COMP 4021
Internet Computing

Dynamic SVG

Adapted from slides prepared by David Rossiter

Approaches to Dynamic SVG

- SVG can be dynamically changed *while* it is being displayed

- There are two different approaches:
  1) Use SVG commands to make changes:
     - There are SVG commands to make changes (transformations)
     - There are SVG commands to animate changes
     - Works in Chrome, Safari, etc, and also IE
     - Older versions of IE and Firefox may not support SVG animation
  2) Use JavaScript to make change to DOM (SVG is just part of the DOM)
     - Should work in all browsers
     - To be discussed in later presentation

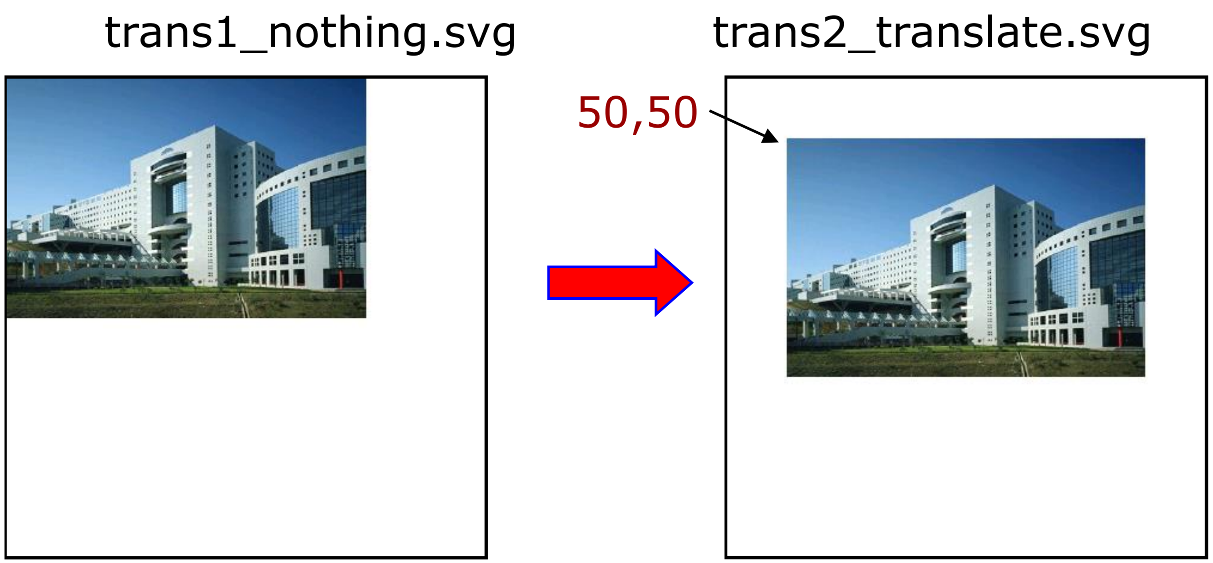
Transformations (without JavaScript)

- All SVG graphic elements have a "transform" attribute to make changes to the graphic elements
- The transformation commands available are
  - translate
  - rotate
  - scale
  - matrix - can be used to do all of the above operations, individually or all at the same time

Translate

- **translate(<tx> [<ty>])** will move the element <tx> units along the x-axis and <ty> units along the y-axis.

```
<image xlink:href="ust.jpg" transform="translate(50,50)"
    x="0" y="0" width="300" height="200"/>
```



Scale

- **scale(<sx> [<sy>])** will scale the element by multiplying <sx> and <sy> to the x and y coordinates
  - If <sy> is not given, it is assumed to be the same as <sx>
  - <sx> or <xy> is 0 it means the corresponding dimension has no change in scale
  - Scaling is relative to the origin (0,0)

