How IBM is making Web applications more accessible with WAI-ARIA

David Todd
IBM Human Ability & Accessibility Center
dltodd@us.ibm.com

Overview

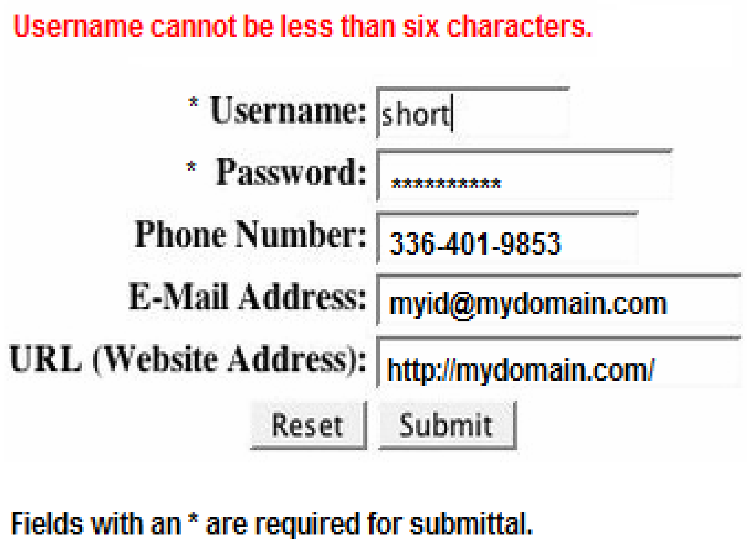
- How IBM Web applications notify screen reader users when:
  - Form input is required.
  - Form input is missing or incorrect.
- How applications use landmarks to enable more efficient page navigation.
- How IBM developers are coding tables so that our automated accessibility testing tool can determine the difference between a layout table and a data table.
- Keyboard navigation requirements for custom widgets.
- IBM requirements for implementing custom widgets so that name, role and value information is communicated to assistive technologies.

Widget problems

- Early Web applications were relatively simple to make accessible because accessibility was built into standard HTML elements.
- Modern Web applications have increasingly complex user interfaces.
- The UIs contain widgets like trees, menus and tab panels.
- Widgets are built using a combination of JavaScript, CSS and standard HTML elements.
- Widgets look like desktop widgets but they aren’t enabled for accessibility.
- Screen reader users find applications containing these widgets difficult if not impossible to use.

Form field problems

- Screen reader users have a difficult time identifying fields that require data before a form can be submitted.
- Screen reader users find it difficult to identify form fields that contain invalid data.



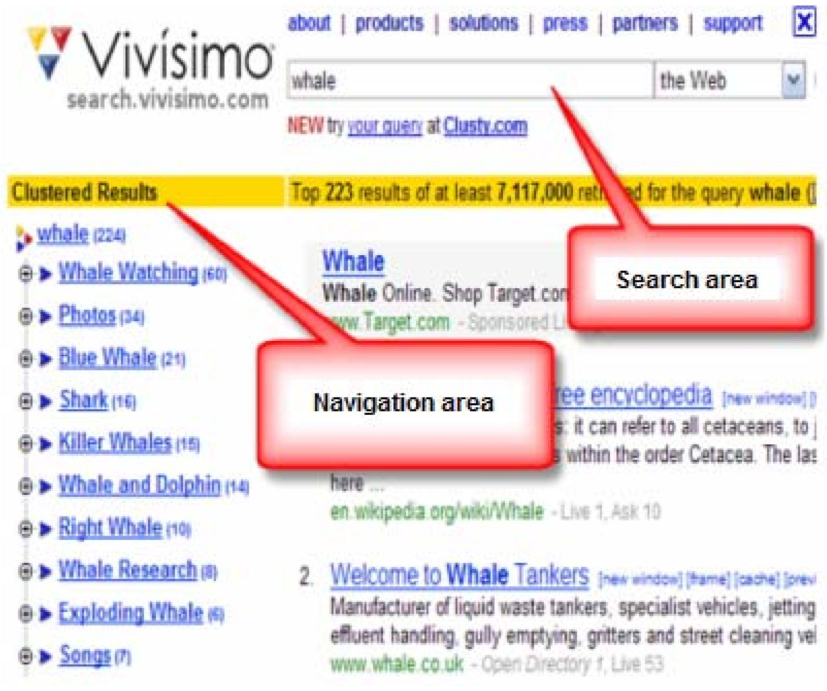
Username cannot be less than six characters.

