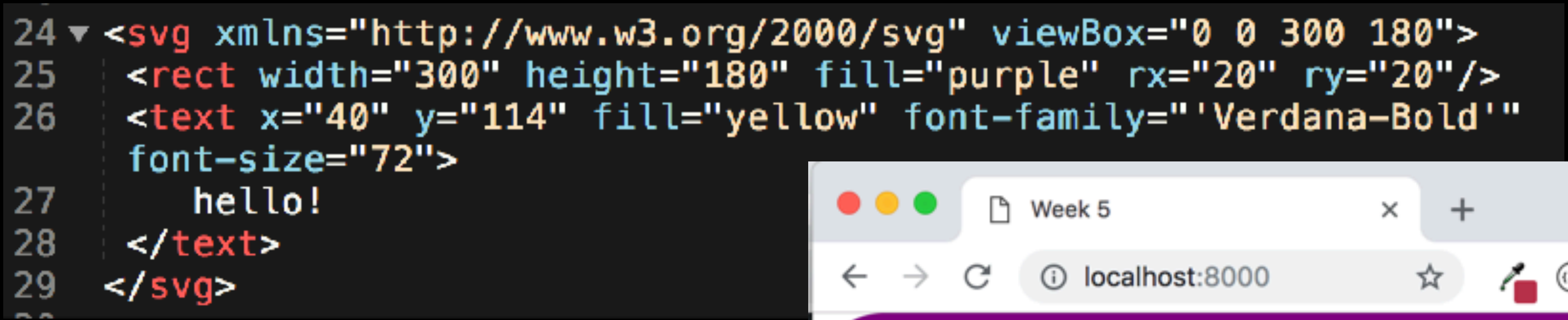
SVG

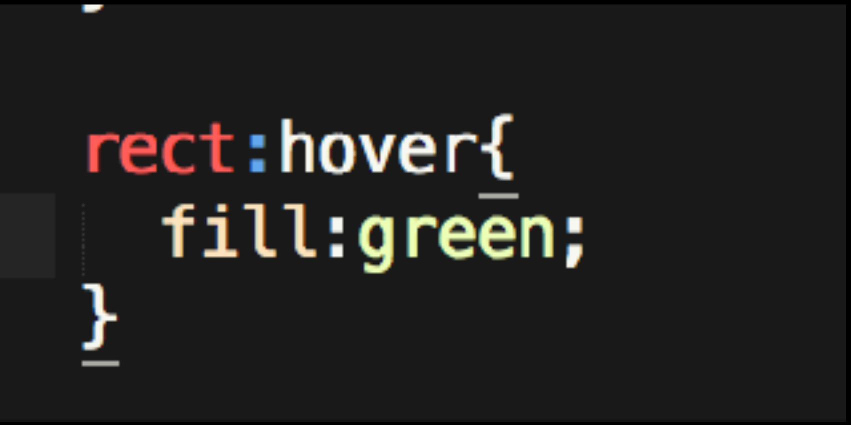
SVG is a 2D vector imaged based on an **XML** (Extensible Markup Language) syntax. It provides the rules and standards for how markup languages should be written and work together. As a result, SVG works well with HTML content.

In SVGs shapes and paths are specified by instructions written out in a text file. Let that sink in: they are images that are written out in text! All of the shapes and paths as well as their properties are written out in the standardized SVG markup language. As HTML has elements for paragraphs **<p>** and navigation **<table>**, SVG has elements that define shapes like rectangle **(rect)**, circle **(circle)**, and paths **(path).**

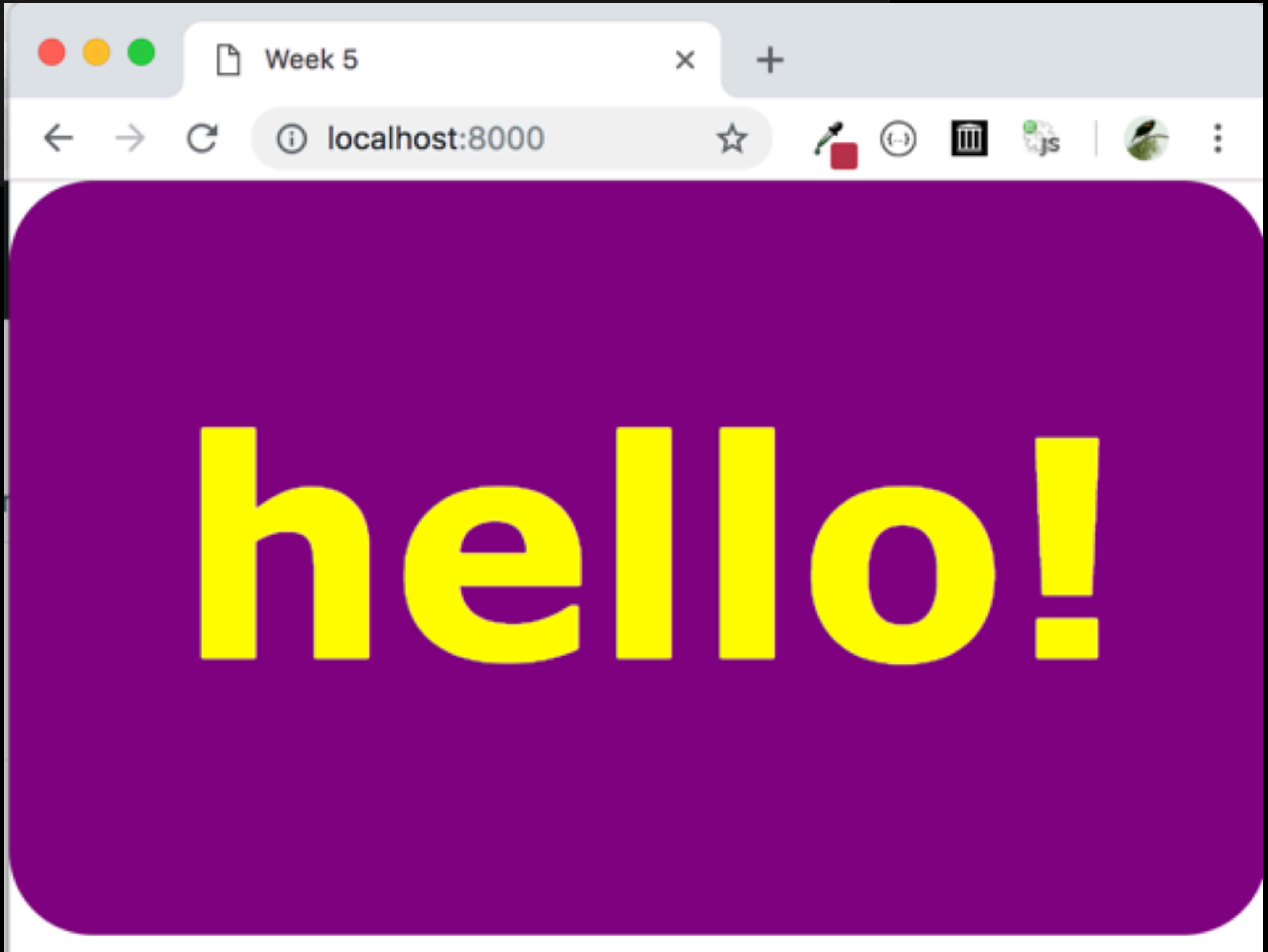
A simple example will give you the general idea. Here is the SVG code that describes a rectangle **(rect)** with rounded corners (rx and ry, for x-radius and y-radius) and the word "hello" set as text with attributes for the font and color. Browsers that support SVG read the instructions and draw the image exactly as designed:



```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 300 180">
  <rect width="300" height="180" fill="purple" rx="20" ry="20"/>
  <text x="40" y="114" fill="yellow" font-family="'Verdana-Bold'"
  font-size="72">
     hello!
  </text>
</svg>
```



```
rect:hover{
  fill:green;
}
```



