EduTyping

Accessibility Assessment – Analysis of Findings

B.E. Publishing

Provided by Knowbility, Inc.
www.knowbility.org

Date: June 15, 2019

Executive Summary

Knowbility reviewed the EduTyping Achievements page submitted by B.E. Publishing to determine if these materials meet the needs of their students with disabilities. To make that determination, Knowbility tested against the Web Content Accessibility Guidelines (WCAG), Version 2.1, Levels A and AA. WCAG Is the global standard, developed by member of the World Wide Web Consortium (W3C), the standards makers for the web. Knowbility tested using a wide variety of browsers, devices, and assistive technologies - including screen readers. We determined that these pages contain significant barriers that will prevent equal access to information and function by people with disabilities. Our findings are arranged by issue and each issue references the applicable WCAG guideline(s) which are included as links in Appendix 1 of this document. In conclusion, we provide a more thorough discussion of the issues and detailed guidance for how to remove the accessibility barriers. We look forward to discussing our findings with you and appreciate the opportunity to do this important work with B.E. Publishing. Major issues are summarized here and detail is provided in the report.

Interactive pages have many issues that prevent keyboard and screen reader users from interacting with the page. Very few controls receive keyboard focus. Once the user reaches the end of the navigation menu within the header there are no additional keyboard focusable elements. This prevents keyboard and screen reader users from changing the content, thus preventing any meaningful interaction at all. Several of the menu items, Choose a Theme, User menu, and All Time, have additional popup menus that are not available via the keyboard for users of screen readers o other assistive technologies.

Many of the controls are implemented using JavaScript. These controls do not work with the keyboard nor is their functionality identified to the screen reader users. Examples are the popup menus previously mentioned and the tabs and tab panel used to control what content is displayed. These controls need the addition of WAI-ARIA roles, states, and properties to identify them to assistive technologies. The keyboard support must also be implemented so these controls work with the keyboard.

There are also images on all of these pages that do not have alternative text. The alternative text is used by assistive technologies to describe the image to the user. In addition, significant text throughout the pages have contrast that is too low against the background color. This makes is difficult for people with low vision to see and read the text. It was difficult using this page with a screen reader and some tasks cannot be performed at all. After the current issues have been fixed, the application must be retested.

Table of Contents

Scope

1	Student Lessons	https://www.edutyping.com/student/lessons
2	Space Bar/Enter Keys	https://www.edutyping.com/student/lesson/110169/home-row-space-bar-and-enter-keys
3	5 and 6 keys	https://www.edutyping.com/student/lesson/110139/5-and-6-keys
4	4-5-6 and Enter Keys	https://www.edutyping.com/student/lesson/110247/4-5-6-and-enter-keys
5	Career Prep	https://www.edutyping.com/student/lesson/118855/career-prep
6	Ready for Workplace	https://www.edutyping.com/student/lesson/125244/are-you-ready-for-the-workplace
7	When Robots Rule	https://www.edutyping.com/student/lesson/399/when-robots-rule-the-world
8	Pre-Test	https://www.edutyping.com/student/lesson/110437/pre-test
9	NotePad	https://www.edutyping.com/student/notepad
10	Progress	https://www.edutyping.com/student/progress
11	Achievements	https://www.edutyping.com/student/achievements
12	Account	https://www.edutyping.com/student/account

Approach and testing tools used

We completed a technical and functional evaluation of representative pages and/or components of the EduTyping Achievements website, as selected by B.E. Publishing staff for compliance against WCAG 2.1 and documented our findings. By combining functional and technical testing, we ensured a comprehensive review when testing for accessibility.

Technical evaluation

By "technical evaluation" we mean the conformance assessment to the Standards for all applicable success criteria in the context of the audited web page(s). No automated tests were run.

Functional evaluation

By "functional evaluation" we mean the use of computer adaptive technology such as voice synthesizers, screen readers, and magnification software, to verify that the results interpreted by these tools corresponds to the result observed during a visual and technical validation of the web page.

Tools

- VoiceOver, Jaws, NVDA
- WebAim color contrast checker
- Firefox web developer toolbar
- Windows High Contrast mode
- Browsers
- Firefox
- Safari
- Chrome
- Edge

Disclaimer

This audit was conducted between June 10-12, 2019 on pages and components included in the scope. Some success criteria may be open to interpretation due to the supporting information provided in the official guidelines. The team erred on the more restrictive interpretation where more than one interpretation is possible.

General issues description

1. Alt text for Image Content

Pages with errors of this type	https://www.edutyping.com/student/lessons https://www.edutyping.com/student/notepad https://www.edutyping.com/student/progress https://www.edutyping.com/student/achievements https://www.edutyping.com/student/account
Affected community	Vision
WCAG Success Criterion	1.1.1 Text Alternatives for NonText Content
WCAG Level	A
Priority	High

Description

Images and other non-text content lack descriptive text on several pages. When content includes graphic images that uniquely convey meaning or function - including images of text and graphic buttons and form controls - proper coding requires the provision of text alternatives using the `alt` attribute. All graphic HTML content must have brief, clear, and unique text description of the meaning or function of the image to allow blind users or those that have images disabled for any reason to understand that meaning or function. When onscreen text is redundant to the meaning of the image, empty alt text <`alt=""`> may be used to allow a screen reader to remain silent and not needlessly repeat the same information. When the image is a link, the alt attribute must describe the function or target and the physical appearance becomes redundant.

Solution

Ensure an appropriate `alt` attribute is provided for all `img` elements

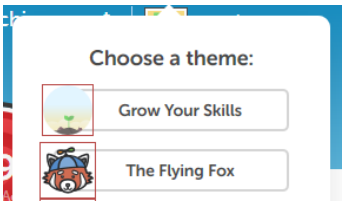
Use background CSS images only for decorative images that add no meaning to the page content

If an image is linked, ensure that the link target is clearly stated

Avoid redundant image and text announcement of links.

Examples

Table 1 Image examples. The following table presents examples of errors in the provision of alt text. There are 3 columns, the page where the error was observed, a screen shot of the image, and the recommendation for how to address the error.

Page	Image	Recommendation
All pages: The chosen theme as well as the section title “EduTyping” are presented as background images, thus unable to receive alt attributes		Validate that it is the intention of the authors to hide this information from screen reader users. If so, using background images is OK- no change needed If it is meaningful information, standard presentation with alt attribute is needed (see code below)
Dropdown theme menu uses images with no text alternative		Add `alt=””` as alt attribute to `img` element Alternatively, wrap the image within anchor `<a…>`element, thus allowing the text to be explicitly associated with the image.
Student Progress	Icons for Stars Earned, Average Speed, Average Accuracy, and Time Spent have no alt attributes.	Add `alt=””` as alt attribute to `img` element -OR- present icons as CSS background images

Code level repair suggestion

Current code

```
<div class="g-b g-b--4of12_xl g-b--6of12_s well well--b">
<div class="achievement achievement--lesson mhc">
<img src="/dist/student/images/avatar/app/avatar-01.svg"></div>
<div class="well tac">
<h4 class="ttc">Typing Seedling</h4>
<p class="mbf twb tss tc-ts">Rank 1 / 5</p>
</div>
</div>
```

Solution

The first part of the solution is to decide if the image is decorative or not. If not, add an empty alt attribute onto the the img element. Either alt with no value or alt="" will cause the screen reader to ignore the image.
If the image conveys meaning or is important, add an alt attribute with descriptive text.
In the recommended code above I added empty alt text. If you feel the image is important, add a description. For example, alt="new seedling growing up from the soil" or something similar.

