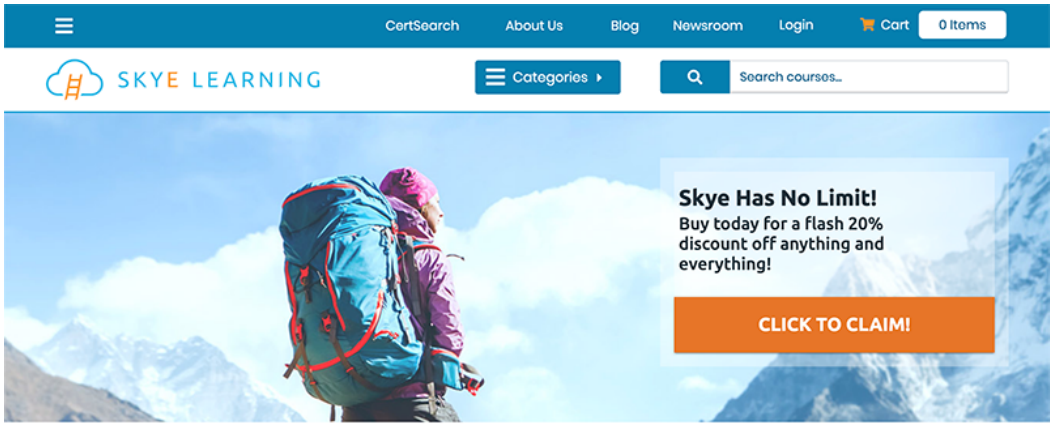
The <picture> Element: Understanding Images that Resize

Consider the popular style of many websites today that start their web page with a large image at the top of the screen, like the example shown below. Large images such as this look great on a desktop or larger screens. However, what happens to that large image file if it is served to a mobile device?



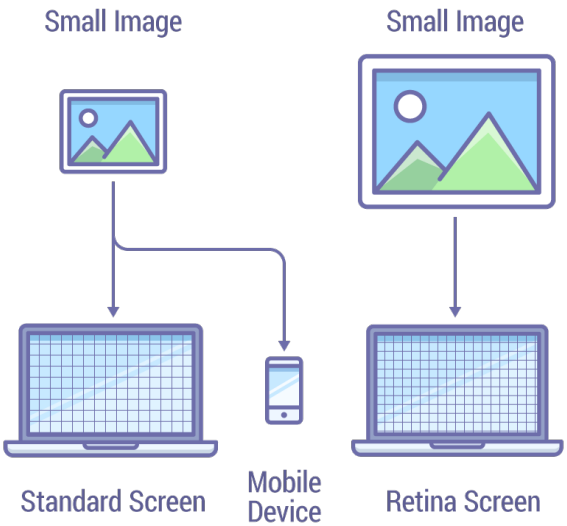
With no adjustments, the image resizes itself proportionately to the screen display. However, more than likely, there will be display problems. For example, consider that the file size for this image has not changed. It's the same giant image that worked well on a desktop computer, hooked up to a wifi signal.

However, what if the website is being viewed on a mobile phone with a 3G connection? That large image file that took no time to download to the desktop screen will take forever to download on the mobile phone. Furthermore, that image will eat up the owner's data plan very quickly.

Or, in some cases, that large image file is nearly impossible to see well on a mobile phone. All of the details are jammed into a tiny space. It begs the question if the image at this small size is even useful. What if we start with a small image on a mobile phone, and then display that on a desktop? Unfortunately, the image gets the **jaggies**, where the details get smudgy and lost as the image scales up from a small size.

The bottom line? None of these scenarios are acceptable from a user experience point of view. Fortunately, there is a solution to address the challenges that images pose when building responsive designs.

The <Picture> Element



The <picture> element affords designers the ability to define different image files to load at different viewport breakpoints. In

other words, rather than having one image that is scaled up or down based on the device's viewport width, several versions of the image that are designed to more appropriately fill the browser viewport are offered. Then, using media queries tied into the <picture> element, the device's browser will pick the image that best matches its viewport, and then *download only that image.*

The benefits of using the <picture> element syntax structure are:

- Only one image file is loaded, thereby improving the speed of page loading and saving bandwidth.
- Improved **art direction** in responsive designs as the image file that is loaded has the proper dimensions for the device upon which it is viewed.
- Ability to offer alternative image formats for situations when certain formats are not supported.

Art Direction Examples

The process of 'art directing' — adjusting image proportions, cropping the image, or replacing the image with a different version — gives a web designer the ability to specify which image should display based on the features of a particular display (desktop, tablet, or mobile). The following graphic illustrates the concept of art direction.

