



Animating SVG Gradients



SVG • TUTORIALS | JONI TRYTHALL • JUNE 04, 2014 • 2 COMMENTS

Share

Share

Tweet

Share

Pin It

Share

SVG gradients are so handy. We can fill complex shapes and create depth and character for our artwork all while having access to it in the DOM.

Within SVG linear and radial gradient elements there are several attribute options available for thorough customization. Animating these gradients takes this customization even further and presents a unique opportunity that can be used to further communicate something to our users.

This article will cover the **basics of SVG gradients** and getting started with `<animate>`, and then dive into some demos for further understanding.



We use cookies to ensure that we give you the best experience on our website.

I disagree

I agree



Browser Support

Basic SVG support is present across desktop and mobile browsers, going at least two versions back across the board for desktop.

Concerning SVG SMIL animation (animating within the SVG element itself) specifically there is **no** support in IE, IE mobile, or Opera Mini, but strong support in current and at least one version back in Chrome, Firefox, Safari, Opera, and on mobile.

SVG Gradients

Before we get started with animation, it's important to have a basic understanding of how SVG gradients work.

Gradients are one of many paint server options we have available to use when we wish to add color to the fills and strokes of SVG. The idea here is that we can define gradients within our SVG but they have no visual output until we call on them with the use of the “fill” and/or “stroke” attributes.

Impress your audience with animated websites and web presentations.

With [Slides](#), we don't make you start from an empty slate. All you have to do is to pick the elements you like best and combine them. Each slide has been carefully crafted to satisfy three key criteria: aesthetic, function and usability. That way you know every element works together seamlessly while enhancing the impact of your content.



The basic structure of a very simple linear SVG gradient looks something like this:

```

1  <svg version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xl
2  width="180px" height="337px" viewBox="0 0 180.36 336.86" enabl
3  <path fill="#7A5034" stroke="7A5034" stroke-width="2.5448" s
4  M159.337,292.35c0.863,0.605,1.073,1.795,0.469,2.658l-20.974,
5  c-0.863-0.605-1.073-1.795-0.469-2.658l20.974-29.954c0.605-0.
6  L25.16,236.893 M149.278,309.079l-77.039-53.943 M73.154,244.3
7  <path fill="#7A5034" stroke="#7A5034" stroke-width="2.5448"
8  M44.187,325.431c-0.863,0.605-2.054,0.395-2.658-0.469l-20.974
9  c0.863-0.605,2.054-0.395,2.658,0.469l20.974,29.954c0.605,0.8
10 l75.509-52.872 M31.907,310.257l77.039-53.943 M118.788,260.87
11 <path class="outer-flame" fill="url(#largeGradient)" stroke=
12 M82.579,210.049c-4.315-0.393-8.678-1.158-12.676-2.352c-19.55
13 c-1.102-5.301-1.471-11.261-0.887-16.656c1.646-15.209,8.889-3
14 c2.079-5.647,3.825-11.421,5.117-17.299c2.019-9.181,3.897-21.
15 c4.029,6.492,7.841,13.117,11.454,19.849c3.744,6.976,7.259,14
16 c4.66,19.078,9.751,43.266-2.968,60.391c-5.683,7.652-12.951,1
17 C86.931,210.352,84.761,210.248,82.579,210.049z">
18 </path>
19 <path class="inner-flame" fill="url(#smallGradient)" stroke=
20 M90.043,116.799c-4.024,9.712-12.607,17.546-17.863,26.662c-5.
21 1.865,8.974,9.366,16.589,18.078,19.191c1.782,0.532,3.726,0.8
22 c5.104,0.041,10.352-1.405,14.945-3.579c3.849-1.822,7.087-4.6
23 c-6.142-14.841-15.085-28.459-22.297-42.793C91.464,111.781,91
24 </path>
25 </svg>

```



The gradient details are written within the `<defs>` element, but would have no output within our work until we call on it using its unique ID. At this point we would also add the shape within the same `<svg>` element, but outside of the `<defs>` element.

The `<stop>` nodes represent the color selections and their locations on our mapped gradient, which we will discuss further in the following section.



to specify the colors and coordinates of our gradients.

