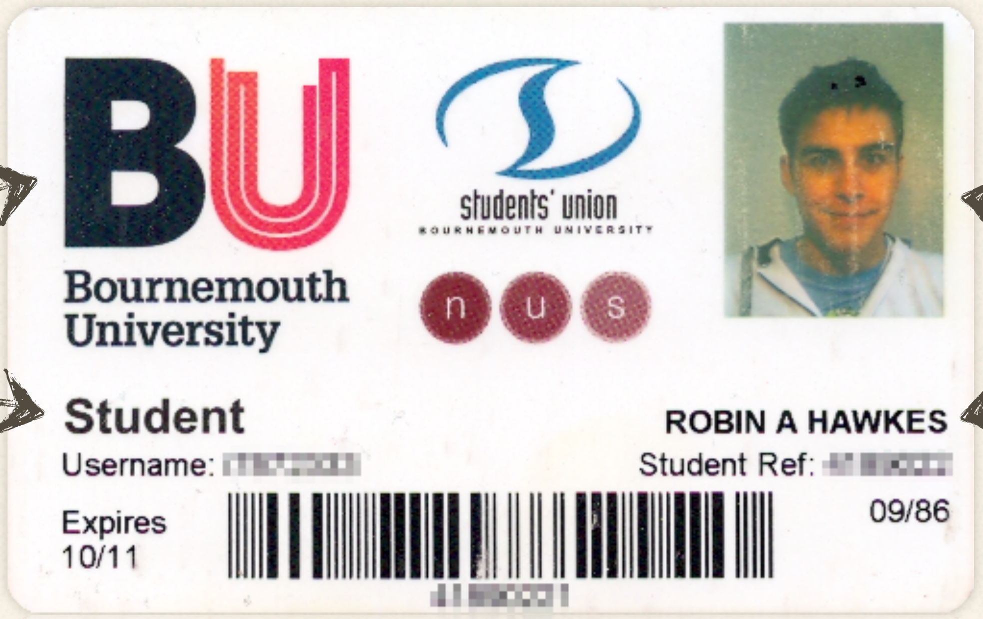
HTML5 Canvas

The Future of
Graphics on the Web

Rob Hawkes

@robhawkes for you social media folk
rawkes.com if you want to see more



Bournemouth University

Student
Username: [redacted]
Expires 10/11

ROBIN A HAWKES
Student Ref: [redacted]
09/86

“Canvas is my favourite part of HTML5, alongside its video and audio support”

Myself, at some point

So what is canvas?

An overview of canvas

- 2D drawing platform within the browser
- Uses nothing more than JavaScript and HTML – no plugins
- Extensible through a JavaScript API
- Created by Apple for dashboard widgets
- Now openly developed as a W3C spec

