Intro To Canvas

What is Canvas?

- Canvas is a drawing surface
- We can use it to draw anything
- Things from graphs to games
- You'll be using it for your projects

The Canvas Element

- We can create a canvas using the <canvas> tag
- We usually include 4 properties
  - id: The name we will refer to the canvas by in JavaScript
  - Width: The width of the Canvas
  - Height: the height of the Canvas
  - style: We use it to give the Canvas a border to so we know where it is on the page.

```
<canvas id="drawingSurface"
    style="border-style: solid" width="600px"
    height="600px"></canvas>
```

The Rendering Context

- Before we can draw onto the canvas we must get the rendering context
- We get the context from the Canvas, which we have called drawingSurface using the id property

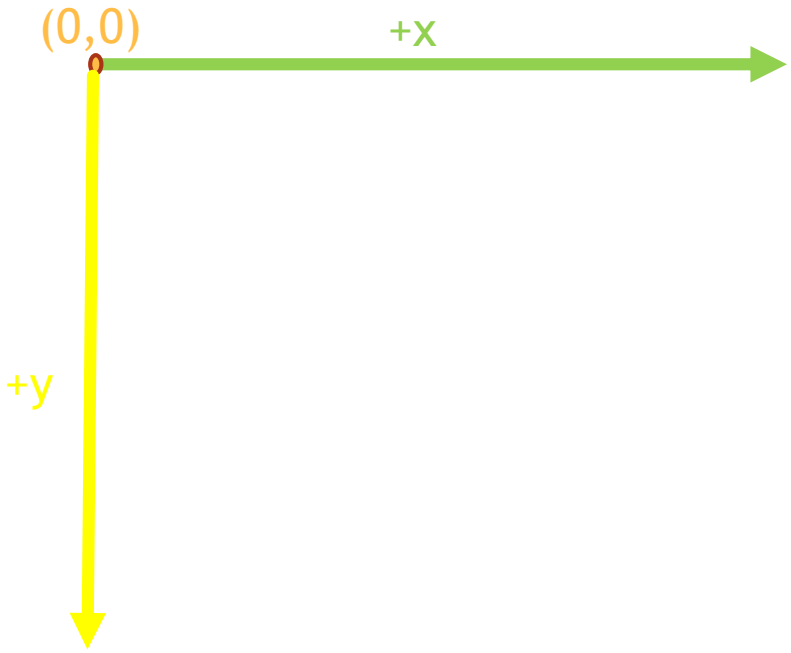
```
var drawingSurface =

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

So we are getting the context with the getContext method, and storing it in the variable ctx

Canvas Coordinate System

- There are two coordinate system, the local coordinate system and global coordinate system.
- All commands are relative to the local coordinate system
- The global coordinate system and local coordinate system are initially the same.
- The Canvas coordinate system:
  1. 0,0 starts at the upper left hand corner
  2. Positive x points to the right
  3. Positive y points down

Visual Representation



The Canvas coordinate system

ctx.fillRect

- `fillRect()` draws a rectangle.

```
fillRect(x,y,L,W);
```

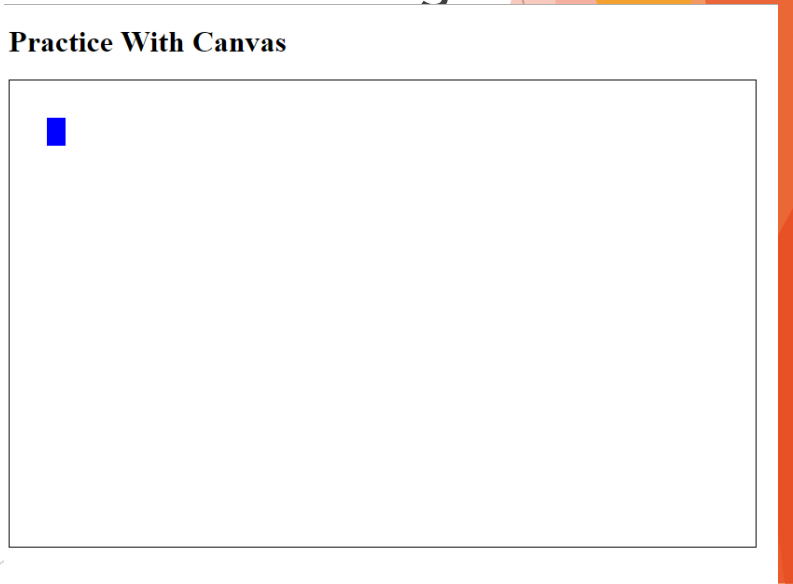
- x is the x-coordinate of the upper left corner of the rectangle
- y is the y-coordinate of the upper left corner of the rectangle
- L is the length in pixels
- W is the width in pixels

ctx.fillStyle

- fillStyle is not a method but a property
- We can us it to set the color of the fill
- `ctx.fillStyle = “red”;`

Example

- `ctx.fillStyle = “blue”`
- `ctx.fillRect(40,40,20,30);`
- The above two lines of code will draw a blue rectangle starting at (40,40) and has a width of 20 pixel and height of 30 pixel
- Note that the color is set first, then the rectangle is drawn



Drawing With Paths

- Canvas has the ability to do line drawings.
- The Drawing Context has an internal list of points/commands.
- It can do this because it's an object!

Commands that modify this list of points

- `beginPath();`
- `lineTo();`
- `moveTo();`

Commands that draw the list

- `stroke()`

To Draw a Path

- We start by calling

