CHAPTER 2
EDITING IN TOUCHDEVELOP

In this chapter we describe how to set up TouchDevelop on your phone, how to enter scripts from scratch, and how to extend existing scripts. At the end of the chapter, you should be familiar with the tiles and buttons that TouchDevelop uses on the phone, and the ones it provides to make programming feasible, and in fact really easy and quick. You will also be shown how to publish your script in the TouchDevelop Bazaar.

2.1 THE TOUCHDEVELOP SCRIPT EDITOR
2.2 CREATING A SCRIPT
2.3 EXAMPLE: DRAWING RANDOM CIRCLES
2.4 MODIFYING A SCRIPT
2.5 MORE EDITING ACTIONS
2.6 USING THE TOUCHDEVELOP BAZAAR
2.7 CAUTION! PROTECTING YOUR INFORMATION
2.8 EXERCISES

2.1 THE TOUCHDEVELOP SCRIPT EDITOR

TouchDevelop includes an editor for creating its scripts. It has been optimized for use with a small touch-sensitive screen as found on a phone. It works by presenting you with only a small number of choices at a time. At every stage, only relevant choices are presented to the user. If there are too many choices to fit on the screen, the most frequently used choices will appear on the first screen and then there will be more choices on a second screen, and so on.

The editor takes a little bit of getting used to. For that reason, it is suggested that you try to follow the sequence of steps shown below for entering a new script named 'add n numbers'. However, before you do that, let's introduce one piece of terminology. The phone has an arrow button at the bottom left, below the screen. We will call that the phone back button; it will be used a lot when entering scripts. Remember – this button is not on the screen itself! Do not confuse it with any other left arrow buttons which may be displayed on the screen.
The editor actions given in the examples below provide just one way of entering the script. There are many alternative paths for getting to the same end result, and you should play around with the editor to find what works best for you.

MISTAKES! If you make a mistake when entering a script, do not worry. There is an UNDO button on every edit menu. Failing that, the PHONE BACK BUTTON will always return you to the previous screen where you can delete an unwanted line of code and try again.

The first script we have chosen is the familiar one to “add n numbers” and the second one is to draw some random circles on the screen. Entering these programs will use several of the different editor actions that you will need later on, so that you can refer back to these steps later when you are entering other scripts. Because the TouchDevelop editor anticipates the possible options that follow on from any given command, and gives you a choice of the next items, you end up actually typing in less than half of the keystrokes that you would for a traditional program on a computer, laptop or tablet. This is the beauty of TouchDevelop, and one we hope you will appreciate as your experience grows.

While reading this material, it might also be useful to watch the Channel 9 video of entering a program, which can be found on:

http://touchdevelop.com/help/video/gettingstarted

### 2.2 Creating a Script

Our first example is a sequence of steps which cause a simple program to be entered as a TouchDevelop script. The program has the following structure:

- set the variable `SUM` to zero;
- for each of the numbers from 1 to N: add the number to `SUM`
- display the final value of `SUM`

It simply sums the integers. If we run such a program with N equal to 5, the answer displayed on the screen would be 15 because 1+2+3+4+5 sums to 15.
START A NEW SCRIPT

1. Start TouchDevelop. (On the home screen, tap the right-pointing arrow at the top right and scroll down to TouchDevelop in the list of apps, and select it.)

2. Tap the plus button at the bottom of the screen – it means you want to add a new TouchDevelop script.

3. Enter the title for the program which, for this example, is “add n numbers” in the pop up text box. Then tap the CREATE box. The screen now contains the following:

```
START THE MAIN ACTION
```

4. Tap the word main in the list of actions. Do not tap the right-pointing arrow in the green square because that causes the script to be played and, at the moment, it does not do very much.

   You should now be looking at the code for an action named 'main'. This is a default line of code which displays 'Touchdevelop is cool!' if you play the script.

   Note that in the code, you will see that the full text of the string constant "TouchDevelop is cool!" is not shown on the screen. That's because the screen is too narrow to display it all. It is shortened to "TouchDevelop is ".... If you were to later modify that scree, you will find that all the characters of the string constant are actually there. We choose to leave this line of code in the script for now and delete it only after we have finished adding our own statements to the script.

ADD A NEW PARAMETER TO AN ACTION

If the main action has a parameter, the user will be prompted to enter a value for that parameter whenever the script is run. We are now going to add that parameter and give a reasonably meaningful name to that parameter. (We’ll choose ‘n’, because it is short.)

