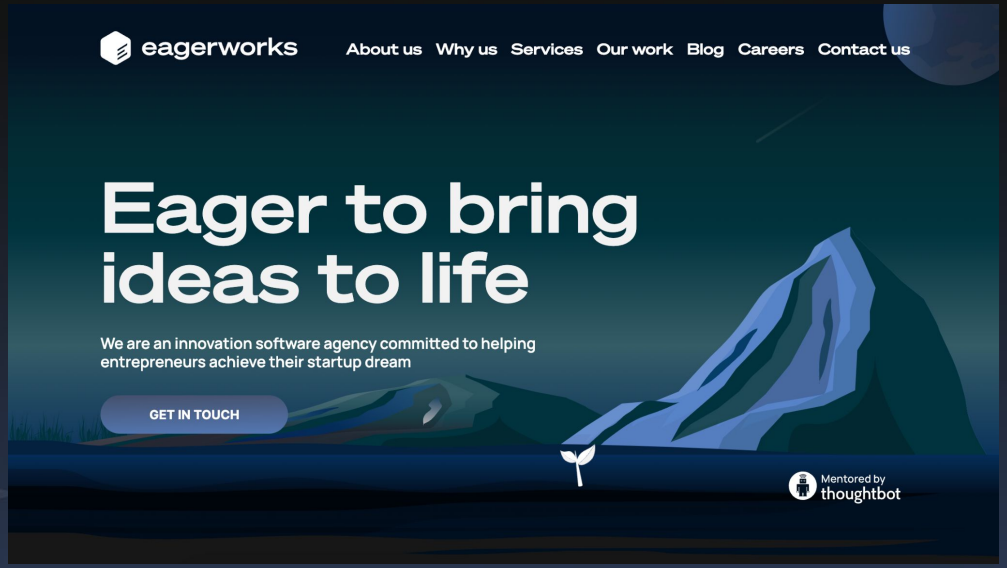
eagerworks

Rails Conf 2023

Faster websites: integrating next-gen images in your Rails apps

JP Balarini
CTO @ eagerworks
@jpbalarini

Background


eagerworks
About us
Why us
Services
Our work
Blog
Careers
Contact us
Eager to bring ideas to life
We are an innovation software agency committed to helping entrepreneurs achieve their startup dream
GET IN TOUCH
Mentored by thoughtbot

Web Images

Current Options

JPEG (1992) → .jpeg/.jpg

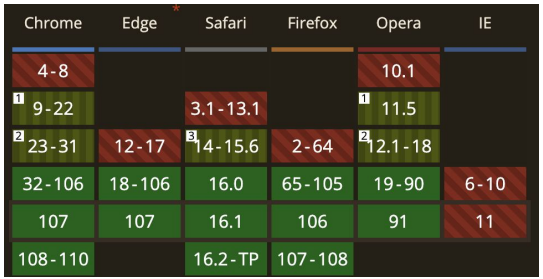
PNG (1996) → .png

JPEG 2000 (2000) → .jp2

Current standards have ~30 years!

Next-gen Options

WebP (2010) → .webp



AVIF (2019) → .avif

JPEG XL (2020) → .jxl

source: caniuse.com

›30% size reduction

Motivation behind next-gen images

Loading time → User conversion

The highest ecommerce conversion rates occur on **pages with load times between 0-2 seconds.** (Portent, 2019)

Website conversion rates drop by an average of 4.42% with **each additional second of load time** (between seconds 0-5). (Portent, 2019)

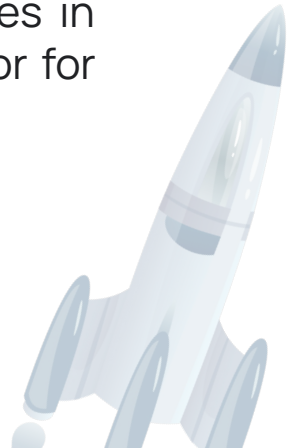
Telefónica improved load times for its mobile site **by 70%** — from six seconds to only two seconds in 3G connections. These improvements helped the company **increase click-through rate by 31%.**

BMW mobile site revamp: people clicking from BMW.com to a BMW sales site went **from 8% to 30%** (~ 4X higher)

SEO I Search Engine Optimization

From Google's Blog:

"Users want to find answers to their questions quickly and data shows that people really care about how quickly their pages load. The Search team announced **speed would be a ranking signal** for **desktop** searches in 2010 and as of this month (July 2018), page speed will be a ranking factor for **mobile** searches too."



