introduction to media computing
week 02

School of creative media, mw

today's topics (week 02)

- **variables & operators**
  - review
  - scope
  - naming
- **conditionals**
- **code comments**

today's topics (week 02)

- **variables & operators**
  - review
  - scope
  - naming
- **conditionals**
- **code comments**

- **built-in variables**
- **mouse interactivity**
- **drawing modes**

review: a basic p5.js sketch

```
<html>
<head>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/p5.js"></script>
  <script>

  function setup() {
    createCanvas(200, 200);
    rect(120, 150, 200, 300);
  }

  </script>
</head>
</html>
```

review: a basic p5.js sketch

open HTML tags

```
<html>
<head>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/p5.js"></script>
  <script>

  function setup() {
    createCanvas(200, 200);
    rect(120, 150, 200, 300);
  }

  </script>
</head>
</html>
```

close HTML tags

review: a basic p5.js sketch

```
<html>
<head>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/p5.js"></script>
  <script>

    function setup() {
      createCanvas(200, 200);
      rect(120, 150, 200, 300);
    }

  </script>
</head>
</html>
```

p5.js library from CDN

your code here

review: basic of p5.js

```
function setup() {
  createCanvas(200, 200);
}

function draw() {
  background(random(0,255));
}
```

review: basic of p5.js

```
function setup() {
  createCanvas(200, 200);
}

function draw() {
  background(random(0,255));
}
```

setup()

runs ONCE ONLY

review: basic of p5.js

```
function setup() {
  createCanvas(200, 200);
}
```

setup()

runs ONCE ONLY

```
function draw() {
  background(random(0,255));
}
```

draw()

LOOPS FOREVER

Variables & Operators

Variables

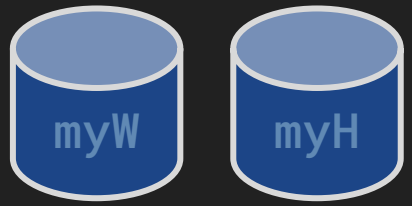
```
let myW;
let myH;
myW = 200;
myH = 150;

function setup() {
  createCanvas(500, 500);
  rect(10, 10, myW, myH);
}
```

Variables

```
let myW;
let myH;
myW = 200;
myH = 150;

function setup() {
  createCanvas(500, 500);
  rect(10, 10, myW, myH);
}
```

myW and myH

are called *Variables*



DECLARATION

let is the keyword

for creating variables

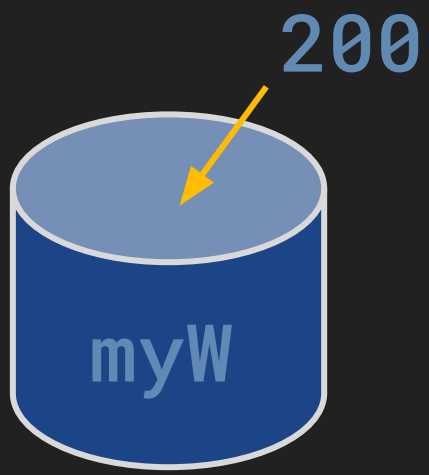
Variables

```
let myW;
let myH;
myW = 200;
myH = 150;

function setup() {
  createCanvas(500, 500);
  rect(10, 10, myW, myH);
}
```

Variable is a *named* container of DATA. The data it stores can be used and changed.

Variables

```
let myW;
let myH;
myW = 200;
myH = 150;

function setup() {
  createCanvas(500, 500);
  rect(10, 10, myW, myH);
}
```

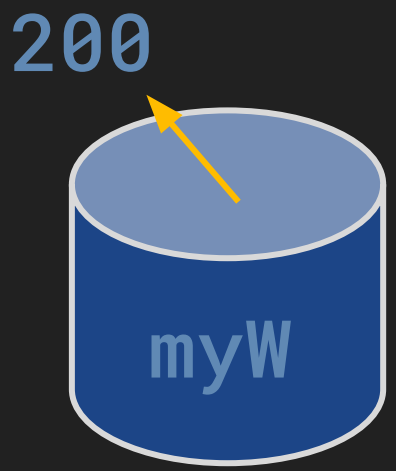


ASSIGNMENT

Data value can be assigned to them

Variables

```
let myW;
let myH;
myW = 200;
myH = 150;

function setup() {
  createCanvas(500, 500);
  rect(10, 10, myW, myH);
}
```



Variable supplies the data (value) when requested.

Variables

```
let myW = 200;
let myH = 150;

function setup() {
  createCanvas(500, 500);
  rect(10, 10, myW, myH);
}
```

We may declare the variables and assign initial values to them at the same time.

