ask: **explain a computational idea used previously** in one of your sketches.

week-5 ask: functions are the basic unit of labor in your code … conceive of an entirely new world of labor. what kinds of labor does it take to make your sketch run?

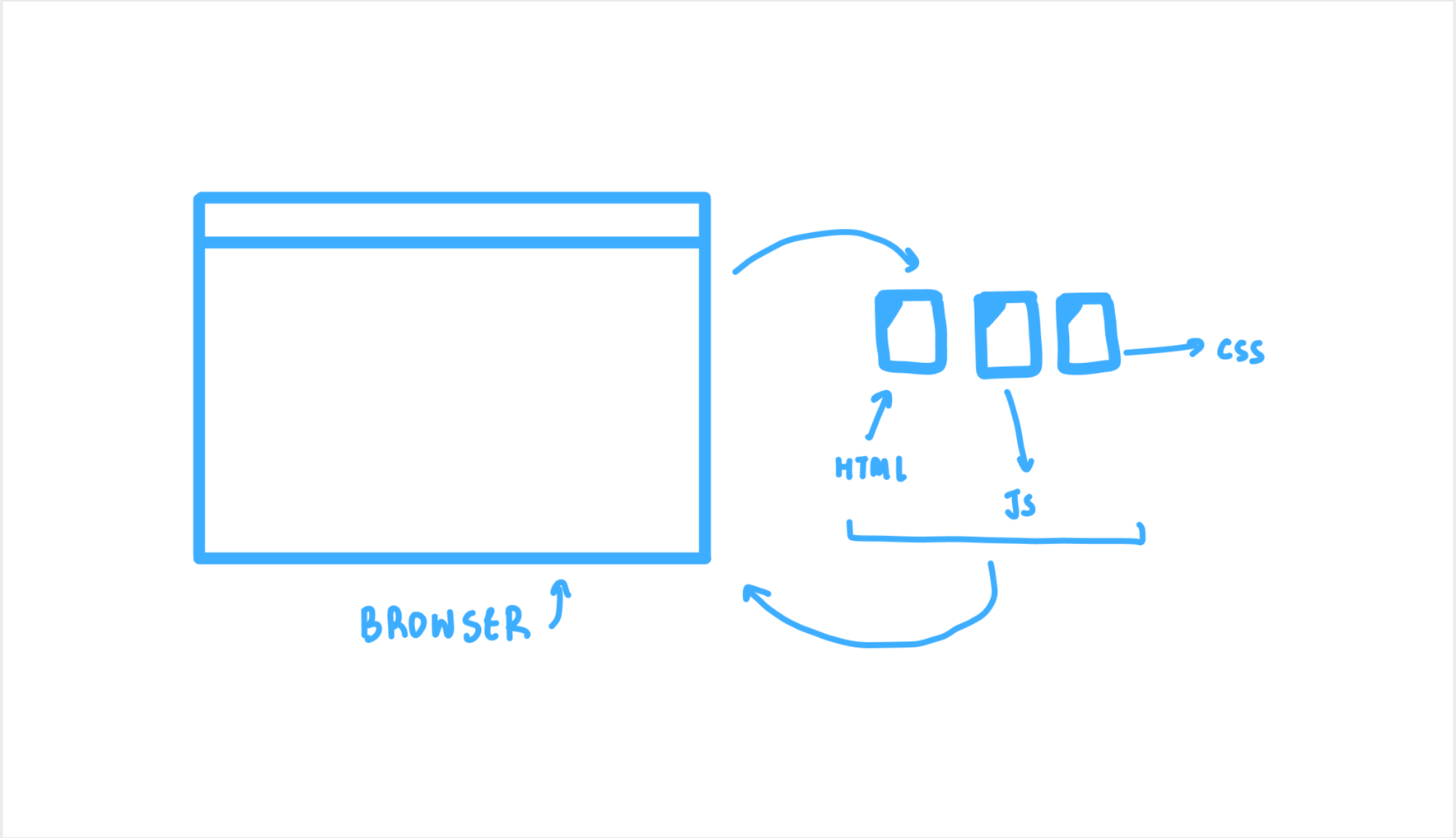
week-5 ask: **functions** are the basic unit of labor in your code … conceive of an entirely new world of labor. what kinds of labor does it take to make your sketch run?

