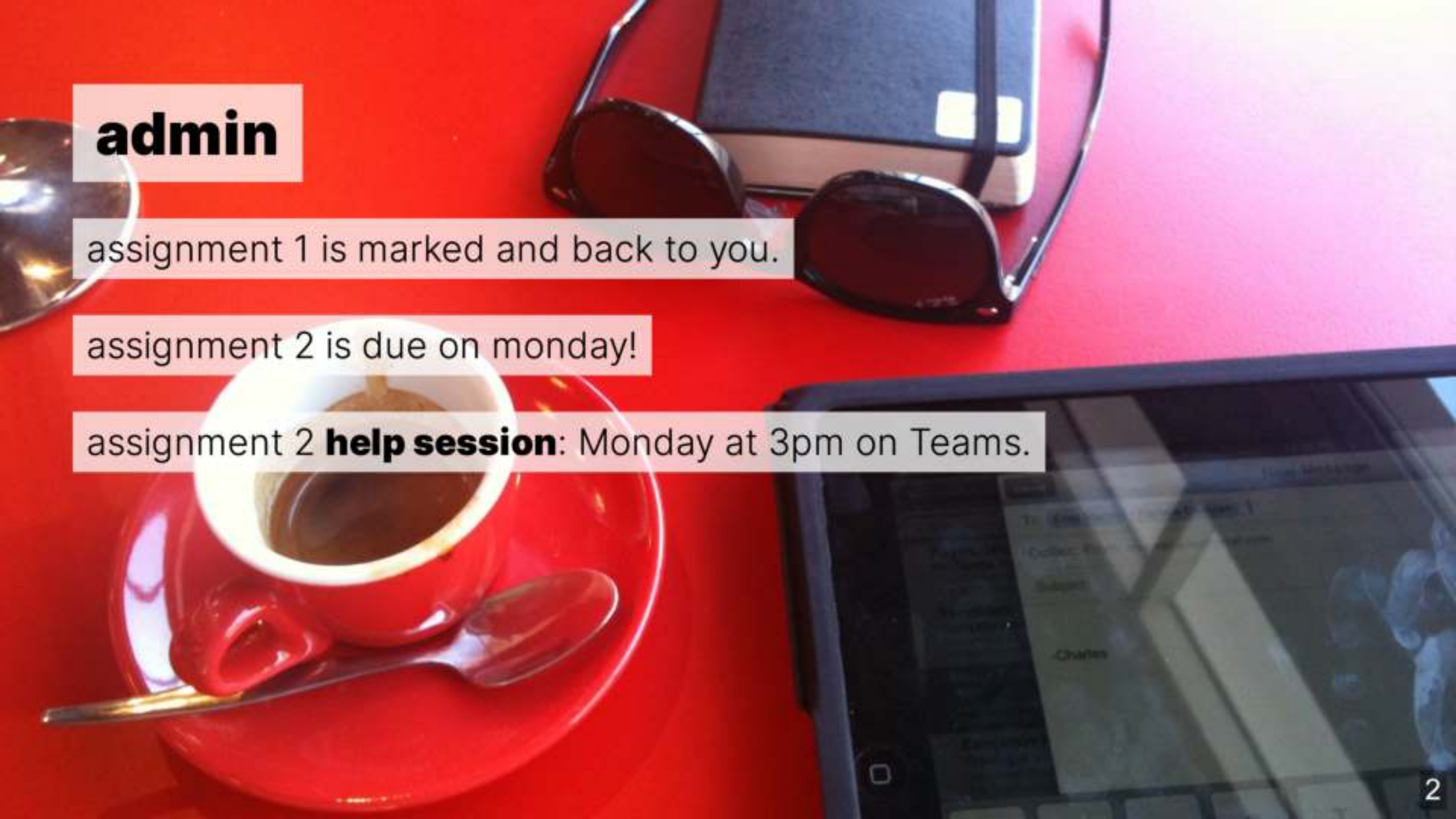
COMP1720

Art & Interaction in New Media

Week 6: code concepts revision

Dr Charles Martin

Semester 2, 2020



admin

assignment 1 is marked and back to you.

assignment 2 is due on monday!

assignment 2 **help session**: Monday at 3pm on Teams.

