SVG For Web Designers and Developers

Hello! (◠‿◠)

— Last name is pronounced ***Swaïdaan***

— Front-End **Developer**

— Author, Codrops CSS Reference

— Co-Author, Smashing Book 5

— SVG articles: http://sarasoueidan.com/articles

SVG (basic support) - REC

Global 91.83% + 4.09% = 95.92%

Method of displaying basic Vector Graphics features using the embed or object elements. Refers to the SVG 1.1 spec.

Current aligned | Usage relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
8			31					1 4.1	
2 9		38	43					1 4.3	
2 10		39	44			7.1		4.4.4	
2 11	**2 12**	**40**	**45**	**8**	**31**	**8.4**	**8**	**44**	**44**
	2 13	41	46	9	32	9			
		42	47		33				
		43	48						

Notes | Known issues (3) | Resources (7) | Feedback

1 Partial support in Android 3 & 4 refers to not supporting masking.

2 IE9-11 desktop & mobile don't properly scale SVG files. Adding height, width, viewBox, and CSS rules seem to be the best workaround.

= Supported = Not supported = Partial support = Support unknown

Need to support older browsers? There's a whole bunch of ways you can do that; and tools to automate the process for you.

When to SVG?

— Raster images are the preferred format when creating or working with continuous-tone images such as photographs.

— SVG is the preferred format for images like user interface controls, logos, icons and vector-based illustrations.

PNG: ~66KB

SVG: ~123KB

Lighting effects and shadows increase SVG file size to ~300KB.

Optimizing it and reducing effects slashes file size down to ~40KB.

PNG version currently served is 5.9KB.

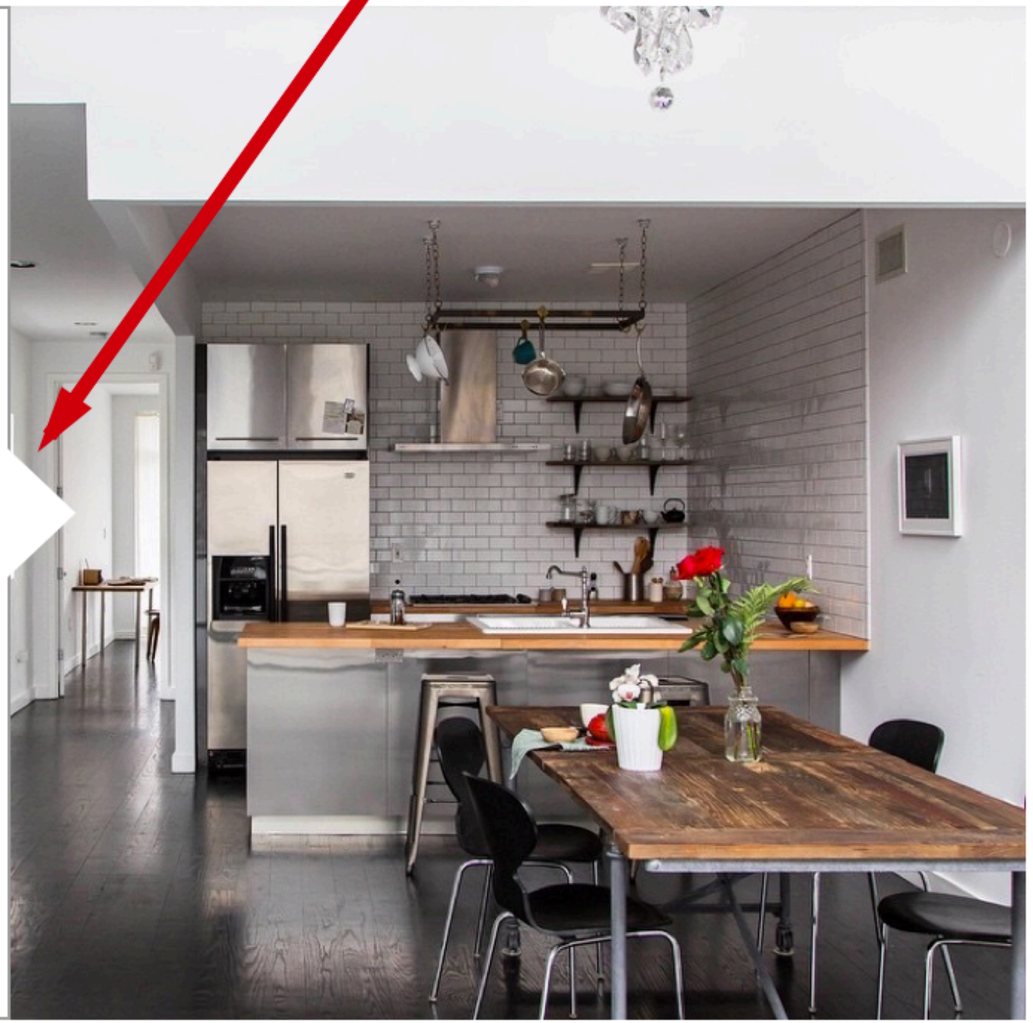
SVG can do more than display images.

— Icon Systems
— Ad banners
— Infographics
— Data visualizations
— Animated illustrations
— Filter effects (on SVG as well as HTML elements, including text)
— Simple UI shapes and arbitrarily-shaped UI components

Contemporary Design

Lorem Ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



CSS triangle:

```
.arrow::after {
  content: '';
  position: absolute;
  width: 0;
  height: 0;
  width: 0;
  height: 0;
  border-top: 30px solid transparent;
  border-left: 100px solid red;
  border-bottom: 30px solid transparent;
  z-index: 1;
  right: -30px;
  top: 50%;
  margin-top: -30px;
}
```

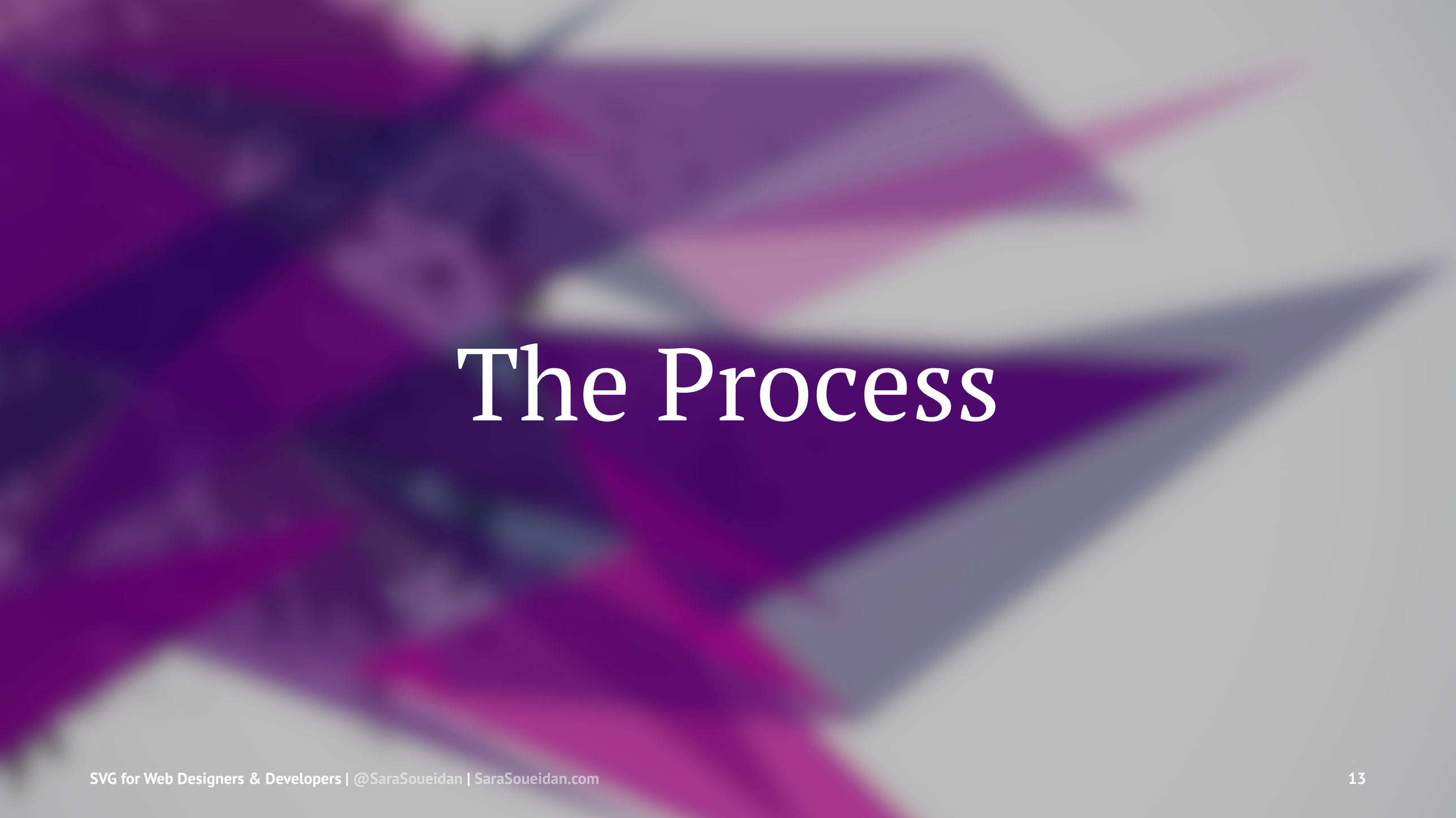
Custom @-rule (as it appears in PostCSS):

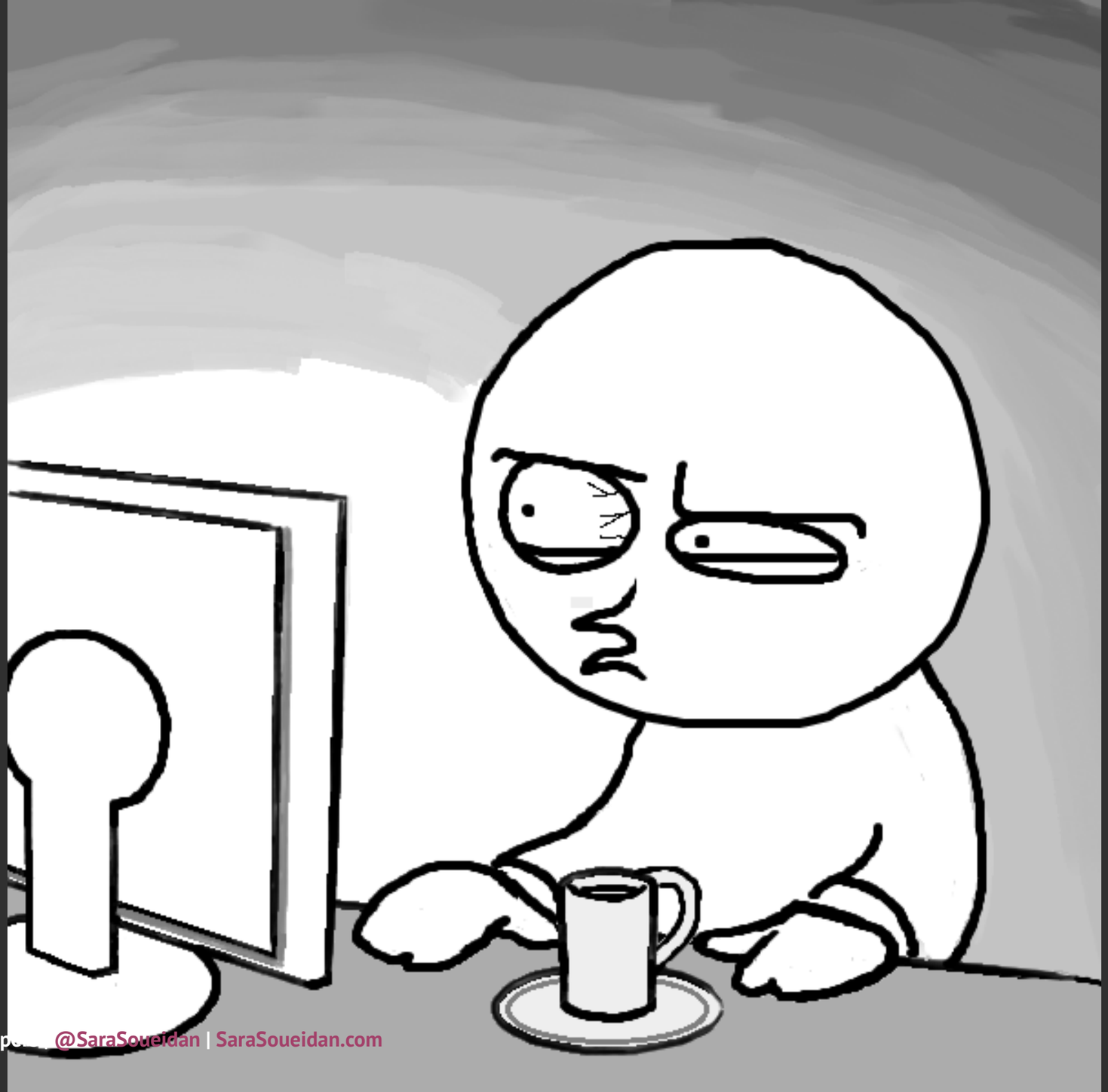
```
.arrow {
    @svg {
        polygon {
            fill: green;
            points: 50,100 0,0 0,100;
        }
    }
}
```