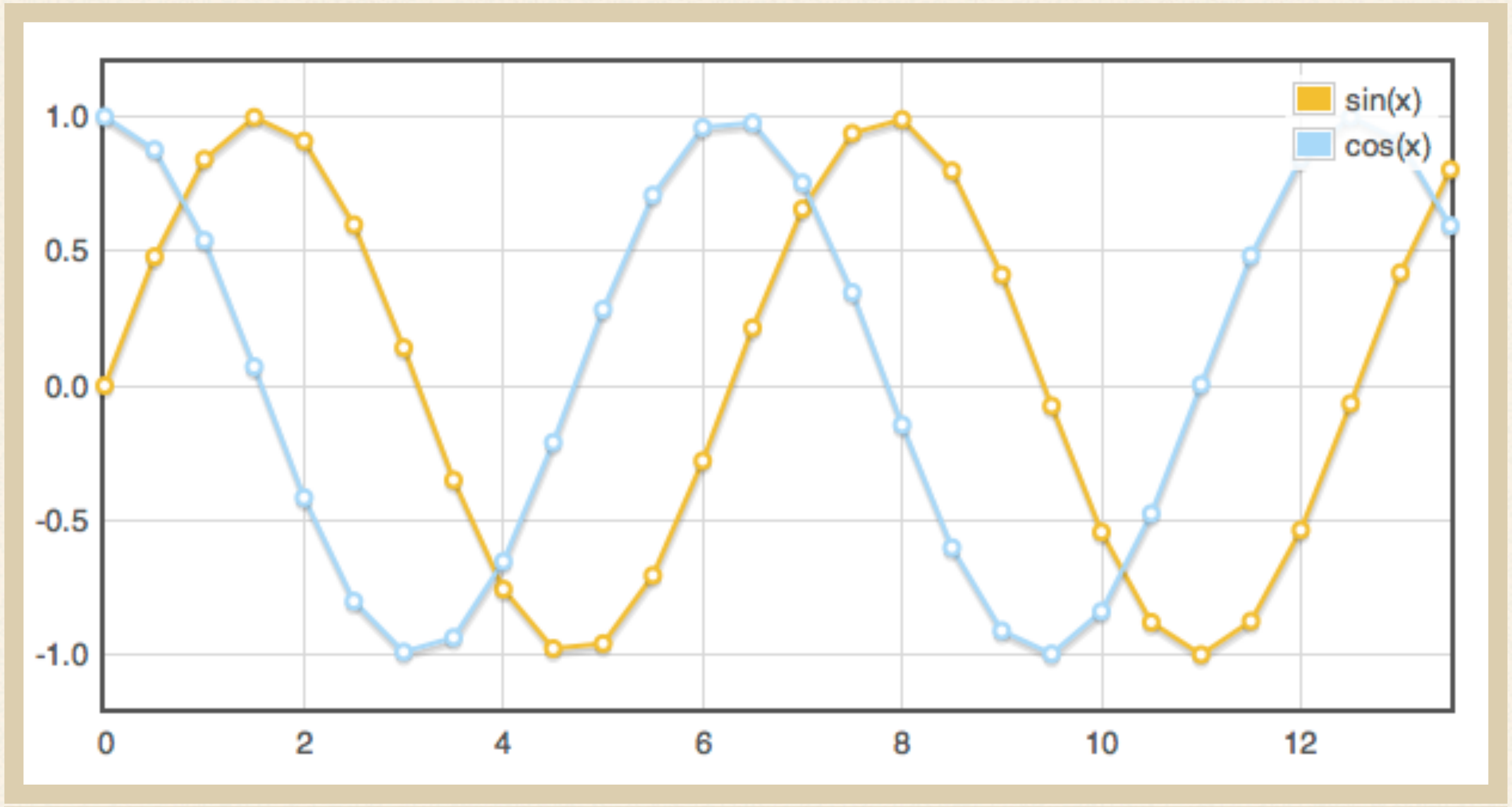
Bitmap vs. vector

- Canvas is a bitmap system
  - *Everything is drawn as a single, flat, picture*
  - *Changes require the whole picture to be redrawn*
- SVG is a vector system
  - *Elements to be drawn are separate DOM objects*
  - *They can be manipulated individually*
- SVG isn’t part of HTML5
  - *Future isn’t as rosy as canvas’*

Browser support

- Most modern browsers
  - *Safari*
  - *Chrome*
  - *Firefox*
  - *Opera*
- No Internet Explorer support by default
  - *However, there are hacks to get it working*

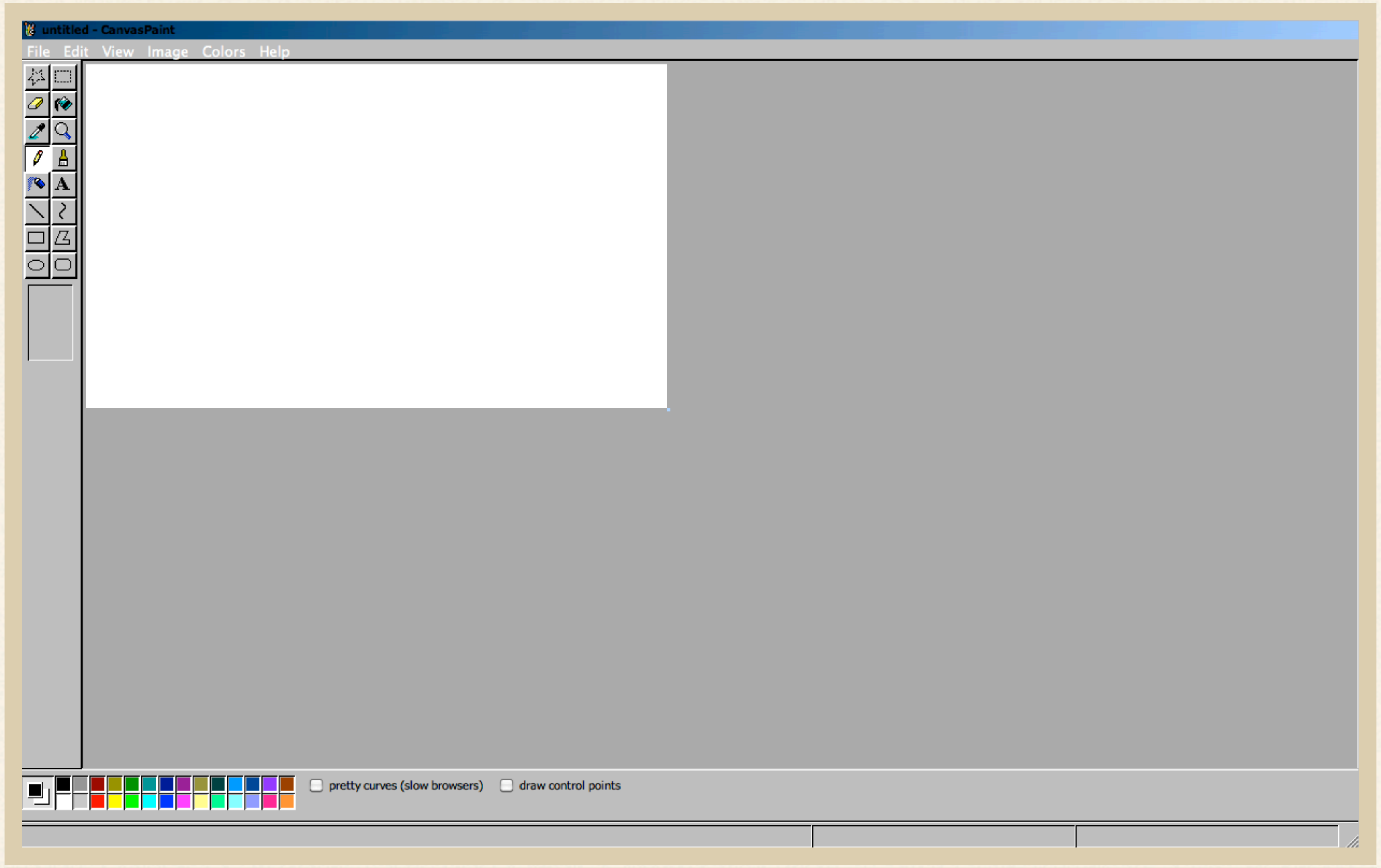
What is it for?



Data visualisation

I hate you, sir Goat!

Animated graphics


untitled - CanvasPaint
File Edit View Image Colors Help
pretty curves (slow browsers)
draw control points

Web applications

GWT Quake II
http://quake2-gwt-port.appspot.com/
Google
HTML5 Canvas

Games

Factored – Top Website Considerations

http://redweb.com/factored/index.html

factored

top website considerations

Click to get started. Then try clicking on a node.