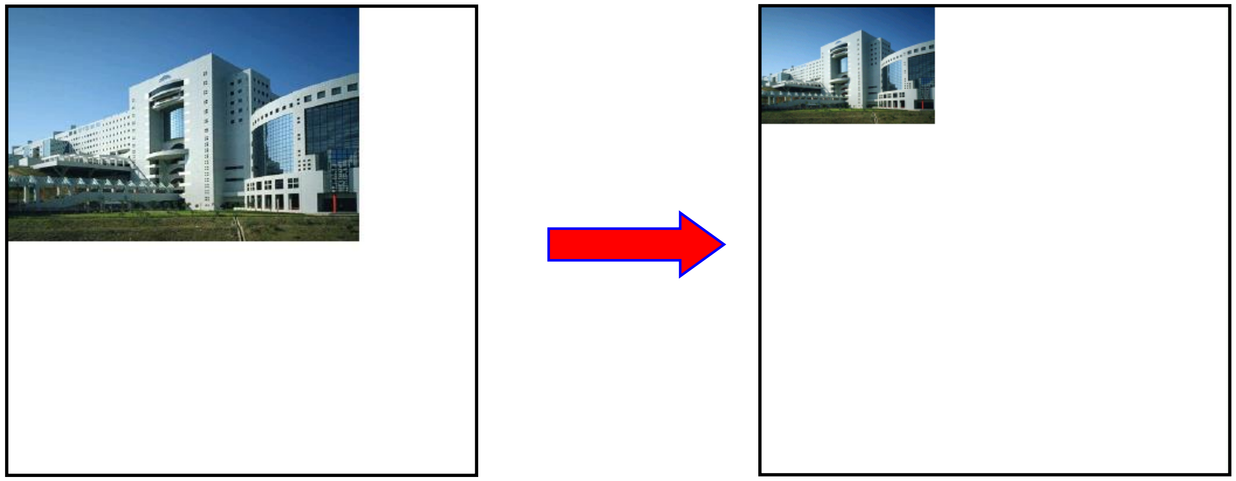
Scale

- Shrink the image to one half of its original size

```
<image xlink:href="ust.jpg" transform="scale(0.5 0.5)"
   x="0" y="0" width="300" height="200"/>
```

Demo – trans1_nothing.svg

Demo – trans3_scale.svg



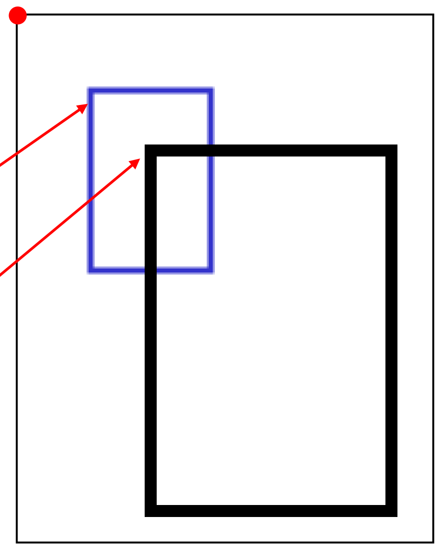
Scale (Cont.)

- Scaling is relative to the origin (0,0)

```
<rect x="10" y="10" width="20" height="30"
    transform="scale(2)" />
```

Original rect at 10,10

Transformed rect at 20,20



- To scale using a different center point, translate the element so that the center point becomes (0,0), perform scale, then translate the element back to its original location

Scale: Think about This

- Scaling an object around the center:
  - Translate center to 0,0
  - Scale
  - Translate back to original center (hard-code center in translate command)

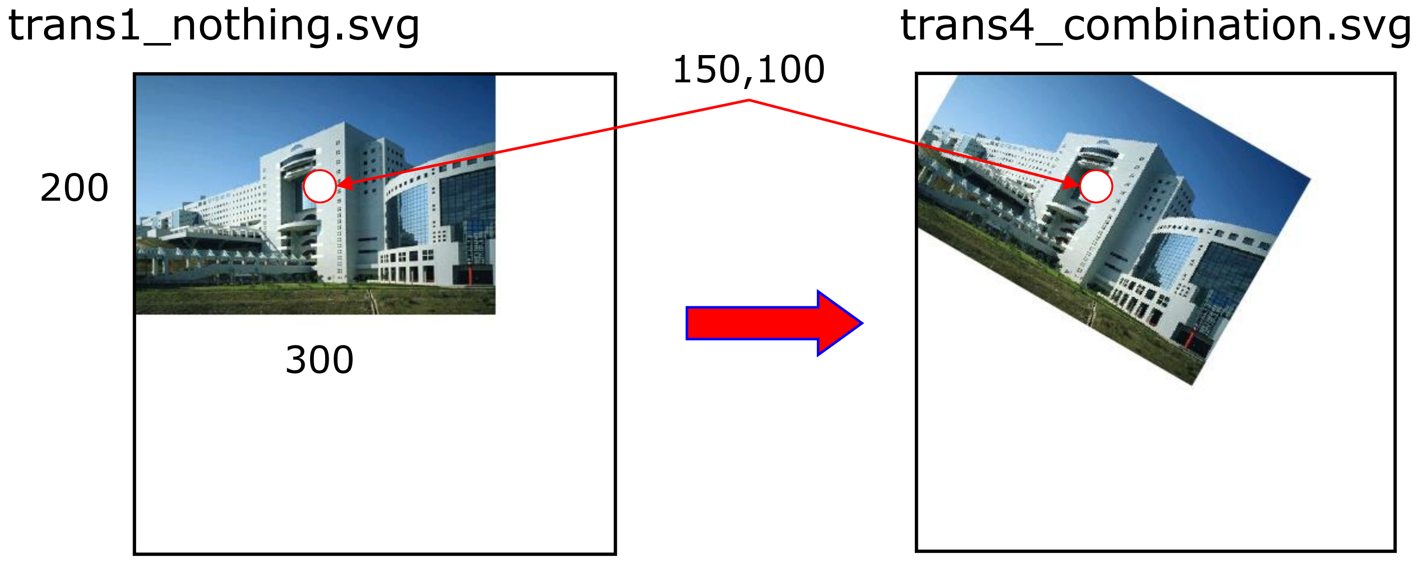
- With JavaScript:
  - Save original center in variables
  - Scale
  - Translate new center to original center

