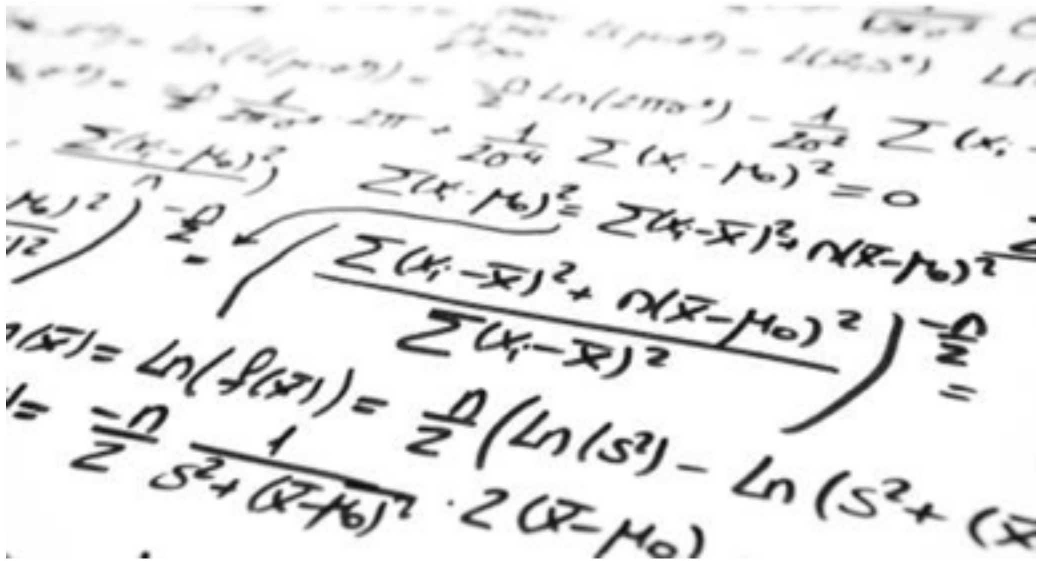
Physics Libraries: Box2D

a physics engine

written by Erin Cato
started out as physics tutorials for GDC in 2006
true physics engine

Box2D

Box2D



Box2D

nothing to do with pixels
translate back and forth between pixel world and physics world

good for when you have many of something; and/or oddly shaped things (not circles)

Typical movement process:

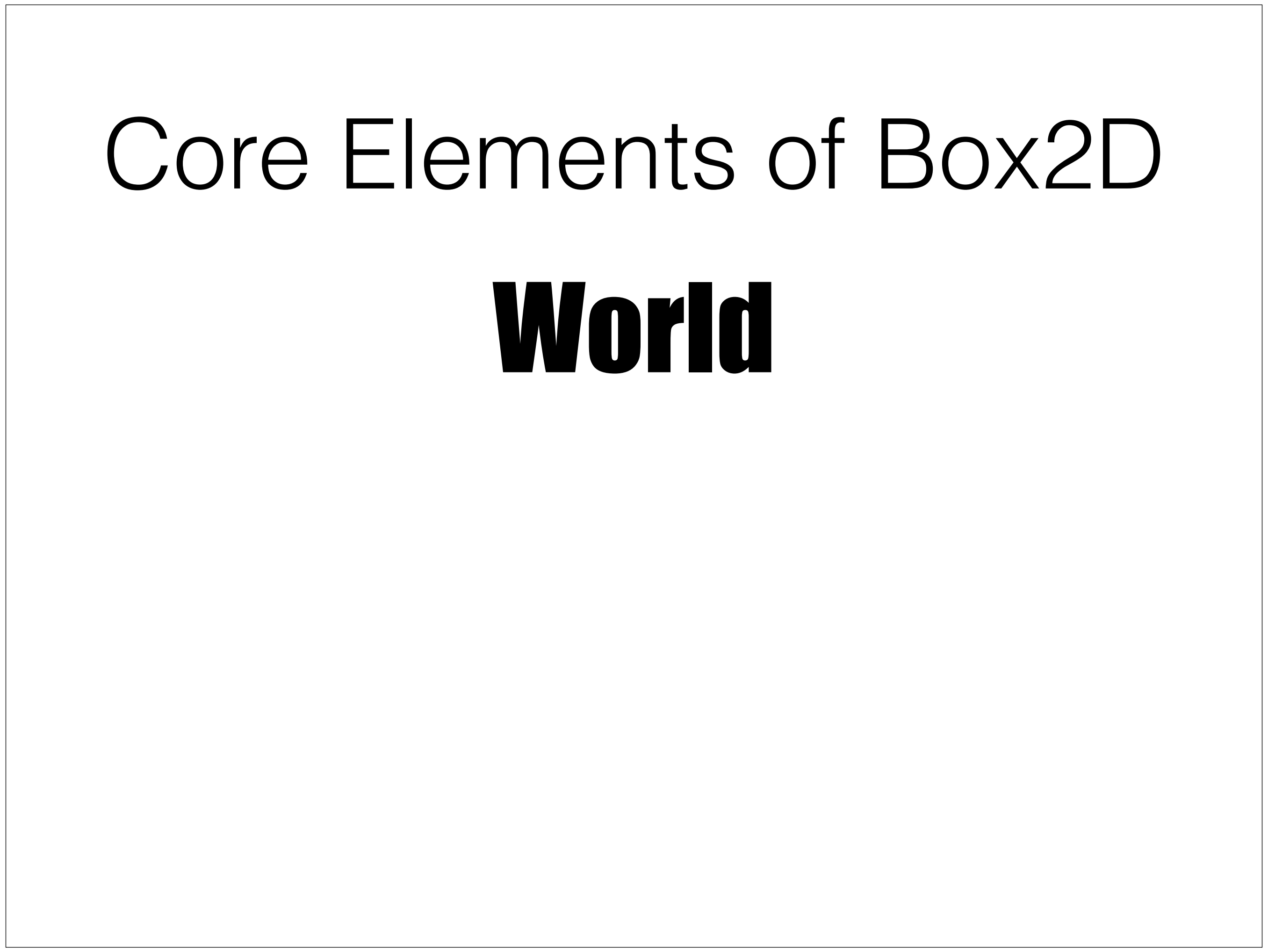
1. Calculate forces (F = M * A)
2. Apply forces to objects
3. Update locations for objects
4. Draw Objects