SVG

Advantages of SVGs over bitmapped counterparts for certain image types:

Because they save only instructions for what to draw, they generally require **less data** than an image saved in a bitmapped format. That means faster downloads and better performance.

Because they are **vectors,** they can resize as needed in a responsive layout without loss of quality. An SVG is always nice and crisp. No fuzzy edges.

Because they are text, they integrate well with HTML/XML and can be compressed with tools like Gzip and Brotli, just like HTML files.

They can be animated.

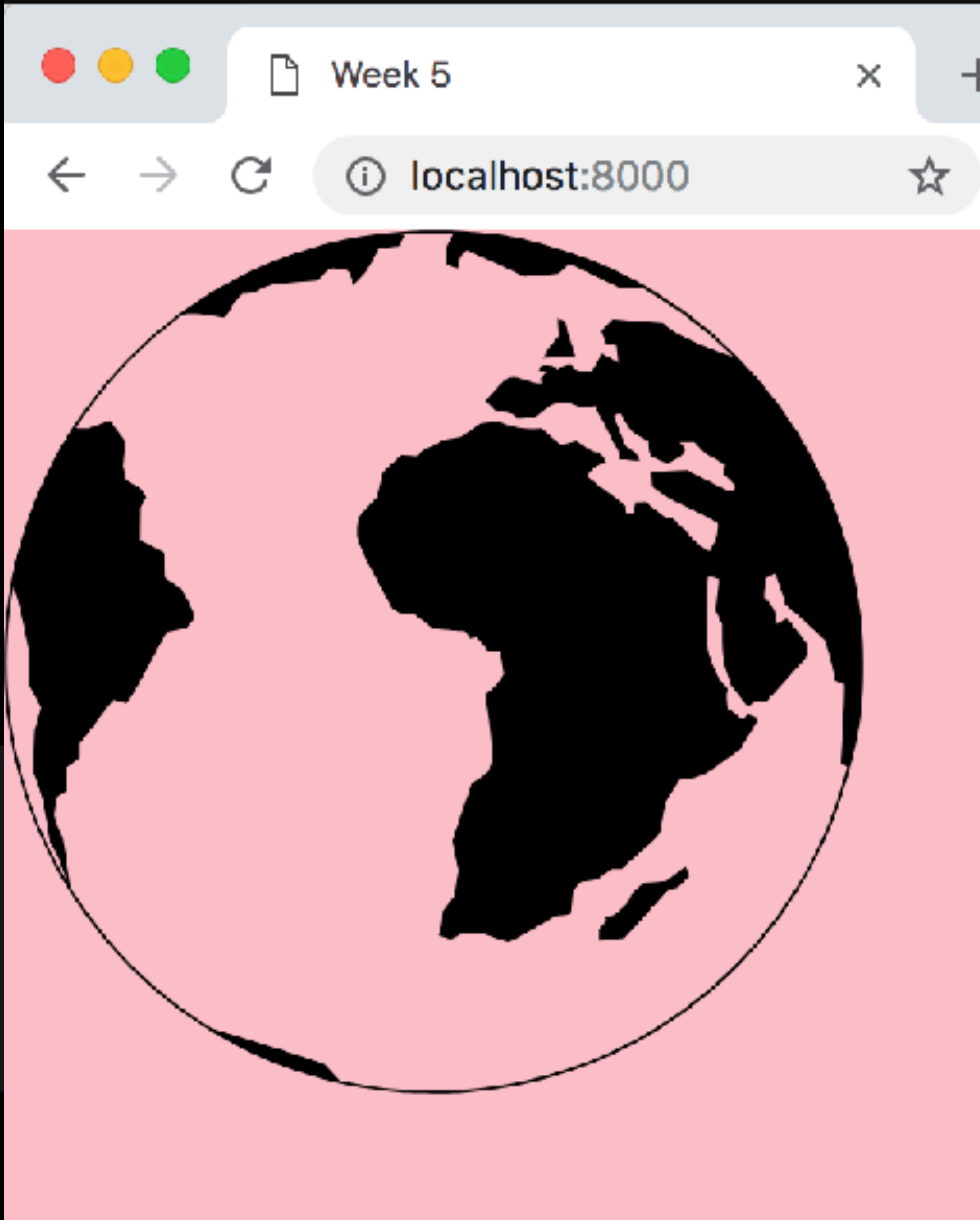
You can change how they look with CSS.

