Vue.js docs performance improvements

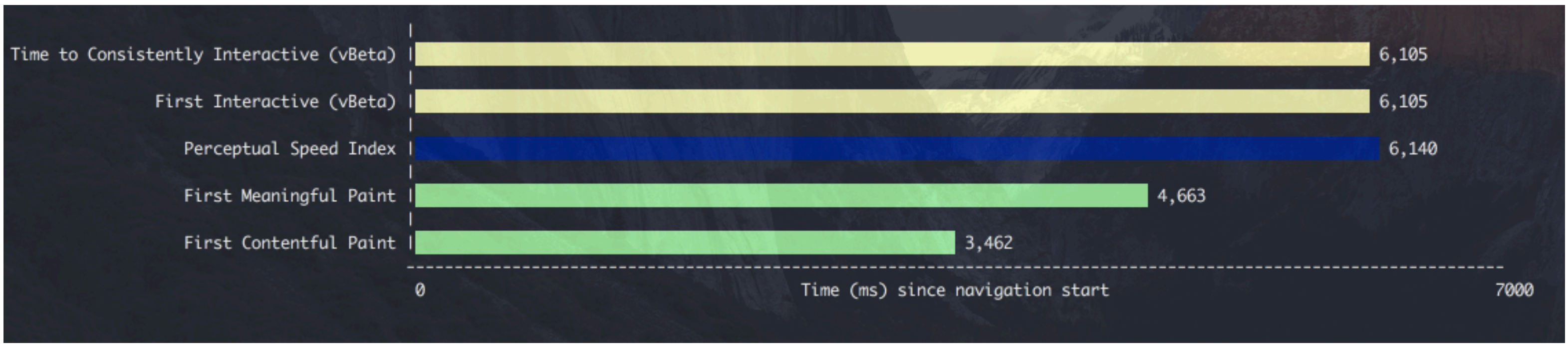
by @denar90

Test setup

- **Device:** Nexus 5 device emulation
- **Network:** 1.6 Mbps network throttling
- **Url:** localhost:4000/v2/guide/

Hi. I wanted to share some ideas about performance improvements for the Vue.js documentation. To set the base line I used pwmetrics which is a tool to gather web performance metrics. So let's have a look in what shape the documentation is:

P.S. It's taking measurements using Nexus 5 device emulation and 1.6 Mbps network throttling.

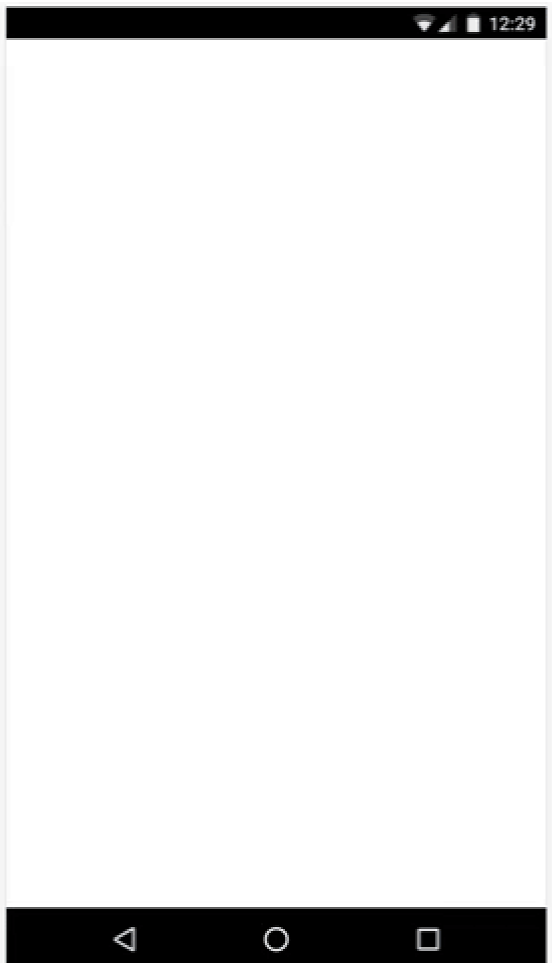


A first meaningful paint at around five seconds and a perceptual speed index around 6000 show that there is an opportunity for performance improvements.

