# Vector Graphics for the Web with Raphaël

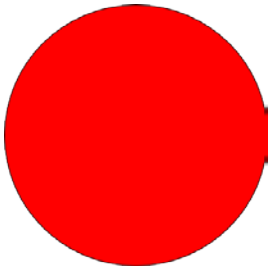
Daniel Hanks | Sr. System Administrator

# Agenda

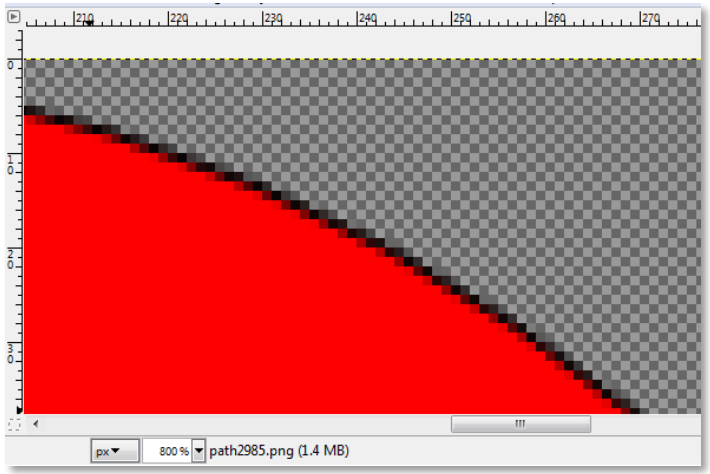
- Vector graphics overview
- SVG overview
- Raphaël overview
- Getting started with Raphaël
- Basic shapes
- Paths
- Raphaël Utilities
- Basic animation

# Vector Graphics Overview

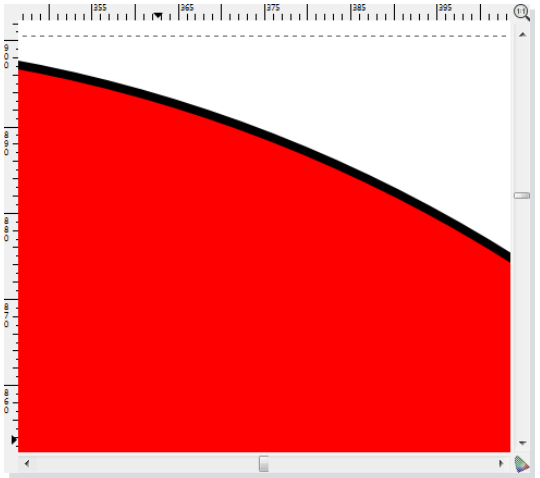
- In contrast to bitmap or raster graphics (think Adobe Illustrator vs. Photoshop)
- Stored as a **description** of the shapes and properties of the image
- Rather than a **pixel-by-pixel** representation of an image
- Can be scaled up in size indefinitely
  - Thus great for print / poster / design applications
  - No pixelating at scale



As a bitmap image




As a vector image



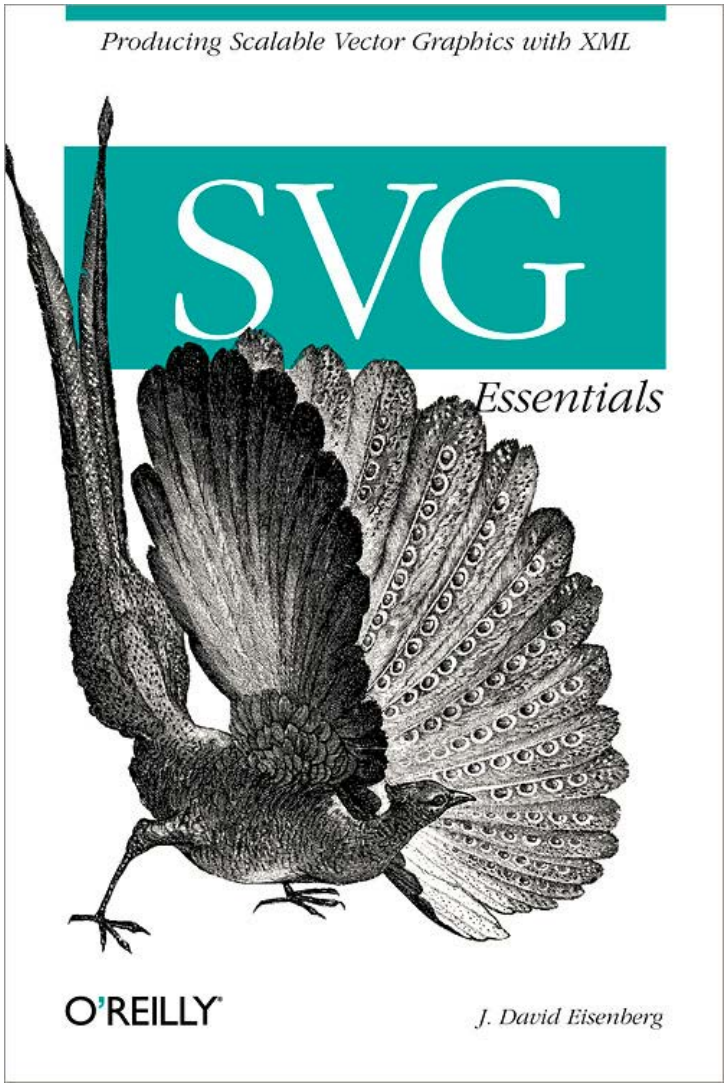
# SVG Overview

"Scalable Vector Graphics (SVG) is an **XML-based** vector image format for two-dimensional graphics that has support for interactivity and animation. The SVG specification is an **open standard** developed by the World Wide Web Consortium (W3C) since 1999.

SVG images and their behaviors are defined in XML text files. This means that they can be searched, indexed, **scripted**, and, if need be, compressed. As XML files, SVG images can be created and edited with any text editor, but it is often more convenient to create them with drawing programs such as Inkscape.

**All major modern web browsers**—including Mozilla Firefox, Internet Explorer 9 and 10, Google Chrome, Opera, and Safari—have at least some degree of support for SVG and can render the markup directly."

– en.wikipedia.org, "Scalable Vector Graphics," *emphasis added.*

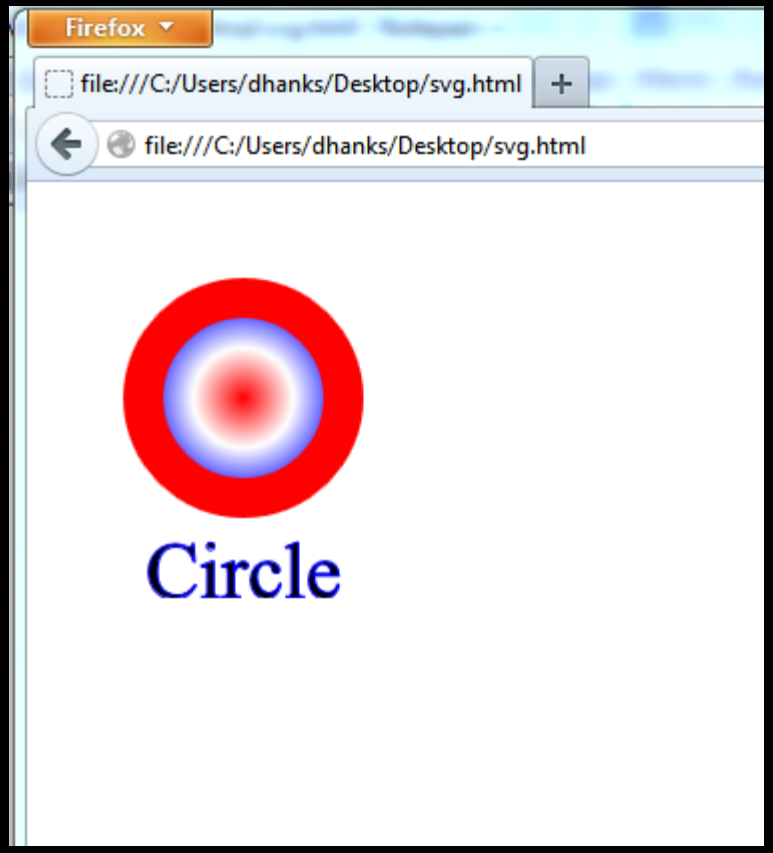


# SVG Example

svg.html

```
<html>
  <svg width="200" height="200">
    <defs>
      <radialGradient id="circlegrad">
        <stop offset="0%" style="stop-color: red;"/>
        <stop offset="50%" style="stop-color: white;"/>
        <stop offset="100%" style="stop-color: blue;"/>
      </radialGradient>
    </defs>
    <circle cx="100" cy="100" r="50" style="stroke: #ff0000;
      stroke-width: 20; fill: url(#circlegrad);"/>
    <text x="100" y="200" style="stroke:blue; text-anchor:middle;
      font-size: 40;">Circle</text>
  </svg>
</html>
```

