

**15** Fun Coding Projects!

# Coding for kids

**Second Edition**

- Learn the basics of coding
- Create apps and games
- No experience required



**Camille McCue, PhD**  
*Kids' coding teacher and author*

**dummies**<sup>®</sup>  
A Wiley Brand





# Coding for kids

2nd Edition

**by Camille McCue, PhD**

for  
**dummies**<sup>®</sup>  
A Wiley Brand

## **Coding For Kids For Dummies®, 2nd Edition**

Published by: **John Wiley & Sons, Inc.**, 111 River Street, Hoboken, NJ 07030-5774, [www.wiley.com](http://www.wiley.com)

Copyright © 2019 by John Wiley & Sons, Inc., Hoboken, New Jersey

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

**Trademarks:** Wiley, For Dummies, the Dummies Man logo, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

**LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.**

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit <https://hub.wiley.com/community/support/dummies>.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at <http://booksupport.wiley.com>. For more information about Wiley products, visit [www.wiley.com](http://www.wiley.com).

Library of Congress Control Number: 2019934591

ISBN 978-1-119-55516-2 (pbk); ISBN 978-1-119-55519-3 (ebk); ISBN 978-1-119-55522-3 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

# Table of Contents

<b>Introduction .....</b>	<b>1</b>
About This Book .....	1
Foolish Assumptions.....	2
Icons Used in This Book .....	3
Beyond the Book.....	4
Where to Go from Here.....	4
 <b>Part 1: Getting Started.....</b>	 <b>5</b>
 <b>Chapter 1: What Is Coding? .....</b>	 <b>7</b>
What Languages Will I Use? .....	8
What Does a Computer Program Look Like? .....	9
A Hello World! Example.....	10
Recipe for a Program .....	11
Planning a Program .....	13
Prepping Yourself to Code .....	15
Coding Cool Stuff.....	16
 <b>Chapter 2: Working with Programming     Languages and IDEs. ....</b>	 <b>18</b>
Basic IDE Setup and Navigation.....	19
Adding Hardware.....	33
Getting Fancier with User Interfaces .....	36
 <b>Chapter 3: When Things Go Wrong. ....</b>	 <b>40</b>
Syntax Errors .....	41
Logic Errors.....	42
Debugging Scratch Programs.....	43
Debugging App Lab Programs .....	45
Debugging MakeCode Programs.....	49
Commenting Out Code when Debugging.....	51

## Part 2: Sounds, Color, Random Surprises ..... 53

### Chapter 4: Orchestra ..... 55

Brainstorm .....	56
Sidebar: Event-driven programming .....	56
Start a New Project .....	57
Add a Backdrop .....	58
Add Instrument Sprites .....	59
Add a Singer Sprite and Modify Its Costume .....	61
Code Each Instrument to Play a Sound .....	63
Sidebar: Parallel processing .....	67
Save, Test, and Debug Your Program .....	67
Share Your Program with the World .....	68
Enhance Your Toy .....	68

### Chapter 5: Foley Sound Generator ..... 69

Brainstorm .....	69
Sidebar: User Interfaces .....	70
Start a New Project .....	71
Add a Background .....	72
Add Sound and Stop Sounds Buttons .....	75
Code the Sound Buttons to Play .....	78
Code the Stop Sounds Button to Stop Sounds .....	80
Save, Test, and Debug Your App .....	82
Share Your App with the World .....	82
Enhance Your App .....	82

### Chapter 6: Lucky Numbers ..... 83

Brainstorm .....	84
Start a New Project .....	85
Code Button A .....	85
Sidebar: Coding Randomness .....	87
Code Button B .....	87
Save, Test, and Debug Your Program .....	89
Transfer Your Program to the micro:bit .....	89
Share Your Program with the World .....	90
Enhance Your Toy .....	90

### Chapter 7: Mondrian Art Toy ..... 91

Brainstorm .....	92
Start a New Project .....	92
Add a Background Color .....	93
Sidebar: RGBA Color .....	95
Add a Title Label .....	96
Add Fill and Clear Buttons .....	97
Code a Canvas and Paintbrush .....	99
Code to Draw a Rectangle .....	101

Code to Fill Rectangles with Color.....	102
Code a Clear Button to Erase a Painting .....	103
Save, Test, and Debug Your App .....	105
Share Your App with the World .....	105
Enhance Your App .....	106

## Part 3: Moving from Here to There, Again and Again..... 107

### Chapter 8: Emoji Explosion ..... 109

Brainstorm .....	110
Start a New Project.....	110
Add a Backdrop.....	110
Add an Emoji Sprite .....	111
Code the Stage to Play a Sound.....	113
Code the Green Flag for the Emoji Sprite .....	115
Sidebar: Cloning and Inheritance .....	116
Code the makeEmojis Block.....	118
Code when I start as a clone for the Emoji Sprite .....	119
Code the explode Block for the Emoji Clones .....	121
Save, Test, and Debug Your Program.....	123
Share Your Program with the World .....	124
Enhance Your Animated Scene .....	124
Sidebar: Setting Position .....	124
Sidebar: Setting Direction .....	127
Sidebar: Moving.....	129
Sidebar: Simple Repeat Loops.....	130
Sidebar: New Blocks (aka Functions).....	131

### Chapter 9: Smelephant..... 133

Brainstorm .....	134
Start a New Project.....	134
Add a Backdrop.....	134
Add a Smelephant Sprite .....	135
Sidebar: Rotation Style in Scratch .....	137
Code the Green Flag Code of the Smelephant .....	138
Code the Smelephant's Up Arrow Key Control.....	139
Sidebar: Animating Shapes .....	141
Code Arrow Keys for Moving the Smelephant Down, Left, and Right.....	144
Add a Flower Sprite.....	145
Code the Green Flag for the Flower Sprite.....	147
Code the makeFlowers Block.....	148
Code when I start as a clone for the Flower Sprite .....	150
Code the getSmelled Block for the Flower Clones.....	151
Add a Monkey Sprite .....	153

Code the Green Flag for the Monkey .....	154
Code the chase Block .....	155
Save, Test, and Debug Your Program .....	158
Share Your Program with the World .....	158
Enhance Your Animated Scene .....	158
Sidebar: Key Control .....	159
Sidebar: Collisions .....	160
Sidebar: Show and Hide .....	162