5. Tap the top line of the code for the main action – this is the SIGNATURE LINE (the line which begins with the keyword action). The screen display should change to look like the following:
6. Now tap the button labeled **edit** and the screen should display a list of ACTION PROPERTIES for the main action, looking like this.

7. Tap the **params** tab. This selects params as the pivot point for the display.

8. Tap the **plus** button at the bottom of the screen, and a new parameter named \( x_1 \) with type **Number** is added to the action. The name and the type are both supplied as defaults and both can be subsequently changed. We will change only the name.

9. Tap the name \( x_1 \) and enter a new name ‘\( n \)’ into the textbox. Tap the **phone back button** to exit the keyboard mode.

10. Tap the **phone back button** again to return to the screen showing the code for the main action.

**INSERT AN ASSIGNMENT**

To enter an assignment, we can first choose the assignment operator, then enter the right hand side expression. A new variable for the left hand side is generated automatically. We will change the name of that variable afterwards.

11. Tap **anywhere** on the line of code which says "TouchDevelop is"... → post to wall" and then tap the **edit** button. A bar colored with your theme accent color should appear to the immediate left of this line.
12. Tap the **plus button** in the **LOWER ROW** to add a line below the current line, and then tap the **expression** button. (The plus button in the upper row is for inserting a new line above the current line.) An edit menu for constructing an expression is now displayed.

13. Tap the button with the label “1 2 3 ...” (the **NUMBERS** button).
14. The menu changes to show characters needed for entering a number. Select the **0 button**, and now an assignment statement of the form “*x0 := 0*” has been created. (If we keep tapping digits or the decimal point, the number on the right-hand side changes.)

15. **Tap in the fix-it bar** on the screen to accept the new expression as being the right-hand side of an assignment to a new variable (whose name has now defaulted to *x*).
16. We choose not to retain the default name for the left-hand side. So tap the **phone back button** to return to the edit-expressions menu where there is a button labeled **RENAME**. We tap the button and the name “*x*” appears and is highlighted in red.
17. We set the new name “sum” by tapping letters on the **keyboard**.
18. To see the modified line of code, tap the **phone back button**.

**INSERT A FOR STATEMENT**

19. Tap in the black space below that line of code again and the edit menu changes. There are only two possible choices, both for adding a new statement to the main action. Tap the choice labeled “*if/foreach ...*”. It’s called the add flow button.
20. The edit menu and its placement changes. There is a thick red line below the assignment statement to indicate where we are about to insert a new line of code.

```
if
  condition is true
then
  -post to wall
var sum := 0
```

Tap the button labeled “for” and the first line of a for loop is inserted.

```
for 0 ≤ ... < i do
```

Although you cannot see it yet, the loop index variable has been given a default name of ‘i’. The cursor position indicates that the editor expects us to enter the upper bound for the loop index next.

**ENTER A SIMPLE EXPRESSION**

We are going to enter the expression $n+1$ as the loop’s upper bound in three steps. (It is $n+1$ because the loop only repeats while $i$ has a smaller value, so the last iteration occurs with $i$ equal to $n$.)

21. Find a button labeled ‘n’ in the edit menu and tap it. The button represents the parameter $n$ which we added to the main action back in steps 6 to 8. If ‘n’ isn’t on the first screen of choices, tap the right arrow, NEXT SUGGESTIONS, to see a second set of choices.

22. Tap the button labeled with the operators “+ − * / ...”, and then select “+”.

23. Tap the numbers button and select “1”.

**ADD A LINE TO THE LOOP BODY**

We are going to provide the statement “sum := sum+i” as the body of the loop.

24. Tap between the line of code and the number entry pad, in the blank area, and two edit choices appear below. Select the “add expression” button which is labeled with a big ‘+’.

25. In the edit menu, select the “sum” button.

26. Select the named operators button labeled “and, or not, wall, ...”, and then select the “:=” button.

27. The cursor position indicates that we should enter the right-hand side of the assignment next. We will construct $sum+i$ as that expression in three steps. This is very similar to entering the $n+1$ expression above. (I.e., Select the “sum” button. Select the operators “+ − * / ...” and choose “+”. Finally, select the ‘i’ button.)
ADD A LINE AFTER ANOTHER ONE

We now want to add a final statement to the script, one which displays the final value of the variable sum.

