Browser Accessibility Inspection Tools

When trying to determine the source of an accessibility of web content there can be several levels where an issue might arise.

1. Improperly or not implemented features in your own code/markup
2. An issue with the browser not publishing to the accessibility api
3. The Assistive Technology not using the information from the accessibility api

These can be tricky to determine, luckily there are a few tools available to help inspect what is being published to the accessibility api. If we can determine that the correct values are being passed to the accessibility api, then the issue is likely at the AT level.

The tools listed below have a number of views. Two of the more relevant views are the accessibility tree (hereafter "a11y tree"), and the interface viewer.

The a11y tree is displayed on the left, and shows the hierarchy of accessible objects on the page. The is *not* a view of the DOM. Browsers use the DOM to construct the a11y tree based on built-in rules as well as explicit ARIA markup. An example of a built-in rule is an empty alt attribute for an <img> element. Empty alt text entails that the <img> is decorative or for layout; or, generally, not semantically relevant. Browsers will not create a node in the a11y tree for such <img> elements. Similarly, an element with an ARIA role of "presentation" will not have a corresponding node in the a11y tree. (Note: this is a generality -- any element that is focusable will appear in the a11y tree even if it is marked "presentation"). Generally, there are fewer branches in the a11y tree than in the DOM since the a11y tree displays the information at a more meaningful level. It abstracts over the nesting of divs, spans, and whatever to show the underlying toolbar and its tools (for example).

Select a node in the a11y tree view to inspect it more closely. The interface viewer on the right will show various accessible properties of the selected node. Properties include aspects of the accessible such as its role, its name and description (if any), and other properties that depend on the nature of the object. For example, if the accessible is a checkbox, then the interface viewer will show its checked state. Another example: If a slider, it will show its current, minimum, and maximum values.

When checking an accessible with respect to ARIA markup, the ARIA User Agent Implementation Guide provides a pair of tables that define the information in the AAPI. The two tables are the role mapping table, and the states and properties mapping table. Each table shows how an ARIA role or state/property is mapped to different AAPIs. Thus, if you are using Accerciser to inspect the a11y tree created by FireFox, the ATK/AT-SPI column of the table is your guide. Similarly, if using AccProbe on Windows with FireFox, use the MSAA+IAccessible2 column.

More documentation is given by the links below for each specific inspection tool.

Accprobe

Platform: WinXP, Win 7

http://accessibility.linuxfoundation.org/a11yweb/util/accprobe/#downloads

Accprobe Documentation

DOMInspector

Platform: Firefox Add-on (no accessibility info on Mac OS)

https://addons.mozilla.org/en-US/firefox/addon/dom-inspector-6622/

DOMInspector Documentation

Accerciser

Platform: GNOME

http://live.gnome.org/Accerciser#Downloads

Accerciser's online documentation is found on the GNOME library site. Also, once installed, documentation is available through Accerciser's help menu.