7B

RESPONSIVE IMAGE MARKUP

OVERVIEW

- How responsive images work
- High-density displays
- Variable-width images
- Art direction with the `picture` element
- Alternative image types

About Responsive Images

Responsive image markup allows us to **deliver images that are appropriate for the user's viewing environment**.

How it works:

- You provide multiple image versions.
- You specify sizes and preferences with responsive image markup.
- The browser makes the final selection.

About Responsive Images (cont’d)

Responsive image markup addresses four scenarios:

- Sending larger images to **high-density screens**
- Sending a variety of image sizes for **different screen sizes**
- Sending **alternately cropped images** for small screens (art direction)
- Providing **alternative image formats** with smaller file sizes to supporting browsers

High-Density Displays
(x-descriptor)

- Screens and images are made up of squares of colored light called **pixels**. T

- On standard screens, the pixels in an image and CSS measurements map one-to-one with the **device pixels**.

- High-density displays fit **2x**, **3x**, or **4x** the number of device pixels in the same amount of space.

- High-density devices use a measurement called **reference pixels** for page layout. Images and CSS measurements map with the reference pixel grid (not the tiny device pixels).

High-Density Displays (cont'd)

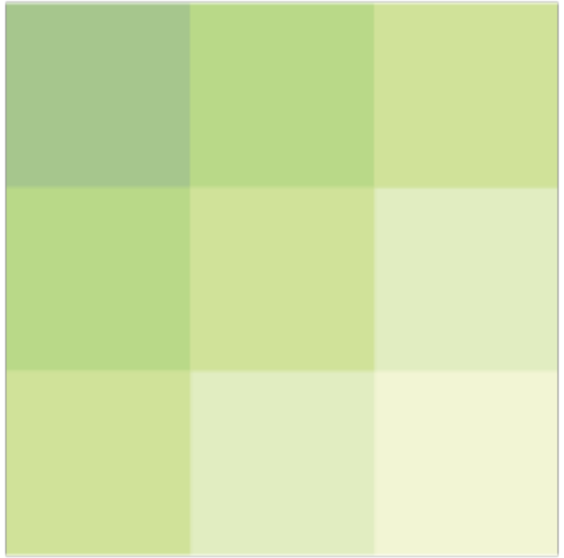
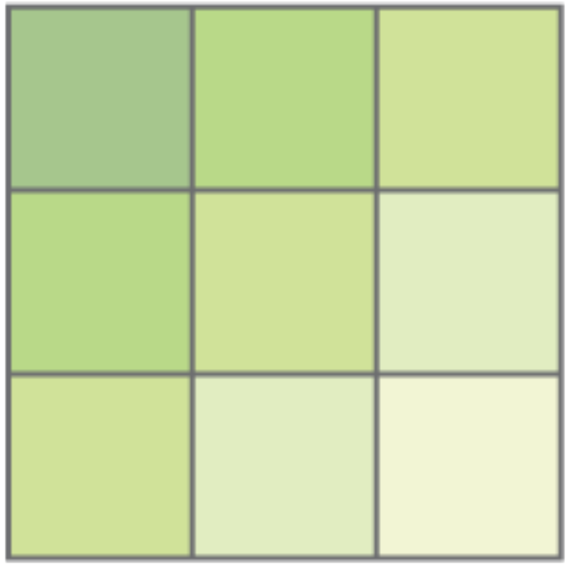
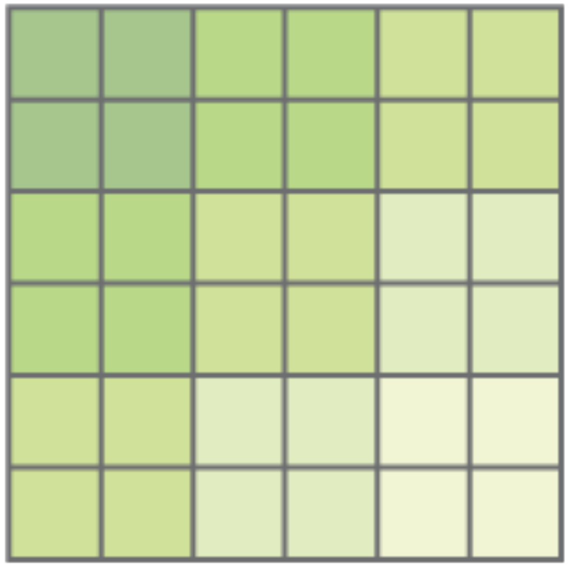