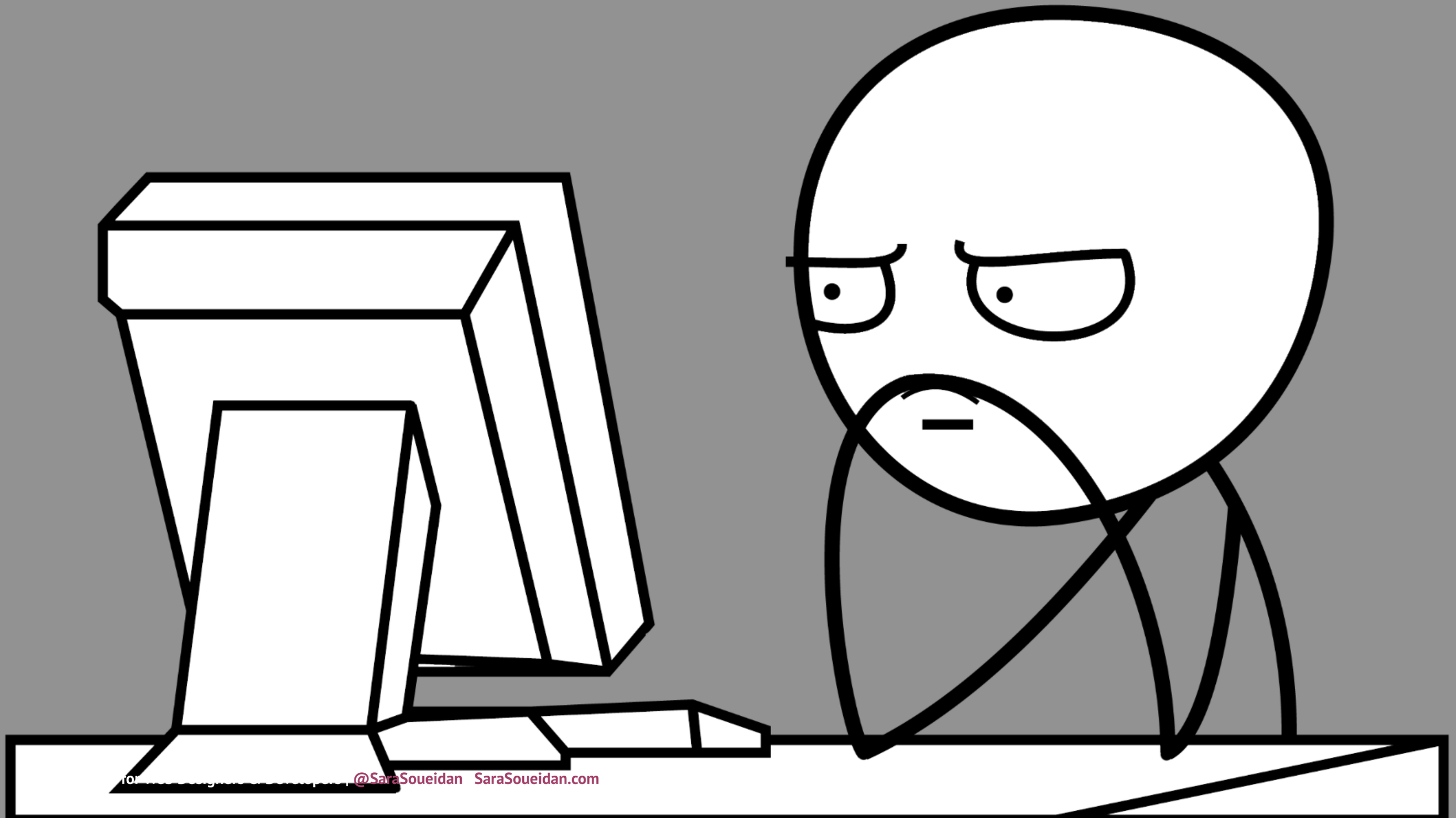
Source: https://github.com/jonathantneal/postcss-write-svg

Sass Mixin: https://github.com/davidkpiano/sass-svg

CSS Spec: *Coming Soon*!


The Process





Knowledge gap

Designers need to know a little bit about code and developers need to know a little bit about the design process in order to suggest alternatives to avoided approcahes. **They can help each other fill that knowledge gap**.

Design

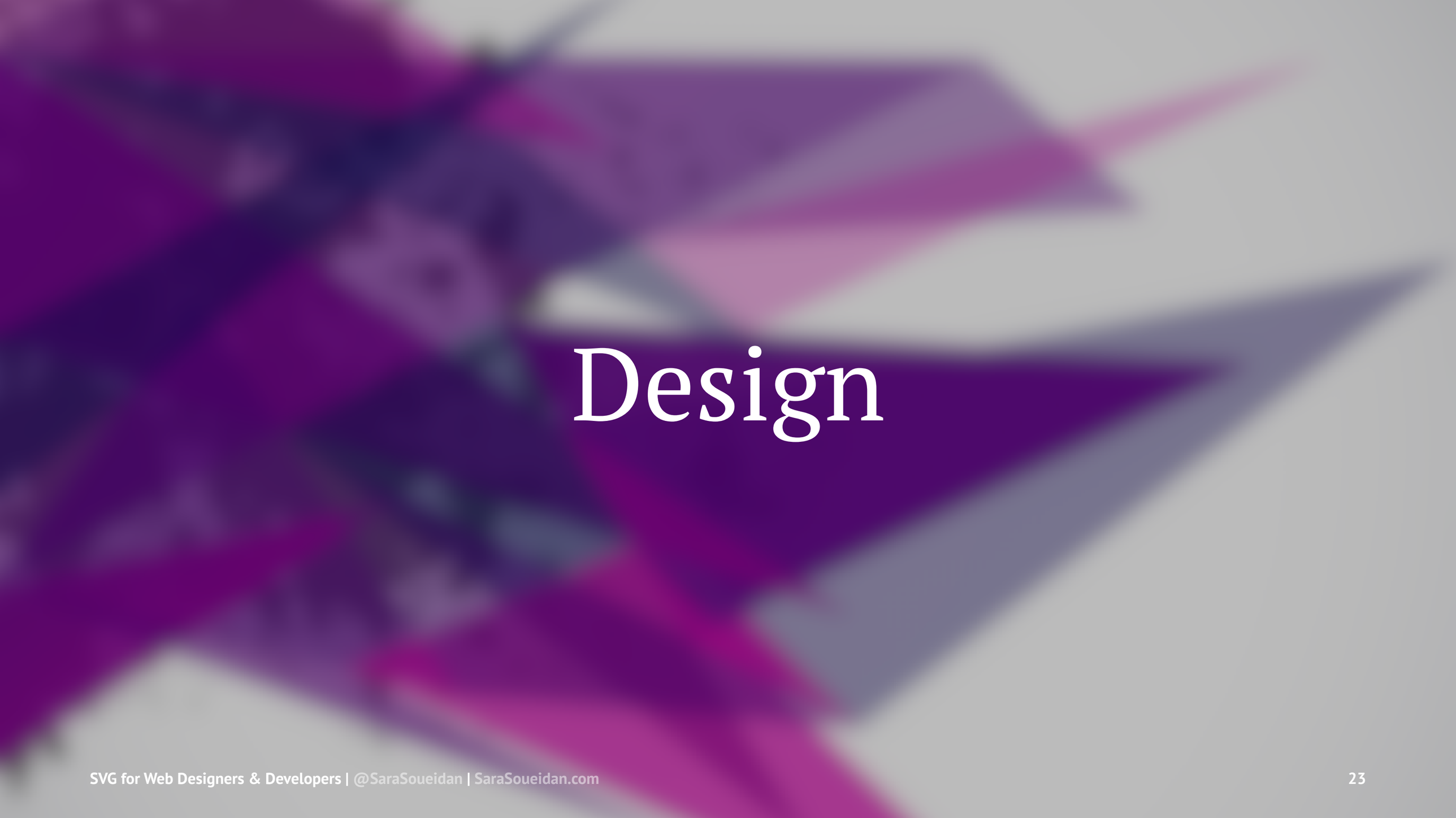
Creating the SVG (e.g. in a graphics editor)

Development

Embedding, Spriting, Scripting, Animating

Export

Optimization


Design

1) Convert text to outlines.. or don't.

- Outlined text is not selectable/searchable/accessible
- Outlined text will preserve the font face you're using w/o needing a web font (good for: logos, lettering)
- Outlining can significantly increase file size depending on font styles
- Outlined text is made up of paths, so might not be as controllable or animatable as individual characters

2) Create simple shapes using simple shape elements instead of <path>s.

— Simple shapes are easier to maintain and edit manually

— Simple shapes are easier to animate (attribute values)

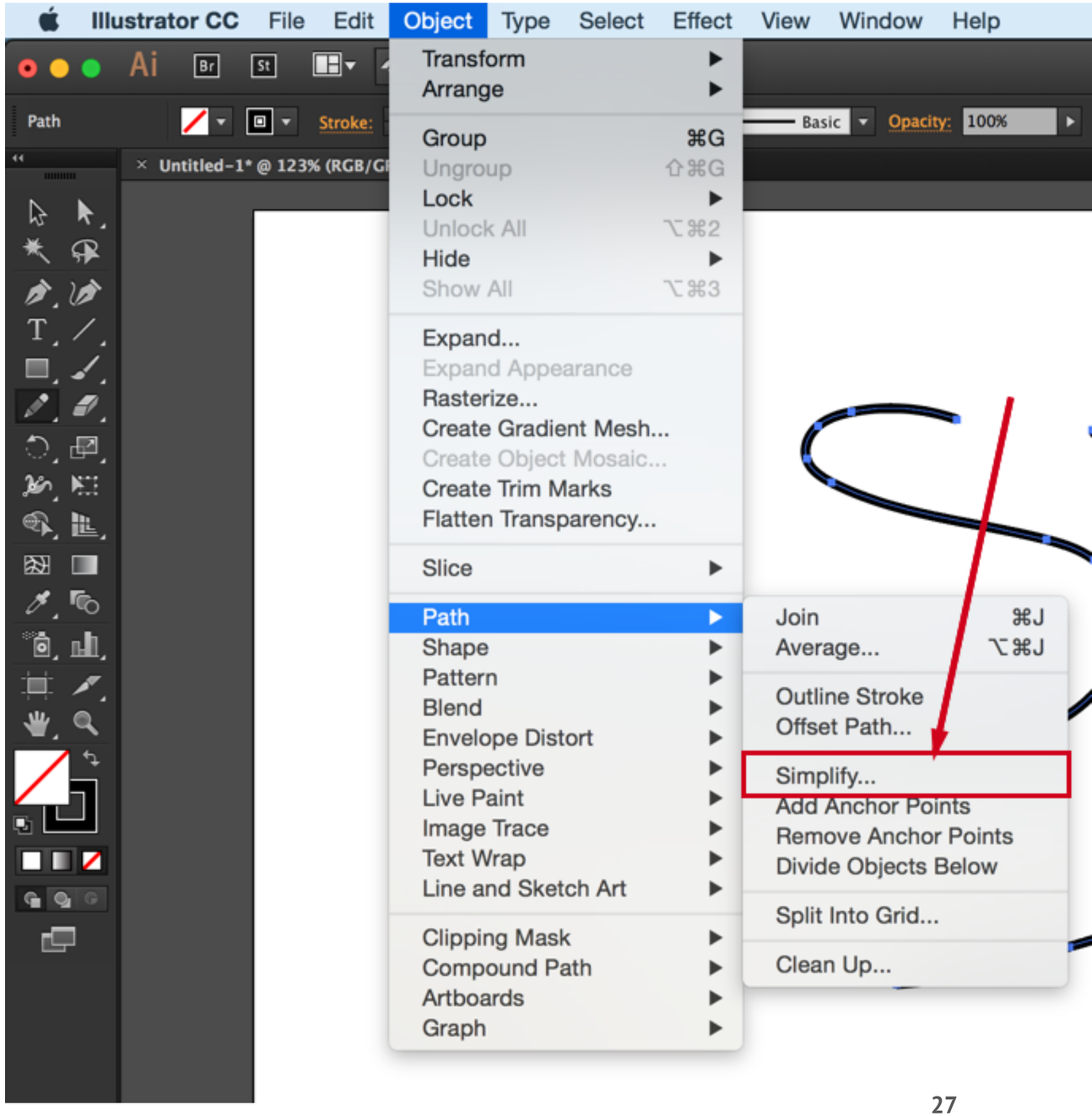
```
<circle fill="#FFFFFF"
        stroke="#000"
        cx="28.1"
        cy="28.1"
        r="27.6"/>
```

Versus

```
<path fill="#FFFFFF"
      stroke="#000"
      d="M55.7,28.1
         c0,15.2-12.4,27.6-27.6,27.6
         S0.5,43.3,0.5,28.1
         S12.9,0.5,28.1,0.5
         S55.7,12.9,55.7,28.1z"/>
```

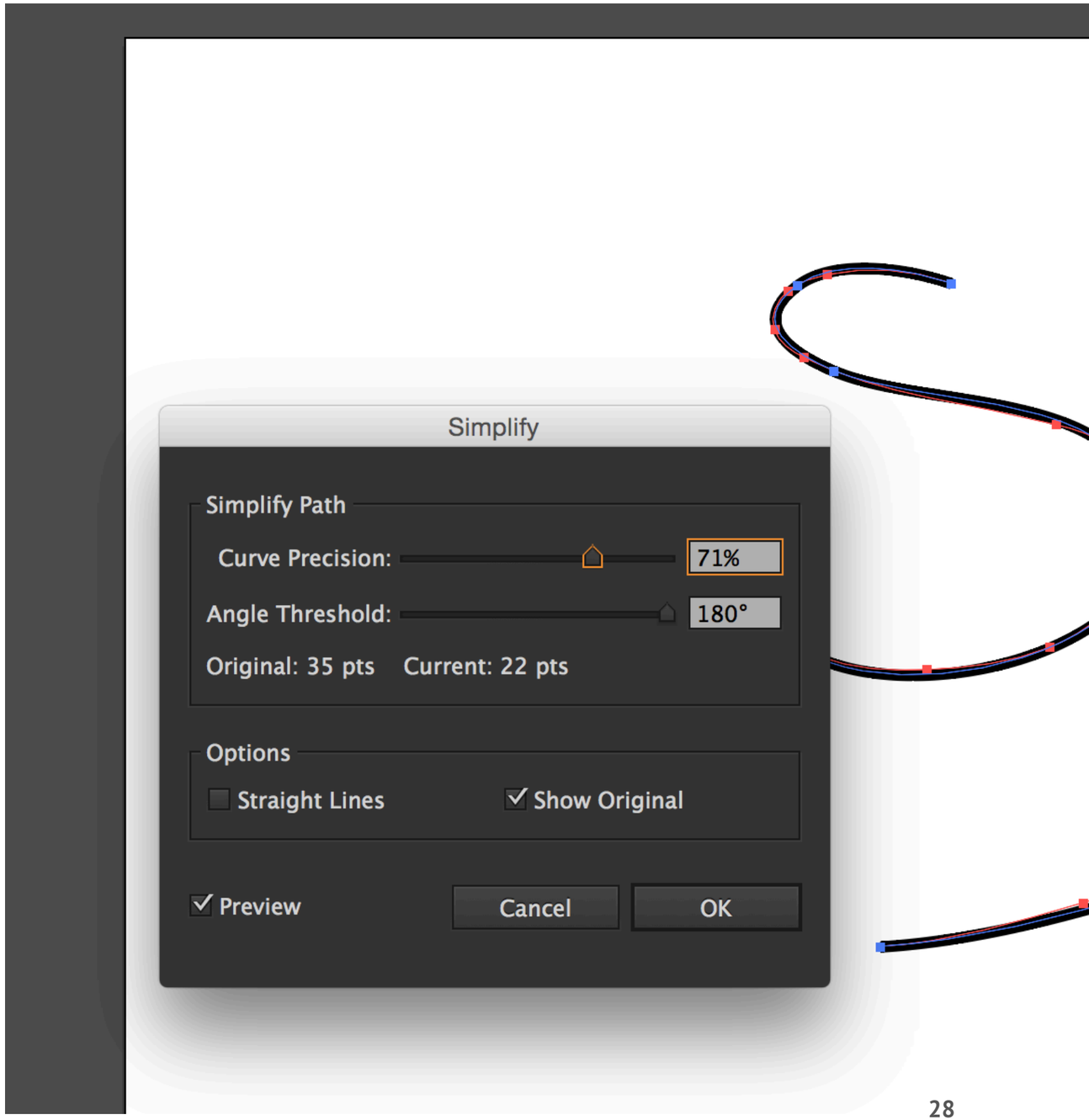
3.a) Simplify Paths

Each point is represented as a pair of coordinates in the <path> data.



