Topic	MULTIPLE CANNONBALLS	
Class Description	Students will learn about the 2D array and ways to iterate on an array. Students will also learn to create a new cannonball and shoot them from the cannon at the press of a button.	
Class	C24	
Class time	45 mins	
Goal	• Learn about the 2D arrays. • Create a new cannonball on the press of the down arrow key.	
Resources Required	• Teacher Resources ○ Laptop with internet connectivity ○ Visual Studio Code ○ Earphones with mic ○ Notebook and pen • Student Resources ○ Laptop with internet connectivity ○ Visual Studio Code ○ Earphones with mic ○ Notebook and pen	
Class structure	WARM-UP Teacher-led Activity Student-led Activity WARM-UP	5 mins 10 mins 25 mins 5 mins

WARM-UP SESSION - 5mins

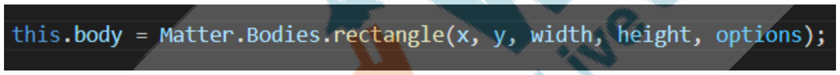
CONTEXT

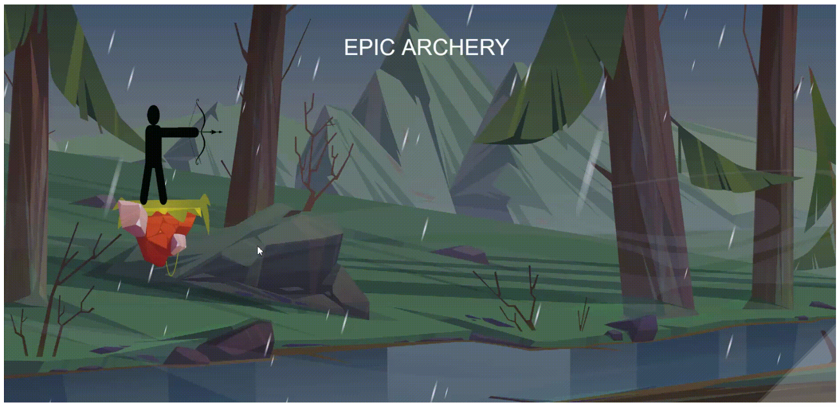
- **Review the concept of class covered in the earlier class.**
- **Talk about the number of chances a player is given in the Pirates Invasion game.**

Teacher starts slideshow from slides 1 to 8
Refer to speaker notes and follow the instructions on each slide.

Activity details	Solution/Guidelines
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Run the presentation from slide 1 to slide 4.** **Following are the WARM-UP session deliverables:** ● Greet the student. ● Revision of previous class activities. ● Quizzes	**ESR**: Hi, thanks, yes I am excited about it! Click on the slide show tab and present the slides

QnA Session

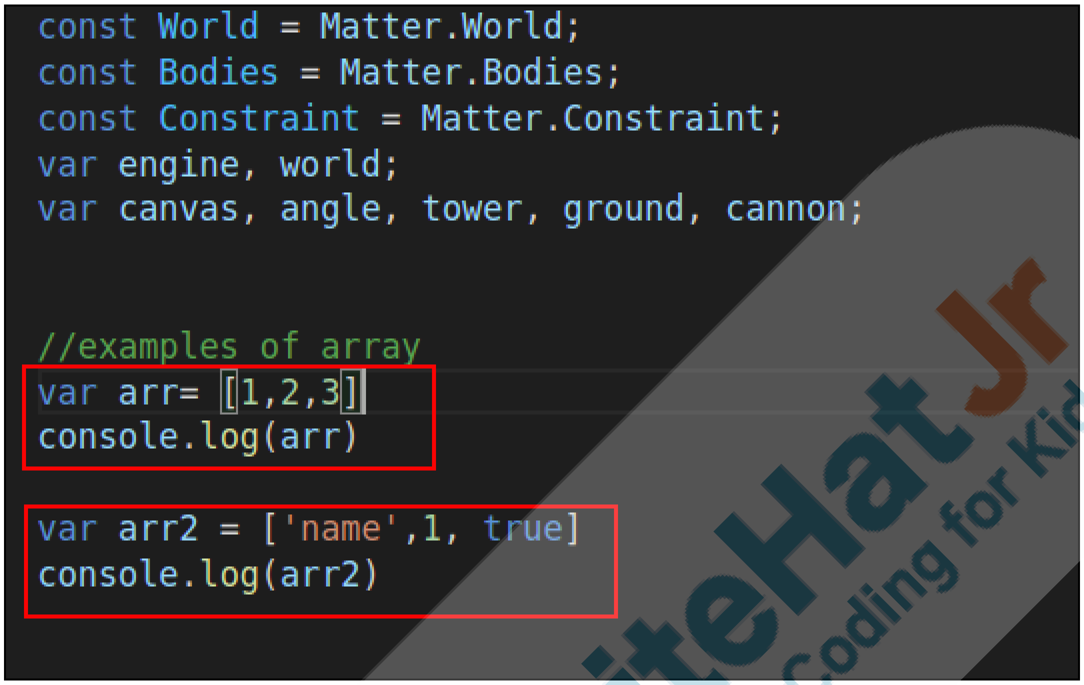
Question	Answer
What will the following block of code do?  `this.body = Matter.Bodies.rectangle(x, y, width, height, options);` A. It will create a circular shaped body. B. It will create an elliptical shaped body. C. It will create a triangular shaped body. D. It will create a rectangular shaped body.	**D**
Select the correct line of code to call the shoot( ) function for the arrow object.	**C**

 A. arrow.(playerArcher.body.angle);
B. arrow.shoot(playerArcher.angle);
C. arrow.shoot(playerArcher.body.angle);
D. arrow.shoot(playerArcher);	

Continue the WARM-UP session	
Activity details	**Solution/Guidelines**
Run the presentation from slide 5 to slide 8 to set the problem statement. **Following are the WARM-UP session deliverables:** ● Appreciate the student. ● Discuss the concept of Array. ● How an array can be used to save coordinates of trajectories.	Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.
Teacher ends slideshow	
TEACHER-LED ACTIVITY - 10 mins	
Teacher Initiates Screen Share	
<u>CHALLENGE</u> ● **Create multiple cannonballs.**	
Teacher Action	**Student Action**

