

Functions

CF1

Gabe Johnson | September 25 & 26, 2024

What's a function?

- A named block of code that has inputs (maybe) and outputs (maybe)
- We've used built-in functions before: draw, rect, ellipse, noStroke, etc.
- Make functions when you have an action that you'd like to name and re-use
- If the function takes parameters, the behavior can depend on them
- ... e.g. a single robot function can take different geometry and color params

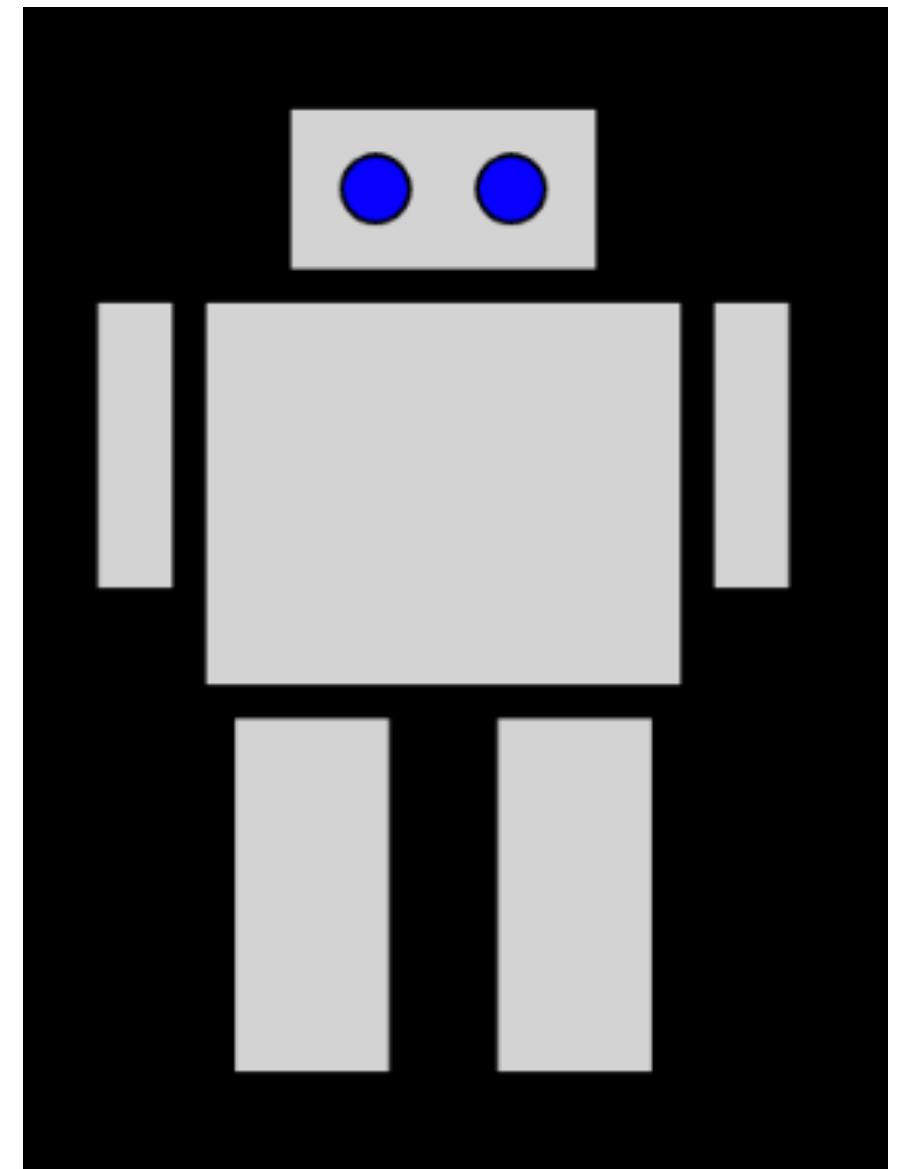
Modularity

What is the minimum information we need to do something?

