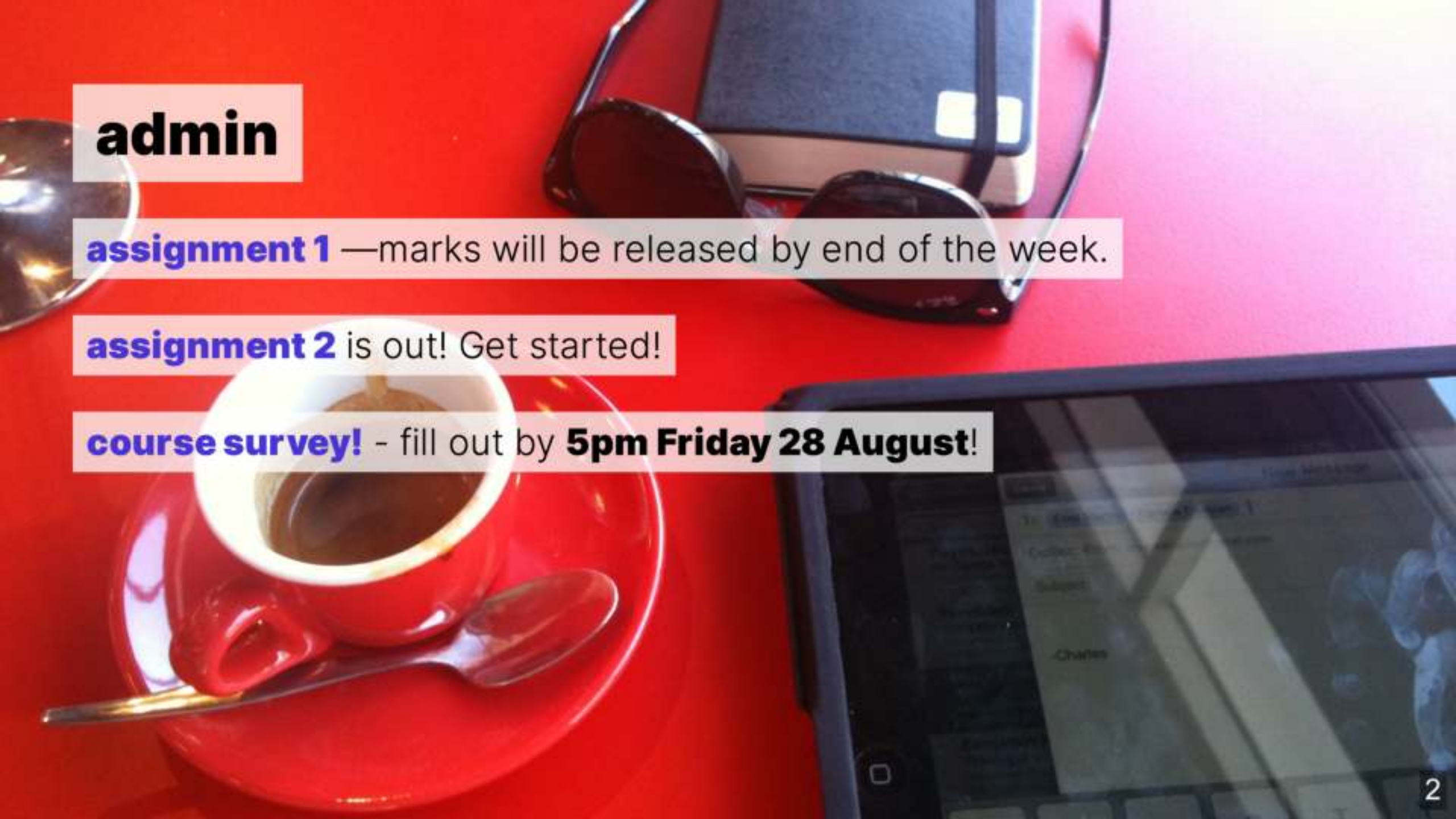
COMP1720

Art & Interaction in New Media

Week 5: objects

Dr Charles Martin

Semester 2, 2020



admin

assignment 1 —marks will be released by end of the week.

assignment 2 is out! Get started!

course survey! - fill out by **5pm Friday 28 August**!

recap

functions: packaging up a bunch of **operations**

arrays: packaging up a bunch of **values**

syntax recap: functions

```
function drawEye(x, y, eyeSize){
  noStroke();
  // first, the whites of the eye
  fill(255);
  ellipse(x, y, eyeSize*4, eyeSize*2);
  // now the pupils
  fill(0);
  ellipse(x, y, eyeSize, eyeSize);
}
```

syntax recap: functions

```
function biggerThanFour(value){
  return value > 4; // return a boolean value
}
```