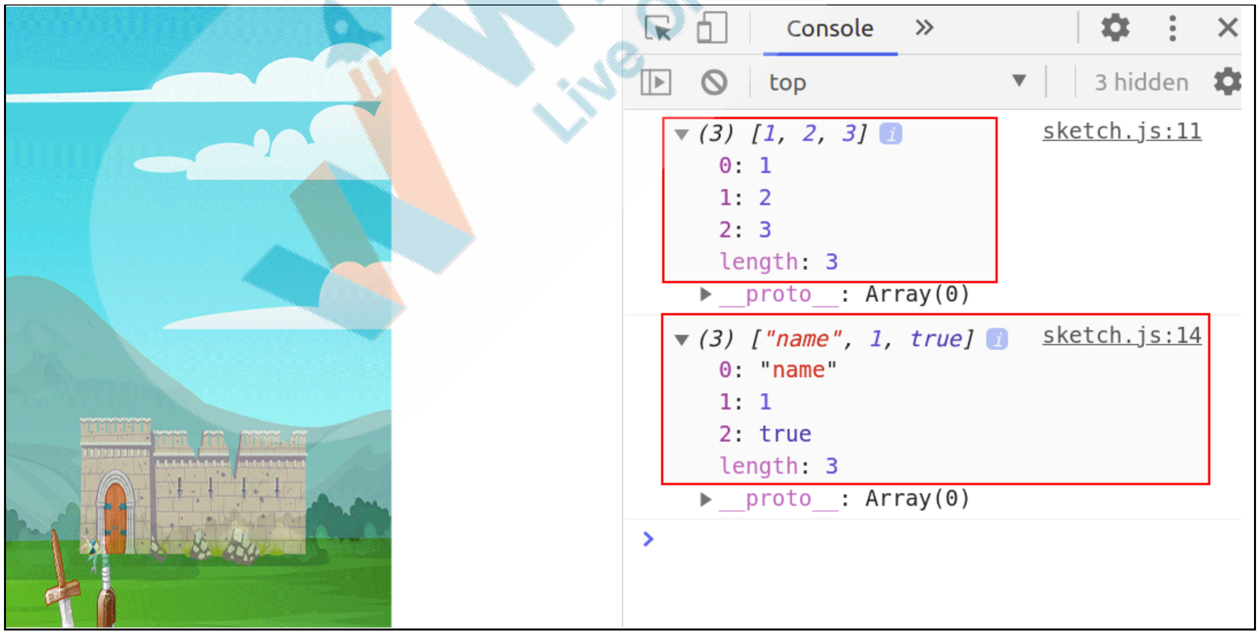
Let us quickly review the code we wrote from the last class. *Teacher can open previous class code from [Teacher Activity 1]()*	*Student reads through the code and explains what each block of code does.*
In the last class, we created a single cannonBall. Today we will start by creating multiple cannonballs. Can you think of a way how we can do that?	**ESR**: Student talks about his/her ideas on how to create multiple cannonballs.
Yes! There can be other ways. We will write a condition to create new cannonballs when we press the arrow key., Which key should we use?	**ESR:** We can create the cannonball when we press the DOWN key.
Before we write the conditions for that, do you recollect about arrays we learned in the previous classes? Let's quickly revise the basics of an array; if you remember it, you can help me write the code. JavaScript and most other languages have different data structures to hold multiple values. One popular data structure to hold multiple values is called an "array". An array is created inside square brackets and can store a list of the same or different types of data separated by a comma.	**ESR:** Varied.

Let's look at an example.



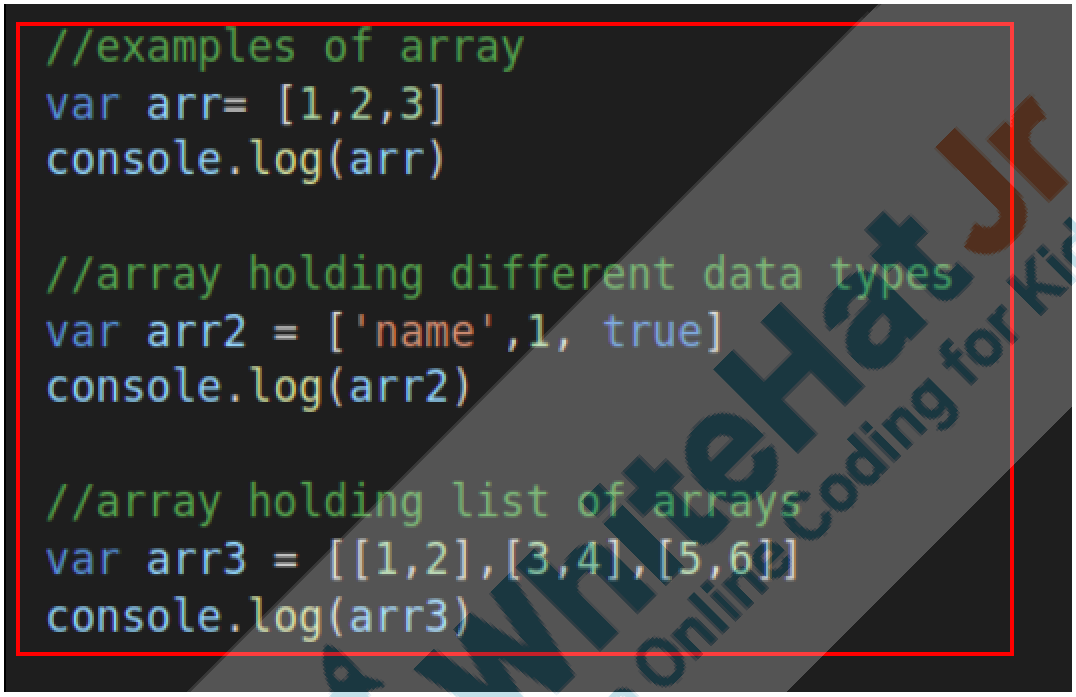
```
const World = Matter.World;
const Bodies = Matter.Bodies;
const Constraint = Matter.Constraint;
var engine, world;
var canvas, angle, tower, ground, cannon;


//examples of array
var arr= [1,2,3]
console.log(arr)

var arr2 = ['name',1, true]
console.log(arr2)
```

Check the output in the console window in the browser:

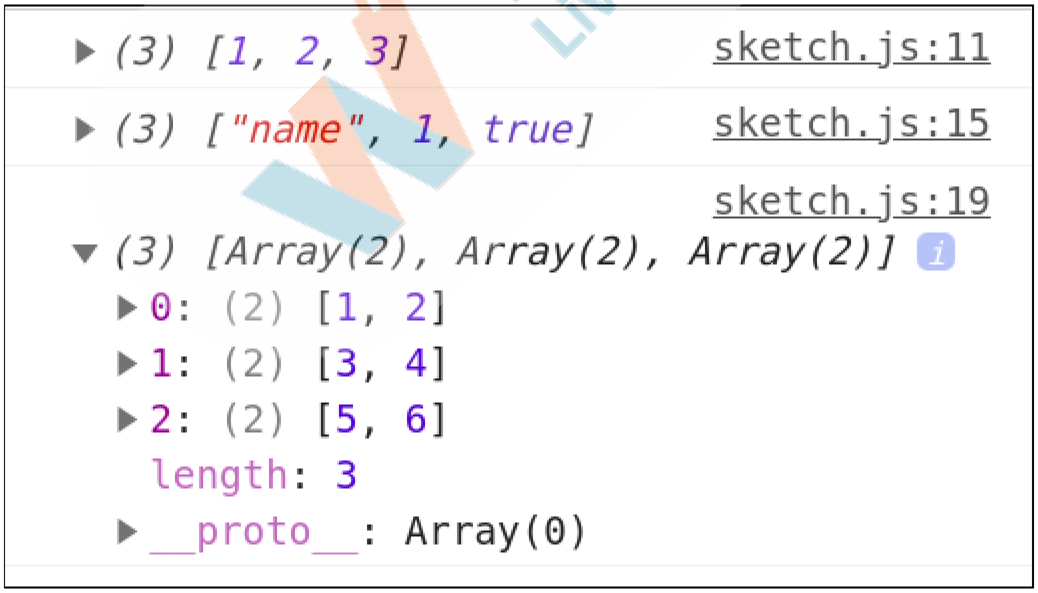


```
▼ (3) [1, 2, 3]                     sketch.js:11
    0: 1
    1: 2
    2: 3
    length: 3
  ▶ __proto__: Array(0)
▼ (3) ["name", 1, true]             sketch.js:14
    0: "name"
    1: 1
    2: true
    length: 3
  ▶ __proto__: Array(0)
>
```