Motivation behind image size: Instagram case study

Uploads: 50k uploaded images per minute (assume 2MB / image)
Storage: 0.002GB * 50000 = **100 GB / min** → **144,000 GB/day** ~ 144 TB/day

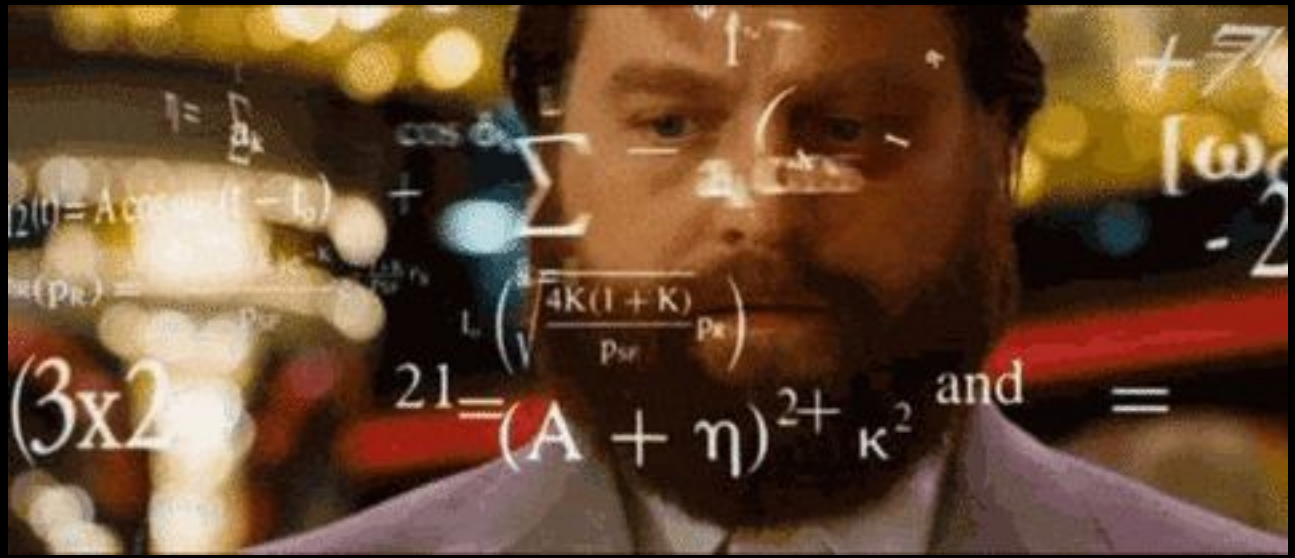
Usage: Assume 50 images/minute/user

29 minutes average/user * 500,000,000 DAU = 14,500,000,000 minutes of usage per day

14,500,000,000 minutes/day * 50 images/minute = 725,000,000.000 images served per day → **~500,000,000 images / min**

Transfer (CDN →user): 500,000,000 images/min * 0.002 GB = **1,000,000 GB/min**

~ 1,440,000 TB/day


(3x2
21=(A + η)^2+ κ^2 and =

Real examples

```
<div class="_aagv" style="padding-bottom: 100%;"> == $0
  <img alt="Photo by Dream BMW E46 in Montevideo, Uruguay." crossorigin="anonymous" class="x5yr21d xu96u0
  xh8yej3" src="https://instagram.fmvd4-1.fna.fbcdn.net/v/t51.2885-15/290654198_56764… AfB144bK5sCSOLyj1H
  &oe=64442D61&_nc_sid=1527a3" style="object-fit: cover;">
```

