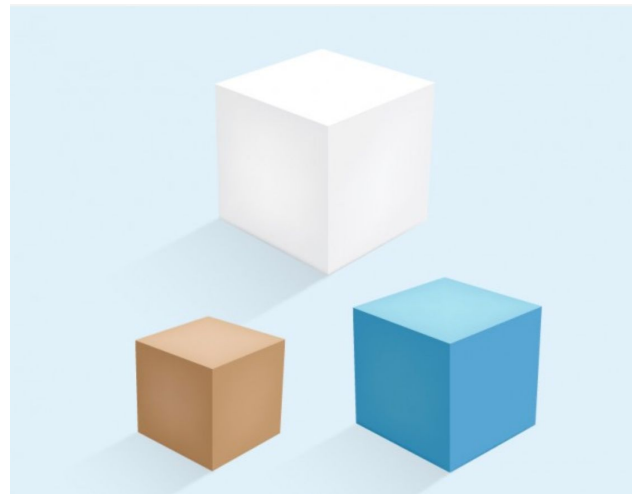


## Class



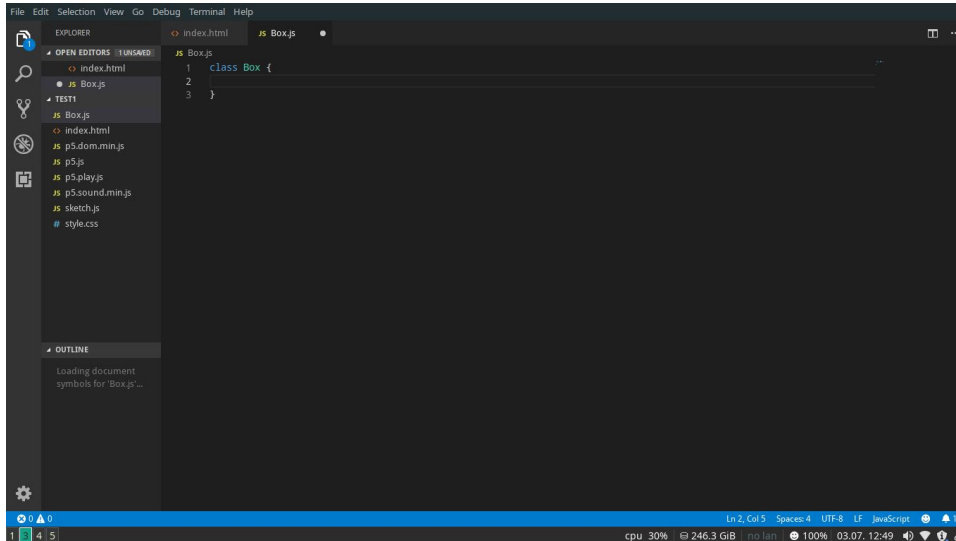
### What we did:

- Create a Box class which creates a template for new objects to be made using the physics engine.
- Create two box objects using the Box class template.
- Tune the physics engine for properties like density, friction etc. for these objects so that they topple over each other
- Display the rectangle so that it can draw with its orientation.

**How we did it:** We need many boxes as obstacles in which our enemy Pigs will be hiding.

**Step 1:** Create a blueprint for our box. A blueprint for an object is called a Class.

Create a new file in the same folder called Box.js.

