The Wonderful World of SVG
There are only a handful of **basic shapes**.
There are only a handful of basic shapes. Technically not a basic shape. Basic Shapes are shortcuts to paths.

- `<line>`
- `<rect>`
- `<circle>`
- `<ellipse>`
- `<polygon>`
- `<polyline>`
- `<path>`
Let’s spend like 15 seconds looking at SVG syntax.

Even though most of us will hardly ever touch it directly because there are great tools for it.
<rect x="10" y="10" width="500" height="500"
    fill="yellow" stroke="blue" stroke-width="5" />
<rect x="10" y="10" width="500" height="500"
     fill="yellow" stroke="blue" stroke-width="5" />
<rect x="10" y="10" width="500" height="500"
   fill="yellow" stroke="blue" stroke-width="5" />

<rect x="10" y="10" width="500" height="500"
    fill="yellow" stroke="blue" stroke-width="5" />
<rect x="10" y="10" width="500" height="500"
fill="yellow" stroke="blue" stroke-width="5" />
<circle cx="50" cy="20" r="15" fill="yellow" stroke="blue" stroke-width="5" />
<circle cx="50" cy="20" r="15"
fill="yellow" stroke="blue" stroke-width="5" />
<polygon points="32.8,72.3 54.2,65.1 78.5,103.4..." fill="yellow" stroke="blue" stroke-width="5" />
“Beaker” by Ben Joyce :: codepen.io/benjoyce/full/myqxad/
<linearGradient id="gradient" gradientUnits="userSpaceOnUse"
  x1="0" y1="0" x2="0" y2="100%">
  <stop offset="0" style="stop-color: #88FF3F" />
  <stop offset="0.56" style="stop-color: #6BD819" />
  <stop offset="1" style="stop-color: #58BF00" />
</linearGradient>
<linearGradient id="gradient" gradientUnits="userSpaceOnUse"
  x1="0" y1="0" x2="0" y2="100%">
  <stop offset="0" style="stop-color: #88FF3F" />
  <stop offset="0.56" style="stop-color: #6BD819" />
  <stop offset="1" style="stop-color: #58BF00" />
</linearGradient>

<ellipse fill="url(#gradient)"
  cx="150" cy="50" rx="150" ry="50" />
<linearGradient id="gradient" gradientUnits="userSpaceOnUse"
    x1="0" y1="0" x2="0" y2="100%">
    <stop offset="0" style="stop-color: #88FF3F" />
    <stop offset="0.56" style="stop-color: #6BD819"/>
    <stop offset="1" style="stop-color: #58BF00" />
</linearGradient>

<ellipse fill="url(#gradient)"
    cx="150" cy="50" rx="150" ry="50" />

"Gradient Reuse in SVG" by Me :: codepen.io/chriscoyier/pen/ByOWYO/
<linearGradient id="gradient" gradientUnits="userSpaceOnUse"
x1="0" y1="0" x2="0" y2="100%">
  <stop offset="0" style="stop-color: #88FF3F" />
  <stop offset="0.56" style="stop-color: #6BD819" />
  <stop offset="1" style="stop-color: #58BF00" />
</linearGradient>

<ellipse fill="url(#gradient)"
  cx="150" cy="50" rx="150" ry="50" />

<path fill="url(#gradient)"
  d="M129.9,68.6c0,18.8-42.2,9.4-42.2,..." />

"Gradient Reuse in SVG" by Me :: codepen.io/chriscoyier/pen/ByOWYO/
```xml
<linearGradient id="gradient" gradientUnits="userSpaceOnUse"
  x1="0" y1="0" x2="0" y2="100%">
  <stop offset="0" style="stop-color: #88FF3F" />
  <stop offset="0.56" style="stop-color: #6BD819" />
  <stop offset="1" style="stop-color: #58BF00" />
</linearGradient>

<ellipse fill="url(#gradient)"
  cx="150" cy="50" rx="150" ry="50" />

<path fill="url(#gradient)"
  d="M129.9,68.6c0,18.8-42.2,9.4-42.2,..." />
```

“Gradient Reuse in SVG” by Me :: codepen.io/chriscoyier/pen/ByOWYO/
“Nature’s Journey” demo file that comes with Adobe Illustrator

25 Years
LET THERE BE ART
But why would you even
USE SVG?

1 Resolution independent
2 Design possibilities
3 Use as a system
SCREEN SIZE & RESOLUTIONS
SVG was born in 1999
Why send pixel data when you can send GEOMETRY?