# SVG Example



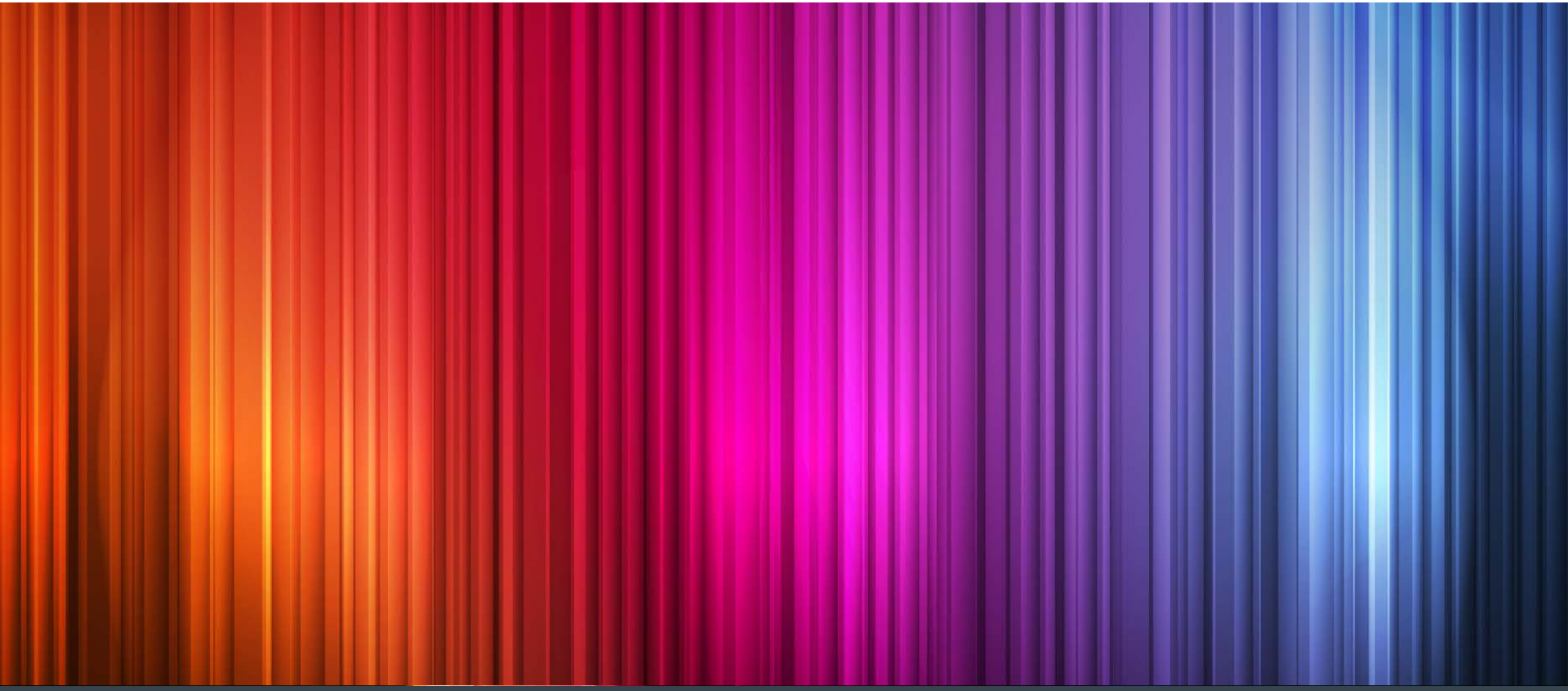
# SVG Capabilities

- Things you can do with SVG
  - Basic Shapes: lines, arrows, circles, ellipses, boxes, etc.
  - Paths: arbitrary lines, polygons, curves
  - Text
  - Patterns, Gradients, Filters, clipping & masking, transformation
  - Animation
- Adobe Illustrator: Save As -> SVG/SVGZ
- SVG is the default file format for Inkscape.

# Raphaël Overview

- Per raphaeljs.org, Raphaël is:
  - "Raphaël is a small JavaScript library that should simplify your work with vector graphics on the web. If you want to create your own specific chart or image crop and rotate widget, for example, you can achieve it simply and easily with this library.
  - Raphaël ['ræfeɪəl] uses the SVG W3C Recommendation and VML as a base for creating graphics. This means every graphical object you create is also a DOM object, so you can attach JavaScript event handlers or modify them later. Raphaël's goal is to provide an adapter that will make drawing vector art compatible cross-browser and easy.
  - Raphaël currently supports Firefox 3.0+, Safari 3.0+, Chrome 5.0+, Opera 9.5+ and Internet Explorer 6.0+."
- A JavaScript library that makes it easy(er) to generate vector art for your web pages.
- Project started by Dmitry Baranovskiy in 2008.

# Getting Started with Raphaël



# Getting Started

sample.html

```
<html>
    <head>
        <!-- Download raphael.js from raphael.com -->
        <script src="raphael.js"></script>
        <script src="my.js"></script>
    </head>
    <body>
        <div id="canvas">
    </body>
</html>
```

my.js

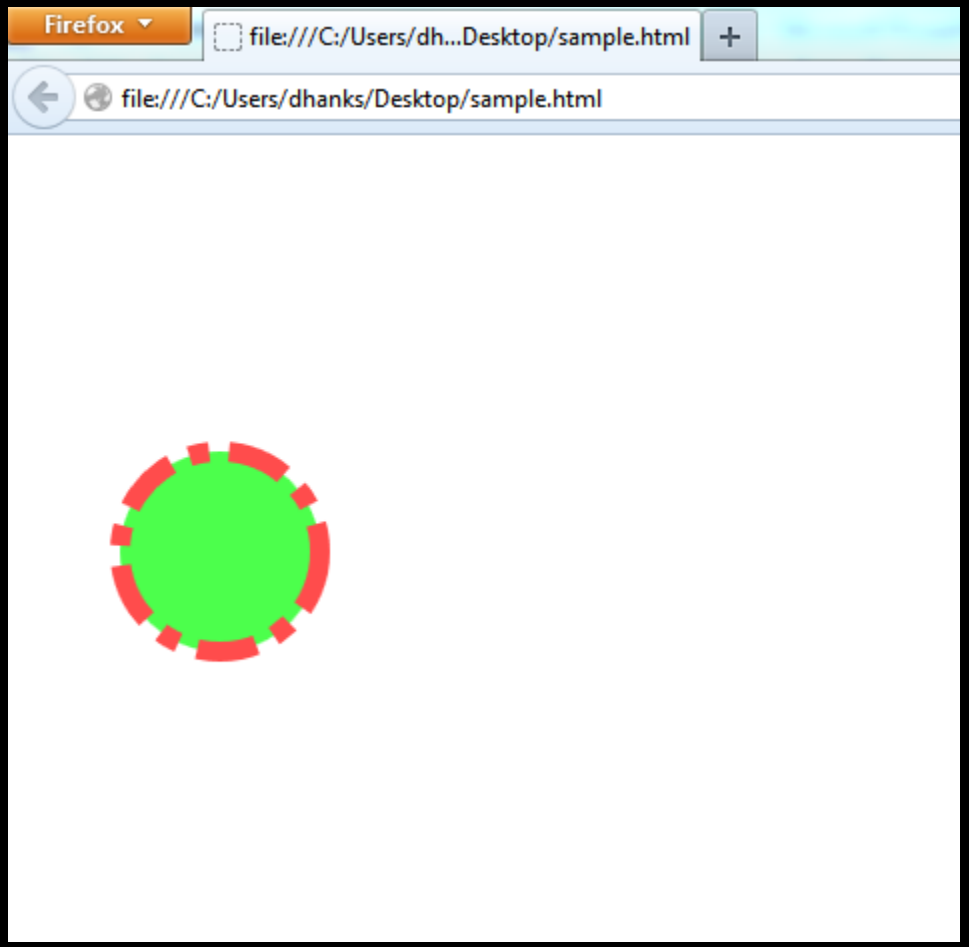
```
window.onload = function() {
    // 400 x 600 canvas
    var paper = new Raphael("canvas", 400, 600);
    // circle with radius 50 at 100, 200
    var circle = paper.circle(100, 200, 50)
    circle.attr({ 'stroke' : '#ff0000', 'stroke-width' : 10 })
    …
}
```

