Image Processing

HTML5 – Canvas

JavaScript

Simple Drawing

Brent M. Dingle, Ph.D.
Game Design and Development Program
Mathematics, Statistics and Computer Science
University of Wisconsin - Stout

2015

Lecture Objectives

- Identify a text editor for coding
- Identify a web browser for testing
- Provide Examples
  - Basic HTML5 canvas
  - Simple Drawing and Manipulation in JavaScript

What this is Not

- To complete your projects
  - You must learn more about HTML5 and JavaScript than what is about to be shown
    - This is an “on-your-own” activity
      - Instructor can help, but you must try on your own
    - A prereq to this course is CS 244
      - So you have programmed before
      - This stuff is “easy” compared to that =)
      - Likewise on the math topics
- In Sum: The following is just a place to start
  - More examples will follow throughout the course

Suggested Text Editor

- Notepad works in Windows
- TextEdit works on Macs
- vi and emacs work on LINUX

- Notepad++ has worked well for students in the past
  - https://notepad-plus-plus.org/

    - *Let the instructor know outside of class if obtaining an editor will be problematic for you*
      - » *You may use what you are comfortable with, but it must result in clean and easy to read (by humans) HTML and javaScript files*
        - *when the files are opened using editors such as the above*

Web Browsers

- Instructor tends to use Firefox
  - used for grading

- Others that you should test your stuff with
  - Safari, Chrome, IE…
    - variations of mobile devices

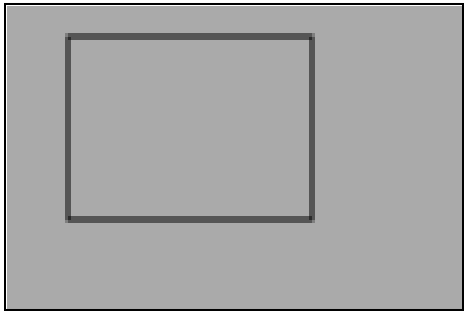
» *Let the instructor know outside of class if testing using Firefox will be problematic for you*

Drawing a Rectangle: HTML

HTML File: simpleRectangle.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <script src="simpleR                ext/javascript"></script>
</head>

<body>
   <div>
      <canvas id="myCanvas" width="300" height="200">
      Your browser does NOT support canvas!
      </canvas>
   </div>
</body>
</html>
```

Declares the Type of document being defined

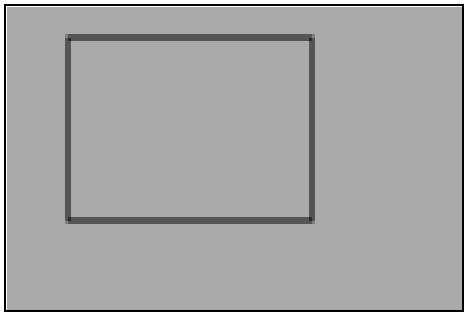


Drawing a Rectangle: HTML

HTML File: simpleRectangle.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <script src="simpleRectan                    t>
</head>

<body>
   <div>
      <canvas id="myCanvas" width="300" height="200">
      Your browser does NOT support canvas!
      </canvas>
   </div>
</body>
</html>
```

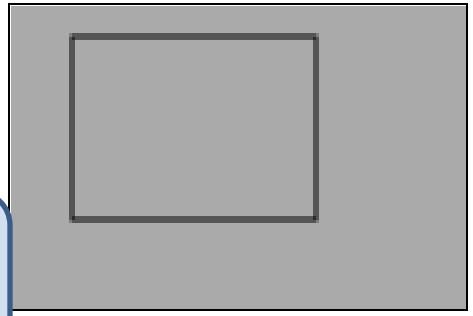
Declares the
Start of the document
and the language it is written in/for



Drawing a Rectangle: HTML

HTML File: simpleRectangle.html

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

<body>
   <div>
      <canvas id="myCanvas
      Your browser does NO
      </canvas>
   </div>
</body>
</html>
```



Header portion of your document
I suggest using this to “include” your javascript file(s)

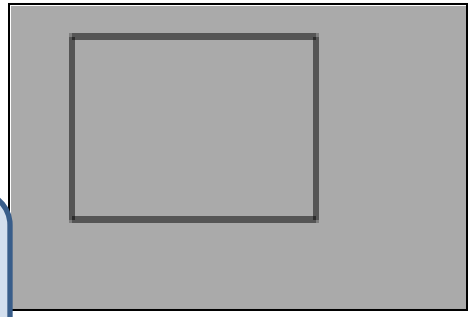
…Rather than writing code in the html file itself…

Drawing a Rectangle: HTML

HTML File: simpleRectangle.html

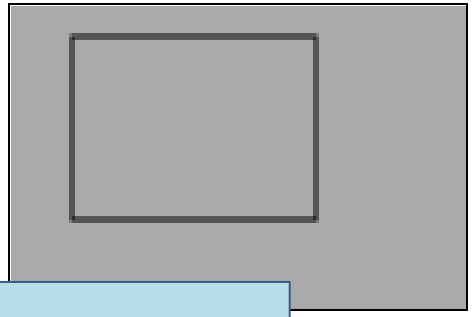
```
<!DOCTYPE html>
<html lang="en">
<head>
  <script src="simpleRectangle.js" type="text/javascript"></script>
</head>

<body>
   <div>
      <can                                    eight="200">
      Your                                    s!
      </canvas>
   </div>
</body>
</html>
```

Body portion of your document with a “division” declared



Drawing a Rectangle: HTML

HTML File: simpleRectangle.html



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

<body>
  <div>
    <canvas id="myCanvas" width="300" height="200">
    Your browser does NOT support canvas!
    </canvas>
  </div>
</body>
</html>
```

Creates a canvas on your webpage
The id="myCanvas" is important as your javaScript code will use that id
Width and Height are also important to note,
They define the size of the canvas display (in pixels) on the webpage

