Library H

The Potential of Web Components for Libraries

Journal:	*Library Hi Tech*
Manuscript ID	LHT-06-2019-0125.R1
Manuscript Type:	Original Article
Keywords:	Web components, Library services, Library systems, Library standards, Libraries, Widgets

The Potential of Web Components for Libraries

Dr Judith Wusteman

School of Information and Communication Studies

University College Dublin

Dublin

Ireland.

Corresponding author: Dr Judith Wusteman
Corresponding Author's Email: judith.wusteman@ucd.ie

☐ *Please check this box if you do not wish your email address to be published*

Acknowledgments (if applicable):
With thanks to Peter Clarke of UCD Library for his helpful input.

Biographical Details (if applicable): -

Structured Abstract:

Purpose: This article highlights the potential of web components for libraries.

Methodology: Case study: The article introduces a working example web component that reimplements an OCLC WorldCat search widget.

Findings: By exploring the case study, the paper explains the functioning of web components and the potential advantages of web components for library web development.

Originality/value: Increasingly, web components are being used within library web development, but there is scope for much greater use of this technology to the advantage of those libraries involved.

Keywords: Web components, libraries, widgets, library services, library systems, library standards.

Article Classification: Case study

Introduction

Library patrons are increasingly demanding improved user interfaces, mobile access, and dedicated apps for a range of library functions (Back & Bailey, 2010; Bomhold, 2015; Clark & Pan, 2014; Matthews, 2016). But meeting these demands can be problematic due to the lack of technical expertise in many libraries. This paper discusses these problems, touching on one current response: the web widget. It goes on to propose an alternative solution: the web component. The latter, an increasingly important suite of technologies, provides a method of creating reusable HTML elements, similar to web widgets, but standardised and thus easier to implement, share and reuse. To illustrate the advantages of web components, an OCLC WorldCat search widget has been reimplemented as a web component. This web component is described and compared to the current WorldCat widget. The paper goes on to explore how web components are currently being deployed in libraries and how this use could be extended in the future.

Web development and libraries

In recent years, and in parallel with the rest of the web, the use of web services within library systems has become widespread. For example, they are used to incorporate information from external sources within OPACs (Back & Bailey, 2010). Such information includes additional bibliographic information, book reviews and tables of contents. In addition, OPACs themselves are being configured as web services, thus making the information they contain available to services external to the OPAC, for example via subject guides (Back & Bailey, 2010).The creation of such mashups is seen as "crucial for increasing the visibility and reach of the digital resources libraries provide" (Back & Bailey, 2010).

More recently, mobile applications and mobile websites have been developed for libraries (Potnis, 2016). These mobile services are both increasingly popular and increasingly expected by library patrons, particularly by the “millennial” generation (Bomhold, 2015). For instance, students want to use their mobile devices when interacting with research databases, the library catalogue, and reference and circulation services (Seeholzer & Salem, 2010). As Pontis (2016) stresses, "Keeping up with these interests and expectations helps libraries establish their relevancy and demonstrate how they are central to communities’ information needs."

The latter examples illustrate just some of the technical demands increasingly being placed on libraries. However, most libraries suffer from a lack of technical expertise, such as programming skills, as well as a lack of time and resources (Clark & Pan, 2014). These challenges may, in part, be responsible for the steady decrease in library-related open-source software projects reported since 2009 (Choi, 2014). This is despite some notable open source projects in the library field, for example [1,2].

Meeting patron expectations of library technology is challenging for many libraries, and various responses to these challenges have emerged (Clark & Pan, 2014). These range from simply outsourcing development work to using "ready- to-go template-based solutions" such as Library Anywhere [3], Boopsie [4] and Solus [5]. Other solutions employ cloud-based app building platforms that require "no coding

knowledge or experience" (Krol, 2018). As Clark & Pan (2014) suggest, those solutions that require no coding skills (for example [6]), tend to offer libraries "very basic and inflexible" solutions. But, as Back & Bailey (2010) comment in relation to web services, solutions that involve minimal programming are more likely to be adopted by libraries. Indeed, "whether a librarian adopts a mashup depends on how much javascript there is to write!! The less the more likely it will be adopted."