28. Tap the phone back button below the screen. This causes all the code for the main action, as entered so far, to be displayed.
29. Tap anywhere on the line which begins with the word 'for'. It causes the entire loop to be selected for editing.
30. Tap the ‘+’ button below the loop (it reads “ADD BELOW” as its name), and then select the expression button.

CONSTRUCT A STRING EXPRESSION FROM SEVERAL PIECES

31. Select the operators button labeled “+ * / ...” and then choose the left parenthesis “(“. This causes a parenthesis pair “( )” to be inserted with the blinking cursor in between.
32. Now tap the button named “STRINGS, ART” and labeled “ ‘...’ pic snd color “, and select the “STRING” button labeled "abc". Enter the text “The sum of the first ” (but without the quote characters – these are supplied automatically).
33. Tap the phone back button.
34. Tap the named operators button and select the “||” operator (string concatenate).
35. Tap the “n” button.
36. Tap the named operators button “and, or not, wall”.
37. Tap the “||” button (string concatenate).
38. Tap the “strings, art” button.
39. Tap the “string” button and enter the text “ numbers is “.
40. Tap the phone back button.
41. Tap the named operators button and then the “||” button (string concatenate).
42. Tap the “sum” button.

At this point, the entire string expression should read

("The sum of the first " || n || " numbers is " || sum)

ADD A POST-TO-WALL CALL TO THE STRING

43. Tap the right-hand “move cursor button” (the RIGHT pointing arrow) to get the cursor to the desired position – which is just to the right of the right parenthesis.
44. Tap the “post to wall” button.
45. Tap the phone back button to see all the code for the main action at this point.
CUT A STATEMENT (AND PASTE THE STATEMENT ELSEWHERE)

The main action still contains the default first line which displays the message “TouchDevelop is awesome” when the script is played. Let’s cut that line and, just for demonstration purposes, we will paste that line back in at the end of the main action.

46. Tap anywhere on the line we want to move or delete. This causes editing options to appear on the screen, surrounding the line to be edited. The line is highlighted with a red bar to its immediate left.
47. Tap the cut button and the line of code is deleted. However, it has not been lost, it has been temporarily remembered by TouchDevelop.
48. If we want to paste the line of code back into the script at the end of the main action, tap the ‘+’ button at the bottom of the screen. This causes the expression editor pad to be displayed.
49. Keep tapping the ‘next suggestions’ button until you see a key named ‘paste’. Tap the paste button.
50. Tap the phone back button to see the new code for the main action.

Your script is now complete. The code after step 50 above should look like that shown in Figure 2.1.

**Figure 2.1: The Final Program**

```
action main(n)
  var sum := 0
  for 0 ≤ i < n + 1 do
    sum := sum + i
  ("The sum of the " || n || " numbers is ")
    || sum) → post to wall
  "TouchDevelop is"... → post to wall
```

PLAY THE SCRIPT

You should test the script to make sure that it runs.

51. Look for the right pointing arrow at the bottom of the screen. That’s the RUN BUTTON. Tap the run button.

If it works, you should be prompted to enter a value for n and then the script should respond by displaying the sum of the integers from 0 to n. A sample run of the program is shown in Figure 2.2.
Congratulations! You've just successfully implemented and run your first program!

### 2.3 Example: Drawing Random Circles

If you completed the exercise of entering the script detailed in Section 2.3 and are ready for another one, here is an example of a script which uses some of the graphics drawing functions. It will display 20 randomly sized circles in randomly chosen colors. This time, the instructions for entering the script will not be given in such detail. (Please refer back to the previous example if you forget how to complete some editing action.)

#### Start a New Script

1. **Begin a new script** and name it “draw random circles”.

#### Add a New Action

2. On the main page for the script, tap the ‘+’ button which appears to the far right of the word ‘actions’. A new action with the default name ‘go’ will be created and displayed.
3. **Tap the line** which reads “action go()” and then select ‘edit’ in the edit menu which appears.
4. Select the word ‘go’ and enter the new name “draw circle” for this action.
5. Click the box labeled ‘private’.