Recommended code

```
<div class="g-b g-b--4of12_xl g-b--6of12_s well well--b">
<div class="achievement achievement--lesson mhc">
<img src="/dist/student/images/avatar/app/avatar-01.svg" alt=""></div>
<div class="well tac">
<h4 class="ttc">Typing Seedling</h4>
<p class="mbf twb tss tc-ts">Rank 1 / 5</p>
</div>
</div>
```

Additional Resources

- An alt decision tree from the W3C
- Accessible Images article from WebAIM

2. Contrast of text and navigation elements against background

Pages with errors of this type	All pages that we reviewed have extensive contrast issues. Examples from several pages are included below
Affected community	Low vision, color blind, partially sighted, cognitive
WCAG Success Criterion	1.4.3 Minimum Contrast
WCAG Level	AA
Priority	Medium

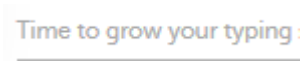
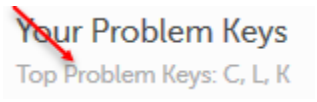
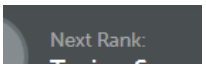
Description

Contrast is an important element of digital accessibility. When text is displayed on the screen and does not mean contrast requirements, it is difficult and sometimes impossible to read for people with low vision or color blindness (nearly 1 in 8 men have color blindness of some kind.) Lack of sufficient contrast can lead to fatigue, increase errors, and decrease reading comprehension.

The color contrast of many items is too low to meet the WCAG requirements. Success Criterion 1.4.3 requires a text contrast of at least 4.5:1 for text that is 24px or less (18.66px if bold) or 3:1 for text that is larger than 24px (18.66px if bold). Many of the colors throughout the application fail to meet these ratios.

Examples

Table 2 Contrast. The examples below are random samples. There are more contrast violations than can be shown in this report. One page alone registered nearly 400 contrast issues. Use these to understand the issue and then use the recommended tools to identify and remediate errors on all pages.

Page	Sample
Navigation on all pages	 3.7 to 1
Student Lessons	 2.8 to 1
Student Progress	 2.4 to 1
Several pages	 2.4 to 1
Navigation	 2.1 to 1
Pop up on mouse hover/progress bar	 3.4 to 1
5 and 6 keys	 2.7 to 1
Student lessons subpage	 2.5 to 1 1.4 to 1

Solution

The contrast of the white text of the navigation menu against the blue background of the header (#1D8CBF) is only 3.8:1. You can solve this by sizing the text up to 24px or modifying the blue background color.

The grey text, Time to grow your typing skills, (#a9a9a9) against the white background is only 2.35:1. Darken to #767676 to make it 4.5:1. This grey text against white is also used in other locations on the page.

The small text (avg Speed, Avg Acc, typing time) color against the dashboard figures is too low.

The white text of the active tab title (Summary, Avatar Levels, Typing Speed, etc) against the blue background (#3295db) is only 3.25:1. Changing the background to #007CBD will up it to 4:5:1. Or, increase the size to 24px or larger or make it 18.66px and bold (the text size is currently 18px). This color combination is also used for the progress bars and must be fixed there as well.

- Use a contrast analyzer tool to measure contrast of text against the background.
- Ensure normal text has a contrast ratio of 4.5 to 1; larger and/or bold text meets the 3 to 1 minimum.
- Avoid text-based images and use actual text wherever possible.

3. Lack of Keyboard Access

Pages	All of the pages reviewed have interactions and operations that cannot be triggered from the keyboard.
Affected community	Vision, mobility
WCAG Success Criterion	2.1.1 Keyboard (A) 4.1.2 Name, Role, Value (A)
WCAG Level	A
Priority	High

Description

When the mouse hovers over navigation elements, video controls, and many page-specific functions, options may popup or be presented to the user – good! However, if the student does not use a mouse due to using assistive technology, there is no option provided for equivalent interaction. Here are some examples, but this is not a complete list. Use these coding suggestions to make repairs to all elements that are currently unavailable to the keyboard.

Example 1: User menu and All Time buttons

The user menu button in the upper right corner and the All time buttons do not work with the keyboard. A keyboard-only user or a screen reader user would not realize that these are buttons. All interactive items within a page must be usable with the keyboard and receive visible focus. The function of the controls, in this case, a button, must be identified to assistive technologies. These buttons open up a menu to provide additional navigation choices. Since the buttons do not receive keyboard focus nor been identified as actionable, the screen reader user is not even aware of their presence as buttons. (WCAG 2.1 4.1.2 Name, Role, Value)

Current code

```
<div class="dropdown dropdown--r dropdown--a">
<div class="tsxs split split--flag split--xxs">
<div class="split-cell dropdown-trigger">
test-1
</div>
<svg class="icon arrow arrow--xs split-cell">
<use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="/dist/student/images/icons.svg#arrow"></use>
</svg>
</div>
<ul class="list dropdown-items list--f">

<li class="list-item dropdown-item"><a href="/student/account"
class="dropdown-link tss">
Account Settings
</a></li>

<li class="list-item dropdown-item"><a href="#" class="js-logout dropdown-link
tss">
Log Out</a></li>
</ul>
</div>
```

Solution

Use semantic HTML to create the user interface controls on the page. Because these are created with the div element the HTML semantics (that these are buttons) and the behavior has not been communicated to the browser nor the screen reader. You can use WAI-ARIA to add these semantics but it is preferable to use existing elements and enhance them with ARIA as needed.

These buttons need to be identified as menu buttons. Refer to the WAI-ARIA Authoring Practices 1.1 Menu Button section to review the details. The pseudo-code below shows adding the aria roles and properties into the existing code. It will certainly need additional JavaScript support and adjustments to fit your development environment.

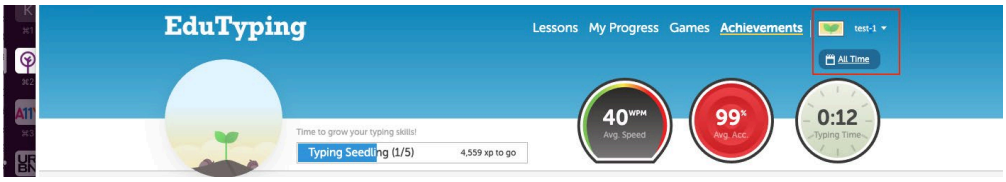
- Add aria-haspopup="true" onto the button/trigger element to indicate that it has a popup associated with it. *Add aria-controls="idOfMenuElementContainter" to indicate the popup list of items that the trigger controls.
- Add role="menu" onto the ul element containing the menu items. Add aria-labelledby="idOfButton" to reference the button text as the label for this menu.
- Add role="none" onto the list items to indicate that they are not functioning as list items. Add role="menuitem" onto the anchor elements within each li element to indicate that it is a menu item.
- Modify your JavaScript code as necessary. Note that this is one example of how to modify this code, there are other ways of implementing this as well.

Recommended code

```
<div class="dropdown dropdown--r dropdown--a">
<div class="tsxs split split--flag split--xxs">
<button id="userMenuButton" onclick="eventHandler" aria-
controls="userDropdown" aria-haspopup="true" class="split-cell dropdown-
trigger">
test-1
<svg class="icon arrow arrow--xs split-cell">
<use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="/dist/student/images/icons.svg#arrow"></use>
</svg>
</button>
</div>
<ul id="userDropdown" role="menu" aria-labelledby="userMenuButton" class="list
dropdown-items list--f">

<li role="none" class="list-item dropdown-item"><a role="menuitem"
href="/student/account" class="dropdown-link tss">
Account Settings
</a></li>

<li role="none" class="list-item dropdown-item"><a role="menuitem" href="#"
class="js-logout dropdown-link tss">
Log Out</a></li>
</ul>
</div>
```

Screenshot(s)

Figure 1: Screenshot of Header highlighting buttons with no keyboard support



Example 2: Typing speed progress bar

On the Student Achievement page, the typing speed progress bar displays a pop-up to show the next rank/level. There is no mechanism for a keyboard user to invoke this popup. All interface components must work over different modalities, touch (if appropriate), keyboard, mouse, etc. so that all users can benefit from the information and interaction.