Rendered size: 97 × 97 px
Rendered aspect ratio: 1:1
Intrinsic size: 1440 × 1440 px
Intrinsic aspect ratio: 1:1
File size: 426 kB
Current source: https://instagram.fmvd4-1.fna.fbcdn.net/v/t51.2885…PmKKST2FSiPlCw2HnXfGrg&oe=64442D61&_nc_sid=1527a3

https://instagram.fmvd4-1.fna.fbcdn.net/v/t51.2885-15/290654198_567643358266590_6221797881216498310_n.jpg?stp=dst-jpg_e35&_nc_ht=instagram.fmvd4-1.fna.fbcdn.net&_nc_cat=106&_nc_ohc=fOikWvHocboAX_dlKQt&edm=ACWDqb8BAAAA&ccb=7-5&ig_cache_key=Mjg3MDkyMTA1OTM0ODMwMjE4NA%3D%3D.2-ccb7-5&oh=00_AfB144bK5sCSOLyj1H_zCUtWPmKKST2FSiPlCw2HnXfGrg&oe=64442D61&_nc_sid=1527a3

2 minute read · November 3, 2022 8:43 PM GMT-3 · Last Updated 4 days ago

Musk orders Twitter to cut infrastructure costs by $1 billion - sources

The company is aiming to find between $1.5 million and $3 million a day in savings from servers and cloud services, said the Slack message, which referred to the project as "Deep Cuts Plan."

Rendered size: 251 × 141 px
Rendered aspect ratio: 251:141
Intrinsic size: 680 × 453 px
Intrinsic aspect ratio: 680:453
File size: 64.6 kB
Current source: https://pbs.twimg.com/media/FiDw7T5XEAA5pEu?format=jpg&name=small

