Developing Web-based English Learning Applications: Principles and Practice

Paul Raine (paul.raine@gmail.com)
J. F. Oberlin University, Tokyo, Japan

Abstract

In the 21st century, there is a near ubiquity of web-connected devices amongst language learners, and the considerable success of mass market web-based language learning applications shows a strong demand for such tools. Where does this leave EFL educators wanting to tap into the global trend, and create their own innovative web applications for learners of English? Having established the global demand for web-based digital English learning tools, this paper discusses the platforms and languages that can be used by English educators themselves to create new online learning activities. The recent development of JavaScript as both a client-side and server-side language is discussed, and the possibility of integrating language learning web-apps with a range of powerful Application Programming Interfaces (APIs) is highlighted. "Lyric Learner", a web applications developed by the author, is briefly examined, and a list of resources for those interested in learning to code is provided. In the second part of the paper, a range of theoretical underpinnings for language learning applications are considered, including structuralist, communicative, and interactional viewpoints. The argument is maintained that engaging and effective web applications can be created under each of these approaches to language pedagogy. The paper concludes with an invitation to English language educators to create their own web applications using sound theoretical principles and technological practices.

Keywords*:* Computer Assisted Language Learning, Mobile Learning, Online Teaching & Learning, Web-Based Instruction

INTRODUCTION

The practice of English language teaching and learning has been greatly affected by the growth of the Internet since the early 1990s, and the subsequent explosion in the number of available mobile devices since the early 2000s. As of 2017, there are an estimated 3.8 billion global internet users (Internet World Stats, 2017), and 2.3 billion smartphone users (Statistica, 2018). China, the US, South Korea, Japan, and Brazil account for 65% of a $2.8 billion global demand for digital English learning products (Adkins, 2016). Web-based Language Learning (WBLL) tools and resources are used autonomously by millions of English language learners, but are also prescribed by their teachers as part of a wide range of motivational and effective blended learning curricula (e.g. Bañados, 2006; Miyazoe & Anderson, 2010; Shih, 2011).

There is now a range of very successful web-based language learning platforms, including the likes of Duolingo with 150 million users (Guliani, 2016), and Busuu with 70 million users (Salter, 2017). However, not every learner need can be satisfied by these mass market solutions. Sometimes learners require fine-tuned practice on specific aspects of language, or need to engage extensively with bespoke language content. Furthermore, grassroots technological innovation should be encouraged, not just as a breeding ground for the next great language learning tool or resource, but also as an avenue for experimentation and research.

When it comes to designing or developing web applications, both practical and theoretical considerations need to be addressed. Practical matters include questions of which programming languages to use, and how to distribute the applications once they are ready to go live. Applications also need to be supported by sound pedagogical foundations, and informed by relevant theories of language learning. In this paper, structuralist, communicative, and interactional theories of language learning are discussed and evaluated with respect to their relevance to web applications. It is maintained that engaging and effective web applications can be created under all three approaches.

PROGRAMMING LANGUAGES AND PLATFORMS

Web-apps versus native apps

When the iPhone was launched in 2007, it included just a small range of native apps, and it wasn't possible for 3rd party developers to create their own native apps for iOS. It was Steve Jobs' original intention for additional apps to be available via iOS Safari in the form of web-apps (Jobs, 2007). Web-apps did not catch on at this time, and Apple quickly changed tack by launching a Software Development Kit (SDK) specifically for iOS. Native apps became popular with both developers and users, and there are now over 2.2 million native apps in the App Store for iOS devices (Statistica, 2017).

However, the trend is now starting to swing back toward browser based web-apps, with some predicting that most apps will be delivered through the browser in the near future (Durkin, 2016). The differences that once existed between native apps and web-apps, such as the range of device functions they could access, and their speed and reliability, are no longer so apparent. Cumulative additions and improvements to HTML5 and JavaScript have leveled the playing field in many respects, and it's now possible to achieve "almost anything we want to accomplish" via applications that run in web browsers (Durkin, 2016).

Another major advantage of web-apps over native apps is that they are device agnostic, i.e. they run on any device (mobile or desktop) with a modern web-browser, such as Google Chrome, Mozilla Firefox, or Safari. Whereas native apps usually need to be rewritten for each target device, which is a time consuming and costly process, web-apps can be written once and distributed to any platform with a modern browser. In addition, the programming languages required to create web-apps are typically “more familiar and easier-to-learn” (Godwin-Jones, 2011, p.5) than those required for native apps. This means that individuals who already have a basic knowledge of HTML, CSS and JavaScript can get started making web-apps relatively easily.