Widgets

Widgets have been the answer of choice for many libraries. As Back and Bailey (2010) point out, "widgets allow the integration and customization of Web services without requiring programming." Indeed, widgets can be deployed on library websites whilst the JavaScript remains "practically entirely hidden from the librarian" (Back & Bailey, 2010). Among the useful library widgets available are the widely-used Librarything book display widgets [7]. Other examples include the OCLC WorldCat widgets [8]; among the latter are a series of three widgets that allow libraries to incorporate a WorldCat search box on a library website, thus enabling library users to search for items in libraries near to them. The search box is illustrated in Figure 1. Figure 2 lists the HTML code that implements this search box. Library developers need only copy and paste the code into the relevant library web page and the widget is deployed.

Figure 1:

OCLC WorldCat Wide Search Box widget in use

Figure 2: OCLC WorldCat Wide Search Box widget code

```
<!-- BEGIN worldcat.org search box -->
<div id="wcs2w">
<form name="wcfw" id="wcfw" method="get" accept-charset="UTF-8"
action="//www.worldcat.org/search" target="_top" style="margin: 0;">
<input type="hidden" name="qt" value="affiliate" />
<input type="hidden" name="ai" value="[Affiliate ID]" />
<table>
```

<tr><td colspan="2" style="text-align: center; font: 11px 'Arial Unicode MS', Arial, Helvetica, Verdana, sans-serif; line-height: 1.3em; margin: 0; text-align: center; width: 250px">Search for an item in libraries near you:
<label for="q" style="color: #666;">Enter title, subject or author</label></td></tr>

<tr>

<td align="right"><input type="text" name="q" id="q" size="20" style="border: 1px solid #999; font: 12px 'Arial Unicode MS', Arial, Helvetica, Verdana, sans-serif; width: 150px;" /></td>

<td align="left"><input type="image" name="wcsbtn2w" id="wcsbtn2w" src="//www.worldcat.org/searchbox/images/wcsbtn2.gif?ai=[Affiliate ID]" alt="Search WorldCat" title="Search WorldCat" /></td>

</tr>

<tr><td colspan="2" style="font: 10px 'Arial Unicode MS', Arial, Helvetica, Verdana, sans-serif; margin: 0; text-align: center;">WorldCat.org >></td></tr>

</table>

</form>

<!-- END worldcat.org search box --> **[End Figure 2]**

This is one of many examples that demonstrate how useful widgets can be to libraries and how simple they can be to deploy. Where a widget does what is required in the way that is required, and where it is customisable to meet user needs, it can be a very attractive option. But this is not always the case: no programming requirements can mean an inflexible solution.

A further disadvantage of widgets is that they do not conform to a standard. This results in many, often incompatible, widget implementations, with no universal rules as to how they should interact with the surrounding web page. It may also result in widgets that are usable only with a specific web development framework. So, for example, in order to utilise a widget developed via the Angular framework [9], the entire website has to be built using that framework.

Web Components

The emergence of HTML5 as a W3C standard in 2014 has meant a step-change in the complexity and size of HTML, as well as an explosion in potential capability. The HTML5 standard comprises HTML and CSS markup plus JavaScript capabilities. But its use also involves the application of an entire family of related standards. Among these standards are Custom Elements, Shadow DOM, HTML Templates and ES modules. Collectively, this suite of technologies is known as "web components". It comprises "a set of web platform APIs that allow you to create new custom,

reusable, encapsulated HTML tags to use in web pages and web apps" [10]. Web components can be used in a similar way as current web widgets. And they share similar advantages for the developer, including a simple interface that does not require JavaScript programming and that hides verbose markup and scripts. However, unlike widgets, web components are based on standards, and thus are truly interoperable and shareable. And, unlike current widgets, they offer genuine encapsulation, meaning that they can be used in web pages as simply as can native HTML elements.

Spearheaded by Google in 2011 (Russell, 2011), the buzz around web components first peaked in 2014, with articles promising a "tectonic shift for web development ...with the promise of permanently reshaping the web design landscape" (Bracey, 2014). As with any hyped advance, the real-life version has been messier and slower than some expected, but the promise of a reshaped landscape remains.