## **Part 4: Variables, Simple Conditionals, and I/O ..... 163**

### **Chapter 10: Mascot Greeter. .... 165**

Brainstorm .....	166
Start a New Project .....	166
Sidebar: Inputs and Outputs (I/O) .....	167
Add a Backdrop .....	168
Add a Mascot Sprite .....	168
Add Text-to-Speech Commands .....	169
Sidebar: Strings and String Operations .....	170
Code the Mascot Sprite to Greet .....	171
Save, Test, and Debug Your Program .....	173
Share Your Program with the World .....	173
Enhance Your Program .....	173

### **Chapter 11: Weird Text Message ..... 174**

Brainstorm .....	175
Start a New Project .....	175
Name the Input Screen for the App .....	176
Add a Background Color to the Input Screen .....	176
Add an Instruction Label .....	177
Add Category Labels and Text Input Fields .....	178
Add a Button to Trigger the Action .....	181
Add and Name an Output Screen .....	183
Add a Message Image to the Output Screen .....	183
Add a Message Label to the Output Screen .....	185
Code the App .....	186
Save, Test, and Debug Your App .....	188
Share Your App with the World .....	189
Enhance Your App .....	189
Sidebar: Dilbert's Jargonator .....	190
Sidebar: ELIZA, the Turing Test, and AI .....	191



**Chapter 12: Vote Machine . . . . . 192**

Brainstorm .....	193
Start a New Project .....	193
Rename the Screen .....	194
Add a Title Label to the App .....	194
Add Images for the Candidates .....	195
Add Labels for Each Candidate .....	197
Code Variables for the First Candidate .....	198
Code the First Candidate to Register a Vote .....	199
Sidebar: Working with Number Variables .....	200
Code Variables for the Remaining Candidates .....	204
Sidebar: Changing and Incrementing Variable Values .....	205
Code Remaining Candidates to Register Votes .....	206
Save, Test, and Debug Your App .....	207
Share Your App with the World .....	208
Enhance Your App .....	208

**Chapter 13: Happy New Year! . . . . . 209**

Brainstorm .....	210
Start a New Project .....	210
Add a Backdrop .....	211
Add a Glittery Ball .....	211
Code the Ball to Drop .....	213
Create a Countdown Variable .....	214
Sidebar: Google Language Translation .....	217
Add Text-to-Speech and Translate Commands .....	217
Add a Cheer Sound to the Ball Sprite .....	218
Code the Countdown Clock .....	219
Sidebar: Decrementing a Variable .....	221
Save, Test, and Debug Your Program .....	223
Sidebar: Simple Conditionals and Booleans .....	223
Share Your Program with the World .....	224
Enhance Your Toy .....	224

**Chapter 14: Light Theremin . . . . . 225**

Brainstorm .....	226
Start a New Project .....	227
Code the First Sound Conditional .....	228
Code More Sound Conditionals .....	231
Sidebar: Advanced Conditionals .....	232
Save, Test, and Debug Your Program .....	236
Sidebar: IoT and Sensors in Circuits .....	236
Transfer Your Program to the micro:bit .....	237
Share Your Program with the World .....	238
Enhance Your Toy .....	238

## Part 5: Lists, Loops, and Logic ..... 239

### Chapter 15: Magic 8-Ball ..... 241

Brainstorm .....	242
Start a New Project.....	242
Code on start .....	243
Sidebar: Simple Lists (Arrays) .....	245
Code on shake .....	246
Save, Test, and Debug Your Program.....	248
Transfer Your Program to the micro:bit .....	249
Share Your Program with the World .....	249
Enhance Your Toy .....	249
Sidebar: eToys.....	250

### Chapter 16: Sock Sort ..... 252

Brainstorm .....	253
Start a New Project.....	253
Add a Backdrop.....	254
Add Red and White Sock Sprites .....	255
Add Mixed, Red, and White Lists.....	256
Code the Green Flag (Create List).....	258
Code the clearLists Block.....	263
Code the Sorting Process .....	264
Save, Test, and Debug Your Program.....	268
Share Your Program with the World .....	268
Enhance Your Program.....	268
Sidebar: Sorting Algorithms.....	269

### Chapter 17: Evil Olive ..... 272

Brainstorm .....	273
Start a New Project.....	273
Add a Background Image to the Screen .....	273
Add an Instruction Label .....	274
Add a Text Input Field .....	276
Create and Add Evil Olive to the Screen .....	277
Add a Message Label to the Screen.....	278
Code the App .....	279
Save, Test, and Debug Your App .....	282
Share Your App with the World .....	283
Enhance Your App .....	283
Sidebar: For Loops.....	284
Sidebar: Searching Algorithms .....	285

**Chapter 18: Sushi Matchup . . . . . 286**

Brainstorm .....	287
Start a New Project .....	288
Draw a Toy Interface on the Backdrop .....	288
Add a Button Sprite .....	292
Add Reels Sprites .....	293
Add a Status Sprite .....	296
Code the Button to Trigger the Spin .....	299
Create wear Variables .....	302
Add Sounds .....	304
Code the Reels to Spin .....	306
Code the checkMatch Block .....	309
Code the status Sprite .....	314
Save, Test, and Debug Your Program .....	315
Share Your Program with the World .....	315
Enhance Your Program .....	316
Sidebar: Broadcasting .....	316
Sidebar: Logical Operators .....	317

**Part 6: Onwards and Upwards .....** 319**Chapter 19: Creating and Sharing. . . . . 321**

Programming Your Own Ideas .....	322
Sharing and Showcasing Your Work .....	325

**Chapter 20: Where to Go from Here. . . . . 336**

Upping Your Game .....	337
Next Steps .....	340

**Index .....** 343



## Coding For Kids For Dummies

---

# Introduction

**So you want to** learn to *code* — awesome! *Coding* — writing computer programs — has something for everyone: creativity, logic, art, math, storytelling, design, and problem solving. From games and simulations to helpful tools and electronic gadgets, this book coaches you step by step through coding *real programs in real programming languages* that you can share with family and friends.

## About This Book

Many kids want to learn to code, but not every kid has computer programming classes at school or a camp he or she can attend during the summer. That's where this book comes in!

*Coding for Kids For Dummies* will help you learn all of the basic coding ideas and skills used by real computer programmers. Everything you do here will be useful in learning new skills and more advanced programming languages in the future. Best of all, the tools in this second edition are free, available online, and easy-to-use.

This edition of the book covers the following:

- ✔ **Scratch**, a learning language developed at MIT that has risen in prevalence to the point where it is arguably the most popular kid programming language available. As such, this book features numerous projects in the most recent version of Scratch — Scratch 3.0. Scratch is a block-based language that offers new coders an easy entry into computer programming. And it's fun!