```
ctx.beginPath()
```

- This call empties the internal list of points and commands inside the drawing context.

- Next we can call lineTo() or moveTo()

ctx.lineTo()

- lineTo() adds a point to the internal list, and the command to draw a line from the last point to the current point. No line is actually drawn yet.

- If lineTo() is called immediately after beginPath(), then only the point is added to list, not drawing command is added because there is no point to connect to

- `ctx.lineTo(x,y);`
  - x is the x-coordinate in pixels in the local coordinate system
  - y is the y-coordinate in pixels in the local coordinate system

ctx.moveTo() & stroke()

- Move to adds the the point to the list, but does not add any drawing commands like the lineTo()

- `ctx.moveTo(x,y);`
  - x is the x-coordinate in pixels in the Local coordinate system
  - y is the y-coordinate in pixels in the local coordinate system

- `ctx.stroke()`
  - stoke actually draws the what is described in the internal list that we have been building up.

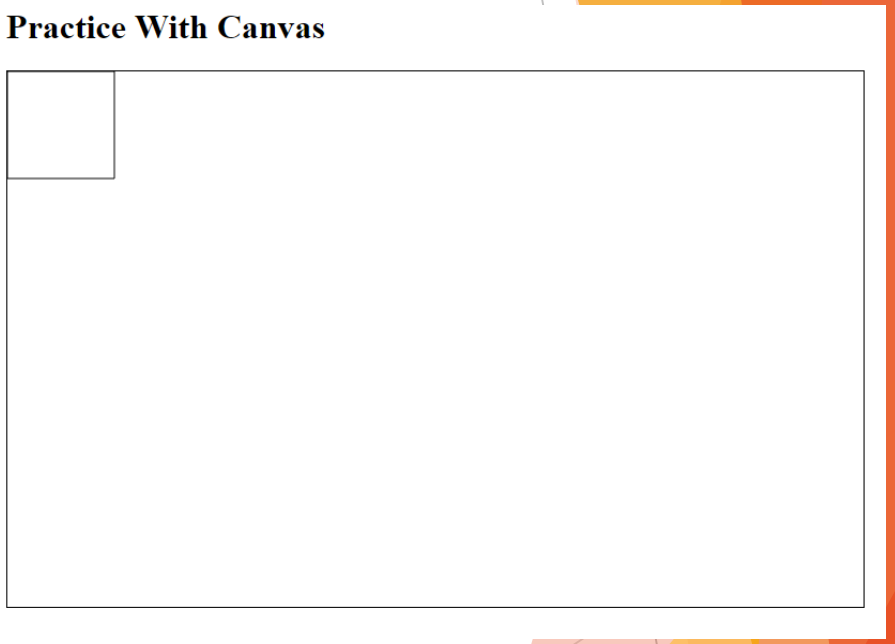
Example

1. ctx.beginPath();
2. ctx.lineTo(0,0);
3. ctx.lineTo(100,0);
4. ctx.lineTo(100,100);
5. ctx.lineTo(0,100);
6. ctx.lineTo(0,0);
7. ctx.stroke();

line	coordinate List
1	[]
2	[(0,0)]
3	[(0,0),(100,0)]
4	[(0,0),(100,0),(100,100)]
5	[(0,0),(100,0),(100,100),(0,100)]
6	[(0,0),(100,0),(100,100),(0,100),(0,0)]
7	[(0,0),(100,0),(100,100),(0,100),(0,0)]

Example cont.

