EMERGING TECHNOLOGIES

TOWARDS TRANSPARENT COMPUTING: CONTENT AUTHORING USING OPEN STANDARDS

Robert Godwin-Jones, Virginia Commonwealth University

With today's rapid developments in digital technologies, technical obsolescence can occur much faster than in the past. Who would have predicted five years ago that Adobe's Flash would have seen the rapid decline it has experienced as a development environment? Using open, internationally accepted standards for materials development is no absolute guarantee of longevity, but it does increase the likelihood that content will continue to be usable and that, if needed, conversion tools will be available. In this column I will be discussing approaches to the development of electronically delivered language learning materials that I believe are the least likely to face short-term obsolescence. I will be arguing in favor of multilaterally developed, open standards, supported by major industry players and educational standards bodies. Specifically, I will be looking at approaches for delivering learning materials through Web browsers or e-book readers (e-readers), so that created content works seamlessly across devices and platforms.

APA Citation: Godwin-Jones, R. (2014). Towards transparent computing: Content authoring using open standards. *Language Learning & Technology 18*(1), 1–10. Retrieved from http://llt.msu.edu/issues/february2014/emerging.pdf

Copyright: © Robert Godwin-Jones

WEB DELIVERY

For the vast majority of language learning projects, the delivery mechanism of choice is likely to be the World Wide Web, accessed through a Web browser, whether that be on a desktop computer or on a mobile device. There are use cases that may dictate a different delivery strategy, such as a compiled application (for an intelligent language tutorial, for example), or a mobile app (if a particular device is being targeted). It's more and more likely, however, that no matter what the programming environment is on the backend, the client-side delivery will be (and should be) Web-based. As I've argued in a previous column (PDF), that holds true for mobile delivery as well (Godwin-Jones, 2011). In fact, developing for the Web with current technology standards makes that content most widely useable. This doesn't mean that the created Web pages will have the identical look and feel in all environments, but it does mean that the content and functionality of the pages will essentially be the same. This is made possible through the most recent version of the Web programming language, HTML5 (by convention capitalized with no space before version number), which, since December, 2012 is a "Candidate Recommendation" of W3C, the standards body for the World Wide Web. HTML5 has features that enable capabilities not available before (or only available through proprietary plug-ins), many of which are of substantial interest to language professionals.

At the Intel Developer Forum in 2012, a language learning application was demonstrated (video clip) that illustrates both the versatility of HTML5 and its interactive features. The demonstration ran on an iPad (using Apple iOS) and on a Windows 8 laptop. This interoperability would be no challenge for typical Web pages. This example was, however, more akin to an application than normal Web pages, as it made rich use of multimedia, animations, and interactivity. This would typically be considered a "Web app". The media playback and interactivity were accomplished without the use of plug-ins such as Microsoft Silverlight or Flash, neither of which are supported on the iPad. The app featured synchronized audio with animated character renditions, character writing with instant feedback (including verification of stroke order), adaptive assessments, self-correcting exercises, and integrated cultural videos. The activities included a drag and drop vocabulary drill, an animated interactive timeline of Chinese history, and active manipulation of 3-D graphics (world map). New characters were introduced with a rich set of contextual examples, easily shown or hidden, all displayed without page refreshes, as the needed data were pre-loaded from a server. The video clips played instantly (also due to preloading) and featured custom playback controls.

The app was demonstrated as an example of "transparent computing", the idea that the consumer should not have to be concerned with which connected device is being used and can easily switch devices/platforms and have a similar user experience, with automatic data syncing. This kind of interoperability is only possible through the adoption and implementation of common standards. In contrast, many development environments today, particularly in the mobile space, represent a world of walled gardens with no connected pathways. A mobile app developed natively for Apple devices (using Objective-C) will not run in Android or Windows devices, or even on Mac computers. This pattern holds true for development on other platforms as well. This means that developers must target a particular mobile OS or, alternatively, create separate apps for each.

