK...yet...

Aims

- The principles underlying web accessibility
- How can we make accessibility issues more visible to us?
- What are the key things to consider?
- What steps can we take today to start moving in the right direction?

- it's invisible to us (obvious if css is broken, not obvious if a11y issues)
- there's a lot to consider and learn - overwhelming - different groups, different needs, standards are lengthy and dull etc

61% of devs who answered survey by edge team dev do no accestblity testing

Web accessibility encompasses all disabilities that affect access to the Web, including:

- auditory
- cognitive
- neurological
- physical
- speech
- visual

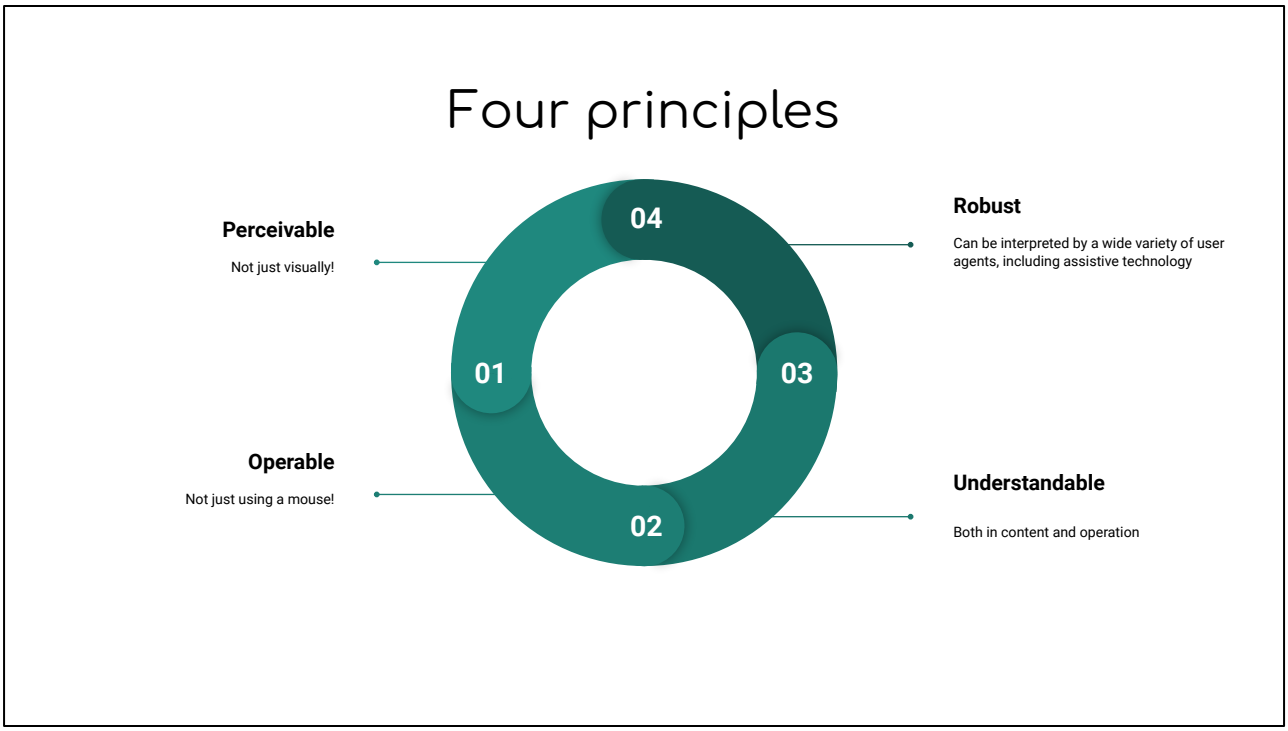
Web accessibility also benefits people *without* disabilities, for example:

- people using mobile phones, smart watches, smart TVs, and other devices with small screens, different input modes, etc.
- older people with changing abilities due to ageing
- people with "temporary disabilities" such as a broken arm or lost glasses
- people with "situational limitations" such as in bright sunlight or in an environment where they cannot listen to audio
- people using a slow Internet connection, or who have limited or expensive bandwidth

https://www.w3.org/WAI/fundamentals/accessibility-intro/

Something like 1 in 5 people in Uk has some type of disability - there's a wide range and wide range of resultant needs...which makes it even more daunting

But it benefits us all...and overlaps with seo so there's extra weight there to the business argument



wcag - web content accessibility guidelines

for each principle there are guidelines with success criteria, along with sufficient and advisory techniques

Tools

- eslint-plugin-jsx-a11y
- Axe
- Lighthouse
- NVDA/Narrator (or Voiceover for Mac)
- Dev tools accessibility inspector

tools not a panacea - don't pick up everything - eg can automate some stuff but no current way of automating screen reader tests’

JAWS most used but nvda is free!