Course Survey Results

61 responses! Great!

- COMP1720: 41 responses
- COMP6720: 20 responses

average time to complete: **1min 7secs**

anonymous to the lecturers and tutors

Course Quality Lectures

Course Quality Labs

3. How would you rate the overall quality of the lab sessions and tasl

More Details

Very poor	1
Poor	5
Average	17
Good	27
Excellent	11

Course Quality Assessment and VDs

4. How would you rate the overall quality of the visual diary and assig

More Details

Very poor	0
Poor	5
Average	15
Good	33
Excellent	8

Feedback: Good!

- pair programming
- lectures are comprehensive and enjoyable
- Charles' clear pronounciation (!)
- Tony "wonderful lecturer"
- tutors are very responsive
- Youtube Channel!
- vd
- labs are interactive
- great atmospshere

Feedback: Not so good!

- Wednesday overtime: can’t stay due to classes?
- Some details in the lectures brushed over, need details to help with math
- Labs: too many tasks, too complicated, what are the deliverables?
- Would prefer whole lab to be live.
- A lot less sharing of what we’ve made
- Visual diary: takes a long time, hard to translate into p5.js?
- Connection between lab marks and vd marks

Feedback: OMG!

“People **leave labs**, can’t find anyone to **live code with me**...”

Struggle with assignments. Not sure **where to get help**?

Can’t

keep

up...

What we will do

- continue updating labs to make most sense for online teaching
- offer more scaffolding exercises in the labs
- continue tuning VD procedures
- keep you accountable for **regular engagement**

