COMP1720

Art & Interaction in New Media

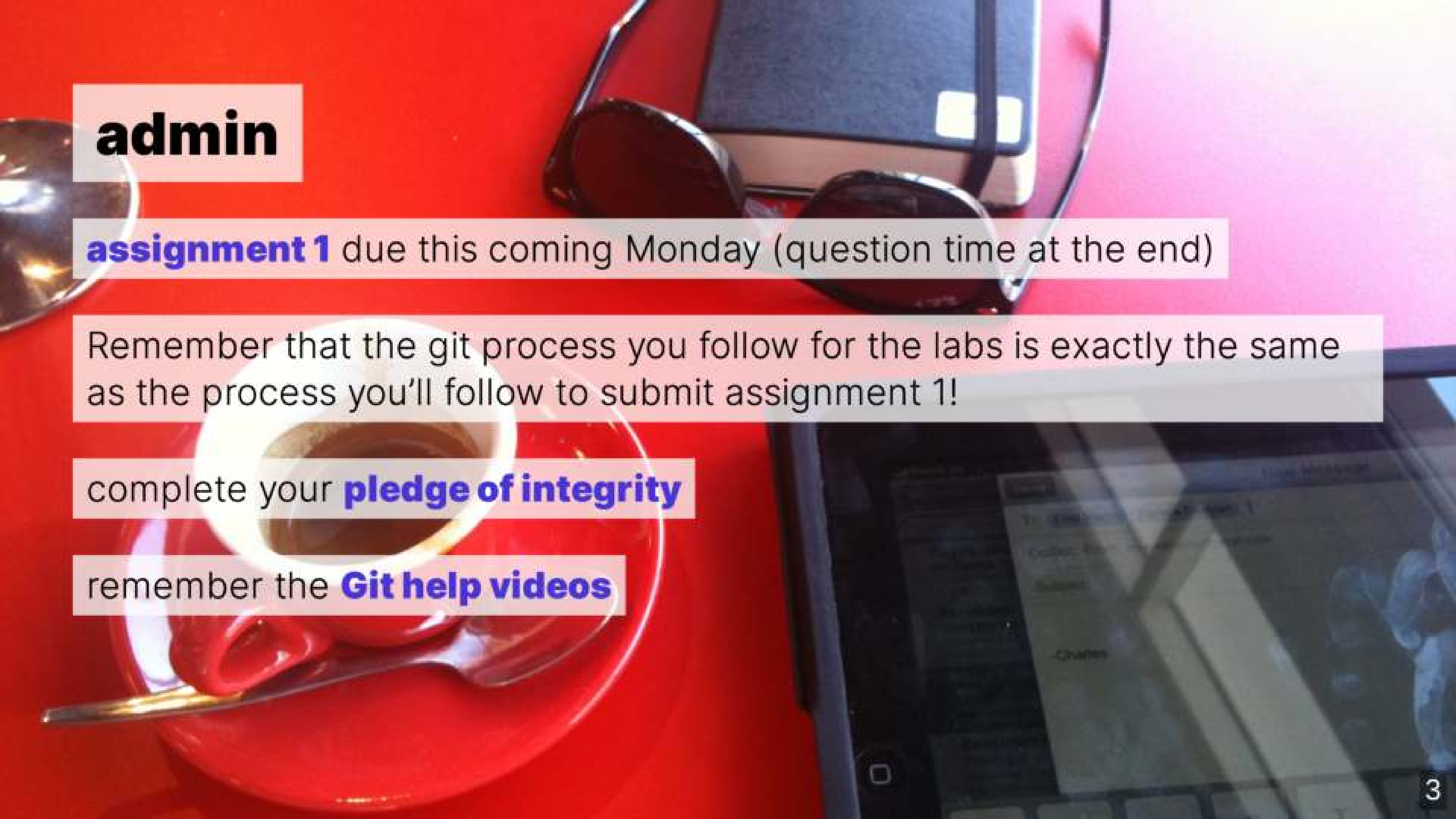
Week 3: conditionals & iteration

Dr Charles Martin

Semester 2, 2020

synopsis

- **recap** of last week's lecture
- ***code* theory** conditionals & Booleans, iteration (loops)
- **assignment 1** example



admin

assignment 1 due this coming Monday (question time at the end)

