



# introduction to media computing

## week 03

# Today's topics (week 03)



- **operators & conditionals**
  - review
  - the modulo operator `' % '`
- **logical operators**
- **coding style**
- **loops I: `while()` loop**

## Today's topics (week 03)



- **operators & conditionals**
  - review
  - the modulo operator `' % '`
- **logical operators**
- **coding style**
- **loops I: `while()` loop**



- **p5.js online editor**
- **keyboard interactivity**
- **drawing text**

# Resources for review



<https://canvas.cityu.edu.hk/courses/31242>

SM1103A Introduction to Media Compt'ng

This course will teach fundamental programming concepts via creative exercises and small projects. Toward this end, students will explore the concepts of variables, sequential programming, loops, conditionals, arrays, functions with the programming of multimedia, such as image, audio, video, animation, and interactivity.

**General Information**

- [Instructors & Sessions](#)
- [Evaluation](#)
- [Readings](#)
- [Freedom](#)

**Lecture Notes**

- [Week 01](#)
- [Week 02](#)

**Assignments**

**Programming Topics**

- [Variables](#)
- [Operators](#)
- [Conditionals](#)
- [Loops](#)
- [Arrays](#)
- [Functions](#)
- [Program Flow](#)

**Resources**

- [Useful Snippets](#)
- [Tips & Tricks](#)
- [The p5.js website](#)
- [The p5.js reference](#)
- [The p5.js online editor](#)
- [The Atom editor website](#)

## Programming Topics

- [Variables](#)
- [Operators](#)
- [Conditionals](#)
- [Loops](#)
- [Arrays](#)
- [Functions](#)
- [Program Flow](#)

## Review: math. operators

```
let a = 10;  
let b = 6;  
let result;
```

```
result = a + b;
```

**addition**

```
result = 16
```

```
result = a - b;
```

**subtraction**

```
result = 4
```

```
result = a * b;
```

**multiplication**

```
result = 60
```

```
result = a / b;
```

**division**

```
result = 1.6667
```

```
result = a % b;
```

**modulo**

```
result = 4*
```

**\*Remainder of integer division**

# Review: assignment operators

```
let r;  
r = 10;  
r = r + 1;  
r += 2;  
r -= 2;  
r *= 2;  
r /= 2;  
r %= 2;
```

<b>assignment</b>	<code>r = 11</code>	<code>(10 + 1)</code>
<b>add. assignment</b>	<code>r = 13</code>	<code>(11 + 2)</code>
<b>sub. assignment</b>	<code>r = 11</code>	<code>(13 - 2)</code>
<b>mul. assignment</b>	<code>r = 22</code>	<code>(11 * 2)</code>
<b>div. assignment</b>	<code>r = 11</code>	<code>(22 / 2)</code>
<b>mod. assignment</b>	<code>r = 1</code>	<code>(11 % 2)*</code>

\*Remainder of integer division

## Review: other operators

```
let x = 200;  
x++;
```

'++' : increment

$x++$  is equivalent to  $x = x + 1$

```
let x = 200;  
x--;
```

'--' : decrement

$x--$  is equivalent to  $x = x - 1$

## Review: if else

```
if (x == 200) {  
    // Do something  
}  
else if (x < 200) {  
    // Do something  
}  
else {  
    // Do something else  
}
```

Only ONE block of  
code will be executed.



## Review: relational operators

```
if (x >= 200) {  
    // Do something  
}  
else {  
    // Do something else  
}
```

operators	meaning
>	larger than
<	smaller than
>=	larger or equal to
<=	small or equal to
!=	not equal to
==	equal to



# Modulo Operator `' % '`

## Modulo Operator `' % '`

- modulo operator `' % '` computes the remainder of an integer division. Example: `5 % 2` returns `1`.
- This operator is particularly useful for some simple looping operation.

