A Platform-Independent Plugin for Navigating Online Radiology Cases

Jason D. Balkman¹ · Omer A. Awan¹

Published online: 3 November 2015
© Society for Imaging Informatics in Medicine 2015

Abstract Software methods that enable navigation of radiology cases on various digital platforms differ between handheld devices and desktop computers. This has resulted in poor compatibility of online radiology teaching files across mobile smartphones, tablets, and desktop computers. A standardized, platform-independent, or “agnostic” approach for presenting online radiology content was produced in this work by leveraging modern hypertext markup language (HTML) and JavaScript web software technology. We describe the design and evaluation of this software, demonstrate its use across multiple viewing platforms, and make it publicly available as a model for future development efforts.

Keywords Software design · Web technology · Medical displays · Electronic teaching · File

Introduction

The use of handheld devices in health care has greatly expanded the number of ways physicians view electronic radiology content [1–4]. Many resources offering scrollable images are not compatible across devices. For example, developers and clinicians alike have created a number of image-rich radiology teaching applications written using Objective C, the native language for Apple devices. Unfortunately, these applications are frequently not compatible with Android devices because of disparate software development kits.

Cross-compatibility also remains an issue for online radiology resources. Radiopaedia is one of many example websites that enable scrolling through online radiology image stacks, but at the time of this writing, content served directly from the website uses older hypertext markup language (HTML) components which are not compatible with mobile devices. These implementations also expose the image source uniform resource locators (URLs) in the publicly viewable HTML code. Adobe Flash has also been used to create image stacks on radiology websites, but this technology is historically incompatible with certain handheld devices [5].

Given the necessity for compliance across platforms and the superior reach of online content, web-based software technology may currently be the best option for creating widely accessible electronic radiology teaching files. However, a unified solution is needed. Radiology cases represent a unique form of web media that requires a dynamic user interface that handles an interactive stack of images. The advent of HTML5 introduced the *canvas* element, which is an enhanced HTML display widget that can manipulate images and respond to user input on both handheld and desktop computers. When combined with a feature-rich JavaScript library like jQuery, the canvas element may serve radiology image stacks and provide platform-independent, PACS-like navigation features through a modern web browser.

In this work, a web application plugin that provides a cross platform web solution for viewing radiology image stacks on a variety of mobile devices and desktop computers is developed, evaluated, and made available for public use and extension.

✉ Jason D. Balkman
balkman.jason@gmail.com

¹ Department of Radiology, Dartmouth-Hitchcock Medical Center, One Medical Center Drive, Lebanon, NH 03766, USA

Methods

All software code, including HTML5 and JavaScript files, were created and edited using GNU Emacs, version 23.4.1 (Free Software Foundation, Boston, MA) running on a

MacBook Pro computer (Apple, Cupertino, CA). The feature-rich object-oriented JavaScript library jQuery version 2.0.2 was utilized from Google Hosted Libraries (Google, Mountain View, CA).

A jQuery-based plugin called *mooscan*, for displaying radiologic images using HTML *canvas* elements, was created in a single JavaScript file, *jquery.mooscan.js*. The file was composed of less than 700 lines of code occupying 21 kilobytes (kB) of memory. The plugin is invoked by programmatically attaching it to a canvas element in an HTML file using a single line of JavaScript code. For example, if the canvas object is called *mycanvas*, invoking the plugin has the general or “pseudocode” format “*mycanvas.mooscan*[argument 1, argument 2, etc....]”. The plugin receives seven arguments or function inputs. These include an empty array used to store a list of URLs for each image slice, *X* and *Y* coordinates that define where to start drawing the images on the canvas, the desired width and height of the image, and the current width and height of the browser window.