The death of Adobe Flash

Adobe Flash was once the predominant technology of interactive websites, and powered well-known language learning sites such as Livemocha (Clark & Gruba, 2010), which is now defunct (EdSurge, 2016). Almost 50% of sites utilized Flash in 2011, but usage drastically dropped to less than 10% in 2016 (Richter, 2016). In 2010, Flash was banned from the iOS platform by Steve Jobs due to being unreliable, inefficient, and proprietary (Jobs, 2010), and in 2017 Adobe themselves announced their intention to discontinue support for the technology by 2020 (Adobe, 2017).

The rise of JavaScript

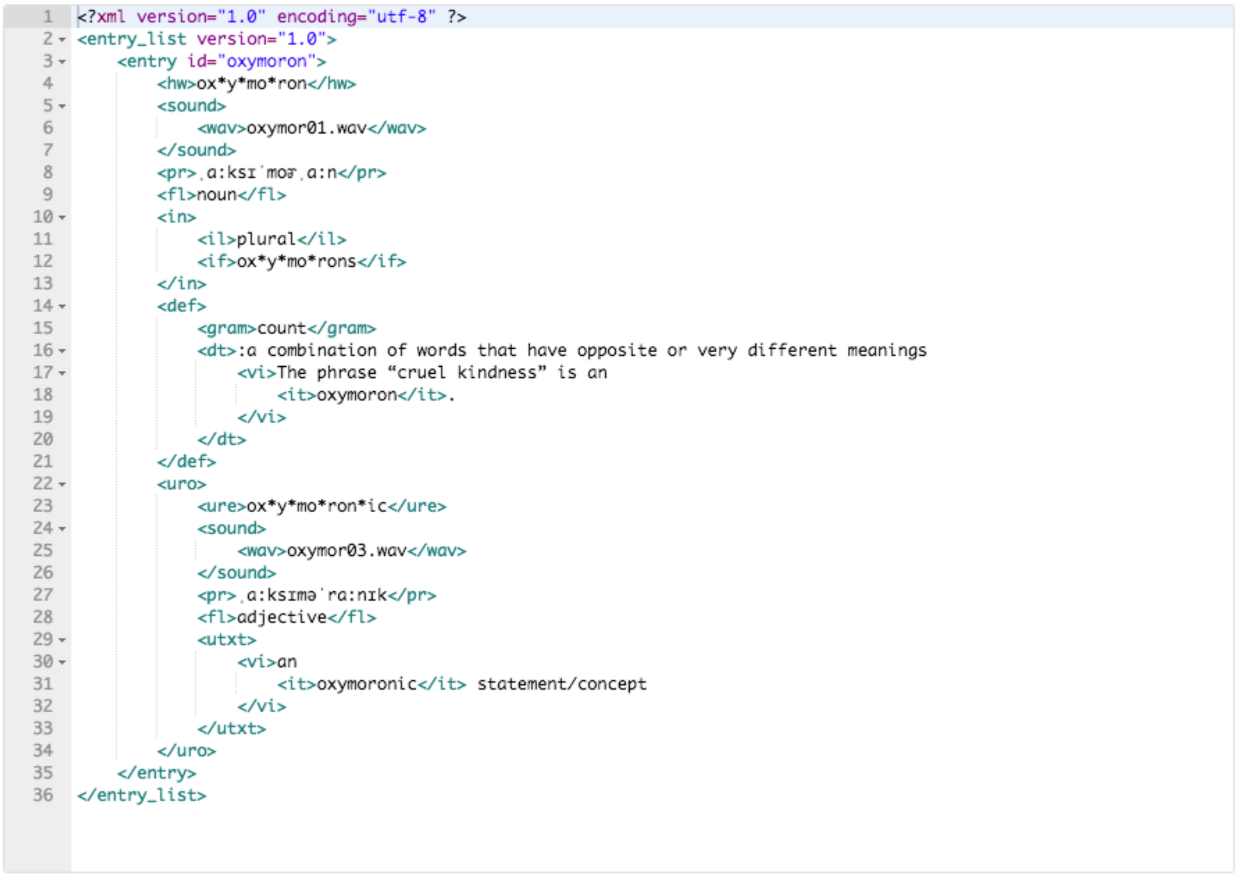
Where Flash lost out, JavaScript has gained ground in a remarkable way. Once purely a client-side scripting language (see below), it can now be deployed both in the browser and on the server by utilizing open source server frameworks such as Node.js (Heller, 2017). It can even be used to create native desktop and mobile apps, using frameworks such as Electron and Cordova, respectively. The author has used Node.js to create real time web-based vocabulary learning activities (Ono & Raine, 2016), and there are many other exciting potential applications of this technology for language pedagogy purposes.

The importance of APIs

An Application Programming Interface (API) is essentially a way to “plug your website into another” (Kiss, 2007), and allows much more sophisticated applications to be built more quickly and easily. APIs allow web applications to send data to servers provided by third parties. The servers then process the application’s data in some way before returning the results. There are over 12,000 APIs available via the web (Iyer & Subramaniam, 2015), including many that provide data useful for language teaching or learning. Macmillan, Oxford, and Merriam-Webster, for example, all offer access to their dictionary data via API, although they require you to sign up for an API key, and implement limits on how many requests your application can make within a certain timeframe.