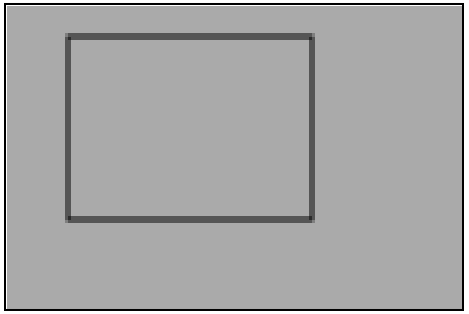
Drawing a Rectangle: HTML

HTML File: simpleRectangle.html

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

<body>
  <div>
    <can
    Your
    </ca
  </div>
</body>
</html>
```

The end of your document

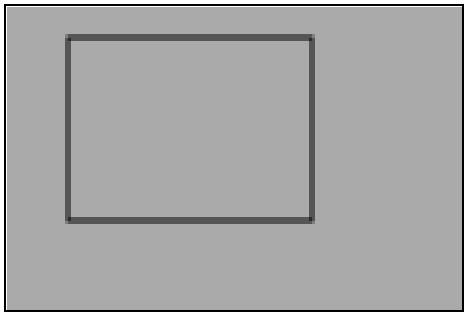
This HTML file is typical of the ones you will use for this class
Most of the changes will be in
- the size of the canvas,
- any words/text you want on the page
- and the rest will (typically) be in the javaScript code file(s)…



Drawing a Rectangle: HTML

HTML File: simpleRectangle.html

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

<body>
   <div>
      <canvas id="myCanvas" width="300" height="200">
      Your browser does NOT support canvas!
      </canvas>
   </div>
</body>
</html>
```



Questions on the HTML ?

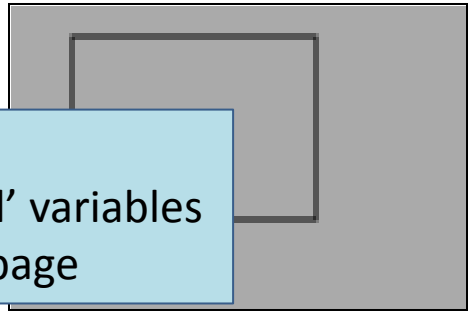
Drawing a Rectangle: JS

JavaScript File: simpleRectangle.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      ctx.fillStyle = "#aaaaaa";
      ctx.fillRect(0, 0, 300, 200);        // draw a grey background
      ctx.strokeRect(20, 10, 80, 60);  // draw a black rectangle
   },

};  // end theProgram variable

window.onload = function()
{
   theProgram.Main();
};
```

theProgram is a Singleton variable object

Useful for isolating your functions and 'global' variables from other "stuff" that might be on the webpage

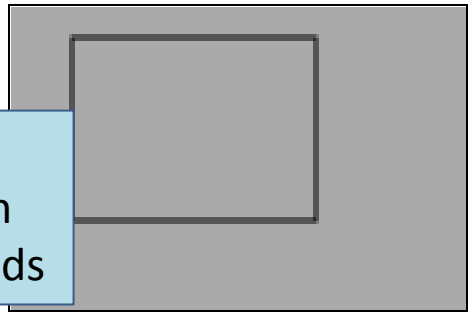


Drawing a Rectangle: JS

JavaScript File: simpleRectangle.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      ctx.fillStyle = "#aaaaaa";
      ctx.fillRect(0, 0, 300, 200);        // draw a grey background
      ctx.strokeRect(20, 10, 80, 60);  // draw a black rectangle
   },

};  // end theProgram variable

window.onload = function()
{
   theProgram.Main();
};
```

Main is a 'member' function of *theProgram*
This helps identify the entry point function which is first called after the webpage loads

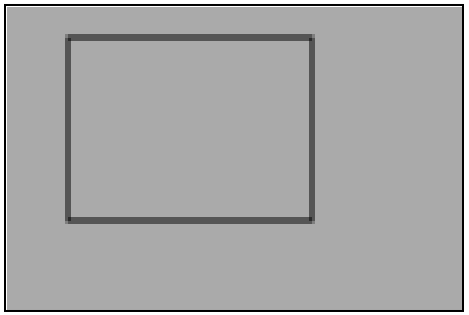


Drawing a Rectangle: JS

JavaScript File: simpleRectangle.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      ctx.fillStyle = "#aaaaaa";
      ctx.fillRect(0, 0, 300, 200);        // draw a grey background
      ctx.strokeRect(20, 10, 80, 60);  // draw a black rectangle
   },

};  // end theProgram variable
```

```
window.onload = function()
{
   theProgram.Main();
};
```

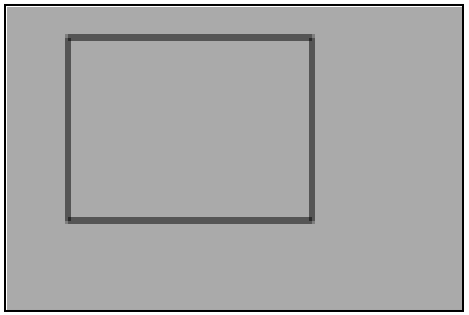


window.onload is automatically called by the browser
This event occurs "after" the html page has been loaded including all the page's content: images, css, scripts…

Drawing a Rectangle: JS

JavaScript File: simpleRectangle.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      ctx.fillStyle = "#aaaaaa";
      ctx.fillRect(0, 0, 300, 200);        // draw a grey background
      ctx.strokeRect(20, 10, 80, 60);  // draw a black rectangle
   },

};  // end theProgram varia

window.onload = func
{
   theProgram.Main();
};
```

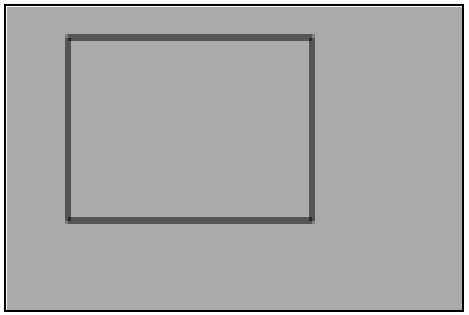


So…
This function's code
is what will be executed when the page finishes loading

Drawing a Rectangle: JS

JavaScript File: simpleRectangle.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      ctx.fillStyle = "#aaa
      ctx.fillRect(0, 0, 300
      ctx.strokeRect(20, 1
   },