Eleven functions that enable user actions are implemented as part of the plugin (Fig. 1). These functions are programmatically bound to *events* that modern hardware and software detect [6]. Depending on the platform, an event may consist of a keystroke, mouse action, or mobile device hand gesture that is transmitted to the web application (Fig. 2). For example, when the user swipes the screen on a mobile device, the “touchstart,” “touchmove,” and “touchend” events trigger the custom “handleStart,” “handleMove,” and “handleEnd”

Fig. 1 Action-event-function mappings. Users predominantly interact with handheld devices using touchscreens (*top row*) and with desktop computers using a mouse and keyboard (*bottom row*). Device hardware and software listen for and process the hardcoded *events* listed in the *middle column*. JavaScript was used to bind these events with the custom plugin functions listed in the *column on the right*. The functions define how images in the HTML *canvas* element respond to user input (not shown)

Fig. 2 Processing user input on handheld devices. Touchscreens and iOS hardware detect a single-finger touch in this example. Software transmits the user’s finger touch to the web browser which categorizes the action as a *touchstart* event. The JavaScript plugin, having programmatically bound the *touchstart* event to the custom *handleStart* function, will execute that function’s code so that the web application responds to user input appropriately

functions. The plugin enables these functions to display the scrolling image stack by ordering the canvas object to redraw the next image slice. Similarly, functions that implement zoom and pan navigation of images are bound to unique mouse and touch events.

The *mooscan* plugin was uploaded to an open source repository for free public use and extension [7]. The plugin was implemented in three public online web applications [8–10]. These applications implement canvas elements in their HTML code and the plugin was programmatically linked to these elements.

Results

An online radiology application implementing the plugin [8] was tested on MAC and PC desktop/laptop computers as well as Android and Apple handheld devices using three modern browsers: Chrome, Firefox, and Safari (Table 1). Because Internet Explorer does not support touch events and the user interface to this particular application, it was not included in the evaluation.

Full functionality of image navigation, which includes image pan, zoom, scroll, and slice increment/decrement, was supported on Apple devices and desktop computers running either Chrome or Safari browsers. Both the mouse wheel as well as mouse click and drag actions were functionally linked to scrolling through images on the desktop. Zoom and pan were controlled on desktop browsers by dragging the mouse and holding the “z” or spacebar keys, respectively. On handheld devices, images could be scrolled using a single-finger swipe gesture, while zoom and pan were controlled using two-finger gestures. Slices could be incremented/decremented by tapping or clicking the left/right side of the screen on handheld devices or desktop platforms, respectively. Scrolling using the

Table 1 Plugin functionality across multiple devices and browsers. Testing of the plugin demonstrates complete functionality on desktop computers and Apple handheld devices running Chrome and Safari browsers. Most functionality operated as expected on the Android device, with the exception of two-finger zooming

	Desktop/Laptop		Handheld device		
	MAC	PC	Android 4.4.2	iPad	iPhone 4S/5S
Chrome	✓	✓	✓ ^a	✓	✓
Firefox	✓ ^b	✓ ^b	✓ ^a	N/A	N/A
Safari	✓	✓	N/A	✓	✓

Checkmarks designate that pan, zoom, scroll, and slice increment/decrement work. N/A (not applicable) designates that the browser is not compatible or was not tested with the listed platform

^a All functions work except two-finger zoom

^b All functions work except scrolling using the mouse wheel. Scrolling using mouse click and drag functions properly

mouse wheel did not work using Firefox, though scrolling was preserved using the mouse click and drag action. All functions mentioned above except for the two-finger zoom were available on Android devices.

The plugin was also successfully integrated with breast density and chest nodule analysis web applications [9, 10]. Both applications were tested on a laptop computer using three modern browsers with the same results as above. Handheld device testing was not applicable for these computer-aided analysis applications as mobile versions had not been made available.

Discussion

Because radiology studies are frequently viewed electronically as interactive images stacks, navigation of radiology cases requires unique software implementations not frequently utilized by other forms of media. Furthermore, little attention has been given to developing standardized open source tools for presenting online radiologic images. As a result, there is currently poor cross-compatibility of electronic radiology resources across viewing platforms, especially for handheld devices.

The development and release of the open source radiology plugin described above represents an initial effort to standardize methods for presenting online radiology images while increasing awareness of underlying compatibility issues. The aim is thus to increase accessibility of radiology content and help radiologists work with web developers and institutional information technology (IT) departments to improve online radiology content.

The current plugin allows compatibility across Apple devices and desktop computers. Android browsers lack the two-finger gesture functionality of the iPhone or iPad due to an existing Apple patent [11]. Thus, the zoom feature of the plugin does not work on Android devices and a workaround has not yet been implemented.