- ✔ **JavaScript**, which is used in everything from apps to websites to electronics. New programming environments have made JavaScript more accessible than ever through interfaces that allow you to switch between block-based and text-based formats. You can begin learning in block-based mode (as in Scratch), and then transition to text-based mode as you build skills and confidence in coding. In this book, JavaScript projects are presented through two different vehicles (officially called *IDEs* — integrated development environments): Code.org's App Lab, for building mobile device apps, and MakeCode, for coding instructions to operate a small electronics board called a micro:bit.
- ✔ **Fundamental computer programming concepts**, which apply to both the projects in this book and additional coding (and, more generally, computer science) work you might pursue in the future.

Additionally, graphic design and animation are incredibly important skills that go hand-in-hand with coding to create great-looking and easy-to-understand digital tools. Although this book provides a little bit of guidance in these areas, the main focus of the content in these pages is coding.

## Foolish Assumptions

Hello person buying this book and reading this intro! I assume you are a kid who wants to learn to code. Awesome! You are starting on an adventure that will take you from being a user of technology to being a maker of technology. And it's a lot easier than you might think.

Here are a few other assumptions I make about you (or your technology) as you get started:

- ✔ You are comfortable typing on a tablet or a computer and using a mouse or touchpad. Your experience can be either on a Windows or Mac system because instructions for coding each project are platform-independent.

- ✓ You have an Internet connection and know how to open a web browser to access websites.
- ✓ For readers choosing to use the micro:bit electronics board, you have a USB port on your computer (via which you'll connect the micro:bit).
- ✓ You've played with a few apps, websites, or games on a computer, so you have some idea regarding how user interfaces (UI) look and how people interact with a computer via the UI.
- ✓ You're comfortable with basic math, math operations such as adding whole numbers, and logical operations such as comparing two whole numbers. I introduce algebraic variables in this book, but you don't need to have any prior knowledge of variables.

Lastly, if you struggle with spelling and punctuation — and you're operating in text-based mode — you may need to spend extra time troubleshooting your code for misspellings. The IDE for a programming language can give you clues about which commands it doesn't understand, but you will need to pay special attention to the details.

## Icons Used in This Book

As you work through the projects in this book, you'll see four icons. These icons point out different things.



TIP

The Tip icon gives you a tip that you can use to make your work easier. You'll see some tips over and over again.



REMEMBER

The Remember icon helps you remember and connect the coding concepts and skills you're working on with the big ideas of coding!



WARNING

The Warning icon tells you to watch out! It marks important information that may save you headaches.



The Technical Stuff icon lets you know more about the nuts and bolts of technical details and hardware help.

## Beyond the Book

On the Dummies.com website, I give you some extra goodies that you won't find in this book. Go online to [www.dummies.com/cheatsheet/codingforkids](http://www.dummies.com/cheatsheet/codingforkids) for a cheat sheet of coding commands in Scratch and JavaScript. (You can also type **Coding for Kids cheat sheet** in the search bar at [www.dummies.com](http://www.dummies.com).)

Download the information, print it, and keep it with your computer!

## Where to Go from Here

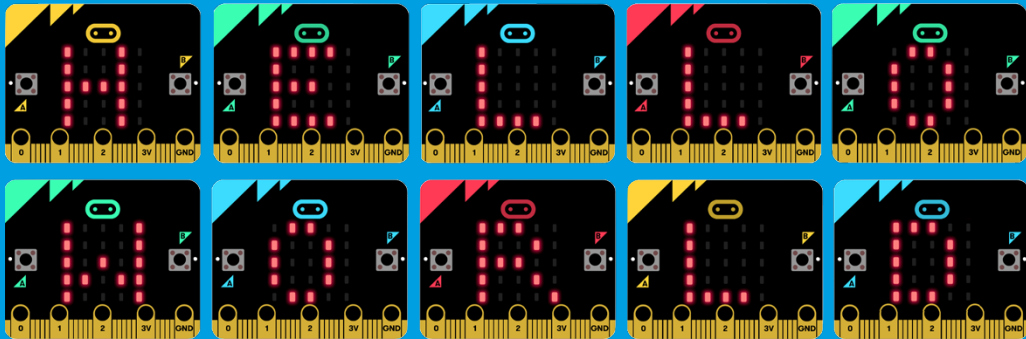
You can work on the projects in order, or you can jump around and work on any project you choose. After you gain a little experience coding, you can go in a bazillion new directions. Learn more advanced concepts in Scratch and JavaScript. Make up your own projects. Work on learning more advanced programming languages.

I hope this book inspires you to continue learning more about coding and making things with tech. Kudos on taking the first step! Now go get started!



# Part 1

# Getting Started

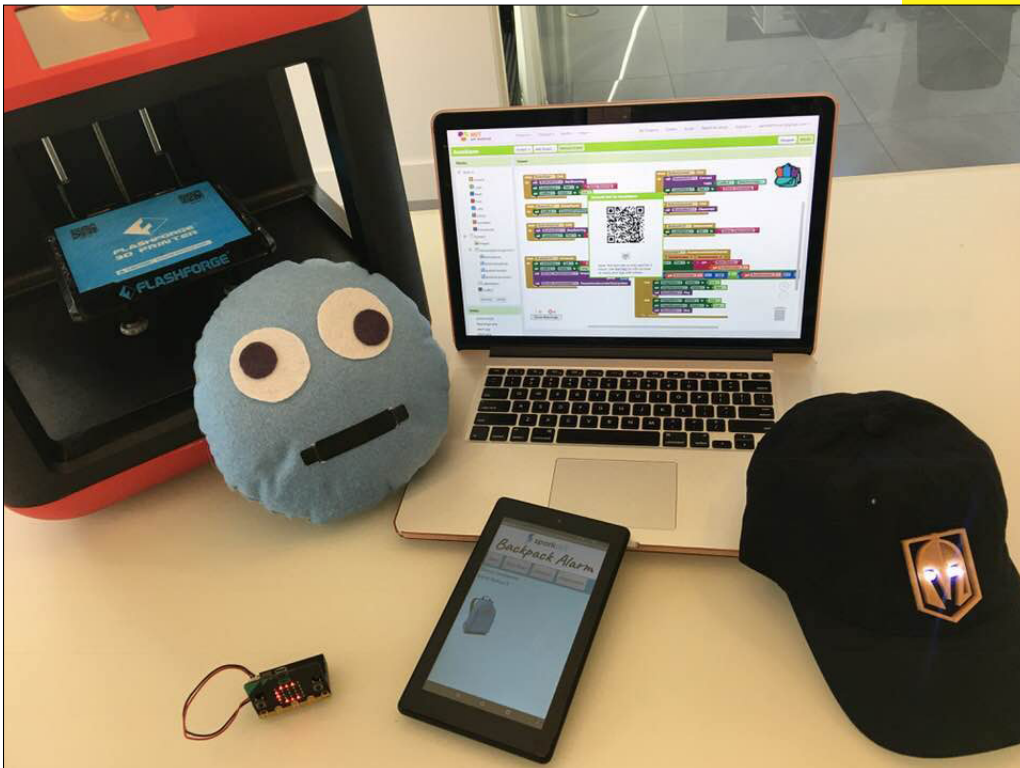


## In this part you'll . . .

- Discover the ingredients in a computer program
- Explore the basics of using Scratch, App Lab, and MakeCode
- Gain strategies for fixing code when things go wrong