Remember that the git process you follow for the labs is exactly the same as the process you'll follow to submit assignment 1!

complete your **pledge of integrity**

remember the **Git help videos**

Course reps

Your course reps are:

- **Jarvis Huang** (COMP6720)
- **Benedict Cusack** (COMP6720)

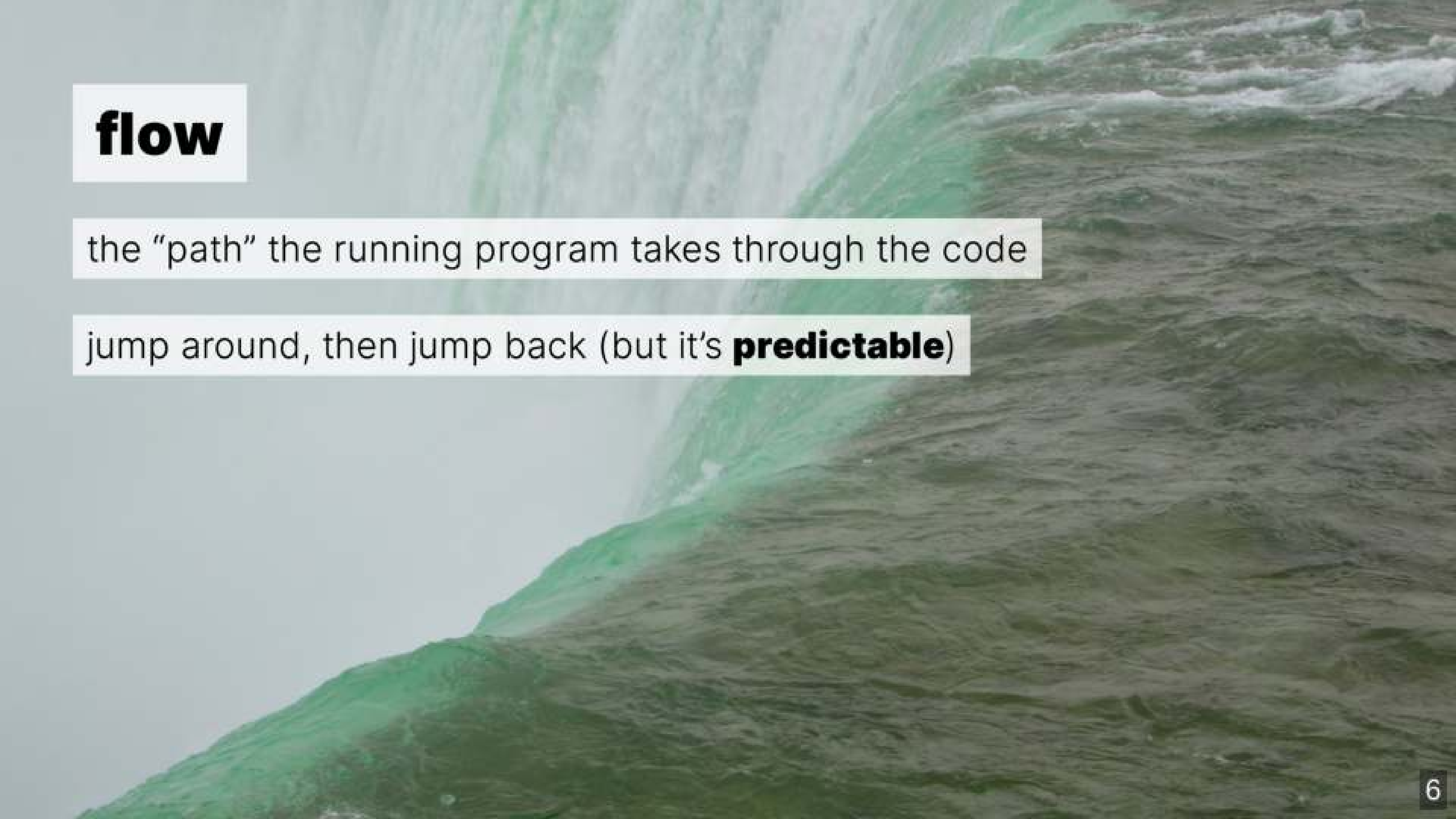
Both Jarvis and Benedict are here to listen to your concerns or issues confidentially, you can contact them over email, Teams, or on **the forum**.