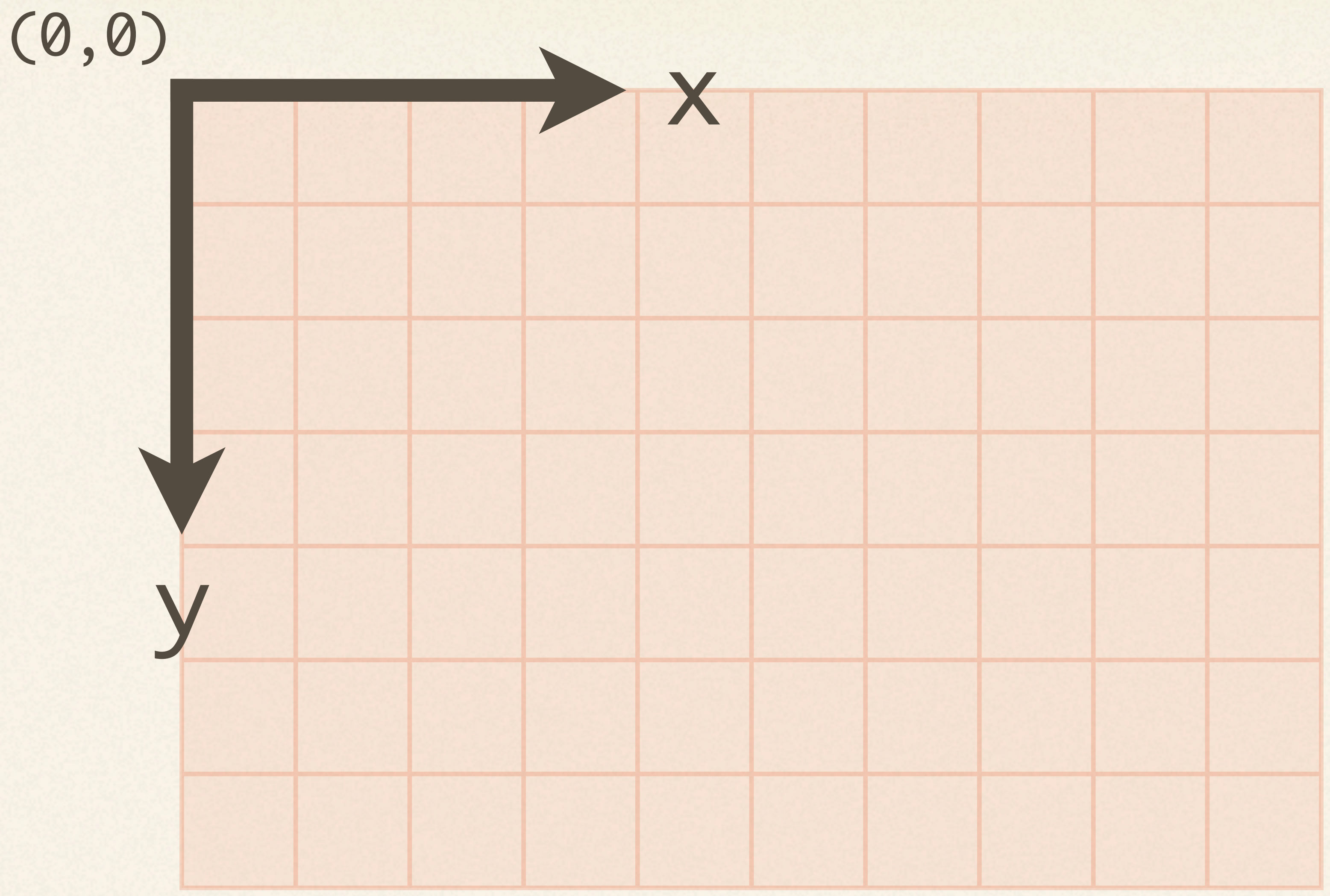
Here's something I made earlier

Getting started

Created using the new HTML5 tag

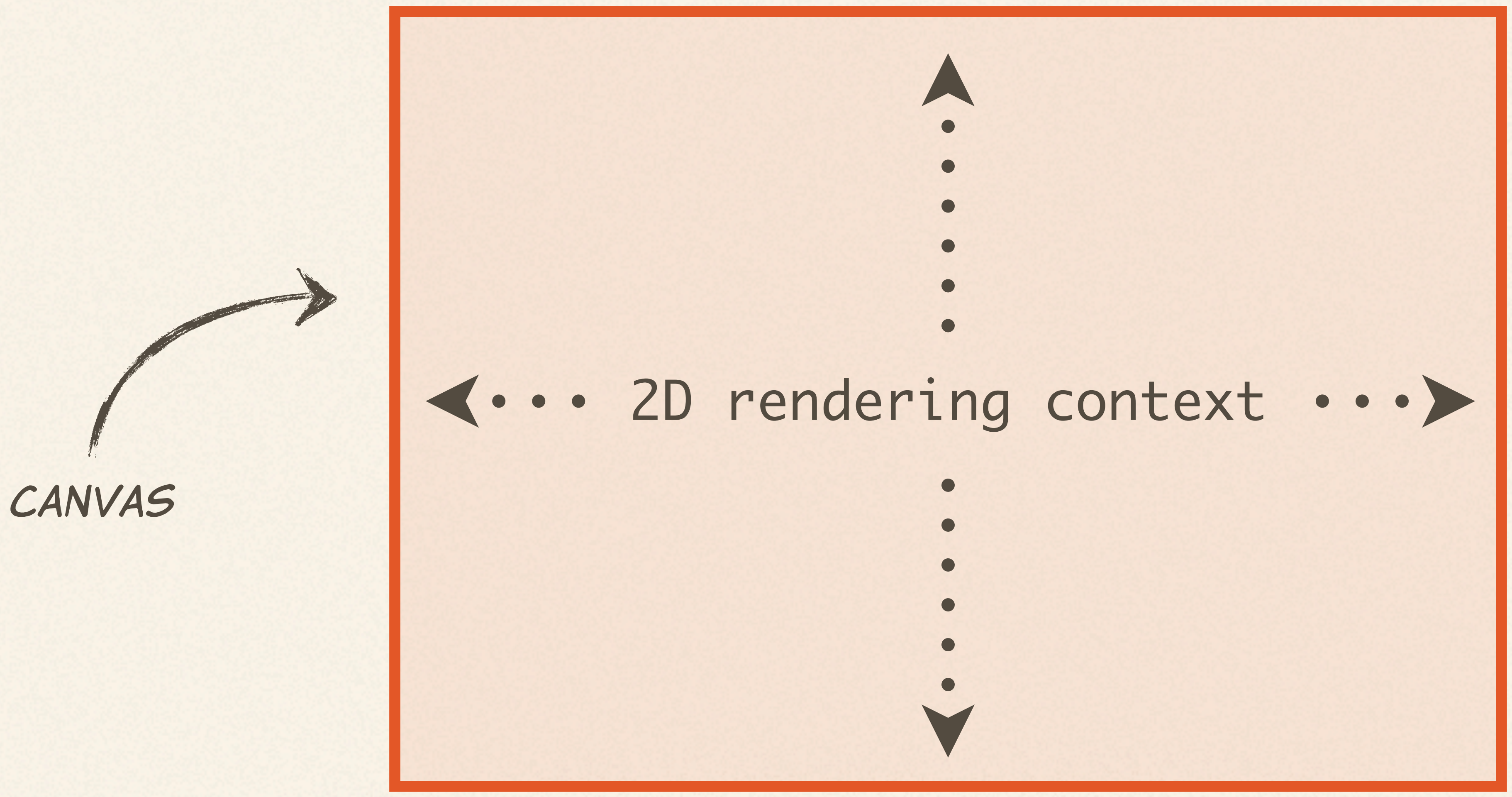
```
<canvas height=”600” width=”800”></canvas>
```

HEIGHT AND WIDTH NEED TO BE SET EXPLICITLY



Uses the standard screen-based coordinate system

Everything is drawn onto the 2D rendering context (ctx)



Use ***getContext()*** to access the 2D rendering context

```
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");
```

THIS IS YOUR FRIEND