- In the videos, he uses the term “modularity”
- I tend to think of this as “behavior”. E.g. bounce, move, display.
- You can think of a function as a verb. It performs some action.
 - It might modify the world, e.g. putting pixels on the screen
 - It might compute something, e.g. `add(2, 4)` could *return* 6

Calling functions

- Simply provide the name of the function, and then arguments in parentheses
- You've done this a zillion times: `rect(10, 30, 200, 70)`
- `robot(200, 80, 140, 200, "blue", "lightgray");`
- Use different arguments to get different results without having to repeat your code a zillion times

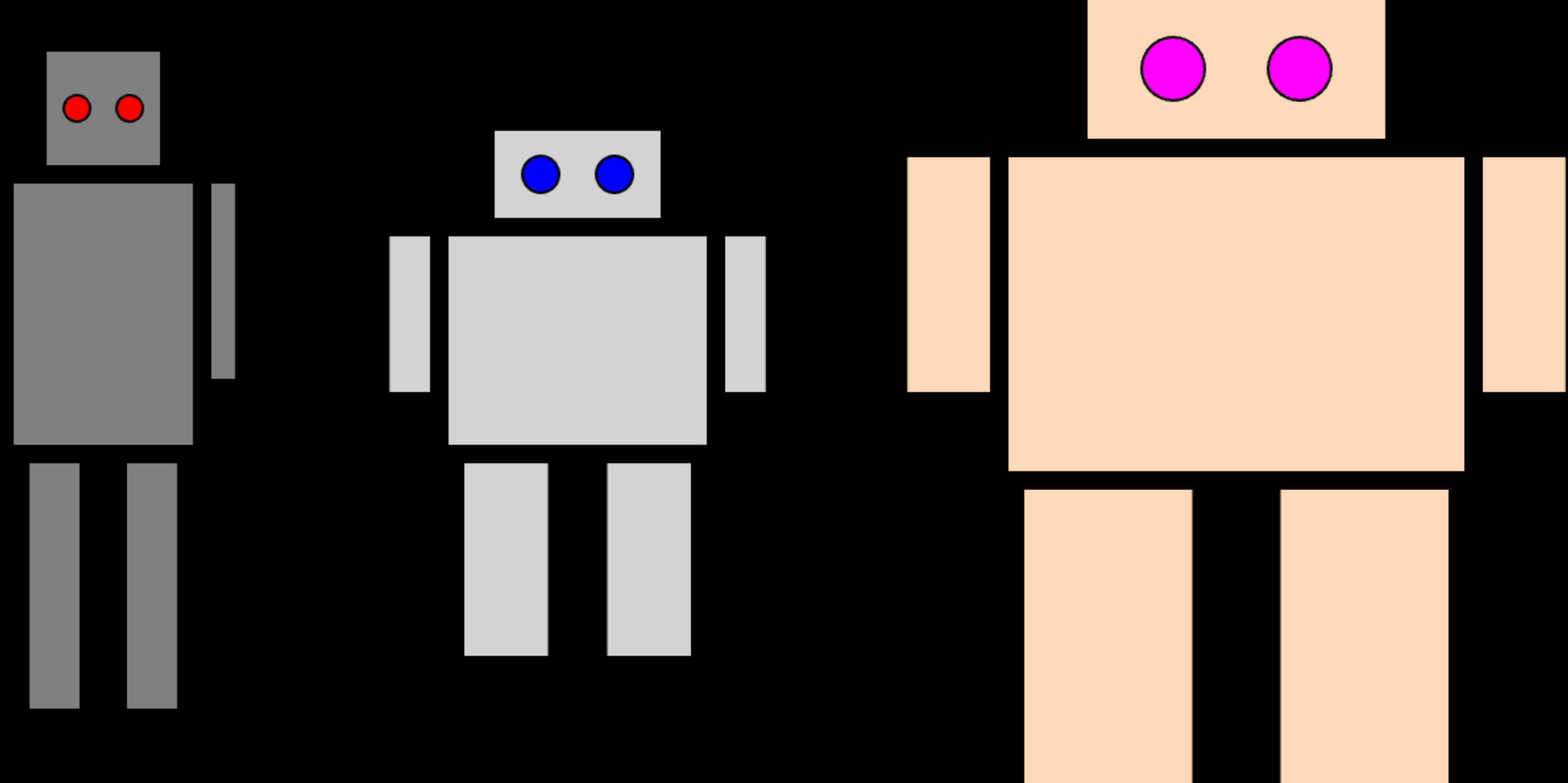


```

1 function setup() {
2   createCanvas(700, 400);
3 }
4
5 function draw() {
6   background(0);
7   robot(40, 50, 100, 250, "red", "gray");
8   robot(200, 80, 140, 200, "blue", "lightgray");
9   robot(400, 30, 240, 300, "magenta", "peachpuff");
10 }
11
12 function robot(x, y, w, h, eyeColor, bodyColor) {
13   // intentionally long passage of code to demonstrate why
14   // it is helpful to pack stuff off into functions.
15   // this one takes parameters - the x/y position, robot
16   // width and height, and colors for eye and body
17   const centerX = w / 2;
18   const eyeRadius = w * 0.1;
19   const eyeOffset = w * 0.1;
20   const leftEyeX = centerX - eyeOffset;
21   const rightEyeX = centerX + eyeOffset;
22   const armGap = 6;
23   const headWidth = 0.5 * w - 6;
24   const headHeight = 0.2 * h - 6;
25   const headLeft = centerX - headWidth / 2;
26   const eyeCenterY = headHeight / 2;
27   const armWidth = 0.16 * w - 6;
28   const armHeight = 0.3 * h;
29   const bodyWidth = 0.75 * w - 6;
30   const bodyHeight = 0.4 * h;
31   const bodyLeft = centerX - bodyWidth / 2;
32   const bodyRight = bodyLeft + bodyWidth;
33   const bodyTop = headHeight + armGap;
34   const bodyBottom = bodyTop + bodyHeight;
35   const leftArmLeft = bodyLeft - armGap - armWidth;
36   const rightArmLeft = bodyRight + armGap;
37   const legHeight = h - (bodyBottom + armGap);
38   const legWidth = armWidth * 2;
39   const legTop = bodyBottom + armGap;
40   const leftLegLeft = centerX - bodyWidth / 2 + armGap;
41   const rightLegRight = centerX + bodyWidth / 2 - armGap;
42   const rightLegLeft = rightLegRight - legWidth;
43
44   push();
45   translate(x, y);
46   // head using body color
47   fill(bodyColor);
48   rect(headLeft, 0, headWidth, headHeight);
49   // eyes, filled with eye color
50   fill(eyeColor);
51   circle(leftEyeX, eyeCenterY, eyeRadius);
52   circle(rightEyeX, eyeCenterY, eyeRadius);
53   // fill using body color again
54   fill(bodyColor);
55   // left arm
56   rect(leftArmLeft, 0.2 * h, armWidth, armHeight);
57   // right arm
58   rect(rightArmLeft, 0.2 * h, armWidth, armHeight);
59   // body
60   rect(bodyLeft, bodyTop, bodyWidth, bodyHeight);
61   // left leg
62   rect(leftLegLeft, legTop, legWidth, legHeight);
63   // right leg
64   rect(rightLegLeft, legTop, legWidth, legHeight);
65   pop();
66 }

