Designing for Keyboard Accessibility

In this module, you'll learn:

- What keyboard accessibility is.
- Why keyboard accessibility is important.
- How you can design for keyboard accessibility.
- Best practices for keyboard accessibility.

What is keyboard accessibility?

At the heart of keyboard accessibility is the idea that any component activated using a mouse or other pointer interactions should also function with a keyboard. As a designer at MURAL, your job is to ensure you design a keyboard compatible UI. While a developer's job is to implement keyboard interactions, your role as a designer is to:

- Define these interactions.
- Document them in the wireframe.
- Ensure they're consistent throughout the interface.

Keyboard accessibility includes elements such as:

- Creating keyboard equivalents for all pointer interactions.
- Ensuring all content is operable through a keyboard interface and doesn't include timed individual keystrokes.
- Ensuring a logical tab order and that focus does not get trapped on any part of the page.

Resource: Watch this Keyboard Compatibility video to learn about the benefits and impact of designing for keyboard support.

Why is keyboard accessibility important?

Using a mouse requires both vision and fine motor skills that can create barriers to interacting with an application, a document, or a webpage. For people with disabilities, keyboard support provides multiple ways of interacting. It helps enable speech interaction, on-screen keyboards, scanning software, and assistive technology such as screen readers and sip and puff devices.

Example: People who experience temporary or permanent physical impairments such as hand tremors or carpal tunnel syndrome may find it easier to use a keyboard than a mouse. Those with vision impairments can press keyboard shortcuts to activate screen reader functions.

Designing for people with disabilities also benefits others. Many people simply don't enjoy using a mouse. Computer programmers frequently use keyboard shortcuts that they find more convenient and efficient than having to switch between a keyboard and a mouse.

At MURAL, one specific challenge/opportunity we have is providing keyboard support for collaborators to navigate the canvas without depending on a mouse. They can move between interactive objects such as sticky notes, shapes, images, and other items.

How can I design for keyboard accessibility?

As a designer, one of the first steps to designing for keyboard accessibility is to think about the overall user experience. Consider the different paths along each user type's journey.

How does the end-user navigate and interact with the content only using a keyboard?

Definitions:

User type: characterization of how a user interacts with an interface (e.g. keyboard user, mouse user, screen reader user, voice control user, etc.)

Navigation: actions and techniques that guide or move end users through an app or website, enabling them complete goals and interact with a product (e.g. how end users get from points A to B to C using focus indicators, menus, tabs, breadcrumbs, etc.).

Interaction: how an end-user engages with a product or carries out actions that allow them to achieve an objective (e.g. end user behaviors or actions such as pressing shortcut keys, clicking, swiping, tapping, etc.)

The best thing you can do to design for keyboard accessibility is to think about how the end-user can complete their journey without a mouse. Then map or walk through that journey in both the planning and implementation stages of your design. At the same time, you should design interactions that try to reduce end user fatigue.

To design for keyboard accessibility, keep these things in mind:

- **Design a keyboard equivalent for all pointer interactions and ensure functionality isn't time-dependent:** Consider how an end-user can achieve the same result or goal using a keyboard.

    - Use keyboard shortcuts for navigation. Typically, it's best to make shortcuts ***an addition to****,* not a replacement for, tab and arrow navigation.
    - Determine whether there are any timed keystrokes (e.g. keyboard navigation within a toast notification that automatically disappears). If any functions are time-dependent, create an accessible keyboard alternative whenever possible.

- **Ensure all mouse-operated components can also be accessed with a keyboard:** Any element that can be operated with a mouse should be reachable and operable by keyboard or have a keyboard equivalent.
    - Ensure interactive content inside hover information is reachable by keyboard and that components with multiple mouse operations have keyboard equivalents.

Example: Don't get tripped up! Consider details such as end users changing the skin tone of the thumbs up reaction in MURAL.

    - Provide a keyboard equivalent for drag and drop.

Resource: Refer to the **Creating Accessible Mouse Pointer Interactions** module in this training to see examples of drag and drop alternatives and to learn more about accessible mouse pointer interactions.

