Iguana – Accessibility & Usability

1 Introduction

1.1 Iguana and Customized interfaces

One of the major goals of Iguana is the delivery of “ideal”, “customized” interfaces. This can apply to customized interfaces for a specific device (for a so-called user-agent, e.g. for a mobile phone), for a specific type of user (e.g. a child, a grown-up) or for a specific disability (e.g. high contrast interfaces, or interfaces for blind people).

Technically this is solved through the concept of “site profiles” (e.g. “kids”, “mobile”), the support for user-agent detection and the concept of themes (e.g. high contrast / low vision skins).

A specific topic in this area is accessibility, - providing accessible interfaces for users with a visual impediment. This document describes in short the main considerations for the development of accessible interfaces within Iguana.

It is relevant to note that we give equal attention to the closely related topic of usability. Whilst accessibility is primarily focusing on a set of technical guidelines, the broad concept of usability focuses on overall providing of a “easy to use” interface.

Note

Infor supplies an accessible ‘base’, it is your responsibility to ensure that the website you create using Iguana meets the accessibility criteria.

1.2 Technical approach : WAI-ARIA

Iguana is a so-called widget based framework for the creation of dynamic interfaces (for websites, mobile interfaces, and others). This architecture implies that Iguana is a so-called “RIA” (Rich Internet Application), relying heavily on so-called AJAX technology (AJAX = Asynchronous JavaScript and XML). The usage of AJAX and widget technology that supports “dynamic” content poses its own specific challenges with regard to accessibility.

Iguana relies heavily on a JavaScript framework called dojo. One of the main reasons why dojo was selected as the JavaScript framework was its native support for WAI-ARIA.

“WAI-ARIA (Web Accessibility Initiative - Accessible Rich Internet Applications) *is a draft technical specification published by the World Wide Web Consortium that specifies how to increase the accessibility of dynamic content and user interface components developed with Ajax, HTML, JavaScript and related technologies. Web developers increasingly use client-side scripts to create user interface controls that cannot be created with HTML alone. They also use client-side script to update sections of a page without requesting a completely new page from a web server. Such techniques on websites are called rich internet applications.”* (from: http://en.wikipedia.org/wiki/WAI-ARIA, April 2011).

“WAI-ARIA, the Accessible Rich Internet Applications Suite, defines a way to make Web content and Web applications more accessible to people with disabilities. It especially helps with dynamic content and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies. Currently certain functionality used in Web sites is not available to some users with disabilities, especially people who rely on screen readers and people who cannot use a mouse. WAI-ARIA addresses these accessibility challenges, for

example, by defining new ways for functionality to be provided to assistive technology. With WAI-ARIA, developers can make advanced Web applications accessible and usable to people with disabilities.

[…]

Web sites are increasingly using more advanced and complex user interface controls, such as tree controls for Web site navigation […]. To provide an accessible user experience to people with disabilities, assistive technologies need to be able to interact with these controls. However, the information that the assistive technologies need is not available with most current Web technologies.

Another example of an accessibility barrier is drag-and-drop functionality that is not available to users who use a keyboard only and cannot use a mouse. Even relatively simple Web sites can be difficult if they require an extensive amount of keystrokes to navigate with only a keyboard.

Many Web applications developed with Ajax (also known as AJAX), DHTML, and other technologies pose additional accessibility challenges. For example, if the content of a Web page changes in response to user actions or time- or event-based updates, that new content may not be available to some people, such as people who are blind or people with cognitive disabilities who use a screen reader.

WAI-ARIA addresses these accessibility challenges by defining how information about this functionality can be provided to assistive technology. With WAI-ARIA, an advanced Web application can be made accessible and usable to people with disabilities.

[…]

More specifically, WAI-ARIA provides a framework for adding attributes to identify features for user interaction, how they relate to each other, and their current state. WAI-ARIA describes new navigation techniques to mark regions and common Web structures as menus, primary content, secondary content, banner information, and other types of Web structures. For example, with WAI-ARIA, developers can identify regions of pages and enable keyboard users to easily move among regions, rather than having to press Tab many times.