COMP1720: Still looking for a COMP1720 person who might like to be a course rep! Get in touch with me if you would like to volunteer.

recap

p5 uses a `setup`, `draw`, `draw`, `draw`... “draw loop”

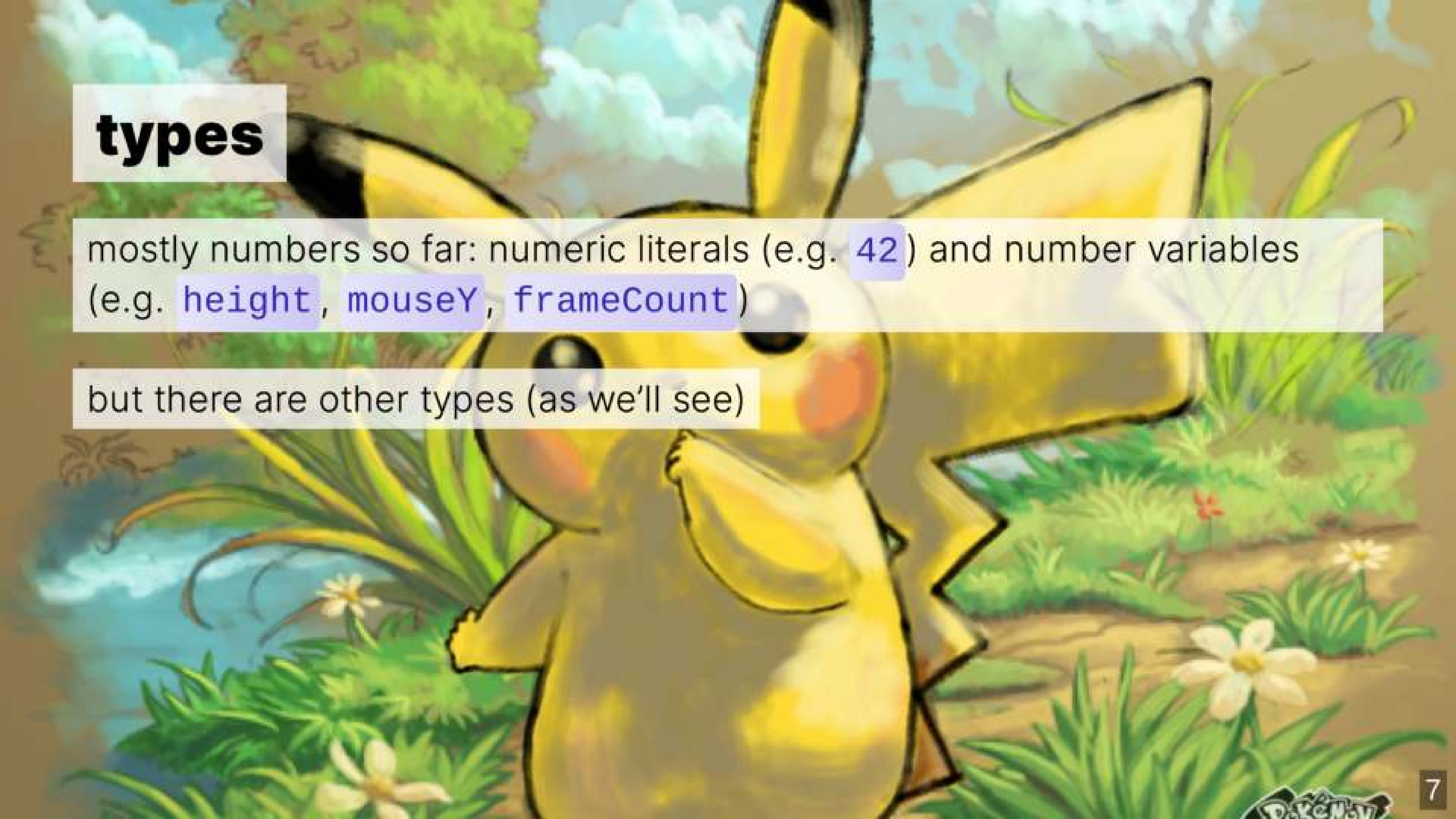
variables are a way to give names to *values* (e.g. numbers, strings, etc.) which might change



