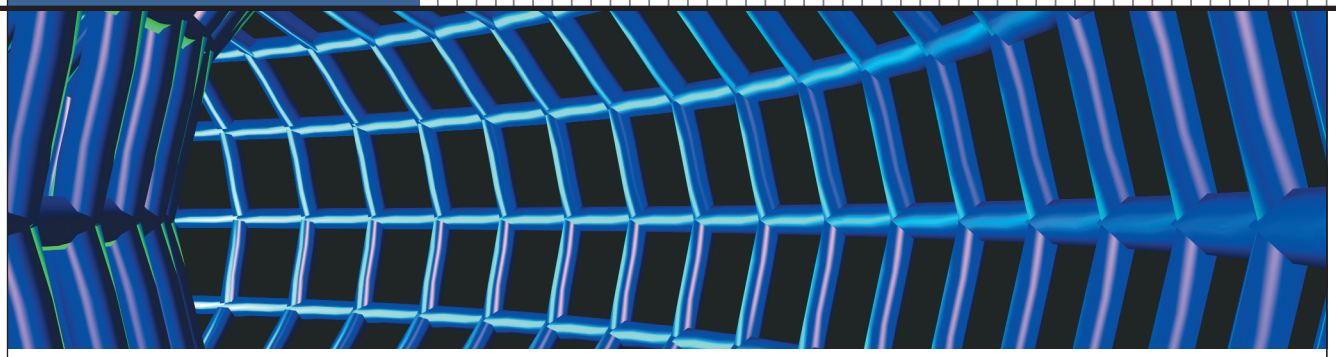

Toward an Equal Opportunity Web: Applications, Standards, and Tools that Increase Accessibility

Lourdes Moreno, Paloma Martínez, Ana Iglesias, and Belén Ruiz-Mezcua
Carlos III University of Madrid

New options offer the promise that those who require assistive technology to access Web content will soon enjoy the full range of interactivity, thereby realizing the equal accessibility goal.

Rich Internet applications (RIAs) and supporting technology, such as JavaScript and Flash, embed components in the user agent or in the browser on the client side, providing more dynamic Web content and more attractive and interactive websites. Through asynchronous client-server communication, RIAs can offer an interactive experience closer to that of a desktop application. However, not all users benefit from this interactivity. Users with disabilities, for example, must access the Web using assistive technologies, such as a screen reader that delivers audio content to the visually impaired. Because RIA technologies are a client-side programming technique, everything loads on that side before the actual interaction. Consequently, the screen reader cannot read the updates, which decreases the user's content accessibility.

Fortunately, with the semantic Web, interface developers should be able to adapt interfaces to meet specific needs, enabling user agents and browsers to understand semantic markup. As the "Road to Web Accessibility" sidebar describes, websites designed in accordance with the Web Content Accessibility Guidelines (WCAG) provide markup semantics that assistive technologies can interpret and successfully relay to the user. Developers can thus use APIs to enable assistive technologies to access dynamic content,[1] adding to techniques that already offer dynamic contextual help through tailored interfaces. More important, interactive technologies might extend learning for disabled users. Users with cognitive and learning disabilities, for example, could experience a wider range of multimodal interaction, since they could tailor the interface to their needs, allowing them to select the most appropriate Web content.

The goal of equal accessibility will not be easy to attain, however. For a fully static webpage, an effective linear presentation is sufficient, but interactive technology involves content updates that preclude a linear approach. The user must be both aware of the update and able to access the new content without disrupting the task in progress. Consequently, assistive technologies must understand the semantics well enough to communicate updates.[2] To assist in promoting accessibility, researchers are exploring development toolkits and frameworks, as well as testing tools to enable faster, easier Web 2.0 application development and to broaden Web accessibility.

 Published by the IEEE Computer Society

ROAD TO WEB ACCESSIBILITY

The myriad perspectives from which to contemplate Web accessibility have yielded a variety of conceptual definitions. According to the Education and Outreach Working Group, web accessibility means that "people with disabilities can use the Web" (www.w3.org/WAI). The International Organization for Standardization's ISO 9241-171 defines it as the "usability of a product, service, environment or facility by people with the widest range of capabilities."

For our purposes, Web accessibility is the development of Web resources that all people can use, regardless of their technical, physical, or cognitive limitations. The number of users affected by website inaccessibility—the digital divide—is growing, largely because users are more frequently encountering Web accessibility obstacles stemming from use context and technological incompatibilities.

Directives, legislation, and standards are attempting to combat this rising inaccessibility, identifying problems and suggesting new, accessible designs. Standards bodies and legislation such as the World Wide Web Consortium (W3C) and Web Accessibility Initiative (WAI) are strong evidence of progress.

The most important WAI component is the Web Content Accessibility Guidelines (WCAG), an official standard in the European Union that is referenced in most legislation worldwide. Other important initiatives include US legislation that requires conformance with Section 508 of the Rehabilitation Act (29 U.S. Code §794d), which describes technical standards related specifically to Web accessibility. Although less extensive, these standards are similar to the WCAG and even map Section 508 to those guidelines.

