Image formats for the web: HEIC and AVIF

There are two formats that I left out of the previous posts about image formats for the web: HEIF and AVIF

These two formats are based on video codecs and, depend on the parent video technology to compress images. This makes them different than existing formats like PNG, JPEG and GIF.

In the table below, I've sumarized basic information about the two formats. If using open formats it's important to your project you may want to take a closer look at these technologies; HEVC is known to have pattent encumbrances and AV1, even though AOM claims it's patent free, has a new patent pool actively issuing licenses and this may render the freely available claim moot.

How will this affect AV1 and AVIF adoption is unknown at this time.

Format	Initial Release	Type	Encoder to Use	Notes
HEIC	2013	Lossless and Lossy	Image Magick or libheif	H265 compression in HEIF container Fully supported in Apple Products (macOS and iOS)
AVIF	2019	Lossless and Lossy	libavif	AV1 compression in HEIF container Google has published an intent to ship a native AVIF decoder in Chrome. See Chrome Status entry Firefox has a basic implementation behind a flag. See Bugzilla Bug 1625363

Getting started

Before we can start comparing images and sizes we need to do the following:

1. Resize the image we want to use
2. Compile libavif on my Mac. This is not always a trivial or self evident process

and it's not documented in

Resizing Images

I chose to resize the image before testing to make sure that the image size will not affect the results of the tests.

To resize we'll use ImageMagick. The command to scale the image to 2000px wide and corresponding height, would look something like this:

```
convert emporium_hi_res.png -resize 2000 \
emporium-resized.png
```

This command will not distort the image. It will scale the height proportionately to meet the width requirements.

Compiling libheif

I've had multiple issues trying to get AVIF images to encode using libheif on my system. To make sure it wasn't a Homebrew issue (it appears to be) I commpiled libheif from scratch following the instructions on the README

First we install the pre-requisites. Most of the time these will already be installed.

```
brew install automake \
make \
x265 \
libde265 \
libjpeg
```

We then run commands to generate the configuration files and build the binaries.

```
./autogen.sh
./configure
make
```

Because the Homebrew installed version is a dependency of ImageMagick and VIPS, Homebrew won't let me remove it. So I've chosen to skip the `make install` step and run them from where they were compiled.

Compiling libavif

In a Mac, Homebrew provides the dependencies we need to run libavif, but not libavif itself. So, to install libavif, we'll install the dependencies first via Homebrew and then compile Libavif from scratch.

The dependecies we need are:

- **aom** provides an encoder and a decoder
- **dav1d** provides a decoder
- **rav1e** provides an encoder

To install the dependecies, run the following command

```
brew install aom dav1d rav1e
```

Later we'll use the two decoders to test if the code we use makes a difference in the size of the final image.

To install libavif, download the latest tagged release from Github.

Once you have downloaded and unzipped the code run the following commands:

First we make a directory to store the build artifacts and binaries before installation.

```
// Make the build directory and CD to it
mkdir build && cd build
```

Libaviif uses CMake to configure the application and provide additional flags for the compiler to use.

From the `build` directory, run the following command:

```
// Use CMake to configure AVIF installation
cmake ..  -DAVIF_CODEC_AOM=ON \
-DAVIF_CODEC_DAV1D=ON \
-DAVIF_CODEC_RAV1E=ON \
-DAVIF_BUILD_APPS=ON
```

The two dots, `..` indicate that the source code is the directory above build.

The `-DAVIF_CODEC_` entries tell libavif the different codecs that we want to use. As discussed earlier we want to enable, aom, dav1d and rav1e.

The final flag, `-DAVIF_BUILD_APPS` tells libavif to build the sample encoder and decoder applications. If you don't add this flag it will only build the libaavif library.

The final process is to compile and install the files.

`make` will compile the needed progras and `make install` will move the compiled programs to default locations in the file system.

```
// Make and install the binaries
make
// install the files we just compiled
make install
```

Because I'm using Homebrew on a Macintosh, installing the binaries and associated files may cause warnings when running `brew doctor`. It is safe to ignore the warnings.

