CSS+SVG: A Designer's Delight

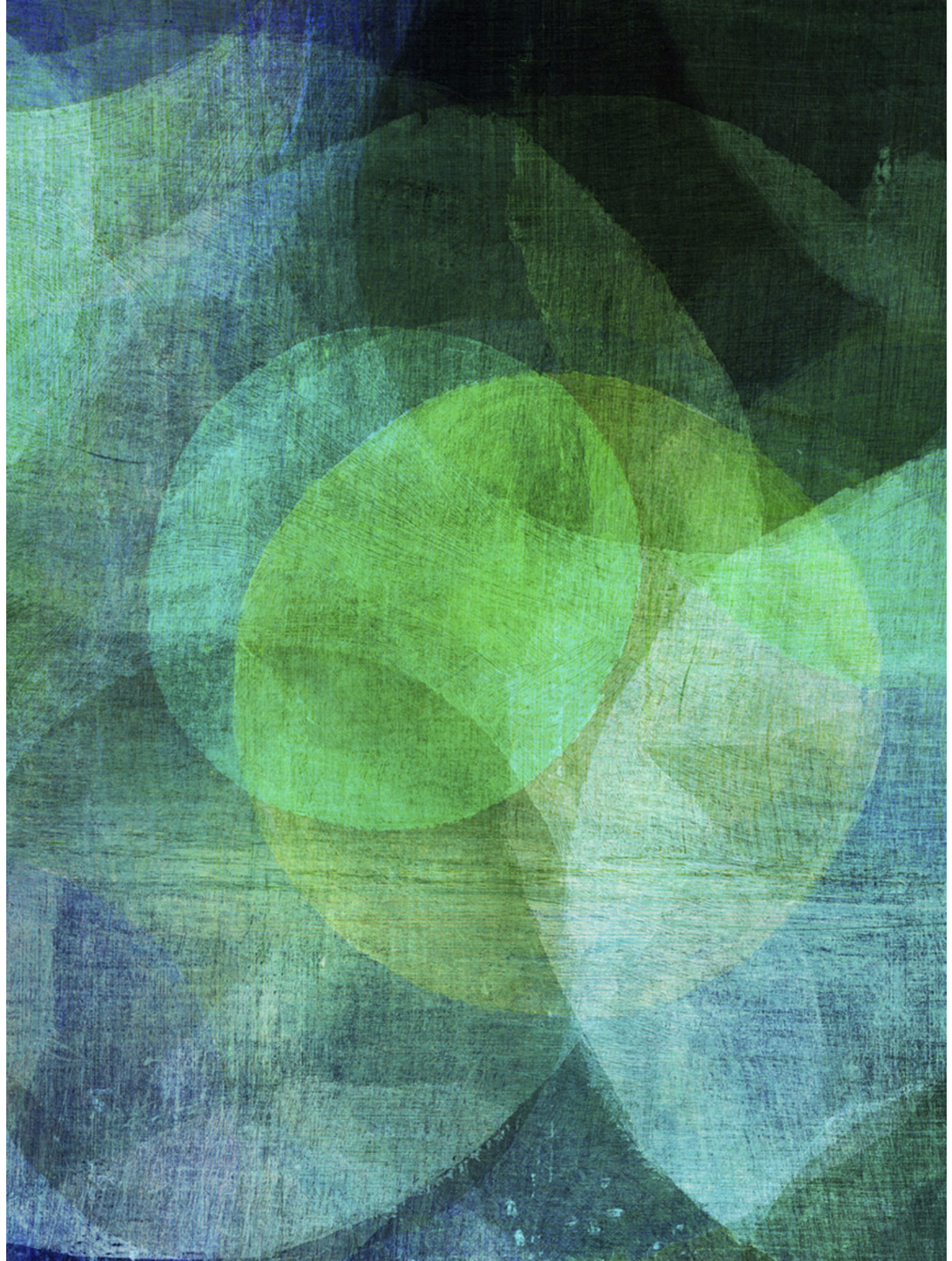
HELLO

- Freelance front-end developer from Lebanon 🇱🇧
- It’s pronounced ‘Swaïdaan’
- Co-author: Smashing Book 5
- Author: Codrops CSS Reference
- SVG Advocate.

SVG can be used to extend the capabilities of CSS

How do you use SVG?

SVG in Living Style Guides: Iconography and Brand Identity



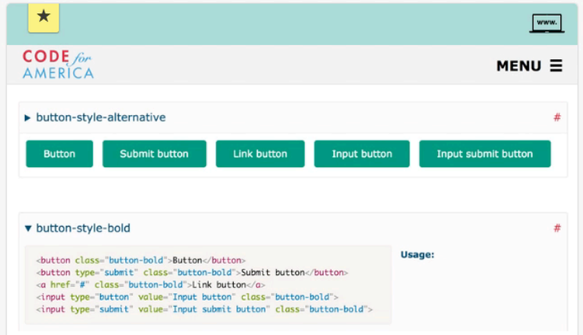
Examples

Real life pattern libraries, code standards documents and content style guides

Filters

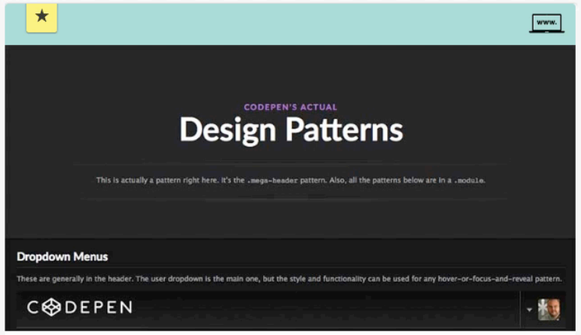
Reset All Filters

Backend | Branding | Code | Design Language | Frontend | Jekyll | Patterns | Voice And Tone | Writing



Code For America

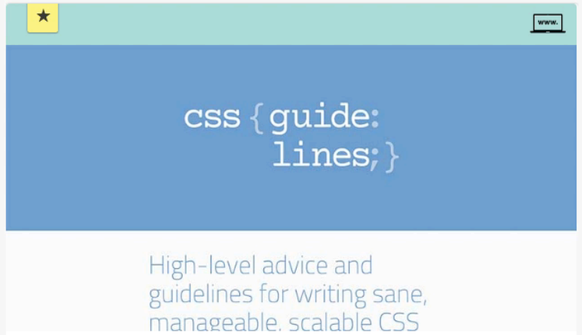
FRONTEND # PATTERNS



Codepen's design patterns

The design patterns used in Codepen

PATTERNS

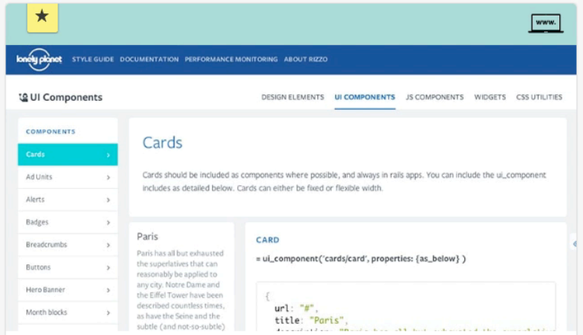


Harry Roberts's CSS Guidelines

High-level advice and guidelines for writing sane, manageable, scalable CSS

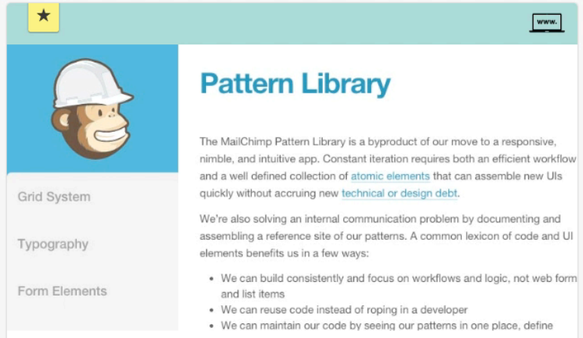
Harry Roberts

CODE



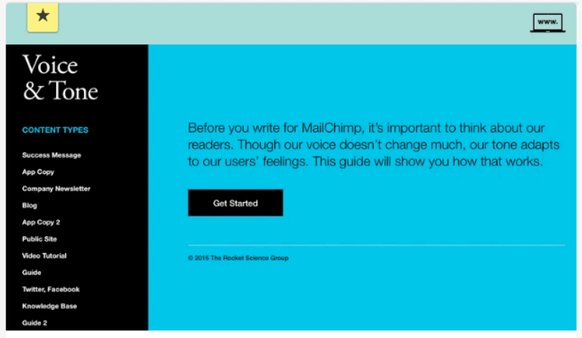
Lonely Planet

FRONTEND # PATTERNS



Mailchimp's Pattern Library

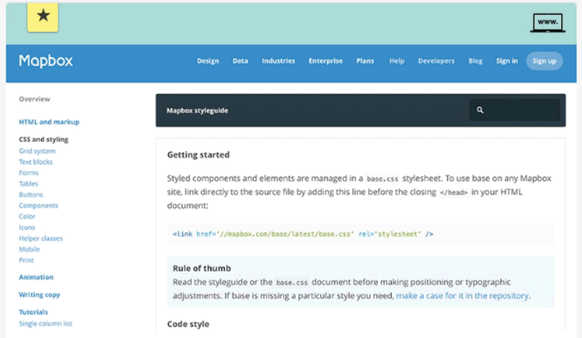
FRONTEND # PATTERNS



Mailchimp's Voice and Tone

Mailchimp

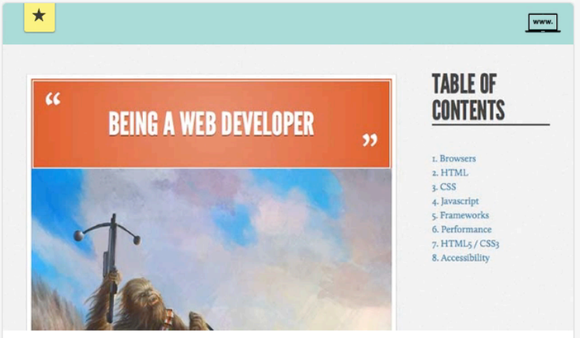
VOICEANDTONE # WRITING # DESIGNLANGUAGE



Mapbox

An extensive guide featuring HTML and CSS patterns, a writing guide, and tutorials for implementation.

FRONTEND # CODE # WRITING # PATTERNS

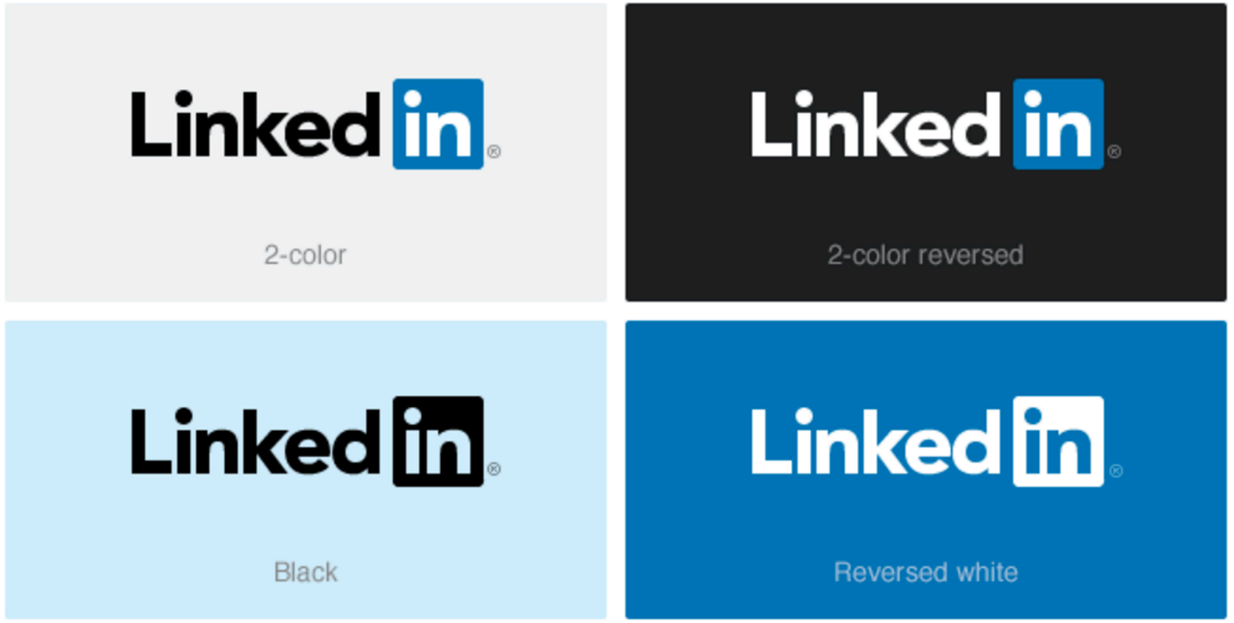


Mark Brown's Standards Guidelines

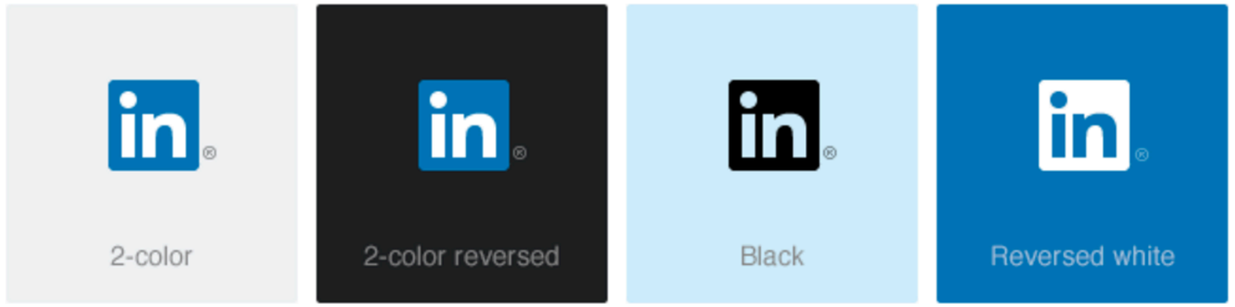
To aid a shared understanding of front-end development best practice at PUP.

Mark Brown

CODE # FRONTEND



The same rules apply to the usage of our [in] mark. A number of logo and [in] box assets in different file formats and sizes are available for download. Never attempt to recreate the logo – always use the artwork provided and adhere to the usage guidelines.

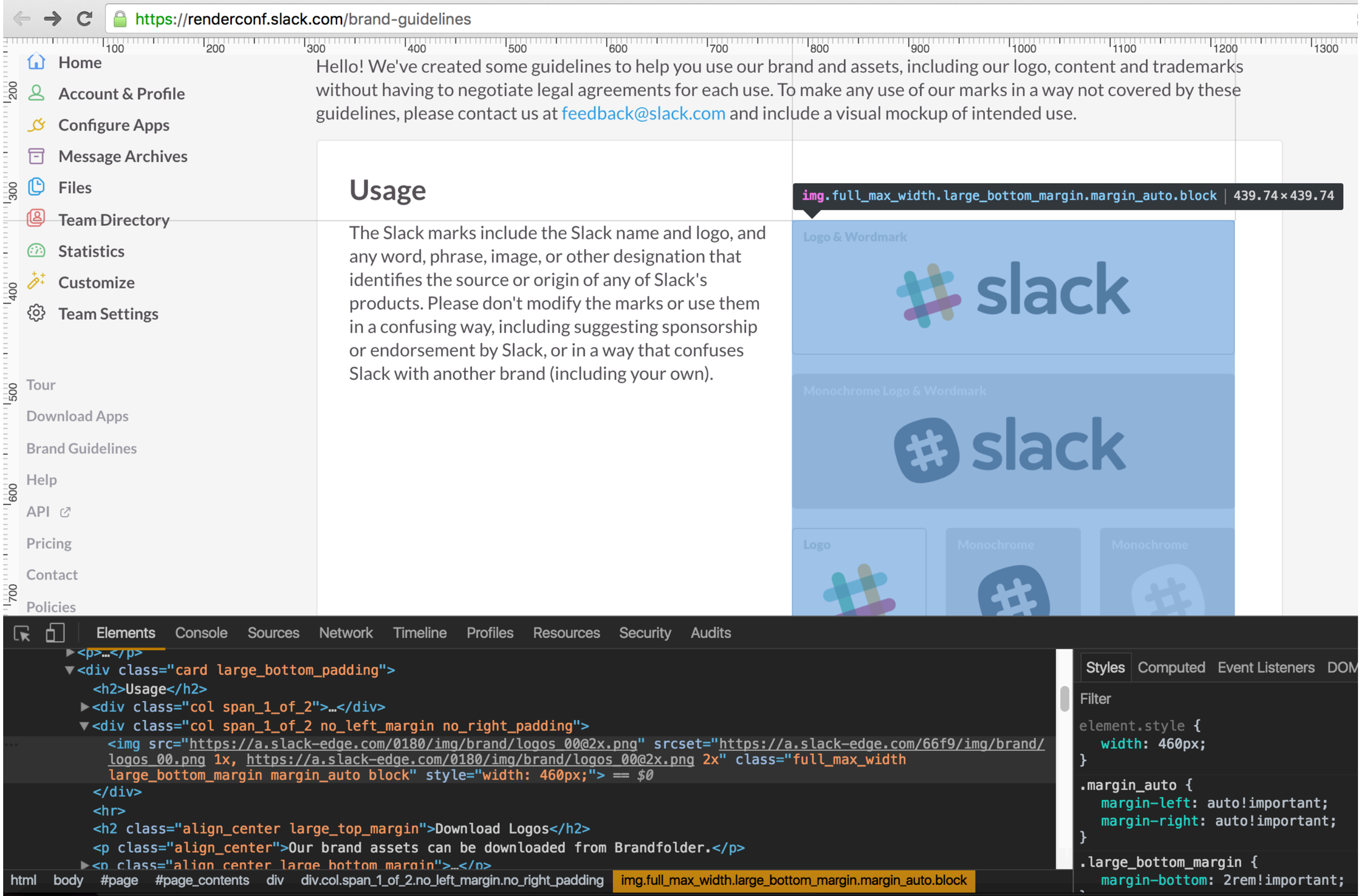


Brand Guidelines

Hello! We've created some guidelines to help you use our brand and assets, including our logo, content and trademarks without having to negotiate legal agreements for each use. To make any use of our marks in a way not covered by these guidelines, please contact us at feedback@slack.com and include a visual mockup of intended use.

