A button function

If you went through the Intro to JS course, then you made a few simple buttons in the Logic challenges Your First Button and Smarter Button. In case you forgot, let's re-hash how to make a simple button.

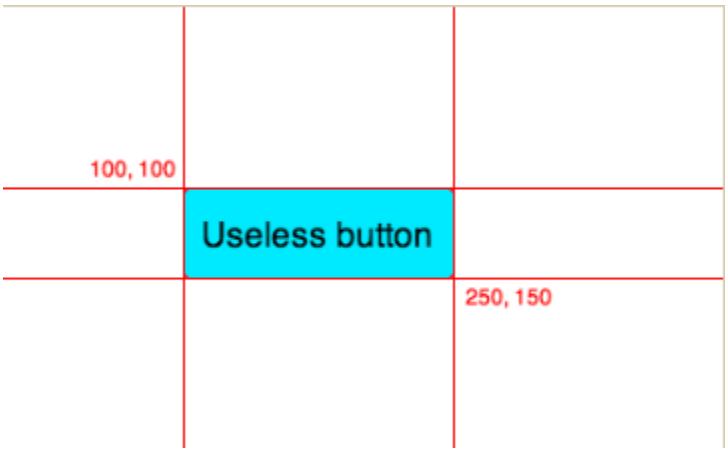
First, what are the bare minimum features of a button?

1. A shape on the canvas (typically a rectangle)
2. Includes a label or icon that describes what it will do
3. Responds to the user clicking on it (but not elsewhere)

We can achieve #1 and #2 pretty easily:

```
fill(0, 234, 255);
rect(100, 100, 150, 50, 5);
fill(0, 0, 0);
textSize(19);
text("Useless button", 110, 133);
```

To achieve #3, we need to define a `mouseClicked` function that will get called when the user clicks, and inside of that, we need to check whether `mouseX` and `mouseY` are within the bounding box of the button. For the button above, it spans from x=100 to x=250, and from y=100 to y=150, as illustrated below:



We can check those coordinates by `&&`ing together four conditions:

```
mouseClicked = function() {
    if (mouseX >= 100 && mouseX <= 250 &&
        mouseY >= 100 && mouseY <= 150) {
        println("Still pretty useless");
    }
};
```

Try clicking on it and off it below to verify that it works:

```
background(255, 255, 255);

fill(0, 234, 255);

rect(100, 100, 150, 50, 5);

fill(0, 0, 0);

textSize(19);
```

```
text("Useless button", 110, 133);


mouseClicked = function() {
   if (mouseX >= 100 && mouseX <= 250 &&
       mouseY >= 100 && mouseY <= 150) {
       println("Still pretty useless");
   }
};
```

It definitely works, but it also worries me. I'm worried that it's not highly reusable code. How much work will I have to do if I want to change the position of the button? (Try it above!) I see a lot of "hard-coded" numbers in the code -- like the coordinates in the `mouseClicked` function, and I immediately start to wonder if there isn't a cleaner way.

To start with, let's make variables of the position and size, so that we can change them in a single place and have the button clicking work still. I've added `btnX`, `btnY`, `btnWidth` and `btnHeight` to the program below. Try changing their values around and clicking the button:

```
var btnX = 100;
var btnY = 100;
var btnWidth = 150;
var btnHeight = 50;


background(255, 255, 255);
fill(0, 234, 255);
rect(btnX, btnY, btnWidth, btnHeight, 5);
fill(0, 0, 0);
textSize(19);
textAlign(LEFT, TOP);
text("Useless button", btnX+10, btnY+btnHeight/4);


mouseClicked = function() {
   if (mouseX >= btnX && mouseX <= (btnX+btnWidth) &&
       mouseY >= btnY && mouseY <= (btnY+btnHeight)) {
       println("Still pretty useless");
```

```
  }

};
```

Well, that's better. But still, how much work will I have to do if I want to add an extra button? Do I have to copy and paste all of that, and make `btn2X`, `btn2Y`? Ugh, that doesn't sound fun at all. This sounds like good motivation to write a function that will take care of doing everything that's the same for buttons, and use parameters to take care of what's different. We could write it like this, turning the variables into parameters:

```
var drawButton = function(btnX, btnY, btnWidth, btnHeight) {
    background(255, 255, 255);
    fill(0, 234, 255);
    rect(btnX, btnY, btnWidth, btnHeight, 5);
    fill(0, 0, 0);
    textSize(19);
    textAlign(LEFT, TOP);
    text("Useless button", btnX+10, btnY+btnHeight/4);
};
```

Then we'd call it like so:

```
drawButton(100, 100, 150, 50);
```

But, uh oh, what about our `mouseClicked` code? Do you see what the problem would be with it?

```
mouseClicked = function() {
    if (mouseX >= btnX && mouseX <= (btnX+btnWidth) &&
        mouseY >= btnY && mouseY <= (btnY+btnHeight)) {
        println("Still pretty useless");
    }
};
```

If we had all this code together, we'd get an error from Oh Noes that "btnX is not defined" - and he's right! We turned `btnX` into a function parameter, which means it's no longer a global variable. That's great for re-usability of the `drawButton` function, but now the `mouseClicked` function has no way to know what coordinates to check.