# Modulo Operator '%'

p5\*

JS

```
let num = 0;

function setup(){
  createCanvas(200,200);
  fill(128);
}

function draw() {
  // Loop thru 0 - 256 for color
  let brightness = num % 256;
  background(brightness);
}
```

Resources

Result



EDIT ON CODEPEN

1x

0.5x

0.25x

Rerun



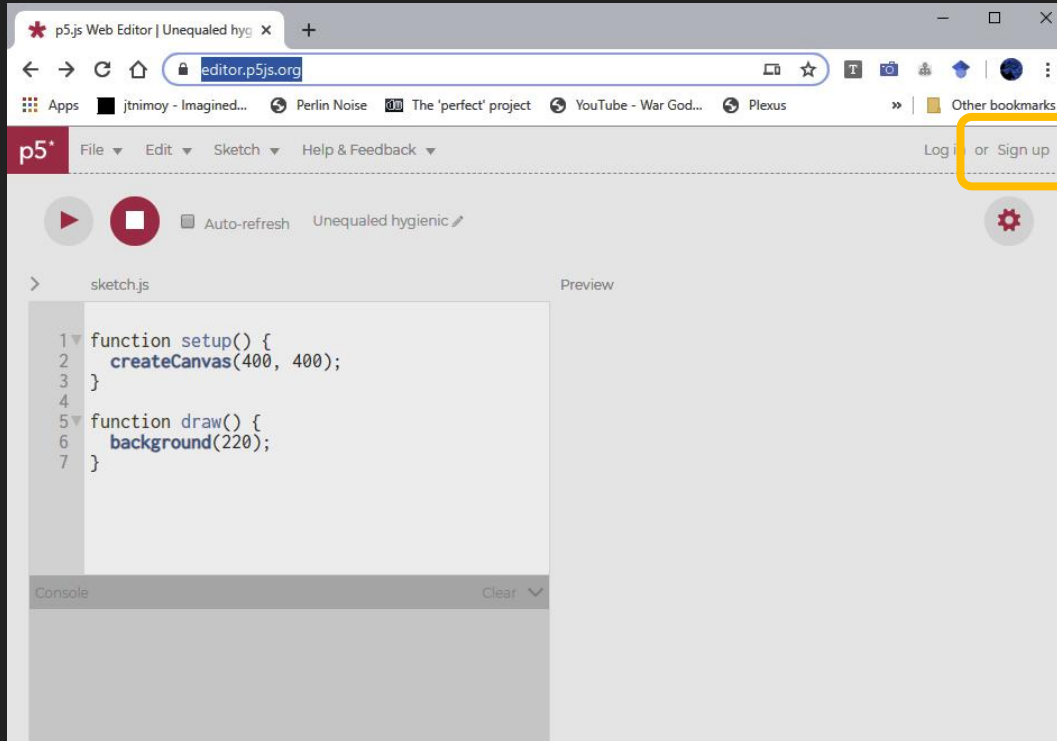
<https://canvas.cityu.edu.hk/courses/31242/pages/week-03#A>



# p5.js online editor

# p5.js online editor

p5\*



Please follow the URL and create an account so you can save your sketches, then do the exercise on the next slide.