#### Add Parameters for the New Action

6. Tap the word ‘params’ at the top of the screen, and tap the ‘+’ button at the bottom.
7. **Rename** the first parameter to have the name ‘pic’. (Tap in the blank area between the parameter and the keyboard to exit from keyboard entry mode.)
8. Tap the **type** of the first parameter in the text box (i.e. tap the word ‘Number’). This causes the screen to display a list of possible types for the parameter. Scroll down and select ‘Picture’ from this list.
9. Tap the ‘+’ button to add a second parameter. **Rename** the parameter to ‘x’ and leave its type as ‘Number’.

10. Tap the ‘+’ button to add a third parameter. **Rename** the parameter to ‘y’ and leave its type as ‘Number’.

11. Tap the ‘+’ button to add a fourth parameter. **Rename** the parameter to ‘radius’ and leave its type as ‘Number’.

12. Tap the ‘+’ button to add a fifth parameter. **Rename** the parameter to ‘color’ and **change its type** to be ‘Color’. The resulting screen should be as shown in Figure 2.3.

**Figure 2.3: The Five Parameters of draw circle**

13. Tap the phone back button.

We are now ready to enter the code for the body of the draw circle action. When we have finished entering that code, the code should display on the screen as shown in Figure 2.4.

**Important note:** this particular way of drawing a circle as a series of dots is not recommended. There is a much more efficient method ‘draw ellipse’ defined for the Picture type; it draws ellipses, which includes drawing circles, and should be used instead.

**Figure 2.4: Code for the draw circle Action**

```action
draw circle(pic, x, y, radius, color)
for 0 ≤ i < 2 * math→π * radius do
    // the circle centre is at (x,y)
    // each dot is at (x+xofs, yofs)
    var xofs := radius * math→cos(i / radius)
    var yofs := radius * math→sin(i / radius)
    pic→set pixel(x + xofs, y + yofs, color)
do nothing
```

— 20 —
ADD A STATEMENT TO THE BODY OF THE DRAW CIRCLE ACTION

14. **Tap anywhere** in the current body which is simply a comment reading ‘do nothing’. This brings up the edit menu.
15. Tap the **add below** button and then select the ‘for’ button.
16. **Insert the expression** $2 \ast \text{math} \rightarrow \pi$ as the upper bound of the ‘for’ loop. (Do this by entering the number 2, then the operator $\ast$, then the global name \text{math}, then the symbol $\pi$.)

ADD A STATEMENT TO THE BODY OF THE ‘FOR’ LOOP

17. **Tap the blank area** below the code. Now the editor actions are a choice of only two: ‘add expression’ and ‘add flow’. Select **add expression** (indicated by the ‘+’ button).
18. **Construct the assignment statement** for the first statement in the loop body – refer to Figure 2.4. (It can be done by entering the right-hand side of the assignment and then tapping the FIX IT button.)

ADD MORE STATEMENTS TO THE BODY OF THE ‘FOR’ LOOP

19. With the cursor to the right of the statement just entered (use the right-arrow button on the edit pad), **tap the area** below the statement, then **tap the add expression** button.
20. Now **construct the assignment statement** for the second statement in the loop body.
21. Similarly **construct the expression statement** for the third and last statement in the body of the ‘for’ loop.

At this point, the code displayed on the screen should look like that in Figure 2.4 only without the comments. What does this action do? It adds dots to a picture at regularly space intervals around the circumference of a circle whose center is at coordinates $(x, y)$ and whose radius is as specified. Each dot is drawn by changing the color of a pixel to the color specified by the fifth parameter. (Optional: you can delete the ‘do nothing’ comment line.)

INSERT COMMENTS

If we intend to let others see our script, it would be a good idea to add some helpful comments. It is easy to insert comments. Here is how we can add the comments shown in Figure 2.4.

22. **Tap anywhere** on the line which reads “var xofs := …” This brings up the menu for editing a statement.
23. **We want to add the comments above the selected line, so tap the ‘+’ add above** button.
24. Tap the button labeled ‘// insert a comment’
25. **Enter the comment** ‘// insert a comment’
26. **Tap the phone back button** when the comment is complete.
CHANGE THE ‘MAIN’ ACTION

27. Tap the **phone back button** to return to the main screen for our ‘draw circles’ script. Tap the action name **main**

28. Tap the **settings button** (the one which looks like a gear wheel) and then **rename** the action to ‘Random circles’. Tap the **phone back button** to return to the code listing for this action.