* Username:	short
* Password:	**********
Phone Number:	336-401-9853
E-Mail Address:	myid@mydomain.com
URL (Website Address):	http://mydomain.com/

Fields with an * are required for submittal.

Navigation problems

- Screen reader users find it difficult to navigate Web pages because there are no standard navigational landmarks to areas like search or navigation.
- Keyboard only users find Web applications containing custom widgets difficult to navigate because they must do excessive tabbing.



Whale
Whale Online. Shop Target.co...
...w.Target.com - Sponsored L...

...ee encyclopedia [new window]
...s: it can refer to all cetaceans, to...
...within the order Cetacea. The las...
here ...
en.wikipedia.org/wiki/Whale - Live 1, Ask 10

2. Welcome to Whale Tankers [new window] [frame] [cache] [prev...
Manufacturer of liquid waste tankers, specialist vehicles, jetting
effluent handling, gully emptying, gritters and street cleaning ve...
www.whale.co.uk - Open Directory 1, Live 53

What’s the solution?

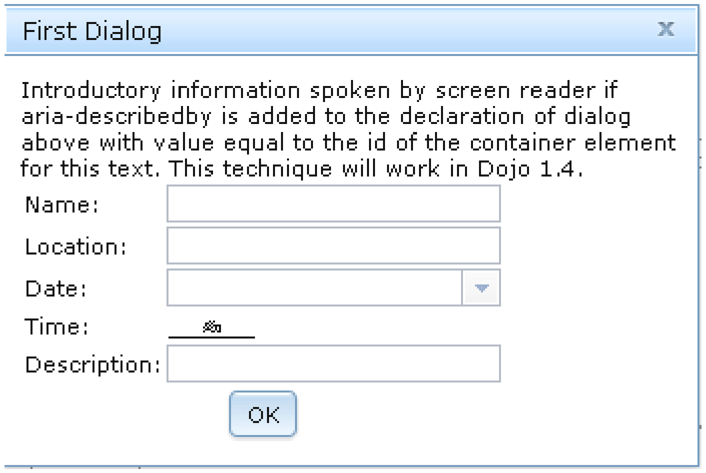
- Throw away HTML, JavaScript and CSS and start over with an accessible technology for building Web applications. NO WAY!

- We must find a way to add information to existing Web technologies in order to make Web applications more accessible.

Solution

- Augment existing Web technologies with WAI-ARIA information that communicates accessibility information to assistive technologies.
- What is WAI-ARIA? It's the Accessible Rich Internet Applications specification, and it defines a way to make Web applications more accessible to people with disabilities.
- The specification enables developers to embed WAI-ARIA markup in widgets and implement arrow key navigation.
- The goal is to enable operation of custom widgets in the same manor as desktop application widgets.
- IBM supports the specification by requiring developers to:
  - Incorporate WAI-ARIA attributes in Web applications.
  - Implement arrow key navigation in custom widgets.

Custom widgets

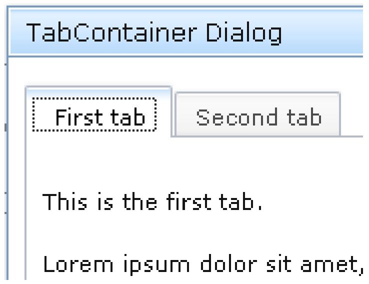

First Dialog

Introductory information spoken by screen reader if aria-describedby is added to the declaration of dialog above with value equal to the id of the container element for this text. This technique will work in Dojo 1.4.