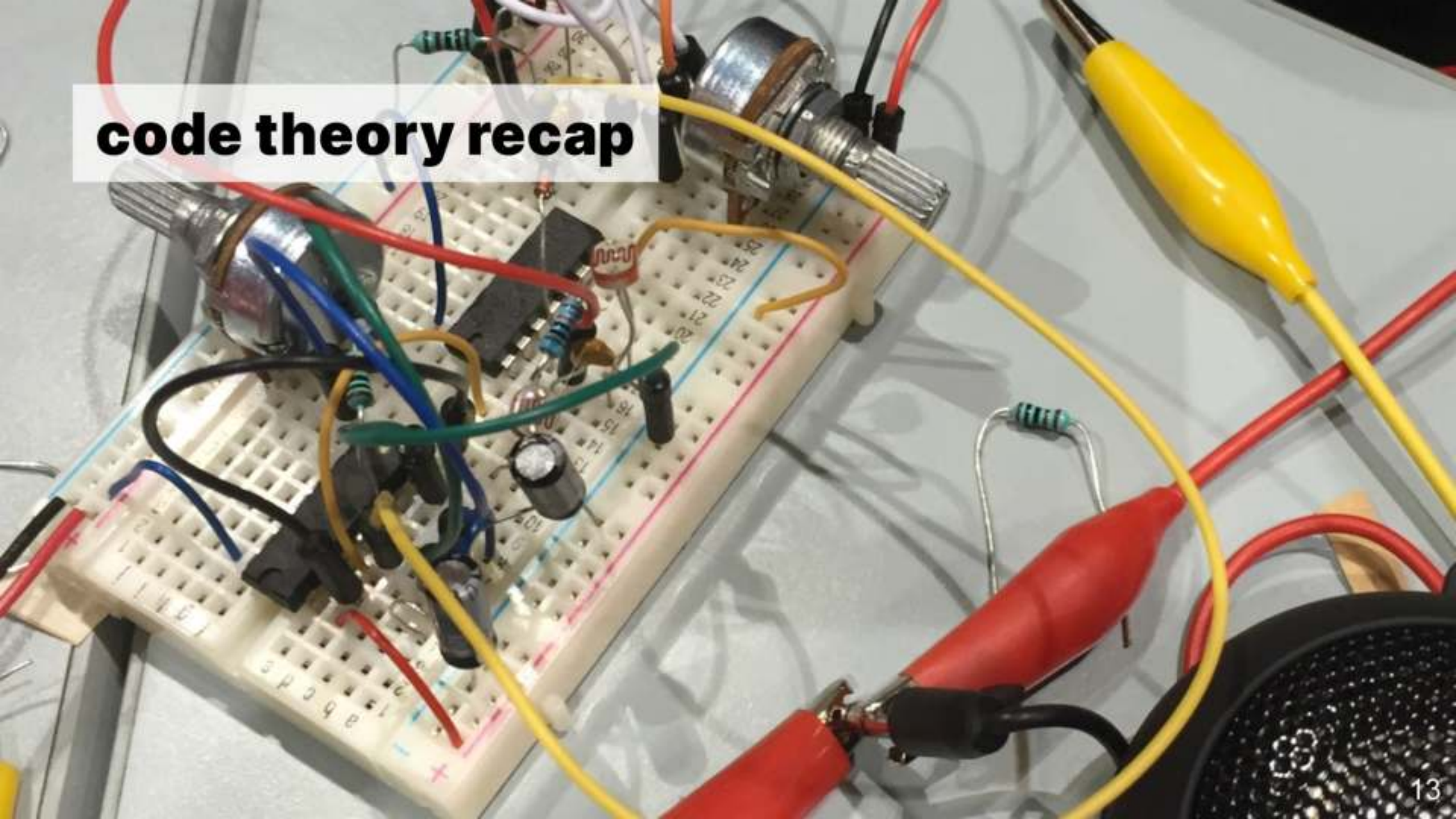
What you can do

- **ask questions** in the forums
- include something **you have made** in p5 in each VD
- **participate in labs** as much as possible; focus on pairs and **peer-learning**
- **share your work** in labs and on the forum
- take ownership over your learning (fill gaps with **videos** and **books**)
- **ask for help**

small personal note...

Charles will be on leave for a bit sometime soon...

code theory recap




painting like a 5yo

first words

- `background()`
- `rect()`
- `ellipse()`
- `fill()`
- `stroke()`
- `random()`

...and more in the **reference**

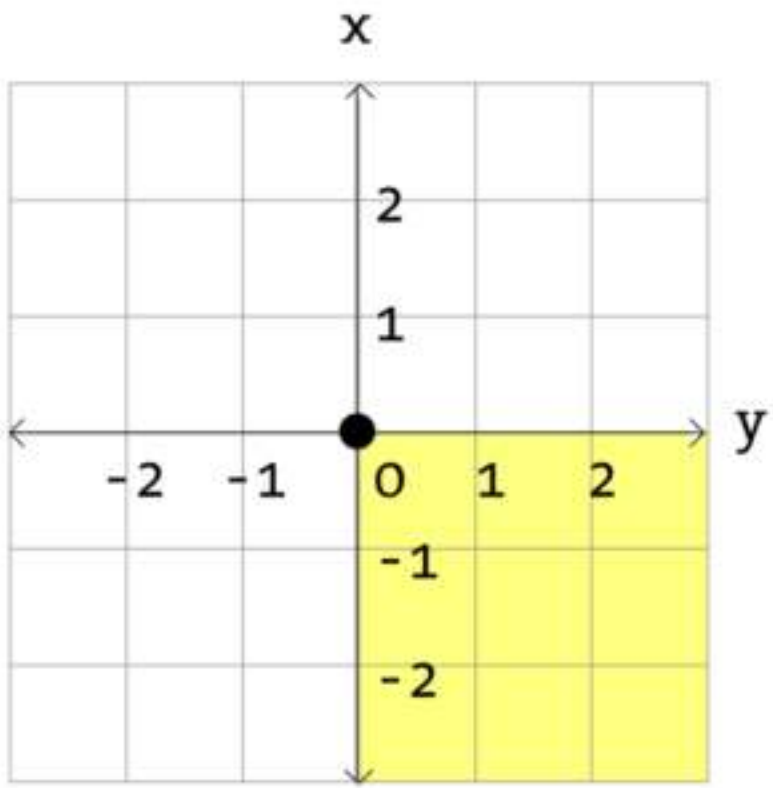
you have to be very specific

computers can't deal with vagueness—it can't figure out what you **mean** if you don't say it clearly

