Unit 3
The Definitions Window

Unit Overview

Students are introduced to the Definitions window, and learn the syntax for defining values of various types. They are also introduced to the syntax of defining functions and creating examples.

Learning Objectives:

- Learn about examples, variables and functions
- Practice Racket syntax and the Circle of Evaluation

Agenda

15 min	Introduction
10 min	Defining Variables
30 min	Game Screenshots
10 min	Fast Functions
10 min	Blue Circle
10 min	Double
5 min	Closing

Product Outcomes:

- Students will write functions to solve simple problems in which a number is used to create an image, when given a word problem and a worked example.
- Students will write examples (unit tests) to check those functions

State Standards See our Standards Document provided as part of the Bootstrap curriculum.
Length: 90 minutes

Materials and Equipment:

- *Student workbook folders with names on covers.*
- *Pens/pencils for the students, fresh whiteboard markers for teachers*
- *Class poster (List of rules, language table, course calendar)*
- *Language Table (see below)*

Preparation:

- *Create student game files. [See the (teachers-only) Teachers Guide]*
- *On student machines: Student Game Files (generated from "Game" template [Game.rkt from source-files.zip | WeScheme])*
- *Write agenda on board, and post along with class posters and the Language Table*
- *Seating arrangements: ideally clusters of desks/tables*
- *Optional: demo machine with projector to show the interations and definitions windows*

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

Introduction (Time 15 minutes)

- You've have done a fantastic job in the last two classes!
- You've learned how to convert expressions into Circles of Evaluation, and how to convert those circles into Racket code.
- You've learned how to think about functions in terms of nested circles, and how to think of them as a relation between the Domain and Range.
- You've extended that knowledge into three data types: Numbers, Strings, and Images.
- *Let's see how much you remember!*
- *Review material from the previous class.*
- You have learned a LOT, and before today's class is over... you will have written your first line of code for YOUR videogame.

Defining Variables (Time 10 minutes)

- *Demo: Have students open their game files, and click Run. They should see a frozen screenshot of their game, using the images they requested. (By now, you should have students' graphics already created, and added to the file)*
- So far, everything that you've been doing has been down in the Interactions window. What happens when you click Run at the top? All the work you did disappears!
- That's because the Interactions window is meant just for trying things out. If you want to define something permanently, you need to use the Definitions window.
- The game in front of you is a bare-bones, totally broken game. It doesn't DO anything...YET!
- Look below Step 0, near the top of the screen.
  Raise your hand if you can read the line of code just below that (Have a volunteer read it aloud).
  What will happen if I type `TITLE` into the Interactions window down at the bottom? (Hint: Try it out!)
- What will happen if you type `TITLE` into the Interactions window down at the bottom? (Answer: This code tells the computer that the name `TITLE` is a shortcut for the string `"My Game"`. When you click Run, the computer learns that name and that shortcut, along with any other definitions.)
- When you click Run, you'll see the title `"My Game"` at the top left hand corner of your new game.
- This kind of name, which is just a shortcut for some value like a Number, String, or Image, is also called a variable. You've seen other names too, like `+` and `string-length` that are the names of functions. You'll name your own functions soon.
- Change the value of this variable from `"My Game"` to YOUR title. Then click Run, and see if the game is updated to reflect the change.
- What is the name of the NEXT variable defined in this file? What is its value? (Answer: `TITLE-COLOR`) Try changing this value and clicking Run, until your title looks the way you want it to.
- For practice, try defining a new variable called `author`, and set its value to be the string containing your names. Don't forget - all strings are in quotes! (This won't do anything in the game, but when you close the game window, you can type `author` and see its value.) Then you can ask `(string-length author)`, etc.
- What other variables do you see defined in this file? What is its name? What is its value?
  Take a volunteer.
- Variables can be more than just strings. They can be numbers, or images! These definitions are where we define the images for your background, player, target, and danger.
- As you can see, there are variables that define the `BACKGROUND`, `PLAYER`, `TARGET` and `DANGER` images.
- What will happen if you type `DANGER` into the Interactions window down at the bottom? (Answer: This code tells the computer that the name `DANGER` is a shortcut for a solid, red triangle of size 30. When you click Run, the computer learns that name and that shortcut, along with any other definitions.)
- You can even define long expressions to be a single value. Look closely at the definition of `SCREENSHOT`. This definition is a complication expression, which puts all of the game images on top of one another ar various coordinates. The `PLAYER` image, for example, is being displayed on top of the `BACKGROUND`, at the position `(320, 240)`.
- According to the definition of `SCREENSHOT`, where is the `DANGER` located? (Answer: `(150, 200)`)
- Try changing the various coordinates and clicking **Run**, then evaluating `SCREENSHOT` in the Interactions window. Can you move the `TARGET` to the top-left corner of the screen by changing it's coordinates?

Game Screenshots (Time 30 minutes)

-  Video: A screencast of this section - follow along to see how to change images in your game.

