



# Crash Course: Functions 2



ST. MARY'S HIGH SCHOOL



# Overloading Functions

We can make our functions take multiple different types of parameters if we overload it.

We can write another function with the same name and different parameters, so we can use the function many ways.

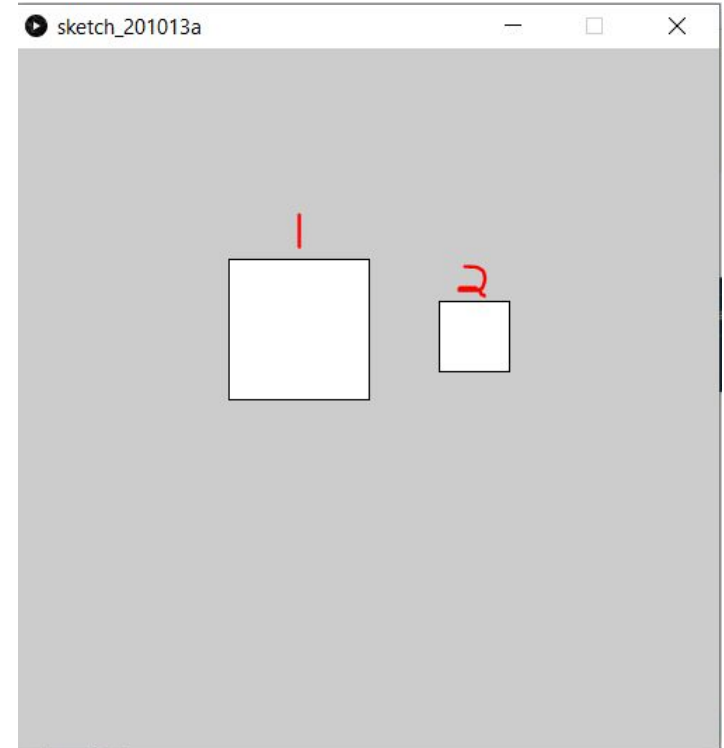
Like giving the function a default value.

```
sketch_201013a
1
2 void drawCube(int x, int y, int size){
3     rect(x, y, size, size);
4 }
5 void drawCube(int x, int y){
6     rect(x, y, 50, 50);
7 }
8
```



The parameters decide which 'drawCube' function we use.

```
1  
2 void drawCube(int x, int y, int size){  
3     rect(x, y, size, size);  
4 }  
5 void drawCube(int x, int y){  
6     rect(x, y, 50, 50);  
7 }  
8  
9 void setup(){  
10    size(500,500);  
11 }  
12  
13 void draw(){  
14    drawCube(150, 150, 100);  
15    drawCube(300, 180);  
16 }  
17  
18  
19
```





# Scope

A variable declared inside a function can only be used inside that function:

'j' is a global variable and can be used anywhere inside the function.

'd' can only be used inside drawCube.

's' can only be used inside setup and

'f' can only be used inside draw.

```
int j = 5;

d | void drawCube(int x, int y){
  |   int d = 5;
  | }

S | void setup(){
  |   int s = 3;
  | }

f | void draw(){
  |   int f = 2;
  | }
```



# Scope Errors

If we try and use 'd' outside of drawCube, we get an error.

This is because 'd' only exists inside drawCube.

```
sketch_201013a  
1 int j = 5;  
2  
3 void drawCube(int x, int y){  
4   int d = 5;  
5 }  
6  
7 void setup(){  
8   int s = 3;  
9  
10  int m = d + s;  
11  
12 }  
13  
14
```

The variable "d" does not exist



# Global Variables

Because we declared 'j' outside of setup or drawCube, we can use 'j' anywhere.

We call these global variables. Use them sparingly to prevent errors.

```
1 int j = 5;
2
3 void drawCube(int x, int y){
4   int d = 5;
5   int f = j + d;
6 }
7
8 void setup(){
9   int s = 3;
10  int m = j + s;
11 }
12
13
14
```



# Why use functions?

As projects get bigger, code gets more complicated and keeping track of variables becomes very difficult.

Using many functions is essential to keeping a project running smoothly.

Not only do functions make your code easier to read and fix, but variable scope can help you keep track of variables.

Using functions help with working in teams. Have different members of the team develop different functions, then bring them all together for the project.



