# CMPT 165

## INTRODUCTION TO THE INTERNET AND THE WORLD WIDE WEB

By Hassan S. Shavarani

## *UNIT6: JAVASCRIPT AND GRAPHICS*

# TOPICS

1. More jQuery Methods
2. JavaScript + SVG: Raphaël
3. About SVG
4. Working with SVG
5. Animating SVG

# `$` / `jQuery`

the jQuery library uses `$`

to mean *exactly* the same thing as `jQuery`

```
jQuery('p').click(say_hello)
$('p').click(say_hello)

jQuery('#changeme').html('Somebody clicked me.')
$('#changeme').html('Somebody clicked me.')
```

# DIFFERENT WAYS TO MODIFY THE PAGE

```
$('#changeme').html('Somebody <em>clicked</em> me.')
$('#changeme').attr('class', 'active')
$('img#changeme').attr('src', 'other_img.png')
$('section#expanding').append('<p>new paragraph.</p>')
newstyle = {
    'font-size': '1.5em',
    'margin-left': '2em'
}
$('#styling').css(newstyle)
$('#styling').animate(newstyle, 2000)
```

* the `.animate()` function can only animate CSS properties that are *numeric*
e.g. it **cannot** animate **font-weight** from normal to bold, or colour values

# JAVASCRIPT + SVG: Raphaël

the library used to create and manipulate SVG graphics on the page

the images are made up of shapes (vector graphics format) not pixels

## JAVASCRIPT + SVG: Raphaël

in `Raphaël`, we will use JavaScript code
to create of manipulate those shapes

working with bitmapped images,
there are libraries to help with that

vector images are just easier to work with to start with!

let's try `Raphaël` for the first time in here!

and Here is a more complicated example

you can find out more about the `Raphaël` library
in this reference

# SVG IMAGES

the SVG image format is strange in some ways

you use it like other image formats (PNG, JPEG, etc.)

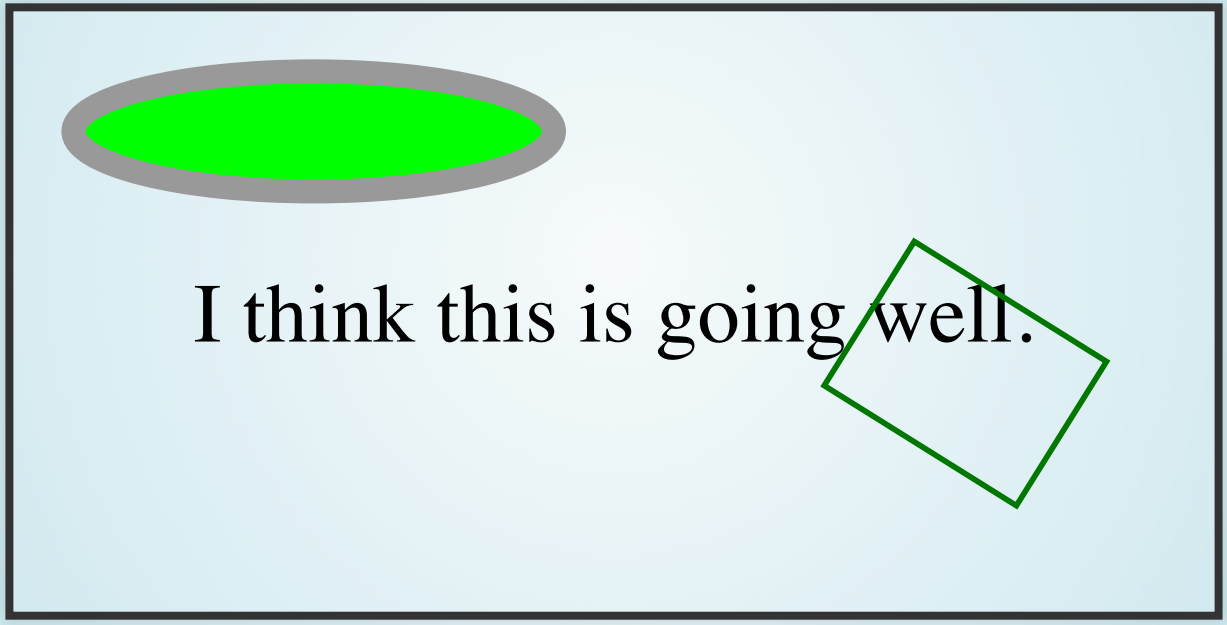
but it technically has more in common with HTML