Figuring out flags and settings

Libheif presents some interesting challenges on figuring out what settings to use. It also takes parameters from the underlying x265 installation for encoding the images.

We'll start with the naive encoding and let it run with only libheif and libavif defaults. This is the simplest way to run the encoders and is what I would first use when compressing images, not worrying about lossless or lossy or specifying

image quality.

```
# aviflib encoder
avifenc emporium-resized.png \
emporium-resized.avif

# libheif encode
heif-enc emporium-resized.png \
-o emporium-resized.heif
```

Format	File Size
AVIF	948KB
HEIF	478KB

Even the default values are decent. But we can do better.

Looking at libheif settings

In addition to the flags that come with the encoder, `heif-enc` can also take parameters from the underlying HEVC encoder (in this caase x265).

To get the available encoder parameters run the following command:

```
heif-enc --params
```

For a default install using x265 it will produce the following output.

```
Parameters for encoder `x265 HEVC encoder (3.4)`:
  quality, default=50, [0;100]
  lossless, default=false
  preset, default=slow, {
    ultrafast,
    superfast,
    veryfast,
    faster,
    fast,
    medium,
```

```
    slow,
    slower,
    veryslow,
    placebo }
  tune, default=ssim, {
    psnr,
    ssim,
    grain,
    fastdecode }
  tu-intra-depth, default=2, [1;4]
  complexity, [0;100]
```

The interesting part, to me, is the use of HEVC presets in the image encoding. According to the x265 documentation:

> x265 has ten predefined --preset options that optimize the trade-off between encoding speed (encoded frames per second) and compression efficiency (quality per bit in the bitstream). The default preset is medium... When you use faster presets, the encoder takes shortcuts to improve performance at the expense of quality and compression efficiency. When you use slower presets, x265 tests more encoding options, using more computations to achieve the best quality at your selected bit rate...

We could modify individual parameters beyond what presets provide but unless there are very specific reasons I don't think it's a wise investment of time and resources, unless you're an HEVC expert.

My assumption when encoding HEIF images is that the slower and faster presets will create a smaller image for different reasons, the faster presets usually take shortcuts and don't do a thorough job in compressing data; the slower presets will take longer in compressing the image and do a better job than the default.

I did the test with two images, the first one, a black and white image of the USS California.

The black and white image produced the opposite results to what I expected. The faster presets were smaller than the slower ones but the difference wasn't as

big as I expected it to be.

The slow preset, the default for libheif compression was only 26KB larger than the superfast present which created the smaller image for this example.

Preset	Size	Notes
Ultrafast	181KB	
Superfast	177KB	
Veryfast	228KB	
Faster	228KB	
Fast	228KB	
Medium	228KB	Default for HEVC
Slow	203KB	Default for Libheif
Slower	203KB	
Veryslow	203KB	
Placebo	203KB	

The second test was with a resized version of emporium-hires. The slower presets still produce smaller sizes than faster presets. What caught my attention is how much larger the results from faster presets (ultrafast and superfast) are. I guess it’s because the shortcuts the faster presets use are more important in a color image.

Preset	Size	Notes
Ultrafast	644KB	
Superfast	628KB	
Veryfast	507KB	
Faster	507KB	
Fast	507KB	
Medium	507KB	Default for HEVC
Slow	478KB	Default for Libheif

Preset	Size	Notes
Slower	478KB	
Veryslow	478KB	
Placebo	479KB	

The final test uses a Nasa Image with a black background and bright, processed, colors. The scale is similar to the two previous examples but it's heavier in file sizes.

Preset	Size	Notes
Ultrafast	591KB	
Superfast	562KB	
Veryfast	642KB	
Faster	642KB	
Fast	642KB	
Medium	642KB	Default for HEVC
Slow	551KB	Default for Libheif
Slower	551KB	
Veryslow	551KB	
Placebo	552kB	

So, for the most part, it's OK to use the libheif default preset. We'll do additional testing when comparing quality for both AVIF and LIBHEIF.