Variables: Assignment

Assign the value on the RIGHT to the variable on the LEFT

using the Assignment Operator `'='`

```
let x = 10;
x = 20;
x = x + 20;
```

Variables: Assignment

Assign the value on the RIGHT to the variable on the LEFT

using the Assignment Operator '='

```
let x = 10;
x = 20;
x = x + 20;
```

X equals to 10

Variables: Assignment

Assign the value on the RIGHT to the variable on the LEFT

using the Assignment Operator '='

```
let x = 10;
x = 20;
x = x + 20;
```

X becomes 20

Variables: Assignment

Assign the value on the RIGHT to the variable on the LEFT

using the Assignment Operator `'='`

```
let x = 10;
x = 20;
x = x + 20;
```

X becomes ?

Variables: Assignment

Assign the value on the RIGHT to the variable on the LEFT

using the Assignment Operator `'='`

```
let x = 10;
x = 20;
x = x + 20;
```

X becomes 40

WHY ?

Variables: Assignment

```
x = 20;
x = x + 20;
```

The value of RIGHT hand side is ALWAYS evaluated first.

```
20 + 20
x = 40
```

More Assignment Operators

```
let x = 200;
x += 100;
```

`'+='`: **Addition assignment**

`x += 100` **is equivalent to** `x = x + 100`

```
let x = 200;
x -= 100;
```

`'-='`: **Subtraction assignment**

`x -= 100` **is equivalent to** `x = x - 100`

More Assignment 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`

Mathematical Operators

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

result = a + b;
result = a - b;
result = a * b;
result = a / b;
result = a % b;
```

Addition	`result = 16`
Subtraction	`result = 4`
Multiplication	`result = 60`
Division	`result = 1.6667`
Modulo	`result = 4*`

***Remainder of integer division**

Variable Naming Rules

- Variable name may consist of characters, digits and underscore `'_'`
- Variable names cannot start with numbers

```
let shape_02 = 6;
let rectHeight = 6;
let let = 6;
let 123abc = 10;
```

OKAY

OKAY

ERROR `let` is a reserved keyword

ERROR Cannot start with numbers

Variable Naming Style

- Choose descriptive and meaningful name which reflects the variable's role in the code
- Choose the names such that they help others to understand your code

```
let abc1 = 6;
let alice = 100;
let i,j,k = 0;
let maxValue = 10;
```

OKAY, but meaning is unclear

OKAY, but does 'alice' refer to anything ?

OKAY, if for general loops

OKAY

Code Comments '//'or '/*...*/'

- Comments lines are automatically ignore by the computer
- Comments are used in-code texts which document the purpose the individual variables or code segments

```
// This is a one-line comment
let oneLine = 100;
/* This is a multi-line
   comment block. */
```

Built-in Variables

Built-in Variables

```
function draw() {
  background(255);

  text(width, 100, 100);    // width: canvas width
  text(height, 100, 120);   // height: canvas height

  // mouseX, mouseY: (x,y) of the mouse cursor in the canvas
  text("mouse x: " + mouseX, 200, 200);
  text("mouse y: " + mouseY, 200, 220);

  // frameCount: no. of frames drawn since the program started
  text(frameCount, 200, 230);
}
```

Built-in Variables

```
function draw() {
  background(255);

  text(width, 100, 100);    // width: canvas width
  text(height, 100, 120);   // height: canvas height

  // mouseX, mouseY: (x,y) of the mouse cursor in the canvas
  text("mouse x: " + mouseX, 200, 200);
  text("mouse y: " + mouseY, 200, 220);

  // frameCount: no. of frames drawn since the program started
  text(frameCount, 200, 230);
}
```

Built-in Variables

```
function draw() {
  background(255);

  text(width, 100, 100);    // width: canvas width
  text(height, 100, 120);   // height: canvas height

  // mouseX, mouseY: (x,y) of the mouse cursor in the canvas
  text("mouse x: " + mouseX, 200, 200);
  text("mouse y: " + mouseY, 200, 220);

  // frameCount: no. of frames drawn since the program started
  text(frameCount, 200, 230);
}
```

Built-in Variables

try p5*

JS