Usage

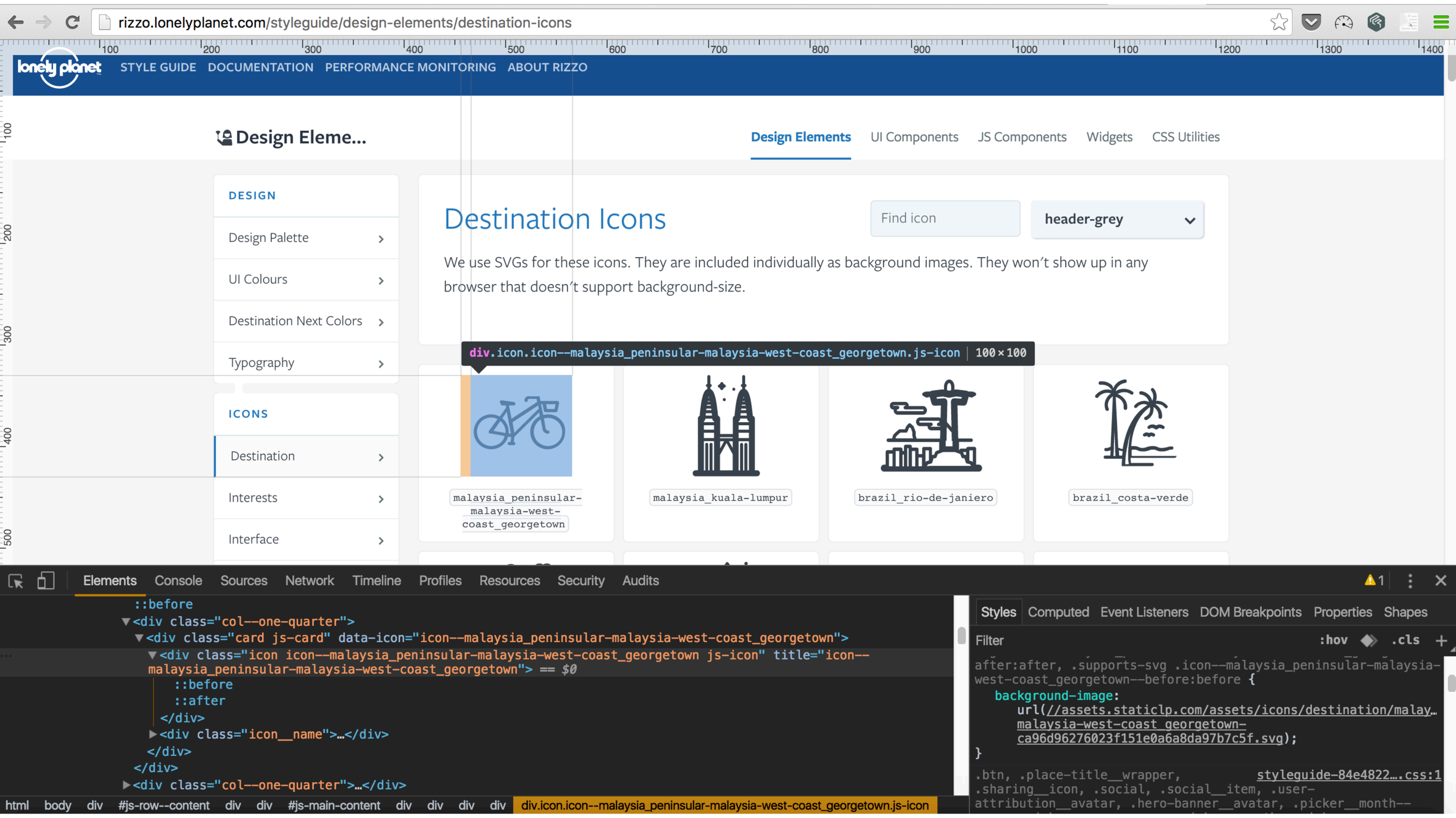
The Slack marks include the Slack name and logo, and any word, phrase, image, or other designation that identifies the source or origin of any of Slack's products. Please don't modify the marks or use them in a confusing way, including suggesting sponsorship or endorsement by Slack, or in a way that confuses Slack with another brand (including your own).


https://renderconf.slack.com/brand-guidelines
Home
Account & Profile
Configure Apps
Message Archives
Files
Team Directory
Statistics
Customize
Team Settings
Tour
Download Apps
Brand Guidelines
Help
API
Pricing
Contact
Policies
Hello! We've created some guidelines to help you use our brand and assets, including our logo, content and trademarks without having to negotiate legal agreements for each use. To make any use of our marks in a way not covered by these guidelines, please contact us at feedback@slack.com and include a visual mockup of intended use.
Usage
The Slack marks include the Slack name and logo, and any word, phrase, image, or other designation that identifies the source or origin of any of Slack's products. Please don't modify the marks or use them in a confusing way, including suggesting sponsorship or endorsement by Slack, or in a way that confuses Slack with another brand (including your own).
img.full_max_width.large_bottom_margin.margin_auto.block | 439.74×439.74
Logo & Wordmark
slack
Monochrome Logo & Wordmark
slack
Logo
Monochrome
Monochrome
Elements Console Sources Network Timeline Profiles Resources Security Audits
…

<h2>Usage</h2>
…

<img src="https://a.slack-edge.com/0180/img/brand/logos_00@2x.png" srcset="https://a.slack-edge.com/66f9/img/brand/logos_00.png 1x, https://a.slack-edge.com/0180/img/brand/logos_00@2x.png 2x" class="full_max_width large_bottom_margin margin_auto block" style="width: 460px;"> == $0

<hr>
<h2 class="align_center large_top_margin">Download Logos</h2>
Our brand assets can be downloaded from Brandfolder.
…
Styles Computed Event Listeners DOM
Filter
element.style {
width: 460px;
}
.margin_auto {
margin-left: auto!important;
margin-right: auto!important;
}
.large_bottom_margin {
margin-bottom: 2rem!important;
html body #page #page_contents div div.col.span_1_of_2.no_left_margin.no_right_padding img.full_max_width.large_bottom_margin.margin_auto.block


rizzo.lonelyplanet.com/styleguide/design-elements/destination-icons
lonely planet
STYLE GUIDE
DOCUMENTATION
PERFORMANCE MONITORING
ABOUT RIZZO
Design Eleme...
Design Elements
UI Components
JS Components
Widgets
CSS Utilities
DESIGN
Design Palette
UI Colours
Destination Next Colors
Typography
ICONS
Destination
Interests
Interface
Destination Icons
Find icon
header-grey
We use SVGs for these icons. They are included individually as background images. They won't show up in any browser that doesn't support background-size.
div.icon.icon--malaysia_peninsular-malaysia-west-coast_georgetown.js-icon | 100×100
malaysia_peninsular-malaysia-west-coast_georgetown
malaysia_kuala-lumpur
brazil_rio-de-janiero
brazil_costa-verde
Elements
Console
Sources
Network
Timeline
Profiles
Resources
Security
Audits
::before

 == $0
::before
::after

…

…
Styles
Computed
Event Listeners
DOM Breakpoints
Properties
Shapes
Filter
:hov
.cls
after:after, .supports-svg .icon--malaysia_peninsular-malaysia-west-coast_georgetown--before:before {
background-image:
url(//assets.staticlp.com/assets/icons/destination/malay…malaysia-west-coast_georgetown-ca96d96276023f151e0a6a8da97b7c5f.svg);
}
.btn, .place-title__wrapper, .sharing__icon, .social, .social__item, .user-attribution__avatar, .hero-banner__avatar, .picker__month--
styleguide-84e4822….css:1
html
body
div
#js-row--content
div
div
#js-main-content
div
div
div
div
div.icon.icon--malaysia_peninsular-malaysia-west-coast_georgetown.js-icon

For every icon, a different sprite is used.

The specific icon is shown by changing the position of the background image in the `<div>`.

To change any icon in the sprite, you need to open the sprite and style the elements in the sprite, and then when the sprite is loaded the changes will appear.

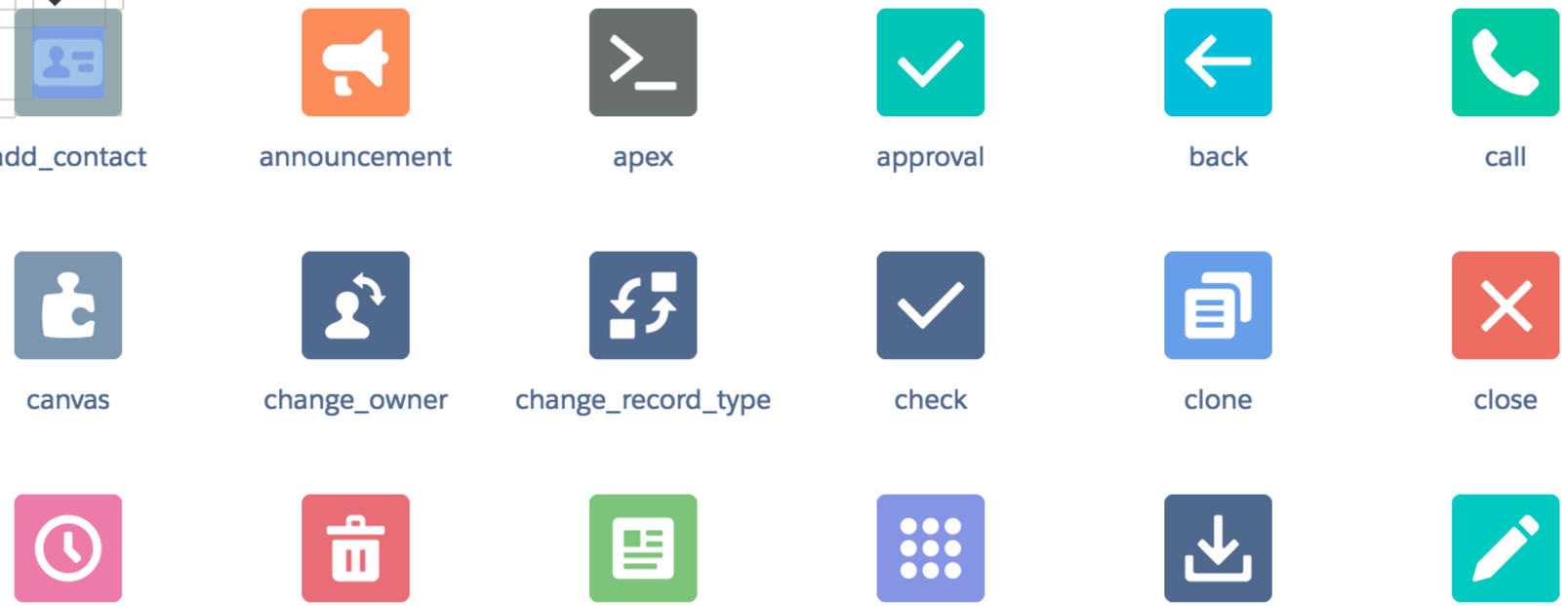
There is a better, more flexible way.

Getting Started
Design
Components
Voice and Tone
Native
Resources
Downloads
Icons
Design Tokens
Links
FAQ

component page. If you are building a Lightning Component, you may require an additional Lightning helper component to use SVGs.

Action Icons

svg.slds-icon.slds-icon-action-add-contact.slds-icon--large.slds-p-around--x-small | 48 × 48



Elements Console Sources Network Timeline Profiles Resources Security Audits

```
<figure>
  <span class="slds-icon_container slds-icon-action-add-contact">
    <svg aria-hidden="true" class="slds-icon slds-icon-action-add-contact slds-icon--large slds-p-around--x-small"> == $0
      <use xlink:href="/assets/icons/action-sprite/svg/symbols.svg#add_contact">
        #shadow-root (user-agent)
          <svg viewBox="0 0 24 24" id="add_contact" width="100%" height="100%">
            <path d="M21.2 4.2H2.8C1.5 4.2.5 5.2.5 6.5v11c0 1.3 1 2.3 2.3 2.3h18.4c1.3 0 2.3-1 2.3-2.3v-11c0-1.3-1-2.3-2.3-2.3zm-9.8 13H4.8c-.7 0-1.3-.8-1.3-1.6.1-1.2 1.3-1.8 2.5-2.4.9-.4 1-.7 1-1.1
```

Styles Computed Event L

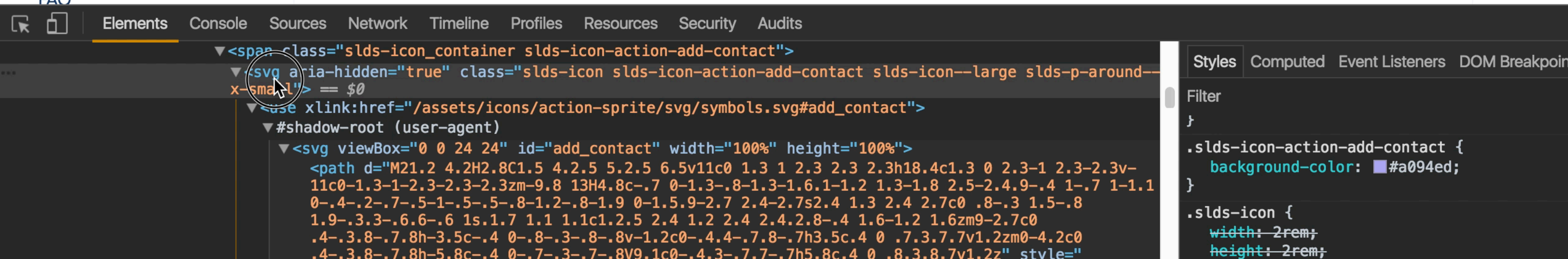
Filter

```
element.style {
}
svg:not(:root) {
  overflow: hidden;
}
```

Choose the icon you need from this page, then follow the implementation instructions on the icon component page. If you are building a Lightning Component, you may require an additional Lightning helper component to use SVGs.

Action Icons

add_contact
announcement
apex
approval
back
call
canvas
change_owner
change_record_type
check
clone
close



So how does it work?

Define the logo inside a `<symbol>` element (the template):

```
<symbol id="logo" viewBox="0 0 100 100">
  <path ...></path>
  <circle ...></circle>
</symbol>
```

then render the logo variations anywhere on the page w/ `<use>`:

```
<use xlink:href="#logo" class="logo--default"></use>

<use xlink:href="#logo" class="logo--light-on-dark"></use>

<use xlink:href="#logo" class="logo--dark-on-light"></use>
```

Using CSS, every instance of the logo can be styled differently.

So multiple **live variations** or ‘themes’ can be created to be used in different contexts.

Need to change the colors for one of the variations? Just change the CSS.

Quick Overview of SVG Structuring, Grouping and Referencing (Re-using) Elements in SVG

There are four main elements in SVG that are used to define, structure and reference SVG code within the document:

`<g>`

`<defs>`

`<symbol>`

`<use>`

The `<g>` element (short for “group”) is used for logically grouping together sets of related graphical elements.

Grouping elements together is useful for when you want to apply a style and have that style be inherited by all the elements in the group, and is particularly useful for when you want to animate a group of elements while maintaining their spatial relationships with each other.

The `<defs>` element is used to *define* elements that you want to reuse later.

Defining elements with <defs> is useful for when you want to create sort of a “template” that you want to use multiple times throughout the document. Elements defined inside a <defs> element are not rendered on the canvas except if you “call” them somewhere in the document.

The `<symbol>` element combines the benefits of both the `<defs>` and the `<g>` elements.

The `<symbol>` element has an important advantage over the other two elements: it accepts a `viewBox` attribute which allows it to scale-to-fit inside any viewport it is rendered in.

```
<symbol id="logo" viewBox="0 0 300 300">
  <!-- contents -->
</symbol>
```

The `<use>` element is used to ‘instantiate’ the ‘templates’ defined in `<symbol>` and `<defs>`.

So, it can be used to display an element or a group of elements multiple times on the same page. It is similar to the copy+paste functionality of the graphics editor. In fact, the ‘instances’ or ‘copies’ of an element are *live* copies.

```
<use xlink:href="#logo"></use>
```

Define the logo inside a `<symbol>` element (the template):

```
<symbol id="logo" viewBox="0 0 100 100">
  <path ...></path>
  <circle ...></circle>
</symbol>
```

then render the logo variations anywhere on the page w/ `<use>`:

```
<use xlink:href="#logo" class="logo--default"></use>

<use xlink:href="#logo" class="logo--light-on-dark"></use>

<use xlink:href="#logo" class="logo--dark-on-light"></use>
```

But the `<use>` element is empty...?

The contents of <use> are cloned into a shadow DOM that cannot be accessed by CSS selectors.

That is, you can't do this:

```
use path {
   fill: white;
}
use circle {
   fill: black;
}
```

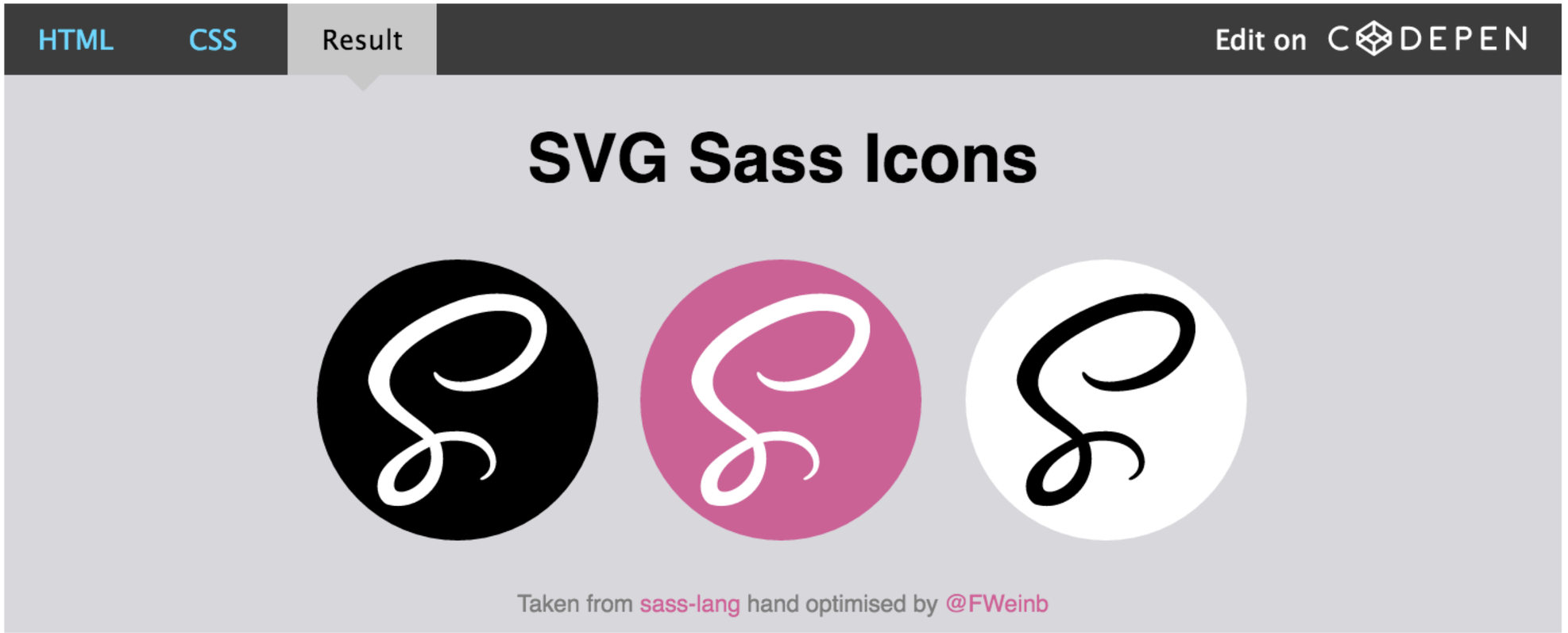
So, how exactly do you style the `<use>`d instances?

About CSS inheritance in SVG:

- Styles applied to a <g> or <use> are inherited by all their content.
- If an element has a style applied using a presentation attribute, the value in the attribute overrides the styles inherited from the container.
- Presentation attribute styles are overridden by every other style delcaration in CSS.

So, let’s put that to practice.

If you set a fill color on the ‹use›, both the path & the circle will inherit that color. But what you want is this:



This is where CSS Variables come in.

When using CSS variables to style SVG images, it's best if you start planning for the code before you actually write the code.

