

Creative Computation for Visual Communication Design

Coding workshop 2.3

Basic concepts in computation

- MEMORY
 - Storing data and accessing it later
 - Variables, arrays, objects
- SEQUENCE
 - Running instructions in order
 - Functions, algorithms
- SELECTION
 - Making choices
 - Conditionals and logic (if, else, and, or)
- REPETITION
 - Doing the same thing more than once
 - Loops (for, while)

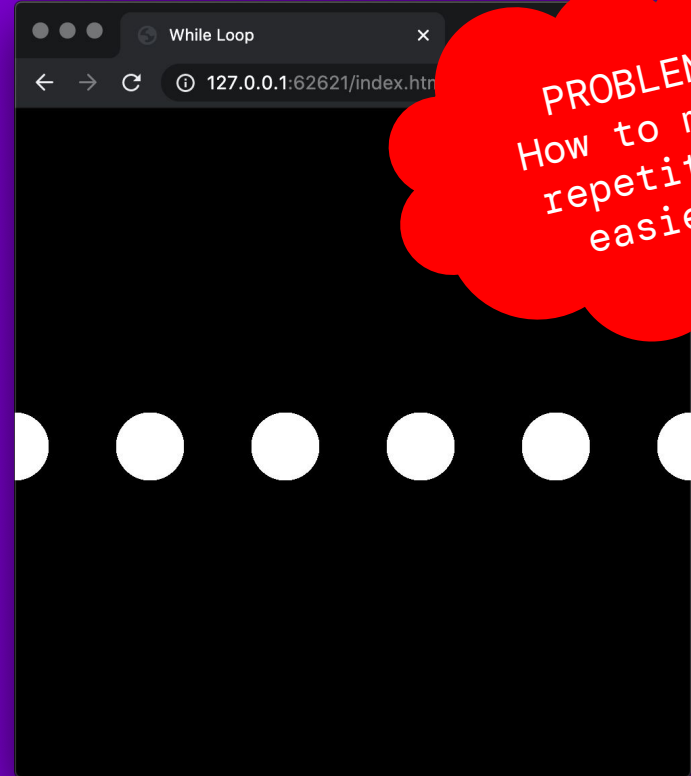
Loops



```
function draw() {  
  
}
```

Loops

```
ellipse(0, 250, 50);  
ellipse(100, 250, 50);  
ellipse(200, 250, 50);  
ellipse(300, 250, 50);  
ellipse(400, 250, 50);  
ellipse(500, 250, 50);
```



Reasons to use a computer

- AUTOMATION
 - Using computational power to complete tasks that are laborious for humans.
- OPTIMISATION
 - Seeking solutions for complex problems that are beyond human cognition.
- IDEATION
 - Reaching unexpected outcomes by seeing motion an autonomous process



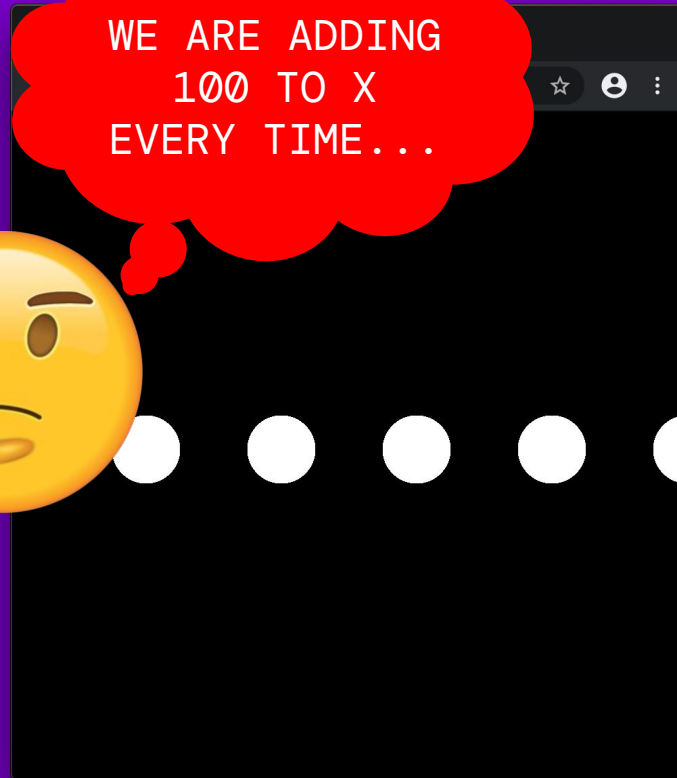
LOOPS!