Any data generated by the user often remains within the walled garden. Through sending that data to a server ("cloud services"), syncing is possible, but not necessarily across different platforms. In any case, the cloud data is reserved for use by that app (or its siblings on other platforms) and is not accessible in other ways. This may be understandable in terms of privacy and security, but it serves to inhibit the development of open cloud-based services for language learning, which could be built on top of such data. Having available user data including achievement milestones, proficiency levels, and interest areas could offer the possibility of creating individualized learner profiles, forming the basis of personalized learning in intelligent language learning environments. This is a vision of the future only achievable if the stored data is encoded and structured in a non-proprietary format.

HTML5 FEATURES

The Chinese language-learning app from Intel uses a variety of HTML5 features. Animations, for example, are created in Canvas, a graphics-rendering environment which uses scripting to draw and manipulate images. Canvas elements can be quite simple and static or extraordinarily rich, complex and animated. HTML5 also supports SVG (Scalable Vector Graphics), which draws graphics in the XML language (Extensible Markup Language). These graphic environments have advantages over other formats in that they do not lose quality when zoomed or re-sized, and each drawn element can be animated and scripted. Both are widely used in Web-based game programming, an area of increasing interest in language learning. One of the caveats that should be mentioned here, and that applies to the graphics capabilities, as well to some other HTML5 features, is that they are not universally usable in all desktop and mobile browsers. Canvas, for example, works across current versions of all the major browsers, but not in older browsers (whose use might be necessitated by older OS versions). It's important to be aware of the implementation status of any new HTML5 features which are crucial to the functionality of a Web app. Sites such as html5test.com (evaluates browser in use) or caniuse.com (support table) can provide help with that process. As always in content development, it's a good idea to implement HTML5 using progressive enhancement, i.e., assuring basic content delivery and functionality for all browsers, while enabling a more advanced, interactive view for users of supported browsers. This is easiest implemented with a JavaScript library such as modernizr.com.

Embedded media are intended to be played in HTML5 without the need for plug-ins, through native support for the new "audio" and "video" tags. Given the importance of multimedia in language learning, this development is welcome, as it means better delivery options, particularly for video. Traditionally, video could be delivered through 1) directly clicking on a link to the video file and waiting for it to download, 2) using a commercial streaming service, or 3) using a proprietary plug-in which may not be available on all platforms. The audio and video elements feature auto-buffering, allow custom controls to be used, and, since they are normal elements of the page's DOM (Document Object Model), can be manipulated by other elements of the page. The playback of a particular segment of a video, for example, could result from user actions such as answering a question. For particular effects, video and Canvas elements can be used together, as in a cartoon video featuring a Canvas-drawn background. There is a W3C working group (Device APIs) that is working on a standard for capturing a user's video camera, if available, and if permission is granted (info and example, supported currently in the Chrome browser). One could also filter the video from a webcam using Canvas.

The main hiccup currently in using HTML5 video is the lack of support among browsers for a common video codec (video compression format), with some browsers supporting h.264 (MP4) and others VP8 (WebM), making it currently advisable to serve up both encodings. A promising aspect of HTML5 video for language learning is its use of WebVTT (Video Text Tracks), to display subtitles, captions, chapter headings, or screen reader descriptions. The ability to switch a caption or subtitle on or off, or to switch languages on the fly, and to do so programmatically, is a great boon to language learning applications. For universal support of this feature currently, using the free video.js (open source HTML5 video player) or a similar program is needed.

The user actions which could result in video playing, jumping to a scene, or turning on subtitles, could be generated in a variety of ways. It might involve filling in a form element, such as a short answer or multiple choice question, or possibly moving items on the page, such as arranging pictures illustrating events in a story. HTML5 offers new native support for drag-and-drop. Designating "draggable" as an attribute to a page element makes it "live". Intel's Chinese language learning app uses drag and drop to have the user select an appropriate image to answer a question. This function may, indeed, seem to target image manipulation, but drag and drop could involve text as well, as in this example. Ordering items, sentence construction from words or phrases, matching, are some of the possible uses in language learning. Drag and drop can also be used for uploading files, allowing, for example, images or text files to be dragged from the desktop to be uploaded.