Determine what elements will change color based on different contexts, and then use CSS Variables as values for these elements.

Then, define the colors for these variables in the CSS.

```
use {
  fill: white;
  color: black;
}
```

```
<symbol viewBox="0 0 50 50" id="sass-logo">
  <circle cx=".." cy=".." r=".."></circle>
  <path d="..." fill="currentColor"></path>
</symbol>
```

Style other variations similarly by specifying different color values per class:

```
use.sass--black-on-white {
  fill: white;
  color: black;
}

use.sass--white-on-black {
  fill: black;
  color: white;
}

use.sass--white-on-pink {
  fill: #CE6499;
  color: white;
}
```

When you have more elements, you need more variables...

```
<symbol id="logo" viewBox="0 0 150 100">
    <path d="..." fill="#fff" />
    <path d="..." fill="#D1312C" />
    <path d="..." fill="#1E8F90" style="fill: var(--primary-color)" />
    <path d="..." fill="#1E8F90" style="fill: var(--primary-color)" />
    <path d="..." fill="#fff" />
    <path d="..." fill="#6A4933" style="fill: var(--tertiary-color)" />
    <path d="..." fill="#1E8F90" style="fill: var(--primary-color)" />
    <path d="..." fill="#6A4933" style="fill: var(--tertiary-color)" />
    <path d="..." fill="#fff" />
    <path d="..." fill="#6A4933" style="fill: var(--tertiary-color)" />
    <path d="..." fill="#F2B42B" style="fill: var(--secondary-color)" />
    <path d="..." fill="#fff" />

    <!-- rest of the shapes -->
</symbol>
```

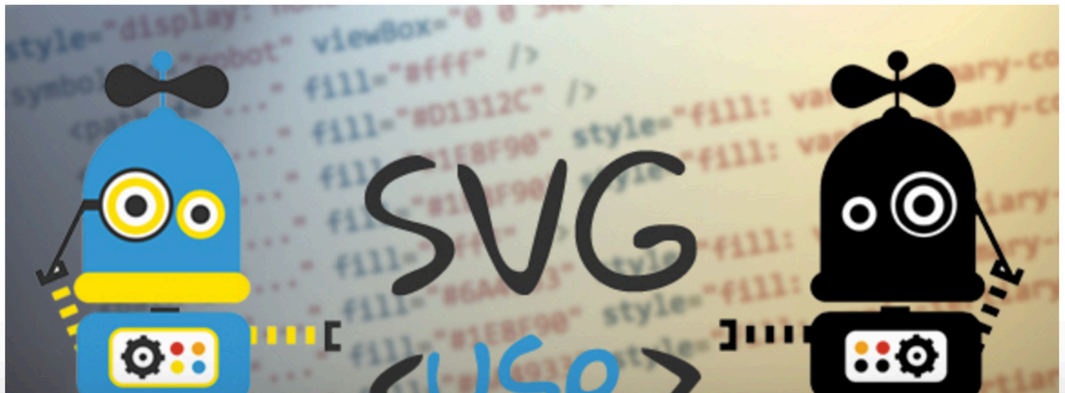
```
<use xlink:href="#logo" class="logo--light-on-dark"/>
<use xlink:href="#logo" class="logo--dark-on-light"/>
<use xlink:href="#logo" class="logo--default"/>

.logo--light-on-dark {
  --primary-color: #009966;
  --secondary-color: deepPink;
  --tertiary-color: #eee;


.logo--dark-on-light {
  --primary-color: black;
  --secondary-color: #eee;
  --tertiary-color: white;
}

.logo--default {
  --primary-color: ...;
  --secondary-color: ...;
  --tertiary-color: ...;
}
```

Using live instances of a logo in a style guide enables you to edit the styles directly in the guide, without having to recreate the entire image, making future maintainance easier for both designers and developers, because the main logo is there, and the variations can be controlled directly from the CSS.

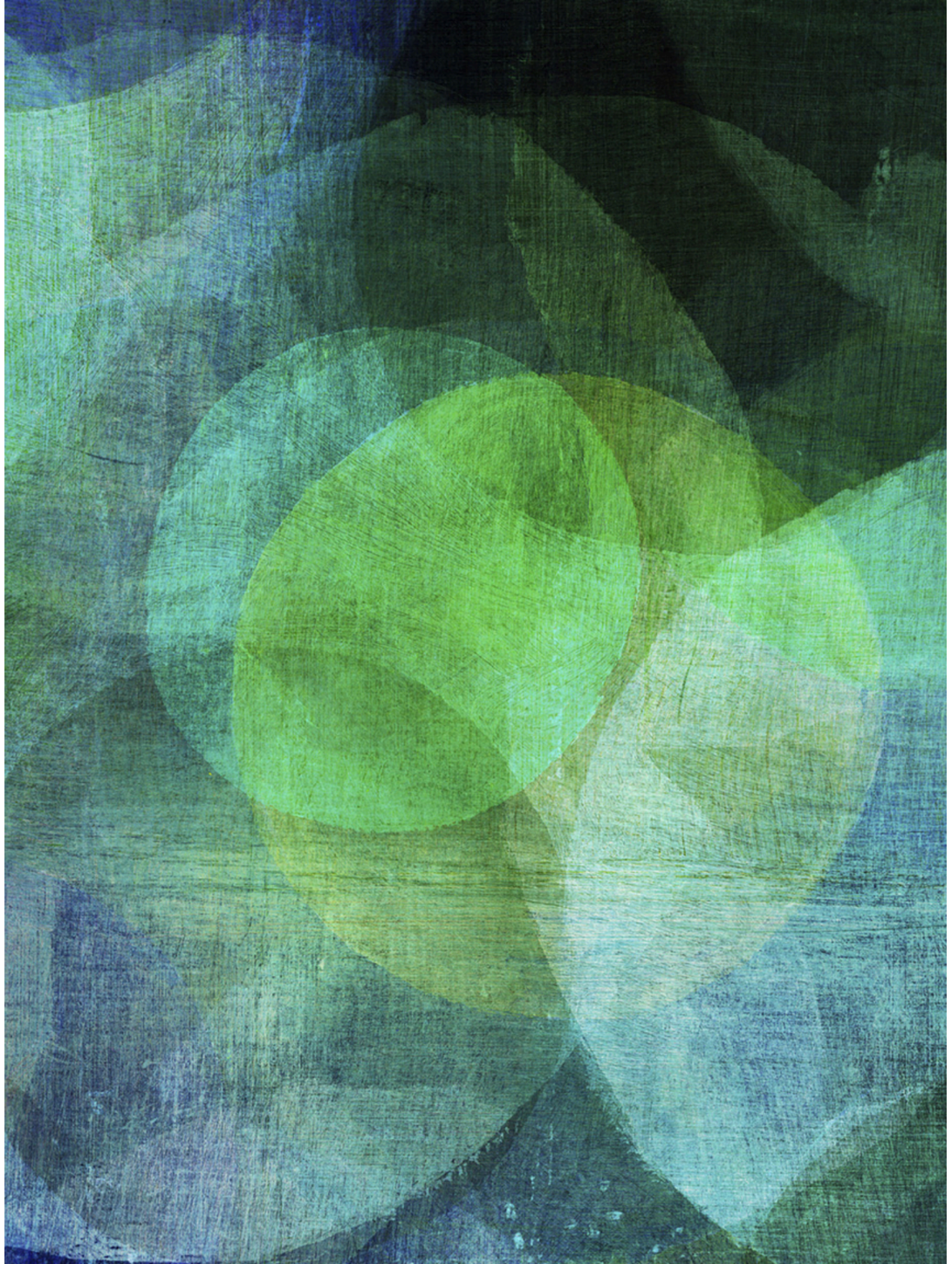
Sara Soueidan in Articles
2015/07/16 · 37 Comments

Styling SVG <use> Content with CSS

An in-depth article on how to style the contents of the SVG <use> element and overcome some challenges it brings.

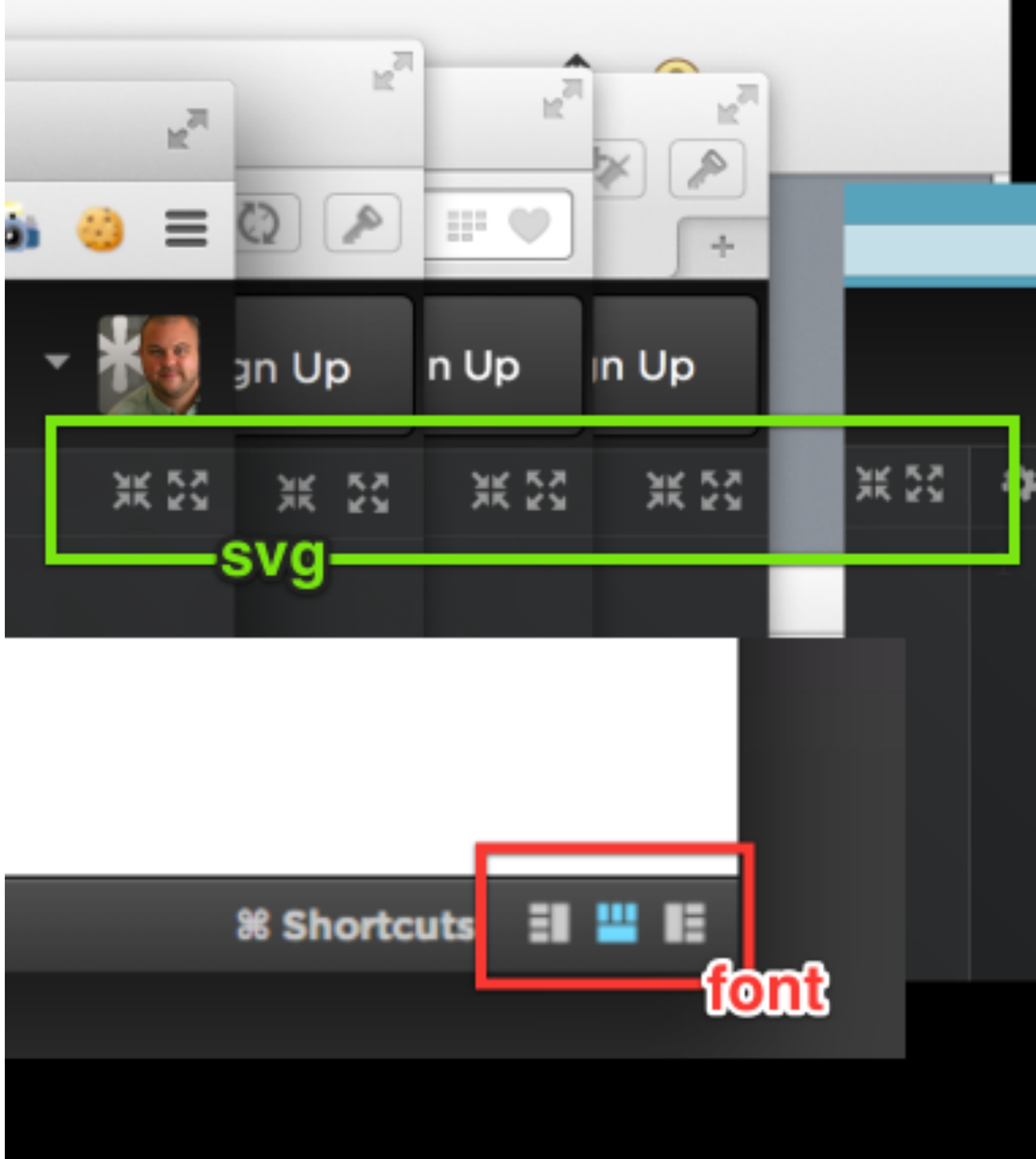


Pixel-Perfect, Flexible Iconography



SVG vs Icon Fonts #1

- SVG icons are resizable up and down without losing quality, extra sharp on retina displays, and small file size among them
- Browsers consider icon fonts as text, so the icons are anti-aliased as such. Can lead to icons not being as sharp as you might expect.



SVG vs Icon Fonts #2

- While icon fonts can be controlled using CSS font styling properties, SVG icons can be controlled the same way, with extra advantages:
    1. You can control individual parts of a multi-part icon and
    2. You can use SVG-specific CSS like stroke properties.

This enables you to create multi-colored icons with SVG, which is not possible with icon fonts without hacks.

SVG vs Icon Fonts #3

- The size and position of an SVG is predictable and behaves just as you'd expect it to.
- It can be frustrating to position a font icon. The icons are inserted via pseudo element, and it depends on line-height, vertical-align, letter-spacing, word-spacing, how the font glyph is designed (does it naturally have space around it? does it have kerning information?). Then the pseudo elements display type affects if those properties have an effect or not.

See how the pseudo element box isn't quite where the glyph actually is.

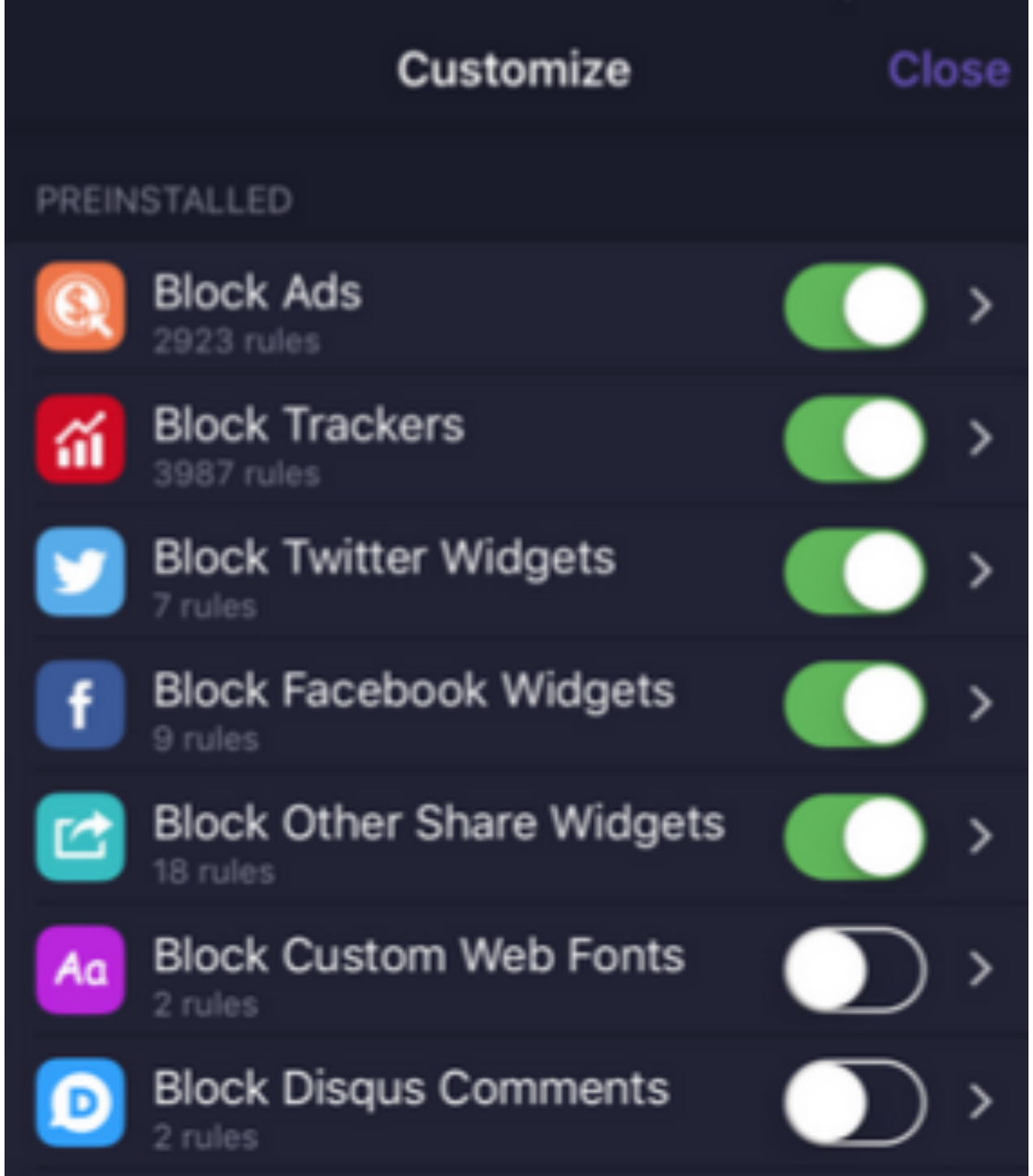
The SVG box is the size of the SVG.

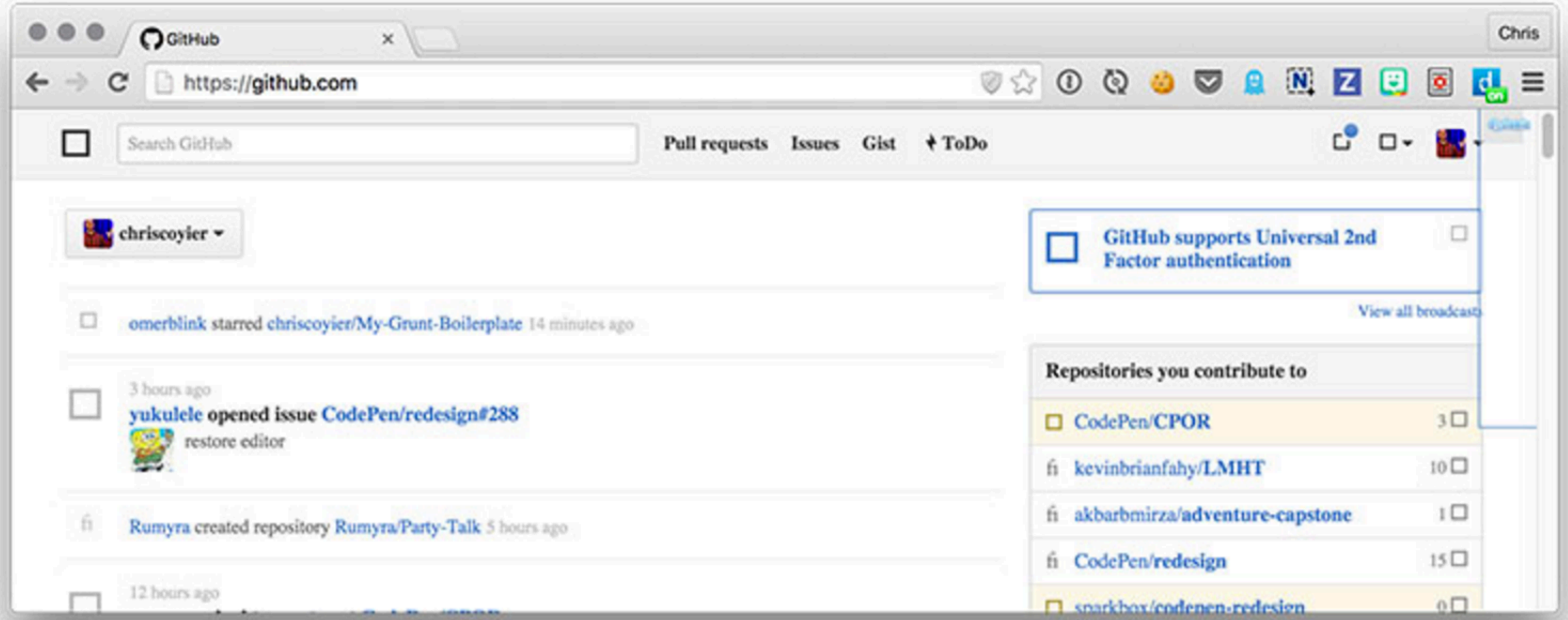
Tip for sizing SVGs with surrounding text:

```
svg {
  height: 1em;
  width: 1em;
}
```

SVG vs Icon Fonts #4

- Icon fonts are subject to many weird and forced failue scenarios and cross-browser issues. They are not displayed when users decide to block web fonts to enhance the speed of the page.
- SVG is right there in the document. If a browser supports it, it displays it. Blockers don't block SVG images, so the icons will be displayed at all times.



