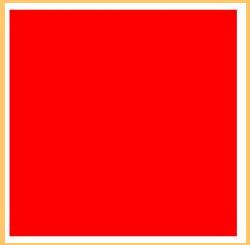
CS 160

Week 2, Section B

Intro to Paper.js

Lucy White, 6/30/2022

Drawing with HTML and CSS



```
<div id="square"></div>
```

```
#square {
  width: 100px;
  height: 100px;
  background-color: red;
}
```

```
<div id="triangle"></div>
```

```
#triangle{
  width: 0;
  height: 0;
  border-top-width: 100px;
  border-top-style: solid;
  border-top-color: transparent;
  border-right-width: 100px;
  border-right-style: solid;
  border-right-color: red;
}
```

?

Enter the canvas element

Added in HTML5, the HTML `<canvas>` element can be used to draw graphics via scripting in JavaScript.
It can be used for:

- Drawing graphs
- Making photo compositions
- Creating animations
- Video processing
- Rendering

```
...
<body>
    <canvas id="myCanvas"></canvas>
</body>
...
```

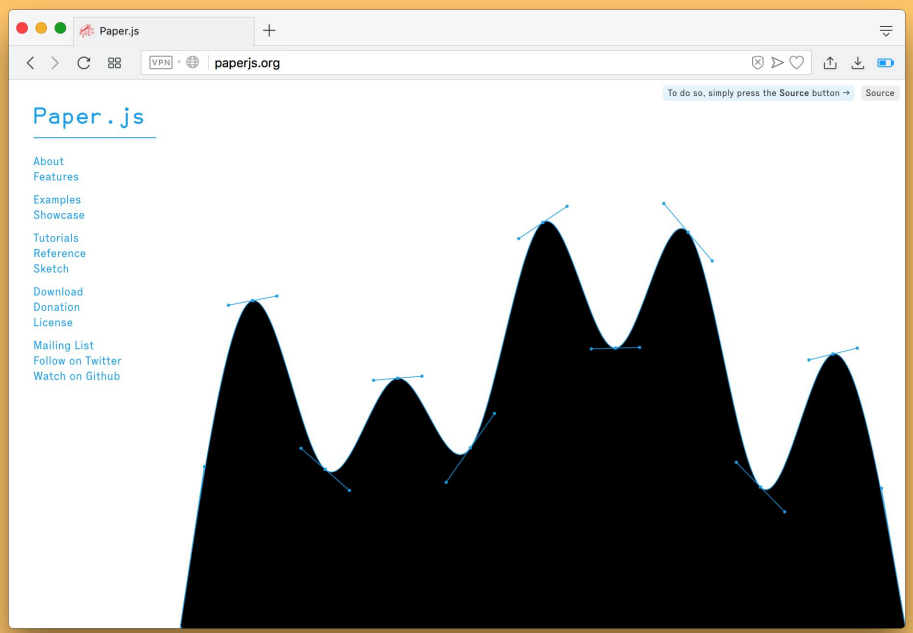
Enter the canvas element

We'll be using a framework called paper.js, which is built on top of the HTML5 canvas. Paper.js includes tools for animating vector graphics. It is usually used along with Paperscript, which its own flavor of Javascript.

- Paperscript vs Javascript

```
...
<body>
    <canvas id="myCanvas"></canvas>
</body>
...
```

“The SWiss Army Knife of Vector Graphics Scripting”



paperjs.org

Getting started with paper.js

```
<!DOCTYPE html>
<html>
<head>



</head>
<body>

</body>
</html>
```

Getting started with paper.js

```
<!DOCTYPE html>
<html>
<head>



</head>
<body>
    <canvas id="myCanvas" resize></canvas>
</body>
</html>
```