- **Follow established keyboard conventions whenever possible** so that users know how to interact with elements on the page:
    - Choose standard HTML elements when possible as these usually include keyboard interactions familiar to end users.
    - Use traditional space and enter keys for activating or triggering an element. This standard method is usually the most familiar to end users.

Example: Nonstandard or custom keyboard interactions can support efficiency but can also confuse end-users and be difficult to learn or remember.

- **Standard keyboard shortcuts** are those typically used in applications or webpages and what users expect.
    - In MURAL, you can still press **Control + Z** to undo an action, just as you can in Microsoft Word, Google Docs, and other tools.
- **Nonstandard shortcuts** are typically those that are unfamiliar to end users and tend to be application-specific.
    - In MURAL, you can press **Alt + H** on a PC or **Option + H** on a Mac to enter or exit nested navigation mode within an area of the canvas.

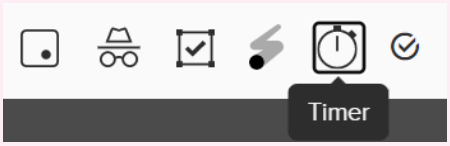
    - Be sure all custom elements have full keyboard support.
    - Rely on standards from the W3C (World Wide Web Consortion) by checking the ARIA (Accessible Rich Internet Applications) authoring practices site to find common conventions you can use for custom components instead of coming up with new conventions. ARIA follows established HTML conventions.

- **Use shortcut key combinations that include modifier keys whenever possible:** When shortcuts don't include modifier keys, such as Ctrl, end users are more likely to accidentally trigger them.
    - **Shortcut keys that include a combination of character and modifier keys are one of the easiest ways to prevent shortcuts from being activated unintentionally.**
    - Ensure keyboard shortcuts that use only letter, punctuation, number, or symbol characters include any of the following:
        - A mechanism to turn off the shortcut.
        - A mechanism to remap the shortcut to modifier keys (e.g. Alt).
        - Ability to activate the shortcut only when it receives focus.

Resource: Refer to the articles **Keyboard shortcuts** and **MURAL screen reader and keyboard features improve canvas accessibility** to see a list of MURAL shortcuts, many of which include modifier keys.

- **Provide a visible and distinguishable focus indicator:** Sighted keyboard users are only aware of their current location on the screen if the focus indicator is visible. These end users may not be able to navigate without it.
    - A focus indicator can be a border around an element, a highlight, or any other visual that clearly shows the end user's location on the screen.
    - Ensure focus indicators meet a 3:1 color contrast requirement.

Example: The focus indicator for the MURAL menu is a black border. It shows the name of the menu item when focused, like in the screenshot below where focus indicates the Timer has been selected. Note that the focus state creates the same impact as a hover state by showing the tooltip!



    - Ensure the keyboard focus responds in a logical way to end-user requests and scripted events.

Example: When modal is opened, the focus automatically moves inside the modal, responding in a logical and expected way.

    - Ensure that multiple keyboard-equivalent operations don't conflict with standard focus movements to and from components.

Example: There should only be one visible focus indicator and one element that has focus.

- **Include a logical keyboard tab order:** A logical tab order will often read like a book to a keyboard user when implemented correctly. A disorganized or illogical tab order can cause confusion about the content on the page.
    - **To create a logical tab order:**
        - Consider the end user's entire journey through the tab order to ensure all content is accessible and follows a logical path.

- Determine what is the highest importance or the most commonly used elements and make sure those are in a tab order that's both understandable and efficient for navigation.
- Create a tab order that moves from left to right and top to bottom.

Example: Components that are anchored, such as modals, don't always follow the existing flow of content or the typical end user journey. A logical tab order for a modal might include making sure the close button is the last element to receive focus. With this order, keyboard and screen reader users don't have to backtrack to access the content of the modal.

- Ensure that information triggered on mouse hover is not missing from the tab order.