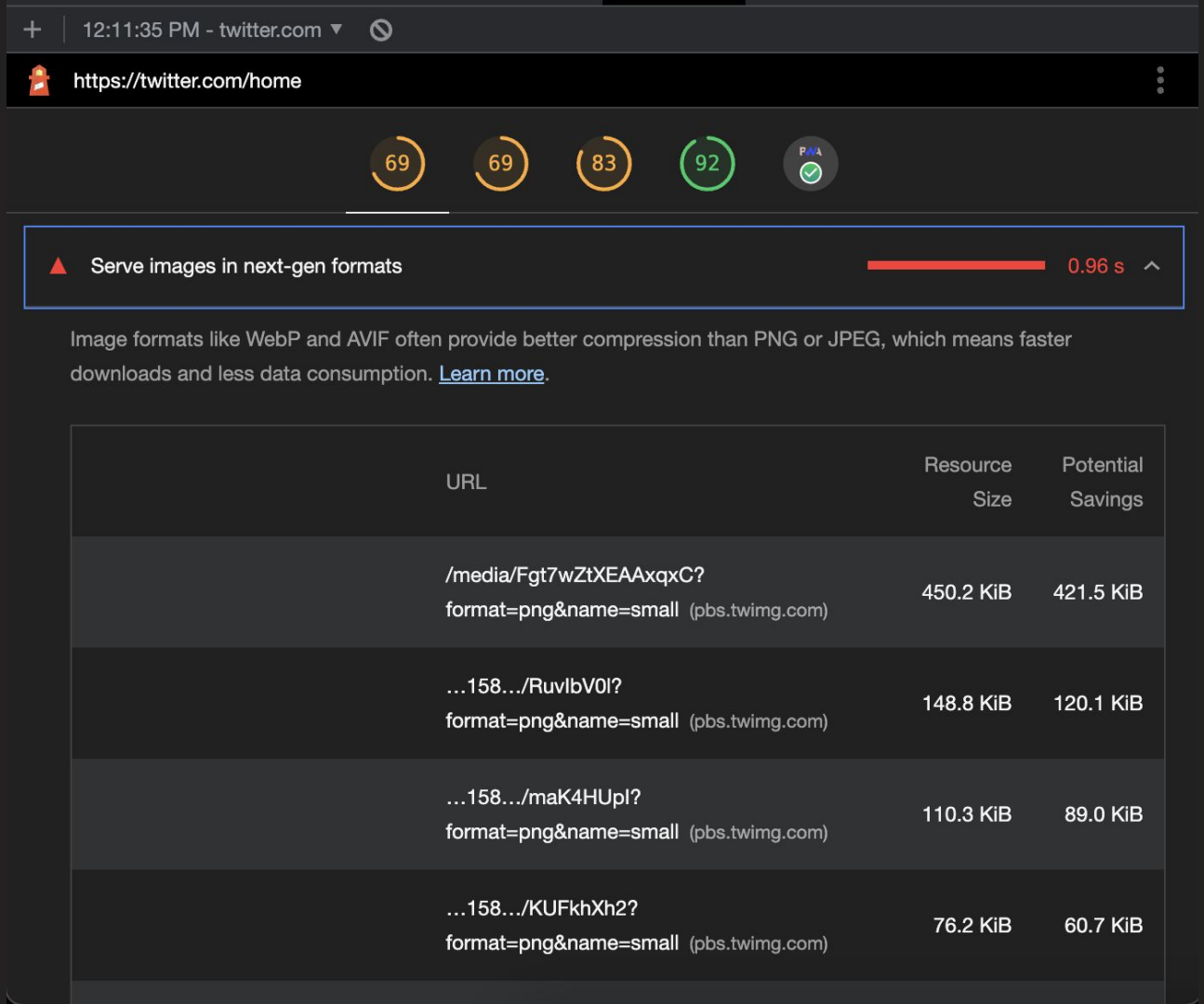


Image formats like WebP and AVIF often provide better compression than PNG or JPEG, which means faster downloads and less data consumption. Learn more.

URL	Resource Size	Potential Savings
/media/Fgt7wZtXEAAxqxC?format=png&name=small (pbs.twimg.com)	450.2 KiB	421.5 KiB
…158…/RuvIbV0l?format=png&name=small (pbs.twimg.com)	148.8 KiB	120.1 KiB
…158…/maK4HUpl?format=png&name=small (pbs.twimg.com)	110.3 KiB	89.0 KiB
…158…/KUFkhXh2?format=png&name=small (pbs.twimg.com)	76.2 KiB	60.7 KiB

Google's case study: 1 million images

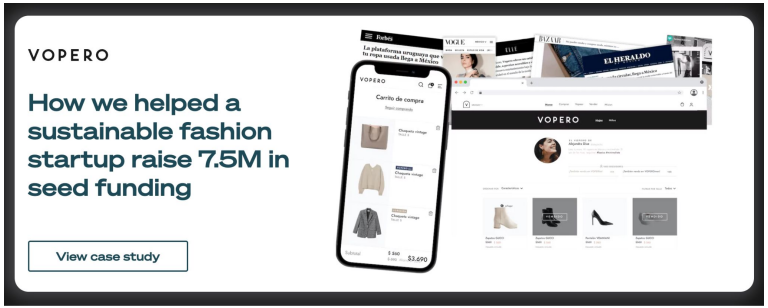
Type	Avg PSNR Obtained	Avg Compression % (non-negative compression gain)	Avg Compression % (negative compression gain)
WebP	39.38	41.30	39.80
JPEG 2000	39.49	27.67	9.71
Re-JPEG	39.36	22.37	14.62