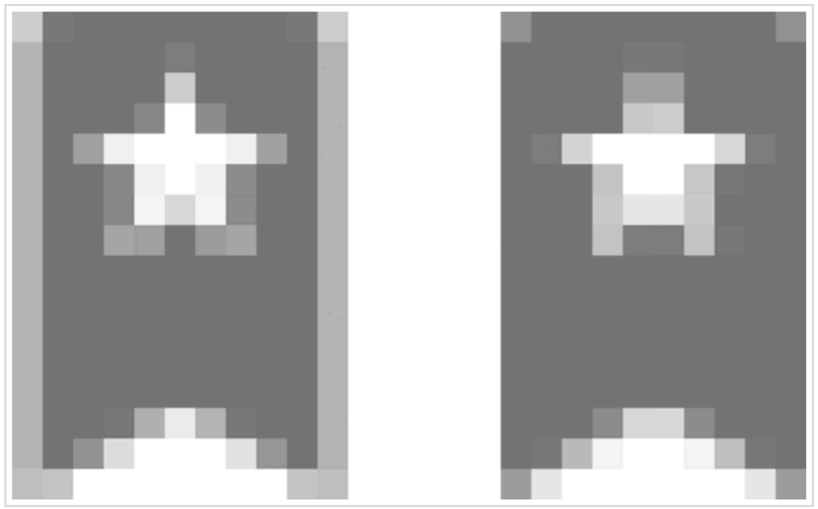


Forcing a different font upon a page can destroy an icon font system.

Delivering Octicons with SVG

February 23, 2016 aaronshekey Engineering

GitHub.com no longer delivers its icons via icon font. Instead, we've replaced all the Octicons throughout our codebase with SVG alternatives. While the changes are mostly under-the-hood, you'll immediately feel the benefits of the SVG icons.



Switching to SVG renders our icons as images instead of text, locking nicely to whole pixel values at any resolution. Compare the zoomed-in icon font version on the left with the crisp SVG version on the right.

Why SVG?

Featured
All Posts
New Features
Engineering
Enterprise
Conferences
Meetups
New Hires
Watercooler
Subscribe

SVG vs Icon Fonts #5

- Icon fonts are not semantic. To use responsibly, you're injecting the icon via a pseudo element on an (empty) . Either bad or no semantics, depending on how you feel about that kind of thing.
- Icons are little images. The semantics of <svg> says "I'm an image."

SVG vs Icon Fonts #6

- SVG comes with default mechanism to make the contents of the SVG accessible to screen readers. Elements such as <title> and <desc> combined with ARIA attributes make SVG icons accessible.
- Icon fonts do not come with semantics or accesisbility. Using ligatures, you can provide basic accessibility, but support starts at IE10.

Accessibility can be baked into reusable elements:

```
<symbol id="logo" viewBox="0 0 300 300">
    <title></title>
    <desc></desc>

    <!-- logo contents -->
</symbol>
```

SVG vs Icon Fonts #7: Browser Support

- SVG is supported in all modern browsers + IE9 and above.
- Icon font support goes back to IE6.

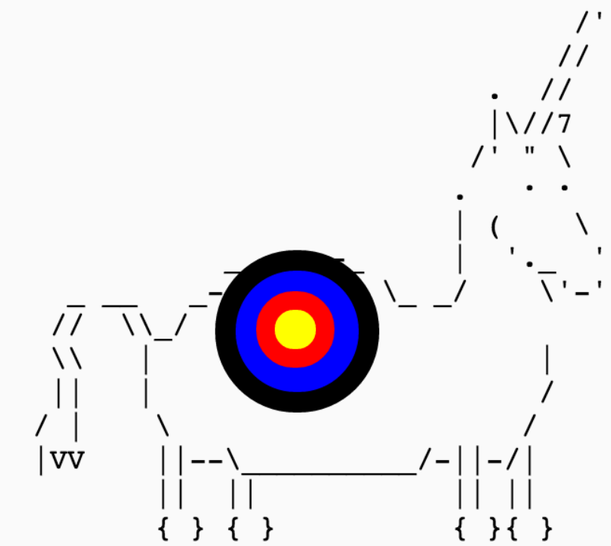
BUT...

There is more than one way you can create and use SVG icon systems.

There are three possible ways to create SVG sprites today. One of these techniques is more similar to icon fonts than you'd think...

Grumpicon creates an SVG sprite inside the CSS file.

- The SVG icons are displayed similar to icon fonts: using CSS classes on empty elements.
- Grumpicon generates all the fallback you need for you, including fallbacks for IE6!
- Grumpicon will automatically switch to using the fallback based on feature detection, so you don't have to worry about doing that yourself.
- Grumpicon generates all the files for you. All you need to do is include the folders in your project and add a small script to the head of your page.



Drag & Drop ur SVGs on the Grumpicon plz.

Or click here 2 upload.

filament group

With special guest, Eric Ponto

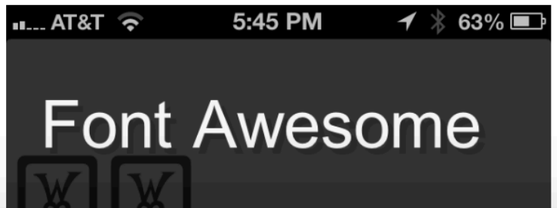
uniconkey head drawn by R.B.Cleary in 1995

A Designer's Guide to Grumpicon, or: How I Learned to Stop Using Font Icons and Love SVGs

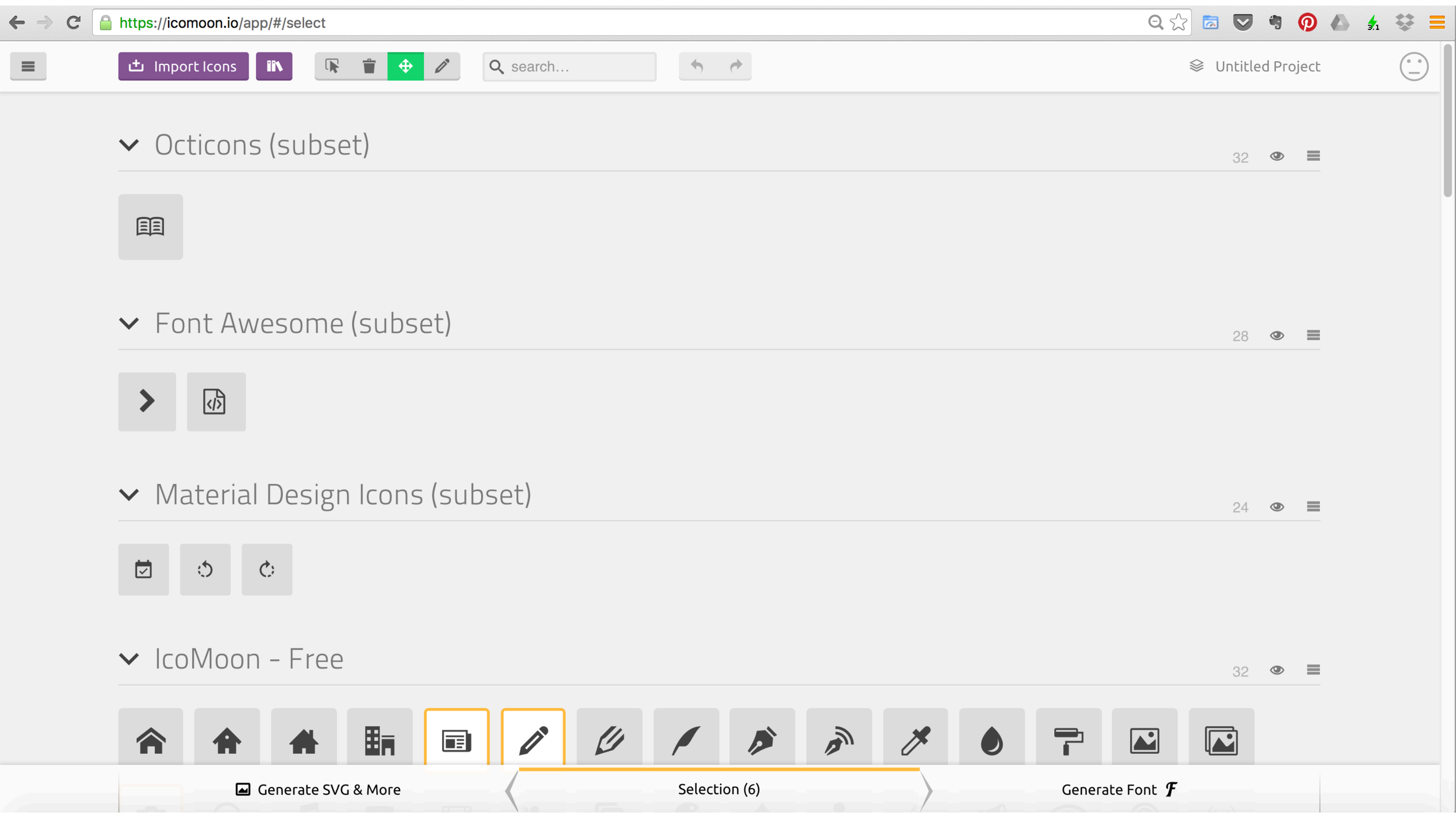
Posted by Todd and Andrew on 07/02/2013

The number of devices with high resolution screens is rising, and with it the need for a simple way to deliver crisp, resolution-independent graphics that don't waste bandwidth. Vector-based graphics, which are compact and scalable, are a great solution.

Icon fonts deliver vector illustrations in an easy format, and there are huge variety of pre-made icon sets to choose from. So, problem solved? Not exactly. They don't work on a number of important mobile platforms like **Opera Mini** (used by 200+ *million* users worldwide!) and **Windows Phone 7.0-7.8** (with 4 million users).



Making the switch from icon fonts to SVG


https://icomoon.io/app/#/select
Import Icons
search...
Untitled Project
Octicons (subset)
32
Font Awesome (subset)
28
Material Design Icons (subset)
24
IcoMoon - Free
32
Generate SVG & More
Selection (6)
Generate Font

font-blast public

Converts any icon-font (FontAwesome etc) into individual SVG files for each icon.

You can use font-blast to extract icons from any icon font - Font Awesome, Foundation, anything from Fontello etc. Font-blast will use the "super-font.svg " file to generate individual SVG/PNG files for each icon.

There are several reasons why you may want to split up the icon font:

- **Use the icons outside of web pages** - so you can import individual icons as SVGs into Sketch, Illustrator, PowerPoint, or any other app to use for visual mockups/presentations.
- **Modify & repackage icons** - you can change a specific icon to suit your taste, then repackage the font (with something like Font Squirrel)
- **Get high quality PNGs** - high-quality PNGs with a transparent backgrounds of any or all icons in the font

Installation

Font-blast is light-weight and has relatively few dependencies. Font-blast does not require PhantomJS or any other binaries on the system, so it should work pretty much everywhere.

Note: PNG images are generated with the embedded batik-rasterizer, and you will need have java installed to do that (but it is not necessary for SVG generation).

```
$ npm install font-blast
```


npm install font-blast

how? learn more

eugene1g published a year ago

0.2.2 is the latest of 3 releases

github.com/eugene1g/font-blast

MIT

Collaborators

Stats

bpierre / fontello-svg

Watch 1 Star 15 Fork 1

Code Issues 3 Pull requests 0 Wiki Pulse Graphs

fontello-svg is a command-line tool that allows to generate an SVG version of a Fontello icon set, with the corresponding CSS file.

32 commits **1** branch **9** releases **2** contributors

Branch: **master** New pull request | New file Upload files Find file | HTTPS https://github.com/bpie | Download ZIP

bpierre 0.5.0 Latest commit `ed0239c` on Dec 16, 2015

bin	Can now specify the filePath as a format e.g. {0}-{1}-{2}.svg will ou…	4 months ago
test	Fontello icon names and GitHub URLs	2 years ago
.gitignore	First!	2 years ago
.travis.yml	Travis CI	2 years ago
LICENSE	First!	2 years ago
README.md	Add --file-format to the README file	4 months ago
index.js	Can now specify the filePath as a format e.g. {0}-{1}-{2}.svg will ou…	4 months ago
package.json	0.5.0	4 months ago

README.md

No execuses!

All about SVG vs Icon fonts: https://css-tricks.com/icon-fonts-vs-svg

Image Editing

free worldwide shipping on every order

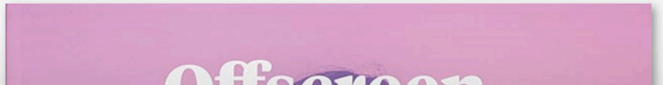
A Print Magazine about the People behind Bits and Pixels

Offscreen is an independent magazine about people who use the internet and technology to be creative, solve problems, and build successful businesses. Captured in enduring print, it documents stories of creativity and passion that shape the digital age.

About Offscreen | View Latest Issue | Buy Latest Issue

Hot off the Press:

Interviewees in Issue No13



Creating a B&W image in modern browsers using CSS became possible with just one line of CSS[1]:

```
img {
    filter: grayscale(100%);
}
```

[1] In CSS, we have no way to specify a composite operation. The default composite operation used is source-over. Both the source and the destination elements remain, and the area where they intersect is blended using the blend mode specified.

```
blur()
brightness()
contrast()
grayscale()
hue-rotate()
invert()
opacity()
saturate()
sepia()
drop-shadow()
url()
```

Adobe Photoshop CC 2015

Feather: 0 px | Anti-alias | Style: Normal | Width: | Height: | Refine Edge... | Essentials

20150330_124134.jpg @ 33.3% (RGB/8) *

Color | Swatches

Libraries | Adjustments | Styles

Add an adjustment

Layers | Channels | Paths

Color | Violet

Normal | Opacity: 100%

Lock: | Fill: 100%

Layer 0

Composite operations[1] and blend modes in CSS using the CSS blending properties:

```
background-blend mode: <blend-mode>

mix-blend-mode: <blend-mode>
```

[1] In CSS, we have no way to specify a composite operation. The default composite operation used is source-over. Both the source and the destination elements remain, and the area where they intersect is blended using the blend mode specified.

Compositing And Blending In CSS

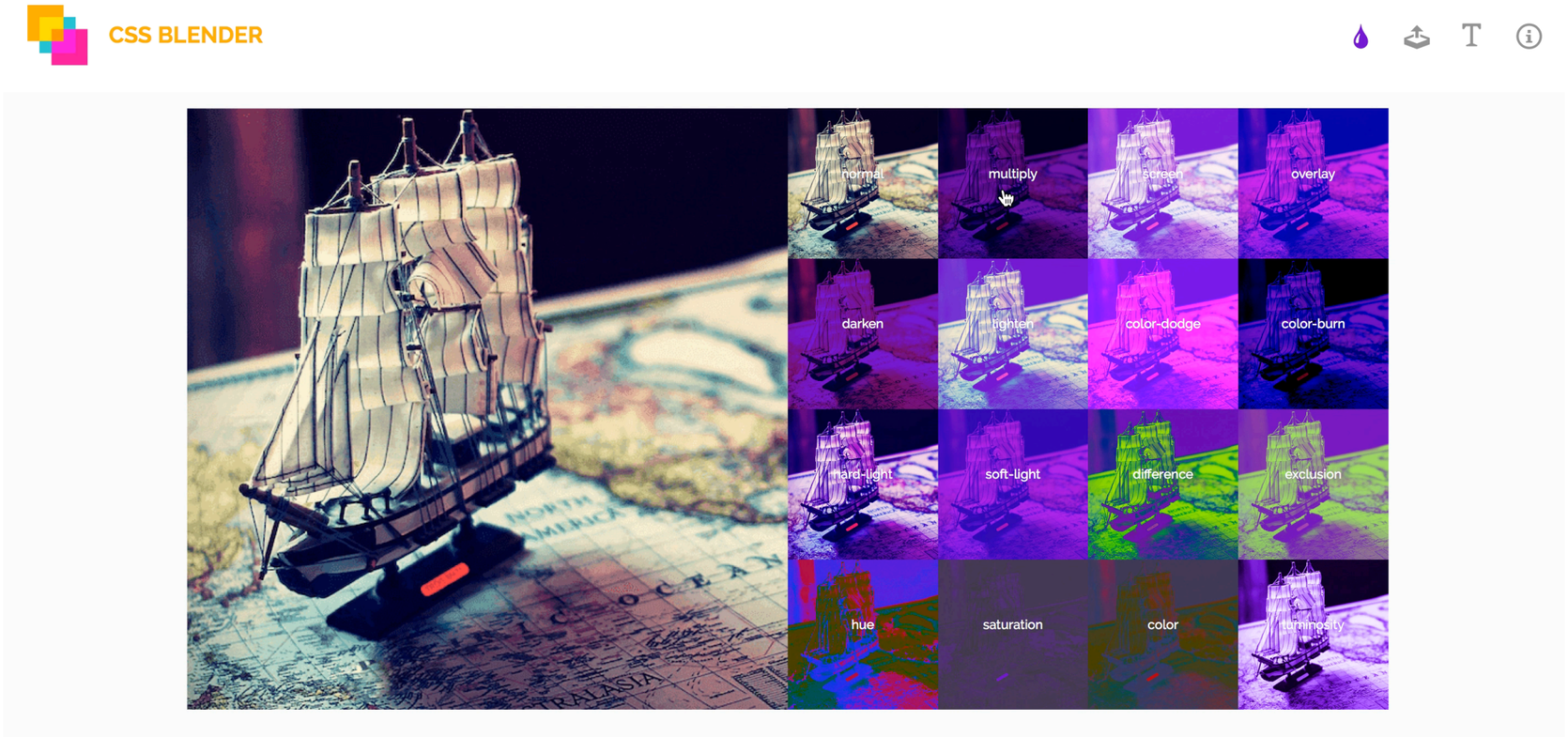
Published January 27, 2015 | Estimated Reading Time: 18 min

If you're a designer, then you've probably already come across blending effects some time or the other. Blending is one of the most frequently used effects in graphic and print design. You can add texture to text by blending it with its textured backdrop, create an illusion of *merged* images by blending these images together, and create a wide range of colorful effects that would not be possible without that fine level of color blending control.

Create a color-tinted effect in CSS:

```
.element {
  background-image: url(...);
  background-color: deepPink;

  /* blend the two background layers together */
  background-blend-mode: luminosity;
}
```

The blend modes can be used to blend multiple background images (and colors) of an element, and to blend elements with their backdrops—i.e. with other elements ‘behind’ them.



Sometimes, you just need more control.


LE PANIER D'OR
Le Panier d'Or
SINT JORIS
HOTEL RESTAURANT CENTRAL

CSS Filters do not offer channel-per-channel color control. But SVG filters do!

`<feColorMatrix>` is a filter type that uses a matrix to affect color values on a per-channel (RGBA) basis[2].

