MarkdownでLazy image loading

2019-10-03

表参道.rb #51

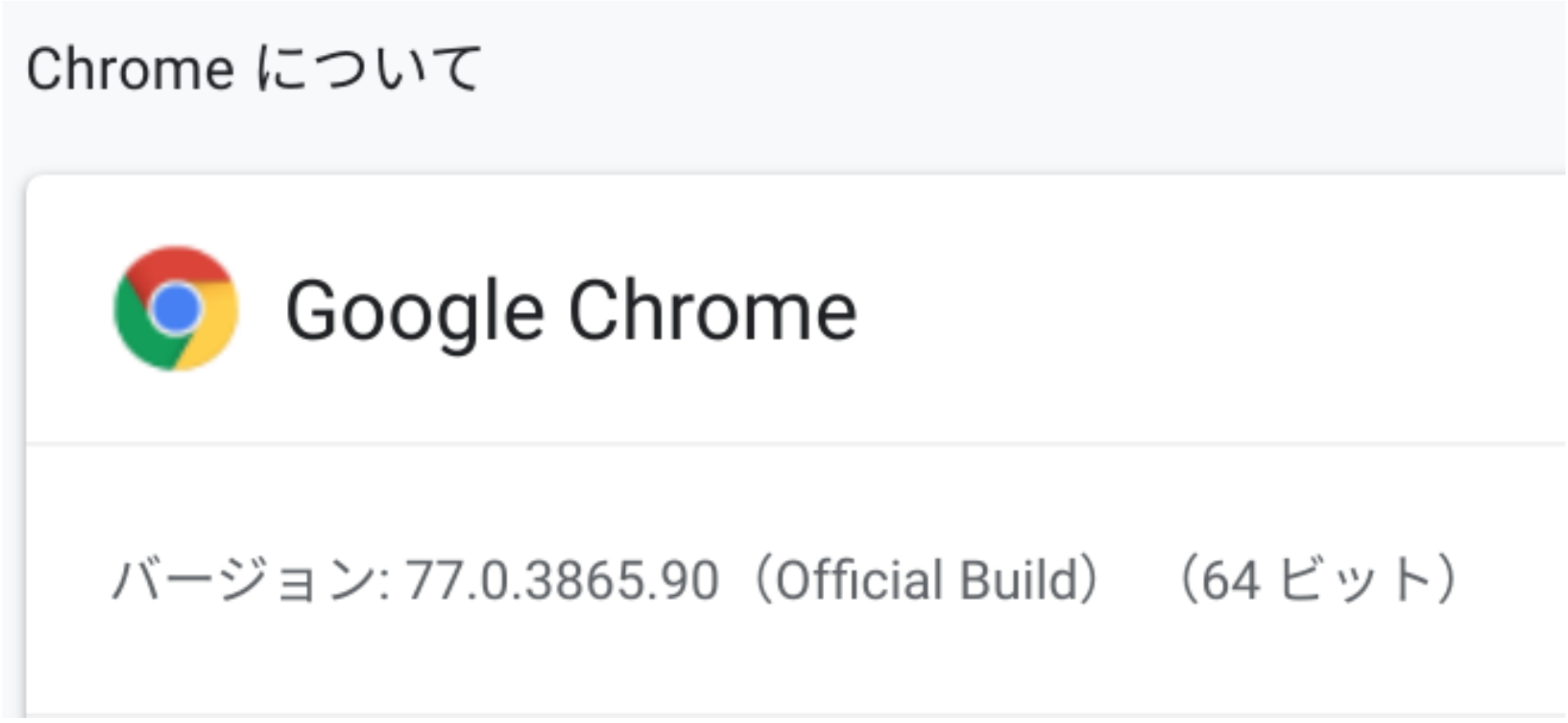
うなすけ

自己紹介

- 名前 : うなすけ
- 仕事 : 株式会社バンク (エンジニア)
  - インフラ寄りサーバーサイドエンジニア
  - Ruby, Rails, Kubernetes…
- GitHub @unasuke
- Mastodon @unasuke@mstdn.unasuke.com
- Twitter @yu_suke1994

はじめに

See your chrome://settings/help



Native lazy-loading from Chrome 75!