WAI-ARIA also includes technologies to map controls, Ajax live regions, and events to accessibility application programming interfaces (APIs), including custom controls used for rich Internet applications. WAI-ARIA techniques apply to widgets such as buttons, drop-down lists, calendar functions, tree controls (for example, expandable menus), and others.

WAI-ARIA provides Web authors with the following:

- *Roles to describe the type of widget presented, such as "menu," "treeitem," "slider," and "progressmeter"*
- *Roles to describe the structure of the Web page, such as headings, regions, and tables (grids)*
- *Properties to describe the state widgets are in, such as "checked" for a check box, or "haspopup" for a menu.*
- *Properties to define live regions of a page that are likely to get updates (such as stock quotes), as well as an interruption policy for those updates—for example, critical updates may be presented in an alert dialog box, and incidental updates occur within the page*
- *Properties for drag-and-drop that describe drag sources and drop targets*
- *A way to provide keyboard navigation for the Web objects and events, such as those mentioned above"*

(from http://www.w3.org/WAI/intro/aria.php, April 2011).

Further information on WAI-ARIA can be found at http://www.w3.org/WAI/intro/aria.php. The full specifications of the WAI-ARIA standard can be found at http://www.w3.org/TR/wai-aria/.

1.3 WAI-ARIA & dojo

Iguana relies heavily on WAI-ARIA technology. Iguana uses the dojo JavaScript framework. dojo's ARIA support is widely considered to be the most mature of all JavaScript frameworks.

Information on dojo and Web 2.0 Accessibility with WAI-ARIA FAQ can be found at http://wiki.codetalks.org/wiki/index.php/Web_2.0_Accessibility_with_WAI-ARIA_FAQ,

"Under utilization of ARIA is being addressed by the developers of ARIA, who are working directly with open source JavaScript toolkits projects such as **dojo**. It often makes sense for the authors of these toolkits, which provide solutions to a number of problems, such as security and cross-browser scripting. Because of this, entire companies and industries are moving forward with applications that use toolkits like **dojo**. Authors that utilize toolkits like **dojo** with ARIA support can thus get accessibility for free."

"Authors want both cross-browser compatibility and accessibility for free -- via easy-to-use, pre-built widgets. A number of JavaScript toolkits are quickly evolving powerful sets of widgets:

- dojo: ARIA support is mature
- YUI: ARIA support growing rapidly
- GWT: ARIA support growing rapidly
- Jquery: ARIA support in beginning stages"

More information on dojo and its WAI-ARIA support can be found at:

- Dijit Accessibility (a11y): http://dojotoolkit.org/reference-guide/dijit/a11y/index.html
- Creating Accessible Widgets : http://www.dojotoolkit.org/reference-guide/quickstart/writingWidgets/a11y.html
- Dijit Accessibility Resources : http://dojotoolkit.org/reference-guide/dijit/a11y/resources
- Testing Widgets for Accessibility : http://www.dojotoolkit.org/reference-guide/quickstart/writingWidgets/a11yTesting.html#id1
- Real World Web Accessibility With WAI-ARIA : http://ewh.ieee.org/conf/accessingthefuture/documents/gibson.pdf.

2 Accessibility features per profile

Iguana provides a specific property per profile which will slightly change Iguana's behaviour to improve accessibility and usability for visually impaired users. The property **Accessibility features** is set per profile.

If Accessibility features is set to Yes, the following behaviour is triggered:

- for all templates Iguana uses templates that have an extension “vi”, - i.e. if Iguana uses profiles A, B, C, D and E if the Accessibility features property is set to No, it will use the profiles Avi, Bvi, Cvi, Dvi and Evi if the Accessibility features property is set to Yes

- the Iguana default pop-up widgets are displayed “in page” instead of as a pop-ups (alert and confirm messages however are shown via the standard browser boxes, unless they appear during a pop-up, in which case they are displayed “in page”)

- the Iguana default dropdown boxes (from dojo) are replaced by standard HTML dropdown boxes

- in My profile, the “Personal data” and ‘Linked accounts” widgets are by default expanded (instead of collapsed)

- in Search and My Profile, sort dropdown boxes are moved to the main widget (instead of the options widget) and a “re-sort” button is added to the dropdown box

- in Search, on the results set page, hyperlinks are added to the top of the main widget for access to the widgets in the side column that contains the Options, Restrictions, Associations and similar widgets.