Will use Axe a couple of times in this talk

DEMO DEV TOOLS

So, what are the key areas to think about?

As I said, there's a huge amount of stuff, far more than I can cover in an hour long talk...so will cover the key areas alhtough its not exhaustive (e.g. not going to talk about concerns around video and audio content and not going to cover some stuff you probably already know like alt tags and link text)

Contrast

- Use tools to review colour contrast (e.g Axe)
- This is one of several areas where design and dev need to work together

- Contrast: OK this is more design, and for some customers will be out of our controle to some extent - obviously a lot of accessibility is rooted in the design of the page so we need to work hand in hand, even though I'm obviously gonna focus more on the dev side

Gov.uk is a beacon of accessibility - simple design and since they are single source of info and have to be accessible design is much lower priority - but run axe...get colour contrast warnings

Oh, and don’t rely on colour alone to indicate things...colour-blindness

Zoom

- Use the browser zoom
- Testers could test the zoom functionality on other devices

- Zoom: responsive design can help with this, try zoom see what happens (200%)...meta tags…

Guardian as example get over 200 and goes to mobile...

Focus Management

- Use focusable elements for interactive items (or use tabIndex = 0)
- Make sure focused element is signalled visually
- Ensure that focus is handled correctly for modals etc:
  - Their first interactive element should get focus
  - Navigation by keyboard should not result in ending up with focus outside the modal while it is open (focus trap)
- Hidden focusable elements should not be able to be tabbed to: toggle visibility:hidden
- Client side routing means need to consider where focus should go (and ensure the change gets announced)

Visibility cf keyboard nav
Also important for screen readers (announcing)

Gov.uk has skip links a fugly but visible style

Santander starts off well but rapidly get nothing visible

```
document.body.addEventListener('focusin', (event) => {
    console.log(document.activeElement)
})
```

Client side routing - no page refresh - less simple to solve, various solutions - Gatsby research - focus on a heading gave best experience for screen readers and having visible focus was crucial for speech recognition software
They also tests with users using zoom. Slightly different results for different techs illustartes the issue...
Conclusion was that using skip links (small non visible areas that get focus -> visible, can announce change and can allow navigation) was maybe best...perhaps with an aria live region that updates on client side navigation

Keyboard Navigation

- Can you navigate just using the keyboard?
- Keyboard interaction patterns are set out in the WAI-ARIA Authoring Practices so you don't need to guess! https://www.w3.org/TR/wai-aria-practices-1.1/

Semantic html and ARIA

- If at all possible use the appropriate html element
  - But some feature may not be implemented in html or you may have design considerations that cannot be accommodated using the native element…
- Aria allows information to be added to elements to allow assistive technology to interpret non-standard widgets etc
- First rule of aria-club: no aria is better than bad aria
  - Roles are a promise of certain behaviour
  - Aria will sometimes override the built-in accessibility semantics
- Aria comprises:
  - Roles - what it is
  - Properties - other immutable information e.g. if its is required
  - States - the current condition of the element

Html gives us a load of stuff for free...

For AT such as screen readers and to some extent speech recognition etc its the markup code that counts - uses the accessibility tree that you can see in dev tools
I have been playing with nvda but no an expert - resources link to video of screen reader user

Sometimes you need to do aria - eg tab panels, accordion which don't exist
Accessible Rich Internet Applications
Aria is mostly for screen readers
No aria better than bad aria - . webAIM million: sites with aria more likely to have accessibility issues!!

eg the broken promise of role="button" on a div

- Tabindex to make it focusable
- onKeyDown Event handler for space/enter key presses because only mouseclick works ootb for onClick (with an actual html button, onClick does the key presses as well)

So not a huge amount of work in this case but it can soon get more complex when using combinations of elements for things like custom selects

The aria authoring practices and mdn should be consulted!

