COMP1720

Art & Interaction in New Media

Week 2: flow & variables

Dr Charles Martin

Semester 2, 2020

synopsis

- **recap** of last week's lecture
- ***code* theory** types, variables, maths, logic
- **praxis** Katharina Grosse

admin

videos up on **lectures page**

labs have started, so you've completed your first **visual diary entry**

(that's 1.25% of the course done!)

assignment 1 released (due Monday week 4)

info

set up your **email notification preferences** for the **course Discourse forum**

it's **super-important** that you check your messages regularly

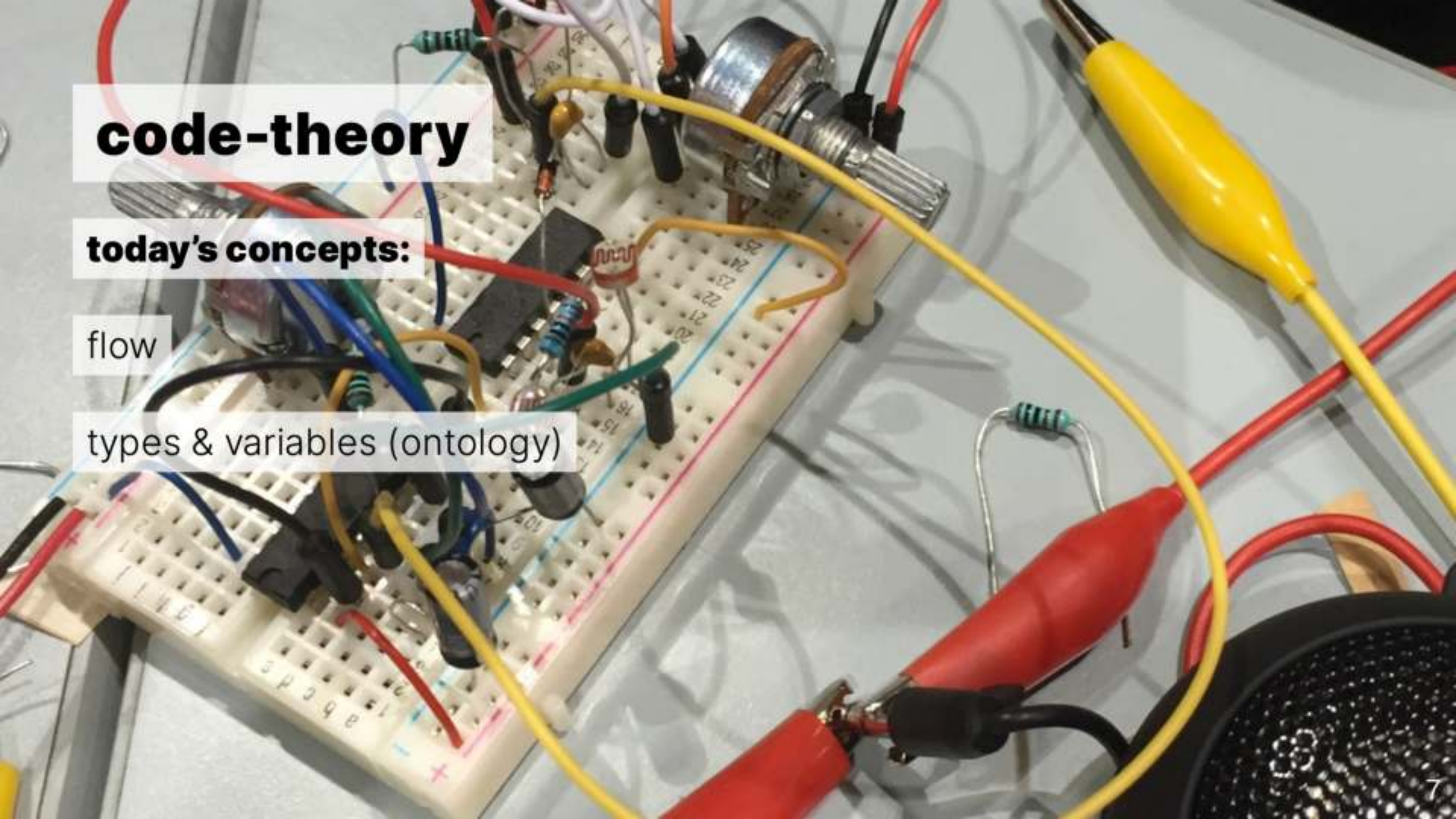
...and don't forget to **attend your lab!**

recap

painting like a 5 year-old: p5 loves to draw, and will follow your instructions, but only has a small vocabulary (recorded in the **reference**)

computers think in **numbers** (rgb for colours, the grid for position, etc.)