Box2D movement process:

1. Create objects for pixel world
2. Translate pixel world info to Box2D
3. Ask Box2D where things are
4. Translate Box2D info to pixel info
5. Draw everything

Core Elements of Box2D

World



Core Elements of Box2D

World

manages the physics simulation
knows everything about coordinate space
stores lists for each element

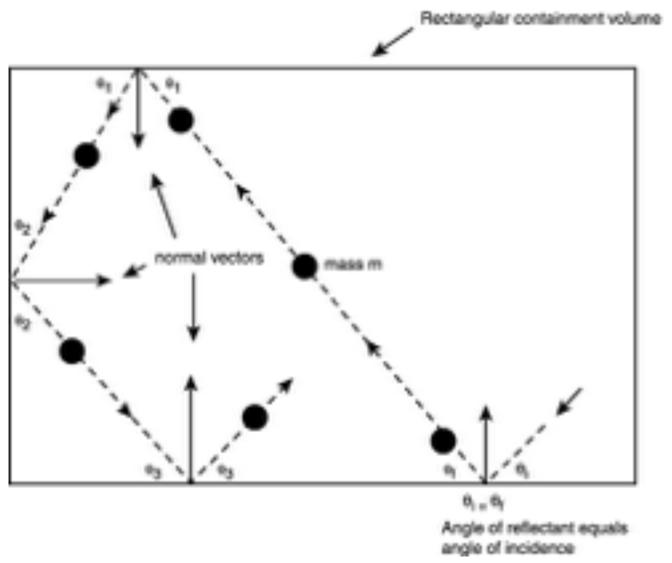
Core Elements of Box2D

Body

Primary Element
Has location + Velocity
moves around the space
experiences forces
no geometry, not physical
has shapes attached to it

Core Elements of Box2D

Shape


Rectangular containment volume
θ1
θ1
θ2
normal vectors
mass m
θ2
θ3
θ3
θi
θi
θi = θi
Angle of reflectant equals angle of incidence

