CS106 W20
Module 1

Introduction to JavaScript p5

and

Recap of CS105

JavaScript p5

- Most students took CS105 in F19 using JavaScript p5
- Some students took CS105 in a previous semester using Processing
- This week in the lectures, labs, and assignment we recap CS105 with both groups taken into consideration
  - Introduction to JavaScript p5
  - Recap of CS105 topics

JavaScript p5

- JavaScript is a programming language
- p5 is a JavaScript library (p5.js)
- We create and edit JavaScript p5 files using the Processing IDE
- We debug JavaScript p5 code using your browser's debugger
  - Also called "debugger console"

Processing IDE

- Integrated Development Environment
  - A JavaScript p5 editor and more
- Download from: https://processing.org/download/
- Must use p5.js mode (upper right corner)

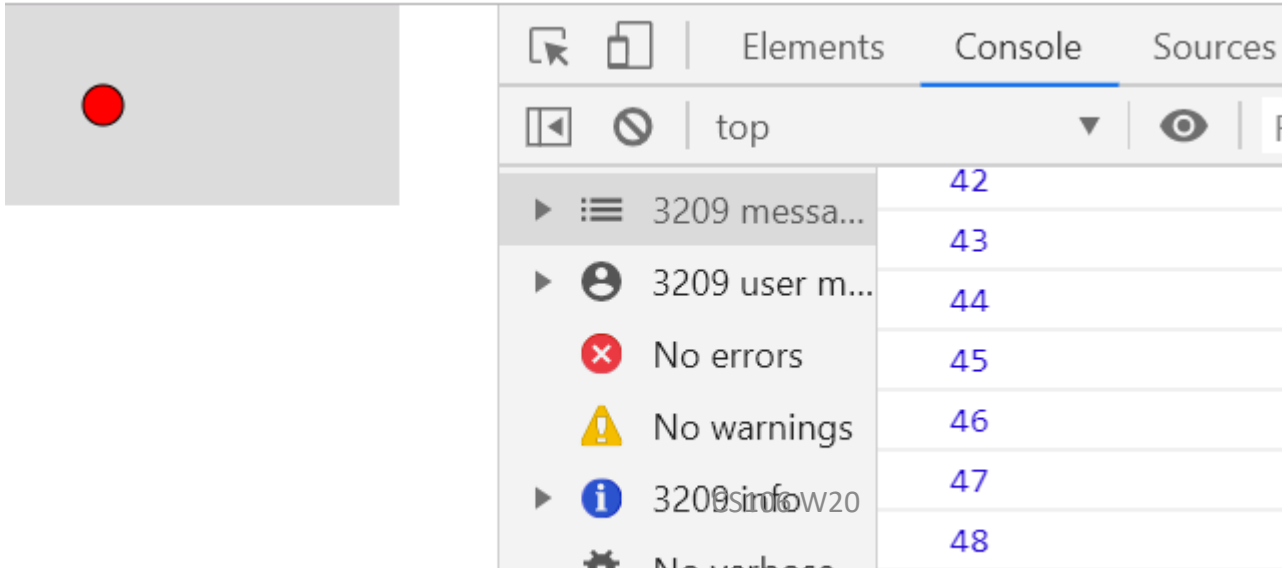
P5.js mode

Processing IDE

- Autoformat
  - CTRL + T
- Comment/uncomment each highlighted line
  - CTRL + /
- File/Save creates a directory and saves several files
  - A library directory
  - Your .js file
  - index.html
  - A sketch properties file
  - … and sometimes more
- Use the CS105 CS106 Code Style Sheet

The Chrome Debugger

- Must first run your JavaScript p5 sketch
- Then, to open the developer **console** window **on Chrome**…
  - Use the keyboard **shortcut** Ctrl Shift J (**on** Windows) or
  - Ctrl Option J (**on** Mac) or
  - Using the **Chrome** menu, select "More Tools," and then "**Developer Tools**."