Name:	
Location:	
Date:	
Time:	
Description:	

OK

Dialog

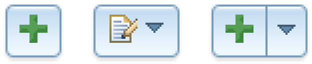

TabContainer Dialog

First tab | Second tab

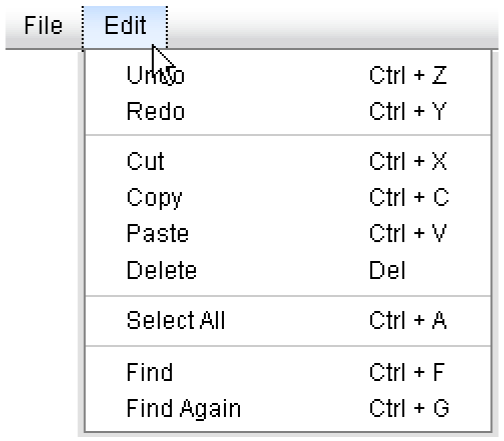
This is the first tab.

Lorem ipsum dolor sit amet,

Tabpanel

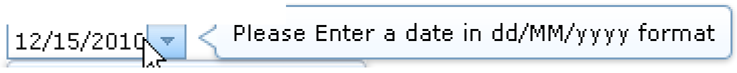


button


File | Edit

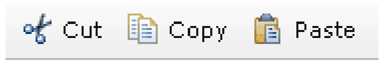
Undo	Ctrl + Z
Redo	Ctrl + Y
Cut	Ctrl + X
Copy	Ctrl + C
Paste	Ctrl + V
Delete	Del
Select All	Ctrl + A
Find	Ctrl + F
Find Again	Ctrl + G

Menu bar and menu

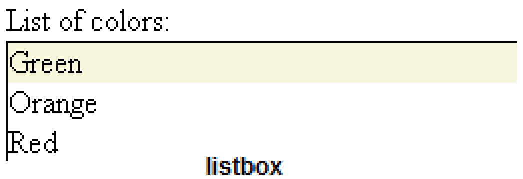

12/15/2010

Please Enter a date in dd/MM/yyyy format

Tooltip

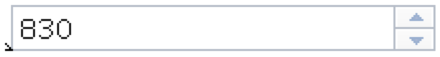

Cut Copy Paste

Toolbar

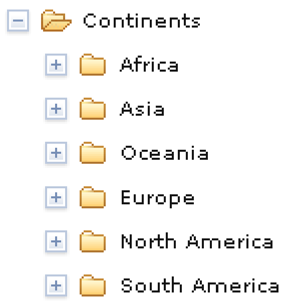

List of colors:

Green
Orange
Red

listbox


830

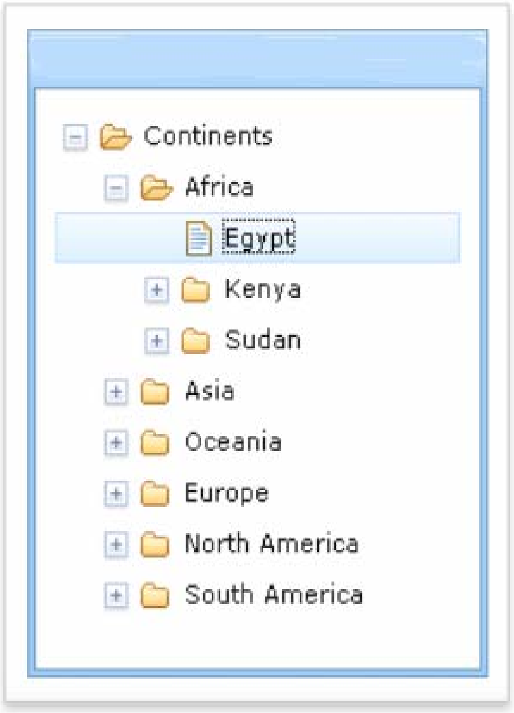
spinbutton


Continents
- Africa
- Asia
- Oceania
- Europe
- North America
- South America

Tree

Tree - missing accessibility information

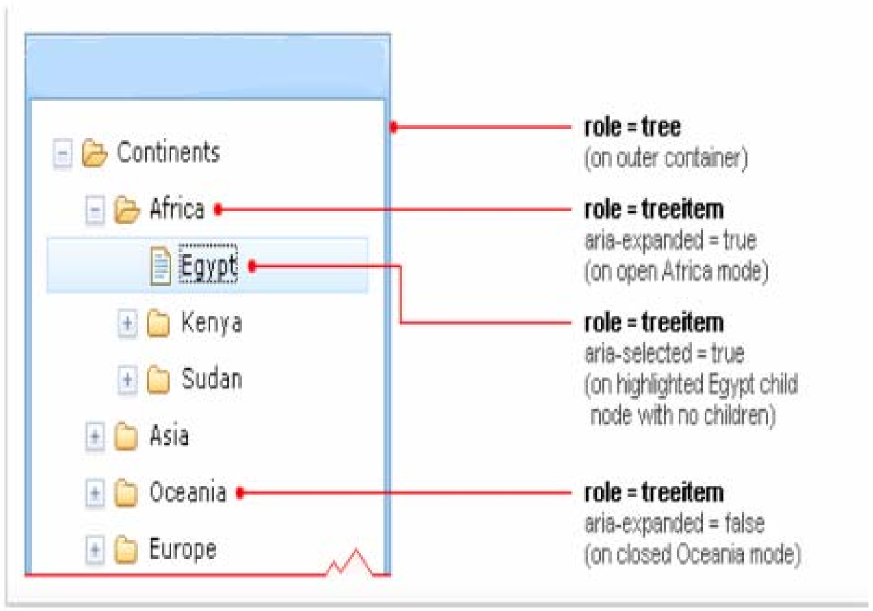
- Tree is developed with a combination of CSS, scripting and HTML elements.
- There's no mechanism to tell assistive technologies that:
  - The widget is a tree
  - It has expandable nodes.
  - It has multiple levels.
- A screen reader user has a difficult time determining:
  - What is this widget.
  - Whether nodes are open or closed.
  - Which node is selected.
  - Current location.
- Navigation: Must tab to each node in the tree.



Tree - WAI-ARIA attributes added

- Adding WAI-ARIA attributes to the tree enables a screen reader to tell the user:
  - The widget is a tree.
  - Which node has focus.
  - Their current location in the tree.
  - Whether a node is open or closed.

- The WAI-ARIA keyboard navigation model is implemented.
  - Navigate the tree with the arrow keys.
  - When a tree node has focus, tabbing places focus on the next focusable element after the tree.



Typical steps for implementing an accessible widget

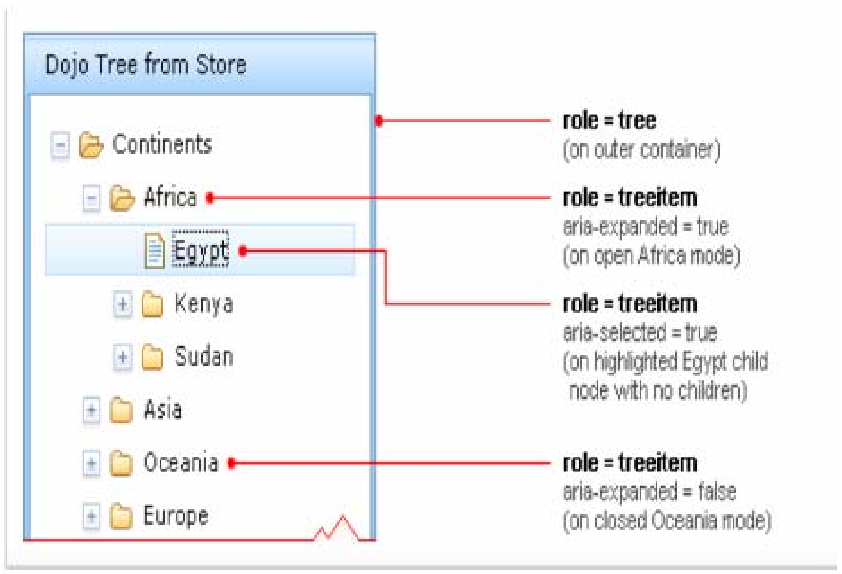
- Pick the widget type from the WAI-ARIA specification (e.g., tree, menu, toolbar, etc).
- See the WAI-ARIA authoring practices to see if there are best practices for implementing the widget.
- Examine an implementation to see how it works (e.g., codetalks.org).
- Establish the widget structure in the markup (parent/child).
- Implement the list of WAI-ARIA supported states and properties (e.g., tree – aria-expanded, aria-selected).
- Repeat this for the children.
- Implement arrow key navigation:
  - The ability to tab to a widget and then navigate its elements with the arrow keys.
  - While the widget has focus, the ability to press tab again to place focus on the next focusable element after the widget.