week-5 ask: **functions** are the basic unit of **labor** in your code … conceive of an entirely new world of labor. what kinds of labor does it take to make your sketch run?

week-5 ask: **functions** are the basic unit of **labor** in your code … conceive of an entirely new world of labor. **what kinds of labor does it take to make your sketch run?**

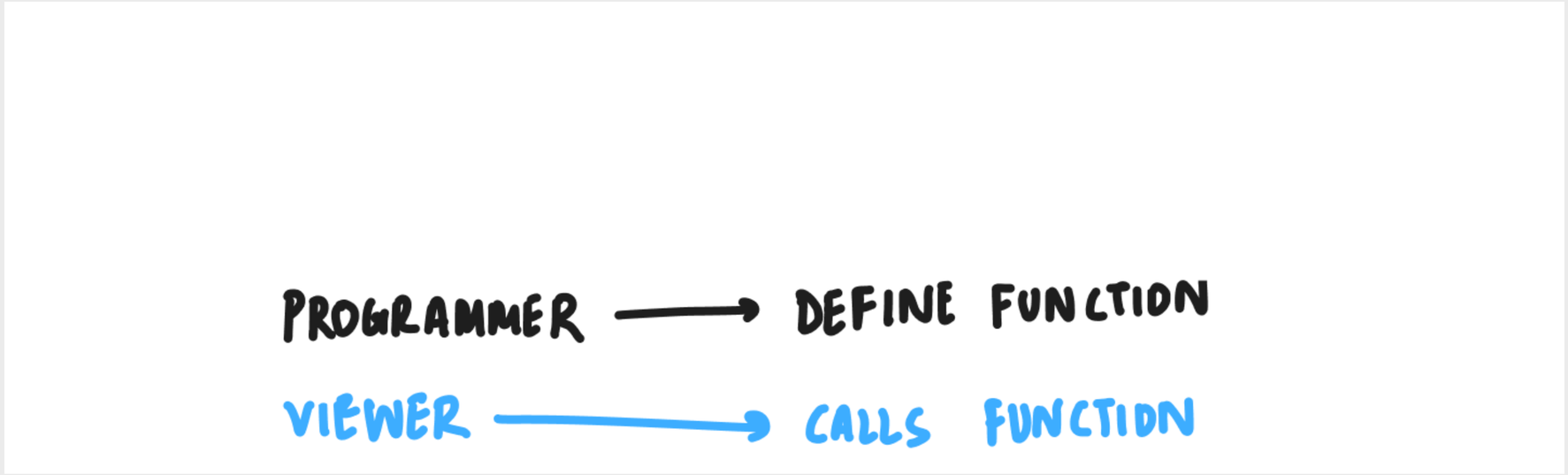
week-5 ask: **functions** are the basic unit of **labor** in your code … conceive of an entirely new world of labor. **what kinds of labor does it take to make your sketch run?**

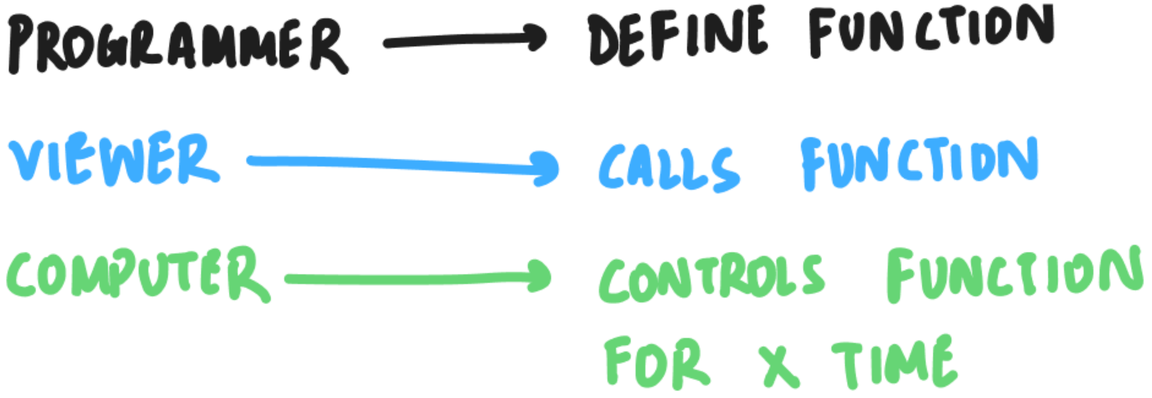
a function that misbehaves, as an act of revolution against the person commanding it.