Math is more efficient!
Let your powerful computer* do the drawing.

*The connection between this idea and client-side MVC is interesting.
shoptalkshow.com/episodes/147-tom-dale/
SVG was conceived in 1999.

Database : Architect
::
JavaScript : Engineer
Database : Architect

::

JavaScript : Engineer

::

SVG : Archaeologist
“Nature’s Journey” demo file that comes with Adobe Illustrator
That one would be a little impractical on the web.

Because it’s like 30 MB.

One of the advantages of SVG is that for simple graphics, the file size is smaller and the quality is higher (best of both worlds).

But there is a complexity limit.
DEFINITELY!
PROBABLY.

(test the zipped file size)
NOPE!
MAYYYYYBBE.
TEST CONTEXT, GZIPPED SIZE, DISPLAY SIZE, ETC.
<div id="icon">
  <div id="controller">
    <div id="leftPad">
      <div id="arrowBG">
        <div id="arrow"></div>
      </div>
    </div>
    <div id="bodyPad"></div>
    <div id="rightPad">
      <div id="buttonBG">
        <div id="button1">
          <div class="circle"></div>
          <div class="circle"></div>
        </div>
        <div id="button2">
          <div class="circle"></div>
          <div class="circle"></div>
        </div>
      </div>
    </div>
  </div>
</div>
RICH PEOPLE STILL RICH PROBABLY STUDIES SHOW
Professional digital design for Mac.

Sketch gives you the power, flexibility and speed you always wanted in a lightweight and easy-to-use package. Finally you can focus on what you do best: Design.

Free Trial  Buy $99

Also available on the Mac App Store. Requires OS X 10.9+

Special prices are available for Educational licenses. You can also Extend an existing license with additional seats.
ABOUT
YWeb Career Academy (YWebCA) is a training program for women and people of color in technology. The goal is to prepare students for high-paying, family-sustaining jobs, particularly in the growing field of technology. The program provides instruction in front-end and web development skills, including HTML, CSS, JavaScript, and SQL, design, and project management.

HOW THE CLASSES WORK
Each cohort is 12 weeks long, with a mix of 40 hours of classroom instruction. Students attend a "flipped classroom" in which lessons are provided online or in other classes for classroom time. These labs provide hands-on experience and individual instruction from industry professionals. The curriculum includes a mix of code challenges, projects, and real-world scenarios, as well as peer and instructor feedback.

CURRENT COURSE

We are currently in Session 7, which runs from April to August. The course concludes with a hands-on presentation of the final project.

COURSE MATERIAL

All course materials are online and accessible to students. YWebCA will provide any necessary hardware or software, and students will be encouraged to bring their own laptops. Students are expected to have basic computer literacy and the ability to work independently.

Apply for Summer 16
How to Help
Pretty useful for `<img srcset>` and/or `<picture>`
<picture>
  <source type="image/svg+xml" srcset="Logo.svg"
  >
  <img srcset="Logo@1x.png, Logo@2x.png 2x"
  alt="Logo description"
  >
</picture>
If you can get your hands on the vector art...

you can easily get that into SVG for use on the web.
There are three* useful ways to use SVG on the web:

1. SVG as `<img>` in HTML
2. SVG as background-image in CSS
3. Inline SVG in HTML

*There are more ways, like `<object>`, `<embed>`, and `<iframe>` - but I don’t think they are very useful so let’s ignore them.
<img> is content.
.main-header {
    background-image: url(texture.svg);
}

CSS
Professor Mode

Allows you to demonstrate coding to you your students in real time. You type, they see it all happen live and can directly participate.
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Inline SVG Demo</title>
</head>
<body>
  
  <svg viewBox="0 0 1000 1000">
    
    <rect x="10" y="10" width="500" height="500"
    fill="yellow" stroke="blue" stroke-width="5" />
  
  </svg>

</body>
</html>
We have these three donuts right now.

