A LIST APART

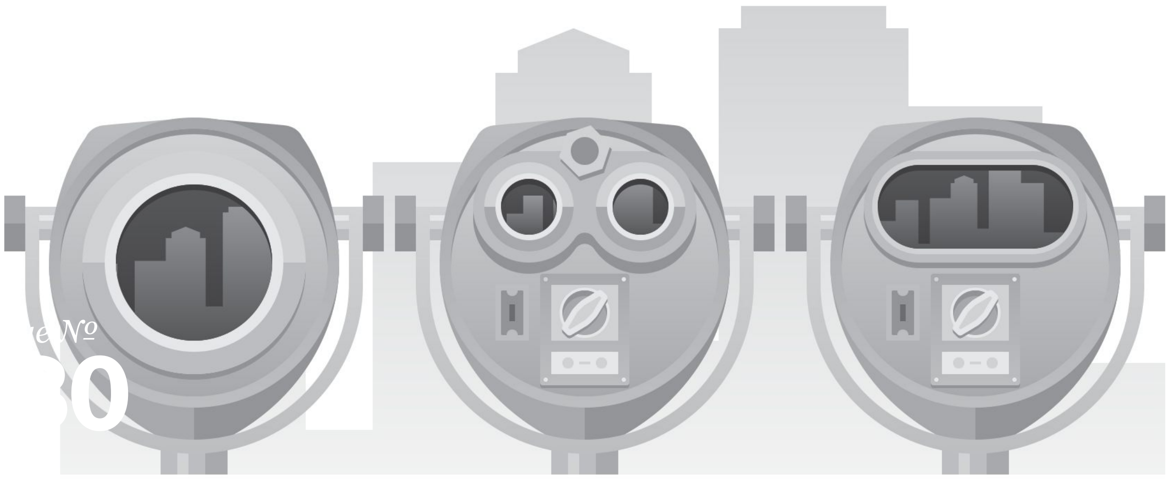

Illustration by Justin Mezzell

Using Responsive Images (Now)

by Chen Hui Jing · October 06, 2015

Published in Responsive Design

Ever since Ethan Marcotte started talking about responsive web design in 2010 *(http://alistapart.com/article/responsive-web-design)*, developers and designers have been scrambling to find ways to deal with the issue of responsive images. It's a thorny problem *(https://css-tricks.com/responsive-images-hard/)* because we're serving the same website, with the same image sources, across a wide range of device widths. Do you want a blurry, pixelated image on a large display? Or do you want to load a huge (but oh-so-pretty) image on your mobile phone? Talk about being stuck between a rock and a hard place.

Loads of smart people, namely the Responsive Issues Community Group (RICG) *(https://www.w3.org/community/respimg/)*, have been working together to solve this conundrum. That's why the `picture` element and the `srcset` and `sizes` attributes are being drafted into the HTML 5.1 specification *(http://www.w3.org/TR/html51/)*. Because we cannot predict where and how users are going to view our websites, we need the browsers themselves to pick the best image for the situation. The new specification will address the following issues:

- Device-pixel-ratio-based selection
- Viewport-based selection
- Art direction-based selection
- Image format-based selection

The spec also introduces two new attributes— `srcset` and `sizes` —to the `img` element. `srcset` lets us declare a set of image sources, which browsers will serve according to certain conditions we specify using descriptors. `x` descriptors indicate the pixel-density of the images, while `w` descriptors indicate the width of the images; the browser will use this information to pick the appropriate image from the list. The `sizes` attribute provides the browser with some context on the size of the image element to be displayed, and must be included when using `srcset` with `w` descriptors. This is especially relevant for variable-width images, which I'll discuss in detail later.

The point is, we now have the option of serving images of different quality or art direction depending on the user's viewport, without some complicated server-side setup. Responsive images will become part and parcel of the HTML specification; eventually, all browsers will support this solution.

Now, let's take a tour of the selection types and how you can make them work for you.

Fixed-width images: device-pixel-ratio-based selection

With the introduction of Retina screens, it became necessary to take into account not only screen resolution but also pixel density. Retina screens, 4K displays, UltraHD—all of these have way more pixels packed into them than a standard resolution display of the same size. More pixels = sharper image quality.

If, for some reason, you have an image that will always display at a certain width regardless of screen size—the site logo or a profile image, say—then device-pixel-ratio-based selection is the way to go. The browser will choose which image to load based on its device-pixel ratio.

The `srcset` attribute basically lists the pool of source images from which the browser can pick to load. It's a comma-separated list. The `x` descriptor indicates the device-pixel ratio of the image. Depending on what environment the browser is operating in, it will utilize this information to select the appropriate image. Any browsers that don't understand `srcset` will simply load the image declared in the `src` attribute.

```
<img srcset="crest-383.jpg 1.5x, crest-510.jpg
2x" src="crest-255.jpg" alt="USWNT crest" />
```

An example of a fixed-width image might be a site's logo, which remains the same size regardless of viewport width. Content-related images, however, are usually responsive; their sizes tend to change depending on the viewport. For those types of images, there's a better method.

Fluid-width images: viewport-based selection

For fluid-width images, we use `srcset` with the `w` descriptor and `sizes`. The `w` descriptor tells the browser the width of each image in the list. The `sizes` attribute is also a comma-separated list containing two values. As of the latest specification, if the `srcset` has any images using the `w` descriptor, then the `sizes` attribute must be present as well.