It used to be the case that text entry and text manipulation were only possible in HTML form fields. HTML5 supports the "contenteditable" attribute, normally used in text fields, which now can be used for any page element, allowing for rich text editing anywhere on the page. An example of a page element made "editable" also illustrates an additional welcome feature of HTML5, support for storing name/value pairs in the browser memory. This used to necessitate using HTTP cookies, which are hampered by size and usage restrictions. Web storage allows for temporary, session-based storage of values ("sessionStorage") or persistent storage, after the browser is closed ("localStorage"). This more robust form of browser storage of variable values is crucial for Web apps to be able to bring users back to finish items only partially completed, with completed items remembered. This functionality is also needed for Web browsers to recall scores or other data from one page to another. Web storage is supported widely in both Web browsers and e-readers. Some developers prefer storage options that are offer more of a database-style format such as WebSQL or IndexedDB, but they are not universally deployable (or supported as W3C standards).

Other new HTML5 features in terms of interactivity include a multitude of new attributes for Web forms and form elements. Pattern matching in text fields (also called use of "regular expressions"), for example, allows for expanded options for evaluating user input. This enables testing for the presence of a core item with the added ability to analyze verb endings or inflections, and provide individualized feedback. Another attribute of possible use in grammar exercises or in writing help is the use of "datalist" controls, which suggest input to be used (inflections, for example), based on a supplied list, providing a kind of auto-complete functionality (example). Also new, but not yet widely supported, is a speech input field (demonstration) which uses speech recognition to convert speech to text. Among other text manipulation features, one that may be of interest for Asian languages is support for ruby annotations. These are glosses used with logographic languages to indicate tones, or to provide other help in pronunciation. They're widely used in Japanese (*furigana*) in which *kanji* are glossed with *hiragana*. Sites such as html5demos.com or others listed in the reference list provide numerous demonstrations of HTML5, which may provide ideas for applications in language learning.

E-TEXTS

HTML5 is the basis for what has become the standard format for electronically delivered books, EPUB 3 (by convention capitalized with space before version number). This is the standard created and promoted by a nonprofit trade and standards organization, the IDPF (International Digital Publishing Forum), with members from 35 countries. EPUB 3 (EPUB stands for electronic publication) was approved as a "Final Recommended Specification" in October, 2011, and is now in wide use by publishers and e-readers. The "big six" U.S. publishers format their book content now in house as EPUB 3 (rather than in XML). The EPUB 3 format can be thought of as a Website in a box, as the content is encoded in HTML5 or XHTML. As is the case with HTML5, a typical EPUB 3 will also include media files (for images, audio/video), CSS files for formatting (Cascading Style Sheets; EPUB 3 supports CSS3), and .js files (JavaScript). That content is accompanied by a manifest file, providing a complete list of files as well as navigational information (i.e., a chapter list), with a mime type declaration ("application/epub+zip"), and a file ("container.xml") pointing to the name and location of the manifest file. Metadata is included in the manifest file and can also be contained in separate, linked files. EPUB 3 incorporates standard Dublin Core metadata but also supports use of fine-grained metadata, essential for cataloging and searching. The folder containing all files for the EPUB 3 is zipped and given an .epub file extension. The IDPF provides a complete description of what comprises a valid EPUB 3 package.

If an EPUB 3 package is basically Web-formatted content, why not just display that content as HTML in a Web browser? In fact, some have argued for that. There are however a number of compelling use cases for EPUB 3. One of the key advantages to EPUB is that once it is loaded onto a device, that content is available whether the device is connected to the Internet or not. HTML5 content can also be cached so as to be available off-line, but that process is cumbersome. Optimized text formatting is another key feature. EPUBs are designed for reflowable content, so the text display is optimized for the device on which it is being used, as contrasted, for example, with the fixed layout of PDF. Text in a Web browser too can be displayed on different sized screens, and techniques of responsive Web design can automatically adjust for different devices. However, these techniques involve adding extra code to the HTML and are infrequently used. With EPUB 3 the text flows to adapt to the size of the display with automatic pagination and navigation specific to the device or app used. Page breaks and pagination on an iPad and on an Android phone, for example, will be adjusted for those devices. E-readers make it easy to adjust font characteristics (size, color, and face), to change background color, or to adjust page turning mechanisms. Additionally, EPUB 3 supports bookmarking, note-taking, and automatic dictionary lookups. For content that is primarily text -- especially if it is of considerable length -- the EPUB format, with its built-in book-friendly features, is preferable to straight HTML5.

