

## Hide and Seek

### Your Tasks (Mark these off as you go)

- ☐ Write code to print the result of a basic numeric operation
- ☐ Use the Math object to perform mathematical operations
- ☐ Write code to create a 2D grid of buttons
- ☐ Write code to make your 2D grid of buttons interactive
- ☐ Write code to add a monster to a random location on your grid
- ☐ Write code to keep track of the game score
- ☐ Receive credit for the group portion of this lab

### ☐ Write code to print the result of a basic numeric operation

In a previous lesson you applied the basic arithmetic operations below to calculate and print the results of various numerical problems.

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

Below are a few examples,

Code	Output
<code>console.log(79 + 3 * (4 + 82 - 68) - 7 + 19);</code>	145
<code>console.log((179 + 21 + 10)/7 + 181);</code>	211
<code>console.log(10389 * 56 * 11 + 2246);</code>	6401870

In addition to the more common operations like addition, subtraction, multiplication and division, we also learned about the modulus operator. The modulus operator prints the remainder of a division operation. For example, `console.log(5%3);` will print 2. This is because when 5 is divided by 3, the remainder is 2. Modulus gives the remainder. Modulus also handles negatives. The answer to `a%b` has the same sign as `a`. The sign of `b` is ignored.

Below are a few more examples,

Purpose	Code	Output
Modulus can be used to print the last digits of a base 10 number	<pre>var num = 12345; console.log(num%10); console.log(num%100); console.log(num%1000);</pre>	<div>5</div> <div>45</div> <div>345</div>
Modulus can be used to determine whether or not a number is even or odd	<pre>var evenNum = 542; console.log(evenNum%2);</pre>	0
Modulus can be used to determine whether or number is divisible by another number	<pre>var multipleOfThree = 51; console.log(multipleOfThree%3);</pre>	0

Write code to print the indicated digits in the numbers shown		
Number	Digits	Code
1234	234	
98765	65	
65467	7	

## □ Use the Math object to perform mathematical operations

The built in Math object in JavaScript allows us to perform calculations that go beyond the simple arithmetic operations we've seen. An example of how the Math object can be applied is illustrated below. The example below computes the square root of 17. The result is assigned to the variable p.

```
var p = Math.square(17); //prints 4.123105625617661
```

In the above example,

- *var p* is the variable to which the result of the Math operation is assigned
- *Math.* is the notation we use to access the library of Math functions in javascript
- *square(17)* is the operation we want to perform on the number 17. In this case, it is the square root.

JavaScript provides an extensive library of Math operations. Below is a description of some of them.

Operation	Example	Description
abs	Math.abs(n);	returns the absolute value of n
pow	Math.pow(n, p);	Returns the the number n raised to the p power
sqrt	Math.sqrt(n);	Returns the square root of n
ceil	Math.ceil(n);	Returns the highest whole number from n
floor	Math.floor(n);	Returns the lowest whole number form n
min	Math.min(a, b);	Returns the smaller of a and b
max	Math.max(a, b);	Returns the larger of a and b
random	Math.random();	Returns a random number in the range (0 <= r < 1)
round	Math.round(n);	Returns n rounded to the nearest whole number
PI	Math.PI	Returns 3.141592653589793

The below example illustrates how to access a number within a base 10 number using a combination of the basic numeric operations and the Math.floor method.

Code	Output
<pre>var num = 12345; num = num/100;//assigns 123.45 to num num = Math.floor(num);//assigns 123 to num var three = num % 10;//assigns 3 to three console.log(three);</pre>	3

Write code to print the number shown backwards,

```
var num = 161517;
```

A commonly used function from the Math class is .random(). The random() function generates a random number between 0 and 1, where 0 is inclusive, but 1 is not. The below code is illustrative,

Consider the outputs below,

Code	Output
<pre>console.log(Math.random()); console.log(Math.random()); console.log(Math.random()); console.log(Math.random()); console.log(Math.random());</pre>	<pre>0.9089451931993995 0.016448827905216623 0.8165579128778573 0.7551019286952072 0.46358292126929634</pre>

Random numbers in this range are not always useful, but it is possible to manipulate the function to get numbers in different ranges. For example, multiplying Math.random() by 10 changes the range and generates a random number between 0 and 10, where 0 is inclusive, but 1 is not.

