Accessibility and Rich Internet Applications

Lisa Seeman[1]

Abstract. La presente investigación se realizó con el fin de identificar, definir y contextualizar los elementos y mecanismos de las competencias interpretativas que se desarrollan alrededor de los ambientes virtuales de aprendizaje (AVA) y, que influyen en la formación y estructuración de los estudiantes de pregrado en la carrera de Ingeniería de Sistemas de la Universidad INCCA de Colombia (Integración, Nivelización, Comunicabilidad, Comunicación, Aceleración).

1 What is Accessibility?

Making your content accessible means accommodating visitors to your web site who will not be viewing it on a monitor and navigating with a mouse. Some users may be using your content via a screen reader which provides speech synthesis for audio output, or a refreshable Braille for tactile output. Others will be using a Braille reader. Some people may be using a specialized input device or keyboard. Some are color-impaired. Typically people with disability rely on assistive technologies that help them accomplish tasks that they cannot perform easily or otherwise.

The Web Accessibility Initiative (WAI) is an initiative by W3 to make the Internet usable for people with disabilities. Most of their guidelines are easy to implement, without detracting from the general style and look of your page.

Web Content Accessibility Guidelines (WCAG): WCAG is a specification that explain how to make Web content accessible to people with disabilities.

The goal of the WAI and WCAG is to permit the maximum amount of people to enter and use each Web site. Web accessibility issues for the disabled community may include one or more of the following:

- Inability to see information
- Inability to hear information
- Inability to read, understand or process, information
- Inability to use a mouse due to mobility limitations.

However access can also be limited by users scenarios such as: The use of Mobile devices and different operating systems

1.1 Semantic as a Requirement for Accessible Web Content

Accessible content and applications will expose to any device, assistive technology or user what the content is and what it can do and what it is doing now. This enables the different user agents to adapt the content for any user. Web Accessibility is about, at least in part, encapsulating and capture of information in a page, that can then be interpreted to create better accessibility. Another way of looking at this is to say Web accessibility is requires putting semantics into Web content.

Examples of information that is required for accessible interoperability and device independence include: What each element is (for example, a checkbox, a form, a paragraph or a link); what each element is doing (such as a checkbox can be checked, or an option selected); and what are the relationships between content elements (for example, a caption may refer to a specific table).

1.2 Problems with DHTML

As explained above, accessibility requires that all the necessary semantics are available in the content such that assistive technology can understand and process the semantics of the content to adapt it for the user. However as content becomes more and more complex, the standard HTML tags and attributes become inadequate in providing semantic reliably. Web application designers will not limit themselves to using components and elements that are well defined in HTML. Modern Web applications often apply scripts to elements such as an HTML DIVs and use scripts and style sheets to enable them to act as a control or other dynamic component.

Scripts often are used to convert HTML elements into custom components. However these custom components or widgets do not provide a way to convey semantic information to the user agent. For example an HTML DIV tag provide no semantic information and the user agent can not know that the role of a DIV may be that if a pop-up menu. Assistive technology can not know what actions can be theoretically performed this element as all semantics are missing. Clearly the user agent also has no information about the current state of the pop-up menu, such as if it is collapsed or expanded.

In another example a tristate checkbox may be scripted by toggling an image of the different states of the checkbox. The assistive technology and accessibility API of the operating system have no mechanism to know that the image is of a checkbox or that it is currently checked. As such, it is impossible to convey this state to the disabled user who can not see the image. Further, as an image does not usually enable focus or tabbing, such custom components that are not keyboard accessible. They also can not convey to the assistive technology when the elements has focus.

To summarize, the problems with DHTML include a lack of knowledge or semantics about:

- The roles of elements and what they are.
- What scripted elements are currently doing (the state of the element).
- What the relationships are between different elements
- Focus is inadequately supported

2 The ARIA Solution

ARIA (Accessible Rich Internet Applications) is a specification that enables accessibility by:

- Enabling each element to be labeled with a role and provide all the semantics to fully describe its supported behavior.
- Proving the semantics so that each element can expose its current states and properties.
- Proving the semantics so that each element can expose its relationships between other elements and groups.
- Enabling elements to support the correct input focus.