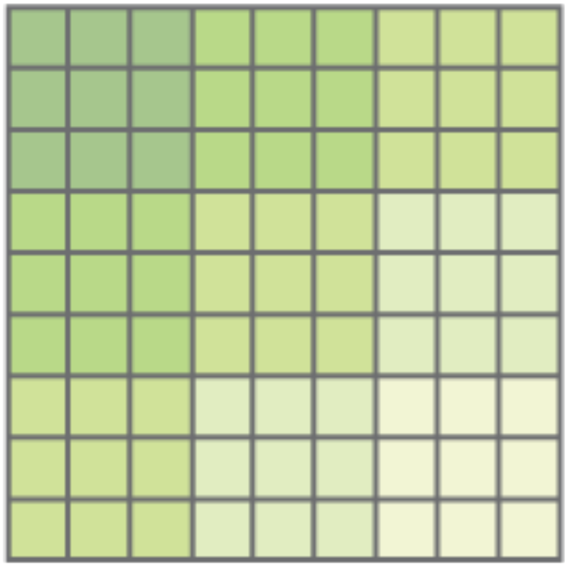
Image or object =
3 x 3 *reference or CSS pixels*



1:1 device-pixel-ratio (1x)
3 x 3 *device pixels*, indicated by grid



2:1 device-pixel-ratio (2x)
6 x 6 device pixels



3:1 device-pixel-ratio (3x)
9 x 9 device pixels

High-Density Displays (cont’d)

In order **for images to look sharp** and not blurry on high-density displays, they need to **map with the device pixels**.

Example:
For an image that displays at **200 pixels wide in the layout**, provide these versions:

- **200px** wide image for standard screens
- **400px** wide image for **2x** high-density screens
- **600px** wide image for **3x** high-density screens
- **800px** wide image for **4x** high-density screens

High-Density Displays (cont’d)

The **srcset** attribute in the **img** element lists image alternatives based on pixel density, specified with an **x-descriptor** (#**x**):

```
<img src="/images/photo-200px.jpg" alt=""
     srcset="/images/photo-400px.jpg 2x,
             /images/photo-600px.jpg 3x" >
```

- The **src** attribute is required. Use it for the standard image.
- The **srcset** value is a comma-separated list of alternative image URLs followed by an x-descriptor.
- A device with a standard screen density gets only the 200px wide version of the image. Higher densities get larger images.

High-Density Displays (cont’d)

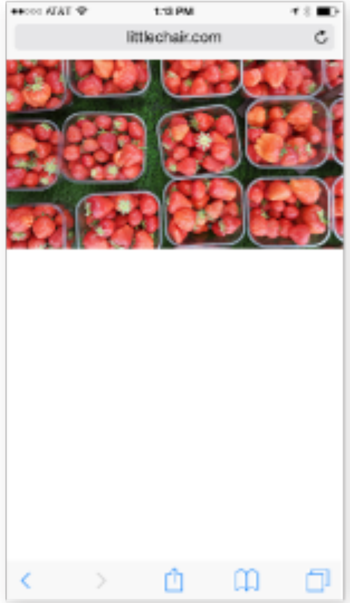
X-descriptors tell the browser to make an image selection based on **screen resolution only**.