```
ctx.beginPath();
ctx.lineTo(0,0);
ctx.lineTo(100,0);
ctx.lineTo(100,100);
ctx.lineTo(0,100);
ctx.lineTo(0,0);
ctx.stroke();
```

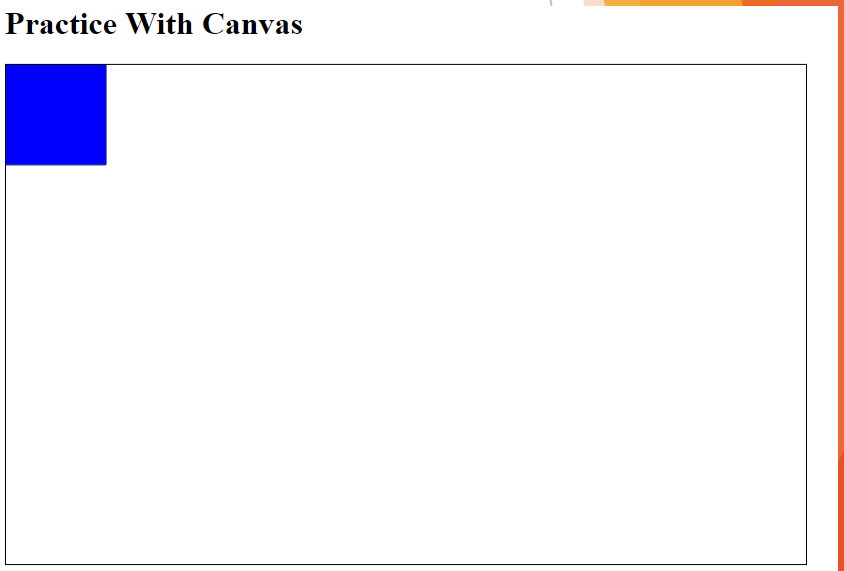


- And this square starting at (0,0) is drawn

Example cont.

- Now if we modify this example slightly by adding two extra lines