# Code without many functions is long and difficult to understand

```
Pac_Man_Sketch

103   return objX;
104 }
105 //setup the screen size and some variable values.
106 void setup(){
107   size(620, 680);
108   pacX = 310;
109   pacY = 260;
110   inkyX = 310;
111   inkyY = 320;
112   pinkyX = 310;
113   pinkyY = 320;
114   blinkyX = 310;
115   blinkyY = 320;
116   clydeX = 310;
117   clydeY = 320;
118   pacDirac = 1;
119   inkyDirac = 3;
120   pinkyDirac = 3;
121   blinkyDirac = 3;
122   clydeDirac = 3;
123   cTimer = 400;
124   bTimer = 300;
125   pTimer = 200;
126   iTimer = 100;
127   go = 1;
128   for(int i = 0; i <= 63; i++){
129     println("right" + Right[i]);
130     println("left" + Left[i]);
131     println("up" + Up[i]);
132     println("down" + Down[i]);
133     println(i);
134     println(pointX[i] + ", " + pointY[i]);
135   }
```

```
noFill();
strokeWeight(20);
line(30,30, 30, 290);
line(30, 290, 70, 290);
line(70, 290, 70, 470);
line(70, 470, 30, 470);
line(70, 530, 30, 530);
line(30, 530, 30, 650);
line(30, 650, 590, 650);
line(590, 650, 590, 530);
line(590, 530, 550, 530);
line(590, 470, 550, 470);
line(590, 470, 550, 290);
line(550, 290, 590, 290);
line(590, 290, 590, 30);
line(590, 30, 30, 30);

rect(490, 90, 40, 20);
line(430, 30, 430, 170);
line(430, 170, 470, 170);
line(530, 170, 530, 230);
line(530, 230, 490, 230);
rect(90, 90, 40, 20);
line(190, 30, 190, 170);
line(190, 170, 150, 170);
line(90, 170, 90, 230);
line(90, 230, 130, 230);

line(250, 90, 250, 170);
line(250, 90, 370, 90);
line(370, 90, 370, 170);
```

```
Super_pong_vers_4
25 }
26 if(start == 1){
27   paddle1x = 0;
28   paddle2x = 569;
29   paddle1y = 225;
30   paddle2y = 225;
31   start = 2;
32   ballx = 300;
33   bally = 300;
34   timer = 120;
35   go = 0;
36   bd = random(1,15);
37   bd = int(bd);
38   timer2 = 70;
39   boxtimer = 370;
40   activ = 0;
41   lengthpu1 = 0;
42   lengthpu2 = 0;
43   ballspu = 0;
44   bd2 = 0;
45   ballx2 = 0;
46   bally2 = 0;
47   secondball=0;
48   timer3=15;
49   timer4 = 120;
50   ballstart2 = 0;
51   paddlespeed1 = 0;
52   paddlespeed2 = 0;
53   PU = "BRD";
54   text_op = 0;
55 }
56 if(start == 2){
57   background(200);
```





## Using functions helps the code make a lot more sense

```
1  
2  
3 void draw(){  
4  
5     int pos = getPlayerPosition();  
6     checkDamage(pos);  
7  
8     updatePlayerPosition();  
9     checkPlayerAttack();  
10  
11    updateEnemies();  
12    checkLevelUp();  
13    updateScore();  
14  
15    updateGameStatus();  
16  
17 }  
18
```

```
2287 var save = 0;  
2288 setInterval(function(){  
2289     var pack = {map: []};  
2290     for(var i in Map.list){  
2291         pack.map[i] = {  
2292             player: Player.update(i),  
2293             bullet: Bullet.update(i),  
2294             slime: Slime.update(i),  
2295             construct: Construct.update(i),  
2296         }  
2297     }  
2298     save--;  
2299     if(save <= 0){  
2300         Map.respawnAllMonsters();  
2301         save = 1000;  
2302     }  
2303 }  
2304 for(var i in SOCKET_LIST){  
2305     for(var n in Map.list){  
2306         if(Player.list[i] && n == Player.list[i].mapNo){  
2307             var socket = SOCKET_LIST[i];  
2308             socket.emit('init', initPack.map[n]);  
2309             socket.emit('update', pack.map[n]);  
2310             socket.emit('remove', removePack.map[n]);  
2311         }  
2312     }  
2313 }  
2314  
2315  
2316 for(var i in Map.list){  
2317  
2318     initPack.map[i] = {player: [], bullet: [], obstacle: [], slime: [], construct: []};  
2319     removePack.map[i] = {player: [], bullet: [], slime: [], construct: []};  
2320 }  
2321  
2322
```

JavaSc length: 47,849 lines: 2,323 Ln: 2,305 Col: 32 Sel: 0 | 0 Windows (CR LF) UTF-8 INS