# What Is Coding?

You know that *coding* has to do with building the apps you use and controlling the technology in your life, but what exactly is coding? *Coding*, also known as *computer programming*, is creating instructions for a computing device to do something. You use a language to communicate with other people, and computers use a coding language to communicate. And just as you can learn to write, speak, and understand languages different from the one you first learned as a toddler, you can learn to communicate in a coding language — so that you can “talk to” computers!



You're probably wondering whether coding is hard to learn. The answer is that it's easy to get started with coding, and easy to write real computer programs to perform all sorts of tasks! Unlike the early days of coding, when computer programmers talked to computers using long sequences of numbers (0s and 1s), you can now write code by using words and symbols that you can understand easily. For example, you can tell an app to play a sound three times ("boom boom boom!") with a command such as `repeat 3 (play sound boom)`. Neat, huh?

This type of human-friendly coding, which is called a *high-level language*, is what you will use when you're first learning to code. (Many professional programmers also use high-level languages.) Later, I talk more about high-level languages and the languages you'll use in this book.

You're probably also wondering whether you can make anything cool as a new coder. Yes, you can! In this book, you write code to build games, toys, and electronic gadgets. Everything you create you can play with and share with your friends and family.

## What Languages Will I Use?

This book is filled with great projects you can do to learn the basics of coding and make real apps. You'll be using two programming languages to code: Scratch and JavaScript.

In the Scratch language, you build code with *blocks* (also called *tiles*) that snap together to make complete programs. Scratch is a learning language, created especially for kids, and has its own *integrated development environment (IDE)*, which is a fancy name for a place where you write and test code.

JavaScript is a professional programming language that real coders use to make all sorts of things from apps to websites. You will use two easy-peasy IDEs to code in JavaScript: App Lab and MakeCode. As with the Scratch IDE, these JavaScript IDEs let you work in block mode, snapping together your coding commands.

But when you feel ready to tackle text-based coding — typing your commands — you can switch to text mode in App Lab and MakeCode.

You can create your code in Scratch, App Lab, and MakeCode on any computer or tablet. Just be sure you have a good Internet connection, and you're ready to go! You learn more about the basics of working with each language and programming environment in Chapter 2.

## What Does a Computer Program Look Like?

A *computer program* consists of the instructions you code to make a computer do something. A program looks like a list of steps, filled with words and symbols. Many words in the list will be familiar, such as *for*, *if*, and *forever*. Words in a computer program are called *commands* because you're commanding the computer to perform some sort of action. Some commands look like combinations of words you know, smushed together into new words. For example, JavaScript uses the `onEvent` command to find out whether a user has pressed a button.

You might also recognize many of the symbols in a computer program. These look like operators you use in math class (+, −, >, =) and also like punctuation marks you use in English class, such as a period (.) and a semicolon (;).

All of the commands and symbols in a computer program are organized in a special order so that the computer can understand what it is supposed to do. Planning that order and then coding it is a bit like writing an essay, solving a math problem, performing dance choreography, or running a play in football. You have to put together and *execute* (run) the program in a specific order — you can't just put the instructions anywhere and expect the program to work correctly.

## A Hello World! Example

Historically, the first computer program a new coding student writes is one that prints the words “Hello World!” on the computer screen. Figure 1-1 shows an example of what that code looks like in Scratch, and its resulting *output* (what it displays onscreen).

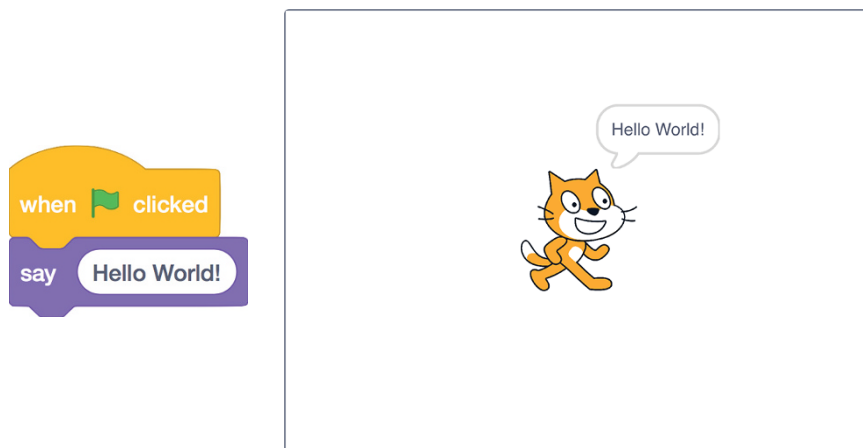


Figure 1-1

Figure 1-2 shows the same code in JavaScript (in block mode and in text mode) using the App Lab IDE, and its resulting output.

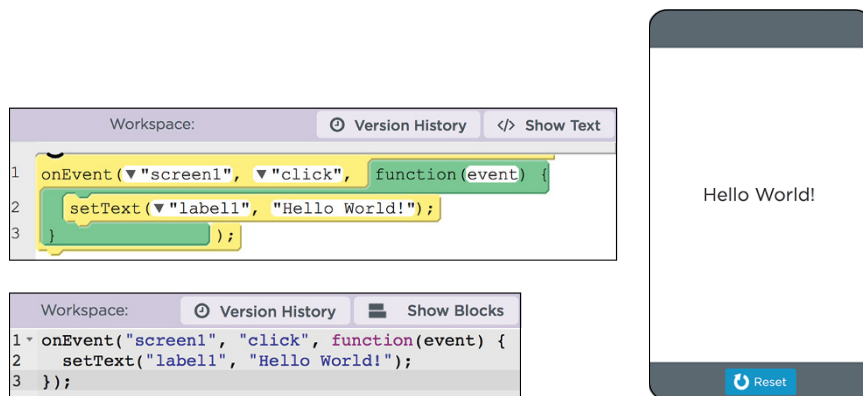


Figure 1-2

Figure 1-3 shows the same code in JavaScript (in block mode and in text mode) using the MakeCode IDE, and the output when displayed on the micro:bit electronics board. Because the micro:bit can scroll only one letter at a time, the figure displays only the letter *H* at the beginning of “Hello World!”