**Glazed Donut**
Our classic recipe; since 1948.
$1.99  Add to Cart

**Sprinkle Donut**
You know, for the kids.
$2.40  Add to Cart

**Striped Donut**
Like an exotic zebra.
$2.92  Add to Cart


Open daily from 6am-11am
Donut be late get it.  123 Hole St.
Sugarsville, MD 12345
We have these three donuts right now.

- **Glazed Donut**: Our classic recipe since 1948. $1.99 | Add to Cart
- **Sprinkle Donut**: You know, for the kids. $2.40 | Add to Cart
- **Striped Donut**: Like an exacto knife. $2.92 | Add to Cart


Open daily from 6am-11pm. Next to the grill. 151 High St. Binghamton, NY 13906
We have these three donuts right now.

- **Glazed Donut**
  - Our global recipe since 1968
  - $3.50
  - Add to Cart

- **Sprinkle Donut**
  - You know, for the kids.
  - $2.40
  - Add to Cart

- **Striped Donut**
  - Like an actual stripe.
  - $2.90
  - Add to Cart


Open daily from 6am-11pm.

155 Hole St.
Downtown, MD 22316.
SVG-as-<img>

SVG as background-image
SVG-as-<img>
SVG as background-image
Inline SVG
SVG is pretty efficient already, but it can also be **heavily optimized** in multiple ways.

**COOL FACT**
SVG gzips very well, because it has a lot of repetitive strings. Be sure to enable that on your server for .svg and when testing to decide to use or not, test different formats against the gzipped sizes.
SVG *gzips* very well, because it has a lot of repetitive strings. Be sure to enable that on your server for `.svg` and when testing to decide to use or not, test different formats against the gzipped sizes.

**HTACCESS**

```
AddType image/svg+xml .svg .svgz

<ifmodule mod_deflate.c>
  AddOutputFilterByType DEFLATE text/html image/svg+xml
</ifmodule>
```
bork@kiwi:~/work/hackerschool/gzip.jl$ julia unzip.jl raven.txt.gz
bork@kiwi:/work/hackerschool/gzip.jl$ julia gunzip.jl raven.txt.gz
px2svg

Turning raster images into SVG files, one pixel at a time. Yes, really.

What?

The PHP accepts a raster image (GIF, PNG, JPEG, that sort of thing) and creates an SVG image that recreates the raster image. It does this by drawing filled rectangles to recreate the pixels in the image.
CRAZY EXPERIMENT

PNG
16 K

SVG
319 K

24 K

? K

SHOULDA BEEN AN SVG AM I RIGHT?
SVGO is great for optimizing .svg files.

This demo is SVGO-GUI :: github.com/svg/svgo-gui
The core library / command-line tools is here :: github.com/svg/svgo
A visual in-browser version by Jake Archibald is here :: jakearchibald.github.io/svgomg/
SVGO is great for optimizing .svg files.

This demo is SVGO-GUI :: github.com/svg/svgo-gui
The core library / command-line tools is here :: github.com/svg/svgo
A visual in-browser version by Jake Archibald is here :: jakearchibald.github.io/svgomg/
SVG makes for an excellent ICON SYSTEM

Icons are incredibly common on the web. Tons of sites make use of them because they are useful visual indicators. The style of them change over time, but the concept isn’t a trend.
Adding CSS-Tricks logo #226

chriscoyler wants to merge 1 commit into LeaVerou-pages from chriscoyler/gh-pages

chriscoyler commented an hour ago

by Lea's generous request at https://twitter.com/LeaVerou/status/4418696454303027201

Add more commits by pushing to the gh-pages branch on chriscoyler/prism.

Write
Preview

Comments are parsed with GitHub Flavored Markdown

Leave a comment

Attach images by dragging & dropping, selecting them, or pasting from the clipboard.

ProTip Add comments to specific lines under Files Changed.
Add New Post

ICONS EVERYWHERE

Publish

- Save Draft
- Draft
- Public
- N/A

Status: Draft Edit
Visibility: Public Edit
SEO: N/A Check
Publish immediately Edit
Publish immediately Edit
Purge from cache
Move to Trash

Publish
The classic problem:

Say a site needs 20 icons. You really don’t want to make 20 separate HTTP requests for those. That would be slow. One of the top ways to make sites faster is to make less requests.