(Show browser output + the resultant Element tree in-browser)

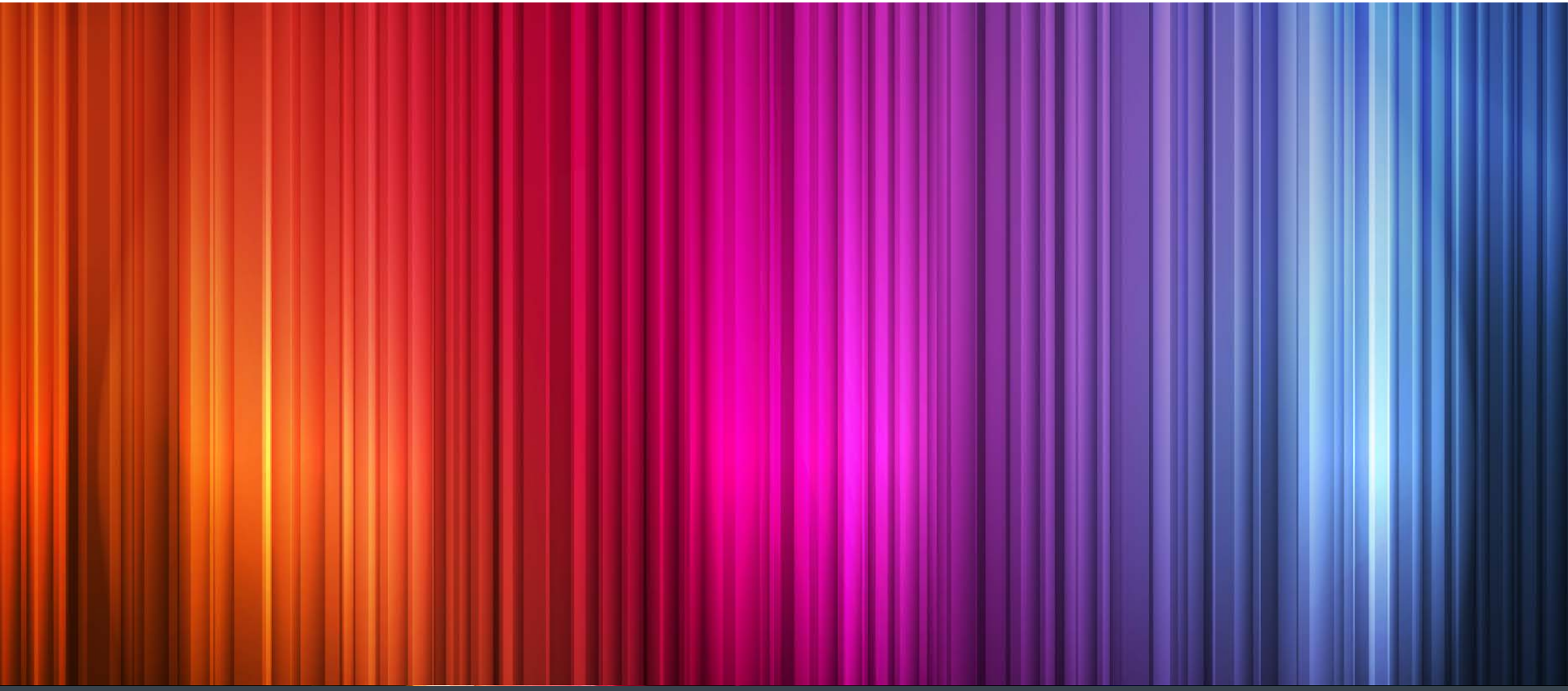
# Getting Started – Getting / Setting attributes

```
// Set multiple attributes – pass in an object
var circle = paper.circle(100, 200, 50);
circle.attr({
  ‘stroke’ : ‘#ff0000’,
  ‘stroke-width’ : 10,
  ‘fill’: ‘#00ff00’,
  ‘opacity’: .7,
  ‘stroke-dasharray’: ‘-.’
  // “”, “-”, “.”, “-.”, “-..”, “. ”, “- ”,
  // “--”, “- .”, “--.”, “--..”
  ...
});

// Or, set one at a time:
circle.attr(‘stroke’, ‘#ff0000’);

// Get the value of attributes
var fill = circle.attr(‘fill’);
var fill_array = circle.attr([‘stroke’, ‘stroke-width’]);
```

# Getting Started – Setting attributes



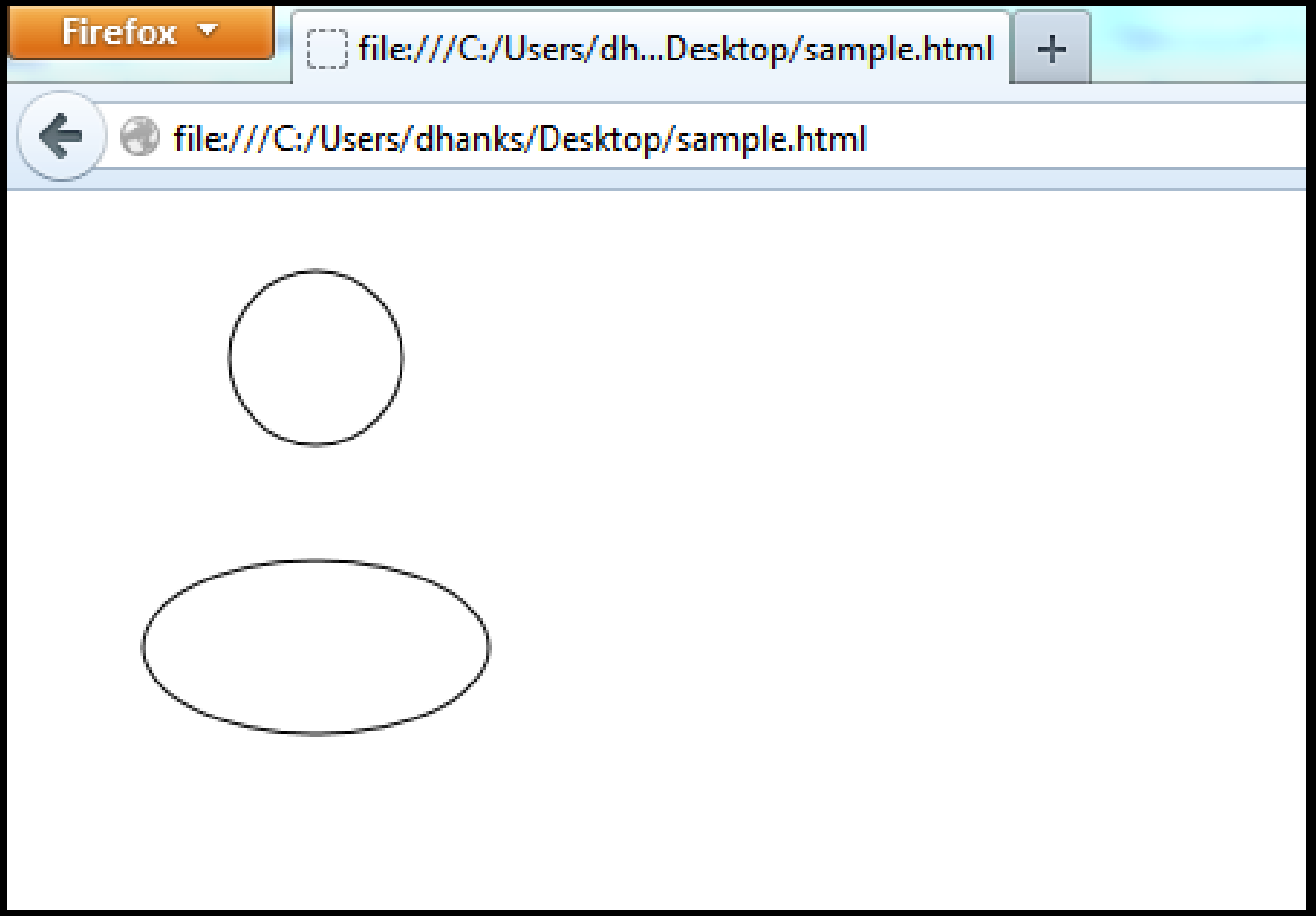
# Basic Shapes



# Basic Shapes – Circles & Ellipses

```
// Circle with radius 30 at 100, 50
var circle = paper.circle(100, 50, 30);

// Ellipse at 100, 150, horiz. radius 60, vert. radius 30
var ellipse = paper.ellipse(100, 150, 60, 30);
```

