Motion of many objects

In the real world, the one we're seeking inspiration from here, we have more than one moving object - we're surrounded by many objects of varying mass and location. Let's look at how we can get our `Mover` to better simulate that aspect of the real world.

To do this, we'll need a quick review of object-oriented programming. Again, we're not covering all the basics of OO programming here. However, since the idea of creating a world filled with objects is pretty fundamental to all the examples in this course, it's worth taking a moment to walk through the steps of going from one object to many.

As a reminder, this is what our current `Mover` looks like. It is identical to the `Mover` object that we created when we first introduced vectors, but with two additions—`mass` and a new `applyForce()` method:

```
var Mover = function() {

   // Set mass equal to 1 for simplicity

   this.mass = 1;

   this.position = new PVector(30, 30);

   this.velocity = new PVector(0, 0);

   this.acceleration = new PVector(0, 0);

};



// Simulates Newton's second law

// Receive a force, divide by mass, add to acceleration

Mover.prototype.applyForce = function(force) {

   var f = PVector.div(force, this.mass);

   this.acceleration.add(f);

};
```

```
Mover.prototype.update = function() {
  // Simulates Motion 101 from the vectors tutorial
  this.velocity.add(this.acceleration);
  this.position.add(this.velocity);
  // Now we make sure to clear acceleration each time
  this.acceleration.mult(0);
};

Mover.prototype.display = function() {
  stroke(0);
  strokeWeight(2);
  fill(255, 255, 255, 127);
  // Scale the size according to the mass, as a simple visualization of mass
  ellipse(this.position.x, this.position.y, this.mass*30, this.mass*30);
};

// Even though we've said we shouldn't check velocity directly,
// there are some exceptions. Here we change it as a quick and easy
// way to bounce our mover off the edges.
Mover.prototype.checkEdges = function() {
  if (this.position.x > width) {
    this.position.x = width;
    this.velocity.x *= -1;
  } else if (this.position.x < 0) {
    this.velocity.x *= -1;
    this.position.x = 0;
  }
  if (this.position.y > height) {
    this.velocity.y *= -1;
    this.position.y = height;
  }
};

var m = new Mover();

var draw = function() {
  background(50, 50, 50);

  var wind = new PVector(0.01, 0);
  var gravity = new PVector(0, 0.1);
  m.applyForce(wind);
  m.applyForce(gravity);

  m.update();
  m.display();
  m.checkEdges();
};
```

Now that our object is set, we can choose to create, say, twenty `Mover` instances with an array, initializing them with a loop

```
var movers = [];

for (var i = 0; i < 20; i++) {
    movers[i] = new Mover();
}
```

But now we have a small issue. If we refer back to the `Mover` object's constructor…

```
var Mover = function() {

    this.mass = 1;

    this.position = new PVector(30, 30);

    this.velocity = new PVector(0, 0);

    this.acceleration = new PVector(0, 0);

};
```

…we discover that every `Mover` object is made exactly the same way. What we want are `Mover` objects with varying mass that start at *varying locations*. Here is where we need to increase the sophistication of our constructor by adding arguments.

```
var Mover = function(m, x, y) {

    this.mass = m;

    this.position = new PVector(x, y);

    this.velocity = new PVector(0, 0);

    this.acceleration = new PVector(0, 0);

};
```

Notice how the mass and location are no longer set to hardcoded numbers, but rather initialized via arguments passed through the constructor. This means we can create a variety of `Mover` objects: big ones, small ones, ones that start on the left side of the screen, ones that start on the right, etc.

```
// A big Mover on the left side of the window

var m1 = new Mover(10, 0, height/2);

// A small Mover on the right side of the window

var m2 = new Mover(0.1, width, height/2);
```

With an array, however, we want to initialize all of the objects with a loop.

```
for (var i = 0; i < movers.length; i++) {
    movers[i] = new Mover(random(0.1, 5), 0, 0);
}
```

For each mover created, the mass is set to a random value between 0.1 and 5, the starting x-location is set to 0, and the starting y-location is set to 0. Certainly, there are all sorts of ways we might choose to initialize the objects; this is just a demonstration of one possibility.

Once the array of objects is declared, created, and initialized, the rest of the code is simple. We run through every object, hand them each the forces in the environment, and enjoy the show.

```
draw = function() {
  background(50, 50, 50);

  for (var i = 0; i < movers.length; i++) {
    var wind = new PVector(0.01, 0);
    var gravity = new PVector(0, 0.1);
    movers[i].applyForce(wind);
    movers[i].applyForce(gravity);
    movers[i].update();
    movers[i].display();
    movers[i].checkEdges();
  }
};
```

Here's how the program looks, all together. Note how in the program, the smaller circles reach the right of the window faster than the larger ones.

This is because of our formula: *acceleration = force divided by mass*. The larger the mass, the smaller the acceleration.

```
var Mover = function(m, x, y) {

   this.mass = m;

   this.position = new PVector(x, y);

   this.velocity = new PVector(0, 0);

   this.acceleration = new PVector(0, 0);

};


Mover.prototype.applyForce = function(force) {

  var f = PVector.div(force, this.mass);

  this.acceleration.add(f);

};


Mover.prototype.update = function() {

  this.velocity.add(this.acceleration);

  this.position.add(this.velocity);

  this.acceleration.mult(0);

};


Mover.prototype.display = function() {

  stroke(0);

  strokeWeight(2);

  fill(255, 255, 255, 127);

  ellipse(this.position.x, this.position.y, this.mass*16, this.mass*16);

};
```

```
Mover.prototype.checkEdges = function() {
  if (this.position.x > width) {
    this.position.x = width;
    this.velocity.x *= -1;
  } else if (this.position.x < 0) {
    this.velocity.x *= -1;
    this.position.x = 0;
  }
  if (this.position.y > height) {
    this.velocity.y *= -1;
    this.position.y = height;
  }
};

var movers = [];

for (var i = 0; i < 20; i++) {
    movers[i] = new Mover(random(0.1, 5), 0, 0);
}

var draw = function() {
  background(50, 50, 50);

  for (var i = 0; i < movers.length; i++) {
    var wind = new PVector(0.01, 0);
    var gravity = new PVector(0, 0.1);
    movers[i].applyForce(wind);
    movers[i].applyForce(gravity);

    movers[i].update();

    movers[i].display();

    movers[i].checkEdges();

  }

};
```