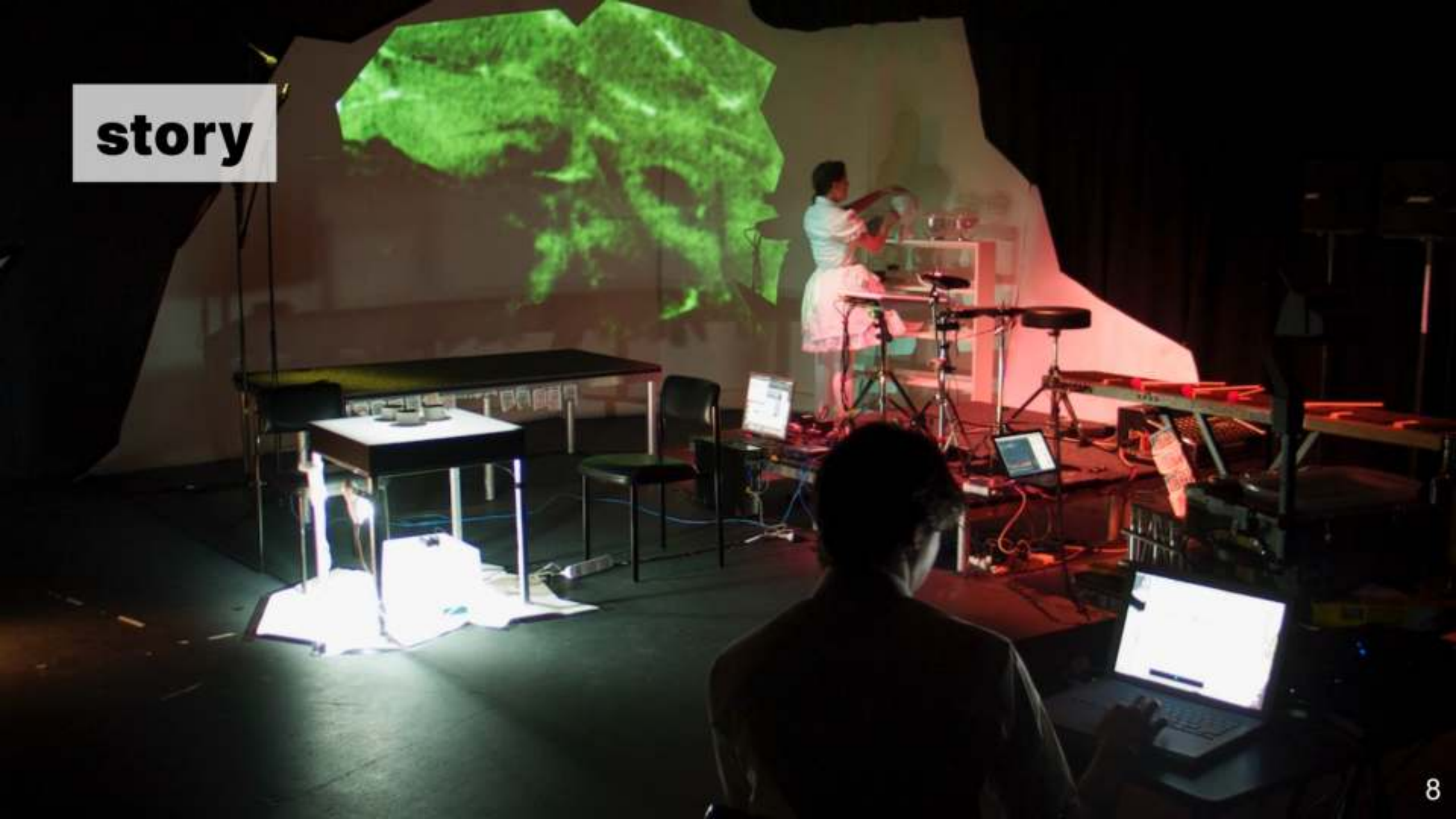
syntax recap: arrays

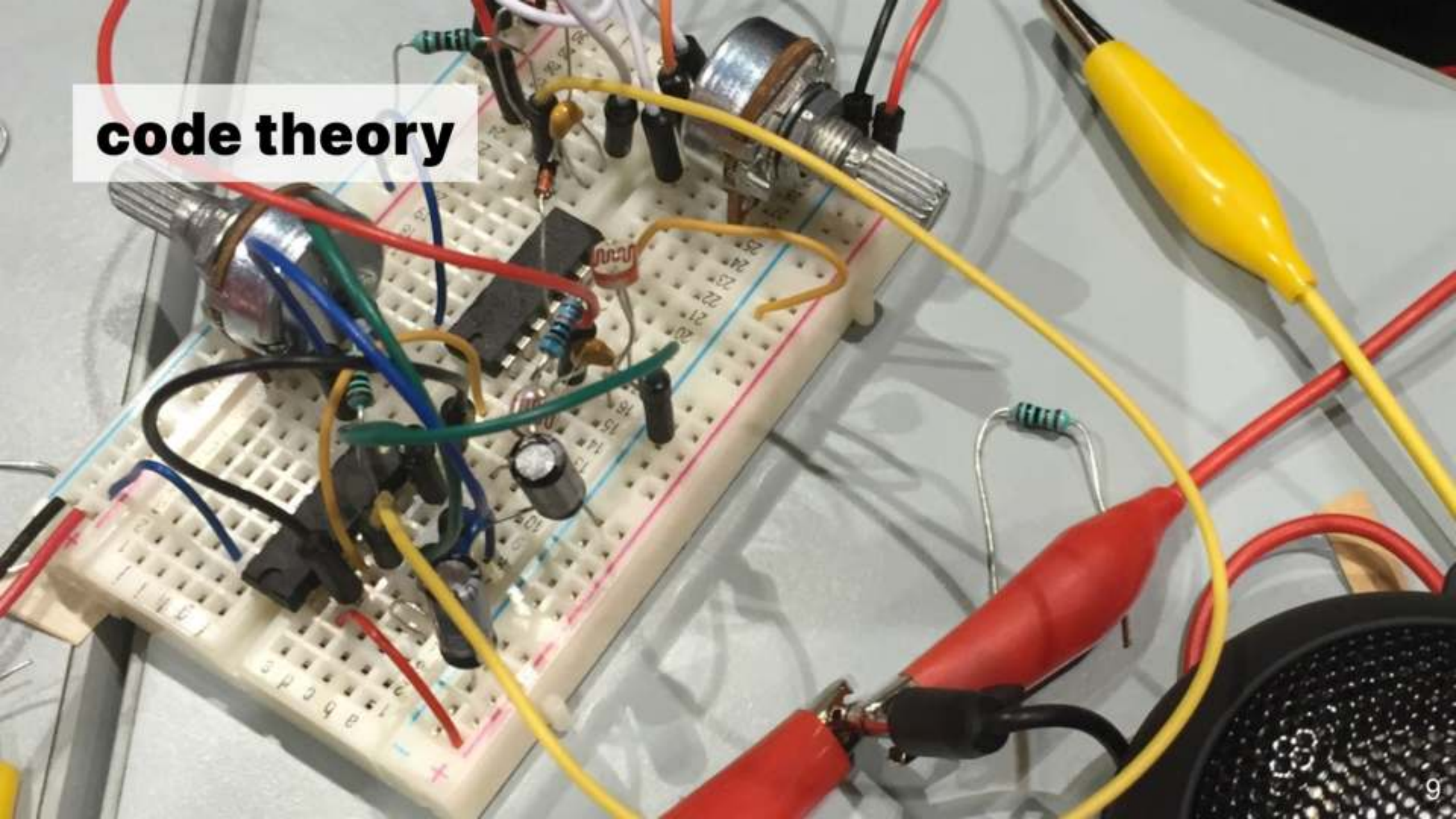
```
var xValues = [5, 15, 55]; // declare & initialise

// get stuff out: use the *first* value in
// the array as a parameter for random()
random(xValues[0]);

// change the value in the 3rd "slot" in the array
// remember that the index starts at zero!
xValues[2] = 544;

// add the number 75 to the end of the array
xValues.push(75);
```

recap: adding & removing things from arrays

```
var things = [1, 2, 3];
```

	add	remove
front	`things.unshift(value)`	`things.shift()`
back	`things.push(value)`	`things.pop()`


story


code theory


what's an object?

talk

name a thing that *exists*

and how you would describe it to someone who has never seen it before?