Over the years, the W3C has published WCAG 1.0 and WCAG 2.0 as W3C recommendations, with the latter becoming official in December 2008. WCAG 2.0 reflects the more interactive nature of Web content and accommodates HTML and Cascading Style Sheets, as well as scripting and non-W3C Web technologies such as RIA.[1]

Although RIA technologies—including JavaScript, Ajax, and Flash—result in more interactive websites, they often impose barriers to accessibility. New specifications can add semantics to a webpage or an application that allow assistive technology to better represent interfaces and interactions to the user. To support this goal, the W3C WAI developed the Accessible Rich Internet Applications (WAI-ARIA) and incorporated ARIA and Flash techniques for WCAG 2.0 (http://www.w3.org/tr/wcag-techs).

The W3C is currently working on HTML 5, aiming to reduce the need for proprietary plug-in-based RIA technologies. HTML 5 will provide Web users and developers with enhanced functionality without using the proprietary technologies that have recently become popular. The new standard, which will replace XHTML standards, should ensure accessibility by fixing the main problems that Web developers are now encountering—such as the need to install plug-ins to access multimedia content and the requirement to separate content and presentation.

HTML 5 will enhance accessibility in several ways: it will add implicit semantic information, be defined on the basis of the document-oriented model, and differentiate content from presentation. HTML 5 will also present better structure than other languages. Div blocks on a webpage will have their own tags—including article, footer, header, and navigation—so that user agents and assistive technologies can recognize them and gather more accurate information. Site authors will be able to embed accessibility multimedia elements natively using <audio> and <video> tags, obviating the need for plug-ins. Although the draft of the new standard that is still under development does not allow the inclusion of HTML 5 elements that provide caption and audio descriptions, it is anticipated that future standards will support adding subtitles and captions using the HTML 5 <video> tag inside webpages.[2] Both user agents and assistive technologies can use this information to enable alternate ways of viewing and navigating a page.

References

1. L.D. Paulson, "W3C Adopts Web-Accessibility Specifications," *Computer*, Feb. 2009, pp. 23-26.
2. S. Pfeiffer and C. Parker, "Accessibility for the HTML5 <video> Element," *Proc. Int'l Cross-Disciplinary Conf. Web Accessibility* (W4A 09), ACM Press, 2009, pp. 98-100.

DEVELOPING ACCESSIBLE RICH APPLICATIONS

Many RIA accessibility challenges arise from the applications' dependency on Ajax (Asynchronous JavaScript and XML) and JavaScript. In addition, RIA toolkits introduce complex user-interface components and dynamically changing content, which is particularly problematic for the keyboard navigation essential to accessibility. In HTML, the focus is only on links and form elements accessible through a keyboard interface; however, in Web 2.0 applications, the focus could also be on span or div elements.

To address accessibility concerns, the W3C Web Accessibility Initiative (WAI) developed specification techniques for RIA technologies. The WAI Accessible Rich Internet Applications (WAI-ARIA) enable keyboard access for all elements by extending the tabindex property applied to any element.

WAI-ARIA provides a navigable page structure or, as Figure 1 shows, an accessible tree widget. The comparison of WCAG 1.0 (without ARIA) and WAI-ARIA semantics shows the severity of Web 2.0 accessibility problems and the inability of some assistive technologies to circumvent these difficulties. However, it also shows the efficiency with which developers can fix these problems by implementing WAI-ARIA recommendations.

The WAI-ARIA suite provides mechanisms to increase Web accessibility, particularly for rich websites and applications with a focus on dynamic content and user interface controls. The addition of semantic data to HTML and XHTML enables assistive technologies to better represent user interface components and dynamic interactions. The suite currently consists of five documents: the WAI-ARIA technical specification, authoring practices, the primer, and the 1.0 user agent implementation guide and roadmap. The documents explain, for example, why JavaScript needs accessibility architecture and relate how the user agent can map such a structure to accessibility frameworks on the native platform.

The *WAI-ARIA technical specification*, a planned W3C Recommendations Web standard that combines the two