Demo of Processing IDE and Debugger

```
let ballX = 0;
let s = 20;
function setup() {
  createCanvas(200, 100);
}

function draw() {
  background(220);
  fill(255, 0, 0);
  ellipse(ballX, height / 2, s, s);
  ballX = ballX + 1.0;
  print(ballX);
}

function mousePressed() {
  ballX = 0;
}
```

Review of topics from CS105

- CS105 CS106 Code Style Sheet
- Variables
- Conditions
- Loops
- Functions
- Arrays
- Program design
- Images
- Sound & Video

Variables

- Built-in (also called System) variables
  - width, mouseX
- Constants (all caps)
  - CENTER, PI, RIGHT
- User-defined variables
  - rectSize, count, i

User-defined variables

- Declared using “let”, In CS105/106, all variables must be declared.
- Variables in JavaScript p5 are not directly associated with any particular type, and any variable can be assigned (and re-assigned) values of all types such as:
  - integer, float, Boolean, string, color, and more

```
let b = 10;
let size = 10.3;
let gameOn = true;
let city = "Waterloo";
let ballColour = color(225, 0, 0);
```

Demo of Data Types and Functions

```
let rectSize = 30;

function setup() {
  createCanvas(300, 300);
}

function draw() {
  background(220);
  rectMode(CENTER);
  rect(width / 2, height / 2, rectSize, rectSize);
}

function keyPressed() {
  rectSize = random(20, 100);
}
```


iclicker

Conditionals

```
let ballX = 0;

function setup() {
  createCanvas(200, 100);
}

function draw() {
  background(220);
  ellipse(ballX, height / 2, 20, 20);
  ballX = ballX + 1.0;
  if (ballX > width) {
    ballX = 0;
  }
}
```

Conditionals using &&

```
function draw() {
  background(220);

  if (keyIsPressed && key === " ") {
    ellipse(mouseX, mouseY, 20, 20);
  }
}
```

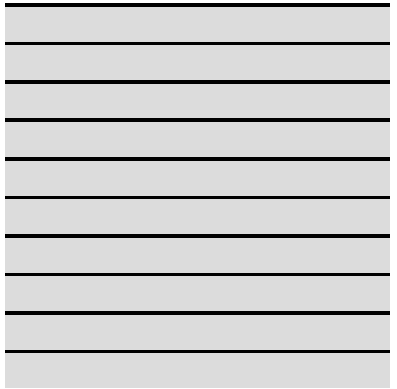
Conditions with an “else” clause

```
function draw() {
  background(220);
  if (keyIsPressed && key === ' ') {
    ellipse(mouseX, mouseY, 20, 20);
  } else {
    rect(mouseX, mouseY, 20, 20);
  }
}
```

Nested Conditions

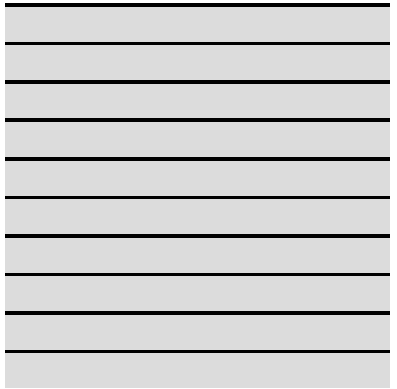
```
function draw() {
  background(220);

  if (keyIsPressed) {
    if (key === 'e') {
      ellipse(mouseX, mouseY, 20, 20);
    } else if (key === 'l') {
      line(10, 10, 100, 100);
    } else {
      rect(mouseX, mouseY, 20, 20);
    }
  }
}
```

While Loop



```
function draw() {
  background(220);
  let y = 0;
  while ( y < height ) {
    line( 0, y, width, y );
    y = y + 10;
  }
}
```

For Loop



```
function draw() {
  background(220);
  for (let y = 0; y < height; y += 10) {
    line(0, y, width, y);
  }
}
```