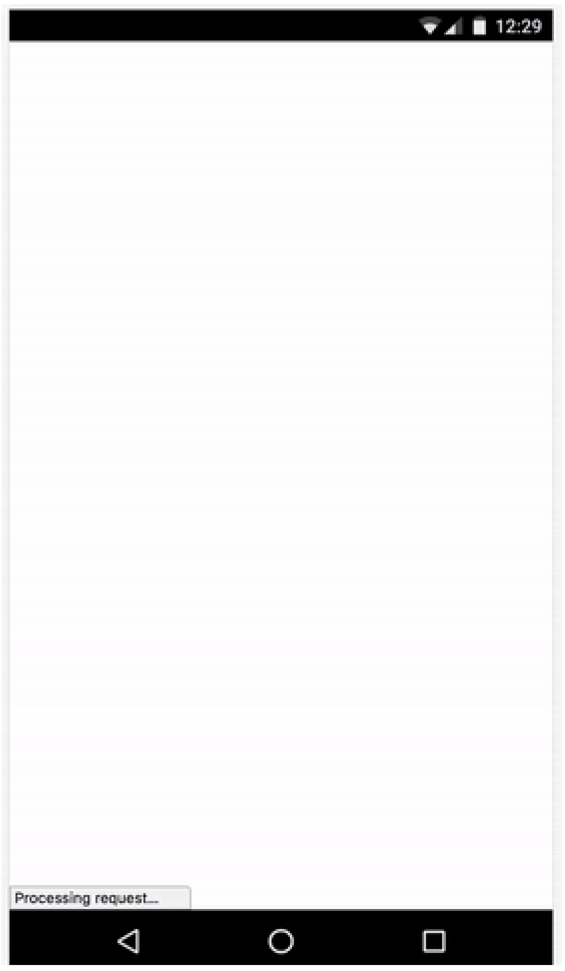
Flash of invisible content vs. flash of unstyled content

A thing I noticed while measuring is that there is also different behavior from user perspective of view of **FOIC** (flash of invisible content) and **FOUC** (flash of unstyled content) through the browsers. Regarding to progressive enhancement strategy,
It will be nicer if core content be loaded and then enhanced.

FOIC



FOUC

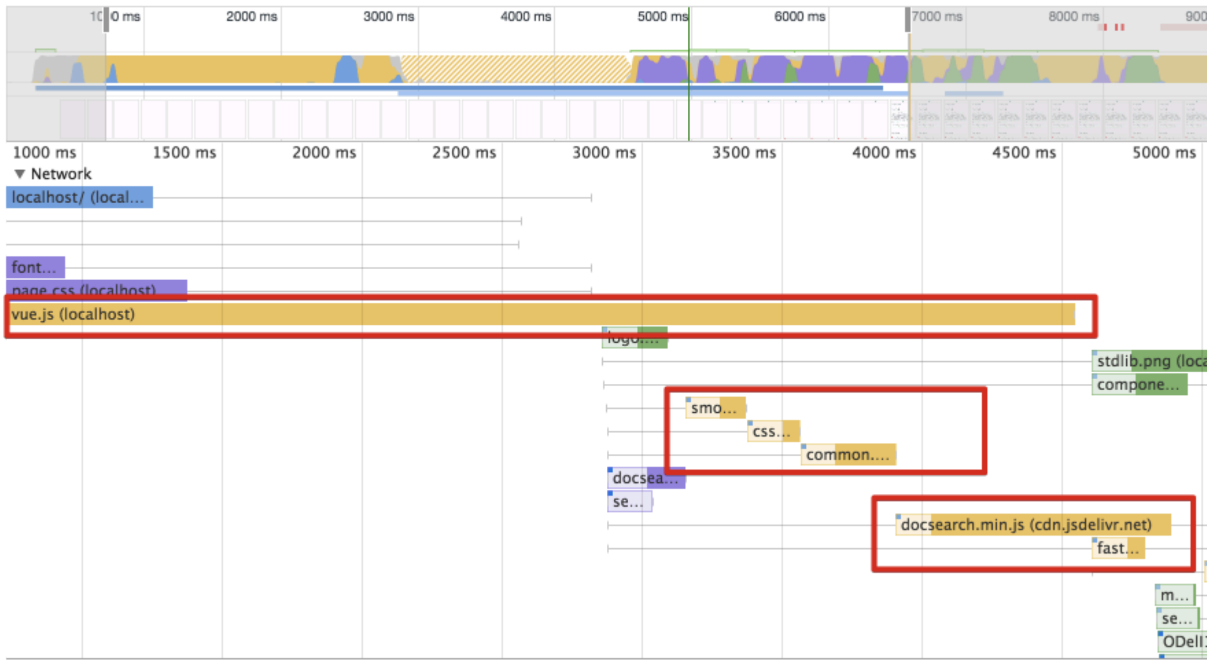


To reach this goal fallback font could be shown as soon as possible.

