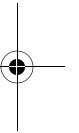
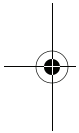


# *Hello App Inventor!*

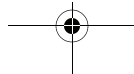
*Android programming  
for kids and the rest of us*



PAULA BEER  
CARL SIMMONS



MANNING  
SHELTER ISLAND



For online information and ordering of this and other Manning books, please visit [www.manning.com](http://www.manning.com). The publisher offers discounts on this book when ordered in quantity.

For more information, please contact:

Special Sales Department  
Manning Publications Co.  
20 Baldwin Road  
PO Box 261  
Shelter Island, NY 11964  
Email: [orders@manning.com](mailto:orders@manning.com)

©2015 by Manning Publications Co. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in the book, and Manning Publications was aware of a trademark claim, the designations have been printed in initial caps or all caps.

⊗ Recognizing the importance of preserving what has been written, it is Manning's policy to have the books we publish printed on acid-free paper, and we exert our best efforts to that end. Recognizing also our responsibility to conserve the resources of our planet, Manning books are printed on paper that is at least 15% recycled and processed without elemental chlorine.

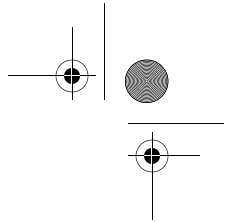
**M** Manning Publications Co.  
20 Baldwin Road  
PO Box 261  
Shelter Island, NY 11964

Development editor: Susanna Kline  
Copyeditor: Tiffany Taylor  
Proofreader: Alyson Brener  
Technical proofreader: Jerome Baton  
Typesetter: Marija Tudor  
Cover designer: Leslie Haimes

ISBN 9781617291432

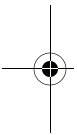
Printed in the United States of America

1 2 3 4 5 6 7 8 9 10 – EBM – 19 18 17 16 15 14



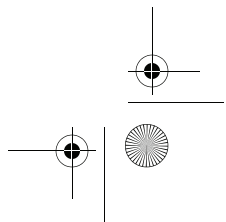
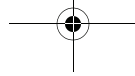
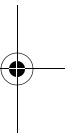
To anyone who really, really tries.  
Come on, keep going, you're nearly there.

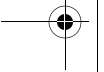
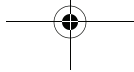
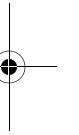
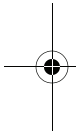
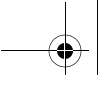
—P.B.

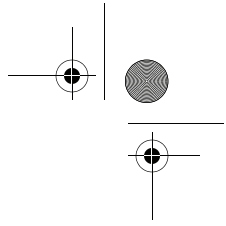


To Frank, who treated us like *geniuses* and loved us fiercely.

—C.S.



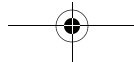


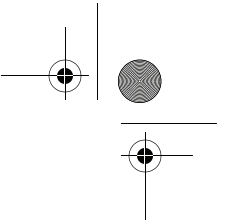


# *Brief contents*

---

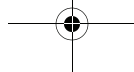
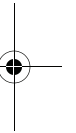
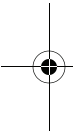
- 1 ■ *Getting to know App Inventor* 1
- 2 ■ *Designing the user interface* 30
- 3 ■ *Using the screen: layouts and the canvas* 48
- 4 ■ *Fling, touch, and drag: user interaction with the touch screen* 65
- 5 ■ *Variables, decisions, and procedures* 79
- 6 ■ *Lists and loops* 112
- 7 ■ *Clocks and timers* 146
- 8 ■ *Animation* 165
- 9 ■ *Position sensors* 183
- 10 ■ *Barcodes and scanners* 203
- 11 ■ *Using speech and storing data on your phone* 218





**vi**    **BRIEF CONTENTS**

- 12** ■ *Web-enabled apps*    **236**
- 13** ■ *Location-aware apps*    **257**
- 14** ■ *From idea to app*    **270**
- 15** ■ *Publishing and beyond*    **312**