An array can also store a list of arrays! *<Teacher shows an example.>*	The student observes and learns.

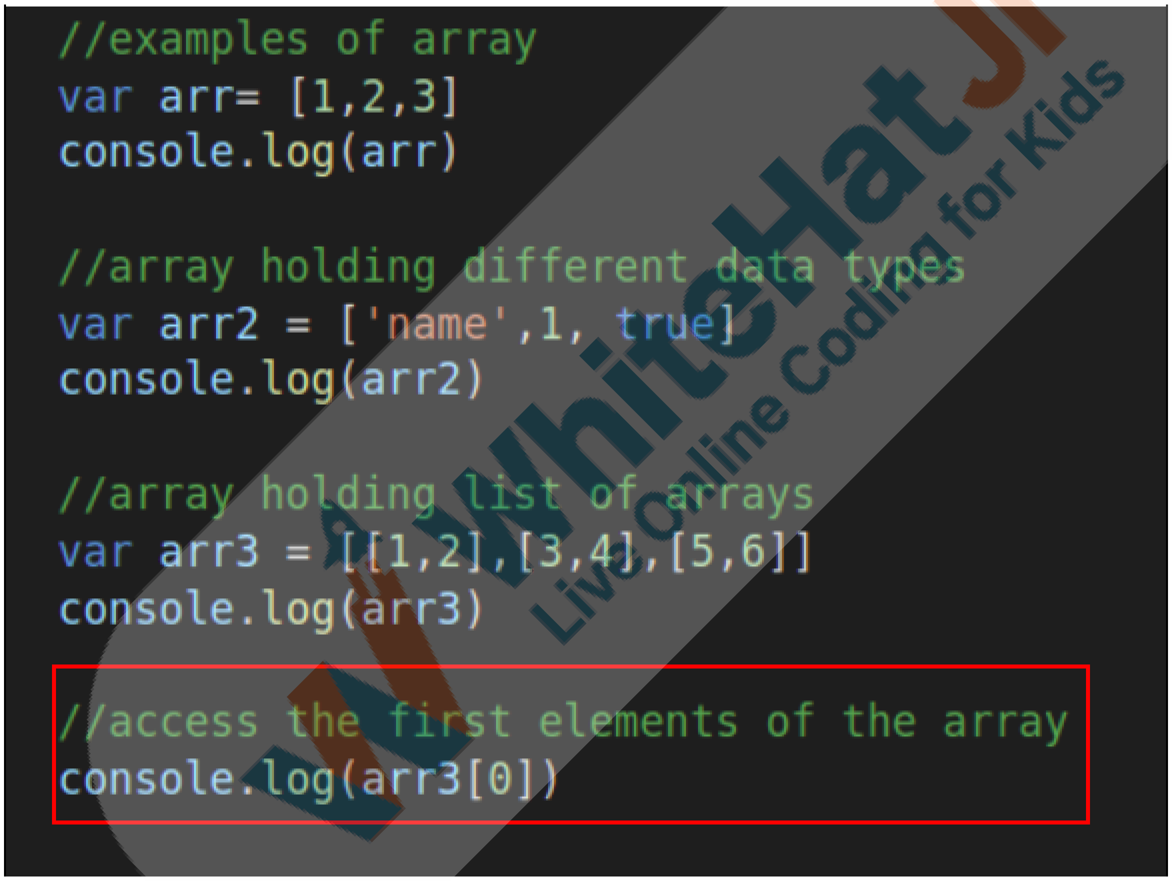


```
//examples of array
var arr= [1,2,3]
console.log(arr)

//array holding different data types
var arr2 = ['name',1, true]
console.log(arr2)

//array holding list of arrays
var arr3 = [[1,2],[3,4],[5,6]]
console.log(arr3)
```

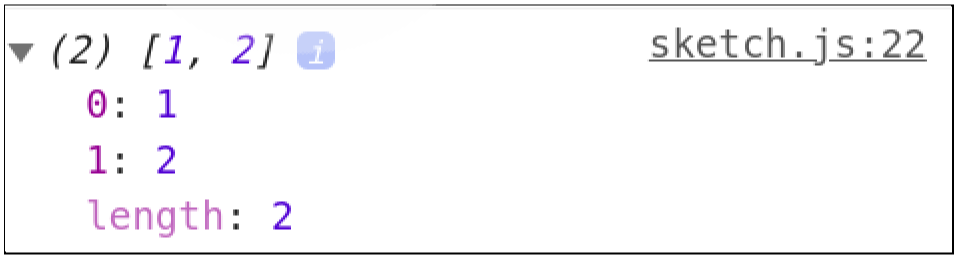


```
▶ (3) [1, 2, 3]                     sketch.js:11
▶ (3) ["name", 1, true]             sketch.js:15
                                    sketch.js:19
▼ (3) [Array(2), Array(2), Array(2)] i
   ▶0: (2) [1, 2]
   ▶1: (2) [3, 4]
   ▶2: (2) [5, 6]
    length: 3
   ▶__proto__: Array(0)
```

Each value in the **array** is indexed by a number. The first value has an index of '0', the second value has an index of '1' and so on. If we want to access the first element of our **arr3 array**, we can access it by using **arr3[0]**. *Teacher shows an example.*	*Student observes and learns.*