```
function setup() {
  createCanvas(250,250);
}

function draw() {
  background(100);
  fill(255,255,0);
//  ellipse(width/2,height/2,40,40);
  ellipse(mouseX,mouseY,20,20);
}
```

Result

EDIT ON CODEPEN

Resources

1x 0.5x 0.25x

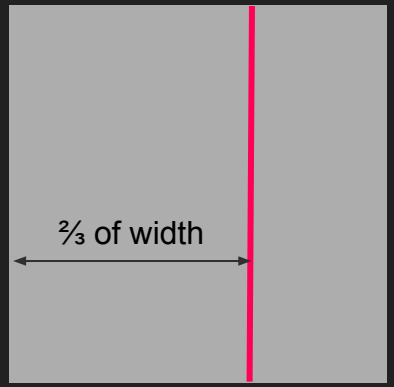
Rerun

exercise

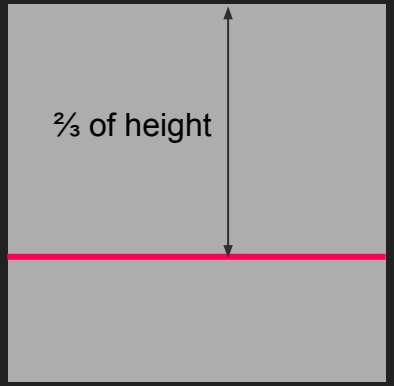
In-class Exercise 1

p5*

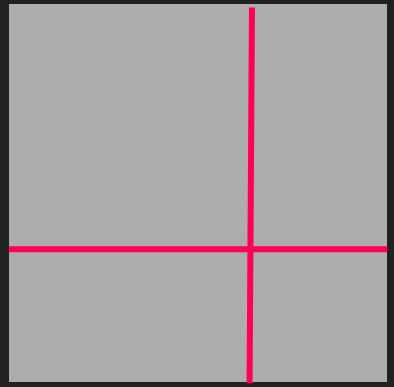
A



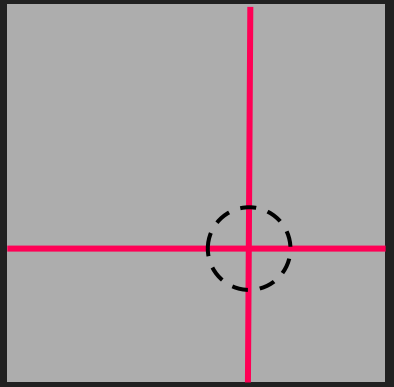
B



C



D



1. Use `stroke()` and `line()` to draw a red line as shown in figure A
2. Use `stroke()` and `line()` to draw the red line as shown in figure B
3. Combine results from step 1 and 2 to as shown in figure C
4. Modify your code with built-in variables `mouseX` and `mouseY` so that the lines always intersect at your mouse cursor (and move accordingly) as shown in figure D.

Variable Scope

Scope of Variables

```
let rectW = 400;
let rectH = 500;

function setup() {
  let foo = 1.234;
  createCanvas(800, 800);
  rect(10, 10, rectW, rectH);
}

function draw() {
  let w = 200;
  let h = 300;
  background(255);
  rect(10, 10, w, h);
}
```

'Scope'

Manages the visibility of variables to different parts of the program.

Scope of Variables

```
let rectW = 400;
let rectH = 500;

function setup() {
  let foo = 1.234;
  createCanvas(800, 800);
  rect(10, 10, rectW, rectH);
}

function draw() {
  let w = 200;
  let h = 300;
  background(255);
  rect(10, 10, w, h);
}
```

GLOBAL scope

Variables (`rectW`, `rectH`) defined at this level are accessible (visible) everywhere in the program.

Scope of Variables

```
let rectW = 400;
let rectH = 500;

function setup() {
  let foo = 1.234;
  createCanvas(800, 800);
  rect(10, 10, rectW, rectH);
}

function draw() {
  let w = 200;
  let h = 300;
  background(255);
  rect(10, 10, w, h);
}
```

GLOBAL scope

Variables (`rectW`, `rectH`) defined at this level are accessible (visible) everywhere in the program.

Local scope A

Variables (`foo`) defined here are only accessible (visible) to this block of code. A block is defined by the pair of parentheses { ..... }).

Scope of Variables

```
let rectW = 400;
let rectH = 500;

function setup() {
  let foo = 1.234;
  createCanvas(800, 800);
  rect(10, 10, rectW, rectH);
}

function draw() {
  let w = 200;
  let h = 300;
  background(255);
  rect(10, 10, w, h);
}
```

GLOBAL scope

Variables (`rectW, rectH`) defined at this level are accessible (visible) everywhere in the program.

Local scope A

Variables (`foo`) defined here are only accessible (visible) to this block of code. A block is defined by the pair of parentheses { ..... }).

Local scope B

Variables (`w, h`) defined here are only accessible (visible) to this block of code.

Review of `random()` and `rect()`

Built-in function random()

- random() serves as a convenient means to make up some numbers in an arbitrary manner.
- random() takes 0, 1 or 2 parameters.