Built-In Functions

- setup()
- draw()
- keyPressed()
- mousePressed()
- Some built-in functions have parameters and return a value
  - random(2, 10)
  - dist(x1, y1, x2, y2)

User-Defined Functions

- Give a name to a block of code
- Benefits
  - Easy of reuse
  - Encapsulation – hides the messy details
  - Abstraction – think about problem solving at a higher level
  - Establish a point of connection between parts of your program
- Must be defined using “function”
  - function myFunc() {
- May have parameters
- May return a value

Demo of Functions

```
let hit;

function draw() {
  background(220);
  ellipse(width / 2, height / 2, 30, 30);
  hit = circleHittest(mouseX, mouseY, width / 2, height / 2, 30);
  text(hit, 10, 10);
}

function circleHittest(x1, y1, cx, cy, s) {
  return (dist(x1, y1, cx, cy) < s / 2);
}
```

Arrays

- A sequence of values
- For example, monthly max temperatures
  - Ontario

Month	Max Temp
Jan	-1
Feb	0
Mar	5
Apr	12
May	18
Jun	24
Jul	27
Aug	26
Sep	21
Oct	14
Nov	8
Dec	2

Arrays – Loop through an array of Strings

```
let month = ["Jan", "Feb", "Mar", "Apr",
  "May", "Jun", "Jul", "Aug",
  "Sep", "Oct", "Nov", "Dec"];

function setup() {
  createCanvas(100, 400);
  background(220);
  textSize(25);
  for (let i = 0; i < month.length; i++) {
    text(month[i], 10, (i * 30) + 30);
  }
}
```

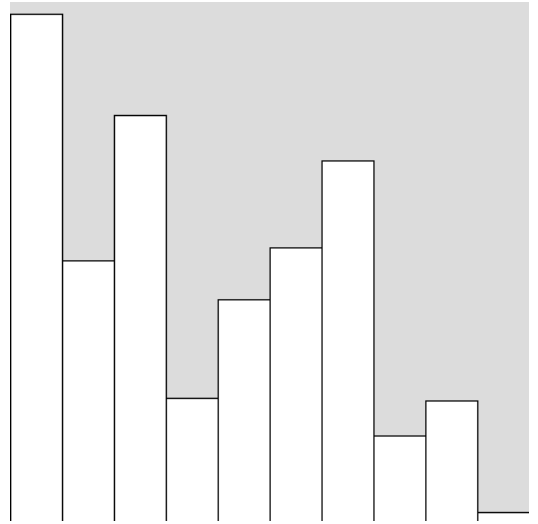
Find the Largest in an Array

```
let hiMonth = [-1, 0, 5, 12, 18, 24, 27, 26, 21, 14, 8, 2];
function setup() {
  createCanvas(400, 100);
  background(220);
  textSize(25);
  let largest = hiMonth[0];
  for (let i = 1; i < hiMonth.length; i++) {
    if (hiMonth[i] > largest) {
      largest = hiMonth[i];
    }
  }
  text("Temperature of hottest month: " + largest, 10, 50);
}
```

Initialize a large Array

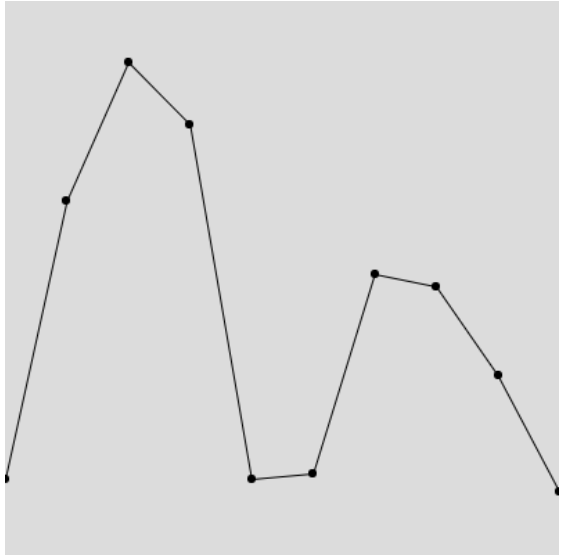
```
let arr = [];
let numBars = 10;

function setup() {
  createCanvas(400, 400);
  background(220);
  for (let i = 0; i < numBars; i++) {
    arr[i] = floor(random(0, height));
  }
```