```
ctx.fillStyle = 'rgb(255, 0, 0)';
ctx.strokeStyle = 'rgba(0, 255, 0, 0.5)';
```

USE RGBA FOR ALPHA
TRANSPARENCY

fillStyle() and ***strokeStyle()*** define the style of shapes to be drawn

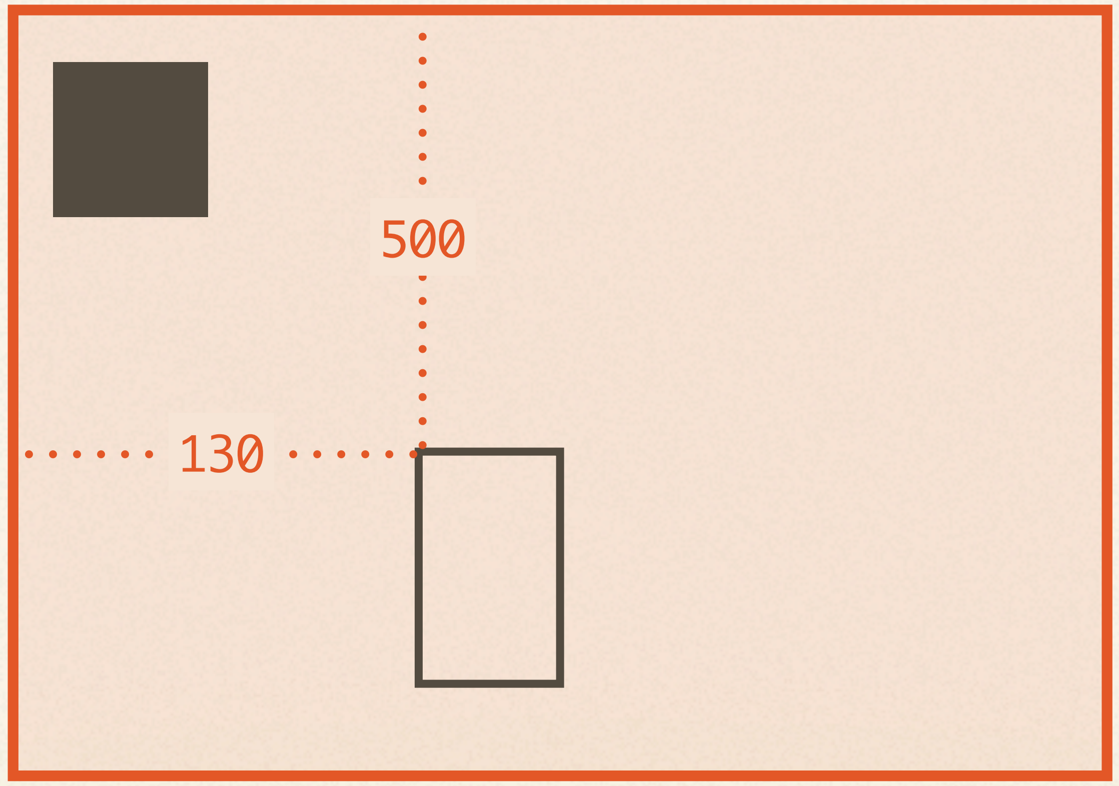
Simple shapes

Method	Action
fillRect(x, y, w, h**)**	Draws a rectangle using the current fill style
strokeRect(x, y, w, h**)**	Draws the outline of a rectangle using the current stroke style
clearRect(x, y, w, h**)**	Clears all pixels within the given rectangle