Loops

```
ellipse(0, height/2, 50);  
ellipse(100, height/2, 50);  
ellipse(200, height/2, 50);  
ellipse(300, height/2, 50);  
ellipse(400, height/2, 50);  
ellipse(500, height/2, 50);
```



WE ARE ADDING
100 TO X
EVERY TIME...

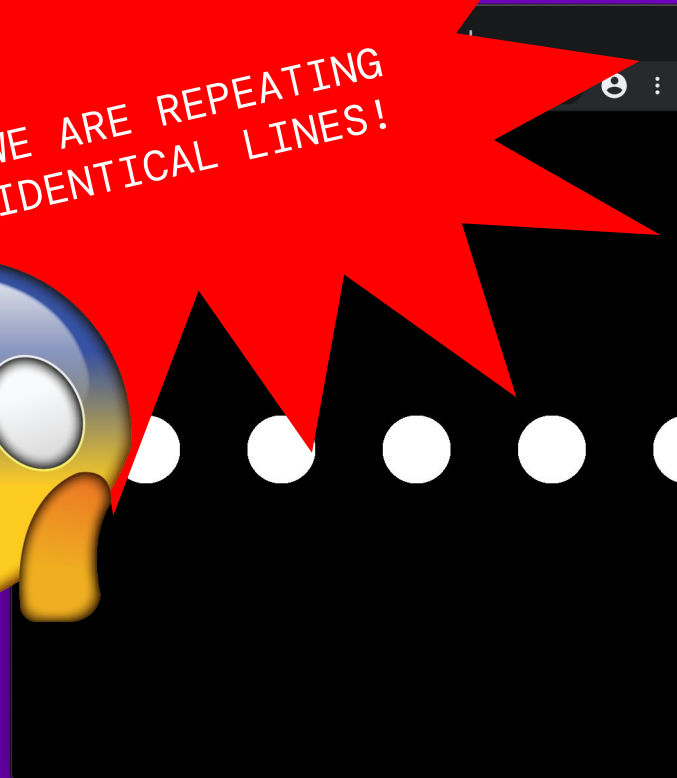


Loops

```
var x = 0;
ellipse(x, height/2, 50);
x += 100; // x is 100
ellipse(x, height/2, 50);
x += 100; // x is 200
ellipse(x, height/2, 50);
x += 100; // x is 300
ellipse(x, height/2, 50);
x += 100; // x is 400
ellipse(x, height/2, 50);
x += 100; // x is 500
ellipse(x, height/2, 50);
```



WE ARE REPEATING
IDENTICAL LINES!