3.b) Simplify Paths

Use Ai's Simplify algorithm or use **the Warp tool**.

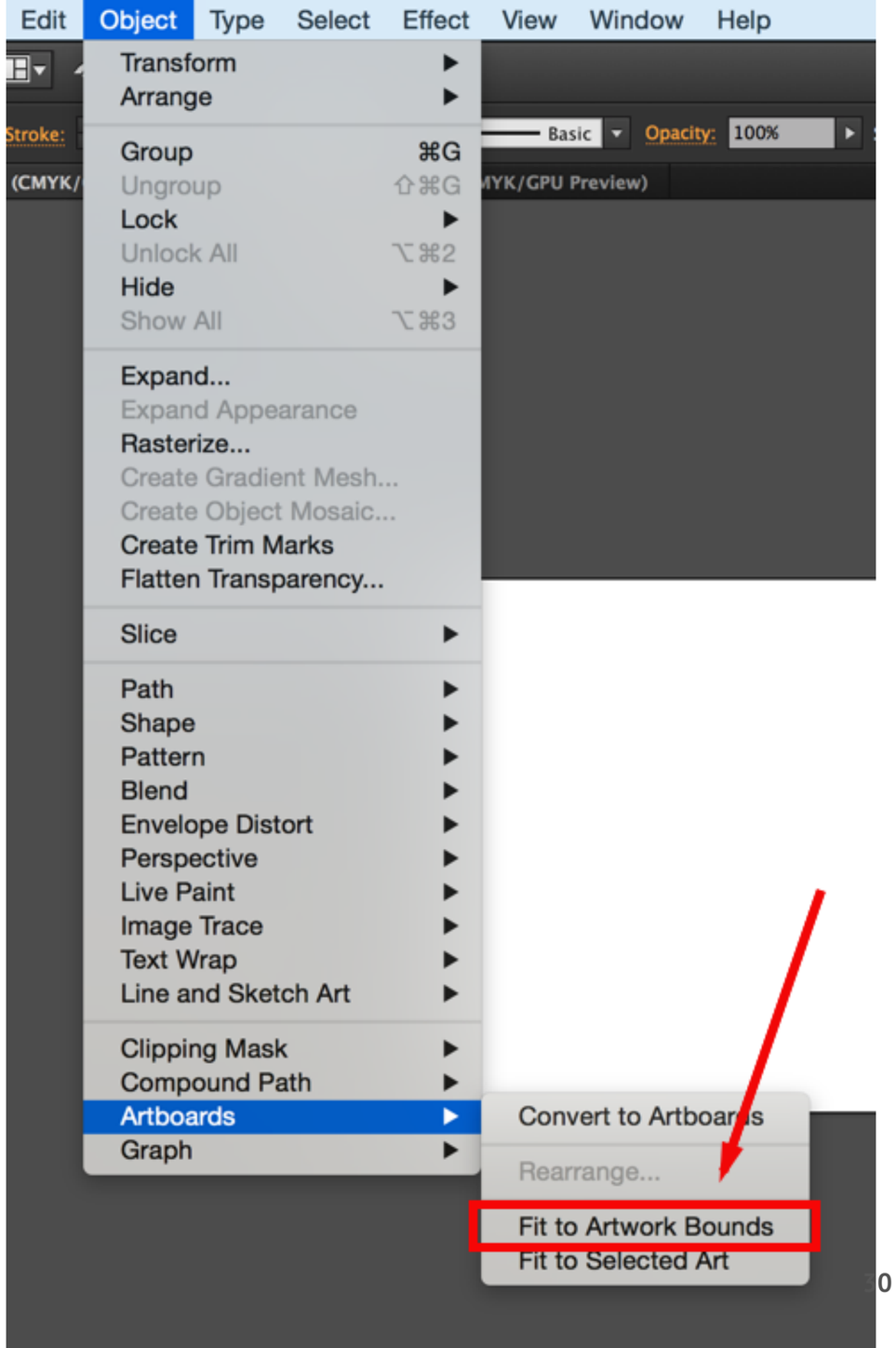


4) Combine Paths Where Possible

… but only if you don't want to animate the individual paths separately.

5) Fit artboard to drawing.

This allows you to crop the SVG and thus get rid of any extra white space around your drawing.



6) Use Good Grouping, Layering and Naming Conventions

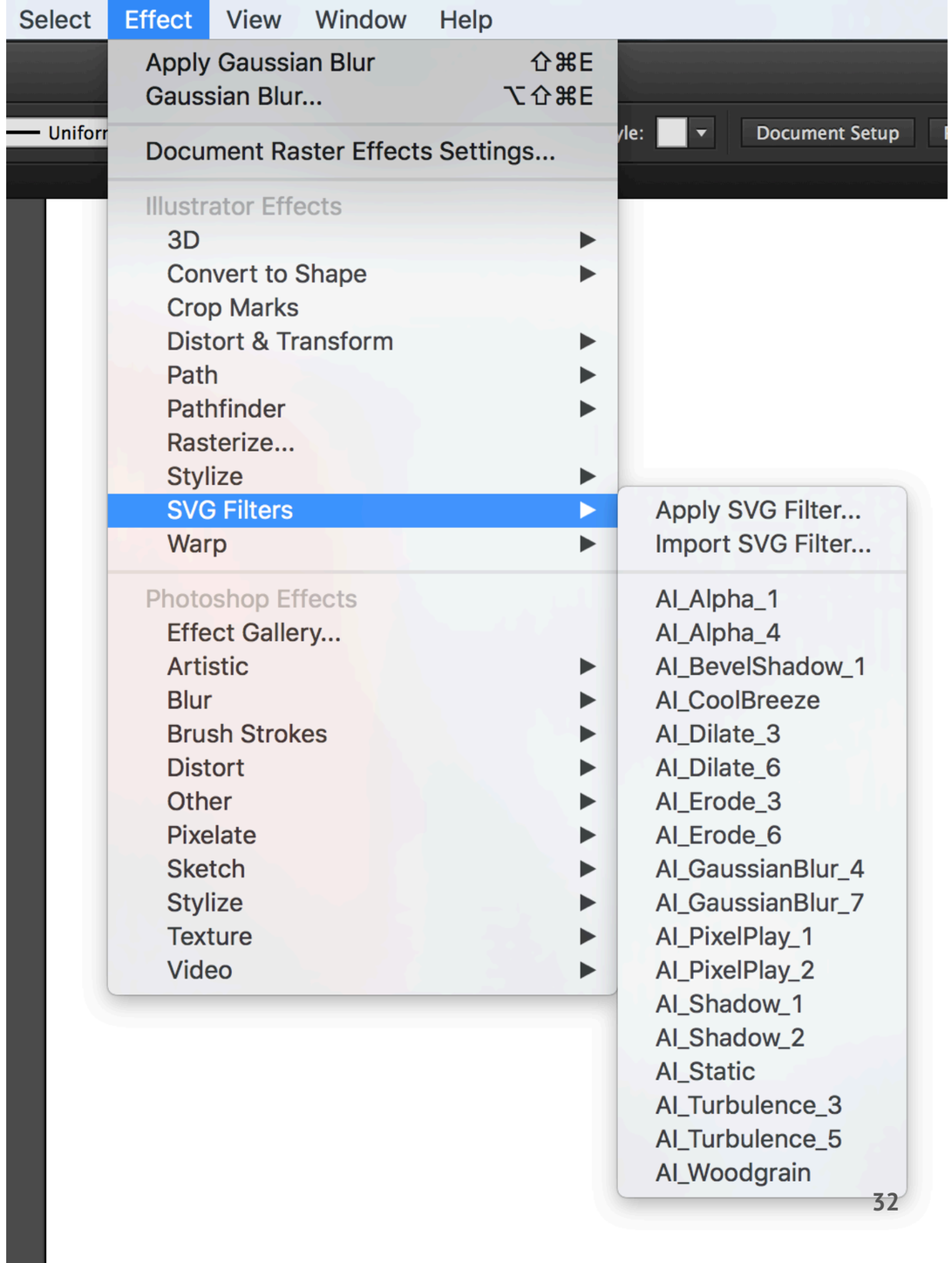
— The layer and group names you use in Illustrator will be translated to IDs in the SVG code.

— Makes scripting, styling and editing SVG code easier.

— Best for automated workflows that use file names in SVG generation.

7) Create filters Using the *SVG Filters* available in Ai, not the Photoshop Effects

Photoshop effects will be exported as bitmaps, not converted into SVG filters.



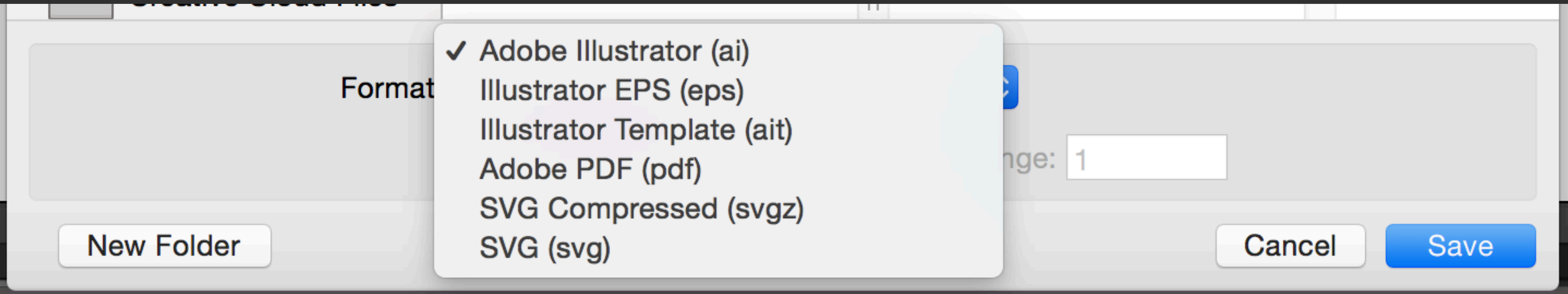
Photoshop can natively export SVG.

1. Select the layers you want to export.
2. Open the *“File > Extract Assets”* window.
3. Select layers and define in which format you want to export them one by one. (You have the choice between JPEG, PNG and SVG.)

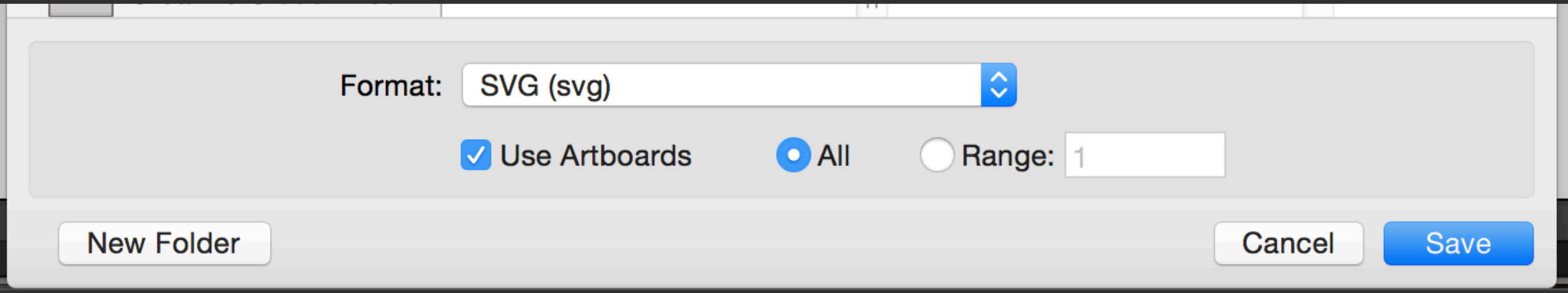
http://creativedroplets.com/generate-svg-with-photoshop-cc-beta/