This open source radiology plugin could have several clinical and educational applications. With the ability to zoom and pan scrollable images on both a desktop and a mobile device, referring physicians and radiologists could have greater and more widespread access to clinical cases. As images can be reviewed remotely by clinicians with various different platforms, there will be inevitably less barriers to timely image review and interpretation. Furthermore, from an educational standpoint, radiology residents and medical students would have the opportunity to learn from scrollable image sets across different platforms, thereby allowing more efficient education without the need to shift from one platform to another.

The current plugin is limited to basic image navigation functions. Since the development of this software, other similar open source projects have been released, some of which allow users to build more versatile PACS-like web viewers that include window width and level functionality [12]. Building upon such existing projects will expedite development of web-based DICOM viewers that rival the performance of currently available commercial products. These may not only be used to share images between institutions but may also be used to extend imaging to patient portals, where individuals can directly access their radiologic studies.

Conclusion

The intricate aspects of navigating online radiology images combined with the rapid development of mobile technologies has led to widespread incompatibilities and little consensus for presenting online radiology content. An open source jQuery-based plugin was created, uploaded to a public repository, and implemented in three web applications to demonstrate the feasibility of standardizing the presentation of online radiology image stacks across multiple web browsers and device platforms.

The plugin allows PACS-like navigation of online radiology stacks and demonstrates full functionality for two of three tested browsers running on desktop computers and handheld devices. Patent protection has limited the implementation of two-finger gesture events on Android devices, affecting the zoom feature of the plugin. Workarounds to this limitation should be investigated in future software versions. Radiologists interested in developing teaching content for a broad online audience should consider the compatibility

issues discussed above and work with experienced web developers to target multiple device platforms.

References

1. Richardson ML: Software Aids for radiologists: Part 2, Essential apps for handheld Devices. *AJR Am J Roentgenol* 199(6):W714–W722, 2012
2. Toomey RJ, Ryan JT, McEntee MF, Evanoff MG, Chakraborty DP, McNulty JP, Manning DJ, Thomas EM, Brennan PC: Diagnostic efficacy of handheld devices for emergency Radiologic consultation. *AJR Am J Roentgenol* 194(2):469–474, 2010
3. Raman B, Raman R, Raman L, Beaulieu CF: Radiology on handheld devices: image Display, manipulation, and PACS integration issues. *RadioGraphics* 24(1):299–310, 2004
4. McLaughlin P, McGarrigle AM, Maher MM: Comparison of handheld devices for Emergency radiology. *AJR Am J Roentgenol* 196(4):W487, 2011
5. Lawler R: Steve Would Be Proud: How Apple Won The War Against Flash. *TechCrunch*, June 30, 2012. Available at <http://techcrunch.com/2012/06/30/steve-jobs-war-against-flash>. Accessed September 27, 2015.
6. “Handling Events,” *iOS Developer Library*, 2014. [Online]. Available at <https://developer.apple.com/library/ios/documentation/appleapplications/reference/safariwebcontent/HandlingEvents/HandlingEvents.html>. Accessed September 27, 2015.
7. Balkman JD: MOO Scan Radiology Web Application Plugin. *GitHub*, 2014. [Online]. Available at <https://github.com/jbalkman/mooscan>. Accessed September 27, 2015.
8. Balkman JD, Loehfelm TW: A Cloud-based Multimodality Case File for Mobile Devices. *RadioGraphics* 34(4):863–872, 2014
9. Balkman JD: QADense. 2014. [Online]. Available at <http://qadense.com/>. Accessed September 27, 2015
10. Balkman JD: Chest Detect. 2014. [Online]. Available at <http://chestdetect.com/>. Accessed September 27, 2015
11. Tolmasky F, Williamson R, Blumenberg C, Coffman P: “Portable multifunction device, method, and graphical user interface for translating displayed content.” U.S. Patent 7966578 B2 2011
12. Hafey C: Cornerstone. *GitHub*, 2014. [Online]. Available at <https://github.com/chafey/cornerstone>. Accessed September 27, 2015