Build the next Great Video Game Using the Hottest Tools

Lesson 3

Here is the code that we will start off with.

```
function setup(){
        createCanvas(800, 400);
}

function draw(){
        background(50);
}
```

Here, we create a variable. Variables are boxlike containers that contain values.

We create a variable using the var keyword. Next, we type the name of the variable.

In this example, we will name our variable x. Then, we type in the equals sign (=). Finally, we give the x variable the value of 70.

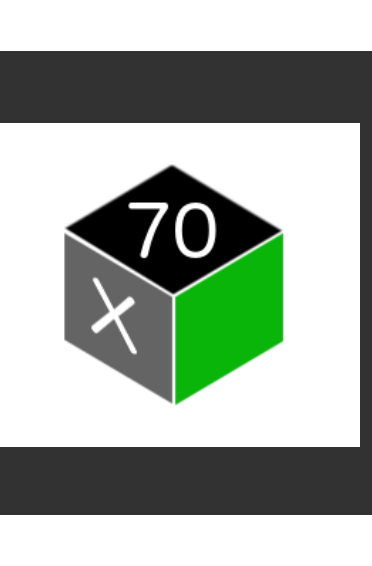
The value of the variable is always on the right side, while the name of the variable is always on the left.

The equals sign (=) is not used to mean equality; it is used instead to assign the value of 70 into the x variable. In this sense, we are moving from right to left. We are taking 70 and putting it into a box named x.

```
var x = 70;

function setup(){
        createCanvas(800, 400);
}

function draw(){
        background(50);
}
```

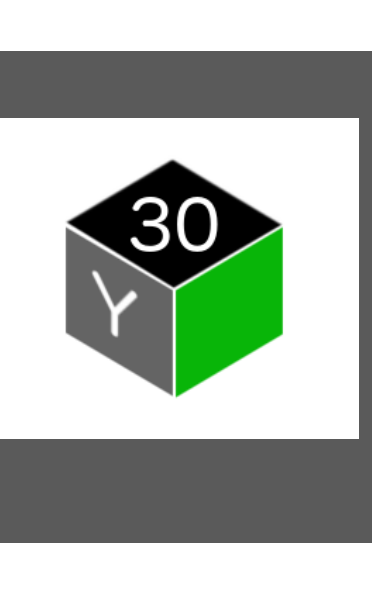


Next, we create another variable named y, and we give it the value of 30.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){
        background(50);
}
```



We now draw a circle using the ellipse() command.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){

        background(50);

        ellipse(70, 30, 50, 50);
}
```

Here, we position the center of our circle at x 70, y 30, with a width and a height of 50 pixels each.

Instead of typing in the number 70, we can instead use the value of the x variable.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){

        background(50);

        ellipse(x, 30, 50, 50);
}
```

We can do the same thing with the y variable, using its value instead of the number 30.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){

        background(50);

        ellipse(x, y, 50, 50);
}
```

We now want to animate the circle moving right. In order to do so, we will increase the value of the x variable.

If we look again at our coordinate system, we can see that a circle positioned at x 400, y 0 is to the right of one positioned at x 0, y 0.

Therefore, if we increase the value of the x variable from 70 to 71, the circle will move right by 1 pixel.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){

        background(50);

        ellipse(x, y, 50, 50);

        x = x + 1;

}
```

When looking at the new code, we move from right to left. If we begin on the right side with x, which has a value of 70, we add 1 to it, which gives us 71.

We then put the value of the right side, which is 71, into x on the left side. The value of x is now 71.

If we continuously add 1 to the value of x each time the draw() function loops, the value of x will steadily increase and the circle will animate right.

It is in this way that we increase the value of the x variable. If the draw() function loops 60 times a second, then for each second we can expect our circle to move 60 times.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){

        background(50);

        ellipse(x, y, 50, 50);

        x = x + 2;

}
```

Now, every time the draw() function loops, we increase the value of the x variable by 2 instead of 1.

If x, on the right side, has a value of 70 and we add 2 to it, we now have 72 on the right side which we put back into the x variable on the left.

This makes the circle animate right twice as fast.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){

        background(50);

        ellipse(x, y, 50, 50);

        x = x + 5;

}
```

Increasing the value of x by 5 each time the draw() function loops allows the circle to animate even faster, 2.5 times faster than before.

It is in this way that we can control the speed of the circle.

Next, we will comment out some code.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){

        background(50);

        ellipse(x, y, 50, 50);

        //x = x + 5;

}
```

By adding two slashes (//) before our code, we render it inactive. P5 now ignores the code.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){

        background(50);

        ellipse(x, y, 50, 50);

        //x = x + 5;

        y = y + 5;

}
```

Here, we add 5 to the y variable every time the draw() function loops.

The y variable controls the vertical direction of the circle; increasing its value will animate the circle moving down the canvas.

The first time through the draw() function, the value of y increases from 30 to 35, moving the circle 5 pixels down the canvas.

The next time the draw() function loops, we increase the value of the y variable from 35 to 40. The circle animates continuously down the canvas.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){

        background(50);

        ellipse(x, y, 50, 50);

        //x = x + 5;

        //y = y + 5;

}
```

We then comment out the code that increases the value of y, rendering it inactive. We also remove the slashes (//) in front of the code that increases the value of x, reactivating it.