So we need to figure out a nice way to pass the information into `drawButton` **and** make that information available to `mouseClicked`. I can think of a few options:

1. Re-introduce global variables for the position and size (`btnX, btnY, btnWidth, btnHeight`)
2. Introduce a global array that stores all the parameters (`var btn1 = [...];`)
3. Introduce a global object that stores the parameters (`var btn1 = {..}`)
4. Use object-oriented principles to define the button and store the properties (`var btn1 = new Button(...)`)

Which to choose? Well, I don't like the first one because we'll have to add **so many** global variables, and I have an allergy to global variables. I don't love the second technique because its hard to read code that grabs data based on array indices. I like the third technique because it introduces only one global variable, and will produce more readable code. I also like the fourth technique, using object-oriented principles to create generic `Button` object types, but let's save that for later.

We can create our global `btn1` object like so:

```
var btn1 = {
    x: 100,
    y: 100,
    width: 150,
    height: 50
```

```
};
```

And change the `drawButton` function to accept a single object that it then grabs properties from:

```
var drawButton = function(btn) {
    background(255, 255, 255);
    fill(0, 234, 255);
    rect(btn.x, btn.y, btn.width, btn.height, 5);
    fill(0, 0, 0);
    textSize(19);
    textAlign(LEFT, TOP);
    text("Useless button", btn.x+10, btn.y+btn.height/4);
};
```

The `mouseClicked` function will check the properties of the global variable:

```
mouseClicked = function() {
    if (mouseX >= btn1.x && mouseX <= (btn1.x+btn1.width) &&
        mouseY >= btn1.y && mouseY <= (btn1.y+btn1.height))     {
        println("Still pretty useless");
    }
};
```

Try it out below! Like before, try changing different parameters of the button and seeing if everything still works:

```
var btn1 = {
  x: 100,
  y: 100,
  width: 150,
  height: 50
};

var drawButton = function(btn) {
  background(255, 255, 255);
  fill(0, 234, 255);
  rect(btn.x, btn.y, btn.width, btn.height, 5);
  fill(0, 0, 0);
  textSize(19);
  textAlign(LEFT, TOP);
  text("Useless button", btn.x+10, btn.y+btn.height/4);
};
```

```
mouseClicked = function() {
   if (mouseX >= btn1.x && mouseX <= (btn1.x+btn1.width) &&
      mouseY >= btn1.y && mouseY <= (btn1.y+btn1.height))    {
      println("Still pretty useless");
   }
};



drawButton(btn1);
```

The whole point of that was to enable us to easily add more buttons, the ultimate re-usability test. Can we do it? Ba Bum BUM.

We'll start with a new global variable, `btn2`, offset in the y direction from the first button:

```
var btn2 = {
    x: 100,
    y: 200,
    width: 150,
    height: 50
};
```

Then we'll draw that button:

```
drawButton(btn2);
```

That will succeed in drawing 2 buttons on the canvas, but only the first will respond to clicks. We can make the second respond by duplicating the logic and swapping `btn2` for `btn1`, like so:

```
mouseClicked = function() {
    if (mouseX >= btn1.x && mouseX <= (btn1.x+btn1.width) &&
        mouseY >= btn1.y && mouseY <= (btn1.y+btn1.height))     {
        println("Still pretty useless");
    }

    if (mouseX >= btn2.x && mouseX <= (btn2.x+btn2.width) &&
        mouseY >= btn2.y && mouseY <= (btn2.y+btn2.height))     {
        println("2nd one still quite useless!");
    }
};
```

But doesn't all that repeated code just make your nose crinkle? Let's make a function `isMouseInside` that knows how to check any button object and return true if the mouse was inside it:

```
var isMouseInside = function(btn) {
    return (mouseX >= btn.x &&
            mouseX <= (btn.x+btn.width) &&
            mouseY >= btn.y &&
            mouseY <= (btn.y+btn.height));
};
```

And now we can use that function inside `mouseClicked` to greatly reduce the amount of repetitive code:

```
mouseClicked = function() {
    if (isMouseInside(btn1))     {
        println("Still pretty useless");
    } else if (isMouseInside(btn2))     {
        println("2nd one still quite useless!");
    }
};
```

And there we have it! We've used functions to draw multiple buttons, and made it relatively easy to add new buttons. Try it out below:

```
var btn1 = {

  x: 100,

  y: 100,

  width: 150,

  height: 50

};



var btn2 = {

  x: 100,

  y: 200,

  width: 150,

  height: 50

};




var drawButton = function(btn) {

  fill(0, 234, 255);

  rect(btn.x, btn.y, btn.width, btn.height, 5);

  fill(0, 0, 0);

  textSize(19);

  textAlign(LEFT, TOP);

  text("Useless button", btn.x+10, btn.y+btn.height/4);

};
```

```
var isMouseInside = function(btn) {
   return (mouseX >= btn.x &&
          mouseX <= (btn.x+btn.width) &&
          mouseY >= btn.y &&
          mouseY <= (btn.y+btn.height));
};


mouseClicked = function() {
   if (isMouseInside(btn1))    {
      println("Still pretty useless");
   } else if (isMouseInside(btn2))    {
      println("2nd one still quite useless!");
   }
};




drawButton(btn1);
drawButton(btn2);
```

We could keep going -- making arrays of all the buttons in a program, making it possible to customize the label and color of a button -- but hopefully this has given you a good basis in how to create simple buttons using functions. Next, we'll walk through how to create buttons using object-oriented principles.