An icon system does two things:

1. All icons are in one request.
2. It makes icons easy to use.
Past solution to this same problem: **CSS Sprites**
Past solution to this same problem: Icon Fonts
It’s looking like **HTTP/2** will make concatenating assets an anti-pattern.

Because. Uh. Reasons.

I think there is no penalty for requesting multiple assets from the same host and no extra cookie overhead. So if you leave all the icons separate, you can change a single icon without breaking the cache on all of them.
Let’s do this thing.

How to make an icon system from inline SVG
<body>

<svg display="none">
  <defs>
    <symbol viewBox="0 0 100 100" id="icon-1">
      <title>Follow us on Twitter</title>
      <path d="M80.893,40.234c-
    ...
    </symbol>
  </defs>
</svg>
To use...

Put this anywhere in the HTML.

<svg>
  <use xlink:href="#icon-1"></use>
</svg>
Give me a global var named "$"+class name

I do a lot of pens on CodePen and I tend to use the same pattern building a prototype as a pen.

For instance I use classnames as handles when inputting and outputting data to the test/prototype. For instance I would have an element with the classname `source` and one with `output`. So I always create vars named `$source` and one called `$output`.

Why not write a function to automatically add global vars named "$" + classname?

This gist is that function! It will...

let and const in ES6

Using let and const in ES6

I've started working on a new project in ES6. Reading through some articles, I was confused about let and const. After figuring it out and writing some demos I thought I'd...

Dynamic Typing in Javascript

Dynamic Typing

You don't tell the Javascript engine what type of data a variable holds, it figures it out while your code is running.

Variable can hold different types of values because it's all...
Give me a global var named "$"+class name

I do a lot of pens on CodePen and I tend to use the same pattern building a prototype as a pen.

For instance I use classnames as handles when inputting and outputting data to the test/prototype. For instance I would have an element with the class name source and one with output. So I always create var named $source and one called $output.

Why not write a function to automatically add global vars named "$" + class name?

This gist is that function! It will...

let and const in ES6

Using let and const in ES6

I've started working on a new project in ES6. Reading through some articles, I was confused about let and const. After figuring it out and writing some demos I thought I'd...

Dynamic Typing in Javascript

Dynamic Typing

You don't tell the Javascript engine what type of data a variable holds, it figures it out while your code is running.

Variable can hold different types of values because it's all...
Step 7: styling an icon group by Benedikte Vanderweeën :: codepen.io/Benedikte/pen/rAjad
"Step 7: styling an icon group" by Benedikte Vanderweeën :: codepen.io/Benedikte/pen/rAjad
Leveling up our icon system

1. Let’s make a build tool do the hard part
2. Let’s ajax for the SVG defs, so we can browser cache
3. Let’s add a fallback for non-supporting browsers
Let’s make a computer build the sprite for us.

We could do it ourselves, but that’s more work and error-prone.
IcoMoon is very awesome.
IcoMoon gave us this.

Using the icons is as easy as this.
The Flame Alphabet
Command line build systems are even more awesome.
How about a little graceful degradation and browser caching?
First, test to see if inline SVG is supported.

```javascript
var supportsSvg = function() {
    var div = document.createElement('div');
    div.innerHTML = '<svg/>';
    return (div.firstChild &&
            div.firstChild.namespaceURI) ==
            'http://www.w3.org/2000/svg';
};
```
if (supportsSvg()) {
    // Ajax for the defs.svg
} else {
    // We’re going to need a fallback
}
Ajax means we can browser cache the response.

```javascript
var ajax = new XMLHttpRequest();
ajax.open("GET", "defs.svg", true);
ajax.responseType = "document";
ajax.onload = function(e) {
  document.body.insertBefore(
    ajax.responseXML.documentElement,
    document.body.childNodes[0]
  );
}
ajax.send();
```
For the fallback, one option is to use **Grunticon**.

Grunticon is a whole system onto itself, so you can definitely just use it exactly as is. But Grunticon doesn’t start with inline SVG in the HTML like we are doing here.

We can still use it and do things our own way.