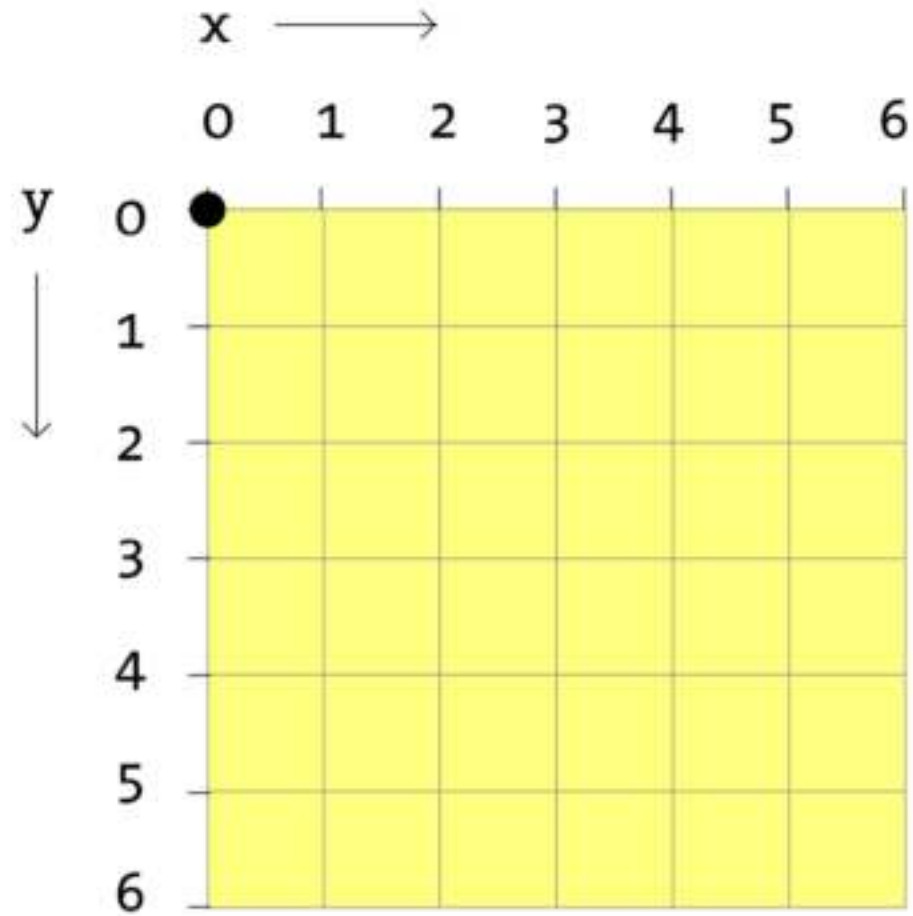
talking to p5 about colour `fill(red, green, blue)`