Simple shapes are drawn without effecting the current path

```
ctx.fillStyle = ‘rgb(65, 60, 50)’;
ctx.fillRect(25, 50, 100, 100);

ctx.strokeStyle = ‘rgb(65, 60, 50)’;
ctx.strokeRect(130, 500, 40, 70);
```



Complex shapes & paths

* Paths are a list of subpaths
* Subpaths are one or more points connected by straight or curved lines
* Rendering context always has a current path
* A new path should be created for each individual shape

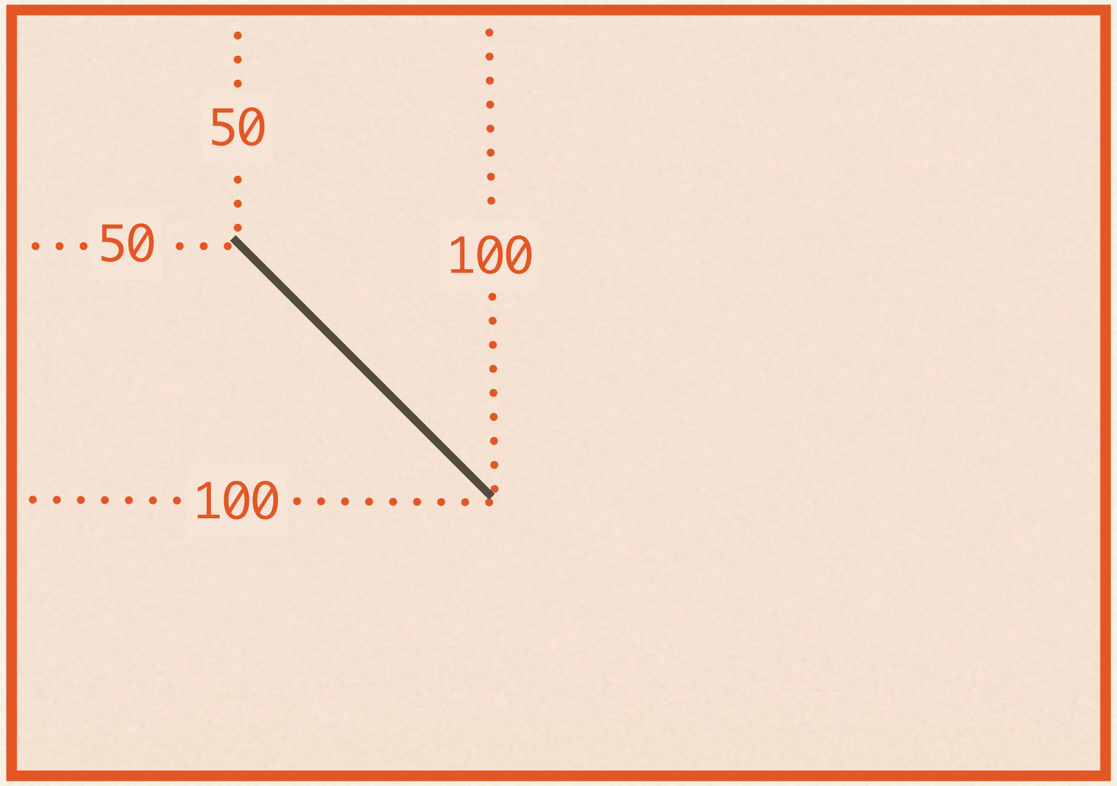
Complex shapes & paths

Method	Action
beginPath()	Resets the current path
closePath()	Closes the current subpath and starts a new one
moveTo(x, y**)**	Creates a new subpath at the given point
fill()	Fills the subpaths with the current fill style
stroke()	Outlines the subpaths with the current stroke style