```

One function made all these robots by using different arguments for position, size, and colors.



<https://editor.p5js.org/johnsogg/sketches/NL0J0Vlv>

Function arguments

Arguments & Parameters

- When you *call* a function, the values you send in are called *arguments*
- When you *define* a function, the values you receive are called *parameters*
- If a function has no arguments, it still needs the parentheses, like:
 - (when invoking) doSomething()
 - (when defining) function doSomething() { }
-

Defining functions

```
function robot(x, y, w, h, eyeColor, bodyColor) {  
    // do what you need using the parameters  
    // listed above  
}
```

Defining functions

Start with ‘function’ keyword



```
function robot(x, y, w, h, eyeColor, bodyColor) {  
    // do what you need using the parameters  
    // listed above  
}
```

Defining functions

Then name give it a descriptive name like ‘robot’

```
function robot(x, y, w, h, eyeColor, bodyColor) {  
    // do what you need using the parameters  
    // listed above  
}
```

Defining functions

After the function name, always include opening and closing parentheses



```
function robot(x, y, w, h, eyeColor, bodyColor) {  
    // do what you need using the parameters  
    // listed above  
}
```

Defining functions

Inside the parens, list the parameters in the order you expect

```
function robot(x, y, w, h, eyeColor, bodyColor) {  
    // do what you need using the parameters  
    // listed above  
}
```