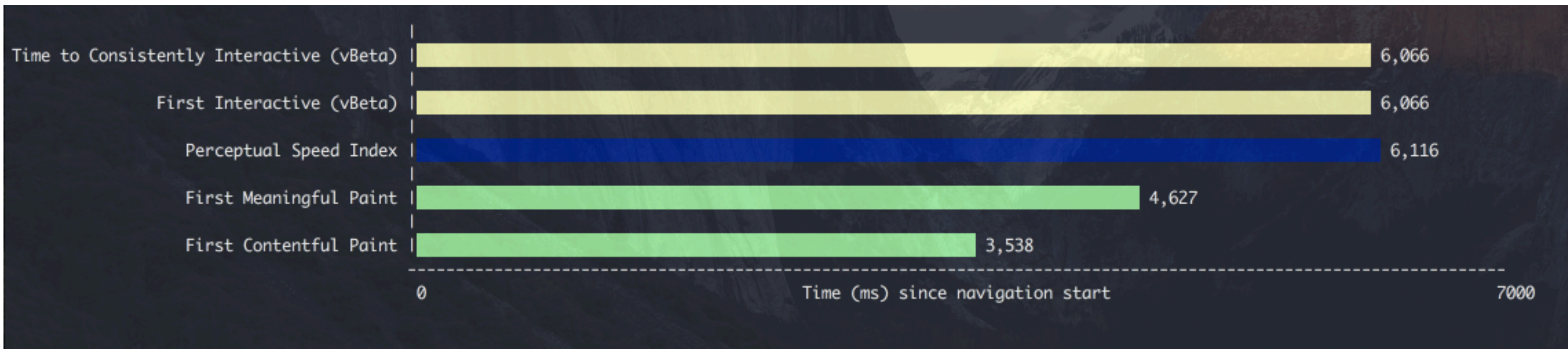
Why takes the first paint so long?

It turns out that the documentation includes a lot of scripts that are render blocking the preventing the first paint from happening.

Timeline trace



You can fix this behavior by adding the `*defer*` attribute to all these blocking scripts (except vue.js, we will manage it later) and you can see immediate improvements.

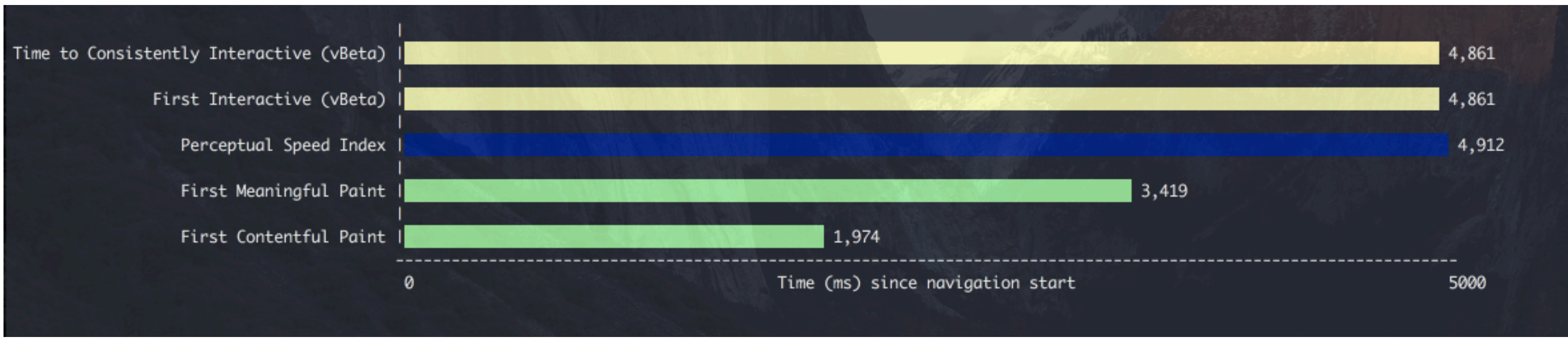


Comparing metrics, Time to Consistently Interactive (**TTCI**), Time to First Interactive (**TTFI**), before and after improvement you can notice *100ms* speedup – this is not so impressive but it's at least a start.
Commit - [b996ecf]

So let's go on...