Code	Output
<pre>console.log(Math.random()*10); console.log(Math.random()*10); console.log(Math.random()*10); console.log(Math.random()*10); console.log(Math.random()*10);</pre>	<pre>2.224839810285044 1.225871756015422 8.962166025213065 7.201893638439694 9.789143389392121</pre>

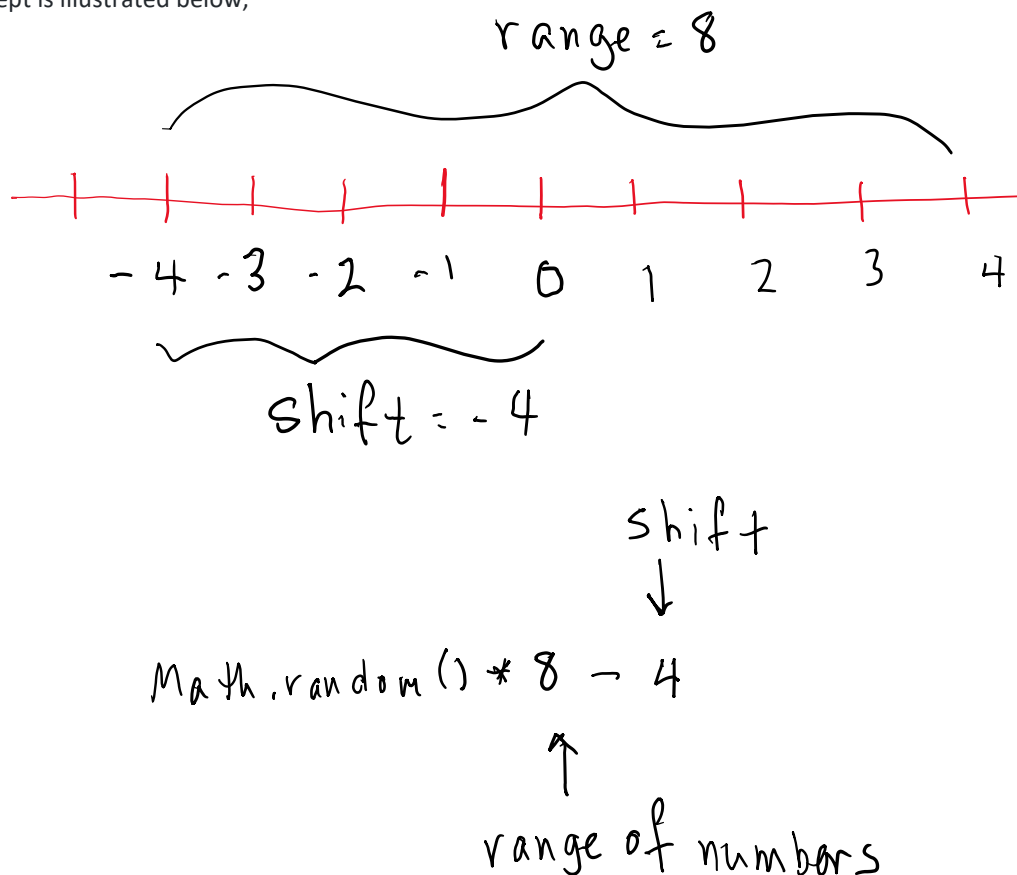
The next example illustrates how we can incorporate the `.floor` method to create whole numbers in the same range,

Code	Output
<code>console.log(Math.floor(Math.random()*10));</code>	0
<code>console.log(Math.floor(Math.random()*10));</code>	5
<code>console.log(Math.floor(Math.random()*10));</code>	3
<code>console.log(Math.floor(Math.random()*10));</code>	7
<code>console.log(Math.floor(Math.random()*10));</code>	8

Finally, it is also possible to shift the range from which the random numbers are generated. By subtracting 5 from the commands above, we can generate random numbers from -5 up to +5.

Code	Output
<code>console.log(Math.floor(Math.random()*10)-5);</code>	-5
<code>console.log(Math.floor(Math.random()*10)-5);</code>	2
<code>console.log(Math.floor(Math.random()*10)-5);</code>	1
<code>console.log(Math.floor(Math.random()*10)-5);</code>	-3
<code>console.log(Math.floor(Math.random()*10)-5);</code>	3

This concept is illustrated below,



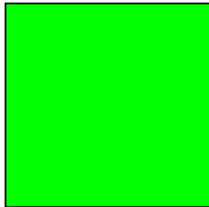
Write a "lucky" number generator that will create random integers between -100 and 100 (100 not inclusive).

The output to the console should be displayed in a single line separated by commas exactly as shown ( your numbers will obviously be different than my numbers ;- ) ).

