Abraham Gebrekidan
EIP 01

Reading #12
August 08, 2024

Section Reading #1—Simple Javascript

1. Bug Squash

In this problem, you will be presented with code containing common bugs that we will all probably write at some point (maybe you won't after taking EIP 01 ☺).

The expected output is provided, but for one reason or another the code is not producing the expected output.

Your job is to determine what the code actually outputs. Then, identify and fix the bugs in the code.


```
function isAbraham(name) {
	if (name = "Abraham"){
		return true;
	}
	return false;
}
isAbraham("Endalk"); //should return false


function climbAxumTower() {
	var currentFloor = 1;
	while (currentFloor <= 12) {
		console.log("On floor " + currentFloor + ". ");
		currentFloor + 1;
	}
}
climbAxumTower(); //should output "On floor 1. / ... / On floor 12."
// Note: the / represents a line break, used here just to save space.

function countToTen() {
	var counts = "";
	for (var i = 1; i < 10; i++) {
		counts += i + " ";
	}
	console.log(counts);
}
countToTen(); //should output "1 2 3 4 5 6 7 8 9 10 "
```

2. Squares

ASCII Square

```
xxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx
xxxxxxxxxxxxxxx
```

```
const SQUARE_LENGTH = 15;
console.log(asciiSquare());
```

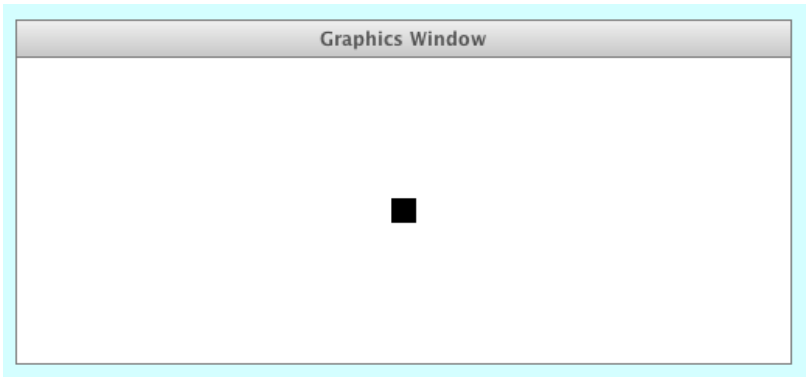
We take amazingly detailed graphics for granted today, but in the past merging computer science and art took the form of ASCII art. Since we are only budding ASCII artists, let's start by trying to draw a square.

Using just the character `'x'`, your job is to write a function that returns a string that has `SQUARE_LENGTH` rows with `SQUARE_LENGTH` x's in each row.

Hint: The character `'\n'` is a line break, and it will be useful here. Also, think about how you can use multiple for-loops.

Graphical Square

Now let's return to the 21st century. Your job is to write a **`GraphicsProgram`** that displays a square in the middle of the graphics window:



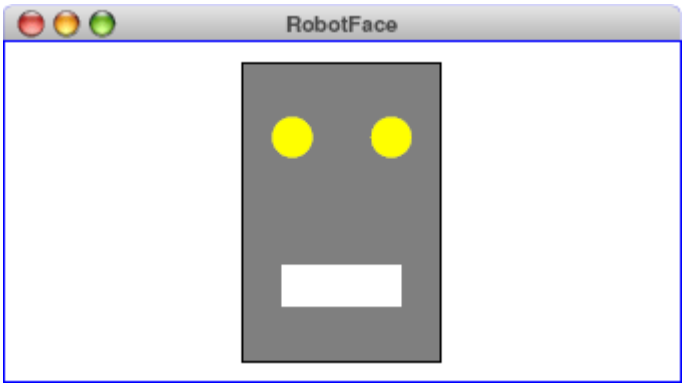

```
import "graphics";

const GWINDOW_WIDTH = 500;
const GWINDOW_HEIGHT = 200;
const SQUARE_LENGTH = 15;

function graphicalSquare() {

}
```

The constants provided in the starter code are shown above. You are encouraged to refer to the methods available in the Stanford Graphics Library found on page 68 of the reader.

Hint: Remember the origin for the coordinates of graphical objects is in the top-left corner. Think about how to use that information to help you center an object.

3. Drawing a face

Your job is to draw a robot-looking face like the one shown in the following sample run:



This simple face consists of four parts—a head, two eyes, and a mouth—which are arranged as follows:

- *The head.* The head is a big rectangle whose dimensions are given by the named constants `HEAD_WIDTH` and `HEAD_HEIGHT`. The interior of the head is gray.
- *The eyes.* The eyes should be circles whose radius in pixels is given by the named constant `EYE_RADIUS`. The centers of the eyes should be set horizontally a quarter of the width of the head in from either edge, and one quarter of the distance down from the top of the head. The eyes are yellow.
- *The mouth.* The mouth should be centered with respect to the head in the *x*-dimension and one quarter of the distance up from the bottom of the head in the *y*-dimension. The dimensions of the mouth are given by the named constants `MOUTH_WIDTH` and `MOUTH_HEIGHT`. The mouth is white.

Finally, the robot face should be centered in the graphics window.

19%3. Food for thought

We learned about the modulo operation (%) in one of the sessions. It turns out that this is a fairly useful and practical operation in programming. Can you think of a few use cases?