Next, we create a condition. Conditions are tests. If the condition is true, we run the code associated with the test.

If the condition is false, we simply ignore the code associated with the test.

Tests will sometimes be false and other times true. Again, the code associated with the test will simply only run if the condition is true.

When the condition is no longer true, P5 will simply ignore the code associated with the test.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){

        background(50);

        ellipse(x, y, 50, 50);

        //x = x + 5;

        //y = y + 5;

        if(x > width){

              x = 0;
        }

}
```

Conditions use the if keyword. Here, if the value of x is greater than the width of the canvas (800 pixels, the right side of the canvas), we want the value of the x variable to be set back to 0 (the left side of the canvas).

The value of x starts off at 70. Because 70 is not greater than 800, the condition is false.

When the value of x increases to 75, 80, or 85, those numbers are still not larger than 800, so the condition remains false, and the code associated with the condition is ignored.

Later, when x becomes 800, the condition is still false, because 800 is not larger than 800. However, when x increases to 805, then the condition becomes now true, because 805 is larger than 800.

The block of code associated with the test is now executed, and the value of x becomes 0. This moves the circle from the right side of the canvas to the left.

When we test the condition again, 0 is not greater than 800, so the condition is once again false, and the code associated with the test is once again ignored.

We continue to add 5 to the x variable. If we look closely, we can see that when half the circle gets to the right edge of the canvas, it goes back to the left side of the canvas. This is because we are calculating the x value from the center of the circle.

This means that when the center of the circle gets to the right side of the canvas, we set x back to 0, where again, the center of the circle begins animating on the left side of the canvas, at x 0.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){

        background(50);

        ellipse(x, y, 50, 50);

        x = x + 5;

        //y = y + 5;

        if(x > width + 25){

              x = -25;
        }

}
```

In order to change the code so that the entire circle fully animates off the right side of the canvas, we add the radius of the circle (half the width of the circle), which is 25 pixels, to the condition.

Now, when the circle animates completely off the right side of the canvas, that it, x is greater than 800 + 25, we send the circle back to the left side of the canvas.

We also make sure that the entire circle is sent completely off the left side of the canvas, giving it an x value of -25.

Next, we will comment out this code using a multi-line comment, thus rendering it inactive.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){

        background(50);

        ellipse(x, y, 50, 50);

        x = x + 5;

        y = y + 5;

        /*
        if(x > width + 25){

              x = -25;
        }
        */

}
```

Multi-line comments are useful when we want to deactivate multiple lines of code. Multi-line comments begin with a slash and a star (/*), and end with a star and a slash (*/). All the code inside the area is now inactive. This also saves us from typing double slashes (//) throughout our code.

We also comment out the code that animates the x value, making it inactive as well. We then reactivate the code that animates the circle moving down the canvas by taking away the two slashes (//).

The circle again animates down the canvas.

Next, we create a conditional statement for the y animation.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){

        background(50);

        ellipse(x, y, 50, 50);

        //x = x + 5;

        y = y + 5;

        if(y > height){

              y = 0;
        }

        /*
        if(x > width + 25){

              x = -25;
        }
```

Here, if the value of y is greater than the height of the page (400), that is, if the circle animates to the bottom of the canvas, we set the value of y back to 0, sending the circle up to the top of the canvas.

Again, we have an issue with the animation. We want the entire circle to animate off the bottom of the canvas, and restart off the screen at the top of the canvas.

In order to fix this, we will add the radius of the circle (half the width) to the condition.

```
var x = 70;
var y = 30;

function setup(){
        createCanvas(800, 400);
}

function draw(){

        background(50);

        ellipse(x, y, 50, 50);

        //x = x + 5;

        y = y + 5;

        if(y > height + 25){

              y = -25;
        }

        /*
        if(x > width + 25){

              x = -25;
        }
```

Now, when the value of the y variable is fully off the canvas, that is, when its y value is greater than the height of the canvas (400) plus 25 pixels, we send the circle to the top of the canvas.

We also send it up past the top of the canvas by 25 pixels, setting the y value to -25, so that the circle begins animating completely off the canvas.

Now, when the circle animates, it loops over and over, going completely off the canvas at the bottom before restarting its animation at the top of the canvas.