flow

the “path” the running program takes through the code

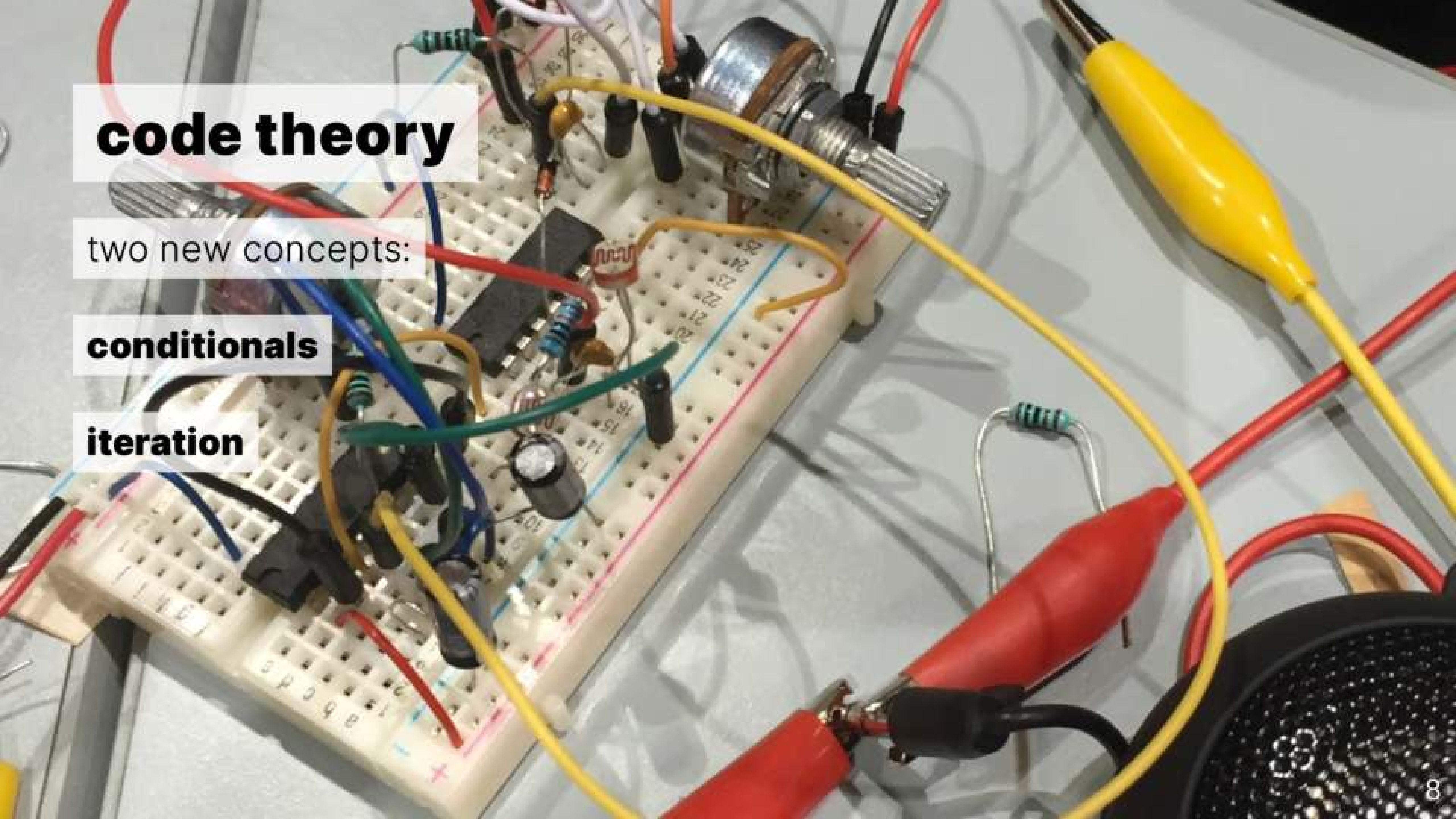
jump around, then jump back (but it’s **predictable**)



