MOBILE ACCESSIBLE RICH INTERNET WEB APPLICATION ENHANCED WITH AMP PUBLISHING TECHNOLOGY

George Alex STELEA[*], Vlad FERNOAGA[*], Cristinel GAVRILA[*], Vlad POPESCU[*], Maurizio MURRONI[**]

[*]Transilvania University, Brașov, Romania (george.stelea@unitbv.ro, vlad.fernoaga@unitbv.ro, cristinel.gavrila@unitbv.ro, vlad.popescu@unitbv.ro)
[**]University of Cagliari, Italy (murroni@diee.unica.it)

DOI: 10.19062/1842-9238.2019.17.1.9

Abstract: *Web applications are becoming more and more complex in an continuously growing Internet and Intranet networks. Nowadays accessibility should not be considered as a barrier to innovation and the possibility of developing optimized accessible solutions in order to favor the reduction or elimination of the gap between those who can independently access web resources and those who can not (in particular people with visual impairment) is a requirement that is indispensable for the modern society. This paper presents a concept and proposes a solution to develop a mobile accessible rich internet web application, presenting its architecture, development environment, as well as the adaptive and responsive capabilities using the AMP (Accelerated Mobile Pages) publishing technology and its advantages.*

Keywords: *accessible rich internet application, accelerated mobile pages, cross-platform development, semantic web, web accessibility, user interface plasticity.*

1. INTRODUCTION

The Internet and Intranets are growing continuously, the web applications are becoming more and more complex, and the semantic web [1] is an ongoing step in the evolution on modern web architectures [2]. Developers and designers often use new self-built controls in applications that can not be represented using the traditional markup language tools. These include drop-down menus, tabs, hierarchical tree structures, sliders, fields that allow an input and at the same time dynamically offer input suggestions in a drop-down menu etc., and all used together are building the modern Rich Internet Applications [3]. These user-side controls and dynamic content updates can generate an increased code density and excessive requests that can slow down the application and decrease the Quality of Service (QoS) and Quality of Experience (QoE) [4], especially when accessed via a mobile device. In addition the Rich Internet Application [5] are often not accessible to users with disabilities, especially for those who use screen readers and users who can not use the mouse or other pointing devices.

Accessibility is the characteristic of a device, a service, a resource or an environment that can be easily accessed by any type of user. The term is commonly associated with the possibility also for people with reduced or impeded sensory, motor, or psychic capacity (that is affected by both temporary and stable disability), to access and move independently in physical environments, to autonomously access and use cultural contents or to benefit from the IT systems and resources available typically through the use of assistive technologies or through compliance with product accessibility requirements.

In this context, accessibility solutions [6] are developed in order to favor the reduction or elimination of the gap between those who can independently access web resources and those who can not (in particular people with visual impairment).

In this paper, we present a solution to develop a cross-platform web and mobile accessible Rich Internet Application using the WAI-ARIA (Web Accessibility Initiative - Accessible Rich Internet Applications) [7] standardized technical specification for users with visual impairment who use screen readers or expandable and refreshable Braille displays. In order to access and use the application, there is no need to install third-party modules or plug-ins because the standard technologies used are natively embedded in the browsers of the modern mobile and desktop devices. Furthermore the solution is developed in accordance and using the new AMP (Accelerated Mobile Pages) publishing technology [8], to ensure improved performance of content and mobile compatibility as well as top indexing in search engine page results.

2. ACCESSIBLE RICH INTERNET APPLICATION

Emerging technology are pushing the development of the Internet as much as possible. But there are also a small number of people with disabilities who are still struggling with these new techniques. Content type as *role*, *state*, and *properties* of widgets and content, which are updated in real-time are often unavailable to assistive technology users. Assistive technologies typically expect the content of the web page to change in response to a navigational event, such as clicking a link or submitting a form [9]. Web applications use techniques such as AJAX (Asynchronous JavaScript And XML) to "hide" content [10], which is sometimes not recognized by assistive technologies, and while they may be affected by content changes, the user may not be aware of it or may not know where it is to locate this updated content.