Keeps track of all of the necessary collision geometry

Core Elements of Box2D

Fixture

Attaches shape to a body
Sets properties such as density, friction + Restitution

Core Elements of Box2D

Joint

acts as a connector between 2 bodies

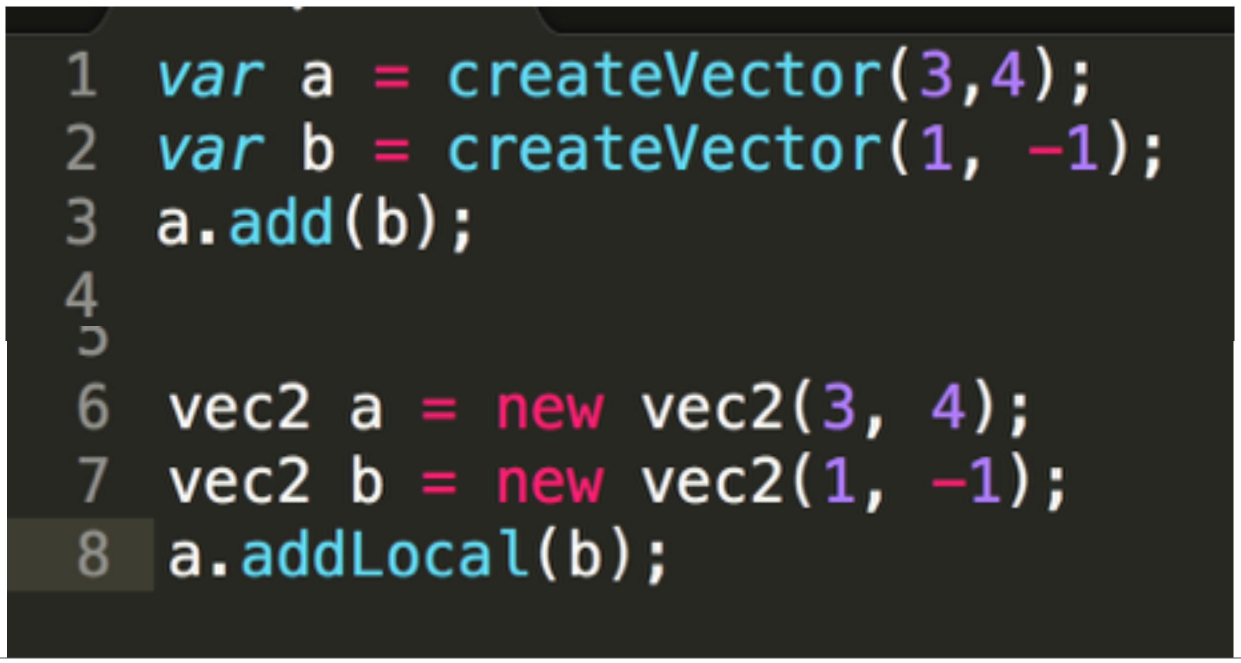
Core Elements of Box2D

Vec2

describes a vector in Box2D
each library tends to have it’s own implementation of a vector

