Images
Best Practices Guide

Version: 2013.11.20

This section includes best practices to consider during development and build phases that are related to optimizations possible when using image resources. Details on each checklist item are discussed later in the document.

Checklist

- ❏ Use less images �
- ❏ Provide proper images sizes for target devices �
- ❏ Reduce image sizes �
- ❏ Scan for and remove unused images �
- ❏ Use CSS image sprites �
- ❏ Use CSS3 instead of images where possible �
- ❏ Use CSS media queries to only load the proper size image for device �
- ❏ Use font icons instead of small image files �
- ❏ Consider using JPEG instead of PNG �
- ❏ Consider using Progressive JPEGs �
- ❏ Consider inlining SVG graphics in HTML as a build step �
- ❏ Consider converting animated SVG to Canvas �

❑ Use Google NinePatch image type for splash screens �

Discussion

Use less Images

Images typically are the most costly items to render on a page. It is ok to push back some when the design relies too heavily on images for the overall look of the page. Images are good, as they convey a general feeling and tone for a page. Too many images slow the page down and often feel cluttered.

Provide proper images sizes for target devices

Having the browser scale images is expensive, especially for mobile devices. Performing a build time resizing of images at specific sizes will result in much better performance at runtime. See also the media queries discussion below.

References:
➠ Why responsive Images is so hard

Reduce image sizes

Reduce image quality where practical. 75% quality is not usually noticeable, but can make a large file size reduction.

References:
➠ Image Optimization Tools, by Addi Osmani
➠ HTML5 Image Compression Article

Scan for and remove unused images

Image directories are often like a junk drawer. Without periodic review, unused images commonly start to accumulate and rarely get cleaned up. For each image file, scan for its use within the code base, and if its not being referenced, remove it from the project.

For hybrid mobile projects, this is very important. Unused images can add considerable bloat to deployed application sizes which can increase application install, update and first-time startup initialization times in some environments.

Use CSS Sprites

Use image sprites and CSS image offsets rather than linking to many smaller image files for things like button states.

Typically, background-image and background-position are used to create css sprites, and thus the images are background images. In some cases, you may want to create foreground image sprites, for example, when you want to keep the images shown even in the high contrast mode. In such cases, you can use the CSS clip property for the <img> element.

Other tips:

- Use lossless compression for CSS sprite images.
- Clever arrangement of image tiles when developing sprite images can result in additional image file size reduction during lossless compression
- Consider the option to inline sprite images into a data-uri's as a build step to further minimize the number of resource requests.

Typically, background-image and background-position are used to create css sprites, and thus the images are background images. In some cases, you may want to create foreground image sprites, for example, when you want to keep the images shown even in the high contrast mode. In such cases, you can use the CSS clip property for the <img> element.

Dealing with sprites can be a major pain as if you need to update an image, you must then rebuild the sprite, and likely update all the image offsets in the CSS rules. Using Compass for Sass can automatically create sprites for you. This can be a huge productivity booster, but comes at the costs of learning how to use Sass/Compass.

References:

➠ A List Apart, Issue# 173, CSS Sprites

➠ Sprites in Compass Video - The whole LevelUpTuts video series on Compass is worth watching

➠ Foreground <img> Sprites – High Contrast Mode Optimization

Use CSS3 instead of images where possible

The old way of providing visual effects for buttons and styling of regions such as rounded corners and gradients used to be done with images. Use CSS3's capabilities for this type of styling. This also reduces resource loading dependencies. Things like gradients, shadows, and rounded corners are faster through CSS than using discrete images.

References:

➠ Smashing Magazine: CSS3 Instead of Images

Use CSS media queries to only load the proper size image for device

Media queries are a CSS technique that permit for conditional loading of CSS files, as well as conditional setting of discrete CSS rules. These are a cornerstone of good responsive design, and can drastically reduce the number of rules to support multiple browser sizes and orientations. Images can be specified based on device widths and/or orientation.

Loading specific images through media queries can be done at the file level like this:

```
<link rel='stylesheet' media='screen and (min-width: 800px)'
href='tablet_images.css' />
```

Using media queries inline in a common CSS file, can be done like this:

```
.logo {
    background-repeat: no-repeat;
}
@media screen and (max-width: 799px) {
    .logo { background-image:url('logo_small.png'); }
}
@media screen and (min-width: 800px and max-width: 1199px) {
    .logo { background-image:url('logo_medium.png'); }
}
@media screen and (min-width: 1200px) {
    .logo { background-image:url('logo_large.png'); }
}
```

References:
➠ Best Practice document on Responsive Design
➠ W3C Media Queries
➠ CSS Media Queries

Review image and SVG data-uri's use in HTML and CSS when targeting mobile devices

Data URI's have known performance issues on mobile browsers. The referenced article includes additional guidelines when using data-uri's.

References:
➠ CSS Sprites vs. Data URI's: Which is faster on mobile?

Use font icons instead of small image files

Font icons can be used effectively for things like bullets, navigation arrows, action indicators, and more. If your design includes many of these small image styles, font icons may be appropriate. These are defined using the `@font-face` rule in CSS.

References:
➠ Font Awesome has a great collection of icons defined this way.

Use proper image types

Different image types should be used for both performance and size. (ie. JPEG for photos, PNG for graphics/screen shots, WebP for anything).

JPEG images render marginally faster than PNG, especially in older browsers. This is probably only a worthwhile exercise if you are using a lot of discreet images. Also, PNG is still recommended if you need transparency.

Consider using Progressive JPEGs

Progressive JPEGs can typically improve render times of JPEG images by up to 7% over normal “Baseline” JPEG images. Progressive JPEGS render by displaying a low resolution version of the image and then progressively enhancing it as more data is returned. Baseline JPEGs render the full resolution image from top to bottom. The progressive fill in may (or may not) enhance the user experience. The overall image size does not change from these two types, but typically progressive rendering completes faster. One caveat is that you should not use progressive for small JPEG images (eg <16K).

Many image manipulation tools provide save options to produce progressive JPEGS. Other command line tools that can convert files in batch include:

- jpegtran - Available for most operating systems.
- Adept - the adaptive JPG Compressor
- Kraken.io - Image optimizer

References:
➠ Performance Matters - Progressive JPEGs FTW!
➠ YUI - Image Optimization, Part 4: Progressive JPEG…Hot or Not?

Consider inlining SVG graphics in HTML as a build step

Benefit: Reduces the number of resource requests by inlining SVG linked file content directly into HTML.

Drawbacks: Inline SVG is more costly to maintain and update if not done as an automated build step.

Caution: If the SVG is not used throughout the application, or on frequently visited views, this technique should be avoided.

Consider converting animated SVG to Canvas

Animated SVG can be slow when you have a lot of data points. Canvas usually faster. It is easy to change from SVG to canvas in D3, but you have to handle mouse events yourself.

Use Google's NinePatch image type for splash screens

Google's 9-Patch image format permits for clean scaling of images. This is very useful when used as splash screens for hybrid applications. The resulting image is a special PNG image that scales in nine different cells, without core image distortion.

References:

- Google's Draw 9-Patch tool