In order to achieve accessibility for users with visual impairment we extended the HTML5 markup with WAI-ARIA semantic metadata as shown in Fig.1 and Fig.2.

```
<div class="tabs">
 <div role="tablist"
      aria-label="Research">
   <button
      role="tab" aria-selected="true" aria-controls="wai-aria-tab" id="wai-aria">
     Accessible Rich Internet Application
   </button>
   <button
      role="tab" aria-selected="false" aria-controls="amp-tab" id="amp" tabindex="-1">
     Accelerated Mobile Pages
   </button>
   <button
     role="tab" aria-selected="false" aria-controls="semantic-web-tab" id="semantic-web"
     tabindex="-1" data-deletable="">
     Semantic Web &amp;amp; Web Accessibility
   </button>
</div>
```

FIG. 1. WAI-ARIA semantic metadata extending the HTML5 markup for control buttons

```
    <div tabindex="0" role="tabpanel" id="wai-aria-tab" aria-labelledby="wai-aria">
      <p>
        In this paper, we present a solution to develop a cross-platform web and ...
      </p>
    </div>
    <div tabindex="0" role="tabpanel" id="amp-tab" aria-labelledby="amp" hidden="">
      <p>
        Another benefit of AMP is that it makes your content more visible ...
      </p>
    </div>
    <div tabindex="0" role="tabpanel" id="semantic-web-tab" aria-labelledby="semantic-web"
hidden="">
      <p>
      The Internet and intranets are growing continuously, the web applications are ...
      </p>
    </div>
  </div>
```

FIG. 2. WAI-ARIA semantic metadata extending the HTML5 markup for displaying the content

WAI-ARIA (Web Accessibility Initiative - Accessible Rich Internet Applications) is a specification that provides help to describe the characteristics of the self-developed widgets, making them identifiable and usable by assistive technology users. In our solution using the custom WAI-ARIA metadata [11] we provided mechanisms to alert these users to updates of the applications content.

One of the most important features of this semantic accessible marking is the flexibility that WAI-ARIA markers are independent of one another and encapsulated in HTML elements to describe the entity of an element. The tags can be easily processed by a program that knows these conventions such as a screen reader [12] or a Braille display [13].

The tags were formatted using CSS (Cascading Style Sheets) methods, shown in Fig.3, without being influenced by attributes and attribute values that specify the metadata, because they are just naming conventions.

```
<style type="text/css">
    *, *:before, *:after {margin: 0; padding: 0; box-sizing: border-box;}
    html, body {height: 100vh;}
    div.tabs {
        min-width: 320px; max-width: 800px; padding: 50px;
        margin: 0 auto; background: #fff;       }
    button#wai-aria, button#amp, button#semantic-web  {
        display: inline-block; margin: 0 0 -1px; padding: 15px 25px;
        font-weight: 600; text-align: center; color: #bbb;
        border: 1px solid transparent;}
    button#wai-aria:hover, button#amp:hover, button#semantic-web:hover{
        color: #888; cursor: pointer;}
    div#wai-aria-tab > p, div#amp-tab > p, div#semantic-web-tab > p {
        margin: 0 0 20px; line-height: 1.5;}
    @media screen and (max-width: 650px) {
    button#wai-aria:before, button#amp:before, button#semantic-web:before{
        margin: 0; font-size: 18px;}}
    @media screen and (max-width: 400px) {
    button#wai-aria, button#amp, button#semantic-web { padding: 15px;} }
</style>
```

FIG. 3. CSS3 rules used to format WAI-ARIA & HTML5 markup

WAI-ARIA is a purely semantic extension for HTML, which does not change the layout of a web page. The accessibility of dynamic pages such as Web 2.0 with its Rich Internet Applications and the general user-friendliness can be improved. Using WAI-ARIA we allowed the web page to be labeled as an application rather than as a static page.