}; // end theProgram variable

window.onload = function()
{
   theProgram.Main();
};
```



Gets a “handle” to the canvas element on the webpage

It is important that the HTML file has a canvas element with
id = “myCanvas”

If not, then the variable *theCanvas* will be assigned a non-value and the rest of the code will not run

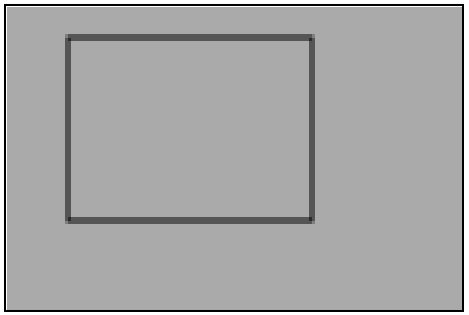
Drawing a Rectangle: JS

JavaScript File: simpleRectangle.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      ctx.fillStyle = "#aaaaaa";
      ctx.fillRec
      ctx.stroke
   },

}; // end theProgram variable

window.onload = function()
{
   theProgram.Main();
};
```

Gets a “handle” to the drawing context of *theCanvas*

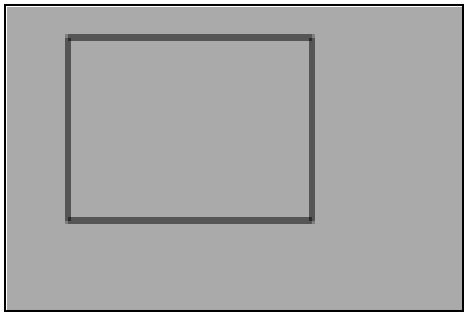
This context defines where “stuff” will actually be drawn



Drawing a Rectangle: JS

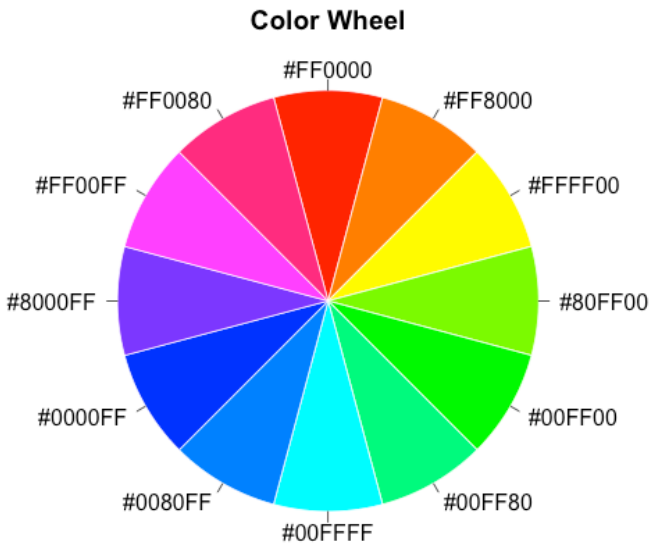
JavaScript File: simpleRectangle.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      ctx.fillStyle = "#aaaaaa";
      ctx.fillRect(0, 0, 300, 200);        // draw a grey background
      ctx.strokeRect(20,
   },

};  // end theProgram variable

window.onload = function()
{
   theProgram.Main();
};
```



Sets the drawing fillstyle to be grey

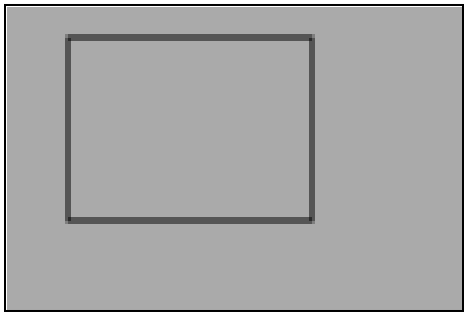
The numbers are hex: RRGGBB



Drawing a Rectangle: JS

JavaScript File: simpleRectangle.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      ctx.fillStyle = "#aaaaaa";
      ctx.fillRect(0, 0, 300, 200);        // draw a grey background
      ctx.strokeRect(20, 10, 80, 60);  // draw a black rectangle
   },

}; // end theProgram var

window.onload = fu
{
   theProgram.Main();
};
```



Draws a filled rectangle with
diagonally opposed corners at (0, 0) and (300, 200)

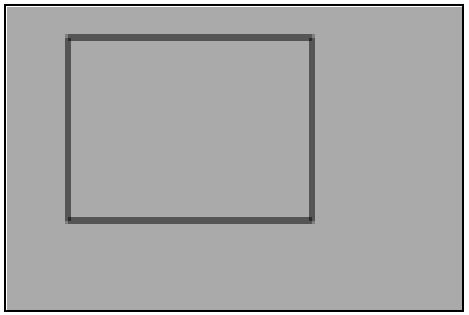
In this case this matches the size of the canvas (hence background)

The fill color is whatever the current fillstyle is set to

Drawing a Rectangle: JS

JavaScript File: simpleRectangle.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      ctx.fillStyle = "#aaaaaa";
      ctx.fillRect(0, 0, 300, 200);        // draw a grey background
      ctx.strokeRect(20, 10, 80, 60);  // draw a black rectangle
   };

}; // end theProgram vari

window.onload = fur
{
   theProgram.Main();
};
```



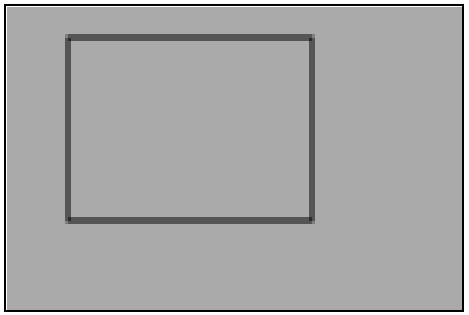
Draws the edges of a rectangle with
diagonally opposed corners at (20, 10) and (80, 60)

The line color is whatever the current strokestyle is set to
(it defaults to 000000 = black)

Drawing a Rectangle: JS

JavaScript File: simpleRectangle.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      ctx.fillStyle = "#aaaaaa";
      ctx.fillRect(0, 0, 300, 200);        // draw a grey background
      ctx.strokeRect(20, 10, 80, 60);  // draw a black rectangle
   },

};  // end theProgram variable

window.onload = function()
{
   theProgram.Main();
};
```



Questions on the JavaScript?