Web Component Libraries

Web components can be easily shared, and components for many common web development requirements are appearing. As of June 2019, there are a range of web component libraries or frameworks [11], including X-Tag [12], Skate.js [13] and Slim.js [14]. One of the most popular libraries is Google's Polymer project [15], now in Version 3. Polymer provides a large and growing library of prebuilt web components, including Polymer elements" [16] and "paper elements"; the latter implements Google's widely-used material design guidelines [17]. Polymer also provides shims [18] to improve cross-browser performance. Web components are now supported by all major modern browsers, including Microsoft Edge as the latter adopts the Chromium browser engine. IE11 usage continues to decline but, while it is still in use, web components are supported by a shim provided by webcomponents.org.

Each new version of HTML has offered the developer added potential in the form of new HTML elements, such as sliders, calendars, input controls and so on. But, in each case, web developers are forced to wait until these elements are implemented by browsers. Web components empower web developers, not only to create their own novel HTML elements, but also to implement them in any browser supporting web components. The methods used to implement Web components are superior to the methods previously used to define standard HTML elements within the browser. As a result, Chrome has re-implemented most of the traditional HTML elements internally as web components.

Web components are superior to widgets because they are independent of any web development framework. They can be built using different technology stacks and then mixed and matched as required.

Case study

In order to illustrate the functioning and advantages of web components, the OCLC WorldCat search widget, detailed above and illustrated in Figures 1 and 2, has been reimplemented as a web component. As can be seen in Figure 2, the OCLC WorldCat Wide Search Box widget code is quite verbose and presentation-focused; it could be time-consuming to code and maintain. It should be pointed out that the WorldCat markup is an example of current good practice; its complexity is simply a reflection of the requirements of a pre-web components world.

Figure 3 illustrates the use of a web component to incorporate the same search box. The two lines of code to be incorporated within the HTML document <head> comprise

- A shim (within the <script> element) to add support for web components to older browsers such as IE11.
- A <script> element that installs a Javascript module which provides the implementation of the new sru-search web component.

Once these two lines of code have been included, our new **sru_search** element can simply be used in the HTML <body>, as figure 3 illustrates. Further, the placeholder text and the icon can be customised within the <body> by simply including different text or pointing to a different image. A developer who wishes to use this new element needs to understand nothing more than this latter simple explanation of Figure 3.

Figure 3: Use of sru-search: a Web component implementation of the OCLC WorldCat Wide Search Box

Within HTML <head>:

```
 <script src="@webcomponents/webcomponentsjs/webcomponents-loader.js"></script>
<script type="module" src="../sru-search.js"></script>
```

Within HTML <body>:

```
<sru-search placeholder="Enter title, subject, or person" max-records="20">
  <img slot="logo"
src="https://static1.worldcat.org/wcpa/rel20171102/images/logo_wcmasthead_en.png"
height="30"/>
</sru-search>
```

Further explanation of Search Box Web Component

There follows an explanation of the web component implementation. Details of how to run the web component can be found in the Appendix. Four main specifications are involved in web components: the HTML template, custom elements, shadow DOM and ES (ECMAScript) modules. They are explained below.

HTML Template

The most widely supported and straightforward specification in the web components family is the HTML template. This defines how to declare a section of markup that remains inert until used at runtime.

Most web components are built out of many lower levels of HTML elements. **sru-search** uses a range of web components from the Polymer library**.** For example, it uses **paper-dialog** within the search results display; **paper-dialog** in turn uses **paper-dialog-scrollable** to create a scrollable list.

Custom Elements

Web components are composable, that is, they may be built from native HTML elements, other web components or a combination of both. The Custom Elements specification (defined using JavaScript) enables the creation of new HTML elements; it also enables the extension of elements created by others [19]. In this case study, a new element called **sru-search** is created. The new element name must include at least one hyphen in order to distinguish it from native HTML elements.

Shadow DOM

Every web component is self-contained: the JavaScript and CSS that it comprises do not affect the rest of the web page in which the web component is used. Conversely, the rest of the page does not affect the functioning of the web component, unless the web component is written to allow this. The Shadow DOM specification defines how this encapsulation of style and markup works.

This encapsulation is at the core of the Web Components concept and is one reason why they are superior to widgets. It is true that, in some widget implementations, the CSS and HTML appear to be encapsulated in the widget. For example, there can be an attempt to scope CSS or Javascript [20] in order to reduce the likelihood of conflicts. However, the encapsulation achieved by web components using the shadow DOM is more complete than that achieved by widgets.