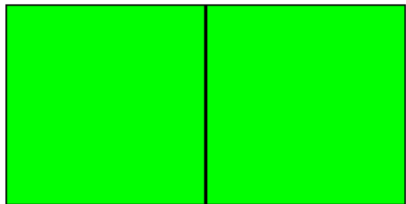
My lucky numbers are: 10, 66, 42

## □ Write code to create a 2D grid of buttons

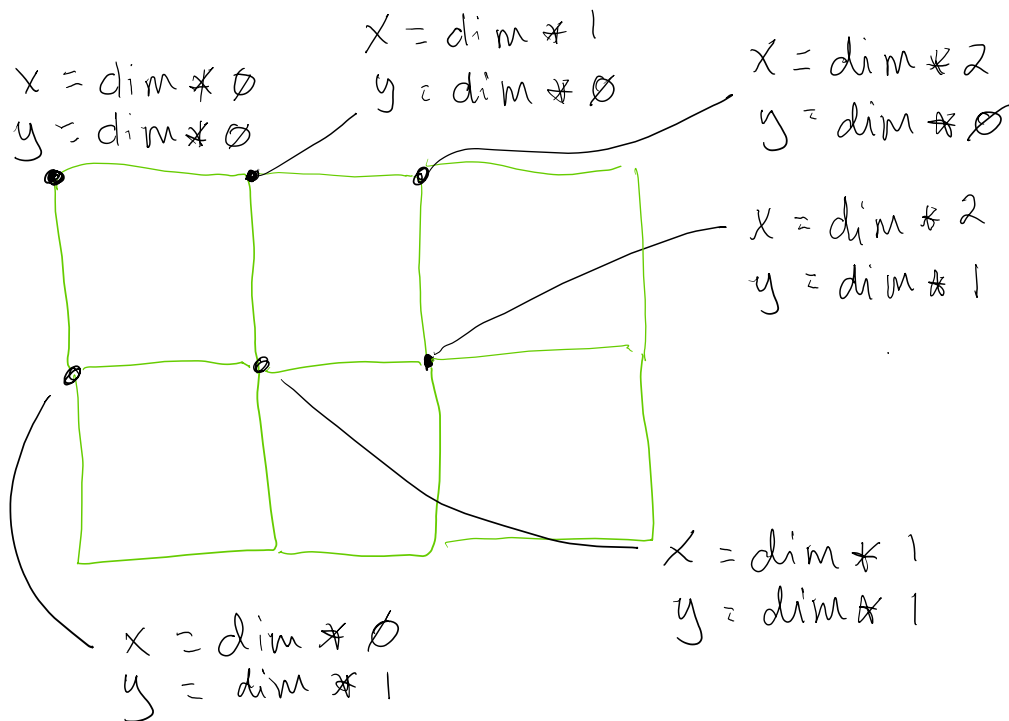
In our previous lab you wrote functions to create various shapes on the screen. The below code for example, creates a rectangle,

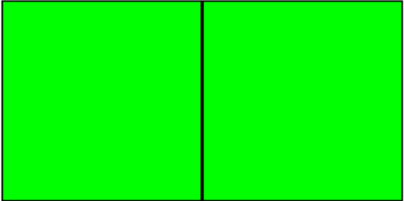
Code	Output
<pre>b = document.createElement("button"); b.style.border = "black solid thin"; b.style.width = "100px"; b.style.height = "100px"; b.style.backgroundColor = "lime"; b.style.position = "absolute"; b.style.left = "0px"; b.style.top = "0px"; document.body.append(b);</pre>	

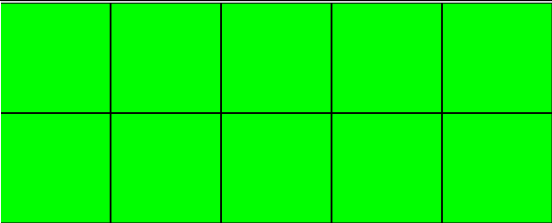
To create many buttons, we can repurpose our code into a function. In the below function we have created three parameters, d, xPos and yPos and assigned them to b.style.width, b.style.height, b.style.left, and b.style.right, respectively.

Code	Output
<pre>function makeButton(d, xPos, yPos){   var b = document.createElement("button");   b.style.border = "black solid thin";   b.style.width = d+"px";   b.style.height = d+"px";   b.style.backgroundColor = "lime";   b.style.position = "absolute";   b.style.left = xPos+"px";   b.style.top = yPos+"px";   document.body.append(b);   return b; }  var b0 = makeButton(100,10,10); var b1 = makeButton(100,110,10);</pre>	

We can make one more abstraction, by creating a new variable to represent the buttons dimensions and define the x and y position of the button in terms of it.