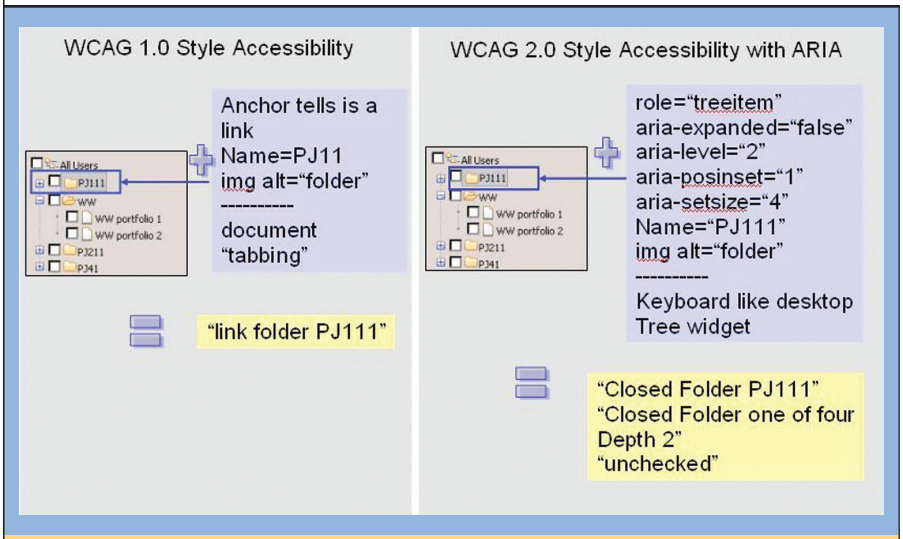

Figure 1. Tree widget using WCAG 1.0 and WCAG 2.0. Within a tree widget, using WCAG 1.0 (left), the screen reader sees only link folder PJ111. Using WCAG 2.0 (right), with WAI-ARIA, the reader sees the information to indicate that the folder is closed (expanded = false) and its depth (aria-level= 2), set position, and set size. Without this information, the user's accessibility to webpage content through the reader is limited. Figure from "WAI-ARIA Primer," W3C Working Draft, 4 Feb. 2008; www.w3.org/TR/2008/WD-wai-aria-primer-20080204.

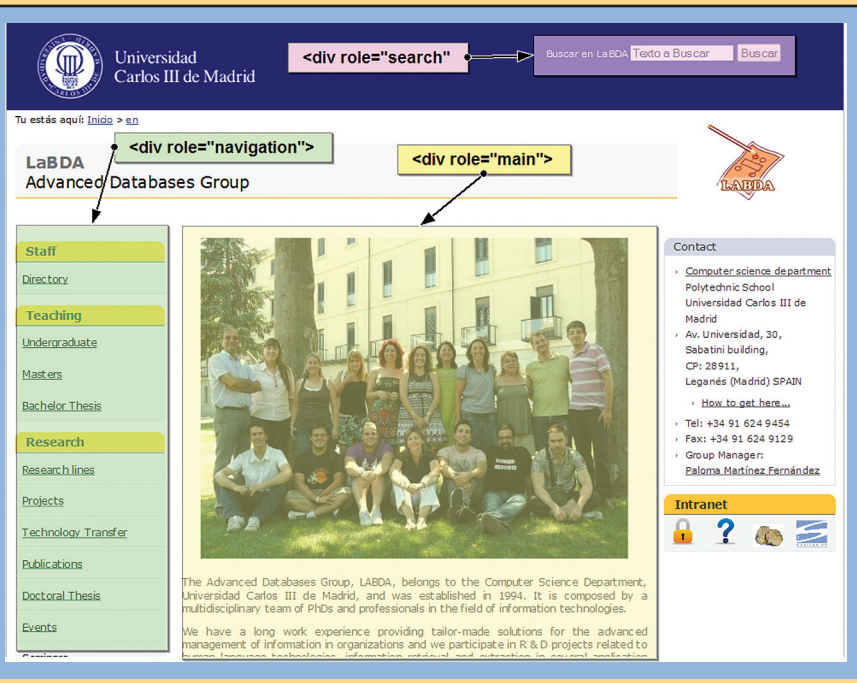

Figure 2. A typical website with regional landmarks indicated. A screen reader aligns keyboard navigation to landmarks, such as navigation, search, and main, allowing the screen reader to provide a visually impaired user accurate information about the page and any updates in the regions the landmarks identify.

previously published WAI-ARIA draft specifications, is intended for developers of Web browsers, assistive technologies, and other user agents. The specification comprises the roles, states, and properties modules. With the WAI-ARIA roles module, the site developer can specify a Web document's functional zones define each component's role, and then use the WAI-ARIA states and properties module to determine each component's characteristics and values.

Applying these modules adds semantics to webpage components and widgets so that assistive technologies can interpret their operation. The result is a guarantee of accessible keyboard navigation, the accessibility of controls and widgets, and accessibility through dynamic content-update notifications.

Figure 2 shows a website with regional landmark roles, which provide a navigable structure within a webpage that a screen reader can access.

Figure 3 shows the HTML code for the regional landmark roles in Figure 2.

Other WAI-ARIA roles, such as those that define page structure, are also useful. The *WAI-ARIA authoring practices* offer guidance on incorporating these additional roles. This document also targets Web developers and provides detailed advice and examples of how to build accessible RIAs.

The *WAI-ARIA primer*, a planned W3C Working Group Note, introduces developers to accessibility-related problems that WAI-ARIA is intended to resolve, as well as to WAI-ARIA's fundamental concepts and technical approach.

The *WAI-ARIA user agent implementation guide*, another planned W3C Working Group Note, describes how browsers and other user agents should support WAI-ARIA. It includes such details as how to expose WAI-ARIA features to platform accessibility APIs.

