Variables are small words or phrases that store all sorts of data in your computer’s memory for you to use later. They can store things like numbers, text and more.

**Variable Declaration**

Before you can use a variable, you must declare it first. The declaration starts with the word var, followed by the name of the variable, then an equal sign, and then the data you want to store in the variable. It looks like this:

**var myVariable** = 10;

Variables can be redefined to store new values. Simply restate the name of the variable with an equal sign and the new value afterwards. You can also add to the value of a variable without restating the value. For example, the variable above has a value of 10. To make it 20, instead of redefining it as equaling 20, you simply say:

**myVariable +=** 10;

This will add 10 to the value of the variable and make it 20. This method also works with subtraction, multiplication and division (using -, \*, or / respectively).

You can also do ++ or -- instead of += or -= to add or subtract 1.

You can use a variable wherever the value of the variable can be used, simply by calling the name of the variable. For example, the variable above holds a number, so it can be used in math statements to add, subtract, multiply or divide. It can also be used as a parameter for drawing functions, such as a coordinate to draw a rectangle.

rect(100, 50, **myVariable**, **myVariable**);

Remember, variables need to be declared first before they can be used anywhere. Variables declared this way are called **global variables**, which means they can be used anywhere you want in your code.

**Functions**

Functions are essentially storing a set of code within a variable name, then running that code whenever the function is called. Functions are declared similarly to variables, like so:

var myFunction = function() { };

Inside of the curly brackets, you would put whatever code you want to run when the function is called (usually on multiple lines). You can draw multiple shapes at once, for example, or draw a certain combination of shapes you commonly draw.

Functions can also ask for parameters, like the drawing functions we’ve been using before (i.e. rect, ellipse, etc.). This is done in the function declaration like so:

var myFunction = function(xPos, yPos) { };

Within the function, you would use the variables xPos and yPos as variables to change the position of the function. When you call the function, you would put values for those variables in the parentheses, instead of leaving them empty.

myFunction(**150**, **100**);

Note: The parameters you create for a function are **local variables**. That means they can be used only in the function itself and nowhere else in your code.

Functions are often used to make your code more reusable. If you use the same code repeatedly, it’s a good idea to put the code in a function and call the function when you need to.

**Returning Functions**

Functions can also be used to calculate values, and then return them to you when they are called. This is done by using the return command. The easiest way to use the return command is to put the value you calculate in a variable, and then return the variable. An example would look like this:

var add20 = function(**num**) {

**num** += 20;

**return num**;

};

Similar to variables, functions that return a value can be called wherever that value can be used. For example, since the function above returns a number, it can be used in math statements to add, subtract, multiply or divide, or as a parameter for a function. For example, if we use myVariable from above again:

rect(150, **add20(myVariable)**, 50, 50);

Keep in mind that if your function will return a value, the return command has to be the very last line of code in your function.

**Special Functions**

Luckily for us, ProcessingJS (the library we use) gives us some useful functions. To define what happens when these functions are called, you do the same thing as declaring your own function, but without the var at the front. For example:

mouseClicked = function() { };

These special functions don’t need to be called within your code; they are automatically called when a certain action occurs (i.e. like above, the function is called whenever the mouse is clicked).

Check the [documentation](https://drive.google.com/open?id=1RwqRWnslltDAfBN2_wFo0WhjpldRVEgDtZinNtRNfI8) for a list of more special functions you can use in your programs.

**Practice Exercises**

1. Define a function that draws a set of shapes to create a picture with multiple colours (i.e. a simple face, house, etc.). Use global variables to define the coordinates of at least 2 shapes and the length/width/radius of 2 other shapes
2. Modify the function above to take in an X and Y coordinate as 2 parameters and draw the picture at the X and Y coordinates provided.
3. Create a mouseClicked function that will draw your picture at the mouse’s coordinates.
4. Modify your function to accept a third parameter sizeFactor that will change the size of your picture by a factor of the value of this parameter (i.e. if a sizeFactor of 2 is used, the picture will be twice as big. If it is 0.5, the picture will be half as big)