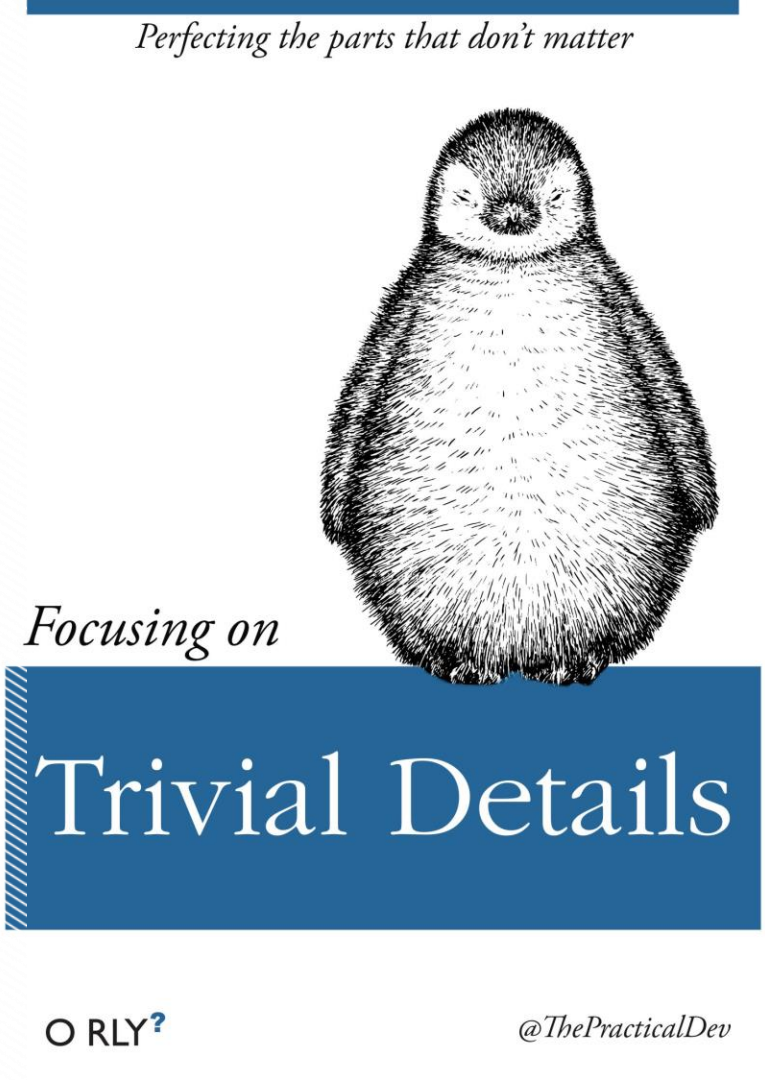
Software Performance

or const length = arr.length;

Frontend Junior Program - 2022



… and they will eat your project on production

"A performance issue won't attack from the front, but from the side. The other two problems you did not even know were there. Because performance is a pack hunter, you see. It uses coordinated attack patterns, and it is out in force today."

But before we move in, we'd need to understand the place of the *software performance* in the product engineering. I promise, it will be surprising and revealing.

Steps of product engineering

The key takeaway here is *the vast majority of the engineering effort will be spent after having a fully functional product* (that cannot be distinguished from the final version at the first sight).

effort spent in engineering

0%

Idea
conception, definition, targeting

Design phase
designing and working on breadboards in short iterations

~5%

breadboard is ready
it is a hand-made, working model without the final design

Prototyping
creating a fully functional prototype with the final design: it looks and works as the real product, but was manufactured manually

~20%

Preparation for production
creating a final product and the production flow

Prototype is ready
the product seems ready from a user's perspective

100%

Production
mass production

Product is ready for mass production

- meets all of the functional requirements
- user support is arranged
- the manufacturing process is established
- tested and certified for **performance**, usability, accessibility, security
- can be manufactured from the available parts (parts do have tolerances as well!)
- conforms standards, safe to use
- works in different environments (temperature, humidity, etc.)
- tolerances meet the specification
- documentation is ready
- …

the most effort will be spent on the *production preparation* by far margin

the product is already "ready" now in this phase

software engineering is no different…

Steps of software engineering

While the flow is very similar, we have notable differences here:

- the product *go live* – depending on the project – is continuous
- we work in short iterations, therefore the border of the prototype ready and product ready phases is usually more blurred.

