introduction to media computing
week 05

School of creative media, mw

Today’s topics (week 05)

- quick review
- `for()` loop
- nested `for()` loops

Today's topics (week 05)

js

- quick review
- `for()` loop
- nested `for()` loops

p5*

- `console.log()`
- mouse interactivity with `pmouseX` and `pmouseY`
- Measure time with `millis()`

Review: Nested Blocks of Code

```
if (mouseX > 200) {
  // press-and-paint
  if (mouseIsPressed) {
    fill(255,255,0);
    text("X", mouseX, mouseY);
    fill(0,0,255);
    text("X", mouseX+2, mouseY+2);
  }
}
else {
  // move-and-paint
  rectMode(CENTER);
  fill(200);
  rect(mouseX, mouseY, 25, 25);
  fill(100);
  rect(mouseX, mouseY, 10, 10);
}
```

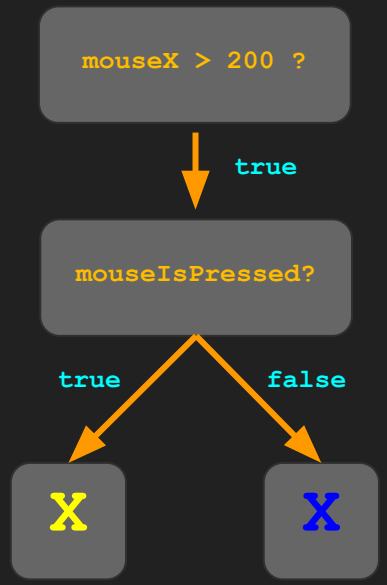
inner block

inner block

Nested blocks

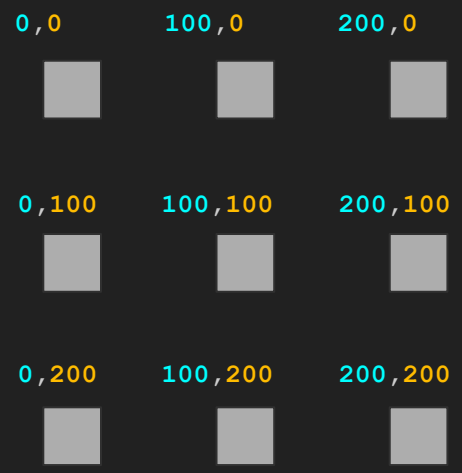
Review: Nested `if-else`

```
if (mouseX > 200) {

  if (mouseIsPressed) {
    fill(255,255,0);
    text("X", mouseX, mouseY);
  }
  else {
    fill(0,0,255);
    text("X", mouseX+2, mouseY+2);
  }

}
```



Review: Nested while() loop

```
let y = 0;
while (y < 300) {
  let x = 0;
  while (x < 300) {
    rect(x,y,30,30);
    x = x + 100;
  }
  y = y + 100;
}
```



`for()` loop

for() loop

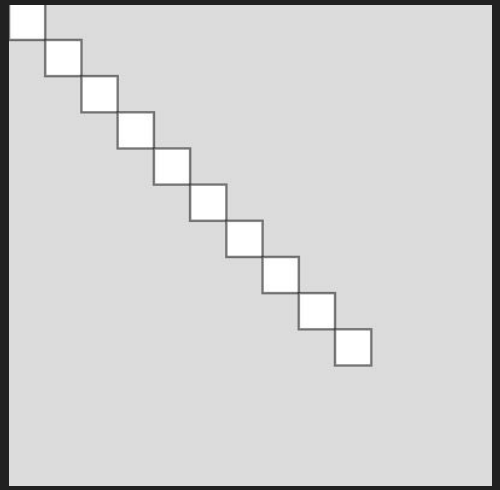
A concise construct for looping

```
for (let i = 0; i < 10; i++) {
  // Some Code here
}
```

for() loop

A concise construct for looping

```
for (let i = 0; i < 300; i += 30){
  rect(i,i,20,20);
}
```



for() loop

A concise construct for looping

```
for (let i = 0; i < 10; i++) {
  // Some Code here
}
```