EPUB 3 was designed with accessibility in mind. HTML5 supports some semantic markup such as tags for article, aside, dialog, or summary. EPUB 3 adds more semantic tags, through the "epub:type" attribute, which was designed specifically for accessibility. Additional tags can specify such text elements as preface, chapter, or sidebar. This allows for tagging items so as to make the text structure clearly understandable to e-readers. It also enables logical reading order to be defined for those using assistive technologies. EPUB 3 provides full support for the ARIA specification (Accessible Rich Internet Applications) of the Web Accessibility Initiative. Also supported is text to speech. The semantic tags in EPUB 3 are not close to being as complete and comprehensive as they are in the TEI specification (Text Encoding Initiative), but they can be helpful in defining the basic structure of documents, a process which benefits all readers. Tagging footnotes, for example, allows for e-readers to display that information in a user-friendly way.

The extensive semantic mark-up is a feature that greatly facilitates EPUB versions of textbooks. This is one aspect of EPUB 3 that has led to the considerable interest we are seeing today among publishers. The International Publishers Association in March, 2013 endorsed EPUB 3 as its preferred global publishing standard. The Association of American Publishers has begun an EPUB 3 implementation project, which calls for the release of a large number of EPUB 3 titles in 2014. Publishers who want to make their content available electronically generally insist on some form of embedded digital rights management (DRM), which is much easier to implement in EPUB 3 than in HTML5. Most commercial e-texts are only available in e-readers supporting DRM, whether they be dedicated reading devices (Kindle, Barnes & Noble Nook) or apps supporting DRM (iBooks, Kindle app, Adobe Digital Editions).

EPUB 3 AND LANGUAGE LEARNING

EPUB 3 incorporates features that will be of particular interest for language learning. It offers robust support for non-Latin writing systems. Supported, for example, are vertical writing and right to left page progression (sample for Japanese). EPUB 3 allows use of non-ASCII characters in names such as HTML anchors or filenames (as used in sample referenced above). Emphasis dots and lines can also be used, where the equivalent in western script would be italics. Sample e-books in Arabic, Hebrew and Devanāgarī illustrate text layout and character display in those languages. O'Reilly's EPUB 3 best practices (highly recommended) provides additional examples of global language support. It's also possible to embed a font in an EPUB 3, which allows for character display in virtually any writing system.

Another benefit for language learning in the EPUB feature set is synchronized audio narration, in which text is highlighted as read, as in this Japanese manga example (direct link to EPUB 3 file; needs to be viewed in one of the e-readers in reference list). This is a feature of obvious interest in children's books (Dickens' Christmas Carol audio e-book; EPUB 3 file), but could be quite useful in language learning as well, not to mention literary study (Dante's Divina Commedia with audio). This is implemented through the use of media overlays. Overlays allow the user to switch easily and instantly from one reading modality to another. One might start reading at home, then switch to audio mode in the car. To create a media overlay, the typical process is to record the audio with a tool such as Audacity, mark and export the synchronization points, then merge this data with the text. Some sample code from the Moby Dick e-book illustrates SMIL use (Synchronized Multimedia Integration Language) to create this kind of "read along" functionality.

EPUB packages consist of a single zipped file. This can be constructed by hand by writing the needed manifest files, making sure content is valid HTML5 or XHTML, using the required naming conventions, zipping the folder, and branding it as EPUB. Validators such as the IDPF EPUB tool can pinpoint errors in the code. There are tools available for creating EPUB 3 e-books or converting from other formats, for those who prefer not to code manually (see reference list). A widely used cross-platform conversion utility is Calibre, which also functions as a desktop e-book reader. It can read and write in a variety of formats, including the proprietary KF8, used by Amazon's Kindle readers. As EPUB 3 becomes more widely supported, it's likely we will see more and better tools for formatting content as EPUB 3. Particularly welcome would be tools which make it easier to take advantage of advanced features such as audio synchronization.