You’ll be making little programs like this, and much bigger programs too, in no time!

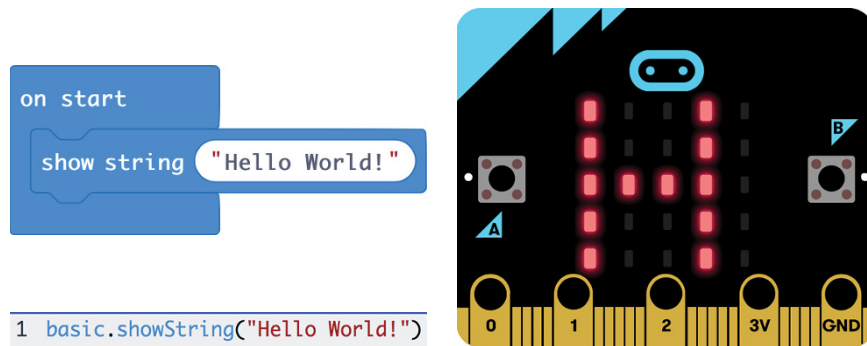


Figure 1-3

## Recipe for a Program

Many mornings, I cook chocolate chip pancakes for my family, following a recipe I created myself. A recipe is like a computer program, and following the recipe is like executing the program. The recipe has parts, including gathering and measuring ingredients, mixing the ingredients to make the pancake batter, and then dropping spoonfuls of the batter onto the griddle to cook it. Similarly, a computer program has parts such as asking the user for information, doing something with that information, and then telling the user the result.

Within each part of a program, you write small chunks of code to perform different processes. A chunk of code that performs a task is called an *algorithm*. For example, one algorithm I perform when making pancakes is testing the surface temperature of the griddle: I plug in the griddle, set it to a certain temperature, and

drip a few droplets of water onto the surface to see how quickly they evaporate.

Constructing algorithms is important in coding programs to run on a computer. Think about the types of algorithms you might make in a favorite game you play on your phone. For instance, an algorithm you might code in a Yahtzee game is rolling the dice. Or an algorithm you might code in a Space Invaders game is flying a spaceship across the sky every so often.

The algorithms you create connect with each other to build your entire program. As coders, we have three fancy terms to describe how our algorithms connect: sequence, selection, and repetition. Here's what each means:

- ✔ **Sequence:** The order in which a process is conducted. Every computer program must be organized so that steps are executed in a logical order. For example, when making pancakes, I must run my algorithm for making batter before running my algorithm for cooking the batter!
- ✔ **Selection:** Choosing a path based on certain conditions. For example, when choosing a movie to attend this weekend, you might decide between an action movie or a comedy. The decision you make then directs you to new sequences — and, consequently, other selections — that relate to your choice. Selection is often coded using conditional statements structured this way: if [condition occurs] then [consequence occurs]. These conditional commands let you create as many paths as needed to respond to the conditions of the program.
- ✔ **Repetition:** The process of repeating something. When you repeat code, you make it loop over and over again. A *loop* is a structure that tells the computer to run the same commands multiple times, without the need to rewrite those same commands. You already know how loops work: In a song, the drumbeat is looped to provide a continuous rhythm pattern from the first note of the song to the last.



Throughout the book, you'll see references to sequence, selection, and repetition. Check back here to refresh your memory of how each is used when coding a program.







## Planning a Program

As you plan and develop your programs, you'll need to have some organized way of writing them down. You can use several methods to represent a program before you translate it into code for use on a computer.

Some people like to draw a picture or a series of pictures (called a *storyboard*) to show how an app, a game, or a website will look onscreen. This type of work is often performed by *graphic designers*, the people who make the images and the animations for computer programs. You will be doing your own graphic design work for the programs you code in this book, so think about doing a little drawing before you touch the keyboard to code.

Other people like to build a flowchart or write pseudocode when planning their programs. A *flowchart* is like a little map with special boxes and arrows that describe the main parts of the program. Table 1-1 shows some of the most important flowchart symbols and what each symbol represents.

**Table 1-1 Symbols Used In Flowcharts**

Symbol	Name	What It Means
	Arrow	Shows the program sequence
	Terminal	Starts or ends the program
	Process	Performs a task
	Decision	Makes a decision, such as yes or no
	Input/output	Accepts input or produces output
	Preparation symbol	Sets up a loop counter

In this book, a couple of programs are planned by using a flowchart. Figure 1-4 is a simple example of a flowchart for a program in which a user searches for, and plays, a song. The great thing about planning a program with a flowchart is that it helps you think about the overall operation of the program. You can think visually and leave writing code for later!

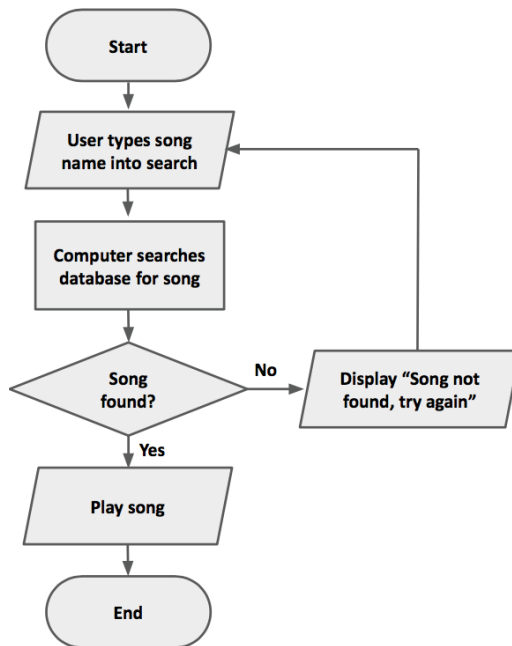


Figure 1-4

Another strategy for planning a program is to jot down your overall ideas in a simplified form of computer code, called *pseudocode*, which means *fake code*. It's not code that the computer can run, but it is written similarly. When writing pseudocode, you don't have to worry about punctuation such as semicolons or curly brackets, so it keeps your mind on the overall operation of the program — not the fine details.