# Contents

---

*Preface xvii*

*Acknowledgments xix*

*About this book xxii*

## **1 Getting to know App Inventor 1**

*First a little history ... 1*

*What can an Android smartphone do? 2*

*App Inventor setup 2*

Option 1: Using Wi-Fi with the App Inventor companion phone software 3

Option 2: using the onscreen emulator 4 ■ Option 3: connecting via USB  
cable 4 ■ Using all three options 5 ■ Troubleshooting 6

*The App Inventor juggling act 6*

1. Designing the app screen 6 ■ 2. Telling the app what to do 7

3. Testing the program 8

*How will it look? App Inventor Designer 8*

*Component properties 9*

*How will it work? The Blocks Editor 9*

Built-in blocks 10 ■ Component-specific blocks 11

## viii CONTENTS

*Running and testing programs 13*

Running and testing: emulator vs. smartphone 13

*Your first app: Hello World! 13*

1. Opening App Inventor 14
2. Starting with a new project 15
3. Adding a Notifier component to the project 16
4. Writing the program using blocks 16
5. Testing the app 19

*Computers never sleep: why you use events 21**Saving in the cloud and using checkpoints 21**Button click: Hello World! app, version 2 23*

1. Saving a new copy of Hello World! 23
2. Adding a Button component 23
3. Programming the blocks 24
4. Testing your app 25
5. Taking it further 26

*Packaging an app for your phone 26*

1. Downloading APKs directly to your phone or computer 26
2. Downloading APKs as files to your computer 27

*Changing the app's icon 28**What did you learn? 28**Test your knowledge 29**Try it out 29***2 Designing the user interface 30***What is a user interface? 30**Using the Designer to make a UI 32**Speeding along: built-in components 33**Getting to Know Ewe app 34*

1. Setting up the project 35
2. Adding a sheep image 36
3. Adding a "baa!" sound 38
4. Programming the blocks, part 1: playing the sound 39
5. Programming the blocks, part 2: vibrating the phone 40

*Extra challenge: Ewe Scared Her! app 42*

1. Saving a new project 43
2. Adding components: accelerometer, reset button, and screen arrangement 43
3. Arranging the screen 45
4. Programming the blocks 45
5. Taking it further 46

*What did you learn? 46**Test your knowledge 47**Try it out 47*



### 3 Using the screen: layouts and the canvas 48

*Layout* 49

Layout: mini-project 50

*Spooky Sound FX app* 52

1. Setting the screen properties: alignment, orientation, and scrolling 53
  2. Setting up the table arrangement and eight buttons 54
  3. Adding the eight Sound components and their source files 55
  4. Coding the blocks 55
- Taking it further 56

*Introducing the Canvas component* 57

*Graffiti Artist app* 58

1. Setting up the user interface 59
2. Coding the blocks 60

*What did you learn?* 64

*Test your knowledge* 64

*Try it out* 64

### 4 Fling, touch, and drag: user interaction with the touch screen 65

*Events up close* 67

- Flung event 67 ▪ Drag event 68

*Flingflung app* 68

1. Dragging out the Ball1.Flung event handler from the Ball1 drawer 69
2. Giving the ball velocity 70
3. Enabling the Dragged event 70
4. Enabling the CollidedWith event 70

*Fling It! app* 71

1. Making the touch sprite react to the user touching the sprite 74
2. Making the drag sprite react to the user dragging the sprite 75
3. Making the fling sprite react to the user flinging the sprite 75
4. Playing the soundtrack 75
5. Stopping the soundtrack 76

*Taking it further* 76

*What did you learn?* 77

*Test your knowledge* 77

*Try it out* 78

### 5 Variables, decisions, and procedures 79

*Remembering useful things* 79

*Total recall: naming and retrieving variables* 80

## x CONTENTS

*Flattery app 81*

- 1. Setting up the screen 83
- 2. Coding the blocks: creating a UserName variable 84
- 3. Getting and storing UserName 84
- 4. Revealing the buttons 85
- 5. Stringing it all together: making the flattery messages 85
- Taking it further 87