**Details!**

css-tricks.com/inline-svg-grunticon-fallback/
Class name that Grunticon automatically creates from the file name.

```html
<svg xmlns="http://www.w3.org/2000/svg" class="icon-home">
  <use xlink:href="#icon-home"></use>
</svg>
```
if (supportsSvg()) {
    // Ajax stuff here.
} else {
    grunticon(
        ['',
         '/fallbacks/icons.data.png.css',
         '/fallbacks/icons.fallback.css'],
    );
}

Don’t load anything in a “supported” scenario
Inline SVG with Grunticon Fallback

Icons by IcoMoon

- icon-home
- icon-pencil
- icon-image
- icon-music
- icon-heart
Should the day come that you don’t need a fallback anymore, just stop running Grunticon and doing the support test.
OK, SHEESEH.

Why do it this way? What are the advantages?
1. **Vector!**
   Typically sharper than icon fonts because of non-text anti-aliasing.
1. **Vector!**
   Typically sharper than icon fonts because of non-text anti-aliasing.

2. **Easy multi-color!**
   More CSS control than any other method.

---

css-tricks.com/icon-fonts-vs-svg/
1. **Vector!**
   Typically sharper than icon fonts because of non-text anti-aliasing.

2. **Easy multi-color!**
   More CSS control than any other method.

3. **Animate!**
   Easy to apply transitions and animations.

[css-tricks.com/icon-fonts-vs-svg/]
1. **Vector!**
   Typically sharper than icon fonts because of non-text anti-aliasing.

2. **Easy multi-color!**
   More CSS control than any other method.

3. **Animate!**
   Easy to apply transitions and animations.

4. **Script away!**
   Everything is in the DOM.
1. **Vector!**
   Typically sharper than icon fonts because of non-text anti-aliasing.

2. **Easy multi-color!**
   More CSS control than any other method.

3. **Animate!**
   Easy to apply transitions and animations.

4. **Script away!**
   Everything is in the DOM.

5. **Better accessibility! Plus fallbacks!**
   Fool-proof, once you set it up well.

[css-tricks.com/icon-fonts-vs-svg/](css-tricks.com/icon-fonts-vs-svg/)
1. **Vector!**
   Typically sharper than icon fonts because of non-text anti-aliasing.

2. **Easy multi-color!**
   More CSS control than any other method.

3. **Animate!**
   Easy to apply transitions and animations.

4. **Script away!**
   Everything is in the DOM.

5. **Better accessibility! Plus fallbacks!**
   Fool-proof, once you set it up well.

6. **Better semantics!**
   `<svg>` = “image” / `<span>` = “nothing”
1. **Vector!**
   Typically sharper than icon fonts because of non-text anti-aliasing.

2. **Easy multi-color!**
   More CSS control than any other method.

3. **Animate!**
   Easy to apply transitions and animations.

4. **Script away!**
   Everything is in the DOM.

5. **Better accessibility! Plus fallbacks!**
   Fool-proof, once you set it up well.

6. **Better semantics!**
   `<svg> = “image” / <span> = “nothing”`

7. **Ease of use**
   Easy to manage individual icons, instant build processes.

css-tricks.com/icon-fonts-vs-svg/
1. **Vector!**
   Typically sharper than icon fonts because of non-text anti-aliasing.

2. **Easy multi-color!**
   More CSS control than any other method.

3. **Animate!**
   Easy to apply transitions and animations.

4. **Script away!**
   Everything is in the DOM.

5. **Better accessibility! Plus fallbacks!**
   Fool-proof, once you set it up well.

6. **Better semantics!**
   `<svg>` = “image” / `<span>` = “nothing”

7. **Ease of use**
   Easy to manage individual icons, instant build processes.

---

css-tricks.com/icon-fonts-vs-svg/
I guess you guys aren't ready for that yet

but your kids are gonna love it.
I guess you guys aren't ready for that yet

but your kids are gonna love it.
Inline SVG vs Icon Fonts [CAGEMATCH]

If you’re building an icon system for a site, you have some options. If you know the icons need to be raster images, then you’ll likely be using [CSS sprites]. If the icons will be vector images (much more common these days), you have some options. Two of those options are using inline SVG and using Icon fonts.

Let’s compare.

Icons are Vector

There are big advantages to vector icons: resizeable up and down without losing quality, extra sharp on retina displays, and small file size among them.

