

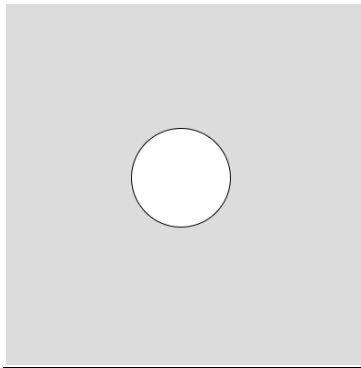
## Lab 6 – Objects Exercises

<https://rkkuhn.github.io/N220Spring2023/>

### Algorithm 1: Animated Object

**Directions:** Create an array with multiple objects with properties for a circle's x, y, and size. In the draw function, loop through the array, change the position of each object by 1 and draw it to the screen.

#### Expected Results:



The circle will move diagonally along the screen while increasing in size.

#### Sudo Code:

HTML – standard, nothing special from what we have been practicing.

JS

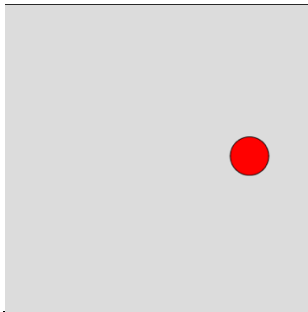
- Create a blank variable labeled circle
- Function setup
  - Create canvas (400, 400)
- Create the properties for the circle
  - Circle = {
    - X: 75,

- Y: 75,
  - Size: 50
- };
- Start the function draw()
  - Function draw(){
    - Background(220);
  - Create the properties for the circle that will increase x and y += 1 and increase the size +=0.5
- Draw the circle using ellipse command
- Run the sketch

## Algorithm 2: Bouncing Ball Refactor

**Directions:** Take your ball bounce assignment and recode it using an object to store the ball's size, color, and velocity. No global variables or hardcoded values should be used in your update function.

### **Expected Results:**



The red circle will bounce off the sides and continue to move around the light gray box Areas.

### **Sudo Code:**

HTML – standard, with the following added

- Code for style in the html so I can call it from the JS side.
  - Display = flex
  - Justify – content = center
  - Align-items = center
  - Height = 100vh
  - Margin = 0

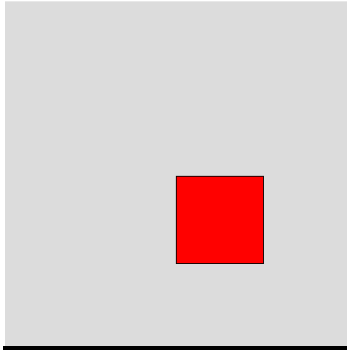
JS

- Create a blank variable labeled ball
- Use the function setup
  - Create the canvase
  - Create the ball object
    - X is width / 2
    - Y is height / 2
    - Color is color(red)
    - Create random movement using velocity on the x and y axis
      - VelocityX = random (-5, 5)
      - VelocityY = random (-5, 5)
- Function draw()
  - Background – light gray
    - Update and display ball
      - updateBall(ball)
      - displayBall(ball)
- Create the function to updateBall
  - Update the ball's position based on velocity
    - Ball.x += ball.velocityX
    - Bally.y += ball.velocityY
    - Use two if statements to check to see if the ball hits the edge of the canvas.
      - One is for the x coordinates
      - The other is for the y coordinates
- Final Junction to display ball
  - Function displayBall(ball)
    - Fill (ball.color)
    - Ellipse(ball.x, ball.y, ball.size)

### Algorithm 3: Data Driven Display

**Directions:** Make an object with all the properties for a graphic on the screen - width, height, color, position. In your create or draw function draw that object using the data in your object.