Software is ready for production

- meets all of the functional requirements
- user support is arranged
- the maintenance process is established
- tested and certified for **performance**, usability, accessibility, security
- conforms regulatory requirements
- works in different environments
- metrics meet the specs
- documentation is ready
- …

Non-functional requirements

0%

Idea
conception, definition, targeting

Design phase
designing (wireframes + design) and working on Proof of Concepts in close iterations

~5%

~20%

Preparation for prod
creating a final product

"Prototype" is ready
the product seems ready from a user's perspective

100%

Production
maintenance

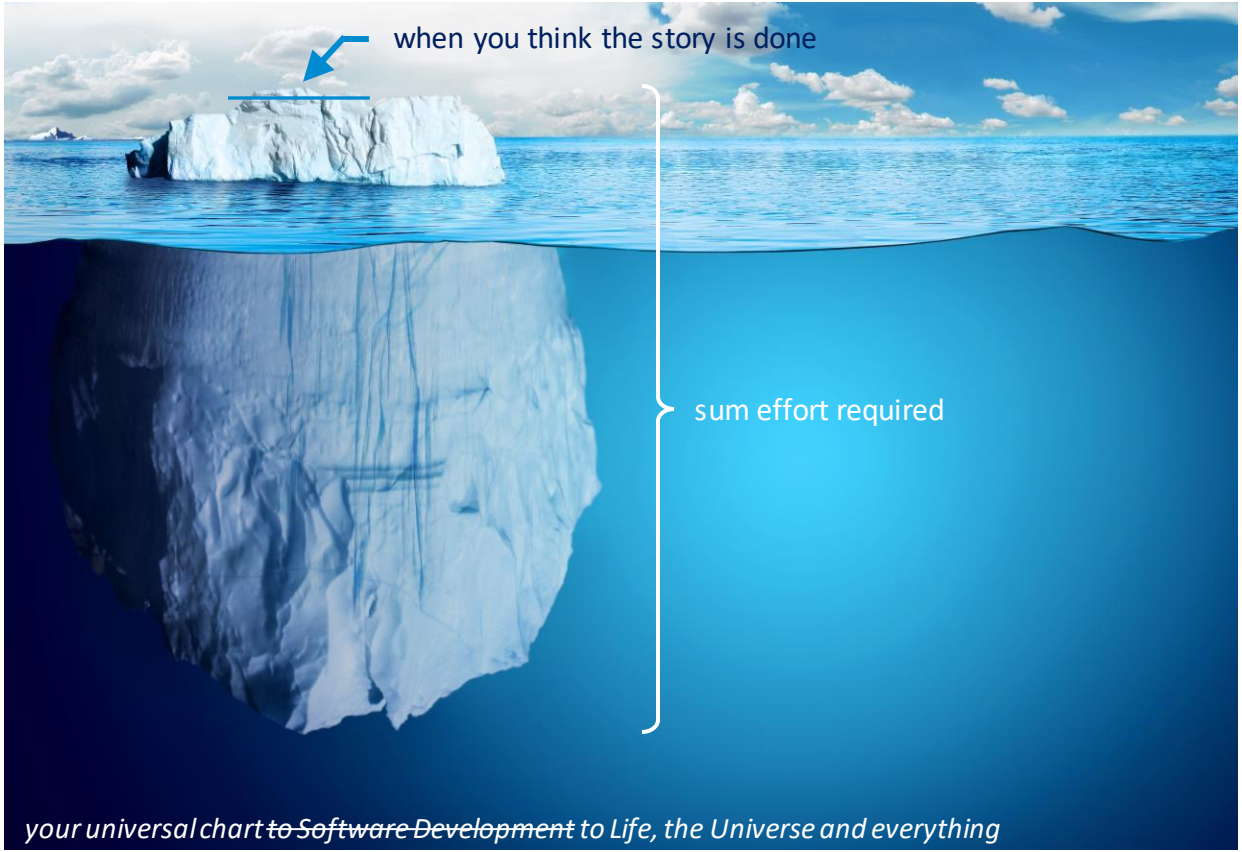
POC is ready
it is working program / component, using quick prototyping tools

"Prototyping"
creating a fully functional software with the final technologies; working on unit tests, automated tests; setting up the environments, deployment processes