For example, if you code part of a *Simpsons* game in which Homer is eating donuts, you might write some pseudocode that looks like this:

```
create variable donuts = 0

if Homer eats donuts
    then add one to donuts

if donuts > 10
    then print "Stop eating donuts!"
    else print "Have another donut!"
```

Whether you choose to draw pictures, flowchart your program, or write pseudocode, it's a good idea to put your plan on paper in some organized form *before* you start coding.

## Prepping Yourself to Code

Although coding is about creating new ideas and bringing them to life through computer programs, remember that it's also about your mindset as a coder. Just as you prepare yourself to run a race or compete in a spelling bee, you can do certain things to get ready to code. Here are a few:

- ✔ **Follow the examples.** The algorithms you learn by coding example programs apply to millions of other programs. For example, learning to make a score increase in one game builds your skill in coding that same process in any game you make. Good coders scour the web and reuse code snippets they find online. Learn by example!
- ✔ **Think top-down.** Start at the top and work your way down when you're developing a computer program. Don't drill down to the nitty-gritty details of a program when you're first brainstorming a new app. Start by mapping out the overall plan. Then get more specific by drawing pictures or storyboarding your designs. Then create a flowchart. Then write pseudocode. Finally, code the app piece-by-piece.

- ✔ **Practice patience and resilience.** No matter what you're making — a computer program, a musical performance, or a gourmet meal — patience is required to learn a new skill, and resilience is required to bounce back from challenges to master that skill. You'll need a good dose of both these traits.
- ✔ **Cultivate your creativity.** Coding is not the cold, calculating discipline many people think it is. From crafting new solutions to a problem to inventing new video games, creativity is a big part of the coding process. Explore as many creative ventures as you can to inspire your programming. Lift your head up from the computer screen, look at the world around you, and listen to the sounds of life. Moving away from your code for a while enables you to come back to it with a fresh new perspective!
- ✔ **Know that debugging is half the process.** Debugging code means tracing and retracing your steps — sometimes by tracing through each line step-by-step, sometimes by isolating and testing smaller sections of code, and sometimes by testing sample data to examine the output — to find and fix problems. You will spend a lot of time debugging your code to get it fully operational. Managing your frustration during the debugging process is vital to being a successful coder. Keep calm and carry on!

## Coding Cool Stuff

Coders make the apps you buy in the app store, but what else do they make? The list of cool things you can create with code is long! For the computer, you can build online games, personal and small business websites, and virtual tours of your photo galleries. For everyday gadgets, you can build backpack alarms, handheld games, health monitors, weather sensors, pet trackers, and remote fish feeders.

If you're into textiles and sewing, you can code programs that control LED lights sewn into fabric so you can craft crazy Halloween and spirit day costumes that literally light up any room you enter! If you're all about security and secret agent activities, you can go cyber, learning Internet basics and encryption techniques to keep online transmissions secure, and information private.

If you have a flair for the theatrical, you can write code to control robots to dance onstage while flying synchronized drones overhead in a stage production. If you love home electronics, you can code devices for use in your smart home, from camera monitors to mood music automated by time of day.

If you want to get wild, you can design, wire, and code an entire yard full of holiday lights, festive music, projection images, and moving figures for the annually televised Great Christmas Light Fight!

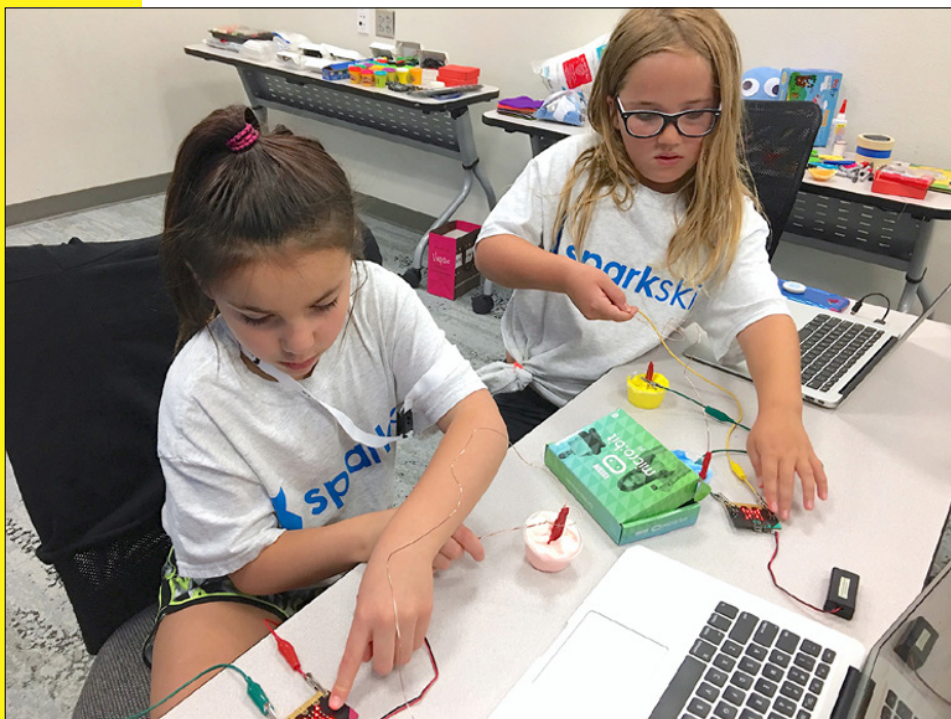
And what about those apps in the app store? You can make all sorts of games, from puzzles to wordplay to arcade games. You can build helper tools that assist people with their daily lives. If you're a good Samaritan, you can code apps to connect people in need with available resources, especially in times of crisis. The programs you can code are limited only by your imagination and your skill set.

As a young coder, you can build simple versions of many of these amazing toys and tools today. By doing so, you'll not only have fun and feel the satisfaction of making a real, usable product but also build the foundation for your future.

# Working with Programming Languages and IDEs

**As a coder, you** will work with lots of programming languages. Just like a traveler must know different languages while going from one country to another, you must know a variety of coding languages for different types of projects you'll create. For example, different languages are used for coding a mobile app, a website, a Nintendo Switch game, and a self-driving car. The projects in this book are written in two coding languages that are great for new coders: Scratch and JavaScript.

Scratch is a teaching language made of blocks that you snap together to create programs. You program in Scratch using a website *IDE*, or *integrated development environment*, which is an all-in-one place to build and run your code. The programs you make in Scratch will run on computers and tablets.



