David Wolber *University of San Francisco* **Harold Abelson** *Massachusetts Institute of Technology* **Mark Friedman** *Google Inc.*

Editors: Nilanjan Banerjee and Sami Rollins

DEMOCRATIZING COMPUTING WITH APP INVENTOR

MIT App Inventor is a visual blocks language that enables beginners and non-programmers to create apps for their phones and tablets. It has empowered thousands to create software with real-world usefulness, and see themselves as creators rather than only consumers in the mobile computing environment. Educationally, it offers a "gateway drug" that can help broaden and diversify participation in computing education.

We now carry our computing with us. Our social networks and interactions are increasingly computer-mediated. Online and offline life are becoming fused. This societal shift is having a profound effect on all walks of life and triggering excitement about computing not seen since the birth of the World Wide Web. Apps like Instagram and Flappy Birds pop up seemingly every day, creating instant phenomenons and instant millionaires.

A few years ago, we helped create a tool with the potential to open up this new mobile world to a broad audience, to empower people to not just use the new technology, but create it. The idea was to democratize app building and not leave it to the purview of a small group of technically elite. The tool, called App Inventor, runs in a browser and allows you to create apps by plugging blocks together in tinker-toy fashion, rather than by typing code. The apps you create are not toys or demonstrations, but full-fledged applications. You can share them, upload them to Google Play, and even sell them.

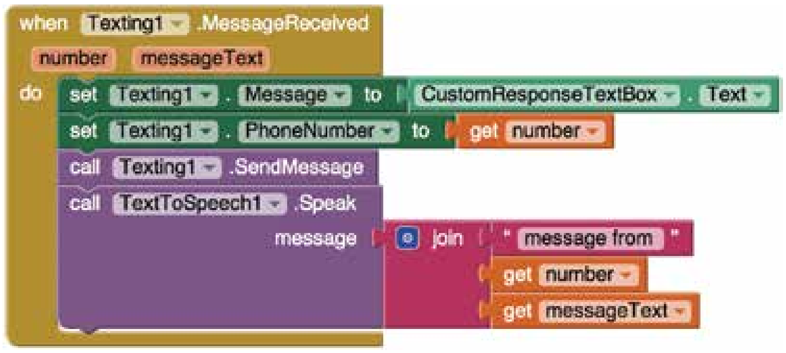
Because you don't need programming experience to use it, App Inventor has spawned thousands of new creators in the mobile space. Over a million people have tried it, together creating 4.7 million apps, with over 30,000 published on Google Play. The apps created (many of them by kids), have been reported on in numerous media outlets. One group of young people even demonstrated their App Inventor app to the First Lady of Nigeria!

We are still trying to measure and understand the impact of App Inventor. Hobbyists can use it to create personalized, situated apps just for themselves or their friends, like the man who wrote a Harry Potter app and on the last screen proposed to his wife (she said yes!). Designers and entrepreneurs can use it to rapidly create prototypes for their ideas or companies.

This article focuses mainly on its impact in education. App Inventor is taught mostly in beginning Computer courses but also in other disciplines like Math, Health, Design and Business. It is taught at numerous colleges and universities, in K-12, and it is being piloted for the new high school Advanced Placement (AP) course in Computer Science Principles (CSP). It is also extremely popular in the burgeoning after-school code camps like Technovation [10]

Coding is now being considered by some as an addition to the 3 Rs (arithmetic, reading, and writing). Code.org [5] has galvanized this movement and induced millions of students to participate in the December "Hour of Code" event, many of whom completed a highly motivating coding maze game based on the Blockly language [4] and "Angry Birds" [5]

These introductory experiences are terrific, but the computing is still something that happens on a computer display screen, not something the students use on their mobile devices in everyday life. With App Inventor, people can create technology that runs on the devices they use every day, and that directly affects their lives and the lives of their friends.



```
when Texting1 .MessageReceived
  number  messageText
do  set Texting1 . Message to CustomResponseTextBox . Text
    set Texting1 . PhoneNumber to get number
    call Texting1 .SendMessage
    call TextToSpeech1 .Speak
                    message  join  " message from "
                                   get number
                                   get messageText
```

FIGURE 1. App Inventor blocks for the "No Texting While Driving" app.