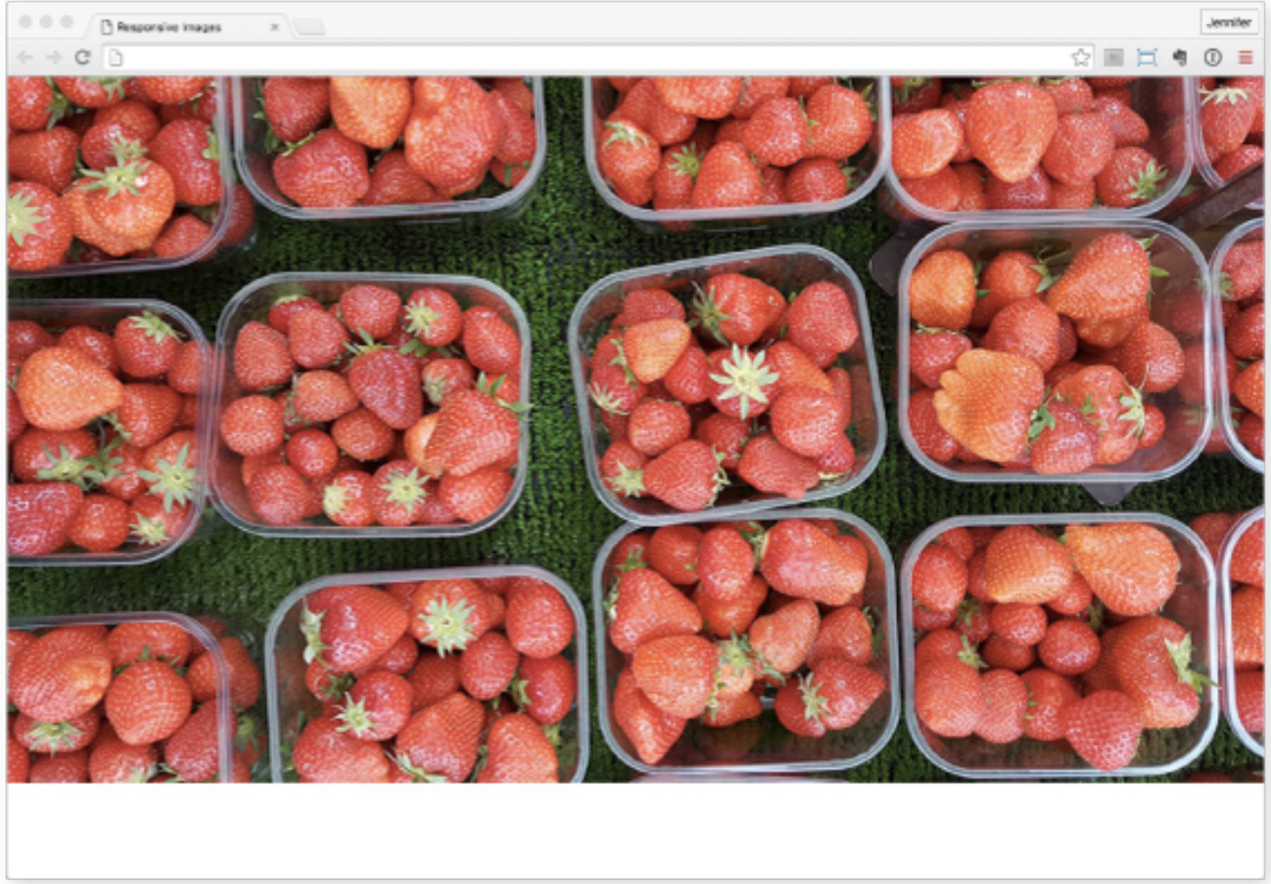
Use x-descriptors for images that stay the same pixel dimensions regardless of the screen size and layout changes.

Variable-Width Images

(w-descriptor)

When images resize based on the size of the screen or browser window (such as in a responsive layout). Use the **`srcset`** attribute with a **w-descriptor** and the **`sizes`** attribute to make a **viewport-based selection**:



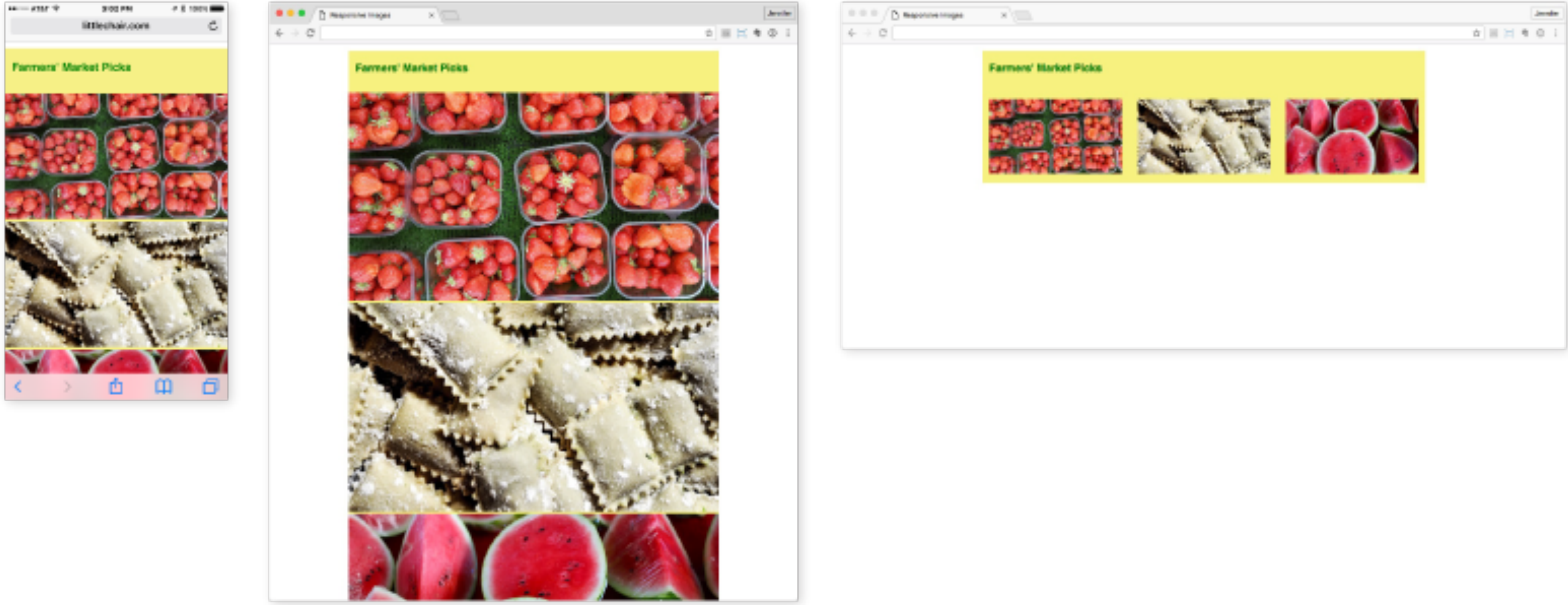


Variable-Width Images (cont’d)

```
<img src="strawberries-640.jpg"
     alt="baskets of ripe strawberries"
     srcset="strawberries-480.jpg 480w,
             strawberries-960.jpg 960w,
             strawberries-1280.jpg 1280w,
             strawberries-2400.jpg 2400w"
     sizes="100vw">
```

- The `src` attribute is required and specifies the **default image size**.
- The value of `srcset` is a comma-separated list of images and w-descriptors.
- **w-descriptors** list the **actual image width (in pixels)**.
- The `sizes` attribute specifies the size at which the image appears in the layout. In this example, it is 100% the width of the browser (100 viewport-width units).

Variable-Width Images (cont'd)



If the image proportion changes based on the size of the viewport, use a **media condition** in the **`sizes`** attribute to test for the screen width and determine the image size:

```
sizes = (media-feature: condition) length
```

Variable-Width Images (cont'd.)

```
<img src="strawberries-640.jpg" alt="baskets of ripe strawberries"
  srcset="strawberries-240.jpg 240w,
          strawberries-480.jpg 480w,
          strawberries-672.jpg 672w"
  sizes="(max-width: 480px) 100vw,
         (max-width: 960px) 70vw,
         240px">
```

- If the viewport is 480 pixels or less, the image fills 100% of the viewport width.
- If the viewport is wider than 480 pixels but less than 960 pixels (`max-width: 960px`), the image appears at 70% of the viewport width.
- If the viewport is wider than 960 pixels, the image gets sized to exactly 240 pixels.
- Now that the browser knows the width of the viewport and how big the image will appear within it, it can select the most appropriate image from the `srcset` list to download.