Challenges for Home

- Code is available online (zipped)

- Download and attempt the following
  - Change the background rectangle to be size: 0, 0, 300, 100

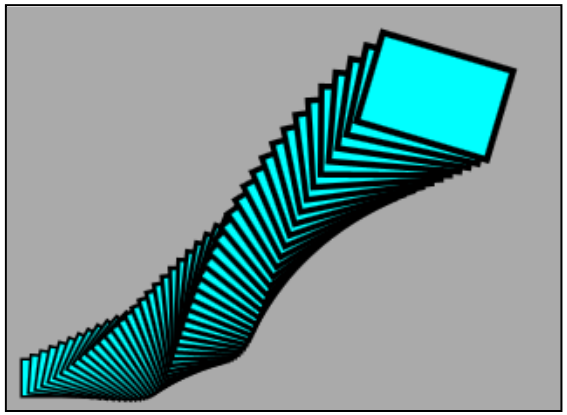
  - Use ctx.scale(s1, s2)
    to scale the drawn rectangle to be bigger

  - Alter the drawn rectangle to be filled RED

Rotating a Rectangle: HTML

loopRotate.html

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

<body>
   <div>
      <canvas id="myCanvas" width="320" height="240">
      Your browser does NOT support canvas!
      </canvas>
   </div>
</body>
</html>
```

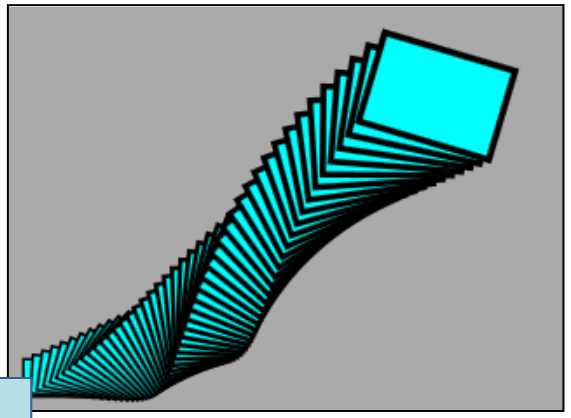


Rotating a Rectangle: HTML

loopRotate.html

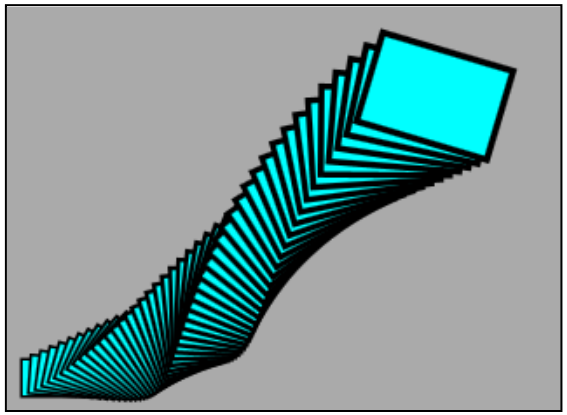
```
<!DOCTYPE html>
<html lang="en">
<head>
  <script src="loopRotate.js" type="text/javascript"></script>
</head>

<body>
   <div>
      <canvas id="myCanvas" width="320" height="240">
      Your browser does NOT support canvas!
      </canvas>
   </div>
</body>
</html>
```

JavaScript filename changed from previous example



Rotating a Rectangle: HTML

loopRotate.html

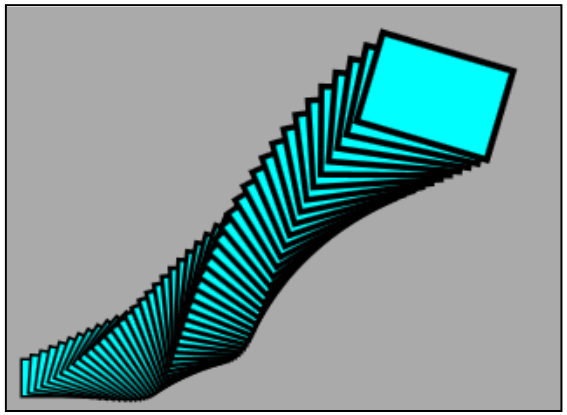


```
<!DOCTYPE html>
<html lang="en">
<head>
  <script src="loopRotate.js" type="text/javascript"></script>
</head>

<body>
  <div>
    <canvas id="myCanvas" width="320" height="240">
    Your browser does NOT support canvas!
    </canvas>
  </div>
</body>
</html>
```

Canvas size changed from previous example

Rotating a Rectangle: HTML

loopRotate.html



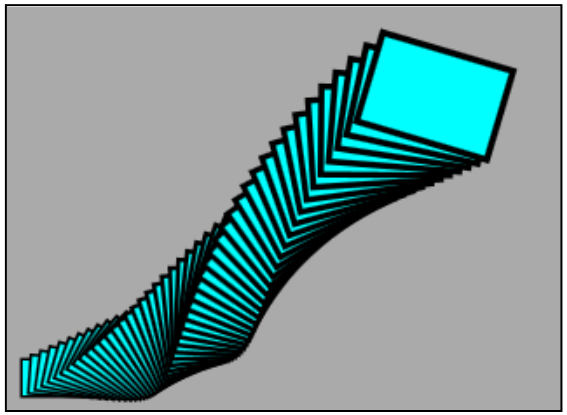
```
<!DOCTYPE html>
<html lang="en">
<head>
  <script src="loopRotate.js" type="text/javascript"></script>
</head>

<body>
   <div>
      <canvas id="myCanvas" width="320" height="240">
      Your browser does NOT support canvas!
      </canvas>
   </div>
</body>
</html>
```

Questions on the HTML ?

Rotating a Rectangle: JS

loopRotate.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      // grey background
      ctx.fillStyle="#aaaaaa";
      ctx.fillRect(0, 0, 320, 240);
      // for subsequent boxes:
      ctx.fillStyle="#00ffff";
      ctx.lineWidth = 3;
```