Here is a quick look at the attributes that will be used in the demos to follow:

ID

A gradient requires a unique ID in order for it to be called on through “fill” and “stroke” attributes.

x1, x2, y1, y2

The values within the x1, x2, y1, y2 attributes specify the start and end coordinates for linear gradients along the appropriate axis. These values default to “0” if left unspecified, except for x2 which defaults to “100%”.

cx, cy, r, fx, fy

Radial gradient attributes are similar to those of linear, except our coordinates will be handled much differently. Instead of being based on a straight line, these gradients will be mapped circularly.

cx, cy, and r (radius) define the outermost circle for the gradient, with this outer perimeter being the 100% stop point. fx and fy define the focal point of the gradient, with the 0% stop mapping to these values.

<stop>

Within the gradient element itself we include the <stop> nodes. The attribute values within <stop> nodes define the colors for our gradient, where they should be placed, and any opacity that we want applied.

The offset property indicates where the gradient stop is located. For linear gradients this location will be a distance along the gradient line, for radial gradients it will be a distance from the focal point to the edge of the mapped circle.



few stops, or create stripes with several closely mapped stops.

<animate>

<animate> element will allow us to select gradient attributes and properties and then assign different values to them over a specified duration of time. In short, we can move them! The <animate> element will reside within the element we wish to animate. Here is the structure for a basic animation on a rectangle:

```
1  <defs>
2    <radialGradient id="smallGradient" fy="90%">
3      <stop offset="0%" stop-color="#f4c708"></stop>
4      <stop offset="100%" stop-color="#f7e69a"></stop>
5      <animate attributeName="fy" dur="700ms" from="90%" to="0%">
6    </radialGradient>
7    <radialGradient id="largeGradient" fy="90%">
8      <stop offset="0%" stop-color="#ff8806"></stop>
9      <stop offset="100%" stop-color="#d9574a"></stop>
10     <animate attributeName="fy" dur="700ms" from="90%" to="0%">
11   </radialGradient>
12 </defs>
```

The animation will move the rectangle along the x axis 400px pixels from its starting point. The animation will complete within 5 seconds, and repeat indefinitely.

Attributes

Within the <animate> element we will use attributes to select our target and specify the details of the animation. Here is a look at the specific attributes we will focus on in the demos:

attributeName

attributeName defines the name of the target attribute to be animated. In taking another look at the animation code above, the **attributeName** value is "x" so that



The “to” and “from” attributes indicate the initial and final value of the targeted attribute.

values

Within the `<values>` attribute we can create a list of values separated by semicolons, and the animation will apply these values in order over the duration of the action.

By including this attribute any “from” or “to” values within the same `<animate>` element will be ignored, meaning that it takes the place of these values once included so there is no need for them.

dur

The duration defines within what time span the animation should complete.

fill

While “fill” often refers to the interior color of SVG, it has a different meaning in the context of an `<animate>` attribute. We can set a value of “freeze” within this attribute so that the animation freezes once it is complete and does not restart.

repeatCount

The repeatCount attribute specifies the number of times an animation should take place. This value can be either a number, or “indefinite”.

Animating Gradients

To target gradients specifically we need to have an idea of what properties we would like to animate, like the **stop-color**, **offset**, and/or the specific coordinates.

When we reviewed the code for the moving rectangle animation above the `<animate>` element resided within the shape’s element, the `<rect>`. In order to



example, we position `<animate>` within the gradient element, but outside of the `<stop>` elements.

Here's a look at the structure for an animation on the **stop-color** of a `<stop>` node.

```
1 | <stop offset="0%" stop-color="#f4c708"></stop>
2 | <stop offset="100%" stop-color="#f7e69a"></stop>
3 | <animate attributeName="fy" dur="700ms" from="90%" to="0%" repe
```

Demos

To demonstrate the possibilities of animating SVG gradients we will look at some potential effects for a fictional camping site. The demos consist of animated gradients on fire, trees, and a skyline.

Demo 1: Fire



Within the fire SVG we are targeting two flames to fill and animate, and as mentioned all the details are listed within a `<defs>` element in the `<svg>`.

```
1 | <defs>
2 |   <radialGradient id="smallGradient" fy="90%">
3 |     <stop offset="0%" stop-color="#f4c708"></stop>
```



```

8      <stop offset= 0% stop-color= #ff8800 ></stop>
9      <stop offset="100%" stop-color="#d9574a"></stop>
10     <animate attributeName="fy" dur="700ms" from="90%" to="0"
11     </radialGradient>
12 </defs>

```

There are two radial gradients, “smallGradient” and “largeGradient”. The same animation is applied to each gradient targeting the fy coordinate point and moving it from 90% to 0% within 700ms. The animation is set to take place indefinitely.