8) Export

File → Save As

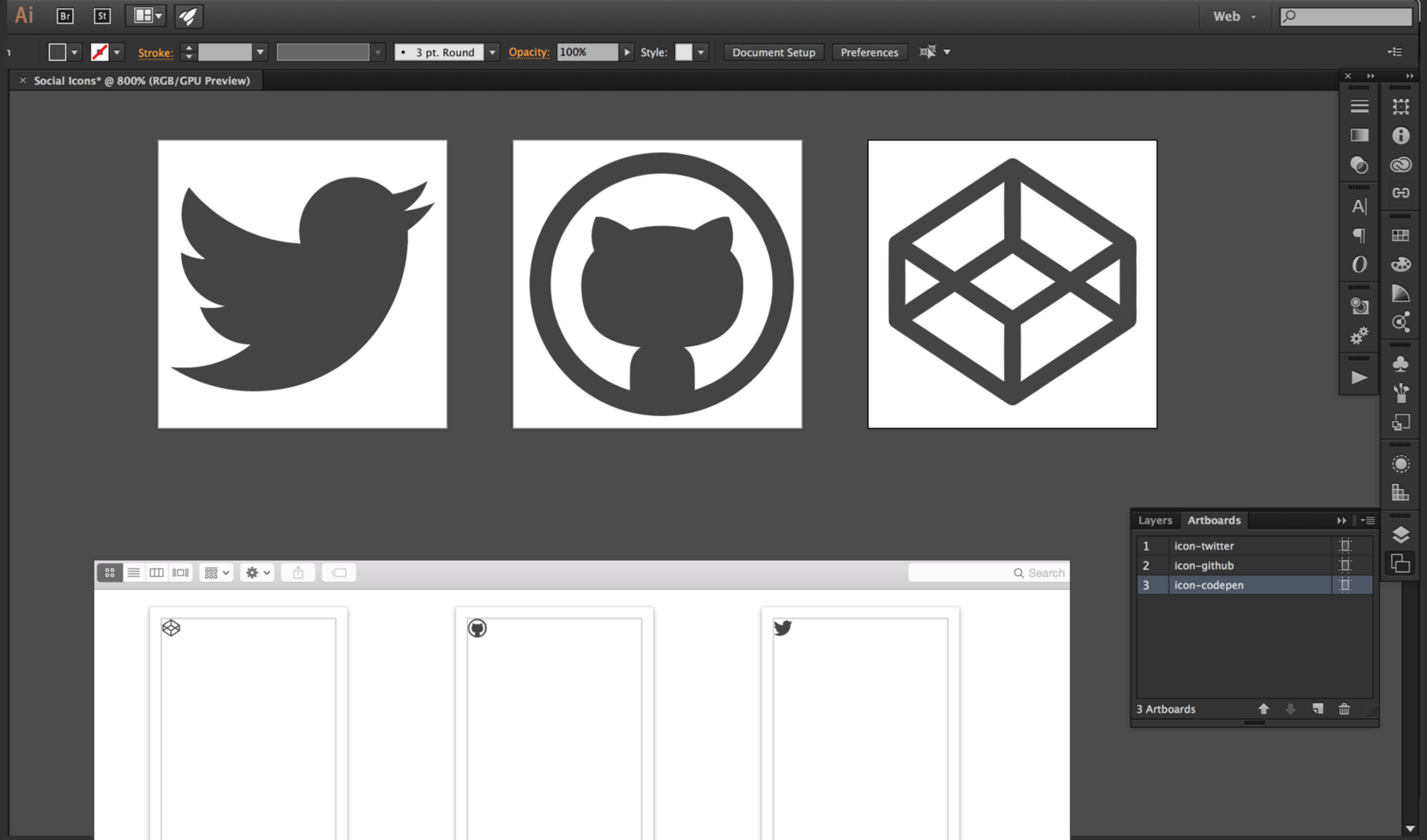


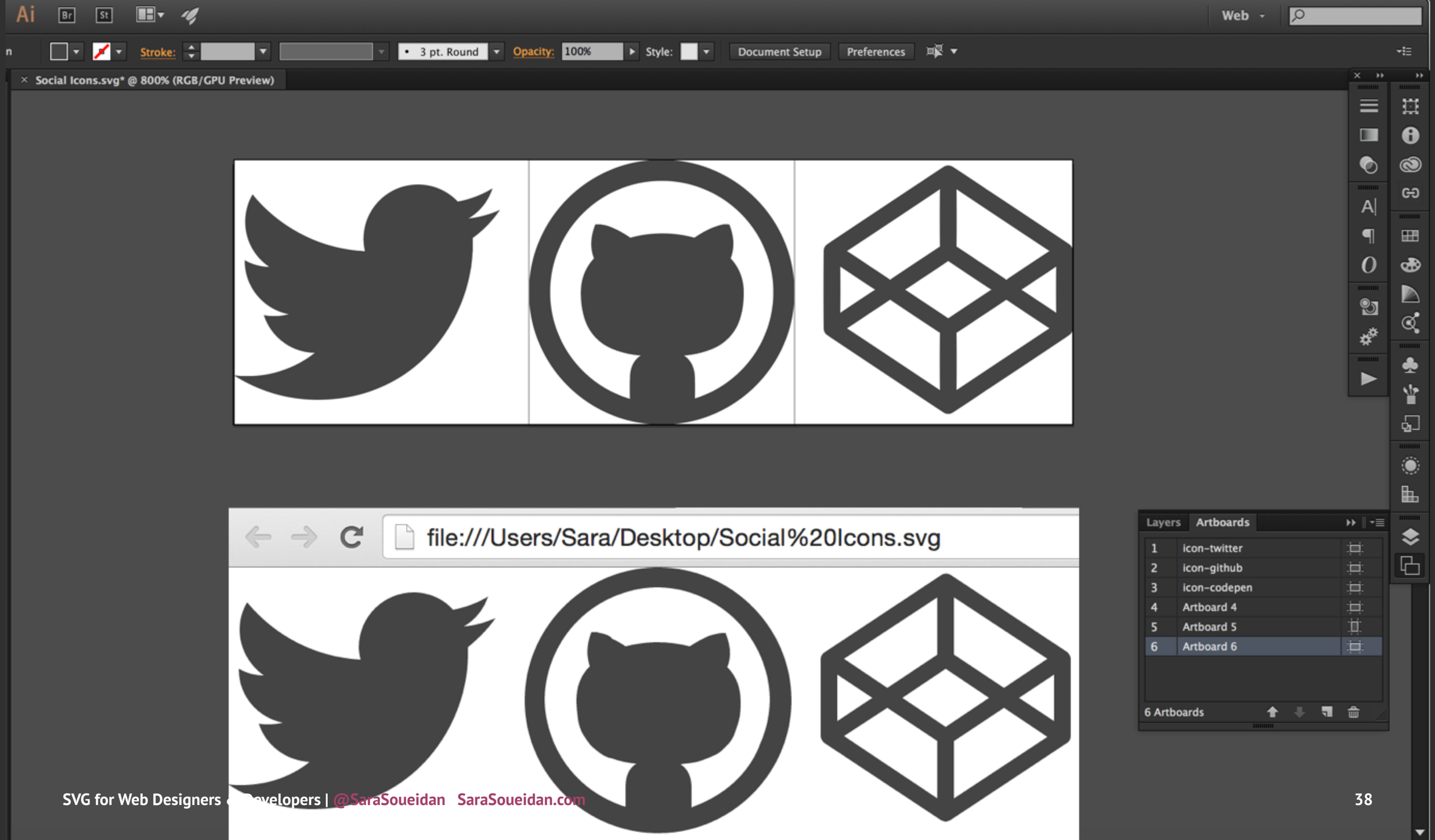
Export multiple SVG files using multiple artboards if/when needed.



One or Multiple Artboards?

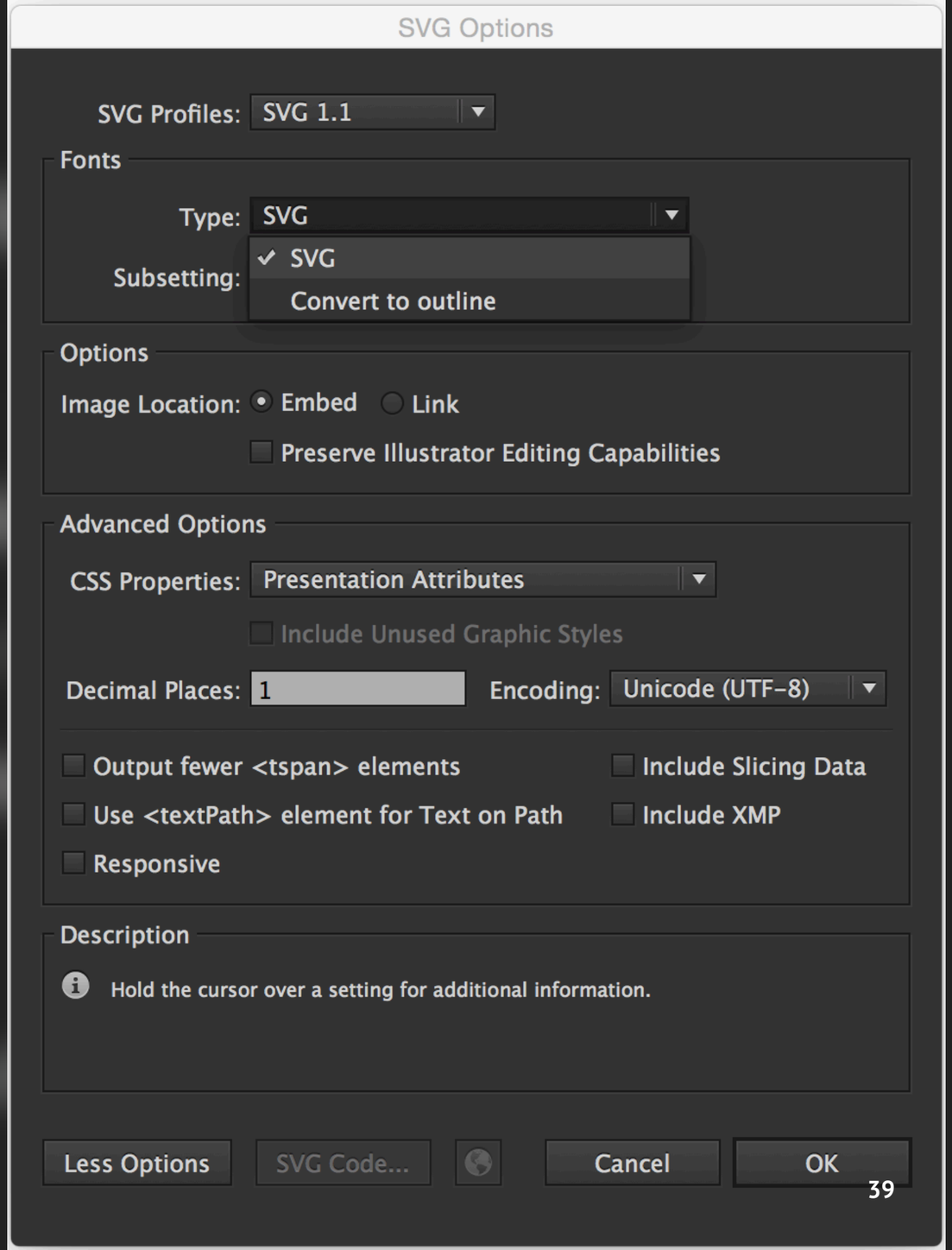
That depends on the spriting technique you intend to use…


Ai
Web
Stroke:
3 pt. Round
Opacity:
100%
Style:
Document Setup
Preferences
Social Icons* @ 800% (RGB/GPU Preview)
Layers
Artboards
1 icon-twitter
2 icon-github
3 icon-codepen
3 Artboards
Search
Social Icons_icon-codepen.svg
Social Icons_icon-github.svg
Social Icons_icon-twitter.svg


Ai
Web
Stroke:
3 pt. Round
Opacity:
100%
Style:
Document Setup
Preferences
Social Icons.svg* @ 800% (RGB/GPU Preview)
file:///Users/Sara/Desktop/Social%20Icons.svg
Layers
Artboards
1 icon-twitter
2 icon-github
3 icon-codepen
4 Artboard 4
5 Artboard 5
6 Artboard 6
6 Artboards

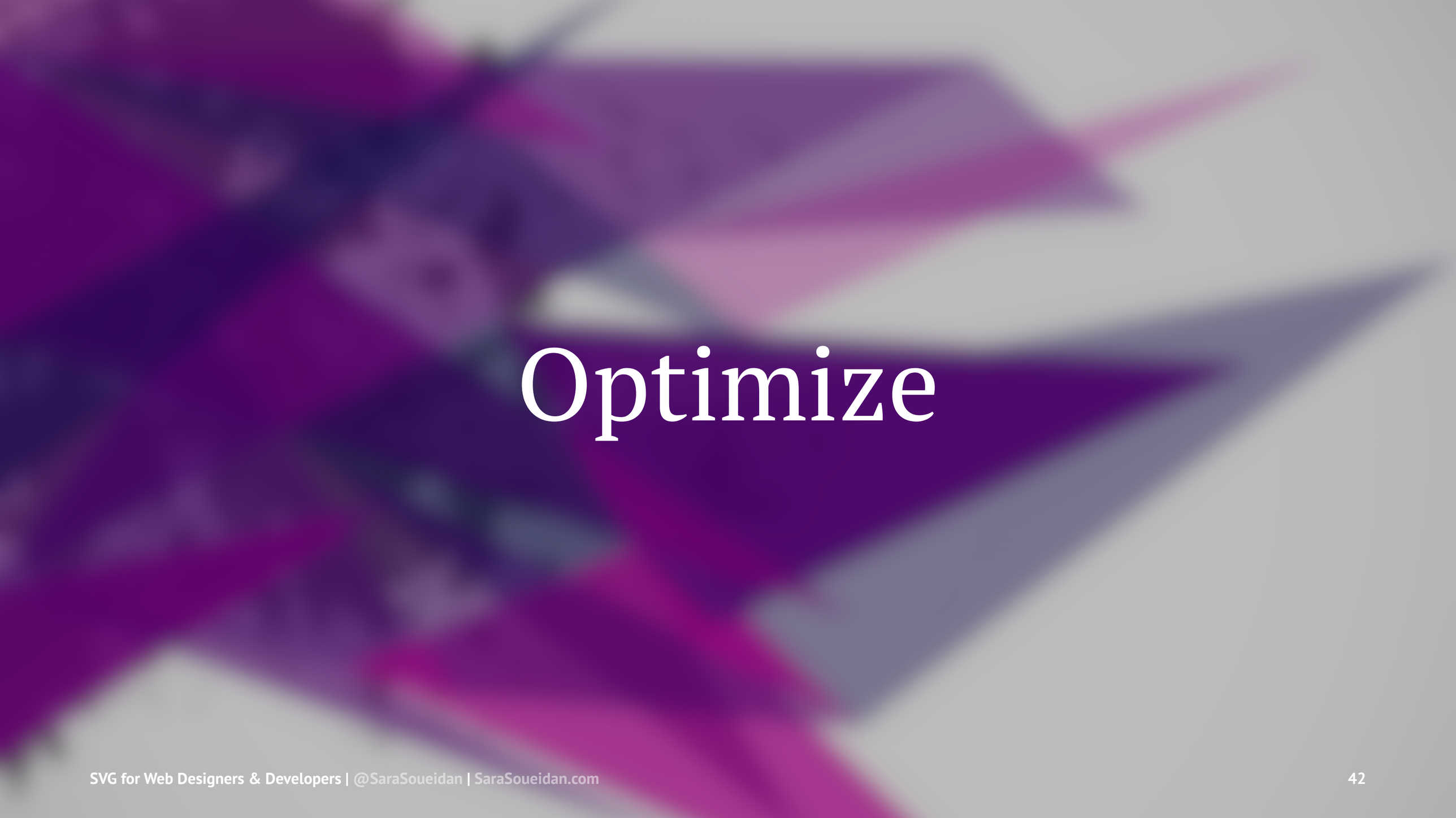
Exporting SVG for the Web with Illustrator:

http://goo.gl/c0Ri4k



Always keep the `width` and `height` attributes on the `<svg>`.