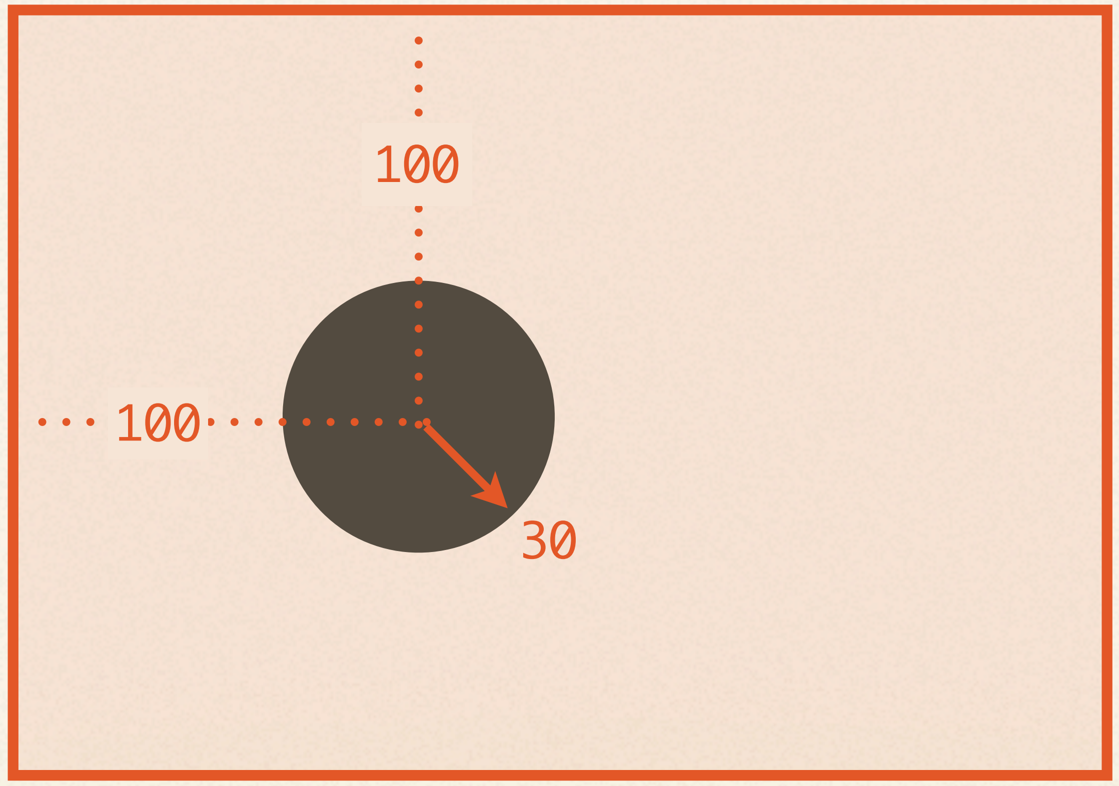
Complex shapes & paths

Method	Action
lineTo(x, y**)**	Draws a straight line from the previous point
rect(x, y, w, h**)**	Adds a new closed rectangular subpath
arc(x, y, radius, startAngle, endAngle, anticlockwise**)**	Adds a subpath along the circumference of the described circle, within the angles defines
arcTo(x1, y1, x2, y2, radius**)**	Adds a subpath connecting two points by an arc of the defined radius
bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y**)**	Adds a cubic Bézier curve with the given control points
quadraticCurveTo(cpx, cpy, x, y**)**	Adds a quadratic Bézier curve with the given control points

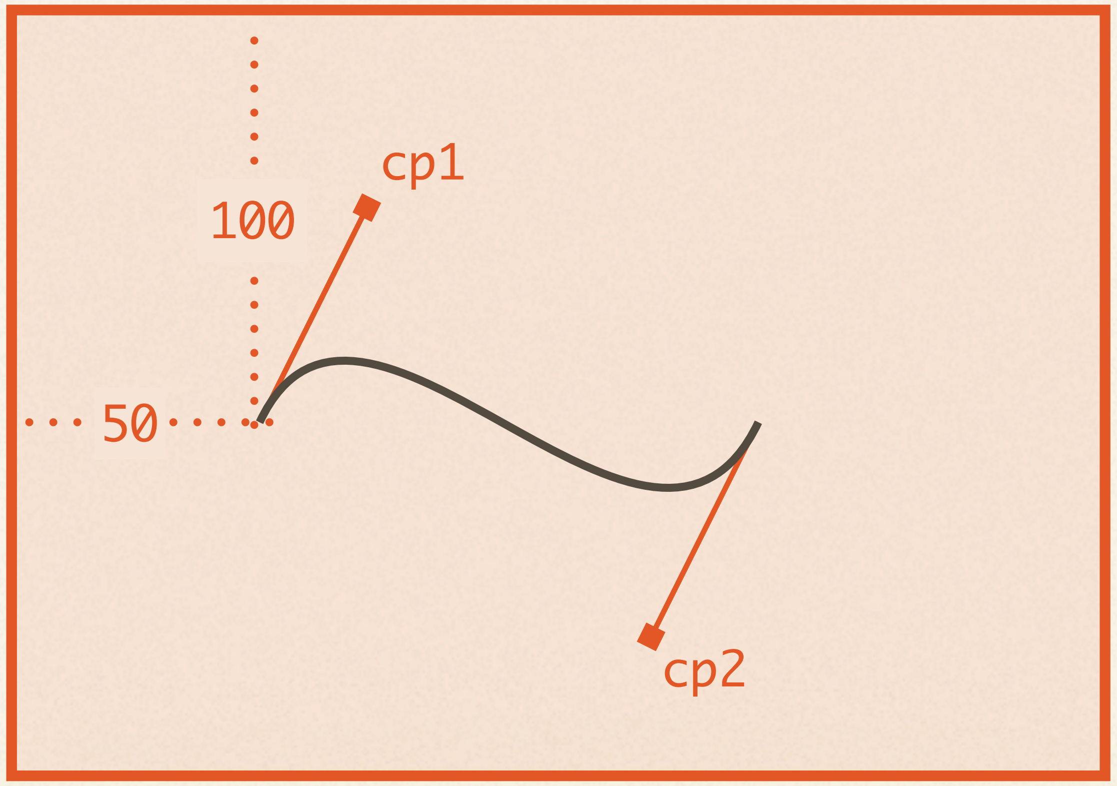
```
ctx.strokeStyle = 'rgb(65, 60, 50)';
ctx.beginPath();
ctx.moveTo(50, 50);
ctx.lineTo(100, 100);
ctx.stroke();
```



```
ctx.fillStyle = ‘rgb(65, 60, 50)’;
ctx.beginPath();
ctx.arc(100, 100, 30, 0, Math.PI*2, true);
ctx.fill();
```



```
ctx.strokeStyle = ‘rgb(65, 60, 50)’;
ctx.beginPath();
ctx.moveTo(50, 100);
ctx.bezierCurveTo(70, 50, 130, 150, 150, 100);
ctx.stroke();
```