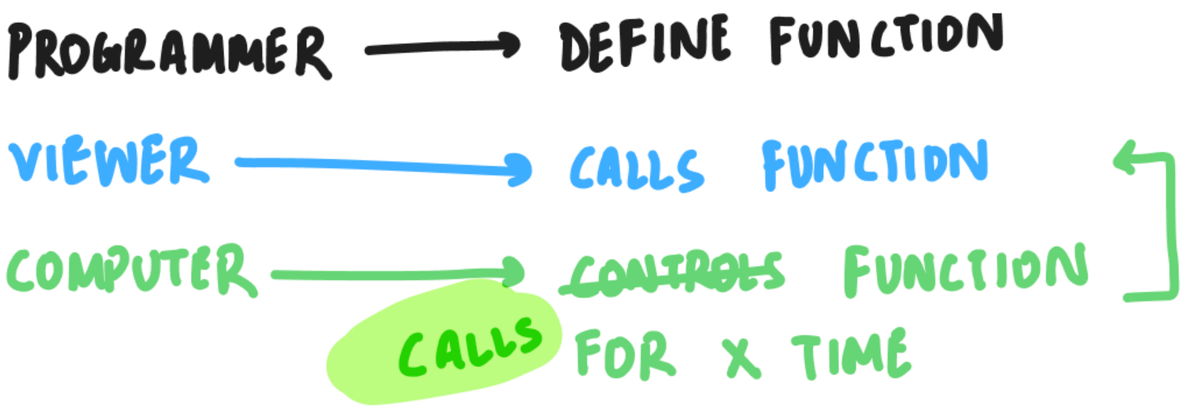


sketch: explaining browser as a runtime environment for javascript.