Rotate

- **rotate(<angle>, centre x, centre y)** rotates the element <angle> degrees around the point (centre x, centre y)

```
<image xlink:href="ust.jpg" transform="rotate(30,150,100)"
    x="0" y="0" width="300" height="200"/>
```

Rotate around the center of the photo



Rotate (Cont.)

- If rotation center is not given, assume the center is 0,0
- The following code has the same effect:

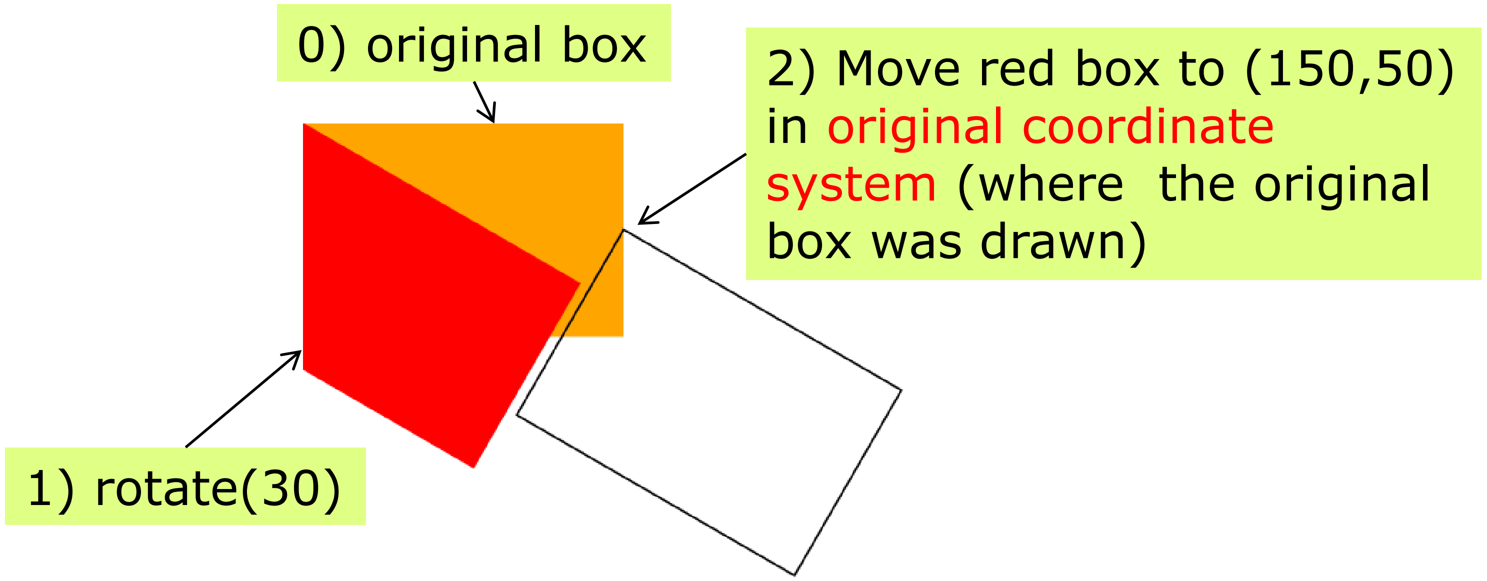
```
<image xlink:href="ust.jpg" transform="
    translate(150 100) rotate(30) translate(-150 -100) "
    x="0" y="0" width="300" height="200"/>
```