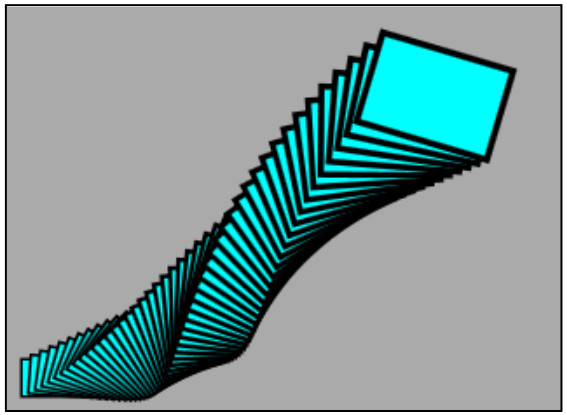
Much the same as in the previous example

```
window.onload = function()
{
   theProgram.Main();
};
```

Rotating a Rectangle: JS

loopRotate.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      // grey background
      ctx.fillStyle="#aaaaaa";
      ctx.fillRect(0, 0, 320, 240);
      // for subsequent boxes:
      ctx.fillStyle="#00ffff";
      ctx.lineWidth = 3;
```



We will be drawing a series of rectangles

Each will be colored light-blue / cyan with a thick border line

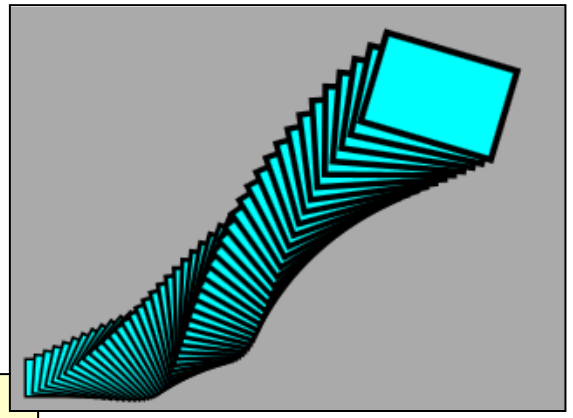
```
window.onload = function()
{
   theProgram.Main();
};
```

setTransform

loopRotate.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      // grey background
      ctx.fillStyle="#aaaaaa";
      ctx.fillRect(0, 0, 320, 240);
      // for subsequent boxes:
      ctx.fillStyle="#00ffff";
      ctx.lineWidth = 3;
```

```
for (var i = 0; i < 50; i++) {
         var t = i / 50.0;   // time parameter [0..1)

         ctx.setTransform(1,0,0,1,0,0); // reset to identity

         ctx.translate(10 + 270 * t, 210 - 120 * t);
         ctx.scale(1 + t * 1.5, 1 + t * 1.5);
         ctx.rotate(3.5 * t);

         ctx.beginPath();
         ctx.rect(0, 0, 30, 20);
         ctx.stroke();
         ctx.fill();
         ctx.closePath();
      }
   },
};  // end theProgram variable
```



```
window.onload = function()
{
   theProgram.Main();
};
```

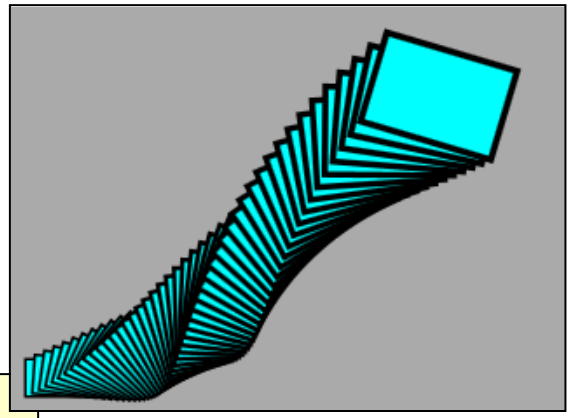
setTransform

loopRotate.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      var ctx = theCanvas.getContext("2d");

      // for subsequent boxes:
      ctx.fillStyle="#00ffff";
      ctx.lineWidth = 3;

      for (var i = 0; i < 50; i++) {
         var t = i / 50.0;   // time parameter [0..1)

         ctx.setTransform(1,0,0,1,0,0); // reset to identity

         ctx.translate(10 + 270 * t, 210 - 120 * t);
         ctx.scale(1 + t * 1.5, 1 + t * 1.5);
         ctx.rotate(3.5 * t);

         ctx.beginPath();
         ctx.rect(0, 0, 30, 20);
         ctx.stroke();
         ctx.fill();
         ctx.closePath();
      }
   },
}; // end theProgram variable
```

Iterative for-loop

i counts from 0 to 49



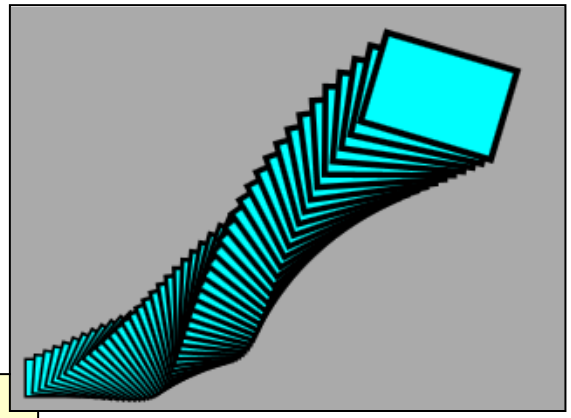
```
window.onload = function()
{
   theProgram.Main();
};
```

setTransform

loopRotate.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      c
      /
      ctx.fillStyle="#dddddd";
      ctx.fillRect(0, 0, 320, 240);
      // for subsequent boxes:
      ctx.fillStyle="#00ffff";
      ctx.lineWidth = 3;
      for (var i = 0; i < 50; i++) {
            var t = i / 50.0;   // time parameter [0..1)

            ctx.setTransform(1,0,0,1,0,0); // reset to identity

            ctx.translate(10 + 270 * t, 210 - 120 * t);
            ctx.scale(1 + t * 1.5, 1 + t * 1.5);
            ctx.rotate(3.5 * t);

            ctx.beginPath();
            ctx.rect(0, 0, 30, 20);
            ctx.stroke();
            ctx.fill();
            ctx.closePath();
         }
   },
};  // end theProgram variable
```