**Expected Results:**



The object with all the properties for a graphic is the red box on the screen

### **Sudo Code:**

JS

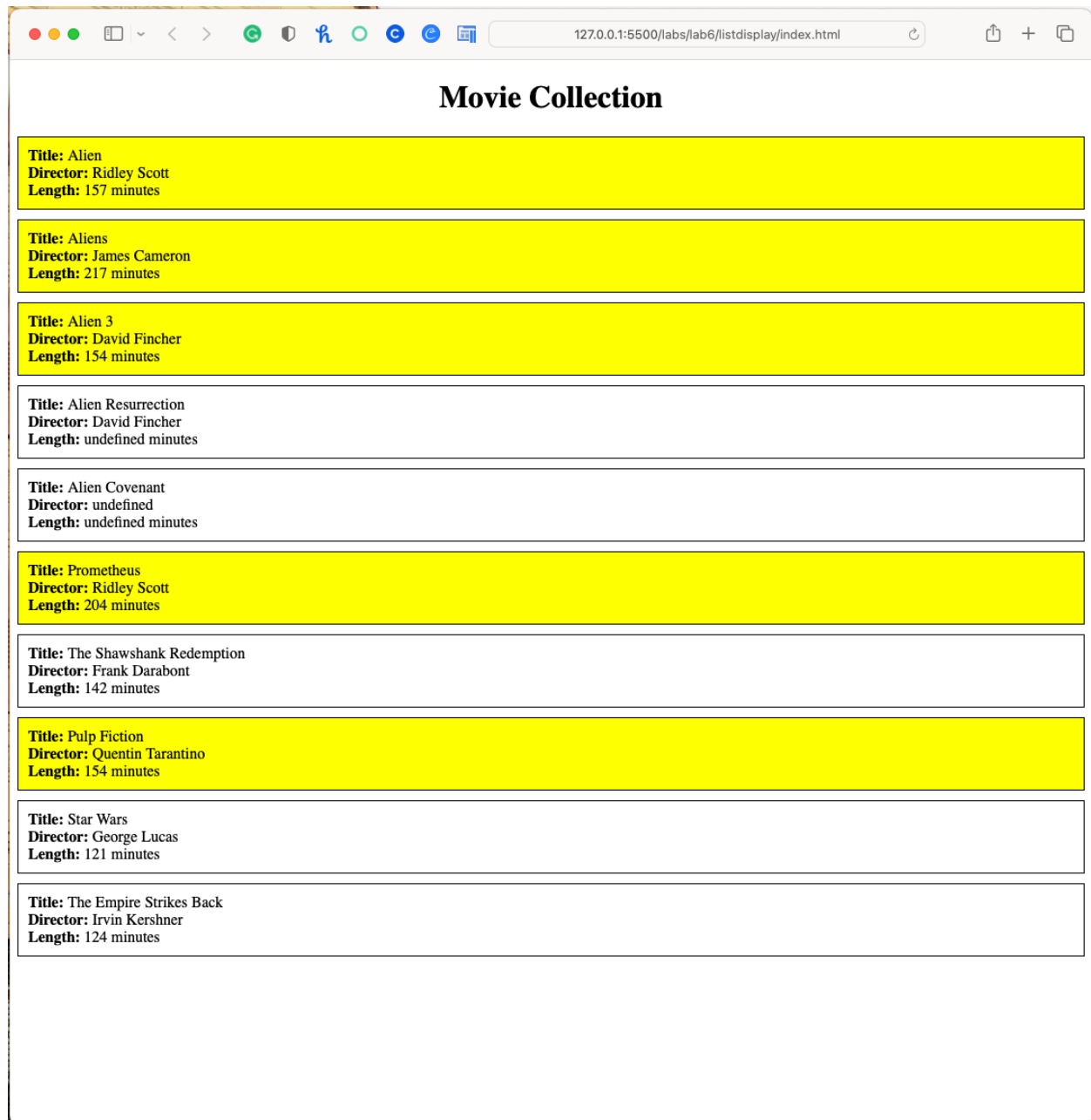
- Create a global let statement = graphic
- Start the function setup
  - Create the canvas 400 by 400
  - Initialize the graphic object
    - Width = 100
    - Height = 100
    - Color = red
    - X = 200
    - Y = 200
- Start the draw function
  - Background color = gray-ish
  - Draw the graphic using the object's properties
    - Fill (graphic.color)
    - Rect(graphic.x, graphic.y, graphic.width, graphic.height)

### **Algorithm 4: List Display**

**Directions:** Create an HTML5 application with a collection (array of objects) in JavaScript that with entries for something you have a collection of (movies, books, rocks, bugs.. what have you). Each object must have at least 3 properties. Use a loop in JavaScript to loop through those elements, and create a div for each that display the data, place that element on the page. Include in this loop at least one IF statement that changes an element's style based on some property. For instance, perhaps if a bug's length property is greater than 5cm, you set the

background color of that element to yellow. Your data must be set up in a way to force this change to happen for at least one of your elements.

### Expected Results:



<b>Title:</b> Alien <b>Director:</b> Ridley Scott <b>Length:</b> 157 minutes
<b>Title:</b> Aliens <b>Director:</b> James Cameron <b>Length:</b> 217 minutes
<b>Title:</b> Alien 3 <b>Director:</b> David Fincher <b>Length:</b> 154 minutes
<b>Title:</b> Alien Resurrection <b>Director:</b> David Fincher <b>Length:</b> undefined minutes
<b>Title:</b> Alien Covenant <b>Director:</b> undefined <b>Length:</b> undefined minutes
<b>Title:</b> Prometheus <b>Director:</b> Ridley Scott <b>Length:</b> 204 minutes
<b>Title:</b> The Shawshank Redemption <b>Director:</b> Frank Darabont <b>Length:</b> 142 minutes
<b>Title:</b> Pulp Fiction <b>Director:</b> Quentin Tarantino <b>Length:</b> 154 minutes
<b>Title:</b> Star Wars <b>Director:</b> George Lucas <b>Length:</b> 121 minutes
<b>Title:</b> The Empire Strikes Back <b>Director:</b> Irvin Kershner <b>Length:</b> 124 minutes

List of movies displayed and movies over 150 minutes are highlighted in yellow.

### Sudo Code:

HTML code is almost the same as we have been using, except I used a .css style here to reference creating the boxes and highlighting movies.

## JS

- Define the movies collection using var movies
- Loop through the movies and create div elements for each
  - Var movieListDiv = document.getElementById("movieList")
  - Create a for loop to go through the movie list
    - Create a new div for the movie
      - Var movieDiv = Document.createElement("div")
      - movieDiv.className = "movie"
    - Create the innerHTML layout for the movies on the page
      - Title of the movie
      - Director
      - Length
    - Using an nested if statement apply the highlight class if the length is > 150 minutes
    - Append the child to the movieDiv

**Reflection:** Normally, I can do the programming for your class with minimum Google searches for answers. This time, I did spend time reviewing the lectures and burning up some bandwidth on Google. With the Virtual Pet, I tried to program it as I did in an assignment under Lab 5 using HTML first. Then Once I got it working in HTML, I could spot the code that needed to be moved to app.js. This time, I do not see the pattern. I have tried moving several pieces of code, and it still works in HTML but not with JS. I have spent a ton of time on the Virtual Pet, but I haven't had much time to get the Battle System working besides the HTML layout. I like that the homework labs are starting to challenge me more. When I research, I learn more and more about JS and P5. An example of stumbling across something new was the Viewport. In Ball Bounce, I was doing research on getting it to work when I found information about Viewport and wanted to play with it. I know we are supposed to use pixels, but I wanted to see if it would work and how it works. I am looking forward to the next lab.

**Note: I am still working on Virtual Pet and Battle System.**