Structure and landmarks

- All pages have a structure and this should be reflected in the markup
  - For example using <main> for the main content
- Headings also form part of this
  - Should be using them in order
  - Shouldn't be using them just for visual impact

Visually it doesn't matter if use or html semantic structure but it really matters for AT

E.g. screen reader can summarise what sections the page has and offer ways of skipping between different header elements (bit like when we glance at the page to take it all in visually) - it can also do this with links which is why your link text should make sense taken out of context (e.g. click here to see lions as opposed to click here)

in MySight we are more using h tags for visual differences - we aren't using them all and sometimes we skip them - actually the question should be 'what is the most important heading on the page' - that's your H1 and it doesn't actually have to look much different or bigger thand h2 etc (although this presents us with something to consider re how we manage styles across pages with perhaps different hierarchies)

(Another thing to consider - again are your hidden elements hidden from AT (e.g. if you show a thank you message on same page after submit it needs to be hidden from screen readers, not just visually) - display: none will do it…
Conversely - you can hide things visually so only screen readers etc can 'see' them to add extra guidance etc)

So essentially something can make sense visually but not via AT...for example have a line to demarcate section visually but nothing else -> screen reader wouldn't know

- information architecture - include how we approach alt text etc

Motion

- Autoplay remains evil
- prefers-reduced-motion
- Do you really need that animation?

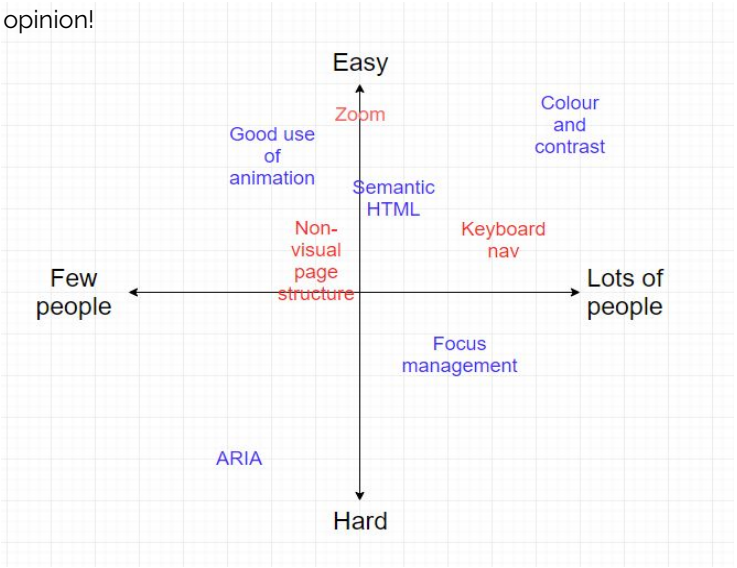
vestibular disorders/migraine/epilepsy/sensory processing disorders/adhd etc...also low quality/mobile devices

https://leedsdigitalfestival.org/

candidate recommedation for Update media query which would allow us to check device ability to update after render

Where should we focus our efforts?

A non-scientific opinion!



Screen readers as a litmus test - small % of users but if can be read by screen reader means its pretty accesible...also bear in mind that not just blind users (dyslexia, illiterate etc)
Need to consider though not only aria but combos of software and browser so it not totally simple - plus of course just like everyone diff screen reader users may have different preferences!

Look at MySight calendar/appt picker - is a good example of where we probably are with this stuff!
can tab through
BUT - once selected date have to tab to end to get to appts, and then back if want to change which is technically accessible but no so much usably accessible -

Wouldn't want to focus the appts on select as may want to just look but there may be a better patter to follow

- Totally unusable using a screen reader - they'd probably realised before now on a hdiing to nothing but it just reads out "1 button", 2 Button etc as you tab.
    - Once you select is silent - no indication anything on right has happened - proably use an aria-live region to announce changes

What can we start doing now?

- Use the right html element for the job
- Navigate using your keyboard
- Zoom in
- Start using a tool to flag issues
- Building a component? Look at the authoring practices and/or projects like Downshift and Reach UI: bake accessibility into our base components (but if using aria - test with a screen reader)
- Add accessibility issues to the mental list of things you look for on code reviews

Resources wiki page

Some stuff is more structural - AC/Requirements/Checklists/more integrated tooling, unit or integration tests/what user groups do we prioritise?
Some stuff is just harder...e.g. time to learn screen reader and researching how to best implement this stuff...Ideally - user testing/expert audit but manual testing is important (hard though if not an expert user of AT -see that gatsby results for how different solutions can impact users of AT differently