Week six contained many things to learn. In week six, we have learned about Logic operators such as “If” and learned about debugging. Let’s start off with logic operators since it is required for the examples I will give for debugging.  
  
  
Logic operators. The brain of the program. Without them, your computer would do EVERYTHING. Without those logic operators, your computer wouldn’t know when it needs to do a specific task, so instead it just does every task. These operators are essentially telling your program or computer, if this happens then do that. Without operators, the key’s I am pressing right now on my computer wouldn’t work, the computer would write all of them every second, have every program close and open and the computer launch and stop.  
  
Enough with talking about why they are useful, I will now explain to you how to use them for yourself! Let’s start off with a simple one, the if statement. All this take is a parameter and it does the task provided if it is true. Let me show you.  
  
var number = 150;

If(number > 100) {

fill(0,0,0);  
}  
  
This asks the program, is the number bigger then 100? If it is, it performs the function “fill()”. If you are wondering what the “>” symbol (bigger than symbol), you can stop wondering since it will be explained in a second along with all its fellow operators. Let’s quickly finish talking about the if statement though. So, to use an if statement, simply write “if()” and write a condition in the parenthesis, for example “if(150 > 100)”, this will be true. This is not done though; The program will give you an error. It might look like it should work but it is missing one more thing, squiggly brackets (the proper name for them for developers, everyone should use the term, obviously). Some languages might allow you to format it differently, I will show an example in java, but most languages use the same way.  
  
Java:  
  
int number = 100;  
If(number >= 100){

System.out.println(“Hello!”);  
}

Also Java:  
  
  
int number = 100;

If(number >= 100) System.out.println(“Hello!”);

Although, the most popular format (by that I mean universal format across different programming languages) is the first example (not the actual functions, that is java unique, I meant just the formatting).  
  
Let’s quickly talk about operators now. Operators are a very simple concept. They are essentially gates (logic gates, such as or, and, bigger than, smaller than, etc…). There are quite a few of these, you can find all of them by searching “JavaScript operators” on the internet but here are the most prevalent ones.  
  
=== Equal in value and type  
== Equal in value  
! not equal to  
> greater then  
< less than  
&& and  
|| or  
  
  
Let’s now go to the else statement, the if statement isn’t the only one there is. The else statement is simply per say an addition to the if statement. Let me show you.

var number = 150

If(number >= 150) {

fill(0,0,0)  
} else {

fill(255,255,255);

}  
  
In this specific case, the else/if doesn’t server much purpose since the number won’t change but… what this does is ask the program if the number is bigger or equal to 150, if it is then change the colour to black. If it is less than 150, then you do what is inside the else statement (in this case, change the colour to white).   
  
After the else statement, you have the “else if” statement. Once again very simple, this is essentially a secondary if statement. Much like the else statement, it is an addition to the original if statement except you can give it another parameter.

var number = 150;

If(number >= 150){

fill(0,0,0);

}else if(number < 150 && number > 100){

fill(100, 100, 100);

}else {

fill(255,255,255);

}

This checks if the number is more than or equal to 150. If it is than change colour to black. If not, it then checks if it is less than 150 and if it is more than 100. Once again, changes the colour and if all those are false then it does the else statement.  
  
Sometimes, you want to give a variable instead in the if statement. If that is the case, let me introduce you to “Booleans”. This is like all the other variables(since it is one), except that it can only be true or false, For example;  
  
var MyBool = true;  
  
This creates a Boolean variable that is true. You can then use this variable in your functions instead of giving it a condition.   
  
if(MyBool) {

…

}

This can also be written as…  
  
if(MyBool === true){

…  
}  
  
You can also say “=== false” but the first example automatically sees if the Boolean is true.  
  
  
Let’s do another variable type. This variable type is random. Actually, it isn’t really a variable type since it returns an int value, anyways, as the name suggests, this function gives a random number from the parameters given. This is how you can use it.  
  
var number = random(0, 100);

This line of code makes a variable with a random value between 0 and 100(including 0 and 100). Careful! This gives a value with decimals. If you do not want decimals you can either use floor, ceil or round.  
  
floor() Rounds to nearest integer(proper rounding, if bigger than or equal to .5 then rounds up, else rounds down).  
ceil() Rounds Up.  
floor() Rounds down.  
  
There is one last thing for this section! After that, we have debugging (which should be really quick). This last thing is a Boolean called mouseIsPressed. As the name suggests, this checks if the mouse is being pressed. If you put it in an if statement (as a parameter), it will check if your mouse is clicked and perform an action based on that.  
  
The next section, which is very quick, is Debugging. Debugging is a very simple concept. Debugging is when you try to find errors/mistakes in computer hardware or software. In other words, your program might have errors, debugging is finding those errors and removing them by changing code.  
  
If you have an IDE(Integrated Development Environment), you might not come across one type of error, although if you are using a plain text editor you might come across spelling mistakes. In this case, if you are not using an IDE, you could either search for errors or you could comment out lines (using “//” in front). Anyways, other than that the most common errors found in code are functions not executing or logic operators not triggering when you think they should. One way to fix this is to search a visualize all of the code in your head but that is too long of a process to find where something does not work. Another thing you can do is use “println()”. This function displays a string of your choice (You can do this by putting a variable/string in the parenthesis). If you have a println() in every function or logic operators, you will find where your code breaks down very quickly. Here is an example.  
  
var number = 150

If(number < 150){

println(“A”);

}else {

println(“B”);

}  
  
Hmmmm… it says B. Ohhh!!! Woops, I used less than instead of greater than. Well, lets fix that. Now it works properly and shows A.  
  
If(number > 150){

println(“A”);  
}else {

println(“B”);

}  
  
And there you go, now you can properly start making interactive programs! With the if statement, boolean, debugging, random and everything stated above, you are unstoppable.