- Native lazy-loading for the web | web.dev
- AddyOsmani.com - Native image lazy-loading for the web!
- 遂に来る！ Chrome 75にてLazyLoadが正式に実装されるようです🎉 | フロントエンドBlog | ミツエーリンクス

loading=”lazy”

Slimの場合

```
img(src="https://example.org/image.png" loading="lazy")
```

Hamlの場合

```
%img{src:"https://example.org/image.png", loading: "lazy"}
```

loading=”lazy”

Markdownの場合

```
<img src="https://example.org/image.png" loading="lazy" />
```

……いやいやいや

Markdownの場合の理想

こう書いたら

```
![](https://example.org/image.png)
```

こうなってほしい

```
<img src="https://example.org/image.png" loading="lazy" />
```

Markdownの場合の理想

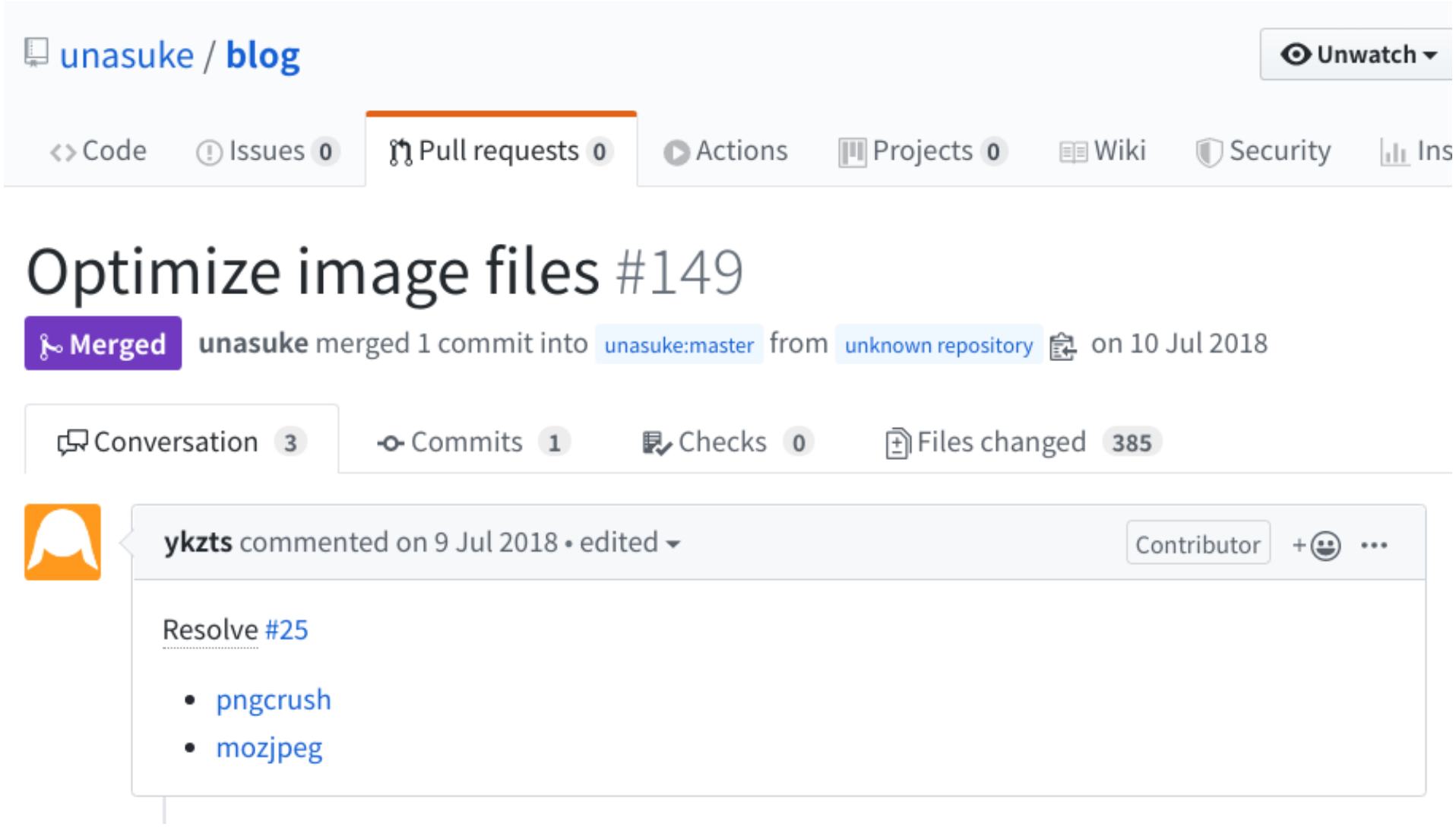
なぜそんなことを気にしているのか

blog.unasuke.comの事情

- Middlemanと、blogプラグインによって生成されている
- 記事自体はMarkdownで書かれている
- 画像が重いというご意見がある

画像が重いというご意見

https://github.com/unasuke/blog/pull/149



つまりこういうこと

- blog.unasuke.comは画像が重いことに定評がある
- img に `loading="lazy"`を付けられたらいいかも？
- いやでもMarkdownやんけ……

どういう手順で進めるか

理想

1. `loading="lazy"` で本当に早くなるのか計測
2. Markdownの変換結果になんとかして `loading="lazy"`をつける

実際

1. Markdownの変換結果になんとかして `loading="lazy"`をつける
2. 本当に早くなるのか計測

なぜ計測ファーストでないか

- 変換結果に介入できることが検証できなければ、早くなることが確認できても実現できないため
- gemを作りたいという気持ちのほうが先にあった
- うっかり

実装やっていきましょう

- blog.unasuke.comではRedcarpetを使っている
  - https://github.com/vmg/redcarpet
- RedcarpetにはCustom Rendererを指定できる
- Custom Rendererで `image(link, title, alt_text)` の返り値をいじればできそう

see also
Railsでカスタムmarkdownを実装する - k0kubun's blog

Minimum code

```ruby