- Let's start changing these image definitions so that they use the images YOU want.
- You already know how to draw various shapes, but suppose you wanted to use an image that you found? (For example, this circular red-and-blue icon on the right?)
- Take a look at the contracts for these two functions:

```
; bitmap : String -> Image
; bitmap/url : String -> Image
```

- Both of these functions take a `String` as their domain, which tells them where the image file is located. If you're using WeScheme, that can be the address of any image file you find on the internet. In DrRacket, the String represents the path to the file.
- If you're using **WeScheme**, you'll want to use `bitmap/url`. If you're using **DrRacket**, you can use `bitmap/url` or `bitmap` to include images from your computer.
- Try replacing the definition of the `BACKGROUND` with an image file, and **click Run.**)
- Now try to find an image for your `PLAYER`. When you click Run, you should see your player appear on the game screen. Typing `PLAYER` into the Interactions window will show you just the player image, by itself
- Suppose you wanted your player to be larger or smaller? There's a function called `scale`:

```
; scale : Number Image -> Image
```

- `scale` resizes the `Image` based on the `Number`. For example, `(scale 3 PLAYER)` will make the PLAYER image three times as large, while `(scale 0.5 PLAYER)` will make it half the size. On the right, you can see what our icon looks like after being `scale`ed.
- Once you've written the contract for `scale`, use it as part of the definition of `PLAYER` to make sure your player looks just right!
- Suppose you want your player to be flipped vertically or horizontally? Copy down the contracts for these two functions, then try using them to flip your `PLAYER`.

```
; flip-vertical : Image -> Image
; flip-horizontal : Image -> Image
```

- You can also rotate any image, so that your player is facing any point on the screen:

```
;rotate : Number Image -> Number
```

- Try rotating your player 45 degrees.
- You can combine these functions together, just like you can use `+, -, /, *, sqrt`, and `sqr` together when working with Numbers. Can you make your player twice as big, *and* rotated 45 degrees, like our example on the right??
- Now try using all of these functions - however you want - to make the definitions of `DANGER, TARGET` and `PLAYER` look exactly the way you want. You can even change their locations inside `SCREENSHOT`, so that you get a real-life screenshot of your game!
- Click "Run", and evaluate `SCREENSHOT`. Does it look the way you expected? On your own, mess with the coordinates until the `TARGET` is placed where you want it to be.

Fast Functions (Time 10 minutes)

- You've learned how to write complex expressions, and define shortcuts so that you can use them later. That's terrific...but we need more!
- The problem is that all of these expressions always return the *same thing* - your screenshot, for example, will always look the same, every single time you evaluate it. What you want is a shortcut to a *pattern*. That way the computer can just fill in the blanks for the stuff that's changed, and get the whole expression back. Up to now, you've been defining values. Now you're going to learn how to define **functions**.
- My favorite shape in the whole world is a triangle, and my favorite color is green. I LOVE making solid green triangles! But right now, I have to type out so much code to do that! Suppose I wanted to make a solid, green triangle of size five. What code would I write? What if I wanted it to be of size 100? Size 24?

Ask students to tell you each of these

```
(triangle 100 "solid" "green")
(triangle 24 "solid" "green")
```

- If only there was a function called `gt`, that would just take in the size and draw a solid green triangle of whatever size I wanted.
- *Skit:*

Who can help me, by acting out `gt`? Take a volunteer. Okay, your name is now "gt." All I need to do is call out your name, give you a number, and you will tell me the code to draw that beautiful triangle. Let's do a test: "gt fifty!". "gt one hundred!" The student should tell you the code to draw the appropriate triangle. Try having other students call out examples, making sure they call out both the name of the function and the number.

- Open your workbooks to Page 8, where it says "fast functions."
- On this page, there is space to write four simple functions.
  We're going to do the first one together, and then we'll have a competition for the rest.
- Let's start with the contract. What are the three parts of a contract?
- *Hey volunteer, what did I say your name was? "gt!" And what information did you need from me to do your job? just a number - the size!. And what did you produce, once I'd given you that number? An Image.*
- Fill in the first contract on the page – it's the one with the shaded, gray bar.
- Now we have some space to write examples.
  Let's think about the examples we saw our volunteer act out...
- When I wanted him to make a solid green triangle of size fifty, what did I tell him? "gt fifty!". So in the first part of the EXAMPLE, we can write `(gt 50)`. So my example so far is

```
; gt : Number -> Image
(EXAMPLE (gt 50) _____________________________)
```

- Then what did he draw for me? A solid green triangle of size fifty! How would we write the code to draw that same shape?

```
; gt : Number -> Image
(EXAMPLE (gt 50) (triangle 50 "solid" "green"))
```

- *Can someone write another example for me, this time using `17` as the size?*

```
; gt : Number -> Image
(EXAMPLE (gt 50) (triangle 50 "solid" "green"))
(EXAMPLE (gt 17) (triangle 17 "solid" "green"))
```