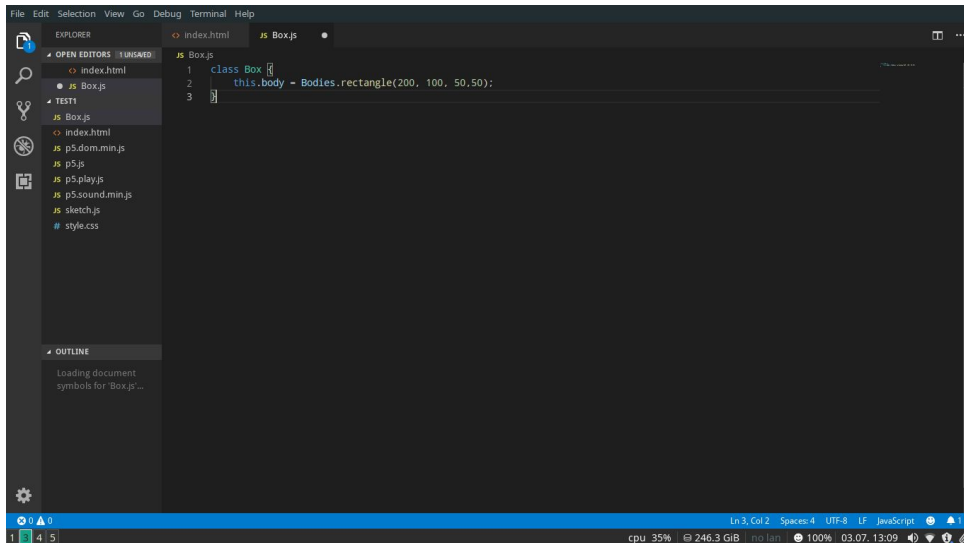


```

1 class Box {
2
3 }
  
```

**Step 2:** Put the x,y, width and height values here as 200,100, 50,50

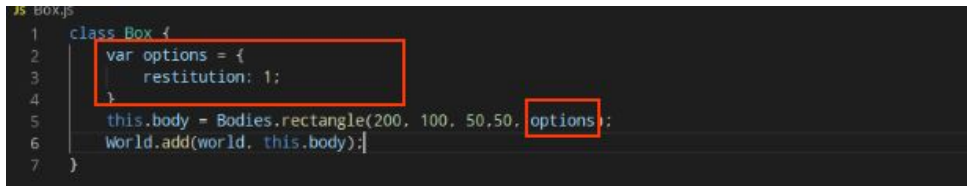
**\*\*Note:** Put all these inside constructor () {}



```

1 class Box {
2   this.body = Bodies.rectangle(200, 100, 50, 50);
3 }
  
```

**Step 3:** Add an option here, which will finetune the physics engine for the object



```

1 class Box {
2   var options = {
3     restitution: 1;
4   }
5   this.body = Bodies.rectangle(200, 100, 50, 50, options);
6   world.add(world, this.body);
7 }
  
```

Add this object to the world

```

1  class Box {
2    var options = {
3      restitution: 1;
4    }
5    this.body = Bodies.rectangle(200, 100, 50, 50, options);
6    world.add(world, this.body);
7  }

```

Display this object— write a display() function to do that.

```

1  class Box {
2    var options = {
3      restitution: 1;
4    }
5    this.body = Bodies.rectangle(200, 100, 50, 50, options);
6    world.add(world, this.body);
7
8    display(){
9      var pos = this.body.position;
10     rectMode(CENTER);
11     fill(255);
12     rect(pos.x, pos.y, this.width, this.height);
13   }

```

**Step 4:** In the sketch.js file, remove all the statements associated with creating the bodies.

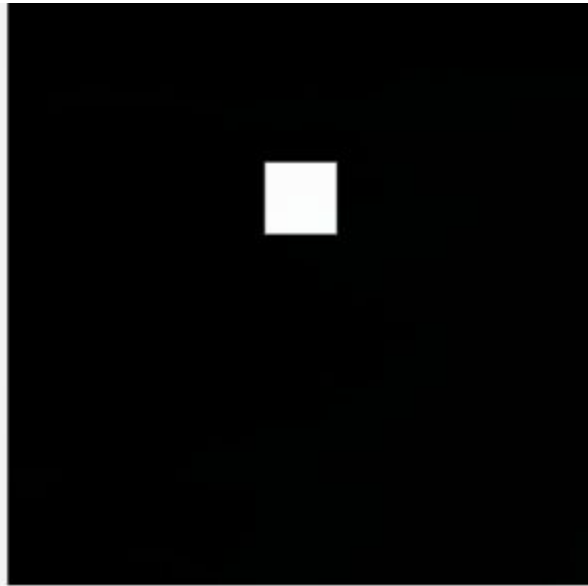
```

1  const Engine = Matter.Engine;
2  const World = Matter.World;
3  const Bodies = Matter.Bodies;
4
5  var engine, world;
6  var box1;
7
8  function setup(){
9    var canvas = createCanvas(400, 400);
10    engine = Engine.create();
11    world = engine.world;
12
13
14  }
15
16  function draw(){
17    background(0);
18    Engine.update(engine);
19
20
21  }

```

**Step 5:** Create a new object and display it with just two statements.

```
1  const Engine = Matter.Engine;  
2  const World= Matter.World;  
3  const Bodies = Matter.Bodies;  
4  
5  var engine, world;  
6  var box1;  
7  
8  function setup(){  
9      var canvas = createCanvas(400,400);  
10     engine = Engine.create();  
11     world = engine.world;  
12     box1 = new Box();  
13  
14 }  
15  
16 function draw(){  
17     background(0);  
18     Engine.update(engine);  
19  
20     box1.display();  
21 }
```