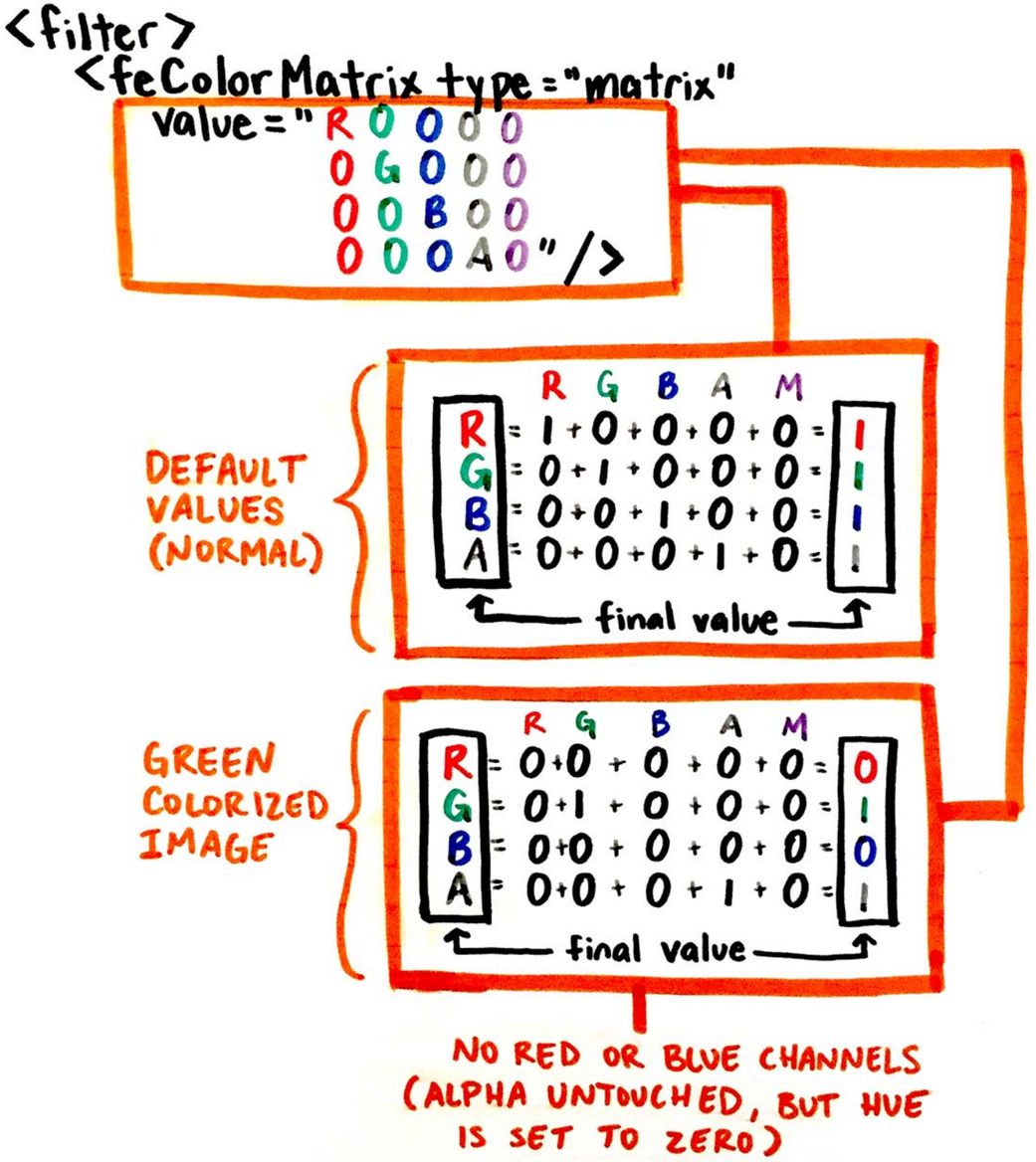
```
<filter id="myFilter">
  <feColorMatrix
        type="matrix"
        values="R 0 0 0 0
                0 G 0 0 0
                0 0 B 0 0
                0 0 0 A 0 "/>
</filter>
```

[2] http://alistapart.com/article/finessing-fecolormatrix

You can colorize images by omitting and mixing color channels.

You can play with the shadow and highlight tones via the alpha channels (fourth column). The fourth row affects overall alpha channels, while the fourth column affects luminosity on a per-channel basis.

Read more here: http://alistapart.com/article/finessing-fecolormatrix



Apply the filter via CSS[3]:

```
img {
   filter: url(#myFilter);
}
```

[3] Not all browsers support SVG filters/effects on HTML elements at this point.

Fall back using an SVG `<image>` instead of `<img>`[4]:

```
<div class="img">
    <svg xmlns:xlink="http://www.w3.org/1999/xlink">
      <image xlink:href="path/to/amsterdam.jpg" width="100%" height="100%"/>
    </svg>
</div>

<svg>
  <defs>
      <filter id="feLime">
        <feColorMatrix
          type="matrix"
          values="1 0 0 0 0, 0 2 0 0 0, 0 0 0 .5 0, 0 0 0 1 0"/>
      </filter>
    </defs>
</svg>
```

---

[4] Source: http://jsfiddle.net/p73938w8

```css
.img {
    display: block;
    position: relative;
    width: 600px;
    height: 400px;
}

.img svg {
    display: block;
    position: absolute;
    width: 100%;
    height: 100%;
    background-color: cyan;
}

.img image {
    filter: url(#feLime);
}
```

Content Clipping

CSS comes with a set of predefined basic shapes:

```
circle()
ellipse()
inset()
polygon()
```

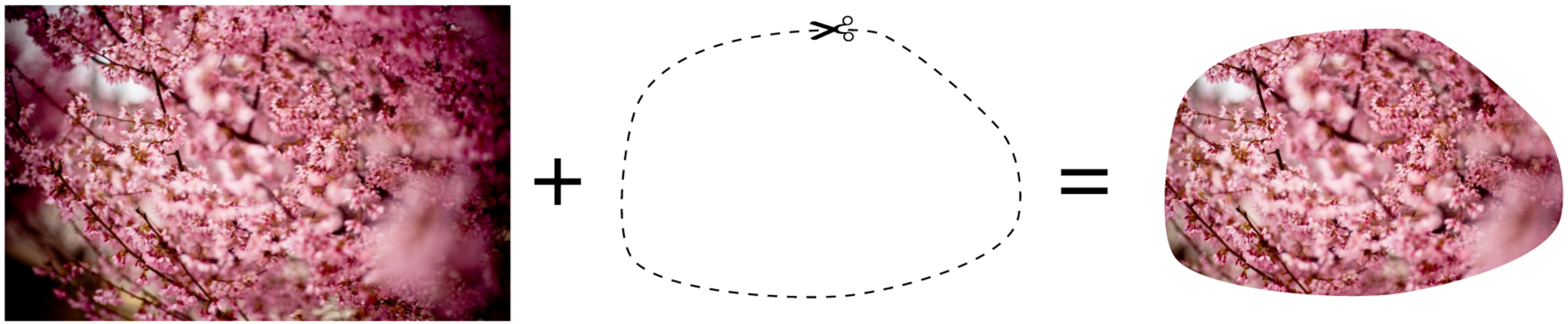
These shapes can be used as values for certain CSS properties such as:

`shape-inside`

`shape-outside`

`clip-path`

The `clip-path` property is used to clip out portions of an element or a container of elements. The shape used to clip the element defines which parts of the elements will be shown (inside of it) and which will be clipped out.



The `clip-path` property accepts either a CSS basic shape or an SVG `clipPath` as values:

```
clip-path: <basic-shape>;

clip-path: url(/path/to/svg-clipPath);
```

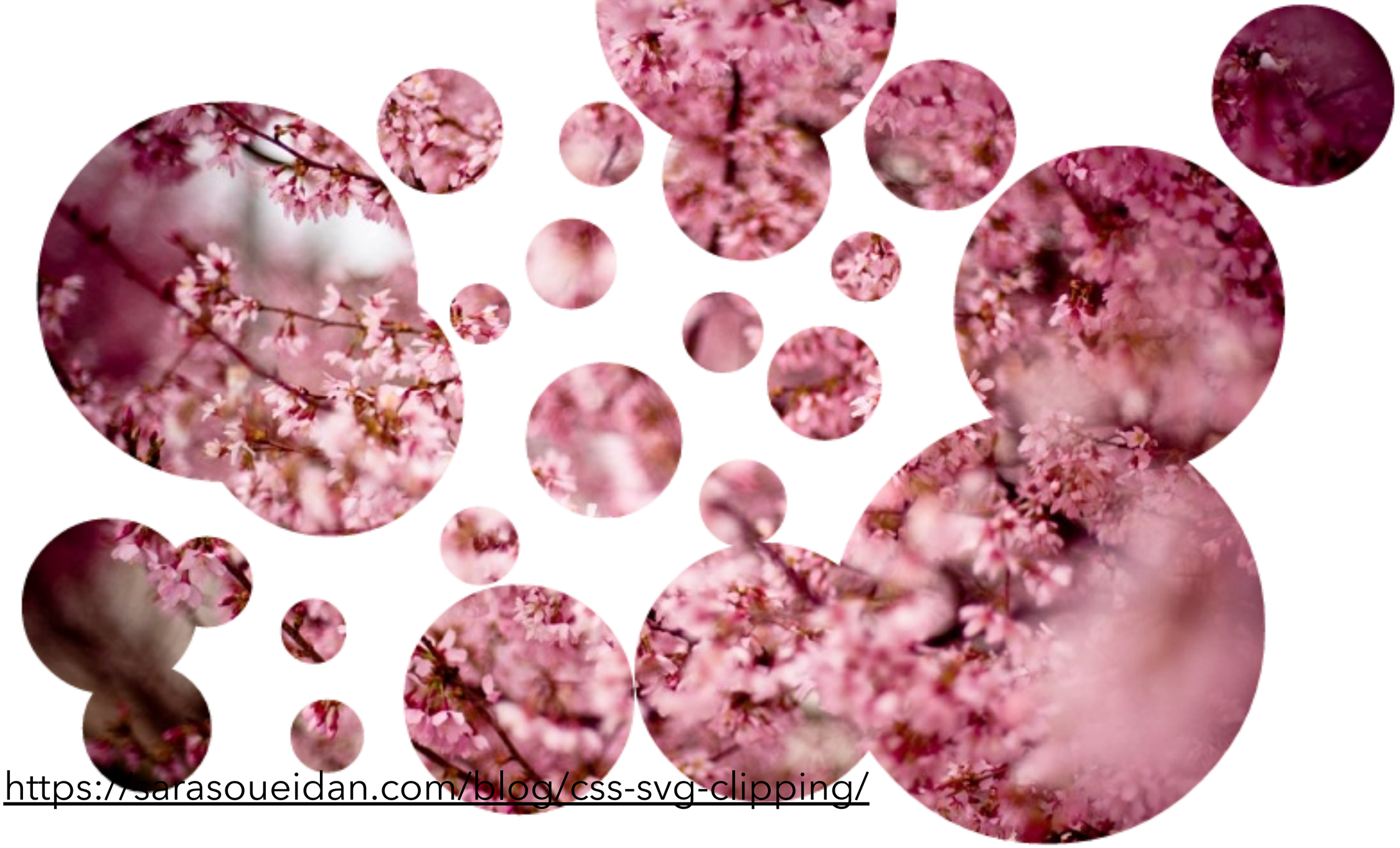
An SVG clip path is defined using the `clipPath` element:

```
<clipPath>
    <!-- elements defined here define the shape
         of the clipping path -->

    <!-- You can create multiple elements. In fact,
         any SVG element is valid, including <text>! -->
</clipPath>
```

```
<svg height="0" width="0">
    <defs>
        <clipPath id="theSVGPath">
            <circle stroke="#000" cx="50" cy="50" r="40" />
            <circle stroke="#000" cx="193.949" cy="235" r="74.576"/>
            <circle stroke="#000" cx="426.576" cy="108.305" r="47.034"/>
            <circle stroke="#000" cx="346.915" cy="255.763" r="43.644"/>
            <circle stroke="#000" cx="255.39" cy="82.882" r="35.17"/>
            <!-- more circles... -->
        </clipPath>
    </defs>
</svg>
```

Apply the clip path via CSS:

```
clip-path: url(#theSVGPath);
```



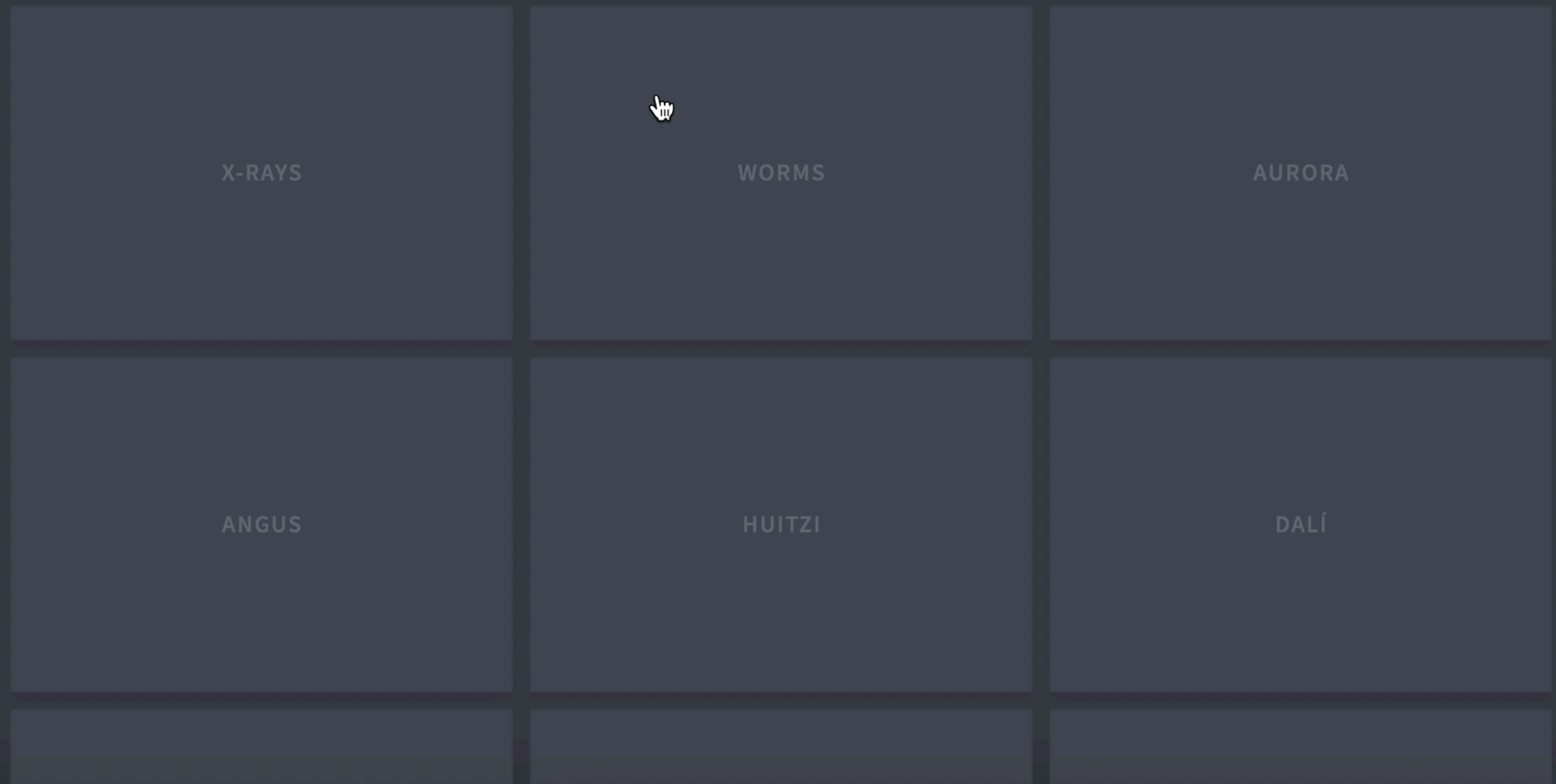
https://sarasoueidan.com/blog/css-svg-clipping/

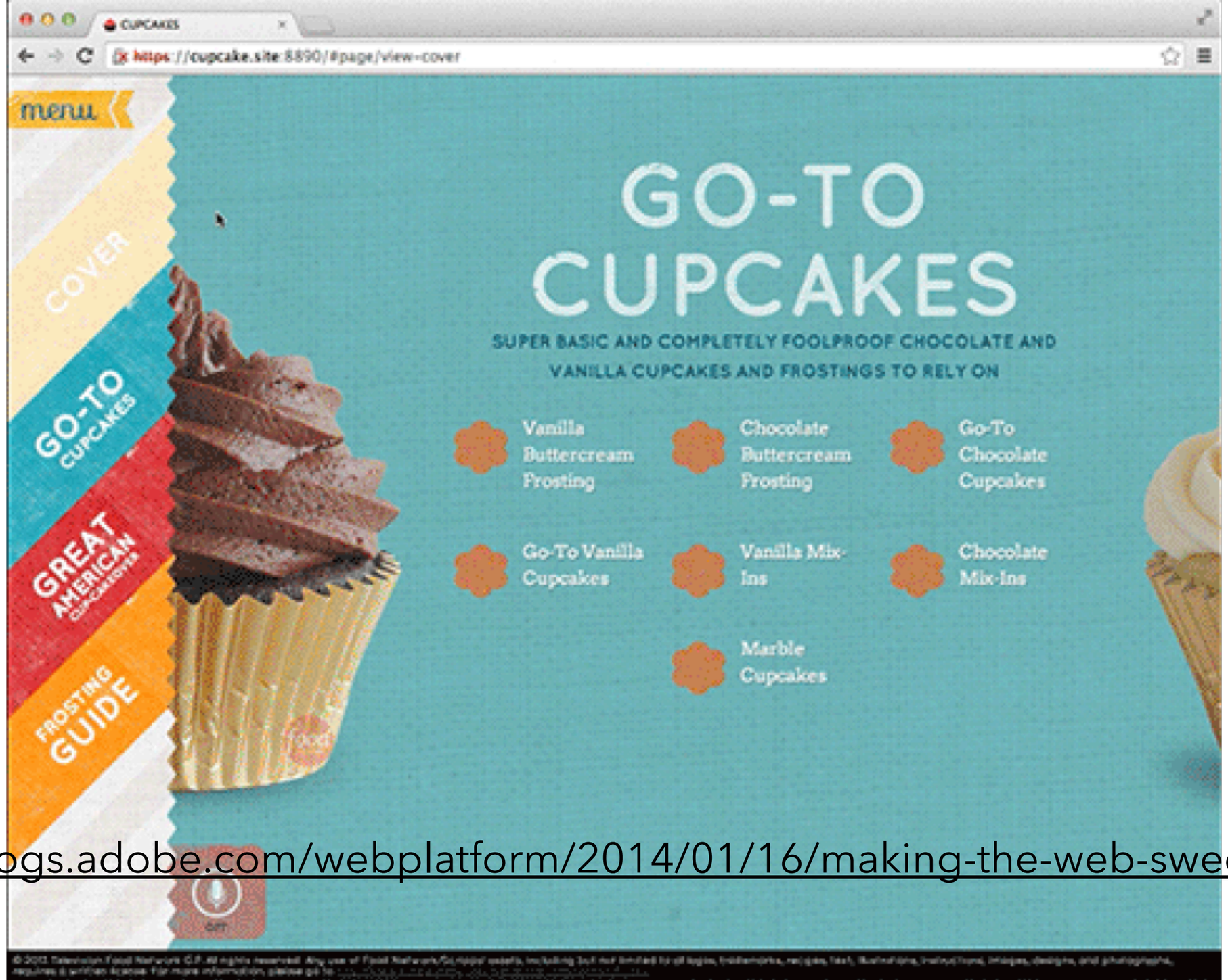
https://css-tricks.com/sketchy-avatars-css-clip-path/

SVG clip-path Hover Effect

Attempt to re-create CJ Gammon's portfolio grid hover effect using SVG and CSS Transitions.

