Web Performance Fundamentals

29/04/2021 1:13 PM

* Psychology Of waiting
  - People want to start
  - Bored / Anxious / Unexplained / Uncertain
    - feels slower
  - People will wait for value

* Measuring Performance — As soon as you start attaching value to a metric it is bound to be gamed. Hence, new metrics were needed.

> New Metrics — Core Web Vitals

1. FCP : First contentful paint
2. LCP : Largest contentful paint
3. CLS : Cumulative layout shift
4. FID : First input delay

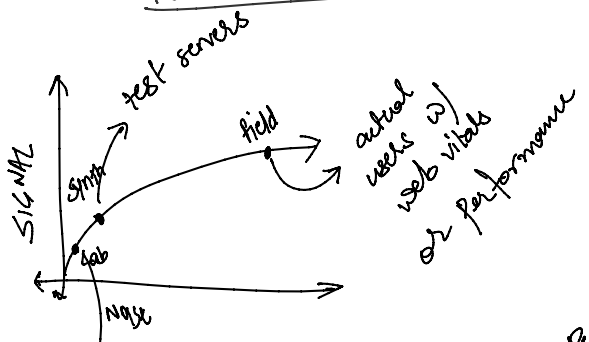
> FCP — Time between link is clicked to the time first bit of content painted
RESPOND QUICK!!

> LCP — Time between clicked and largest area painted
GIVES AN IDEA ABOUT WEBSITE IS READY!

> CLS — Recorded over the lifetime of the app
Total movement distance
DONT MOVE STUFF AROUND!

All of them have a certain range of values.

> FID — Delay from the first input / interaction and your browser registering it & starting to respond to it
DONT LOAD TOO MUCH



* Reading these data → p50 p75 p95 → worst users
  - percentile
  - median

* Performance API → provided by the browser to capture events such as paint, render, interactions, etc.
↳ PerformanceObserver
↳ Using these you can build your own web vitals on top of it

* Improving FCP → for improving anything one of the easiest thing you can do is → DO LESS

> for FCP → (1) servers (2) Doc size (3) network

> servers: — sized correctly — minimal processing — network bandwidth

> document: — doc size — compression → gzip, brotli

> network: CDN
Content Distribution Network

* Improving LCP — (1) Defer resources until later (2) Optimize images (3) Reduce request overhead [caching] & [HTTP 2]

> Use "view original trace" under "performance" to get a waterfall image of the requests

> Defer resource: script — async → loads later, but not much later. → also execution is based on which JS is loaded first
— defer → does not execute it after loading. Executed later, after LCP
— doesn't work as expected. So, defer is used

→ change order the things happen, by putting the scripts at the end of HTML as HTML evaluates from top to bottom

> Optimize Images: lazy loading — loading = "lazy" → but not compatible with all browsers
↳ a native JS script to lazy load.
replace src when in the viewport.
use [data-src] to store the src
↳ this can lazy load any elements with various attributes like src.
Does not care for the element type

> Responsive images: use src-set & sizes on images to dynamically fetch images of different sizes

> Crunching images: remove unused attributes eg. metadata.

dynamically fetch images of appropriate size

> Crunching Images: remove unused attributes ex. metadata.
(tiny png) or build a native one

> Reduce Overhead: HTTP2 — Pros: faster
: reuse connections
Cons: Server setup & compatibility (ex. expressJS no support for HTTP2)
: Requires SSL certificate

> Cache (only on second requests)

> Cache-control
> expires
> etag
} Headers

> Preloading: rel = "preconnect" → to establish earlier connection
rel = "preload"

* Improve CLS: Avoid moving things around and changing the map of the page
↳ get to 0.01
> layout dims → add width & height to the img
if it is lazy loading
↳ aspect-ratio is maintained