class CustomRenderer < ::Redcarpet::Render::HTML
  def image(link, title, alt_text)
    "<img loading=\"lazy\" src=\"#{link}\" alt=\"#{alt_text}\" />"
  end
end
```

これでやりたいことは実現できる

gemができました

https://github.com/unasuke/redcarpet-render-html_lazy_img

便利QRコード

gemができました

```ruby
markdown = Redcarpet::Markdown.new(Redcarpet::Render::HTMLLazyImg::Lazy)
markdown.render('![example image](https://example.com/img.png)')
# => '<p><img loading="lazy" src="https://example.com/img.png" alt="example image" /></p>'


markdown = Redcarpet::Markdown.new(Redcarpet::Render::HTMLLazyImg::Auto)
markdown.render('![example image](https://example.com/img.png)')
# => '<p><img loading="auto" src="https://example.com/img.png" alt="example image" /></p>'


markdown = Redcarpet::Markdown.new(Redcarpet::Render::HTMLLazyImg::Eager)
markdown.render('![example image](https://example.com/img.png)')
# => '<p><img loading="eager" src="https://example.com/img.png" alt="example image" /></p>'
```

`loading` attributeの他の値にも対応
(auto, eager)

Middlemanで使用するには

config.rbにこのように書けばOK

```
set :markdown, renderer: Redcarpet::Render::HTMLLazyImg::Lazy
```

では計測しましょう

使用するのは僕のブログ記事でも大量に画像を使用しているこの記事！

TEX Yoda Trackpoint Keyboardを買った | うなすけとあれこれ

計測方法

1. 上記記事をhtmlとしてローカルに保存
2. `<img>` に何もなし、`loading="lazy"`, `loading="eager"`を指定したものの3つを用意
3. `chrome://flags/#enable-lazy-image-loading` を Enabledに
4. Chrome Developer toolsでloadingやperformanceを見る

※ もちろん Disable cacheの状態で

結果 (6回計測)