29. Now replace the default action body with the code shown in Figure 2.5. At this point, you should not need the step-by-step instructions.

![Figure 2.5: Code for the ‘Random circles’ Action](image)

```plaintext
action random circles()
    var pic := media→create picture(400, 400)
    for 0 ≤ i < 20 do
        ▷ draw circle(pic, math→random(400), math→random(400),
                        5 + math→random(95), colors→random)
        pic→post to wall
```

What does this code for this action do? It first creates a blank picture which measures 400 pixels by 400 pixels. Then the ‘for’ loop iterates 20 times and on each iteration, it invokes the `draw circle` action to draw a new random circle on the picture. The center of the circle is placed at a random position in the picture, the radius of the circle is chosen to be a random number of pixels in the range 5 to 49, and the color of each circle is randomly selected. Finally, the picture which contains the 20 random circles is displayed on the screen.

PLAY THE SCRIPT

30. Finally, play the script by tapping the **RUN button** at the bottom of the editor window.

2.4 MODIFYING A SCRIPT

Let us suppose that after entering the ‘draw random circles’ script and running it, you are somewhat dissatisfied. The circles drawn with single pixels spaced around each circumference are barely visible on your screen. You want to modify the script so that each circle is filled with color, it’s not an empty circle with a thin line around the outside.

If we check the documentation for the TouchDevelop API documentation on the web or if we tap the `pic` variable in our `draw circles` action and see which methods the editor will allow us to invoke for a variable of type `Picture`, we see that there are functions for generating both hollow and solid ellipses. A circle is simply a special case of an ellipse. This will make the code much simpler and faster than the loop which displays individual pixels.
The fill ellipse function is documented as having these parameters:

\[
\text{fill ellipse}(\text{left : Number, top : Number, width : Number, height : Number, angle : Number, color : Color})
\]

The first two parameters (left and top) control where the ellipse is drawn; the next two parameters (width and height) determine the dimensions of the ellipse – but we will have width equal to height for a circle; the fifth parameter (angle) allows us to rotate the ellipse by some angle before it is drawn – but this parameter is irrelevant when we draw circles; the last parameter specifies the color to fill the ellipse shape with. Having understood all that, we should replace the code for the draw circle action in our previous script with the new version shown in Figure 2.6. So let’s begin.

**Figure 2.6: Replacement Code for draw circle**

```
action draw circle(pic, x, y, radius, color)
var left := x−radius
var top := y+radius
var diameter := 2*radius
pic→fill ellipse(left, top, diameter, diameter, 0, color)
```

**OBTAIN THE SCRIPT**

1. In the start screen of TouchDevelop, **swipe right-to-left** to show the list of installed scripts.
2. **Select** the script with the correct name from the list – it should be at the top. Tapping it should cause the two actions of this script to be listed.

**MODIFY THE DRAW CIRCLE ACTION**

3. **Tap the name** draw circle in the list of actions. The code for this action will be displayed.
4. **Tap anywhere in the first line of code** for the for loop. This will cause an edit window to appear, surrounding the whole for loop.
5. Tap the **cut button** and the for loop which is the entire action body is deleted and replaced with a “do nothing” comment.
6. **Tap** the “do nothing” comment and **replace** it with the first statement shown in Figure 2.6.
7. Now add each of the other statements one at a time. When you get to the fill ellipse function, the editor will supply default values for each of the parameters. You should select each parameter, one at a time, delete the parameter and then insert the desired parameter value.

Running the modified script leads to a much more pleasing rendering of random circles. It should like something like that shown inFigure 2.7.
This chapter has not covered all the ways in which the editor can be used to enter statements, insert references to data variables, move statements from one place to another, and so on. It is a powerful editor which you can only become comfortable with using by practice and by experimentation.

Amongst the features which we have not been covered in this chapter, we might mention these:

- Refactoring a group of one or more statements to become a separate action. This is covered in Chapter 7.
- Refactoring an expression or subexpression to become a new local variable. This too is covered in Chapter 7.
- Inserting a use of a global data variable into a statement, or insert a use of a local variable which does not appear as a button in the expression editor. Here, the answer is to find the button labeled EXPLORE ALL, as in the bottom row of this screenshot:
Tapping that button will expose a list of all the services provided in the TouchDevelop API. Also in the list, one can find a code and a data item, amongst others. Tapping on those will yields lists of all the actions and all the global data items the included so far in the script.