If an application requires a dictionary definition, it can send the relevant headword to the Merriam-Webster dictionary API, which will then return the matching definitions (*Figure 1*).

Another example of an API which provides data suitable for language learning is the digital flashcard site Quizlet, which allows access to billions of term/definition pairs. Sets of these digital flashcards can be retrieved with either an ID number, or by searching through set names for specific keywords or phrases.



```xml
<?xml version="1.0" encoding="utf-8" ?>
<entry_list version="1.0">
    <entry id="oxymoron">
        <hw>ox*y*mo*ron</hw>
        <sound>
            <wav>oxymor01.wav</wav>
        </sound>
        <pr>ˌɑːksɪˈmoɚˌɑːn</pr>
        <fl>noun</fl>
        <in>
            <il>plural</il>
            <if>ox*y*mo*rons</if>
        </in>
        <def>
            <gram>count</gram>
            <dt>:a combination of words that have opposite or very different meanings
                <vi>The phrase “cruel kindness” is an
                    <it>oxymoron</it>.
                </vi>
            </dt>
        </def>
        <uro>
            <ure>ox*y*mo*ron*ic</ure>
            <sound>
                <wav>oxymor03.wav</wav>
            </sound>
            <pr>ˌɑːksɪməˈrɑːnɪk</pr>
            <fl>adjective</fl>
            <utxt>
                <vi>an
                    <it>oxymoronic</it> statement/concept
                </vi>
            </utxt>
        </uro>
    </entry>
</entry_list>
```

Figure 1: XML data returned by the Merriam-Webster dictionary API (dictionaryapi.com) for the headword “oxymoron”

The latest advances in Natural Language Processing (NLP), text-to-speech (TTS), and automatic speech recognition (ASR) are also accessible via API. Google Cloud offers advanced NLP functions via its Cloud Natural Language service, and speech recognition via its Cloud Speech API. The suitability of cloud-based ASR services for language learning has been recently examined (Daniels & Iwago, 2017) and Google’s Cloud Speech API was found particularly useful for administering online speaking tasks which allow for automated scoring and feedback. In relation to text-to-speech, a free TTS service that works on any device is available from Responsive Voice. Several studies have shown the utility of TTS for language learning and teaching purposes (e.g. Gonzales, 2007; Handley, 2009; Pellegrini, Costa & Trancoso, 2012).

Server-side and client-side methodologies

When it comes to developing applications for the web, there are two basic methodologies that must be understood: server-side and client-side. In server-side methodologies, the application which processes the data runs on the server using languages such as PHP, ASP, and, more recently, JavaScript. In client-side methodologies, the application runs within the browser using JavaScript. Many web-based applications use a combination of both server and client side methodologies.

When a user opens their web browser (the "client") and requests a web page via the Hypertext Transfer Protocol (HTTP) by typing in a Uniform Resource Locator (URL), the server delivers the page as a combination of resources, including HTML files, JavaScript files, CSS (cascading style sheet) files, image files, and audio files. (*Figure 2*).

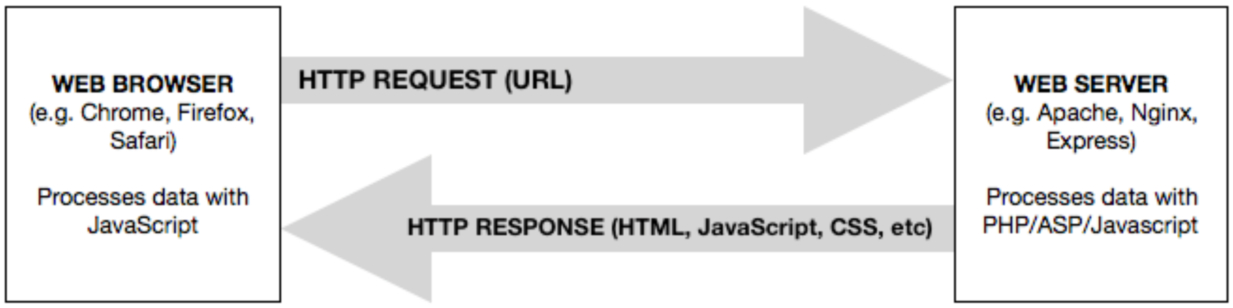

WEB BROWSER
(e.g. Chrome, Firefox, Safari)
Processes data with JavaScript
HTTP REQUEST (URL)
HTTP RESPONSE (HTML, JavaScript, CSS, etc)
WEB SERVER
(e.g. Apache, Nginx, Express)
Processes data with PHP/ASP/Javascript

