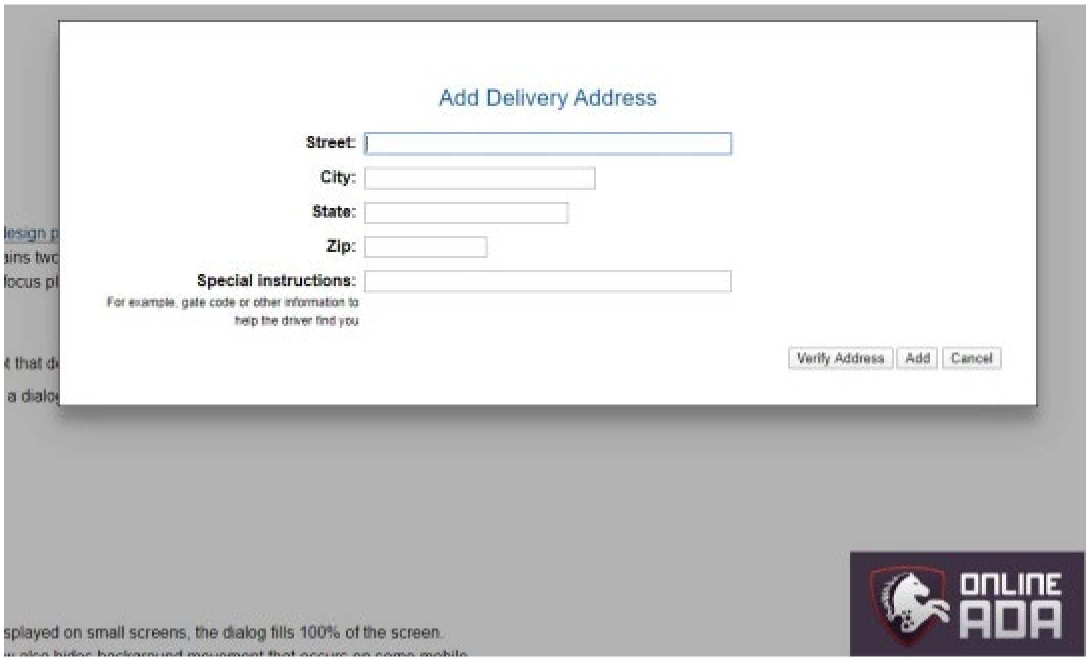
Online ADA Quick Guide

Making Pop-Ups & Modals Accessible

By the Online ADA Auditing Team

What are modals?

Modals and pop-ups have a variety of uses and come in many forms. Some common examples of modals include pop-up ads, alerts, and interfaces for capturing user input.



For example, gate code or other information to help the driver find you

splayed on small screens, the dialog fills 100% of the screen.

Example of a modal with user input.

Modals are intended to be used as a quick and simple way to capture an interaction from a user. They trap the user's focus (visual and navigational) in a window that is separated from the rest of the page content, blocking access to the contents on the main page until the modal is closed by the user. These modal windows are overlaid over the main page content, trapping keyboard focus in their windows and blurring out or dimming the main page content.

At their best, modals are non-intrusive and simple, adding value to the user experience. At their worst, modals can be cumbersome and confusing, frustrating users and forcing them to leave your website. Modals should be used sparingly and should be tested thoroughly before deployment to ensure that they can be used effectively by all users.

For a modal to be accessible, several criteria must be met:

1. Focus must be placed on the modal when it is opened.
2. Focus must be returned to the last focused element on the main page when the modal is closed.
3. Keyboard focus must be contained inside of modal while it is open.
4. ESC key must close the modal.
5. Proper ARIA roles must be assigned to the modal overlay.
6. The modal's CLOSE button or icon needs a proper label describing its function.

These elements are important to the overall accessibility of a website because they provide needed context to assistive technology. They create a detailed map of the page's content and ensure that a user does not get lost or distracted on their journey.

Common Issues

It's impossible to anticipate the needs of every unique user and combination of assistive technology, but thinking about users that use keyboards and screen readers for navigation can help us understand why the following problems are important to address and prevent.

1. Not trapping focus inside of the modal window so that users are able to access main page content via keyboard navigation while the modal is open.
   - Keyboard users and screen reader users have no way of closing the modal window if they leave the modal.
   - Screen reader users can access the main page content without knowing that they have left the modal.
2. Focus not assigned to the first element in the modal window after it is opened.
   - Keyboard and screen reader users could miss important instructions and / or information about the modal's function.
3. Input elements and interactive elements lacking proper labels.
   - Screen reader users could miss important instructions on the intended functionality of interactive elements and what type of information to provide for an input.
4. Elements lacking tabindex, out of tab order.
   - Keyboard and screen reader users will not be able to navigate the modal in a logical order.