# THE SVG FORMAT

SVG images are stored as text files, and contain tags and attributes, much like HTML

here you can find more about the rules on svg standard

# THE SVG EXAMPLE



# THE SVG EXAMPLE - SOURCE CODE

```
<svg height="300" width="600" xmlns="http://www.w3.org/2000/svg">

    <ellipse cx="150" cy="60" rx="120" ry="30"
        fill="#00ff00" stroke="#999999" stroke-width="12">
    </ellipse>

    <text x="300" y="150" text-anchor="middle" font-size="42px">
        <tspan dy="15">
            I think this is going well.
        </tspan>
    </text>

    <rect x="450" y="150" rx="0" ry="0" fill="none"
        width="120" height="90" stroke="#007700" stroke-width="3"
        transform="matrix(0.8,0.5,-0.5,0.8,165,-230)">
    </rect>

</svg>
```

for editing a SVG file you can use Inkscape;
a free, open-source, vector graphics editor

it is designed as an SVG editor
the format that it works with by default and best is SVG

# DYNAMICALLY MODIFYING SVG

Since we can use JavaScript code to modify SVG images

we can do that work anywhere we want in our code

Lets try it here!

# `paper.path()` - DRAWING LINES AND CURVES

we can use the **`paper.path()`** function
to draw lines and curves on the canvas!

we need to pass it a *path string* upon calling it

## `paper.path()` - DRAWING LINES AND CURVES

before delving into the details,
let's try an example first, here!

# `paper.path()` - DRAWING LINES AND CURVES

the *path string* is a series of these commands that will be used to draw the path

the path will contain the following commands [remember that (0,0) is in the upper-left]

- **M**: move to the point (x, y), e.g. **'M50,20'**
- **L**: draw a straight line from current position to (x,y), e.g. **'L390,30'**
- **Z**: close the path (draw a straight line back to the starting point)
- **T**: curve through br(draws a curve ending to the selected point), e.g. **'T150,80'**
- **Q**: control point (chooses a point that "pulls" the curve toward it), e.g. **'Q150,180 150,120'**

# paper.text()

draws a text string as a part of the SVG Image

```
var t = paper.text(50, 50, "Raphaël\nis\ncool!");
```

# ANIMATING SVG

## `.animate()` FUNCTION

all the Raphaël shape objects have a **`.animate()`** function (as the exact function jQuery objects do)

when using **`.animate()`**, the *shapes' attributes* would change in an animated manner

# `.animate()` FUNCTION - EXAMPLE

```
initial_attr = {
    'fill': '#fff',
    'stroke-width': '1'
}
final_attr = {
    'fill': '#f00',
    'stroke-width': '5'
}
rect = paper.rect(10, 10, 20, 20)
rect.attr(initial_attr)
rect.animate(final_attr, 2000)
```

if we have a variable referring to the shape (**rect** in that example),
we can perform the animation any time, such as later when the user clicks something

# *TRANSFORM STRING* ATTRIBUTE

one of the most useful attributes to animate with Raphaël is the element's transformation

*transform string* is used to specify the transformation and it can be done using following commands

- **t** for translation (moving)
- **r** for rotation
- **s** for scaling

# *TRANSFORM STRING* ATTRIBUTE EXAMPLE

```
rotated = {
    'transform': 'r180'
}
shape.animate(rotated, 5000)

slide_grow = {
    'transform': 't100,200s3'
}
shape.animate(slide_grow, 1000)
```

# ANIMATING SVG - EXAMPLES

let's study these two examples carefully

- Example 1: User-Initiated Animation
- Example 2: Repeating Animation

# RAPHAËL QUESTIONS

# QUESTION 1

How can I change the background color of the Raphaël canvas itself?

```
p2.canvas.style.backgroundColor = '#F00';
```

# QUESTION 2

How can I define a function for a Raphaël shape ?

```
Raphael.el.red = function () {
    this.attr({fill: "#f00"});
};
paper.circle(100, 100, 20).red();
```

# QUESTION 4

How can I access the equivalent DOM Element of a Raphaël shape?

```
var c = paper.circle(10, 10, 10);
c.node.onclick = function () {
    c.attr("fill", "red");
};
```

# QUESTION 5

## How can I define a bunch of functions that I can use later for Raphaël shapes?

```
Raphael.fn.arrow = function (x1, y1, x2, y2, size) {
    return this.path( ... );
};
// or create namespace
Raphael.fn.mystuff = {
    arrow: function () {...},
    star: function () {...},
    // etc ...
};
var paper = Raphael(10, 10, 630, 480);
// then use it
paper.arrow(10, 10, 30, 30, 5).attr({fill: "#f00"});
paper.mystuff.arrow();
paper.mystuff.star();
```

# Any Questions?