Adding defer to `vue.js` saved us
FCP - `3,528 sec` -> `1,972 sec`
FMP - `4,627 sec` -> `3,419 sec`

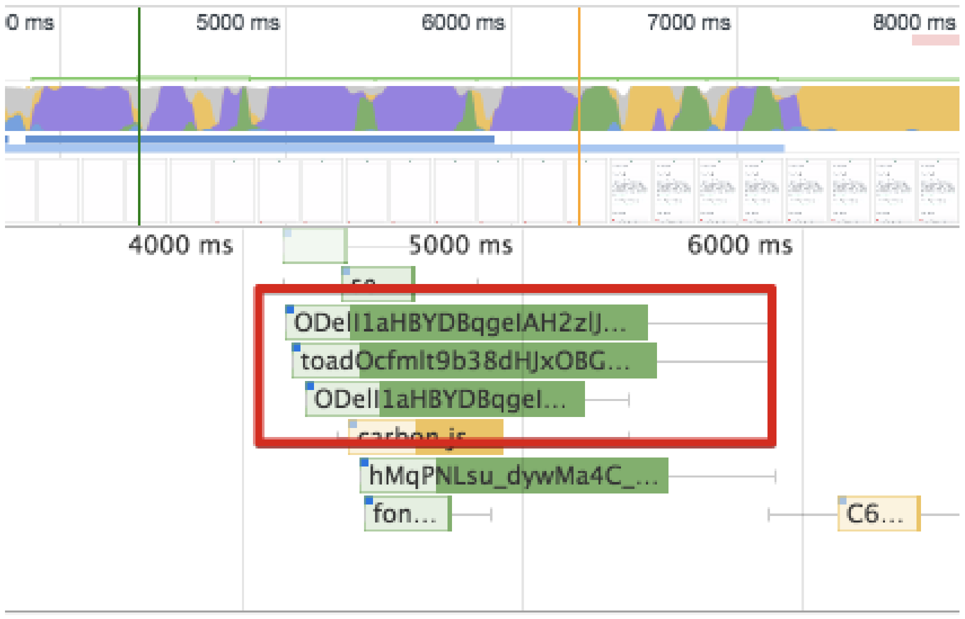
PSI - `6,116 sec` -> `4,912 sec`
TTFI - `6,066 sec` -> `4,861 sec`
TTCI - `6,066 sec` -> `4,861 sec`



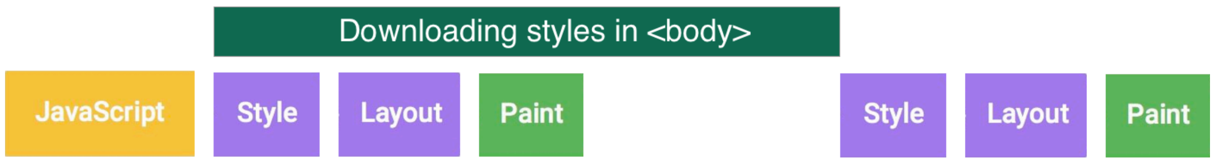
Good, but we can do more.
Commit - [97ea5db]

Looks like fonts loading is blocking **FMP**

[Timeline trace]

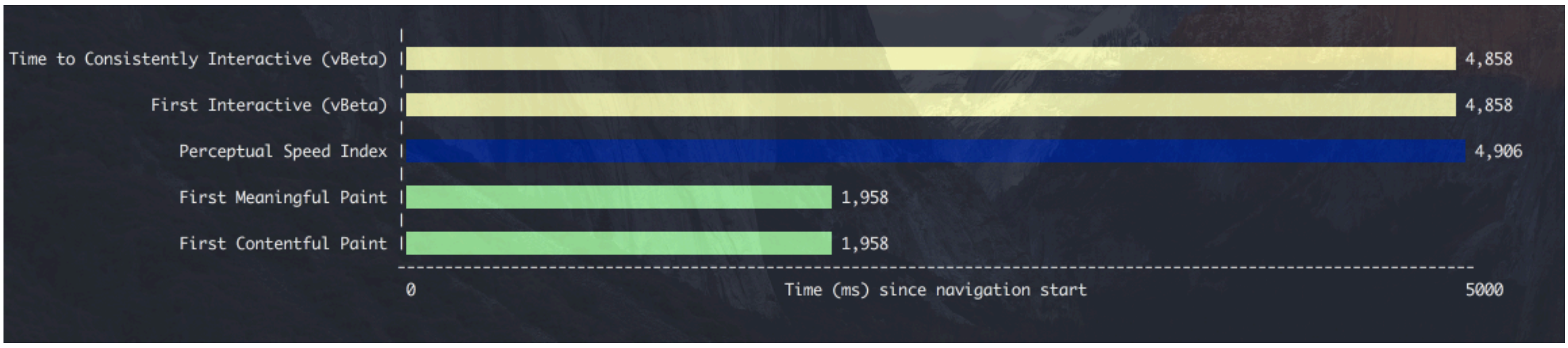


Since browsers repaint after loading resources found in the body element.