Core Elements of Box2D

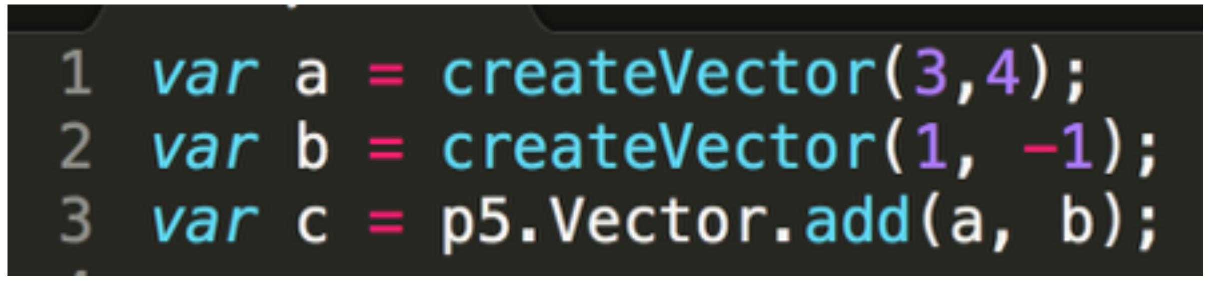
Vec2



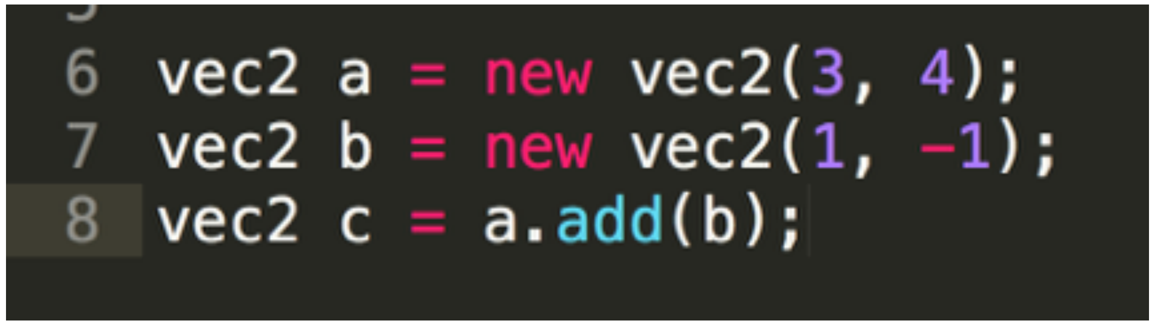
```
var a = createVector(3,4);
var b = createVector(1, -1);
a.add(b);


vec2 a = new vec2(3, 4);
vec2 b = new vec2(1, -1);
a.addLocal(b);
```

Core Elements of Box2D

Vec2



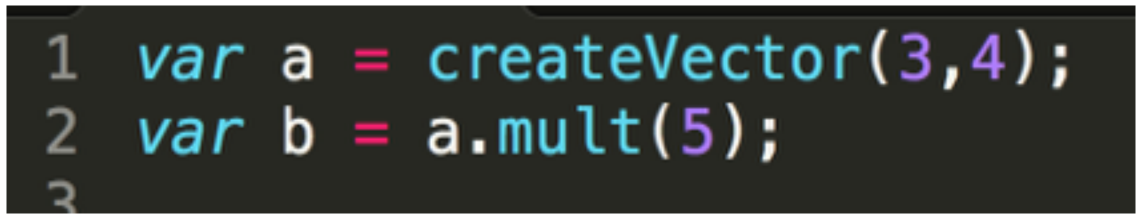
```
var a = createVector(3,4);
var b = createVector(1, -1);
var c = p5.Vector.add(a, b);
```



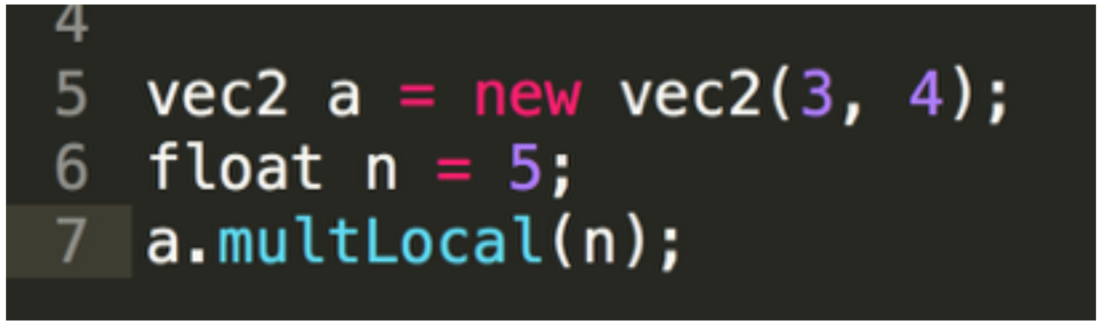
```
vec2 a = new vec2(3, 4);
vec2 b = new vec2(1, -1);
vec2 c = a.add(b);
```

Core Elements of Box2D

Vec2



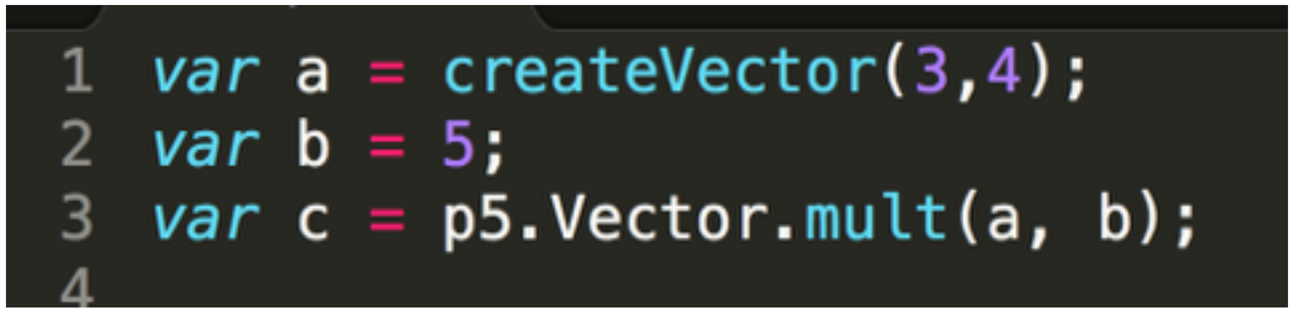
```
var a = createVector(3,4);
var b = a.mult(5);
```