<table>
<thead>
<tr>
<th>Icon Font</th>
<th>Inline SVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browsers consider it text, so the icons are anti-aliased as such. Can lead to icons not being as sharp as you might expect.</td>
<td>Straight up vector</td>
</tr>
</tbody>
</table>

Front End Design & Development Jobs

*Intesolv* is hiring a Full Stack .Net Developer with Strong Front End Skills

*Chevy* is hiring a Senior Front End Developer

*Chevy* is hiring a Senior Interaction Designer

What’s the web performance situation on your main project?
Speaking of icons, **The Noun Project** is the best site ever for finding simple vectors for about anything.
SVG is pretty great at animation

1. Animate with CSS
2. Animate with JavaScript
Animating SVG with CSS is just like animating HTML with CSS.

If it’s inline SVG, the CSS can be anywhere, like with the rest of the CSS for your page.

If you use the SVG any other way, you have to embed the CSS within the SVG.
```html
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1" width="0" height="0" viewBox="0 0 600 800" preserveAspectRatio="" fill="url(#C2332)"

cx="386" cy="783.4" r="89"

<path id="left_arm" fill="#BAD3B7" d="M184 438.9c-44.8-84.8 8-94.2 44.1-18.9 3.3-22.1 1.3-28.8 5.5-1-1.8 2.6-3.2 2.1-4.5 5.2-6.2 8.3-5.8 10-2.3 23.1 7.7 32.4 3.8-7.7-7 3.8-6.8-4.6-10.6-14.6-18.5-22.3 0.1 7.7 3.7 15.9 10.5 22.1 8.7-8.7c-4.8-5.1-6.7-11.7-4.8-17.3 5.6-5.5 13.8-3.4 21 0.8 8.7 5 13 14.3

<head

css[
animation: hover .5s ease-in-out 0s infinite alternate]

#head{
animation: hover 1.1s ease-in-out 0.05s infinite alternate}

#body{
animation: hover 1.1s ease-in-out 0.1s infinite alternate

#arms{
animation: hover 1.1s ease-in-out 0.1s infinite alternate

```
IDEAS FOR YOUR PROJECTS

Design Patterns

Accordions

Menu Icons
The ubiquitous hamburger.

Grids
The layout helper of choice.

Charts & Graphs
Destiny awaits.

Tabs
Peek-a-boo!

Star Ratings
From one to five.

Buttons
Push it. Push it real good.

Breadcrumbs
Experientially delicious.

Dialogs
Answer the question, Claire.
Animating SVG with JavaScript can mean many different things.

JavaScript can do anything!
This is just a dumb ol’ loop that changes an attribute. **But that’s animation!**

```javascript
var circle = document.getElementById("orange-circle"),
    positionX = 0;

setInterval(function() {

    positionX += 10;

    circle.setAttribute("cx", positionX);

    if (positionX > 500) {
        positionX = 0;
    }

}, 20);
```
Typically, it means use a library. All of these work with SVG, but all have slightly different capabilities, approaches, and focuses.

1. **Greensock (GSAP)**
greensock.com - does some cool normalization stuff too

2. **Snap.svg**
snapsvg.io - jQuery for SVG - kinda like newer Raphaël

3. **Velocity.js**
julian.com/research/velocity

4. **SVG.js**
svgjs.com

5. **D3**
d3js.org - data powerhouse
"High Five SVG Animation" by MailChimp UX :: codepen.io/mailchimpux/pen/Gblcs
“Yes/No SVG Tick Animation” by Chris Gannon :: codepen.io/chrisgannon/pen/ogEjRa/
```javascript
TweenLite.set("svg", {position:"absolute", left:"50%", xPercent:-50});
TweenLite.defaultEase = Power4.easeInOut;
var tl = new TimelineMax({delay:8.3}),
    flap = new TimelineMax({repeat:10}),
    scrubTween = TweenLite.to("#scrubber", 10, {x:882, ease:Linear.easeNone, paused:true});

tl.to("#M", 2, {morphSVG:{shape:"cape", shapeIndex:-19}, fill:"#444"}, 0);
    .to("#H", 2, {morphSVG:"#torso", fill:"#777"}, 0.1);
    .to("#P", 2, {morphSVG:{shape:"legs", shapeIndex:-24}, fill:"#777"}, 0.1);
    .to("#D", 2, {morphSVG:"#head", fill:"#777"}, 0.66);
    .to("#R", 1.5, {morphSVG:{shape:"sock", shapeIndex:-11}, fill:"#48CE02", transformOrigin:"-100 60"}, 0.5);
    .set("#GUY", capeFlow1, {visibility:"visible", opacity:1}, 1.8);
    .to("#M", 0, "{0, R, P, H}"), 0.7, {autoAlpha:0, ease:Linear.easeNone}, 1.6);"
```javascript
TweenLite.set("svg", {position:"absolute", left:"50%", xPercent:-50});
TweenLite.defaultEase = Power4.easeInOut;
var tl = new TimelineMax({delay:0.3});
flap = new TimelineMax({repeat:18});
scrubTween = TweenLite.to("#scrubber", 18, {x:882, ease:Linear.easeNone, paused:true});