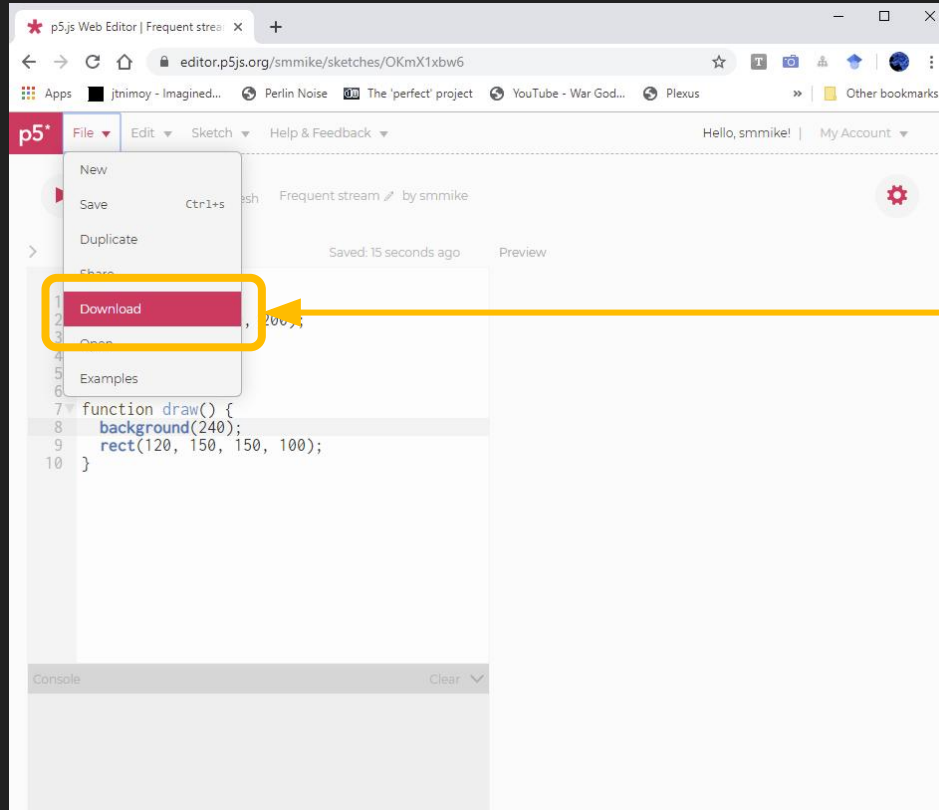
```
function setup() {  
  createCanvas(200, 200);  
}  
function draw() {  
  background(frameCount % 256);  
}
```

Modify this program to use **ONE** conditional and the modulo operator such that the background alternates every 60 frames, between black and white.

Don't change the frameRate

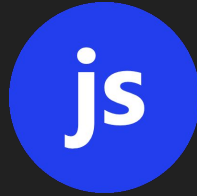
# p5.js online editor

p5\*



Once you have saved your sketch, you may download the sketch as a zipped archive.





# Logical Operators

# Logical operators

- logical operator helps us to compose more flexible 'conditions' for various JavaScript conditionals.
- All 'conditions' evaluation in JavaScript returns a logical (boolean) value 'true' or 'false'.

```
if (x == 200) {  
    // Do something  
}
```

Simple SINGLE condition,  
what if we want to  
combine two or more  
conditions ?

# Logical operators

operators	meaning
	Logical OR
&&	Logical AND
!	Logical NOT

```
if (x == 0 || x == 200) {  
    // Do something  
}
```

This block will run only if  
X equals to 0  
OR  
X equals to 200

# Logical operators

operators	meaning
	Logical OR
&&	Logical AND
!	Logical NOT

```
if (x > 1 && x != 200) {  
    // Do something  
}
```

This block will run only if  
X is larger than 1  
AND  
X is not equal to 200

# Logical operators

operators	meaning
	Logical OR
&&	Logical AND
!	Logical NOT

```
if (!(x > 1)) {  
    // Do something  
}
```

The NOT operator always inverses the result of the condition `(x > 1)`. So the block runs when `x` does not fulfill `(x > 1)`, i.e. the block runs when `x` is NOT larger than 1.

# Truth table

## Logical OR '||'

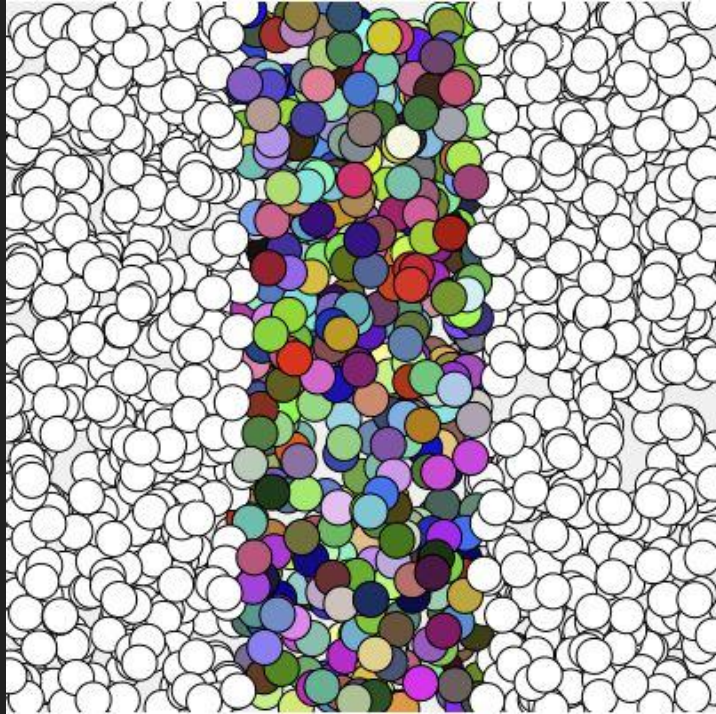
A	B	(A    B)
true	true	true
true	false	true
false	true	true
false	false	false