motivation

often, you need multiple “attributes” to fully describe a thing

- a **student**: name, age, uni course
- an **address**: street name, street number, perhaps unit number?
- a **pet**: name, species, breed, owner

javascript has **Objects** to keep related bits of data together

some definitions

> **object** (*noun*): something mental or physical toward which thought, feeling, or action is directed **https://www.merriam-webster.com/dictionary/object**
>
> **object** (*noun*): a data structure in object-oriented programming that can contain functions as well as data, variables, and other data structures **https://www.merriam-webster.com/dictionary/object**


let's make a Pokemon

talk

what properties does a Pokemon have?

species

level

hit points

owner

captured

as a javascript Object

```
// declare and initialise in one go
var sally = {
  species: "Pikachu",
  level: 1,
  hp: 100,
  owner: "Ash",
  captured: true
};
```

object vocabulary

the components of an object are called **properties**, each property has a *name* and a *value*

```
var sally = {
// prop name: value
          hp: 42
};
```

try not to get confused between the name of the *object* (`sally` in this case) and the name of its *properties* (just one property in this case: `hp`)

more jargon

in everyday language, we've got a bunch of "interchangeable" terms:
property, attribute, characteristic, trait, aspect

but in programming they mean different things! 🤷

so, be careful, and remember that **objects** have **properties**

object syntax

for an object we use the `{` and `}` squiggly braces

the spaces don't matter, only the braces

```
// these are all exactly the same
var sally = {hp: 50};

var sally = { hp: 50 };

var sally = {
  hp: 50
};
```

designer data

let's return to some of the things you talked about earlier—can we represent them as javascript objects?

what can you do with an object?

just like with arrays, there are a few main things you want to do with your object

- **get** the value of a property (this is often called *accessing* a property)
- **set/update** the value of a property
- add a **new** property & value (sometimes)

getting the value of a property

there are two main ways to access (get) the value of a property

```
sally.species     // "Pikachu"
sally["species"] // "Pikachu"
```

both the “dot” version and the “square brackets” version are equivalent—they return the same value

when to use the dot (.) syntax

pros: a *bit* easier to type

cons: doesn't work if the property name isn't a **String**

```
// here, the property name is actually a number
var benObject = {5: "howdy"};

benObject.5  // Uncaught SyntaxError: Unexpected number
benObject[5] // "howdy"
```

setting the value of a property

just like with arrays, you can assign a new value to a property with `=`

these two lines of code are exactly equivalent

```
sally.species = "Raichu"
```

```
sally["species"] = "Raichu"
```

updating the value of a property

there's even a syntax for doing the *get* and *set* all in one line

these two lines of code are exactly equivalent

```
sally.hp = sally.hp - 10;
```

```
sally.hp -= 10; // a bit quicker to type
```

adding new properties to an object

if a property with that name doesn't exist, you can create it by just assigning a new value to that property name

these two lines of code are exactly equivalent

```
sally.hat = true;
```

```
sally["hat"] = true;
```

using the new property

then, you can use it in your code from that point on

```
if(sally.hat){
   // draw a hat, or something
}
```

how can things go wrong?

what happens if we try to access (get) a property which the object doesn't have?

everyone knows that Pikachu don't wear pants (**except when they do**)

```
sally.pants // undefined
```

your program might not break *completely* (`undefined` is an actual thing in javascript) but it probably won't do what you want it to

why should I care?

p5 is making objects (under the bonnet)

for example the `color()` function

```
var magenta = color(220,0,220);
print(magenta);
```

moar p5 objects

or `loadImage()`, or `createVector()`, or...

```
var img = loadImage("assets/kangaroo.jpg");
print(img);
```

```
var position = createVector(40, 52);
print(position);
```

nested object examples

```
// nested objects
var car = {
  make: "toyota",
  paint: {
    colour: "red",
    metallic: true
  }};

car.paint.colour; // what's the value?
```

arrays and objects together!

```
// array of objects
var myAnts = [{species: "fire", size: 100},
              {species: "giant", size: 500}]

// object of arrays
var pokerHand = {hearts: [2, 5, 7],
                 diamonds: [8, "Q", "K"],
                 clubs: [5, 6],
                 spades: []}

pokerHand.clubs[1] // what's the value?
```

objects vs arrays

how does this relate to the arrays stuff from last week? a lot of this stuff looks really familiar...

