Icons

- Scalable Vector Graphics
- Icons and text
- Standalone icons
- Styling icons
- Resources

Most images are part of your content, but icons are part of your user interface. They should scale and adapt in the same way that the text of your UI scales and adapts.

Scalable Vector Graphics

When it comes to photographic imagery, there are lots of choices for the image format: JPG, WebP, and AVIF. For non-photographic imagery, you have a choice between the Portable Network Graphics (PNG) format and the Scalable Vector Graphics (SVG) format.

Both PNGs and SVGs are good at dealing with areas of flat color, and they both allow your images to have transparent backgrounds. If you use a PNG you'll probably need to make multiple versions of your image in different sizes and use the `srcset` attribute on your `img` element to make the image responsive. If you use an SVG, it's responsive by default.

PNGs (and JPGs, WebP, and AVIF) are bitmap images. Bitmap images store information as pixels. In an SVG, information is stored as drawing instructions. When the browser reads the SVG file the instructions are converted into pixels. Best of all, these instructions are relative. Regardless of the dimensions you use to describe lines and shapes, everything renders at just the right crispness. Instead of creating multiple SVGs of different sizes you can make one SVG that will work at all sizes. There's no need to use the `srcset` attribute.

```
<img src="image.svg" alt="A description of the image." width="25" height="25">
<img src="image.svg" alt="A description of the image." width="250" height="250">
```

https://codepen.io/web-dot-dev/pen/YzrKoWY

XML is used to write the instructions in an SVG file. This is a markup language, like HTML.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE svg>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg" viewBox="-21 -21 42 42" width="100"
height="100">
  <title>Smiling face</title>
  <circle r="20" fill="yellow" stroke="black"/>
  <ellipse rx="2.5" ry="4" cx="-6" cy="-7" fill="black"/>
  <ellipse rx="2.5" ry="4" cx="6" cy="-7" fill="black"/>
  <path stroke="black" d="M -12,5 A 13.5,13.5,0 0,0 12,5 A 13,13,0 0,1 -12,5"/>
</svg>
```

Smiley face.

You can even put SVG inside HTML.

```
<figure>
  <svg version="1.1" xmlns="http://www.w3.org/2000/svg" viewBox="-21 -21 42 42" width="100"
height="100">
    <title>Smiling face</title>
    <circle r="20" fill="yellow" stroke="black"/>
    <ellipse rx="2.5" ry="4" cx="-6" cy="-7" fill="black"/>
    <ellipse rx="2.5" ry="4" cx="6" cy="-7" fill="black"/>
    <path stroke="black" d="M -12,5 A 13.5,13.5,0 0,0 12,5 A 13,13,0 0,1 -12,5"/>
  </svg>
  <figcaption>
  A description of the image.
  </figcaption>
</figure>
```

If you embed an SVG like that, that's one less request that the browser needs to make. There'll be no wait for the image to download because it arrives with the HTML …*in* the HTML! Also, as you'll soon find out, embedding SVGs like this gives you more control over styling them too.

Icons and text

Icon images often feature simple shapes on a transparent background. SVG is ideal for icons.

If you have a button or a link with text and an icon inside it, the icon is presentational. It's the text that matters. When using an `img` element, an empty `alt` attribute indicates that the image is presentational.

An empty `alt` attribute is not the same as a missing `alt` attribute. Always provide an `alt` attribute even if an image is presentational and the `alt` attribute has no content.

```
<button>
<img src="hamburger.svg" alt="" width="16" height="16">
Menu
</button>
```

https://codepen.io/web-dot-dev/pen/qBPWzam

Because CSS is for presentation, you could put the icon in your CSS as a background image.

```
<button class="menu">
Menu
</button>
```

```
.menu {
  background-image: url(hamburger.svg);
  background-position: 0.5em;
  background-repeat: no-repeat;
  background-size: 1em;
  padding-inline-start: 2em;
}
```

https://codepen.io/web-dot-dev/pen/gOGYNwj

If you put the SVG inside your HTML, use the `aria-hidden` attribute to hide it from assistive technology.

```
<button class="menu">
  <svg aria-hidden="true" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 100 80" width="16"
height="16">
    <rect width="100" height="20" />
    <rect y="30" width="100" height="20"/>
    <rect y="60" width="100" height="20"/>
  </svg>
  Menu
</button>
```

https://codepen.io/web-dot-dev/pen/bGobPwz

If you use the same icon multiple times in one page, it would be inefficient to repeat the entire SVG markup each time. There's an element in SVG called `use` which allows you to “clone” part of an SVG, even from a different SVG element.

Standalone icons

Use text inside your buttons and links if you want their purpose to be clear. You can use an icon without any text but don't assume that everyone understands the meaning of a particular icon. When in doubt, test with real users.

If you decide to use an icon without any accompanying text, the icon is no longer presentational. A background image in CSS is not an appropriate way to display the icon. The icon needs to be given an accessible name in HTML.

If you use an `img` element, the icon gets its accessible name from the `alt` attribute.

Usually the `alt` attribute describes the contents of the image (“Three horizontal lines.”) but with standalone icons, the `alt` attribute describes the meaning of the image (“Menu”).

```
<button>
<img src="hamburger.svg" alt="Menu"" width="16" height="16">
</button>
```

https://codepen.io/web-dot-dev/pen/MWEgMbY

Another option is to put the accessible name on the link or button itself and declare that the image is presentational. Use the `aria-label` attribute to provide the accessible name.

```
<button aria-label="Menu">
<img src="hamburger.svg" alt="" width="16" height="16">
</button>
```

https://codepen.io/web-dot-dev/pen/VwMZJmr

If you put the SVG inside your HTML, use the `aria-label` attribute on the link or button to give it an accessible name and use the `aria-hidden` attribute on the icon.

```
<button aria-label="Menu">
  <svg aria-hidden="true" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 100 80" width="16"
height="16">
    <rect width="100" height="20" />
    <rect y="30" width="100" height="20"/>
```

```
    <rect y="60" width="100" height="20"/>
  </svg>
</button>
```

https://codepen.io/web-dot-dev/pen/OJxLebB

Styling icons

If you embed your SVG icons directly in your HTML you can style parts of them just like any other element in your page. You can't do that if you use an `img` element to display your icons.

In the following example, the `rect` elements inside the SVG have a `fill` value of `blue` to match the styles for the button's text.

```
button {
 color: blue;
}
button rect {
  fill: blue;
}
```

You can apply `hover` and `focus` styles too.

```
button:hover,
button:focus {
  color: red;
}
button:hover rect,
button:focus rect {
  fill: red;
}
```

https://codepen.io/web-dot-dev/pen/rNGBEje

Resources

- If you need to style SVGs that are background images in your CSS, read Una's article on colorizing SVG backgrounds.

- Sara Soueidan has written about accessible icon buttons.
- Scott O'Hara has written about contextually marking up accessible images and SVGs.
- If you're using a graphic design tool to export SVGs, use SVGOMG to optimize the output.

Icons are an important part of your site's branding. Next you'll find out how to make other aspects of your branding responsive through the power of theming.

□□□ #3
□ □□□ □□□ 2026-10-31 22:47:17 UTC
□ cocobook-sync □□□ 2026-11-04 17:39:14 UTC