- Now, on your own, fill out two examples for `gt` on your Fast Functions worksheet.
- If only we had a function like `gt`! Well, let's build one!
- Right now, I'm telling the computer how to deal with a shortcut for `(gt 17)` - but what if I wanted the shortcut to work for ALL sizes, not just 50 and 17?
- That's the final step: replace the stuff that changes between examples with a variable. So let's look at these two lines, and circle everything that changes. What did we circle? Just the numbers 10 and 17! What do those numbers mean? Is it the number of circles we're drawing? No! It's the SIZE. So let's make a little note to ourselves, to remind us that those numbers mean the size of the circle.
- Now we can write the code – instead of an EXAMPLE we'll use `define`. After that, we're just going to copy everything from our examples except the stuff that we circled. What do you think we'll write instead? We'll use the name we wrote down: size.
  Go character-by-character with the students, asking them if both examples have an open paren, the name "gt", etc...

```
; gt : Number -> Image
(EXAMPLE (gt 50) (triangle 50 "solid" "green"))
(EXAMPLE (gt 17) (triangle 17 "solid" "green"))
(define (gt size) (triangle size "solid" "green"))
```

- What we've learned here is a *recipe* for solving programming problems. By starting with a **word problem**, we can act out

the function, then write it's contract, imagine a few examples, and then use those examples to write the code! Let's do another for practice - you'll need to get really good at these to build the functions you'll need for your videogame!

Blue Circle (Time 10 minutes)

- Now it's your turn!
- *Raise your hand if you want to help me act out this next function. We'll come up with some examples together, and your group will have to write two more on paper!*
- *Hand the student the sign that says "bc" and ask them to come to the whiteboard.*
- *Skit:*

> *When I say "bc 50", you'll draw a solid blue circle of size 50. Let's try it out. "bc fifty!". Wait for student to draw a circle. Then have several other students give examples to your function, by calling out "bc" and a number. Make sure that the student answering gives an appropriately sized circle return every time.*

- I want to write a function called `bc`, which takes in a number and draws me a solid, blue circle that is whatever size the number was.
  Just like our volunteer here.
- First, you need to write down the CONTRACT for this function.
  Once again, everyone in your group needs to have the correct answer! You'll have 2 minutes. GO!
- Now it's time to write some examples. Let's look at the first example "bc" drew on the board, for `(bc 50)`. What shape did they draw? What color? What size? How would you write the code to draw that shape? (Answer:
  (write on the board):
  ```
  (EXAMPLE (bc 50) (circle 50 "solid" "blue" ))
  ```
- You have 2 minutes for EVERYONE in your group to write out 2 examples of your own. ALL OF THEM have to be correct for your team to get this point. GO!
- *Countdown: 30... 10... 5... 4... 3... 2... 1... PENCILS DOWN, EYES ON ME. (Don't forget to wait for total silence, attention.)*
- *Give points, praise kids for neat handwriting and good teamwork.*
- *Give the countdown, then review answers with the class and assign points.*
- Time for the last part: writing the function header and body. Your team will have 2 minutes to complete this. GO!
- *Give the countdown, then review answers with the class and assign points.*

Double (Time 10 minutes)

- *I want a volunteer to be a function called "double", which takes in a number and multiplies it by two.*
  Hand the sign to the student.
  So if I say "double 3", what will I get back?"
- *Have a couple of students try out the function by giving examples*
- *You will have TWO minutes to write down that contract and two examples. Once you've got your examples, RAISE YOUR HAND and call me over, so I can check them. Two minutes, ready - go!*
- *Give the countdown, then review answers with the class and assign points.*
- *Raise your hand if you think you know how you could write an example for "double". (If you get blank stares, give them ONE example on the board. Otherwise, smile and move on.)*
- Your groups will now have FIVE minutes to write two examples, and then circle and label what has changed. Then you can fill out the function header and body. Once you've got your examples, RAISE YOUR HAND and call me over, so I can check them. Do NOT go on to the function header and body until I have checked your examples! Any questions? GO!
- *Give the countdown, then review answers with the class and assign points. If time allows, do another example, preferably one where the domain is something besides numbers.*

Closing (Time 5 minutes)

- *Who can tell us one thing we learned today?*
- We learned about how to define variables and functions!
- Say that we wanted to define a shortcut of a solid, purple star of size 10. Would we define a value or make a function? (Answer: It would be an expression, since it's always the same image, of the same size.)
- Say that we wanted to make a shortcut that would take in a size, and draw a purple stars of that size. Would we define a value or make a function? (Answer: It would be a function, since it takes in a size, and returns a purple star based on the size that you gave it.)

- *Who saw someone else in the class do something great?*
- Well done! You have officially started your games! The next step is to make your characters animate, which we'll be doing in our next class. See you then!
- *Cleanup, dismissal.*