Initialization

To declare the loop counter variable and its initial value.

for() loop

A concise construct for looping

```
for (let i = 0; i < 10; i++) {
  // Some Code here
}
```

condition

The condition to fulfill <u>BEFORE</u> each iteration

for() loop

js

A concise construct for looping

```
for (let i = 0; i < 10; i++) {
  // Some Code here
}
```

iteration

To execute AFTER each iteration

for() loop

A concise construct for looping

```
for (let i = 0; i < 10; i++) {
  // Some Code here
}
```

```
for (<init>; <cond>; <iter>) {
  // Some Code here
}
```

for() loop

Comparison with while() loop

```
for (let i = 0; i < 10; i++) {
  // Some Code here
}
```

```
let i = 0;
while (i < 10) {
  // Some Code here
  i++;
}
```

for() loop

Comparison with while() loop

initialization: declare & initialize loop counter variable

```
for (let i = 0; i < 10; i++) {
  // Some Code here
}
```

```
let i = 0;
while (i < 10) {
  // Some Code here
  i++;
}
```

for() loop

Comparison with while() loop

condition: the condition to check

```
for (let i = 0; i < 10; i++) {
  // Some Code here
}
```

```
let i = 0;
while (i < 10) {
  // Some Code here
  i++;
}
```

for() loop

Comparison with while() loop

iteration: update loop counter variable

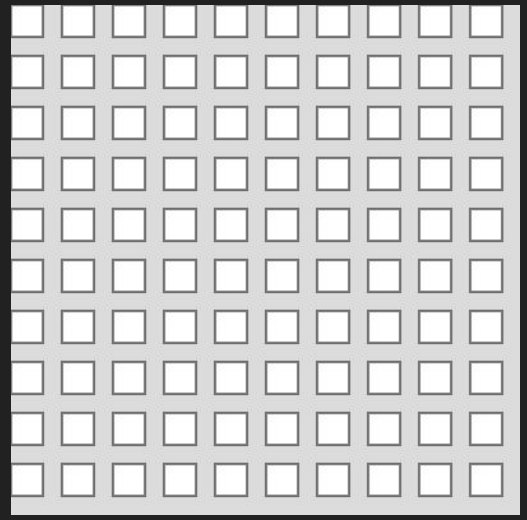
```
for (let i = 0; i < 10; i++) {
  // Some Code here
}
```

```
let i = 0;
while (i < 10) {
  // Some Code here
  i++;
}
```

Nested `for()` loop

Nested use of `for()` loop

```
for (let y = 0; y < 10; y++) {
  for (let x = 0; x < 10; x++) {
    rect(x*40, y *40, 25, 25);
  }
}
```



Nested `for()` loop

Nested use of `for()` loop

```
for (let y = 0; y < 10; y++) {
  for (let x = 0; x < 10; x++) {
    rect(x*40, y *40, 25, 25);
  }
}
```

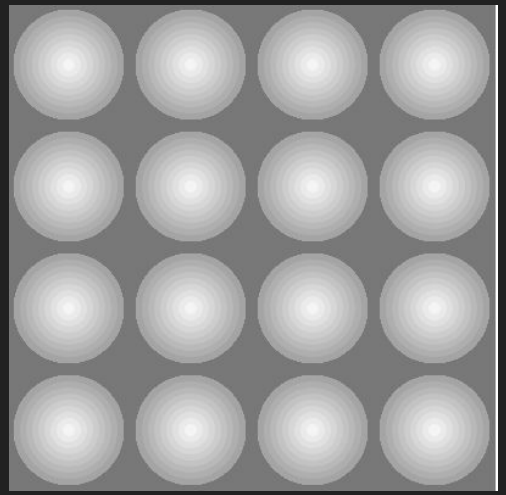
More concise

```
let y = 0;
 while (y < 10) {
   let x = 0;
   while (x < 10) {
     rect(x*40, y *40, 25, 25);
     x++;
   }
   y++
 }
```