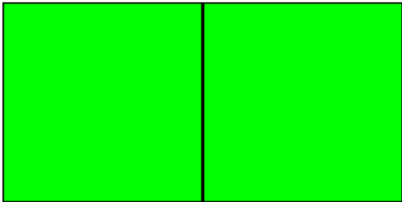
Code	Output
<pre> var dim = 100;  function makeButton(d, xPos, yPos){   var b = document.createElement("button");   b.style.border = "black solid thin";   b.style.width = d+"px";   b.style.height = d+"px";   b.style.backgroundColor = "lime";   b.style.position = "absolute";   b.style.left = xPos+"px";   b.style.top = yPos+"px";   document.body.append(b);   return b; }  var b0 = makeButton(dim, dim*0, dim*0); var b1 = makeButton(dim, dim*1, dim*0); </pre>	

Refer to the function above. Write code that calls the function to create the grid shown below. The position of each button should be defined in terms of <code>dim</code> .	
Code	Output
	

### □ Write code to make your 2D grid of buttons interactive

To make our grid interactive requires that we add action listeners to each of the buttons. We also need a way to tell which button is clicked. To do this, we will associate an id with each of our buttons. As we create our buttons we will assign an id to each button and an action listener. `check` is the function called when a button is clicked.

In the below example, we have added a forth parameter to represent the id. In the body of the function we assign the id to the id of the button and we also create an event listener to listen for a click. `check` is the function called when a button is clicked.

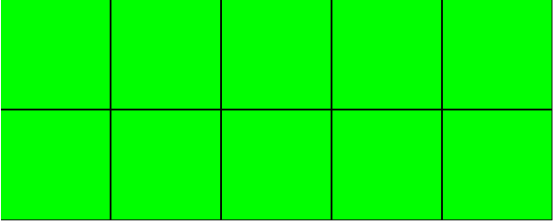
Code	Output
<pre> var dim = 100;  function makeButton(d, xPos, yPos, id){   var b = document.createElement("button");   b.style.border = "black solid thin";   b.style.width = d+"px";   b.style.height = d+"px";   b.style.backgroundColor = "lime";   b.style.position = "absolute";   b.style.left = xPos+"px";   b.style.top = yPos+"px";   b.id = id;   b.addEventListener("click", check);   document.body.append(b);   return b; }  var b0 = makeButton(dim, dim*0, dim*0, 0); var b1 = makeButton(dim, dim*1, dim*0, 1); </pre>	

The check function below is called when a button is clicked. Notice in the function that we pass a parameter. The parameter allows us to capture the object (or button in our case) that was clicked. Once we have the button, we can get its id log it to the console.

```
function check(e){  
    console.log(e.target.id);  
}
```

the event  
the ID of the target  
the target clicked

Modify your code calls from the previous example to include an id parameter. Each button you create using the parameter should have a unique id.

Code	Output
	

### ☐ Write code to add a monster to a random location on your grid

Now that we have a grid of clickable buttons, we will need to randomly assign one button to the “monster”.

In the body of the makeMonster function below create a random integer 0 thru 10 (10 not inclusive). Register a click event listener with the button that corresponds to the random number. When the button is clicked the found function should be called.

Inside the body of the found function, write code to change the background color of the random button to red. Recall, that the following command can be used to capture the id of the button clicked,

e.target.id;

```
function makeMonster(){  
  
}  
  
function found(e){  
  
}
```



## ❑ Write code to keep track of the game score

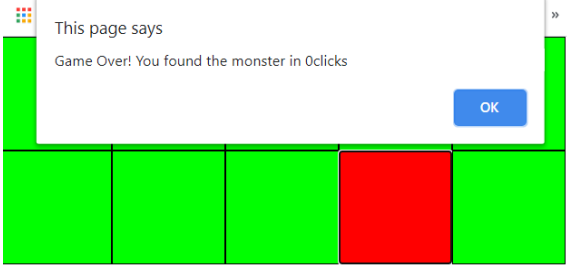
To finish our game we will need a way to keep track of how many clicks it took to find our monster. We will need a variable called score to keep track of this. We will initialize score to the number of buttons on our grid,

```
var score = 10;
```

Each time button is clicked that does not have the monster, you will need to deduct one from the score,

The check function is called each time a button is clicked. Inside the body of the check function deduct one from the score.

In the body of the found function alert the user "Game Over! You found the monster in" + score + "clicks".

Code	Output
<pre>function check(e){  } function found(e){  }</pre>	

## ❑ Receive Credit for this lab guide

Submit this portion of the lab to Pluska to receive credit for the lab guide. Once received, your completed code challenges will also be graded and will count towards your final lab grade.