types

mostly numbers so far: numeric literals (e.g. `42`) and number variables (e.g. `height`, `mouseY`, `frameCount`)

but there are other types (as we'll see)



code theory

two new concepts:

conditionals

iteration

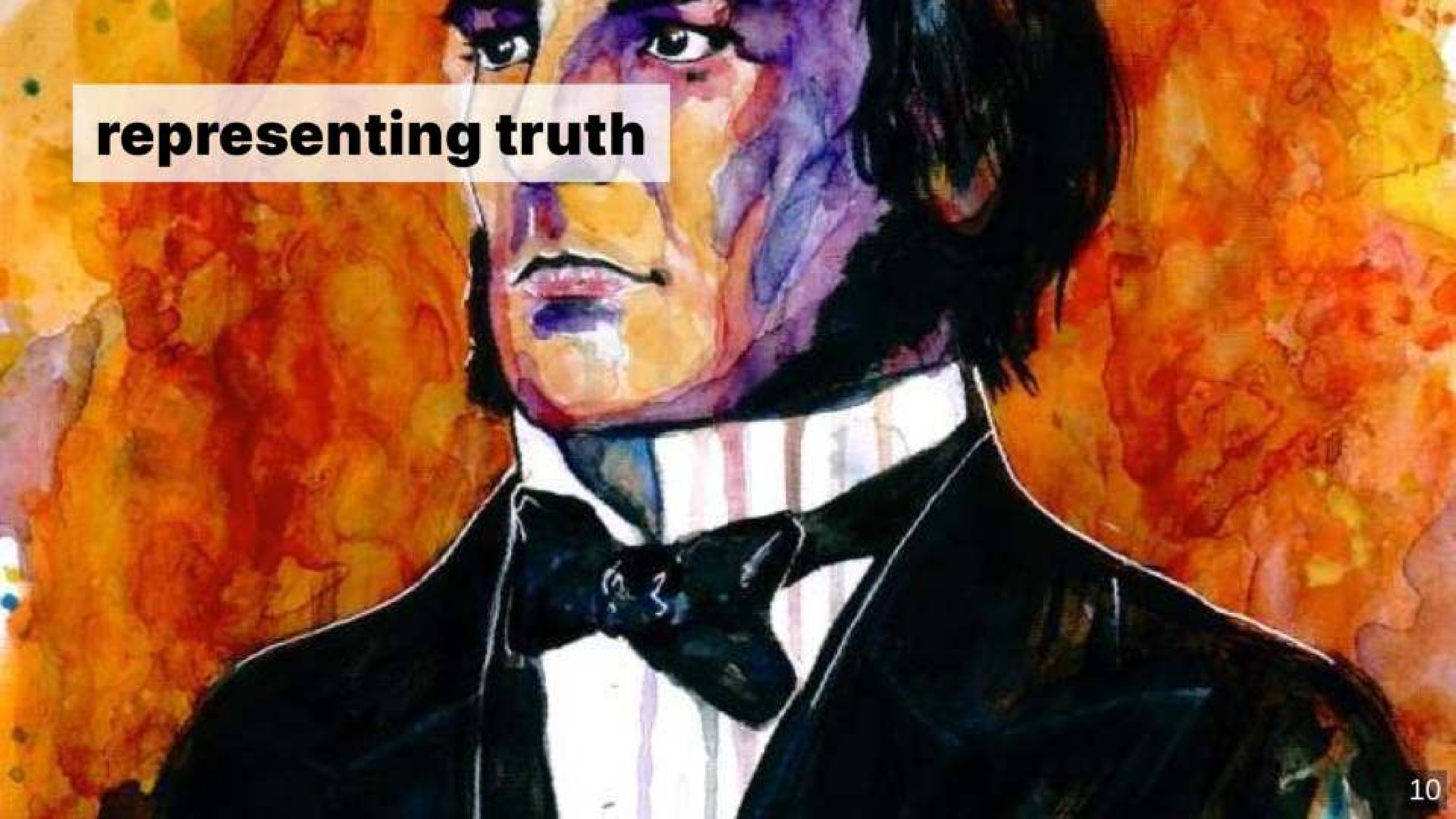
conditionals

"**if** I run, I can make it to the tram stop in time"

"**if** I have a spare month I'll read *War and Peace*"

"I'll start assignment 1 **when** I've finishd watching Netflix"

representing truth



Boolean type

there’s a special data type for representing truth values called **Boolean**

unlike **Number** (which can take any numeric value, e.g. `54` or `2.55`) or **String** (which are collections of characters e.g. `"lorem ipsum"`), **Boolean** variables must be either `true` or `false`

```
// here are some boolean variables
var the_sky_is_blue = true;
var the_sea_is_red = false;
```

Boolean logic

you can also get a Boolean from the result of a logical expression (**MDN docs**)

- `<` (less than)
- `<=` (less than **or** equal to)
- `>` (greater than)
- `>=` (greater than **or** equal to)
- `==` (equal to, note the double equals sign)
- `!=` **not** equal to

Boolean logic

all these variables are also Boolean

```
var x_is_small = mouseX < 100;
var y_is_big = mouseY >= 500;
var jan_is_happy = !true;
```

combining Boolean expressions

we can use logic to check multiple things at once:

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

combining Boolean expressions

this allows us to do things like

```
(the_sky_is_blue && the_sea_is_red)
(the_sky_is_blue || the_sea_is_red)