Communication = 💖


Optimize

SVGO Grunt Plugin

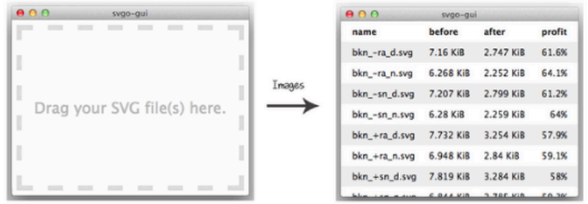
SVGO Gulp Plugin

SVGO

SVG NOW for Adobe Illustrator

OS X Folder Action

SVGOMG (Online GUI)



The SVGO drag-and-dropGUI

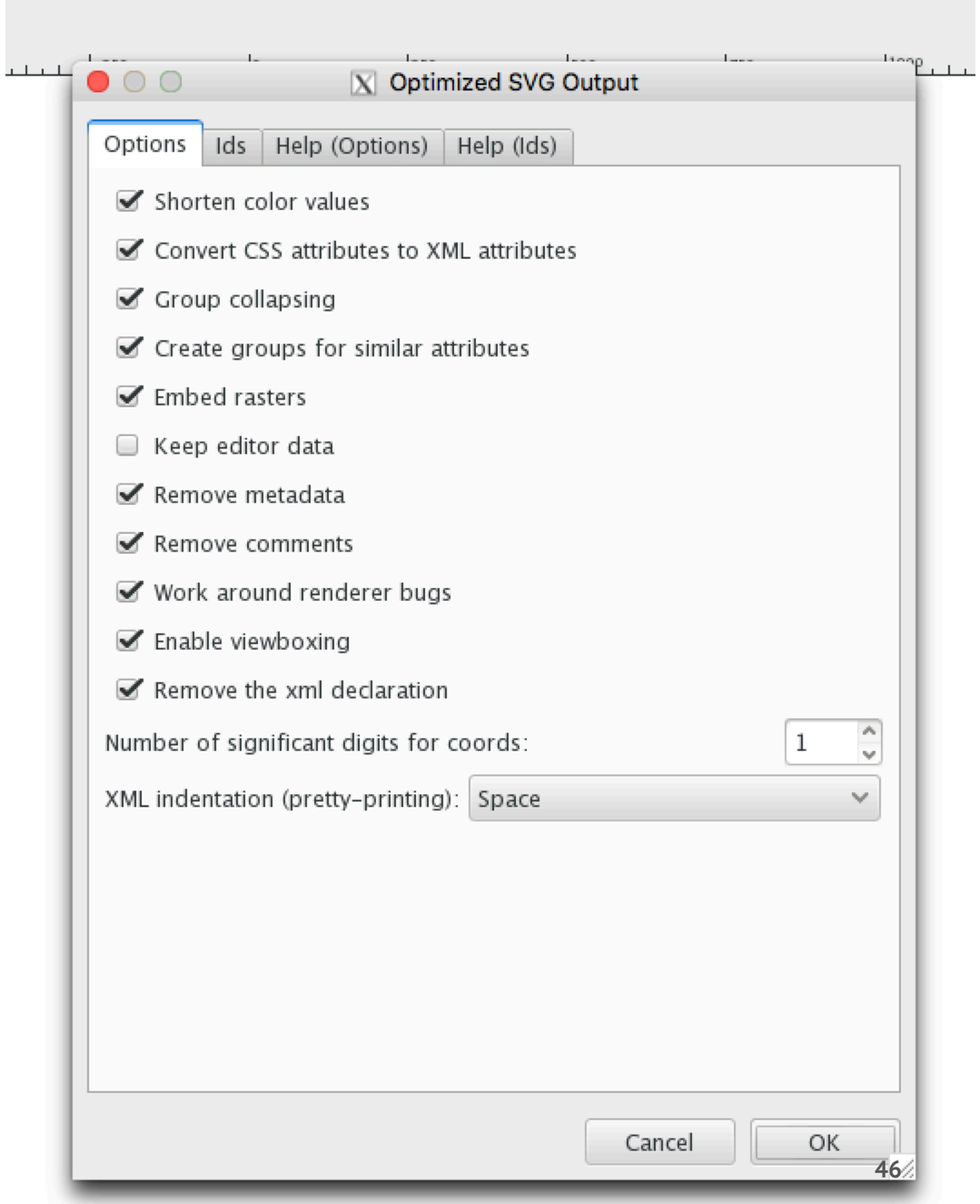
http://sarasoueidan.com/blog/svgo-tools/

You may not always need or want to optimize…

Some editors offer more optimization options that yield cleaner code.

Inkscape is a great example of that.

Ai will be getting better.



Remember: optimization (esp. using standalone tools) can and will change the document structure, thus affecting any animation/scripting.

Develop

Spriting Technique #1

Multiple SVG files (e.g. icons) are combined into one SVG file using <symbol>s

```
<svg style="display: none;">

    <symbol id="icon-twitter" viewBox="0 0 35 35">
        <!-- Twitter icon markup -->
    </symbol>

    <symbol id="icon-github" viewBox="0 0 35 35">
        <!-- Github icon markup -->
    </symbol>

    <!-- more icons here... -->
</svg>
```

Accessible icons

```
<svg style="display: none;">

   <symbol id="icon-twitter" viewBox="0 0 35 35"
           role="img" aria-labelledby="title description">
    <title>Twitter</title>
    <description>...</description>
      <!-- Twitter icon markup -->
   </symbol>

   <symbol id="icon-github" viewBox="0 0 35 35"
           role="img" aria-labelledby="title description">
    <title>Github</title>
    <description>...</description>
      <!-- Github icon markup -->
   </symbol>

   <!-- more icons here... -->
</svg>
```

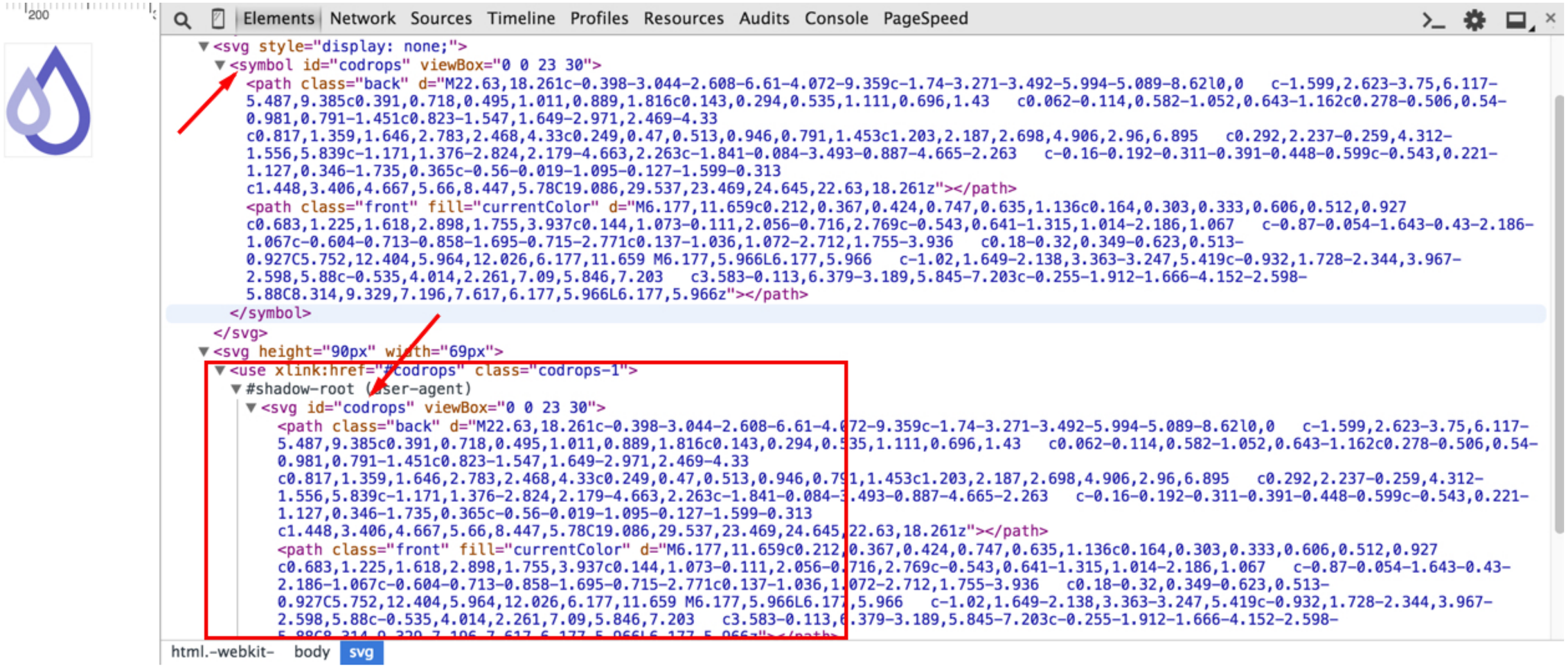
Display the icon(s) anywhere in the page:

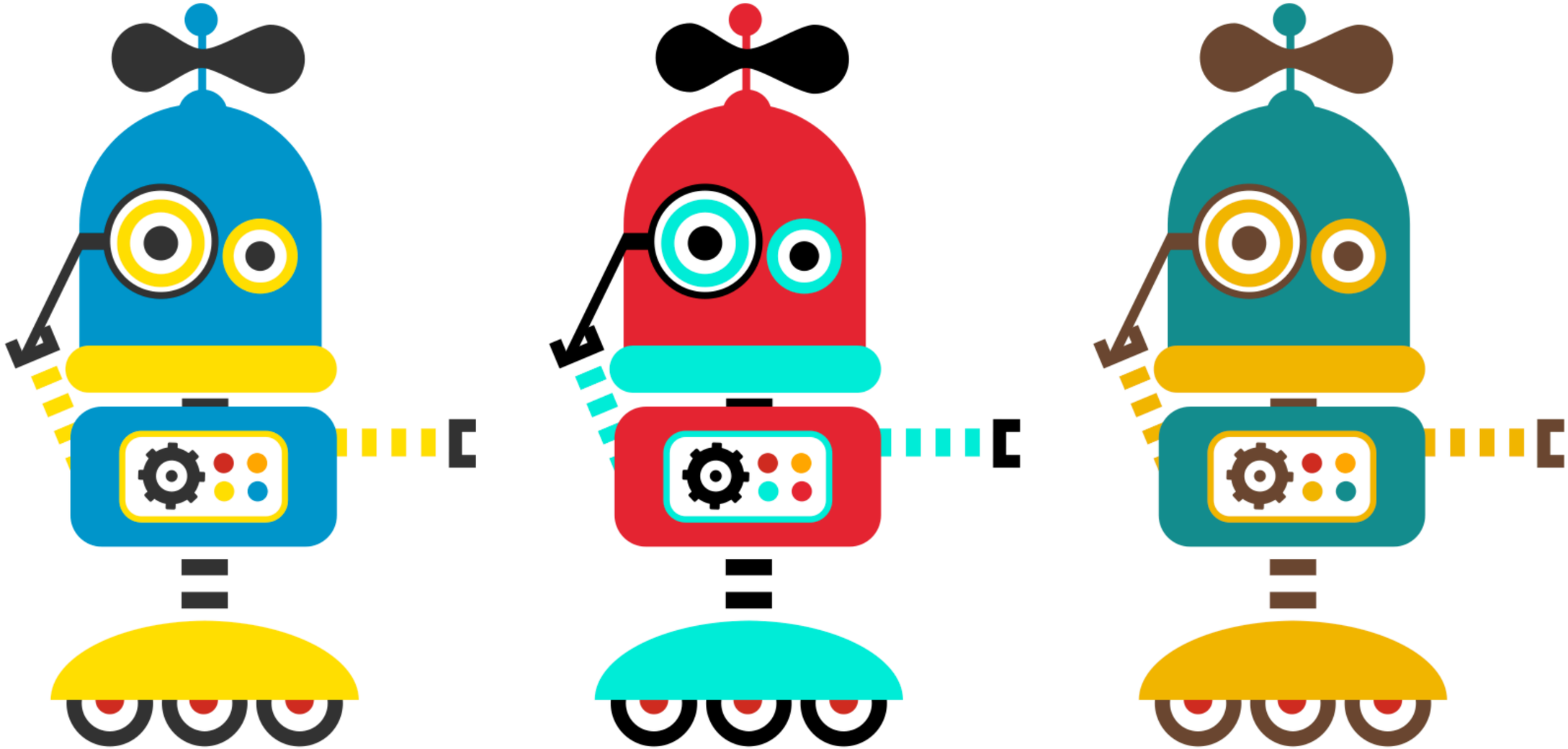
```
<svg class="icon-twitter">
  <use xlink:href="#icon-twitter"></use>
</svg>
```

or

```
<svg class="icon-twitter">
  <use xlink:href="icons.svg#icon-twitter"></use>
</svg>
```

What about styling <use>d images?