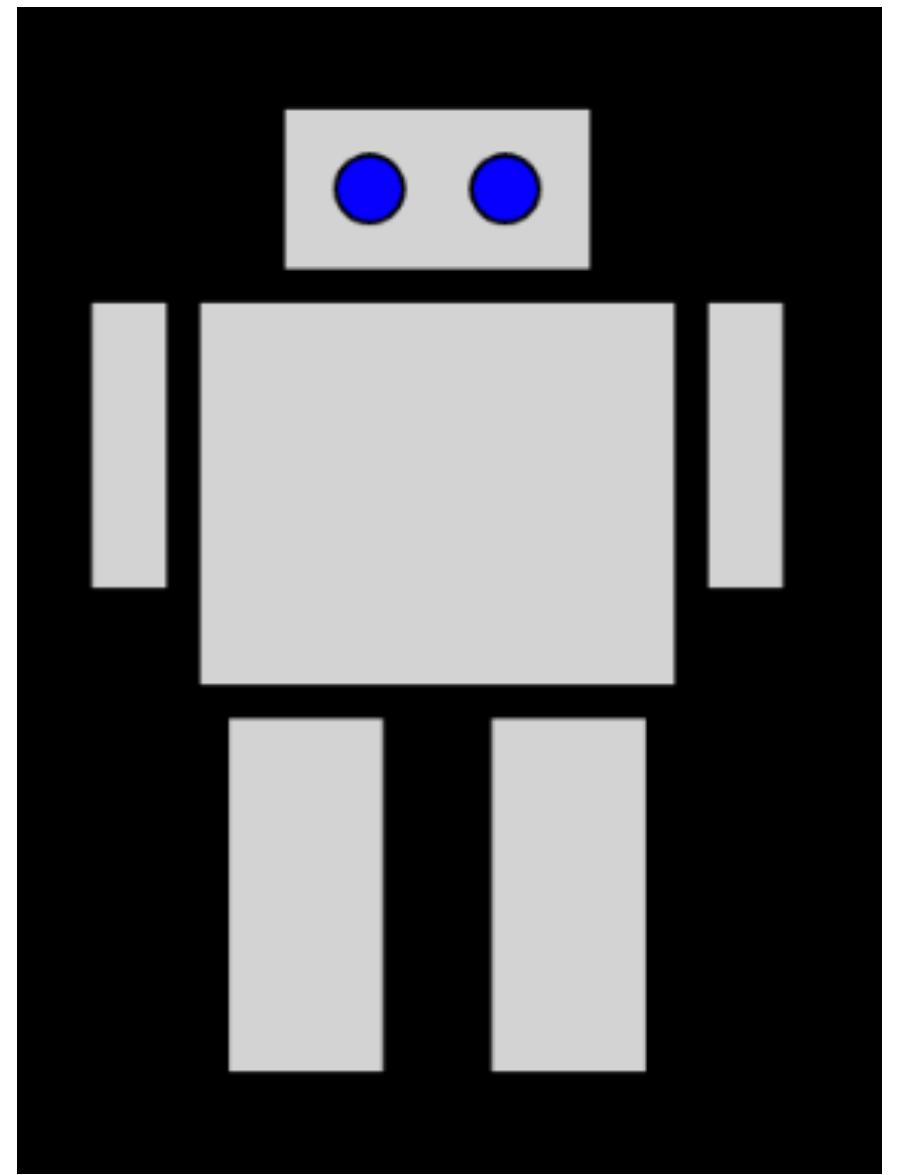
Defining functions

Then use curly braces to define the body of the function.

```
function robot(x, y, w, h, eyeColor, bodyColor) { ←  
    // do what you need using the parameters  
    // listed above  
}  
←
```

Calling functions

- Your function can now be invoked, called, executed
- (all those words mean the same thing)
- `robot(200, 80, 140, 200, "blue", "lightgray");`



Return values

This is super useful

- In p5.js you often see functions that simply do work, like drawing robots
- But you can always define functions that compute some data:

