Accessibility is for everyone

Tiffany Prosser
Leicester City Council
tiffany.prosser@leicester.gov.uk

Overview

- Why does Web Accessibility matter?
- What do we mean by ‘Web Accessibility’?
- Making websites accessible
- The Umbraco accessibility project

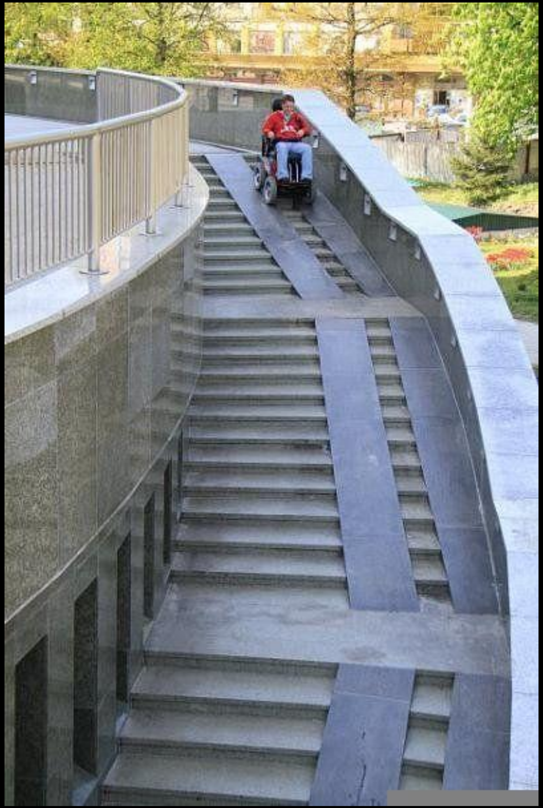
What do we mean by Accessibility?

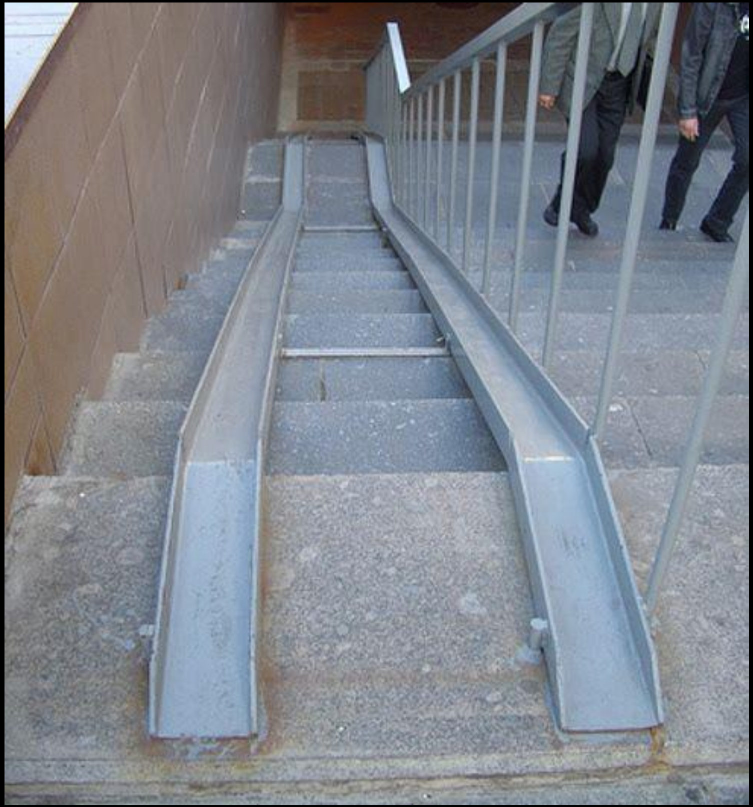
What do we mean by accessibility?

What do we mean by accessibility?

What do we mean by accessibility?

What do we mean by accessibility?





What do we mean by accessibility?