200
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
PageSpeed
<svg style="display: none;">
<symbol id="codrops" viewBox="0 0 23 30">
<path class="back" d="M22.63,18.261c-0.398-3.044-2.608-6.61-4.072-9.359c-1.74-3.271-3.492-5.994-5.089-8.62l0,0 c-1.599,2.623-3.75,6.117-5.487,9.385c0.391,0.718,0.495,1.011,0.889,1.816c0.143,0.294,0.535,1.111,0.696,1.43 c0.062-0.114,0.582-1.052,0.643-1.162c0.278-0.506,0.54-0.981,0.791-1.451c0.823-1.547,1.649-2.971,2.469-4.33 c0.817,1.359,1.646,2.783,2.468,4.33c0.249,0.47,0.513,0.946,0.791,1.453c1.203,2.187,2.698,4.906,2.96,6.895 c0.292,2.237-0.259,4.312-1.556,5.839c-1.171,1.376-2.824,2.179-4.663,2.263c-1.841-0.084-3.493-0.887-4.665-2.263 c-0.16-0.192-0.311-0.391-0.448-0.599c-0.543,0.221-1.127,0.346-1.735,0.365c-0.56-0.019-1.095-0.127-1.599-0.313 c1.448,3.406,4.667,5.66,8.447,5.78C19.086,29.537,23.469,24.645,22.63,18.261z"></path>
<path class="front" fill="currentColor" d="M6.177,11.659c0.212,0.367,0.424,0.747,0.635,1.136c0.164,0.303,0.333,0.606,0.512,0.927 c0.683,1.225,1.618,2.898,1.755,3.937c0.144,1.073-0.111,2.056-0.716,2.769c-0.543,0.641-1.315,1.014-2.186,1.067 c-0.87-0.054-1.643-0.43-2.186-1.067c-0.604-0.713-0.858-1.695-0.715-2.771c0.137-1.036,1.072-2.712,1.755-3.936 c0.18-0.32,0.349-0.623,0.513-0.927C5.752,12.404,5.964,12.026,6.177,11.659 M6.177,5.966L6.177,5.966 c-1.02,1.649-2.138,3.363-3.247,5.419c-0.932,1.728-2.344,3.967-2.598,5.88c-0.535,4.014,2.261,7.09,5.846,7.203 c3.583-0.113,6.379-3.189,5.845-7.203c-0.255-1.912-1.666-4.152-2.598-5.88C8.314,9.329,7.196,7.617,6.177,5.966z"></path>
</symbol>
</svg>
<svg height="90px" width="69px">
<use xlink:href="#codrops" class="codrops-1">
#shadow-root (user-agent)
<svg id="codrops" viewBox="0 0 23 30">
<path class="back" d="M22.63,18.261c-0.398-3.044-2.608-6.61-4.072-9.359c-1.74-3.271-3.492-5.994-5.089-8.62l0,0 c-1.599,2.623-3.75,6.117-5.487,9.385c0.391,0.718,0.495,1.011,0.889,1.816c0.143,0.294,0.535,1.111,0.696,1.43 c0.062-0.114,0.582-1.052,0.643-1.162c0.278-0.506,0.54-0.981,0.791-1.451c0.823-1.547,1.649-2.971,2.469-4.33 c0.817,1.359,1.646,2.783,2.468,4.33c0.249,0.47,0.513,0.946,0.791,1.453c1.203,2.187,2.698,4.906,2.96,6.895 c0.292,2.237-0.259,4.312-1.556,5.839c-1.171,1.376-2.824,2.179-4.663,2.263c-1.841-0.084-3.493-0.887-4.665-2.263 c-0.16-0.192-0.311-0.391-0.448-0.599c-0.543,0.221-1.127,0.346-1.735,0.365c-0.56-0.019-1.095-0.127-1.599-0.313 c1.448,3.406,4.667,5.66,8.447,5.78C19.086,29.537,23.469,24.645,22.63,18.261z"></path>
<path class="front" fill="currentColor" d="M6.177,11.659c0.212,0.367,0.424,0.747,0.635,1.136c0.164,0.303,0.333,0.606,0.512,0.927 c0.683,1.225,1.618,2.898,1.755,3.937c0.144,1.073-0.111,2.056-0.716,2.769c-0.543,0.641-1.315,1.014-2.186,1.067 c-0.87-0.054-1.643-0.43-2.186-1.067c-0.604-0.713-0.858-1.695-0.715-2.771c0.137-1.036,1.072-2.712,1.755-3.936 c0.18-0.32,0.349-0.623,0.513-0.927C5.752,12.404,5.964,12.026,6.177,11.659 M6.177,5.966L6.177,5.966 c-1.02,1.649-2.138,3.363-3.247,5.419c-0.932,1.728-2.344,3.967-2.598,5.88c-0.535,4.014,2.261,7.09,5.846,7.203 c3.583-0.113,6.379-3.189,5.845-7.203c-0.255-1.912-1.666-4.152-2.598-5.88C8.314,9.329,7.196,7.617,6.177,5.966z"></path>
html.-webkit-
body
svg



```
<use id="robot-1" xlink:href="#robot" ... />
<use id="robot-2" xlink:href="#robot" ... />
```

You can define the colors that make up your theme in the style sheet:

```
#robot-1 {
  --primary-color: #0099CC;
  --secondary-color: #FFDF34;
  --tertiary-color: #333;
}

#robot-2 {
  --primary-color: #E52A39;
  --secondary-color: #11EBD8;
  --tertiary-color: #000;
}
```

and then use those variables in the SVG:

```
<svg style="display: none">
    <symbol id="robot" viewBox="0 0 340 536">
        <path d="..." fill="#fff" />
        <path d="..." fill="#D1312C" />
        <path d="..." fill="#1E8F90" style="fill: var(--primary-color, #1E8F90)" />
        <path d="..." fill="#1E8F90" style="fill: var(--primary-color, #1E8F90)" />
        <path d="..." fill="#fff" />
        <path d="..." fill="#6A4933" style="fill: var(--tertiary-color, #6A4933)" />
        <path d="..." fill="#1E8F90" style="fill: var(--primary-color, #1E8F90)" />
        <path d="..." fill="#6A4933" style="fill: var(--tertiary-color, #6A4933)" />
        <path d="..." fill="#fff" />
        <path d="..." fill="#6A4933" style="fill: var(--tertiary-color, #6A4933)" />
        <path d="..." fill="#F2B42B" style="fill: var(--secondary-color, #F2B42B)" />
        <path d="..." fill="#fff" />
         <!-- rest of the shapes -->
    </symbol>
</svg>
```

Read all about styling the contents of use at: http://tympanus.net/codrops/2015/07/16/styling-svg-use-content-css/

“SVG Parameters are a way to set CSS custom properties on an "external" SVG image, by passing them through a special fragment scheme on the URL. This gives a limited, but powerful, subset of the customizability that "inline" SVG images have to "external" SVG images.”

Source: https://tabatkins.github.io/specs/svg-params/

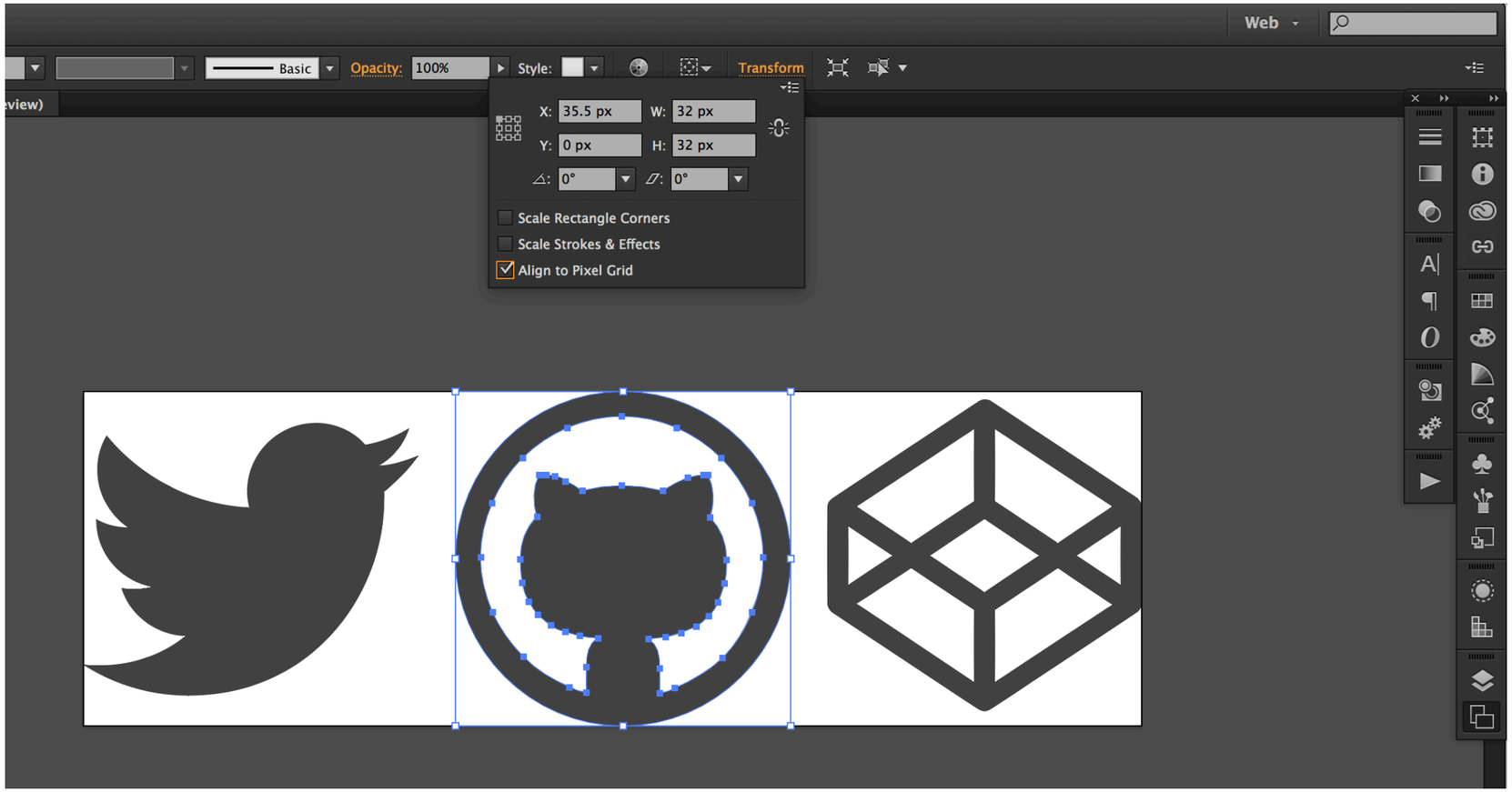
```
<img src="http://example.com/image.svg#param(--text-color%20blue)" alt=".." />`
```

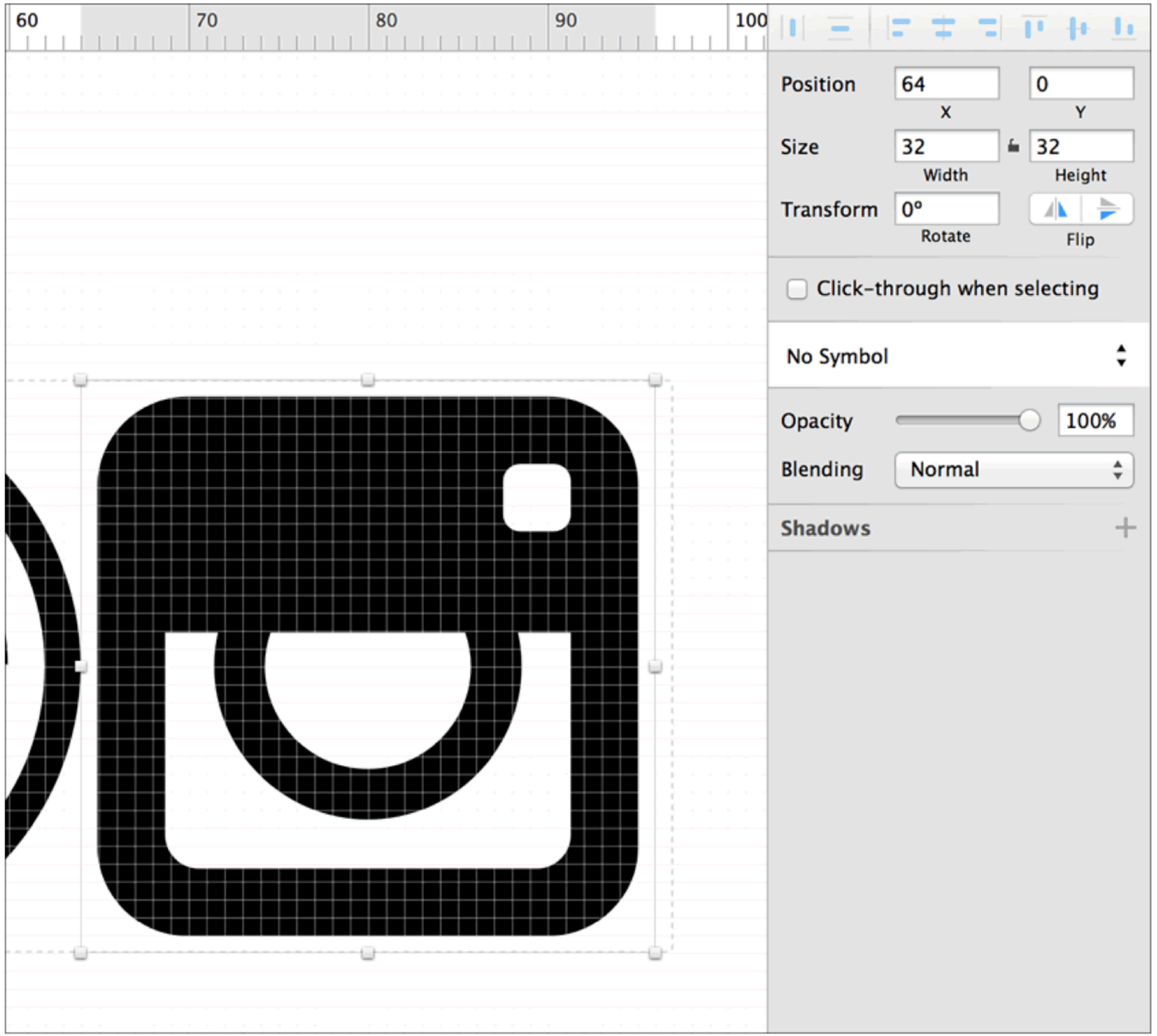
Spriting Technique #2

Using Fragment Identifiers

— A visual approach to spriting SVG (Similar to spriting using a PNG)
— Uses one sprite that contains all icons
— Uses the position and bounding box an icon inside the sprite
— to create a "view" for that icon
— You display each icon by referencing its view