how does p5 communicate back to us?
DUAL Direct



code-theory

today's concepts:

flow

types & variables (ontology)

what's the flow through this code?

```
rect(300, 200, 200, 200);
ellipse(400, 300, 100, 100);
line(200, 200, 400, 400);
```

what's the flow through this code?

```
rect(300, 200, 200, 200);
ellipse(400, 300, 100, 100);
line(200, 200, 400, 400);
```

what's the flow through this code?

```
rect(300, 200, 200, 200);
ellipse(400, 300, 100, 100);
line(200, 200, 400, 400);
```

what's the flow through this code?

```
rect(300, 200, 200, 200);
ellipse(400, 300, 100, 100);
line(200, 200, 400, 400);
```

how about this one?

```
line(200, 200, 400, 400);
ellipse(400, 300, 100, 100);
rect(300, 200, 200, 200);
```

does it even matter?

remember: the “painting” metaphor

each shape in p5 is “painted” **on top** of what was already on the canvas (so the order matters!)

to “clear” the canvas (to paint it all with one colour) use the `background` function (**link to reference**)

the `background` function has several different syntax options depending on how you want to set the colour

the setup-draw loop

```
function setup() {
    createCanvas(windowWidth, windowHeight);
    // any additional setup code goes here
}

function draw() {
    // your "draw loop" code goes here
}
```

note: if a line starts with `//` it's called a "comment"; it's ignored by p5, it's just there for meatbags (humans)


flow

more loops

later in the course we'll learn to create our **own** loops, so we can do more than just setup, draw, draw, draw...

for now, see the **flow**—it'll keep going forever...

if does the same thing each time (e.g. draws a static rectangle) then we'll get a still image

but if starts to do different things...

functions & flow

p5 uses **braces** `{` and `}` (*aka* sqiggly brackets) to show where flow starts and stops

remember from **last week**: a function is a *re-usable* chunk of code which takes parameters

the `draw()` function takes zero parameters (hence the `()`) and then executes the code between the braces (called the *body*) of the function

look for the matching pairs

when you see a `{`, there will be a matching `}` further on

same for `(`, `)`, `[`, `]`

more flow

as our programs get more complex, the flow will get more complex

but at the top level it's still the same `setup`, `draw`, `draw`, `draw`, `draw`, ...

as a programmer, you are the *master* of the flow

and if you're ever wondering why your program isn't doing things in the order you expect...

remember the flow

like a series of photos (frames)

data types & variables

what numbers are these?

- 7
- 654
- 5.77
- number of planets in the solar system
- your age
- 

first steps towards animation

```
function setup(){
  createCanvas(800, 600);
}

function draw(){
  background(255);
  rect(100, 100, frameCount, 100);
}
```

what's this `frameCount` thing?

variables

some numbers are always the same (e.g. **100**)

some numbers are always the same, but have “names” (e.g. **pi**)

some numbers change (or *vary*) over time (e.g. **your age**)

variables: definition

in programming, a **variable** is how we give a *name* to a *value* (e.g. a number) which (can) change

this is really handy, because in lots of cases you're dealing with things which will change

the **name** stays the same

but the **value** can change

time in p5

time represented with **numbers** (just like position & colour were)

there are a few different ways to represent time, but the main one we'll use is the `frameCount` variable (although it's just a number)

it's *relative* (e.g. it can't tell you if it's 4pm, but it can tell you the time since your sketch started running)

using this variable in your sketch allows for **change**

more variables

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

function draw(){
  background(255);
  rect(mouseX, mouseY, 100, 100);
}
```

p5 has a bunch of useful variables built-in (as usual, the **reference** has the full list)


types

types

as well as a name, every variable has a *type* (sometimes called a data type)

in p5, values can have the following types:

- **Number** (e.g. `100`, `4.5`)
- **String** (e.g. `"Hot Potato"`—note the double quotes)
- **Boolean** (`True` or `False`)
- **Undefined** (p5 doesn't know what it is)
- **Object** (wait until week 4)

the **Parameters** section of the reference for a function tells you what types the parameters should be

***mostly* numbers so far**

but not always

declaring your own variables

we can **make our own** variables—we're not stuck with the built-in ones

there are three steps to this process:

1. **declare**: `var age;` means *there's a variable called "age"*
2. **initialise**: `age = 34` means *set the age variable to the number 34*
3. **use**: *when you do refer to the variable in your code*, e.g. `2*age`

declaring your own variables

you can combine the **declaration** and **initialisation** steps in one line (this can appear anywhere in your code)

```
//  name  value
var max = 100;
var min = 10;
```

declaring your own variables

a variable can be initialised using the value of another variable, or even with the result of a calculation

```
//  name     value
var range = max - min;
var randomValue = random(13);
```

modifying variables

the names doesn't change

but we *can* change the value

```
// name     value
   range = range + 1;
```

note: there's no `var` declaration the second time (it's already declared)

in general, we'll learn about these things by *using* them

a note about maths

most of the mathematical operators we'll use in this course you learned in primary school maths (`+`, `-`, `*`, `/` etc.)

so I won't dwell on them here, but we'll cover them extensively in the labs

you can also check out the **arithmetic operators docs on MDN**

your TODO list

week 3 visual diary!

get access to the tools

practice!

start assignment 1!

further reading/watching

variables on MDN

control flow on MDN

Shiffman video: variables

Katharina Grosse in “Fiction”