The picture element

- source
- Image formats
- Image sizes
- Cropping

The previous module demonstrated how the `srcset` attribute allows you to provide different-sized versions of the same image. The browser can then decide which is the right version to use. If you want to change the image completely, you'll need the `picture` element.

In the same way that `srcset` builds upon the `src` attribute, the `picture` element builds upon the `img` element. The `picture` element wraps around an `img` element.

```
<picture>
  <img src="image.jpg" alt="A description of the image.">
</picture>
```

If there is no `img` element nested inside the `picture` element, the `picture` element won't work.

Like the `srcset` attribute, the `picture` element will update the value of the `src` attribute in that `img` element. The difference is that where the `srcset` attribute gives suggestions to the browser, the `picture` element gives commands. This gives you control.

`source`

You can specify multiple `source` elements inside a `picture` element, each one with its own `srcset` attribute. The browser then executes the first one that it can.

Image formats

In this example, there are three different images in three different formats:

```
<picture>
  <source srcset="image.avif" type="image/avif">
  <source srcset="image.webp" type="image/webp">
  <img src="image.jpg" alt="A description of the image."
    width="300" height="200" loading="lazy" decoding="async">
</picture>
```

The first `source` element points to an image in the new AVIF format. If the browser is capable of rendering AVIF images, then that's the image file it chooses. Otherwise, it moves on to the next `source` element.

The second `source` element points to an image in the WebP format. If the browser is capable of rendering WebP images, it will use that image file.

Otherwise the browser will fall back to the image file in the `src` attribute of the `img` element. That image is a JPEG.

This means you can start using new image formats without sacrificing backward compatibility.

In that example, the `type` attribute told the browser which kind of image format was specified.

Image sizes

As well as switching between image formats, you can switch between image sizes. Use the `media` attribute to tell the browser how wide the image will be displayed. Put a media query inside the `media` attribute.

```
<picture>
  <source srcset="large.png" media="(min-width: 75em)">
  <source srcset="medium.png" media="(min-width: 40em)">
  <img src="small.png" alt="A description of the image."
    width="300" height="200" loading="lazy" decoding="async">
</picture>
```

Here you're telling the browser that if the viewport width is wider than `75em` it must use the large image. Between `40em` and `75em` the browser must use the medium image. Below `40em` the browser must use the small image.

This is different to using the `srcset` and `sizes` attributes on the `img` element. In that case you're providing suggestions to the browser. The `source` element is more like a command than a suggestion.

You can also use the pixel density descriptor inside the `srcset` attribute of a `source` element.

```
<picture>
  <source srcset="large.png 1x" media="(min-width: 75em)">
  <source srcset="medium.png 1x, large.png 2x" media="(min-width: 40em)">
  <img src="small.png" alt="A description of the image." width="300" height="200"
loading="lazy" decoding="async"
    srcset="small.png 1x, medium.png 2x, large.png 3x">
</picture>
```

In that example you're still telling the browser what to do at different breakpoints, but now the browser has the option to choose the most appropriate image for the device's pixel density.

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

Cropping

If you only need to serve differently sized versions of the same image, `srcset` is your best option. But if an image doesn't look good at smaller sizes, you can try making a cropped version of the image instead.

The different images might have different width and height ratios to suit their context better. For example, on a mobile browser you may want to serve a crop that's narrow and tall, whereas on a desktop browser, you might want to serve a crop that's wide and short.

Here's an example of a hero image that changes its contents and its shape based on the viewport width. Add `width` and `height` attributes to each `source` element.

```
<picture>
  <source srcset="full.jpg" media="(min-width: 75em)" width="1200" height="500">
  <source srcset="regular.jpg" media="(min-width: 50em)" width="800" height="400">
  <img src="cropped.jpg" alt="A description of the image." width="400" height="400"
loading="eager" decoding="sync">
</picture>
```

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

Remember that you can't change the `alt` attribute depending on the image source. You'll need to write an `alt` attribute that describes both the full size image and the cropped image.

You probably won't need to use the `picture` element for most of your responsive images—the `srcset` and `sizes` attributes on the `img` element cover a lot of use cases. But for those situations

when you need more fine-grained control, the `picture` element is a powerful tool.

There's one kind of image where you might not need either solution: icons. That's what's next.

□□□ #3
□ □□□ □□□ 2026-10-31 22:48:35 UTC
□ cocobook-sync □□□ 2026-11-04 17:39:15 UTC