Looking at AVIF encoders

AVIF presents different challenges. We begin by considering the three different encoders:

- Libheif (with a special flag)
- AOM for libavif (encoder and decoder)
- Rav1e for libavif (encoder only)

I wouldn't expect major different using the three codecs but it's always good to

test and be sure. Using libheif for all our tests and future encodings reduce the number of applications we have to use but, if we can get better performance in terms of encoding time and quality then the additional software may be warranted.

All these tests use default settings for the encoders. We can probably dive deeper into specific encoder settings to improve file size but that's not the goal of these particular tests.

The results using `emporium-resized` proved my assumptions wrong again. Encoding using libheif produced the smallest results and using rav1e produced the largest.

format	encoder	size
AVIF	heif-enc --avif	702KB
AVIF	avifenc -c aom	948KB
AVIF	avifenc -c rav1e	1MB

Using the `USS_California` image as a grayscale example the results are similar in terms of encoder performance but the results are smaller across the board.

I suspect the numbers are smaller because there are fewer colors that can be compressed more than the other two examples.

format	encoder	size
AVIF	heif-enc --avif	207KB
AVIF	avifenc -c aom	262KB
AVIF	avifenc -c rav1e	341KB

The final example is a NASA image `STScI-H-p2031b-f` and it produces the largest sizes accross the board with larger file sizes overall.

format	encoder	size
AVIF	heif-enc --avif	621KB
AVIF	avifenc -c aom	978KB

format	encoder	size
AVIF	avifenc -c rav1e	1.2MB

Considering the results of the three tests, it's safe to assume that libheif is the best alternative for creating HEIF images.

Evaluating different quality values

As with WebP, PNG and JPG, AVIF and HEIF have a quality setting that will allow finner level of control over the quality.

AVIF doesn't have a straight quality value like HEIF but it has the following elements that, as far as I know, produce the same effect as the quality setting in libheif

```
--min Q: Set min quantizer for color
(0-63, where 0 is lossless)
--max Q: Set max quantizer for color
(0-63, where 0 is lossless)
--minalpha Q: Set min quantizer for alpha
(0-63, where 0 is lossless)
--maxalpha Q: Set max quantizer for alpha
(0-63, where 0 is lossless)
```

Because the values are not the same as those from libheif I would be guessing as to what would be equivalent.

I've created a different scale for compressing images with `avifenc`. Rather than going from 50 to 100 I decided to go from 22 to 62 in 10 step increments.

I chose to use the same value for both color and alpha rather than leave alpha quality as lossless.

I also chose the rav1e encoder to make sure if there is a difference in this scenario. It appears that when encoding with non-default quality values for both color and alpha Rav1e produces smaller results than the AOM encoder.

format	encoder	(color) quality	α quality	size
avif	avifenc	62	62	28KB
avif	avifenc	52	52	73KB
avif	avifenc	42	42	192KB
avif	avifenc	32	32	402KB
avif	avifenc	22	22	649KB

When enconding with `libheif` both formats gain the ability to define quality in a scaller fron 20 to 100 and I took this scale and used values from 50 to 100.

Keeping in mind that the quality values for HEIF are not the same as the values we used with `avifenc` the table below presents the AVIF results created with `heif-enc`.

format	encoder	(color) quality	α quality	size
avif	libheif	50	N/A	478KB
avif	libheif	60	N/A	771KB
avif	libheif	70	N/A	1.2MB
avif	libheif	80	N/A	1.6MB
avif	libheif	90	N/A	2.0MB
avif	libheif	100	N/A	2.1MB

The final result is for HEIF images compressed with `libheif`. The results follow a progression in file sizes but not a linear one. The one surprising item is the size of the HEIF images, they are significantly larger than thei AVIF equivalents.

format	encoder	(color) quality	α quality	size
heif	libheif	50	N/A	478KB
heif	libheif	60	N/A	771KB
heif	libheif	70	N/A	1.2MB
heif	libheif	80	N/A	1.8MB

format	encoder	(color) quality	α quality	size
heif	libheif	90	N/A	2.0MB
heif	libheif	100	N/A	2.1MB

SSIM testing

Testing for structural similarity is harder with newer formats. As far as I know there is no current SSIM-like testing for AVIF images. I will write a separate post when the functionality becomes available so we can do a complete and fair comparison.