Go live preparation
in a greenfield project there is a dedicated go live preparation phase as well: last bugfixes, refinements, production setup

effort on Functional Requirements

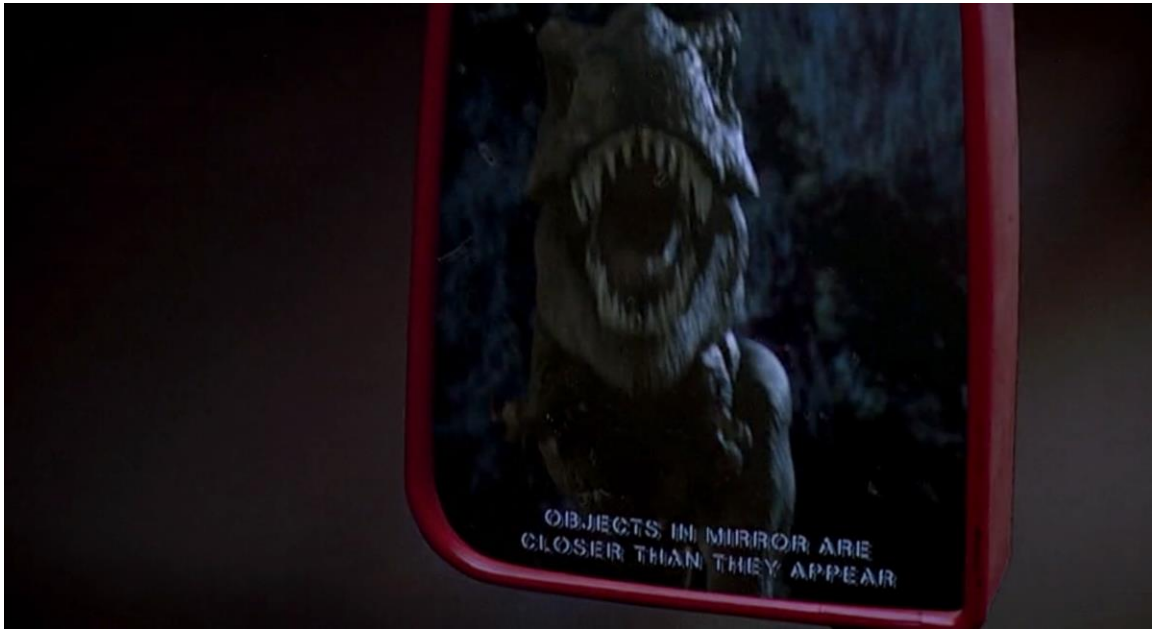
effort on Non-functional Requirements (NFRs)



On *Isla Nublar,* the UAT (User Acceptance Test) and the audits started well before they went through the production preparation phase. While from the Product Owner (*John Hammond*) perspective the product was ready, in fact, it was just a fully functional prototype – *don't let your project become the Isla Nublar*!

top 10 reasons why your story might not be ready to merge

1. You simply missed an AC point
   sadly, that was the main goal
2. You misunderstood an AC point
   and implemented something else
3. You've probably overlooked an edge case
   "Welcome, undefined!"
4. Some unit tests are still missing
   please add 80% coverage to reach the 80%
5. Your branch is up to date with the master
   opps, not really. np, it's just all the commits from the last 2 weeks
6. Some review items are still to do
   such as: *"these 2000 lines seem to be duplicated, could you please take a look?"*
7. Those *TypeErrors* were not part of the story
   if so, please double check with the BA
8. *"Settimeout 100"* is still not the proper solution
   unless you convince the customer
9. Yes, it is easier to go live with the mock BE
   at least it consistently responds with *"John Doe"*
10. We use i18n on the project
    and those translation copies should have been sent to the translation team last week



the deadlines are always closer than they appear

But what if John Hammond pushes the delivery?

We'd need to *listen to the PO and clearly understand the requirements*. If the product is already in production, every incremental changes must be in a production ready state. Bugs, security or performance issues could and will have serious financial consequences.

Many times, however, the POs require quick feedback loop, therefore components must be roll out in an earlier state. There are no contradictions here – the requirements are obviously different!

