Unit 6
Booleans: Teaching Functions to Compare

Unit Overview

Agenda

15min Introduction
5min Booleans
25min onscreen? 1.0
15min Boolean Functions
25min onscreen? 2.0
5min Closing

Students discover Boolean types, and use them to create programs that test values, and then model scenarios using these programs.

Learning Objectives
Students will:

- Understand how to declare Boolean values
- Learn functions to generate and manipulate Booleans and Strings

Product Outcomes

- Students will write functions that use conditionals and Booleans
- Students write functions to perform tests on various domains

State Standards
See Bootstrap Standards Matrix provided as part of the Bootstrap curriculum.

Length: 90min

Materials and Equipment

- ☐ Computers w/DrRacket or WeScheme
- ☐ Student workbooks
- ☐ Class posters (rules, basic skills, calendar, warnings, design recipe, language table)
- ☐ Pens or pencils for students

Preparation

- ☐ Write agenda on board
- ☐ Student games open on their computers
- ☐ "Cage" [DrRacket file + teachpack | WeScheme] preloaded on students' machines, in front
- ☐ Class posters
- ☐ Seating arrangements: ideally clusters of desks/tables
- ☐ Load pre-class assessment on student machines

Pre-Class Assessment: Review of Unit 5

- *This data helps us track student performance. All data collected is anonymous, and participation is optional.*
- If you would like us to see the assessment results for your class, contact us.

Language Table

Types	Functions
Number	+ - * / sq sqrt expt
String	string-append string-length
Image	rectangle circle triangle ellipse radial-star scale rotate put-image

Introduction — Time: 15 minutes

- *Have students begin with the online assessment.*

Booleans — Time: 5 minutes

- We're going to play a quick game, called "True or False."
- When I say a lie, I want you to yell out "False", and when it's not a lie you yell "True."
- *Give them some simple Boolean statements ("I am holding a red pen", "I am a boy", etc.)*
- So far, our language can only add, subtract, multiply and divide numbers. It can't compare them or decide whether something is true or false! We'll be adding the power of comparison to our language today.
- What would DrRacket say is the value of the following pieces of code?

```
(+ 1 4), (/ 4 2), (- 0 9), (< 3 4)
```

- With the exception of the last one, all the above are Numbers. We can solve any of those expressions using our language. Why isn't the last one a number? What is it?
- The expression `(< 3 4)` is a test. It is either true or false that 3 is less than 4. `<` is another function, just like + and string-length. There are several more tests we can do.
- Try using them to compare numbers. What values do you get back? (Use Circle of Evaluation)

```
(> 0 5)
(= 1 9)
(<= 2 2)
(string=? "dog" "cat")
```

- *Have students explain what each of these tests does.*
- As you have found, there are only two values that can be returned from a test: true or false. These values are a very special category, called "Booleans". Let there be Booleans!
- *Model one of the contracts for a Boolean function, then have them guess the rest.*
- Make sure you're writing these contracts down in your notes!
- Booleans are really important in videogames - you can use them to test if someone's health is down to zero, or if they've bumped into a wall. Without booleans, there are no tests, and the game can't ever react to changes.
- Let's try making a function that uses booleans.

onscreen? 1.0 — Time: 25 minutes

- *Draw a rectangle on the board, to represent the screen. Then turn students' attention to the computers, and have them analyze the last function in the code, called* `onscreen?`.
- What are the Domain and Range of `onscreen?` What does this function do? What will happen when it's given a sample input? *It always returns true, no matter what!*
- *When they click Run, a window with a butterfly in it will pop up. They can use the arrow keys to move the butterfly around.*
- This butterfly is Sam - he's a happy middle schooler like you! He likes to fly around outside, enjoy the fresh air, etc.
- Just like our rectangle here, his yard is 640 pixels wide and 480 pixels tall. If the bottom-left corner is (0, 0), what are the coordinates of the other corners?
- It's safe to play there, but it's not as big as he'd like. So sometimes, Sam flies out of the box. He's free! Free as a bird! (well, as a butterfly!) Look at his smile! He's so happy to be free! Why shouldn't he be?
- Well...there's some bad news.
- Sam doesn't realize that there's a monster outside his yard, waiting to eat him! As long as ANY part of Sam is still in the yard, he is safe...but the moment he disappears completely he might get eaten!
- It's up to you to protect him!!
- We want to Sam inside the box. He can fly left or right and there is nothing to stop him from moving off the screen... yet!
- How far can Sam go to the left before NONE of him is onscreen? *-50*. So he is onscreen as long as x > -50!
- *Draw another, outer rectangle on the board to represent the 50 pixel buffer zone.* Raise your hand if you can tell is why we

have to go out to -50 put Sam off the screen, when the screen only goes from 0 to 640.

