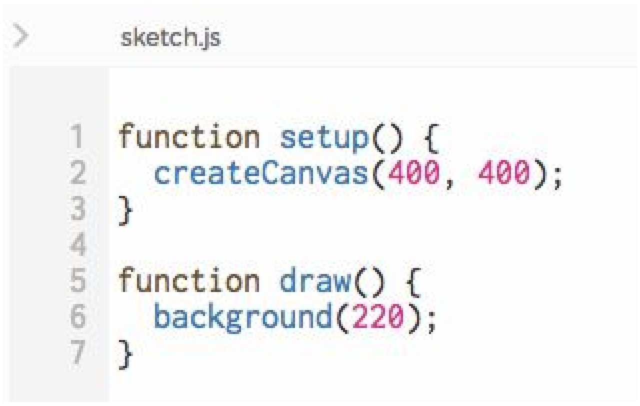
MODULE 7: VARIABLES AND MOVEMENT

Agenda:

1. Setup and Draw (10 min)
2. What is a variable? (10 min)
3. Using variables (mouseX and mouseY) (20 min)
4. Creating your own variables (20 min)

Setup and Draw

When you first open the p5.js web editor (http://editor.p5js.org) you'll notice that several lines of code have already been written for you. So far we've ignored these lines and added our own code to draw things. Now we'll discuss what these lines mean and how you can use them yourself to create dynamic, interactive animations.



```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
}
```

The default code in the p5.js editor

Notice that there are two "functions" in these lines of code. Each "function" serves a separate purpose. So far, the drawings you've made with code have been static -- they don't move, or change color, or do anything interesting. Today we'll use setup() and draw() to start making interactive drawings.

setup()

As you might expect, the “function setup() { }” block of code is used to create initial conditions for your drawing, to set up the parts of your drawing that will always be the same. Everything inside the { } curly brackets of setup() will only be run one time. This code will not change once your program is running.

An example of something to put inside the setup() function is how big your want your drawing canvas to be. By default, there is a line of code inside setup() that says “createCanvas(400,400);”. This creates a canvas (a place where you can draw things) that is 400 pixels wide and 400 pixels high. If you change the numbers inside the createCanvas() function, the size of your canvas will change.

draw()

All of the code inside the “function draw() { }” block of code will loop over and over again. This code will run continuously, from top to bottom, until you tell the program to stop. Inside this block of code is where you draw things that will change over time, or animate. However, in order to make your drawing change over time, we have to introduce the concept of a variable.

What is a variable?

A variable is something that stores information. That information can be a number (like the width of a rectangle), or it can be a color (“this circle is red”), or it can be any other kind of information you want to remember.

However, variables are so important because they allow us to do more than remember information. A variable, as its name suggests, is also something that can vary or change over time. We can change the information stored inside the variable using code.

For example, let’s say we create a variable called “circle_color” which stores information about the color of a circle we are drawing with code. In the setup() block of code, we may tell the program that circle_color is red (255,0,0). When the circle first appears on the screen, it will be red. Then later, when the program is running, we can change circle_color to blue whenever we click the mouse. When the mouse is clicked we will see the color of the circle become blue.

This is probably still a little confusing. So let’s look at a few examples.

Using variables (mouseX and mouseY)

So far, every drawing you've made in code hasn't not moved or changed. Let's add one line of code to your sketch which will make your drawings much more exciting.

```
function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
  ellipse(mouseX,mouseY,50,50);            ← add this line of code
}
```

Now when you press "Run", move your mouse around the gray canvas. The circle is following your mouse wherever you move it! This is because we have made the X and Y position of the circle be represented by two variables (a value that changes over time).

This is an example of a variable in action. p5.js has several variables included in its language that you can use in your code. The variables mouseX and mouseY store information (the x-location and the y-location of the mouse). The information it stores changes over time (when you move the mouse, the XY-location of the mouse changes), so the location of the circle changes as well. This is the power of variables.

So how does this work? In the ellipse function, instead of putting a permanent X position and Y position, like

```
ellipse(200,150,50,50);
```

You told the ellipse to use a variable for the X location and a variable for the Y location:

```
ellipse(mouseX,mouseY,50,50);
```

So whenever you change those variables (by changing the location of the mouse), the code changes the location of the ellipse.