Finally, the *WAI-ARIA roadmap* defines the path for making rich Internet content more accessible, including steps already taken, future steps, and a timeline. It describes the technologies to map controls, Ajax live regions, and events to accessibility APIs, including cus-

tomized controls for RIAs. It also describes techniques to mark common Web structures, among them menus, primary content, and secondary content.

To understand JavaScript's power, consider the document object model (DOM) node in Figure 4, which is part of a model-view-controller architecture. Without JavaScript, assistive technologies acquire accessibility information only through the HTML element's tag name—and then only the accessibility attributes that the tag can provide. Because the data node (the model) is separate from the user interface node (the view), the user agent manages the document element according to the element's default behavior, and the user agent's default behavior at that element acts as the controller.

With JavaScript, the default user agent's behavior no longer serves as the controller. JavaScript overrides the default user agent behavior at the DOM node, manipulating data, content, and style in response to user interaction events. The result is custom widgets. In this scenario, default accessibility information is no longer valid, so the contract is also invalid. The asterisks in front of the blue text (role, state, actions, and so on) represent potential accessibility errors, as well as gaps in the base markup that result from the author's inability to provide the new semantic data to support the contract.

Both browsers and assistive technologies are providing WAI-ARIA support. All four major browsers have either implemented support or plan to do so. Opera 9.5 and Firefox 1.5+ already include support for ARIA, as does Internet Explorer 9 beta. WebKit, the open source application framework behind Safari, has begun to add support for ARIA.

Assistive technologies that are starting to widely support ARIA include JAWS 7.1+, Window-Eyes 5.5+, NVDA, and Zoomtext 9+. The trends we've observed indicate that this support is likely to increase.

```
<html>
<body>
      ...
      <div role="search">
      The search area
      </div>

      <div role="navigation">
      The navigation area
      </div>

      <div role="main">
      The main content area
      </div>
      ...
</body>
</html>
```

Figure 3. HTML code with regional landmark roles. The roles make navigation easier for a user who must rely on a screen reader or similar assistive technology.

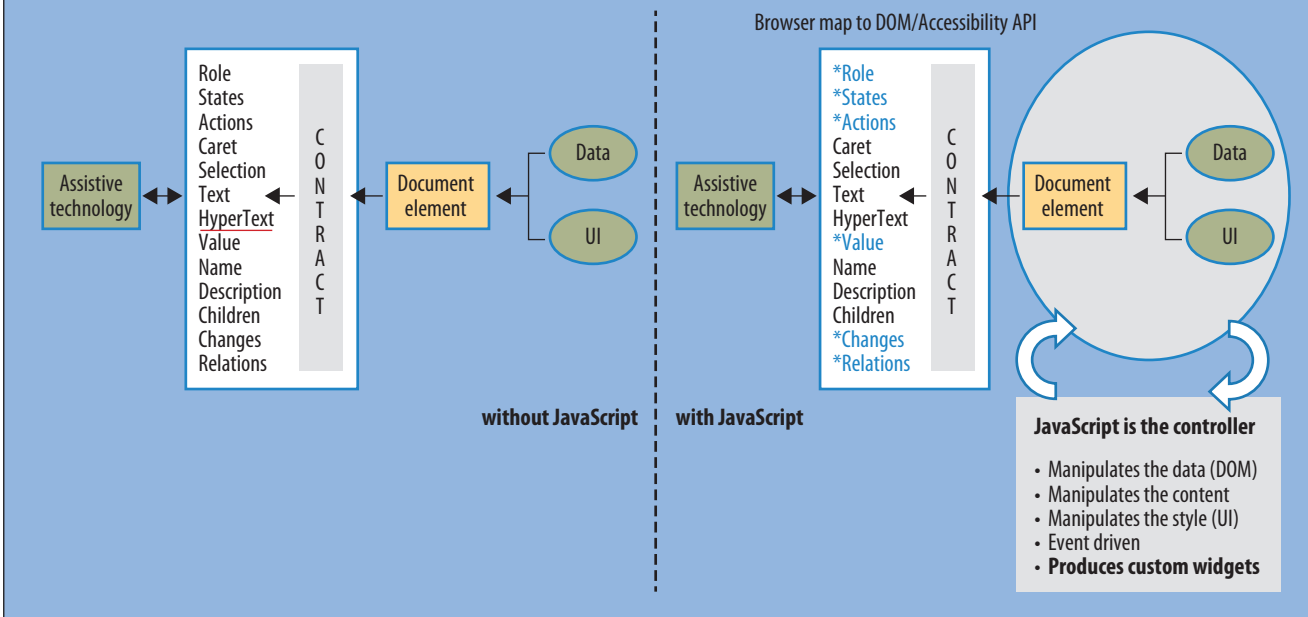

Figure 4. Accessibility interoperability at a DOM node without and with JavaScript. Without JavaScript (left), the data node (Data), which should include semantic information, is separated from the user interface (UI) presentation. Default user agent behavior forms the controller. With JavaScript (right), the same DOM node now has JavaScript as the controller, which overrides the default user agent behavior at the node. Figure from "WAI-ARIA Primer," W3C Working Draft, 4 Feb. 2008; www.w3.org/TR/2008/WD-wai-aria-primer-20080204.