*Decisions, decisions: using variables to choose an action 88*

- Comparison operators 88
- How can a dumb machine make a decision? 90

*Prankster Flattery app (Personality Judge app) 91*

- 1. Setting up the app 91
- 2. Coding the if ... else blocks 92
- Using creative comparison conditions 93
- Taking it further 95

*Keeping track with comments 95**Changing variables 96**Dice Roll app 97*

- 1. Programming the blocks 98
- 2. Taking it further 98

*Guess What I Am Thinking app 100*

- 1. Setting up the screen 102
- 2. Initializing 103
- 3. Gameplay 106
- 4. Resetting the game 109
- Taking it further 110

*What did you learn? 110**Test your knowledge 110**Try it out 111***6 Lists and loops 112***Readymade lists 113**Making your own list picker 115*

- Ready for ice cream? 116

*Excuse Generator app 118*

- 1. Setting up the screen 119
- 2. Coding the blocks: a slot-machine algorithm 119
- 3. Setting up the lists 120
- 4. Building the sentence 122

*Graffiti Artist 2: the spray-paint returns! 124*

- 1. Updating the screen layout 124
- 2. Coding the blocks: choosing a background image 126
- 3. Changing the line and dot sizes 126
- 4. Offering a rainbow of colors 127

*Loop the loop 129*

- for range ... do ... 129
- for each ... in list ... do ... 130
- while (test) do ... 131

*Multiplication Table Generator app 132*

1. Setting up the screen 133
2. Coding the blocks: defining the programming problems 133
3. Looping the loop: nested loops 134

*Secret SMS Sender app 136*

1. Setting up the screen 138
2. Coding the blocks: encrypting the message 138
3. Adding agent contacts 140
4. Deleting agent contacts 141
5. Sending the message 142

*What did you learn? 143**Test your knowledge 144**Try it out 145***7 Clocks and timers 146***What time is it? 146*

- Time to experiment 147
- Outputting the system time 148
- Instant answers 148
- Making time 149
- Taking it further: validation 149

*Using timers 150**Beeper app 150*

1. Setting up the screen 151
2. Coding the blocks 152

*Where Are Ewe Hiding? app 153*

1. Setting up the screen 154
2. Coding the blocks 155

*Multiple timers 156**Splat the Rat app 157*

1. Setting up the screen 158
2. Coding the blocks: showing and hiding the rat 160
3. Splat that rat! Changing the score 161
4. Timing, ending, and restarting the game 161
5. Setting the rat free 162
- Taking it further 162

*What did you learn? 163**Test your knowledge 163**Try it out 164***8 Animation 165***Animation type 1: moving sprites using properties 166**Ballybally app 166*

1. The properties explained 167
2. Coding the blocks: identifying that the edge has been reached, and testing which way the ball is heading 168
3. Responding to the direction of travel 169
4. Changing the color of the canvas when the edge is reached 169

**xii CONTENTS**

*Animation type 2: Creating movement with a clock timer and user interaction 169*

*Cheeky Hamster app 170*

1. Coding the blocks: setting the BowlClock timer 172
  2. Moving the hamster 172
  3. Shooting seeds 173
  4. Making the seed disappear at the edge of the screen 173
  5. Scoring when the seed hits the bowl 174
  6. Resetting the score 174
- Taking it further 175

*Animation type 3: creating movement using lists and a clock timer 175*

*Creepy Spider app 176*

1. Coding the blocks: making the spider appear to crawl 177
  2. Making the spider move repeatedly 180
  3. Making the spider bite and stay still 181
  4. Making the spider start moving again 181
- Taking it further 181