HTML does not provide the ability to create dynamic content or advanced controls for the user interface, but allows the inclusion of applets (Flash, Java) and client-side scripts (typically JavaScript). These user-side controls and dynamic content updates are often not accessible to users with disabilities, especially for those who can not use the mouse or other pointing devices.

When creating desktop components for web applications, such as *menus*, *tree views*, *rich text fields* or *tab panels* it is usually used JavaScript. The components generally consist of * * and ** elements, which do not inherit the same functionality as real desktop components. Using WAI-ARIA we were able to easily re-propose the same web content on different platforms without loss of accessibility support [14]. In order for the keyboard to be used to activate elements, all handlers associated with the mouse events were also linked to keyboard events.

The discoverability of updated content is one of the biggest hurdles for assistive technology users, to ensure that the controls are fully keyboard usable in the application, the behavior of the self-made controls in JavaScript was implemented, as shown in Fig. 4.

```
function toggleTabs(bool) {
  var tabItem = formItems[formItems.length-1];
  if(bool) {
    tabItem.input.disabled = false;
    tabItem.label.style.color = '#1a9dd3';
    tabItem.input.setAttribute('aria-disabled', 'false');
    hiddenAlert.textContent = 'Tab section is now enabled.';
  } else {
    tabItem.input.disabled = true;
    tabItem.label.style.color = '#e5e5e5';
    tabItem.input.setAttribute('aria-disabled', 'true');
    tabItem.input.removeAttribute('aria-label');
    hiddenAlert.textContent = 'Tab section is now disabled.';
  }
}
```

FIG. 4. JavaScript implementation of keyboard usable self-made controls

For visually impaired people, user guidance and orientation improves immensely, since they can browse sections such as navigation, search or main content at any time. In addition, they immediately understand the area of the page they are currently in.

A screen reader, also called a read-aloud application, is a software that provides the blind and visually impaired with an alternative user interface instead of the text mode or a graphical user interface [15]. A screen reader communicates the information that is usually displayed on screen using non-visual output devices.

The controls and texts are acoustically reproduced mostly via a sound card or tactile via a Braille display by means of speech synthesis. Figure 5 presents the front-end view of the HTML5 block items, extended with WAI-ARIA markup, which is displayed in the web browser using the ChromeVox Screen Reader plugin, when a user visualizes and interacts with the application in order to access the information and interface with it, even modifying it, transforming its values into text; in this way users can find all the features that the applications make available to them.

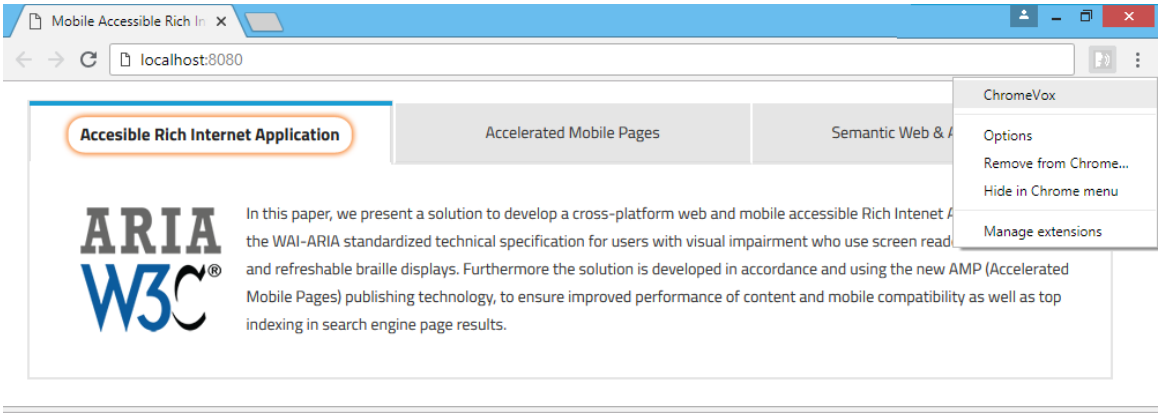

FIG. 5. Front-end view displayed in the browser using CromeVox Screen Reader

3. ACCELERATED MOBILE PAGES

Accelerated Mobile Pages (AMP) is an open source, cross-platform framework that can significantly increase the speed of loading mobile websites. AMP is based on the reduction of CSS and JavaScript, a Content Delivery Network and custom HTML [16]. The Accelerated Mobile Pages Project is supported by Google and compared to mobile-optimized or responsive-designed applications, AMP documents are loaded much faster even over low-bandwidth connections, and are rendered faster in HTML browser-based popular web browsers in less time [17].