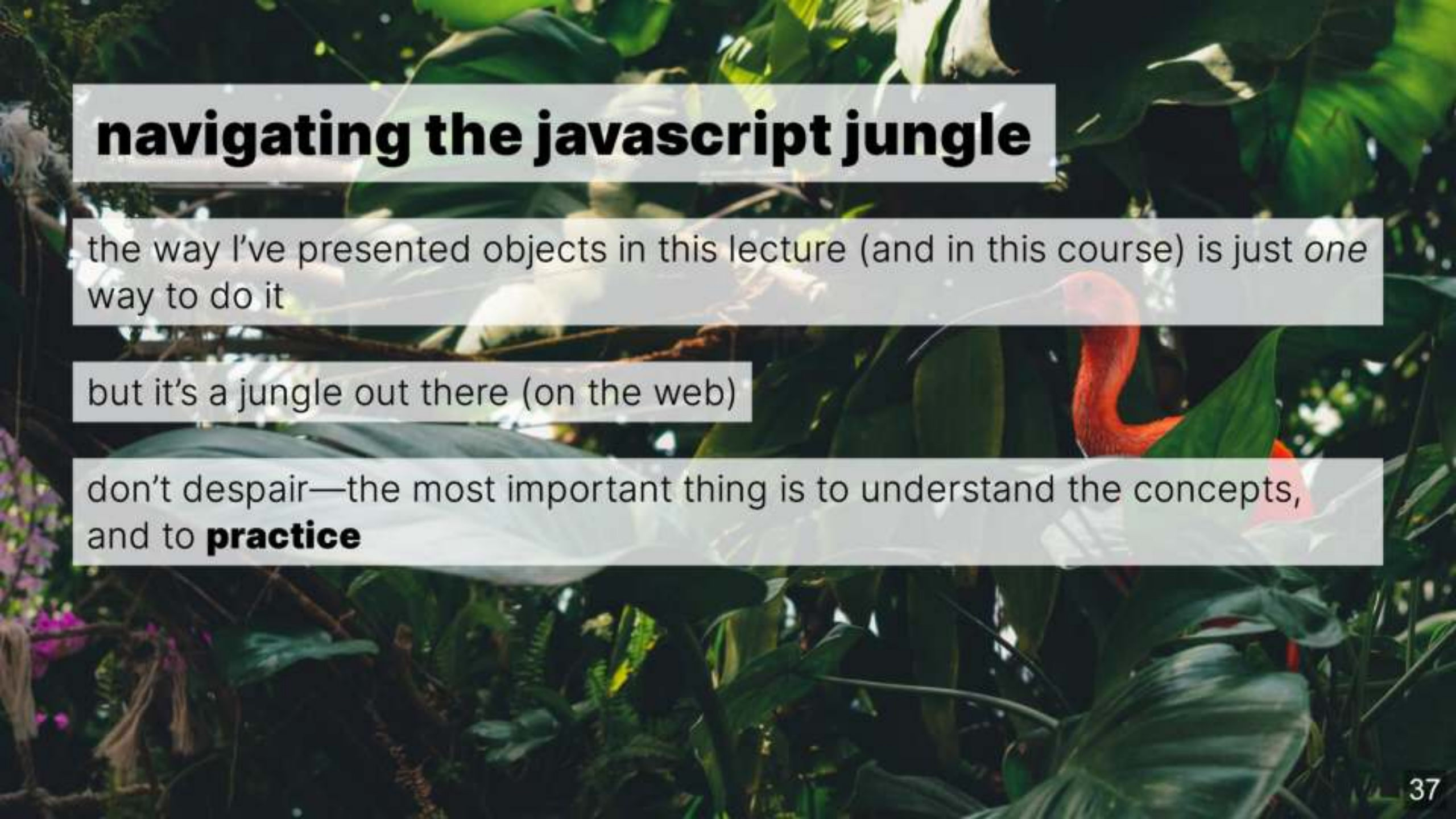
arrays **are** objects, the property names are just the numbers `0`, `1`, ..., `N-1` (where `N` is the length of the array)

which should I use?

a rule of thumb:

- use an **object** when you have several 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)



navigating the javascript jungle

the way I've presented objects in this lecture (and in this course) is just *one* way to do it

but it's a jungle out there (on the web)

don't despair—the most important thing is to understand the concepts, and to **practice**