Typical steps for implementing an accessible widget (continued)

- Implement arrow key navigation.
  - Widget captures keyboard events.
  - Set the tabindex="-1" on child elements to allow them to be focusable.
  - ChildElement.focus()
  - Browser will fire a focus change event to the assistive technology.
  - If focus isn't visible, draw focus border using CSS styling.
- Font size - use "em" or % instead of "px" so users can so users can increase font size.
- Do not hard-code container sizes. Containers won't scale when font size is increased.
- Support high contrast mode.
- Test with a WAI-ARIA aware browser and screen reader.
  - Verify screen reader speaks proper role, label and state information.

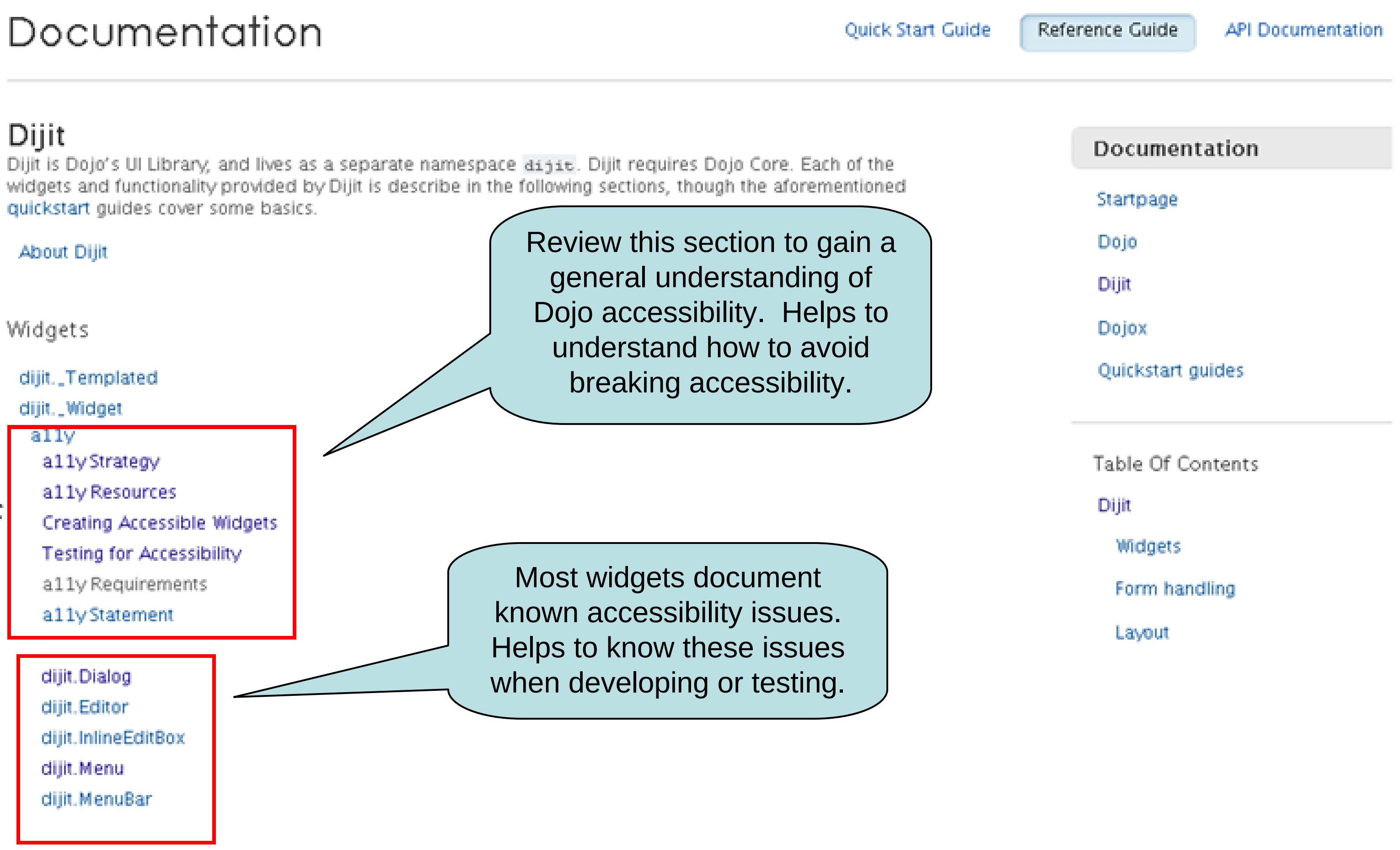
Dojo – IBM de facto standard for Web application user interfaces