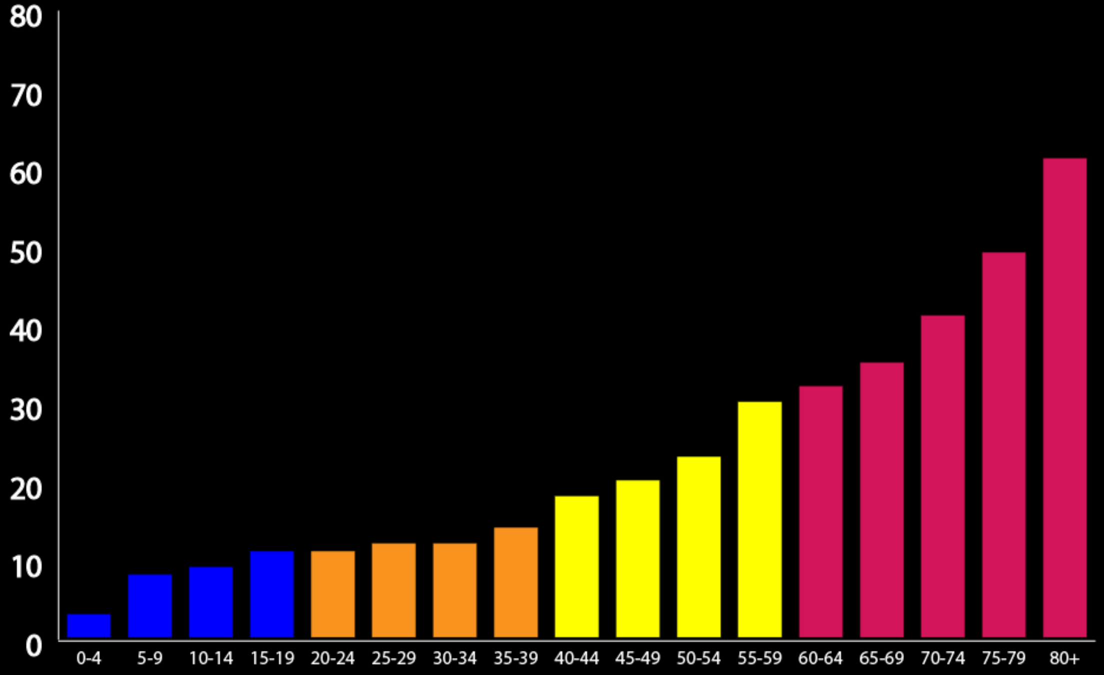
Underpinned by core concepts:

- Universal Design
- Usability
- Technology

Why does Web Accessibility matter?

Who does it affect?

Why does accessibility matter?



Disability by age 2016/17 UK

Family Resources survey
Department Work & Pensions

Why does accessibility matter?

25%

of the entire global population has a disability or impairment

Source: Towards Disability Confidence - Community Business April 2011

Who does it affect?

- Physical: Motor, mobility and dexterity
- Visual: Blindness, sight loss, colour blindness
- Auditory impairments: Hearing loss, deafness, Tinnitus
- Neurological: Seizure and vestibular disorders
- Cognitive, learning, attention and information processing
- Multiple disabilities

But that’s not all…

- Ageing population
- Temporary disability
- Language barriers
- Cultural barriers
- Economical factors

Enter…

Web Accessibility

What is Web Accessibility?

Accessibility is essential for developers and organizations that want to create high quality websites and web tools, and not exclude people from using their products and services.

World Wide Web Consortium (W3C)

Web Accessibility Standards

Web Content Accessibility Guidelines (WCAG)

Web Content Accessibility Guidelines (WCAG) is developed through the W3C process in cooperation with individuals and organizations around the world, with a goal of providing a single shared standard for web content accessibility that meets the needs of individuals, organizations, and governments internationally.

Web Accessibility Standards

Success criteria for web accessibility is divided into 4 key principles known as POUR

- Perceivable
- Operable
- Usable
- Robust

Web Accessibility Standards

These principles are then divided into levels in order to measure how well a site meets the criteria

- A
- AA
- AAA

How do I find out my site is accessible?

Number of other initiatives to improve web accessibility:

- The a11y project
- SiteImprove
- aXe by Deque
- Google Lighthouse
- Visual Studio accessibility extension
- Visual Studio Code accessibility extension
- Your own common sense

What would an accessible website look like?

- No mouse or touch
- No vision
- No hearing
- Few distractions, clear path, understandable language
- No seizure or motion issues
- Any combination of the above

Site accessibility checks

- Validate your code
- Test with a keyboard
- Check for alt-text
- Check the colour
- Check for proximinity
- Check form labels and form validation

How does it work?

Platform Accessibility API’s:

- Assistive technology uses API’s to get at information in the operating system
- HTML is read by the browser which assistive technology retrieves via the operating system API and fed back to the user

How does it work?