Example: When thinking about tab order as a MURAL designer, you'll need to consider the order of the various toolbars and areas within the MURAL canvas (the top bar, left bar, bottom bar, and the canvas itself) and determine the most important or commonly used items. Ensure these items are included in the tab order in a way that's understandable and efficient for end user navigation.

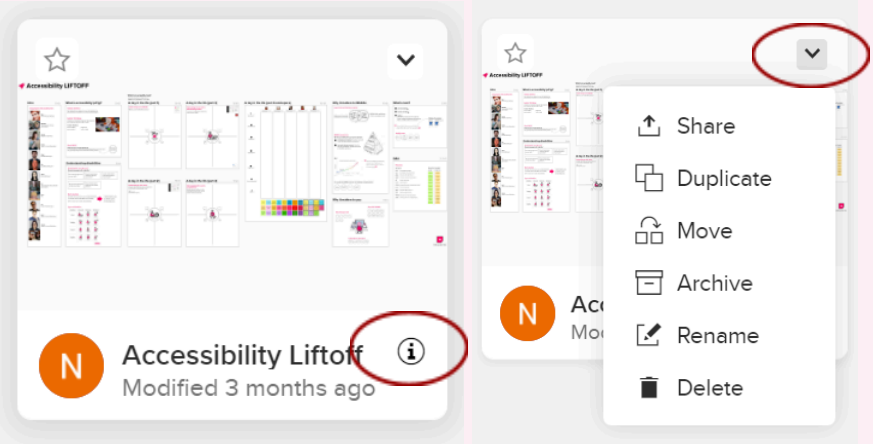
- **Reduce tabbing by grouping related components:** Keyboard users are often forced to move through multiple tabs to begin a task. You can lessen tab fatigue by reducing tab stops.
  - Consider how to group related components. One way to do this is to group components as a single tab stop with elements that can be navigated by arrow keys.

Example: The screenshots below show two versions of the same MURAL card, an example of a component with multiple tab stops. You can click the card to open the mural.

- The first screenshot shows that you can also click the information icon on the card to find out when the mural was created and last modified.
- The second screenshot presents items from the dropdown menu, which is opened when you click the arrow on the top right of a mural card.

Keyboard users are able to tab to these features and press Enter to access them. Each card has 4 to 5 tab stops without a lot of unique context, which can cause end user fatigue.

MURAL cards present an opportunity for designers to experiment like an owner. For example, each card on the dashboard could be a tab stop but the individual actions within (favorite, more information, actions dropdown, etc) could be navigated through arrows or shortcuts.

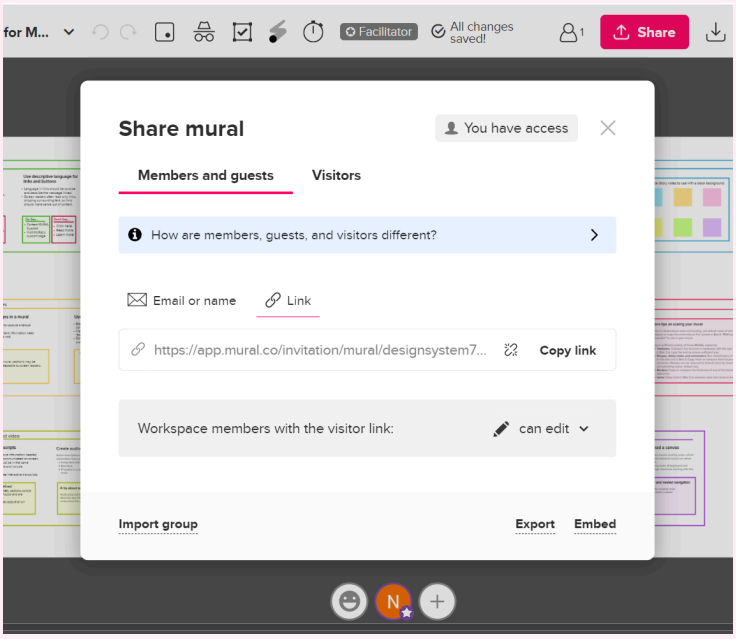


- **Prevent keyboard traps:** An end user should be able to navigate away from content in an intuitive and logical way without a mouse.
    - **Ensure the Escape key always cancels, closes, or exits a mouse action. This is one of the easiest ways to reduce keyboard traps.**
        - Allow users to press Escape to exit from dialog or activated components and to dismiss notices, modals, and overlays where possible.
    - Provide a control for end users to exit a widget or other content if needed.
    - Include instructions to end users on how to exit a website or app that contains complex keyboard interactions or other potential traps.