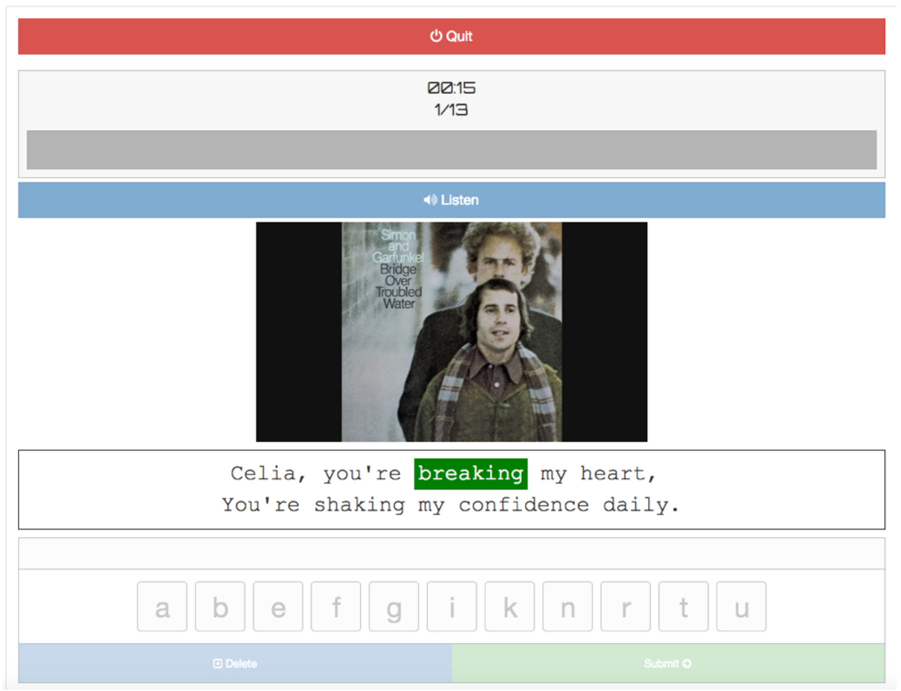
Figure 2: HTTP request and response process between client and server

In general, HTML files define the structure of the web page, CSS files define the appearance, and JavaScript files define the logic. Most web pages also use images and sound to provide a rich user interface. A web application is a specific kind of web page that provides a similar functionality and user experience to traditional desktop applications.

Web application example: Lyric Learner

An example of an English learning web-app developed by the author, which uses both client-side and server side methodologies, in addition to interacting with a database and APIs, is Lyric Learner. This app allows English learners to listen to and learn from a variety of different music videos. It requires users to input missing words using an onscreen keyboard that contains all the letters of the word in a scrambled order, in addition to some extra ("distractor") letters (*Figure 3*). Its functionality is therefore similar to that provided by traditional cloze activities (e.g. Oiler & Conrad, 1971).

Lyric Learner uses the YouTube IFrame Player API to manipulate the video content, and retrieves song lyrics from a database using server-side scripting. It then uses client-side scripting to create and position all the user interface (UI) elements in the browser. Lyric Learner is just one example of the kind of web-based pedagogical tool that can be created for English language learners, and a huge number of other ideas are possible with a combination of client-side, server-side, and API-based methodologies.



Celia, you're breaking my heart,
You're shaking my confidence daily.

Figure 3: The "Lyric Learner" web application

Learning to code

There are numerous free online resources available for those who wish to learn how to code server-side or client-side web applications. The following resources are recommended by the author:

- Learn Code the Hard Way
- Team Tree House
- Lrn
- Code School
- W3 Schools
- Stack Overflow

In terms of how long it takes to learn to code, some have suggested that the investment of time is comparable to that of acquiring conversational competence in a second language, i.e. about 500-700 hours (Orosz, 2015). However, the wide range of libraries (e.g. JQuery), frameworks (e.g. Bootstrap), APIs, and online support options are increasingly making complex and sophisticated applications much easier and quicker to develop and deploy.

Having selected an appropriate platform and programming language for application development, coder-educators should carefully consider applicable language learning theories, both in the design and implementation of their tools. The next part of this paper discusses three major theoretical underpinnings to language learning application design.

THEORETICAL UNDERPINNINGS OF LANGUAGE LEARNING APPLICATIONS

When it comes to adopting sound pedagogical principles in designing CALL applications, it is useful to conceptualize three distinct views of language learning: the structuralist view, the communicative view, and the interactional view (Warschauer & Healey, 1998; Blake, 2013). Depending on the view adopted, the focus of the pedagogical endeavor changes. Although this trifurcated analysis of CALL paradigms has been criticized by some (e.g. Bax, 2002), it remains useful for understanding the different kinds of applications that can be created, and their effect on the user's language knowledge and skill.