The assignment of these gradients is done by adding the gradient ID to the “fill” of the shape via a URL, for example:

```

1 | fill=&quot;url(#largeGradient)&quot;;

```

Demo 2: Trees



For the trees demo we will be adding the animation to the gradient **stop-color**. There are two gradients to represent different colors for the trees, one with dark shades of green and one with light. Each animation is essentially the same, the only difference being the colors used.

```

1 <defs>
2   <linearGradient id="darkGradient" x1="30%" y1="70%">
3     <stop offset="0%" stop-color="#0b2d13">
4       <animate attributeName="stop-color" values="#0b2d13; #79c98c"
5     </stop>
6     <stop offset="100%" stop-color="#79c98c">
7       <animate attributeName="stop-color" values="#79c98c; #0b2d13"

```




```

12     <stop offset="0%" stop-color="#054f16" >
13         <animate attributeName="stop-color" values="#054f16; #91bc9c"
14     </stop>
15     <stop offset="100%" stop-color="#91bc9c">
16         <animate attributeName="stop-color" values="#91bc9c; #054f16"
17     </stop>
18 </linearGradient>
19 </defs>

```

The `<animate>` element resides within the 100% stop point for each gradient and targets the **stop-color**. The color values are animated through the **values** attribute, over a period of 3 seconds, and the animation is set to stop, or “freeze” once it completes one cycle.

Demo 3: Sky



The concept of a sunset is an ideal example of when animating a stop’s offset would be needed. There are two gradients here, each having a series of animated color changes. With an additional animation on the **offset** of the 100% stop point the point moves down (to 0) and then back up again.

```

1  <defs>
2    <linearGradient id="skyGradient" x1="100%" y1="100%">
3      <stop offset="0%" stop-color="lightblue" stop-opacity=".5">
4        <animate attributeName="stop-color" values="lightblue;blue"
5      </stop>
6      <stop offset="100%" stop-color="lightblue" stop-opacity=".5">
7        <animate attributeName="stop-color" values="lightblue;orange"
8        <animate attributeName="offset" values=".95;.80;.60;.40;"
9      </stop>

```



through within 14s. The additional **offset** animation changes the location of the 100% stop point through a series of numerical values, also within 14s.

It is important to mention here that while you should technically be able to use percentages when animating **offset**, and can do so without problems in Firefox, Chrome and Safari will not render the animation on the **offset** when percentage values are used within `<animate>`. The work around here is to simply avoid using percentages, as shown in the code above.

Conclusion

For this article, we discussed the basic structure of SVG gradients, how to utilize SVG SMIL animation once they are created, and reviewed some demos regarding how this can be handy in communicating to users.

Hopefully, all this inspires you to take a look at writing gradients within SVG code and get them moving all within the same SVG element.

Like what you're reading? Subscribe to our top stories.



☐ I agree to storage of my email according to Privacy Policy.

Share post



Share



Tweet



Share




Joni Trythall

Joni Trythall learns web design by day, and forgets it all by night. She is constantly trying to combine her love of learning code with her long time obsession of

[SHOW 2 COMMENTS](#)

Related Articles

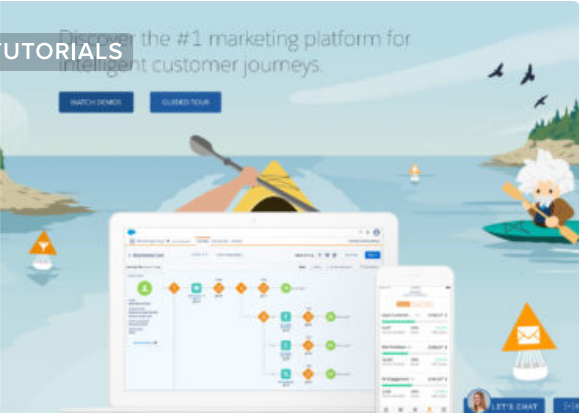
TUTORIALS



ANDRIAN VALEANU

How to Import Custom HTML Email Template from Postcards to Intercom (YouTube Tutorial)