Browser reads the HTML code and builds the DOM

Browser uses the DOM to build an accessibility tree

Events such as user interaction trigger the browser to rebuild the DOM and the accessibility tree

Assistive technology access this information by querying the operating system API’s

How does it work?

So in general HTML is highly accessible

- It is focusable
- It is operable via keyboard
- It relays information about semantic HTML

With some exceptions

- Span tags
- Div tags

So what’s the problem?

We want to build web apps that do things that just aren’t there in HTML

- Spans and Divs allow us to nest objects and customise the dom
- Scripting frameworks such as javascript allows us to provide extra functionality e.g. controls, widgets and interaction
- HTML is flexible and open to manipulation

Let's look at some code:

```
<span onclick="goToLink(event, 'http://www.w3.org/')">
        W3C website
</span>
```

- Provides no information to the accessibility tree
- Not be focusable by a keyboard

Let's look at some code:

Add a tabIndex:

```
<span onclick="goToLink(event, 'http://www.w3.org/')" tabindex="0">
         W3C website
</span>
```

Let's look at some code:

Add some css:

```
 #accessLink {
  color: #009;
  background: transparent;
  text-decoration: underline;
}
        #accessLink:hover,
        #accessLink:focus {
  color: #000;
  cursor: pointer;
  outline: solid 1px white;
}
```

Let's look at some code:

Add some keyboard functionality with javascript:

```
   <span tabindex="0" onclick="goToLink(event, 'http://www.w3.org/')"
   onkeydown="goToLink(event, 'http://www.w3.org/')">

       function goToLink (event, url) {
       var type = event.type;
   if (
  (type === 'click') ||  (type === 'keydown' && event.keyCode === 13)
) {
  window.location.href = url;
   }
```

Let's look at some code:

And we can add an Aria role to provide information to the accessibility tree:

```
<span tabindex="0" role="link" onclick="goToLink(event,
'http://www.w3.org/')" onkeydown="goToLink(event, 'http://www.w3.org/')">
```

ARIA…. eh?

Accessible Rich Internet Applications:

Useful for when native HTML or other methods cannot be applied

Provides information about the state of an element to the accessibility tree

However can’t be used to change the way an element behaves

Should only be a last resort

So what can be done?

We can add extra information into the HTML

We can provide extra styling

We can use event listeners

Lot of extra work for something that already exists in native HTML

Don’t get the semantic benefits of native HTML

To conclude...

Use native html where possible

Always make things focusable with a keyboard

Find ways of feeding back the purpose of an element to a screenreader

As a last resort use ARIA

Test Test Test

Umbraco Accessibility

Umbraco Accessibility

Umbraco Accessibility Project

- Focused on making the Umbraco backend accessible
- Users with accessibility needs should not be prevented from engaging in the backoffice experience
- Community-led initiative

Umbraco Accessibility

Umbraco Accessibility Project

- First met on April 4th
- Trello board and Slack channel created
- ‘Accessibility’ category created on the Umbraco Github Issues channel
- Danny Lancaster and WeAreSigma carried out audit of backoffice accessibility
- Agreed to submit 5 pull requests before Codegarden