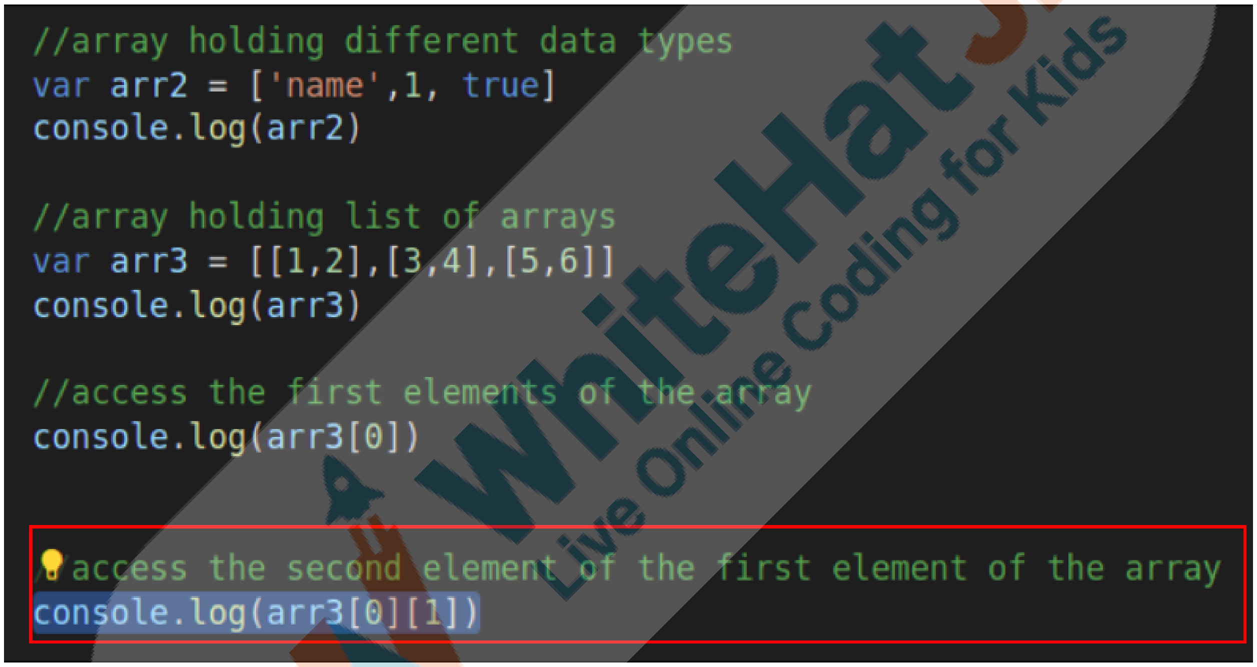
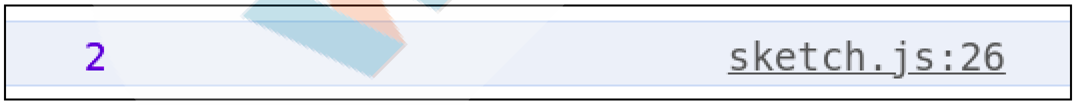


```
//examples of array
var arr= [1,2,3]
console.log(arr)

//array holding different data types
var arr2 = ['name',1, true]
console.log(arr2)

//array holding list of arrays
var arr3 = [[1,2],[3,4],[5,6]]
console.log(arr3)

//access the first elements of the array
console.log(arr3[0])
```



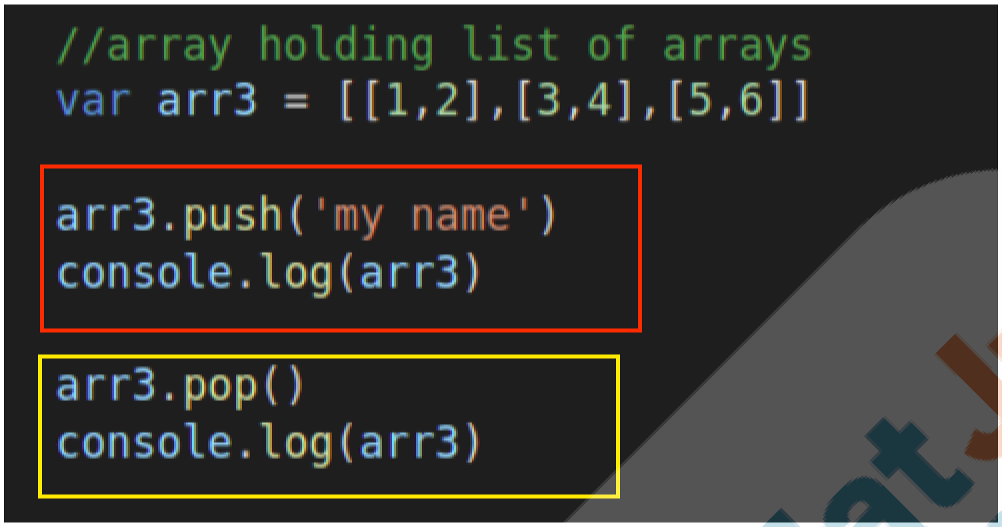
```
▼ (2) [1, 2]                    sketch.js:22
    0: 1
    1: 2
    length: 2
```

But what if we wanted to access the first element inside the first element of the array? We do this by adding a sub-index like this **arr3[0][0]**. The second element of the first element in the array can be accessed with **arr3[0][1]**. *Teacher shows an example.*	ESR: Varied.

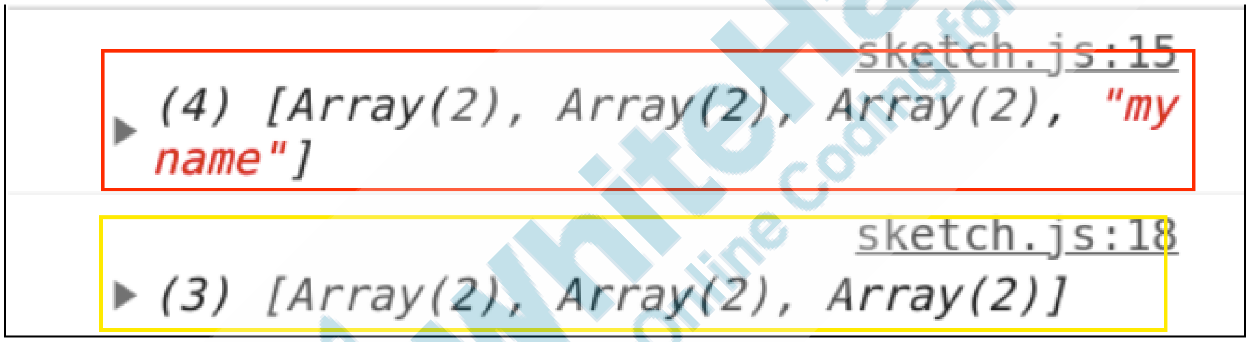

`//array holding different data types`
`var arr2 = ['name',1, true]`
`console.log(arr2)`
 	
`//array holding list of arrays`
`var arr3 = [[1,2],[3,4],[5,6]]`
`console.log(arr3)`
 	
`//access the first elements of the array`
`console.log(arr3[0])`
 	
`//access the second element of the first element of the array`
`console.log(arr3[0][1])`
 	

`2    sketch.js:26`	

A new value can be pushed inside an array by using **array.push()**. Similarly, the last values can be popped out of the arrays using **array.pop()**.	*Student observes and learns.*

Teacher shows with examples.



```
//array holding list of arrays
var arr3 = [[1,2],[3,4],[5,6]]

arr3.push('my name')
console.log(arr3)

arr3.pop()
console.log(arr3)
```



```
                                              sketch.js:15
▶ (4) [Array(2), Array(2), Array(2), "my
  name"]
                                              sketch.js:18
▶ (3) [Array(2), Array(2), Array(2)]
```

Teacher starts slideshow :Slide 9 to Slide 11

Run the presentation slide to set the student activity context.