Swiping from left to right will also change the pivot and yield other lists, including a list of all the local variables declared so far in the current script.

RECOVERY FROM ERRORS

Oops! I tapped the wrong button and now my code is messed up. How can we go back and try again? Sometimes it’s easy to guess how to recover and sometimes you may have to play around until you have found a sequence of steps which work. As you gain familiarity with the editor, the less hard you will find it to recover. Table 2.1 lists some of the standard ways to undo a mistake.

<table>
<thead>
<tr>
<th><strong>Table 2.1: Recovering from Errors in the Editor</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Button</strong></td>
</tr>
<tr>
<td><strong>Undo button</strong></td>
</tr>
</tbody>
</table>
### Phone Back Button

This is probably for use only as a last resort. Touching the phone back button will take you back up one level in the editor where you should be able to see all the code for the current action or event. You can look at that code and touch any line which needs to be deleted or modified. You can, alternatively, insert missing statements above or below the line you touch.

### Delete Button

If an expression or statement contains an incorrect item, you can touch that item to select it and then touch the delete button in the expression editor menu to delete it, then carry on using the expression editor to insert the item(s) you really wanted to appear in that position. You can also use the cursor left and right buttons to position the cursor to the right of an incorrect item and then touch the delete button.

### Trash Can

If you have added a parameter or a result parameter to an action or event which you do not want, you can select the properties of the action or event, then display the params or result listing, and then touch the parameter which is unwanted. There should be a trash can icon displayed at the bottom of the screen; touching that will remove the parameter.

### Trash Can

Touching any undesired item in a line should cause the expression editor menu to be displayed. You can often just enter a new item with the editor which will replace or modify the item you touched.

If you entered an incorrect string or number value, you can touch the editor buttons for entering a string or number and you will be able to edit the value which had been erroneously entered.

### 2.6 Using the TouchDevelop Bazaar

As discussed in Chapter 1, the Bazaar is a place where you can save and share your scripts so others can see them, use them, and adapt them.

### Publishing a Script in the Bazaar

Here’s how to publish your script to the cloud and let others use and view your brilliant script:

- Touch the **phone back button**, as needed, until you’re on the main page for the script.
- Touch the **publish** button on the bottom (the one with an upward pointing arrow).
- Touch the **publish** box.
• Log in, if necessary, with your Windows Live Id or Facebook credentials and ... you’re published!

• Once your script is published, run your script again and publish a screenshot by clicking on the ‘take screenshot’ menu item in the script execution page.

DOWNLOADING AND MODIFYING AN EXISTING SCRIPT

The bazaar is full of useful scripts. You can access these scripts via TouchDevelop on your phone or with your web browser on a computer.

When you start up TouchDevelop or when you visit the webpage

http://touchdevelop.com/

the welcome screen shows a series of screenshots produced by some sample programs, plus there are lists of scripts provided under the headings ‘new’ ‘top’ and ‘featured’. Touching (or clicking if using a browser) one of these headings will display a list of TouchDevelop scripts in that category, with a ‘show more’ link at the bottom if you wish to keep exploring. Each script has a title which is usually somewhat descriptive of what the script does. Selecting that title causes a lot more information about the script to be displayed.

If you are viewing on a web browser, you can see the code for the script by scrolling down. Reading other people’s code is a probably a good way to learn how to use features of the TouchDevelop API. However, there is no guarantee that it is good code! To get a sense of the code quality and the general usefulness of the script, take a look at the heart symbol and the number alongside. It’s displayed on the webpage like this

bouncy balls
TouchDevelop Samples
3 months ago
17

and almost identically on the phone. What we see is that the script was published 3 months ago and that since it was published, 17 TouchDevelop users have indicated that they ‘like’ the script. This is a reasonably good endorsement. Tapping the name of the script displays some details of the script.

Some scripts in the listings will say that they are based on another script. This is the normal way to create a TouchDevelop script. You search for an existing script which does at least some of the things you want your script to do, and then you edit it. This reduces the amount of work you have to do yourself!
DOWNLOADING A SCRIPT TO YOUR PHONE

If you have identified a script you like amongst those displayed by TouchDevelop on the phone, you can touch the square icon for the script which appears on the left. Here is one such icon, reproduced on the right.

Touching it will cause your phone to connect to the cloud and download it (possibly asking you to sign in first).