While it's obvious how using EPUB 3 for delivery of long-form texts supplies a number of benefits, it may appear that aside from interactive multimedia capabilities, the format is largely static and quite different from the dynamic environment of HTML5, with its support for the use of highly interactive Web apps. While it's certainly the case that complex, graphics-intensive Web apps would not be the best candidates for the EPUB format, scripted interactions are in fact supported in the EPUB 3 specs. Scripting support is not required, although most current e-readers include JavaScript support, although, unfortunately, they do not have any script debugging. Scripted interactivity is normally created using JavaScript and embedded in the HTML or included in external .js files -- in other words in the same way as it would be for Web browser delivery. Typically, textbooks in EPUB 3 format include assessments at the end of each chapter. A sample EPUB 3 (EPUB 3 file) from Infogridpacific illustrates a variety of scripted activities similar to what one finds in electronic textbooks. Although e-texts formatted in EPUB 3 allow scripted interactions, connecting to external Web resources, although permitted, can be problematic. In such situations, one is better off using straight HTML in a Web browser. This is also the case for integration of social media, which is also limited within EPUB 3.

OUTLOOK

For a project constructed in HTML5, it is not a major extra effort to create an EPUB 3 version as well. Doing so adds some potential book-related features, but the greater benefit is the additional delivery options it provides. This could be of particular interest in schools with tablet initiatives, that are looking to distribute content that will be usable at home without Internet access. Creating an EPUB 3 for use in iBooks on an iPad or iPhone (or now on Mac OS computers as well) allows creators to bypass the lengthy iBookstore approval process. In contrast, if one were to use Apple's e-book creation tool, iBooks Author, the resulting product would be in a proprietary format, displayable only on Apple devices. Standard EPUB 3s, on the other hand, can be shared easily with students in a variety of ways, including a simple download link on a Web page.

There are some features missing from the EPUB spec that would be helpful in an educational setting. One such feature is a standard way to create annotations and glosses, obviously an important feature for language learners. Although it is possible to integrate into EPUB 3 look-ups from dictionaries or other reference sources, there is as yet no widely supported method for annotations, despite the presence in the specification of tags for glosses. Some e-readers have developed a means for achieving this function while staying within the EPUB 3 specs. Apple iBooks, for example, relies on the footnote tag to create pop-up overlays to display footnotes or annotations, rather than taking the reader to the end of the document. For a true seamless e-book experience across devices, annotations should -- as is not the case now -- transfer across reading apps and platforms, as should bookmarks and notes. Ideally, sharing such data with others should be possible as well. The EDUPUB initiative is looking to deliver a way to do that.

Another important area for educational use is a robust and compatible scripting environment. Despite the fact that most of the popular e-readers are based on WebKit, Apple's open Web browser engine, they differ markedly in JavaScript execution. The IDPF has recognized this issue and is making its own JavaScript engine (ReadiumJS) which will be made freely available to e-reader producers. IDPF is also working with IMS Global, the education standards body, to support both the QTI specification (Question and Testing Interoperability), widely used by publishers for creating test banks, and LTI (Learning Tools Interoperability), which would allow for better integration of EPUB 3 into an LMS (Learning Management System). A sample LTI and QTI integration demonstrates how that might be done. That particular prototype is of interest as well in that it demonstrates how to create fallbacks should network connections be unavailable, or should the connection be lost. An additional consideration in using EPUB 3 for content delivery is that revising content requires creating a new EPUB 3 file for downloading. In contrast, with HTML, a text editor can be used to do revisions to a single page, without the necessity of redistributing the entire unit of content in a package.

Following the development strategy and roadmap I have outlined here is not likely to appeal universally to developers, particularly those targeting mobile delivery. Mark Zuckerberg famously announced in 2012 that his biggest mistake in developing mobile apps for Facebook was the decision to use HTML5, citing slow performance and design constraints. These are common complaints from developers, particularly those working on large-scale projects such as the mobile Facebook app. As one developer discusses, development in such cases using a native development environment might be preferable. That necessitates, however, creating compiled applications in different programming languages (for iOS, Android, Windows 8), not a task to be undertaken lightly. A possibility for creating native apps is to port the HTML5 code using a tool such as PhoneGap or Intel's XDK. It's also possible to create hybrid apps, written in HTML5 with some native code integrated. Another concern for developers is the perceived inability of Web apps to integrate deeply into mobile device hardware. It's not the case that creating an HTML5-based Web app necessarily denies access to particular device features, but that access may be different from device to device. In recognition of this issue, the W3C System Applications Working Group is working to develop standards and methodologies for accessing such typical device features as GPS, accelerometers, Bluetooth and NFC (Near Field Communication).