Solution

Since this is generally not a keyboard focusable element, add a button or other trigger that is keyboard focusable so the keyboard user can invoke the Next Level popup or find a different mechanism to provide that information.

Example 3: Page Contents selection tabs are not keyboard accessible

The controls to select the page contents do not work with the keyboard. All interactive items and user interface controls on the page must work with the keyboard and have a visible focus indication. There

are no keyboard focusable items after the main navigation. A user trying to tab through this page will get to Achievements in the banner area and then the next tab stop takes them out of the page. Keyboard users and screen reader users rely on all items being properly identified and usable via the keyboard.

Current code

```
<div class="dashboard">
<div class="dashboard-path paths ttc">
<select class="paths-select select js-tab">
<option value="summary">Summary</option>
<option value="avatars">Avatar Levels</option>

<option value="1">Typing Speed</option>

<option value="5">Accuracy Stars</option>

<option value="6">Typing Tests</option>

<option value="8">Time Spent Typing</option>

<option value="9">Characters Typed</option>

<option value="7">Games Played</option>

<option value="badges">Lesson Badges</option>
</select>
<div class="path path--s tab js-tab is-active" data-id="summary">
<div class="path-title">Summary</div>
</div>
</div>
<div class="js-panel-item dashboard-list">
<div class="dashboard-listInner">
<div class="dashboard-listContent tab-panel lessons is-active">
<div class="lesson pt">
```

Solution

The class names indicate that you intend this to be a tab panel. Each element that is a tab must have the role=tab. The tab must have aria-controls=<id of panel this tab controls>. The currently selected tab is identified using aria-selected=true all others have aria-selected=false. The item where the panel contents are loaded must have role="tabpanel". It must also have an aria-labelledby=<id of controlling tab>. The element that contains all of the tabs must have the role="tablist" and must be labeled either via aria-label=<description> or an aria-labelledby=< id of the element with identifying text label>.
I have added those roles to the existing code snippets to demonstrate. See the ARIA Authoring Practices Tab section for complete details.

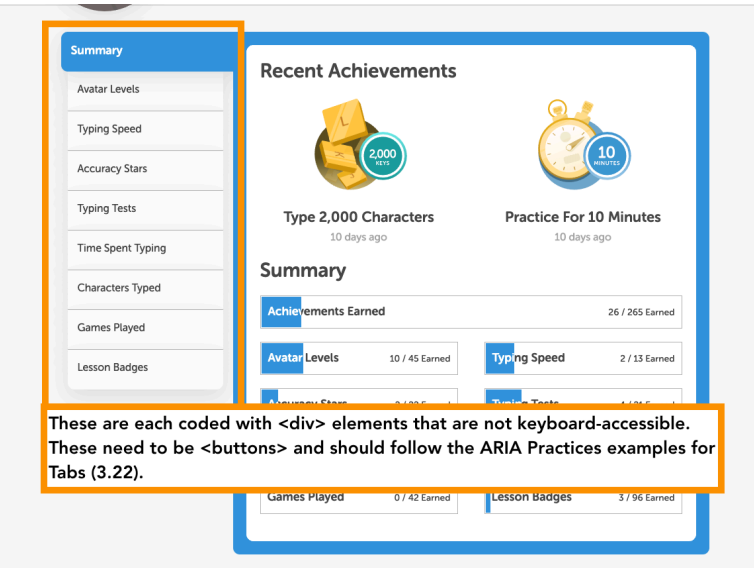
I am also not sure how you are using the <select> elements without seeing the actual code. This select is not visibly rendering on the page.

Recommended code

```
<div class="dashboard">
<div class="dashboard-path paths ttc" role="tablist" aria-label="Content
Selection" aria-orientation="vertical">
<select class="paths-select select js-tab">
<option value="summary">Summary</option>
<option value="avatars">Avatar Levels</option>

<option value="1">Typing Speed</option>

<option value="5">Accuracy Stars</option>

<option value="6">Typing Tests</option>

<option value="8">Time Spent Typing</option>

<option value="9">Characters Typed</option>

<option value="7">Games Played</option>

<option value="badges">Lesson Badges</option>
</select>
<div role="tab" aria-selected="true" aria-controls="panelContents" class="path
path--s tab js-tab is-active" data-id="summary">
<div id="tabSummary" class="path-title">Summary</div>
</div>
</div>
<div class="js-panel-item dashboard-list">
<div class="dashboard-listInner">
<div id="panelContents" role="tabpanel" aria-labelledby="tabSummary"
class="dashboard-listContent tab-panel lessons is-active">
<div class="lesson pt">
```

Screenshot:

Figure 2 : Screenshot of Tab list with suggested code



Example 4: Choose a theme button does not work with the keyboard

The Choose a theme button like many other interactions cannot be performed with the keyboard. It does not receive keyboard focus and opens only on mouseover. This is not identified as a button and does not contain a text description so the screen reader user doesn't even encounter this. The keyboard only user also has no way to interact with or open this menu and may not even know what it represents. Since the button face is only an image, it also needs to have alternative text.

Current code

```
<li class="list-item has-tooltip " data-tooltip="" id="id0.8760279316317998"
aria-describedby="tooltip_2rvossrrz4">
```

Solution

The solution is similar to the User menu and All Time menus as reported. Implement this as a button menu. See the WAI Tutorial on Fly-out Menus for code examples.

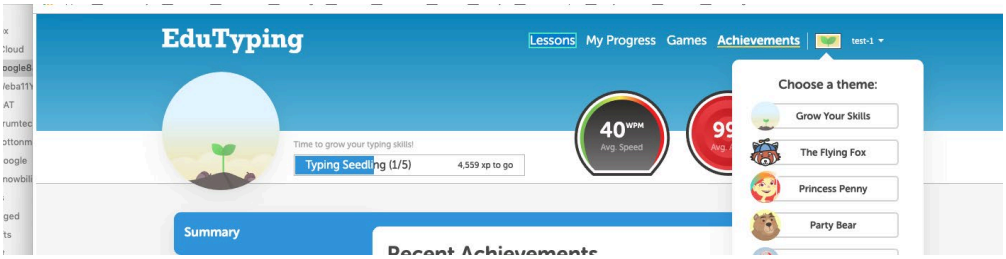
The current code that attempts to add a label to the button is implemented incorrectly. The current code has an aria-describedby="tooltip_2rvossrrz4". However, the item with id="tooltip_2rvossrrz4" is further down the page and has an aria-hidden="true" attribute. Anything with an aria-hidden value of true causes the screen reader to ignore it. The recommended code modifies the aria-describedby to an aria-label. This change will help the screen reader user. You could use aria-labelledby as long as the element that it points to is visible to the screen reader. A better solution is to provide the actual text on the button to help all users identify the button and its function. This is especially important for cognitive impaired users.