Example: In many use cases, you'll find keyboard shortcut support in the context menu of an application. As an end user navigates, it may display to provide that information. At MURAL, keyboard shortcuts can be reach in the context menu by pressing Shift +?.

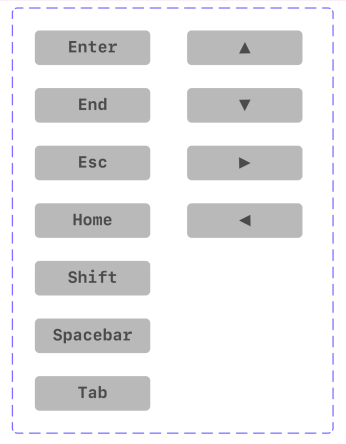
- Treat modals as a unique page. They're an exception to keyboard traps in that the content behind a modal should be inaccessible to keyboard users.

Example: The screenshot below demonstrates the modal exception to keyboard traps. When the Share mural modal pops up, the background is grayed out and inaccessible, and the modal functions as a unique page. You can still press Escape to exit the modal and return to the canvas, eliminating any keyboard traps.



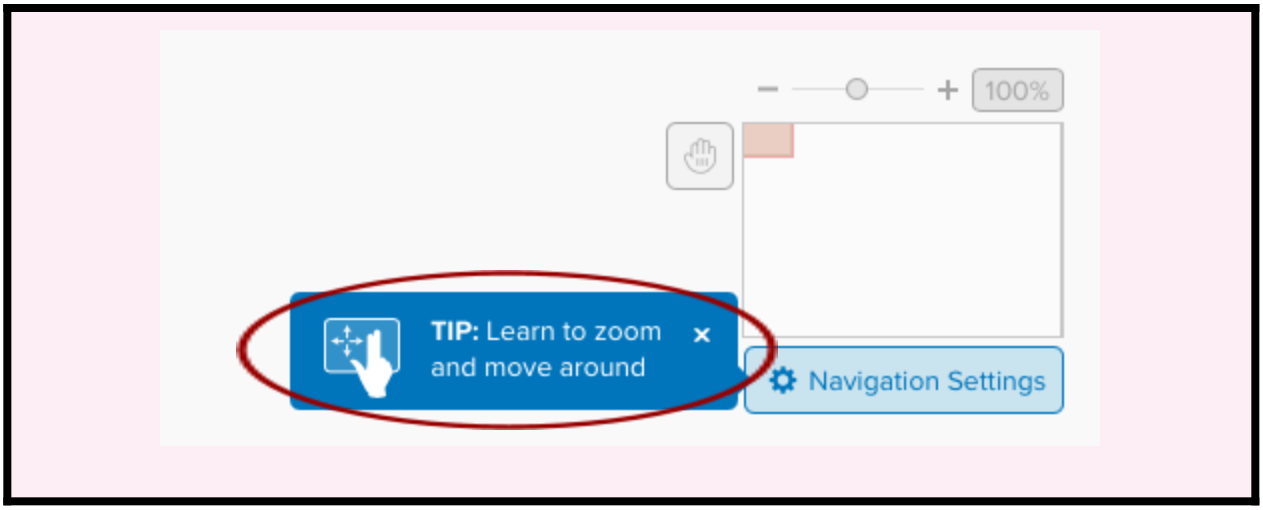
- **Document keyboard interactions throughout the wireframe and interface as needed:** While a developer implements keyboard interactions, it's your job to ensure they're defined consistently and documented throughout.
  - Provide annotations of the entire keyboard user journey to ensure developers implement them and test for keyboard support and potential focus traps.
  - Document all mouse operable components and the navigation and operation of their keyboard equivalents, especially components that have multiple mouse operations.
  - Include annotations of the intended keyboard function of custom components.