Further, because web components are part of the HTML specification, browser vendors have produced browser-native implementations of web components. This means that such browsers will have a native implementation of custom elements and the Shadow DOM. Thus users do not need to download all the JavaScript, CSS and HTML to implement the component framework. This, in turn, results in faster loading of pages and better performance in comparison to equivalent widget solutions.

ES (ECMAScript) Modules

Earlier browser implementations of web components used a method called HTML Imports, but most browsers (except IE 11) now use ES6 modules. The ES or ECMAScript Modules specification defines "the inclusion and reuse of JS [JavaScript] documents in a standards based, modular, performant way" [21].

Conclusion: Libraries and standards

In 2006, Wusteman commented that "The use of library-specific systems and formats has prevented libraries from integrating their resources and services with others from outside the library world." Wusteman (2006) went on to propose that, if web services were allowed to achieve their potential, they offered a method of ending this isolation. More than a decade later, web services, those "interoperable building blocks for constructing applications" (Gardner, 2001), have, indeed, played a role in helping library services become more standard and interoperable. Web components offer further support in this journey.

Increasingly, common components that can be shared across similar sites will emerge. The idea of components is not new but, with web components, they become browser-native. In a parallel to the situation of web services over a decade ago (Wusteman, 2006), three types of web component can be identified as of relevance to libraries: core, function-specific and industry-specific.

- **Core** web components could be used to describe those used as the basis for many other web components. For example, common web component libraries such as Polymer include Polymer elements and Paper elements.

- **Function-specific** web components could describe those developed by the information industry to perform domain-independent functions. The **sru_search** web component described in this article is an example of this type. Further examples, such as Mapbox [22] and Leaflet [23], are used in University College Dublin Digital Library to support geospatial visualisations (Clarke, 2017).

- In the case of libraries, **industry-specific standards** are still comparatively sparse, but could include Universal View (UV) [24, 25], an open source project to "enable cultural heritage institutions to present their digital artifacts in a IIIF-compliant and highly customisable user interface".

Whenever possible, the library sector needs to adopt standard web components, borrowing from industry-specific standards in other sectors where feasible. And, just as with web services, where this is not the case, "new standards should be developed by appropriate library-related consortia in conjunction with standards bodies" (Wusteman, 2006). This three-pronged approach supports the increasingly common model in libraries of "leveraging services for standardized services and focusing our staff's time on initiatives where they can create the most value" (Dehmlow, 2017).

The reduction in the pursuit of open source software development in libraries in the last decade has already been noted. This could be seen as a reflection of the lack of expertise, time and resources in libraries. Alternatively, it could be seen as further evidence that library software is moving into the mainstream and is increasingly able

to use and benefit from commercial and open source software from other industries.

Where restraints allow, libraries can and should get involved in the creation of industry-specific web components, and in the expansion and customisation of existing web components. But, the beauty of web components is that the use of an existing Web Component is as simple as adding a line of code to the head of a web page; after that, the web component can be used just as if it were a native HTML element.

Appendix: sru-search web component

A working example of this web component is available at: https://github.com/jwusteman/sru-search

The web component can be downloaded and installed on a web server. To view the web component in action, the index.html file (directly within the public_html folder) should be viewed from the web server.

Notes

[1] LibrarySimplified: http://www.librarysimplified.org/community.html
[2] Folio: https://www.folio.org/
[3] Library Anywhere: https://www.proquest.com/products-services/Library-Anywhere.html
[4] Boopsie: http://bfl.boopsie.com/
[5] Solus: http://www.yourlibraryapp.com/marketing/
[6] Seattle Clouds: https://seattleclouds.com/
[7] LibraryThing: https://www.librarything.com/forlibraries
[8] OCLC WorldCat Web Services https://www.worldcat.org/wcpa/content/affiliate/
[9] AngularJS: https://angularjs.org/
[10] Webcomponents.org: https://www.webcomponents.org/
[11] Web Component libraries: https://www.webcomponents.org/libraries
[12] X-Tag Web Components Library: https://x-tag.github.io/
[13] Skate.js Web Components Library: https://github.com/skatejs
[14] Slim.js Web Components Library: http://slimjs.com
[15] Polymer: https://www.polymer-project.org/
[16] Polymer elements:https://www.webcomponents.org/author/polymerelements
[17] Material Design: Introduction: https://material.io/design/introduction/