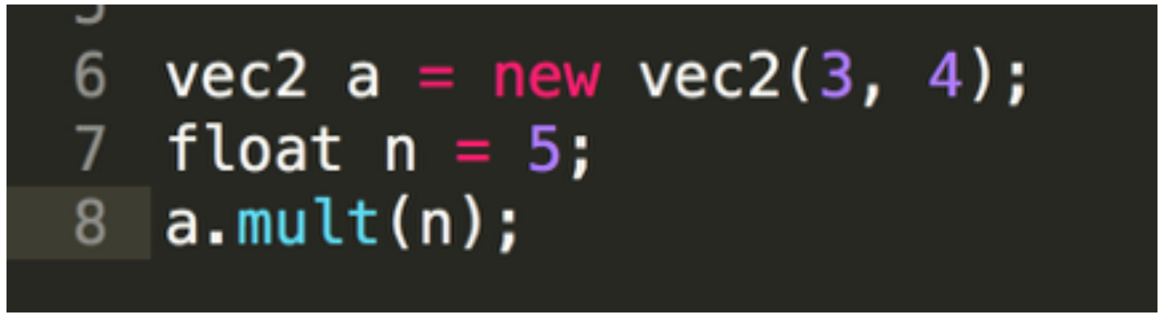
```
vec2 a = new vec2(3, 4);
float n = 5;
a.multLocal(n);
```

Core Elements of Box2D

Vec2



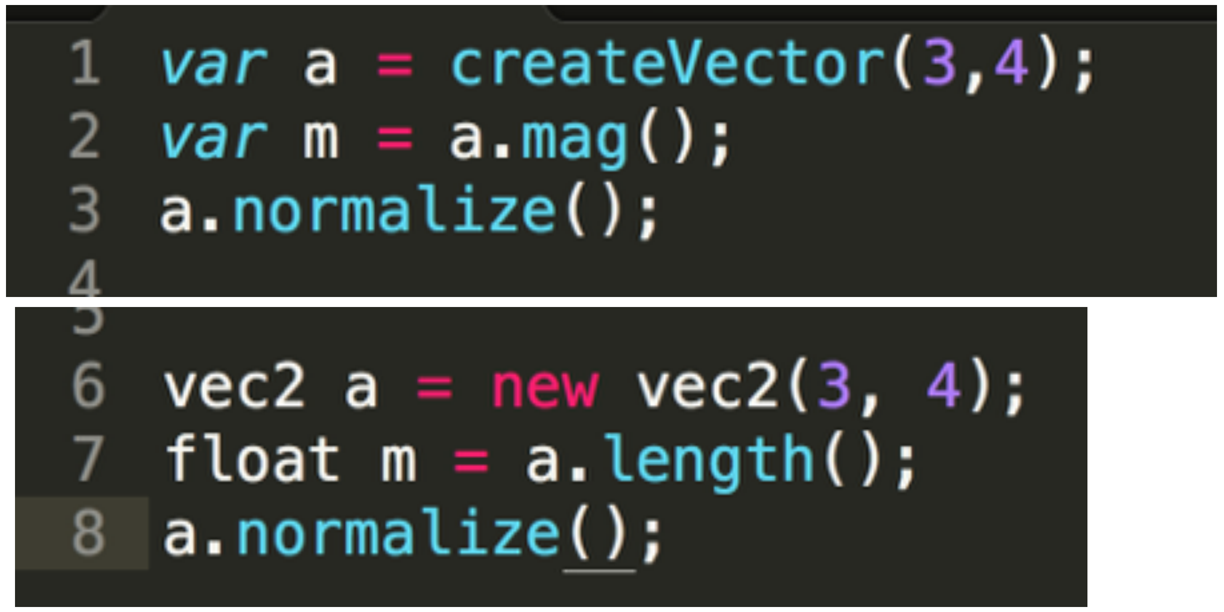
```
var a = createVector(3,4);
var b = 5;
var c = p5.Vector.mult(a, b);
```



```
vec2 a = new vec2(3, 4);
float n = 5;
a.mult(n);
```