tl.to("#M", 2, {morphSVG:{shape:"cape", shapeIndex:-19}, fill:"#444"}, 0)
  .to("#H", 2, {morphSVG:"torso", fill:"#777"}, 0.1)
  .to("#P", 2, {morphSVG:{shape:"legs", shapeIndex:24}, fill:"#777"}, 0.1)
  .to("#D", 2, {morphSVG:"head", fill:"#777"}, 0.65)
  .to("#R", 1.5, {morphSVG:{shape:"sock", shapeIndex:-11}, fill:"#88CE02", transformOrigin:"-100% 60%"}, 0.5)
.set("#GUY", capeFlow1, {visibility:visible, opacity:1}, 1.0)
.to("#M", 0, "R", "P", ", #H", 0.7, {autoAlpha:0, ease:Linear.easeNone}, 1.6);
flap.add[
waveSVG(document.getElementById("capeBottom1"), {taperEnd: 80, taperStart:2, loose:true, length:128, angle:-52,
```

MorphSVG Plugin
From GreenSock

---

```
CSS-TRICKS
You can do **CLIPPING & MASKING**

**Clipping paths** are always vector. Inside the vector shape is shown, outside the vector path is hidden.

**Masks** are images. They can be vector too, but they don’t have to be.
SVG Masking Experiment.
Move your mouse/finger over the image.

div ello.me

```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
<defs>
  <clipPath id="mask">
    <circle id="mask-circle" cx="50%" cy="50%" r="8%"
            style="fill:#fff"/>
  </clipPath>
</defs>
<clip-path url="#mask">
  <rect width="100%" height="100%"
         fill="#272727"/>
  <image xlink:href="http://www.w3.org/1999/xlink"/>
</clip-path>
</svg>
```

```
var svgElement = document.querySelector('svg');
var maskedElement = document.querySelector('#mask-circle');
var circleFeedback = document.querySelector('#circle-shadow');
var svgPoint = svgElement.createSVGPoint();

function cursorPoint(x, y) {
  svgPoint.x = e.clientX;
  svgPoint.y = e.clientY;
  return svgPoint.matrixTransform(svg.getScreenCTM());
}
```
It’s overused already but who cares it’s awesome
"Animated SVG Headphones" by Chris Gannon :: codepen.io/chrisgannon/pen/zxWowX
SVG is pretty great at **CHARTING**


SVG doesn’t have charting-specific features. It has features that lend themselves well to charting.
Using SVG Filters

You can use SVG filters with our charts and maps. The background of this chart are real graphs (not just an images) with blur filter applied. This opens wide horizons for you, designers!
How to Make Charts with SVG

In my first post about making charts, I looked at methods that solely relied on CSS. I argued that this wasn’t the best option in most cases; there are just too many tricky design and development hurdles to overcome. Generally speaking, it's best to make charts with a combination of SVG, JavaScript, and CSS.

Why not canvas?
The original gangsters

SVG FILTERS

“The Gooey Effect” by Lucas Bebber :: css-tricks.com/gooey-effect/
More, more, MORE

All this has been the tip of the iceberg. There is a TON to know about SVG that we didn't cover. I mostly wanted to just get you more excited about it.

Huge list of information about SVG:
css-tricks.com/mega-list-svg-information/
THANKS!

@chriscoyier
codepen.io