- Cannonball Images at the position of the Trajectory.

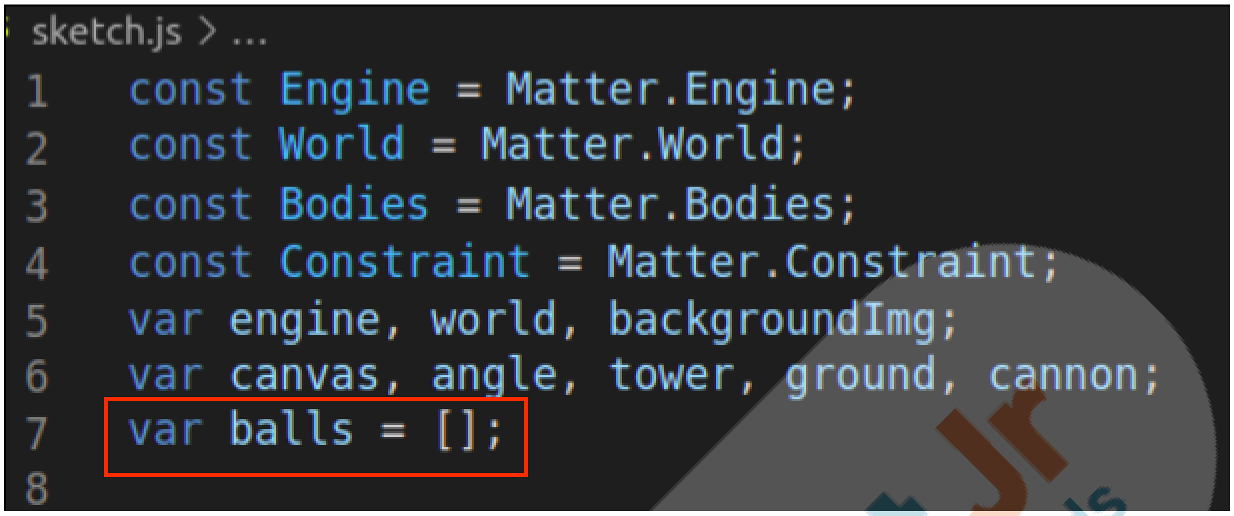
Teacher ends slideshow

Teacher Stops Screen Share

STUDENT-LED ACTIVITY - 25 mins

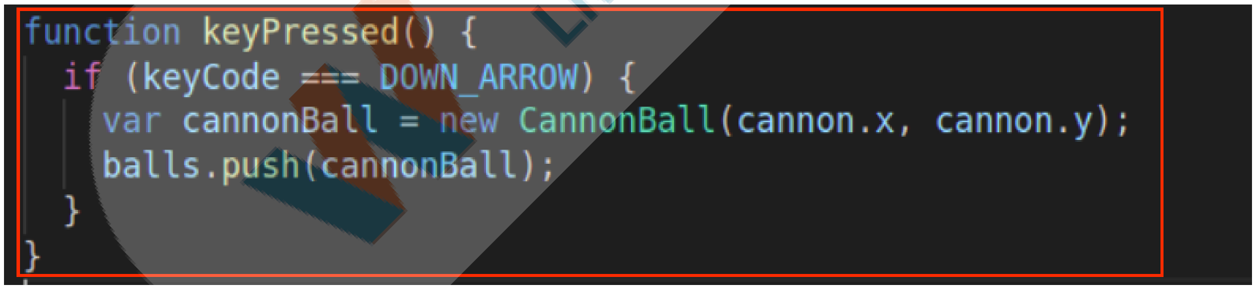
ACTIVITY	
• **Ask the student to press the ESC key to come back to the panel.** • **Guide the student to start Screen Share.** • **The teacher gets into Fullscreen.**	
ACTIVITY • **Write code to create multiple cannonballs.**	
Teacher Action	**Student Action**
Teacher guides the student to open the code from the previous class or download the code from [Student Activity 1](#). We are going to use these concepts of Arrays to create multiple cannonballs instead of creating multiple variables for storing many balls. We can create an empty array to store all the cannonballs we will create in the game. Isn't that a more efficient way? So let's begin by creating an array variable named **balls.** *Guide the student to create an empty balls array.*	*Student opens the code from the previous class or download the code from [Student Activity 1](#).* **ESR:** No. *Student codes to create an empty balls array.*

WhiteHat Jr
Live Online Coding for Kids



```
sketch.js > ...
const Engine = Matter.Engine;
const World = Matter.World;
const Bodies = Matter.Bodies;
const Constraint = Matter.Constraint;
var engine, world, backgroundImg;
var canvas, angle, tower, ground, cannon;
var balls = [];
```

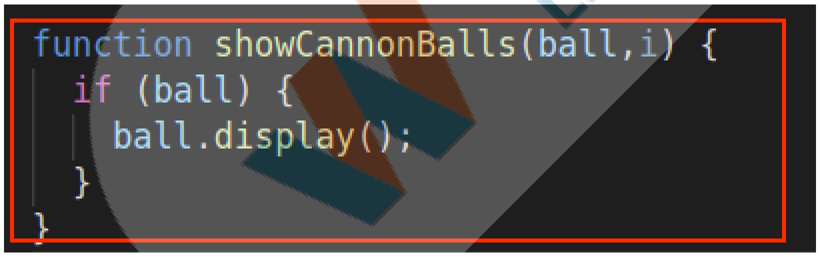
As discussed before we will create a new ball when the down arrow key is pressed. We will create a **keyPressed()** function and inside this function when a key is pressed, we'll create a new cannonball and push it in the balls array. *Guide the student to create a new cannonball when the **DOWN_ARROW** key is pressed and push the cannonball in the balls array.*	*Student codes to create a new cannonball when the DOWN_ARROW key is pressed and push the cannonball in the balls array.*



```
function keyPressed() {
  if (keyCode === DOWN_ARROW) {
    var cannonBall = new CannonBall(cannon.x, cannon.y);
    balls.push(cannonBall);
  }
}
```

We've now created the new cannonballs. What is the next thing that we need to do?	**ESR** We need to display the cannonballs.