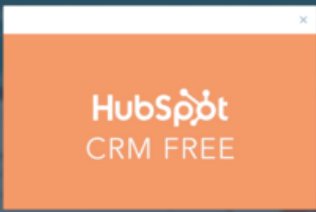
TUTORIALS



ANDRIAN VALEANU

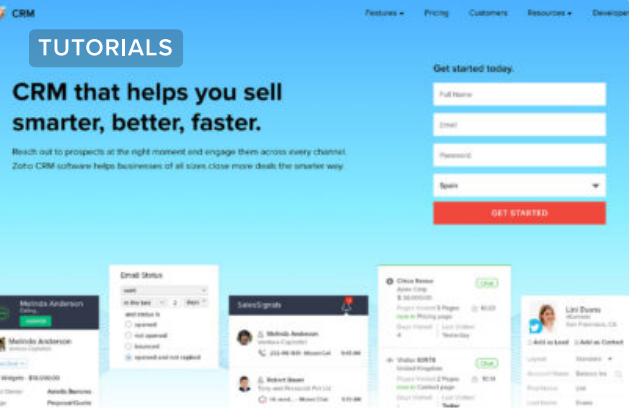
How to Import a Custom HTML Email Template from Postcards to Salesforce [YouTube Tutorial]

TUTORIALS

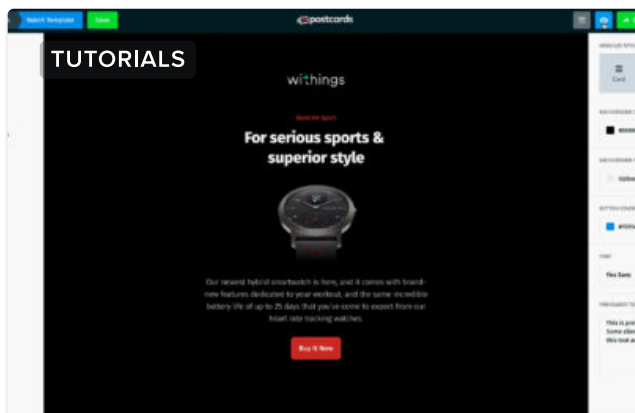


ANDRIAN VALEANU

TUTORIALS



ANDRIAN VALEANU



ANDRIAN VALEANU

How to Create and Customize an Email Newsletter Template [YouTube Tutorial]



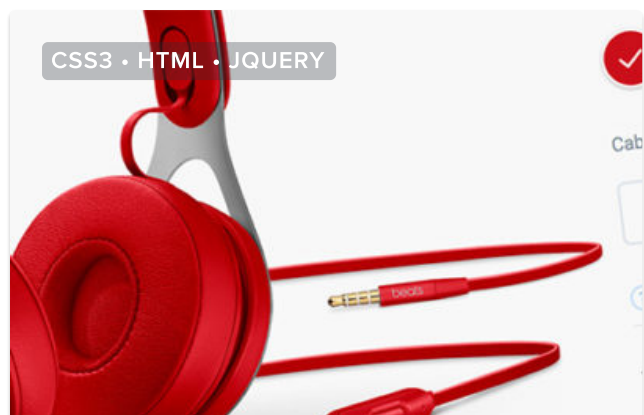
ANDRIAN VALEANU

How to Create a Minimalist Website Design [YouTube Tutorial]



CALEB KINGSTON

6 Creative Ways to Use Repeat Grids in Adobe XD



RAUL DRONCA

Create a Product Page with Interactive Colors in HTML, CSS3 & jQuery



We build high-quality products

Designmodo offers advanced drag and drop website and email builders for web designers and developers, we have everything you need to make money.

[Estimate Revenue](#)[Become an Affiliate >](#)

designmodo

Copyright © 2010-2018.
All Rights Reserved.

ARTICLES

- News
- Coding
- Design
- Inspiration
- Freebies
- Tutorials
- WordPress

PRODUCTS

- Postcards new
- Slides Framework
- Startup Framework
- Qards WordPress
- Job Board
- Affiliate Program
- Banners

MARKET

- Submit Product
- Free Goods
- Mockups
- UI Kits
- Icons
- Fonts
- Tools

COMPANY

[About Us](#)

FOLLOW US

[Twitter](#)





- [Privacy Policy](#)
- [Terms and Conditions](#)
- [Cookies Policy](#)
- [Merch](#)
- [Google+](#)
- [Youtube](#)
- [Medium](#)
- [Dribbble](#)