Our experience suggests that this type of power is appealing to a very broad group of young people, not just those who typically show up at a computer science classroom. We think this has implications for broadening participation in computer science, and improving the historically high attrition rate [15]. We also think App Inventor can have an even broader societal impact and give people a different understanding and perspective of the technology they use every day [11].

Snapshot: No Texting While Driving

Daniel Finnegan was an English major at the University of San Francisco in the fall of 2009. He was taking an app building class because he was a senior and needed to fulfill the university Math requirement. He had no coding experience and was not particularly motivated to learn about computing.

Early in the semester he happened to read an article on the 2009 rash of accidents caused by people texting while driving their cars. It occurred to him that the app building he was learning about in class might offer a potential solution. Because App Inventor has a high-level texting block, Daniel was able to get to work on his idea. The result was a fairly simple but extremely powerful app, and one that did not exist at the time. The blocks for Daniel's app are shown in Figure 1.

Can you tell what the app does? You turn the app on when you're driving and it auto-responds to any texts that come in. Daniel even made it so the app speaks the texts aloud, further reducing one's desire to pick the phone up!

You can study computer science for years before creating such a useful app, and here an English student in a first semester programming course had done it. Clive Thompson of *Wired* [17] picked up on the novelty and wrote this:

Software, after all, affects almost everything we do. Pick any major problem – global warming, health care, or, in Finnegan's case, highway safety – and clever software is part of the solution. Yet only a tiny chunk of people ever consider learning to write code, which means we're not tapping the creativity of a big chunk of society.

App Inventor is about tapping the creativity Thompson mentions, about opening up the world of software creation to everyone. The digital divide has been partially eliminated, but there is still a "programmer divide" which is blocking many from prospering in today's society. Not everyone needs to become a programmer, but gaining a fundamental under- standing of code, of how software works, is quickly becoming a core skill needed in today's workplace.

Snapshot: Community Service Clean-ups

Four high-school girls from East Palo Alto came together to form the "Chica Squad" and to participate in the "Technovation Challenge," an App Inventor app development contest sponsored by the non-profit educational company Iridescent [10]. Their app, called "Tag It!", allows users to take a picture of graffiti, vandalism or trash in their neighborhood, tag its location and create a community event to get it cleaned up. The girls won their regional contest and were invited to present their work before a group of local and state government officials at a conference on strengthening communities through social media – the only students to present.

Snapshot: Help for Traffic Law Enforcement

Princeton Girls in IT is a group of five high-school girls in Lagos, Nigeria who created a mobile app to help local traffic police. Rather than chasing after the offender, the officer submits information with the mobile app, and this is collected in a shared database that can be accessed by police throughout the country. The team was a winner of the International Telecommunications Union "Tech Needs Girls" award. They were honored at a reception hosted by Nigeria's First Lady at the Presidential Villa, as well as at an award presentation at the European Parliament in Brussels.

Snapshot: School Bus Tracking for Kids and Parents

A lot of kids ride the bus in India, and those buses are often delayed, causing enormous worries for parents. Arjun Kumar, a twelve-year old from Chennai, had an idea: how about an app that allows students to check-in when they get on the bus then tracks their whereabouts for parents?

People have such ideas all the time. But for most of us, these great ideas have a short shelf-life. We mention it to a friend, who says, "Isn't there already an app for that?" and the idea dies.

But Arjun was no ordinary twelve-year old. He had heard of App Inventor so he started playing around with his idea and built a prototype. When he showed his prototype to his friends and family they were amazed. How did you build that? His headmaster at school was astounded–he knew very well of the bus issues and parent's worries and here was a digital solution staring right at him! He asked Arjun to build the app out and run a pilot program for the upcoming school year.

The point of the story is not that Chennai will have a new bus tracking system – actually deploying software, especially something as complicated as a tracking system for a district's school buses, presents huge hurdles, both technically and socially. The takeaway here– to all of these snapshots– is that some kids with little or no coding experience were able to explore solving real-world problems with a mobile computing component, something that has, until recently, been restricted to an elite group of engineers.