Note: this is an experiment, it does not seem to work on Firefox 43.0.4 neither have touch support.
Tested on Chrome 47.0.2526.106, Opera 34.0 and Safari 8.0.6.





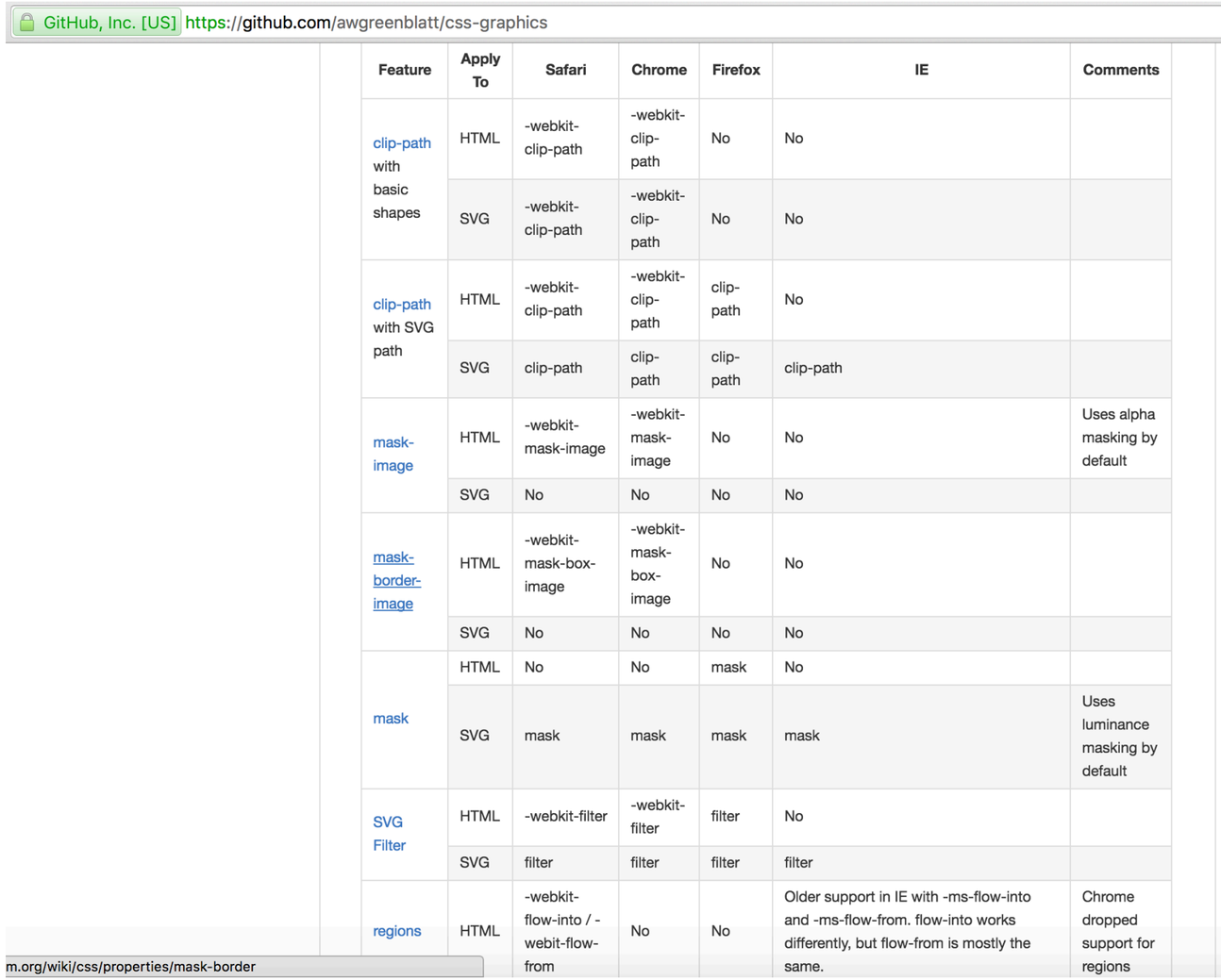
http://blogs.adobe.com/webplatform/2014/01/16/making-the-web-sweeter/

How about clipping videos?!

Theoretically, it’s also as simple as applying a clip path via the `clip-path` property. And it works when using CSS shapes.

Practically, however, it's a bit tricky because of browser support of SVG effects on HTML elements.

But there is a workaround...


GitHub, Inc. [US] https://github.com/awgreenblatt/css-graphics

Feature	Apply To	Safari	Chrome	Firefox	IE	Comments
clip-path with basic shapes	HTML	-webkit-clip-path	-webkit-clip-path	No	No	
	SVG	-webkit-clip-path	-webkit-clip-path	No	No	
clip-path with SVG path	HTML	-webkit-clip-path	-webkit-clip-path	clip-path	No	
	SVG	clip-path	clip-path	clip-path	clip-path	
mask-image	HTML	-webkit-mask-image	-webkit-mask-image	No	No	Uses alpha masking by default
	SVG	No	No	No	No	
mask-border-image	HTML	-webkit-mask-box-image	-webkit-mask-box-image	No	No	
	SVG	No	No	No	No	
mask	HTML	No	No	mask	No	
	SVG	mask	mask	mask	mask	Uses luminance masking by default
SVG Filter	HTML	-webkit-filter	-webkit-filter	filter	No	
	SVG	filter	filter	filter	filter	
regions	HTML	-webkit-flow-into / -webit-flow-from	No	No	Older support in IE with -ms-flow-into and -ms-flow-from. flow-into works differently, but flow-from is mostly the same.	Chrome dropped support for regions

m.org/wiki/css/properties/mask-border

```
<svg class="svg">
    <defs>
      <clippath id="cp-circle">
        <circle r="180" cx="50%" cy="42%"></circle>
        <text text-anchor="middle" x="50%" y="98%">
          LOOK UP
        </text>
      </clippath>
    </defs>

    <g clip-path="url(#cp-circle)">
      <foreignObject width="853" x="0"
                     y="0" height="480">
        <body xmlns="http://www.w3.org/1999/xhtml">
          <iframe width="853" height="480"
                  src="//www.youtube.com/embed/fCT5n23OExE"
                  frameborder="0" allowfullscreen></iframe>
        </body>
      </foreignObject>
   </g>
</svg>
```

CSS and SVG Masks

Live demo

CSS clip

1

Specification

CSS

```
.item {
  position: absolute;
  clip: rect(10px, 190px, 190px, 10px);
  }
```

CSS clip-path

2

Specification

CSS

```
.item {
  clip-path: circle(100px, 100px, 100px);
  clip-path: circle(100px at center);
  }
```

Syntax for circle was chnaged, to radial gradient syntax

CSS clip-path maker

Tweet

Triangle | Trapezoid | Parallelogram | Rhombus

Pentagon | Hexagon | Heptagon | Octagon

Nonagon | Decagon | Bevel | Rabbet

Prefix: -webkit

Demo Size: 280 × 280

Demo Background

Custom URL...

Show outside clip-path: Off

```
-webkit-clip-path: polygon(50% 0%, 0% 100%, 100% 100%);
clip-path: polygon(50% 0%, 0% 100%, 100% 100%);
```

About Clip Paths

The `clip-path` property allows you to make complex shapes in CSS by clipping an element to a basic shape (circle, ellipse, polygon, or inset), or to an SVG source.

Text Shadows

ELEGANT SHADOW

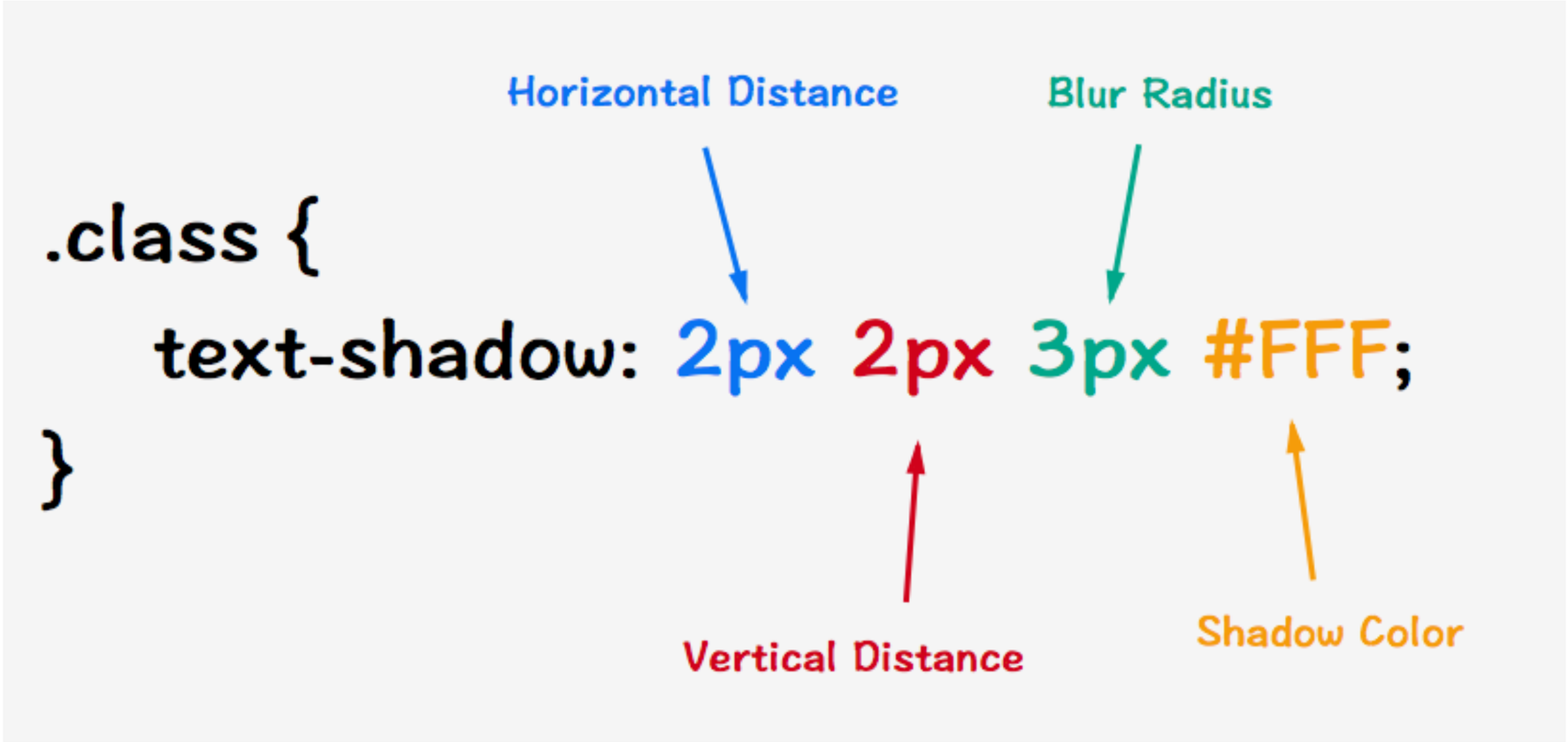
DEEP SHADOW

http://codepen.io/juanbrujo/full/yGpAK

INSET SHADOW

RETRO SHADOW

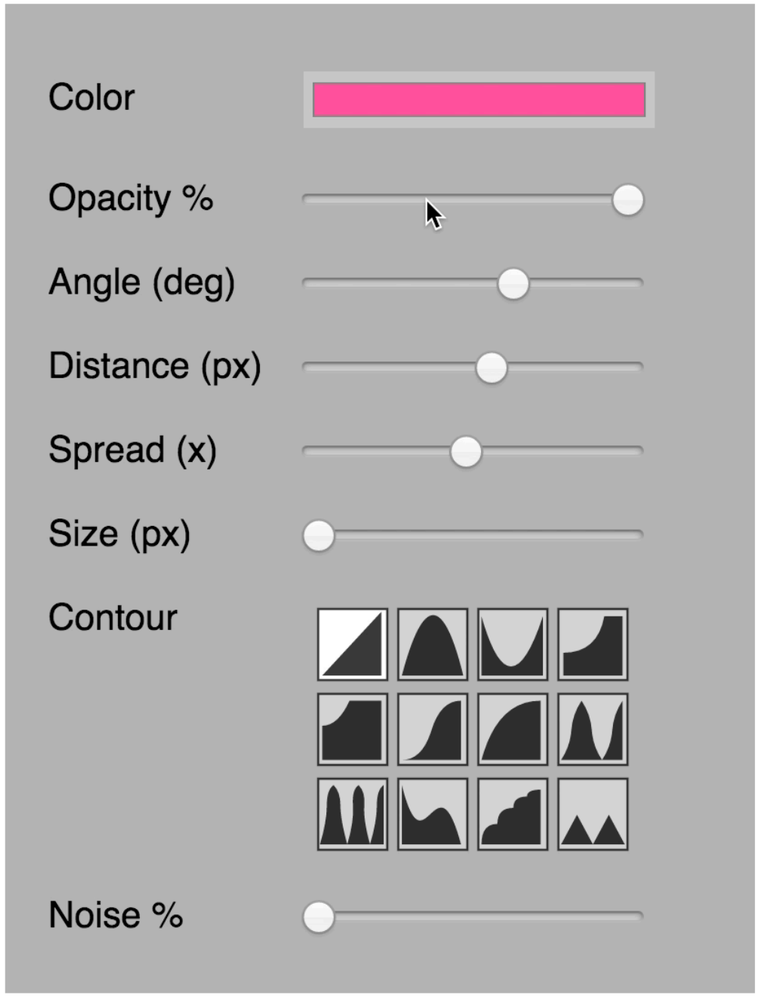
But there's only so much the `text-shadow` can do...



