Preface

You're on your regular running route, just jogging along, and an idea for the next killer mobile app hits you. All the way home, you don't even care what your time is, all you can think about is getting your idea out there. But how exactly do you do that? You're no programmer, and that would take years, and time is money, and...well, someone has probably done it already anyway. Just like that, your idea is dead in the water.

Now imagine a different world, where creating apps doesn't require years of programming experience, where artists, scientists, humanitarians, health-care workers, attorneys, firefighters, marathon runners, football coaches, and people from all walks of life can create apps. Imagine a world where you can transform ideas into prototypes without hiring programmers, where you can make apps that work specifically for you, where you can adapt mobile computing to fit your personal needs.

This is the world of App Inventor, a visual programming tool for building mobile apps. Based on a visual "blocks" programming method that's proven successful even with kids, App Inventor dramatically lowers the barriers to creating apps for Android phones and devices. How about a video game where the characters look like you and your friends? Or a "did you pick up the milk?" app that reminds you if it's after 3 p.m. and you're near the grocery store? Or a quiz app you give your significant other that's in fact a surprise marriage proposal? "Question 4: Will you marry me? Press the button to accept by sending a text message." Someone really created an App Inventor app to propose marriage like this, and she said yes!

A Blocks Language for Mobile Phones

App Inventor is a visual, drag-and-drop tool for building mobile apps on the Android platform. You design the user interface (the visual appearance) of an app using a web-based graphical user interface (GUI) builder, then you specify the app's behavior by piecing together "blocks" as if you were working on a puzzle.

Figure P-1 shows the blocks for an early version of an app created by Daniel Finnegan, a university student who had never programmed before. Can you tell what the app does?

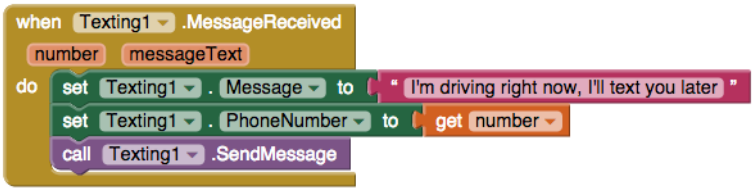

Figure P-1. An app that autoresponds to texts

The app is a text "answering machine." You launch it when you're driving and it autoresponds to the texts you receive.

Because the blocks are more understandable than traditional programming code, you're immediately drawn in, and the real-world utility gets you asking questions like: Can I make it so the received texts are spoken aloud? Can I make it so the response sent back could be customized? Can I write an app that lets people vote for something by text, like on *American Idol*? The answer to all these questions is "yes," and in this book, we'll show you how.

What Can You Do with App Inventor?

Lots of stuff!

Play
: Creating apps for your phone is fun, and App Inventor promotes exploration and discovery. Just open App Inventor in a web browser, connect your phone, and start putting together blocks like those in the app shown in Figure P-1. You can immediately see and interact with the app you're building on the phone. So you're programming, but you're also emailing your friend to send you a text to test your app, or you're controlling a LEGO NXT robot with the app you just built, or you're unplugging the phone and walking outside to see if your app is using the location sensor correctly.

Prototype
: Have an idea for an app? Instead of writing it down on a napkin or letting it float off into the ether, build a quick prototype. Prototypes are incomplete and unrefined working models of your idea. Expressing an idea in text is like writing a to a friend or loved one with prose; think of an App Inventor prototype as poetry to a venture capitalist. In this way, App Inventor can serve as an electronic napkin for mobile app development.

Build apps with personal utility
: In the current state of the mobile app world, we're stuck with the apps we're given. Who hasn't complained about an app and wished it could be personalized or adjusted in some way? With App Inventor, you can build an app exactly how you want it. In Chapter 3, you'll build a MoleMash game that lets you score points by touching a

randomly moving mole. But instead of using the image of the mole in the tutorial, you can customize it so that you mash a picture of your brother or sister—something that only you might want to do, but who cares? In Chapter 8, you'll write a quiz app that asks questions about US Presidents, but you can easily customize it to ask questions on any topic you want, from your favorite music to your family history.

Develop complete apps
: App Inventor is not just a prototyping system or an interface designer—you can build complete, general-purpose apps. The language provides all the fundamental programming building blocks like loops and conditionals, but in block form.