A ‘time’ parameter *t*
controls translation and rotation amounts

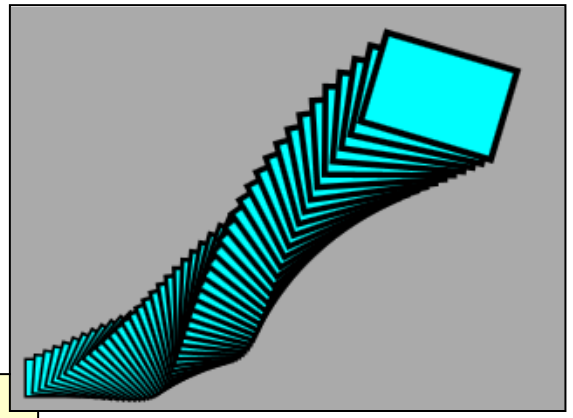


```
window.onload = function()
{
   theProgram.Main();
};
```

setTransform

loopRotate.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      // grey background
      ctx.fillStyle="#aaaaaa";
      ctx.fillRect(0, 0, 320, 240);
      // for subsequent boxes:
      ctx.fillStyle="#00ffff";
      ctx.lineWidth = 3;
```



```
for (var i = 0; i < 50; i++) {
        var t = i / 50.0;   // time parameter [0..1)

        ctx.setTransform(1,0,0,1,0,0); // reset to identity

        ctx.translate(10 + 270 * t, 210 - 120 * t);
        ctx.scale(1 + t * 1.5, 1 + t * 1.5);
        ctx.rotate(3.5 * t);

        ctx.beginPath();
        ctx.rect(0, 0, 30, 20);
        ctx.stroke();
        ctx.fill();
        ctx.closePath();
      }
   },
};  // end theProgram variable
```

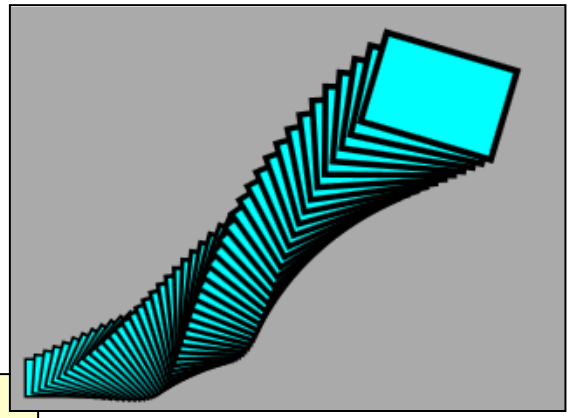
This is interesting

```
window.onload = function()
{
   theProgram.Main();
};
```

setTransform

loopRotate.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      // grey background
      ctx.fillStyle="#aaaaaa";
      ctx.fillRect(0, 0, 320, 240);
      // for subsequent boxes:
      ctx.fillStyle="#00ffff";
      ctx.lineWidth = 3;
```



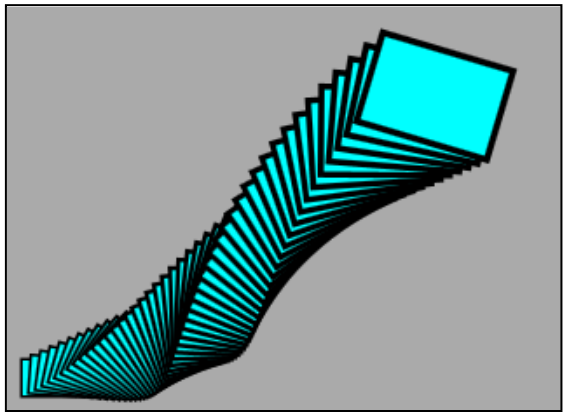
```
for (var i = 0; i < 50; i++) {
        var t = i / 50.0;   // time parameter [0..1)

        ctx.setTransform(1,0,0,1,0,0); // reset to identity

        ctx.translate(10 + 270 * t, 210 - 120 * t);
        ctx.scale(1 + t * 1.5, 1 + t * 1.5);
        ctx.rotate(3.5 * t);

        ctx.beginPath();
        ctx.rect(0, 0, 30, 20);
        ctx.stroke();
        ctx.fill();
        ctx.closePath();
      }
   },
};  // end theProgram variable
```

This is interesting

```
window.onload = function()
{
   theProgram.Main();
};
```

setTransform

loopRotate.js

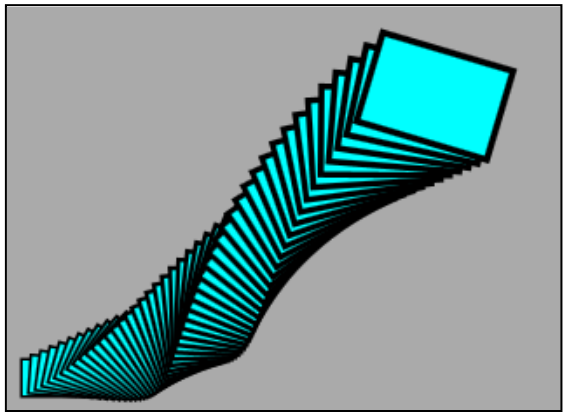


```
ctx.setTransform(1,0,0,1,0,0); // reset to identity
```

setTransform

loopRotate.js

```
ctx.setTransform(1,0,0,1,0,0); // reset to identity
```



Resets the transform matrix to the identity matrix:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Its parameter order is a little strange:

setTransform(A, b, c, D, e, f)

$$\begin{bmatrix} A & c & e \\ b & D & f \\ 0 & 0 & 1 \end{bmatrix}$$

Conceptually:

A	Scales the drawings horizontally
b	Skews the drawings horizontally
c	Skews the drawings vertically
D	Scales the drawings vertically
e	Moves the drawings horizontally
f	Moves the drawings vertically

Transforms in General

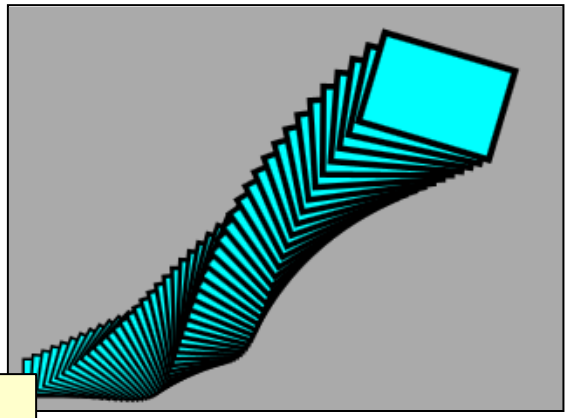
- setTransform()
  - good for clearing transform matrix back to identity

