Disclaimer: This resource provides general guidance and is not legal advice. Consult qualified counsel for jurisdiction-specific requirements.

Keyboard Navigation Patterns

Ensuring keyboard accessibility for all interactive elements

Prepared by: AccessGuard

Version: 1.0

Date: October 18, 2025

Standards: WCAG 2.2 AA, Section 508

Keyboard Navigation Patterns

Table of Contents

[Back to TOC]

Keyboard accessibility is fundamental to digital accessibility. Many users rely on keyboards for navigation, including users with motor disabilities, visual impairments, and those who simply prefer keyboard shortcuts.

Core Principles

[Back to TOC]

1. **All interactive elements must be keyboard accessible**
2. **Focus indicators must be visible**
3. **Focus order must be logical**
4. **No keyboard traps**
5. **Keyboard shortcuts must be documented**

WCAG Requirements

[Back to TOC]

WCAG 2.1.1 Keyboard (Level A)

All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes.

WCAG 2.4.3 Focus Order (Level A)

If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability.

WCAG 2.4.7 Focus Visible (Level AA)

Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible.

Basic Keyboard Commands

Back to TOC

Standard Navigation

- **Tab**: Move forward to next focusable element
- **Shift+Tab**: Move backward to previous focusable element
- **Enter/Space**: Activate buttons and links
- **Arrow keys**: Navigate within components (menus, radio groups, etc.)
- **Escape**: Close dialogs, cancel operations
- **Home/End**: Go to beginning/end of list or line

Form Controls

- **Enter**: Submit form
- **Escape**: Reset form
- **Arrow keys**: Navigate radio buttons and some select menus
- **Space**: Toggle checkboxes and buttons
- **Ctrl/Cmd + Arrow**: Move by word in text inputs

Focus Management

Back to TOC

Tabindex

tabindex="0" - Add element to natural tab order

```
<div role="button" tabindex="0">Click me</div>
```

tabindex="-1" - Remove element from tab order but keep it focusable

```
<div role="alert" tabindex="-1" id="error-message"></div>
```

tabindex="1+" - NEVER USE (creates confusing tab order)

Focus Indicators

```
/* Default focus styles */
*:focus {
  outline: 2px solid #2563EB;
  outline-offset: 2px;
}

/* Custom focus styles */
button:focus,
a:focus,
input:focus,
select:focus,
textarea:focus {
  outline: 2px solid #2563EB;
  outline-offset: 2px;
  box-shadow: 0 0 0 4px rgba(37, 99, 235, 0.1);
}

/* High contrast mode support */
@media (prefers-contrast: high) {
  *:focus {
    outline: 3px solid;
  }
}
```

Common Patterns

Back to TOC

Buttons

Native Button

```
<button type="button">Click me</button>
```

Custom Button

```
<div
  role="button"
  tabindex="0"
  onclick="handleClick()"
  onkeydown="handleKeyDown(event)"
  aria-label="Click to submit form"
>
  Submit
</div>

<script>
function handleKeyDown(event) {
  if (event.key === 'Enter' || event.key === ' ') {
    event.preventDefault();
    handleClick();
  }
}
</script>
```

Links

Standard Link

```
<a href="/about">About Us</a>
```

JavaScript Link

```
<a
  href="#"
  onclick="handleClick(); return false;"
  onkeydown="handleKeyDown(event)"
>
  Show more
</a>

<script>
function handleKeyDown(event) {
  if (event.key === 'Enter') {
    handleClick();
  }
}
</script>
```

Dropdown Menus

```
<nav aria-label="Main navigation">
  <ul>
    <li>
      <button
        aria-expanded="false"
        aria-haspopup="true"
        id="menu-button"
        onclick="toggleMenu()"
      >
        Products
      </button>
```

```
      <ul
        role="menu"
        aria-labelledby="menu-button"
        id="menu"
        hidden
      >
        <li role="none">
          <a href="/products/web" role="menuitem">Web</a>
        </li>
        <li role="none">
          <a href="/products/mobile" role="menuitem">Mobile</a
        </li>
        <li role="none">
          <a href="/products/desktop" role="menuitem">Desktop<
        </li>
      </ul>
    </li>
  </ul>
</nav>

<script>
function toggleMenu() {
  const button = document.getElementById('menu-button');
  const menu = document.getElementById('menu');
  const isExpanded = button.getAttribute('aria-expanded') ===

  button.setAttribute('aria-expanded', !isExpanded);
  menu.hidden = isExpanded;

  if (!isExpanded) {
    menu.querySelector('[role="menuitem"]').focus();
  }
}

// Close menu on Escape
document.addEventListener('keydown', (e) => {
  if (e.key === 'Escape') {
    const button = document.getElementById('menu-button');
```

```
    const menu = document.getElementById('menu');
    button.setAttribute('aria-expanded', 'false');
    menu.hidden = true;
    button.focus();
  }
});
</script>
```