*What did you learn? 182*

*Test your knowledge 182*

## **9 Position sensors 183**

*Compass app 184*

1. Setting up the screen 185
2. Coding the blocks: where are you heading? 186

*Astonishing Prediction! app 188*

- Understanding the secret of the app 189
  1. Setting up the screen 190
  2. Coding the blocks: abracadabra! 191
- Testing it 192

*Hungry Spider app 193*

1. Setting up the screen 194
  2. Coding the blocks: moving the spider 194
  3. Freeing the flies 195
  4. Feeding the spider 199
  5. Restarting the game 200
- Taking it further 200

*What did you learn? 200*

*Test your knowledge 201*

*Try it out 201*

## **10 Barcodes and scanners 203**

*Barcode Petfinder app 204*

1. Opening the scanner and scanning a barcode 206
  2. Collecting the barcode scanner result 206
  3. Looking for secret codes in the barcode scanner result 206
  4. Revealing the hamster 207
  5. Revealing all the pets 207
- Taking it further 208

*Book Finder app 208*

1. Creating the event handler to scan the barcode 210
2. Using the scanner result to print the ISBN number 210
3. Searching the web 211
- Taking it further 212

*QR Treasure Hunt app 213*

- How does the game work? 213
1. Scanning the QR codes 215
2. Revealing the treasure and the clues 216
- Taking it further 217

*What did you learn? 217**Test your knowledge 217***11 Using speech and storing data on your phone 218***My Dream Recorder app 220*

1. Defining the password 221
2. Checking the password and opening the second screen 222
3. Adding the second screen 222
4. Opening the SpeechRecognizer component 224
5. Using the SpeechRecognizer text 224
6. Initializing Dreamscreen (or, setting the rules for Dreamscreen) 225
- Taking it further 225

*Inspiration Scrapbook app, part 1 225*

1. Inserting the title button 228
2. Defining the list variable 228
3. Adding the quote button 229

*Inspiration Scrapbook app, part 2 229*

4. Storing the text from the text box in TinyDB 231
5. Taking the photograph 232
6. Submitting the photo taken with the camera to TinyDB 232
7. Activating Ownpicturebutton 233
8. Understanding how to check TinyDB 233
- Taking it further 234

*What did you learn? 235**Test your knowledge 235***12 Web-enabled apps 236***Browsing the web 236*

- Using WebView 237
- Using ActivityStarter 240

*Using data from the web 242*

- API keys 242

*Weather Watch app 243*

1. Request an API key 244
2. Work out the API call 244
3. Setting up the screen 247
- Taking it further 250

*Storing and sharing data in the cloud with TinyWebDB 251*

- Limitations of the TinyWebDB test service 251

## xiv CONTENTS

*Dream Sharer app 252*

Retrieving the username and dream 253 ■ Checking that a username has been entered 253 ■ Dream Reader 254

*What did you learn? 256*

*Test your knowledge 256*

*Try it out 256*

**13 Location-aware apps 257***Lost & Found app 259*

Taking it further 261

*Homing Pigeon app 261*

ActivityStarter 264 ■ Layout of the screen 264 ■ Coding the blocks 264 ■ 1. Recording the starting location 265 ■ 2. The Remember Current Location button 265 ■ 3. Saving the location to Tiny DB, and activating the directions button 265 ■ 4. The Get Directions button 266 ■ 5. Creating the tempaddress variable 267 ■ 6. Initializing the app 267 ■ 7. The if block 267 ■ Taking it further 268

*What did you learn? 268*

*Test your knowledge 269*

*Taking it further 269*

**14 From idea to app 270***General problem-solving strategies (useful for app inventing) 271**Zombie Alarm! app 272*

User involvement 272 ■ Decomposing: get out your pencil 274 ■ Solving the timer problems 274 ■ Solving the graphics and animation problems 279