- rotate(), translate(), and scale()
  - tend to be more intuitive to use
  - HOWEVER
    - they are cumulative
    - so ORDER matters
    - and what you ALREADY DID matters
      - thus resetting transform matrix back to identity becomes important

translate, scale, rotate

loopRotate.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      // grey background
      ctx.fillStyle="#aaaaaa";
      ctx.fillRect(0, 0, 320, 240);
      // for subsequent boxes:
      ctx.fillStyle="#00ffff";
      ctx.lineWidth = 3;
```



```
for (var i = 0; i < 50; i++) {
         var t = i / 50.0;   // time parameter [0..1)

         ctx.setTransform(1,0,0,1,0,0); // reset to identity

         ctx.translate(10 + 270 * t, 210 - 120 * t);
         ctx.scale(1 + t * 1.5, 1 + t * 1.5);
         ctx.rotate(3.5 * t);

         ctx.beginPath();
         ctx.rect(0, 0, 30, 20);
         ctx.stroke();
         ctx.fill();
         ctx.closePath();
      }
   },
};  // end theProgram variable
```

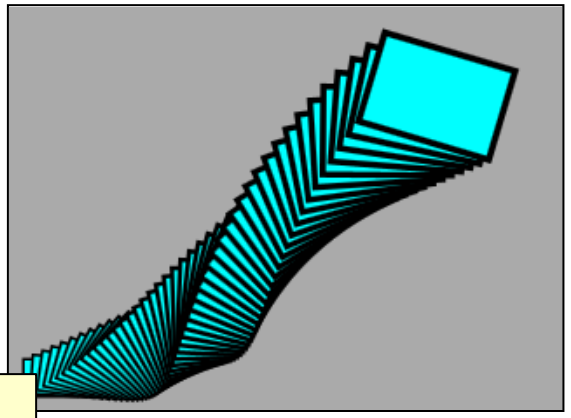
Examples of
translate()
scale()
rotate()

```
window.onload = function()
{
   theProgram.Main();
};
```

translate, scale, rotate

loopRotate.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      // grey background
      ctx.fillStyle="#aaaaaa";
      ctx.fillRect(0, 0, 320, 240);
      // for subsequent boxes:
      ctx.fillStyle="#00ffff";
      ctx.lineWidth = 3;
```



```
for (var i = 0; i < 50; i++) {
         var t = i / 50.0;   // time parameter [0..1)

         ctx.setTransform(1,0,0,1,0,0); // reset to identity

         ctx.translate(10
         ctx.scale(1 + t *
         ctx.rotate(3.5 * t

         ctx.beginPath();
         ctx.rect(0, 0, 30, 20);
         ctx.stroke();
         ctx.fill();
         ctx.closePath();
      }
   },
};  // end theProgram variable
```

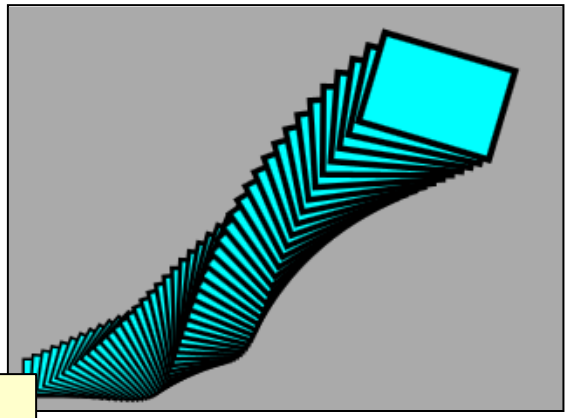
Begins or resets a path
Needed here to make certain each box is fully drawn

```
window.onload = function()
{
   theProgram.Main();
};
```

translate, scale, rotate

loopRotate.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      // grey background
      ctx.fillStyle="#aaaaaa";
      ctx.fillRect(0, 0, 320, 240);
      // for subsequent boxes:
      ctx.fillStyle="#00ffff";
      ctx.lineWidth = 3;
```



```
for (var i = 0; i < 50; i++) {
        var t = i / 50.0;   // time parameter [0..1)

        ctx.setTransform(1,0,0,1,0,0); // reset to identity

        ctx.translate(10 + 270 * t, 210 - 120 * t);
        ctx.scale(1 + t * 1.5, 1 + t * 1.5);
        ctx.rotate(3.5 * t);

        ctx.beginPath();
        ctx.rect(0, 0, 30, 20);
        ctx.stroke();
        ctx.fill();
        ctx.closePath();
      }
   },
};  // end theProgram variable
```

Closes a path
Needed here to make certain each box is fully drawn