Core Elements of Box2D

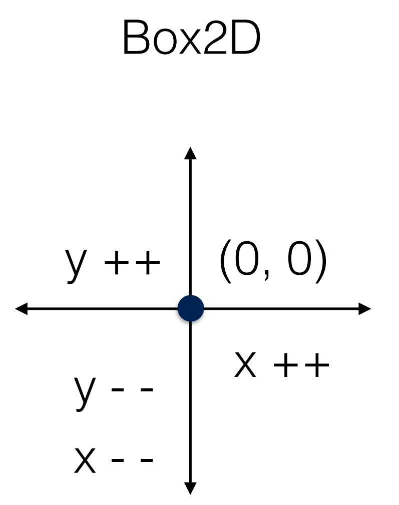
Vec2

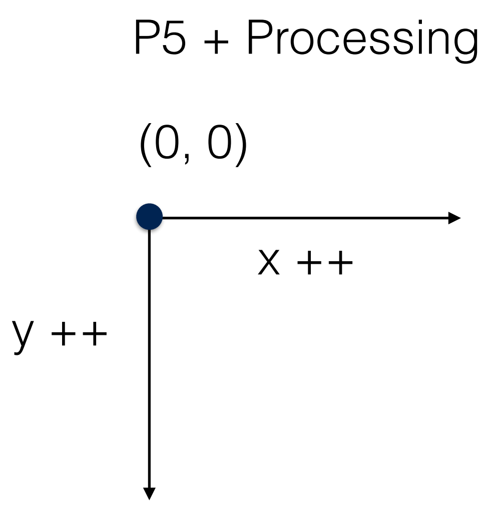


```
var a = createVector(3,4);
var m = a.mag();
a.normalize();


vec2 a = new vec2(3, 4);
float m = a.length();
a.normalize();
```

Core Elements of Box2D

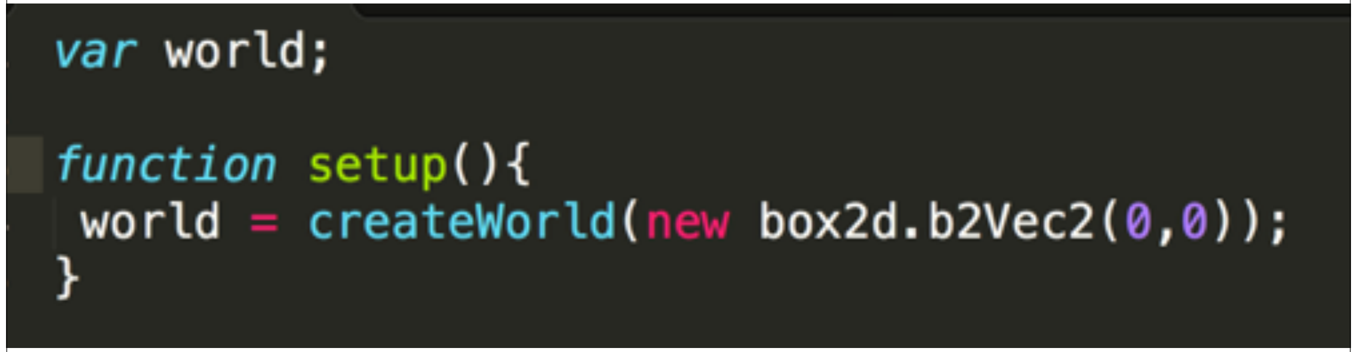
Vec2





Using Box2D

0. Put Box2D in the libraries folder
1. Reference the libraries folder in your index.html file
2. Create world

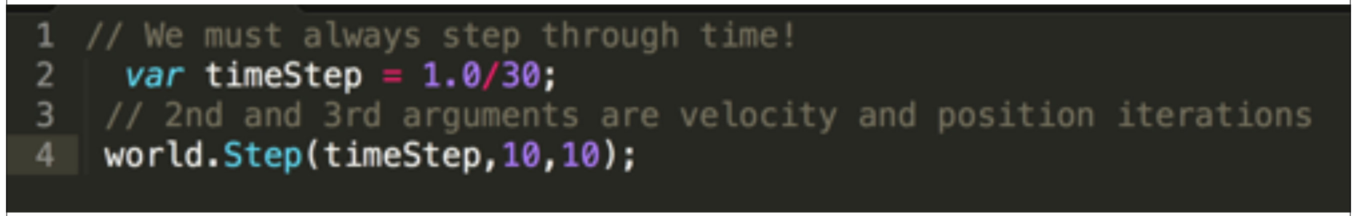


```
var world;

function setup(){
  world = createWorld(new box2d.b2Vec2(0,0));
}
```