From a structuralist viewpoint, the focus is on breaking down utterances into their constituent parts (phonemes, lexemes, clauses, etc.), and obtaining mastery over these parts in order to construct original utterances. In a communicative view of language, the focus is on conveying authentic meaning, obtaining mastery over the functions of language (apologizing, describing, inviting, etc.), and expressing oneself fluently without becoming too preoccupied with accuracy. In an interactional view of language, the focus is on building and maintaining social relations with one's interlocutors, and using appropriate forms of language according to different kinds of social situations.

Each of these views of language learning came in and out of vogue over the course of the 20th century. Some claim that the interactional view, based on a sociocultural theory of language learning, is the most appropriate and efficacious for 21st century CALL applications (e.g. Blake, 2013). However, in this paper the argument is maintained that pedagogically motivational and effective applications can be created under all three views, each of which is discussed below.

Structuralist applications

From a programmatic perspective, it's a relatively simple endeavor to produce applications which focus on breaking down and reconstructing the elements of language. Natural Language Processing (NLP) libraries, such as the Natural Language Toolkit and Google Cloud Natural Language, are very useful for this function. An example of a structuralist web-app created by the author is Auto Cloze (*Figure 4*). Auto Cloze uses a part-of-speech (POS) tagger to identify, to an accuracy of over 96% (Giesbrecht & Evert, 2009), the POS categorizations of the words appearing in the selected text. It then removes the words which match the POS category selected by the user, and the user must reconstruct the text by choosing the correct word from a dropdown menu containing a list of words of the same type.

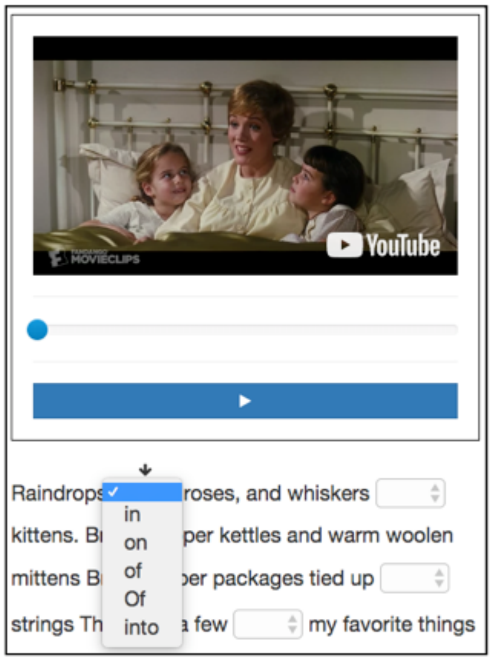

YouTube
MOVIECLIPS
Raindrops ✓ roses, and whiskers
kittens. B per kettles and warm woolen
mittens B per packages tied up
strings Th a few my favorite things
in
on
of
Of
into

Figure 4: The “Auto Cloze” web application

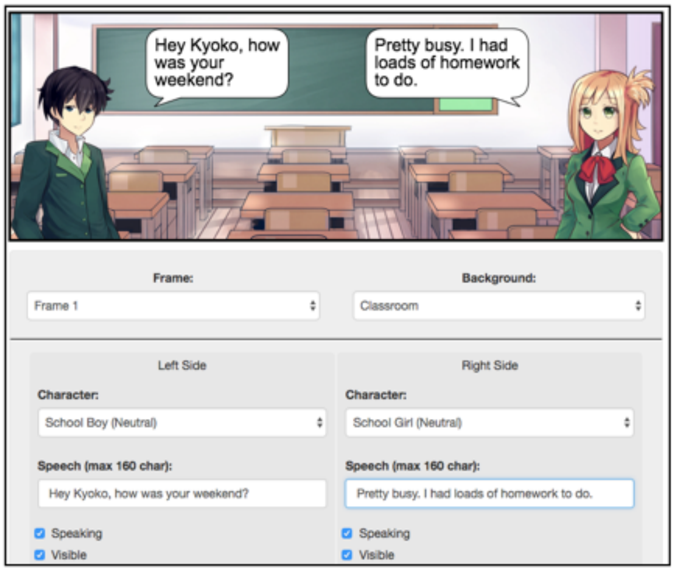
The Auto Cloze web-app is further enhanced by integrating with the English Wikipedia via the MediaWiki API and also with captioned YouTube videos via the YouTube Data API. This allows users to select their own texts or videos from millions of choices, and thereby increases learner autonomy.

Blake (2013, p.40) has argued that although form-focused ‘drill-and-practice’ type activities “should not constitute the driving concept behind a Web-based L2 curriculum”, they are nevertheless useful for students to acquire knowledge of new morphology and syntax. Others have highlighted the fact that computers are simply much more efficient than humans at generating structuralist type activities, thus making such activities readily available for deployment in the classroom without the need for extensive planning or preparation (Warschauer & Healey, 1998).