we can cheat and move loading resources fonts there.
Commit - 7fba6b7

Results:

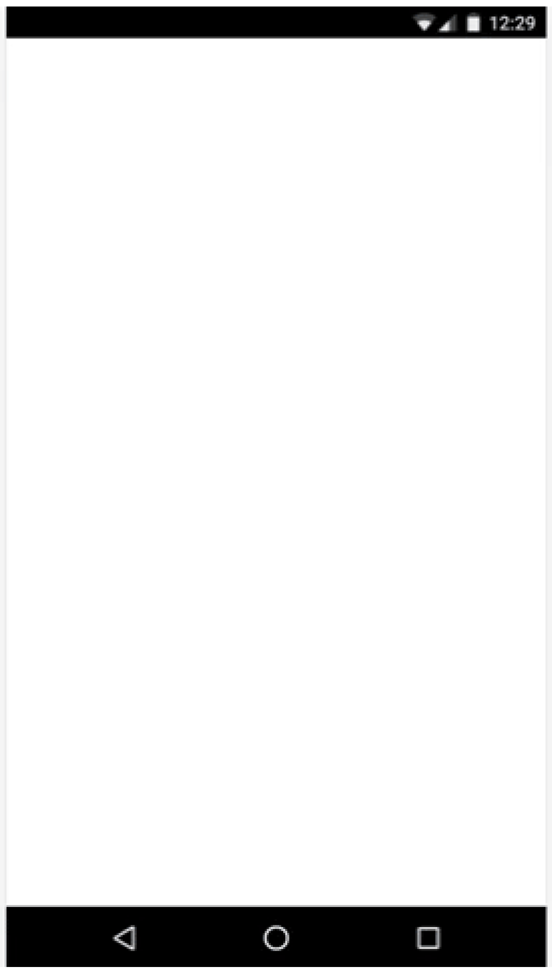


Really really nice results on chart.

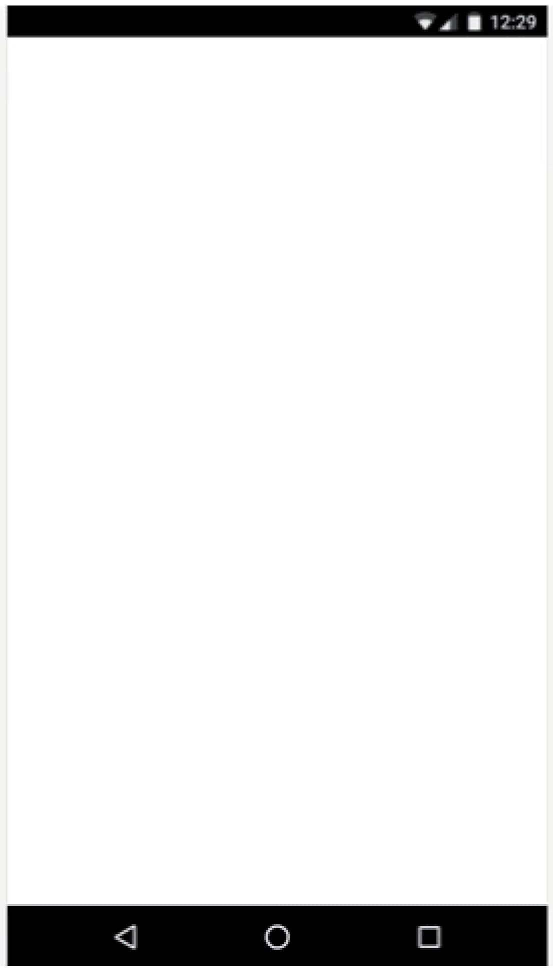
FMP, **FCP** under `**2 sec**`
TTCI, **TTCI**, **PSI** under `**5 sec**`