Recommended code

```
<li class="list-item has-tooltip " data-tooltip="" id="id0.8760279316317998"
aria-label="Choose a Theme">
```

Screenshot

Figure 3: Screenshot of Choose a Theme dropdown menu

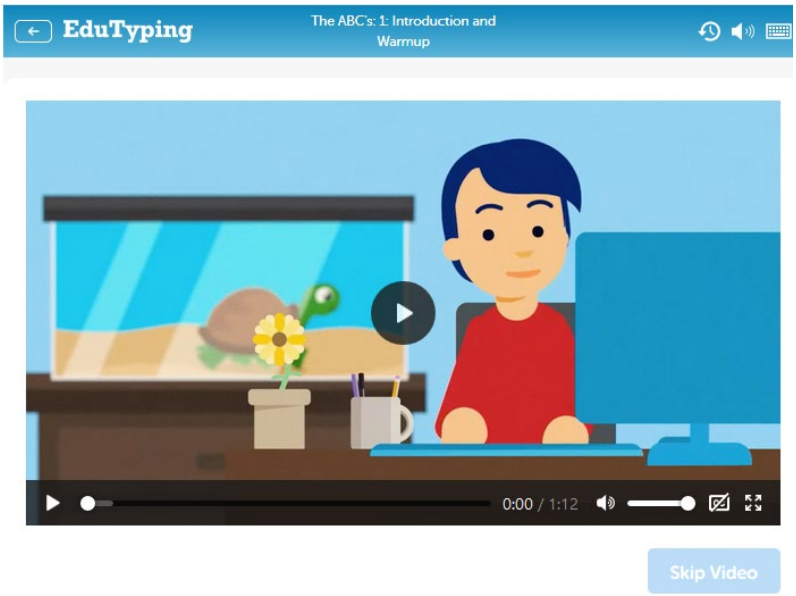


Example 5: Video players are not operable by the keyboard

Several lessons have video intros. However, the tab key has been disabled and so the screen reader user is unable to start or operate any of the functions. As well, there is no access to the Skip button. The user is entirely blocked from accessing the content of the lesson.

Screenshot

Figure 4: The ability to start or skip or control any of the video function is unavailable to keyboard users



Solution

As demonstrated above, enable keyboard access to all content and function.

Resources

List of Accessible Media Players from Digital A11y

4. Page structure

Pages	All pages reviewed lack defined, consistent structure
Affected community	Vision, Motor
WCAG Success Criterion	1.3.1 Info and Relationships 2.4.1 Bypass Blocks 2.4.6 Headings and Labels
WCAG Level	A, AA
Priority	High

Description

Semantic structure allows people who do not see the screen to understand how it is organized and how and where to find the information they need. Screen readers provide a way for users to navigate

quickly through the page by pulling up a list of headings, regions, and/or landmarks. None of the pages we viewed have a clearly described and defined series of headings, skip navigation, landmarks or other structural elements required for navigating with assistive technology.

The heading structure is erratic and inconsistent, there are no skip navigation links and landmarks are missing from all pages. By using proper heading and landmark structure along with skip links, you dramatically improve the ability of users of assistive technology to successfully navigate through the sections and interactions of even very complex pages and applications.

Issue1: Skip Links

There is no mechanism for the keyboard or screen reader use to skip over the repeated navigation elements on the page to quickly arrive at the main content. Once a user is familiar with the layout of the page, she often wants to skip over the navigation to quickly reach the main content. Adding landmarks (see linked issue) helps assistive technology users. Adding a skip link that becomes visible on focus aids keyboard users.

Solution

The solution is to add a skip to main content link at the top of all pages. Use CSS to make this link visible when it receives focus. Thus, when a keyboard user tabs into the page, this is one of the first items they will encounter and they can easily activate this link to skip to the main content of the page, bypassing the header and navigation. See good examples of skip links at the Knowbility blog post, Skip Links Design Showcase.

Recommended code

```
<a class="skip-link" href="#main-content">Skip to Main Content</a>
........
<main id="main-content">
```

Issue2: Headings

There are no visible H1 elements on many of the pages. Headings are not nested properly on any of the ages. In some cases, such as the NotePad page, there is only one heading on the entire page, omitting much of the content and function. Screen reader users rely on headings to have an overview of the page's content and to navigate a page. Missing headings and improperly nested headings are confusing. Someone relying on headings to understand the relationship of page content may wonder what they are missing when heading levels are not logically hierarchical.

Generally, there should be only one H1 element and it should represent the main heading or topic of the page. It should be visible. For example, on the Achievements page, one of the H1 elements appears to be in a pop out or flyout section that we were unable to discover, but suspect may relate to reaching a new achievement level.

On that same page, an H4 is used as the heading, Choose Theme, in the popup menu. This does not flow in the hierarchy since there is no H3 parent for this heading.

On the Summary panel the H2 Summary is followed by H4 headings (Achievements Earned, Avatar levels, etc.) these should be H3's.

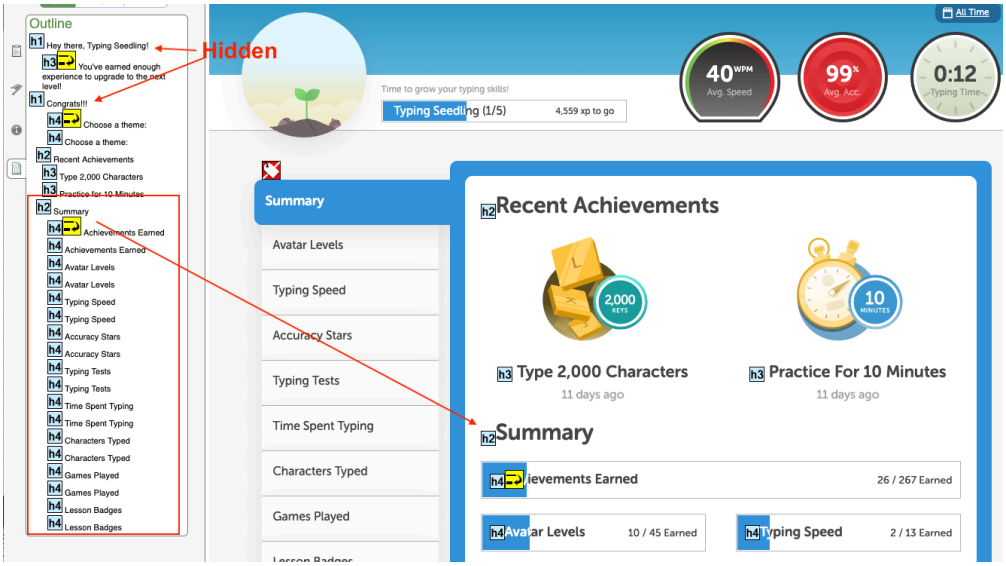
Solution

Review the heading structure of these pages and add a visible H1 heading on the page that describes the main content, perhaps something that highlights that that page is about Achievements. Don't use an H3 for the Choose a Theme menu heading, use styled text. Headings are not generally used on items that are not always visible on the page. Make certain the headings are arranged hierarchically.

On the Summary panel the H2 Summary is followed by H4 headings (Achievements Earned, Avatar levels, etc.) these should be H3's. Also, note that these are repeated within the code. The screen reader reads each instance. This is due to how the "progress bars" are implemented. While not a failure, it is very annoying for the screen reader user. You may wish to explore a less verbose implementation of those progress indicators.

Screenshot

Figure 5: Screenshot of Achievements page heading structure and problems



Issue 3: Landmarks

Landmark roles (or "landmarks") programmatically identify sections of a page. Landmarks help assistive technology users orient themselves to a page and help them navigate easily to various sections of a page. Without these landmarks, the user is forced to scroll through and listen to all of the content until the desired region is found.