- *Take some ideas from the class, then explain if necessary*. When the butterfly is drawn at some coordinate, it is centered at that coordinate. So if it's drawn at 0, half of the butterfly is actually off the screen. We add the extra 50 pixels on all sides, because a piece of him is still visible as long as he is within 100 pixels of the screen's edge.
- Turn to **Page 16**. Take thirty seconds to fill out what we've discovered.
- Turn to **Page 17** and read the problem statement for the `protect-left` function.
- Who would like to act out `protect-left`? *Take a volunteer*.
- According to the problem statement, what's your name? *protect-left*. What's your Domain? *Number* Your Range? *Boolean*. So I give you Sam's x-coordinate, and you tell me if any part of him is still on the screen. Let's try it out. "protect-left forty-five!" "protect-left three!" (volunteer should answer "true" for both of these). "protect-left negative one hundred and one!" (volunteer answers false). How did you know Sam wasn't protected at -101? *because the number was less than -100!*.
- Complete the design recipe for `protect-left`. Raise your hand when you've completed each step.
- *When a team has completed the Design Recipe for `protect-left`, have them type in the examples and function body onto the computer*.
- So now we have a function that will protect Sam on the left side of the screen. But at the moment, `onscreen?` doesn't know how to use that function. We need to change the body of `onscreen?` so that it can talk to `protect-left`:

```
(define (onscreen? x)
    (protect-left x))
```

- Click "Run", and try to move Sam off the left edge of the screen. Congrats! You've protected Sam on one side!
- Unfortunately, Sam can still escape on the right hand side. We need a `protect-right`! *(act out with another volunteer, if necessary)*.
- Turn to Page 18, and write another function called `protect-right`. You know when to call me over...
- Good job! Now we can protect Sam on the left side, or we can protect Sam on the right side. However, we don't have a way to protect him from going off from both sides at the same time! We need to learn something new to save him in both directions.

Boolean Functions — Time: 15 minutes

- You've already learned many functions that allow you to create Booleans. < and >, for example, will all return a boolean when applied to the appropriate input.
- There are also many functions that work with Booleans, and let you do really complex tests.
- Pick up two, easily-distinguishable objects (a pen and an eraser, for example). Tell me if the following statements are true:
  - I am holding a Pen
  - I am holding an Eraser
  - I am holding a Pen AND I am holding an Eraser. *(drop the pen)*
  - I am holding a Pen AND I am holding an Eraser.
  - I am holding a Pen OR I am holding an Eraser. *(drop the eraser)*
  - I am holding a Pen OR I am holding an Eraser.
- Did you notice how we joined the Boolean statements? What words did we use? (AND, OR).
- *Try to get students to write the contracts for these functions, after giving them AND*.

```
; and: Boolean Boolean -> Boolean
; Returns true if BOTH inputs are true.

; or: Boolean Boolean -> Boolean
; Returns true if EITHER of the inputs are true.
```

- Turn to **Page 19**. *Have a student read the directions*.
- Take five minutes with your team, to draw the circle for each of the statements. Think about what Boolean functions you'll need for each one! When you're done, convert those circles into Racket code.

onscreen? 2.0 — Time: 25 minutes

- Now it's time to put it all together. Right now, you know how to protect Sam on the left side of the screen, and how to protect

him on the right side. But how do we protect him on BOTH sides at the same time?

- Turn to **Page 20** and read the problem statement.
- I need a volunteer to be the `onscreen?` function. *Take a volunteer from the class*
- What is your name? *onscreen?* What is your domain? *Number* What is your Range? *Boolean*.
- Sam is onscreen only when he is protected on the left AND the right. How can we use one of those functions we just learned to do this? I'd like my volunteers `protect-left` and `protect-right` to stand up.
- When I yell out "onscreen? fifty", `onscreen?` is going to call `protect-left` and `protect-right` with that number. Depending on what they give back, `onscreen?` will tell me true or false.
- *Try this out, making sure that students are actually talking to one another, calling the functions properly by name. When you call out "onscreen eighty!", the `onscreen?` volunteer should call out "protect-left eighty" and "protect-right eighty", and listen their answers before giving back an answer of their own. Run through a bunch of examples here, to make sure it's clear.*
- Great job, everyone. Back to your seats. It's time to write the contract and examples! Don't forget to use what we've just acted out! Raise your hand so I can check your work at each step!
- *Guide students as necessary, and eventually have them write the code on the computer.*
- Guess what? This function can be dropped into your game file, and used to have your characters come back once they've gone off the screen!
- *Have students copy and paste the three functions into their game files.*

Closing — Time: 5 minutes

- Who can tell us one thing we learned today?
- Who saw someone else in the class do something great?
- Cleanup, dismissal.