[18] A shim is "a piece of code used to correct the behavior of code that already exists, usually by adding new API that works around the problem." [https://developer.mozilla.org/en-US/docs/Glossary/Shim]

[19] Custom Elements v1: Reusable Web Components: https://developers.google.com/web/fundamentals/web-components/customelements

[20] AMD (http://requirejs.org/docs/whyamd.html) can be used to encapsulate Javascript

[21] ES6 and modules: https://www.polymer-project.org/3.0/docs/es6

[22] Mapbox: https://github.com/PolymerVis/mapbox-gl

[23] Leaflet Map: https://github.com/leaflet-extras/leaflet-map

[24] UV: https://github.com/Brown-University-Library/universal-viewer

[25] IIIF (International Image Interoperability Framework): https://iiif.io/

References

Back, G., & Bailey, A. (2010). Web services and widgets for library information systems. Information Technology and Libraries, 29(2), 76–86. http://dx.doi.org/10.6017/ital.v29i2.3146

Bracey, K. (2014). How to Create Your Own HTML Elements With Web Components. Blog post, enovatotuts+, 4 Jul. Retrieved 6 June 2019 at https://webdesign.tutsplus.com/articles/how-to-create-your-own-html-elements-with-web-components--cms-21524

Bomhold, C. (2015). Research and discovery functions in mobile academic libraries: Are university libraries serving mobile researchers?, Library Hi Tech, Vol. 33 Issue: 1, pp.32-40, https://doi.org/10.1108/LHT-09-2014-0084

Choi, N. (2014). The application profiles and development characteristics of library Open Source Software projects. Library Hi Tech, Vol. 32 Issue: 2, pp.260-275, https://doi.org/10.1108/LHT-09-2013-0127

Clark,P. (2017). Personal communication.

Clark, J. & Pan, R. (2014). Strategic mobile library development: the place of library apps and the options for creating them, ANLTC/Swets Research Award 2012 Project Report, july 2012. Retrieved 6 June 2019 from https://www.ucd.ie/t4cms/ANLTC_SWETS%20REPORT%202012.pdf

Dehmlow, M. (2017) Editorial Board Thoughts: Developing Relentless Collaborations and Powerful Partnerships. Information Technology and Libraries, June, 36.2, 3-6.

Gardner, T. (2 Oct 2001). “An Introduction to Web services”, *Ariadne*, issue 29. Retrieved 6 June 2019 from: http://www.ariadne.ac.uk/issue/29/gardner/

Krol, J. (2018). 10 excellent platforms for building mobile apps, MashableUK. Retrieved 6 June 2019 from: https://mashable.com/article/build-mobile-apps/?europe=true#yDtEpeeFyZqV

Matthews, J (2016) A nostalgic look back at library hi tech(nology), Library Hi Tech, Vol. 35 Issue: 1, pp.92-98. Retrieved 6 June 2019 from: https://doi.org/10.1108/LHT-10-2016-0116

Potnis,D., Reynard Regenstreif- Harms, and Edwin Cortez (2016). Identifying Key Steps for Developing Mobile Applications and Mobile Websites for Libraries. September, Information Technology and Libraries.

Russell, A. (2011). Web Components and Model Driven Views. Fronteers Conference, 6-7 October, Amsterdam, NL. Retrieved 6 June 2019 from: https://fronteers.nl/congres/2011/sessions/web-components-and-model-driven-views-alex-russell

Seeholzer & Salem, (2010). Library on the go: A focus group study of the mobile web and the academic library. College & Research Libraries, 72 (1), 9–20. doi: 10.5860/crl-65r1.

Wusteman, J. (2006). Realising the Potential of Web Services. OCLC Systems and Services: International Digital Library Perspectives, 22 (1) 2006, pp.5-9. Retrieved 6 June 2019 from: http://hdl.handle.net/10197/8071 DOI: 10.1108/10650750610640739

School of Information and Communication Studies
Room C116, Newman Building,
University College Dublin
Belfield, Dublin 4, Ireland.
15 July 2019

Editor, Library Hi Tech

Dear Dr Kazmer

Please find below my responses (in bold) to the reviewer's comments.

