

You can view this presentation  
with full written notes here:



# The Accessible Web

A better Internet for Everyone

Accessibility is all about making the internet more usable for everyone.

Some people need more help than others, which is why we often focus on accessibility for deaf and blind people with captions or extra descriptions.

But there are small changes web developers and designers can make which make their websites much more usable for everyone.

Here I will outline some of these methods and hopefully encourage you all to consider accessibility not just in web, but whatever platforms you find yourselves working on.

```
<html lang="en">
  <body>
    <h1> Hello World!</h1>
  </body>
</html>
```

While most of you are probably familiar with HTML, here is a quick refresher.

The web runs on HTML (Hyper Text Markup Language). Even if you are working in React, Vue or Elm, it all ends up as HTML which is interpreted by the browser.

You have tags, which describe a type of component, embedded between angle brackets.

Tags can also have attributes, such as the lang attribute on the html tag, which describes our website as being en, or English.

This is already an important accessibility feature, as it tells the browser what language your website is, so it can offer translations for non English users. Just like most websites assume their users have perfect vision and comprehension, they also primarily cater towards English speakers.



# HTML 5

The first thing you should do for improving accessibility is to make full use of HTML5. This isn't new tech, it was released in 2008, with it's last major update in 2014. It has since been replaced with the HTML living standard.

But it was a major shift for web development

**<h1>**  
**<div>** **<b>**  
**<span>**  
**<img>** **<ul>**

But it was a major shift for web development by expanding on the limited set of HTML tags, like div, span, h1 to h5 ...

**<date>   <header>   <nav>**  
**<track>   <article>   <time>**  
**<color>   <audio>   <details>**  
**<article>   <ruby>   <video>**  
**<output>   <footer>   <mark>**

To a whole set of semantic tags for content, form inputs and even multimedia.

This is particularly a reminder for developers who rely on frameworks like React, Angular and Vue - there are more elements than `div` available for you.

Take for example the `<nav>` tag.



**<nav>**

Every website has a navigation menu, where you can find links to other pages on a site.

Inside a HTML file, these could be located anywhere, because they are positioned by CSS.

Imagine how that would be read out by a screen reader.

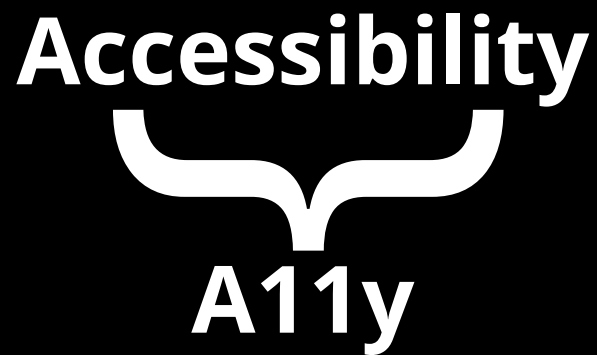
By placing your navigation menu inside a <nav> tag, you make it easy to find for screen readers and keyboard navigation.

# Accessibility

One thing which developers rarely get right are acronyms.

I know, it's rich coming from someone who named a community Galway Information Technology, or GIT - but that just means I know what I'm talking about.

If you are familiar with Kubernetes, you can guess how we decided to abbreviate



**Accessibility**

**A11y**

That's right, we took a 13 letter word, kept the first and last letter, and replaced the rest with the number 11.

I guess it's pronounced Ally, which is a nice sentiment. But imagine how a screen reader would read that out loud: Aeleveleny.



```
<span  
aria-label="accessibility">  
A11y  
</span>
```

At least when including this acronym in your website we have a solution:

The aria-label tag can be used to give an alternative text for a screen reader to read out.

There is also a specialized `<abbr/>` tag, but there is some debate on it's usage. I've included links with more details on the final reference page if you want to read more about it.

The aria-label is also great for icon-only menu items / links, though I would discourage those in general.

# **ARIA**

## **Accesible Rich Internet Applications**

Let's talk a little more about ARIA - it stands for Accessible Rich Internet Applications. It defines a list of HTML attributes which can be used to enhance your website's content with more descriptions and context.

We already saw the aria-label example, here are some more...

- **aria-description**
- **aria-describedby**
- **aria-labelledby**
- **aria-required**

Description is a longer version of label, useful for images and videos

If you already have a description in text on your page for an image, you can use describedby to link to it to save duplicating it

Similarly you can mark any element as labelled by another element (referencing it with it's ID). Which is normally only available for form input elements.

You can mark any element as required, such as terms and conditions (which are never read)

These are great for creating composite elements, such as those PIN input fields with separate boxes for each digit, to make them accessible for everyone.