One of the other issues in developing EPUB 3 is the degree of support in e-readers for the EPUB 3 specification. Not all features are supported in all e-readers, so that developers need to proceed cautiously in implementing advanced features such as scripted interactions or the recently developed fixed-layout option. It's important to provide adequate fallbacks for features not supported. O'Reilly Media, one of the first publishers to move to EPUB 3 for its publications, offers concrete examples of how that can be achieved. The Book Industry Support Group offers a helpful EPUB 3 support grid for developers. Currently the desktop e-readers which support the greatest number of features are iBooks (Mac), Azardi (Linux, Mac, Windows), and Readium, an extension of the Chrome browser (Linux, Mac, Windows). For mobile devices, iBooks (iOS) and Helicon Books EPUB3 reader (Android) are recommended.

At VCU this year I am serving on a university e-text task force, and I suspect many institutions have such groups operating, looking at textbook affordability, the open access e-text movement, and the integration of electronic texts into the rest of the university's digital infrastructure, especially its LMS. In fact, a number of U.S. universities have engaged in pilot e-text initiatives, including a major collaborative project sponsored by Educause and Internet2. That project is using Courseload for delivery of e-texts, which formats commercial textbooks with proprietary DRM which is then distributed to authenticated users from its servers. Similar services are available from VitalSource and Inkling. The project has gotten mixed reviews, with a major problem being that Courseload did not until recently support use of portable devices. The e-text distribution services currently in use in the U.S. tend to start with HTML5 or EPUB 3 content, then add proprietary elements. Inkling, for example, uses its own markup system, S9ML, for interactive elements.

Many faculty members are seeking more open options for implementing e-texts and are looking to freely available e-texts from sources such as Wikibooks or the Internet Archive. This bypasses the need for any server authentication handshake, and makes it more likely that a wide number of devices and environments can be supported. A number of open access language e-texts are available (see reference list), which range from full textbooks to lessons or simple activities. As is always the case with open education resources (OER), it's not always easy to find appropriate materials and, if found, they may not meet specific needs. Usually OER come with a Creative Commons share-alike license, enabling content to be tailored to need and to be combined with other materials, not something possible with traditional commercial textbooks, even if they are available in a digital format. Another concern in OER materials is quality. Examining the language e-texts available from OER Commons, for example, shows a wide range of quality and scope. Using repositories which encourage peer review of content submissions, such as Merlot, can be helpful in finding reliable content. For course-related open materials, the Open Courseware Consortium is a good resource. There is likely to be considerably more activity in the near future in this area, given U.S. federal funding for OER development for community colleges as well as state initiatives in California, Florida, and Utah and in the Canadian province of British Columbia. South Korea is transitioning to use electronic books exclusively in its schools by 2015.

The long-sought developer nirvana is "write once, run anywhere," enabling true interoperability and transparent computing. It's too early to say whether HTML5 will fit this bill. Not long ago, it seemed that Java would hold the keys to the kingdom, but that has not panned out, due to problems like slow performance, memory issues, and significant platform and plug-in differences. Java applets running in a browser also, like Flash, require a plug-in. In contrast, HTML5 is the native environment of the Web browser and, unlike Java, does not need to be compiled into an executable program. While there are plenty of arguments in favor of open standards, there are also powerful forces working in the opposite direction. Many companies have a strategic and financial interest in exclusivity and in preventing interoperability. Both Apple and Amazon, for example, use proprietary versions of EPUB 3 for their e-books. For educational institutions, interoperability through open standards is crucial, so that the ability to change institutional software platforms (such as the LMS) is maintained. HTML5 development is also less costly with a shallower learning curve, important considerations in educational settings. Additionally, HTML5 is the most likely development platform to be compatible with future OS's or devices, such as the upcoming Firefox OS or Tizen. No one can see into the future, but it does seem a reasonable bet that the mobile environment will continue to be fragmented, hence the added importance of interoperable standards in developing learning materials to be accessed from tablets and phones.

REFERENCES

Garrish, M. and Gylling, M. (2013). *EPUB 3 best practices: Optimize your digital books.* Sebastopol, CA: O'Reilly Media.