Multiple Operations in Transform (1)

- Rotate a picture then translate it

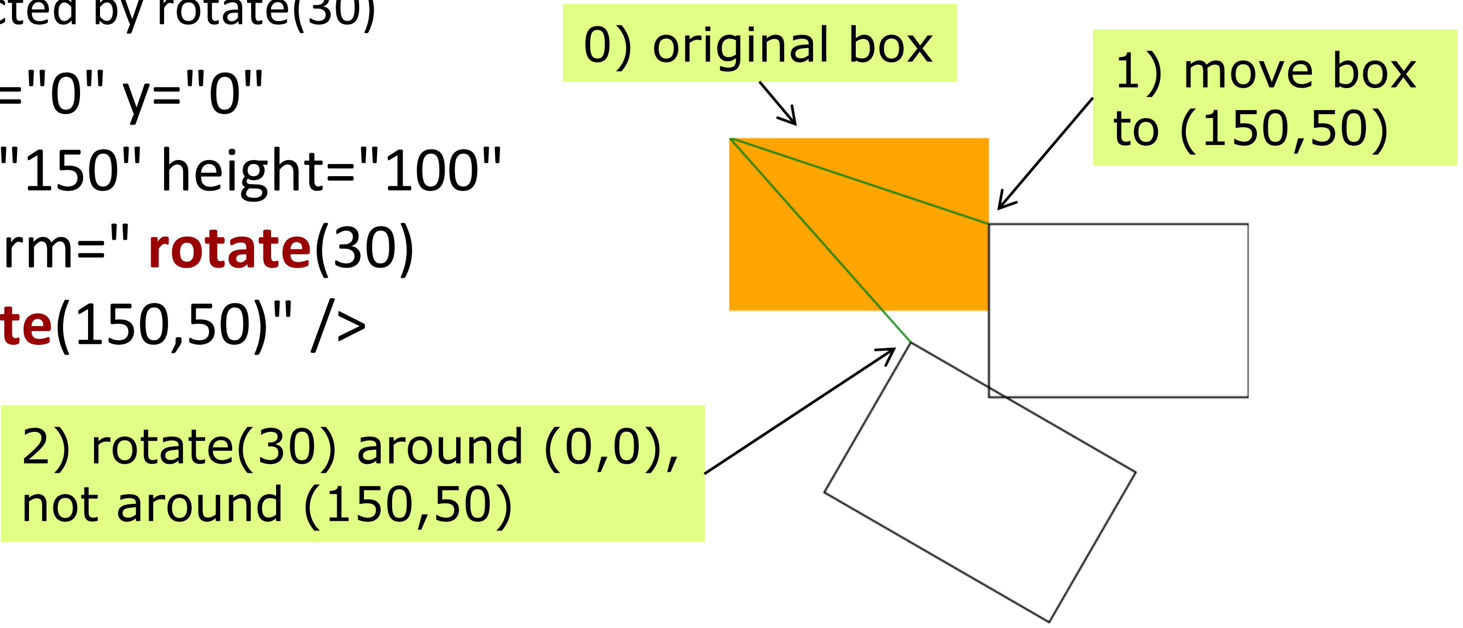
<rect x="0" y="0" width="150" height="100" transform="**translate**(150,50) **rotate**(30)" />

- Transform operations are performed from right to left, i.e., first perform rotate(30) then translate(150,50)


0) original box
2) Move red box to (150,50) in original coordinate system (where the original box was drawn)
1) rotate(30)

Multiple Operations in Transform (2)

- All transform operations perform in the original coordinate system; an operation does not change the coordinate system of subsequent operations
  - In previous example, coordinate system of translate(150,50) is not affected by rotate(30)
- <rect x="0" y="0" width="150" height="100" transform=" **rotate**(30) **translate**(150,50)" />



