Image Replacement in Blink (public)

sclittle@chromium.org
Last Modified 2016-10-14
https://crbug.com/605350

Overview

In order to save data for the user, Chrome should support removing or postponing image loads, while preserving page layout. To achieve this, Chrome should support issuing range requests for just the first few bytes of images, and then attempt to extract the image dimensions from these chunks and display placeholders with the same dimensions in their place.

Potential use cases of this functionality include:

- Replace images with placeholders on heavy pages or extremely slow networks for Data Saver users, similar to Lo-Fi.
- Defer loading of below-the-fold images.

This doc proposes an implementation of the Blink support for this feature, including fetching and building these placeholders.

Objective

When Chrome determines that it should attempt to display a placeholder image, it should attempt to get the image in the following ways in order of precedence:

- If the full image is present and fresh in the cache, then use that.
- Otherwise, if the server supports range requests, and the image dimensions can be decoded from the first 2KB of the image, then generate and show an image placeholder with the same dimensions.
- Otherwise, fetch the entire full image from the server as usual.

The first 2KB is used for the range because (a) it's likely to capture the entire image for smaller images and icons and (b) the first 2KB is highly likely to contain the image dimensions for larger images, according to UMA data (Blink.DecodedImage.EffectiveDimensionsLocation.* histograms).

It also needs to be possible for placeholder images to be reloaded and replaced with the complete original image later on, such as if the user decides to load the original image through the UI.

Detailed Design

Fetching Placeholders

Add a new field to FetchRequest:

```
enum class PlaceholderImageRequestType {
    DisallowPlaceholder = 0,
    AllowPlaceholder,
};
PlaceholderImageRequestType FetchRequest::placeholderImageRequestType();
```

By default, this field would be set to DisallowPlaceholder. The caller of ImageResource::fetch(), e.g. the ImageLoader or CSSImageValue, can decide if it wants a placeholder image or not, and set the field accordingly.

If DisallowPlaceholder is passed in, the image fetch will behave like it does currently and won't attempt to fetch a placeholder, with no noticeable additional performance or memory penalty. Also, if the image was already loaded as a placeholder or is currently being loaded as a placeholder, calling ImageResource::fetch() with the new field set to DisallowPlaceholder will cancel the in-progress placeholder fetch (if applicable) and issue a regular fetch for the full image instead.

If AllowPlaceholder is passed in, and there isn't already a fetch in progress for this image, then modify the request in ResourceFetcher::requestResource() to add a range request header for the first 2KB of the image, and also set a new request bit ResourceRequest::ignoreRangeHeaderIfFullyCached (see below section), so that the entire image can be used from cache if it's fully present and fresh.

If the received response consists of the entire resource according to a new response bit ResourceResponse::isEntireResource (see below section), and the response is either decoded without error or has a response code other than a 4XX or 5XX error, then use the returned response to show the full image normally. Otherwise, attempt to decode the returned range to extract the image dimensions.

If the image dimensions can't be decoded from the subrange, then issue a new request for the full image (by calling ImageResource::reloadIfPlaceholder, see section below) from within ResourceFetcher::didFinishLoading/didFailLoading, bypassing the cache just in case the server returned a bad range earlier. Otherwise, generate a placeholder image with the same dimensions as the original image, and show that in the image's place. See the Placeholder Images section for more details.

New Request and Response Bits

```
// Indicates to the embedder that if the full resource response would have been
// returned for the same request minus the range header without sending any
// requests over the network, then the embedder should respond with that full
// resource response.
ResourceRequest::ignoreRangeHeaderIfFullyCached();

// Indicates if this response represents the entire resource. This will be true if
// the response code is anything other than a 206, or if the 206 Partial Content
// response has a Content-Range header indicating that the returned range contains
// all of the bytes of the resource, e.g. "Content-Range: bytes 0-2047/2048".
ResourceResponse::isEntireResource();
```

Placeholder Images

Placeholder images will be generated images consisting of a translucent gray box. A new Image subclass will be added, PlaceholderImage, that has custom drawing methods for this.

Reloading Placeholders

A method already exists to reload Lo-Fi images, ImageResource::reloadIfLoFi. This method will be renamed to:

```
ImageResource::reloadIfPlaceholder(ResourceFetcher*);
```

This method will be changed to reload images if they are either Lo-Fi placeholders or placeholders generated from this range request mechanism. In both cases, it'll be a full reload bypassing the cache, just like how Lo-Fi placeholders are reloaded currently.

Potential Improvements

If the original image is a progressive JPEG, and the first 2KB range of the image contains the full low-resolution version of the image, then it might be worth considering using that low-resolution version of the image as the placeholder.