exercise

In-class exercise 1

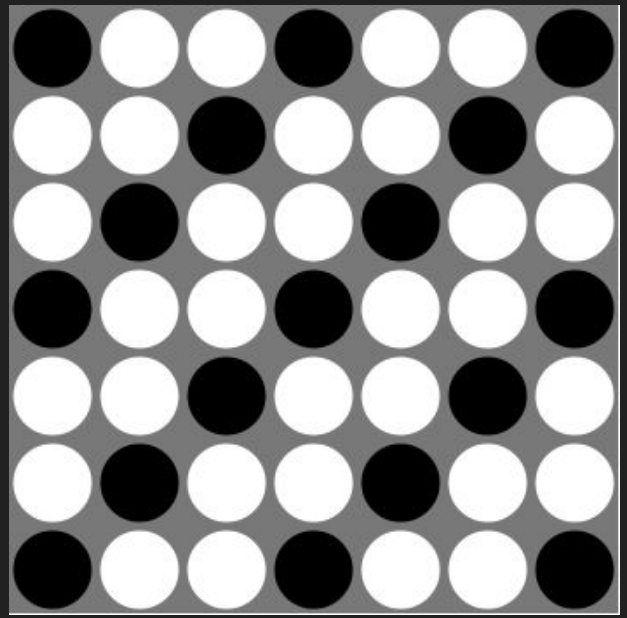
1. Use a `for()` loop to draw the shape shown on the left



2. Now, use the code in step 1 as a basic block for drawing the shape. Create a new <u>nested</u> `for()` loop to supply a sequence of `(x,y)` coordinates, and re-use the 'step 1' block to to fill the whole canvas.

exercise

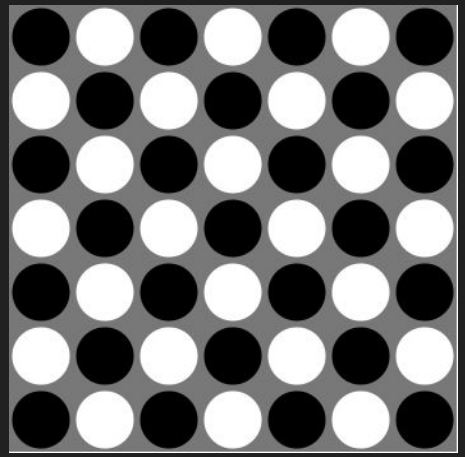
In-class exercise 2

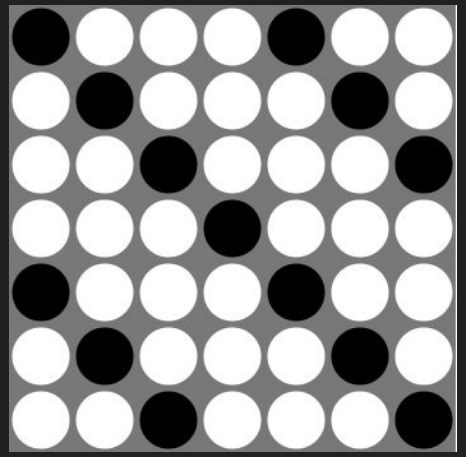


Use nested `for()` loop and to produce the pattern as shown on the left.

exercise

In-class exercise 2





If your code is general enough, you may produce other patterns quickly by changing one single constant in your code.