Current code

```
<div class="has-header hero hero--01 row">
<div class="hero-cell cell cell--xl">
<div class="hero-asset"></div>
<div class="hero-split well--p split split--flag">
<div class="hero-splitCell split-cell">
```

```
<a href="/student/lessons" class="logo logo--hero logo--edutyping mrs db"></a>

</div>
<div class="hero-splitCell split-cell">
<ul class="list list--flag list--inline list--xs list--wrap hero-nav">
<li class="list-item">
<ul class="list list--inline list--inlineNav list--s stall stall--right stall-
-hxlt stall--xs stall_l">
<li class="list-item">
<a href="/student/lessons" class="navLink link--nu ">
Lessons</a>
</li>


<li class="list-item">
<a href="/student/progress" class="navLink link--nu wsnw ">
My Progress</a>
</li>

<li class="list-item">
<a href="/student/games" class="navLink link--nu ">
Games</a>
</li>

<li class="list-item">
<a href="/student/achievements" class="navLink link--nu is-active">
Achievements</a>
</li>

</ul>
</li><li class="list-item">
<ul class="list list--inline list--flag list--xs hero-navBlockList">
<li class="list-item has-tooltip " data-tooltip="" id="id0.8760279316317998"
aria-describedby="tooltip_2rvossrrz4">
<div href="/student" class="js-skin-flag flag flag--app is-active cup"></div>
<div class="tooltip-template">
<div class="tooltip tooltip--i">
<div class="tooltip-arrow"></div>
<div class="tooltip-inner"></div>
</div>
</div>
<div class="tooltip-html">
<h4 class="tac well well--xs">
Choose a theme:
</h4>
<ul class="list list--f">

<li class="js-skin list-item" data-id="1">
<a class="btn btn--stroke-s btn--i btn--s theme theme--app">

<div class="theme-image">
<img src="/dist/student/images/avatar/app/avatar-01.svg">
</div>

Grow Your Skills
</a>
</li>
```

```
<li class="js-skin list-item" data-id="2">
<a class="btn btn--stroke-s btn--i btn--s theme theme--fox">

<div class="theme-image">
<img src="/dist/student/images/avatar/fox/avatar-01.svg">
</div>

The Flying Fox
</a>
</li>

<li class="js-skin list-item" data-id="3">
<a class="btn btn--stroke-s btn--i btn--s theme theme--princess">

<div class="theme-image">
<img src="/dist/student/images/avatar/princess/avatar-01.svg">
</div>

Princess Penny
</a>
</li>

<li class="js-skin list-item" data-id="4">
<a class="btn btn--stroke-s btn--i btn--s theme theme--forest">

<div class="theme-image">
<img src="/dist/student/images/avatar/forest/avatar-01.svg">
</div>

Party Bear
</a>
</li>

<li class="js-skin list-item" data-id="5">
<a class="btn btn--stroke-s btn--i btn--s theme theme--sea">

<div class="theme-image">
<img src="/dist/student/images/avatar/sea/avatar-01.svg">
</div>

Cappy the Sailor
</a>
</li>

<li class="js-skin list-item" data-id="6">
<a class="btn btn--stroke-s btn--i btn--s theme theme--pixels">

<div class="theme-image">
<img src="/dist/student/images/avatar/pixels/avatar-01.svg">
</div>

The Pixel Knight
</a>
</li>

<li class="js-skin list-item" data-id="7">
```

```
<a class="btn btn--stroke-s btn--i btn--s theme theme--space">

<div class="theme-image">
<img src="/dist/student/images/avatar/space/avatar-01.svg">
</div>

Eek From Space
</a>
</li>

<li class="js-skin list-item" data-id="8">
<a class="btn btn--stroke-s btn--i btn--s theme theme--superhero">

<div class="theme-image">
<img src="/dist/student/images/avatar/superhero/avatar-01.svg">
</div>

Super Eli
</a>
</li>

<li class="js-skin list-item" data-id="9">
<a class="btn btn--stroke-s btn--i btn--s theme theme--nitrotype">

<div class="theme-image">
<img src="/dist/student/images/avatar/nitrotype/avatar-01.svg">
</div>

Nitro Type
</a>
</li>

<li class="js-skin list-item" data-id="10">
<a class="btn btn--stroke-s btn--i btn--s theme theme--basic">

Basic Skin
</a>
</li>

</ul>
</div>
</li>

<li class="list-item">
<div class="dropdown dropdown--r dropdown--a">
<div class="tsxs split split--flag split--xxs">
<div class="split-cell dropdown-trigger">
test-1
</div>
<svg class="icon arrow arrow--xs split-cell">
<use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="/dist/student/images/icons.svg#arrow"></use>
</svg>
</div>
<ul class="list dropdown-items list--f">
```

```
<li class="list-item dropdown-item"><a href="/student/account"
class="dropdown-link tss">
Account Settings
</a></li>

<li class="list-item dropdown-item"><a href="#" class="js-logout dropdown-link
tss">
Log Out</a></li>
</ul>
</div>
</li>

</ul>
</li>
</ul>
</div>
</div>
</div>
</div>
```

Solution

Use semantic HTML elements and aria-labels and or roles to identify the sections of the page. Add at least the following HTML elements to the page; header, main, nav, footer.

The recommended code adds header and nav - you may want to modify the nesting. Also, modify the div containing the main contents into a main element.

See [ARIA Landmark roles] for more info.

Recommended code

```
<header class="has-header hero hero--01 row">
<div class="hero-cell cell cell--xl">
<div class="hero-asset"></div>
<div class="hero-split well--p split split--flag">
<div class="hero-splitCell split-cell">
<a href="/student/lessons" class="logo logo--hero logo--edutyping mrs db"></a>

</div>
<nav class="hero-splitCell split-cell">
......
</nav>
</header>
```

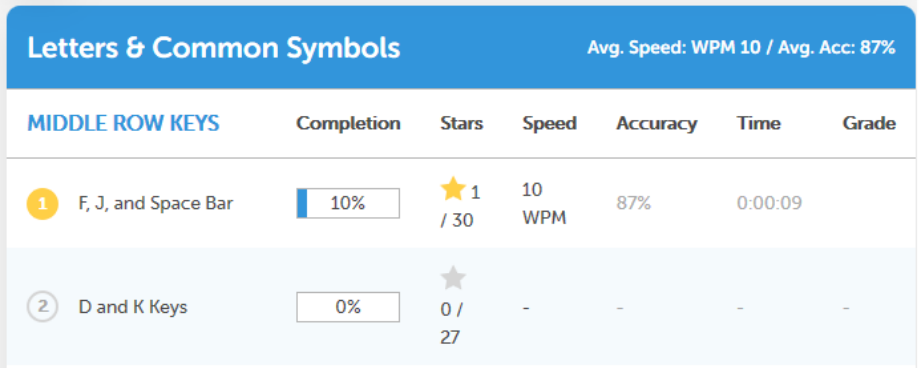
Issue 4: Table Structure

When information is included in a table format to show relationships between data items, the table must be properly structured and coded in order for blind users to understand the relationships. Accessible tables need HTML markup that indicates header cells and data cells and defines their relationship. Assistive technologies use this information to provide context to users.

Header cells must be marked up with <th>, and data cells with <td> to make tables accessible. For more complex tables, explicit associations may be needed using scope, id, and headers attributes.

For example, on the Progress page, https://www.edutyping.com/student/progress, the titles of each unit are presented at H3 and span columns within each table. Column headers within the tables are not identified as such. The blind user has no way to know what column is active and so does not understand the relationship between the different column data.

Figure 6 Sighted user can see column headings and understand relationship of data to heading category. Blind user has no support for that same understanding.



Solution

Use standard table markup – column and row headings – to make the relationship between the data and the identifying column heading clear and explicit.