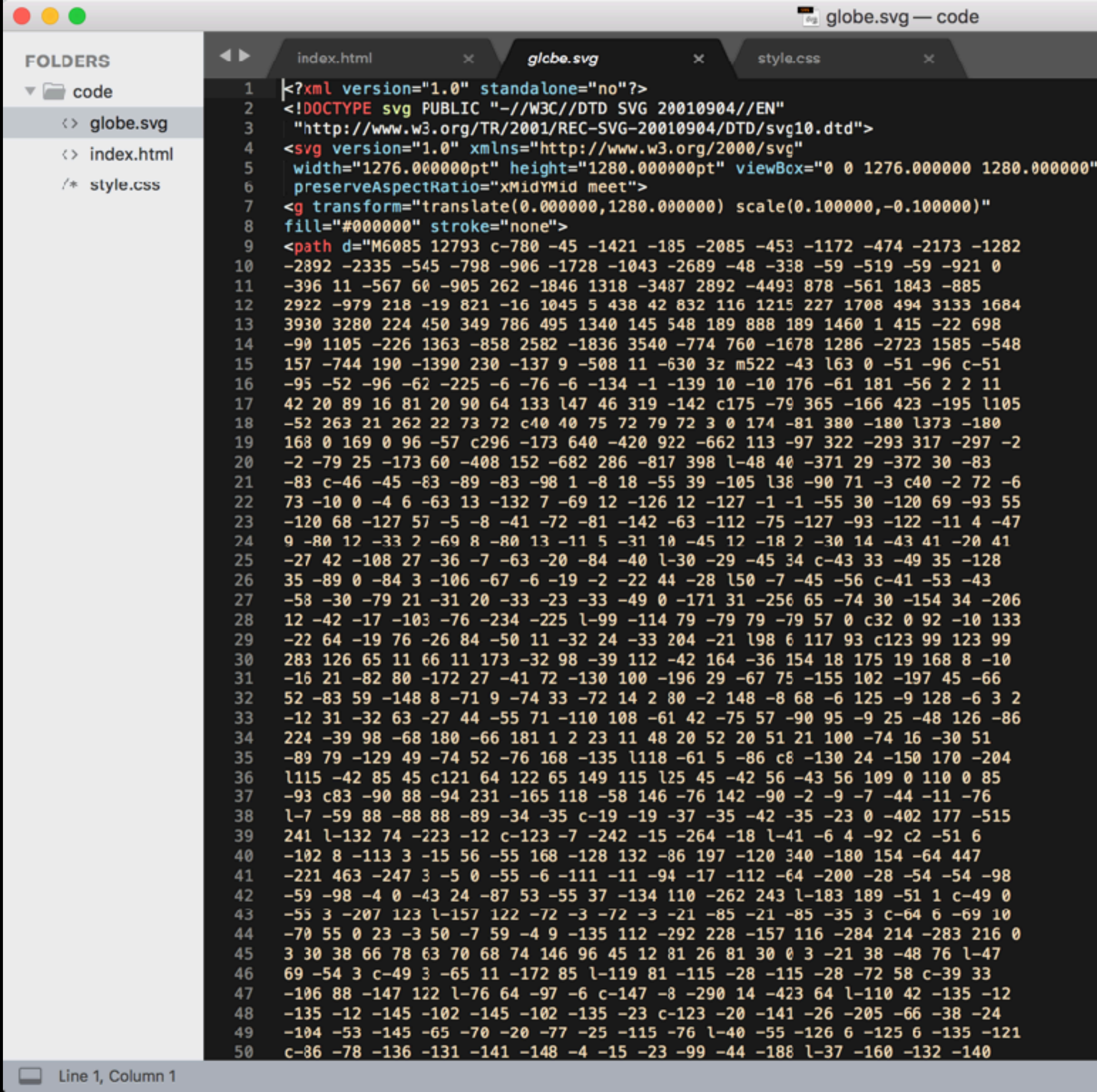
You can add interactivity with JavaScript so you can add interaction design.

SVG

Embedded with the img element - this will act as a static image.
You can not apply styles, animation or javascript:

```
<img src="globe.svg" width=300>
```