This is achieved by consistent streamlining the code of the pages. Technical standards include "AMP HTML", "AMP JS" and "Google AMP Cache". AMP HTML is an HTML5 markup that is enhanced with some special AMP tags. AMP-JS is a JavaScript framework that causes all resources to load asynchronously. Above all, the visible elements of a page are first loaded, and only afterwards the "invisible" elements loaded. In addition, it was used the Google proxy-based content delivery network "AMP Cache", as shown in Fig. 6.

Optional, the proposed AMP solution can cache it's pages and their performance optimized to deliver faster. This also allows for a snapshot of the AMP page on Google search results pages. Images were scaled to the required size on the server side and content that can not be immediately displayed on the screen is not requested until the user starts to scroll.

```
<!doctype html>
<html>
 <head>
   <meta charset="utf-8">
   <link rel="canonical" href="index.html">
   <meta name="viewport" content="width=device-width,minimum-scale=1,initial-scale=1">
   <style amp-boilerplate>body{-webkit-animation:-amp-start 8s steps(1,end) 0s 1 normal
both;-moz-animation:-amp-start 8s steps(1,end) 0s 1 normal both;-ms-animation:-amp-start 8s
steps(1,end) 0s 1 normal both;animation:-amp-start 8s steps(1,end) 0s 1 normal both}@-
webkit-keyframes -amp-start{from{visibility:hidden}to{visibility:visible}}@-moz-keyframes -
amp-start{from{visibility:hidden}to{visibility:visible}}@-ms-keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}@-o-keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}@keyframes -amp-
start{from{visibility:hidden}to{visibility:visible}}</style><noscript><style amp-
boilerplate>body{-webkit-animation:none;-moz-animation:none;-ms-
animation:none;animation:none}</style></noscript>
   <script async src="https://cdn.ampproject.org/v0.js"></script>
 </head>
```

FIG. 6. Accelerate Mobile Pages integration

In Fig. 6, above, is presented the integration of the 3 Accelerate Mobile Pages components in the proposed solution:

- **AMP HTML** - It is a markup language with some restrictions to adapt to the objectives sought with the AMP project. In practice, the code reports many of the classic HTML tags, while some are replaced by AMP tags. These elements make some of the common patterns that affect mobile load speed lighter and better performing;
- **AMP Java Script** - The library of the AMP JS is responsible for making operational the features of the AMP, manages the loading of resources and provides the custom tags, the sandboxing of all the iframes and the loading of each resource only after having calculated the overall layout of the page;
- **Google AMP Cache** - This is a proxy-based content delivery network that loads HTML AMP pages and automatically speeds up deployment operations. When the Google AMP Cache is in use, the documents, JavaScript files and images are loaded from the same source, using the HTTP 2.0 protocol.

In Fig.7. is shown the front-end view user interface plasticity of the responsive accelerated mobile web application in a mobile browser:

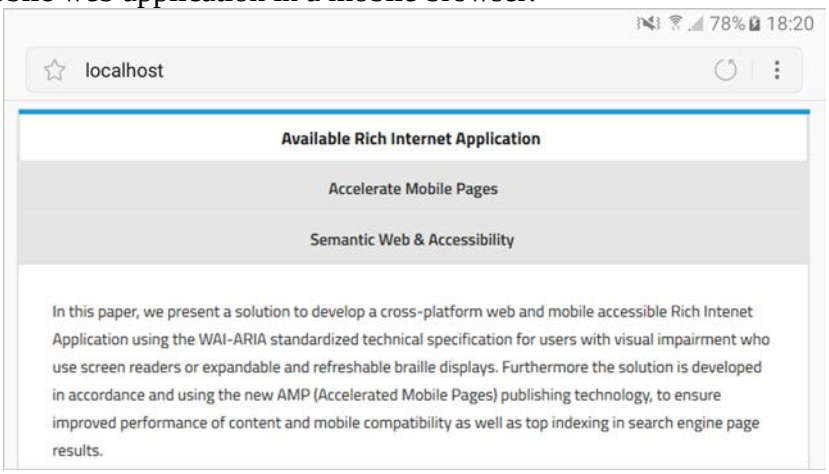

FIG. 7. Front-end view of the applications interface in a mobile browser

4. CONCLUSIONS

Accessibility means no obstacles. Accessibility is available in buildings, on the street and even in product design. Supporting computer technologies allow people with and without disabilities, regardless of age, equal support in school, education, work and leisure, so they can participate as fully as possible in the real and digital world.

Computer accessibility includes both people with technical or age-related limitations (visual deficiencies), as well as web crawlers, with which search engines capture the content of a page, correlating the web accessibility for humans and accessibility for robots. Dynamically prepared and interactively accessible information represents an innovation in information sociology to a similar extent as the Internet itself.

Nowadays, with the evolution of technology, operating systems are considered “barrier-free”. However, there are big differences of accessibility from one operating system to another. In this paper we presented and proposed a solution using standardized and cross-platform compatible technology that works independently of the operating system and the devices used, extending the “border-free” limits to web and mobile users.

The required strict separation of the structure of a document (Document Object Model) and its presentation (layout) was achieved using standard HTML5, CSS3, JavaScript and WAI-ARIA producing a "barrier-free user interface", without compromising on the applications design and scalability making it perceptible and tangible for all users.

Moreover because we used open web standards, a visually impaired user is not limited to a typical form of assistive technology, and can choose what type of software (e.g. screen reader) or hardware (e.g. Braille terminal reader) would like to use. The presented application using WAI-ARIA semantic markup is compatible with all of modern browsers and screen readers, without the necessity to install additional plugins or third party modules, that can cause security breaches, can generate unnecessary resources consumption and can make the QoE problematic and troublesome due to additional dependencies. Even if an old browser or assistive technology, that does not understand the used semantic markup, are used, the information will be ignored because WAI-ARIA comes as an extension to enhance HTML markup language and increase its informational value without altering it (visually as well as acoustically nothing will change).

One of the most important thing of the proposed application is that all the components are focusable and can be operated, continuous updates to the accessibility tree are transmitted to screen readers and Braille displays, making the information accessible to the user without having to re-read the entire page. Integrating AMP technology in the presented solution, full compatibility to mobile devices was achieved, making the application accessible and optimized to smartphone and tablet users. Because only standardized open source technologies were used, the application's graphical user interface is called by simple browser access and without the need to install third-party modules and plugins that can produce security breaches, can generate unnecessary resource consumption and may cause user difficulties and complications due to additional dependencies.

Furthermore optimizing the application for web and mobile devices and using valid and standardized technology and markup languages, made the code more viable, supporting search engines to evaluate semantically correct and accurate the content, in order to generate better search result.

ACKNOWLEDGEMENT

This paper has been supported by the Autonomous Region of Sardinia-Italy (POR FESR 2014-2020 - Asse 1, Azione 1.1.3., Project Marinanow 3.0 - RICERCA_1C-103).

REFERENCES

[1] Jodi Schneider, Tudor Groza, and Alexandre Passant. 2013. A review of argumentation for the Social Semantic Web. Semant. web 4, 2 (April 2013), 159-218, ISSN: 1570-0844;

[2] Christian Fürber and Martin Hepp. 2011. Towards a vocabulary for data quality management in semantic web architectures. In Proceedings of the 1st International Workshop on Linked Web Data Management (LWDM '11) ACM, New York, NY, USA, 1-8. dx.doi.org/10.1145/1966901.1966903;.

