

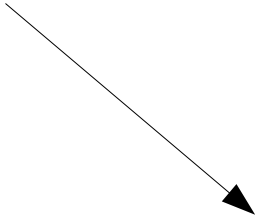


Raphael.js

Paths
Animation
Interface with WebIOPi

A Look Back at Events

- The parameter in Raphael event listener functions is the **HTML DOM event object**
- Example:



```
stuff.mouseover(function(e) {  
    label.attr({  
        text: this.attr("fill"),  
        x: e.clientX,  
        y: e.clientY  
    });  
});
```

Paths

- Paths are represented as a long string of characters, similar to translations

```
var paper = Raphael(0, 0, 300, 300);  
var d = "M 10,30 L 60,30 L 10,80 L 60,80";  
var mark = paper.path(d);
```

- This translates to: “Move (M) to the coordinates (10,30), draw a line (L) to the coordinates (60,30), then a line (L) to (10,80), and then a line (L) to (60,80)”
 - Paths should always begin with an M command
 - Commas between numbers can be replaced with spaces
 - Spaces between letters and numbers are not necessary

More Path

- Add style attributes: **example**
 - Try adding fill
 - Try adding a 'z' to the end of the path specification

```
var d = "M10 30L60 30L10 80L60 80z";
```
- Upper-case path designations (L,M,H,V...) are with reference to the global coordinate system with origin at the upper-left corner of the canvas
- Lower-case path designations (l,m,h,v...) are with reference to the local coordinate system, i.e., offsets are relative to the previous coordinate.
 - Equivalent path designation as above:

```
var d = "M10,30l50,0l-50,50l50,0";
```
- Can mix upper and lower-case designations

Paths

- Raphael stores paths as an array in which each object represents one command with a letter and some numbers

```
var tri = paper.path([["M", 90, 90], ["L", 10, 70], ["L", 50, 5], ["L", 90, 90]]);
```

- Raphael paths following the **W3C SVG recommendations**

Elliptical Curves

- Elliptical Curves
 - Example of the **A Command**: A 100,70 0 0,1 250,220
- Those numbers represent:
 - The horizontal and vertical radii of the imaginary ellipse we're using as a guide
 - An angle rotating the curve's axis (for advanced users)
 - A Boolean value (or "flag") that is either 0 or 1, representing whether a curve goes clockwise or counterclockwise
 - A Boolean value (or "flag") representing whether the curve goes the long way or the short way
 - The ending point
 - Note: the arc starts at the current point
- An **example** to play with

Cubic Bézier curve

- The **C command** takes three pairs of coordinates: a destination and two control points that determine how the line bends
 - Two control points come first and then the destination as the third coordinate.
 - Path begins wherever the previous command left off: at the current point
 - **Example:**

```
var path = "M 50,100 C 100,50 200,150 250,100";
```

 - Try changing the control points
 - **Demo of control points**

Animations

- An animation is a change in a shape's attribute values over time
 - Almost all attributes support being animated
- A basic animation consist of
 - The final shape state; an attribute object which the animated shape will have when the animation is finish
 - The duration, this is the number of milliseconds that the animation runs
 - An animation formula that determines HOW the final state is reached from the initial state
 - The default **designations** are: linear, easeIn, easeOut, easeInOut, backIn, backOut, elastic, and bounce.
- Example:

```
circle1.animate({fill: "blue", transform: "s2.0"}, 1000, "linear");
```
- The animate method has a fourth optional parameter which is a callback function fired when the animation finishes

Animation Examples

- **Example** On JSFiddle
 - Try changing the characteristics of the animation
- Path **animation example**
- A simple **simulation**
- Final note: Animation objects have two methods of their own: `delay()` and `repeat()`
 - `delay()` takes an input of milliseconds that cause the animation to wait before firing
 - `repeat()` takes an integer for the number of times the animation should loop

Adding Functions

- Add your own method to the canvas

- Raphael.fn example:

```
Raphael.fn.arrow = function (x1, y1, x2, y2, size) {  
    return this.path( ... );  
};  
var paper = Raphael(10, 10, 630, 480);  
paper.arrow(10, 10, 30, 30, 5).attr({fill: "#f00"});
```

- Add your own method to elements

- Raphael.el example

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

- Represent a set of attributes as a function

- paper.customAttributes

```
paper.customAttributes.hue = function (num) {  
    num = num % 1;  
    return {fill: "hsb(" + num + ", 0.75, 1)"};  
};  
c.animate({hue: 1}, ...);
```

Interfacing with WebIOPi

- Can use animations to produce display changes based on sensor output
 - Approach taken in demo1 project
 - `wget http://www.cs.unca.edu/~bruce/Fall14/demo1.tar.gz`
 - Can use `setInterval()` method to trigger the animation
 - Can use `onclick()` to drive the animation during development
- **Strongly** recommend doing development in Firebug and using console output to help debug

Example Development

- Download a tar file containing an orb development:

```
wget http://www.cs.unca.edu/~bruce/Fall14/jsTest.tar.gz
```

- Download the WebIOPi implementation:

```
wget http://www.cs.unca.edu/~bruce/Fall14/orbDemo.tar.gz
```

In-Class Exercise

- Modify `orbTest.html` (included in the earlier tar file) to use a path animation. Show me your completed work before leaving to receive credit.