- IBM committed resources to the Dojo Open Source project to help make the core widgets and the DataGrid widget accessible.
- The widgets contain the proper WAI-ARIA attributes to make them accessible when using a WAI-ARIA aware screen reader and browser.
- Keyboard support in IE 6, IE 7, Firefox 2, Firefox 3.
- Low Vision Support.
  - Windows high contrast support in IE 6, IE 7, FF2, FF3.
  - No fixed font sizes.
  - Support images off.



Dojo Dijit accessibility documentation

- When extending a widget, IBM developers consult the Dojo accessibility documentation in order to avoid breaking accessibility.



Documentation

Quick Start Guide | Reference Guide | API Documentation

Dijit

Dijit is Dojo's UI Library, and lives as a separate namespace `dijit`. Dijit requires Dojo Core. Each of the widgets and functionality provided by Dijit is describe in the following sections, though the aforementioned quickstart guides cover some basics.

About Dijit

Widgets

- dijit._Templated
- dijit._Widget
- a11y
  - a11y Strategy
  - a11y Resources
  - Creating Accessible Widgets
  - Testing for Accessibility
  - a11y Requirements
  - a11y Statement
- dijit.Dialog
- dijit.Editor
- dijit.InlineEditBox
- dijit.Menu
- dijit.MenuBar

Review this section to gain a general understanding of Dojo accessibility. Helps to understand how to avoid breaking accessibility.