How it looks in browser

FOIC



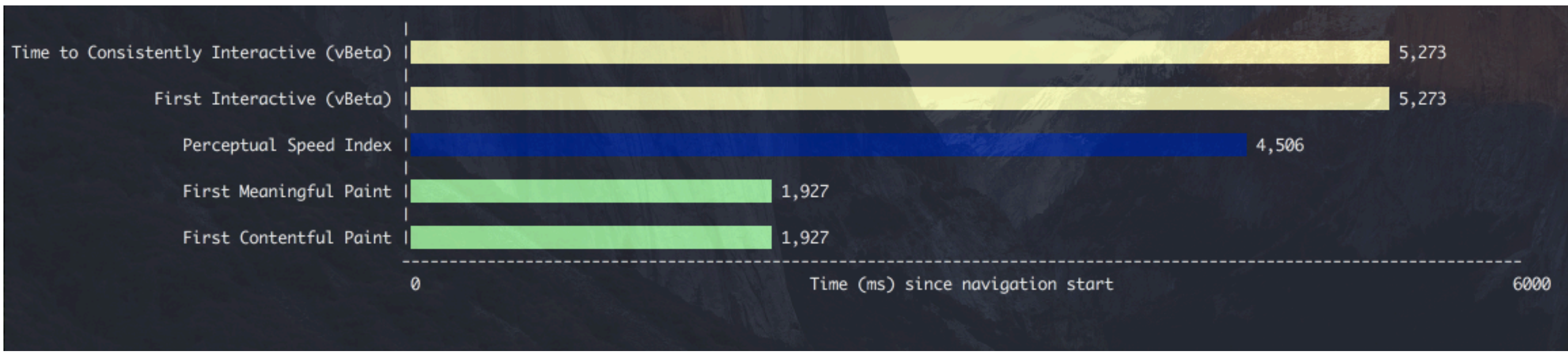
FOUC



There some blink In browsers with FOIC strategy between fallback font and styled font which is not nice behavior form user experience.

There several libraries handling loaded fonts, I propose to use fontfaceobserver.

So, results are:
Timeline trace

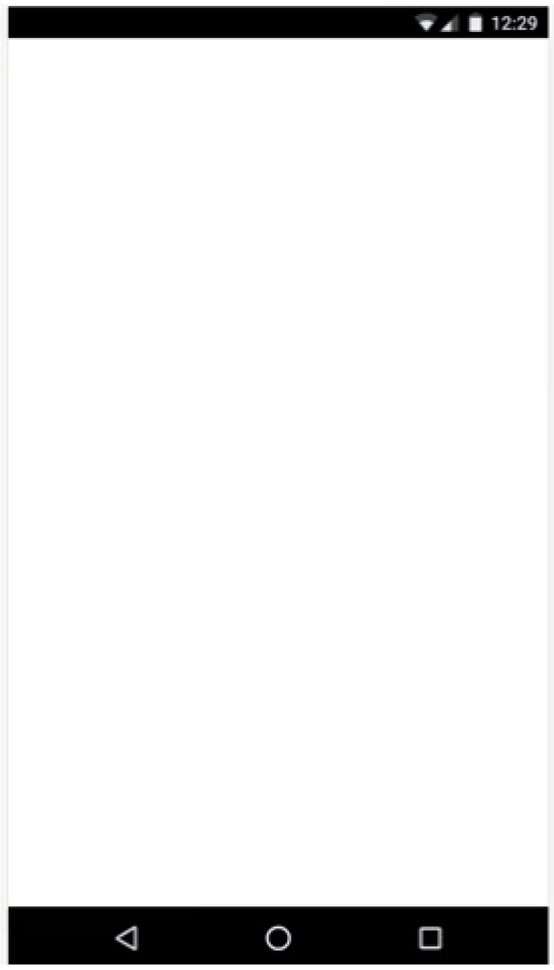


FMP, **FCP** under `2 sec`, still the same
TTCI, **TTCI**, **PSI** under `5,2 sec` instead of `4,8 sec`