**Step 6:** Write code in the Box class to show how a constructor of a class can take arguments: Tell the computer where to draw the rectangle and of what dimensions, by passing the x,y, width and height to the constructor.

```

1  class Box {
2      constructor(x,y,width,height) {
3          var options = {
4              restitution:0.8
5          }
6          this.body = Bodies.rectangle(x,y,width,height);
7          World.add(world, this.body);
8      }
9      display() {
10         var pos = this.body.position;
11         rectMode(CENTER);
12         fill(255);
13         rect(pos.x, pos.y, this.width, this.height);
14     }
15 };
```

**Step 7:** Create the second box object using the Box class.

```

1  const Engine = Matter.Engine;
2  const World= Matter.World;
3  const Bodies = Matter.Bodies;
4
5  var engine, world;
6  var box1;
7
8  function setup(){
9      var canvas = createCanvas(400,400);
10     engine = Engine.create();
11     world = engine.world;
12
13     box1 = new Box(200,100,50,50);
14     box2 = new Box(100,50,50,100);
15 }
16
17 function draw(){
18     background(0);
19     Engine.update(engine);
20
21     box1.display();
22     box2.display();
23 }
```

Create a Ground class blueprint and then create a ground object using it

```

class Ground {
  constructor(x,y,width,height) {
    var options = {
      isStatic: true
    }
    this.body = Bodies.rectangle(x,y,width,height,options);
    this.width = width;
    this.height = height;
    World.add(world, this.body);
  }
  display(){
    var pos =this.body.position;
    rectMode(CENTER);
    fill(255);
    rect(pos.x, pos.y, this.width, this.height);
  }
};

```

```

1 <!DOCTYPE html><html><head>
2   <script src="p5.js"></script>
3   <script src="p5.dom.min.js"></script>
4   <script src="p5.sound.min.js"></script>
5   <script src="https://unpkg.com/matter-js@0.14.2/build/matter.min.js"></script>
6   <script src="p5.play.js"></script>
7   <script src="Box.js"></script>
8   <script src="Ground.js"></script>
9   <link rel="stylesheet" type="text/css" href="style.css">
10  <meta charset="utf-8">
11
12 </head>
13 <body>
14   <script src="sketch.js"></script>
15
16
17 </body></html>

```

```

1 const Engine = Matter.Engine;
2 const World= Matter.World;
3 const Bodies = Matter.Bodies;
4
5 var engine, world;
6 var box1;
7
8 function setup(){
9   var canvas = createCanvas(400,400);
10   engine = Engine.create();
11   world = engine.world;
12
13   box1 = new Box(200,300,50,50);
14   box2 = new Box(200,100,50,100);
15   ground = new Ground(200,390,400,20)
16 }
17
18 function draw(){
19   background(0);
20   Engine.update(engine);
21
22   box1.display();
23   box2.display();
24   ground.display();
25 }

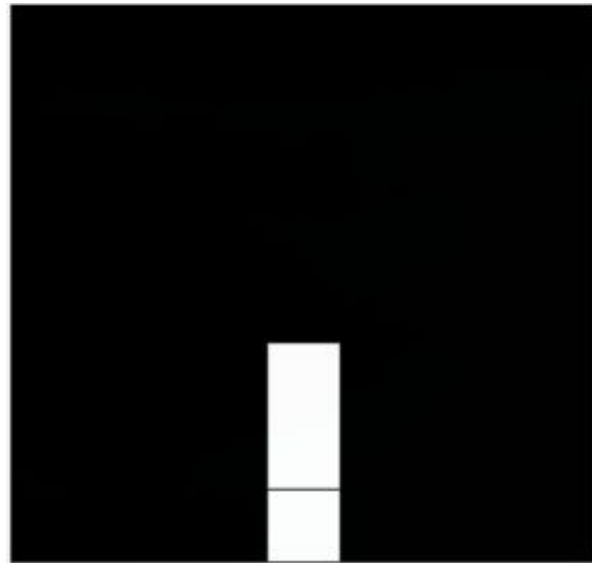
```