(!the_sky_is_blue)
```

note: each whole line above evaluates to `true` or `false` depending on whether the parts are true or false

```
function setup() {
    createCanvas(windowWidth, windowHeight);
    noStroke();
    fill(240, 50, 10);
}

// unconditional love
function draw() {
    background(255);
    arc(width/2-100, height/2-100, 250, 250, PI, TWO_PI);
    arc(width/2+100, height/2-100, 250, 250, PI, TWO_PI);
    triangle(width/2-225, height/2-100,
             width/2+225, height/2-100,
             width/2, height/2+200);
}
```

but unconditional love is hard to find!

what if you want your sketch to do one thing some of the time, and another thing the rest of the time?

javascript has an `if` statement:

```
if (condition) {
  // do something
} else {
  // do something else
}
```

this also has a human-language intuition

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

conditional love

mini-praxis

Boolean madness!

there's a whole course on this:
http://programsandcourses.anu.edu.au/2019/course/PHIL1005

the take-home message is that javascript has a special place in its heart for true/false logic stuff, and that place is the **Boolean** type

and I guess that if you want to take a course about love, maybe try **https://programsandcourses.anu.edu.au/2020/course/PSYC1003**?

what could Boolean variables be used for?

monitoring some aspect of a program which is either `true` or `false`:

- has something been clicked?
- is something finished?
- enhancing **readability** of your code (e.g. if you create variables called `isFinished`, `gameStarted`, etc.)