The result of using only one image that is scaled up or down based on the viewport width.

Use several different images to more appropriately fill the browser viewport.

Code Syntax: The <picture> Element

The <picture> element holds two different elements:

1. Zero or more <source> elements
2. One <img> element

Inside the <picture> declaration block are the versions of an image file for different display or device scenarios. Browsers will consider each <source> element, and choose the best match. If no matches are found (or if the browser does not support the <picture> element) the <img> element's src attribute is selected and displayed.

The following is one example of the syntax used with <picture> element. Note, that the media query associated with <picture> the tag isn't exclusively a "min-width value." Any type of valid media query may be used here. Min-width is commonly used due to the mobile-first approach, therefore, that is what we are showing here in this example.

```
<picture>
    <source media="(min-width:     )"
            srcset="image-file-1.jpg">
    <img src="image-file-2.jpg">
</picture>
```

Where:

- The srcset attribute (required) defines the URL of the image to display
- The media attribute — media="(min-width:)" — defines the media query parameters.
- The <img> element is required to be the last child in the picture declaration block. It is displayed if none of the media queries in the <source> elements matched, or if the browser does not support the <picture> element.

It is All About Options

The <picture> element provides options for displaying different images for situations such as when an image cannot load or is not appropriate for a given screen size.

Example: Working with <picture>

Let's start by considering the following example.

In the code tool below, the browser displays three images of the same subject — in this example, hot air balloons. The first image on the page is the full photo (1280px wide by 847px tall), followed by a medium-sized image (640px wide by 409px tall), and then a small image that is cropped to display only one of the balloons (200px wide by 200px tall). The cropped example illustrates a distinctly different type of picture that still communicates the same idea, yet will work well at mobile dimensions.

Instructions:

- Click 'run code' to see display in the browser window of the code tool
- Place your tooltip inside the code tool browser window, and then scroll up/down and left/right to see the various image sizes in the code tool

Run Code | Save | Export | Reset

index.html styles.css

```
<!doctype html>
<html>
 <head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="/styles.css" />
</head>
 <body>
   <p>Photo by <cite>Snapwire from Pexels</cite></p>

    <img img src="https://cdn-d.mindedgeonline.com/1678/balloons1.jpg" alt="Hot air balloons.">

    <img src="https://cdn-d.mindedgeonline.com/1678/balloons1-small.jpg" alt="hot air balloons.">

    <img src="https://cdn-d.mindedgeonline.com/1678/balloons1-cropped.jpg" alt="hot
```

Theme Select:

rubyblue

It is important to note that with the HTML syntax in the example above, *all three images will load on any display*. However, we want our page structure to be responsive to different screen sizes. Therefore, we only want to display the large image file on large displays, the medium image file on medium displays, and the small cropped image file on small displays. How do we structure the code to display the correct image on the appropriate device? Below are two possible solutions.

Please click on the following tabs to view each solution.

Solution #1: Use the display property in the CSS (Not Optimum)

One possible solution might be to use the display property in combination with media queries in the CSS. This code structure hides all but the appropriate image for a given display size.

In the HTML code, class attributes (small, medium, large) have been added to the image files. Then, within each media query in the CSS code, the display properties for each image size are turned "off" or "on" based on the screen size.