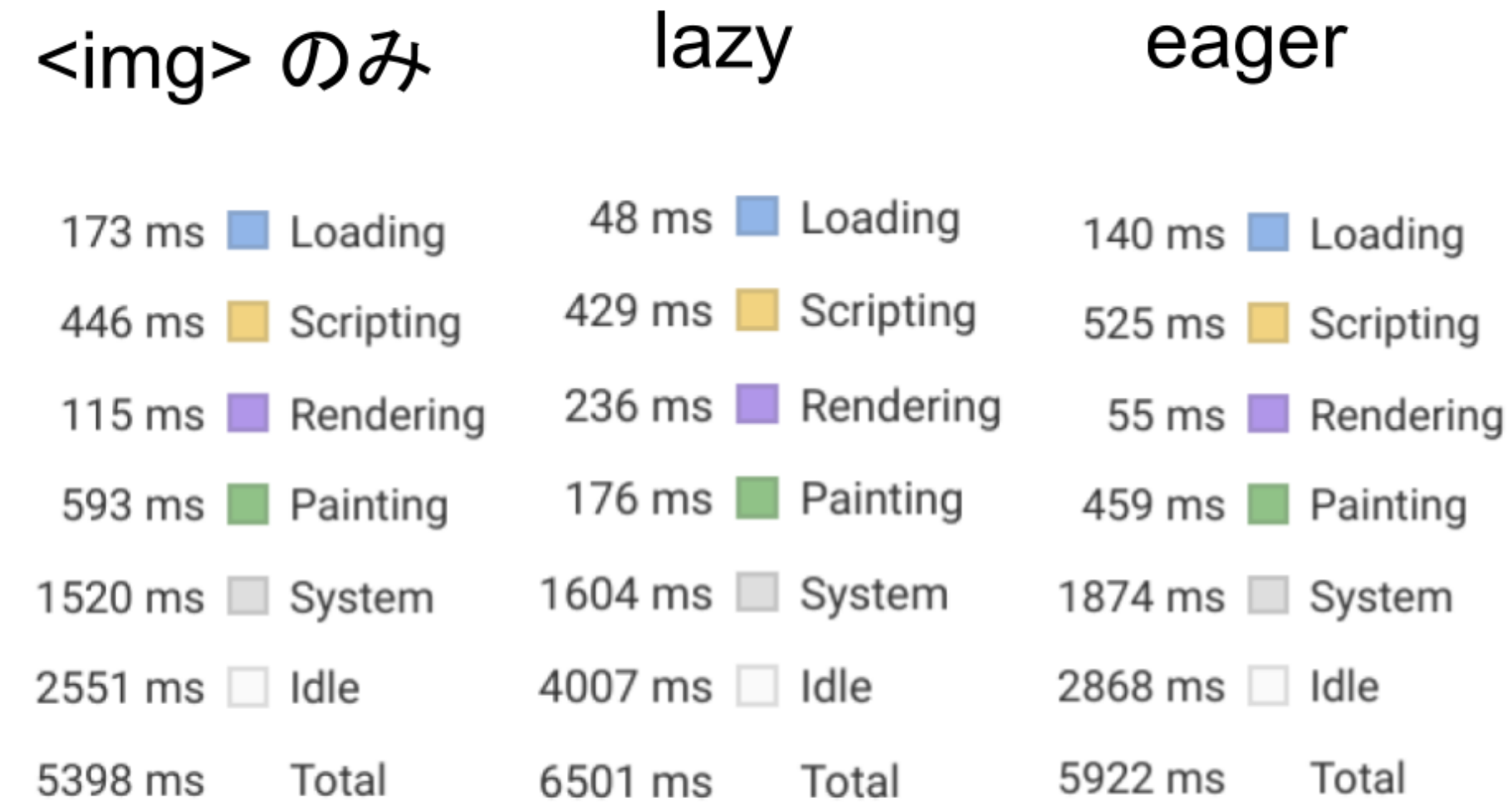
Load eventが発火するまでの時間

	最悪値	最速値	平均値
pure img	8.33	5.42	6.25
lazy	6.89	4.96	5.63
eager	7.83	4.97	6.47

Load eventはページ内の要素が全て読み込まれたら発火するのであまり意味のない計測だったかな……

結果 (1回のみ取得)

Performanceの結果



まとめ

- Loadingにかかる時間が削減されておりよさそう
- 画像のサイズは変わらないが、ページ自体は軽くなったのでは(サイズの話ではない)
- Webの最新技術を使うのは楽しい！！！