```
ctx.beginPath();
ctx.lineTo(0,0);
ctx.lineTo(100,0);
ctx.lineTo(100,100);
ctx.lineTo(0,100);
ctx.lineTo(0,0);
ctx.stroke();
ctx.fillStyle = "blue";
ctx.fill();
```



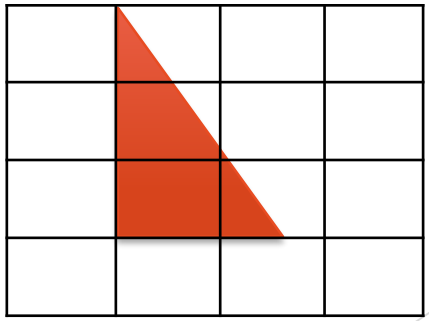
ctx.strokeStyle

- In the previous example you may have noticed the final box was filled with blue, but still had a black outline
- We can change the color of the stroke
- stokeStyle is a property of the rendering context

- `ctx.strokeStyle = “blue”;`

Drawing shapes

- If we're going to draw anything, we need to know the coordinates of the points
- Easiest way to do this is to superimpose a grid on top of your drawing
- Then you know exactly what points you need to construct your drawing



Changing the Local Coordinate System

- We can change the local coordinate system
- This is convenient, since we can draw everything around the origin (0,0) and then just move them to the right place

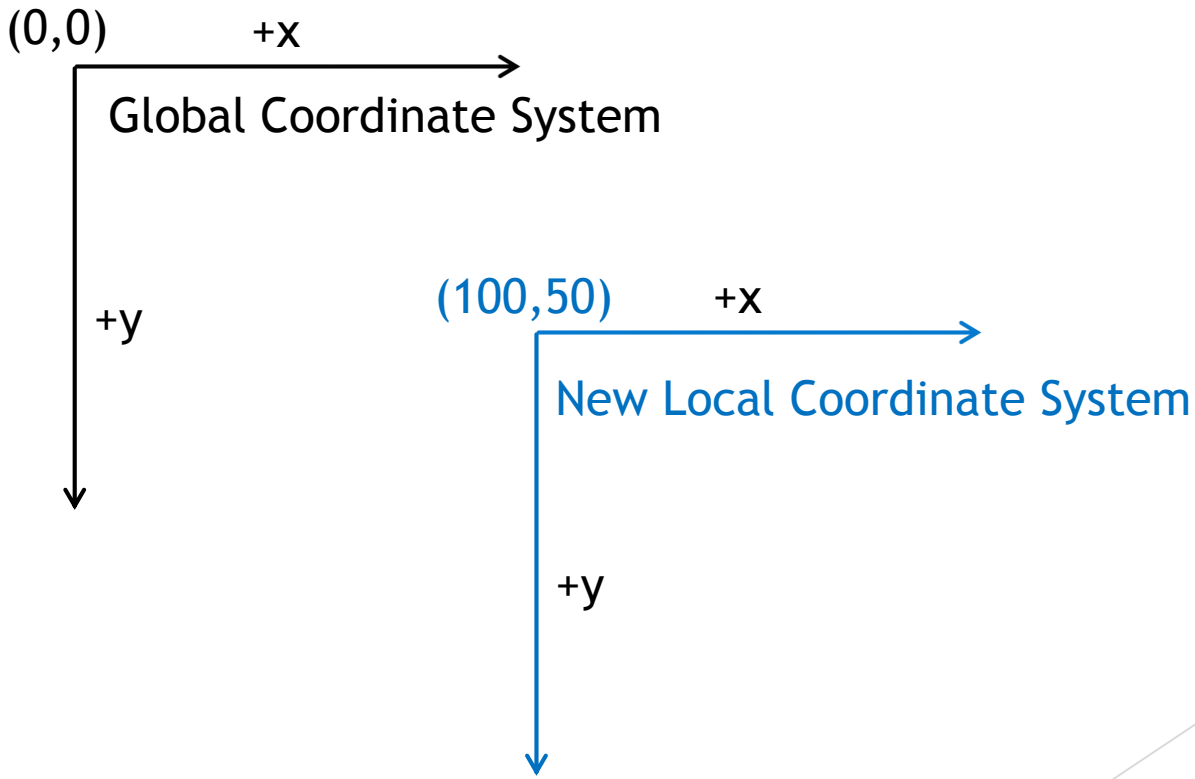