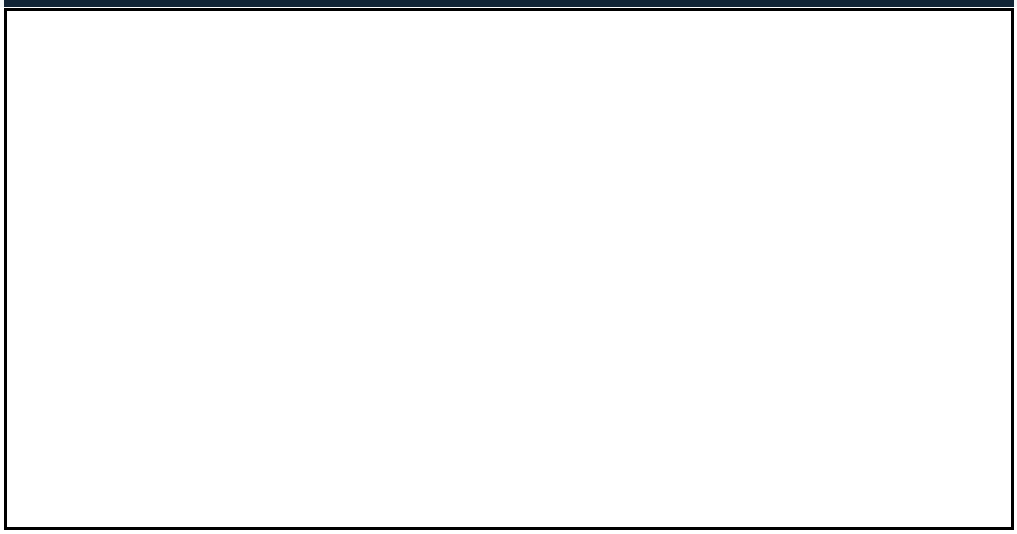
```
<!doctype html>
<html>
 <head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="/styles.css" />
 </head>
 <body>
  <p>Photo by <cite>Snapwire from Pexels</cite></p>

  <img img src="https://cdn-d.mindedgeonline.com/1678/balloons1.jpg" alt="Hot
air balloons." class="large">

  <img src="https://cdn-d.mindedgeonline.com/1678/balloons1-small.jpg"
alt="hot air balloons." class="medium">

  <img src="https://cdn-d.mindedgeonline.com/1678/balloons1-cropped.jpg"
alt="hot air balloons." class="small">
 </body>
</html>

 /* mobile first: do not display on small devices */
  .medium, .large {
    display: none;
  }

  @media (min-width: 350px) {
    .small {display: none;}
    .medium { display: inline; }
  }

  @media (min-width: 500px) {
    .medium {display: none;}
    .large { display: inline; }
  }
```

Instructions:

- Click 'run code' to see display in the browser window of the code tool
- Resize your browser window and watch how the image display changes at different screen widths

Run Code | Save | Export | Reset

index.html styles.css



Theme Select:

rubyblue

The problem with this code syntax is that while we are setting the display property to hide two of the images at any given breakpoint, **all of the images still download to the device**. This means extra time and data to load a webpage for images that never display — this is particularly consequential on mobile phones.

Solution #2: Use the <picture> Element (Optimum)

As previously discussed, the <picture> element allows the browser to pick the appropriate image based on its current state. Here, the browser has a choice of which image file to use, based on the size of the window.

CSS is not involved with this code structure; rather the picture element syntax and associated media queries are added directly to the HTML document. The image files are grouped within <picture> tags, with each image file wrapped in <source> tags that define the image URL with the srcset attribute, and the media query parameters using the media attribute. This code syntax is illustrated below — notice the mobile-first approach with the smaller screen sizes listed first.

```
  <picture>
  <source media="(min-width: 350px)"
          srcset="https://cdn-d.mindedgeonline.com/1678/balloons1-small.jpg"
alt="hot air balloons.">
```

```
  <source media="(min-width: 500px)"
          srcset="https://cdn-d.mindedgeonline.com/1678/balloons1.jpg"
alt="Hot air balloons." >

  <img src="https://cdn-d.mindedgeonline.com/1678/balloons1-cropped.jpg"
alt="hot air balloons.">

  </picture>
```

The code syntax above specifies that if the screen width is at least 500px wide, the browser will choose the first image. If the screen width is more than 350px, but less than 500px wide, the browser will choose the second image. The final <img> tag will load only if the screen width is less than 350px wide, or if the browser does not support the picture element. With this syntax, in addition to art direction, **only one resource will load, even though several are listed.**

Instructions:

- Click 'run code' to see display in the browser window of the code tool
- Resize your browser window and watch how the image display changes at different screen widths

Run Code Save Export Reset

index.html styles.css

Theme Select:

rubyblue

Images & Responsive Design

According to *Google Developers*, images account for more than 60% of the bytes on average needed to load a web page. Therefore, it is crucial that images respond to different viewport sizes and usage scenarios, so that your site will load quickly on any device.

The <picture> Element and Browser Compatibility

Currently, the <picture> element is not supported by Internet Explorer (IE).

Review Checkpoint

To test your understanding of the content presented in this assignment, please click on the Questions icon below. If you have trouble answering any of the questions presented here, you are always free to return to this or any assignment to re-read the material.

1. True or False?

The <picture> element contains two elements: one or more <source> elements and one <img> element.

a. True

Correct. This statement is true. The <picture> element contains two elements: one or more <source> elements and one <img> element.

b. False

Incorrect. Try again.

2. Which browser does not support the <picture> element?

a. Firefox

Incorrect. Try again.

b. Google Chrome

Incorrect. Try again.

c. Internet Explorer

Correct. Currently, the <picture> element is not supported by Internet Explorer.

d. Mosaic

Incorrect. Try again.