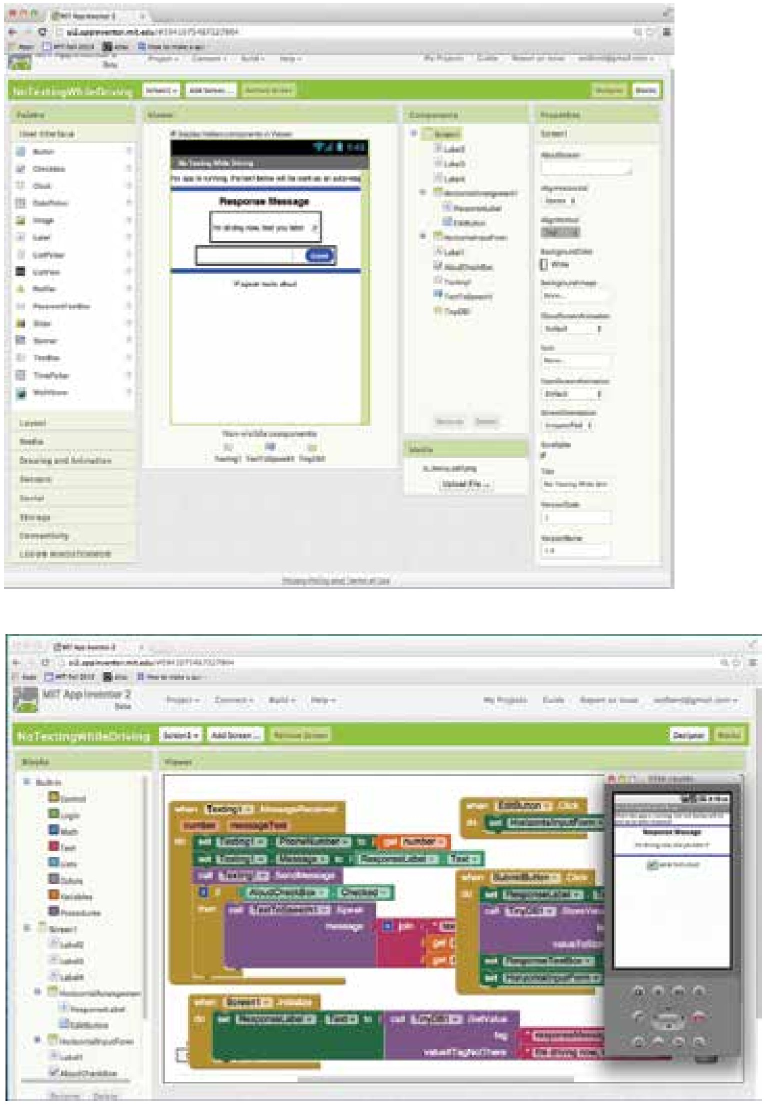


FIGURE 2. The App Inventor Environment: Designer, Blocks Editor, and Emulator

APP INVENTOR'S HISTORY AND ITS USE TODAY

The MIT App Inventor service [7] includes tutorials, sample curricula, news and events, and the service itself. As of October 2014, there are 87,000 people actively using the service each week. In total, there are 1.9 million registered users from 194 countries, who together have created (or at least started) 4.7 million apps. The App Inventor code base is open source and available on GitHub so that anyone can download it and mount their own App Inventor service.

App Inventor's origins trace to 2008 when Hal Abelson (author) taught an MIT class on mobile computing using the very first publicly available Android devices. That experience helped him appreciate the power of students building apps. The following year, on sabbatical at Google, he proposed creating a better tool to let even pre-college beginners build the same kind of creations his advanced MIT computer science students had done. He partnered with Google engineer Mark Friedman (author) to lead the development of the project, and together with primary developers Sharon

Perl, Liz Looney, Ellen Spertus, and Deborah Wallach they released the first pilot system in 2009. University of San Francisco (USF) professor David Wolber (author) was one of the teachers brave enough to teach an actual class using the incomplete earliest version of App Inventor, and he immediately recognized the motivating force that it was for computer science education.

Google continued to provide App Inventor as an experimental service until the end of 2011, when App Inventor moved to MIT under Abelson. The MIT App Inventor service began operating in March 2012.

HOW APP INVENTOR WORKS

The App Inventor environment is shown in Figure 2. The developer toggles between the Component Designer for developing the user interface (top of Figure 2) and a Blocks Editor for programming the behavior (bottom of Figure 2). You can test your app using your phone/tablet over Wi-Fi, or with the provided emulator (shown).