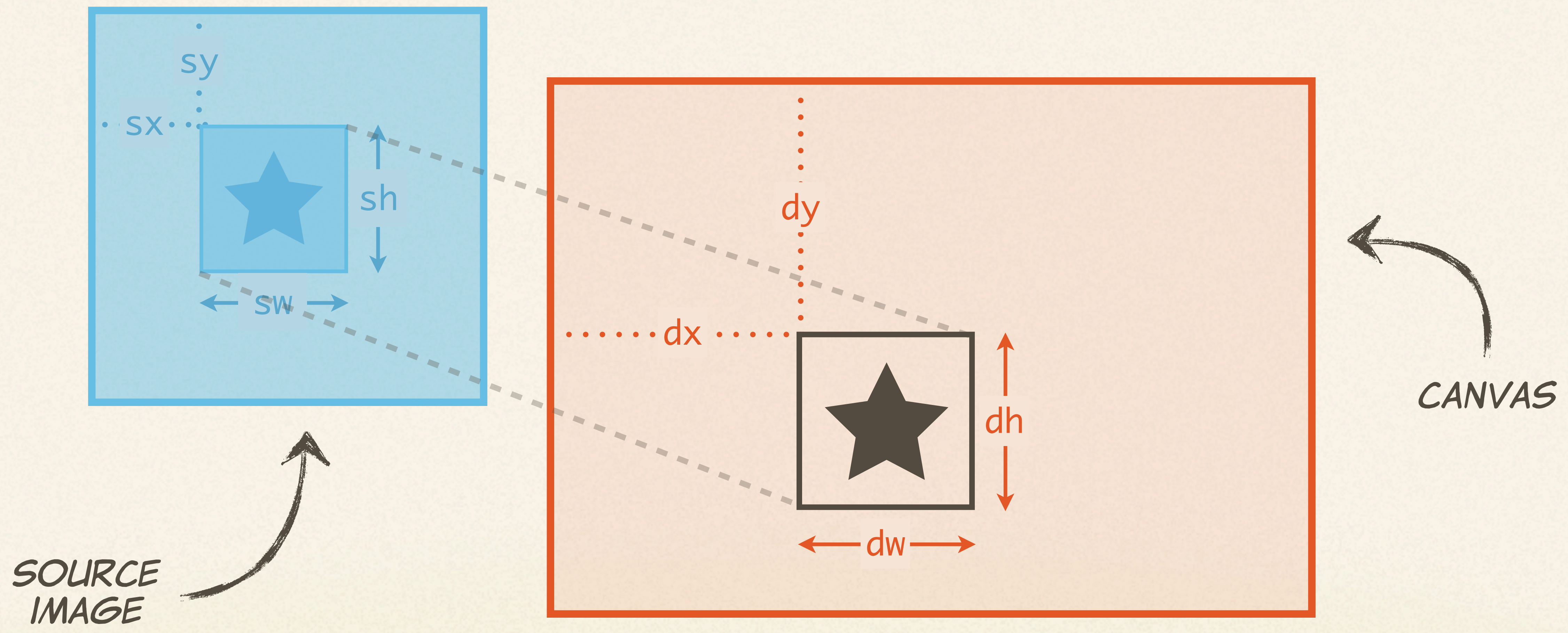
Other cool stuff

* Text
* Shadows & blurring
* Line styles; width, cap, etc.
* Saving state of drawing context
* Transformations

Pixel manipulation

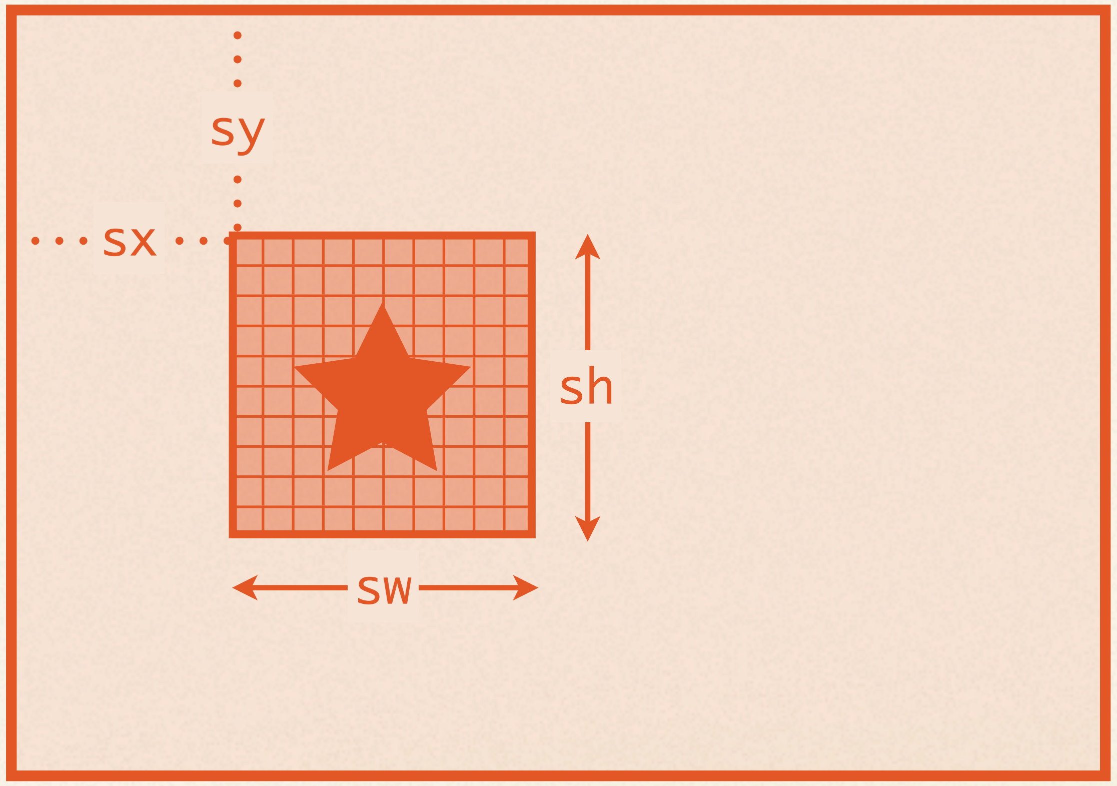
Images can be drawn onto the canvas

```
ctx.drawImage(image, dx, dy);
ctx.drawImage(image, dx, dy, dw, dh);
ctx.drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh);
```