- `ctx.translate(deltaX, deltaY);`
  - deltaX : The amount to move in the x-direction relative to the local coordinate system.
  - deltaY: The amount to move in the y-direction relative to the local coordinate system.

Example

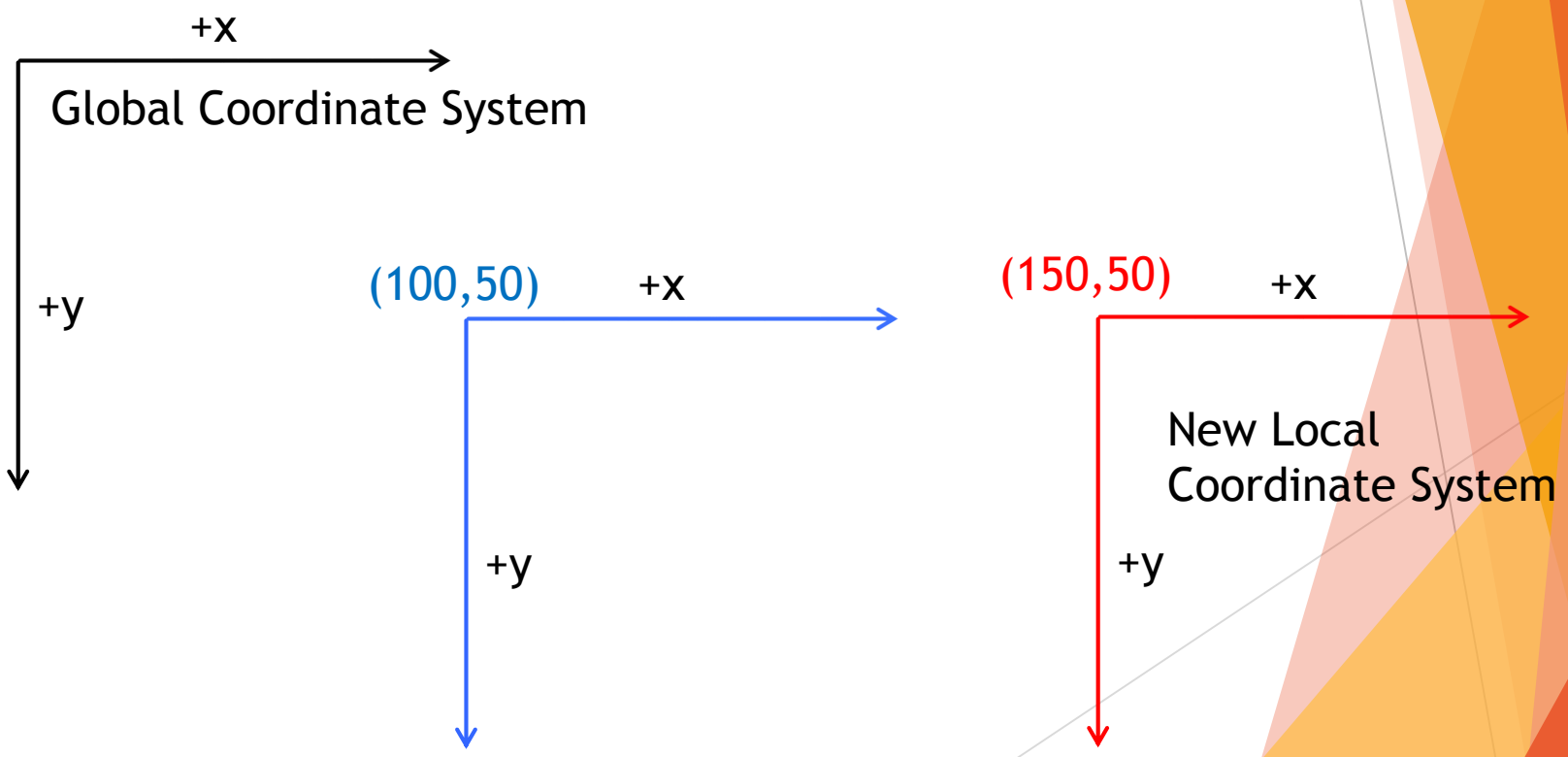
- `ctx.translate(100,50);`
- will move the local coordinate system, and all subsequent commands will use the new local coordinate system.
- The above command will move
  1. Move 100 in the current local x-direction
  2. Move 50 in the current local y-direction
- After the command we have a new local coordinate system.

Example Part 1

- `ctx.translate(100,50);`



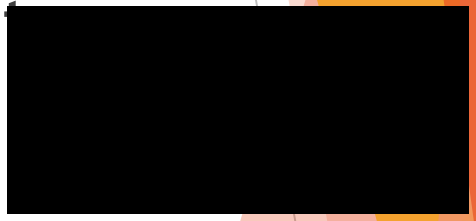
Example Part 2



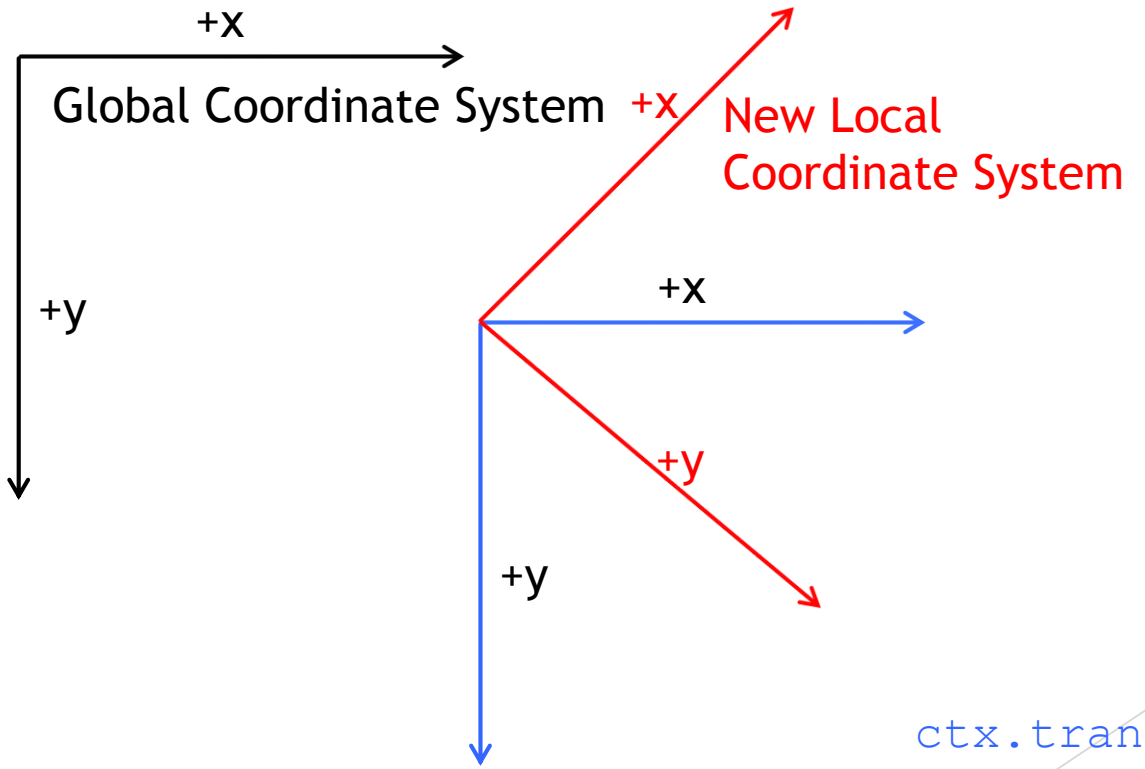
ctx.rotate()

- ctx.rotate() rotates the coordinate system.
- The angle is specified in radians
- Sorry
- Going to have to convert degrees t
  radians
- Don’t panic! This is easy in JavaScript