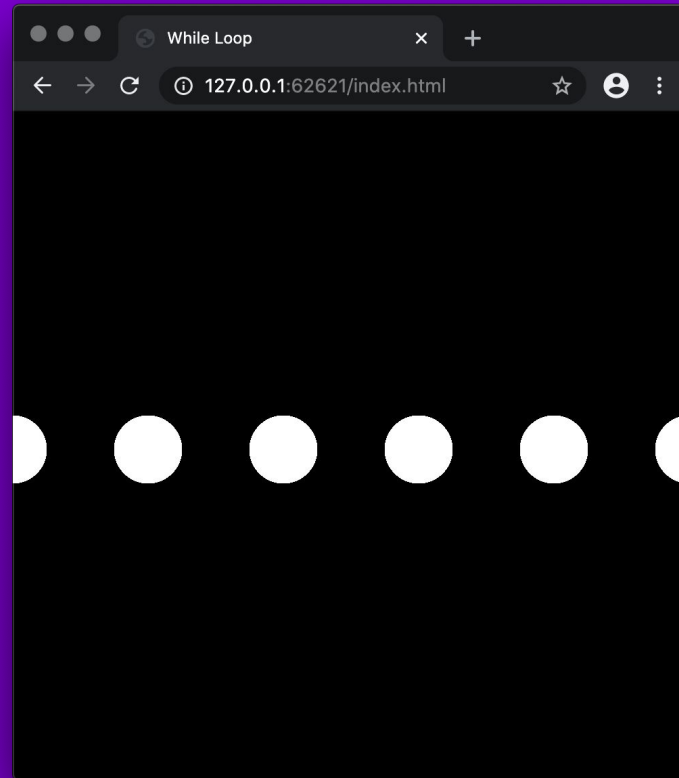


Loops



REPEAT THIS
UNTIL $x = 500$

```
ellipse(x, height/2, 50);  
x += 100;
```



While-loop

boolean
expression

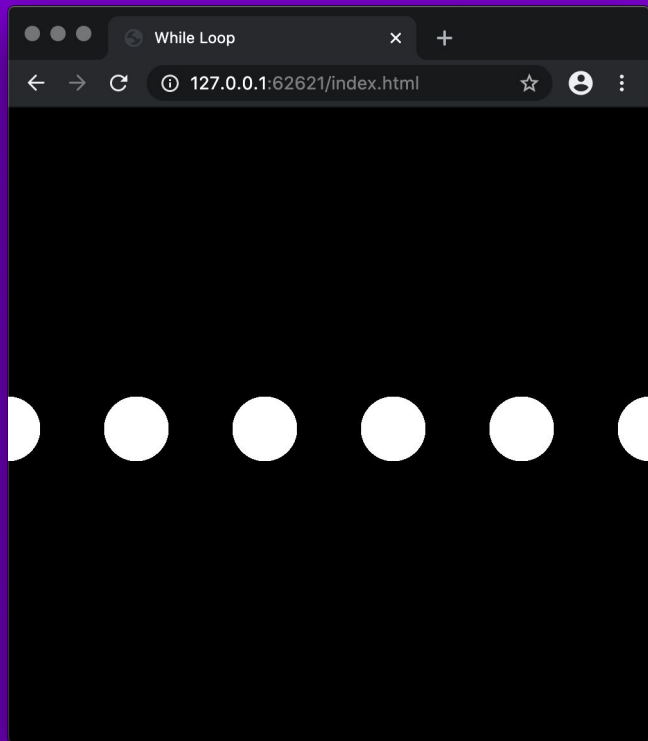
curly
brackets

```
while(condition){  
    //repeat while condition is true  
}
```

```
var x = 0; //define variable  
while(x <= 500){ //repeat while condition is true  
    ellipse(x, height/2, 50); //draw ellipse  
    x += 100; //increment variable  
}
```

BEWARE THE
INFINITE LOOP!
The condition has
to become false
at some point!

Exercise 1: While-loop



Recipe for loops

```
var x = 0;  
while(x <= 500){  
    ellipse(x, 250, 50);  
    x += 100;  
}
```

INITIAL STATE

CONDITION

INCREMENT



For-loop

Semicolons between
statements

```
for(initial state; condition; increment){  
    //code block to be repeated
```

```
}
```