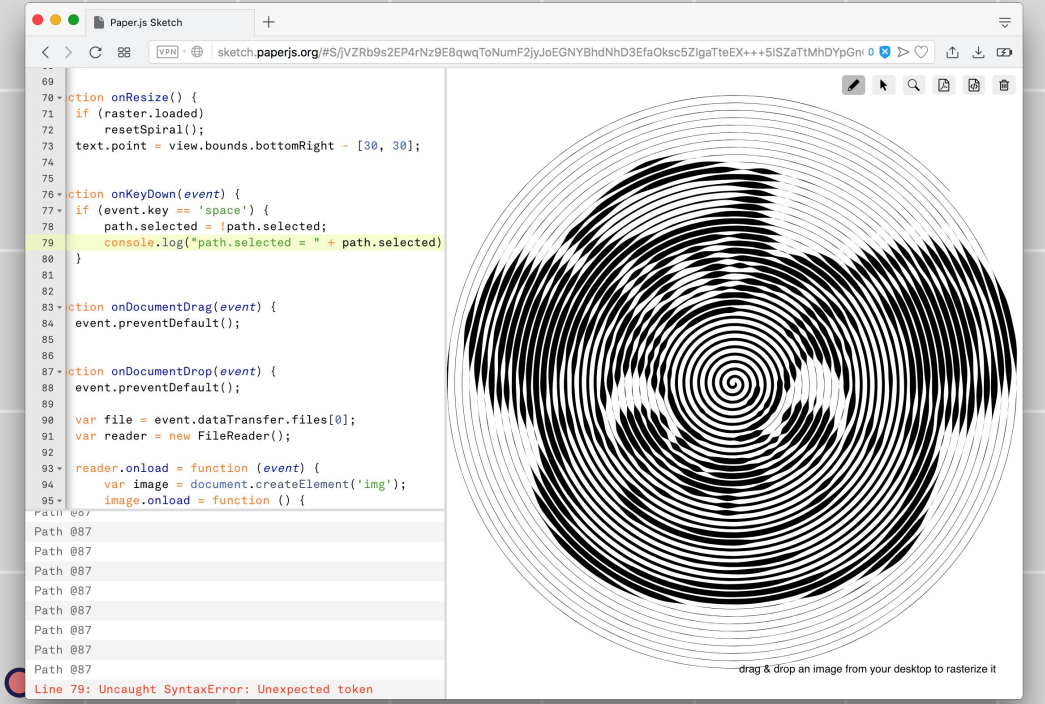
Getting started with paper.js

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript" src="js/paper.js"></script>

</head>
<body>
    <canvas id="myCanvas" resize></canvas>
</body>
</html>
```

Getting started with paper.js

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript" src="js/paper.js"></script>
<script type="text/paperscript" canvas="myCanvas"></script>
</head>
<body>
    <canvas id="myCanvas" resize></canvas>
</body>
</html>
```

online editor: sketch.paperjs.org/



Points and paths

```
// arg 1 = x pos, arg 2 = y pos
var point1 = new Point(10, 10);
var point2 = new Point(100, 100);
```

output

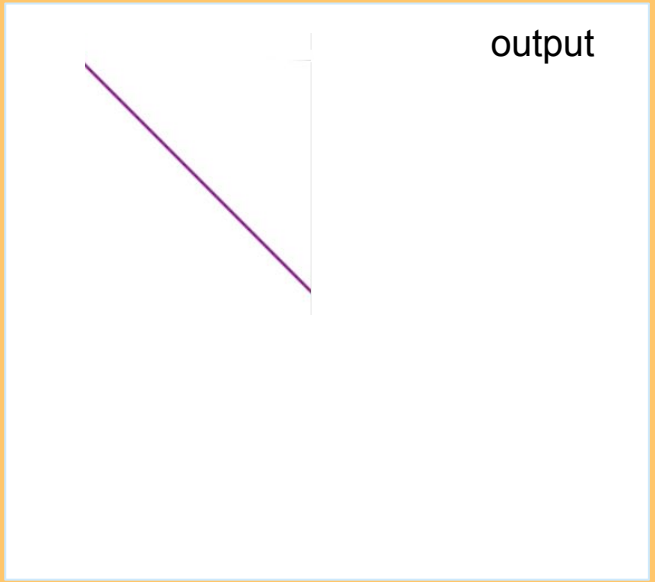
Points and paths

```
// arg 1 = x pos, arg 2 = y pos
var point1 = new Point(10, 10);
var point2 = new Point(100, 100);

var myPath = new Path();
myPath.strokeColor = "purple"
```