[3] P. Fraternali, G. Rossi and F. Sánchez-Figueroa, "Rich Internet Applications," in IEEE Internet Computing, vol. 14, no. 3, pp. 9-12, May-June 2010. doi: 10.1109/MIC.2010.76;

[4] M. Fiedler, T. Hossfeld and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," in IEEE Network, vol. 24, no. 2, pp. 36-41, March-April 2010;

[5] Marino Linaje, Adolfo Lozano-Tello, Miguel A. Perez-Toledano, Juan Carlos Preciado, Roberto Rodriguez-Echeverria, Fernando Sanchez-Figueroa, Providing RIA user interfaces with accessibility properties, Journal of Symbolic Computation, Volume 46, Issue 2, 2011, p. 207-217, ISSN 0747-7171;

[6] Giulio Mori (2012). Web Accessibility and Collaboration to Support Learning for Blind People (Doctoral dissertation). Retrieved from SISTEMA ETD - (URN etd-04242012-121149);
[7] Xabier Valencia, Myriam Arrue, J. Eduardo Pérez, and Julio Abascal. 2013. User individuality management in websites based on WAI-ARIA annotations and ontologies. In Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility. NY, USA, , Article 29 , 10 pages;
[8] P. Smutný, "Mobile development tools and cross-platform solutions," Proceedings of the 13th International Carpathian Control Conference (ICCC), High Tatras, 2012, pp. 653-656. doi: 10.1109/CarpathianCC.2012.6228727;
[9] Lilit Hakobyan, Jo Lumsden, Dympna O'Sullivan, Hannah Bartlett, "Mobile assistive technologies for the visually impaired", Survey of Ophthalmology, Eksevier, 2012, DOI: https://doi.org/10.1016/j.survophthal.2012.10.004;
[10] Al Mesbah, Arie van Deursen, and Stefan Lenselink. 2012. Crawling Ajax-Based Web Applications through Dynamic Analysis of User Interface State Changes. ACM Trans. Web 6, 1, Article 3 (March 2012), 30 pages. DOI=http://dx.doi.org/10.1145/2109205.2109208;
[11] Peter Thiessen. 2011. WAI-ARIA live regions and HTML5. In Proceedings of the International Cross-Disciplinary Conference on Web Accessibility (W4A '11). ACM, NY, USA, Article 27, 4 pages;
[12] Sylvia Söderström & Borgunn Ytterhus (2010) The use and non-use of assistive technologies from the world of information and communication technology by visually impaired young people: a walk on the tightrope of peer inclusion, Disability & Society, 25:3, 303-315, DOI: 10.1080/09687591003701215;
[13] Shiri Azenkot and Emily Fortuna. 2010. Improving public transit usability for blind and deaf-blind people by connecting a braille display to a smartphone. In Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility. ACM, New York, NY, USA, 317-318;
[14] Andy Brown and Simon Harper. 2013. Dynamic injection of WAI-ARIA into web content. In Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility (W4A '13). ACM, New York, NY, USA, , Article 14 , 4 pages. DOI=http://dx.doi.org/10.1145/2461121.2461141;
[15] Ravi Kuber, Amanda Hastings, Matthew Tretter, and Dónal Fitzpatrick0,"Determining the Accessibility of Mobile Screen Readers for Blind Users" Imaging and Signal Processing in Health Care and Technology / 772: Human-Computer Interaction 2012, DOI: 10.2316/P.2012.772-003;
[16] Ruadhan O'Donoghue, AMP: Building Accelerated Mobile Pages: Create lightning-fast mobile pages, Packt Publishing - October 2017, ISBN 139781786467317;
[17] M. E. Joorabchi, A. Mesbah and P. Kruchten, "Real Challenges in Mobile App Development," 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, Baltimore, MD, 2013, pp. 15-24. doi: 10.1109/ESEM.2013.9.