Simple logging `console.log()`

More mouse tracking `pmouseX,pmouseY`

Measure time with `millis()`

Simple logging `console.log()`

`console.log()` helps us to keep track of variables

Example:

```
console.log(x);
```

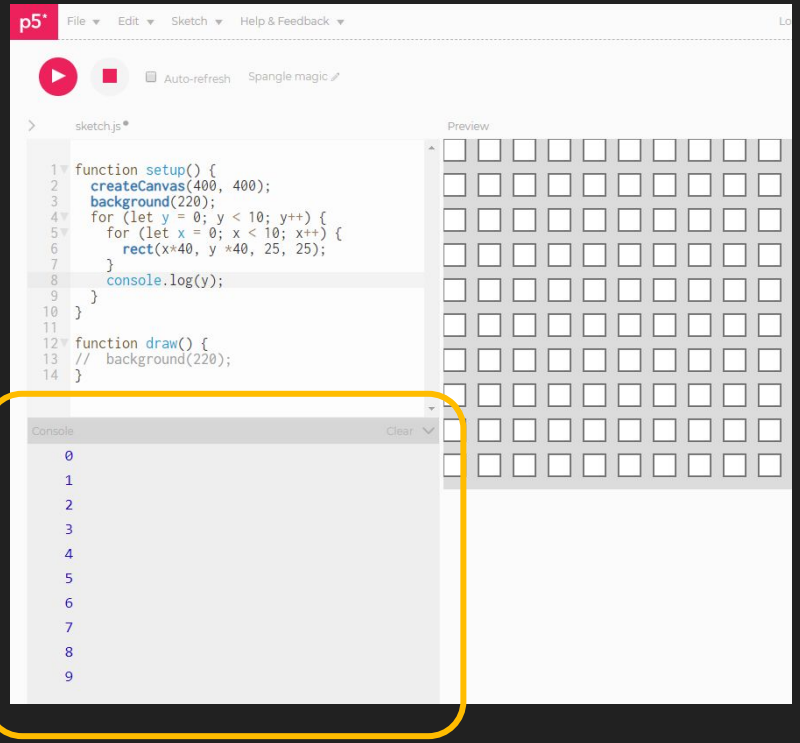
prints the value of variable `'x'` to the console.

** where is the console ? **

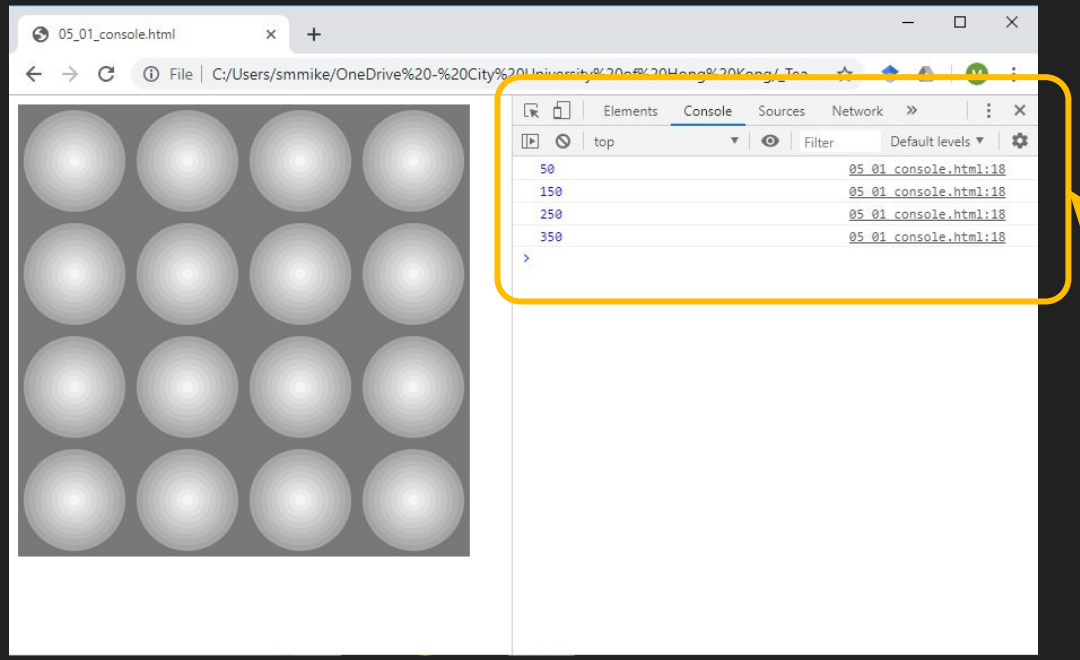
```
console.log(x);
```

Simple logging `console.log()`

p5.js editor console



Simple logging `console.log()`



Chrome browser's 'Developer Tools' **console**

Invoked via
'More Tools → Developer Tools'

More mouse `pmouseX`, `pmouseY`

`pmouseX` and `pmouseY` store the values of `mouseX` and `mouseY` of previous frame (drawn by `function draw()`) respectively.