Example: The screenshot below shows an example of wireframe documentation for keyboard shortcuts in MURAL.



- Note the tab order. Make sure the wireframe shows how interactive components or hover triggers are included in the tab order or can be operated with a keyboard.
- Document groups of related components, including tab stops and the keys (usually the arrows) that will enable interaction between groups.
    - Notate the arrow keys between individual components and the single tab stop for groups of components.
- While your documentation should be thorough and avoid making assumptions about keyboard implementations, you should also avoid overwhelming the wireframe with notes. Include separate notes if needed.
    - This is typically done with numbered annotations or by using multiple artboards (e.g. one of which shows the overall tab order of tab stops and another that highlights the arrow key navigation within a specific related group.)
- Provide visible keyboard instructions to end users in the interface for any custom components or to explain how to exit a website or app when keyboard operation is complex or less familiar. Don't always rely on the user to access the help menu or search through and try to remember key commands for nonstandard shortcuts.

Example: The screenshot below is an example of in-context instructions. It's one method you Can use as a MURAL designer to provide keyboard support guidance



Best practices for keyboard accessibility, in a nutshell

Be sure to do the following when you're designing for keyboard accessibility:

- **Design a keyboard equivalent for all pointer interactions and ensure functionality isn't time-dependent.**
- **Ensure all mouse-operated components can also be accessed with a keyboard whenever possible.**
- **Follow established keyboard conventions whenever possible.**
- **Use shortcut key combinations that include modifier keys whenever possible.**
- **Provide a visible and distinguishable focus indicator.**
- **Include a logical keyboard tab order.**
- **Reduce tabbing by grouping related components.**
- **Prevent keyboard traps.**
- **Document keyboard interactions throughout the wireframe and interface as needed.**

Test your knowledge

Choose one or more answers to the following questions.

1. As a MURAL designer, you're tasked with creating keyboard shortcuts for MURAL features. What's the most accessible solution?

    A. Make sure a feature can be accessed using several different character key shortcuts. The more the better, right?

    B. Provide custom keyboard shortcuts to show off your fancy design skills.

    C. Create custom shortcuts that replace tab and arrow keys. Novelty keeps users engaged in navigation.

    D. Provide a standard keyboard shortcut that includes a modifier key if possible. Familiar is better.

2. You're tasked with trying to make a MURAL feature more accessible! You realize keyboard users need to tab through several components of one feature before being able to complete a task. How should you handle this as a designer?

    A. Reduce tabbing by grouping related components as a single tab stop.

    B. Create custom shortcuts that use the same character keys.

    C. Do nothing. Tabs are accessible and this feature is perfect as-is.

    D. Make sure the end user can just press escape and give up.

3. One of the easiest ways to reduce keyboard traps is to:

    A. Create shortcut keys that include modifier keys.

    B. Ensure the Escape key always cancels, closes, or exits a mouse action.

    C. Ponder, pontificate, contemplate, and think about how the end user can complete their entire journey with a mouse.

    D. Make sure the component can be accessed with a mouse.

4. A designer should ensure keyboard shortcuts that use only letter, punctuation, number, or symbol characters:

    A. Are customized to prevent confusion.

    B. Include the ability to activate the shortcut only when it receives focus.

    C. Provide a control for the end user to exit a widget or other content if needed.

    D. Include instructions for end users on how to navigate with them.

5. You've designed several custom shortcuts for MURAL features. Based on the answers below, how can you make sure these are implemented and accessible to the end user?

A. Provide instructions or tips for end users in the UI.

B. Notate these in the wireframe so that a developer knows to implement them.

C. Rely on the user to access the help menu to learn about the shortcuts.

D. None of the above.

E. Both A and B.