Teach and learn
: Whether you're at a middle school, high school, or university, App Inventor is a great teaching and learning tool. It's great for computer science, but is also a terrific tool for math, physics, entrepreneurship, and just about any other discipline. The key is that you learn by creating. Instead of memorizing formulas, you build an app to, say, find the closest hospital (or mall!). Instead of writing an essay on Black History, you create a multimedia quiz app with video and speeches from Martin Luther King, Jr., and Malcolm X. We think App Inventor, and this book, can be a great tool in classes throughout the curriculum.

Why App Inventor Works

Most people say that App Inventor is easy to use because of its visual, drag-and-drop interface. But what does this mean? Why is App Inventor so easy to use?

You don't have to remember and type instructions
: One of the biggest sources of frustration for beginning programmers comes from typing in code and having the computer spit back indecipherable error messages. This frustration discourages many beginners from programming before they even get to the more fun, logical problem solving.

You choose from a set of options
: With App Inventor, the components and blocks are organized into drawers that are readily available to you. You program by finding a block—which helps specify the functionality you want to build—and dragging it into the program. You don't have to remember what the instructions are or refer to a programming manual.

Only some blocks plug in to each other
: Instead of chastising programmers with cryptic error messages, App Inventor's blocks language restricts you from making many mistakes in the first place. For instance, if a function block expects a number, you can't plug in text. This doesn't eliminate all errors, but it sure helps.

You deal with events directly
: Traditional programming languages were designed when programming was like working with recipes, or sets of instructions. But with graphical interfaces, and

especially with mobile apps where events can happen at any time (for example, receiving a text message or phone call), most programs are not recipes, but are instead sets of event handlers. An event handler is a way of saying, "When this happens, the app does this." In a traditional language like Java, you have to understand classes, objects, and special objects called listeners to express a simple event. With App Inventor, you can say, "When a user clicks this button..." or "When a text is received..." by dragging out a "When" block.

What Kind of Apps Can You Build?

You can build many different types of apps with App Inventor. Use your imagination, and you can create all kinds of fun, useful apps.

Games
: People often begin by building games like MoleMash (Chapter 3) or apps that let you draw funny pictures on your friend's faces (Chapter 2). As you progress, you can build your own versions of more complex games like Pac-Man and Space Invaders. You can even use the phone's sensors and move characters by tilting the phone (Chapter 5).

Educational software
: App building is not limited to simple games. You can also build apps that inform and educate. You can create a quiz app (Chapter 8) to help you and your classmates study for a test, or even a create-a-quiz app (Chapter 10) that lets the users of your app create their own quizzes (think of all the parents that would love this one for those long road trips!).

Location-aware apps
: Because App Inventor provides access to a GPS-location sensor, you can build apps that know where you are. You can build an app to help you remember where you parked your car (Chapter 7), an app that shows the location of your friends or colleagues at a concert or conference, or your own custom tour app of your school, workplace, or a museum.

High-tech apps
: You can create apps that scan bar codes, talk, listen (recognize words), play music, make music (Chapter 9), play video, detect the phone's orientation and acceleration, take pictures, and make phone calls. Smartphones are like Swiss Army knives for technology, and App Inventor makes it easy to control that technology.

SMS Texting apps
: No Texting While Driving (Chapter 4) is just one example of the SMS processing apps you can build. You can also write an app that periodically texts "missing you" to your loved ones, or an app like Broadcast Hub (Chapter 11) that helps coordinate large events. Want an app that lets your friends vote for things by texting, like on *American Idol*? You can build it with App Inventor.

Apps that control robots
: Chapter 12 shows how to create an app that acts as a controller for a LEGO robot. You can use the phone as a remote control, or you can program it to be a "brain" that the robot carries around with it. The robot and phone communicate via Bluetooth, and App Inventor's Bluetooth components let you create similar apps that control other Bluetooth devices.

Complex apps
: App Inventor dramatically lowers the entrance barrier to programming and lets you build flashy, high-tech apps within hours. But the language also provides loops, conditionals, and other programming and logic constructs necessary to build apps with complex logic. You'll be surprised at how fun such logic problems can be when you're trying to build an app.

Web-enabled apps
: App Inventor also provides a way for your apps to communicate with the Web. You can write apps that pull in data from Twitter or an RSS feed, or an Amazon Bookstore Browser that lets you check the online cost of a book by scanning its barcode.