The fact that the hugely popular Duolingo employs a variety of manipulative linguistic activities implies that such activities are in demand by language learners. Although we should not mistake the popularity of an approach for its efficacy, independent research on Duolingo’s methods seems to suggest that such activities are both in demand and effective (Vesselinov & Grego, 2012). Again, there are pragmatic reasons for adopting a structural approach for a mass market language learning website, including the fact that it’s a relatively trivial task to get a computer to chop up a sentence into its constituent parts to be presented to the learner for rearrangement. Users' interactions with structuralist type applications can also be easily tracked and scored, with appropriate motivational or correctional feedback being displayed (Heift, 2004; Carey, 2004).

Communicative applications

Creating web applications that fulfill the precepts of the communicative paradigm requires a little more imagination and effort. One example created by the author is an app called Manga Maker (*Figure 5*). Manga Maker allows the user to create a dialogue between two characters, whose expressions and gestures can be changed. A choice of background environments is also provided. This application allows learners to practice using language in a functional-notional sense (Finocchiaro & Brumfit, 1983), by first conceiving of the situation the two characters are in, and then generating suitable utterances for that situation. Follow up activities can include having the students print out their dialogues and then rehearse them together.


Hey Kyoko, how was your weekend?
Pretty busy. I had loads of homework to do.
Frame:
Background:
Left Side
Right Side
Speaking
Visible

Frame: Frame 1
Background: Classroom

Left Side
Character: School Boy (Neutral)
Speech (max 160 char): Hey Kyoko, how was your weekend?

Right Side
Character: School Girl (Neutral)
Speech (max 160 char): Pretty busy. I had loads of homework to do.

Figure 5: The "Manga Maker" web-app

Communicative CALL activities such as Manga Maker clearly go beyond simple manipulation of basic grammar forms. In requiring students to generate their own original conversations, they need an understanding of how the relationship between the characters, and the situation they are in, affects both what they say to each other and how they say it. Although many students struggle to make dialogues realistic initially, this problem is also faced by professional textbook writers (Nguyen & Ishitobi, 2012) and is therefore to be expected. Unnatural sounding conversations can be checked by the teacher and revised by the students and thereby capitalized on as an additional learning opportunity.

Interactional applications

The interactional view emphasizes the importance of developing sociocultural relationships with one’s interlocutors. In a CALL setting, this usually means utilizing the affordances of technology to bring about some form of Computer Mediated Communication (CMC) between learners and teachers, or between learners themselves. The mode of CMC can either be synchronous (SCMC), where communication occurs in ‘real time’, or asynchronous (ACMC), where there is a delay before receiving a response. An early example of SCMC on the Internet is Internet Relay Chat (IRC) which dates back to 1988 (Stenberg, 2011) and allows language learners to interact with others who are in the same ‘virtual space’ (Healey, 2016). IRC has largely been superseded by more technologically advanced forms of SCMC, which feature both audio and video transmission. Modern examples include the likes of Skype, Facebook Messenger, and Google Hangouts. ACMC, on the other hand, is largely associated with e-mail (e.g. Gmail, Hotmail, Yahoo! Mail), bulletin boards (e.g. phpBB), and other tools which store messages on a server for later retrieval.

In general, SCMC applications are more difficult to develop than ACMC applications, especially when it comes to transmitting live audio and video, which has significant implications for memory and bandwidth. In recent years, however, the technology has become available to achieve full SCMC functionality via the web browser, using a set of capabilities provided by WebRTC. The rise in real time live language tutoring services such as Okpanda and YourTutor has been powered by WebRTC technology. The prevalence of browser-based SCMC is expected to continue to increase over the next few years, with over 2 billion potential users predicted by 2019 (Lumiaho, 2015).

The author has experimented with SCMC functionality in an application that allowed learners to chat with each other in real time (Ono & Raine, 2016), but the tool lacked a suitable pedagogical framework and was not developed further at that time. It would be fruitful to pursue the development an SCMC application specifically for language learners that followed a principled framework (e.g. Hampel, 2006) that allows for a variety of multimodal communication channels, including audio, video, and virtual whiteboards.

CMC is not the only possible way to use technology to deliver interactional language learning experiences to the user. Rapid technological improvements in Artificial Intelligence (AI), powered by Artificial Neural Networks (ANN), has made an increasing array of artificial interlocutors available to learners. The best known among these are Apple’s Siri, Google Assistant, Amazon’s Alexa, and Microsoft’s Cortana. Although none of these virtual assistants are designed specifically for teaching a language, they are able to have basic conversational interactions about an increasing array of topics, and have been used in English language classrooms with promising results (e.g. Moussalli & Cardoso, 2016; Underwood, 2017). There has yet to emerge a fully conversational virtual assistant with a specific remit for teaching English as a second language. However, simpler “chat-bots” have been available online for many years. Rong Chang’s award-winning EFL chat-bot Tutor Mike, and general purpose chat-bots Jabberwacky and ALICE have been employed in EFL classrooms with positive results (e.g. Jia, 2004; Fryer & Carpenter, 2006; AlKhayat, 2017), although they are based on deprecated Flash technology, and are therefore incompatible with mobile devices.