Codes for the Previous Examples

```
<svg width="300" height="400">
<rect x="0" y="0" width="150"
height="100" style="fill:orange"/>
<rect x="0" y="0" width="150"
height="100" style="fill:red"
transform="rotate(30)"/>
<rect x="0" y="0" width="150"
height="100"
style="fill:none;stroke:black"
transform="translate(150,50)
rotate(30)"/>
</svg>
```

```
<svg width="300" height="400">
<rect x="0" y="0" width="150" height="100"
style="fill:orange"/>
<rect x="0" y="0" width="150"
height="100“ style="fill:none;stroke:black"
transform="translate(150,50)"/>
<rect x="0" y="0" width="150" height="100"
style="fill:none;stroke:black"
transform="rotate(30) translate(150,50)"/>

<line x1=0 y1=0 x2=150 y2=50
style="stroke:green"/>
<line x1=0 y1=0 x2=150 y2=50
style="stroke:green"
transform="rotate(30)">
</svg>
```

Animation in SVG

Animation (Without JavaScript)

- So far we have looked at SVG commands to change an SVG element (once)
- But how can we continually apply a change over time, to get some kind of animation effect?
- SVG has commands for this also, called animate/ animateColor/ animateMotion/ animateTransform

SVG Animation Commands

- animate - for animating any attribute
- animateColor - for animating color attributes only
  - The animate command can also do exactly the same thing that animateColor can do
- animateMotion - for animating any object in a motion path
- animateTransform - for animating any object by changing any transformation (I.e. animating translation/ scale/ rotation/ matrix parameters)

animate

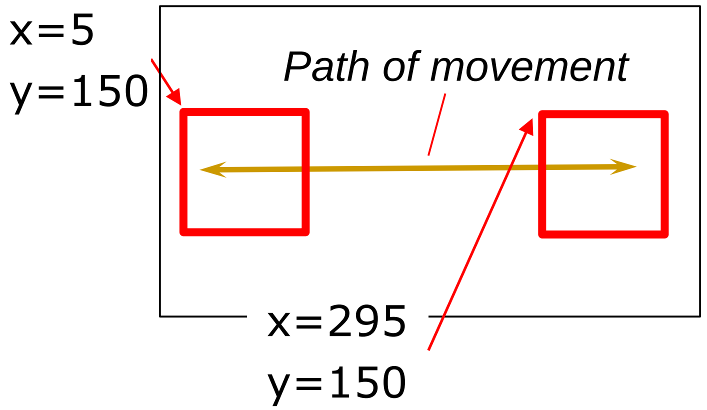
Anim01_animate.svg

```
<rect x="5" y="150" width="100" height="100" style="fill:none;
    stroke:red; stroke-width:5" >

        <animate attributeName="x" attributeType="XML"
            dur="5s" values="5; 295; 5"
            repeatCount="indefinite"/>

</rect>
```

From x=5 to x=295 inc 5



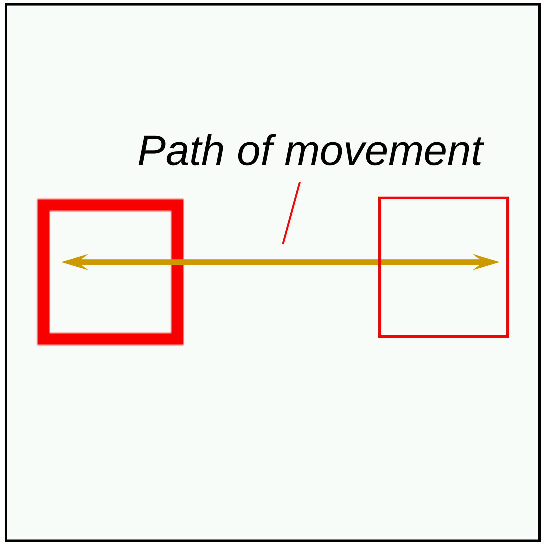
- The x position is changed over a period of five seconds, from x=5 to x=295, and then back to x=5
- Values are interpolated between the three key values: 5, 295, 5

Animate Two Parameters

Anim02_animate.svg

```
<rect x="5" y="150" width="100" height="100" style="fill:none;
    stroke:red; stroke-width:5" >

    <animate attributeName="x" attributeType="XML"
          dur="5s" values="5; 295; 5“
          repeatCount="indefinite"/>
    <animate attributeName="stroke-width"
          attributeType="CSS" dur="5s"
          values="10; 1; 10"
          repeatCount="indefinite"/>
</rect>
```



colorAnimate

Anim03_color.svg

```
<rect x="5" y="150" width="100" height="100"
   style="fill:none;stroke:red;stroke-width:5" >

  <animateColor attributeName="fill"
     attributeType="CSS" from="rgb(255,255,255)"
     to="rgb(255,0,0)" begin="0s" dur="5s"
     fill="freeze" />
</rect>
```