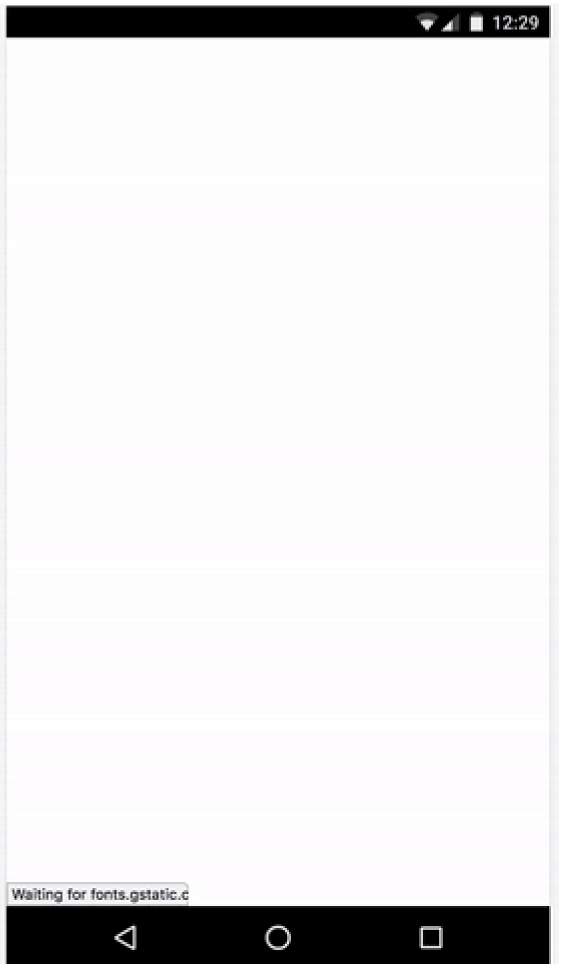
This fix cut another `400ms`! 🎉
Commit - [6b13c3]

So, last gifs

FOIC

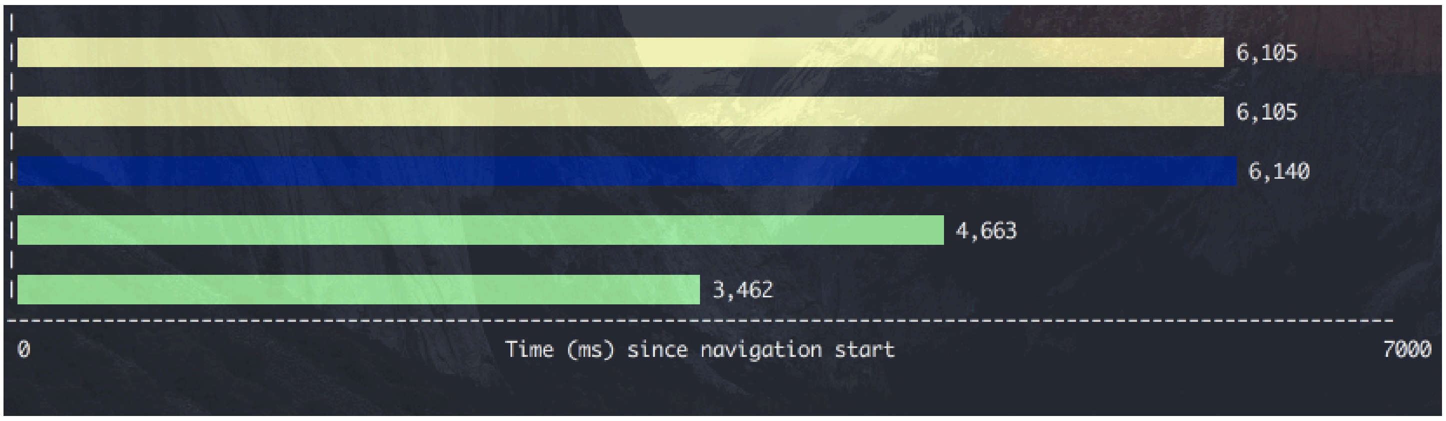


FOUC

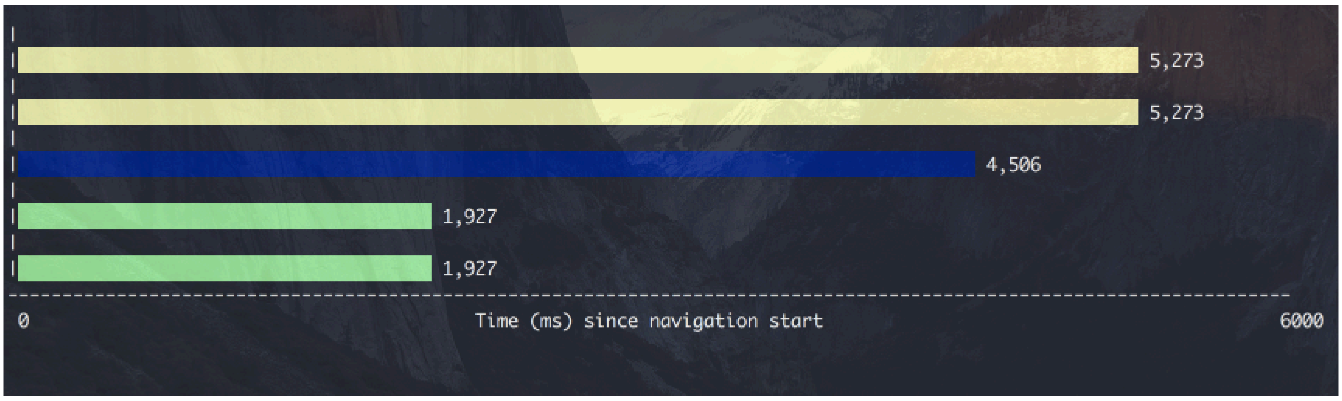


And to compare the before and after...