*JavaScript* is a professional language that is text-based, meaning you type commands to create your programs. But don't panic! In this book, you use online IDEs that let you choose whether you want to program JavaScript using text mode or block mode. When you code in the App Lab IDE, you make programs that run on computers and mobile phones. When you code in the MakeCode IDE, you code programs that power an electronics board called a micro:bit. Your programs run on a simulator on the computer screen and — if you want — also on a real micro:bit.

This chapter helps you set up and understand the basic layouts of Scratch, App Lab, and MakeCode. You also learn a little bit about working with images and sounds, so that you can build fun and engaging *UIs*, or *user interfaces*, for your projects.

## Basic IDE Setup and Navigation

Scratch, App Lab, and MakeCode are all free, but you need to set up accounts in Scratch and App Lab before you can start coding. MakeCode doesn't require an account.

### Setting up your account in Scratch

It's quick and easy to set up an account in Scratch. Just follow these steps:

1. In any web browser, navigate to <https://scratch.mit.edu>.
2. On the Scratch home page, select Join Scratch.
3. In the Scratch dialog box, type a Scratch username and a password, as shown in Figure 2-1. Then click the Next button.
4. Type your birthdate, gender, and country. Then click Next.
5. Type the email of your parent or guardian. Then click Next.

A screen appears letting you know that you are signed up to use Scratch and that a confirmation email has been sent to your parent or guardian.

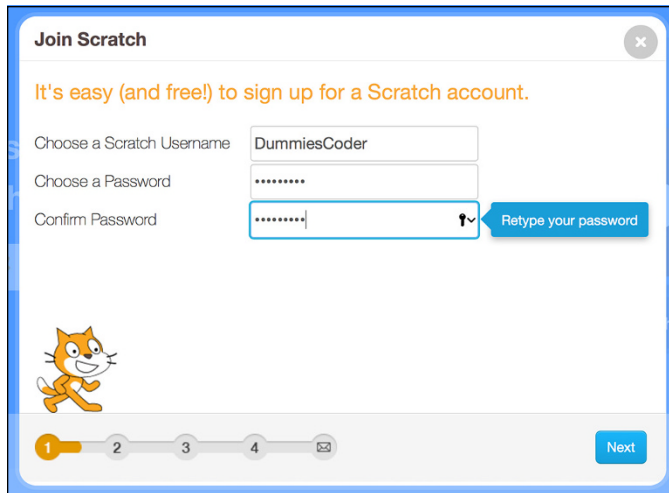
The image shows a web form titled "Join Scratch" with a close button (X) in the top right corner. Below the title is a message: "It's easy (and free!) to sign up for a Scratch account." The form contains three input fields: "Choose a Scratch Username" with the text "DummiesCoder", "Choose a Password" with masked characters "\*\*\*\*\*", and "Confirm Password" with masked characters "\*\*\*\*\*". A blue button labeled "Retype your password" is positioned to the right of the "Confirm Password" field. At the bottom left of the form is the Scratch logo (an orange cat). At the bottom right is a blue "Next" button. A progress bar at the bottom shows four steps: 1 (highlighted with an orange circle), 2, 3, and 4, followed by an envelope icon.

Figure 2-1

6. Your parent or guardian must open the email and confirm that you're permitted to share your work publicly on Scratch.

If the adult doesn't confirm, you can still work in Scratch — but you won't be able to share your programs.

7. Click OK on the final screen to complete the sign-up.

After your Scratch account is set up, you can log in to your account at any time by clicking the Sign In button in the upper-right of the Scratch home page and typing your username and password.

## Getting around in Scratch

After you set up your Scratch account, you are taken to the Scratch home page. Here are some of the things you'll see, and some of the actions you can perform.

### The Scratch IDE

Select Create on the Scratch home page to open the Scratch IDE. In the Scratch IDE, you see a new, blank project, as shown



in Figure 2-2. This is also the same screen you see whenever you choose File⇨New when working in Scratch.

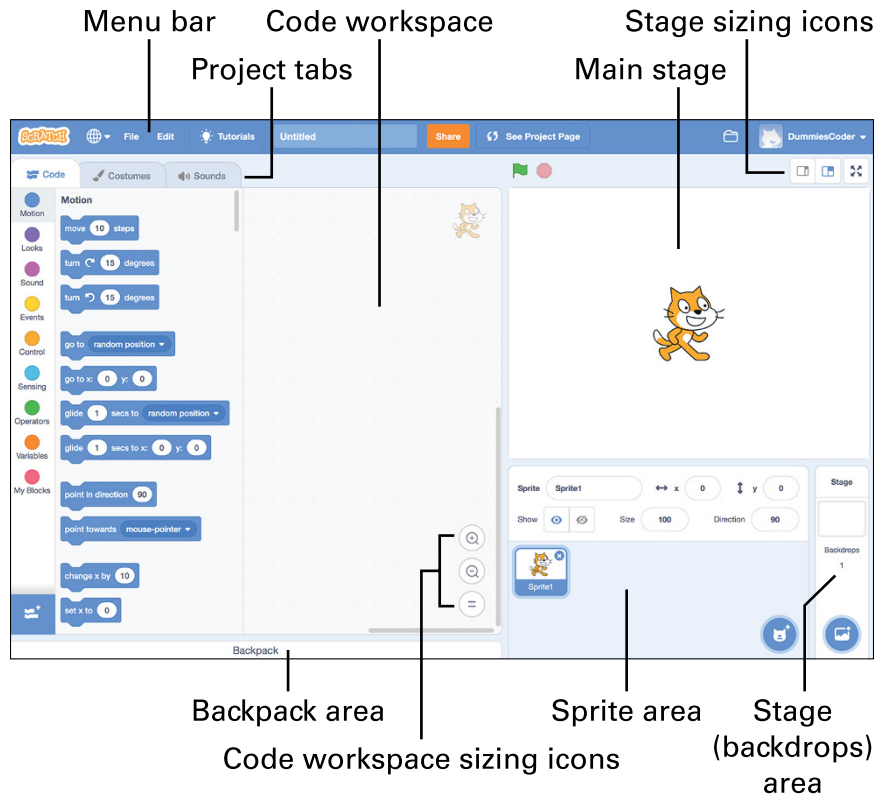


Figure 2-2

The *menu bar* at the top of the Scratch IDE features just a few choices. These are the most important:

- ✓ **File⇨New:** Create a new project.
- ✓ **Edit:** Allows you turn Turbo mode on or off. When on, this mode executes the program at the fastest possible speed.
- ✓ **Project name field:** Name your project.
- ✓ **Share button:** Share your project (see Chapter 19).

✓ **See Project Page button:** Go to the project page for your current project, where you can give users directions on how to use your program. On the project page, click the See Inside button to get back to the Scratch IDE shown in Figure 2-2.