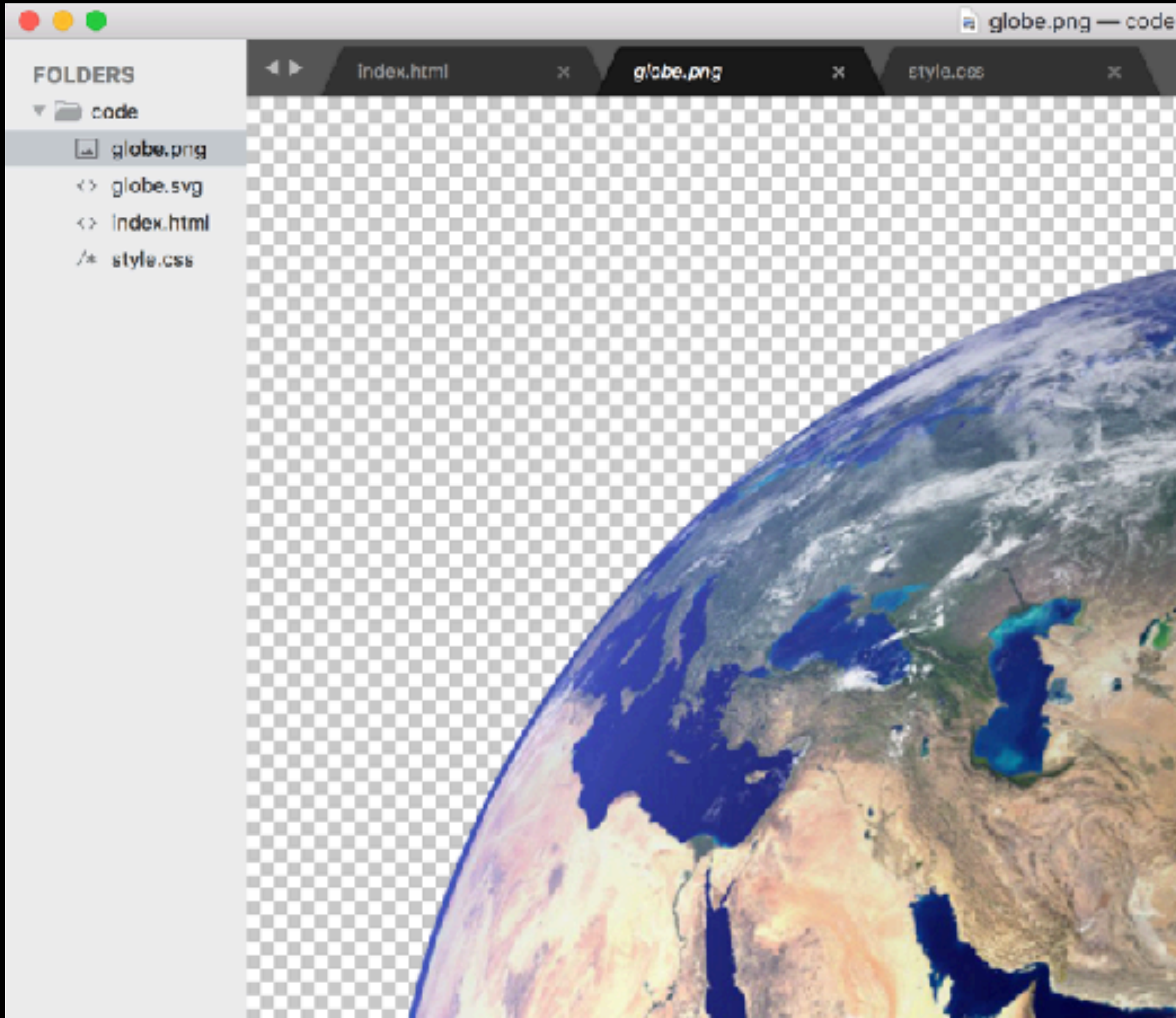
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg version="1.0" xmlns="http://www.w3.org/2000/svg"
 width="1276.000000pt" height="1280.000000pt" viewBox="0 0 1276.000000 1280.000000"
 preserveAspectRatio="xMidYMid meet">
<g transform="translate(0.000000,1280.000000) scale(0.100000,-0.100000)"
fill="#000000" stroke="none">
<path d="M6085 12793 c-780 -45 -1421 -185 -2085 -453 -1172 -474 -2173 -1282
-2892 -2335 -545 -798 -906 -1728 -1043 -2689 -48 -338 -59 -519 -59 -921 0
-396 11 -567 60 -905 262 -1846 1318 -3487 2892 -4493 878 -561 1843 -885
2922 -979 218 -19 821 -16 1045 5 438 42 832 116 1215 227 1708 494 3133 1684
3930 3280 224 450 349 786 495 1340 145 548 189 888 189 1460 1 415 -22 698
-90 1105 -226 1363 -858 2582 -1836 3540 -774 760 -1678 1286 -2723 1585 -548
157 -744 190 -1390 230 -137 9 -508 11 -630 3z m522 -43 l63 0 -51 -96 c-51
-95 -52 -96 -62 -225 -6 -76 -6 -134 -1 -139 10 -10 176 -61 181 -56 2 2 11
42 20 89 16 81 20 90 64 133 l47 46 319 -142 c175 -79 365 -166 423 -195 l105
-52 263 21 262 22 73 72 c40 40 75 72 79 72 3 0 174 -81 380 -180 l373 -180
168 0 169 0 96 -57 c296 -173 640 -420 922 -662 113 -97 322 -293 317 -297 -2
-2 -79 25 -173 60 -408 152 -682 286 -817 398 l-48 40 -371 29 -372 30 -83
-83 c-46 -45 -83 -89 -83 -98 1 -8 18 -55 39 -105 l38 -90 71 -3 c40 -2 72 -6
73 -10 0 -4 6 -63 13 -132 7 -69 12 -126 12 -127 -1 -1 -55 30 -120 69 -93 55
-120 68 -127 57 -5 -8 -41 -72 -81 -142 -63 -112 -75 -127 -93 -122 -11 4 -47
9 -80 12 -33 2 -69 8 -80 13 -11 5 -31 10 -45 12 -18 2 -30 14 -43 41 -20 41
-27 42 -108 27 -36 -7 -63 -20 -84 -40 l-30 -29 -45 34 c-43 33 -49 35 -128
35 -89 0 -84 3 -106 -67 -6 -19 -2 -22 44 -28 l50 -7 -45 -56 c-41 -53 -43
-58 -30 -79 21 -31 20 -33 -23 -33 -49 0 -171 31 -256 65 -74 30 -154 34 -206
12 -42 -17 -103 -76 -234 -225 l-99 -114 79 -79 79 -79 57 0 c32 0 92 -10 133
-22 64 -19 76 -26 84 -50 11 -32 24 -33 204 -21 l98 6 117 93 c123 99 123 99
283 126 65 11 66 11 173 -32 98 -39 112 -42 164 -36 154 18 175 19 168 8 -10
-16 21 -82 80 -172 27 -41 72 -130 100 -196 29 -67 75 -155 102 -197 45 -66
52 -83 59 -148 8 -71 9 -74 33 -72 14 2 80 -2 148 -8 68 -6 125 -9 128 -6 3 2
-12 31 -32 63 -27 44 -55 71 -110 108 -61 42 -75 57 -90 95 -9 25 -48 126 -86
224 -39 98 -68 180 -66 181 1 2 23 11 48 20 52 20 51 21 100 -74 16 -30 51