```
 radians = degrees *
(Math.PI/180);
```



Example Part 3



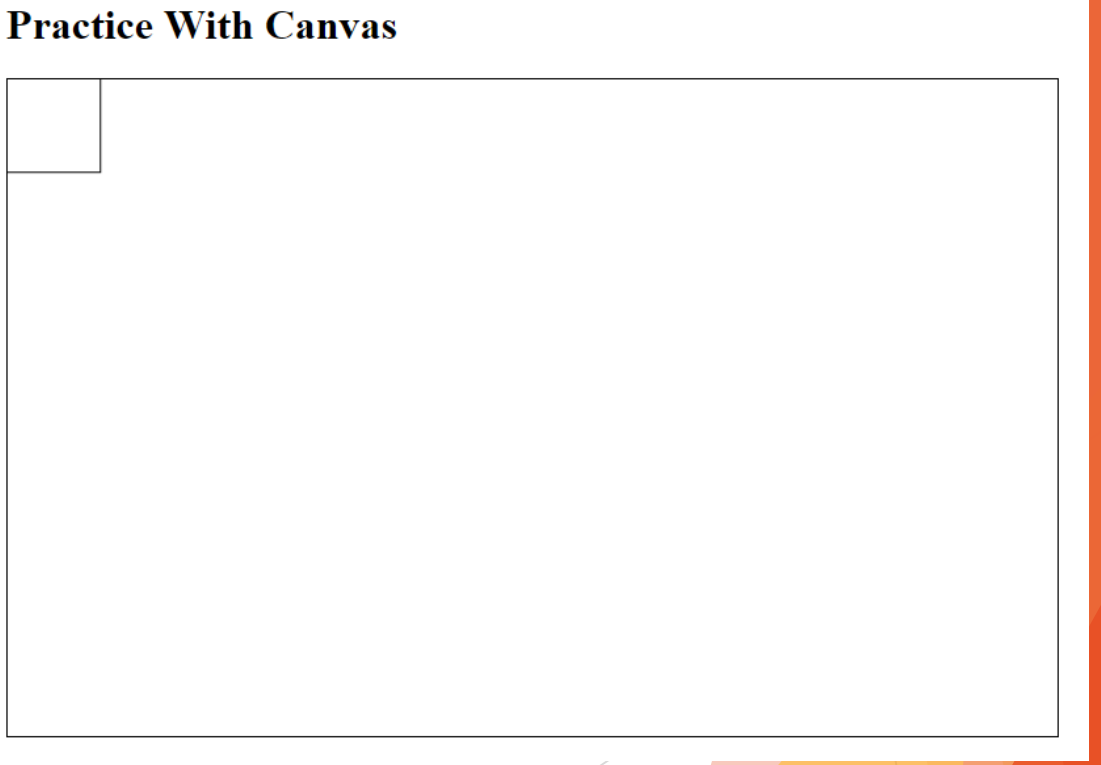