*Zombie Alarm! app: full specification 287*

Taking it further 289

*Designing a complete game 289*

Developing the characters and rules 290 ■ Designing the levels 292

*A-Mazeing Penguin app 293*

1. Setting up the screen 296 ■ 2. Coding the blocks: setting up the variables 296 ■ 3. Updating the game information 297 ■ 4. Animating and moving the penguin 297 ■ 5. Making a splash 299 ■ 6. How to lose a life 300 ■ 7. A fishy interlude 301 ■ 8. Get that Yeti moving 302 ■ 9. When penguins collide ... 303 ■ 10. The snowballs

strike back! 304 ■ 11. Level up! 307 ■ 12. Load a new level 308  
Taking it further 310

*What did you learn?* 310

*Test your knowledge* 311

## 15 Publishing and beyond 312

*Publishing apps* 312

*Young app developers* 313

Thomas Suarez: believes in sharing knowledge with others 313

Nick D'Aloisio: app development millionaire 313 ■ Jordan Casey: My Little World 314 ■ Charley Hutchinson: "Why don't I make my own app?" 314

Where are the girls? 315 ■ A career in app development? 316

*Steps in the publishing process* 316

Step 1: gather materials for release 317 ■ Step 2: configuring your application for release 318 ■ Step 3: building your application for release 318 ■ Step 4: preparing external servers and resources 318

Step 5: testing your application for release 319

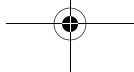
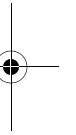
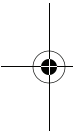
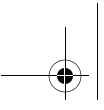
*The future of App Inventor* 319

*Other platforms* 319

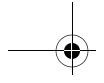
*Staying with App Inventor for the time being?* 320

What about trying some advanced features? 321 ■ Final words 322

*Index* 323







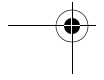
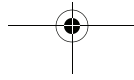
## Preface

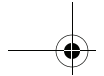
---

We think writing computer programs is fun and that those who can do it can make a difference to the world around them—sometimes in an almost magical way. But when you start out, it's often hard to see how the programs you write can make any difference to anyone—for example, you might just be drawing simple shapes or adding up a bunch of numbers.

Back in 2012, we started using App Inventor with teachers and children and discovered it was a brilliant way to make computer programs that worked in the real world. Beginners could perform useful, imaginative, and fun tasks like a GPS treasure hunt or a homework excuse generator. The App Inventor books that helped us learn were great, but we wanted to focus on helping school-age kids and beginners become app creators. Paula proposed that we write a book, and within a day we had the original contents page and app list. This initial speed lulled Paula into a false sense of security, and colleagues now remind her that she waved her arms around and said "It'll only take us 12 weeks!" She was only off by a factor of 8 ...

But what a couple of years it has been! Throughout the book, we've woven in key facts and resources useful to beginning programmers and always tried to develop original (or inspired!) working examples. We feel that the power of visual programming languages is brought out through the concepts and the huge variety of apps that can be produced by users of the book. Our companion website provides great graphics and sounds for each of the apps and a really useful table layout so users can set up their designs quickly and get down to learning to program with App Inventor right away.

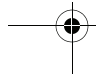
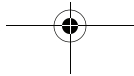
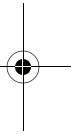
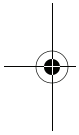


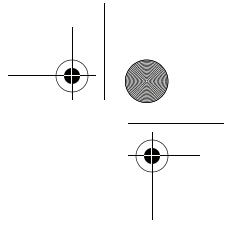


**xviii** PREFACE

We've used the resources from this book to teach App Inventor to primary-age kids, secondary-age kids, trainee teachers, experienced programmers, and experienced teachers—and what we always see is fun, satisfaction, and engagement.

We hope you get a lot out of the book and make fun apps for you and your friends. Who knows? You might even make the next award-winning killer app!





# Acknowledgments

---

Many people helped bring this book to fruition—mentors, colleagues, reviewers, editors, friends, and family. We thank you all.