Resources

WAI Tables Tutorial from the W3C Web Accessibility Initiative
Creating Accessible Tables from WebAIM

5. Top level menu items spoken twice by screen reader

Pages	https://www.edutyping.com/student/lessons https://www.edutyping.com/student/notepad https://www.edutyping.com/student/progress https://www.edutyping.com/student/achievements https://www.edutyping.com/student/account
Affected community	Vision, Cognitive
WCAG Success Criterion	2.4.3 Focus Order (A), 4.1.2 Name, Role, Value (A)
WCAG Level	A
Priority	Medium

Description

When navigating with a screen reader, the top level menu items are announced twice. This is due to the two menus on the page. One is in the header/banner area and the other is the navigation menu that gets "fixed" onto the top of the screen when the banner is scrolled out of view.

This is very confusing to the screen reader user as they don't expect to encounter the menu items twice and may be confused as to which one to use or wonder if they are different. This is even more confusing to someone who may use a screen reader but who can also see the screen because they will hear the fixed and currently offscreen navigation but will not know where those items are located. And, for the keyboard only user the fixed menu items come first on the page but they do not receive focus. So, the user will encounter several tab stops with no visible focus when they first enter the page.

Current code

```
<div  class="js-fixed-nav fixedNav row row--o pillar pillar--b pillar--f">
<div class="well--xxs_p split split--flag">
<div class="js-dashboard-mini"><div><div class="avatar avatar--01 avatar--xs
fixedNav-avatar">
<div class="avatar-circle"></div>
</div>
<div class="fixedNav-progress progress progress--a well well--xxs well--t
show--s">
<div class="progress-fill" style="width: calc(35% - 2px)">
<div class="progress-lt split split--flag" style="width: 285.7142857142857%">
<div class="split-cell plxs show--m">
<p class="truncate mbf">Typing Seedling (1/5)</p>
</div>
<div class="split-cell prxs show--xl">
<div class="tsxs">4,559 xp to go</div>
</div>
</div>
</div>
<div class="progress-dk split split--flag">
<div class="split-cell plxs show--m">
<p class="truncate mbf">Typing Seedling (1/5)</p>
</div>
<div class="split-cell prxs show--xl">
<div class="tsxs">4,559 xp to go</div>
</div>
</div>
</div></div></div>
<div class="split-cell">
<ul class="list list--flag list--s list--inline">
<li class="list-item fixedNav-navBlock">
<ul class="list list--inline list--inlineNav list--s stall stall--right stall-
-xs stall_l">
<li class="list-item fixedNav-navLinks">
<a href="/student/lessons" class="fixedNav-navLink navLink navLink--s navLink-
-b link--nu ">
Lessons</a>
</li>


<li class="list-item fixedNav-navLinks">
```

```
<a href="/student/progress" class="fixedNav-navLink navLink navLink--s
navLink--b link--nu ">
My Progress</a>
</li>


<li class="list-item fixedNav-navLinks">
<a href="/student/games" class="fixedNav-navLink navLink navLink--s navLink--b
link--nu ">
Games</a>
</li>

<li class="list-item">
<a href="/student/achievements" class="fixedNav-navLink navLink navLink--s
navLink--b link--nu is-active">
Achievements</a>
</li>
</ul>
</li>

<li class="list-item show--l">
<ul class="list list--inline list--flag list--xs">
<li class="list-item has-tooltip ">
<div class="js-skin-flag flag flag--app is-active"></div>
<div class="tooltip-template">
<div class="tooltip tooltip--i">
<div class="tooltip-arrow"></div>
<div class="tooltip-inner"></div>
</div>
</div>
<div class="tooltip-html">
<h4 class="tac well well--xs">
Choose a theme:
</h4>
<ul class="list list--f">

<li class="js-skin list-item" data-id="1">
<a class="btn btn--stroke-s btn--i btn--s theme theme--app">

<div class="theme-image">
<img src="/dist/student/images/avatar/app/avatar-01.svg">
</div>

Grow Your Skills
</a>
</li>

<li class="js-skin list-item" data-id="2">
<a class="btn btn--stroke-s btn--i btn--s theme theme--fox">

<div class="theme-image">
<img src="/dist/student/images/avatar/fox/avatar-01.svg">
</div>

The Flying Fox
</a>
</li>
```

```
<li class="js-skin list-item" data-id="3">
<a class="btn btn--stroke-s btn--i btn--s theme theme--princess">

<div class="theme-image">
<img src="/dist/student/images/avatar/princess/avatar-01.svg">
</div>

Princess Penny
</a>
</li>

<li class="js-skin list-item" data-id="4">
<a class="btn btn--stroke-s btn--i btn--s theme theme--forest">

<div class="theme-image">
<img src="/dist/student/images/avatar/forest/avatar-01.svg">
</div>

Party Bear
</a>
</li>

<li class="js-skin list-item" data-id="5">
<a class="btn btn--stroke-s btn--i btn--s theme theme--sea">

<div class="theme-image">
<img src="/dist/student/images/avatar/sea/avatar-01.svg">
</div>

Cappy the Sailor
</a>
</li>

<li class="js-skin list-item" data-id="6">
<a class="btn btn--stroke-s btn--i btn--s theme theme--pixels">

<div class="theme-image">
<img src="/dist/student/images/avatar/pixels/avatar-01.svg">
</div>

The Pixel Knight
</a>
</li>

<li class="js-skin list-item" data-id="7">
<a class="btn btn--stroke-s btn--i btn--s theme theme--space">

<div class="theme-image">
<img src="/dist/student/images/avatar/space/avatar-01.svg">
</div>

Eek From Space
</a>
</li>

<li class="js-skin list-item" data-id="8">
```

```
<a class="btn btn--stroke-s btn--i btn--s theme theme--superhero">

<div class="theme-image">
<img src="/dist/student/images/avatar/superhero/avatar-01.svg">
</div>

Super Eli
</a>
</li>

<li class="js-skin list-item" data-id="9">
<a class="btn btn--stroke-s btn--i btn--s theme theme--nitrotype">

<div class="theme-image">
<img src="/dist/student/images/avatar/nitrotype/avatar-01.svg">
</div>

Nitro Type
</a>
</li>

<li class="js-skin list-item" data-id="10">
<a class="btn btn--stroke-s btn--i btn--s theme theme--basic">

Basic Skin
</a>
</li>

</ul>
</div>
</li>

<li class="list-item">
<div class="dropdown dropdown--r dropdown--b">
<div class="tsxs split split--flag split--xxs">
<div class="split-cell">
test-1
</div>
<svg class="icon arrow arrow--xs split-cell">
<use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="/dist/student/images/icons.svg#arrow"></use>
</svg>
</div>
<ul class="list dropdown-items list--f">

<li class="list-item dropdown-item"><a href="/student/account"
class="dropdown-link tss">
Account Settings
</a></li>

<li class="list-item dropdown-item"><a href="#" class="dropdown-link tss js-
logout">
Log Out</a></li>
</ul>
</div>
</li>
```

```
</ul>
</li>
</ul>
</div>
</div>
</div>
```

Solution

Add Display:none on the fixed menu item section when this menu is scrolled off of the screen. This will hide the items from all users until it is made visible. Remove the display attribute when the item is fixed and visible at the top of the screen. You should always hide this fixed menu from screen reader users since a screen reader user encounters items in source code order. When this is marked as aria-hidden, the screen reader user will only encounter the menu in the banner. Once that is properly marked as a navigation landmark, the screen reader user will use that to quickly navigate to the navigation within the banner.