To create a view, get the position and dimensions of the icon from the gaphics editor




60
70
80
90
100
Position
64
X
0
Y
Size
32
Width
32
Height
Transform
0°
Rotate
Flip
Click-through when selecting
No Symbol
Opacity
100%
Blending
Normal
Shadows

```
<img src='uiIcons.svg#svgView(viewBox(64, 0, 32, 32))' alt="..."/>
```

```
<view id='instagram-icon' viewBox='64 0 32 32' />

<img src='sprite.svg#instagram-icon' alt="Instagram icon" />
```

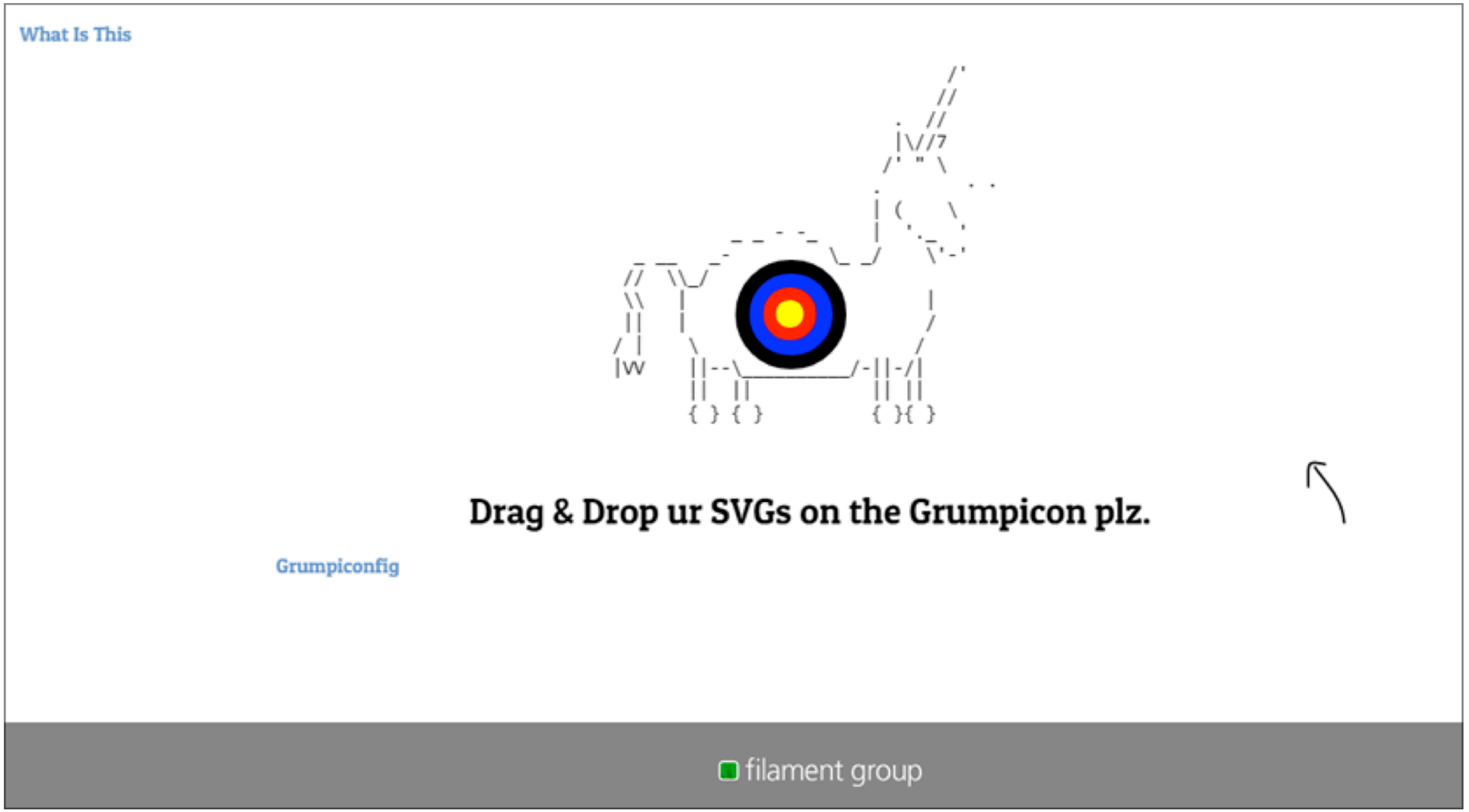
Important: *Remember to specify width and height for the* `<img>` *referencing the sprite views.*

Spriting Technique #3

Icons as background images in CSS

— The closest you can get to icon fonts, using SVG
— Support back to IE6
— Same animation capabilities as icon fonts, plus the advantages of SVG.
— There are tools to automate and speed up the spriting process.

Grunticon and/or Grumpicon for automation



SVG is embedded as data URI in CSS:

```
.twitter-icon{
    background-image: url('data:image/svg+xml;…');
    background-repeat: no-repeat;
    background-position: 50% 50%;
    height: 2em;
    width: 2em;
    /* etc. */
}
```

```
<span class="twitter-icon"></span>
```

A script takes care of loading the appropriate version for the different browsers.

Overview of all spriting techniques: https://24ways.org/2014/an-overview-of-svg-sprite-creation-techniques/

WHY would I want to use SVG over icon fonts?

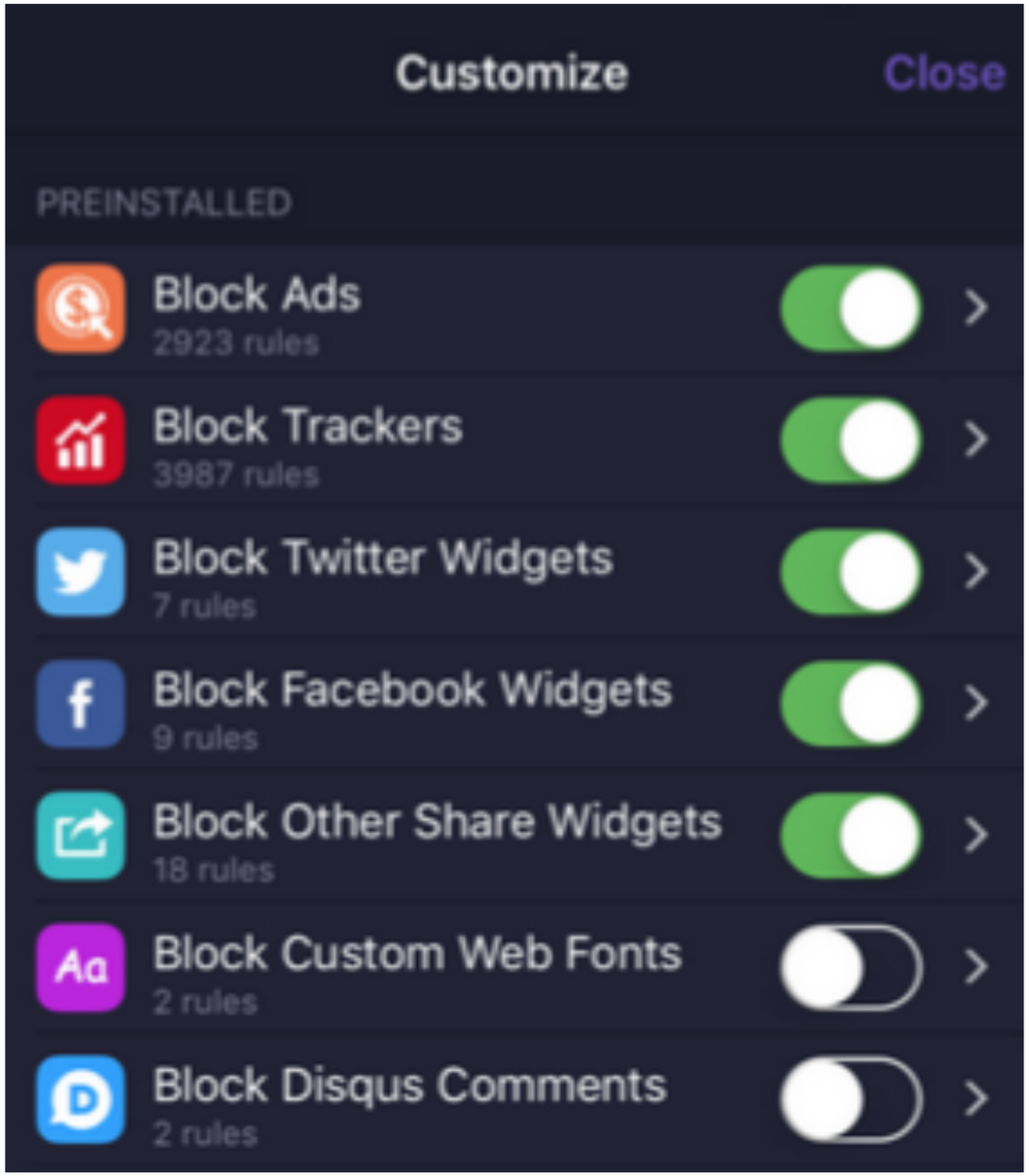
— Infinitely scalable & look crisp on all resolutions
— Styleable via CSS (multi-colors)
— Easier size control
— Interactive and animatable using CSS, SMIL and/or JavaScript
— Semantic markup
— Fully accessible (incl. text contained in an icon)
— Many tools available to make creation, interaction, embedding and spriting easier
— Different embedding and fallback techniques

Just moved @selz from icon fonts (woff2) over to SVG sprites. Saved 7.2% in CSS and 51.6% woff2 to SVG.

— Sam Potts

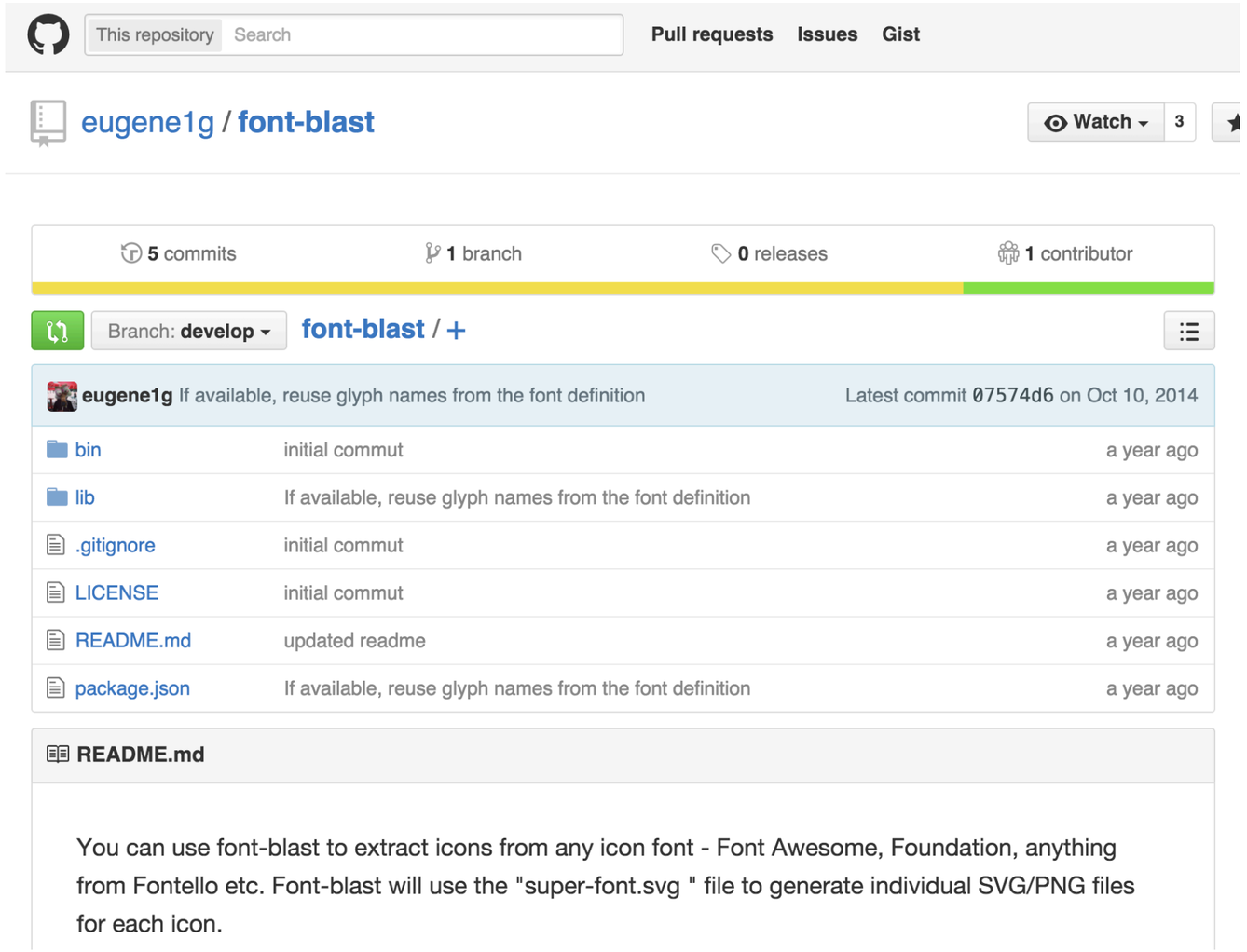
We don't download fonts; icons are what svg is for.

