Next topic is to generate on Function App Lab

**FUNCTION:**

They will be introduced to functions as a form of abstraction that enables them to write code in larger, more logical chunks and focus on what something does, rather than how it does it. As they explore the use of functions through a sequence of activities in App Lab, they will be asked to think about where they see the need for functions and how these functions can make their code clearer or more concise.

**Introduction of Function:**

Programming languages must necessarily define the meaning of a set of commands which are generally useful and applicable. In order to extend their functionality, nearly all programming languages provide a means for defining and calling new commands which programmers design to fit the needs of the problems they are trying to solve.

Calls a user defined function that takes no parameters, and optionally generates a return value.

When you define a function you give a name to a set of actions you want the computer to perform. When you call a function you are telling the computer to run (or execute) that set of actions.

Defining functions is an example of how computer scientists use abstraction to solve problems. A programmer will design a solution to a small, well-defined portion of the task and then give a name to the associated code. Whenever that problem arises again, the programmer can invoke the new function by name, without having to solve the problem again. This ability to refer to complex functionality by simple, meaningful names allows for programs to be written in more intuitive ways that reflect the relationships between different blocks of code.

**Declaring and Calling Function:**

A function definition can be provided anywhere in your code - in some ways the function definition lives independently of the code around it. It actually doesn't matter where you put a function definition. And you can call it from anywhere, either before or after the function definition. We will follow the convention of always putting function definitions at the bottom of our program, and the code for calling functions at the top of our program.

On code for declare and calling function:

// Call functions to draw a figure eight using two squares.

square();

turnLeft();

turnLeft();

square();

// Define a function to draw a square using left turns.

function square(){

moveForward();

turnLeft();

moveForward();

turnLeft();

moveForward();

turnLeft();

moveForward();

turnLeft();

}

**Sample Output:**

It can be designed on simple work in shaping using above given code.

The purpose of a function is to help you organize your code and to avoid writing the same code twice. You can you define a function once, and then call the function a number of times.

A common error is defining a function but forgetting to call the function. A function does not automatically get executed.

A function that does not explicitly return a value returns the JavaScript value undefined.

**Rearrange on Function Call:**

arrange() orders the rows of a data frame by the values of selected columns. Unlike other dplyr verbs, arrange() largely ignores grouping; you need to explicitly mention grouping variables.

// Grow an image on mouseover,

// back to normal size on mouseout.

image("logo", "https://code.org/images/logo.png");

setPosition("logo", 160, 240, 32, 32);

onEvent("logo", "mouseover", function(event){

setSize("logo", 48, 48);

});

onEvent("logo", "mouseout", function(event){

setSize("logo", 32, 32);

});

**Sample Output:**

N

O

W

C

This above program followed by output resolve that the arrangement on function in app lab can be moveable frame work. So that user can play on puzzle game.

The UI element must be defined in your code before the matching [onEvent()](https://studio.code.org/docs/applab/onEvent/) event handler.

The callback function receives an event object as its parameter, which can be used to gain more information about the event. You can ignore the App Lab warning event is defined but not called in your program.

The preferred placement in your code is screen elements at the top, event handlers in middle, other code/functions at end.

**Function to make the Face:**

**Smiley Face** Draw a smiley face. The second canvas is smaller and placed where the mouth should go. Since drawing only occurs within the bounds of a canvas, the circle drawn on the second canvas is cut off, or clipped, so that only the portion inside the canvas is visible.

// Draw a smiley face. The second canvas is smaller and placed where the mouth should go. Since drawing only occurs within the bounds of a canvas, the circle drawn on the second canvas is cut off, or clipped, so that only the portion inside the canvas is visible.

createCanvas("face");

setFillColor("yellow");

circle(160, 240, 100);

setFillColor("black");

circle(125, 215, 20);

circle(195, 215, 20);

createCanvas("mouth", 120, 50);

setActiveCanvas("mouth");

setPosition("mouth", 100, 260);

setStrokeWidth(15);

circle(60, -15, 50);

**Sample Output:**

Only one canvas can be active at a time. A canvas must be first created with an id using [createCanvas](https://studio.code.org/docs/applab/createCanvas).

Change the position (or display size) of the canvas using [setPosition](https://studio.code.org/docs/applab/setPosition). Changing the display size of the canvas does not change the bounds of the canvas (the range of x and y values that are valid for drawing). Instead, the drawing within the canvas will be stretched (or squished) to fit the size specified by [setPosition](https://studio.code.org/docs/applab/setPosition).

Drawing outside of the dimensions of the canvas will not be visible. The dimensions span from 0 to the width of the canvas horizontally (x), and from 0 to the height of the canvas vertically (y).

When creating more than one canvas, the canvas elements are layered one on top of the other. The second canvas created will be "in front" or "on top" of the first canvas. This means that any drawing on the second canvas will cover the first canvas when the two canvas elements overlap.