Godwin-Jones, R. (2011). Mobile apps for language learning *Language Learning & Technology 15*(2), 2–11. Retrieved from http://llt.msu.edu/issues/june2011/emerging.pdf

RESOURCE LIST

HTML5 Info

- HTML 5 Official specs from W3C
- HTML5 Introduction - What is HTML5 Capable of From WebDesigner
- HTML 5 Doctor Helpful site on HTML 5 compatibility issues

HTML5 Demos

- HTML 5 Demos and Examples Includes browser support info
- HTML5 Website Showcase: 48 Potential Flash-Killing Demos HTML5 Canvas demos By Kevin Roast
- 9 Mind-Blowing Canvas Demos
- Gaming your way to language learning Example of an HTML5 language learning game (from Shai Shapira)
- HTML5 Hackathon - building a Language Learning App YouTube video
- BlahBlahLearning takes the winning prize at Onswipe's HTML5 Hackathon

Mobile Development: HTML5 versus Native Apps

- HTML5 rocketing in popularity, finds Developer Economics report From Telefónica Digital
- Debunking five big HTML5 myths From Telefónica Digital
- Intel commits to "fabulous" HTML5 From PCPro
- Developing a Cross-Platform HTML5 Offline App From Grinning Gecko
- To HTML5 or not to HTML5, that is the mobile question From Webdesignerdepot
- HTML5 Vs. Native Mobile Apps: Myths and Misconceptions From Forbes
- Here's why HTML-based apps don't work Argument in the developer debate
- Native vs. HTML5 -- looked at objectively, the debate is over Another argument in favor of native apps

- Best Of Both Worlds: Mixing HTML5 And Native Code

EPUB 3 Info

- EPUB 3 Overview From International Digital Publishing Forum
- EPUB 3 links Nice list from epubtest.com
- PIGS, GOURDS, AND WIKIS Liz Castro's blog, excellent source for information
- EPUB Resources and Guides From O'Reilly Media
- Digital Publishing and the Web From A List Apart

E-reader Applications with Support for EPUB 3

- Comparison of e-book readers Wikipedia
- EPUB 3 Support Grid Good reference for updated information on e-readers
- iBooks Mac only
- Readium Plug-in Google Chrome browser (Linux, OS X, Windows)
- Azardi Linux, Mac, Windows
- Calibre Linux, Mac, Windows, clunky interface & limited EPUB 3, but able to convert from and to many formats

Mobile E-reader Apps with Support for EPUB 3

- Apple iBooks iOS
- Helicon Books EPUB3 reader Android

Sample E-books

- EPUB 3 Sample Documents
- Sample ePub 3 books From Azardi
- EPUB 3 Samples From Infogrid Pacific
- EPUB3 with embedded QTI assessments
- eBooks: A Language Learner's Best From Langology
- Language Learning eBooks
- Two examples of EPUB3 Manga
- Dante's Comedy as EPUB 3 Sample w. Media Overlays From Smuuks
- 300 Enhanced eBooks Published for Language Learning From the Samsung Learning Hub
- Language e-books List for over 400 languages

EPUB 3 Development

- Epub Validator From idpf
- EPUB 3 Best Practices Book from O'Reilly Media
- Create Your Own eBooks Review of different tools by Richard Byrne
- Build a digital book with EPUB How to create an epub 3 by hand
- Adobe InDesign Windows/Mac
- Apple iWork Pages Mac only
- oXygen XML Editor Windows/Mac/Linux

EPUB 3 Language Info

- Requirements for Japanese Text Layout W3C Working Group
- Richer Internationalization for eBooks Workshop from W3C
- Suprasegmental Phonological representation of Tone in Kikuyu Example of ruby markup

- What are situations with western languages where you'd use HTML 5's Ruby element? From stackoverflow

Acessibility Info

- Sample accessible (test) books in EPUB 3 format LIA Project
- Accessibility and eBooks: International Endorsement
- Tips for Creating Accessible EPUB 3 Files DIAGRAM Center website
- EPUB 3: Accessibility Guidelines IDPF
- Accessible EPUB 3 free publication on O'Reilly site
- EPUB 3 Accessibility forum IDPF website