PROGRAMMER ⟶ DEFINE FUNCTION


PROGRAMMER
DEFINE FUNCTION
VIEWER
CALLS FUNCTION


PROGRAMMER
DEFINE FUNCTION
VIEWER
CALLS FUNCTION
COMPUTER
CONTROLS FUNCTION
FOR X TIME



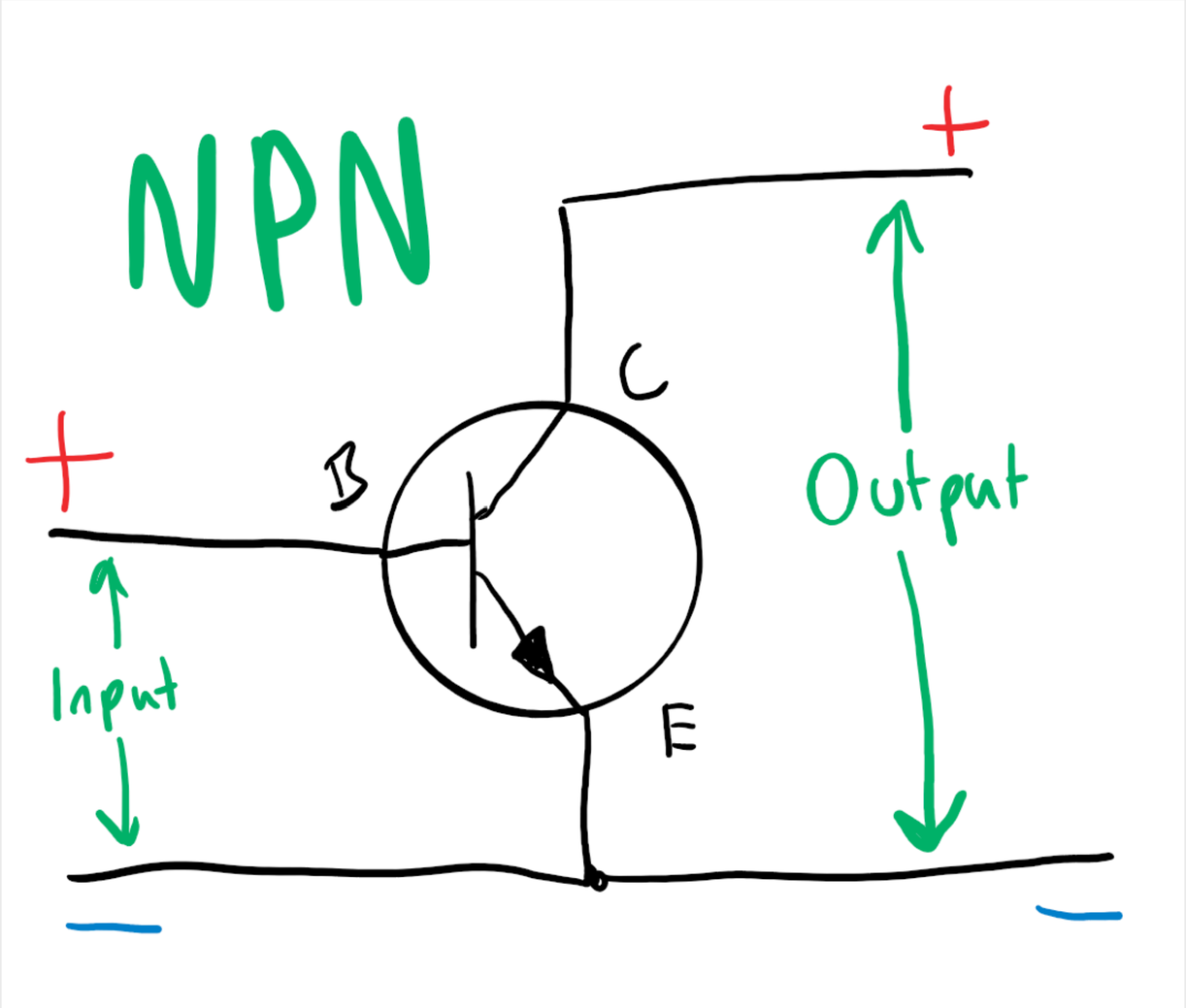
sketch: explaining browser as a runtime environment for javascript.

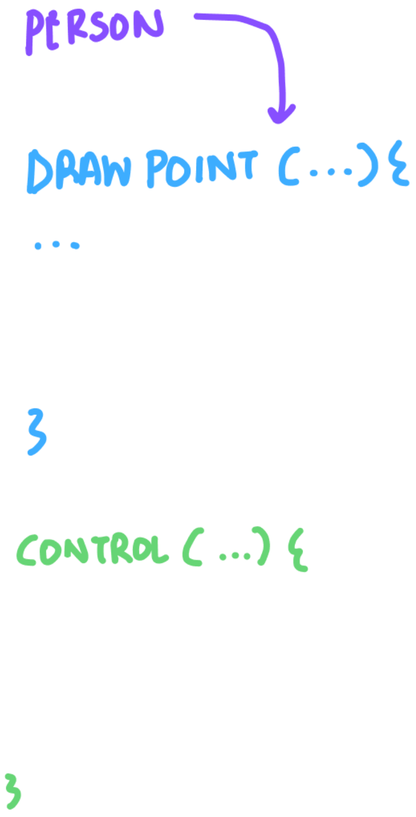
```
let sw = 5; //variable for stroke_weight.

function draw_point(x, y) {
  //do what you're told.
  strokeWeight(sw);
  stroke(255);
  point(x, y);

  //then, give over control to the computer.
    control(x, y);

  return "oops"; //return this in the console instead of undefined.
}

let min_run = 5;
let max_run = 100;

function control(x, y) {
  let times_to_run = int(random(1, 1000));

  for (let i = 0; i < times_to_run; i++) {
    //syntax: setTimeout(function, delay, arg1, arg2, ...); (from: https://www.geeksforgeeks.org/javascript/
    how-to-delay-a-function-call-in-javascript/)
    let new_x = (x += random(-sw, sw));
    let new_y = (y += random(-sw, sw));
      draw_point(new_x, new_y);
  }
}
```