— Bruce 'Captain Wonderful' of Opera (a.k.a Bruce Lawson)


Customize
Close
PREINSTALLED
Block Ads
2923 rules
Block Trackers
3987 rules
Block Twitter Widgets
7 rules
Block Facebook Widgets
9 rules
Block Other Share Widgets
18 rules
Block Custom Web Fonts
2 rules
Block Disqus Comments
2 rules

If you still need convincing, read this: https://css-tricks.com/icon-fonts-vs-svg/

Want to make a switch? This tool might help you:



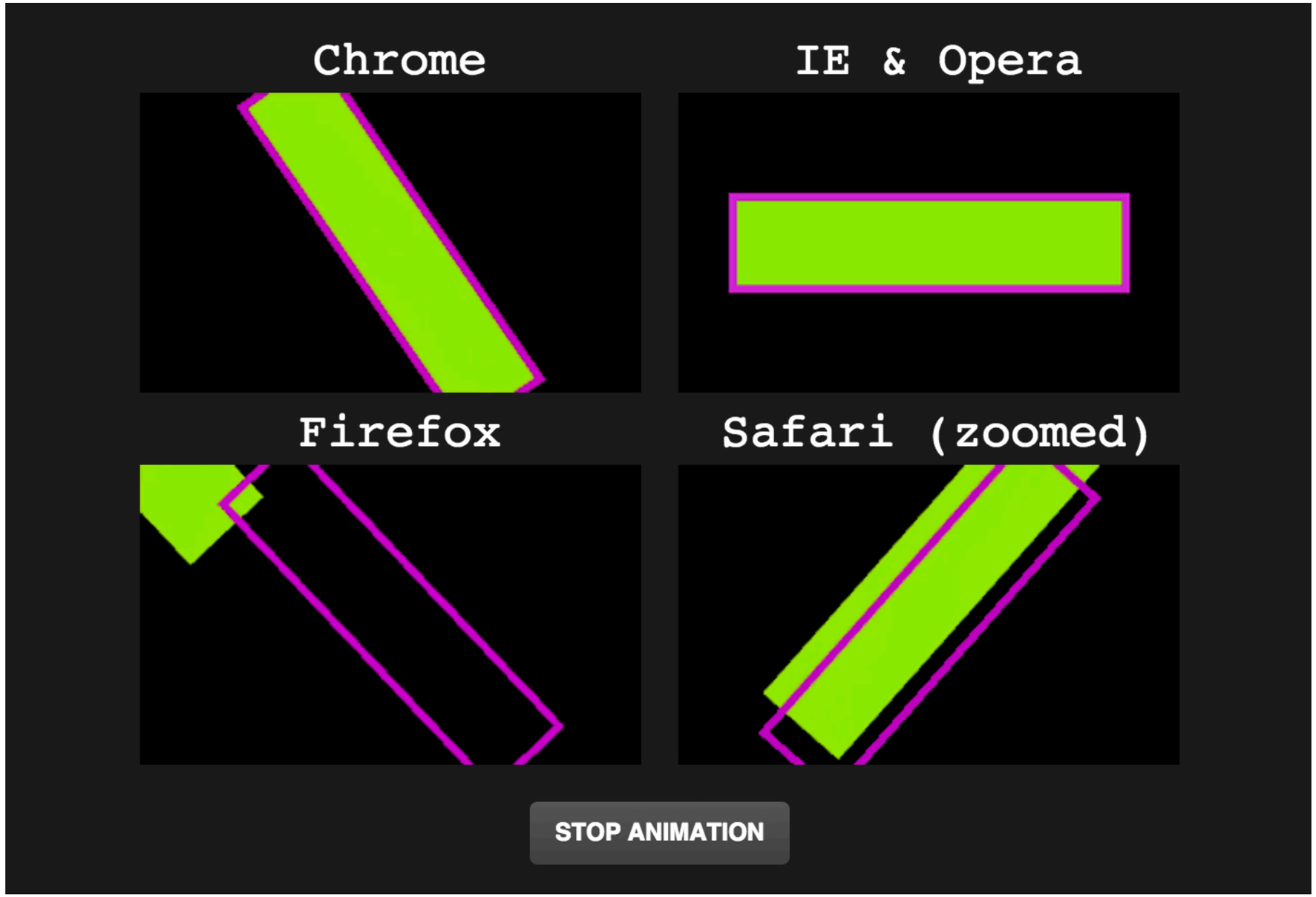
#nomoreexcuses

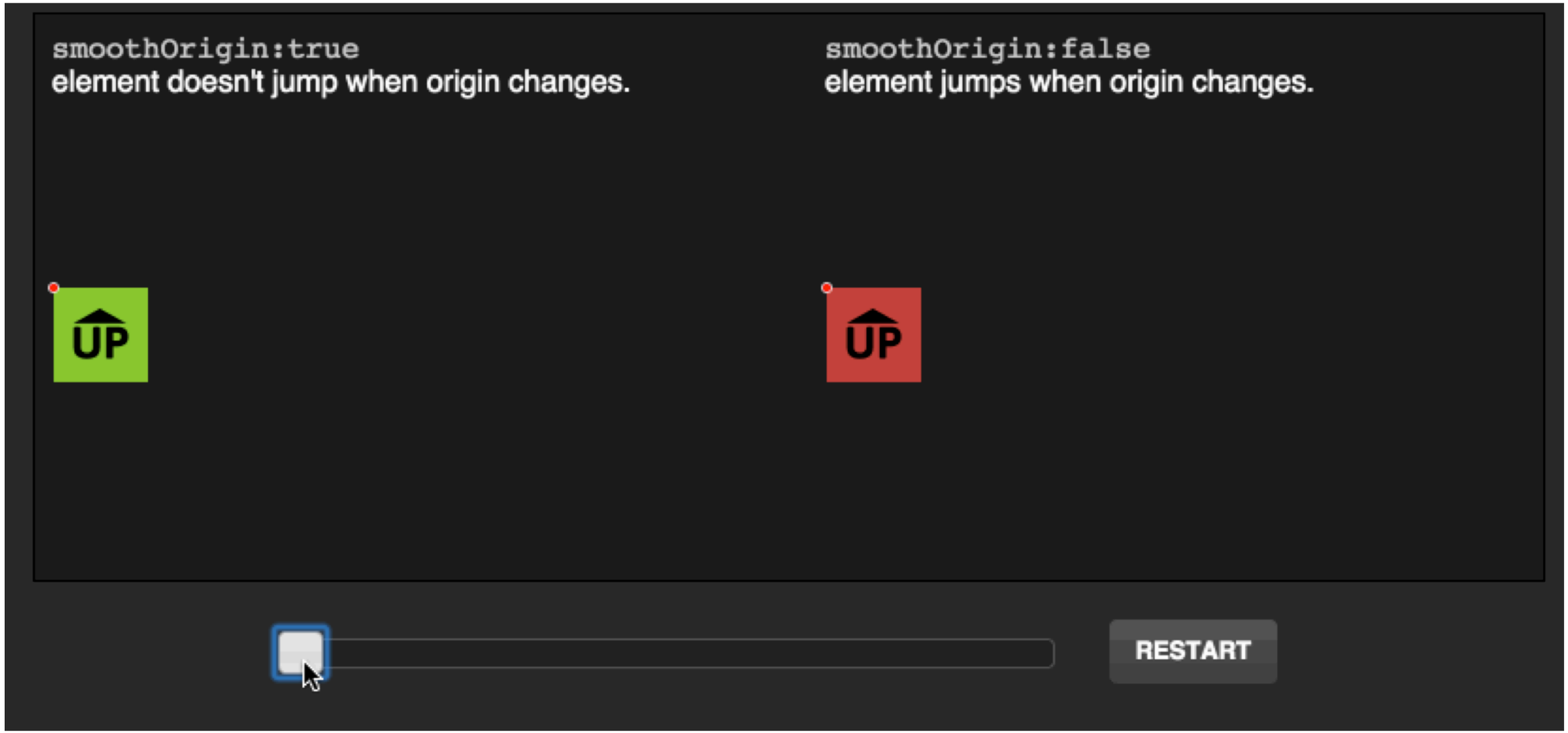
Animating SVG: SMIL, CSS or JavaScript?

— Don’t use SMIL.
— Use CSS only for simple animations.
— Use JavaScript for complex animations.

Popular SVG JavaScript animation libraries:

1. GreenSock Animation Platform (GSAP)
2. Snap.svg (“*The jQuery of SVG*”)
3. Velocity.js
4. D3.js


Chrome
IE & Opera
Firefox
Safari (zoomed)
STOP ANIMATION


smoothOrigin:true
element doesn't jump when origin changes.
smoothOrigin:false
element jumps when origin changes.
UP
UP
RESTART

Read more here: http://greensock.com/svg-tips

Which Embedding Technique Should You Choose?

1.
```
<img src="mySVG.svg" alt="..."/>
```

2.
```
background-image: url(path/to/mySVG.svg);
```

3.
```
<object type="image/svg+xml" data="mySVG.svg">
   <!-- fallback here -->
</object>
```

4.
```
<iframe src="mySVG.svg">
   <!-- fallback here -->
</iframe>
```

5.
```
<svg xmlns="http://www.w3.org/2000/svg" …>
   <!-- svg content -->
</svg>
```

6.
```
<picture>
   <source type="image/svg+xml" srcset="path/to/image.svg">
   <img src="path/to/fallback.png" alt="Image description">
</picture>
```

Filter your options down; set priorities (and compromises when needed).

- Is the SVG animated?
- Is it interactive?
- What kind of animation? (Does it require JS?)
- What browser support do I need (for the animation)?
- What kind of content and fallback do you need? (e.g. infographics)

Embedding Technique	Animations	Interactions?
<img>	CSS (, SMIL)	No
url();	CSS (, SMIL)	No
<picture>	CSS (, SMIL)	No
<iframe>	CSS, JavaScript (, SMIL)	Yes
<object>	CSS, JavaScript (, SMIL)	Yes
<svg>	CSS, JavaScript (, SMIL)	Yes

Embedding Technique	CSS Animations Location	JS Animations Location
<img>	Inside <svg>	-
url();	Inside <svg>	-
<picture>	Inside <svg>	-
<iframe>	Inside <svg>	Anywhere
<object>	Inside <svg>	Anywhere
<svg>	Anywhere	Anywhere

Why I love `<object>`

— Enables modularity of content without sacrificing modularity and reusability of styles/animations

— Scriptable, animatable, interactive

— Cacheable resource

— Default fallback mechanism

— Fallback can be image *or* text, and even both.

— All the benefits of using SVG: accessible content, searchable and selectable text, interactive shapes, etc.

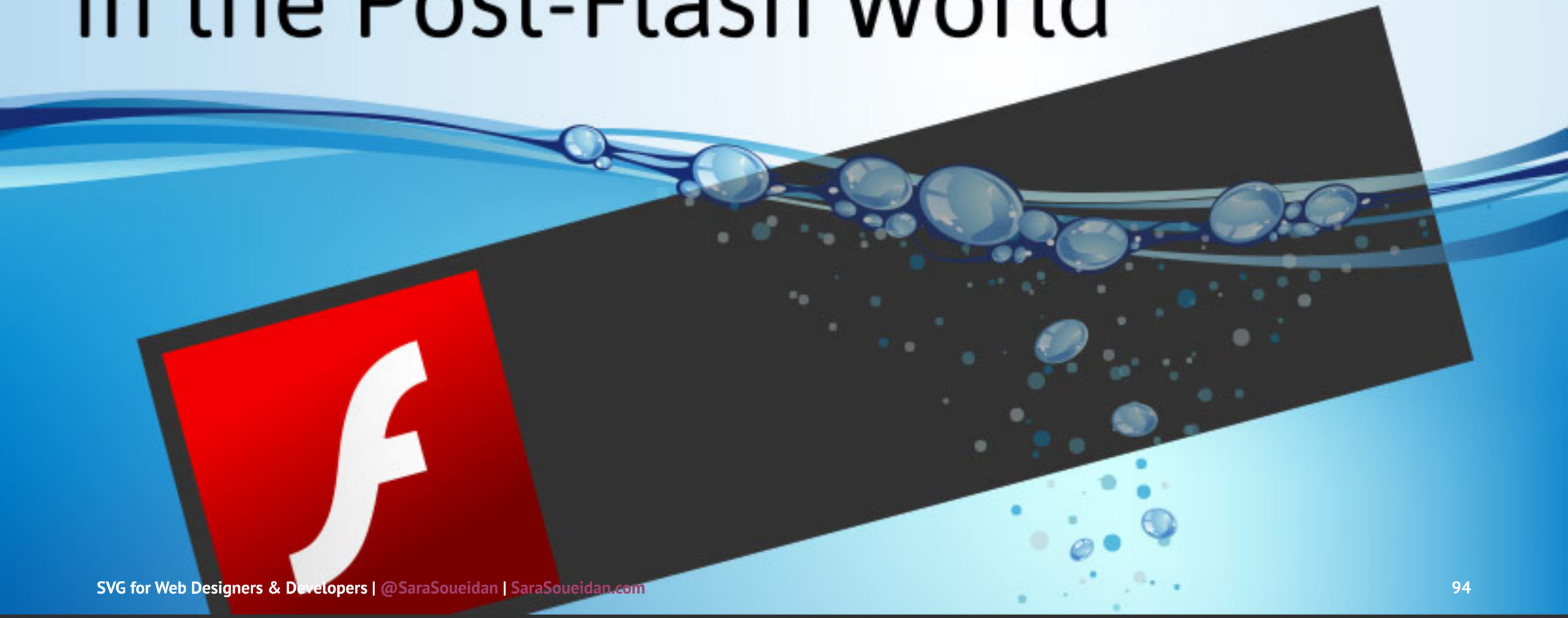
Practical Example #1: Ad Banners

— The banner content including scripts and animations are "encapsulated" and easily reusable

— Separation of concerns

— Plug `<object>` in and play, while scripts and interactions are handled and maintained using JavaScript.

Solutions for Banner Ads
in the Post-Flash World



Recommended reading: http://codepen.io/chrisgannon/blog/my-first-svg-banner-ad

Practical Example #2: Infographics

- Infographic is animated and can be interactive
- Best kind of fallback is text content of the data presented in the graphic, goes between opening and closing `<object>` tags.

Image source: http://provatahealth.com

Credits

— Geometric background designed by Freepik.
— GSAP Screenshots by GreenSock.

Thank you!