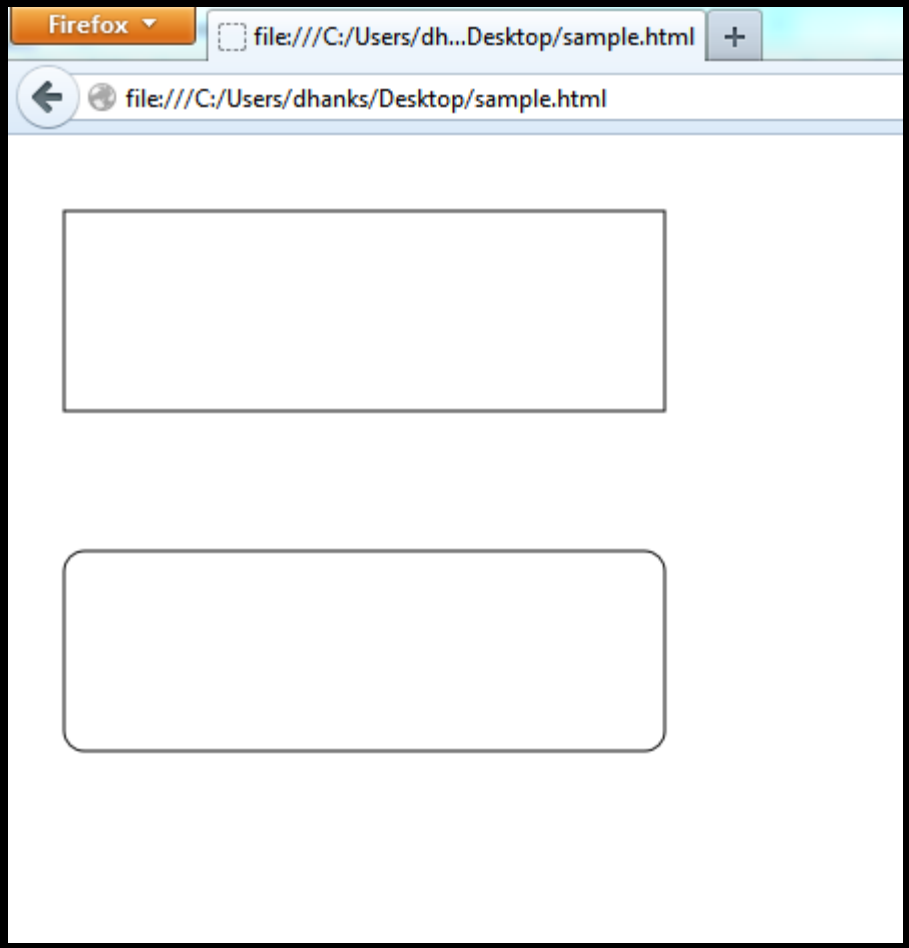
# Basic Shapes – Circles & Ellipses



# Basic Shapes – Boxes

```
// Box at 20, 30, width 300, height 100
var box = paper.rect(20, 30, 300, 100);

// Same thing with rounded corners (corner radius = 10)
var rounded_box = paper.rect(20, 200, 300, 100, 10);
```

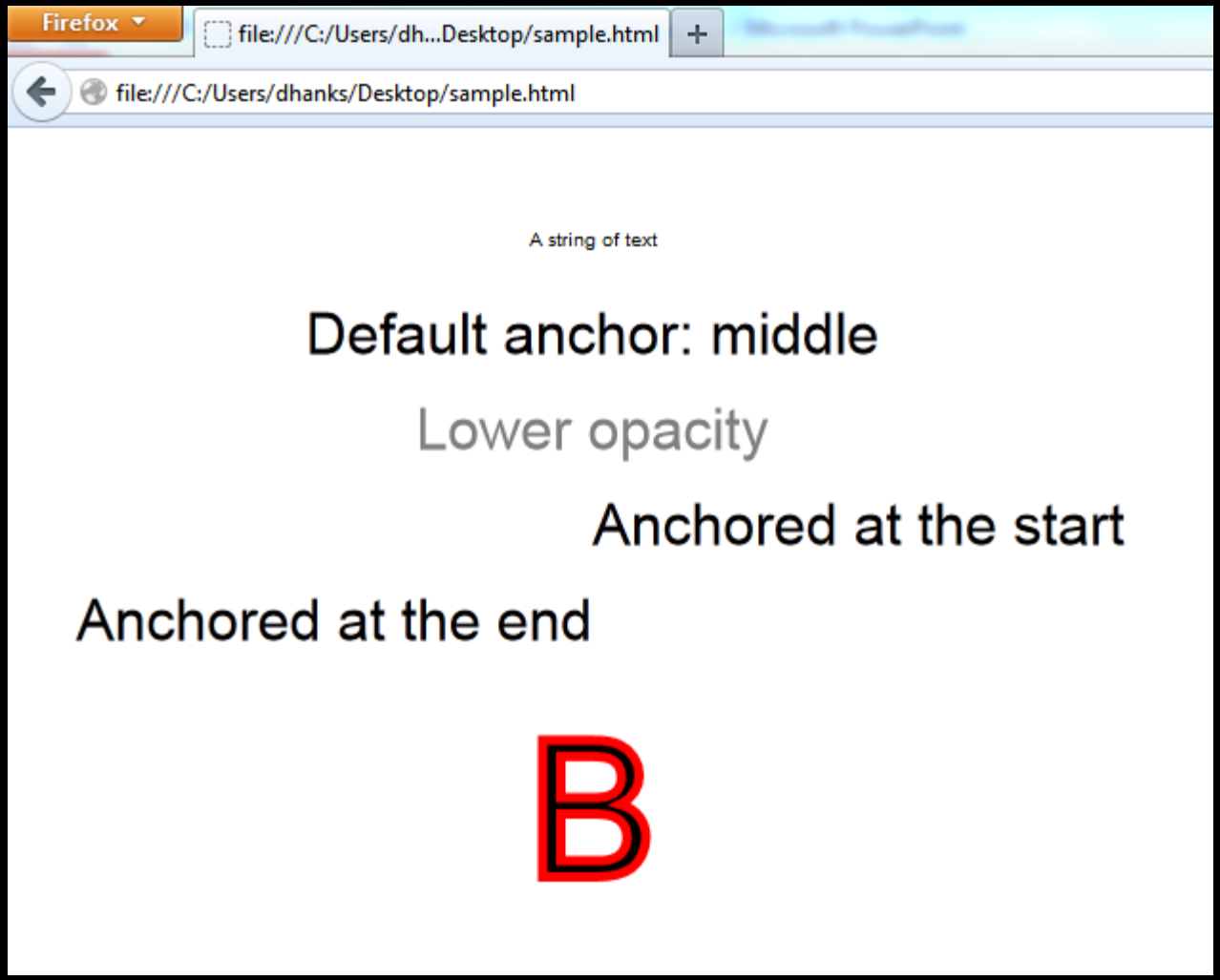
# Basic Shapes – Boxes



# Basic Shapes – Text

```
var text = paper.text(300, 50, "A string of text");
var text = paper.text(300, 100, "Default anchor: middle").attr({
 'font-size' : 30
});
var text = paper.text(300, 150, "Lower opacity").attr({
 'font-size' : 30, 'opacity' : .5
});
var text = paper.text(300, 200, "Anchored at the start").attr({
 'font-size' : 30, 'text-anchor' : 'start‘
});
var text = paper.text(300, 250, "Anchored at the end").attr({
 'font-size' : 30, 'text-anchor' : 'end'
});
var text = paper.text(300, 350, "B").attr({
 'font-size' : 100,
 'text-anchor' : 'middle',
 'stroke' : 'red',
 'stroke-width' : 5,
 'fill' : 'black‘
});
```