output

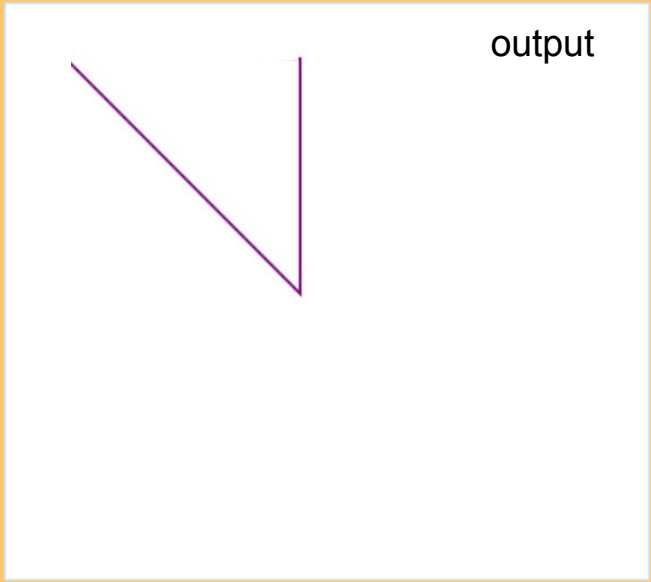
Points and paths

```
// arg 1 = x pos, arg 2 = y pos
var point1 = new Point(10, 10);
var point2 = new Point(100, 100);

var myPath = new Path();
myPath.strokeColor = "purple"

// args* = points
myPath.add(point1, point2);
```



Points and paths

```
// arg 1 = x pos, arg 2 = y pos
var point1 = new Point(10, 10);
var point2 = new Point(100, 100);

var myPath = new Path();
myPath.strokeColor = "purple"

// args* = points
myPath.add(point1, point2);
myPath.add(new Point(200, 15));
```



Points and paths

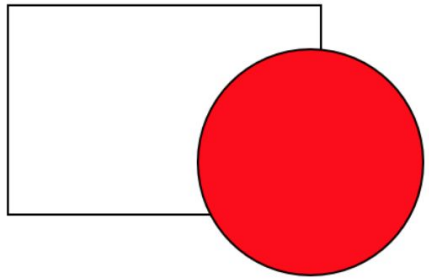
```
// arg 1 = x pos, arg 2 = y pos
var point1 = new Point(10, 10);
var point2 = new Point(100, 100);

var myPath = new Path();
myPath.strokeColor = "purple"
// args* = points
myPath.add(point1, point2);
myPath.add(new Point(200, 15));
myPath.removeSegment(1);
```

output

Predefined shapes

```
// args: top left point and size
var rect = new Rectangle(new Point(5, 25),
                         new Size(150, 100));
var newPath = new Path.Rectangle(rect);
newPath.strokeColor = "black";

// args: center point, radius length
var myCircle = new Path.Circle(
                        new Point(150, 100), 54);
myCircle.strokeColor = 'black';
myCircle.fillColor = 'red';
```

output



Item

Using the Item class, you can modify the items (paths, points, text, etc.) in your Paper.js project. Some properties of the Item class include:

- `className` - 'Group', 'Layer', 'Path', 'CompoundPath', 'Shape', 'Raster', 'SymbolItem', 'PointText'
- `name` - get/set string name, which you can use to reference the item with
- `style` - contains fillColor, strokeColor, and strokeWidth. Helpful when cloning items or applying the same style to multiple items
- `visible` - get/set boolean to hide or unhide an item
- `opacity` - opacity between 0 and 1

More info on items

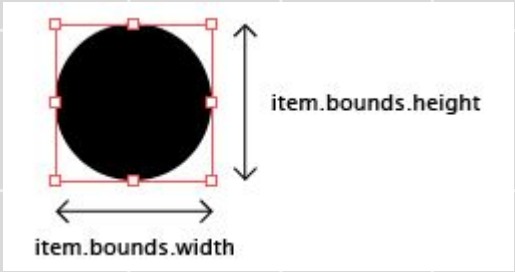
Item - positioning

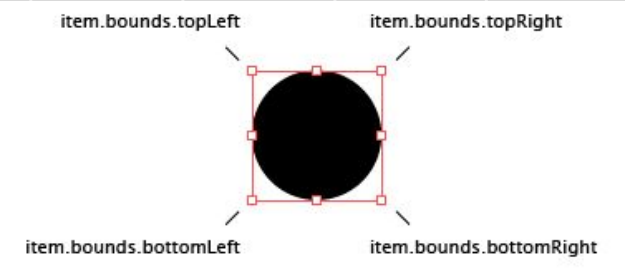
Commonly used properties

- **`position`** - the `rectangle.center` of the item's bounds rectangle
- **`bounds`** - The bounding rectangle of the item excluding stroke width.