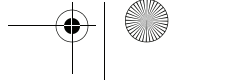
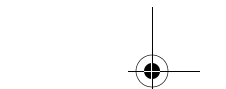
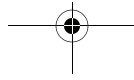
Thanks to the reviewers who read the manuscript in various stages of its development and provided invaluable and encouraging feedback: Aditya Sharma, Alain Couniot, Andrei Bautu, Brent Stains, Chris Davis, Ezra Simeloff, Ian Stirk, John D. Lewis, Mark Elston, Michael Knoll, Phanindra V. Mankale, Richard Lebel, Rick Goff, and Ron Sher.

Thanks to the student reviewers from Avon Grove Charter School, West Grove, PA, whose comments helped make this a much better book for our target audience: Alex Wilson, Ian Khan, Jacob Snarr, Jake Kerstetter, James Cottle Vinson, Lewis Arscott, Rhys Cottle Vinson, and their teacher Jacqueline Wilson.

Thanks to Diane Blakeley and the brilliant pupils of Wellfield College for trying out the early chapters of this book and coming up with brilliant ideas for apps, especially the Homework Excuse Generator.

Thanks to the readers of Manning's Early Access Program (MEAP) for their comments and their corrections to our chapters as they were being written. You helped us to improve our manuscript.

Thanks to Shay Pokress at MIT for keeping us in the App Inventor loop. Thanks for your encouragement and also to the team at MIT who continue to support App Inventor as a great educational tool.



**xx ACKNOWLEDGMENTS**

Our Manning editor, Susanna Kline, always made great suggestions and knew when to push and when to wait with patience and understanding while we grappled with difficult chapters; thanks for being a constant and encouraging presence.

Finally, thanks to the super-professional and efficient team at Manning who worked with us and supported us throughout: Marjan Bace, Scott Meyers (for taking a chance on us), Cynthia Kane, Candace Gillhoolley, Jerome Baton, Kevin Sullivan, Tiffany Taylor, Mary Piergies, and many others who helped along the way.

## Paula Beer

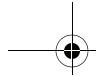
I would like to give huge thanks to my friends and family. In “how long I have known them” order: my mum, Carol, who taught me the white-hot fear of not having a good book on the go; my dad, Bernie, who taught me how to teach using confidence-building examples; my brother, Andy “Analogy Man,” who taught me how to explain myself clearly; my sister, Netty, who taught me about priorities and how to juggle (career and family, not batons; she is rubbish at that); and my twin cousin Stephen and best chum Janet, who have always convincingly feigned interest in my progress and inspired me with their own achievements. Thanks to my in-laws, Gillian and John, whose own love of writing inspired me toward this possibility. To my Edge Hill colleagues Claire, Dawn, and Colette: your encouragement means so much.

To my coauthor and sparring partner Carl: it’s been fun, especially as I won the last round (don’t you dare edit this out!). To my three delicious children, Sam, Ella, and Sophia: thanks for testing my apps, telling me when they were up to the mark, and making everything wonderful. Thank you for not spilling juice on my keyboard when occasionally I sat among you writing apps on treat night while watching Harry Hill. Finally, to my incredible husband Rufus: it is impossible to fail at anything with a man like you in my life.

## Carl Simmons

Thank you to our Edge Hill colleagues who supported us throughout and patiently endured our bizarre conversations about hungry spiders, cheeky hamsters, and amazing penguins. Thanks also to the many teachers and students who tried out chapters and gave us great feedback.

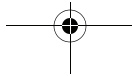
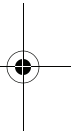
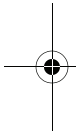
To Mum and Dad: thanks for indulging my early geekhood. Not only for all the computers, disk drives, monitors, and books, but also for the self-esteem that comes from constant encouragement and support. Huge thanks to my family for putting up with the long hours I sat staring at screens of various sizes, my wandering around holding a phone trying to get a



ACKNOWLEDGMENTS **xxi**

GPS signal, and my occasional shouts of frustration or triumph. Also for offering lots of helpful advice about ideas, characters, games, and graphics. Special thanks to the children for testing the apps, especially A-Mazeing Penguin, which needed lots of play-testing to get right. To Daniel: thanks for the brilliant Zombie picture and for testing the early chapters to see whether a 10-year-old could follow them. To Ellie: thanks for ensuring that I didn't become a total hermit, dragging me away from the computer to have fun with the family and providing a constant soundtrack of music and dance while I worked. To Lynne: without your support, this simply wouldn't have happened; thanks for being the chauffeur, chef, and chief of our home to give me the space to write.