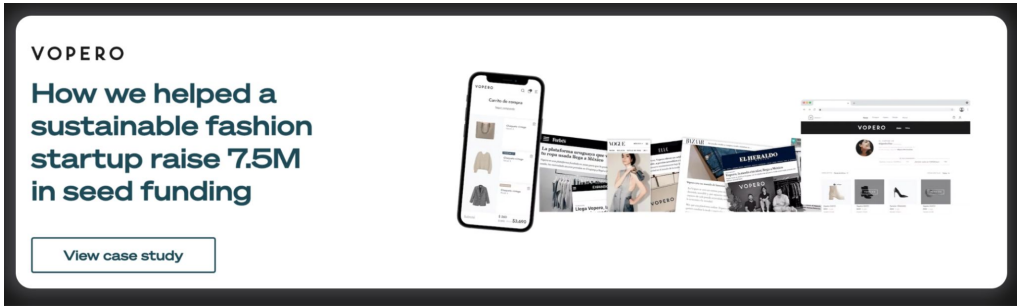
WebP

How to display a WebP image: enter picture tag

Image tag: ‹img›
- resolution switching → srcset

Picture tag: ‹picture› (brother of ‹video› and ‹audio›)
- resolution switching → srcset
- art direction → media





```
<picture>
  <source srcset="test.avif" type="image/avif">
  <source srcset="test.webp" type="image/webp">
  <img src="test.png" alt="test image">
</picture>
```

```
<picture>
  <source srcset="img-small.webp" type="image/webp" media="(max-width: 600px)">
  <source srcset="img-big.webp" type="image/webp">
  <source srcset="img-small.png" type="image/png" media="(max-width: 600px)">
  <source srcset="img-big.png" type="image/png">
  <img src="img-big.png" class="ps-xl-4" alt="car">
</picture>
```

```
<picture>
  <source media="(max-width: 799px)"
          srcset="480w-close-portrait.webp
                  480w-close-portrait-2x.webp 2x">
  <source media="(min-width: 800px)" srcset="img-full.webp">
  <img src="img-full.jpg">
</picture>
```

WebP + Rails

WebP + Rails

No native support for:

- Pre processing: **JPEG/PNG → WebP**
- Displaying WepP: there's no **picture_tag** helper
- Serving WebP images

No good 3rd-party options:

- assume a lot of things (specific reality of who implemented it)
- solve part of the problem (conversion, display, serving)
- don't even work
- no stars / community (unmaintained)
- bad documentation

WebP + Rails

Rails 6.1.4 - Ruby 3.0.1 - ActiveStorage and WebP image format

Asked 7 months ago Modified 2 months ago Viewed 432 times

3

There is no good info about how to setup Rails for serving WebP images with ActiveStorage.

Can someone explain how to do it?

I try:

application.rb

```
config.active_storage.web_image_content_types = %w(image/jpeg image/png image/webp image/jpg)
```

And in **View**:

```
<% image_tag( f.image_1.variant(resize_to_limit: [800,600], format: :webp) ) %>
```

But this works on development (I see link to jpeg but when I use right mouse button and "save image as" image is saved as .WebP

On production I see **no image default icon** for browser and link to .jpg

RoR is dead...

Carrierwave::WebP

stability experimental

gem version 0.1.0 maintainability A Dependency Status Still Maintained

Coderwall

This gem provides an ugly, but working solution for converting uploads to the WebP format.

I created it as an expirement, so gem's API can change in the future. You've been warned!

Motivations for this gem

In my experience, the existing gems for processing webp images in the Rails Asset Pipeline made questionable assumptions. More importantly, I could not use the native libraries used by those gems in my team's acceptance/production environment. ImageMagick was an easy choice for this reason because it's widely implemented, used, and understood (my opinion).

Multiple problems to solve

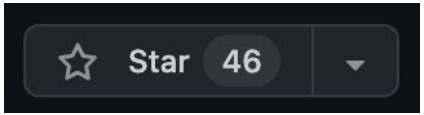
- Converting images to WebP
- Handle both static and dynamic assets
  - /app/assets/images
  - Carrierwave, ActiveStorage, etc
- Correctly handling the picture tag: view helper
- Serving WebP: Nginx redirection?
- Asset digests
- Handling different files for different sizes (art direction)
- Working in development → what about images that are not yet converted?
- WebP images not still in the asset path (sprockets-manifest file)