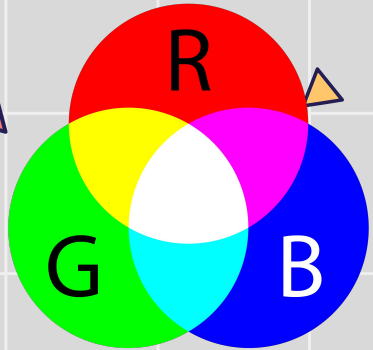
Commonly used scaling functions

- **`translate`**`(delta)` - moves the item to the arg Point
- **`rotate`**`(angle, (optional) center)` - Rotates the item by a given angle around the given center point.
- **`scale`**`(scale, (optional) center)` - scales (resizes) item from either its center point, or from the point provided in arg

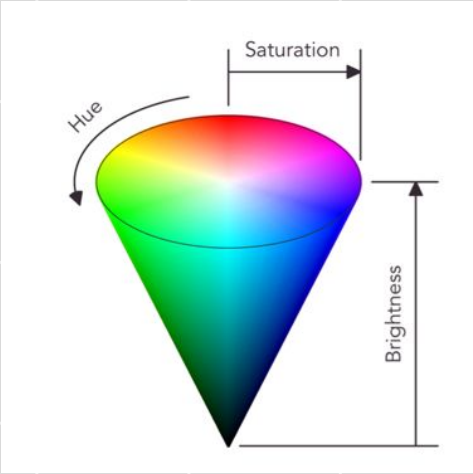




Color - RGB, Hex, Gradient, and HSB



#666600	#666633	#666666	#666699	#6666cc	#6666FF
#663300	#663333	#663366	#663399	#6633cc	#6633FF
#660000	#660033	#660066	#660099	#6600cc	#6600FF
#996600	#996633	#996666	#996699	#9966cc	#9966FF
#993300	#993333	#993366	#993399	#9933cc	#9933FF
#990000	#990033	#990066	#990099	#9900cc	#9900FF
#cc6600	#cc6633	#cc6666	#cc6699	#cc66cc	#cc66FF
#cc3300	#cc3333	#cc3366	#cc3399	#cc33cc	#cc33FF
#cc0000	#cc0033	#cc0066	#cc0099	#cc00cc	#cc00FF
#FF6600	#FF6633	#FF6666	#FF6699	#FF66cc	#FF66FF
#FF3300	#FF3333	#FF3366	#FF3399	#FF33cc	#FF33FF
#FF0000	#FF0033	#FF0066	#FF0099	#FF00cc	#FF00FF
#000000	#666666	#999999	#cccccc	#dddddd	#FFFFFF



Color - constructors

```
// RGB
Color(red, green, blue, (optional) alpha)

// pass in a percent (1 = white, 0 = black, between = gray)
Color(gray, (optional) alpha)

// Create a HSB, HSL or gradient color from the param object
Color(object)

// Creates a gradient Color object.
Color(gradient, origin, destination, (optional) highlight)
```

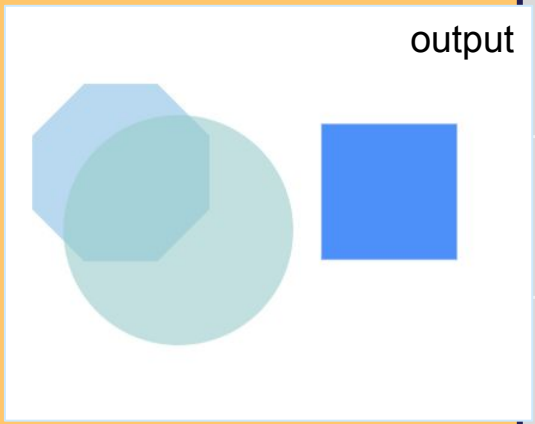
Color - RGB

```
// RGB colors! args: red, green, blue,
// and alpha (Opacity)
oct.fillColor = new Color(0, .5, .8, .3)
circle.fillColor = new Color(.6, .8, .8,
.6);
```