Yours sincerely

Judith Wusteman

Reviewer: 1

Comments:

##########

1. The phrase,

"So, for
example, in order to utilise a widget developed via the Angular framework [9], the
entire website has to be built using that framework."

Is not quite accurate. Though you may be using the Angular framework, you could still implement a widget using pure Javascript or something else entirely.

The reviewer said that if you are using the Angular framework you could still implement a widget in pure JavaScript (for example a web component using pure JavaScript), but that's the opposite of the point I am making. I have said that if you built the widget/component using the Angular framework then you'd have to use the Angular framework in your website to use the widget.

##########

2. This section is cumbersome.

"Polymer elements" [16] which are functionality-oriented,
and the "paper elements" which are design-oriented; the latter implements Google's widely-used material design guidelines [17]. Polymer also provides shims to improve cross-browser performance. Web components are now supported by all major modern browsers, including Microsoft Edge as the latter adopts the Chromium browser engine. IE11 usage continues to decline but, while it is still in use, web components are supported by a shim provided by webcomponents.org

If you are going to provide this much detail about Polymer, you might want to explain in more detail "polymer elements" and "paper elements" or leave them out entirely. I'm not sure you need this level of detail. Also, if you are going to mention "shims" you might want to explain what they are.

I have added a definition of shim. And I have removed some of the more technical language.

##########

3. Page 5 I would change the sentence,

As a result of this, Chrome has reimplemented most of the traditional HTML elements internally as web components.

to

As a result, Chrome has re-implemented most of the traditional HTML elements internally as web components.

Done

##########

4. Page 5 I would re-write the sentence,

Central to the success of web components, and to their advantage over widgets, is that they are independent of any web development framework.

to something like

Web components are superior to widgets as they are independent of any web development framework

Done

##

5. The subsequent sentence,

They can be built

using different technology stacks and then mixed and matched as required, their

differences being encapsulated inside the component.

Is not clear. Please re-write.

Done

##

1. Originality: Does the paper contain new and significant information adequate to justify publication?: A search of Google Scholar as well as EDS at our library found a number of articles related to web widgets in libraries.

There are articles published on web widgets in libraries - and I have referenced them. But this article is about web components, not web widgets.

The article below references web components but does not go into an in-depth explanation of how they are created.

Chad, K., & Miller, P. (2005). Do libraries matter. The rise of Library, 2(0).

The above article appeared 6 years before web components were invented. The single reference to "web components" isn't referring to the same technology that I describe.

2. Relationship to Literature: Does the paper demonstrate an adequate understanding of the relevant literature in the field and cite an appropriate range of literature sources? Is any significant work ignored?: The number and quality of the sources are appropriate for a paper of this length.

3. Methodology: Is the paper's argument built on an appropriate base of theory, concepts or other ideas? Has the research or equivalent intellectual work on which the paper is

based been well designed? Are the methods employed appropriate?: This paper is not a research study but rather an explanation of a technology with examples of how that technology might be implemented. As well as it's potential impact on libraries.

4. Results: Are results presented clearly and analysed appropriately? Do the conclusions adequately tie together the other elements of the paper?: There aren't really results. However, the case study approach is appropriate for this type of paper.

5. Implications for research, practice and/or society: Does the paper identify clearly any implications for research, practice and/or society? Does the paper bridge the gap between theory and practice? How can the research be used in practice (economic and commercial impact), in teaching, to influence public policy, in research (contributing to the body of knowledge)? What is the impact upon society (influencing public attitudes, affecting quality of life)? Are these implications consistent with the findings and conclusions of the paper?: The paper presents the practice of implementing web components, their advantages over web widgets and the potential impact. Implications for libraries are clearly stated.

6. Quality of Communication: Does the paper clearly express its case, measured against the technical language of the fields and the expected knowledge of the journal's readership? Has attention been paid to the clarity of expression and readability, such as sentence structure, jargon use, acronyms, etc.: As noted, one of the main advantages of web components in libraries is the ability of people with limited technical skills to implement them. If the intended audience is people with limited technical skills, the technical explanations may need to be explained at a more rudimentary level.

Addressed in responses to comments above.