2.1 ARIA Roles

Roles are used to build accessible applications by providing any missing information about what each element is and any other information that the assistive technology needs to anticipate the behavior of the elements inside the Web application.

Roles (unlike states and properties) are used to provide element types and semantics that do not change with time.

Often, to support accessibility the browser may create accessible objects of a web page via an Accessibility API that is part of the operating system. When the full semantics are provided and understood by the browser, it can correctly map the page or internet application to the accessible objects of the operating system.

2.2 ARIA States and Properties

States and Properties can be used to build accessible applications by providing information of what each element is doing at any given time. This includes any missing states, properties, and relationships information that the assistive technology may need to follow the changeable behavior of the elements inside the application. This further includes any missing properties and relationships information needed to understand the element in a given context.

In contrast to Roles, States and Properties can be used for assigning meaningful semantics that change with time and events. States and Properties often controlled by the user.

3 Building accessible applications

A simplified summary of the steps involves in using ARIA to make applications and DHTML accessible include:

Step 1: Use your native mark up as correctly as possible. Assistive technology work robustly with HTML tags and attributes. When these controls are appropriate for an application control or element it is strongly recommended to use them.

Step 2: Find the right roles: When elements behaviors are not fully described by the native markup language, set the roles attribute to correctly describe the supported behavior and function of each element within your application. Roles should ensure that every element behave predictably.

Step 3: Look for groups: Look for groups within a page or application, and mark them using the most appropriate role that best describes their usage. A group can include a group of form elements, a live AJAX region, a Menu bar etc.

Step 4: Build relationships: For example: If one region contains search results, which are controlled by a second region containing a search form, then this region can expose the relationship by setting the aria property controls to point to the search results.

Further examples of ARIA supported relationships include: Describedby, labeledby, Hierarchical relationships, Group structure such as in trees, and regions. In many instances relationships are an essential part of what the element is.

Sometimes the relationships are already made clear via the native mark up language. Alternatively relationships can sometimes be clearly implied via the position and the order of the content within the source code or the DOM (Document Object Model). It is only when this is inadequate to describe all the relationships, that the application should use the ARIA States and Properties.

Step 5: Set properties until the behavior of the element is defined and described (For example: If the user is required to fill in a form element one can use the aria property required, to convay this).

Step 6 Control the behavior of the element using device independent events, states, and properties. For example, a script could toggle the ARIA state based on a device independent event.

Events such as mouseover are dependent on the use of a mouse and are therefore not device independent. The keyup event on the other hand, is device independent.

Step 7: Set Focus. Using the extending ARIA tabindex property, the author is able to give any element keyboard focus (and not just form elements or anchors). The author can ensure the correct tabbing order, so that tabbing can be used to skim the page and reach the section of interest to the user.

3.1 A Simple Example

The following example shows the use of ARIA (via the ARIA DOCTYPE) to provide role information (menubar, menuitem, menu), and the use of ARIA properties to define relationships (aria-labelledby) and further details about the element (aria-haspopup).

```
<?xml version="1.1" encoding="us-ASCII"?>
<!DOCTYPE html PUBLIC "Accessible Adaptive Applications//EN"
  http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<body>
<ul role="menubar">

 <li role="menuitem" aria-haspopup="true" aria-labelledby="myFileLabel">
<span id="myFileLabel">File</span>
<ul role="menu">
        <li role="menuitem" aria-haspopup="false">New</li>
        <li role="menuitem" aria-haspopup="false">Open…</li>
        …
        </ul>
 </li>
…
</ul>
</body> </html>
```

4 ARIA and Accessibility API of the Operating System

To support accessibility, the browser may create accessible objects of a web page though the Accessibility API of the operating system. With ARIA, the full semantics are provided and understood by the browser, so that the browser can properly map the content to accessible objects of the operating system.
Roles are mapped to the right MSAA object (such as *ROLE_MENUITEM* or *ROLE_LINK*) and to right states (*STATE_FOCUSABLE, STATE_CHECKABLE*).
Scripts activate on toggled states that AT can understand and respond to. It exposes the state information. AT can know what is going on and can tell user. For example, the ARIA properties that map to *EVENT_STATE_CHANGE* are: Checked , expanded, readonly, disabled, required and invalid.