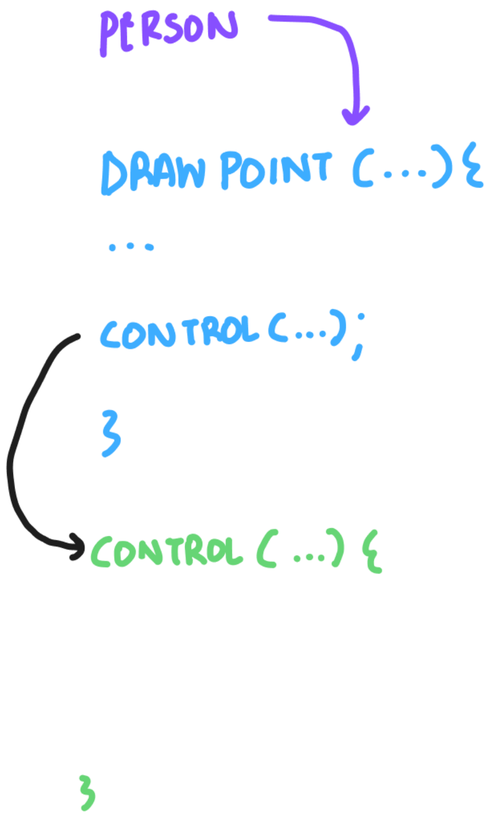
code snippet to explain recursion.

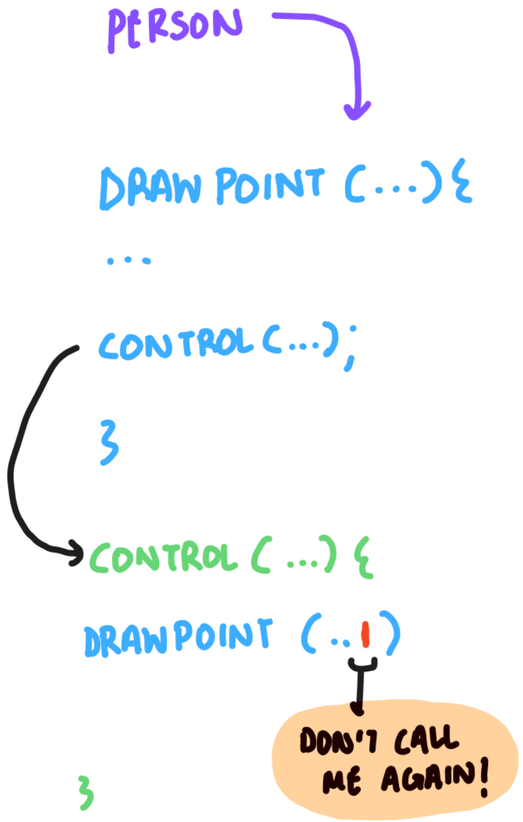
```
let sw = 5; //variable for stroke_weight.

function draw_point(x, y) {
  //do what you're told.
  strokeWeight(sw);
  stroke(255);
  point(x, y);

  //then, give over control to the computer.
    control(x, y);

  return "oops"; //return this in the console instead of undefined.
}
```