CONCLUSION

There is a clear and increasing demand for web-based English language learning tools, and educators may soon be expected to be able to design and create such tools in addition to recommending and prescribing them to their students.

In terms of theoretical underpinnings, the three main categories of CALL applications originally delineated by Warschauer & Healey (1998) still hold true for a wide range of currently available and future possible tools, notwithstanding revisions suggested by Bax (2002). However, the rapid development of Artificial Intelligence (AI) in recent years may soon require whole new ways of thinking about CALL applications, and the language learning theories that underpin them.

Small scale innovation and experimentation can lead to world-changing breakthroughs. Even on the occasions that it does not, it is nevertheless a valuable learning experience, and can provide avenues for further research and experimentation. Individuals with some knowledge of HTML and CSS can easily begin making web applications, and the vast range of libraries, frameworks, and APIs available make sophisticated and powerful features available to coder-educators.

There is a wide range of free resources for learning to code, and although the investment of time and energy required to do so is significant, the rewards are worthwhile. The next generation of web-based CALL applications should be based on proven principles and innovative practices. Who better to be involved in the design and development process than language teachers themselves?

ACKNOWLEDGEMENTS

The author gratefully acknowledges the helpful comments and suggestions of Todd Cooper and Brian Gallagher.

REFERENCES

Adobe. (2017). Flash & The Future of Interactive Content. Retrieved from https://theblog.adobe.com/adobe-flash-update/

Adkins, S. S. (2016). The 2015-2020 Worldwide Digital English Language Learning Market. Retrieved from http://www.ambientinsight.com/Resources/Documents/AmbientInsight_2015-2020_Worldwide_Digital_English_Market_Sample.pdf

AlKhayat, A. (2017). Exploring The Effectiveness of Using Chatbots in the EFL classroom. In *Teaching English Reflectively with Technology,* 20-36.

Bañados, E. (2006). A blended-learning pedagogical model for teaching and learning EFL successfully through an online interactive multimedia environment. *Calico Journal*, *23*(3), 533-550.

Bax, S. (2003). CALL—past, present and future. *System*, *31*(1), 13-28.

Blake, R. J. (2013). *Brave new digital classroom: Technology and foreign language learning*. Georgetown University Press.

Carey, M. (2004). CALL visual feedback for pronunciation of vowels: Kay Sona-Match. *CALICO Journal*, *21*(3), 571-601.

Clark, C. & Gruba, P. (2010). The use of social networking sites for foreign language learning: An autoethnographic study of Livemocha. In C.H. Steel, M.J. Keppell, P. Gerbic & S. Housego (Eds.), *Curriculum, technology & transformation for an unknown future. Proceedings ascilite Sydney 2010*, 164-173.

Daniels, P., & Iwago, K. (2017) The suitability of cloud-based speech recognition engines for language learning. *JALT CALL Journal*, *13*(3), 229-239.

Durkin, H. (2016). Browsers, not apps, are the future of mobile. Retrieved from https://medium.com/swlh/browsers-not-apps-are-the-future-of-mobile-c552752ff75

EdSurge. (2016). Rosetta Stone to Shut Down Language Learning Service, Livemocha. Retrieved from https://www.edsurge.com/news/2016-03-23-rosetta-stone-to-shut-down-language-learning-service-livemocha

Finocchiaro, M., & Brumfit, C. (1983). *The functional-notional approach: From theory to practice*. Oxford University Press.

Fryer, L. K., & Carpenter, R. (2006). Bots as language learning tools. *Language Learning & Technology*, *10*(3), 8-14.

Giesbrecht, E., & Evert, S. (2009). Is part-of-speech tagging a solved task? An evaluation of POS taggers for the German web as corpus. In *Proceedings of the fifth Web as Corpus workshop*, 27-35.

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

Gonzales, D. (2007). Text-to-Speech Applications Used in EFL Contexts to Enhance Pronunciation. *TESL-EJ, 11*(2). Retrieved from http://tesl-ej.org/ej42/int.html.bu2

Guliani, P. (2016). Duolingo Looks to Dominate the Mobile Education Market with New Flashcard App TinyCards. Retrieved from https://www.forbes.com/sites/parulguliani/2016/07/22/duolingo-looks-to-dominate-the-mobile-education-market-with-new-flashcard-app

Handley, Z. (2009). Is text-to-speech synthesis ready for use in computer-assisted language learning? *Speech Communication, 51*(10), 906-919.