Visualize the data as a Bar Graph



```
let arr = []; // declare array
let barWidth;
let numBars = 10;

//… setup() goes here, as on previous slide …

  barWidth = width / numBars;
  for (let i = 0; i < arr.length; i++) {
    rect(i * barWidth, height - arr[i], barWidth, arr[i]-1);
  }
}
```

Visualize the same data as a line graph



```
for (let i = 0; i < numBars; i++) {
   arr[i] = floor(random(0, height));
 }
 pointsWidth = width / (numBars - 1);
 for (let i = 0; i < arr.length; i++) {
   // draw the points
   let x = i * pointsWidth;
   let y = height - arr[i];
   strokeWeight(6);
   point(x, y);
   // draw the connecting lines
   if (i > 0) {
     let px = (i - 1) * pointsWidth;
     let py = height - arr[i - 1];
     strokeWeight(1);
     line(px, py, x, y);
   }
 }
```

Snake: Modular Design

```
let snakeX = [];
let snakeY = [];

function setup() {
  createCanvas(500, 500);
  initialization();
}

function draw() {
  background(220);
  updateSnake();
  drawSnake();
}
```

Snake: Modular Design con't

```
function initialization() {
  for (i = 0; i < 50; i++) {
    snakeX[i] = 0;
    snakeY[i] = 0;
  }
}
```

Snake: Modular Design con't

```
function updateSnake() {
  for (let i = 0; i < snakeX.length - 1; i++) {
    snakeX[i] = snakeX[i + 1];
    snakeY[i] = snakeY[i + 1];
  }
  snakeX[snakeX.length - 1] = mouseX;
  snakeY[snakeY.length - 1] = mouseY;
}

function drawSnake() {
  for (let i = 0; i < snakeX.length; i++) {
    ellipse(snakeX[i], snakeY[i], i, i);
  }
}
```

Creating a Timer

- Let’s say we want something to happen every three seconds
- Using snake from previous slides
- Change fill every 3 seconds
- Built-in function millis()
  - Returns # of milliseconds since program started

Demo: Snake with a Timer to change Font

- Add these variables

```
let savedTime;
let changeFillTime = 3000;
```

- Initialize saveTime to the current millis()

```
savedTime = millis();
```

- Determine when 3 seconds have passed

```
if (millis() - savedTime > changeFillTime) {
    fill(random(255), random(255), random(255));
    savedTime = millis();
  }
```

Demo: Snake ends after 10 seconds

- Add variables

```
let gameoverTime = 10000;
let gameOn = true;
```

- Check to determine if 10 seconds have passed

```
if (millis() > gameoverTime) {
   gameOn = false;
 }
```

Demo: Snake ends after 10 seconds con't

- Modify draw()

```
if (gameOn) {
   background(220);
   updateSnake();
   drawSnake();
 }
```

Demo: Animated

```
let counter = 0;

function setup() {
  createCanvas(500, 500);
}
```

Demo: Animated

```
function draw() {
  background(0);
  fill(200);
  stroke(255);
  strokeWeight(3);

  counter = counter + 1;
  if (counter === 61) {
    counter = 0;
  }
```

Demo: Animated

```
let aa = map( counter, 0, 60, 100, 400 );

  ellipse( aa, 100, 80, 80 );
  ellipse( 100, 500 - aa, 80, 80 );
  ellipse( 500 - aa, 400, 80, 80 );
  ellipse( 400, aa, 80, 80 );
}
```


iclicker

What is printed to the console ?

What is printed to the console ?

What is printed to the console ?

What is printed to the console?

What is printed to the console?

What does this draw after 1,000,000 frames?

What does this draw after 1,000,000 frames?