Non-functional requirements describe implicit requirements that are not included in user stories

- some of these can be (but not necessarily) well defined in a separate document
- not all will be, though – most of them are common sense or even industry standard
- while these are not explicitly stated in every user story, the product must fulfill these

NFRs are a must

major NFRs

usually, 3rd party agencies are responsible to measure and certify against these

- performance
- compatibility
- accessibility (a11y)
- usability
- security

there are 11 letters between the "a" and "y" in the word: "accessibility"

agile team analyzing the status reports

Performance matters…

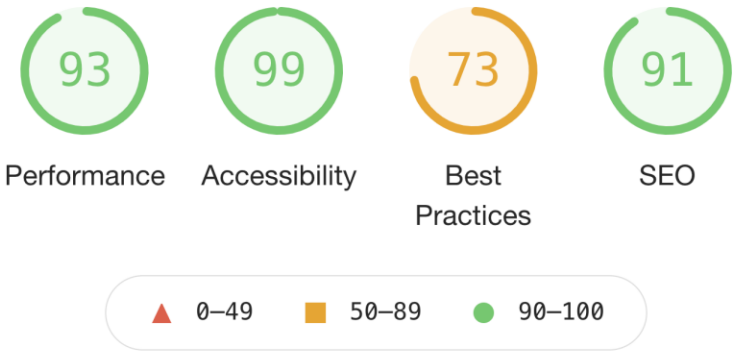
…however, probably not in a way one could expect – let's see what is important and what is not!

don't believe in everything ~~you can see in films: dinosaurs went extinct~~:
the performance score depends

on **your** network speed

it is not a completely useless metric, but you'd need to understand what to measure and how

don't believe in everything ~~you can see in films: dinosaurs went extinct~~ : SEO is pseudoscience!



A lighthouse report could help to identify the targets

Browsers (Chrome, Edge) could advise on useful details on some NFRs. However, the targets are always the official reports of the audits provided by our partners.

if this *blue pill* is not ugly enough,
I don't know what is – design is a
hard job, even for Google

Metrics

- **First Contentful Paint** 0.6 s
  First Contentful Paint marks the time at which the first text or image is painted. Learn more.
- **Time to Interactive** 1.8 s
  Time to interactive is the amount of time it takes for the page to become fully interactive. Learn more.
- **Speed Index** 1.8 s
  Speed Index shows how quickly the contents of a page are visibly populated. Learn more.
- **Total Blocking Time** 110 ms
  Sum of all time periods between FCP and Time to Interactive, when task length exceeded 50ms, expressed in milliseconds. Learn more.
- **Largest Contentful Paint** 0.9 s
  Largest Contentful Paint marks the time at which the largest text or image is painted. Learn more
- **Cumulative Layout Shift** 0.006
  Cumulative Layout Shift measures the movement of visible elements within the viewport. Learn more.

a detailed view of the Lighthouse report

Welcome | Elements | Memory | Console | Sources | Network | Performance | Application | Security | Lighthouse | AdBlock

hu.hotels.com #1 | Screenshots | Memory | Web Vitals

Network

apihotels.net/ (apihotels.net)

0 (bat.bing.com)

search.do (hu.hotels.com)

Frames: 2568.2 ms, 577.6 ms

Interactions: Animation, Animation

Timings: FP, FCP, LCP, DCL, L

Experience: Layout Shift

Main — https://hu.hotels.com/search.do?destination-id=643376&q-check-in=2021-06-03&q-check-out=2021-06-04&q-rooms=1&q-room-0-adults=2&q-room-0-children=0&sort-order=BEST_SELLER

Summary | Bottom-Up | Call Tree | Event Log

Range: 8.10 s – 15.83 s

43 ms	Loading
997 ms	Scripting
157 ms	Rendering
38 ms	Painting
331 ms	System
6169 ms	Idle
7734 ms	**Total**