Applicable WCAG 2.1 Guidelines

1.3.1

WCAG 2.1, Article 1.3.1 training video

Article 1.3.1 is the article that governs information and relationships. The text of 1.3.1 reads:

> Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text.

This article addresses the proper semantic structure of a page, which includes using HTML elements in a meaningful way and providing accessible labels, names, and identifiers to interactive elements such as forms, buttons, and links. It ensures that elements that provide visual order to a page provide order in the code as well. Since assistive technology relies on the underlying code of a webpage for its functionality, this guideline ensures that screen readers and other assistive technologies function predictably and reliably.

2.1.1

WCAG 2.1, Article 2.1.1 training video

WCAG Article 2.1.1 is the article that governs keyboard functionality of a webpage. The text of 2.1.1 reads:

> All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints.

This article ensures that all users have equal access to inputs, interactive elements, and widgets (buttons, inputs, date pickers, photo carousels, etc.) on a website. This means that all functionality that is operable with a mouse must be operable using a keyboard as well. Ensure that the button that triggers the modal can be navigated to using a keyboard, that the ESC key closes the modal, and that all elements on the inside of the modal are keyboard accessible. Keep in mind that the first element in the modal should receive focus after it is opened, and that keyboard focus must be contained within the modal until a user closes it.

2.1.2

WCAG 2.1 Article 2.1.2 training video

WCAG article 2.1.2 is the article that ensures that components of a webpage do not trap keyboard focus and that a keyboard can be used to close / exit the component. The text of 2.1.2 reads:

> If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved a way from that component using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away.

In the case of modals and pop-ups, this article ensures that users have a reliable way to close the component once activated. While keyboard focus being trapped in a pop-up or modal is desired functionality, a user must have a keyboard accessible way to exit / close the modal. This includes a close or "X" button at the top of the modal, a button under the main content of a modal, and using the ESC button as a trigger to close the modal.

4.1.2

WCAG 2.1, Article 4.1.2 training video

WCAG Article 4.1.2 is the article that governs states, properties, and values of components. The text of 4.1.2 reads:

> For all user interface components (including but not limited to: form elements, links, and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies.

This article ensures that assistive technology can determine the functionality of components such as modals and pop-ups by programmatic means, such as semantic roles and values. If desired functionality of a component cannot be achieved through using semantic HTML elements, then those elements must have a ROLE attribute that communicates their function to screen readers. For instance, if a modal uses an ANCHOR element for a close button, it must have a ROLE="BUTTON" attribute, since the element functions as a button.

Functionality

Modals and pop-ups should make use of some supplementary ARIA elements along with appropriate semantic HTML. This can help to ensure that they have the functionality expected by users.

ARIA Attributes and Roles

- ARIA-HIDDEN: The parent container of the modal needs a mechanism that toggles ARIA-HIDDEN from "TRUE" to "FALSE" based on the modal's state.
- ROLE="DIALOG": This role should be added to the modal's parent container.
- ARIA-MODAL: This attribute should be applied to the modal's parent container. Note: *ARIA-MODAL is fairly new and lacks support across some browsers / OSs. Do not rely on this attribute alone to communicate to assistive technologies that a modal is present.*
- ARIA-DESCRIBEDBY: Gives the modal an accessible description.
- ARIA-LABELLEDBY: Gives the modal an accessible name.

Code Example

Following is an example of a modal that captures a user's input using form input fields. The example demonstrates modal functionality described in this document including: proper form error labels, keyboard focus and navigation, proper use of input labels, and the proper use of ARIA attributes. It is written with HTML, CSS, and JavaScript.

Codepen Example: https://codepen.io/Online-ADA/pen/OJjLXyg

Keyboard Support

It's important to align the keyboard controls for your modal with the controls that habitual keyboard users expect.

Key	Function
Tab	• Moves focus to the next focusable element in the modal. • Moves focus to the first focusable element in the modal when focus is on the last focusable element in the modal.
Shift + Tab	• Moves focus to the previous focusable element in the modal. • Moves focus to the previous focusable element in the modal when focus is on the last focusable element in the modal.
Esc	Closes modal.

Keyboard Controls and their Corresponding Functions

Conclusion

Modals and pop-ups are excellent tools for ensuring that your users get all the information they need and for giving them ways to perform actions without going to a new page. They must be built carefully, however, to ensure that they can be operated by as wide a range of users as possible. Ensuring that they are keyboard accessible, that all interactive elements have programmatically associated labels, and that focus goes where expected when the modal is opened or closed will help all your users have a better experience on your website.