Main code

Setting up Box2D

0. Put Box2D in the libraries folder
1. Reference the libraries folder in your index.html file
2. Create world
3. Step through time

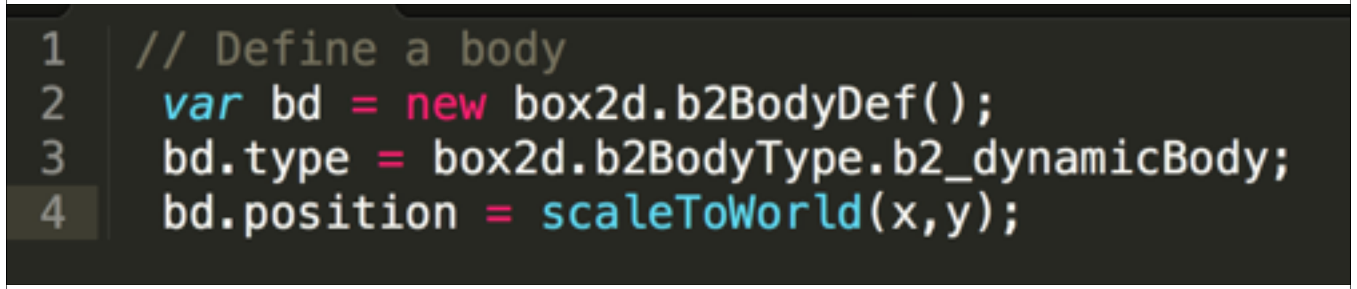


```
// We must always step through time!
 var timeStep = 1.0/30;
// 2nd and 3rd arguments are velocity and position iterations
world.Step(timeStep,10,10);
```

main code

Setting up Box2D

0. Put Box2D in the libraries folder
1. Reference the libraries folder in your index.html file
2. Create world
3. Step through time
4. Create body DEFINITIONS



```
// Define a body
 var bd = new box2d.b2BodyDef();
 bd.type = box2d.b2BodyType.b2_dynamicBody;
 bd.position = scaleToWorld(x,y);
```

class
always create you body definitions first
configure the body - properties + attributes, position

Setting up Box2D

0. Put Box2D in the libraries folder
1. Reference the libraries folder in your index.html file
2. Create world
3. Step through time
4. Create body DEFINITIONS

    A. Set the body type:

    - Dynamic
    - Static
    - Kinematic

Setting up Box2D

4. Create body DEFINITIONS

A. Set the body type:

Dynamic	Static	Kinematic
• fully simulated	• can't move	• can be moved manually by setting velocity
• collides	• platforms	• user controlled objects can only collide w/dynamic bodies
• responds to forces	• boundaries	

Setting up Box2D

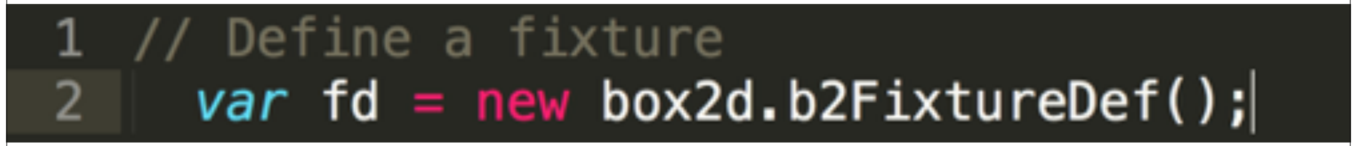
4. Create body DEFINITIONS

    A. Set other attributes:

    - fixed rotation - boolean
    - linear damping - float
    - angular damping - float
    - bullets - boolean

Setting up Box2D

0. Put Box2D in the libraries folder
1. Reference the libraries folder in your index.html file
2. Create world
3. Step through time
4. Create body DEFINITIONS
5. Create fixture DEFINITIONS