The accessibility features do not require a specific CSS and are included in the base.css.

3 Testing

Iguana’s accessibility features are developed and tested in co-operation with Libraries for the Blind, who have played a major role in testing the accessibility features.

Testing was primarily done with JAWS (screen reader), ZoomText (magnifier) and NVDA (screen reader).

“JAWS (*Job Access With Speech*) is a computer screen reader program in Microsoft Windows that allows blind and visually impaired users to read the screen either with a text-to-speech output or by a Refreshable Braille display.” (from http://en.wikipedia.org/wiki/Job_Access_With_Speech, January 2012). JAWS is the most used screen reader program worldwide (see e.g. http://webaim.org/projects/screenreadersurvey2/ for some market share data).

“ZoomText Magnifier enlarges and enhances everything on your computer screen, making all of your applications easy to see and use.” (from http://www.aisquared.com/zoomtext, January 2012).

“NonVisual Desktop Access (NVDA) is a free and open source screen reader for the Microsoft Windows operating system. Providing feedback via synthetic speech and Braille, it enables blind or vision impaired people to access computers running Windows for no more cost than a sighted person. Major features include support for over 20 languages and the ability to run entirely from a USB drive with no installation.” (from http://www.nvda-project.org/, February 2012).

3.1 Minimum requirements

The minimum requirements for these applications are:

- JAWS 11 and up

- ZoomText 9 and up
- NVDA 2011.3.

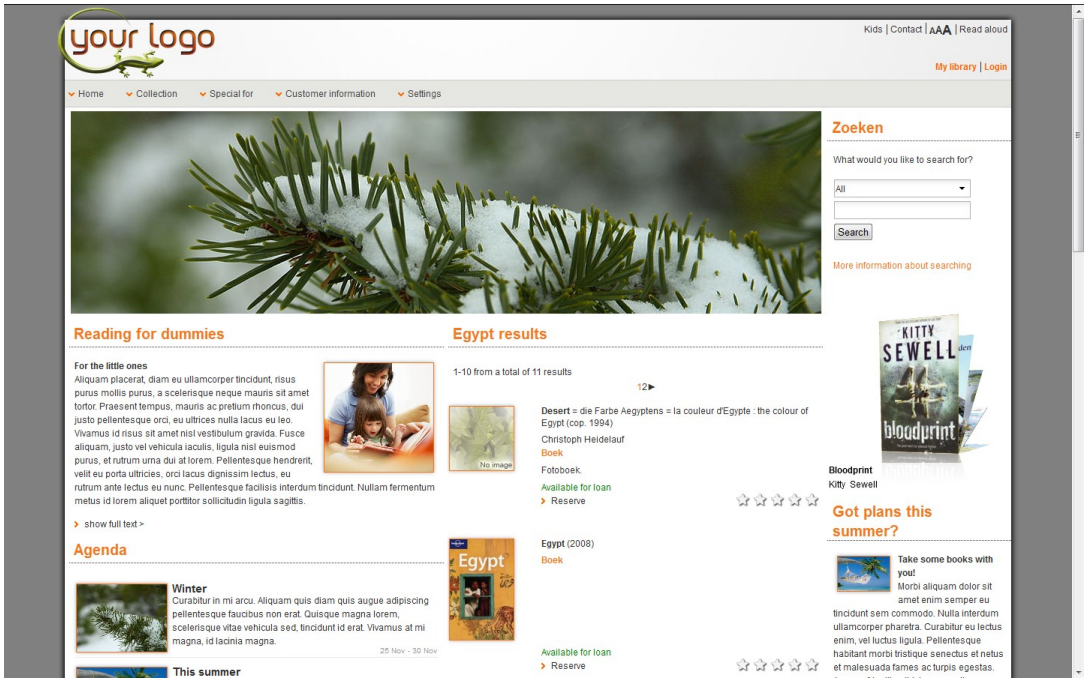
4 A high contrast theme

Iguana also provides a theme called “highcontrast”. This is a CSS file that can be applied to any Iguana site and that primarily uses white and yellow on a black background.