Table 1. Summary of non-ARIA technology for accessible websites.

Technology (base)	License type	Integrated development environment	Compatible browsers	WCAG/ Section 508 support	Accessibility support	Accessibility statement
Flex 3.0 (Flash)	Open source	Adobe Flex Builder	Any browser with Flash player	Section 508 with exceptions	Flash player integration support for MSAA	Section 508 support
Silverlight 4.0 (Windows)	Proprietary software; free license	Visual Studio 2008, Expression Web	IE 6+, Firefox 1.5+, Opera 2.0+, Chrome	Section 508 with exceptions	User interface automation API	Not provided
Java FX 1.3 (Java)	Partially open source	NetBeans for JavaFX	Any browser with JRE and mobiles with JME	Section 508 with exceptions	No accessibility API; not compatible with Java accessibility API	No support

Table 2. Summary of ARIA-based technology for accessible websites.

Technology	License type	Integrated development environment	Compatible browsers	WCAG/ Section 508 support	WAI-ARIA support	Accessibility statement
Dojo 1.4.2	Open source	Aptana or Komodo	IE 6+, Firefox 1.5+, Safari 3.1+, Opera 9.6+, Konkeror 3.5+, older browsers with Dijit	Both	With Dijit (forthcoming)	WCAG/ Section 508
Bindows	Commercial software	Bindows	IE 5.5+, Firefox 1.4+, Netscape 7.1+, K-Meleon, Camino, WebKit 525+	Section 508	Limited (forthcoming)	WCAG/ Section 508
Google Web Toolkit 1.7	Open source	Eclipse and other Java IDEs	IE 6+, any browser with JRE and mobiles with JME	Not specified	Limited (forthcoming)	Not provided
ExtJS 3.2.1	Commercial and open source	Ext Designer	IE 6+, Firefox 1.5+, Safari 3+, Opera 9+	Section 508 extended (forthcoming)	Limited (forthcoming)	Not provided
Fluid Infusion 1.2	Open source	Any JavaScript IDE	Firefox 3+, IE 6+, Opera 9.6+, Safari 3.2+	Not specified	Limited (forthcoming)	Not provided
jQuery User Interface 1.8.1	Open source	Any JavaScript IDE	Firefox 2+, IE 6+, Safari 2.0.2+, Opera 9+	Section 508 extended (forthcoming)	Limited (forthcoming)	Not provided
Yahoo User Interface Library 3.1.1	Open source	Any JavaScript IDE	Firefox 3+, Opera 9.6+, IE 6+, Safari 3.2+	Not specified	Limited (forthcoming)	Not provided
BBC Glow Widgets 1.7.3	Open source	Any JavaScript IDE	IE 6+, Firefox 2+, Chrome 1+, Opera 9+, Safari 2+	Not specified	Limited (forthcoming)	Not provided

ACCESSIBILITY IN WEB PLATFORMS

At present, the Web uses two kinds of platforms.[3] *Non-Ajax platforms* are environments that depend on components embedded in the HTML document, and operation requires additional browser software or a plug-in. *Ajax platforms* are RIA development frameworks based on Ajax technologies. The platforms feature JavaScript implementation and are completely independent of the operating system, since only a browser with Ajax support is necessary.

Tables 1 and 2 summarize the differences between the two platforms, drawing from information we extracted from the websites detailing the various technologies.

Non-Ajax platforms

Flash is the most common RIA technology in the non-Ajax category, with websites being entirely Flash-based or using components, such as calendars, news feeds, and media players.

Sites that are entirely Flash-based present a challenge to accessibility. Because they do not use standards like HTML or Cascading Style Sheets (CSS), responsibility for accessibility falls to the browser and built-in technologies like Microsoft Active Accessibility and accessibility APIs. Problems arise when Web user agents cannot access Flash websites. Developers could offer alternative versions of these sites that any user agent can access, but the alternative sites would have minimal functionality and generally no updates.

Flash components are small programs that any developer can create, and their unknown origin can often mean the lack of an accessibility guarantee before the site author adds the component. If the Flash component does prove inaccessible, the entire site's accessibility suffers. Even if the visible layer is not HTML, the component developer must satisfy all WCAG and Section 508 accessibility requirements.

As a first step in installing these components, the site author should carefully validate the accessibility level and look at possible alternatives if the component doesn't meet requirements. Authors often add Flash animations to their sites, for example, because the animation has a relatively light format and is easy to include, thanks to Flash players.

As is true of other HTML elements on the page, the site author should follow accessibility guidelines, such as Guideline 1.2 of WCAG 2.0 and should offer alternative content to audio tracks (such as captioning), Flash techniques for WCAG 2.0, and other elements where applicable.[4] The WCAG 2.0 suggests adding a group name to the accessible name of a form control, such as a radio button. Figure 5 shows an example of how an author can do this. WCAG 2.0 further states that the suggested Flash technique is applicable for Adobe Flash Professional MX and Adobe Flex.