curly
brackets

indent




For-loop

Executed
when loop
starts

Condition for
executing the
code block

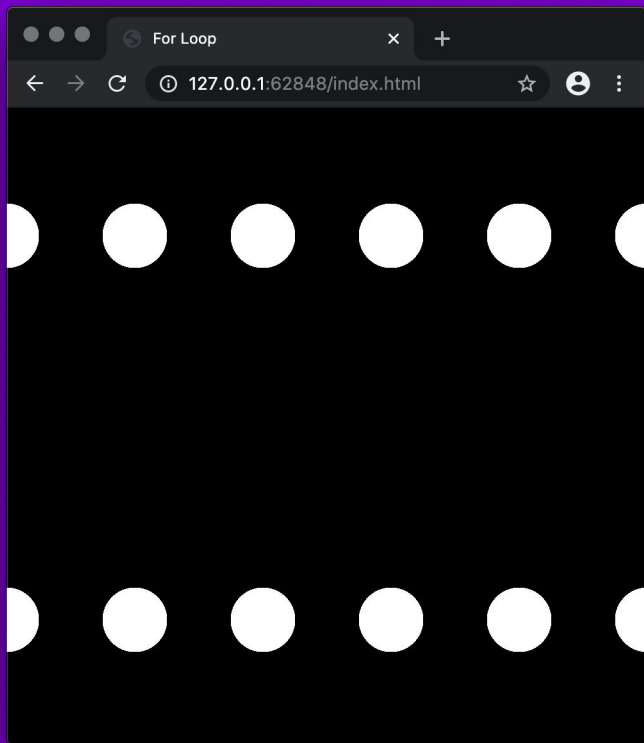
Executed
after code
block



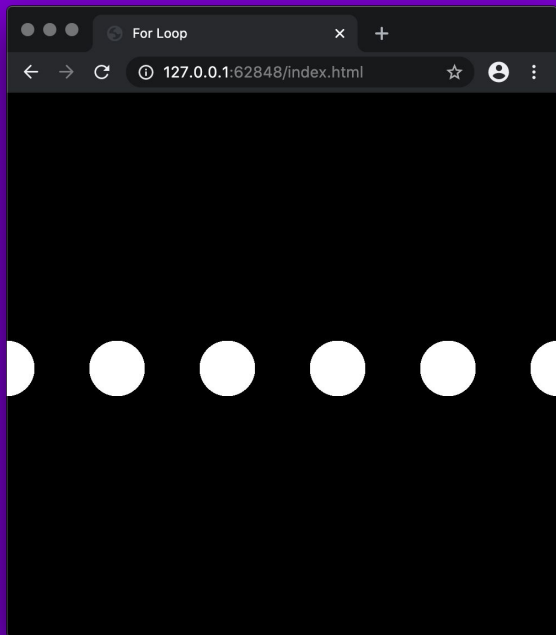
```
for(initial state; condition; increment){  
    //code block to be executed  
}
```

```
for(var x = 0; x <= 500; x += 100){  
    ellipse(x, width/2, 50);  
}
```

Exercise 2: For-loop



For-loop



LOOPING ACROSS THE DISTANCE

```
for(var x = 0; x <= width; x += 100){  
  ellipse(x, 250, 50);  
}
```

LOOPING A NUMBER OF TIMES

```
for(var i = 0; i < 6; i++){  
  ellipse(i*100, 250, 50);  
}
```