Umbraco Accessibility

Umbraco Audit

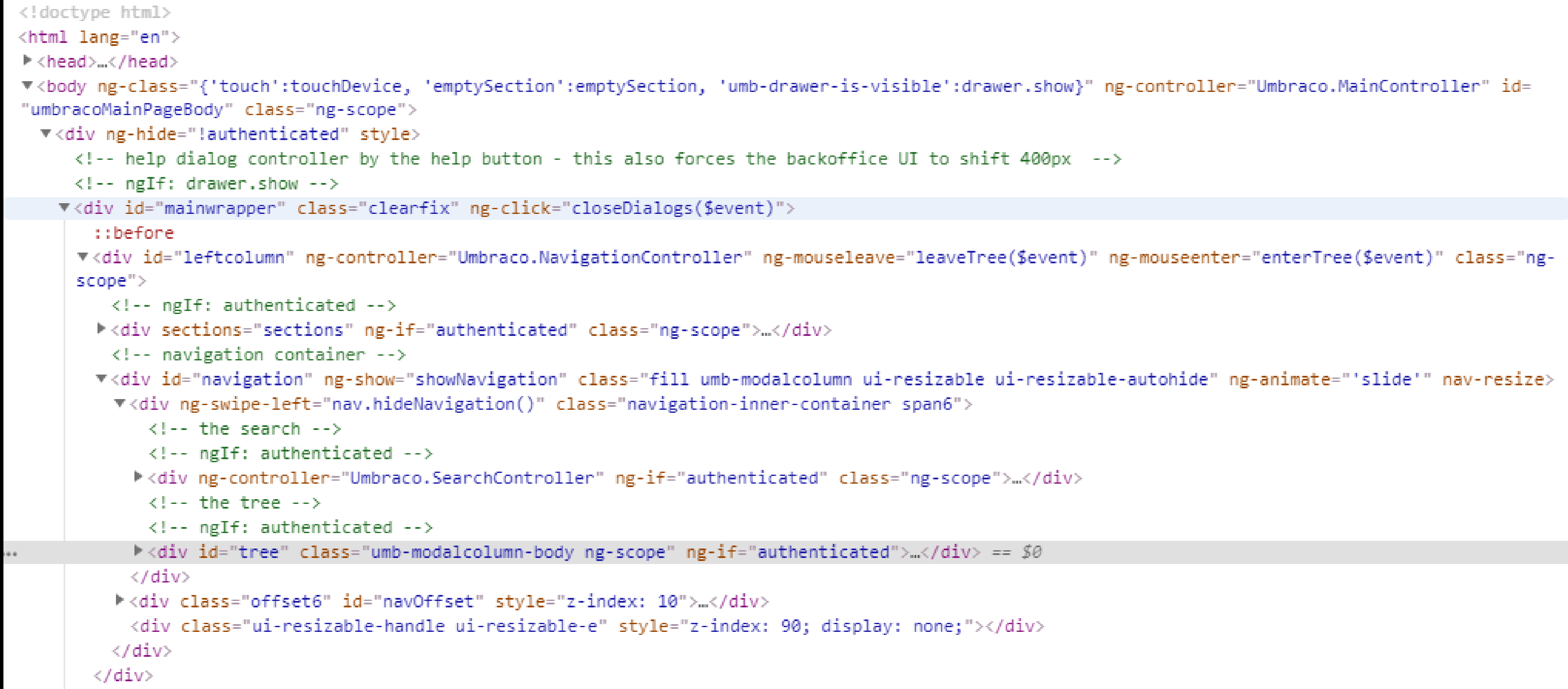
- 151 issues with accessibility identified
- Issues range from simple to complex
- Issues span the whole spectrum of POUR and all levels of A, AA, AAA
- Visit https://github.com/umbraco/Umbraco-CMS/issues/5277 to read the list

Umbraco Accessibility

So what’s the problem?

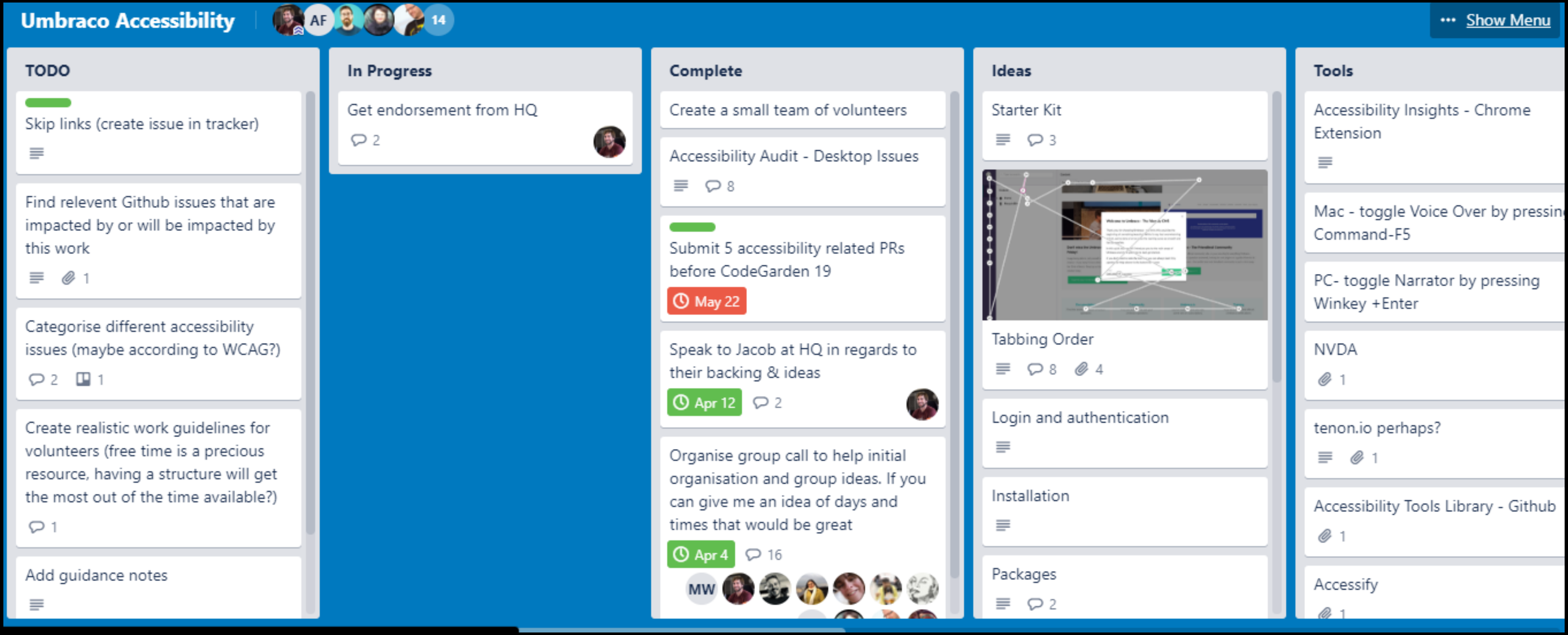
- The vast majority of the backoffice is not identifiable by assistive technology
- Backoffice is built on AngularJS and Bootstrap amongst others
- Elements can easily be broken
- No testing framework used that targets accessibility
- Accessibility as a development concept has only really come to the attention of developers in recent years