Technical and best practices documents already recommend making Flash more accessible—in essence by offering alternative content to every Flash object—and these criteria are quite similar to WCAG requirements. Another way to guarantee website accessibility is to provide a logical organization of content that ensures keyboard access and correct tabulation with technical aids.[5]

Silverlight, Microsoft's alternative to Flash, is a plug-in for free Web browsers that adds certain functionalities such as video, vector graphics, and animation display. Like Flash, Silverlight is not an open standard, but multiple browsers and platforms support it, and its functionalities are similar to those of Flash. Although Silverlight is free, it is not open code; however, Microsoft has published parts of the code under a permissive license.

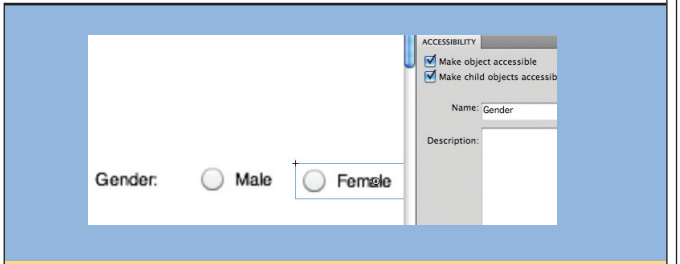

Figure 5. Using the technique suggested in WCAG 2.0, Flash concatenates the group name with each button's individual name to produce "Gender:Female." Figure from Flash Techniques for WCAG 2.0; www.w3.org/WAI/GL/WCAG20-TECHS/Flash.html.

Silverlight presents alternatives for each of its objects and passes any component accessibility concerns to the browser through the platform's accessibility APIs. Because Silverlight uses the Extensible Application Markup Language (XAML) to define the user interface, it can dynamically upload and manipulate XML code with the DOM. Using XAML technology, the developer can separate presentation, content, and events and include metadata with each object at the XHTML code level, thus enabling assistive technologies to interpret content.

Flex Framework applies the same model using component library code to facilitate the development of Flash-based RIAs. Instead of XAML, Flex Framework uses the Multimedia Extensible Markup Language (MXML) to define user interfaces, communicate with application servers, and execute other operations. As with XAML, MXML lets authors separate presentation, content, and events. It also has APIs defined to incorporate properties and accessible metadata in objects, allowing assistive technologies to interpret content.[6]

Flex Framework's principal advantages for accessibility are its compatibility with assistive technologies, such as screen readers, as well as its wide support for Section 508 requirements. According to Adobe, Flex best practices are nearly the same as those for Flash, although Adobe has adapted them to suit MXML and included new characteristics, such as specific recommendations for each accessible Flex component.

JavaFX is a non-Ajax platform that offers very limited accessibility support and is incompatible with Swing's Java Accessibility API. Some independent implementations, such as fxaccessible, use a Swing component to add support for the Java Accessibility API, which in turn allows screen readers to work with the JavaFX-based application.

AJAX PLATFORMS

Ajax-based technologies include but are not limited to asynchronous JavaScript and XML. Their advantages include high upload speed and usability. Google Maps and Gmail are among the numerous Web 2.0 applications that rely on these technologies. On the client side, JavaScript provides full functionality by interacting with the Ajax engine.

TECHNOLOGIES TO ENHANCE ACCESSIBILITY

The following is a list of interface libraries and evaluation and development tools used to promote the development of more-accessible websites. Most of these tools are open source.

- Accessibility Evaluation Toolbar: https://addons.mozilla.org/es-es/firefox/addon/5809
- Adobe Accessibility Best Practices for Flex: www.adobe.com/accessibility/products/flex/best_practices.html
- Adobe Flash Platform: www.adobe.com/flashplatform
- Adobe Flex 3: www.adobe.com/es/products/flex
- ASP.NET Ajax Roadmap: www.asp.net/ajax/documentation/live
- BBC Glow widgets: www.bbc.co.uk/glow/docs/1.5/api/glow.widgets.shtml
- Dojo Toolkit: www.dojotoolkit.org
- Ext JS Cross-Browser Rich Internet Application Framework: www.extjs.com/products/extjs
- Flash Techniques for WCAG 2.0: www.w3.org/tr/wcag20-techs/Flash.html
- Fluid Infusion: http://fluidproject.org/products/infusion
- Firefox Accessibility Extension: http://firefox.cita.uiuc.edu
- Firefox Juicy Studio Accessibility Toolbar: https://addons.mozilla.org/en-US/firefox/addon/juicy-studio-accessibility-tool
- Fxaccessible Accessibility for JavaFX: http://code.google.com/p/fxaccessible
- Google Web Toolkit: http://code.google.com/intl/en/webtoolkit
- jQuery UI: http://jqueryui.com
- MB Technologies, Bindows: www.bindows.net
- Microsoft Silverlight: http://silverlight.net
- The Paciello Group Blog: WAI-ARIA Implementation in JavaScript UI Libraries–updated, www.paciellogroup.com/blog/?p=313
- Web Accessibility Toolbar: www.visionaustralia.org.au/info.aspx?page=614
- Yahoo User Interface Library widgets: http://developer.yahoo.com/yui

Functionalities are grouped into libraries that a JavaScript engine in the user agent or the client browser runs.