```
let result_0 = random();
// result_0: a random number from 0 to 0.999

let result_1 = random(10);
// result_1: a random number from 0 to 9.999

let result_2 = random(-10, 10);
// result_2: a random number from -10 to 9.999
```

Built-in function `random()`

try p5*

JS

```
function setup() {
  createCanvas(200, 200);
}

function draw() {
  background(255);
  let w = random(10, 100);
  let h = random(10, 120);
  let x = width / 5;
  let y = height / 5;
  rect(x, y, w, h);
}
```

Result

EDIT ON CODEPEN

Resources

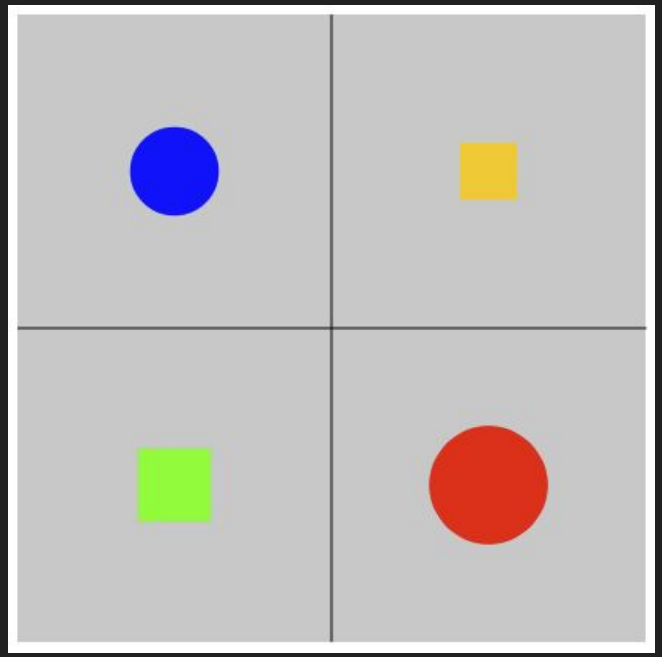
1× 0.5× 0.25×

Rerun

exercise

In-class Exercise 2

p5*



https://p5js.org/reference/

1. Divide a square shaped canvas into 4 equal quadrants. In each region, draw a square or circle as shown in the figure. Each shape must be in the center of its region. HINT : you need to use variables for their sizes, and you may want to use `rectMode(CENTER)` and `ellipseMode(CENTER)` too.

2. Vary the size of each shape using the `random()` function.

3. Now vary the size of the shapes at most by 1-pixel per frame, i.e. their sizes change GRADUALLY. Keep color unchanged.

Conditionals

Conditional: if-else

```
let x;

x = 200;

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

'Conditionals'

A construct which allows code execution only when certain condition is met.

Conditional: if-else

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

Conditional: 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

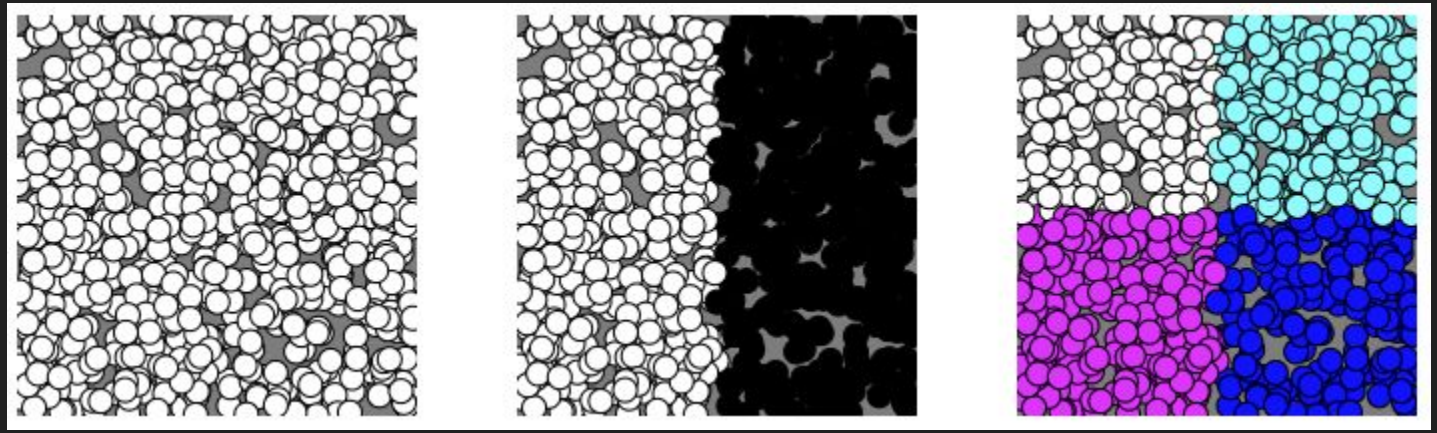
Conditional: else if

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

exercise

In-class exercise 3

p5*



1. Fill the canvas with small WHITE circles where each has a random position.

2. Modify your code such that the circles located on the right hand side are now colored in BLACK.

3. Partition the canvas into four regions as above. In each region, circles are colored differently. HINT : relate color (R,G,B) to the regions.