App Inventor was designed to get from zero to app in as little time as possible. The key features that enable that are:

- A high level component architecture that exposes complex functionality via simple and concrete properties, events and methods.
- Event handling is front and center.
- A drag-and-drop blocks-based programming language that is fun to use and that minimizes syntax and type errors.
- A ”live development” mode that allows users to instantly see their app as they develop it and experiment with blocks and components.
- You can build real apps that run on phones and tablets!

High-Level Components

App Inventor components attempt to encapsulate complex functionality, of both UI and behavior, in simple forms which can be dropped WYSIWYG-style from onto a browser-based representation of a phone screen. For example, you can just drop a ListPicker on the screen, set the contents of its corresponding list and you're done. In your app when you click on the ListPicker's corresponding button you can select from the items in the list. Behavior-wise, you can drop a Twitter component that handles all the details of connecting to Twitter, then use the provided high-level blocks to get and send tweets, follow users, etc.

Event-First Programming

Modern mobile devices are much more interactive, reactive to their environment and connected to external data sources than their desktop and laptop brethren. App Inventor emphasizes those interactive and reactive behaviors and makes handling events easy by providing explicit ”when” blocks for each interesting event. These events are things like button clicks, tweets or texts received, incoming location readings, web content obtained, etc. You simply drop “when” blocks into the program then add the blocks that are to run when the event occurs. There is no ”main” program and App Inventor hides all the behind-the-scenes event-dispatching code (which is too complex for beginners, e.g., programming Java Listeners).

Blocks Language

App Inventor's drag-and-drop, blocks-oriented language reduces the kinds of syntactic errors that are common in text-based languages and requires much less typing. Having explicit, concrete blocks makes it clear to users what choices are available to them at all times, and the Blocks editor also does some static type checking, letting the user know as a block is dragged which other blocks it can be plugged into. In blocks form, App Inventor also provides the same coding constructs – conditionals, loops, etc. – as you find in a traditional textual language. There are also blocks for traditional functionality like lists and text, and much of the Android library is exposed. There are limitations, as Android is always changing and only the MIT App Inventor team can expose new functionality (there isn't currently a way to import components or blocks from third parties).

Live Development

App Inventor provides ”live development”, where changes made in either the User Interface Designer or the Blocks Editor are immediately reflected in the developer's phone, such that she sees the app and can interact with and test it instantly at every stage of its development. The developer can even test partial apps by hooking some blocks together, performing a 'Do it' operation and seeing what the result is. This all means that users get immediate feedback on their actions in the most direct and relevant way possible, i.e. in the app as it's running on the phone.

WHO TEACHES APP INVENTOR?

Through the App Inventor forums and appinventor.org, we interact with many teachers all over the world using App Inventor in the classroom. In this section we'll provide an overview of what is happening.

App Inventor in Colleges/Universities

Most college-level App Inventor courses are introductory courses for non-majors (CS0), but there have also been courses developed in other disciplines as well, including Health Informatics, e.g., [13], Business, and a Design/CS course co-taught by professors from both disciplines. [14]

One of the first courses was a CS0 course taught by David Wolber at USF. [19] Wolber started teaching with App Inventor as part of the original Google pilot program in 2009, and now teaches multiple sections of the course each semester. The course targets the general student population (mostly business and humanity students) and follows a constructionist approach [16], with students building apps the first day, then building successively more complex apps throughout the semester. At each stage, students customize sample apps, participate in discussions on terminology and concepts, and then are assigned the task of creating an app that someone will really use.

The students build portfolios of their work as they would in an Art or Design class. The students are excited to show their friends and family what they have created, which motivates learning, and they begin driving the learning process to learn the concepts they need to build their next app they envision. Imagine having design and business majors coming to office hours enthusiastic about learning about if- statements!

Because the language is high-level and not all topics required in a CS1 course are taught, the students are exposed to many topics and given a broad introduction to the software ecosystem. They explore topics like sensors, web persistence, and web access that generally don't fit in an introductory course, and they are also exposed to thinking about a problem space and what should be built, prototyping, business plans, intellectual property,user-centered design and testing. Our experience suggests that this broad view of computer science helps prepare students for the world they are entering, even if they don't ever take another

FIGURE 3. High School Girls learn entrepreneurship and App Inventor coding in the Technovation Challenge

computer science course.