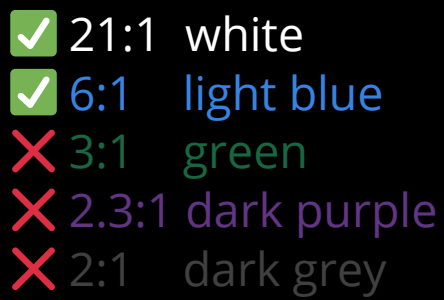
# Contrast

Now I've focused a lot on improving your website for screen readers. Let's look at some visual aspects:

- ✓ 21:1 white
- ✓ 6:1 light blue
- ✓ 3:1 green
- ✗ 2.3:1 dark purple
- ✗ 2:1 dark grey

I've made this presentation white on black for maximum contrast. This measures the max of 21:1 contrast ratio.

The recommendation is a minimum contrast of 3:1



- ✓ 21:1 white
- ✓ 6:1 light blue
- ✗ 3:1 green
- ✗ 2.3:1 dark purple
- ✗ 2:1 dark grey

But once font size decreases from the huge text I use, and we make the text non-bold, the legibility changes.

For such "normal" text a minimum contrast ratio of 4.5:1 applies.

# Fonts

While Contrast applies to everything in your website, let's take a closer look at fonts

- ✓ Strong, Bold Fonts
- ✓ Traditional Print with Serifs
- ✓ Anti Dyslexic
- ✗ ELEGANT
- ✗ *Cursive or Handwriting*

The default font of Open Sans / Sans Serif is a great choice

Traditional Print fonts with serifs like Times New Roman are great too for helping distinguish your I, l and 1s

There are even fonts like Open Dyslexic which are designed to help those with difficulty processing written words and numbers.

Fonts you should avoid using heavily in websites are elegant fonts - tall and thin, especially when not using good contrast.

Various cursive or handwritten fonts are probably the worst for legibility.



# CSS

This leads nicely to styling websites with CSS, or cascading style sheets.

Because thanks to CSS, it is possible for users to set their own preferred styles for websites, such as overriding your choice of font family, color or size.

- ✗ Over use `!important`
- ✗ Use `content`
- ✓ Set `speak` or `speak-as`
- ✓ Dark / light modes

What you shouldn't do, is use the important modifier to force styles to take precedence, because this can prevent users from using their own styles.

Similarly any text set via the css content property is to be avoided as it doesn't get picked up by screen readers.

Something new I learned for this talk is the speak css property, which let's you set if an elements content should be read out loud or skipped by screen readers.

the speak-as property lets you tell screen readers to spell out abbreviations or skip punctuation.

Websites can detect if users prefer dark / light mode and create a color scheme for each - this again lets users make websites easier for them to process.

# ✗ Scroll Hijacking

Now for a few important things not to do:

This is my personal biggest annoyance. Websites which interfere with scrolling to show / hide content, or rotate 3d objects look pretty but are frustrating and a usability nightmare.

It's fine for a tech demo, but if there's content you want people to read, I recommend against it.

# Full screen popups

Full screen popups are the worst part to come out of European GDPR restrictions.

If you need these, make them appear instantly, so that they don't appear as I'm already clicking on them.

Test them in all browsers to ensure they display correctly, and don't require scrolling into view.

And make them easily dismissable.

# Autoplay audio/video

Thankfully autoplay is disabled in modern browsers, but there are some workarounds possible.

Please don't use these and leave it to users to decide what to play and when.

# Javascript links

This is a another personal appeal.

HTML anchor links work wonderfully for moving between pages and can be styled to look like anything.

Please use them rather than click listeners which navigate via javascript. Such links cannot be opened in new tabs using my favorite ctrl+click or middle mouse click.

Use the HTML elements available rather than reinventing the wheel.

# **breathe...**

Phew....

OK that was a lot, and I know I tend to talk fast, so I need to remind myself to take a deep breath.

So how can you remember to do all of this when working on websites?

Luckily there are a lot of tools to help evaluate your websites accessibility.

I have listed them in the references at the end of the slides, but the most important to know is probably...



# Google Lighthouse

Google Lighthouse is built into the Chrome browser and can give your website an accessibility score and recommendations on how to improve it.

And this also leads onto the hidden link between accessibility and SEO / analytics.

If you make your website easier for humans to read, it's also easier for bots to crawl, therefore it can rank higher in search results and reach more people!



demo / end

# References

- [Web Content Accessibility Guidelines](#)
- [Web Accessibility Initiative - Accessible Rich Internet Applications](#)
- [Mozilla Accessibility Documentation](#)
- [abbr vs. aria label](#)
- [HTML5 Tags](#)
- [HTML standard](#)
- [css speak](#)
- [HTML Living Specification \(for devs\)](#)

# References (2)

- [contrast checker](#)
- [Google Chrome Lighthouse](#)
-