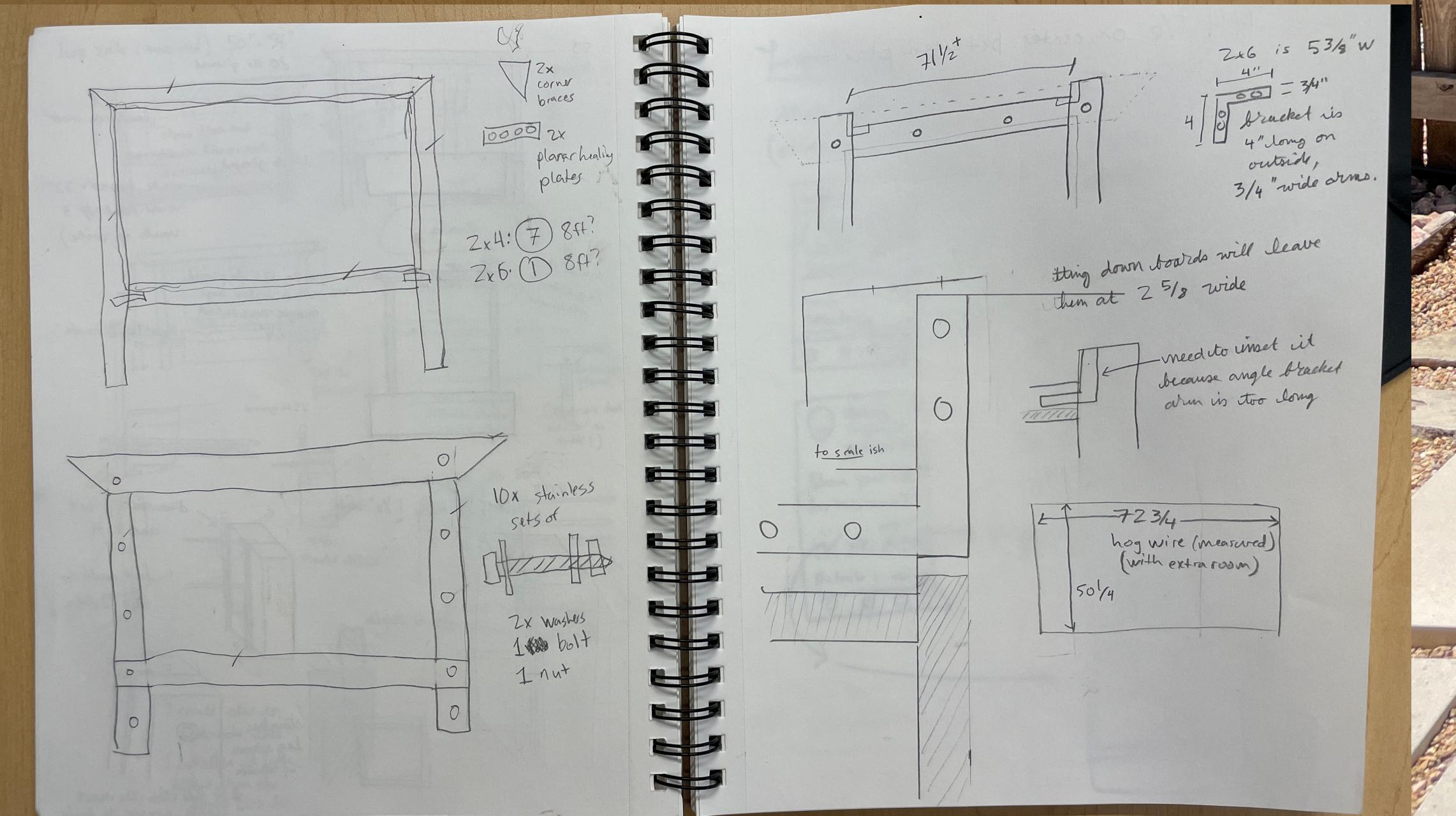
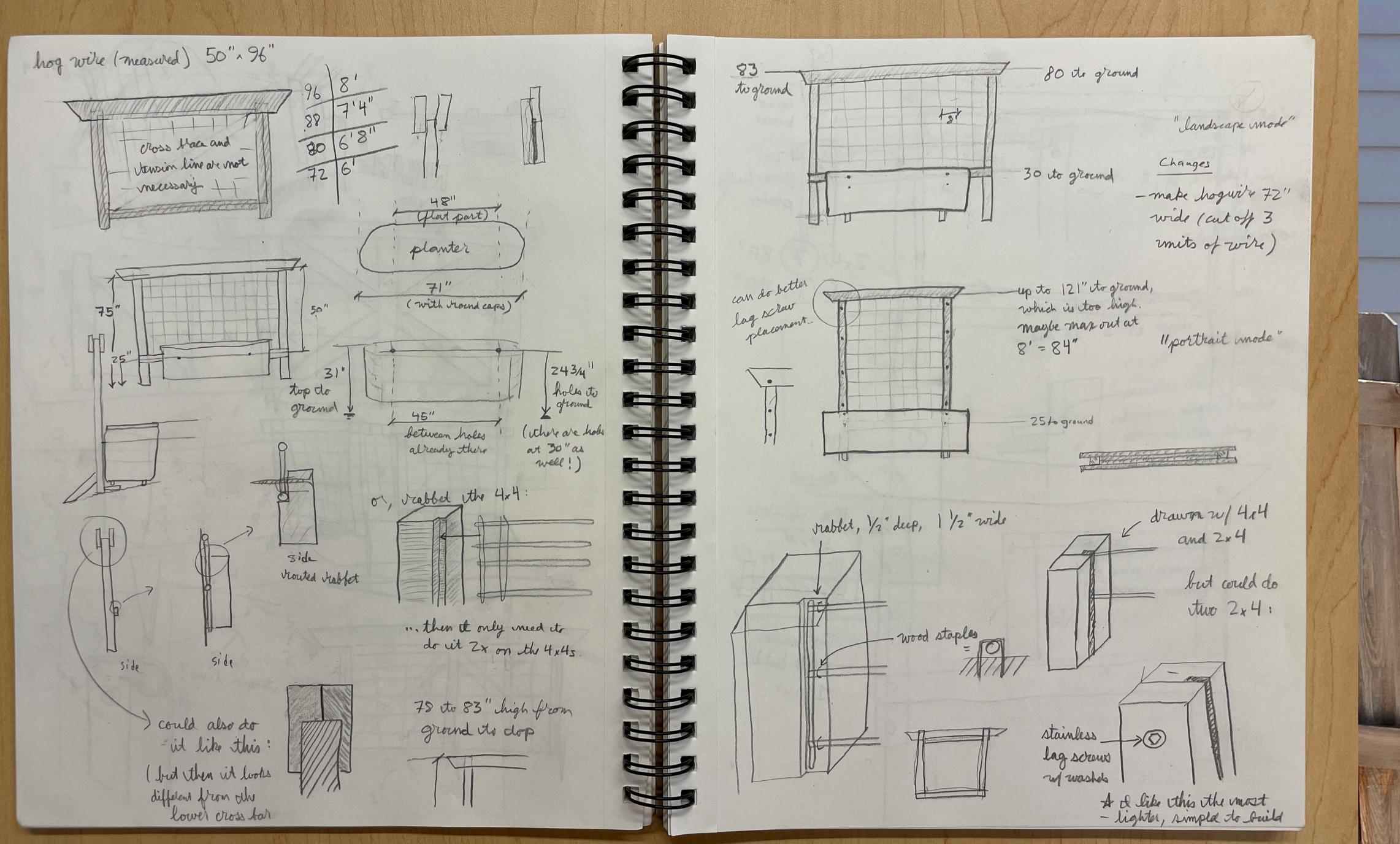
```
function numberCubed(x) {  
    return x * x * x;  
}
```

- and then use it:

```
const v = numberCubed(5); // v will be 125
```

Use Functions Smartly

- I tend to sketch out what I want to help me spot patterns, points of interest, and things that I wouldn't "see" when braining my way through it
- You can decompose problems into subproblems, and those are good candidates for turning into functions
- This works beyond software - if a function is just a process and the parameters are just knobs, you can think of building physical things as just executing a "program" built out of "functions"



Homework

If we have time, live coding to give a head start

- Re-purpose homework from last week (or before)
- Identify spots where you could modularize your code with functions
- Anyone have a good candidate sketch to work out in class?
- If so, email me - gabriel.johnson@colorado.edu