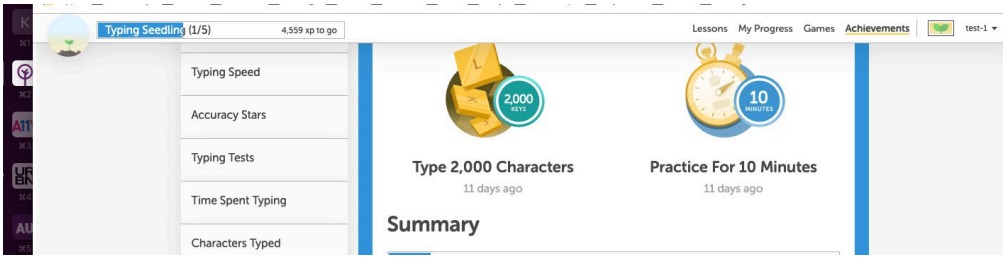
Recommended code

```
<div aria-hidden="true"  style="display:none;" class="js-fixed-nav fixedNav
row row--o pillar pillar--b pillar--f">
```

Screenshot(s)

Figure 7 Screen shot of main navigation

Figure 8 Screenshot of "fixed" navigation



6. Empty Link text

Pages	On all pages with the linked logo
Affected community	Vision, Cognitive
WCAG Success Criterion	1.1.1 Non-text content (A), 2.4.4 Link Purpose (In Context) (A)
WCAG Level	A
Priority	High

Description

The link to the EduTyping home page has no link text. Thus, it is not announced by screen readers. A screen reader user just hears that there is a link but not where it goes. This would be very confusing to screen reader users as they would wonder what they were missing.

The problem is that the link "text" has been implemented using a background image of the the text "EduTyping". Background images are ignored by screen readers. All images must have a text equivalent that can be accessed by assistive technologies.

Screenshot

Figure 9: Wherever the EduTyping logo is displayed as a link, there is no actual link text for screen readers and other assistive technology

Current code

```
<div class="hero-splitCell split-cell">
<a href="/student/lessons" class="logo logo--hero logo--edutyping mrs db"></a>
</div>

.logo--edutyping {
    background-image: url(/dist/student/images/logo-edutyping.svg);
    height: 35px;
    max-width: 200px;
```

Solution

Use a real img element within the anchor element. Add the alt text, alt="EduTyping home" to the img. Remove the background image from the logo--edutyping style. Note that you may need to add other style adjustments, beyond what is shown.

Recommended code

```
<div class="hero-splitCell split-cell">
<a href="/student/lessons" class="logo logo--hero logo--edutyping mrs db"><img
src="/dist/student/images/logo-edutyping.svg" alt="EduTyping home"></a>
</div>

.logo--edutyping {
    height: 35px;
    max-width: 200px;
```

7. Inaccessible media content

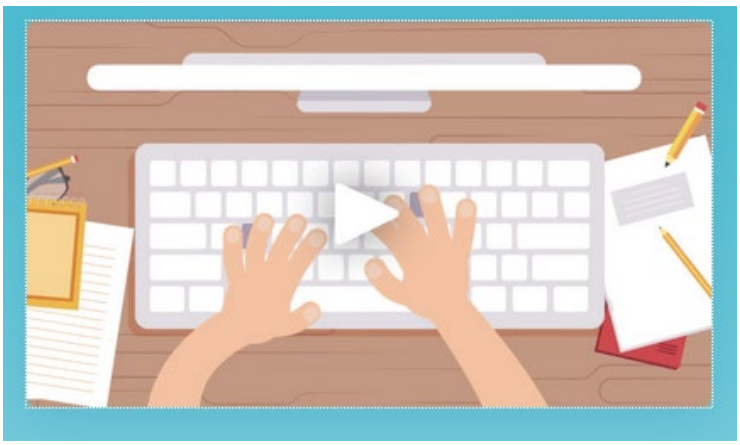
Lesson pages that have video content or intros	https://www.edutyping.com/student/lesson/110169/home-row-space-bar-and-enter-keys https://www.edutyping.com/student/lesson/110139/5-and-6-keys https://www.edutyping.com/student/lesson/110247/4-5-6-and-enter-keys https://www.edutyping.com/student/lesson/110117/the-abc-s https://www.edutyping.com/student/lesson/110316/review-all-of-the-middle-row https://www.edutyping.com/student/lesson/110362/parts-of-a-computer
Affected community	Vision, Cognitive, Mobility
WCAG Success Criterion	1.1.1 Non-text Content 1.2.2 Captions (Prerecorded) 1.2.5 Audio Description (Prerecorded)
WCAG Level	A, AA
Priority	High

Description

Many videos and animations in these pages have audio content that is not captioned. In some instances demonstrations or illustrations in these video components are not described for blind users. The result is that people who are deaf, hard of hearing, and/or have limited access to speakers or audio output will miss important audio content. Similarly, animations and video content that provide visual information will be inaccessible to blind students.

Screenshot

Figure 10 Videos are embedded in players that are not keyboard accessible and that have no captions, audio description, or captions to provide content to students with visual, hearing, or cognitive disability.



Solution

Learn about elements that make media content accessible to all users. For example, different design choices are required to make media accessible to people who cannot hear, people who cannot see, and others. Some media needs all of these elements, and some may need only a transcript.

- Captions/Subtitles — Captions provide content to people who are Deaf and others who cannot hear the audio. Captions are a text version of the speech and non-speech audio information needed to understand the content.
- Transcripts — Basic transcripts are a text version of the speech and non-speech audio information needed to understand the content. *Descriptive transcripts* also include visual information needed to understand the content.
- Audio Description — Audio description provides content to people who are blind and others who cannot see the video adequately. It describes visual information needed to understand the content. If the audio content describes the activities sufficiently audio description may not be needed.
- Media player — Media players have different levels of accessibility support. For example, some provide a separate audio track for description and some use the caption file to provide an *interactive transcript*.
- Video and Audio Content — There are also accessibility requirements for the video or audio content itself. For example, in videos, avoid flashing that can cause seizures.

Resources

Creating Accessible Video from the University of Washington

8. Pages are incompatible with screen readers and other assistive technology

Pages	All pages
Affected community	Vision, Cognitive, Mobility
WCAG Success Criterion	4.1.1 Parsing (A), 4.1.2 Name, Role, Value (A)
WCAG Level	A
Priority	High

Description

Using these pages with a screen reader is not practical or possible as a learning experience. The JAWS screen reader mostly doesn't respond, I believe that is due to the amount of scripting used to generate the page. The lack of screen reader response happens with both JAWS and NVDA in Chrome and Firefox. Once the user reaches the end of the header section, the screen reader basically stops responding. We can not further navigate normally through the page. We were able to continue some navigation by using the screen reader features to pull up a list of headings or links. Although some general areas of the EduTyping website are navigable with a screen reader, the typing lessons are entirely inaccessible, and thus a blind person cannot access the core features of the product. We tested using JAWS versions 2018 and 2019 and NVDA 2019.1 plus Chrome 74 and verified that setup/installation worked correctly on other internet pages. Note that there is some improvement with the VoiceOver screen reader on OS X. Here is the narrative of a screen reader user trying to work through lessons and track progress:

Typing Lessons | Learn Touch Typing – EduTyping:
https://www.edutyping.com/student/lessons
Unit and lesson titles are organized by heading, but their order is not hierarchical. Units are at H5, lessons at H3. There are images with unhelpful alt text. Elements within units (Letters & Common Symbols, Words & Sentences, etc.) are not identified as links or buttons.

