A Mini Momentum Dojo:

Animating Scalar Vector Graphics

Duart Breedt

# Introduction

I don’t know much about SVG’s. I learnt about them at University and thought nothing more of them. Some fancy math, good for fonts, that’s what I knew.

I am sure some of you saw this project on CodePen a few months ago.

*Enter Yeti*

This small project fascinated me. I wondered how he managed to do that? I wanted to do that. So I downloaded the project, refactored, and played around a bit with it.

This tinkering gave me the idea to show others who may also want to know how to make awesome shit how to make awesome shit.

So here is some awesome shit.

# What are SVG’s?

Go find out: <https://en.wikipedia.org/wiki/Scalable_Vector_Graphics>

# Can’t I use animated DOM elements instead?

Yes. However, the animations that can be applied to them are limited to basic transformations such as rotations, scaling, skewing, and translations.

Furthermore, animating DOM elements generally has some sort of underlying functional reasoning such as hiding or showing an input field. Whereas animating SVG’s should be purely a design choice such as loading bars, hover effects, and visual representations of state changes.

# Do I need a library?

For this MOJO we will be using a library called KUTE.js

However, most of the functionality that SVG animation libraries offer are already built into modern browsers. Libraries are useful for making the animations easier to read and more compatible with older browsers.

# Are SVG animation used?

They are everywhere. The reason we don’t know much about them or we have this perception that they have become irrelevant is because they are old news. SVG’s are not revolutionary.

So there seems to be a disconnect between the lack of SVG knowledge in our design and animation toolbox and the fact they are used basically everywhere in this fashion.

## Three types of animations:

Which transformation type is a design choice. Generally, a particular effect can be achieved a multitude of ways.

### Transformations

These include translations, rotations, scaling, and skewing. Or any combination of those.

It should be noted that transformations can also be applied to DOM elements.

Examples:

<https://tympanus.net/Development/IconHoverEffects/#set-6>

<https://tympanus.net/Development/AnimatedSVGIcons/>

### Drawing

This involves the stroke of an SVG image to gradually be traced overtime as if it were to be drawn.

This functionality is the most attractive feature of SVG’s. However, it is achieved in a rather hacky way.

*Enter hacky illustration*

There isn’t an efficient way of mimicking this with DOM elements.

Examples:

<http://www.baeldung.com>

<https://tympanus.net/Development/AnimatedCheckboxes/>

<https://tympanus.net/Development/LetterInteractions/weaver.html>

<https://tympanus.net/Development/LetterInteractions/redraw.html>

<https://tympanus.net/Development/AnimatedLetters/>

### Morphing

To morph an SVG you need a second SVG which it will morph to. Overtime the first SVG transforms into the second SVG.

Careful planning is important for morphing to be effective.

Once more, there isn’t an efficient way of mimicking this with DOM elements.

Examples:

The play button | <http://www.youtube.com>

<https://tympanus.net/Development/OrganicShapeAnimations/>

<https://tympanus.net/Development/ShapeMorphIdeas/index5.html>