The sections for the USF course fill up immediately, with over half women, and students report a high-level of motivation in post-course evaluations. Many students go on to take additional CS courses or become CS minors/majors. A number of students have gone on to volunteer at local schools and after-school programs, including a group of four women who served as teachers and role-models to the high school girls in the Technovation program [10]. For first-hand student testimonials, check out the videos on the home page at appinventor.org [2].

Because the students can build such interesting apps, they now present their apps at the CS department's annual CS night, along with the Senior and Master project students. The upper class students and alums often grumble, "Why didn't we get to do cool stuff like that when we started!", and alums often mistake the App Inventor students for CS majors. The event, and the course in general, has exposed many business, design, and humanities students to computer science and raised awareness of the field and the CS department within the greater university population.

The teaching materials from the course, including video lessons, a textbook, and a modular course-in-a-box curriculum, are available at appinventor.org [2]

App Inventor in K-12

At the K-12 level, few schools offer computer science as there are too few qualified teachers and it is not a core course. The current Advanced Placement (AP) course, taught in Java, is not widely offered. Fortunately, a new AP course in computer science principles (CSP) is being piloted across the country. The new CSP initiative offers a less coding-centric and language-agnostic scheme for teaching computational thinking to a broad audience. Because visual languages like Scratch, Alice, and App Inventor can be taught, more teachers can be trained and more students can have success.

Trinity College professor Ralph Morelli and High School teacher Uche Chinma are leading the App Inventor CSP pilot. The pilot has been successful in a number of schools and the teachers, most of whom had little prior coding experience, have been able to teach the course after just weeks of instruction. The "Mobile CSP" curriculum is now offered on-line with branches for both students and professional development. [8]

There are many more examples of successful App Inventor courses both at high schools and the middle school levels. As an example of the type of excitement these courses elicit, check out this video about the class offered by Massachusetts STEM teacher of the year, Kelly Powers. [6]

After-School Programs

Code camps and after school programs, many of them based on App Inventor, are popping up all over the world. One terrific example is the Technovation program from Iridescent (see Figure 3), which targets high school girls and focuses on technology entrepreneurship. The girls come up with an idea for an app, formulate a business plan, and use App Inventor to build prototypes for the apps they are designing and pitching. The program is now serving over 800 teams world-wide.

Other after-school programs teaching mobile computing with App Inventor include Apps for Good [1], which reaches thousands of students in the UK, Oakland Youth Radio's App Lab [9], which teaches app building to media-savvy youth, and Black Girls Code [3], which empowers young women of color in STEM fields.

BRIDGING TO CS1

Though it is not the primary goal of most App Inventor courses, many students have been motivated to take additional courses and even add a CS minor or major. An interesting research area involves tracking this diverse group, with backgrounds far different than the typical CS student, to see how they fare.

We have had some anecdotal feedback from the students at USF who have continued. Those questioned commonly expressed that the fundamentals they learned in the App Inventor course were vital to making it through the traditional CS1 course (taught in Python). They also commonly expressed frustrations switching to the syntax of a textual language, and disappointment and some loss of motivation in going from building real-world mobile solutions to less motivating programs.

In Spring 2015 USF will explore a strategy which will allow students to stay in the mobile world through a CS0- CS1 sequence. The course will be based on the App Inventor Java Bridge, a code library with Java classes corresponding to each of the components in App Inventor [7]. Motivation is expected to be high as the students will get to build mobile apps throughout the course. And because the terminology and structure of the Java code parallels the blocks of App Inventor, the transition to Java will be facilitated.

SUMMARY

The common thread amongst most App Inventor educational efforts is that ”what is being built” is the driving force, not the abstract goal of learning to code. Though some students are motivated to learn coding for coding's sake, many more get excited about solving real-world problems and improving their lives and those of their friends. ”What is worth making?” is discussed as much or more as ”how to make it?” It is why the App Inventor book [12] has chapters organized by the apps you can build, rather than programming topics like conditionals and iteration (though those topics are ”hidden” within the chapters!).