Before:



After:

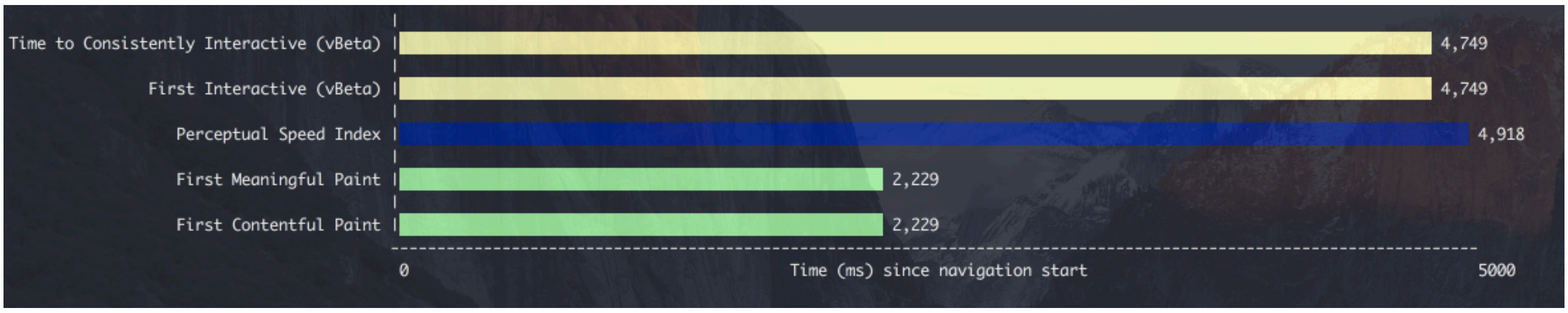


(Image altered to match horizontal scale)

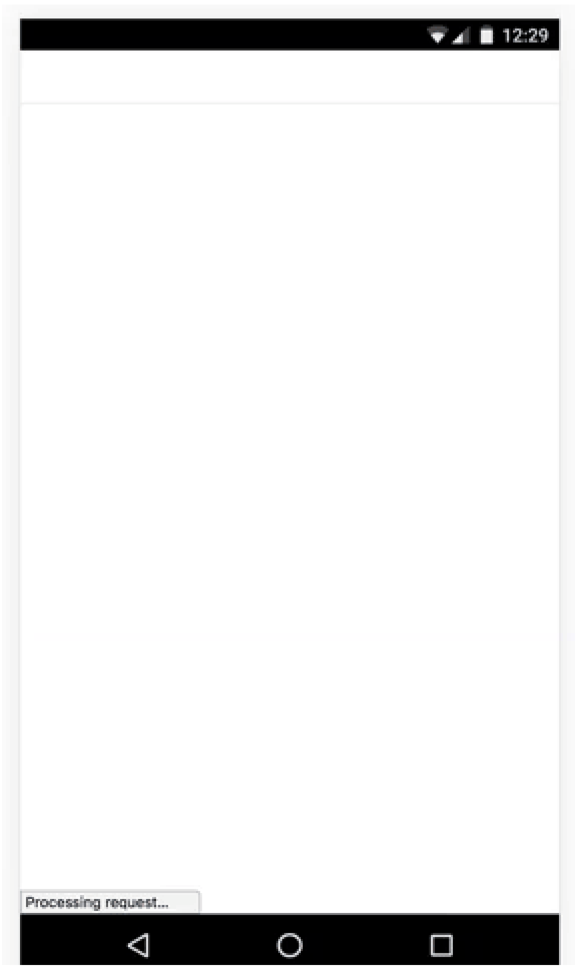
You can take a look at pull request
Thanks to @paulrish

P.S. In case supporting only evergreen browsers `[font-display](font-display)` is a new way to manage FOIC.

I also reached some results adding `*font-display: swap*`



And gif for FOIC.



Processing request...