✓ **My Stuff folder icon:** Open projects you've created.

The *tabs* on the left side of the Scratch IDE change depending on whether you are creating code for a *sprite* (an object in Scratch) or for the *stage* (the backdrop where your sprites live):

✓ **Code tab:** Access command categories and their blocks.

✓ **Costumes tab:** Access sprite costumes and the sprite editor for these images.

✓ **Backdrops tab:** Access backdrops for the stage and the backdrop editor for these images.

✓ **Sounds tab:** Access sounds and the sound editor.

The *workspace* is the place where you drag command blocks into and assemble command blocks together to make larger *code blocks*. Click the sizing icons to zoom in or zoom out as you build code.



TIP

The Backpack area at the bottom of the Scratch IDE is a place where you can save, store, and reuse project items, sharing them among projects. Just drag a code block, sprite, costume, or sound into the Backpack and it will be available in all projects! Drag any item out of the Backpack for use in any project. Delete an item in the Backpack by right-clicking (Windows) or Ctrl-clicking (Mac) the item.

The *main stage* is the large window that shows you what your user sees. It consists of the sprites and the backdrop. Click a stage sizing icon to resize your view of the stage. Just above the stage are the *green flag* and the *stop* icons; use these to start and stop, respectively, most projects you create.

The *sprite area* displays the sprites (objects) in your project. Each time you start a new project, the Scratch Cat sprite is placed in your project by *default* (something the computer does

automatically). Information about this sprite is located just below the stage. The name of the default Scratch Cat sprite is Sprite1. You can find out additional information about Sprite1 in the Sprite area, including position (the *x* and *y* values), *size*, *direction*, and *show* (whether the sprite is showing or hiding, indicated by the eyeball icons).



If you have more than one sprite in your project, each sprite will appear in the sprite area. The *active*, or selected, sprite has a blue outline around its icon. Clicking a sprite icon to make it the active sprite allows you to write code for that sprite. To add a new sprite, click the *choose a sprite* icon (shown in the margin). To delete a sprite, click the X in the corner of the sprite icon.

The small *stage*, or *backdrops area*, is located in the lower-right corner of the Scratch IDE. Here, an icon shows the current backdrop. Clicking this icon makes the stage active, which means you can write code for the stage, change its background, and add sounds to it.



WARNING

Only one sprite or the stage can be active at a time! Pay attention to which of these is currently selected. If you're writing code, you're writing it for the currently selected object.

### Code tab

Figure 2-2 shows the Code tab of the Scratch IDE. You use this tab for both sprites and the stage. It is organized by command categories, as follows:

- ✓ **Motion:** Commands to tell sprites (objects) how and where to move. (The stage does not have motion commands.)
- ✓ **Looks:** Commands to change the costumes of sprites and backgrounds.
- ✓ **Sound:** Commands to play music and sound effects.
- ✓ **Events:** Commands to start and end code execution.
- ✓ **Control:** Commands to select code or repeat code for execution. (See Chapter 1 for help on code sequence, selection, and repetition.)

- ✓ **Sensing:** Commands to sense color, sound, position, or user input.
- ✓ **Operators:** Commands for math and logic operations.
- ✓ **Variables:** Commands to create and change variables.
- ✓ **My Blocks:** New commands you define for your project.

To write code for a sprite or for the stage, select a command category and then drag and drop one block into the workspace. Add the next command by dragging a new command below and then snapping it (bringing it close to) the previous command to build a code block. To *execute*, or run, the code block, click the block. The code block will “light up” with a bright yellow outline, and the commands in that block will execute.

You can remove a command block in Scratch by clicking it and dragging it back into the command categories.



TIP

You’re using Scratch 3.0, the version released in 2019. It has some cool new features, including text-to-speech and language translation. Click the Add Extension icon below the code categories to check out these features. Also, in this new version, you can create and run code on both computers and tablets!



TIP

Scratch automatically saves your projects, but you can choose to save at any time by choosing File⇨ Save Now from the menu bar.

## Setting up your account in App Lab

Follow these steps to set up an account in App Lab:

1. In any web browser, navigate to <https://code.org>.
2. On the Code.org home page, click the Sign in button.
3. On the Sign In page, shown in Figure 2-3, click the Create an Account button.
4. Complete the form on the page that appears, and then click the Sign Up button.

A kid can sign up as a student, or a parent can sign up as a teacher. Due to the difficulty level (the inclusion of text-based code), App Lab is designated as a 13+ area of Code.org.

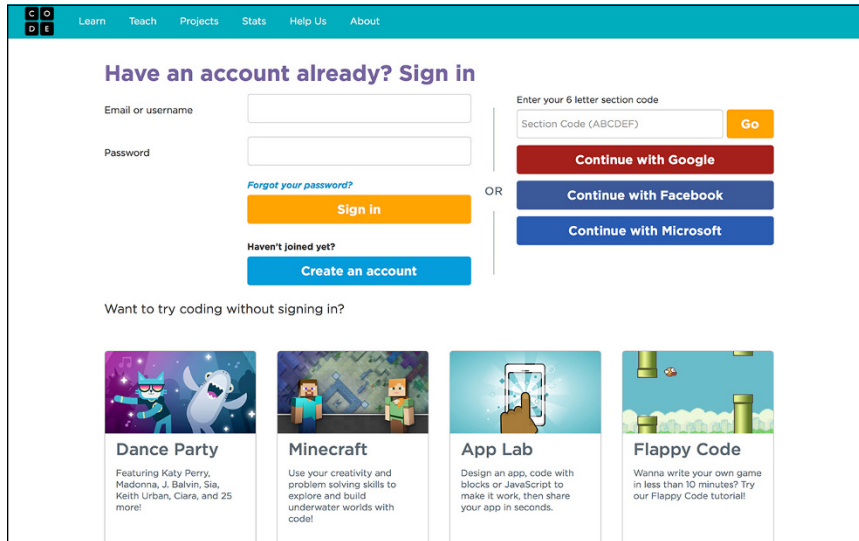


Figure 2-3

5. On the My Dashboard screen, scroll down and click the App Lab icon.

After your Code.org account is set up, you can log in to your account by clicking the Sign In button in the upper right of the Code.org home page and then typing your username and password.

## Getting around in App Lab

After you set up your Code.org account, you can sign in at any time and then access App Lab from My Dashboard. Here are some things you can see and do in App Lab.

### The App Lab IDE

App Lab opens with a new, blank project, as shown in Figure 2-4. This is also the same screen you see whenever you click the Create New button in App Lab.