UNIT 3 ACTIVITY

DEBUG IT!

SUGGESTED TIME
15–30 MINUTES

OBJECTIVES
By completing this activity, students will:
+ investigate the problem and find a solution to five debugging challenges
+ explore a range of concepts (including events and parallelism) through the practices of testing and debugging

ACTIVITY DESCRIPTION

- ❑ Optionally, have the Unit 3 Debug It! handout available to guide students during the activity.
- ❑ Help students open the Debug It! programs from the Unit 3 Debug It! studio or by following the project links listed on the Unit 3 Debug It! handout. Encourage students to click on the "Look Inside" button to investigate the buggy program, tinker with problematic code, and test possible solutions.
- ❑ Give students time to test and debug each Debug It! challenge. Optionally, have students use the remix function in Scratch to fix the bugs and save corrected programs.
- ❑ Ask students to reflect back on their testing and debugging experiences by responding to the reflection prompts in their design journals or in a group discussion.
- ❑ Create a class list of debugging strategies by collecting students' problem finding and problem solving approaches.

RESOURCES

- ❑ Unit 3 Debug It! handout
- ❑ Unit 3 Debug It! studio
  http://scratch.mit.edu/studios/475554

REFLECTION PROMPTS

+ What was the problem?
+ How did you identify the problem?
+ How did you fix the problem?
+ Did others have alternative approaches to fixing the problem?

REVIEWING STUDENT WORK

+ Were students able to solve all five bugs? If not, how might you clarify the concepts expressed in the unsolved programs?
+ What different testing and debugging strategies did students employ?

NOTES

+ Being able to read others' code is a valuable skill and is critical for being able to engage in the practices of reusing and remixing.
+ This activity is a great opportunity for pair programming. Divide students into pairs to work on the debugging challenges.
+ Students can explain their code revisions by right-clicking on Scratch blocks to insert code comments.

NOTES TO SELF

- ❑ ___________________________
- ❑ ___________________________
- ❑ ___________________________
- ❑ ___________________________

DEBUG IT!

HELP! CAN YOU DEBUG THESE FIVE SCRATCH PROGRAMS?

In this activity, you will investigate what is going awry and find a solution for each of the five Debug It! challenges.

START HERE

- ❑ Go to the Unit 3 Debug It! Studio: http://scratch.mit.edu/studios/475554
- ❑ Test and debug each of the five debugging challenges in the studio.
- ❑ Write down your solution or remix the buggy program with your solution.

❑ **DEBUG IT! 3.1** http://scratch.mit.edu/projects/24269007

In this project, the Scratch Cat teaches Gobo to meow. But when it's Gobo's turn to try -- Gobo stays silent. How do we fix the program?

❑ **DEBUG IT! 3.2** http://scratch.mit.edu/projects/24269046

In this project, the Scratch Cat is supposed to count from 1 to the number the user provides. But the Scratch Cat always counts to 10. How do we fix the program?

❑ **DEBUG IT! 3.3** http://scratch.mit.edu/projects/24269070

In this project, the Scratch Cat is doing roll call with Gobo's friends: Giga, Nano, Pico, and Tera. But everything is happening all at once! How do we fix the program?

❑ **DEBUG IT! 3.4** http://scratch.mit.edu/projects/24269097

In this project, the Scratch Cat and Gobo are practicing their jumping routine. When Scratch Cat says "Jump!", Gobo should jump up and down. But Gobo isn't jumping. How do we fix the program?

❑ **DEBUG IT! 3.5** http://scratch.mit.edu/projects/24269131

In this project, the scene changes when you press the right arrow key. The star of the project -- a dinosaur -- should be hidden in every scene except when the scene transitions to the auditorium backdrop. In the auditorium, the dinosaur should appear and do a dance. But the dinosaur is always present and is not dancing at the right time. How do we fix the program?

FEELING STUCK?

THAT'S OKAY! TRY THESE THINGS...

- ❑ Make a list of possible bugs in the program.
- ❑ Keep track of your work! This can be a useful reminder of what you have already tried and point you toward what to try next.
- ❑ Share and compare your problem finding and problem solving approaches with a neighbor until you find something that works for you!

FINISHED?

- + Add code commentary by right clicking on blocks in your scripts. This can help others understand different parts of your program!
- + Discuss your testing and debugging practices with a partner, and make note of the similarities and differences in your strategies.
- + Help a neighbor!

DEBUG IT! REFLECTIONS

NAME: ____________________

RESPOND TO THE FOLLOWING REFLECTION PROMPTS USING THE SPACE PROVIDED BELOW OR IN YOUR DESIGN JOURNAL.

+ What was the problem?

+ How did you identify the problem?

+ How did you fix the problem?

+ Did others have alternative approaches to fixing the problem?