```
window.onload = function()
{
   theProgram.Main();
};
```

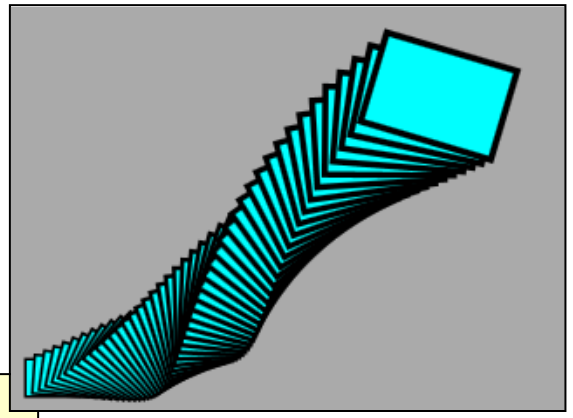
translate, scale, rotate

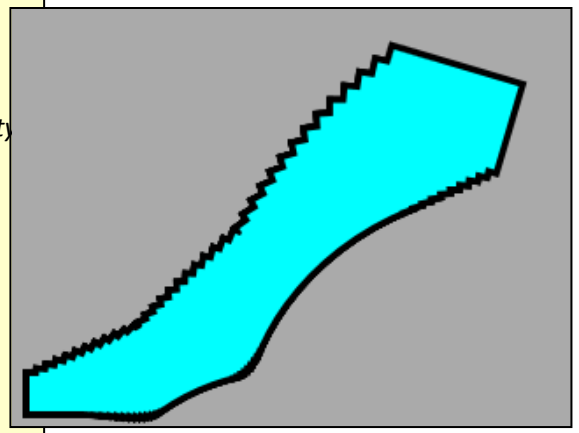
loopRotate.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      // grey background
      ctx.fillStyle="#aaaaaa";
      ctx.fillRect(0, 0, 320, 240);
      // for subsequent boxes:
      ctx.fillStyle="#00ffff";
      ctx.lineWidth = 3;
```

```
for (var i = 0; i < 50; i++) {
         var t = i / 50.0;   // time parameter [0..1)

         ctx.setTransform(1,0,0,1,0,0); // reset to identity

                                  ) * t, 210 - 120 * t);
                                  . + t * 1.5);

         ctx.beginPath();
         ctx.rect(0, 0, 30, 20);
         ctx.stroke();
         ctx.fill();
         ctx.closePath();
      }
   },
};  // end theProgram variable
```

Without the beginPath and closePath
- The boxes are all drawn ‘at the same time’
- Resulting in a different image





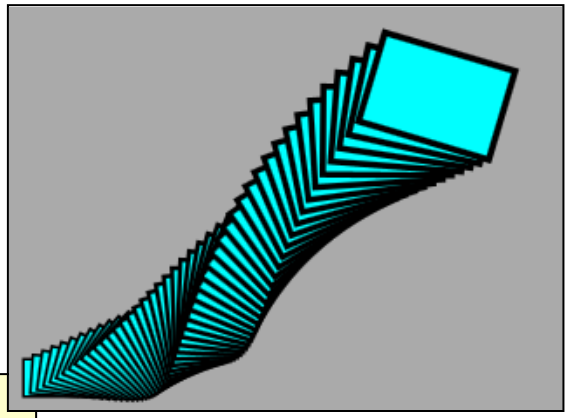
```
window.onload = function()
{
   theProgram.Main();
};
```

translate, scale, rotate

loopRotate.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      // grey background
      ctx.fillStyle="#aaaaaa";
      ctx.fillRect(0, 0, 320, 240);
      // for subsequent boxes:
      ctx.fillStyle="#00ffff";
      ctx.lineWidth = 3;
```

```
for (var i = 0; i < 50; i++) {
        var t = i / 50.0;   // time parameter [0..1)

        ctx.setTransform(1,0,0,1,0,0); // reset to identity
```



Draws a rectangle with opposing
diagonal corners of (0, 0) and (30, 20)

Note: It will be transformed based on current Transform Matrix

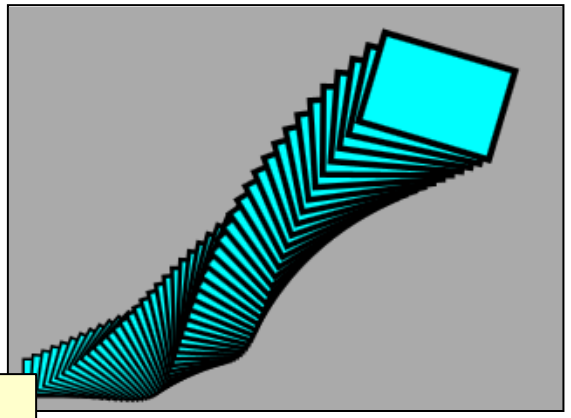
```
        ctx.rect(0, 0, 30, 20);
        ctx.stroke();
        ctx.fill();
        ctx.closePath();
      }
   },
};  // end theProgram variable
```

```
window.onload = function()
{
   theProgram.Main();
};
```

translate, scale, rotate

loopRotate.js

```
var theProgram =
{
   Main: function()
   {
      theCanvas = document.getElementById("myCanvas");
      ctx = theCanvas.getContext("2d");
      // grey background
      ctx.fillStyle="#aaaaaa";
      ctx.fillRect(0, 0, 320, 240);
      // for subsequent boxes:
      ctx.fillStyle="#00ffff";
      ctx.lineWidth = 3;
```



```
for (var i = 0; i < 50; i++) {
         var t = i / 50.0;   // time parameter [0..1)

         ctx.setTransform(1,0,0,1,0,0); // reset to identity

         ctx.translate(10 + 270 * t, 210 - 120 * t);
         ctx.scale(1 + t * 1.5, 1 + t * 1.5);
         ctx.rotate(3.5 * t);

         ctx.beginPath();
         ctx.rect(0, 0, 30, 20);
         ctx.stroke();
         ctx.fill();
         ctx.closePath();
      }
   },
};  // end theProgram variable
```

Questions on the JavaScript?

```
window.onload = function()
{
   theProgram.Main();
};
```

Challenges for Home

- Code is available online (zipped)
  - Download it and experiment
- Try adding a ctx.transform() call to skew each rectangle before it is drawn
- Try using ctx.save() and ctx.store() to “push” and “pop” the transform matrix’s state
  - instead of explicitly resetting it to the identity
- Try other things
  - arcs, gradients, pixel manipulation, pattern fills

Questions?

- Beyond D2L
  - Examples and information can be found online at:
    - *http://docdingle.com/teaching/cs.html*

    - *Continue to more stuff as needed*

Extra Reference Stuff Follows

Credits

- Much of the content derived/based on slides for use with the book:
  - *Digital Image Processing,* Gonzalez and Woods

- Some layout and presentation style derived/based on presentations by
  - Donald House, Texas A&M University, 1999
  - Bernd Girod, Stanford University, 2007
  - Shreekanth Mandayam, Rowan University, 2009
  - Igor Aizenberg, TAMUT, 2013
  - Xin Li, WVU, 2014
  - George Wolberg, City College of New York, 2015
  - Yao Wang and Zhu Liu, NYU-Poly, 2015
  - Sinisa Todorovic, Oregon State, 2015
  - Beej's Bit Bucket / Tech and Programming Fun
    - http://beej.us/blog/
  - w3schools.com