igrigorik@gmail.com - last update: Feb 10, 2014.

Context: https://bugzilla.mozilla.org/show_bug.cgi?id=935216#c20

If you zoom on desktop, the device pixel ratio changes without all HTTP requests for the parts of the page getting replayed. If you change the window size on desktop or rotate the screen on mobile, the width of the weep or changes without all HTTP requests for the parts of the page getting replayed. So both cases are definitely different from how things would work with CH unless CH forces a reload upon rotate, window size change (hello Netscape 4!) or desktop zoom…. if these are meant to imply re-issuing the HTTP requests, then CH isn't actually a pure HTTP-layer feature and the impact on the higher layers needs to be specified and assessed (at which point we should really be putting the effort into client-side respimg and CSS OM opt-out instead)

Right, I think I'm with you. Let's try to work through some hand-on examples below.

Let's start with the "simple" DPR-switching case...

 # 1 ----------

```
<style>
  #bg-image { background: (pic1x.jpg) }
  @media only screen and (min-device-pixel-ratio: 2.0) {
      #my-image { background: (pic2x.jpg) }
  }
</style>

<div id=”bg-image”></div>
```

Above example is what we have been using for some time now:

- we have to manually define the criteria for when the asset should be switched
- we have to manually size the containing container
- different assets have to have a different URLs

 # 2 ----------

```
<picture>
    <source srcset="pic1x.jpg 1x, pic2x.jpg 2x">
    <img alt="A rad wolf." src="pic1x.jpg">
</picture>
```

This provides all the same behaviors as #1, plus some additional benefits:

- UA figures out the width/height of the container if they are not specified
- UA automatically performs all the intrinsic size calculations based on specified mix of X

queries and/or sizes attributes

That said, the user still has to manually specify distinct URLs for each and every variant of the resource - e.g DPR variants and resolution variants.

 # 3 ----------

```
<img alt="A rad wolf." src="pic.jpg">
```

This last example is the interesting case. Could we turn the humble <img> tag into an auto-multi-DPR friendly tag? A couple of different cases to consider:

I. We're on a device where DPR is fixed - aka, zoom does not rescale DPR. This is the simple case, since we could simply send the CH-DPR header to the server and let it pick the right asset. Once it's downloaded, our job is done.
II. We're on a ("desktop") device where zoom affects the DPR. As a result, when the DPR value is updated, should the UA dispatch a new request? Several plausible behaviors:
    A. No request is made and original image is used until page is refreshed.
    B. Request is sent with new CH-DPR header.

Arguably, (A) is not unreasonable: we're retrofitting <img> to be auto-DPR friendly, and the site is not providing any hints for when new request should be initiated, hence we just reuse and rescale the same asset when DPR is affect.

- If the site owner wants the image to be updated with a different asset, they should upgrade to <picture>.
- "No update" behavior is consistent with current implementations of srcset in Blink and Webkit: the UA does not initiate new requests when DPR is updated on zoom -- whether this is the right behavior is a separate question, but that's the way it behaves today and the spec does not define any explicit behaviors for this case.

Conversely, (B) is not unreasonable either, especially if srcset is updated to trigger requests when zoom change → DPR update hits a defined breakpoint. However, if this behavior is put in place, it may result in a lot of requests if only "Vary: CH-DPR" is used: every zoom update technically invalidates the previous asset and new request can/should be made. To address this, either the UA needs to provide some coarse-grained triggers, or we rely on Key header to resolve this - e.g. *"Key: CH-DPR;r=[1.0:3.0]"* indicates that returned image asset should be reused for DPR values between 1.0 and 3.0.

Handling resource width updates

For resource width updates (communicated via CH-RW), the logic is very similar…

Working backwards, let's start with #3 (<img>)... We may or may not have the resource width when we parse the <img> tag:

- if width attribute or an inline CSS style defining the width is specified, then we could use that and append the CH-RW header
- if width cannot be determined and we we're not willing to wait for layout, we omit CH-RW.

The benefit of appending CH-RW is that it could help with delivering "pixel perfect" images. On the other hand, there is a good argument to be made here that it may not be worth it: we may not have the resource width in enough cases; we have to consider what to do when resource is resized, etc. - all the same Vary vs. Key considerations here.

Cases #1 and #2 are basically the same, so let's use #2 as an example…

```
<picture>
  <source sizes="(max-width: 30em) 100%, (max-width: 50em) 50%, calc(33%-100px)"
          srcset="pic.jpg">
  <img src="pic400.jpg" alt="The president giving an award.">
</picture>
```

In the case above we can immediately calculate the resource width and append CH-RW to the outgoing image request. Then, let's say the viewport is resized and the resource width has changed: we need to decide if we're going to dispatch a new request.

Naively, the answer is yes, but that may result in a lot of requests and overhead: do we initiate requests for every pixel change, or is there a larger / coarser update? Once again, this is likely best addressed via Key - e.g. server can return "Key: CH-RW;r=[320:640]" to indicate that the same asset should be used for widths between 320 and 640 pixels.

How does the server determine these resource width and DPR buckets? It can be configured to use widths based on available pre-generated assets, or it can be configured to generate a set or dynamic number of variants to optimize cache hit rates both on the client and server, <insert own smart strategy here>.

TL;DR:

- DPR and resource-width updates may need to trigger new asset requests with CH
- Without a predefined set of breakpoints, this may require new request on every update
    - Implementing Key resolves this by allowing the server to specify the parameter range for which the returned asset should be considered valid.

- The parameter range can be based on available asset variants, custom configuration, or dynamic optimization to optimize cache hit rates or other criteria.
- Alternatively, perhaps there is also an in-between strategy here to allow the UA to make some reasonable decisions as to when request should be dispatched, instead of forcing a new request on each and every update.