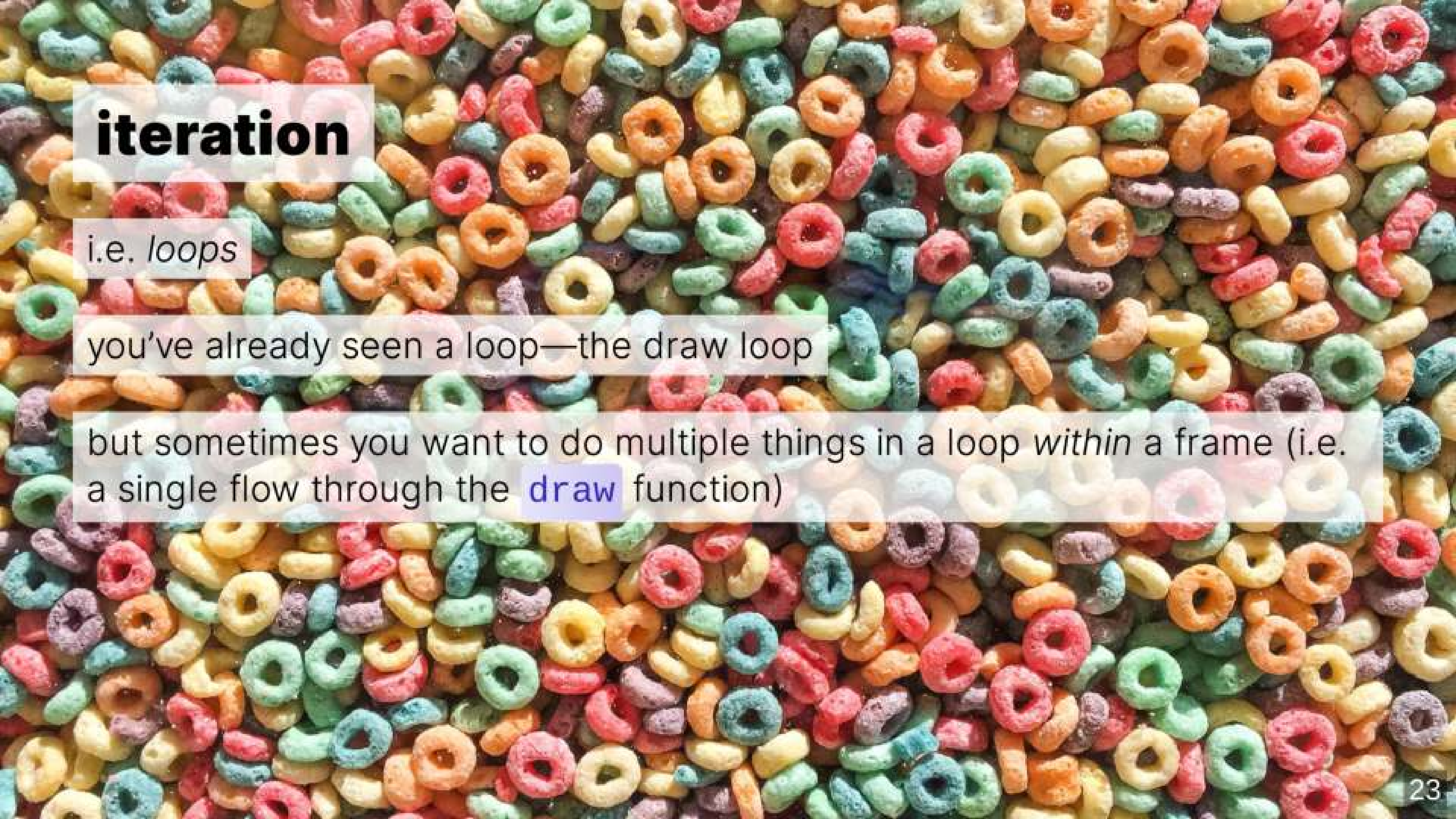
a note about the docs

here we come up against the javascript vs p5 thing—I'll try not to make things more complicated than they need to be, but it's also important not to lie to you (and there is a *bit* of trickyness here)

language features like `if` statements (and `while`/`for` loops later) are part of the javascript programming language (which is what p5 is written in)

the best javascript docs are on **the Mozilla Developer Network (MDN)** (p5 reference is no good here)