PhotoShop DropShadows Using SVG Filters

A PEN BY Michael Mullany

Fork | Change View

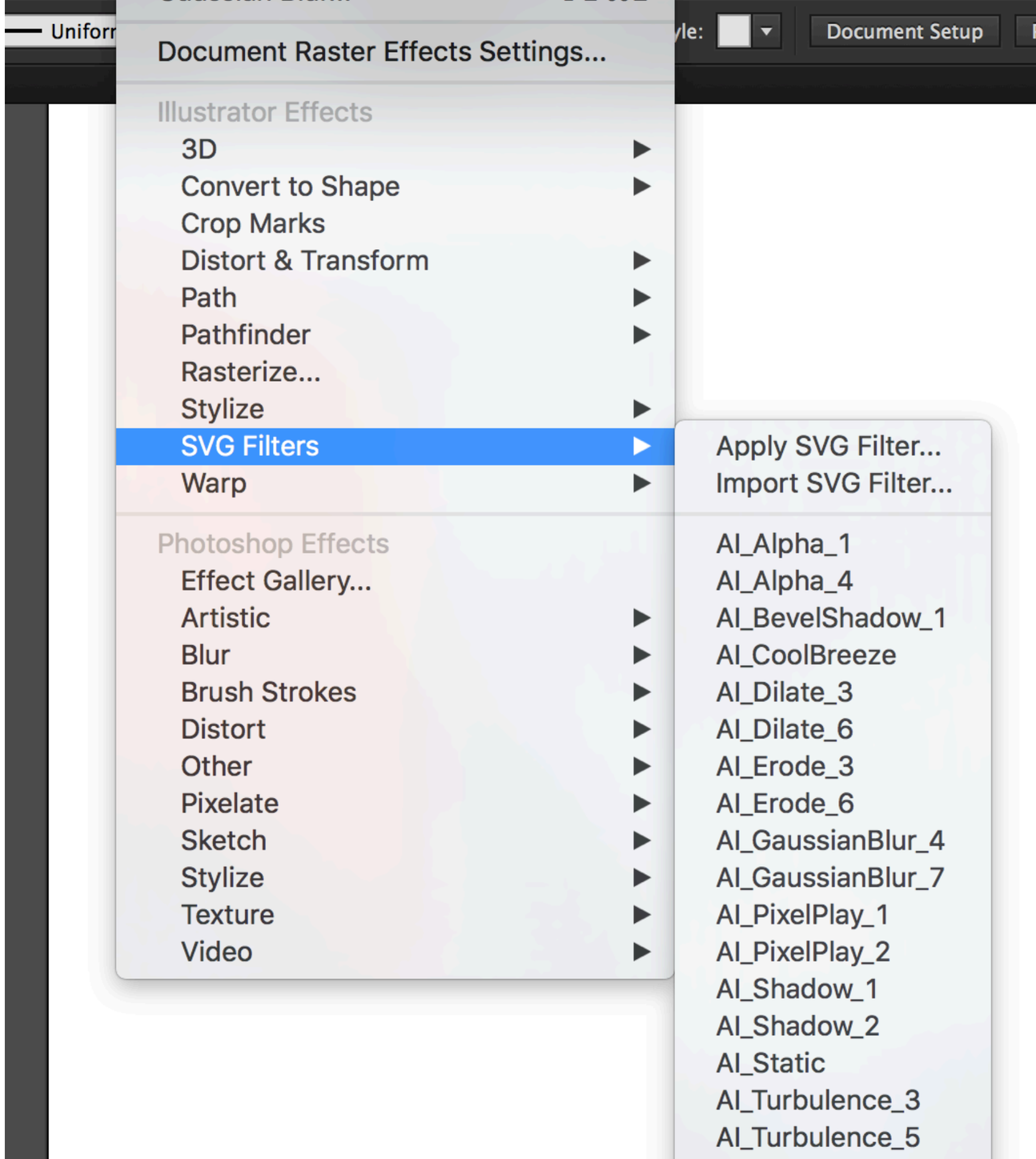


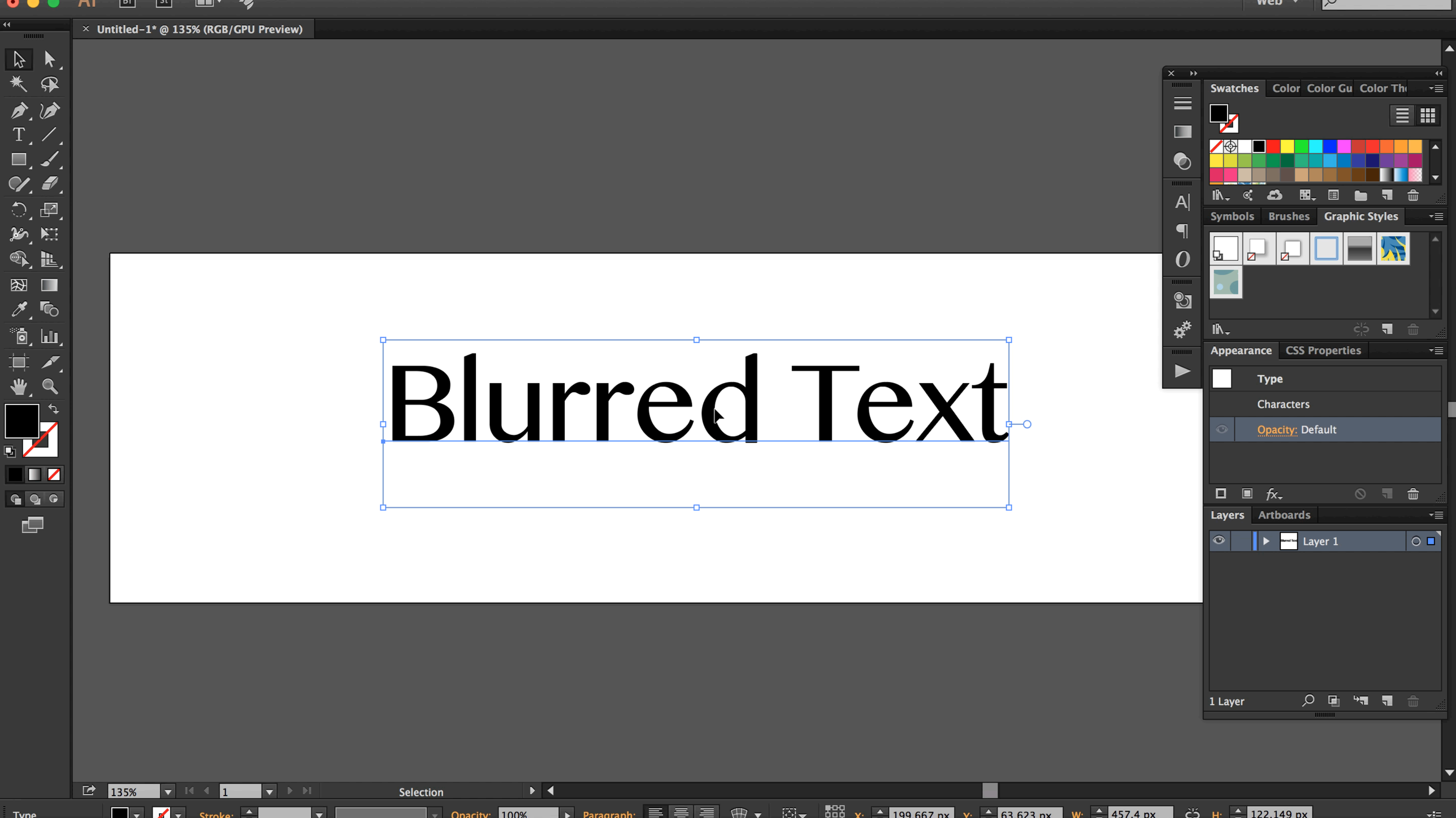

```
<filter id="drop-shadow" color-interpolation-filters="sRGB" x="-50%" y="-50%"
height="200%" width="200%">
```

```
<filter id="drop-shadow" color-interpolation-filters="sRGB" x="-50%" y="-50%" height="200%" width="200%">

<!-- Take source alpha, offset it by angle/distance and blur it by size -->
<feOffset id="offset" in="SourceAlpha" dx="-22.9" dy="-14.31" result="SA-offset"/>
<feGaussianBlur id="blur" in="SA-offset" stdDeviation="3.75" result="SA-o-blur"/>

<!-- Apply a contour by using a color curve transform on the alpha and clipping the result to the input -->

<feComponentTransfer in="SA-o-blur" result="SA-o-b-contIN">
  <feFuncA id="contour" type="table" tableValues="0 .5 0 .5 0"/>
</feComponentTransfer>

<feComposite operator="in" in="SA-o-blur" in2="SA-o-b-contIN" result="SA-o-b-cont"/>

<!-- Adjust the spread by multiplying alpha by a constant factor --> <feComponentTransfer in="SA-o-b-cont" result="SA-o-b-c-sprd">
  <feFuncA id="spread-ctrl" type="linear" slope="8.4"/>
</feComponentTransfer>

<!-- Adjust color and opacity by adding fixed offsets and an opacity multiplier -->
<feColorMatrix id="recolor" in="SA-o-b-c-sprd" type="matrix" values="0 0 0 0 1 0 0 0 0 0.184 0 0 0 0 0.573 0 0 0 1 0" result="SA-o-b-c-s-recolor"/>

<!-- Generate a reasonably grainy noise input with baseFrequency between approx .5 to 2.0. And add the noise with k1 and k2 multipliers that sum to 1 -->
<feTurbulence result="fNoise" type="fractalNoise" numOctaves="6" baseFrequency="1.98"/>
<feColorMatrix in="fNoise" type="matrix" values="1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 7 -3" result="clipNoise"/>
<feComposite id="noisemix" operator="arithmetic" in="SA-o-b-c-s-recolor" in2="clipNoise" k1="0.32" k2="0.68" result="SA-o-b-c-s-r-mix"/>

<!-- Merge the shadow with the original -->
<feMerge>
  <feMergeNode in="SA-o-b-c-s-r-mix"/>
  <feMergeNode in="SourceGraphic"/>
</feMerge>
</filter>
```

To export Illustrator's filter effects as SVG `<filter>`s, make sure you choose the filters in the **`SVG Filters`** drop-down.

Effect → SVG Filters




Untitled-1* @ 135% (RGB/GPU Preview)
Blurred Text
Swatches
Color
Color Gu
Color Th
Symbols
Brushes
Graphic Styles
Appearance
CSS Properties
Type
Characters
Opacity: Default
Layers
Artboards
Layer 1
1 Layer
135%
Selection
Type
Stroke:
Opacity:
100%
Paragraph:
X:
199.667 px
Y:
63.623 px
W:
457.4 px
H:
122.149 px

Creative Text Effects

The Photoshop Way

Replace the text with the image) by hiding the text and appylying the image as a background:

```
.hide-text {
    text-indent: 100%;
    white-space: nowrap;
    overflow: hidden;
    background-image: url(...);
}
```

THE #1 COOKING MAGAZINE IN THE WORLD.

New healthy and delicious recipes every week. Subscribe to the weekly issue of COOK magazine and stay up-to-date on the latest kitchen trends and tips anc tricks from the world's #1 chefs. **Download our app available for Android, iOS and Windows phones.**

```
<div class="clipped">
    <h1>CO<br/>OK</h1>
</div>

.clipped {
    background: url(../img/kitchen.jpg)
                cover / no-repeat center center;
    color: #fff;

    /* -webkit-background-clip clips
        the background of the element
        to the text */
    /* overrides the white text color
        in webkit browsers */

    -webkit-text-fill-color: transparent;
    -webkit-background-clip: text;
}
```

```
<defs>
    <clipPath id="svgTextPath">
        <text x="0" y="300" textLength="800px"
              lengthAdjust="spacing" font-family="Vollkorn"
              font-size="230px" font-weight="700" font-style="italic">
              Blossom </text>
    </clipPath>
</defs>
```

```
<defs>
    <linearGradient id="filler" x="0%" y="100%">
        <stop stop-color="olivedrab" offset="0%"></stop>
        <stop stop-color="peru" offset="20%"></stop>
        <!-- other stops -->
    </linearGradient>
</defs>
<text x="100" y="70%" font-size="205" fill="url(#filler)"> GRADIENT</text>
```

```
<defs>
    <pattern id="filler" patternUnits="userSpaceOnUse"
             width="400" height="400" >
        <image xlink:href="img/fire.gif"
               width="1200" height="600"
               preserveAspectRatio="none" />
    </pattern>
</defs>
<text x="100" y="70%" font-size="200"
      fill="url(#filler)">
    WOOD
</text>
```

http://tympanus.net/codrops/2015/02/16/create-animated-text-fills/

Sara Soueidan in Tutorials
2013/12/02 · 24 Comments

Techniques for Creating Textured Text

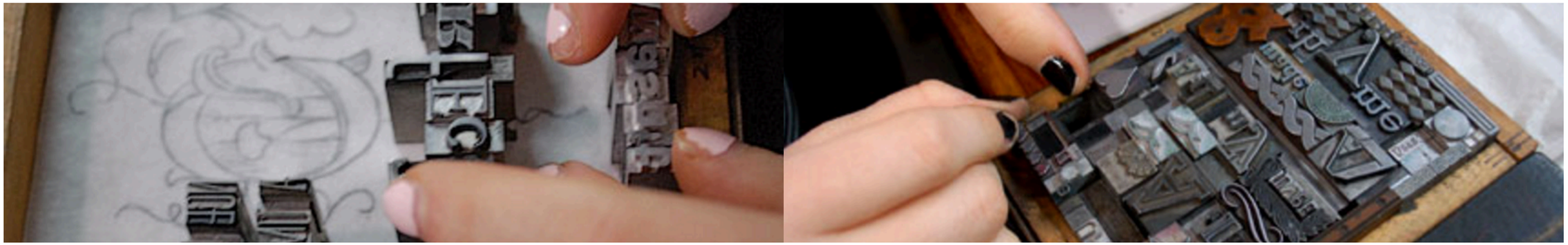
In this article we'll explore all the current techniques for creating image or texture filled text and show you how to apply them.

The text is fully selectable, searchable, and accessible!

Enhance.

Type Lockups

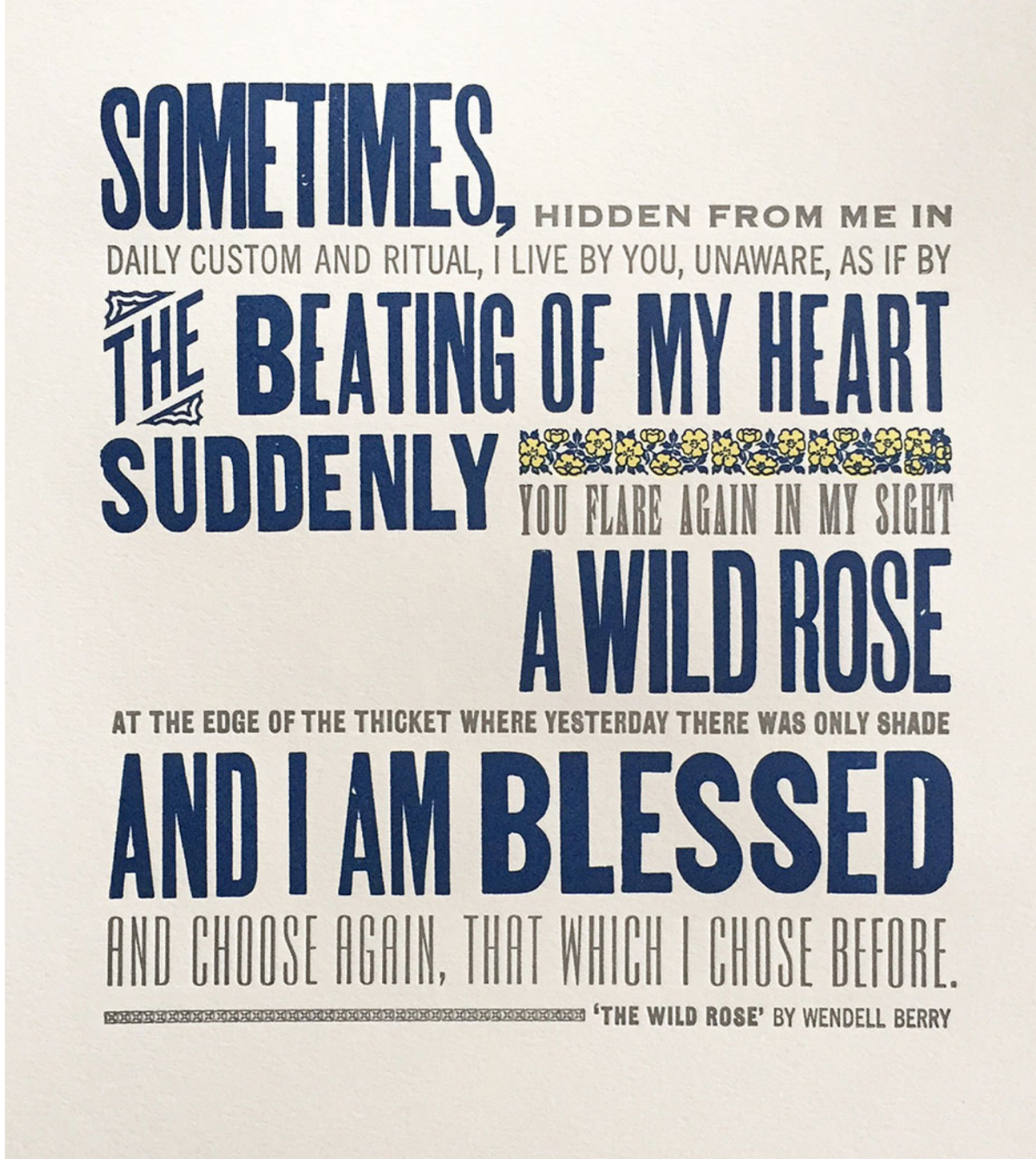
"A type lockup is a typographic design where the words and characters are styled and arranged very specifically. Like the design is literally locked in place. […] A "type lockup" comes from "locking up type", as in actual wooden or metal blocks of type arranged together with other blocks of wood and metal to make a design that you can literally clamp together, ink up, and press to paper." –Chris Coyier



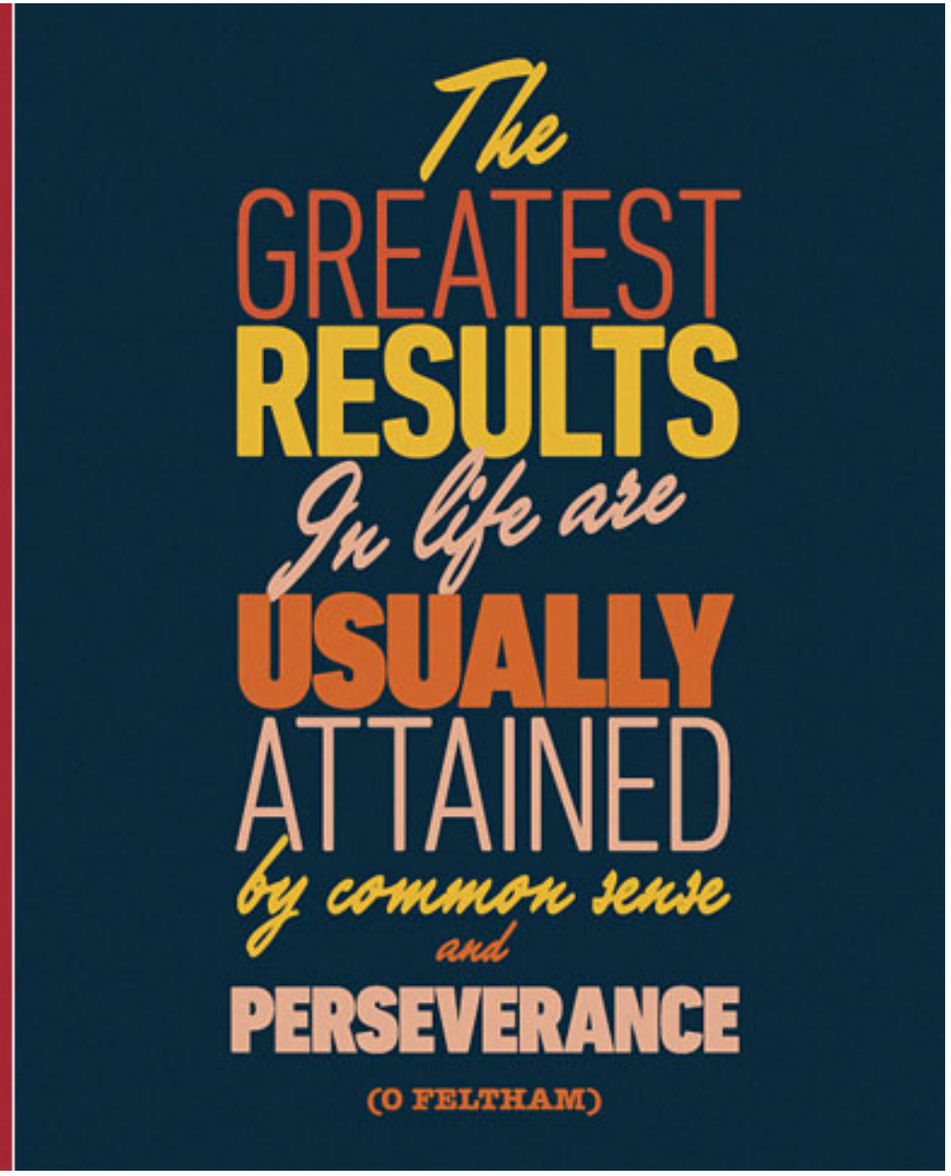
Creative type posters

Posters with creative text layouts, that would normally be hard to replicate in the restrictions of Web page layouts and content flows.

Image Source




Our
GREATEST
GLORY
IS NOT IN NEVER
FALLING
But in
RISING
EVERY TIME WE FALL
(CONFUCIUS)


The
GREATEST
RESULTS
In life are
USUALLY
ATTAINED
by common sense
and
PERSEVERANCE
(O FELTHAM)

Type Lockup

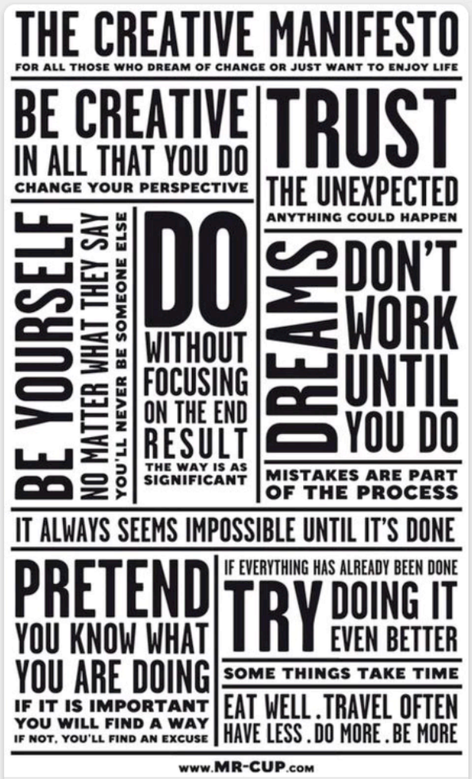
Various Type lock ups from the web

Tom Sullivan

32 Pins

20 Followers

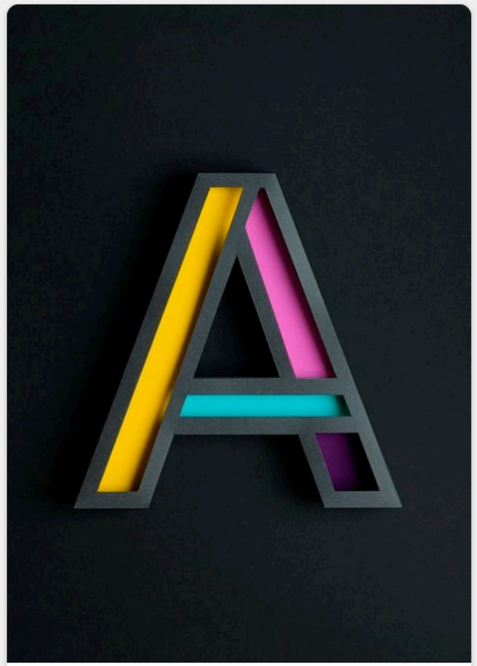
Follow board



Great Type / type.is

1

Pinned from

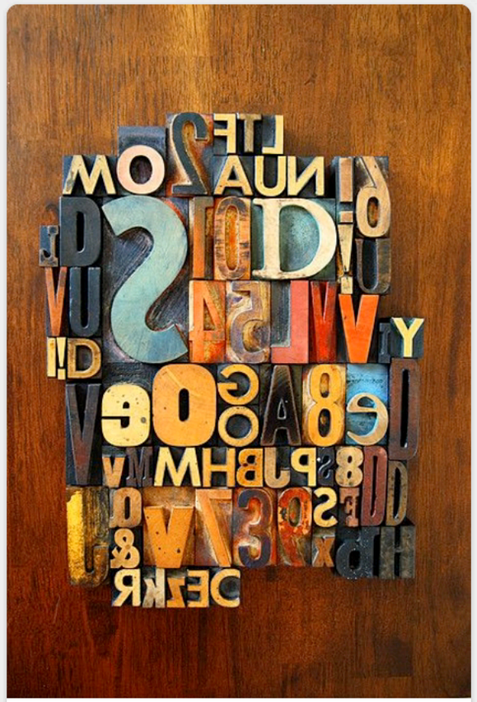


Atype on Behance

by Lobulo Design

1

Pinned from
behance.net



Typography

from Flickr - Photo Sharing!