There are two values in the `sizes` attribute. The first is a media condition. The second is the source-size-value, which determines the width of the image given that particular media condition. One important thing to note is that you can't use percentages as the source-size-value; the only relative CSS length you can use is `vw`.

```
<img srcset="uswnt-480.jpg 480w,
             uswnt-640.jpg 640w,
             uswnt-960.jpg 960w,
             uswnt-1280.jpg 1280w"
     sizes="(max-width: 400px) 100vw,
            (max-width: 960px) 75vw,
            640px"
     src="uswnt-640.jpg" alt="USWNT World Cup
victory">
```

Here, I'm telling the browser that for viewport widths up to 400 pixels, make the image 100% of the viewport width. At viewport widths up to 960 pixels, make the image 75% of the viewport width. And for everything above 960 pixels, make the image 640 pixels. If you're unfamiliar with `vw`, take a look at Tim Severien *(https://timseverien.com/)*'s great article explaining viewport units *(https://web-design-weekly.com/2014/11/18/viewport-units-vw-vh-vmin-vmax/)*.

The browser utilizes the information from `srcset` and `sizes` to serve the image that best matches the stated conditions. If my browser's viewport is 600 pixels, it would most likely display the image at `75vw`. The browser will try to load the first image larger than 450 pixels, which is `uswnt-480.jpg`. If I'm on a Retina display with a device-pixel ratio of 2, then the browser will try to load the first image larger than 900 pixels, which should be `uswnt-960.jpg`. We can't be certain of exactly which image will be served because each browser has some leeway in how their algorithm picks an appropriate image based on the information we provide. This is what "viewport-based selection" means.

Because the first two examples display the same image at different quality levels, the `srcset` attribute alone is sufficient. Again, if you're worried about legacy browsers, that's what the `src` is for—those browsers will just treat it as a regular image and load from `src`. If you want to show slightly different images at different widths, for example, showing only the critical parts of an image at smaller widths, then use the `picture` element.

`picture`: art direction-based selection

The `picture` element is like a wrapper for the image and its sources. Browsers still need `img` to recognize that an image needs to be served; without `img`, nothing will render at all. `source` provides the browser alternate versions of the image to display. Art direction-based selection is used for situations when we want a specific image to display at a specific breakpoint. There is no ambiguity in terms of image selection when you use the `picture` element.

```
<picture>
  <source media="(min-width: 960px)"
srcset="ticker-tape-large.jpg">
  <source media="(min-width: 575px)"
srcset="ticker-tape-medium.jpg">
  <img src="ticker-tape-small.jpg" alt="USWNT
ticker-tape parade">
</picture>
```

In this example, when the viewport is larger than 960 pixels, a landscape-oriented version of the image (`ticker-tape-large.jpg`) is loaded. For widths larger than 575 pixels, the browser loads a cropped portrait-oriented image (`ticker-tape-medium.jpg`) instead. And for widths smaller than 575 pixels, the image (`ticker-tape-small.jpg`) loaded has been cropped to focus just on one player.

The `picture` element is backwards compatible; browsers that don't support the `picture` element will display `img` as usual. All standard attributes for images, like `alt` , should be applied to `img` , not `picture` .

`source` : image format-based selection

A number of new image formats have come into existence in recent years. These new image formats offer better quality at lower file sizes. Sounds good, right? Until you realize that none of these formats is universally supported across all browsers. Google's WebP performs very well, but is only natively supported by Chrome and Opera. JPEG-XR, originally known as HD Photo, was a proprietary image format released by Microsoft, supported only by Internet Explorer. If you want to learn more, Zoltan Hawryluk wrote an in-depth examination *(http://www.useragentman.com/blog/2015/01/14/using-webp-jpeg2000-jpegxr-apng-now-with-picturefill-and-modernizr/)* of these new formats.

```
<picture>
  <source type="image/vnd.ms-photo"
src="wwc2015.jxr">
  <source type="image/jp2" src="wwc2015.jp2">
  <source type="image/webp" src="wwc2015.webp">
  <img src="wwc2015.png" alt="WWC 2015">
</picture>
```

Because `source` also has a type attribute, by specifying the MIME type of each image, browsers can choose the first source that has a type attribute of a supported MIME type. The order of the source matters, in this case, but if the browser doesn't recognize any of the image types, it will just fall back to the original `img` element.

Can I use all this right now?

At time of writing, `picture` is supported by the latest stable releases of Firefox, Chrome, and Opera. Safari and Internet Explorer do not support `picture` natively at all. `srcset` fares slightly better, with full support on the latest stable releases of Firefox, Chrome, and Opera, and partial support on Safari 8 and Internet Explorer Edge, where they allow for use of the `x` descriptors for resolution switching, but not the `w` descriptors.

Quite a few polyfills out there address this support problem. The most well-known is probably Scott Jehl's picturefill *(http://scottjehl.github.io/picturefill/)*. I currently use Alexander Farkas's respimage *(https://github.com/aFarkas/respimage)* on my own site. We've finally reached a point where we've agreed on a solution for how to deal with responsive images, and that solution is getting implemented across all major browsers. Even though the specification is still being refined, we're really close to a native responsive solution.

And if you want to stay up-to-the-minute current, I highly recommend checking out the Responsive Issues Community Group *(https://www.w3.org/community/respimg/)*. You can also sign up for their newsletter *(https://responsiveimages.org)* or follow them on Twitter *(https://twitter.com/respimg)*.

About the Author

Chen Hui Jing

Hui Jing Chen is a self-taught designer and developer from Singapore. Reducing lines of code in her web projects makes her extremely happy. She used to play basketball full-time and launched her web career during downtime between training sessions.

ISSN 1534-0295 · Copyright © 1998–2016 A List Apart & Our Authors