Healey, D. (2016). Language learning and technology: Past, present and future. In F. Farr & L. Murray (Eds.) *The Routledge handbook of language learning and technology*, (pp. 9-23). Routledge.

Heift, T. (2004). Corrective feedback and learner uptake in CALL. *ReCALL*, *16*(2), 416-431.

Heller, M. (2017). What is Node.js? The JavaScript runtime explained. Retrieved from https://www.infoworld.com/article/3210589/node-js/what-is-nodejs-javascript-runtime-explained.html

Hampel, R. (2006). Rethinking task design for the digital age: A framework for language teaching and learning in a synchronous online environment. *ReCALL*, *18*(1), 105-121.

Internet World Stats. (2017). Internet Usage Statistics. Retrieved from http://www.internetworldstats.com/stats.htm

Iyer, B., & Subramaniam, M. (2015). The Strategic Value of APIs. Retrieved from https://hbr.org/2015/01/the-strategic-value-of-apis

Jia, J. (2004). The study of the application of a web-based chat-bot system on the teaching of foreign languages. In *Society for Information Technology & Teacher Education International Conference*, 1201-1207.

Jobs, S. (2007). “Keynote Address.” Apple Worldwide Developers Conference, 11 June 2007, Moscone West, San Francisco, CA.

Jobs, S. (2010). Thoughts on Flash. Retrieved from https://www.apple.com/hotnews/thoughts-on-flash

Kiss, J. (2007). The Nutshell: A beginners' guide to APIs. Retrieved from https://www.theguardian.com/media/pda/2007/dec/14/thenutshellabeginnersguide

Lumiaho, L. (2015). Five Ways of Connecting People, Devices, and Systems with WebRTC. Retrieved from https://www.callstats.io/2015/06/05/five-ways-of-connecting-with-webrtc/

Miyazoe, T., & Anderson, T. (2010). Learning outcomes and students' perceptions of online writing: Simultaneous implementation of a forum, blog, and wiki in an EFL blended learning setting. *System*, *38*(2), 185-199.

Moussalli, S., & Cardoso, W. (2016). Are commercial ‘personal robots’ ready for language learning? Focus on second language speech. In S. Papadima-Sophocleous, L. Bradley & S. Thouësny (Eds.), *CALL communities and culture – short papers from EUROCALL 2016*, 325-329.

Nguyen, H. & Ishitobi, N. (2012). Ordering Fast Food: Service Encounters in Real-Life Interaction and in Textbook Dialogs. *JALT Journal, 34*(2), 151-186.

Oiler, J. W., & Conrad, C. A. (1971). The cloze technique and ESL proficiency. Language Learning, 21(2), 183-194.

Ono, T., & Raine, P. (2016). Apps 4 EFL: Synchronous Learning through Real Time. *Annual Report of JACET SIG on ESP*, *18*, 38-41.

Oroz, G. (2015). How long does it actually take to learn to code? Retrieved from http://gergelyorosz.com/2015/09/how-long-does-it-actually-take-to-learn-to-code/

Pellegrini, T., Costa, Â., & Trancoso, I. (2012). Less errors with TTS? A dictation experiment with foreign language learners. In *INTERSPEECH-2012*, 1291-1294.

Raine, P. (2017). Web-based language learning with creative commons data. *Accents Asia*, *9*(2), 41-55.

Richter, F. (2016). The Web Is Turning Its Back on Flash. Retrieved from https://www.statista.com/chart/3796/websites-using-flash/

Salter, P. (2017). Busuu is breaking down language barriers in nearly every country across the world. Retrieved from http://www.cityam.com/268678/busuu-breaking-down-language-barriers-nearly-every-country

Shih, R. C. (2011). Can Web 2.0 technology assist college students in learning English writing? Integrating Facebook and peer assessment with blended learning. *Australasian Journal of Educational Technology*, *27*(5), 829-845.

Statistica. (2017). Number of apps available in leading app stores as of March 2017. Retrieved from https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores.

Statistica. (2018). Number of smartphone users worldwide from 2014 to 2020 (in billions). Retrieved from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide

Stenberg, D. (2011). History of IRC (Internet Relay Chat). Retrieved from https://daniel.haxx.se/irchistory.html

Underwood, J. (2017). Exploring AI language assistants with primary EFL students. In K. Borthwick, L. Bradley & S. Thouësny (Eds.), *CALL in a climate of change: adapting to turbulent global conditions–short papers from EUROCALL 2017*, 317-321.

Vesselinov, R. & Grego, J. (2012). Duolingo Effectiveness Study. Retrieved from http://static.duolingo.com/s3/DuolingoReport_Final.pdf

Warschauer, M., & Healey, D. (1998). Computers and language learning: An overview. *Language Teaching*, *31*(2), 57-71.