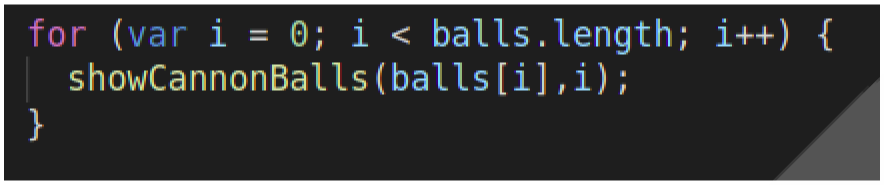
Yes, because the **cannonBall** is stored in an array, we can traverse through each element of an array using a **for loop**. To keep our code organized, let us create a separate function called **showCannonBalls()** in **Sketch.js.** Now that we are creating a separate function for showing cannonballs; let us remove **cannonball.display from** the **draw()** function that we wrote in the last class. *Teacher declares a **function showCannonBalls().** This function will take two parameters, ball, and index.* Using this index we can access each element of the ball array. For now, let us just write one line **cannonBall.display()** inside this function. (We'll be adding more code to the showCannonBall function regarding the cannonballs.) We'll then call the function in the **draw()** function. *Write the **showCannonballs()** to display the cannonball.*	*Student codes to create the **showCannonballs()** to display the balls*



```
function showCannonBalls(ball,i) {
  if (ball) {
    ball.display();
  }
}
```

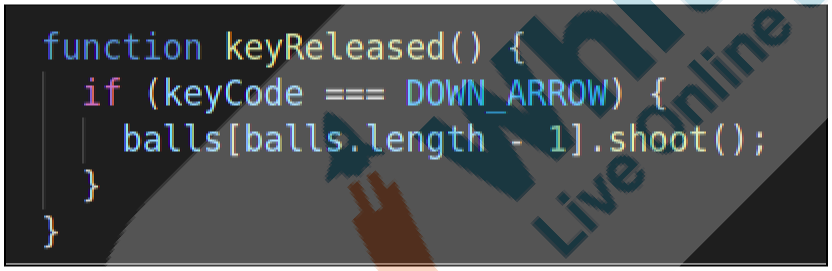
Now using a **for loop** over the ball array we'll get all the cannonballs from the balls array. It allows us to call **showCannonBalls()** multiple times.	*Student codes to use the for loop on the balls array and*

Guide the student to use the for loop on the balls array to get all the cannonballs. *Call the* ***showCannonBalls()*** *function and pass the balls I index to it.*	*call the* ***showCannonBalls()*** *function and pass the ball index to it.*



```
for (var i = 0; i < balls.length; i++) {
  showCannonBalls(balls[i],i);
}
```

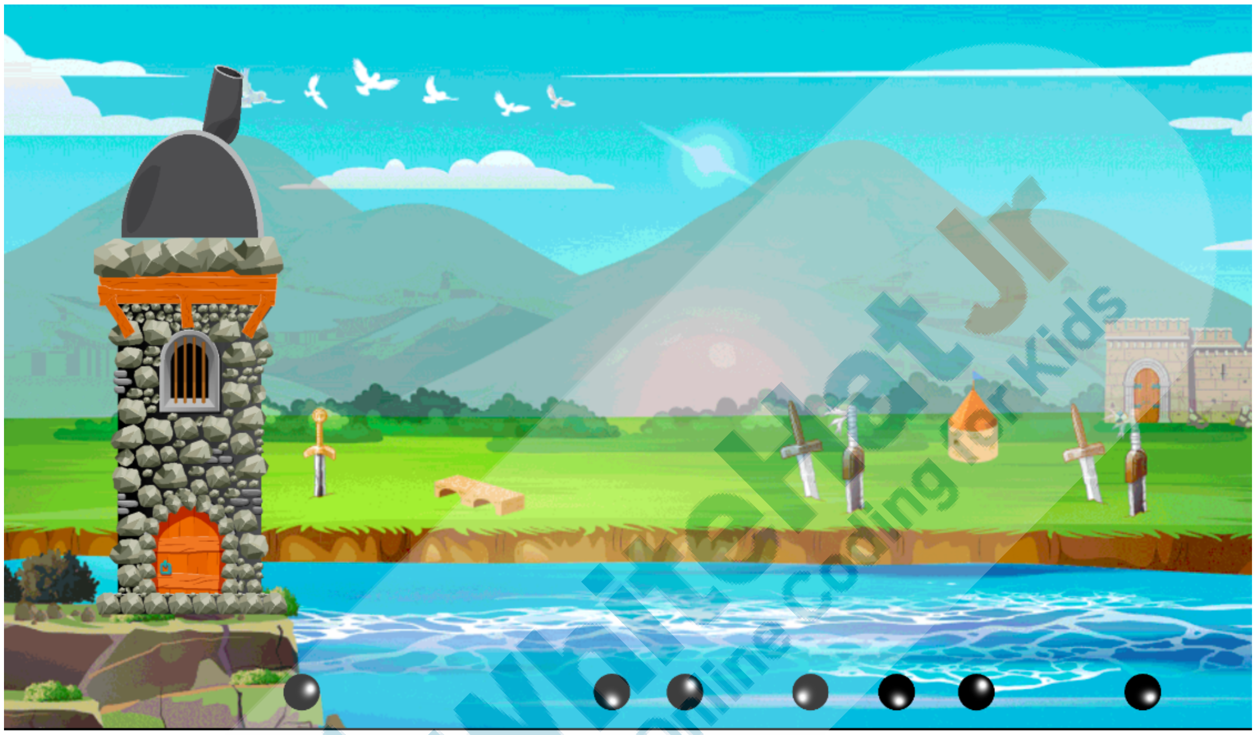
Now the cannonballs are ready to be shot. We'll shoot them when the down arrow key is released. To do so we'll use the **keyReleased()** function. *Guide the student to use the* ***keyReleased()*** *function.* *When the key is released, call the* ***shoot()*** *function on the balls array.*	*Student codes to shoot the cannonball when the DOWN_ARROW key is released.*



```
function keyReleased() {
  if (keyCode === DOWN_ARROW) {
    balls[balls.length - 1].shoot();
  }
}
```

ball.length is the number of elements in an array; we are using **ball.length -1,** as we are accessing the last ball from the array. The index of an array starts from 0, and the length of the array will always be 1 more than the last index of the array. So to get the last element we write length -1.

ex: ar=[100,200,300]
ar.length = 3
and to access 300 we need to write
ar[2] which is as good as ar[ar.length-1] (3-1 = 2)

Now let's run the code to check the output. *Ask the student to run the code to check the output*	*Student runs the code to check the output.*
	
Teacher Guides Student to Stop Screen Share	
WRAP-UP SESSION - 5 Mins	
Teacher starts slideshow from slide 12 to slide 24	
Activity details	**Solution/Guidelines**
Run the presentation from slide 14 to slide 18. **Following are the WARM-UP session deliverables:** ● **Appreciate the student.** ● **Revise the current class activities.** ● **Discuss the quizzes.**	Discuss with the student the current class activities and Student will ask doubts related to the activities.