Alternatively, if you are using a web browser to look through the scripts or if you want to download the script later, make a note of the 4 or 5 letter codename for the script. For example, you may see the notation “/fpkj” appear alongside the script. You should note that codename and then, later, when you have started TouchDevelop on your phone, you can touch the magnifying glass icon on the TouchDevelop screen (NOT the similar looking one on the phone’s frame) to open a search window. Entering the letters “fpkj” into the search window and touching the even smaller magnifying glass at the right end of the search window will cause the script to be located and downloaded to your phone.

After downloading has finished, the script will run automatically. If you do not want to continue with that execution, tap the phone back button to stop the script (if it has not completed execution already) and then tap the phone back button to return to the main TouchDevelop window.

You can tap (or swipe) to view the list of installed scripts (look for the ‘installed’ tab along the top). The one you just downloaded will be at the top of the list. Tapping the icon or the name of that newly installed script will display a sentence describing the script (if provided by the script’s author) followed by a list of all the components of the list, starting with the code component – which is comprised of all the actions. It will look similar to Figure 2.8.

Some possible actions are:

- If we want to view or edit the code of the action named ‘main’, we simply touch the name ‘main’. Touching anywhere within that code listing will activate the script editor. Then you have the opportunity to make changes to the line which was touched, or to insert new code above or below the line, or to ‘cut’ the line (i.e. delete it with the possibility of pasting it somewhere else).
- If we want to view or edit the code of an event, we can touch its name. (However this script does not have any events.)
- Similarly, for viewing or editing the definitions for any global data variables. (Again, this particular script does not declare any.)
- If we want to add a completely new action to the script, we should touch the plus symbol to the right of the word ‘code’. (Similarly for adding new events or new global variables.)
2.7 CAUTION! PROTECTING YOUR INFORMATION

Before you download a new script to your phone and begin using it or editing it, do take a few moments to check that this script does not transmit personal information about you to someone else. (At least, not unless that is the intended purpose of the script.)

The information displayed about a script may include some special symbols. For example, you may see these two symbols included in the script’s description:

The symbols provide a summary of what kind of information is accessed by the script and what happens to it. If we see the two symbols shown above, we are being told that the script accesses the collection of music on the phone (the first symbol) and that the script makes unchecked accesses to the web (the second symbol). However, it would only be information about your music collection (the names of albums, etc.) that might escape. Most people would not worry about that potential loss of information. An explanation of the symbols and how the information flow analysis is performed can be found on the TouchDevelop website at this URL:

http://touchdevelop.com/help/scriptprivacy
In addition to the analysis performed by TouchDevelop, there is a second level of protection. When you install a script and run it for the first time, you can configure whether TouchDevelop uses your real or anonymized information. For example, your anonymized location is a random place on earth. With the default configuration, TouchDevelop will use your real information only if you will get a chance to review the information before it escapes the phone and potentially be seen by someone else. The privacy settings can also be changed on the script properties page.

### 2.8 Exercises

These exercises are aimed at giving you practice in editing. They are intended to be quite simple and to build only on the examples in this chapter.

1. In the Add n Numbers script in Figure 2.1, change the string to read "The total is “ || sum). 
2. Consider the Add n Numbers script in Figure 2.1. It adds the numbers 1 .. n. Follow the steps in Section 2.3 and change the variable name n to be limit instead. 
3. The Add n Numbers script would be more interesting if we could enter different numbers to add, as in a calculator. We might want to enter prices we paid for gas recently, as in 23.95, 30.54, 42.60, 20.00, 35.75. Change the line sum := sum+ i to be

   `sum := sum + wall→ask number("Enter the next value")`

   Note that with this data, we would still enter the number 5 for n at the start of the script. In Chapter 3, we’ll look at while loops which are more flexible for entering a variable number of numbers.

4. The random circles action in Figure 2.5 is set to draw 20 circles. Add a parameter to the action, similar to that in the Add n Numbers script of Figure 2.1, so that you can say how many circles you want at the beginning.

5. Following on from Exercise 4, there is a statement to clear the screen that we can issue after the number is entered and before the circles start appearing at the start of the action. It is

   `wall→clear`

   Add this as the first line of the draw circles action.

6. Section 2.3 describes how to add comments to a script. Add your name and today’s date as comments to your scripts.