Finally, thanks to my coauthor, Paula: you've made this a huge amount of fun, and that's kept me going throughout. But just remember, it's not the last round that counts, it's the total score!



## About this book

---

This book will help you become an App Inventor—someone who doesn't just use a phone or tablet, but takes control of it! You'll learn to create some fun apps, and along the way you'll also learn programming skills that you can use in lots of other programming languages. We wrote this book for kids, but anyone who's curious about programming and mobile devices will find it useful.

We aren't expecting you to have any programming knowledge at all—we'll start from the very beginning (you'll be surprised how quickly you learn). You need to know your way around a keyboard and mouse, how to save files, and how to use a web browser like Chrome, Firefox, or Safari. You'll also find it useful to know how to use an Android smartphone and access its menu settings. If you can do that, you can jump right in and get started making apps.

### What is an App Inventor?

Before we answer that question, we need to answer this one: *"What is an app?"*

An *app* is a computer program—a list of instructions that tells a computer what to do. Normally, the word *app* is used to talk about programs written for smartphones and other mobile devices like tablets. It's going to get a bit longwinded to keep talking about "smartphones and other mobile devices like tablets," so throughout this book we'll use the term *phone* or *smartphone* to mean any mobile Android device such as a smartphone or tablet.

An *App Inventor* is two things:

- App Inventor is a programming language you can use via an internet browser to design and make apps for Android phones. It's a graphical environment—that means you don't have to type complicated code. Instead, you drag and drop objects on screen and plug blocks of code together like a jigsaw puzzle. If you've used the Scratch programming language, this will be familiar.
- *You are an App Inventor.* Once you've done the first exercise in chapter 1, you can proudly claim, "I am an App Inventor!" That's someone who can create code that runs on a phone.

## What is Android?

We've mentioned Android a couple of times already. Android is an operating system (OS) that runs on lots of mobile devices. An OS manages a computer's *hardware*—that's all the bits you can touch, including computer chips and circuit boards and cameras and touch screens.

Windows, Mac OS, and Linux are OSs that are mainly used in desktop and laptop machines. Android, iOS, and Windows Phone are the main OSs you find in mobile devices. So App Inventor lets you program pretty much any Android device, but it won't work on iPhones or Windows phones.

## Why should you be an App Inventor?

Learning to program computers is fun! It can sometimes be frustrating when things aren't working, but the joy you get from solving these problems is huge.

Programming is a great skill to learn because it helps you think in a certain way—this is sometimes called *computational thinking*. What this really means is that once you can program, you can solve lots of problems in the real world, too. Even if you become an architect, an artist, an engineer, or a scientist, the skills you learn from programming are useful. These skills are things like

- Thinking creatively
- Problem solving
- Sequencing steps
- Logic and math skills
- Understanding people's needs

**xxiv** ABOUT THIS BOOK

- Patience and tenacity (sticking with it)
- At the moment, the world needs computational thinkers and good programmers, and becoming an App Inventor is a great place to start a career as a coder, game designer, or entrepreneur.