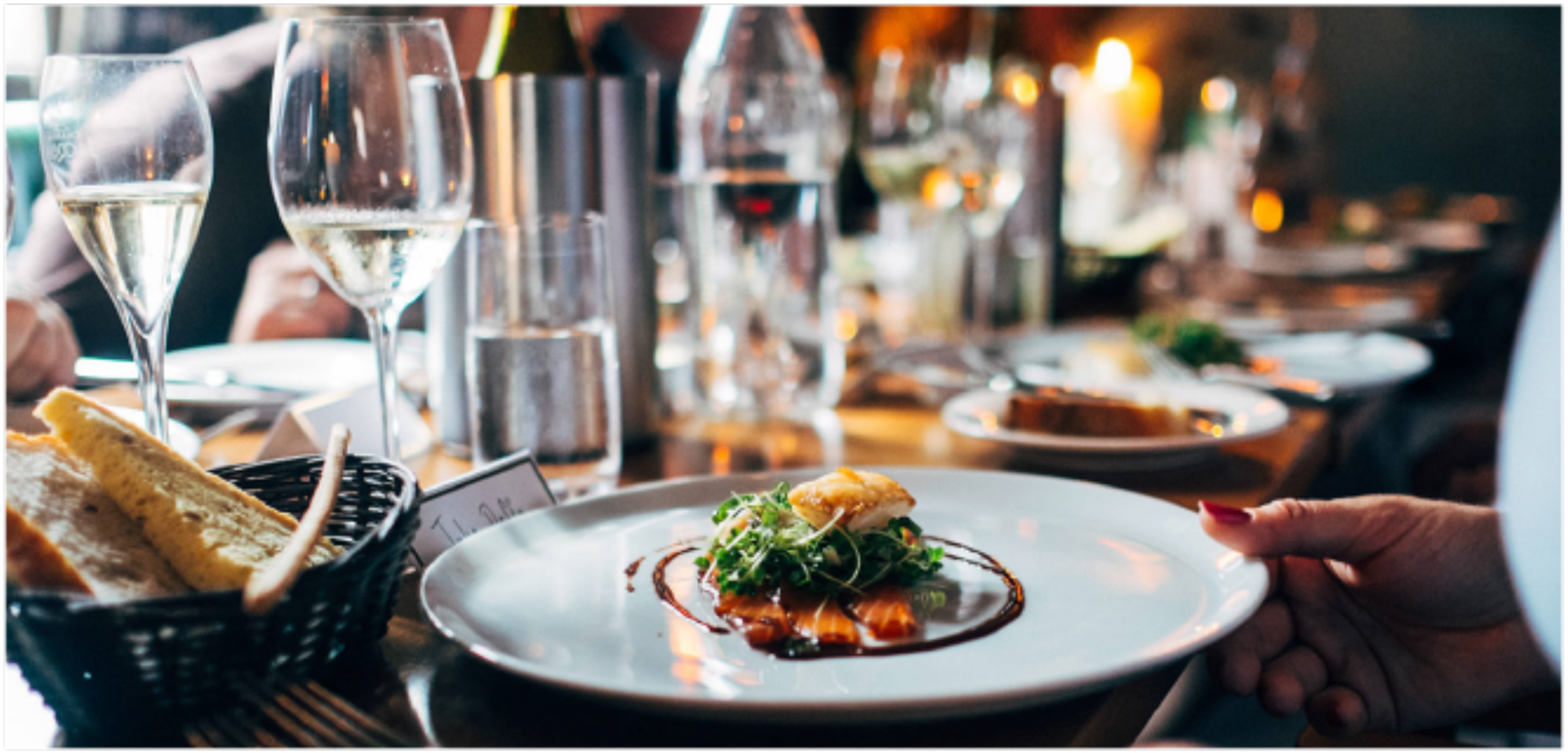
Art Direction Selection
(picture element)

Details may be lost in images scaled down to fit small screens.

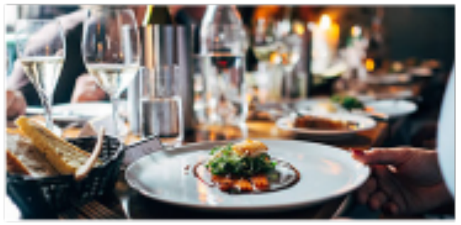
To change the image itself rather than just resizing it, make an **art-direction-based selection** with the **`picture`** and **`source`** elements:

- Provide differently cropped versions.
- Provide both landscape (wide) and portrait (tall) versions.
- Change or remove text in images to keep it legible.

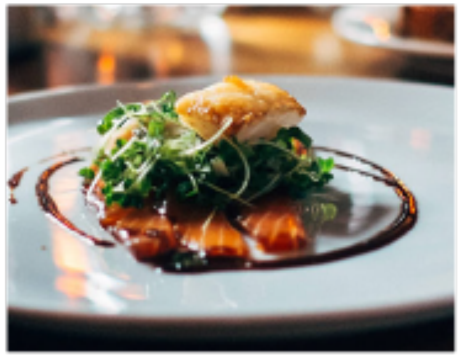
Art Direction Selection (cont'd)



That dinner looks delicious on desktop browsers.
(1280px wide)



Detail is lost when the full image is shrunk down on small devices.
(300px wide)

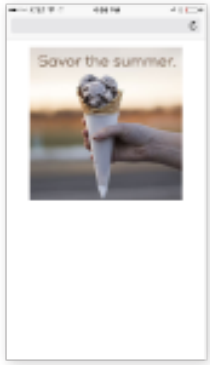


Cropping to the most important detail may make better sense.
(300px wide)

