What is the aria-controls attribute, and how does it work?

The aria-controls attribute is used to create a programmatic relationship between an element that controls the presence of another element on the page, and the element it controls. This relationship can help screen reader users better understand how the control works. Common uses include a set of tabs that control the visibility of different sections of content, or a button that toggles the visibility of a menu.

Let's take a look at an example to see how this works. In this example of a tabbed interface, we have a div element with a set of three buttons:

<div role="tablist">

<button role="tab" id="tab1" aria-controls="section1" aria-selected="true">

Tab 1

</button>

<button role="tab" id="tab2" aria-controls="section2" aria-selected="false">

Tab 2

</button>

<button role="tab" id="tab3" aria-controls="section3" aria-selected="false">

Tab 3

</button>

</div>

Each button represents a tab and has a role attribute set to tab. In most tabbed interfaces, the first tab panel is visible by default, so the first tab button has an aria-selected attribute set to true to indicate that its associated section of content is currently visible The aria-controls attribute is used to associate each button with the section of content that it controls.

Here are the three sections of content that correspond to the tabs:

<div id="section1" role="tabpanel" aria-labelledby="tab1">

Section 1 content

</div>

<div id="section2" role="tabpanel" aria-labelledby="tab2" hidden>

Section 2 content

</div>

<div id="section3" role="tabpanel" aria-labelledby="tab3" hidden>

Section 3 content

</div>

Each section of content has a role attribute set to tabpanel and an aria-labelledby attribute that points to the corresponding tab to give the panel an accessible name. The hidden attribute is used to hide the sections of content that are not currently selected. Each section id matches the value of the aria-controls attribute on the corresponding button. The section1 id matches the aria-controls attribute on the first button, section2 matches the aria-controls attribute on the second button, and section3 matches the aria-controls attribute on the third button. This is how the association between the tabs and their sections of content is established.

To switch between the different sections you will need to use JavaScript to update the hidden attribute on the section and the aria-selected attribute on the button based on which section is currently visible. Only one section can be visible at a time and it should not have the hidden attribute and aria-selected should be set to true on its button. The remaining hidden sections should all have the hidden attribute and aria-selected should be set to false on their buttons.

Tabs are a common use case for the aria-controls attribute, but it can be used in other scenarios where one element controls the visibility or behavior of another element. Other examples include accordions, dropdown menus, and modals.

The next time you design a user interface that involves controlling the visibility of elements, consider using the aria-controls attribute to establish the relationship between the controlling element and the controlled element.