Pinned from
flickr.com

Peace & Quiet by Bryan Patrick Todd

Pinned from
bryanpatricktodd.com

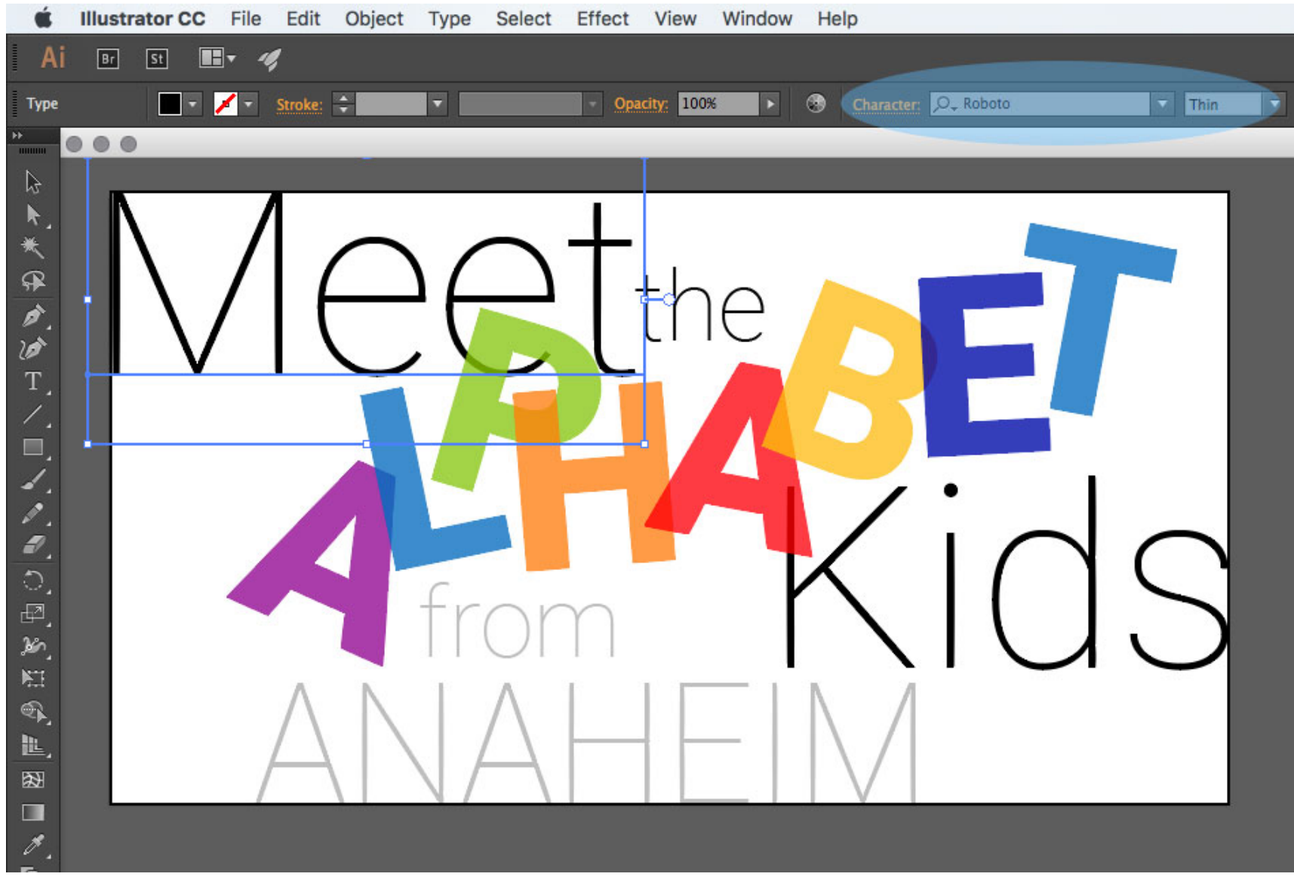
Iampeth envelope by Barbara Calzolari

by Barbara Calzolari

Pinned from
flickr.com

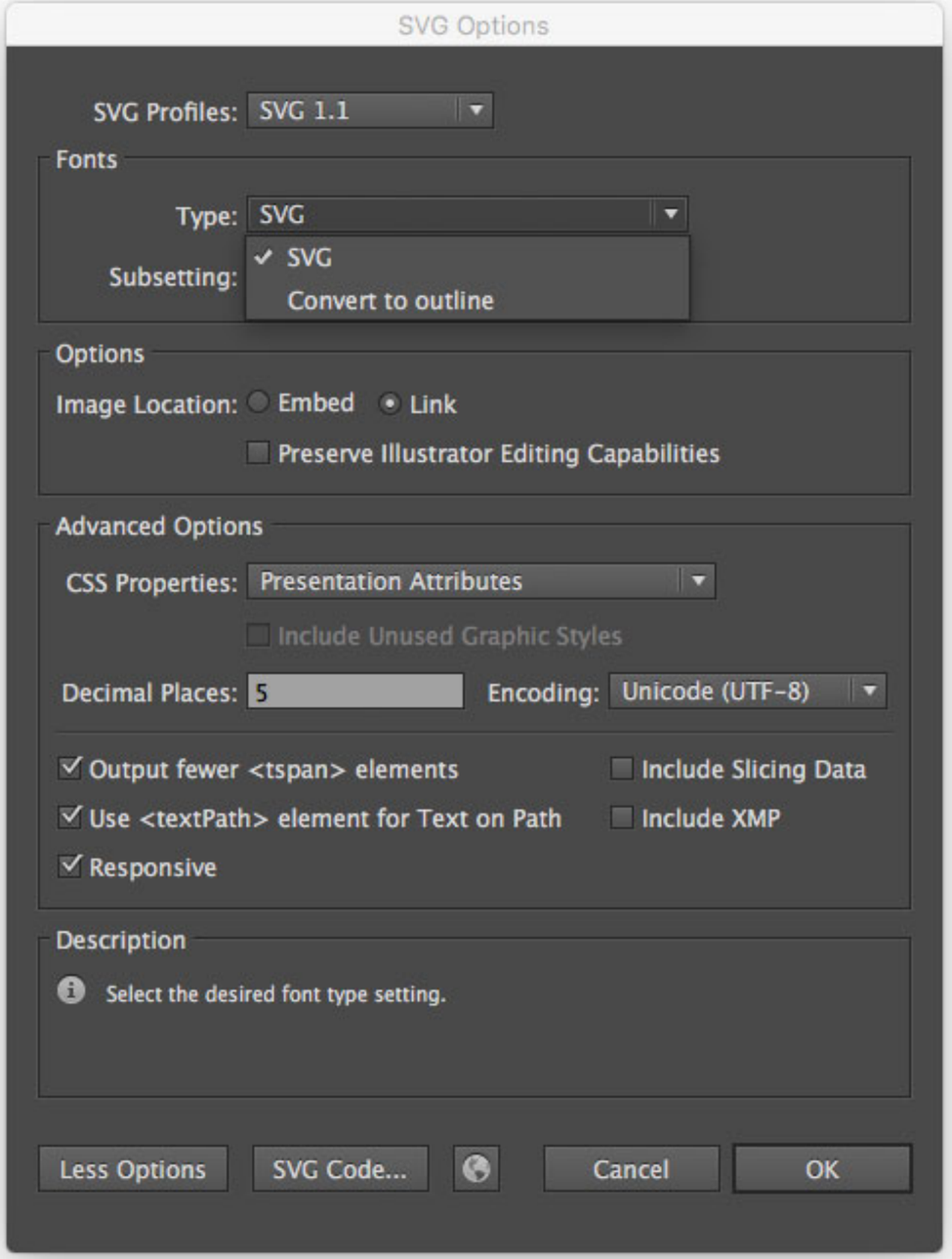
Get creative

- Download/Buy the fonts you need.
- Install the fonts.
- Fire up your favourite graphics editor.
- Create the typographic effects you want.
- **Get Creative.**
- Export your poster as SVG...



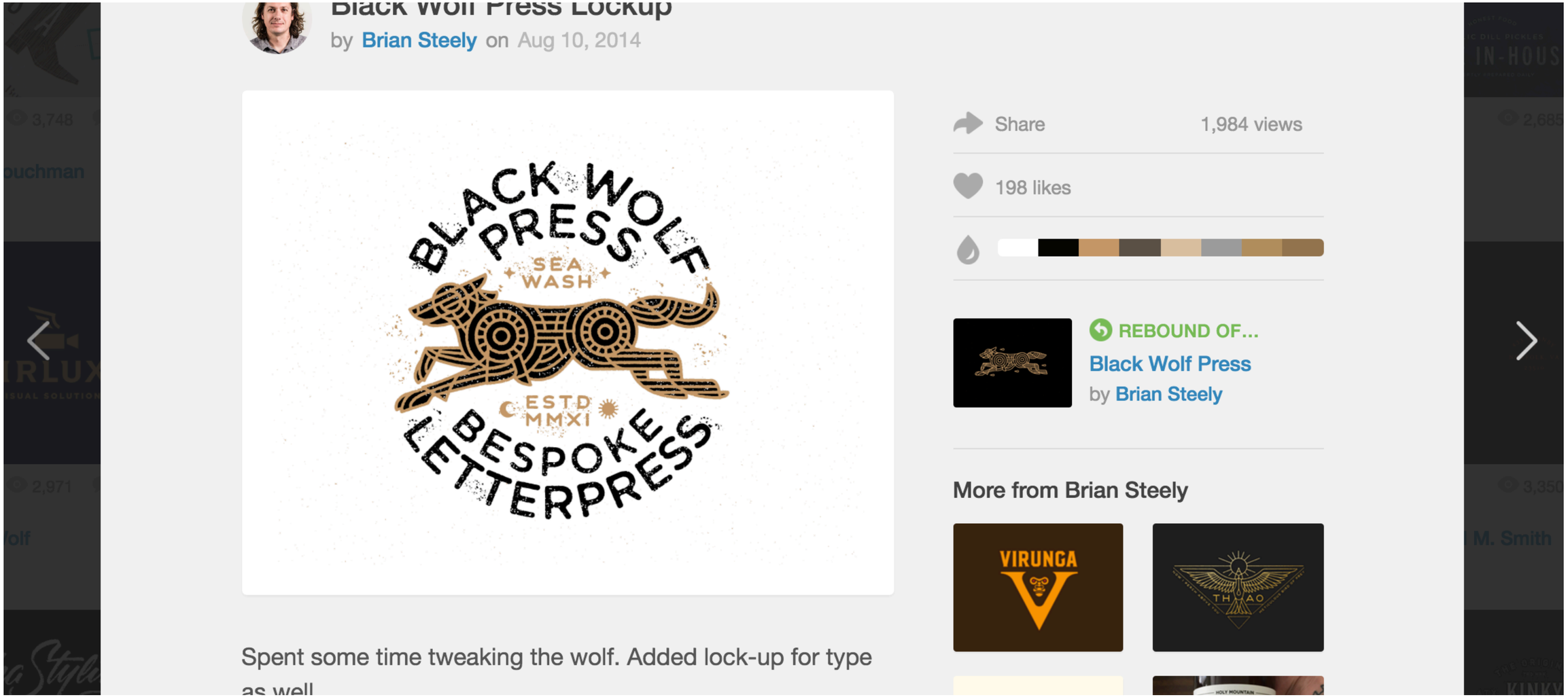
Save the SVG. **Make sure the SVG text is exported as real text, not converted to outlines.**

The whole idea is to create creative text effects that retain the characteristics of real text: searchability, selectability and accessibility. **Outlines have none of these.**



More: https://css-tricks.com/creating-web-type-lockup/

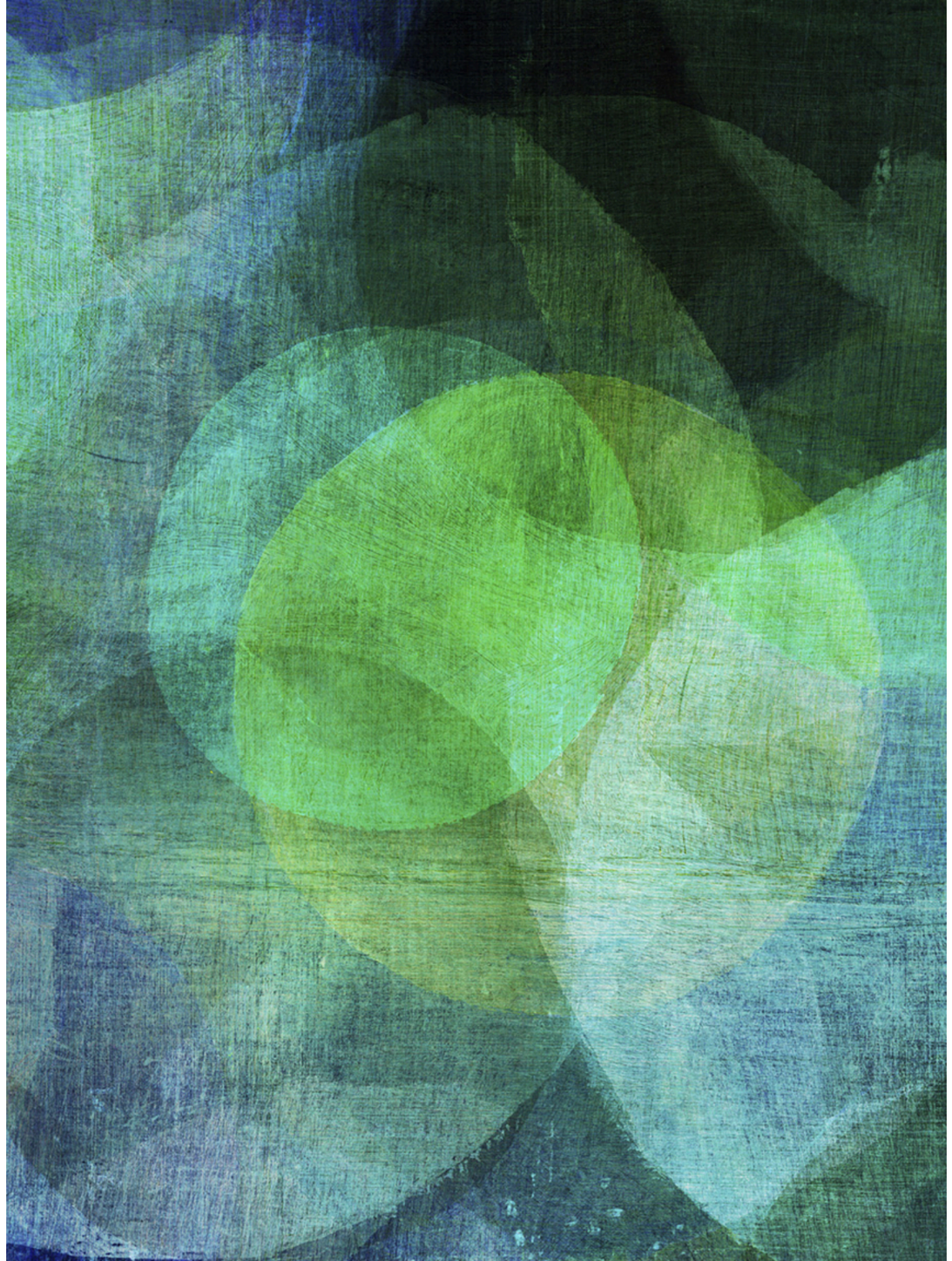
Typical examples of type lockups are logos.



Do convert text to outlines for logos.

Using CSS, SVG logos can be ‘responsified’, and they can be made to adapt to different size and color contexts…

Responsive Logos



responsivelogos.co.uk

Responsive Logos

You don’t need multiple SVG images to create responsive logos.

Making the SVG responsive

Generally speaking, it is recommended that you *preserve* the width and height of the SVG when you export it from the graphics editor—it helps avoid the FOUSVG and makes it possible to fix scaling problems when the SVG is used as a background image.

Leveraging the style cascade, you can make the SVG responsive using CSS without having to use an SVG image sprite.

To make an SVG responsive:

1. Make sure the `viewBox` is not missing.
2. Keep the `width` and `height` attributes. (Override them in CSS.)
3. Embed the SVG using one of the 7 possible ways.
4. Apply a hack/fix to make it cross-browser responsive[5].

---

[5] http://tympanus.net/codrops/2014/08/19/making-svgs-responsive-with-css

If the SVG is embedded using `<img>` or `<picture>` or `<object>` (or `<embed>`):

```
img, object {
  width: 100%;
}
```

This fix is required for IE. Without it, the SVG height will be fixed to 150px.

If the SVG is embedded using `<iframe>` or `<svg>`:

Apply the padding hack…

How does the padding hack work?

1. Wrap SVG in a container.
2. Collapse container's height.
3. Apply (top or bottom) padding such that the resulting height to width ratio = the svg's height to width ratio.
4. Creating positioning context inside the container.
5. Position svg absolutely inside the container.
6. Give svg height and width = 100%.

```
<div class="container">
    <iframe src="logo.svg" alt="Company Logo">
        <!-- fallback -->
    </iframe>
</div>
```

```
.container {
    height: 0;
    width: width-value;
    padding-top: (svg 'height' / svg 'width') * width-value;
    position: relative;
}
```

```
iframe {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
}
```

Making the SVG content adapt (w/ Media Queries)


ENJOY DE SUNSHINE
SUMMER ON THE SEA


ENJOY DE SUNSHINE
SUMMER ON THE SEA

```svg
<svg viewBox="0 0 194 186">
    <style>
        svg * {
            transition: fill .1s ease-out, opacity .1s ease-out;
        }
        @media all and (max-width: 250px) {
            #curved_bg {
                opacity: 0;
            }
            #secondary_content, #primary_content {
                fill: #195463;
            }
        }
        @media all and (max-width: 200px) {
            #secondary_content {
                opacity: 0;
            }
        }
        @media all and (max-width: 150px) {
            #inner-circle, #middle-circle {
                opacity: 0;
            }
        }
    </style>
    <path id="curved_bg" fill="#195463" d="..."/>
    <!-- ... -->
</svg>
```

SVG Media Queries are element queries, except where SVG is embedded inline (`<svg>`).

One SVG image. CSS Media Queries. No Sprite required.

Thank You! ❤️