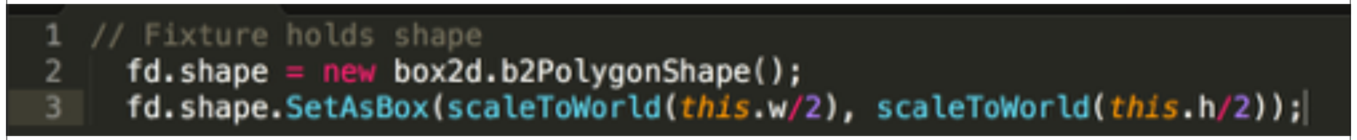
```
// Define a fixture
  var fd = new box2d.b2FixtureDef();
```

body alone doesn’t exit in the world
needs to have mass - a shape

but in order to attach a shape to a body, you need a fixture
fixture holds the shape

Setting up Box2D

0. Put Box2D in the libraries folder
1. Reference the libraries folder in your index.html file
2. Create world
3. Step through time
4. Create body DEFINITIONS
5. Create fixture DEFINITIONS
6. Create shape and attach the shape to the fixture

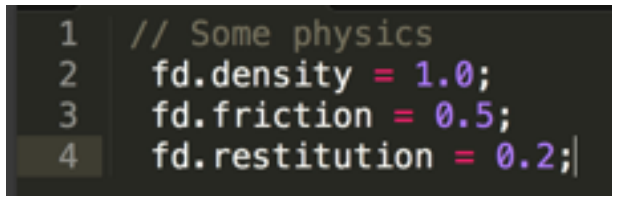


```
// Fixture holds shape
  fd.shape = new box2d.b2PolygonShape();
  fd.shape.SetAsBox(scaleToWorld(this.w/2), scaleToWorld(this.h/2));
```

shapes keep track of all the necessary collision geometry
can attach multiple shapes to a body to get more complex forms

Setting up Box2D

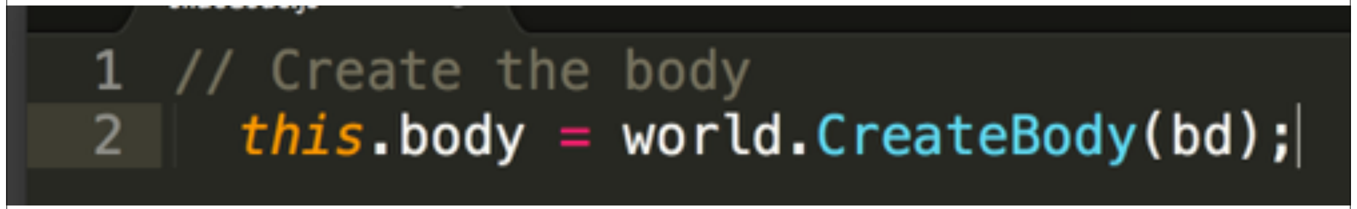
0. Put Box2D in the libraries folder
1. Reference the libraries folder in your index.html file
2. Create world
3. Step through time
4. Create body DEFINITIONS
5. Create fixture DEFINITIONS
6. Add the shape to the fixture
7. Add the physics to the fixture



```
// Some physics
 fd.density = 1.0;
 fd.friction = 0.5;
 fd.restitution = 0.2;
```

Setting up Box2D

8. Create the body in the world, passing it the definition

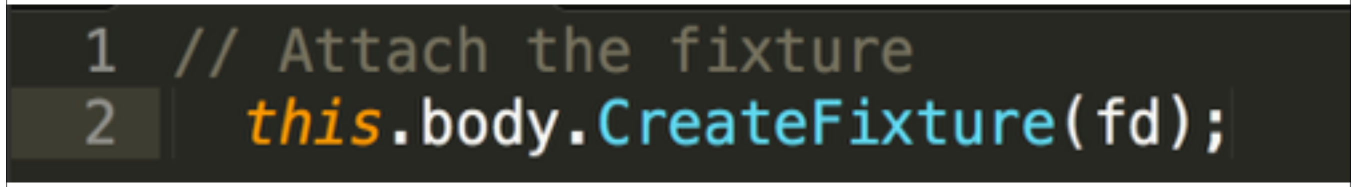


```
// Create the body
   this.body = world.CreateBody(bd);
```

Setting up Box2D

8. Create the body in the world, passing it the definition

9. Attach the fixture to the body



```
// Attach the fixture
   this.body.CreateFixture(fd);
```