Who Can Build Apps?

App Inventor is freely available for anyone to use. It runs online (instead of directly on your computer) and is accessible from any browser. You don't even need a phone to use it: you can test your apps on an included Android emulator. As of September 2014, there were 1.9 million registered App Inventor users from 195 countries. Together they have created nearly five million apps.

Who are these app builders? Were they already programmers when they started? Some of them were, but most were not.

One of the most telling experiences has been the courses that coauthor David Wolber teaches at the University of San Francisco. At USF, App Inventor is taught as part of a general education computer science course targeting primarily business and humanities students. Many students take the course because they are afraid of math, and the course fulfills the dreaded Math Core requirement. The vast majority have never even dreamed of writing a computer program.

Despite having no prior experience, the students have been successful in learning App Inventor and building great apps. An English major created the first No Texting While Driving app, two communications majors created Android, Where's My Car? (Chapter 7), and an International Studies major created the Broadcast Hub app (Chapter 11). When an art major knocked on Wolber's office door one night well after hours asking how to write a `while` loop, Wolber knew that App Inventor had dramatically changed the computer science education landscape.

The media grasped the significance as well. The *New York Times* called App Inventor "Do-It-Yourself App Creation Software." The *San Francisco Chronicle* reported on the

USF students' work in an article, "Google brings app making to the masses." *Wired* magazine featured Daniel Finnegan, the author of No Texting While Driving, and wrote that "Finnegan's story illustrates a powerful point: It's time for computer programming to be democratized."

The cat is, as they say, out of the bag (your first app will involve a kitty, by the way). App Inventor is now used in middle school and high school courses around the world; by over 2,500 girls in 28 countries who have participated in in the Technovation Challenge, an after-school program for high school girls; in pilot courses for the new high school Computer Science Principles Advance Placement course; and in new introductory courses at several universities. There are now thousands of hobbyists, businesspersons, marriage proposers, and tinkerers roaming the App Inventor site building apps. Want to get in on the action? No programming experience is required!

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

Tip This element signifies a tip or suggestion.

Note This element signifies instructions for testing the app you are building.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *https://appinventor.org/bookFiles*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*App Inventor 2* by David Wolber, Hal Abelson, Ellen Spertus, and Liz Looney (O'Reilly). Copyright 2015 David Wolber, Hal Abelson, Ellen Spertus, and Liz Looney, 978-1-491-90684-2."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals.

Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que,

Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/app-inventor2*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

The educational perspective that motivates App Inventor holds that computing can be a vehicle for engaging powerful ideas through active learning. As such, App Inventor is part of an ongoing movement in computers and education that began with the work of Seymour Papert and the MIT Logo Group in the 1960s, and whose influence persists today through many activities and programs designed to support computational thinking.

App Inventor's design draws upon prior research in educational computing and upon Google's work with online development environments. The visual programming framework is closely related to the MIT Scratch programming language. The specific implementation for App Inventor 2 is based on Blockly, which is developed at Google and led by Neil Fraser. The compiler that translates the visual blocks language for implementation on Android uses the Kawa Language Framework and

Kawa's dialect of the Scheme programming language, developed by Per Bothner and distributed as part of the GNU Operating System by the Free Software Foundation.

The authors would like to thank Google and the original App Inventor team for their support of our work and teaching efforts at USF, Mills College, and MIT. Special thanks go to App Inventor Technical Lead Mark Friedman, Project Manager Karen Parker, and engineers Sharon Perl and Debby Wallach.

We would also like to thank the MIT App Inventor team for their work and continued development of App Inventor. Special thanks go to Technical Lead Andrew Mckinney, all-around guru Jeff Schiller, education and outreach directors Shaileen Pokress and Josh Sheldon, unsung hero and engineer Jose Domínguez, and key "sabbatical" contributors Franklyn Turbak and Ralph Morelli.

We also owe a special thanks to University of San Francisco student Cayla Shaver for her extraordinary editing work and helping convert this book for App Inventor 2.

Finally, we'd like to acknowledge the support of our respective spouses: Ellen's husband, Keith Golden; Hal's wife, Lynn Abelson; Liz's husband, Kevin Looney; and David's wife, Minerva Novoa. New mother Ellen is also grateful for the help of nanny Neil Fullagar.