HTTP Headers and Responsive Images

This is an old one but it still worries me, and made me search for a possible solution in the context of web APIs without requiring a Node package manager or a Node infrastructure.

> I never want to make anything that even remotely resembles this. pic.twitter.com/QsfusOtxkI
>
> — Brad Frost (@brad_frost) May 16, 2015

No, no ones wants code like what Brad posted in 2015, but that's the reality when using responsive images, or is it?

If we don't mind doing the work on the server rather than the client we can do something like the code below, taken from Adapting to Users with Client Hints, to load WebP images for browsers that support them and JPG/PNG for browsers that don't.

```
<?php
// Check Accept for an "image/webp" substring.
$webp = stristr($_SERVER["HTTP_ACCEPT"], "image/webp") !== false ? true :
?>"whats-up.webp"?>

<img src="<?php echo($imageFile); ?>" alt="I'm an image!">
```

We can then shrink our responsive images by removing the formats that are not necessary using server-side code, in this case PHP.

```
<?php
// Check Accept for an "image/webp" substring.
$webp = stristr($_SERVER["HTTP_ACCEPT"], "image/webp") !== false ? true :
$format = $webp ? "webp" : "jpg";
"company-photo"?>
```

```
<picture>
  <source srcset="
    <?php echo($name); ?>-256w.<?php echo($format); ?> 256w,
    <?php echo($name); ?>-512w.<?php echo($format);?> 512w,
    <?php echo($name); ?>-768w.<?php echo($format); ?> 768w,
    <?php echo($name); ?>-1024w.<?php echo($format); ?> 1024w,
    <?php echo($name); ?>-1280w.<?php echo($format); ?> 1280w"
    type="image/<?php echo($format)">
      src="company-photo-256w.jpg"
      sizes="(min-width: 560px) 251px, 88.43vw"
      alt="The Sconnie Timber Staff!">
</picture>
```

The code first checks if the browser supports PHP by testing if the string `image/webp` is included in the Accept header. We record the result.

We then create two variables. One with the name of the file and the other one with the file extension that we use based on WebP support.

Then for every image in the `srcset` attribute we compose it using PHP echo statements and the variables that we created. In essence, if the browser supports WebP we use it, otherwise we use JPG images.

This is great but it requires a server-side script and it mixes business loogic with the HTML and it definitely gets cumbersome with more than a few images.

Looking at the client hints article in Web Fundamentals, I thought that they may be the solution to writing something like the PHP code in the previous example. Unfortunately, *Client Hints have no equivalent to the Accept HTTP header that we can parse for WebP support*.