This emphasis on solving real-world problems helps attract a broader clientele: artists, designers, entrepreneurs, business majors, humanity students, women, people of color, and in general folks who have been underrepresented or left out of the software process altogether.

Consider this email from an App Inventor user, which is indicative of many we receive:

I'm a visual guy. I have no room in my head for remembering languages, syntax, formulas, etc. Instead, I am a Google addict, always finding solutions to my problems on the fly as they arise. With App Inventor, it removes the language and syntax portions from my thought process so I can focus on what I want the app to look like... on what I want it to do. If the puzzle pieces don't fit, there's a reason for it.

App Inventor enables creative people like Jared to participate in creating mobile solutions, to be producers and not just consumers of technology. This empowerment can certainly can help them in their jobs and lives, but it is also a big win for society: it infuses the mobile computing world with great thinkers and big-picture problem solvers!

This newly broadened and diversified group of creators offers many research questions to be answered. What are the long-term effects, for a non-CS-student, of taking an App Inventor course? Will experiences like those described in the ”Snapshots” section motivate those students to take another coding course or to try coding in their job? Do App Inventor students learn the fundamentals and fare better in follow-up traditional CS courses than they would have without such a course? Let's say a student takes one course and never codes again. Do the computational thinking [18] concepts and problem-solving techniques they learn help them in their everyday lives? Are they better prepared for solving problems involving software and communicating with software engineers? Will they have a different relationship with their mobile devices, and feel empowered to tinker with their phones and create apps for every-day life? ■

The authors thank Google, the National Science Foundation, and the W.M. Keck Foundation for their support.

REFERENCES

[1] App for Good Website. http://appsforgood.org. [Online;accessed 05-May-2014].

[2] AppInventor.org Website. [Online;accessed 05-May-2014].

[3] Black Girls Code Website. https://blackgirls code.com. [Online;accessed 05-May-2014].

[4] Blockly Website. http://code.google.com/p/ blockly/. [Online;accessed 05-May-2014].

[5] Code.org Website. http://code.org. [Online; accessed 05-May-2014].

[6] Kelly Powers– Massachusetts STEM Summit 2013. https://www.youtube.com/ watch?v=ThZ5ZkY0IIUt=14. [Online;accessed 05-May-2014].

[7] MIT App Inventor Website. appinventor.mit. edu. [Online;accessed 05-May-2014].

[8] Mobile CSP Website. http://mobile-csp.org. [Online;accessed 05-May-2014].

[9] Oakland Youth Radio's Innovation Lab Website. https://youthradio.org/creative- studio/desk/ innovation-lab/. [Online;accessed 05-May-2014].

[10] Technovation Website, Iridescent. http:// technovationchallenge.org. [Online;accessed 05-May-2014].

[11] H. Abelson. Mobile ramblings. Ecucause Review, On-line, 34(1), March 2011.

[12] David Wolber, Harold Abelson, Ellen Spertus, and Liz Looney. App Inventor: Create your own Android Apps. O'Reilly, 2011.

[13] B. MacKellar and M. Leibfried. Designing and Building Mobile Pharmacy Apps in a Healthcare IT Course. In Proceedings of the 14th annual ACM SIGITE conference on Information technology education (SIGITE '13)., pages 188–197. ACM, 2013.

[14] F. Martin and K. Roehr. UMass-Lowell Class: Introduction to App Design and Mobile Computing. http://appdesignf13.wiki.uml.edu/. [Online;accessed 05-May-2014].

[15] A. B. Maureen Biggers and T. Yilmaz. Student Perceptions of Computer Science: a Retention Study Comparing Graduating Seniors with CS Leavers. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, SIGCSE ˆaA˘Z´08, pages 402–406. ACM, 2008.

[16] S. Papert. Mindstorms: *Children, Computers, and Powerful Ideas.* Basic Books, Inc., 1980.

[17] C. Thompson. Clive Thompson on Coding for the Masses. http://www.wired.com/2010/11/ stthompsonwereallcoders/, 201 May – 2014].

[18] J. M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, March 2006.

[19] D. Wolber. App Inventor and Real-World Motivation. In *Proceedings of the 42nd ACM Technical Symposium on Computer science education* (SIGCSE '11), New York, NY, 2011. ACM.