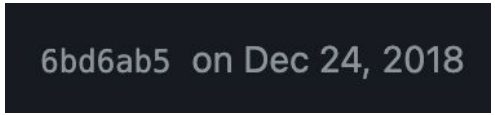
Option 01

Use something that already exists:

https://github.com/kavu/sprockets-webp

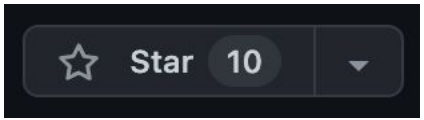
→ Doesn't work. Only compiles assets

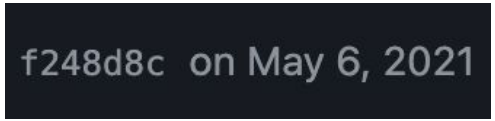




https://github.com/0xjmp/rails-webp

Bug with assets digest. Only compiles assets





Option 02

Let's manually maintain WebP images alongside PNGs/JPEGs

- really hard to maintain
- what about dynamic assets?
- helper to display them alongside PNGs/JPEGs?

Option 03

- Have some script that converts to WebP
  - Same digest as original asset
- Handle redirection in Nginx (or similar)
  - Rewrite rule
- Development?

Option 04

WebP assets

- Static assets
  - add them somehow to assets/images
  - precompile them to public/assets
- Dynamic assets have to be handled separately

Picture tag

- Specify everything if needed (picture tag attributes, source, etc)
- Automatically take JPEG/PNGs and inference WebP images files paths
- Take into account both static/dynamic assets

Let’s build this!

Selected approach

WebP assets

- Convert existing static JPEG/PNG images to WebP with
  - **.enhance()** the **assets:precompile** task
- Dynamic assets are converted separately
  - helper to generate WebP versions

Picture tag

- Add a **picture_tag** helper for 3 use cases:
  - full control → pass a block
  - Array
  - automatic inference → add_webp option

WebP Assets

```ruby
require 'webp-ffi'

namespace :assets do
  desc 'Create .webp versions of assets'
  task webp: :environment do
    image_types = /\.(?:png|jpe?g)$/

    public_assets = File.join(
      Rails.root,
      'public',
      Rails.application.config.assets.prefix)

    Dir["#{public_assets}/**/*"].each do |filename|
      .
      .
      WebP.encode(filename, webp_file, quality: 80)
      .
      .
    end
  end

  # Hook into existing assets:precompile task
  Rake::Task['assets:precompile'].enhance do
    Rake::Task['assets:webp'].invoke
  end
end
```

Carrierwave

```ruby
class ImageUploader < CarrierWave::Uploader::Base
  include CarrierWave::MiniMagick
  include NextGenImages::CarrierwaveHelpers

  version :small, from_version: :small do
    process resize_to_limit: [400, 400]
  end

  version :webp do
    process convert_to_webp: [{ quality: 80 }]

    version :small do
      process resize_to_limit: [400, 400]
      process convert_to_webp: [{ quality: 80 }]
    end
  end
end
```

ActiveStorage

```ruby
class User < ApplicationRecord
  has_one_attached :avatar do |attachable|
    attachable.variant :small, resize_to_limit: [400, 400]
    attachable.variant :webp, {
      convert: :webp,
      format: :webp,
      saver: { quality: 80 }
    }
    attachable.variant :webp_small, {
      resize_to_limit: [400, 400],
      convert: :webp,
      format: :webp,
      saver: { quality: 80 }
    }
  end
end
```