THESE TWO
DO THE
SAME THING

While vs. for

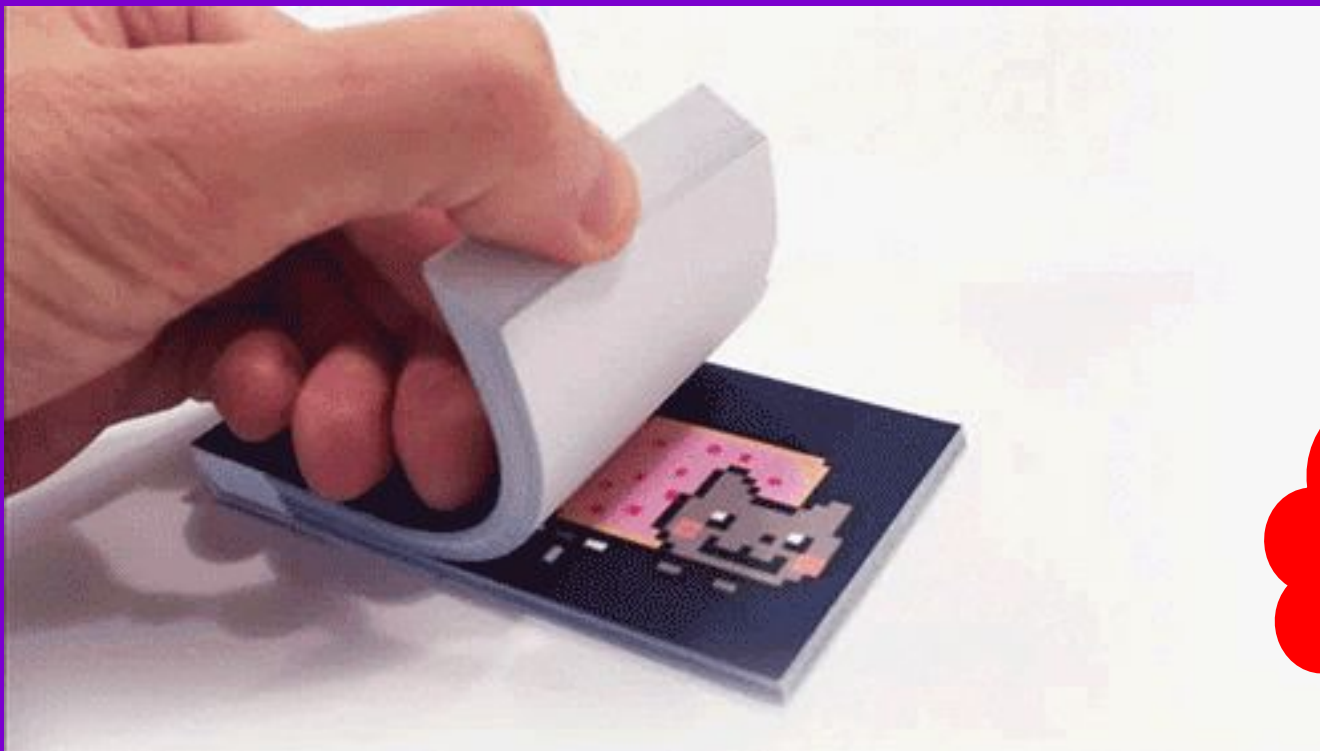


When you don't know how many times you have to repeat the code



When you know how many times you want to repeat the code

Understanding loops

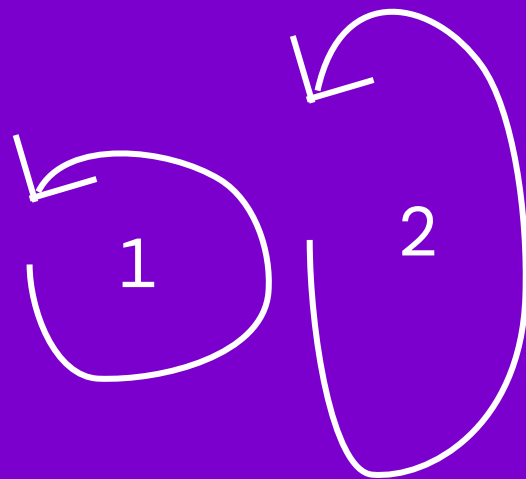


LOOPING \neq
ANIMATING

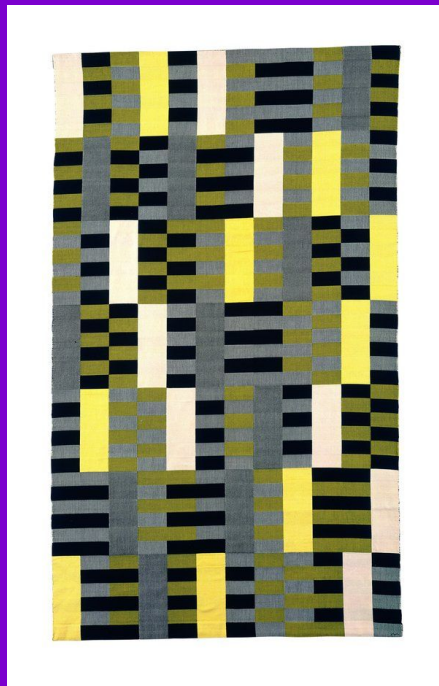
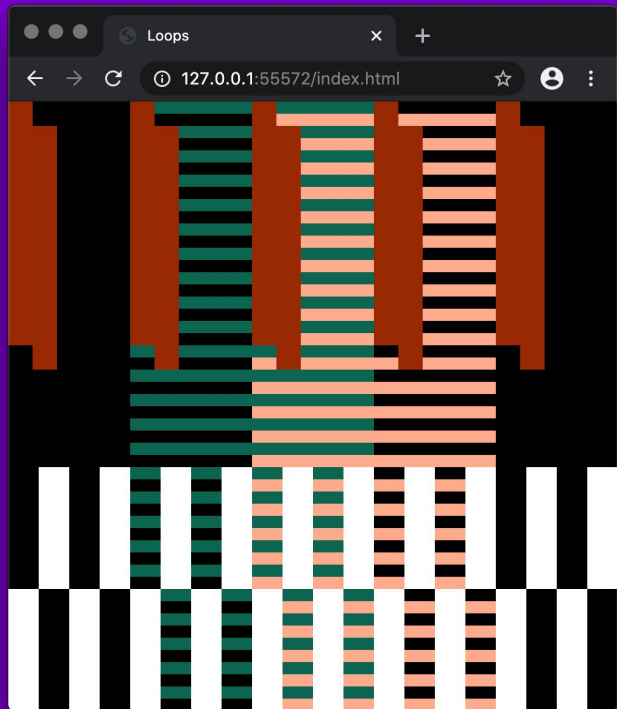
Understanding loops