## Logical AND '&&'

A	B	(A && B)
true	true	true
true	false	false
false	true	false
false	false	false

## Logical NOT '!'

A	!(A)
true	false
false	true



1. Fill the canvas (400 x 400) with circles of size 20, each with a random position.
2. Divide the screen into 3 regions as shown in the figure. Circles in the middle region are randomly colored, and the rest are in white.
3. Use only one conditional. Hint: Use a 'logical operator'.



# Coding Style



# Coding Style

js

p5\*

```
let x=0;

function setup() {
  createCanvas(400, 400);
  background(220);
}

function draw() {
  if (x % 2 == 1) {
    fill(150);
  } else {
    fill(0);
  }
  ellipse(50 * x, 0, 50, 200);
  x++;
}
```

It is **VERY IMPORTANT** to use proper indentation and spacing in your code.

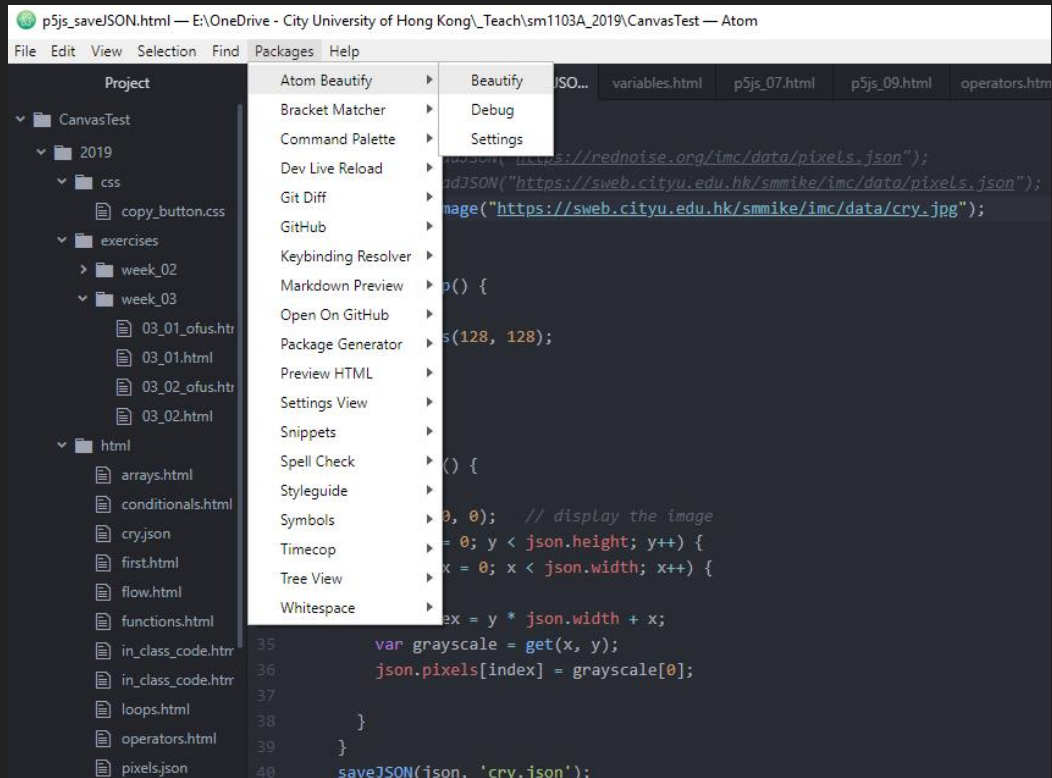
## WHY ?