Picture tag helper

```
= picture_tag 'img-big.png', image: { alt: 'car', class: 'ps-xl-4' } do
  = source_tag srcset: "#{image_path('img-small.png')}.webp", media: '(max-width: 600px)'
  = source_tag srcset: "#{image_path('img-big-png')}.webp"
  = source_tag srcset: image_path('img-small.png'), media: '(max-width: 600px)'
  = source_tag srcset: image_path('img-big.png')
```

```
<picture>
  <source srcset="img-small.png.webp" type="image/webp" media="(max-width: 600px)">
  <source srcset="img-big.png.webp" type="image/webp">
  <source srcset="img-small.png" type="image/png" media="(max-width: 600px)">
  <source srcset="img-big.png" type="image/png">
  <img src="img-big.png" class="ps-xl-4" alt="car">
</picture>
```

```
= picture_tag [post.cover_image.webp.medium.url, post.cover_image.medium.url]
    , image: { class: 'rounded-2' }
```

```
<picture>
  <source srcset="blog-cover-medium.jpeg.webp" type="image/webp">
  <source srcset="blog-cover-medium.jpeg" type="image/jpeg">
  <img src="blog-cover-medium.jpeg" class="rounded-2">
</picture>
```

```
= picture_tag 'vopero.png', image: { class: 'mb-3' }, add_webp: true
```

```
<picture>
  <source srcset="vopero.png.webp" type="image/webp">
  <source srcset="vopero.png" type="image/png">
  <img src="vopero.png" class="mb-3">
</picture>
```

Internals

```ruby
def picture_tag(source, options = {}, &block)
  picture_options = options.except(:image)

  content_tag :picture, picture_options do
    build_picture_content(source, options, block)
  end
end
```

```ruby
def build_picture_content(source, options, block)
  image_options = options.fetch(:image, {})
  image_options[:src] = build_img_src(source)
  add_webp = options.fetch(:add_webp, false)

  content = ''.html_safe
  if block.present?
    content += capture(&block).html_safe
  else
    [source].flatten.each do |img_src|
      content += build_source_from_img(image_path(img_src), add_webp)
    end
  end
  content += tag('img', image_options)
end
```

```ruby
def build_source_from_img(img_path, add_webp)
  # try to find WebP image from "normal" image
  ...
end

def source_tag(options = {})
  tag :source, options
end

def build_img_src(source)
  ...
end

def image_type(image_path)
  ...
end

def file_exist_in_public_path?(path)
  ...
end
```

▲ Serve images in next-gen formats 0.8 s

Image formats like WebP and AVIF often provide better compression than PNG or JPEG, which means faster downloads and less data consumption. Learn more.

	URL	Resource Size	Potential Savings
vopero image `<img alt="vopero image" class="d-none d-xxl-block me-3 me-xl-5 img-fluid" src="/assets/startups/vopero_img_big-9074d045dba7c49978c0faccf7eacdd0fb4e59ac70…">`	…startups/vopero_img_big-9074d04….png (localhost)	715.4 KiB	435.6 KiB
the appraisal image `<img alt="the appraisal image" class="w-50 me-4 me-lg-6 me-xl-10 me-xxl-0 img-fluid" src="/assets/startups/tal_img-520a877….png">`	…startups/tal_img-520a877….png (localhost)	382.5 KiB	289.8 KiB

Test it out!

github.com/eagerworks/next_gen_images

Next steps:

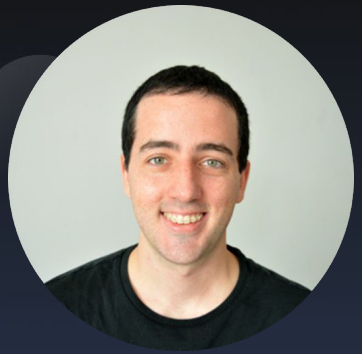
- support other image formats
- user-configurable

Reference

- https://developer.chrome.com/blog/search-ads-speed/
- https://blog.hubspot.com/marketing/page-load-time-conversion-rates
- https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/mobile-site-speed-importance/
- https://developers.google.com/speed/webp/docs/c_study
- https://developers.google.com/speed/webp/docs/webp_study
- https://blog.bitsrc.io/why-you-should-use-picture-tag-instead-of-img-tag-b9841e86bf8b
- https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Responsive_images
- https://www.reuters.com/technology/musk-orders-twitter-cut-infrastructure-costs-by-1-bln-sources-2022-11-03/
- https://stackoverflow.com/questions/71587757/rails-6-1-4-ruby-3-0-1-activestorage-and-webp-image-format

eagerworks

Thanks!



JP Balarini
CTO @ eagerworks
@jpbalarini

next_gen_images