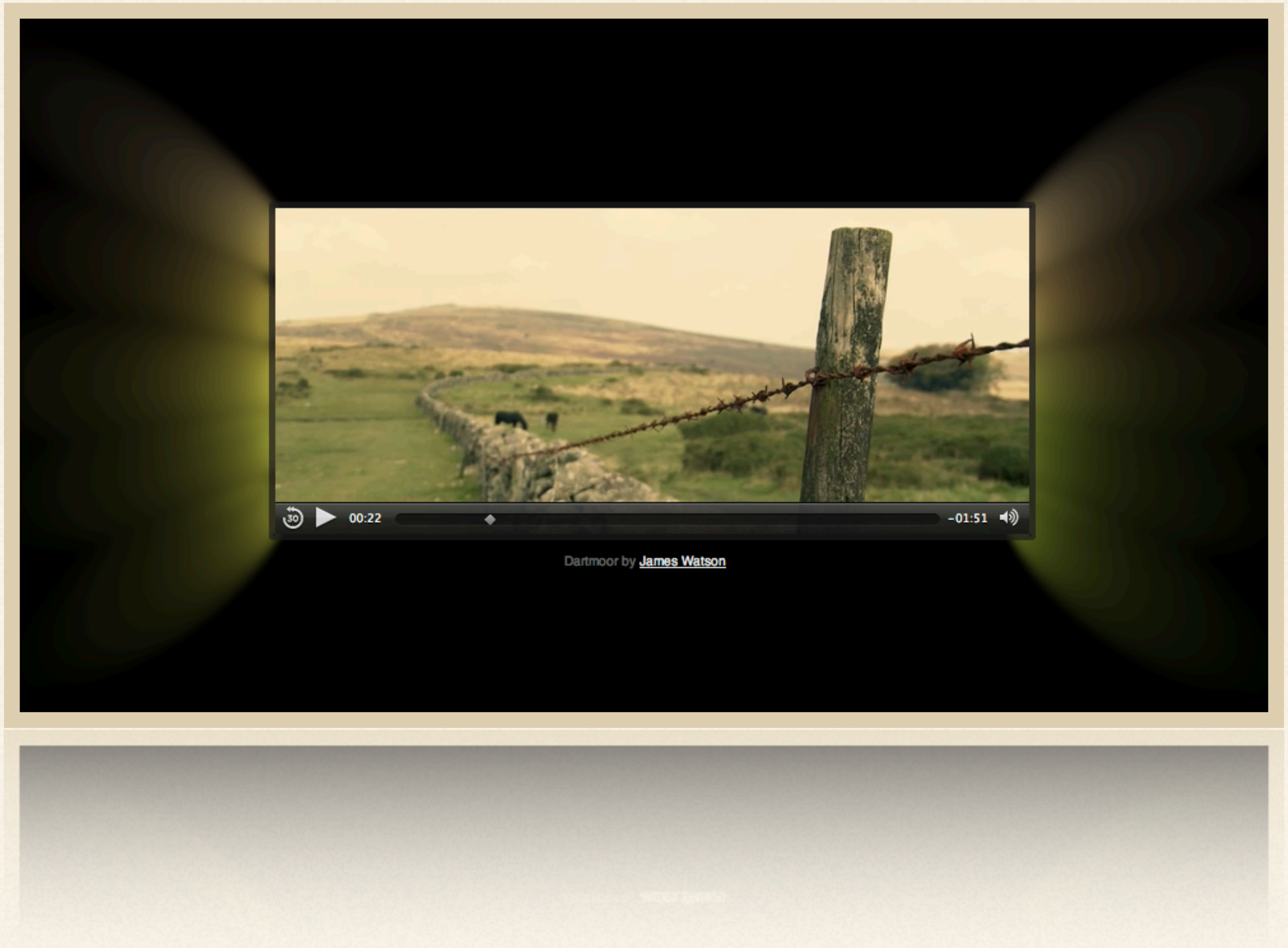


Individual pixel values can be retrieved

```
ctx.getImageData(sx, sy, sw, sh);
```



Returns an array of pixel values



Canvas ambilight

Making things move

Harnessing the power

* Remember all the shapes on the canvas
* Move them, transform them, and make them interact
* Redraw the all the shapes in their new positions
* Rinse and repeat, really quickly

My workflow

* Treat each shape as a JavaScript object
* Each shape object has position values
* Store the shape objects in an array
* Run a timeout function every 40 ms
* Clear the canvas
* Make any changes to the shape objects
* Loop through and redraw every shape

The future of canvas

The future of canvas

* OOP programming allows much to be achieved through canvas
* It's flexible and powerful
    * *Animation engines*
    * *Pseudo 3D graphics*
* Reading pixel values opens a lot of doors
* Integration with other HTML5 elements is a killer feature

Is it a Flash killer?

Canvas vs. Flash

* Canvas will flourish with future development of frameworks
* Its animation capabilities are only just being realised
* Flash has tight integration with the offline world, but HTML5 is changing that
* There is a gap in the market for a GUI for developing canvas applications

Thank you!