output

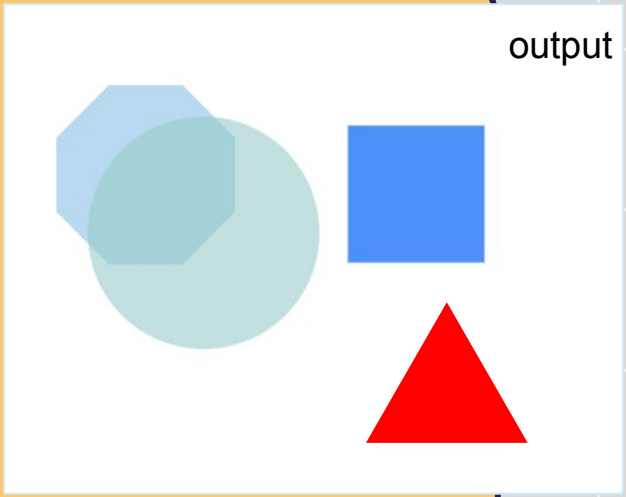
Color - RGB and Hex

```
// RGB colors! args: red, green, blue,
// and alpha (Opacity)
oct.fillColor = new Color(0, .5, .8, .3)
circle.fillColor = new Color(.6, .8, .8, .6);

// hex colors
rectangle.fillColor = '#498dfc';
```



Color - RGB and Hex

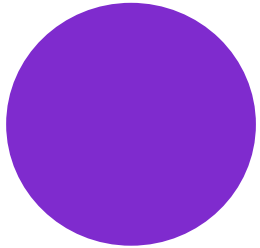
```
// RGB colors! args: red, green, blue,
// and alpha (Opacity)
oct.fillColor = new Color(0, .5, .8, .3)
circle.fillColor = new Color(.6, .8, .8, .6);


// hex colors
rectangle.fillColor = '#498dfc';
// named colors
myTriangle.fillColor = 'red';
```



Color - adjusting hsb

```
var circle = new Path.Circle(
                    new Point(90, 90), 60);
circle.fillColor = new Color(.6, 0, .8, 1);
```

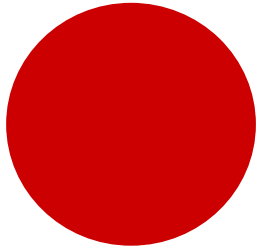
output



Color - adjusting hsb

```
var circle = new Path.Circle(
                    new Point(90, 90), 60);
circle.fillColor = new Color(.6, 0, .8, 1);
circle.fillColor.hue += 60;
```

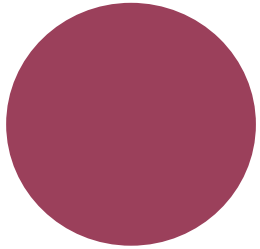
output



Color - adjusting hsb

```
var circle = new Path.Circle(
                    new Point(90, 90), 60);
circle.fillColor = new Color(.6, 0, .8, 1);
circle.fillColor.hue += 60;
circle.fillColor.saturation = .5;
```

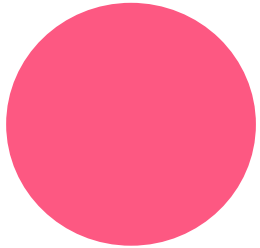
output



Color - adjusting hsb

```
var circle = new Path.Circle(
                    new Point(90, 90), 60);
circle.fillColor = new Color(.6, 0, .8, 1);
circle.fillColor.hue += 60;
circle.fillColor.saturation = .5;
circle.fillColor.brightness += .4;
```

output



Handlers and Events

```
function onMouseDown(event) {
    // ...
}

function onMouseDrag(event) {
    // ...
}

function onMouseUp(event) {
    // ...
}

function onMouseMove(event) {
    // ...
}
```

Use handlers to make drawings interactive!

Handlers receive an `Event` object. Events hold information about the position and direction of the action.

Event properties

1. `type` - 'mousedown', 'mouseup', 'mousemove', or 'mousedrag'
2. `point` - position of mouse when the event was fired.
3. `lastPoint` - position of mouse when the previous event was fired
4. `downPoint` - position of mouse when the mouse button was last clicked
5. `middlePoint` - midpoint between lastPoint and point.
6. `delta` - difference between the current and last position of the mouse when the event was fired
7. `item` - the item at the position of the mouse (if any).
8. `count` - the number of time the event was called

Binding Mouse Handlers

```
// binding an event to an element
var myPath = new Path();
myPath.strokeColor = 'orange';

myCircle.onMouseDrag = function(event) {
    this.position = event.point;
}
```

```
// binding an event to whole canvas
var myPath = new Path();
myPath.strokeColor = 'orange';

function onMouseDrag(event) {
    myCircle.position = event.point;
}
```