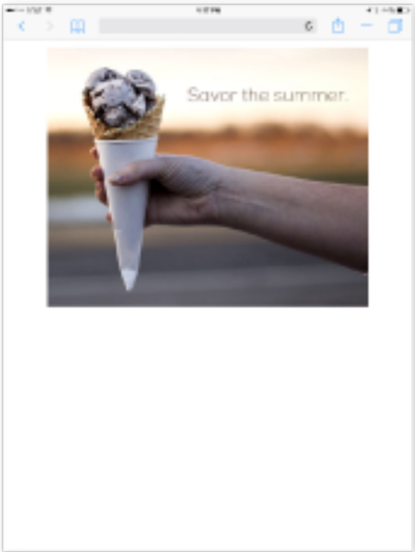
Art Direction Selection (cont'd)

```
<picture>
  <source media="(min-width: 1024px)" srcset="icecream-large.jpg">
  <source media="(min-width: 760px)" srcset="icecream-medium.jpg">
  <img src="icecream-small.jpg" alt="hand holding ice cream cone and text
that reads Savor the Summer">
</picture>
```

iPhone

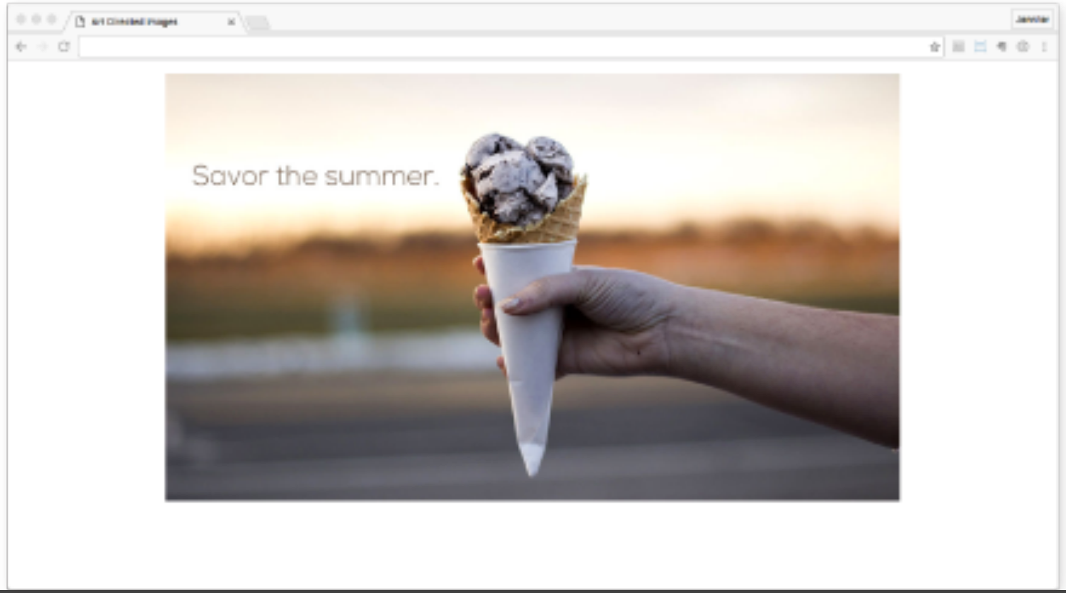


iPad



- The **picture** element has no attributes; it is a wrapper for **source** and **img** elements.
- The **img** element is **required** and must go last in the list.
- **source** elements test for media conditions (**media** attribute) and provide alternative image options for each size (**srcset**).

Chrome browser on desktop



Alternative Image Formats
(type Attribute)

- New, efficient image formats (WebP, JPEG 2000, and JPEG XR) are available, but aren't supported by every browser.

- Use the **`picture`** and **`source`** elements with the **`type`** attribute for **image-format-based selections**:

```
<picture>
  <source type="image/webp" srcset="pizza.webp">
  <source type="image/jxr" srcset="pizza.jxr">
  <img src="pizza.jpg" alt="">
</picture>
```

- Browsers use the first image format they support and don't download the rest.

Browser Support

- **`picture`**, **`source`**, **`srcset`**, and **`sizes`** are supported by all current browser versions.

- Every technique includes a standards **`img`** element as a reliable fallback.

- For better support, use the **Picturefill** polyfill script that makes older browsers support the new markup (scottjehl.github.io/picturefill).