-89 79 -129 49 -74 52 -76 168 -135 l118 -61 5 -86 c8 -130 24 -150 170 -204
l115 -42 85 45 c121 64 122 65 149 115 l25 45 -42 56 -43 56 109 0 110 0 85
-93 c83 -90 88 -94 231 -165 118 -58 146 -76 142 -90 -2 -9 -7 -44 -11 -76
l-7 -59 88 -88 88 -89 -34 -35 c-19 -19 -37 -35 -42 -35 -23 0 -402 177 -515
241 l-132 74 -223 -12 c-123 -7 -242 -15 -264 -18 l-41 -6 4 -92 c2 -51 6
-102 8 -113 3 -15 56 -55 168 -128 132 -86 197 -120 340 -180 154 -64 447
-221 463 -247 3 -5 0 -55 -6 -111 -11 -94 -17 -112 -64 -200 -28 -54 -54 -98
-59 -98 -4 0 -43 24 -87 53 -55 37 -134 110 -262 243 l-183 189 -51 1 c-49 0
-55 3 -207 123 l-157 122 -72 -3 -72 -3 -21 -85 -21 -85 -35 3 c-64 6 -69 10
-70 55 0 23 -3 50 -7 59 -4 9 -135 112 -292 228 -157 116 -284 214 -283 216 0
3 30 38 66 78 63 70 68 74 146 96 45 12 81 26 81 30 0 3 -21 38 -48 76 l-47
69 -54 3 c-49 3 -65 11 -172 85 l-119 81 -115 -28 -115 -28 -72 58 c-39 33
-106 88 -147 122 l-76 64 -97 -6 c-147 -8 -290 14 -423 64 l-110 42 -135 -12
-135 -12 -145 -102 -145 -102 -135 -23 c-123 -20 -141 -26 -205 -66 -38 -24
-104 -53 -145 -65 -70 -20 -77 -25 -115 -76 l-40 -55 -126 6 -125 6 -135 -121
c-86 -78 -136 -131 -141 -148 -4 -15 -23 -99 -44 -188 l-37 -160 -132 -140
```

text editor can
interpret +
display XML

PNG

Portable Network Graphics

is a **raster-**graphics file-format that supports lossless data compression. PNG was developed as an improved, non-patented replacement for Graphics Interchange Format (GIF)



Lempel-Ziv Welch
compression algo
1997, Unisys

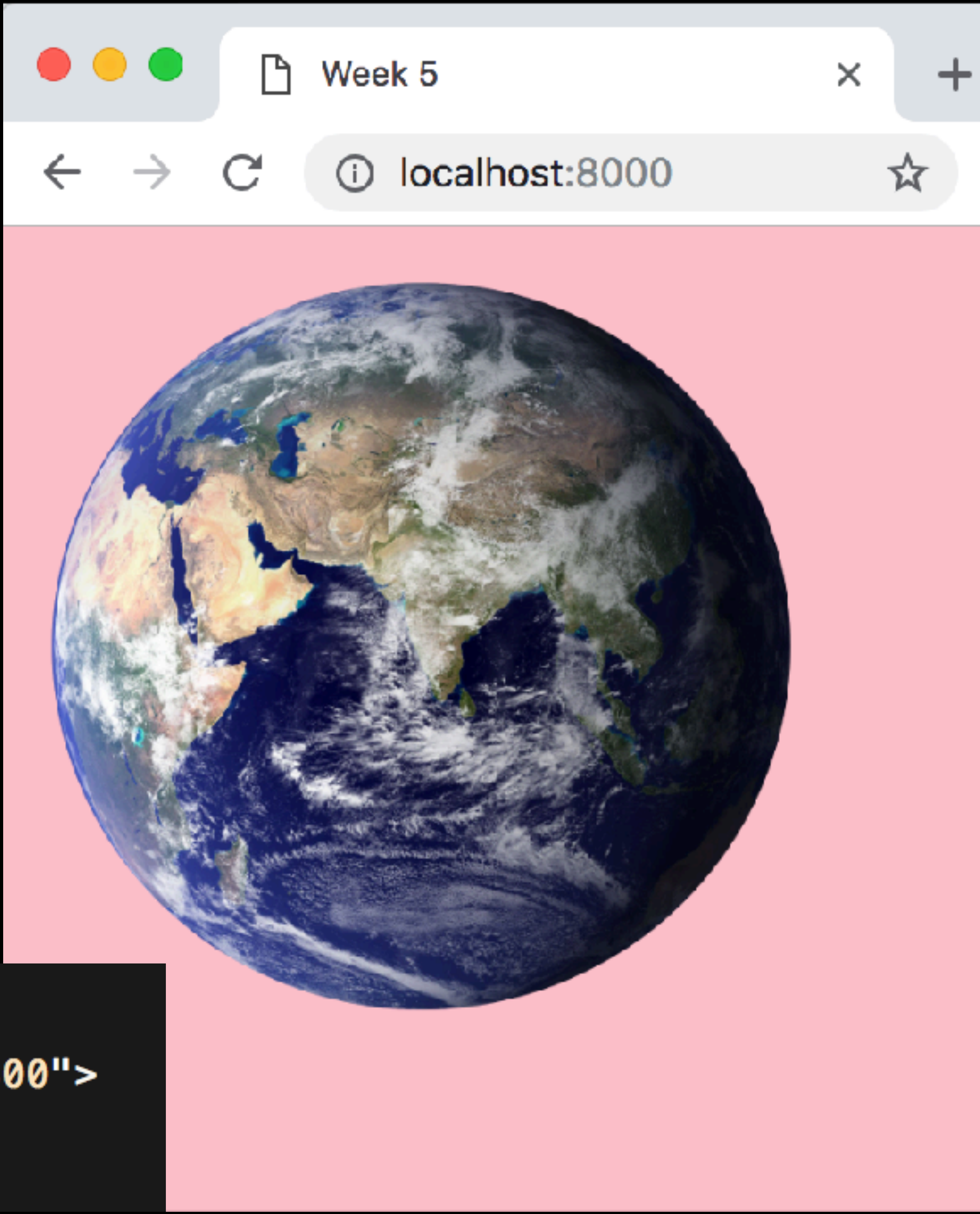
text editor (like browser) interprets as image.

PNG

A raster image is a grid of pixels. Each discreet pixels each have an (R,G,B,A)

red
green
blue
alpha - transparency,
0 -1
in this case it is set to 0.

Using the Canvas we can start to manipulate pixels using Javascript.



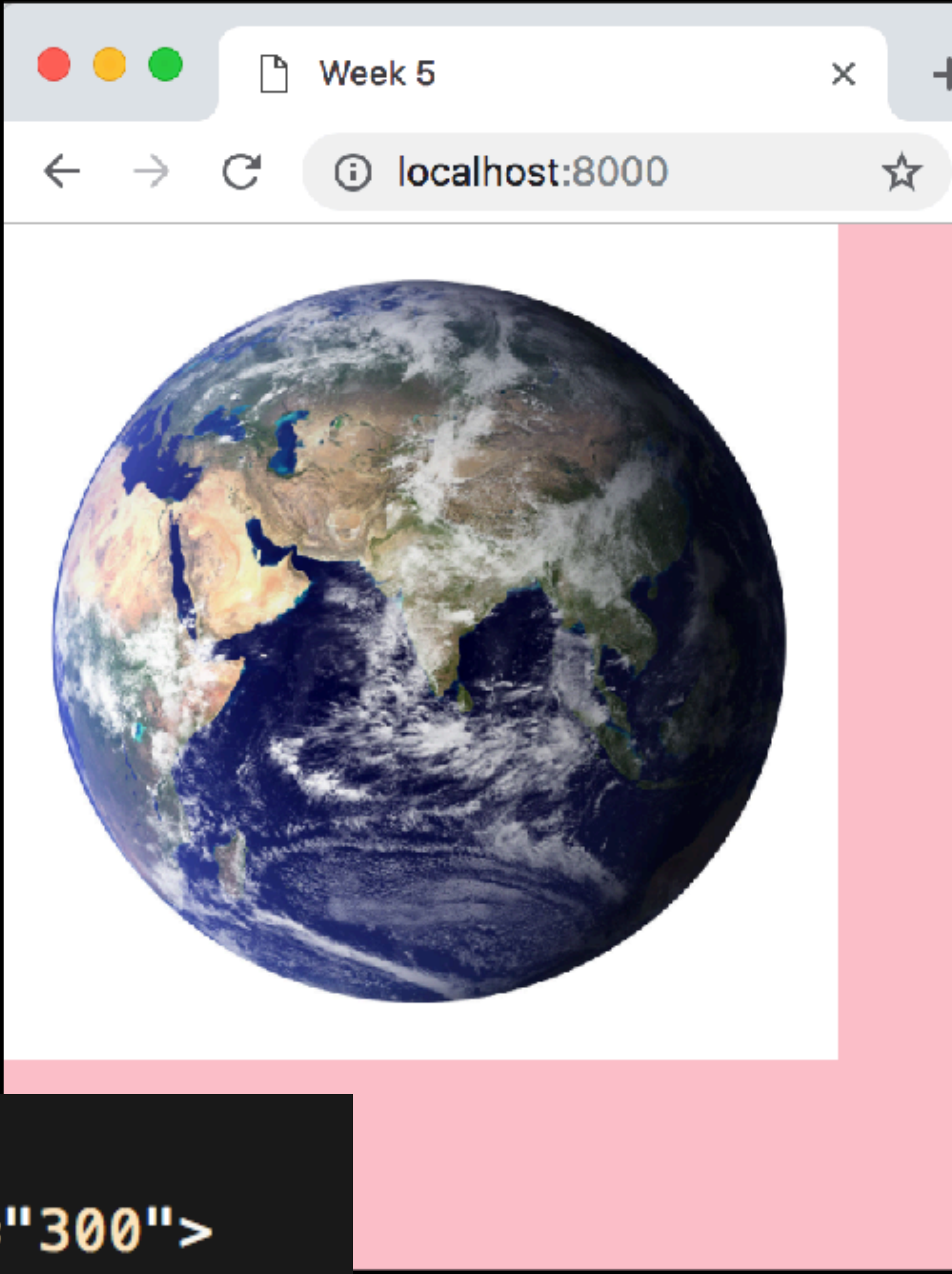
```
<img src="globe.png" width="300">
```

JPG

Joint Photographic Networks Group

a commonly used method of **lossy compression** for digital images, particularly for those images produced by digital photography. The degree of compression can be adjusted, allowing a selectable tradeoff between storage size and image quality.

.jpg files have no alpha channel.

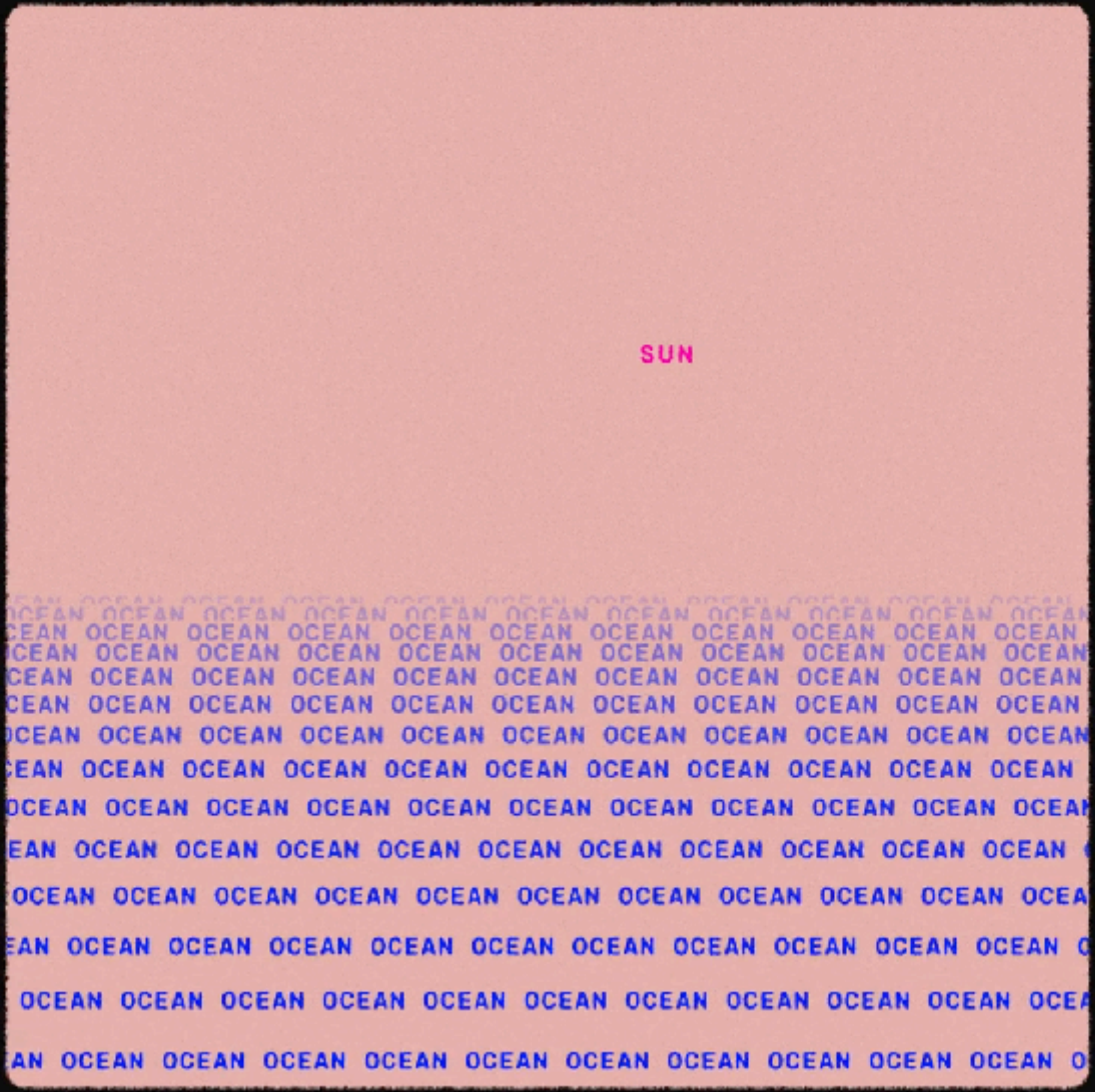