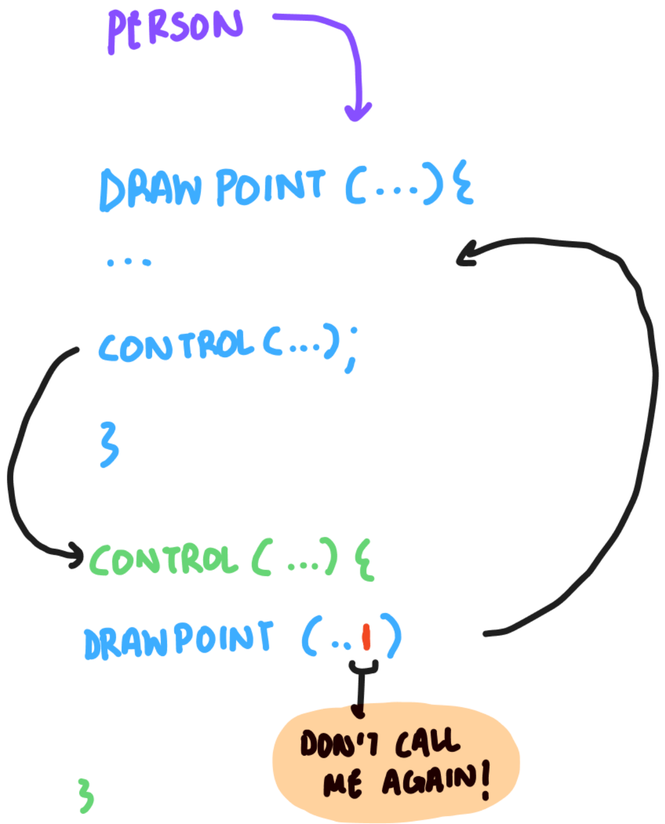
```
let sw = 5; //variable for stroke_weight.

function draw_point(x, y) {
  //do what you're told.
  strokeWeight(sw);
  stroke(255);
  point(x, y);

  //then, give over control to the computer.
    control(x, y);

  return "oops"; //return this in the console instead of undefined.
}

let min_run = 5;
let max_run = 100;

function control(x, y) {
  let times_to_run = int(random(1, 1000));

  for (let i = 0; i < times_to_run; i++) {
    //syntax: setTimeout(function, delay, arg1, arg2, ...); (from: https://www.geeksforgeeks.org/javascript/
    how-to-delay-a-function-call-in-javascript/)
    let new_x = (x += random(-sw, sw));
    let new_y = (y += random(-sw, sw));
      draw_point(new_x, new_y);
  }
}
```

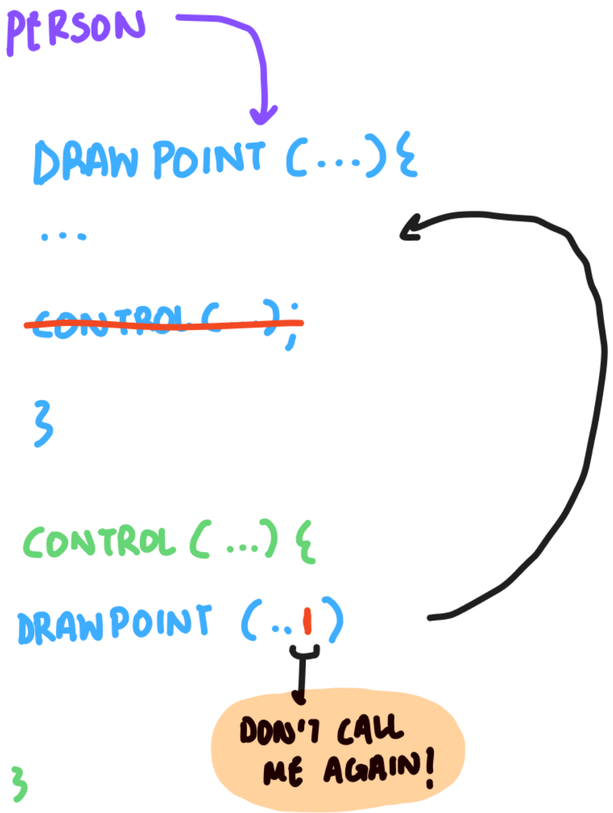

NPN
+
C
+
B
Output
Input
E
−
−


PERSON
DRAW POINT (...){
...
}
CONTROL (...) {
}


PERSON
DRAW POINT (...){
...
CONTROL(...);
}
CONTROL(...){
}


PERSON
DRAW POINT (...){
...
CONTROL(...);
}
CONTROL (...) {
DRAWPOINT (..!)
DON'T CALL ME AGAIN!
}


PERSON
DRAW POINT (...){
...
CONTROL(...);
}
CONTROL (...) {
DRAWPOINT (..!)
DON'T CALL ME AGAIN!
}


PERSON
DRAW POINT (...) {
...
CONTROL ( );
}
CONTROL (...) {
DRAWPOINT (..)
DON'T CALL ME AGAIN!
}