Modals and Dialogs

```
<!-- Trigger button -->
<button onclick="openModal()">Open dialog</button>

<!-- Modal -->
<div
  role="dialog"
  aria-modal="true"
  aria-labelledby="modal-title"
  id="modal"
  hidden
>
  <div class="modal-content">
    <h2 id="modal-title">Confirm action</h2>
    <p>Are you sure you want to continue?</p>
    <div class="modal-actions">
      <button onclick="confirmAction()">Confirm</button>
      <button onclick="closeModal()">Cancel</button>
    </div>
  </div>
</div>

<script>
let previousFocus;

function openModal() {
  const modal = document.getElementById('modal');
```

```
  const trigger = event.target;

  // Store previous focus
  previousFocus = trigger;

  // Show modal
  modal.hidden = false;

  // Focus first focusable element
  const firstFocusable = modal.querySelector('button');
  firstFocusable.focus();

  // Trap focus within modal
  trapFocus(modal);
}

function closeModal() {
  const modal = document.getElementById('modal');
  modal.hidden = true;

  // Restore focus to trigger
  if (previousFocus) {
    previousFocus.focus();
  }
}

function trapFocus(modal) {
  const focusableElements = modal.querySelectorAll(
    'button, [href], input, select, textarea, [tabindex]:not(
  );

  const firstElement = focusableElements[0];
  const lastElement = focusableElements[focusableElements.leng

  modal.addEventListener('keydown', (e) => {
    if (e.key === 'Tab') {
      if (e.shiftKey) {
        if (document.activeElement === firstElement) {
```

```
          e.preventDefault();
          lastElement.focus();
        }
      } else {
        if (document.activeElement === lastElement) {
          e.preventDefault();
          firstElement.focus();
        }
      }
    }

    if (e.key === 'Escape') {
      closeModal();
    }
  });
}
</script>
```

Tabs

```
<div role="tablist" aria-label="Product features">
  <button
    role="tab"
    aria-selected="true"
    aria-controls="panel-1"
    id="tab-1"
    onclick="switchTab(1)"
  >
    Overview
  </button>
  <button
    role="tab"
    aria-selected="false"
    aria-controls="panel-2"
    id="tab-2"
    onclick="switchTab(2)"
```

```
  >
    Pricing
  </button>
  <button
    role="tab"
    aria-selected="false"
    aria-controls="panel-3"
    id="tab-3"
    onclick="switchTab(3)"
  >
    Support
  </button>
</div>

<div
  role="tabpanel"
  id="panel-1"
  aria-labelledby="tab-1"
>
  <h3>Overview</h3>
  <p>Product overview content...</p>
</div>

<div
  role="tabpanel"
  id="panel-2"
  aria-labelledby="tab-2"
  hidden
>
  <h3>Pricing</h3>
  <p>Pricing information...</p>
</div>

<div
  role="tabpanel"
  id="panel-3"
  aria-labelledby="tab-3"
  hidden
```

```
>
  <h3>Support</h3>
  <p>Support information...</p>
</div>

<script>
function switchTab(tabNumber) {
  // Update all tabs
  document.querySelectorAll('[role="tab"]').forEach((tab, inde
    const isSelected = index + 1 === tabNumber;
    tab.setAttribute('aria-selected', isSelected);
    tab.setAttribute('tabindex', isSelected ? '0' : '-1');
  });

  // Update all panels
  document.querySelectorAll('[role="tabpanel"]').forEach((pane
    panel.hidden = index + 1 !== tabNumber;
  });
}

// Arrow key navigation
document.querySelector('[role="tablist"]').addEventListener('k
  const tabs = Array.from(document.querySelectorAll('[role="ta
  const currentIndex = tabs.indexOf(e.target);

  let nextIndex;
  if (e.key === 'ArrowRight') {
    nextIndex = (currentIndex + 1) % tabs.length;
  } else if (e.key === 'ArrowLeft') {
    nextIndex = (currentIndex - 1 + tabs.length) % tabs.length
  } else if (e.key === 'Home') {
    nextIndex = 0;
  } else if (e.key === 'End') {
    nextIndex = tabs.length - 1;
  }

  if (nextIndex !== undefined) {
    e.preventDefault();
```

```
    tabs[nextIndex].focus();
    tabs[nextIndex].click();
  }
});
</script>
```

Skip Links

```
<!-- Skip to main content -->
<a href="#main-content" class="skip-link">Skip to main content

<header>
  <nav>...</nav>
</header>

<main id="main-content">
  <!-- Main content -->
</main>

<style>
.skip-link {
  position: absolute;
  top: -40px;
  left: 0;
  background: #000;
  color: #fff;
  padding: 8px;
  text-decoration: none;
  z-index: 100;
}

.skip-link:focus {
  top: 0;
}
</style>
```

Testing Checklist

Back to TOC

Manual Testing

- [ ] All interactive elements are keyboard accessible
- [ ] Focus indicators are visible on all elements
- [ ] Tab order is logical and intuitive
- [ ] No keyboard traps in modals or dropdowns
- [ ] Escape key closes dialogs and dropdowns
- [ ] Arrow keys work in menus and tabs
- [ ] Enter and Space activate buttons
- [ ] Focus returns to trigger after closing modals

Automated Testing

- [ ] axe DevTools reports no keyboard accessibility violations
- [ ] No elements with positive tabindex values
- [ ] All custom interactive elements have keyboard handlers
- [ ] Focus styles meet 3:1 contrast ratio

Common Mistakes

Back to TOC

❌ Don't

- Remove focus indicators with `outline: none`
- Use positive tabindex values
- Create keyboard traps
- Rely solely on mouse events
- Hide interactive elements from keyboard navigation

✅ Do

- Always provide visible focus indicators
- Use logical tab order
- Implement keyboard event handlers for custom components

- Test with keyboard only
- Document keyboard shortcuts

Resources

Back to TOC

- **WCAG 2.1.1 Keyboard**: https://www.w3.org/WAI/WCAG22/Understanding/keyboard
- **WCAG 2.4.3 Focus Order**: https://www.w3.org/WAI/WCAG22/Understanding/focus-order
- **WCAG 2.4.7 Focus Visible**: https://www.w3.org/WAI/WCAG22/Understanding/focus-visible
- **ARIA Authoring Practices**: https://www.w3.org/WAI/ARIA/apg/patterns/
- **WebAIM Keyboard Testing**: https://webaim.org/techniques/keyboard/