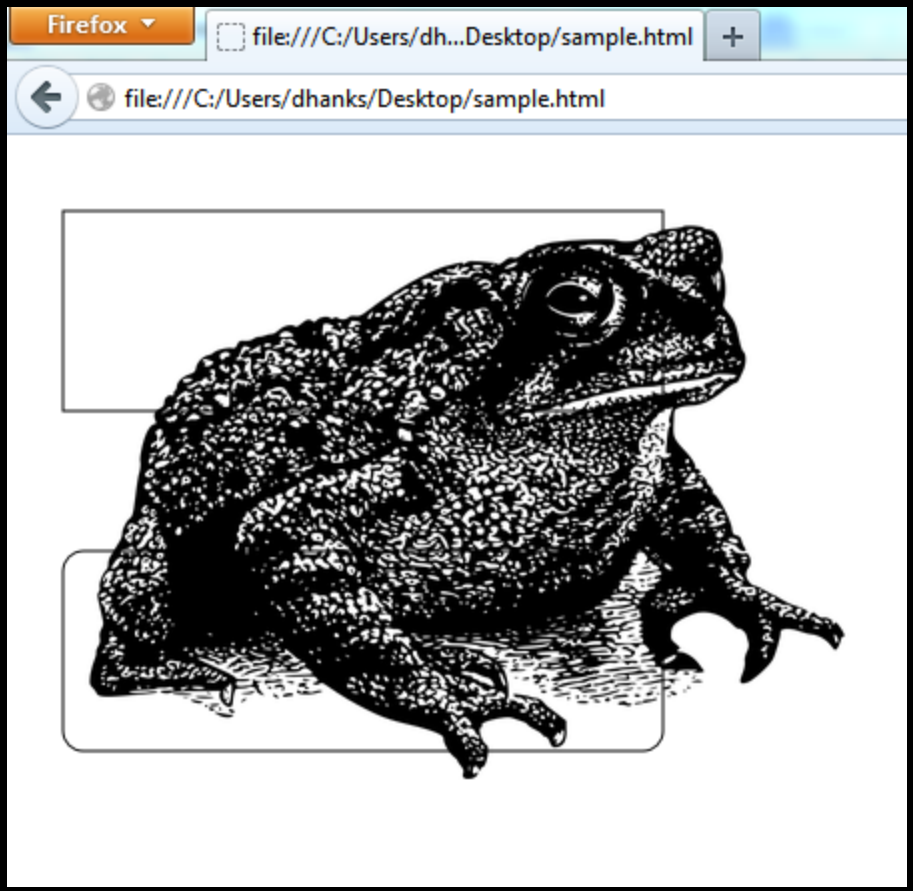
# Basic Shapes – Text



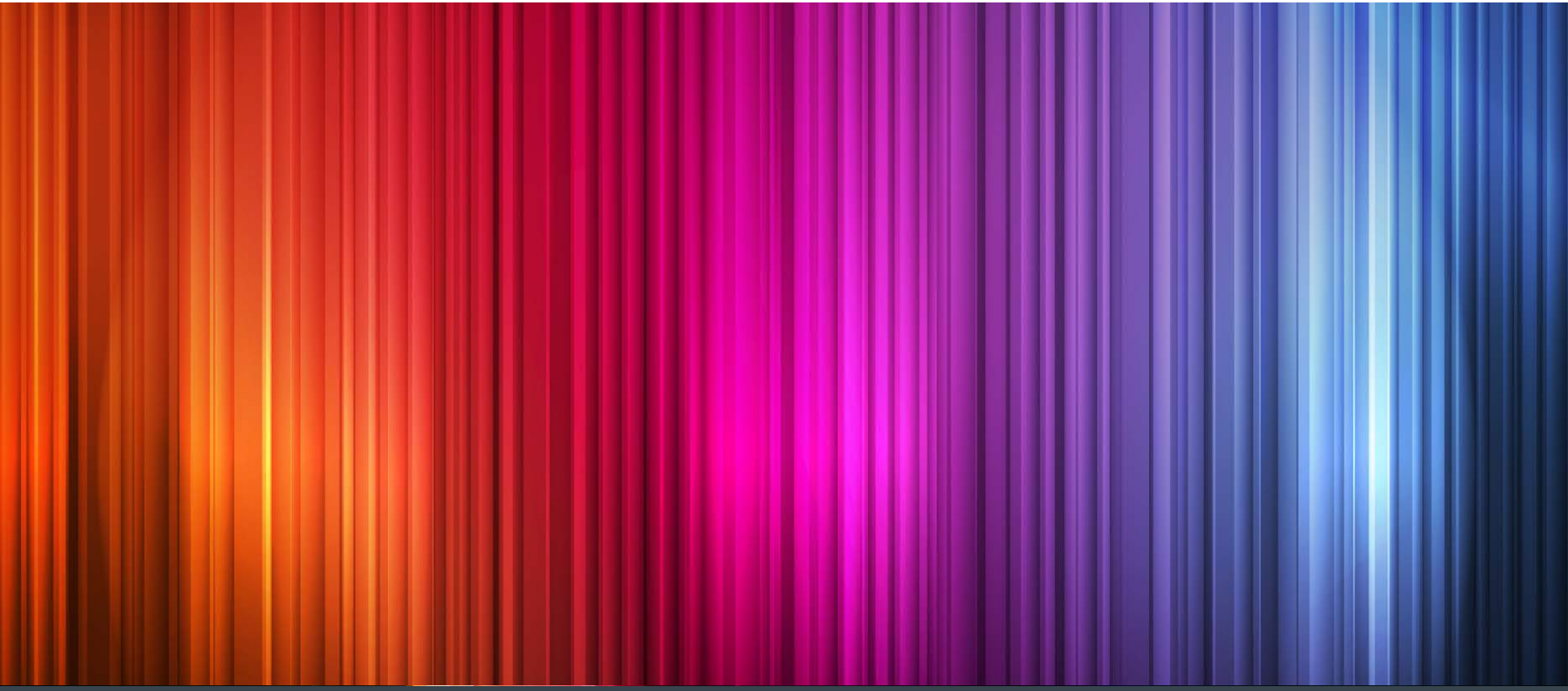
# Basic Shapes – Images

```
// Box at 20, 30, width 300, height 100
var box = paper.rect(20, 30, 300, 100);

// Same thing with rounded corners (corner radius = 10)
var rounded_box = paper.rect(20, 200, 300, 100, 10);

// Image at 20, 20, width 400, height 294
var image = paper.image("Toad.png", 20, 20, 400, 294)
```

# Basic Shapes – Images

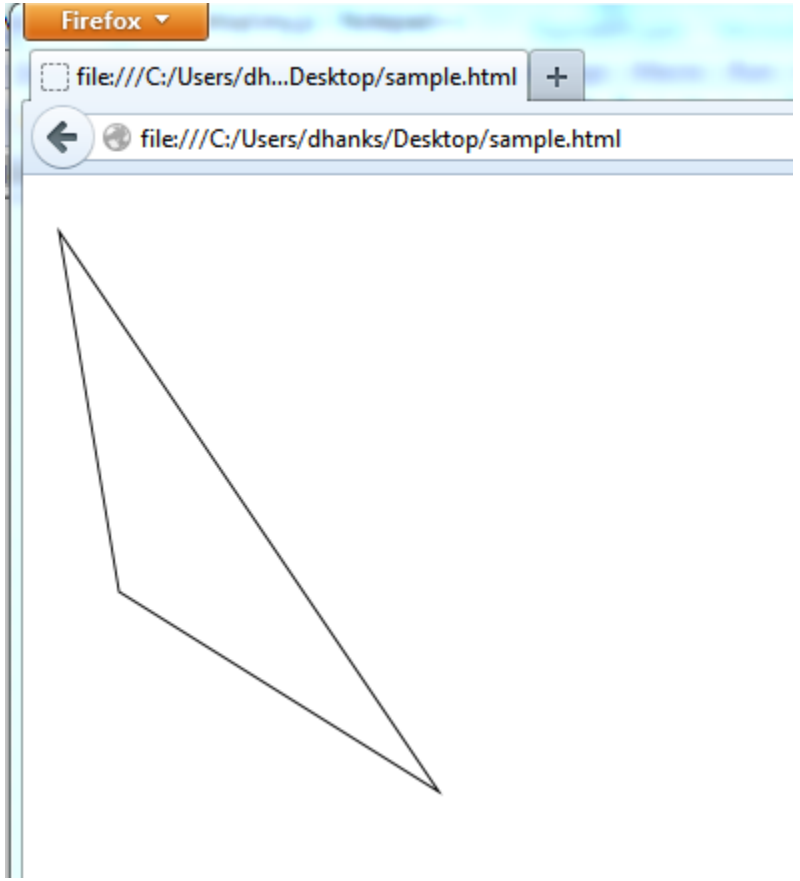


# Paths



# Paths – Basics

- Arbitrary lines, curves, and polygons
- The other shapes are just shortcuts for Paths
- Create a path with a Path string:
  - E.g., "M 10 20 L 200 300 L 40 200 z"
  - Means:
    - **M**ove to (10,20),
    - draw a **L**ine to (200,300),
    - another **L**ine to (40,200),
    - then go back to the starting point (z)."
  - Looks like this:



# Paths – Straight Line Path Strings

- Path String options:
  - **"M** x y" Pen up and **M**ove to x, y (absolute)
  - "**m** x y" Pen up and **M**ove to current location + x, current location + y (relative)
  - "**L** x y" Draw a **L**ine to x, y (absolute)
  - **"l** x y" Draw a **l**ine to current location + x, current location + y (relative)
  - Z or z Draw a line back to the starting point
  - "H x" Draw a line **H**orizontally to position x (absolute)
  - "h x" Draw a line **h**orizontally to current x + x (relative)
  - "V y" Draw a line **V**ertically to position y (absolute)
  - "v y" Draw a line **v**ertically to current y + y (relative)

# Paths – Curved Path Strings

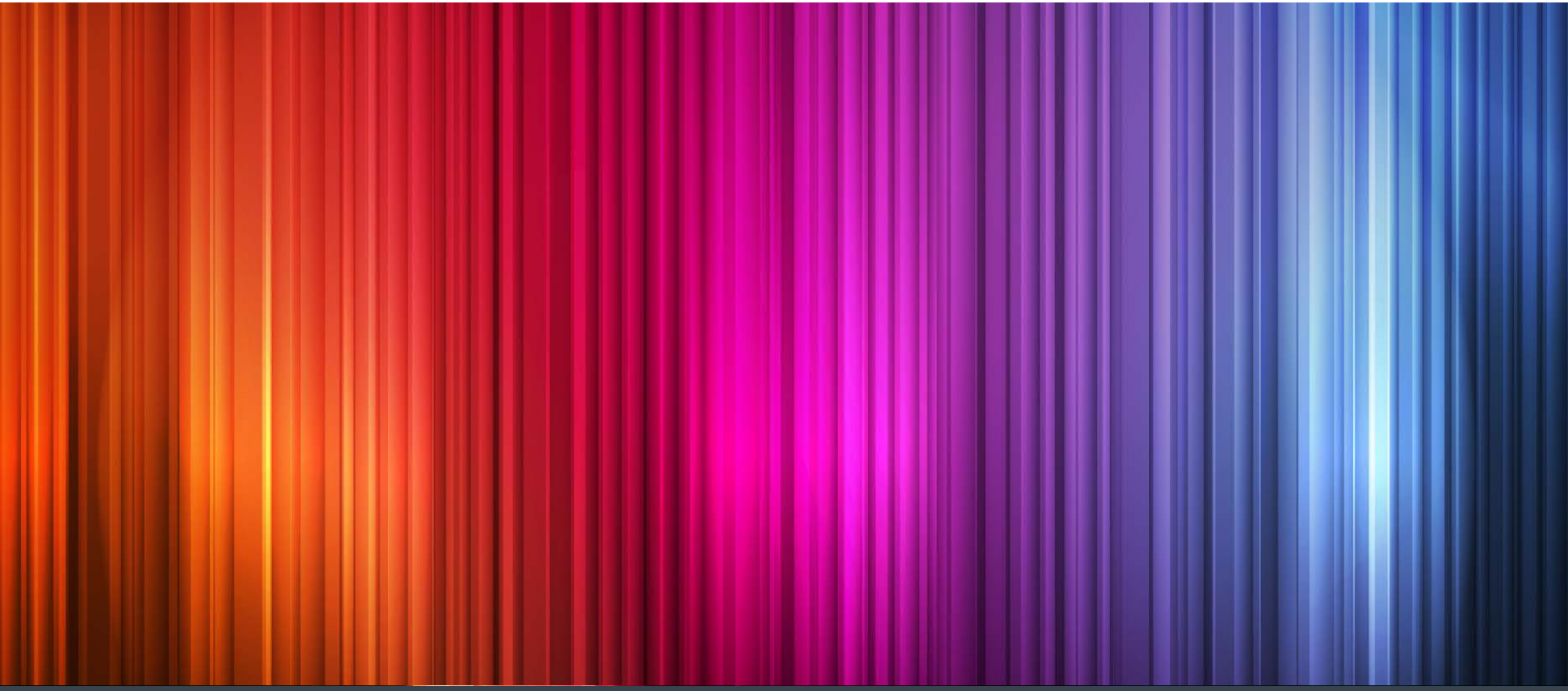
- Curved Path String options (left as an exercise for the reader…):
  - C Curve to
  - S Smooth curve to
  - Q Quadratic curve to
  - T Smooth Quadratic Bezier curve to
  - A Elliptical arc

# Paths Example – Worldmap.svg

Path string for the country outline of Jordan:

<path xmlns="http://www.w3.org/2000/svg" class="landxx jo" d="M 1530.4073,505.50955 C 1530.2693,503.09555 1531.2413,500.79655 1531.2643,498.39655 1531.2913,495.63755 1532.5223,493.59155 1532.8563,490.96555 1532.9963,489.87155 1532.6803,488.75555 1532.7663,487.64455 1532.8483,486.59455 1533.1978,485.56155 1533.2368,484.53855 1533.3168,482.39455 1533.0232,479.8195 1532.9442,477.6555 1532.8782,475.8185 1535.1703,476.38355 1536.2173,477.10755 1537.9423,478.29955 1539.5016,480.10541 1541.8206,479.64941 1544.1556,479.18941 1546.3043,476.99355 1548.2353,475.70655 1550.7033,474.06255 1553.1673,472.24455 1555.7573,470.80555 1556.5563,473.36555 1557.6043,475.83655 1558.4873,478.36655 1557.5973,478.46055 1558.1783,479.67555 1558.7513,479.82255 1559.6593,480.05555 1560.8049,479.91275 1559.3359,481.45373 1554.7274,484.27094 1550.1117,485.76424 1544.5722,487.146 L 1552.8316,495.81314 C 1552.0886,496.17814 1550.1241,496.13361 1549.5951,496.79961 1548.9871,497.56561 1549.6761,499.11214 1549.0259,499.87814 1547.9349,501.16314 1545.181,500.06891 1543.767,501.41891 1542.154,502.95891 1541.6265,505.85414 1539.3225,506.76114 1537.2375,507.58014 1532.5203,506.05655 1530.4073,505.50955" id="jo"/> (http://commons.wikimedia.org/wiki/File:BlankMap-World6.svg)

# Raphaël Utilities & Useful functions

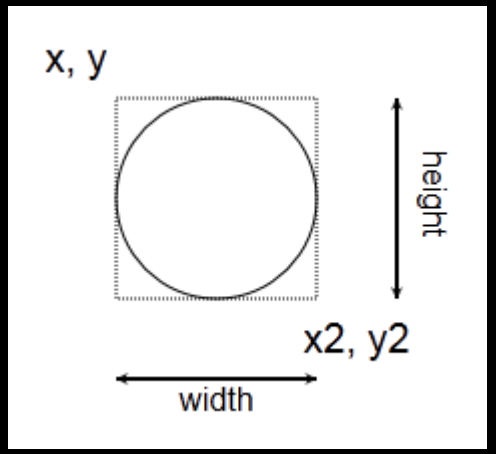


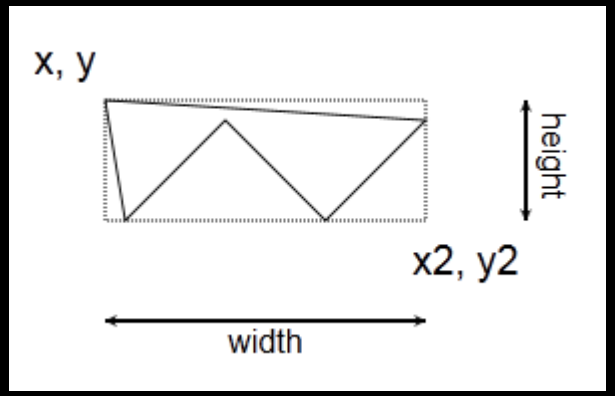
# Useful functions – Path Formats

```
// Usr Raphael.format() to generate path strings

var x = 10, y = 20, line_x = 50, line_y = 50, vert_y = 100;

var my_path = paper.path(
    Raphael.format(
        “M{0} {1} L {2} {3} V {4}”,
        x,
        y,
        line_x,
        line_y,
        vert_y
    )
);

// Produces “M10 20 L 50 50 V 100”
```