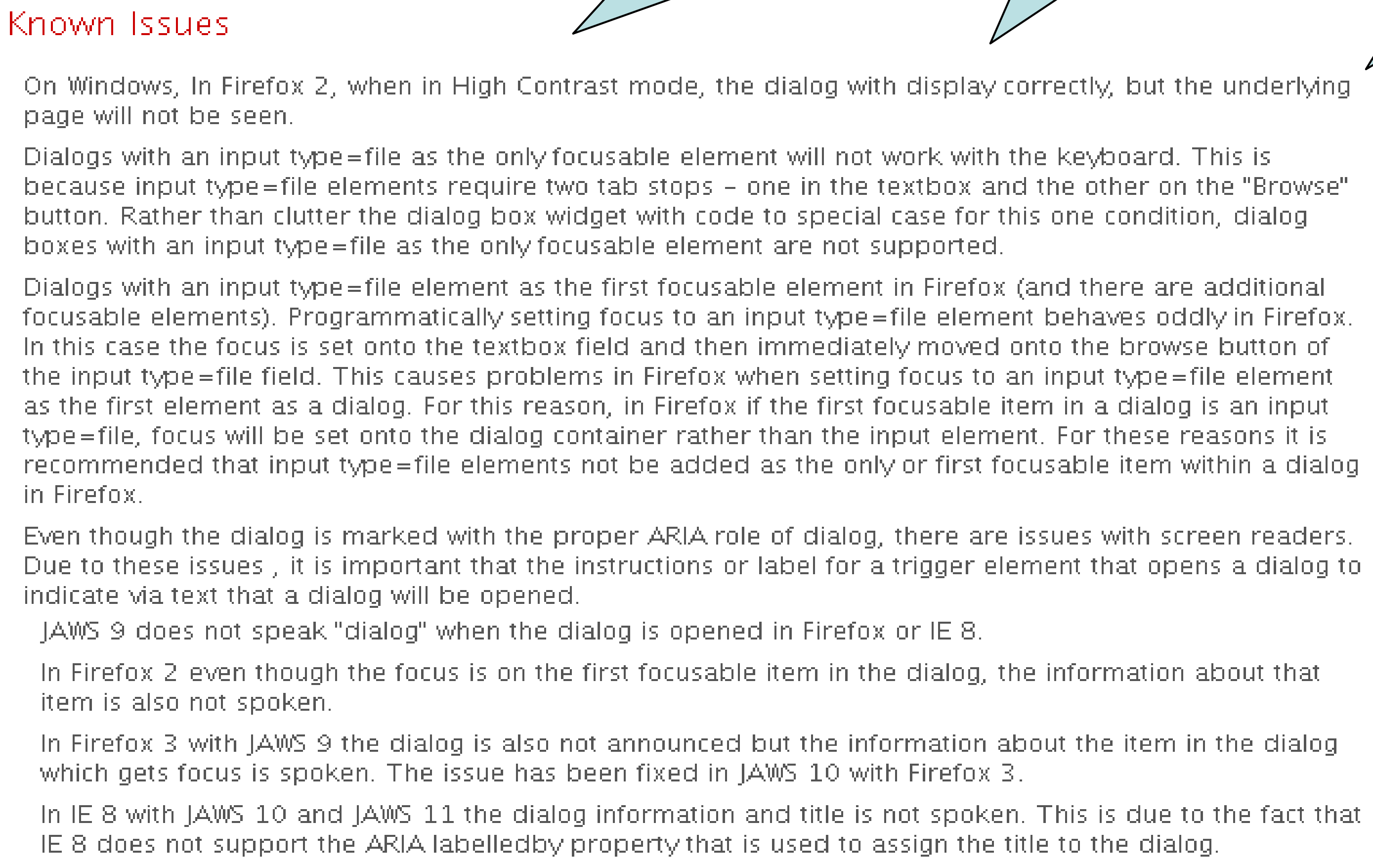
Most widgets document known accessibility issues. Helps to know these issues when developing or testing.

Dialog documentation

Why isn't JAWS behaving as expected?

Known keyboard navigation issues?

Known issues with high contrast mode or large fonts?



Known Issues

On Windows, In Firefox 2, when in High Contrast mode, the dialog with display correctly, but the underlying page will not be seen.

Dialogs with an input type=file as the only focusable element will not work with the keyboard. This is because input type=file elements require two tab stops - one in the textbox and the other on the "Browse" button. Rather than clutter the dialog box widget with code to special case for this one condition, dialog boxes with an input type=file as the only focusable element are not supported.

Dialogs with an input type=file element as the first focusable element in Firefox (and there are additional focusable elements). Programmatically setting focus to an input type=file element behaves oddly in Firefox. In this case the focus is set onto the textbox field and then immediately moved onto the browse button of the input type=file field. This causes problems in Firefox when setting focus to an input type=file element as the first element as a dialog. For this reason, in Firefox if the first focusable item in a dialog is an input type=file, focus will be set onto the dialog container rather than the input element. For these reasons it is recommended that input type=file elements not be added as the only or first focusable item within a dialog in Firefox.

Even though the dialog is marked with the proper ARIA role of dialog, there are issues with screen readers. Due to these issues , it is important that the instructions or label for a trigger element that opens a dialog to indicate via text that a dialog will be opened.

JAWS 9 does not speak "dialog" when the dialog is opened in Firefox or IE 8.

In Firefox 2 even though the focus is on the first focusable item in the dialog, the information about that item is also not spoken.

In Firefox 3 with JAWS 9 the dialog is also not announced but the information about the item in the dialog which gets focus is spoken. The issue has been fixed in JAWS 10 with Firefox 3.

In IE 8 with JAWS 10 and JAWS 11 the dialog information and title is not spoken. This is due to the fact that IE 8 does not support the ARIA labelledby property that is used to assign the title to the dialog.