CSS: “fill” is a parameter of style attribute

- The fill colour is interpolated from white (255,255,255) to red (255,0,0) over five seconds

colorAnimate

Anim04_color.svg

```
<rect x="5" y="150" width="100" height="100"
    style="fill:none;stroke:red;stroke-width:5" >

  <animateColor attributeName="fill"
      attributeType="CSS"
    values="red;orange;yellow;green;blue;indigo;violet"
      begin="0s" dur="8s" repeatCount="indefinite"/>
</rect>
```

Colour is interpolated between these 7 key values

- The fill colour shows all the colours of the rainbow, in a cycle lasting 8 seconds

attributeType

- Each node can have a variety of attributes
- Some are from style sheet parameters; there are others such as those added by the programmer (these are called XML attributes)
- So the *attributeType* can be one of
  - "CSS" (if the attribute being controlled is a CSS property)
  - "XML" (if the attribute being controlled is an XML property)
  - or "auto" (this is the best value if you're not sure – the browser will search through all the attributes and use the right one)

animateMotion

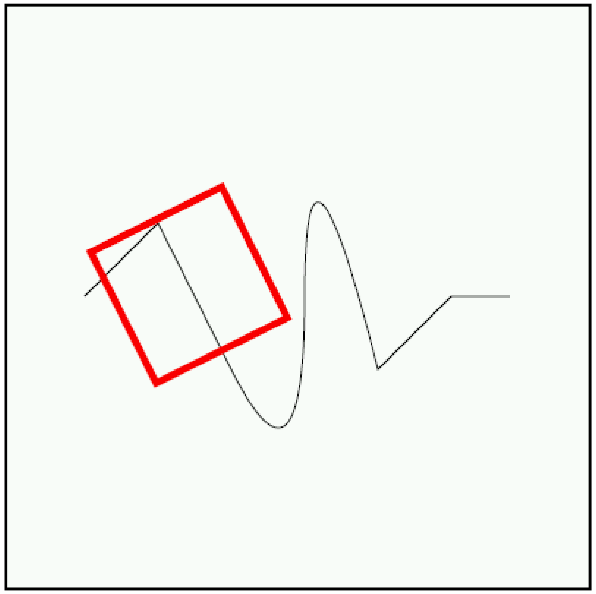
Anim05_motion.svg

- SVG elements can be animated along a path specified by path data in the <animateMotion> element

```
<rect x="-50" y="-50" width="100" height="100"
    style="fill:none;stroke:red;stroke-width:5" >

    <animateMotion
        path="M55,200 l50,-50 t50,100
        t50,-50 t50,50 l50,-50 L345,200"
        dur="3s" fill="freeze" rotate="auto"/>
</rect>
```

Draw quadratic Bezier to the specific points with implicit control points



Recall:
Small letter: relative
Capital letter: absolute
t/T: smooth quadratic
q/Q: quadratic Bezier

animateTransform

Anim06_transform.svg

- animateTransform is for animating translation/ rotation/ scaling

```
<g transform="translate(200, 200)">
<rect x="-50" y="-50" width="100" height="100"
    style="fill:none;stroke:red;stroke-width:10">
    <animateTransform type="scale"
    attributeName="transform" attributeType="XML" dur="5s"
    values="1;2;1" repeatCount="indefinite"/>
</rect>  </g>
```

- The rectangle is made larger and smaller in a 5 sec period

animateTransform

Anim07_transform.svg

```
<g transform="translate(200, 200)">
<rect x="-50" y="-50" width="100" height="100"
    style="fill:none;stroke:red;stroke-width:10">
    <animateTransform type="rotate"
    attributeName="transform" attributeType="XML" dur="5s"
    from="0" to="360" repeatCount="indefinite"/>
</rect>   </g>
```

- The rectangle is constantly rotated

Take Home Message

- SVG does not just display simple graphics; it can transform and animate graphics
- All of these are done with a markup language, without complex programming
  - The idea of markup languages is that non-programmers can do what they want without programmers’ help
- JavaScript is not an essential requirement although it can further enhance interactivity