# Useful functions – Bounding Boxes

```
// Need to know the boundaries of an object?
var c = paper.circle(100, 200, 100);

var circle_bbox = circle.getBBox();

// Yields the following object:
// {
//     x : top_left_corner_x
//     y : top_left_corner_y
//     x2 : bottom_right_corner_x
//     y2 : bottom_right_corner_y
//     width : width
//     height : height
// }
```



# Useful functions – Event Handlers

```
var c = paper.circle(100, 100, 50);

// Call a handler when the object is clicked…
c.click(function () { ... });

// When an object is double-clicked
c.dblclick(function () { ... });

// When you hover over an object
c.hover(
  f_hover_in,
  f_hover_out
  [, hover_in_context_object ]
  [, hover_out_context_object ]
);

// And others…
// drag(), mousedown(), mousemove() mouseout(),
// mouseover(), mouseup(), touchcancel(), touchend(),
// touchmove(), touchstart()
```

# Useful functions – Miscellaneous

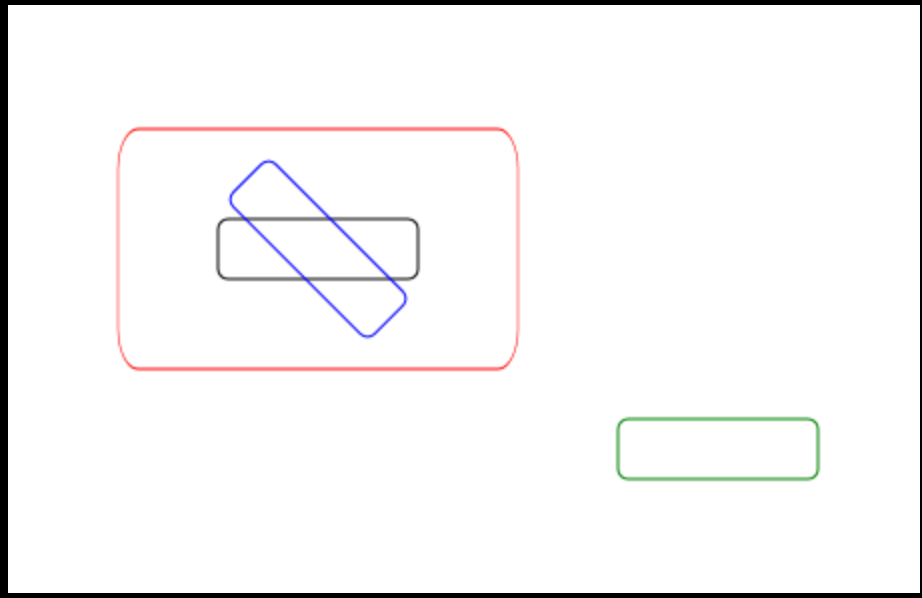
```
var c = paper.circle(100, 100, 50);

c.hide();
c.show();
var c_copy = c.clone();

// Store data in your object:
c.data('key', 'value');
var c_data = c.data('key');

c.toBack();
c.toFront();
Paper.clear();

Var c_id = c.id;
my_circle = Paper.getById(c_id);

Raphael.ninja() // Removes all trace of itself…
```

# Useful functions – Transformation

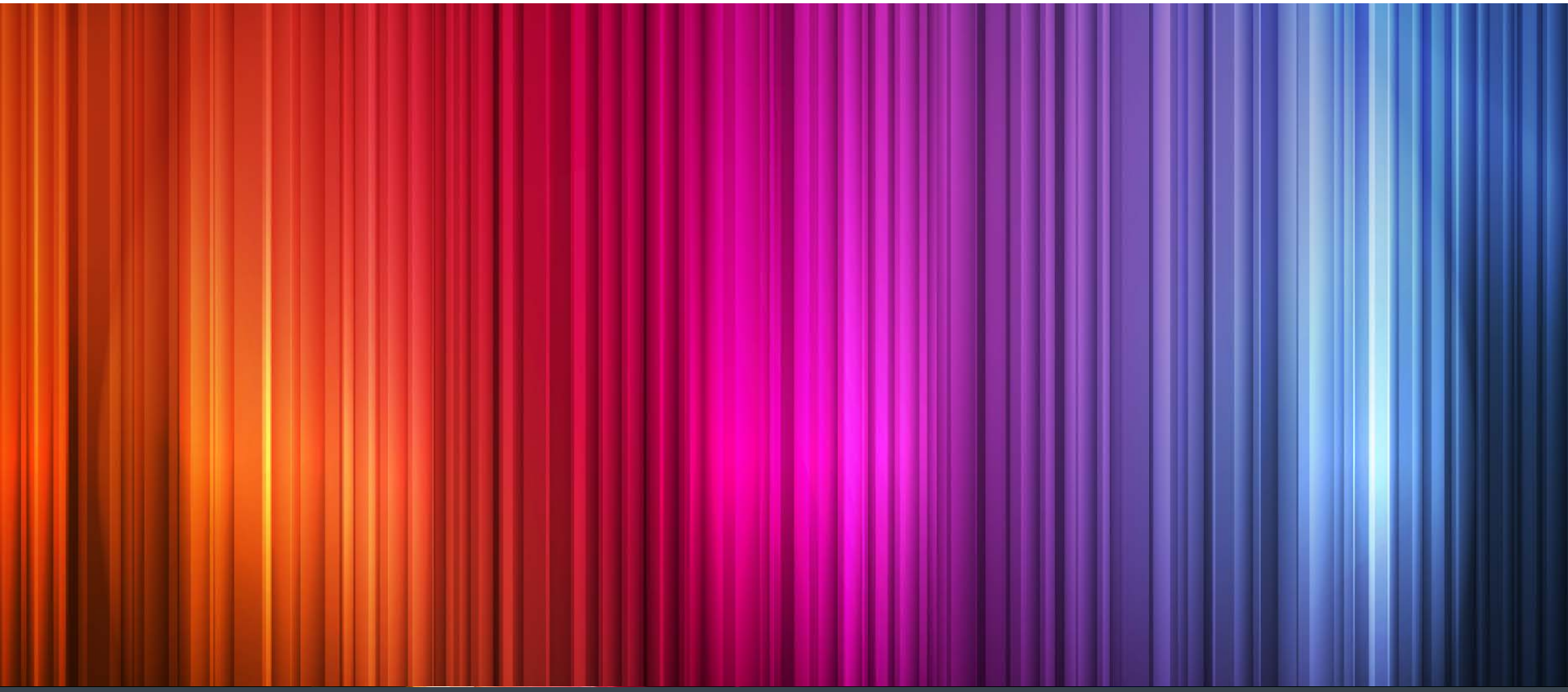
```
// Box at 10, 10, 100 x 200, with rounded corners
var box = paper.rect(10, 10, 100, 200, 5);

// Scale, and rotate the box by passing in a transform
// string (similar to paths)
box.transform("s1.5 r45");

// Available transforms
// sx[,y]     = Scale by x_factor, y_factor
// rN[,x,y]   = Rotate by N degrees, around point x,y
// tN,M       = Translate by Nx,My
// m          = Matrix (takes 6 parameters, not sure what
it does…
// Use capital S, R, T, M for 'absolute' transforms (which
// don't take previous transformations into account).
```

# Useful functions – Transformation

```
// Box at 10, 10, 40 x 20, with rounded corners
var box = paper.rect(100, 100, 100, 30, 5);
var scale_clone = box.clone();
scale_clone.transform("s2,4").attr('stroke','red');
var rotate_clone = box.clone();
rotate_clone.transform("r45").attr('stroke','blue');
var translate_clone = box.clone();
translate_clone.transform("t200,100").attr('stroke','green');
```