```
<img src="globe.jpg" width="300">
```

GIF

Graphic Interchange Format

Steve Whilhite
1987, CompuServe



Words

Enle Li + Liz Xiong

When applying a transition, you have a few decisions to make, each of which is set with a CSS property:

- Which CSS property to change (**transition-property**) (Required)
- How long it should take (**transition-duration**) (Required)
- The manner in which the transition accelerates (**transition-timing- function**)
- Whether there should be a pause before it starts (**transition-delay**)

Transitions require a **beginning state** and an **end state**. The element as it appears when it first loads is the beginning state. The end state needs to be triggered by a state change such as :hover, :focus, or :active…

CSS Animation Selectors

: hover

: focus

: active

transition-property

identifies the CSS property that is changing and that you want to transition smoothly. In our example, it's the background-color. You can also change the foreground color, borders, dimensions, font- and text-related attributes, and many more. TABLE 18-1 lists the animatab CSS properties as of this writing. The general rule is that if its value is a color, length, or number, that property can be a transition property.

Backgrounds

background-color
background-position

Borders and outlines

border-bottom-color
border-bottom-width
border-left-color
border-left-width
border-right-color
border-right-width
border-top-color
border-top-width
border-spacing
outline-color
outline-width

Color and opacity

color
opacity
visibility

Font and text

font-size
font-weight
letter-spacing
line-height
text-indent
text-shadow
word-spacing
vertical-align

Position

top
right
bottom
left
z-index
clip-path

Transforms

transform
transform-origin

Element box measurements

height
width
max-height
max-width
min-height
min-width
margin-bottom
margin-left
margin-top
padding-bottom
padding-left
padding-right
padding-top

Animateable CSS Properties

Timing Functions