Make Web applications easier to navigate with WAI-ARIA landmarks

- Add a WAI-ARIA `main` landmark to enable screen reader users to jump directly to the page main content.

  <h1 role="main">Main content heading</h1>

- If a page contains navigation links, assign a WAI-ARIA `navigation` role to the links container.

  <h2 role="navigation">Navigation</h2>

- If a search field is present, assign a WAI-ARIA `search` role to the search container.

  <div role="search">div containing a search field</div>

- If the same landmark is used multiple times on a page, each instance must have a unique label.
  - <h2 role="navigation" aria-label="site search">
  - <h2 role="navigation" aria-labelledby="searchID">

Make Web applications easier to navigate with WAI-ARIA landmarks

- Additional application landmarks for making Web applications easier to navigate:
  - application – designates a region of a page that executes a set of tasks for the user (e.g. calendar).
  - banner – contains site oriented content (e.g., logo).
  - complementary – marks content that is relevant to the main content, but remains meaningful when separated from the main content. For example, movie show times or related articles.
  - contentinfo – a region that contains information about the parent document (e.g., copyright, privacy statement). Only one of these can be used per page.
  - form – used to mark scripted controls when a standard form element isn’t used, but the controls cause a form submit.
- Screen reader user can navigate landmarks sequentially or select one from a list of landmarks.

Notify users when input is required

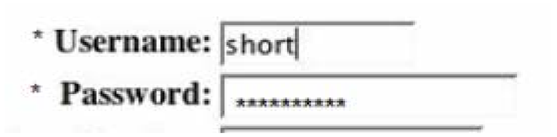
- Add an `aria-required=“true”` property to form fields that require input before a form can be submitted.

- An `aria-required=“true”` property is mandatory in addition to other indicators that a field is required (e.g., asterisk).

- When an form field receives focus and it has an `aria-required=“true”` attribute, a screen reader indicates that information is required.

- Example:

```
<label for="reqfield">* Username:</label>
<input name="Name" id="reqfield" type="text" aria-required="true" />
```



Notify users when input is incorrect

- Add an `aria-invalid =“true”` property to input fields when user input falls outside the required format or values.

- For example, if a user enters an invalid phone number in the following field, JavaScript updates the DOM to add an `aria-invalid` property to the input element:

```
<label for=”phone”>Phone: </label><input type="text" id="phone" ...>
```

- The JavaScript snippet that updates the DOM follows:

```
…
If (invalidTest) {
    document.getElementById(“phone”).setAttribute(“aria-invalid”, “true”);
}
…
```

Add a `presentation` role to layout tables

- Often, tools that are used for automated accessibility testing can’t tell the difference between a data table and a layout table.
- Hence, accessibility tools often generate false error messages about tables.
- The only way for an automated testing tool to definitively identify a layout table is by finding a `role=“presentation”` attribute on the table.
- Therefore, developers are required to add a presentation role to layout tables.
  - <table role=“presentation”> … </table>
- The `presentation` role causes a screen reader to ignore the table structure and read only the table content.

Automated accessibility testing with IBM Rational Policy Tester

- Policy Tester helps determine a site's level of compliance with Section 508 of the U.S. Rehabilitation Act and the W3C WCAG 2.0 guidelines.
- Developers can do a site crawl or inspect individual pages for accessibility problems.
- IBM and the Open Ajax Alliance Accessibility Working Group are developing a set of Open Source accessibility rules that Policy Tester will use to validate WAI-ARIA markup.
- The group is working on rules that, for example:
  - Validate correct parent/child role relationships.
  - Validate correct WAI-ARIA properties for roles.
  - Validate that WAI-ARIA property references are valid.

Summary of WAI-ARIA requirements for IBM Web applications

- Add WAI-ARIA properties to rich Internet widgets to communicate name, role and value information to assistive technologies.
- Implement WAI-ARIA arrow key navigation to mimic desktop widget navigation.
- Add WAI-ARIA landmarks to enable easier page navigation.
- Notify screen reader users when input is required by adding an `aria-required` property to form fields.
- Notify screen reader users when input is invalid by adding an `aria-invalid` property to form fields.
- Add role="presentation" to layout tables to enable more accurate table testing results.