Example: to measure the mouse displacement:

```
dist(pmouseX, pmouseY, mouseX, mouseY);
```

Measure time with `millis()`

`millis()` returns the number of milliseconds since the program has started.

(1 millisecond = 0.001 sec)

Measure time with `millis()`

try p5*

A simple time counter which updates the text once a second.

JS

```
function setup(){
  createCanvas(200,200);
  textSize(100);
  last = millis();
}

let second = 0;

function draw() {

  let now = millis();
  if (now - last > 1000) {
```

Result

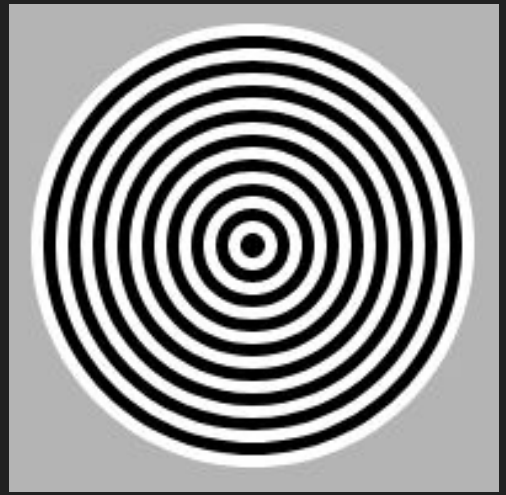
EDIT ON CODEPEN

Resources

drag to resize

exercise

In-class exercise 3

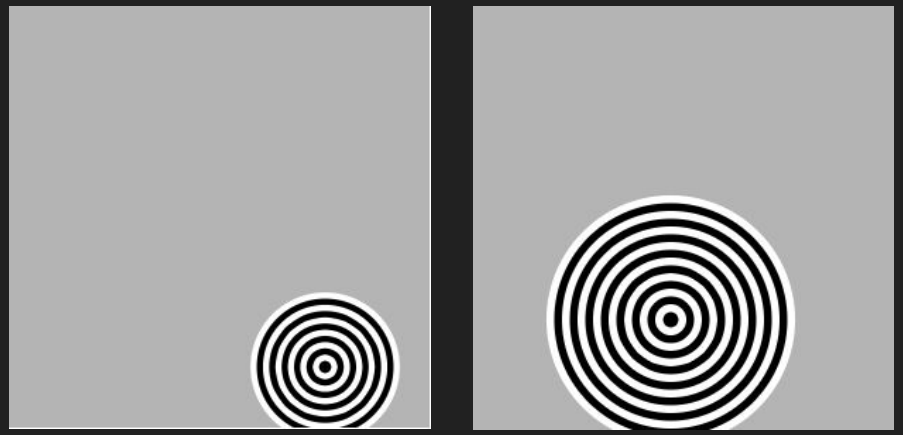


Pattern Clock

Use p5.js time measuring function `millis()` and `for()` loop to create a growing pattern (gets larger each second) as shown, the animated sample is on Canvas Week 05 page. The pattern resets itself after a certain period of time.

exercise

In-class exercise 4



Interactive pulse (Live sample on Canvas)

1. Use the result of exercise 3 as your building block for drawing the pattern but make it grow faster, say every 10 milliseconds.

2. The pattern always follows the mouse cursor, and its size adapts to the movement of mouse (hint: use `dist()`,`pmouseX`, `pmouseY` etc.). The faster the mouse moves, the larger the pattern is.