talking to p5 about position `ellipse(x, y, width, height)`

the grid



Eighth Grade



Computer

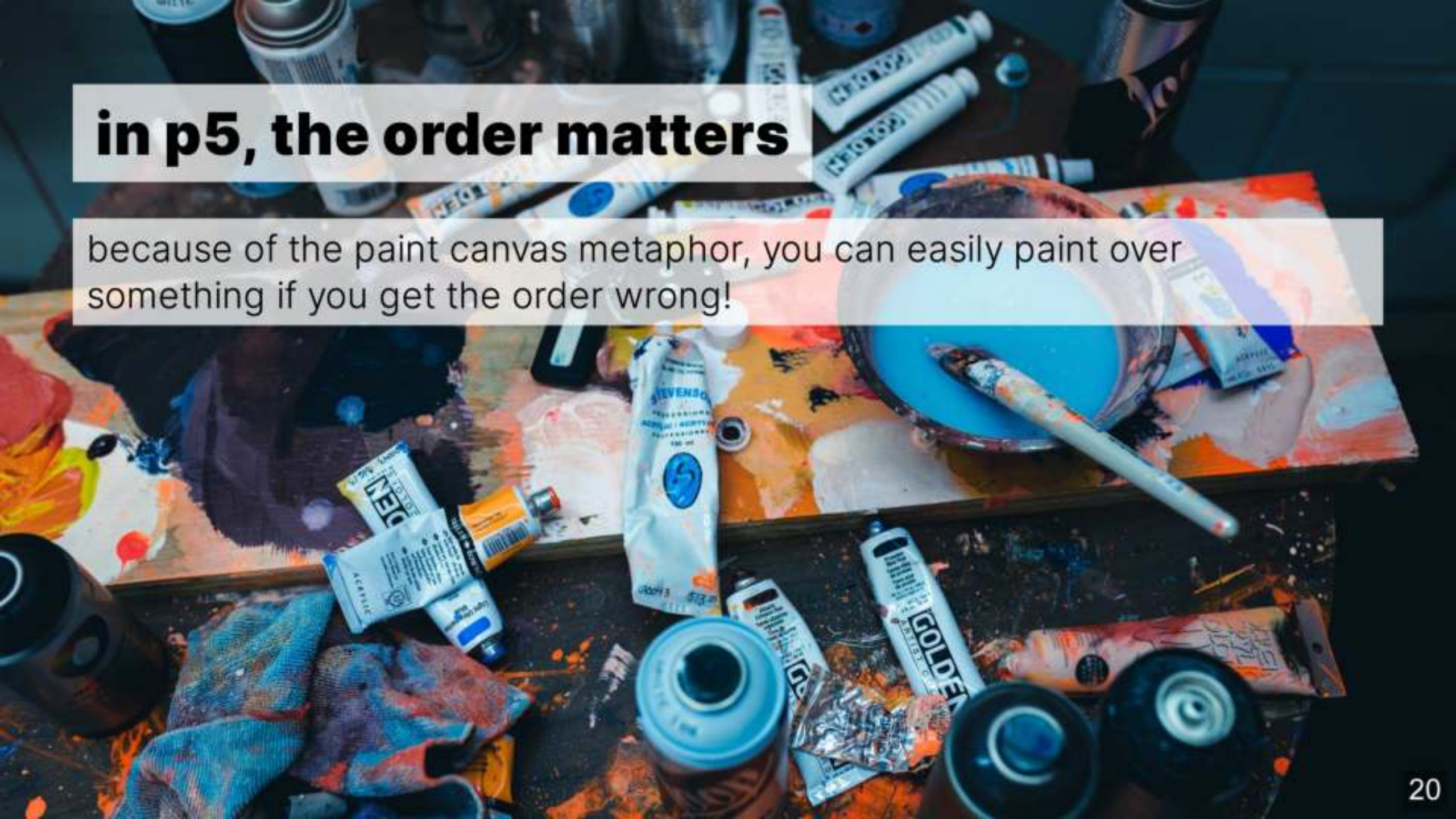
how does the computer read?

```
function draw() {
    background(40);
    ellipse(10, 10, 20, 20);
    drawStuff();
    ellipse(310, 100, 200, 20);
}

function drawStuff() {
    fill(255,0,127);
    rect(100, 100, 20, 20);
    fill(0,127, 255);
    rect(300, 200, 200, 120);
}
```

flow

setup, draw, draw, draw, draw...



in p5, the order matters

because of the paint canvas metaphor, you can easily paint over something if you get the order wrong!

Important!

At this point you have all the tools you need to draw **anything** in p5!

Everything else is for making **structured** programs to save you time, save you typing, and to be **interactive** and **dynamic**!

Variables

a stable *name* for a value (which might change)

```
//   name          value
var dogeCount = 15;
var halfRange = width/2;
var randomValue = random(13);
```

To use variables:

1. **declare**: `var age;` means *there's a variable called "age"*
2. **initialise**: `age = 34` means *set the age variable to the number 34*

Special Variables

```
mouseX
mouseY
mouseIsPressed
width
height
```

These are declared and initialised by p5.js, but are for **you** to use!