```
ctx.translate(100,50);
ctx.rotate(-45*Math.PI/180);
```

Note:

- All coordinates in the ctx list is stored in the global coordinate system
- So, calling translate or rotate does not effect existing points already on the list, just points added after the translate or rotate is called

Example trying to use rotate() to make life easier

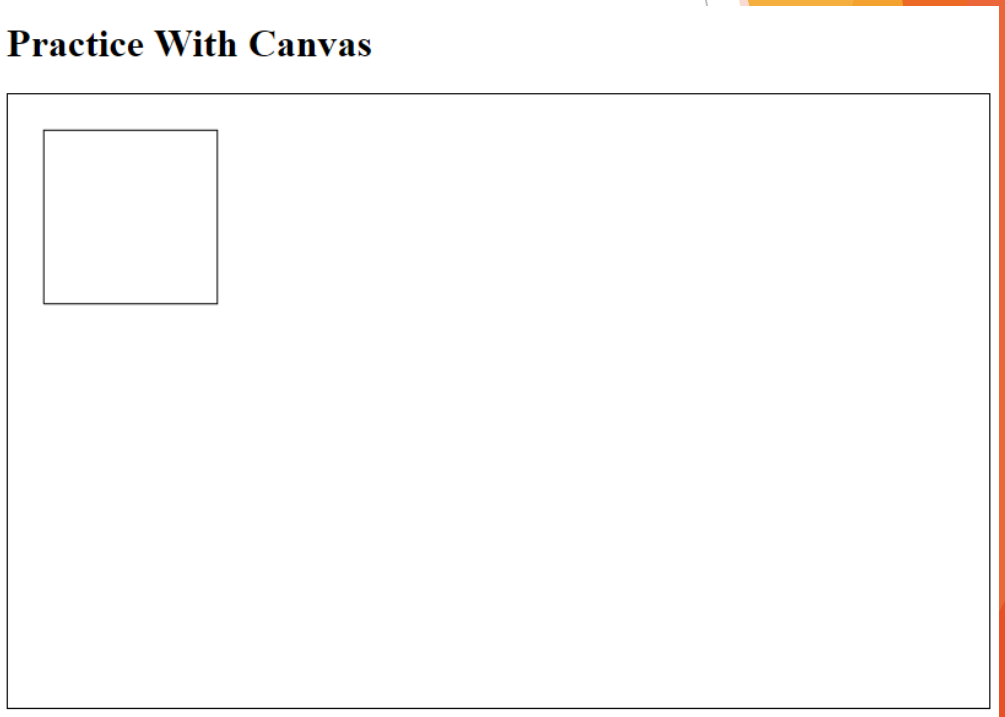
```
ctx.beginPath();
ctx.rotate(45*Math.PI/180);
ctx.lineTo(0,100);
ctx.rotate(90*Math.PI/180);
ctx.lineTo(0,100);
ctx.rotate(90*Math.PI/180);
ctx.lineTo(0,100);
ctx.rotate(90*Math.PI/180);
ctx.lineTo(0,100);
ctx.rotate(90*Math.PI/180);
ctx.lineTo(0,100);
ctx.stroke();
```



Draws a square, but notice that all the lineTo commands look the same

Example – easier to see if we add a translate()

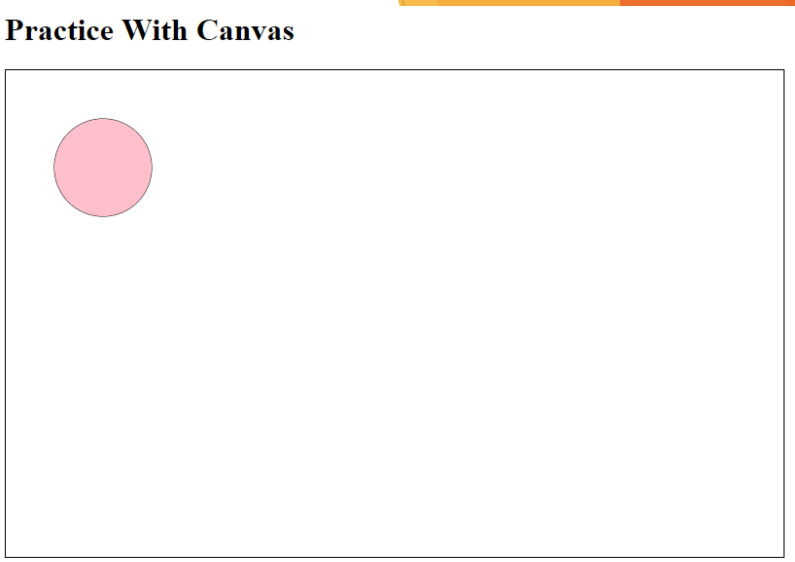
```
ctx.translate(100,100);
ctx.beginPath();
ctx.rotate(45*Math.PI/180);
ctx.lineTo(0,100);
ctx.rotate(90*Math.PI/180);
ctx.lineTo(0,100);
ctx.rotate(90*Math.PI/180);
ctx.lineTo(0,100);
ctx.rotate(90*Math.PI/180);
ctx.lineTo(0,100);
ctx.rotate(90*Math.PI/180);
ctx.lineTo(0,100);
ctx.stroke();
```



What If I Want to Draw Circles?

- Great Question
- `ctx.arc(x, y, r, start, stop);`
  - X and Y give the location of the CENTER of the circle
    - x is the x-coordinate in pixels
    - y is the y-coordinate in pixels
  - r is the radius of the circle you want to draw
  - start is the angle you want to start drawing your circle at (radians)
  - stop is the angle you want to stop drawing your circle at (radians)

My First Circle