an even more detailed view in the Performance tab

Performance is a tricky metric

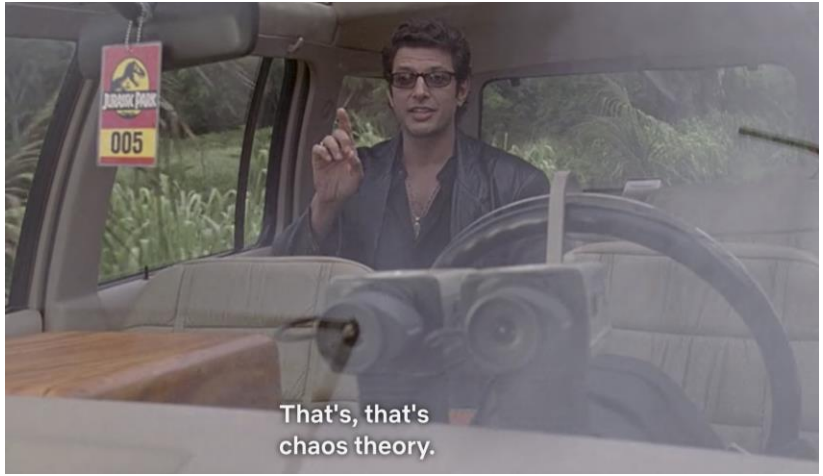
it depends on a lot of properties, and these properties affect each other strongly.

The page load metrics *(FCP, TTI - choose your poison)* depend on

- the asset sizes (image, script, CSS)
- the application / site technology (server-side rendered, SPA)
- the page content
- network speed and type
- device CPU performance
- CDN caches (e.g., Akamai)

in a very convoluted way:

- if the network speed is fast, then the CPU speed could be significant – *if there is a large JavaScript file that needs to be parsed before the page can render / load anything*
- if the network is slow, then the assets sizes could be important – *if the device does have a retina display, and the page relies on images heavily*



complex systems, well, behaves in a complex way

Some properties depend on…

- *the asset sizes* ← devs / CMS system / content manager
- *the application / site technology* ← devs / architect
- *the page content* ← devs / client
- *network speed and type*
- *device CPU performance*
- *CDN caches (e.g., Akamai)*

external factors

While we cannot change the external factors, we can design the application to fulfill the requirements according to the preferences.

To be able to do that, we'd need detailed information on actual usage statistics or about the planned target usage behaviors (in a form of performance requirements).

Without those, it would be a blind flight to design / optimize the performance profile.

performance

built-in

- prototyping, pre-production phases (sometimes even must happen even in PoC)
- general requirements defined in DoD, NFRs, industry standards and common sense
- expected on all competency levels
- should be incorporated in the first version
- component level
- usually correlates with less complexity (well tested, cleaner solutions, limited library utilization, covered edge cases, identified scalability targets, dynamic event handlers, no duct type solutions, etc.)

optimization

- usually happen in production
- very specific requirements, clear and well-defined goals
- complex, costly, requires strong domain knowledge and a high level of expertise
- iterative, sometimes contains preliminary refactoring only
- predominantly architectural level
- introduces more complexity (local and CDN caching, pre-rendering, load balancing, adaptive design, device targeted retina images, etc.)

How to design / optimize the performance profile?

The most sensible way is to identify the bottlenecks and mitigate them, because until a major factor is not eliminated, it is pretty hard to reason about any page performance analysis.

Basically, this approach matches with the optimal development flow: after you have a good understanding and overview, you should go step by step, running targeted analysis after every step.

remember? this is basically a definition of the TDD

we need a careful analysis, a plan and a step-by-step implementation

Setup JavaScript

```
const arr = Array.from({length: 1000000}, (_, i) => i);
let length = arr.length;
```

just don't

enter test case name

```
for (let i = 0; i < arr.length; i++) {
    arr[i];
}
```

DEFER

finished

1841.52 ops/s ± 1.25%
Fastest

enter test case name

```
for (let i = 0; i < length; i++) {
    arr[i];
}
```

DEFER

finished

1810.45 ops/s ± 1.43%
1.69 % slower

jsbench.me: please don't introduce new variables for solving imaginary performance problems

Setup JavaScript

```
const length = 1000000;
```

enter test case name

```
Array.from({length}, (_, i) => i);
```

50 ms

finished

20.7 ops/s ± 0.82%
59.96 % slower

enter test case name

```
Array(length).fill().map((_, i) => i);
```

20 ms

finished

51.68 ops/s ± 2.71%
Fastest

Does it matter?

It depends.

If this code part runs 100 times, could.

If only once, do not bother – while the intention is clear, both are perfect.

However, if you run into the very rare situation when it could matter, you really need to have that user statistics / edge case analysis, because the difference could be huge on different situations.

Also, browsers do have differences in performance, even between versions of the same browser could provide different results.

But in this case, you should consider a completely different approach, not to fine tune a fundamentally wrong solution.

Q&A

edu_hu@epam.com