Umbraco Accessibility



```
<!doctype html>
<html lang="en">
▶<head>…</head>
▼<body ng-class="{'touch':touchDevice, 'emptySection':emptySection, 'umb-drawer-is-visible':drawer.show}" ng-controller="Umbraco.MainController" id=
"umbracoMainPageBody" class="ng-scope">
  ▼<div ng-hide="!authenticated" style>
     <!-- help dialog controller by the help button - this also forces the backoffice UI to shift 400px  -->
     <!-- ngIf: drawer.show -->
    ▼<div id="mainwrapper" class="clearfix" ng-click="closeDialogs($event)">
       ::before
      ▼<div id="leftcolumn" ng-controller="Umbraco.NavigationController" ng-mouseleave="leaveTree($event)" ng-mouseenter="enterTree($event)" class="ng-
      scope">
         <!-- ngIf: authenticated -->
        ▶<div sections="sections" ng-if="authenticated" class="ng-scope">…</div>
         <!-- navigation container -->
        ▼<div id="navigation" ng-show="showNavigation" class="fill umb-modalcolumn ui-resizable ui-resizable-autohide" ng-animate="'slide'" nav-resize>
          ▼<div ng-swipe-left="nav.hideNavigation()" class="navigation-inner-container span6">
             <!-- the search -->
             <!-- ngIf: authenticated -->
            ▶<div ng-controller="Umbraco.SearchController" ng-if="authenticated" class="ng-scope">…</div>
             <!-- the tree -->
             <!-- ngIf: authenticated -->
            ▶<div id="tree" class="umb-modalcolumn-body ng-scope" ng-if="authenticated">…</div> == $0
           </div>
          ▶<div class="offset6" id="navOffset" style="z-index: 10">…</div>
           <div class="ui-resizable-handle ui-resizable-e" style="z-index: 90; display: none;"></div>
         </div>
       </div>
```

Umbraco Accessibility



Umbraco Accessibility AF 14 ••• Show Menu

TODO

- Skip links (create issue in tracker)
- Find relevent Github issues that are impacted by or will be impacted by this work 1
- Categorise different accessibility issues (maybe according to WCAG?) 2 1
- Create realistic work guidelines for volunteers (free time is a precious resource, having a structure will get the most out of the time available?) 1
- Add guidance notes

In Progress

- Get endorsement from HQ 2

Complete

- Create a small team of volunteers
- Accessibility Audit - Desktop Issues 8
- Submit 5 accessibility related PRs before CodeGarden 19 May 22
- Speak to Jacob at HQ in regards to their backing & ideas Apr 12 2
- Organise group call to help initial organisation and group ideas. If you can give me an idea of days and times that would be great Apr 4 16 MW