- Loop diverts the program flow
- Loop structures inside draw() are **completed before** the frame is refreshed
 - Loop does not appear as animated

```
function draw(){  
  //this stuff happens first  
  for(var i = 0; i < 10; i++){  
    //this stuff happens until loop exits  
  }  
  //this stuff happens last  
}
```



Variation: Anni Albers



Anni Albers: *Black
White Yellow* (1926)



Anni Albers: *Smyrna rug design*
(1925)

Mapping

PROBLEM:
How to convert
numbers from one
range to another?

```
fill(mouseX, 100, 100);  
ellipse(mouseX, mouseY, 50, 50);
```

Mapping

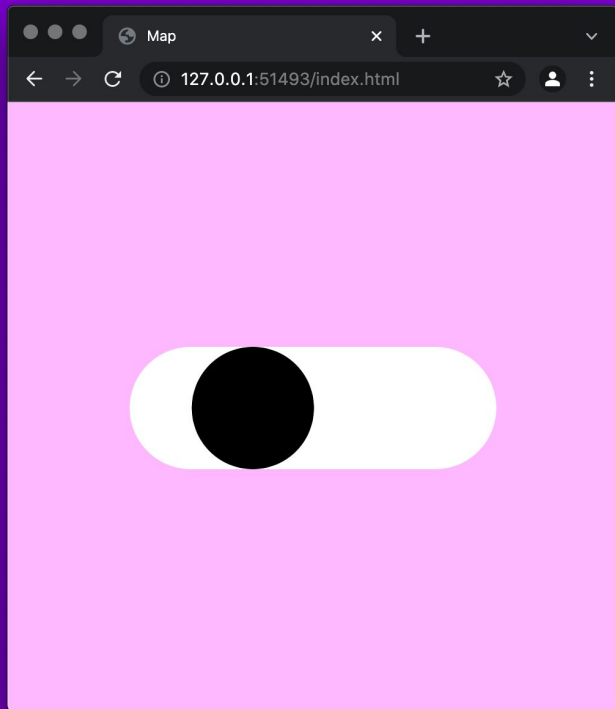
```
map(value, start1, stop1, start2, stop2);
```

- Converts values from one range to another
 - VALUE: the incoming value to be converted
 - START1: lower bound of the value's current range
 - STOP1: upper bound of the value's current range
 - START2: lower bound of target range
 - STOP2: upper bound of target range

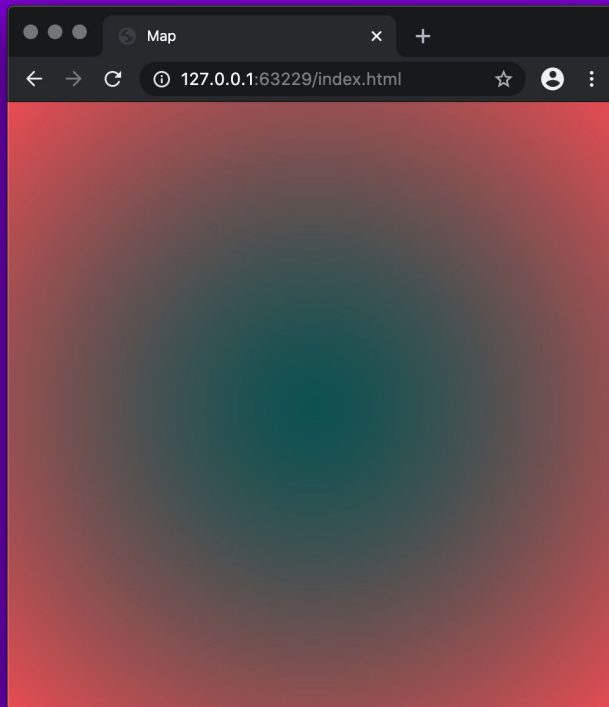
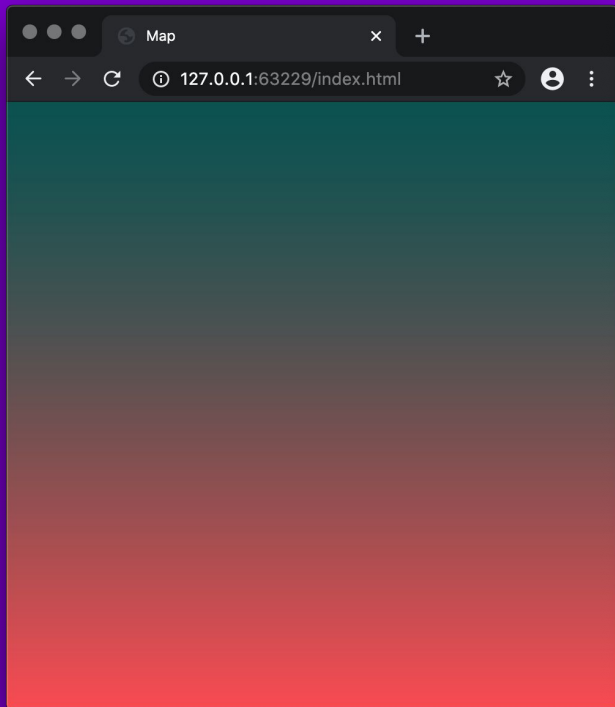
```
var x = map(mouseX, 0, width, 100, 200);
```