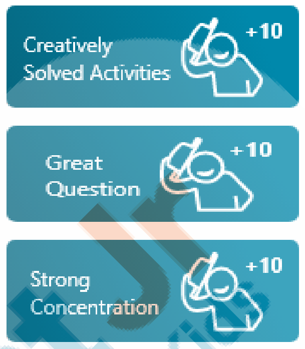
Quiz time - Click on in-class quiz	
Question	**Answer**
Which of the following data structures is used to capture the multiple positions of cannonballs? A. Arrays B. JSON C. Variable D. lists	**A**
Which of the following instructions are used to remove a body from the world? A. slice() B. World.remove() C. World.add() D. splice()	**B**
Which code block can be used to make the body static? A. isStatic= true B. isStatic= false C. isStatic:false D. isStatic: true	**D**
End the quiz panel	
Activity details	**Solution/Guidelines**
Run the presentation from slide 19 to slide 24 **Following are the WARM-UP session deliverables:** ● **Explain the facts and trivia** ● **Next class challenge** ● **Project for the day** ● **Additional Activity**	Guide the student to develop the project and share with us
FEEDBACK ● **Appreciate the student for their efforts in the class.** ● **Ask the student to make notes for the reflection journal along with the code**	

they wrote in today's class.

You get Hats Off for your excellent work!

In the next class we'll create the pirate boats.

Make sure you have given at least 2 Hats Off during the class for:



*** This Project will take only 30 mins to complete. Motivate students to try and finish it immediately after the class.**

Project Overview

EPIC ARCHERY STAGE 3

Goal of the Project:

In Class 24, you have learned how to create multiple cannonballs, when the space key is pressed. In this project, you will apply the same concept to create multiple PlayerArrow.

* This is a continuation of Project 23 & Project 24. Make sure to complete those projects before attempting this one.

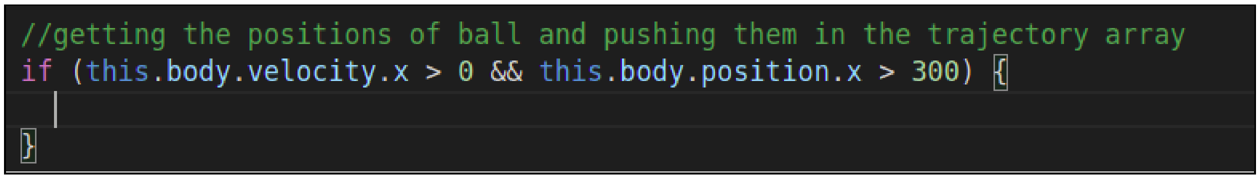
Story:

Archery is one of the oldest arts which is still practiced. After reading the information about Archery in a book, your friend Georgie wants to play Archery. To give him a virtual experience, you want to use your coding expertise and physics engine concepts to create an Archery game for him.

Note: You can assign the project to the student in class itself by clicking on the Assign Project button which is available under the projects tab.

Student engages engages with the teacher over the project.

Can you create multiple arrows for the player? I am very excited to see your project solution and I know you will do really well. Bye Bye!	
Teacher ends slideshow	
Teacher Clicks ✖ End Class	
ADDITIONAL ACTIVITIES	
Additional activity to create the trajectory path for the cannonball when launched from the cannon. To draw the trajectory line first we'll need all the positions that the ball has traveled to. Then we can draw the lines/image to create the trajectory line. So when do we want to start to collect the positions? We'll start to collect the positions from the current positions of the cannonball. *<Teacher asks student to use console.log(this.body.position) in cannonBall.js to find the position when ball toques the water, use corresponding value in the if condition.>*	**ESR:** When we shoot the cannonball. *<Student codes to add the condition to check if the velocity of ball is greater than 0 and the position is greater than one captured using console.log().>*
In cannonBall.js file	



```
//getting the positions of ball and pushing them in the trajectory array
if (this.body.velocity.x > 0 && this.body.position.x > 300) {
    
}
```

Now inside the code to capture the positions we'll create a new array called as position and put the positions inside it.

There are multiple ways to add elements to the array such as creating an empty array and then pushing the elements inside it using the **push()** method.
This method requires additional lines of code.

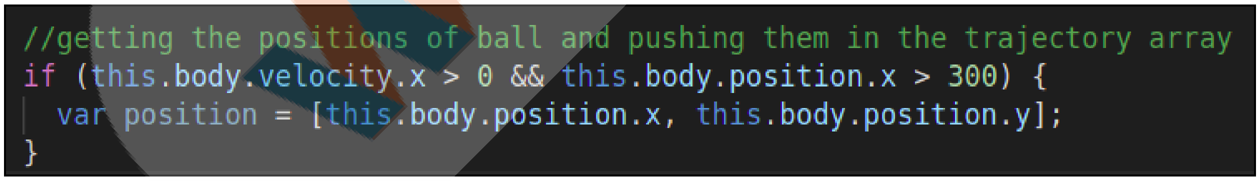
The smart method that we have is to declare a list and add the variables directly inside the list.
Which is what we are doing here.

we declare a position array and inside the array add the variables position.x and position.y which will automatically fill the values.

<The teacher guides the student with the code.>

<Student codes to create a new position array and add the x and y positions of the cannonball in it.>

in the cannonBall.js file



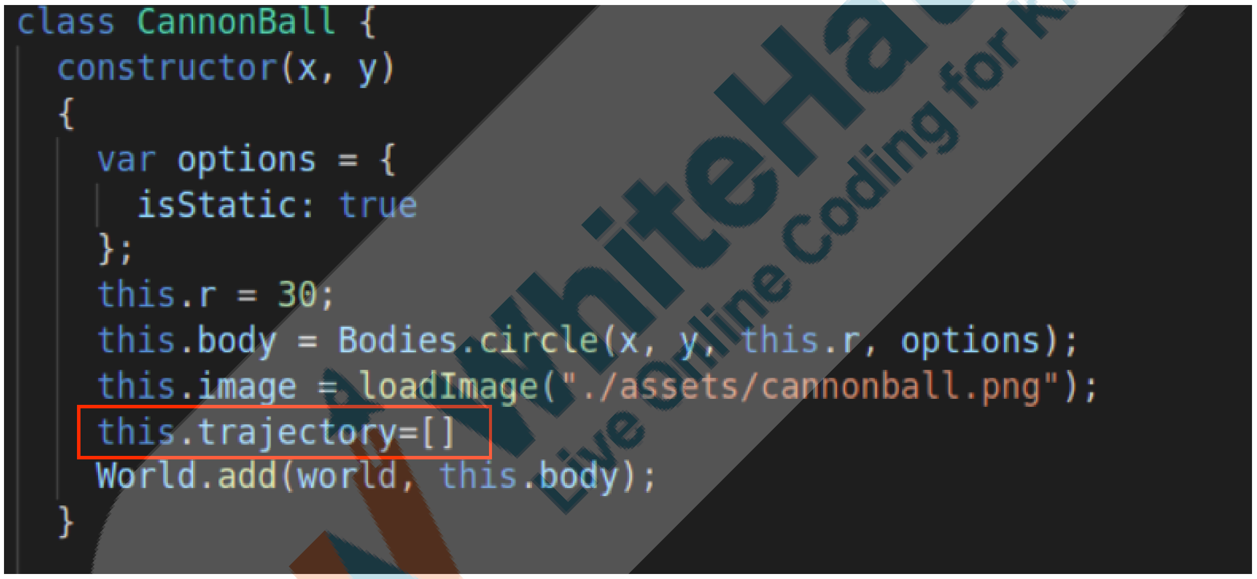
```
//getting the positions of ball and pushing them in the trajectory array
if (this.body.velocity.x > 0 && this.body.position.x > 300) {
    var position = [this.body.position.x, this.body.position.y];
}
```