What can we do now?

We can set out sketches with **simple maths**.

And we can start to be **interactive!**

representing truth

one little bit of maths...

the boolean type: `true` and `false`.

`&&` (and) `||` (or) `!` (not)

- `<` (less than)
- `<=` (less than **or** equal to)
- `>` (greater than)
- `>=` (greater than **or** equal to)
- `==` (equal to, note the double equals sign)

conditionals and loops

syntax recap: conditionals

```
var the_sky_is_blue = false;
if (the_sky_is_blue) {
    ellipse(50,50,100,100);
} else {
    rect(50,50,100,100);
}
```

```
if (frameCount > 60) {
    text("you lose!");
} else {
    text("wow, you're so good at this!");
}
```

syntax recap: loops

```
var i = 0;
while (i < 10) {
 // loop code goes here
 i=i+1;
}
```

```
for (var i = 0; i < 10; i=i+1) {
 // loop code goes here
}
```

What can we do?

- **dynamic** sketches
- make **decisions** in our code
- **repeat code** to save typing!


functions

syntax recap: functions

```
// name <---parameters--->
   rect(100, 100, 100, 100);
```

writing your own functions

```
function polkadot(x, y){
  fill(255,0,0);
  ellipse(x, y, 20, 20);
}
```

```
function double(x) {
  return x * 2;
}
```


arrays

syntax recap: arrays

```
var arrayOfNumbers = [100, 24, -2];
var arrayOfStrings = ["John", "Paul", "George", "Ringo"];
var arrayOfBooleans = [true, false, true, true, false];
var emptyArray = [];

// arrays in arrays
var points = [[0, 5], [50, 100], [0, 200]];
```

the **Array reference is on MDN**, but several p5 functions use arrays as well

working with arrays

```
var arr = [1, 2, 3, 4, 5];

arr[1] = 42 // change the first element

arr.push(9000) // add to the end of an array
arr.unshift(0) // add to the start of an array

arr.pop() // remove from the end of an array
arr.shift() // remove from the start of an array

arr.length // get the length of the array
```


YO DAWG, I HEARD YOU LIKE ARRAYS
SO I PUT AN ARRAY IN YOUR ARRAY

looping over arrays

```
var points = [[0, 5], [50, 100], [0, 200]];

for(var i = 0; i < points.length; i = i+1){
  var x = points[i][0];
  var y = points[i][1];
  // do stuff here
}
```

objects

objects

objects: a bunch of properties, each of which has a *name* and a *value*

a useful way of grouping related bits of data together

syntax recap: objects

```
var sally = {
  species: "Pikachu",
  level: 1,
  hp: 100,
  owner: "Ash",
  captured: true
}
```

get/set property values

```
// get the current value
sally.species

// set a new value
sally.species = "Raichu"
```

remember that `sally.species` and `sally["species"]` **are the same**

arrays, or objects?

sometimes we want arrays, and sometimes we want objects

- use an **object** when you have a bunch of related values with *different* types
- use an **array** when you have a bunch of values with the *same* type


don’t be afraid to have an array of objects (and vice versa)

p5 is making objects (under the bonnet)

for example the `color()` function

```
var magenta = color(220,0,220);
console.log(magenta);
```

talk

what do you think the `bubble` object might look like?

```
function drawBubble(bubble) {
  if(bubble.popped){
    fill(100);
  }else{
    fill(20, 50, 250, 150);
  }
  ellipse(bubble.x, bubble.y, 50, 50);
}
```

looping over arrays of objects

```
// loop over all the bubbles
for(var i = 0; i < bubbles.length; i = i+1){
  popIt(bubbles[i]);
  drawBubble(bubbles[i]);
}
```


if it's on fire, don't panic
and check the console

further reading/watching

check out the sketches we've done together **(link)** and extend them!

play **p5 reference golf** (pick a few random p5 functions and see if you can use them to make something cool)

read this book: **Make: Getting Started with p5.js**

MDN javascript docs (**tutorials**, **reference**)


questions?