Instead of the ellipse() function, try entering this line of code:

```
line(200,200,mouseX,mouseY);
```

See how this works?

Let’s try one more thing with our program so you can understand more how setup() and draw() work. Move the code for the background() inside the setup() function, like this:

```
function setup() {
  createCanvas(400, 400);
  background(220);
}

function draw() {
  ellipse(mouseX,mouseY,50,50);
}
```

What happened? Why do you see more than one circle now? Why don’t the other circles disappear like they did before?

When you move the background(220); line of code into the setup() function, it only draws the background one time at the very beginning of the program. Remember that everything you put inside the setup() function only happens once. But everything you put in the draw() function happens over and over until you stop the program.

As the program is running, your code tells the computer to draw a circle over and over again wherever you move the mouse. Because the background is only drawn once, and not over and over again just before you draw a circle, all of the circles you’ve drawn remain on the screen.

Creating your own variables

Using the mouseX and mouseY variables that p5.js provides for you can be fun, but eventually you’ll want to use your own variables to make your drawings more interesting.

First, let’s draw a circle on the screen using variables.

Creating a variable in p5.js is as easy as typing

```
var x = 100;
```

This simple statement has three parts. The **first**, “var”, tells the program you want to create a variable. The **second**, “x”, says you want to call that variable x. You don’t have to call the variable “x”. You could give this variable almost any name you want. The **third** part of this line of code, “ = 100;” tells the program you want to set the initial value of the variable to 100. This number could be anything you want.

In sum, this statement says “Create a variable called ‘x’ and set x’s value to 100.”

Now, how do we put this into our code to make a circle appear on screen? Try this:

```
var x = 100;

function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
  ellipse(x,200,50,50);
}
```

What happens when you press “Play”? A circle appears on the screen at location (100,200). The program knows that when you put the variable “x” for the ellipse’s location, that you want to use whatever number you set the variable x to earlier. In this case you set the variable’s value to be 100, so the program displays the the circle 100 pixels from the edge of the canvas.

Notice that the “var x = 100;” line was put at the very beginning, even before the setup() function. For now, this is the best place to put the variables you create.

Now, let’s image you want to create a circle that moves from left to right across the screen. To do this, you would use a variable that controls the circle’s X location. We’ll start the same, except this time the initial value of x will be zero (because you want the circle to start on the left side, where the x location equals 0).

```
var x = 0;

function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
  ellipse(x,200,50,50);
  x = x + 1;
}
```

The only other change is adding the line “x = x +1;” inside the draw() function. This line of code does all the magic of animating the circle. Remember that all the code inside the draw() function is repeating over and over until you stop the program. So, every time the program runs the code inside the draw() function, the value of ‘x’ is increased by 1. The variable ‘x’ starts at 0 (the left side of the screen), and slowly increases as the program runs. As ‘x’ increases, its x-location on the screen also increases, so it appears to move across the canvas.

Let’s see one more example of how to use a variable to change your drawing over time. This time we’ll use a variable to control the size of a circle. Start by creating a variable and assigning it a value, in this example, the variable is called ‘r’ and its initial value is zero. Then put that variable in the part of the ellipse() function that controls its width and height:

```
var r = 0;

function setup() {
  createCanvas(400, 400);
}

function draw() {
  background(220);
  ellipse(200,200,r,r);
  r = r + 1;
}
```

Don’t forget to add the line “r = r + 1;” at the end or else the variable will never change its value and you won’t see anything appear on the screen. Press “Run” and watch the circle grow!

Variables don’t just control size or location. They can also control color.

Try adding this line of code your program right before the ellipse() function.

```
  fill(mouseX,mouseY,100);
```

What does this do?

Basically, any time you see a number in your code, you can replace that number with a variable and change that variable’s value in your program. You can use a built-in variable like mouseY or your own variable you created.

Exercise: Bring your character to life

In this exercise you'll add some animation and interaction to the character you created in the last session. Think about the things you learned during this module and figure out what you want modify on your character using variables. You can add several variables to your sketch to control different aspects of your character. Just add all the variables you want to use at the very beginning of your code, like this:

```
var x = 100;
var y = 60;
var headSize = 10;
var b = 255;
```

Then, within your draw sketch, add lines of code to change them as the program runs. You can make values increase or decrease as time passes:

```
x = x + 2;
y = y - 10;
headSize = headSize + 3;
```