Our position array just has the single position of the cannonball, similarly, we need to store all the positions traveled by the cannonball.

<The student codes to create an empty trajectory array and push the positions in it.>

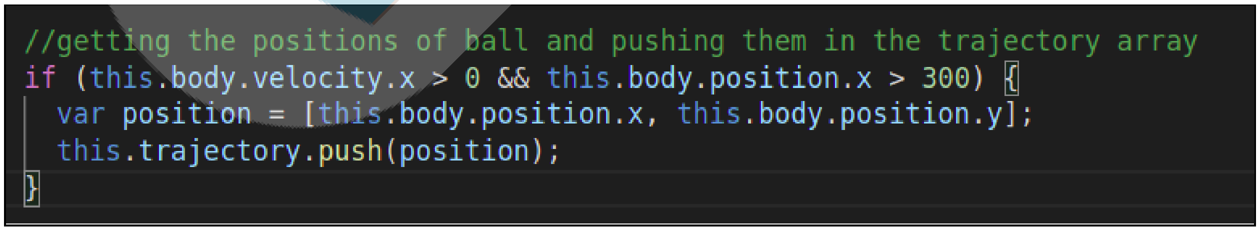
To do so we'll create a **trajectory array** and push all the positions inside this trajectory array. So the all the position array will be inside the trajectory array making it an array of arrays. ex**. trajectory =[[position1], [position2], [position3].........]** *<The teacher helps the student with the code.>*	

in cannonBall.js file



```
class CannonBall {
  constructor(x, y)
  {
    var options = {
      isStatic: true
    };
    this.r = 30;
    this.body = Bodies.circle(x, y, this.r, options);
    this.image = loadImage("./assets/cannonball.png");
    this.trajectory=[]
    World.add(world, this.body);
  }
```

Add positions to the trajectory array.

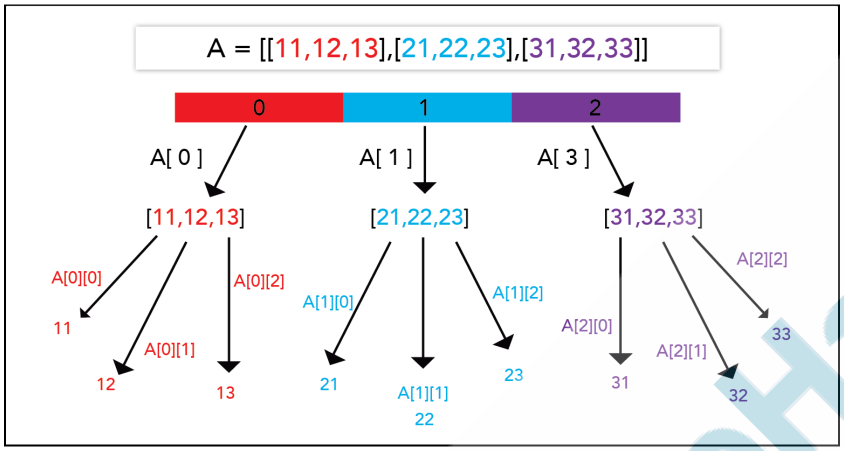


```
//getting the positions of ball and pushing them in the trajectory array
if (this.body.velocity.x > 0 && this.body.position.x > 300) {
  var position = [this.body.position.x, this.body.position.y];
  this.trajectory.push(position);
}
```

Good job. Now we have all the positions traveled by the cannonball.	*<The student codes to loop over the trajectory array and draw image at all the*

All we need to do is to draw the image on the positions stored in the **trajectory array**.

We can just use a **for loop** to get all the values inside the **trajectory array** and draw an image on it.



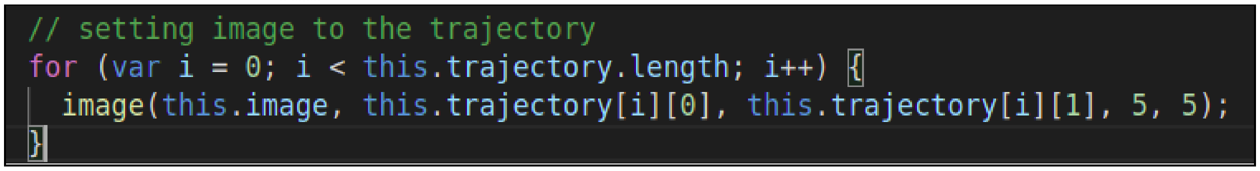
Here we are using the for loop to get the values inside the array.
We are using a loop on the length of the trajectory array.
While looping on the array the variable **i** will act as the index number for all the elements inside the trajectory array.

trajectory[i][0] here the **trajectory[i]** means accessing the first array inside the trajectory array.

trajectory[i][0] means accessing the first element inside the array using index number 0.

positions traveled by the cannonball>

in cannonBall.js file.



```
// setting image to the trajectory
for (var i = 0; i < this.trajectory.length; i++) {
  image(this.image, this.trajectory[i][0], this.trajectory[i][1], 5, 5);
}
```

We are using a cannonball image for trajectory, and reducing its size to look smaller.

Activity	Activity Name	Links
Teacher Activity 1	Previous Class Code	https://github.com/pro-whitehatjr/PRO-C23-Reference_code
Teacher Activity 2	Teacher Reference	https://github.com/pro-whitehatjr/PRO-C24-Reference_code
Student Activity 1	Boilerplate code.	https://github.com/pro-whitehatjr/PRO-C23-Reference_code
Teacher Reference Visual Aid Link	Visual Aid Link	https://s3-whjr-curriculum-uploads.whjr.online/e50b8e5d-8e70-4a50-b4f7-38c32c819d92.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/ec4416bb-d40a-491a-ba65-f074c0a147ea.pdf
Project Solution	Epic Archery Stage 3	https://github.com/pro-whitehatjr/pro-c24-project-sol