```
ctx.fillStyle = "pink";
ctx.beginPath();
ctx.arc(100,100,50,0,2*Math.PI);
ctx.stroke();
ctx.fill();
```

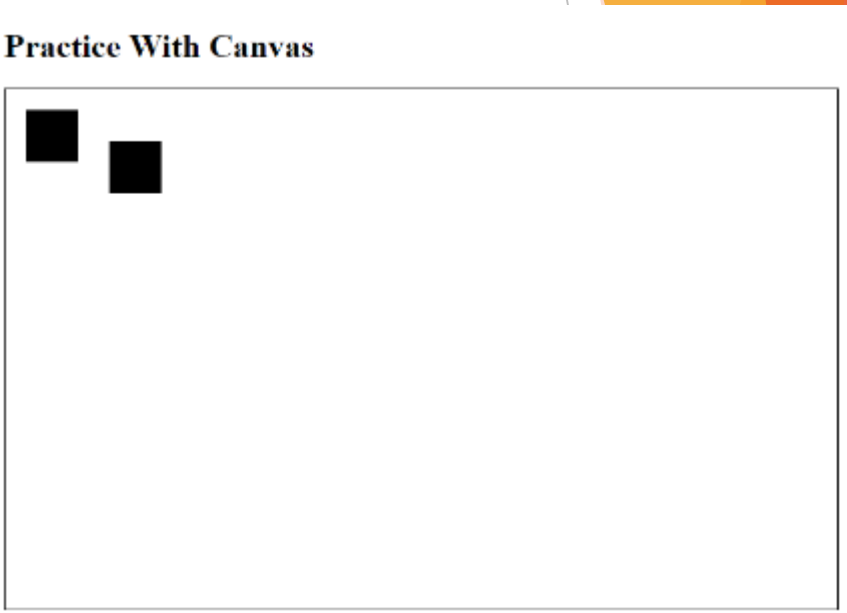
- Note about 2*Math.pi
- If you want to draw a full circle, you want to start at 0 degrees and go a full 360 degrees around
- Converting 360 degrees to radians is 360*(Math.PI/180)
- This is the same as 2*Math.PI

ctx.save() & ctx.restore()

- ctx.save saves the current local coordinate system to a list
- ctx.restore restores last local coordinate system saved to the list, and removes it from the list
- This allows us to isolate our effects of translate and rotate to a small section of code

Example of save/restore

```
ctx.save();
ctx.translate(100,50);
ctx.fillRect(0,0,50,50);
ctx.restore();
ctx.save();
ctx.translate(20,20);
ctx.fillRect(0,0,50,50);
ctx.restore();
```



The above will draw two rectangles at different locations

Summary

- We can create images on HTML pages using canvas
- We will unlock its power more when we start using loops and arrays
- Things we can draw
  - Rectangles
  - Polylines
  - There are more you can draw
  - Ref: http://www.w3schools.com/tags/ref_canvas.asp