```
. thisAwesomeClass {
        transition-timing-function :

                ease

                linear

                ease-in

                ease-out

                ease-in-out
                step-start

                step-end

                steps

                cubic-bezier(#,#,#,#)

    }
```

the css property

possible values you can set

The **property** and the **duration** are required and form the foundation of a transition, but you can refine it further. There are a number of ways a **transition** can roll out over time.

For example, it could start out fast and then slow down, start out slow and speed up, or stay the same speed all the way through, just to name a few possibilities. I think of it as the transition "style," but in the spec, it is known as the timing function or easing function.

The timing function you choose can have a big impact on the feel and believability of the animation, so if you plan on using transitions and CSS animations, it is a good idea to get familiar with the options.

ease

Starts slowly, accelerates quickly, and then slows down at the end. This is the default value and works just fine for most short transitions.

linear

Stays consistent from the transition's beginning to end. Because it is so consistent, some say it has a mechanical feeling.

ease-in

Starts slowly, then speeds up.

ease-out

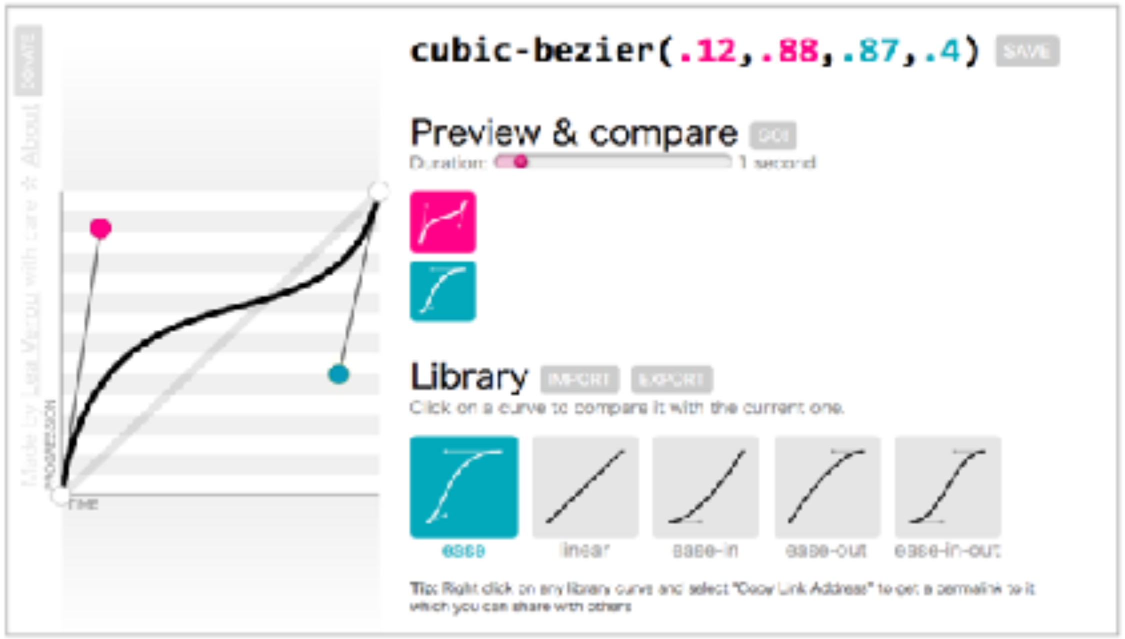
Starts out fast, then slows down.

ease-in-out

Starts slowly, speeds up, and then slows down again at the very end. It is similar to ease, but with less pronounced acceleration in the middle.

You can see that the ease curve is a tiny bit flat in the beginning, gets very steep (fast), then ends flat (slow). The linear keyword, on the other hand, moves at a consistent rate for the whole transition.

You can get the feel of your animation just right by creating a custom curve. The site Cubic-Bezier.com is a great tool for playing around with transition timing and generating the resulting code. The four numbers in the value represent the x and y positions of the start and end Bezier curve handles (the pink and blue dots.



```
cubic-bezier(.12,.88,.87,.4)
```

FIGURE 18-2. Examples of Bezier curves from Cubic-Bezier.com. On the left is my custom curve that starts fast, slows down, and ends fast.

steps(#, start|end)

Divides the transitions into a number of steps as defined by a stepping function. The first
is the number of steps, and the **start** and **end** keywords define whether the change in sta
happens at the begin- ning (**start**) or end of each step. Step animation is especially useful
keyframe animation with sprite images. For a better explanation and examples, I recomm
the article "Using Multi-Step Animations and Transitions," by Geoff Graham on CSS-Tricks
(*css-tricks.com/using-multi- step-animations-transitions/*).

step-start

Changes states in one step, at the beginning of the duration time (the same as **steps(1,st**
The result is a sudden state change, the same as if no transition had been applied at all.

step-end

Changes states in one step, at the end of the duration time (the same as **steps(1,end)**).

The Shorthand transition Property

The authors of the CSS3 spec had the good sense to give us the shorthand transition property to combine all of these properties into one declaration.You've seen this sort of thing with the shorthand border property. Here is the syntax:

```
transition: property duration timing-function delay;
```

```
.theClass {
    transition: background-color 0.3s ease-in-out 0.2s;
}
```

The values for each of the **transition-*** properties are listed out, separated by character spaces. The order isn't important as long as the **duration** (which is required) appears before **delay** (which is optional). If you provide only one time value, it will be assumed to be the duration.

The sub-properties of the **animation** property are:

animation-delay	Configures the delay between the time the element is loaded and the beginning of the animation sequence.
animation-direction	Configures whether or not the animation should alternate direction on each run through the sequence or reset to the start point and repeat itself.
animation-duration	Configures the length of time that an animation should take to complete one cycle.
animation-iteration-count	Configures the number of times the animation should repeat; you can specify infinite to repeat the animation indefinitely.
animation-name	Specifies the name of the **@keyframes** at-rule describing the animation's keyframes.

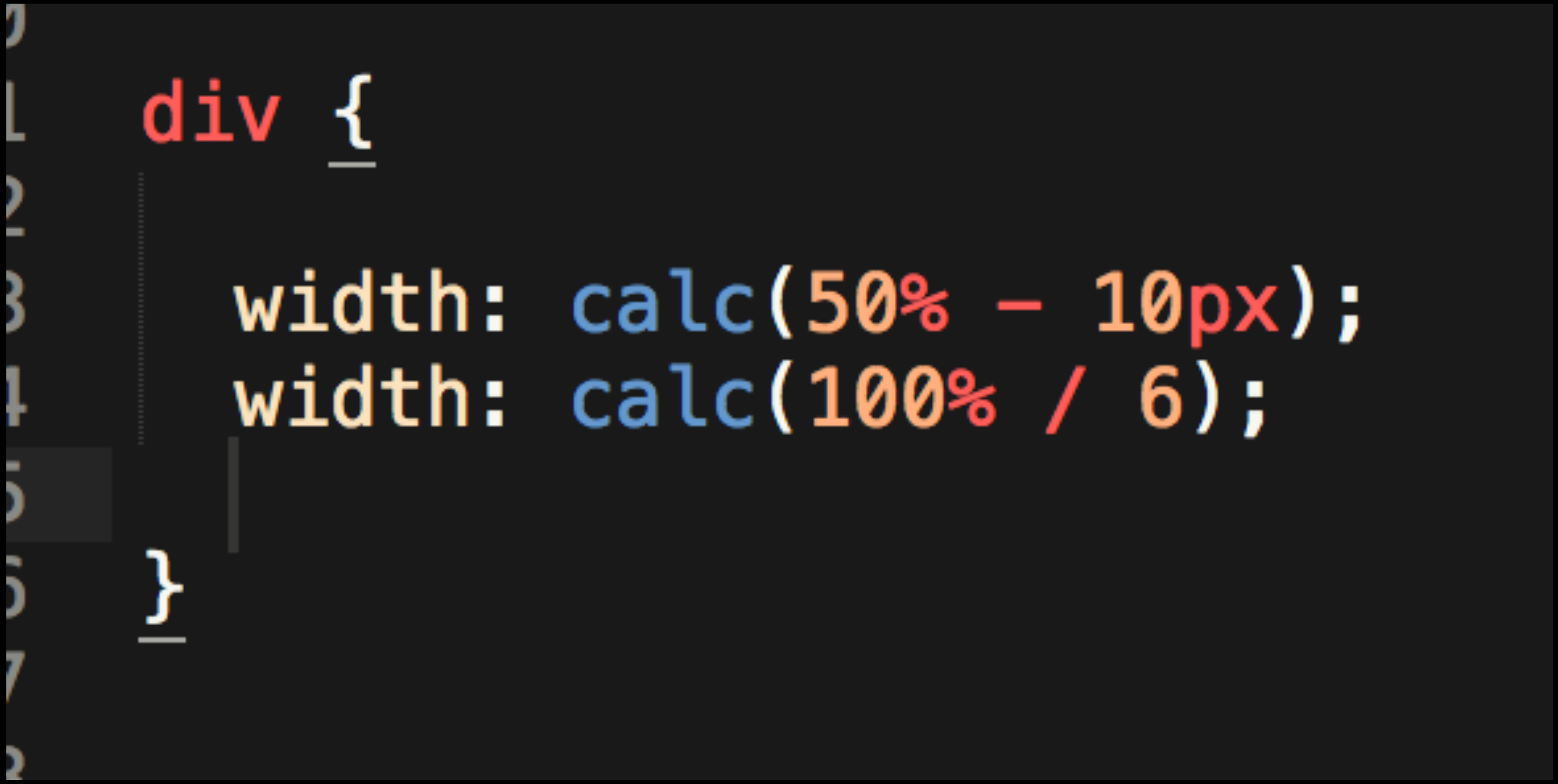
animation-play-state	Lets you pause and resume the animation sequence.
animation-timing-function	Configures the timing of the animation; that is, how the animation transitions through keyframes, by establishing acceleration curves.
animation-fill-mode	Configures what values are applied by the animation before and after it is executing.

@keyframes + animation property

calc

you can use the calc CSS function to define numeric values in terms of expressions:

/* property: calc(expression) */



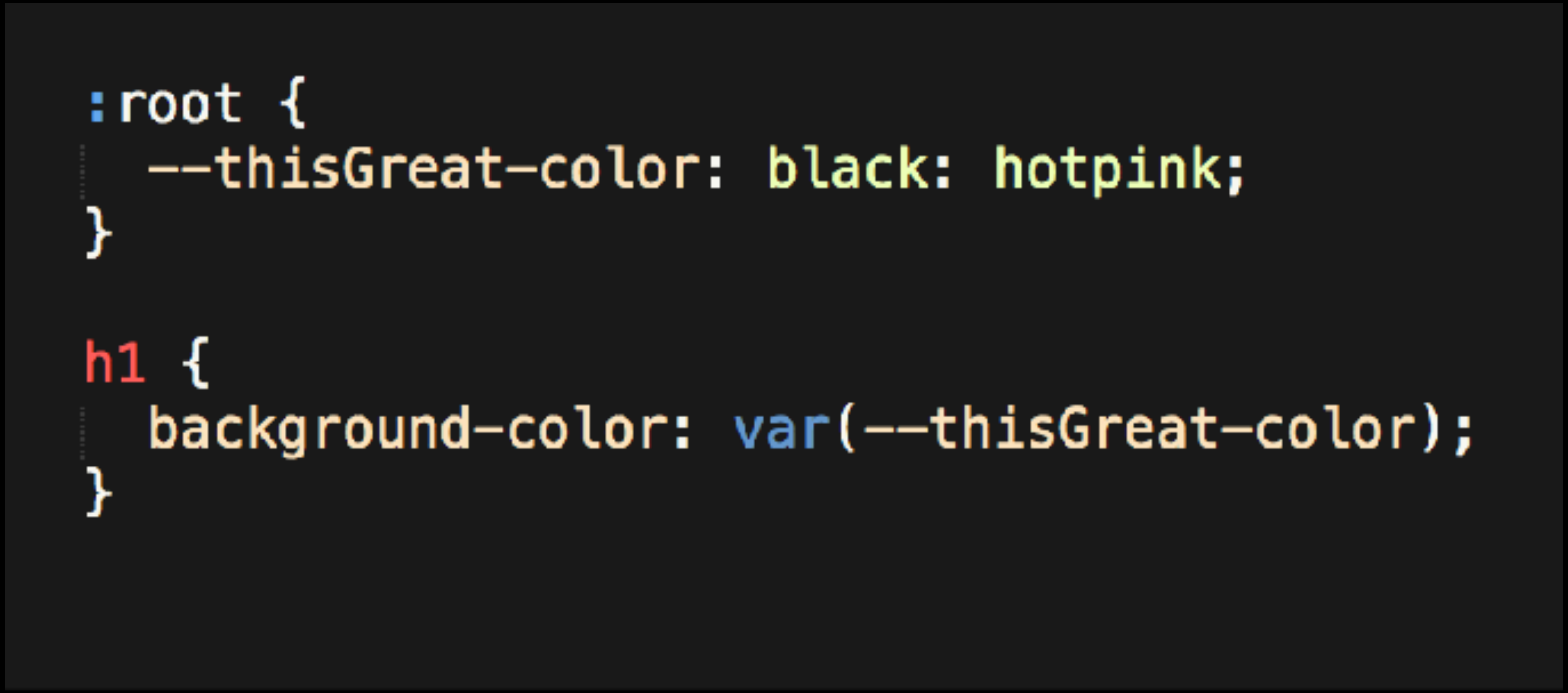
```
div {

  width: calc(50% - 10px);
  width: calc(100% / 6);

}
```

CSS variables

variables are a relatively new CSS feature.

CSS variables are entities defined by CSS authors that contain specific values to be reused throughout a document. They are set using custom property notation (e.g., --main-color: black;) and are accessed using the **var( )** function (e.g., color: var(—main-color););.

CSS variables are subject to the cascade and inherit their value from their parent.



```
:root {
  --thisGreat-color: black: hotpink;
}

h1 {
  background-color: var(--thisGreat-color);
}
```