YOU SHOULD HAVE
AN IDEA OF THE
INCOMING
NUMBER'S RANGE!

Exercise 3: Map



Variations



Make gradients using
`map()` and a loop!

Can you figure out
how to make a radial
gradient?

Recap

```
//while-loop
while(condition){
    //repeat while condition is true
}
//for-loop
for(initial state; condition; increment){
    //code block to be executed
}
//mapping values from one range to other
map(value, start1, stop1, start2, stop2);
```

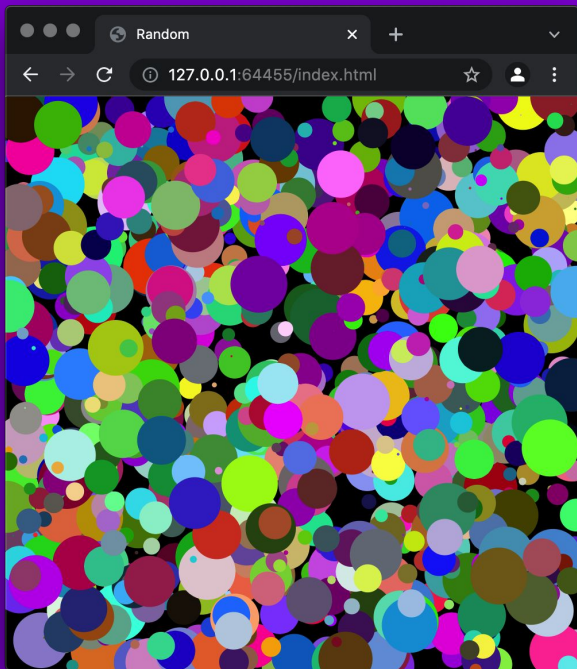
Coding Assignment II: Clock

Coding Assignment II:



Make a program that visualises time. It can be an abstract visualisation that shows the passing of time, or it can be a more literal clock that counts seconds, minutes and hours
CONSTRAINT: The visualisation should change over time.

Coding Assignment II: TIPS



We have looked at different ways to animate shapes. Eg. how to use incrementation, `frameRate`, `rotation` etc. Think how to use these methods to make an engaging visualisation.

Also check out the `millis()` function that gives the number of milliseconds since starting your sketch. You can use this to time events.

Coding Assignment II: TIPS



You can also use the `second()`, `minute()` and `hour()` functions that give the current time.

You can make a more “conventional” clock by using these values to draw and transform shapes.

Coding Assignment II: INSTRUCTIONS

- Comment your code well!
 - Add your name and date on the top
 - Explain how the code works
 - Explain the key features of your clock.
How does it visualise time?
- Make sure the code runs without errors
- If you borrow code, reference it well and be explicit how you have modified it
- Submit a **.zip folder** containing everything that is needed to run the code
 - index.html, sketch.js, p5.min.js