# Basic animation



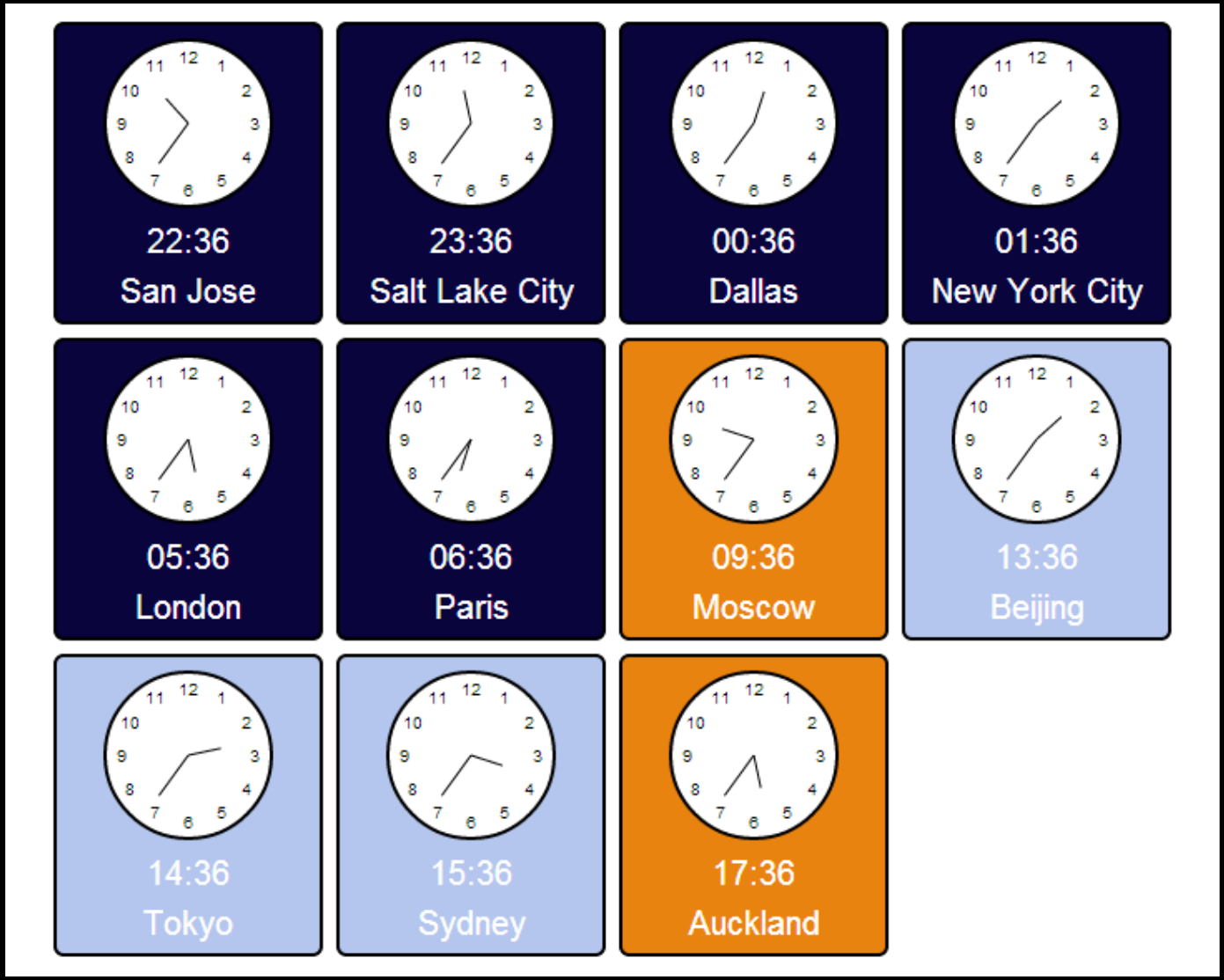
# Animation

```
// Box at 100, 100, 100 x 200, with rounded corners
var box = paper.rect(100, 100, 100, 200, 5).attr('fill', 'blue');

// Animate by passing in an array of new attributes,
// duration of the animation, and string indicating which
// ‘easing’ formula to use (http://raphaeljs.com/easing.html)
Box.click(function() {
  box.animate(
    {
      'width' : 400,
      'height' : 300,
      'transform' : 'r45',
      'fill' : 'red',
      'opacity' : .5,
      'stroke-width' : 10,
    },
    3000,
    'elastic'
  );
}                                    // Demo...
}
```

# Animation

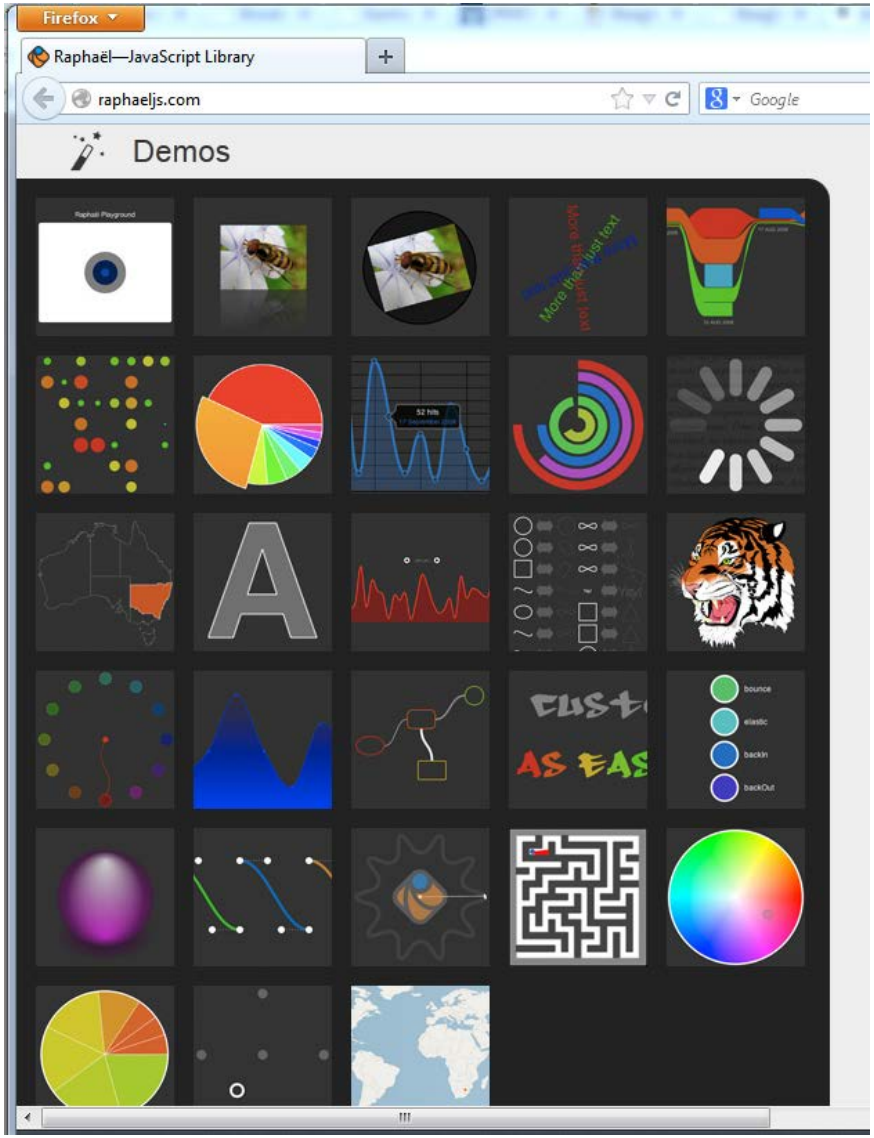
```
// Create an object, and transmogrify it...
var obj = paper.path("M 40 40 L 50 100 L 100 50 L 150 100 L
200 50 z").attr('fill', 'blue');

obj.click(
  function() {
    obj.animate(
      {
        'path' : "M 60 40 L 70 120 L 200 10 L 10 200 L 200 50
z",
           'fill' : 'red',
      },
      5000,
      'elastic'
    );
  }
);

// Local Demo + Helvetica demo on raphaeljs.com...
```

# Demo – World Clock



# Info

- And More!
- Raphaeljs.com
  - Downloads
  - Documentation
  - Demos, etc.
- Me
  - Brainshed.com
    - Slides
  - @danhanks
  - danhanks@gmail.com



# Questions?

# Other sessions of interest…

- Next session on this track:
  - **“Using D3, Cubism, and WebSockets to Make Your Little Heart Go Pitter-Pat”**

# Raffle!

- Special thanks to Marsee & co at O'Reilly and Associates!


Adobe