## Why choose the App Inventor language?

There are loads of choices of programming languages—LOGO, Python, Small Basic, JavaScript, Logo, Scratch, and Kodu. We think you should try them all! What's different about App Inventor is that it gives you access to some powerful hardware that you can carry in your pocket—a smartphone. That means you can create apps that

- Can do very cool things like use your GPS location, make phone calls, send texts, read barcodes, and take pictures or videos.
- Are useful in the real world. You might make an app that
  - Reminds you to do important things, like take medication
  - Sends an alert text with your phone's location
  - Uses pictures to tell small children what time of day it is
- Being able to create apps that your friends and family can use on their phones is an amazing motivator to learn to program and to make your apps the best they can possibly be. If you get really good, you can even sell your apps—for example, on the Google Play store.
- The other reason we're excited about App Inventor is that because it's a drag-and-drop graphical language, beginners tend to make fewer mistakes, and it's easier to spot them when you do (this is true of languages like Scratch and Kodu, too). It can be frustrating to hunt down missing periods or capital letters in typed programming languages—and that's one thing you don't need to worry about when you're starting out in App Inventor.

## What you need

We've included setup instructions in chapter 1, and there's more on the App Inventor website. To get started, you'll need

- A computer running Windows, Mac OS X, or Linux (we used Windows).
- A web browser. Firefox, Chrome, or Safari is fine (Internet Explorer doesn't work at the moment, but there are plans for an update soon).
- An Android phone isn't essential, but it makes things a lot more fun! A wireless internet connection or USB lead from your phone to your computer is required too.



- A Google account (more about this in chapter 1).
- Website resources. All the files we use (like graphics and sound clips) can be found at [www.manning.com/HelloAppInventor](http://www.manning.com>HelloAppInventor).
- Good ideas and a little patience!

## Ways of using this book

We've written this book in sequence so you get the essential ideas of computing and programming in an order that we think makes sense. The early examples in the book include step-by-step instructions (walkthroughs). Later, we assume you already know lots of things; and as you progress, the activities become more challenging as your skill level increases. The apps tend to get more complicated as you go through the book, too, and those in chapter 14 bring all the ideas you've learned together in two quite big and complicated apps.

If you're a beginner, we suggest that you work through the book in order; it probably makes the most sense that way. If you know a little about programming, you might find it useful to skip around and try different sections that interest you.

We've included "Try It Out" and "Taking It Further" sections that give you additional challenges. This is how you find out just how much you know! It's easy to follow a set of instructions, but can you apply those skills to something else? We strongly suggest that you try these exercises and suggestions. You'll also find some quiz questions at the end of each chapter to check that you understand the computing concepts covered.

## Symbols you'll see

We've used these symbols to highlight important information throughout the book:



*Learning points* give you general computer science concepts that you can research further. They explain the "Why is this important?" questions and give you keywords that you'll see in this book and others. They're also useful for teachers in planning lessons.



The *Let's invent!* symbol indicates the beginning of a practical activity. You'll see this whenever we start a new app.

**xxvi** ABOUT THIS BOOK

Some apps only work properly on a real phone—you can make a phone vibrate, but that won't work if you use the onscreen *emulator* (more about this in chapter 1). The *Phones only!* symbol means this part of an activity can only be tested on a phone rather than in the emulator.

## Code conventions and downloads

All source code in listings or in text is presented in a **fixed-width font like this** to separate it from ordinary text. Code annotations accompany many lines of code, highlighting important concepts.

The code used in this book, along with graphics and sound clips, is available from the publisher's website at [www.manning.com/HelloAppInventor](http://www.manning.com/HelloAppInventor).

## Author Online

Purchase of *Hello App Inventor!* includes free access to a private web forum run by Manning Publications where you can make comments about the book, ask technical questions, and receive help from the authors and from other users. To access the forum and subscribe to it, point your web browser to [www.manning.com/HelloAppInventor](http://www.manning.com/HelloAppInventor).

This page provides information on how to get on the forum once you're registered, what kind of help is available, and the rules of conduct on the forum. Manning's commitment to our readers is to provide a place where a conversation between individual readers and between readers and the authors can take place. We invite you to visit the forum and to share your questions and comments with the authors and other readers of this book.

OK, that's enough introduction—now let's go be App Inventors!