talk

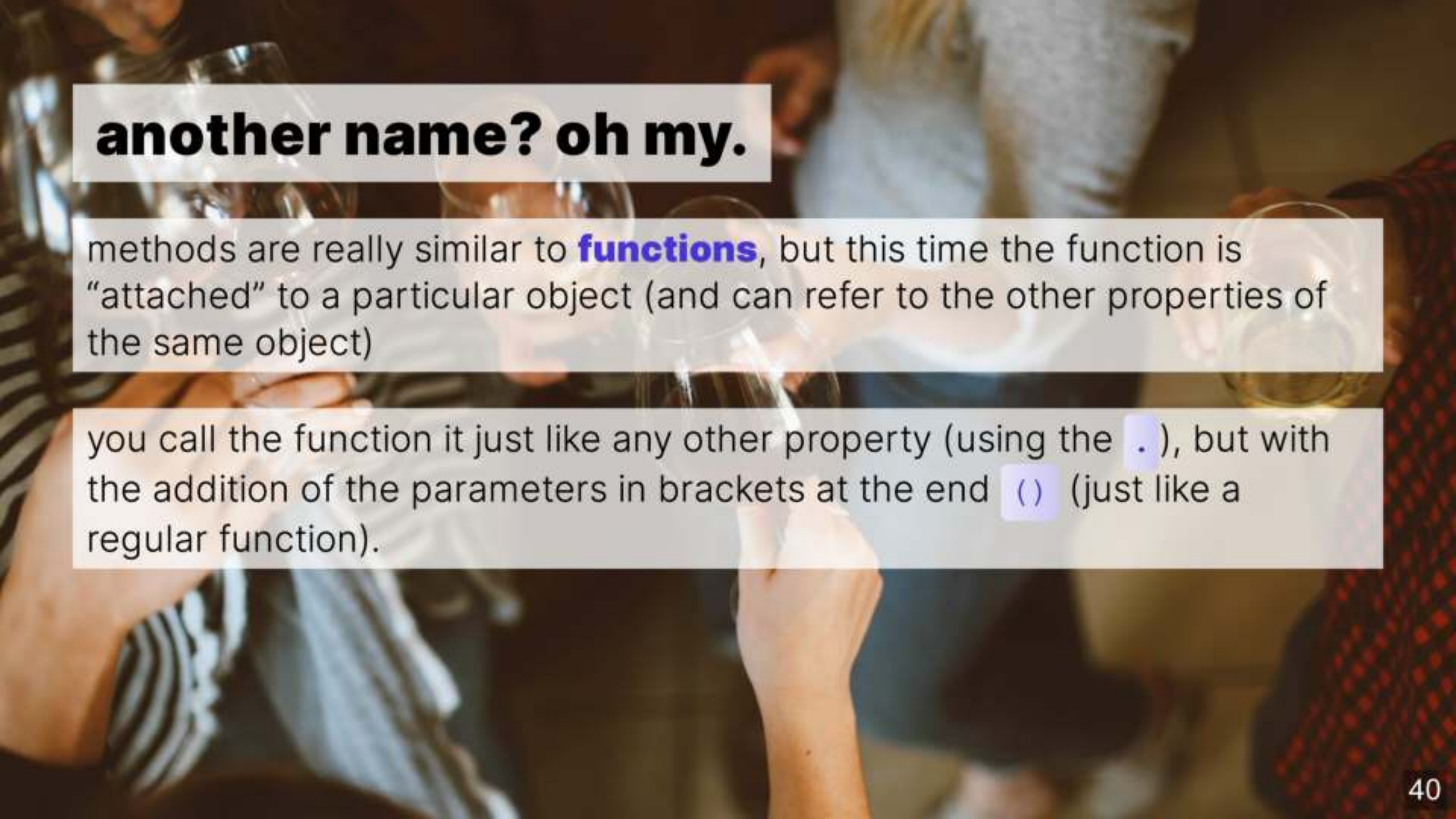
```
// this seems legit
var point = {x: 100, y: 200};
```

```
// what about this?
var point = {
  x: 100,
  redBackground: function() {
    background(255, 0, 0);
  }
};
```

methods: what if the property is a function?

in javascript, a *function* is a value just like a number, so the value of a property can be a function

in this case, the property is called a **method**



another name? oh my.

methods are really similar to **functions**, but this time the function is “attached” to a particular object (and can refer to the other properties of the same object)

you call the function it just like any other property (using the `.`), but with the addition of the parameters in brackets at the end `()` (just like a regular function).

example: `p5.Vector` object

```
var position, momentum;
function setup() {
  createCanvas(400, 400);
  // createVector() creates a p5.Vector object
  position = createVector(width/2, height/0);
  momentum = createVector(0, 0);
}

function draw() {
  background(0);
  ellipse(position.x, position.y, 40,40);
  position.add(momentum);
  momentum.add(createVector(0,1));

  if(position.y > height - 20){
    momentum.y*=-1;
  }
  momentum.mult(.98);
}
```


further reading/watching
Shiffman on Objects
MDN article on Objects


questions?