Ajax controls HTML documents as well as their updates in a way that is transparent to the user, which is not advantageous for accessibility, since the user might not notice these updates. Ajax operations include, among others, the dynamic modification of Web content with DOM and event capture. Ajax carries out interactions asynchronously using the XMLHttpRequest object for data exchange with the webserver without the need to reload the page. For the presentation layer, Ajax uses W3C standards (HTML, CSS, and XHTML) as well as other embedded components, such as Flash and multimedia content.

Most popular platforms integrate Ajax and, although not all Web user agents implement the XMLHttpRequest object in the same way, most of the better-known user agents generally agree.

Whether they are frameworks, toolkits, or libraries, numerous Ajax-based solutions are available. Two solutions, Dojo and Bindows, deserve special consideration for their support of WAI-ARIA and tagging that permits access for assistive technologies. Additionally, both solutions include components that offer keyboard support, and both provide an accessibility statement. For these reasons, Dojo and Bindows have received special recognition from the W3C for their compliance with the WAI-ARIA specification.

Dojo is an open source toolkit for developing accessible Web applications, and it has the official support of companies like IBM and Sun. Dojo's core widget set, Dijit, has included accessibility support since its 1.0 release, and Dojo claims to be the only fully accessible open source toolkit available for Web 2.0. Because Dijit uses the WAI-ARIA specification, all Dijit widgets are accessible using the latest versions of the JAWS or Window-Eyes screen readers with Firefox 3.

In addition to its WAI-ARIA support, Dojo is compatible with the latest versions of Web browsers and screen readers. It offers the user complete keyboard access for elements, including tabulation, keyboard events, and focus assignments. Dojo also supports visually impaired users by guaranteeing assistive technologies proper access. Its inclusion of alternative themes and texts for multimedia content ensures that assistive technology will be able to detect any content changes and updates and accommodate changes in font sizes—features that enable visually impaired users to adapt Web content to suit their specific needs.[7]

The Bindows Ajax-based solution is the first framework to be officially recognized for its Section 508 compliance. Unlike Dojo, Bindows is not an open source framework, but it does offer many of Dojo's advantages, such as complete keyboard navigation support, the inclusion of alternative texts, and support for notifying users of dynamic content changes. Bindows improves framework accessibility by supporting assistive technologies, such as screen readers and magnifiers, and providing reference material and a module to bring legacy applications up to Section 508 standards.

Table 2 also lists several popular JavaScript user-interface libraries that are adding WAI-ARIA support to their widgets and components; however, their conformance to Section 508 or WCAG guidelines has not yet been documented. The "Technologies to Enhance Accessibility" sidebar provides links for these libraries, most of which are open source:

- *Google Web Toolkit*. This toolkit partially supports WAI-ARIA and fully supports keyboard navigation,

including tabulation, keyboard events, and focus assignments.

- *Ext JS*. Although details are lacking, the literature reports real accessibility improvements.[8] Version 3.0 supports both Section 508 and WAI-ARIA.
- *Fluid Infusion*. This new JavaScript application framework built on top of jQuery lets both developers and users customize user interfaces. Although it does not provide any documentation about Section 508 or WCAG conformance, its developer states that assistive technology, such as screen readers, has been tested.
- *Yahoo User Interface Library widgets*. WAI-ARIA plugins exist for many of these widgets. Although Yahoo does not provide documentation about Section 508 or WCAG conformance, it states that most of these widgets would withstand a rigorous Section 508 testing. Yahoo plans to implement more WAI-ARIA support in the upcoming version 3.0.
- *BBC Glow widgets and ASP.Net Ajax Roadmap*. Both the widgets and roadmap have supported WAI-ARIA since mid-2009.

Other technological solutions, such as the combination of Google AxsJAX and the SADIe transcoder, improve accessibility by dynamically injecting WAI-ARIA statements into Web content.[9] AxsJAX is an open source framework that lets developers use high-level patterns developed from the underlying WAI-ARIA markup. The SADIe transcoder uses CSS annotations to generate AxsJAX framework code and insert it into webpages. This approach can, for example, improve page access for visually impaired users who use a screen reader. Similarly, the Hearsay-Dynamo browser[10] lets users with disabilities experience RIA content, even content that the author has annotated according to WAI-ARIA specifications.

ACCESSIBILITY EVALUATION TOOLS

Currently, tools that can evaluate how well a webpage conforms to WAI-ARIA requirements are either still in trials or permit only a limited inspection. Consequently, a complete accessibility evaluation of pages with and without WAI-ARIA support requires manually validating specifications and good practices. Even so, it is worth investigating some of the automatic tools that provide WAI-ARIA testing features, such as the Web Accessibility Toolbar, Firefox Accessibility Extension, and Firefox Juicy Studio Accessibility Toolbar. The "Technologies to Enhance Accessibility" sidebar provides links to these tools.

