Multi-stroke text effect in CSS

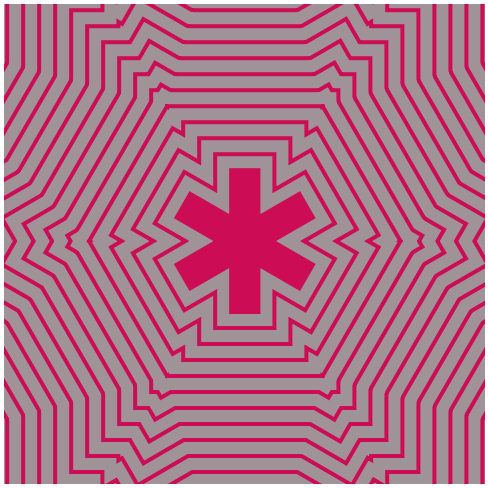
11 April 2026

I used to see that retro multi-stroke text effect quite often and tried to replicate it using the CSS `text-stroke` property, but the results never quite matched. Because `text-stroke` accepts a single value, stacking elements was the only workaround I could think of, though it didn't seem to work.

One evening late last year, I was eager to give it another shot after seeing the text effect again in the book Graphic Japan : from woodblock and zen to manga and kawaii.

Text stroke effect from the book

I kept stacking several elements and accidentally varied the `text-stroke-width` for each layer. To my suprise, the result was getting closer this time.

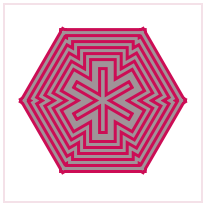


```
--c: #cc0d55;
--n: @i(-1);

@grid: 36x1 / 240px;
@content: '*';

position: absolute;
inset: 0;
font: 100px/0 sans-serif;
color: var(--c);
z-index: @I(-@i);

-webkit-text-stroke-color: @pn(--c, #f4e1e8)
-webkit-text-stroke-width: $em(.08n+.02(1-(-
```

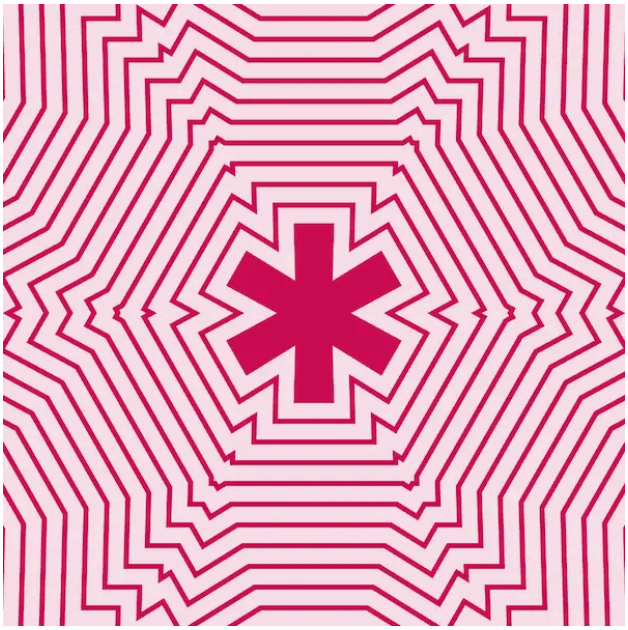
How it works

For different values of the `text-stroke-width`, browsers will automatically draw outlines of the charater, The larger you set the stroke width, the thicker the outline will get, while still maintain its original shape.

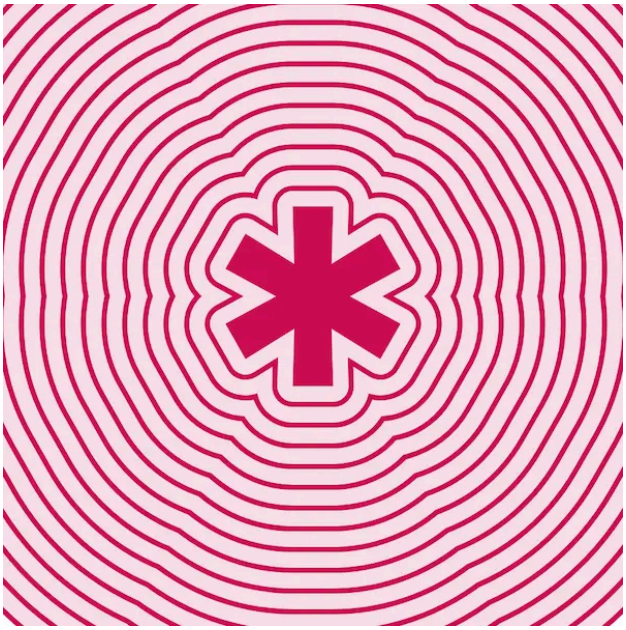
The next step is to use different colors and put them in order.



The interesting part is how the browsers outlining the character shapes differently. FireFox offers more smoother rendering than in Chrome and Safari.

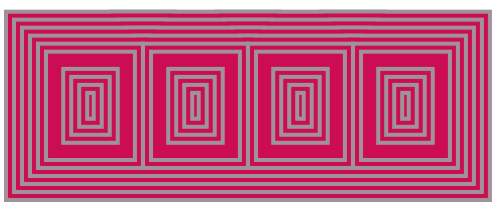


Chrome/Safari



Firefox

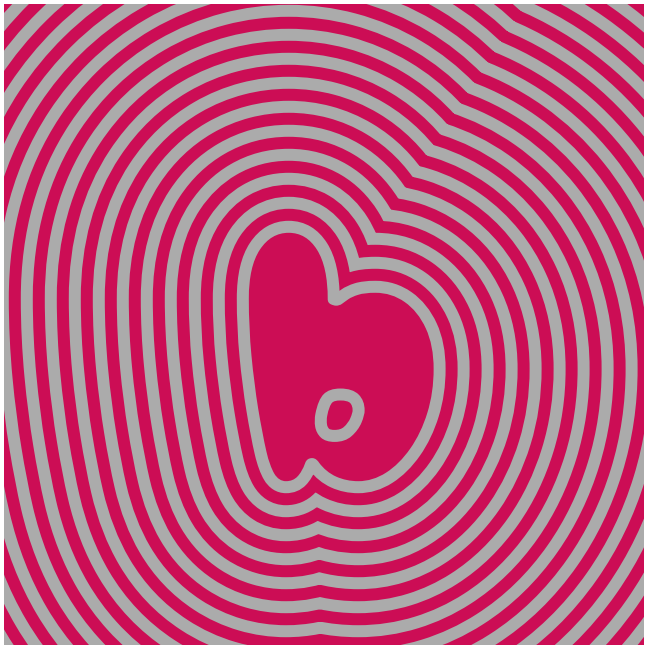
Another interesting part is when there are more text put inline, the character shapes will be merged.



```
/* ... */
@content: '□□□□';
```

Trying different fonts

The final result really depends on the font you choose. To help experimenting with different fonts more quickly, I added the `@google-font` function for faster font loading.



```
font-family: @google-font(Matemasie);
@content: 'b';
```

```
  font-family: @google-font(Tangerine);
  @content: 'Love';
```

```
  font-family: @google-font('Cherry Bomb One');
  @content: '+';
```

Unfortunately, the performance is as bad as CSS filters, especially when the font-size is getting bigger, you may have noticed some flicking above. It's fine for experiments like this, or for generating images with css-doodle, but it's not well-suited for production usage.

More examples

Here are two more examples to play around with different colors and characters, generated with css-doodle, just for fun.

CodePen link for the first one: https://codepen.io/yuanchuan/pen/ogzarGo