Project Hierarchy

Each paper.js project has a list of layers: `project.layers`
Every new project starts out with one layer, which you can access using `project.activeLayer`. All new items are automatically added to this layer, as elements of its `children` array.

```
var path1 = new Path.Rectangle(view.center, new Size(50,50));
var path2 = new Path.Circle(view.center, new Size(100,100));

project.activeLayer.children[0].fillColor = 'red';
project.activeLayer.children[1].strokeColor = 'green';
```

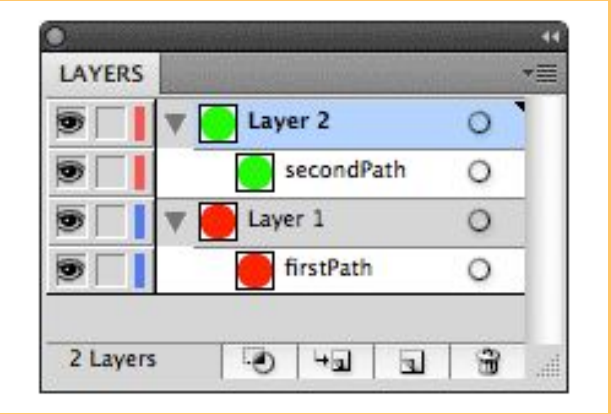
Project Hierarchy

You can also access children items from the children array by name:

```
var path1 = new Path.Rectangle(view.center, new Size(50,50));
path1.name = "my favorite square";

project.activeLayer.children['my favorite square'].fillColor = "blue";
```

Project Hierarchy - layers

Create new layers with the `Layer` class. When you create a new layer, it becomes the new `project.activeLayer` (if you want to avoid this, use the Group class instead)



Project Hierarchy - layers examples

```
var path1 = new Path.Circle(new Point(80, 50), 35);
path1.fillColor = "blue"

// Create a new layer
var secondLayer = new Layer();
var path2 = new Path.Circle(new Point(150, 50), 35);
var path3 = new Path.Circle(new Point(220, 50), 35);

secondLayer.fillColor = 'green';
// Both path2 and path3 will be green. What will happen if I
set project.activeLayer.fillColor = "blue"?
```

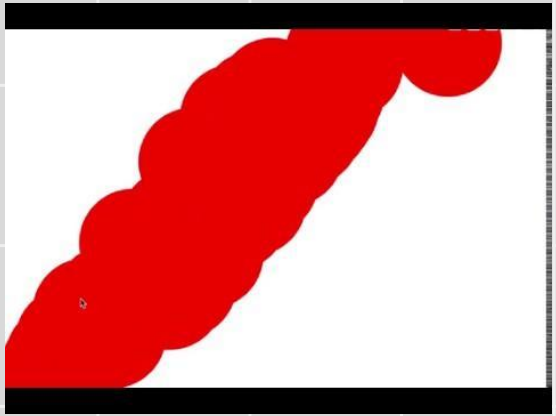
Project Hierarchy - groups examples

When using groups, you must explicitly add items to the group's list of children. Groups do not affect what the `project.activeLayer` is

```
// new group: doesn't have any
items in it
var group = new Group();

// add paths to the group
group.addChild(path1);
group.addChild(path2);
group.strokeColor = 'black';
```

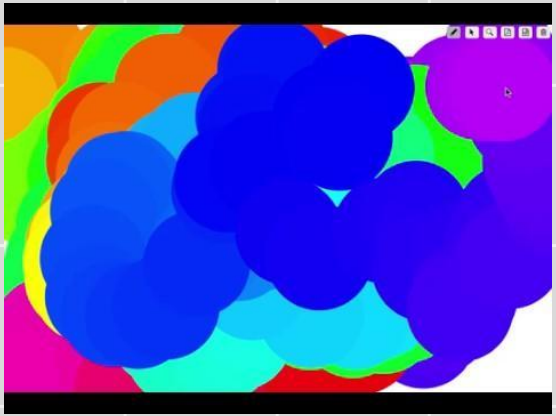
EXERCISE: sketch.paperjs.org

Tasks



Task 1: Paintbrush

Start with creating a circle on screen (ignore event handlers at first). Use event.middlePoint.



Task 2: Color Changing Paintbrush

Use the hsb Color constructor.

Task 3: Paint "worm"

Use project hierarchy.

Task 4: make something cooler and show the class!