Although RIA technologies make websites more attractive for many users, they can also raise new accessibility barriers for users with special needs. Because problems arise for keyboard navigation when these applications automatically change pages or content, users who access the Web through assistive technology can feel lost while surfing the Web. To ensure Web accessibility, the Web community has developed new techniques such as WAI-ARIA.

At present, only a handful of solutions are oriented toward guided accessible Web development using RIA technologies. Rather than offer full guided support, existing solutions generally provide only the resources necessary for accessible development. Therefore, the degree of accessibility is largely in the hands of the developer.

The immediate challenge is to build Web 2.0 development frameworks and technologies with a greater degree of WAI-ARIA support so that developers do not have to be experts in what constitutes accessibility. Work must continue in creating tools to support the authoring process in Web applications development. By making it easier to add accessibility, there is a greater opportunity for all users to benefit from the depth and breadth of Web content.

Acknowledgments

This research work is supported by the Research Network MAVIR (S2009/TIC-1542) (www.mavir.net), project TIN2007-67407-C03-01, and the Spanish Center of Captioning and Audio Description (www.cesya.com).

References

1. B. Gibson, "Enabling an Accessible Web 2.0," *Proc. Int'l Cross-Disciplinary Conf. Web Accessibility* (W4A 07), ACM Press, 2007, pp. 1-6.
2. M. Cooper, "Accessibility of Emerging Rich Web Technologies: Web 2.0 and the Semantic Web," *Proc. Int'l Cross-Disciplinary Conf. Web Accessibility* (W4A 07), ACM Press, 2007, pp. 93-98.
3. G. Lawton, "New Ways to Build Rich Internet Applications," *Computer*, Aug. 2008, pp. 10-12.
4. L. Moreno, P. Martínez, and B. Ruiz, "Disability Standards for Multimedia on the Web," *IEEE MultiMedia*, vol. 15, no. 4, 2008, pp. 52-54.
5. B. Regan, "Best Practices for Accessible Flash Design," Macromedia white paper, 2005; www.adobe.com/resources/accessibility/best_practices/best_practices_acc_flash.pdf.
6. R. Tretola, S. Barber, and E. Renaun, *Professional Adobe Flex 2*, John Wiley & Sons, 2007.
7. B. Gibson, "Dojo: An Accessible JavaScript Toolkit," IBM Human Ability and Accessibility Center; www-03.ibm.com/able/resources/dojo.html.
8. Z. Mikovec, J. Vystrcil, and P. Slavik, "Web Toolkits Accessibility Study," *SIGACCESS Accessibility Computing*, June 2009, pp. 3-8.
9. D. Lunn, S. Harper, and S. Bechhofer, "Combining SADIe and AxsJAX to Improve the Accessibility of Web Content," *Proc. Int'l Cross-Disciplinary Conf. Web Accessibility* (W4A 09), ACM Press, 2009, pp. 75-78.

10. Y. Borodin et al., "What's New?: Making Web Page Updates Accessible," *Proc. 10th Int'l ACM SIGACCESS Conf. Computers and Accessibility* (Assets 08), ACM Press, 2008, pp. 145-152.

Lourdes Moreno *is an assistant professor in the Advanced Databases Group in the Computer Science Department at Carlos III University of Madrid and a member of the Spanish Center of Caption and Audio Description. Her research interests include Web accessibility, development of accessible Web applications, and inclusive education. She is a member of the Spanish Association of Human-Computer Interaction and the Mavir consortium. Moreno received a PhD in computer science from Carlos III University of Madrid. Contact her at lmoreno@inf.uc3m.es.*

Paloma Martínez *is a member of the Advanced Databases Group in the Computer Science Department at Carlos III University of Madrid, where she teaches database design and management. Her research interests include Web accessibility and human language technologies, such as multilingual information extraction and retrieval in several domains, question answering, name-entity recognition, and temporal information management. She is a member of the Spanish Society for Natural Language Processing and the Mavir consortium. Martinez received a PhD in computer science from the Polytechnic University of Madrid. Contact her at pmf@inf.uc3m.es.*

Ana Iglesias *is a faculty member in the Computer Science Department at Carlos III University of Madrid. Her research interests include inclusive education, assistive technologies, adaptive intelligent educational systems, natural-language processing and information retrieval, and advanced database technologies. She is a member of the Mavir consortium. Iglesias received a PhD in computer science from Carlos III University of Madrid. Contact her at aiglesia@inf.uc3m.es.*

Belén Ruiz-Mezcua *is an adjunct vice chancellor of research and a lecturer in the Department of Computer Science at Carlos III University of Madrid, where she is the head of the Spanish Center of Caption and Audio Description. Her research interests include speech and speaker recognition, Web accessibility, and accessibility in products, services, applications, and technologies applied to persons with disabilities. Ruiz-Mezcua received a PhD in physics from the Telecommunications School at the Polytechnic University of Madrid. Contact her at bruiz@inf.uc3m.es.*