**Step 8:** Stack the two box objects on top of each other: Allow the box2 to fall on the top of box 1.

```

1  const Engine = Matter.Engine;
2  const World= Matter.World;
3  const Bodies = Matter.Bodies;
4
5  var engine, world;
6  var box1;
7
8  function setup(){
9    var canvas = createCanvas(400,400);
10   engine = Engine.create();
11   world = engine.world;
12
13   box1 = new Box(200,300,50,50);
14   box2 = new Box(200,100,50,100);
15   ground = new Ground(200,390,400,20)
16 }
17
18 function draw(){
19   background(0);
20   Engine.update(engine);
21
22   box1.display();
23   box2.display();
24   ground.display();
25 }

```



Shift box 2 slightly to the right

```

1  const Engine = Matter.Engine;
2  const World= Matter.World;
3  const Bodies = Matter.Bodies;
4
5  var engine, world;
6  var box1;
7
8  function setup(){
9    var canvas = createCanvas(400,400);
10   engine = Engine.create();
11   world = engine.world;
12
13   box1 = new Box(200,300,50,50);
14   box2 = new Box(240,100,50,100);
15   ground = new Ground(200,390,400,20)
16 }
17
18 function draw(){
19   background(0);
20   Engine.update(engine);
21
22   box1.display();
23   box2.display();
24   ground.display();
25 }

```



**Step 9:** Store the new translation and rotation setting and then revert back to the old setting when the object is drawn.

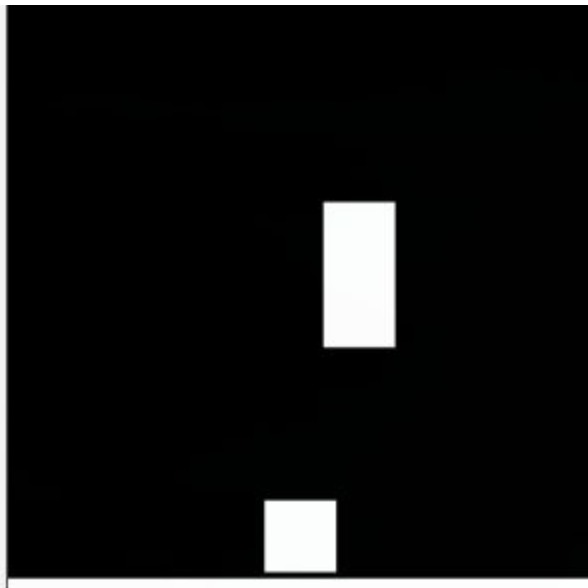
This is done using push() and pop().

push() -> captures the new setting

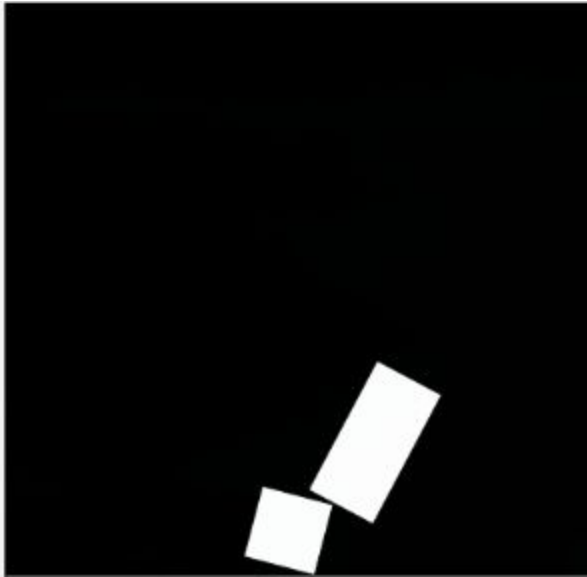
pop() -> reverts back to the old setting

translate() -> to change the 0 of the axis to a given x and y position.

```
1 class Box {
2   constructor(x, y, width, height) {
3     var options = {
4       'restitution':0.8
5     }
6     this.body = Bodies.rectangle(x, y, width, height, options);
7     this.width = width;
8     this.height = height;
9
10    World.add(world, this.body);
11  }
12  display(){
13    var pos =this.body.position;
14    var angle = this.body.angle;
15    push();
16    translate(pos.x, pos.y);
17    rotate(angle);
18    rectMode(CENTER);
19    fill(255);
20    rect(0, 0, this.width, this.height);
21    pop();
22  }
23 };
```







You can play around with more properties of objects like restitution, density, friction etc.

**What's next?:** Using what we learned today to create the stack of obstacles for the pig in the Angry Birds Game.