Typing Lessons | F, J, and Space Bar – EduTyping:
https://www.edutyping.com/student/lesson/110090/f-j-and-space-bar

1. There are no headings nor landmarks, so navigation is tedious.
2. The normal function of the Tab key has been disabled, so tab navigation is mostly nonexistent. Tabbing within a series of links is fine, but tab functionality quickly breaks down throughout most of the page.
3. Diagram of Qwerty keyboard can be found, but no elements of the keyboard are labeled, so it is unusable by the screen reader user. Each element is on a line of its own, and there is no way to know where a new row of the keyboard starts.
4. Input keystrokes in the lesson are not communicated between screen reader and application. Must either first issue a “pass key through” command and then type a letter, or close the screen reader, type a series of letters, and then reload the program. Both are unacceptable.

Notepad – EduTyping:
https://www.edutyping.com/student/notepad

The text input field on this page is unlabeled. “EduTyping Notepad” is at H2 and there are no further headings or landmarks on the page.

My Progress – EduTyping:
https://www.edutyping.com/student/progress

The titles of each unit are presented at H3 and span columns within each table. Column headers within the tables are not set up properly, so I don’t know what column I’m in as I move across the rows of the tables.

Achievements – EduTyping:
https://www.edutyping.com/student/achievements
Types of achievements are organized by heading, but their order is not hierarchical. Recent Achievements are at H3, specific achievement types are at H4. There are images with unclear alt text. Achievement categories (Summary, Avatar Levels etc.) are not identified as links or buttons. Pressing Enter on an element updates the page, but this is far from ideal behavior.

Account Settings – EduTyping:
https://www.edutyping.com/student/account
Change Password is at H4. Apart from this heading, there are no other headings or landmarks on the page. Text input fields are labeled correctly. Update Password is neither a link nor a button. Pressing Enter does submit the form, but this is not ideal since the user has to guess at expected behavior.

Solution

We strongly recommend fixing the issues based on those that are reported in detail and then retesting. Use semantic elements wherever possible rather than generic elements and scripting.

Conclusion

The B.E. Publishing pages reviewed in this assessment contain a significant number of barriers that will block access to information by users with disabilities.

Particular attention should be paid to keyboard accessibility of all controls. Wherever possible use native HTML elements for user interface controls. These elements include support for keyboard and correctly identify themselves to assistive technology. Where scripting is necessary to create the user interface controls, add full keyboard support and WAI-ARIA roles, states, and properties to identify the control and its behaviors.

Review the page and update the colors to meet color contrast requirements. Check all images for appropriate alt text.

Ensure there is alternative text for image content.

Provide captions for all audio content and audio description if needed for video content.

Once your team has reviewed these findings, we welcome the opportunity to provide more detailed, in person feedback that will help you make this useful resource more broadly accessible. Thanks for the opportunity to work with B.E. Publishing.

Contact Information

Knowbility, Inc.
1033 La Posada, Suite 372
Austin, Texas 78754
512 527-3138

Appendix 1: Web Content Accessibility Guidelines

Table 1: Success Criteria, Level A and AA

Criteria	Level	URL
1.1.1 Non-text Content	A	https://www.w3.org/TR/WCAG20/#text-equiv-all
1.2.1 Audio-only and Video-only (Prerecorded)	A	https://www.w3.org/TR/WCAG20/#media-equiv-av-only-alt
1.2.2 Captions (Prerecorded)	A	https://www.w3.org/TR/WCAG20/#media-equiv-captions
1.2.3 Audio Description or Media Alternative (Prerecorded)	A	https://www.w3.org/TR/WCAG20/#media-equiv-audio-desc
1.2.4 Captions (Live)	AA	http://www.w3.org/TR/WCAG20/#media-equiv-real-time-captions
1.2.5 Audio Description (Prerecorded)	AA	http://www.w3.org/TR/WCAG20/#media-equiv-audio-desc-only
1.3.1 Info and Relationships	A	https://www.w3.org/TR/WCAG20/#content-structure-separation-programmatic
1.3.2 Meaningful Sequence	A	https://www.w3.org/TR/WCAG20/#content-structure-separation-sequence

1.3.3 Sensory Characteristics	A	https://www.w3.org/TR/WCAG20/#content-structure-separation-understanding
1.4.1 Use of Color	A	https://www.w3.org/TR/WCAG20/#visual-audio-contrast-without-color
1.4.2 Audio Control	A	https://www.w3.org/TR/WCAG20/#visual-audio-contrast-dis-audio
1.4.3 Contrast (Minimum)	AA	http://www.w3.org/TR/WCAG20/#visual-audio-contrast-contrast
1.4.4 Resize Text	AA	http://www.w3.org/TR/WCAG20/#visual-audio-contrast-scale
1.4.5 Images of Text	AA	http://www.w3.org/TR/WCAG20/#visual-audio-contrast-text-presentation
2.1.1 Keyboard	A	https://www.w3.org/TR/WCAG20/#keyboard-operation-keyboard-operable
2.1.2 No Keyboard Trap	A	https://www.w3.org/TR/WCAG20/#keyboard-operation-trapping
2.2.1 Timing Adjustable	A	https://www.w3.org/TR/WCAG20/#time-limits-required-behaviors
2.2.2 Pause, Stop, Hide	A	https://www.w3.org/TR/WCAG20/#time-limits-pause
2.3.1 Three Flashes or Below Threshold	A	https://www.w3.org/TR/WCAG20/#seizure-does-not-violate
2.4.1 Bypass Blocks	A	https://www.w3.org/TR/WCAG20/#navigation-mechanisms-skip
2.4.2 Page Titled	A	https://www.w3.org/TR/WCAG20/#navigation-mechanisms-title
2.4.3 Focus Order	A	http://www.w3.org/TR/WCAG20/#navigation-mechanisms-focus-order
2.4.4 Link Purpose (In Context)	A	http://www.w3.org/TR/WCAG20/#navigation-mechanisms-refs

2.4.5 Multiple Ways	AA	http://www.w3.org/TR/WCAG20/#navigation-mechanisms-mult-loc
2.4.6 Headings and Labels	AA	http://www.w3.org/TR/WCAG20/#navigation-mechanisms-descriptive
2.4.7 Focus Visible	AA	http://www.w3.org/TR/WCAG20/#navigation-mechanisms-focus-visible
3.1.1 Language of Page	A	http://www.w3.org/TR/WCAG20/#meaning-doc-lang-id
3.1.2 Language of Parts	AA	http://www.w3.org/TR/WCAG20/#meaning-other-lang-id
3.2.1 On Focus	A	http://www.w3.org/TR/WCAG20/#consistent-behavior-receive-focus
3.2.2 On Input	A	http://www.w3.org/TR/WCAG20/#consistent-behavior-unpredictable-change
3.2.3 Consistent Navigation	AA	http://www.w3.org/TR/WCAG20/#consistent-behavior-consistent-locations
3.2.4 Consistent Identification	AA	http://www.w3.org/TR/WCAG20/#consistent-behavior-consistent-functionality
3.3.1 Error Identification	A	http://www.w3.org/TR/WCAG20/#minimize-error-identified
3.3.2 Labels or Instructions	A	http://www.w3.org/TR/WCAG20/#minimize-error-cues
3.3.3 Error Suggestion	AA	http://www.w3.org/TR/WCAG20/#minimize-error-suggestions
3.3.4 Error Prevention (Legal, Financial, Data)	AA	http://www.w3.org/TR/WCAG20/#minimize-error-reversible
4.1.1 Parsing	A	http://www.w3.org/TR/WCAG20/#ensure-compat-parses
4.1.2 Name, Role, Value	A	http://www.w3.org/TR/WCAG20/#ensure-compat-rsv