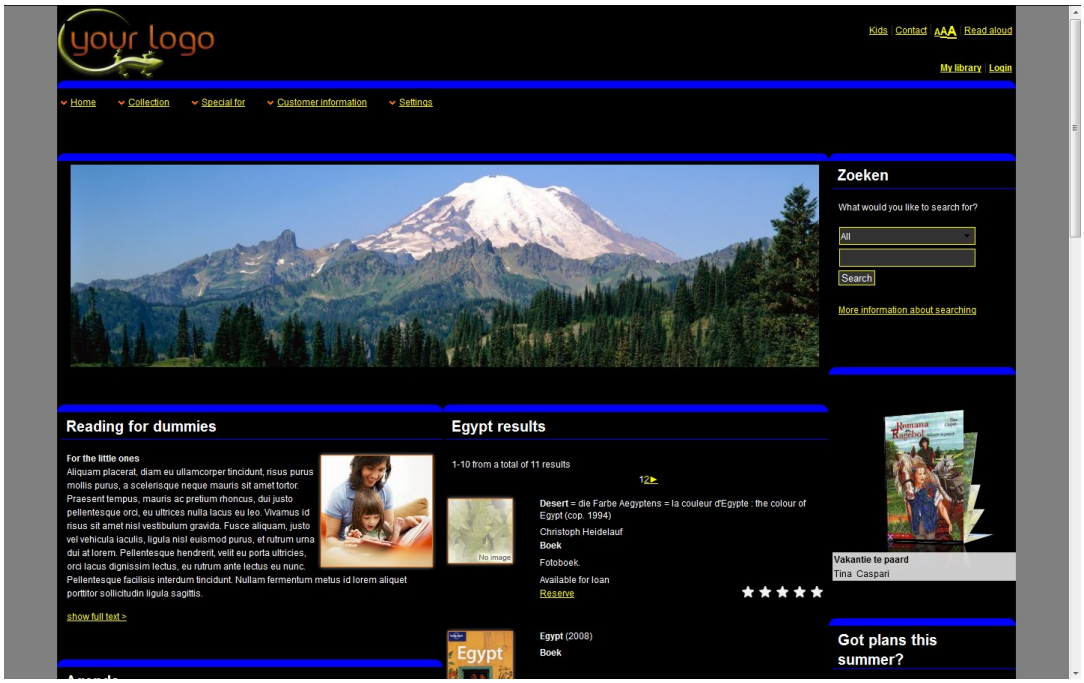
The “highcontrast” theme is based on “base.css” (the default theme), but has its own set of properties.

This high contrast theme can be invoked from any page by adding the URL parameter ?theme=highcontrast to the URL.

The following screen shots show the same screen, first with the default “orange” theme, and then with the “highcontrast” theme.



The “orange” theme



The same page with the “highcontrast” theme

See the document on Structure & Style for more information on CSS files and how to use them.

4.1 Characteristics of the ‘highcontrast’ theme

The ‘highcontrast’ theme has the following characteristics:

- black background
- white texts
- links in yellow
- a selected link (focus/hover) has a red border
- a thick blue border at the top of each widget
- a thin blue line immediately beneath the widget title
- a thin green line between sections in the same widget (e.g. between records in a result set or between personal data and linked accounts in the My profile environment)
- buttons have a yellow border and a black background
- additional space between elements
- moderator login will not cause the page sections to get a grey background, but a thick red border will appear at the top and bottom of each section

- **Document control – Change History**

Version	Date	Change description	Author
1.0	April, 2011	Creation (Introduction only)	
1.1	November 2011	Added section on vi URL parameter and on the highcontrast theme; Rewrite of many sections following the introduction of Search profiles	
1.2	December 2011	Added characteristics of the highcontrast theme	
1.3	January 2012	Added new information on Accessibility features property (replacing vi=1 setting); Added header	
1.4	January 2012	Reviewed Added section on Testing with JAWS and ZoomText	
1.5	February 2012	Added additional information on testing and requirements; added additional information on vi=1 effects	
2.0	May 2012	Reformat for online help doc	
3.0	December 2012	Added warning in intro part of 3.0 updates	