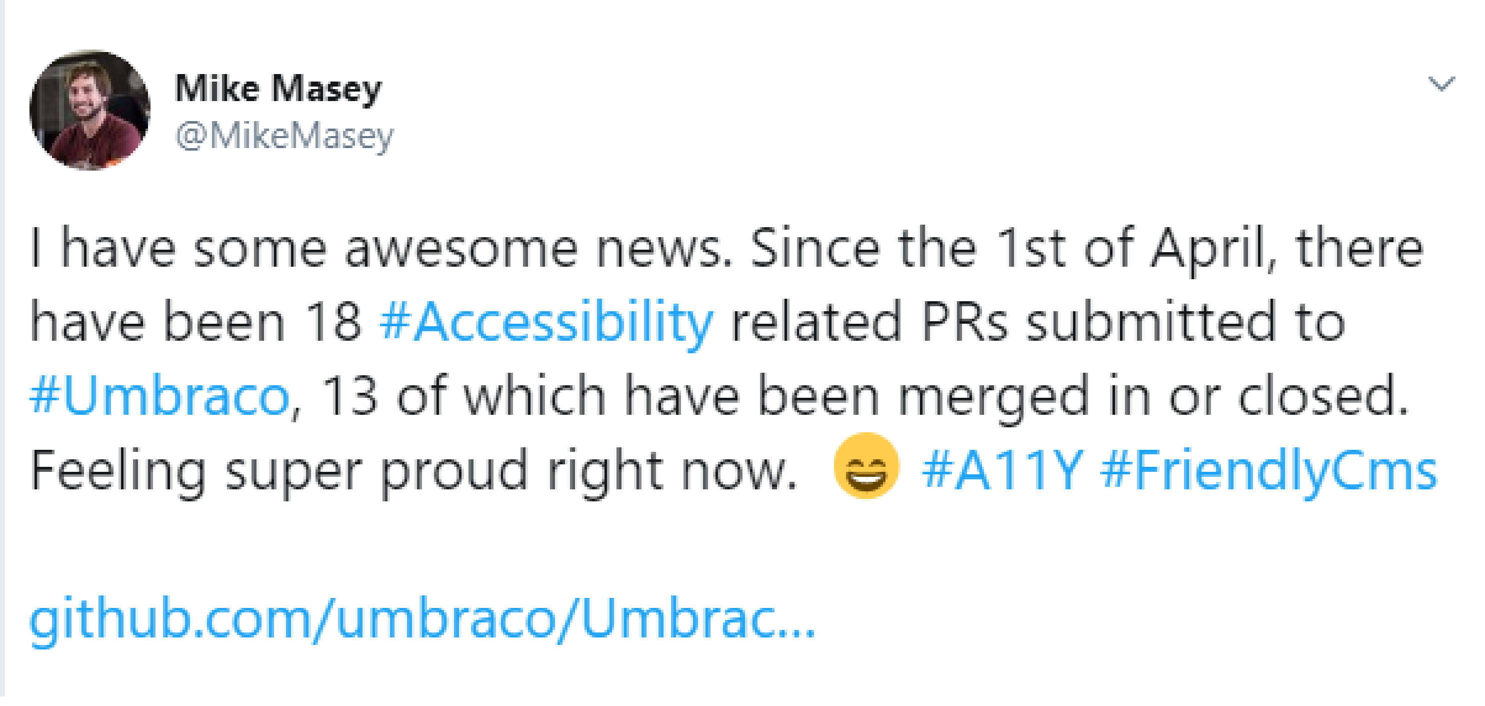
Ideas

- Starter Kit 3
- Tabbing Order 8 4
- Login and authentication
- Installation
- Packages 2

Tools

- Accessibility Insights - Chrome Extension
- Mac - toggle Voice Over by pressin Command-F5
- PC- toggle Narrator by pressing Winkey +Enter
- NVDA 1
- tenon.io perhaps? 1
- Accessibility Tools Library - Github 1
- Accessify 1

Umbraco Accessibility



Mike Masey
@MikeMasey

I have some awesome news. Since the 1st of April, there have been 18 #Accessibility related PRs submitted to #Umbraco, 13 of which have been merged in or closed. Feeling super proud right now. 😄 #A11Y #FriendlyCms

github.com/umbraco/Umbrac...

Thank you

Twitter: @tiffanyMprosser
Email: tiffany.prosser1@gmail.com
Umbraco accessibility team:
https://trello.com/b/MwD8xuz3/umbraco-accessibility