I'll point you to the reference material which is most appropriate



iteration

i.e. *loops*

you've already seen a loop—the draw loop

but sometimes you want to do multiple things in a loop *within* a frame (i.e. a single flow through the `draw` function)

`while` loop

the first type of loop we'll look at is the *while* loop:

```
while(condition){
  // do some things while "condition" evaluates to true
}
```

does this look familiar?

`while` loop intuition

a human-language intuition:

```
while(agatha_is_sitting){
  draw_circles_on_slide();
}
```

```
while(triangles_have_three_sides){
  draw_circles_on_slide();
}
```

a working example

```
var count = 0;

// count < 5 is a logical comparison
// (with a true/false answer)
while(count < 5){
  ellipse(
    random(width), random(height), // x and y
    10*count, 10*count             // width and height
  );
  count = count + 1;
}
```

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  noStroke();
  frameRate(5);
}

function draw() {
  background(30);

  var x = 0;
  while (x < width) {
    fill(x/2, random(255), 255);
    rect(x, 0, random(10), random(height));
    x += random(10);
  }
}
```

looping tips

make sure the Boolean expression will evaluate to `false` at some stage—otherwise you'll loop forever!

you probably want to have some variable which you modify in the loop, and then when the modifications are "done" the expression in the `while` loop should be `false`

when writing a loop, look at your code: *be* the computer (think about types & flow)

`for` loop

the other type of loop in javascript is the `for` loop

it’s like a `while` loop, but it has the “modify the loop variable” (e.g. `count = count + 1;`) part built-in

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

in this example we called the loop variable `i` (although you can give it whatever name you like)

differences between `for` and `while` 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
}
```

code theory recap

Boolean expressions (e.g. `mouseX < 500`) and `if` statements let us control the *flow*: do this or that

loops (`for`, `while`) allow us to control the flow: do this over and over again

you'll get *lots* of practice in the labs

further reading/watching

Shiffman videos:

conditionals,

`while` **and** `for` **loops**

MDN javascript docs (have a look at the *Tutorials* & *References* menus in the sidebar)

ass1 artist statement

> what are you trying to achieve with your name tag as a work of art?

- what is the inspiration for your design?
- what concepts are you trying to explore (e.g. shape, colour, etc.)?
- how have you tried to balance the goals of interestingness and readability?

KIERAN

example artist statement

My nametag is inspired by one of the earliest examples of popular generative art. `10 PRINT CHR$(205.5+RND(1)); : GOTO 10` is a one line program for the Commodore 64 computer that would randomly pick between two characters `/` and `\` to generate a maze on the screen. This program is an incredible example of the power of a simple generative process to produce complex imagery.

My sketch uses the same basic mechanic (a random choice between two alternatives) to produce the background of my nametag. Because the forward and backslash look very different on modern computers, I used p5 functions to create the classic shape of those characters. The blue colour is inspired by the screen colour of the C64 but I've brightened it for visual effect.

For my name I've chosen a font which has a similar "computery" style to match with the background and aligned it as closely as possible with the rows and columns of the maze. I've used a red outline to contrast with the background to ensure that my nametag is easily readable and also to create a focal point for the image.

artist statement tips

don't explain your code line-by-line—instead, think about the overall goals and effect you're trying to achieve

talk about what inspired you to come up with your design

short & sweet is ok (but make sure you *say something*)

don’t explain your code line-by-line

read the website

questions?