- Easier for you to read
- Easier for others to read
- Help you to spot mistakes
- It shows you understand *the craft of coding*
-

# Coding Style

js

p5\*

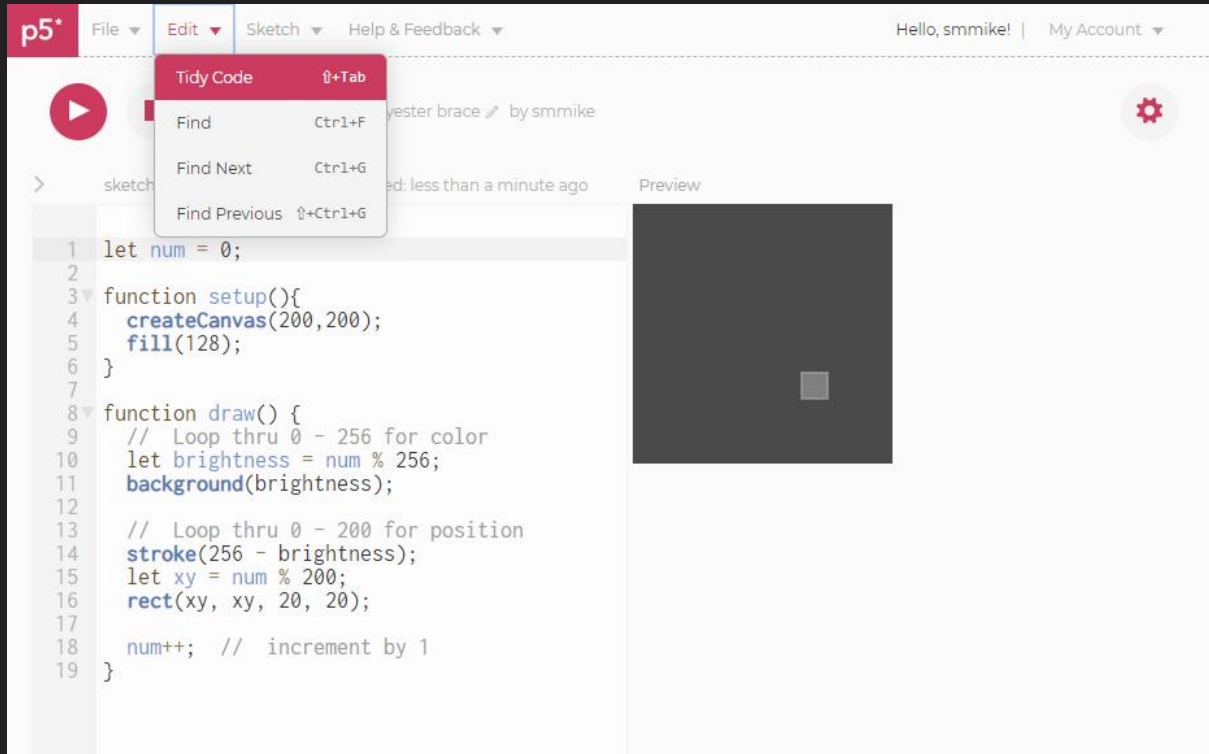


Most modern code editor like **Atom** often has code 'beautify' or 'prettify' package which helps with code formatting and syntax highlighting.

# Coding Style

js

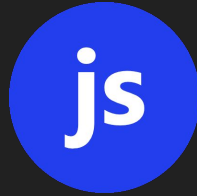
p5\*



**p5.js editor also offers convenient code formatting functions.**

Properly formatted code is  
required in all assignments.  
Poorly formatted code will  
lead to point deduction.





## Loops 1: The 'while()' loop

## Loops 1: the `while()` loop

js

p5\*

- A 'Loop' allows a block of code to be executed repeatedly (aka iteration).
- A `while()` loop repeats a block of code to as long as certain condition is fulfilled in each iteration.

## Loops 1: the `while()` loop

p5\*

