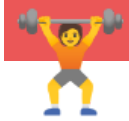


Overcome Technological Change And Pressure (pdf)

1. Introduction

What are the first thoughts that spring to mind when we hear the word "pressure"?

- What is the pressure of?
- Who will be able to withstand this pressure?

These and other basic questions will be raised. In reality, the answers to these questions will allow us to conduct a more concentrated examination of the issue.

Dealing with the Pressure of What

1. Technological change and development pressure
2. Financial issues exert pressure
3. Health issues
4. Artificial Intelligence (AI)-generated pressure

Who will be able to withstand this pressure?

In this post, I will concentrate on folks who work as **software developers**. People working on the **Frontend** in particular.

Of course, within a certain logic, we may build on the preceding factors and human profile. Because there are challenges that might be related to a variety of reasons and positions.

2. Pressure Factors

2.1 The Impact of Technological Change

Even as I type this, new libraries/frameworks are being launched, and announcements about these libraries and technologies are being made someplace. The developer is under pressure.

- Avoid using the new,
- the need to keep up with technology

It drives us to crush more and more every day, to spend more and more time in front of the computer. It pushes us to spend the majority of our time chasing technology.

Abstraction is a notion that helps us to create larger and more complete projects by creating libraries that propel us ahead as developers. So, has recent progress and change placed us developers in an untenable situation?

In reality, while the subject is broad (Backend, DevOps, Frontend, Mobile, Machine Learning, Big Data, etc.), I will explore it through the lens of Frontend.

Beginning in 2011, I began working with **JQuery**. We might create Web Applications using **ExtJS** and Backbone basic libraries during this procedure. **Adobe Flash, Microsoft Silverlight, and JavaFX** were built with plugin logic on the browser for more interactive, rich content displays. **Adobe Flash** and other programs dominated the market. Suddenly, Steve Jobs announced that the iPad/iPhone would no longer support Flash, that HTML5 was on its way, and that he would base his whole business on it, resulting in the phenomenal rise of JS and the demise of **Adobe Flash.**

The notion of utilizing JS on mobile and even developing the entire App using these libraries came up later in the process.

JQuery Mobile, Sencha Touch, and even libraries like **PhoneGap** with HTML rendering capabilities in Native Wrapper developed as Mobile Web rage. Before these were fully established, mobile applications were developed with HTML5 on them, but due to the inability to provide the desired interaction performance, Facebook - Zuckerberg and LinkedIn -CEO jointly stated that the web does not yet provide enough performance to develop mobile applications, and the wind in the mobile field turned to Native App. When Apple joined these techniques and stated that hybrid applications degrade the phone's user interaction quality, everyone switched to native applications.

During this time, **Blackberry** was losing popularity, and **Microsoft** was attempting to compete with **Android/iPhone with Windows Phone** by purchasing Nokia.

For a long time, iPhone was able to stay stable with Objective C in the Native field. The only change was the Storyboard change from Xib. Android Eclipse was abandoned and we started using Android Studio. Fabric was enough to catch bugs and TestFlight was enough for Beta Tests. Windows Phone lost the race to be in the phone world, Blackberry and Nokia were completely erased.

Later, as a result of dynamic languages such as JS,

- On the iOS side, the **Swift** language will develop; on the Android side, the **Kotlin** language will emerge.
- **Xamarin** will be bought by Microsoft in the Cross Platform Native area, leaving Google **Flutter**, - **React Native** will be able to construct Native projects for any environment.
- We'd be able to create desktop apps using JS, **Electron**.

Twitter was the first to use Bootstrap CSS as a foundation for online apps. Infrastructures for MVC (Model View Controller) frameworks were being released one after the other. Adobe Flash would now be replaced by Angular1, Ember, and Knockout. These frameworks would allow people to create rich content Web apps.

Then we found that NodeJS was being utilized not only on the server side, but also in our local development environments.

JS technology suddenly began to release new libraries all over the place. Hundreds of our Component Based JS, CSS, HTML files that we produced could be built and combined into single files using **npm/yarn** package managers, **npm scripts**, **gulp** task runners, **webpack, parcel** combining tools.

Why not add another degree of abstraction? Let us write our code in many languages and let the transpiler turn it into the chosen environment in the background. We discovered this for CSS SASS, PostCSS, LESS, JS TypeScript, and Flow.

Developer editors such as Atom, Visual Studio Code, and other tools such as Vim, SublimeText, and IntelliJ were all supporting these new functionalities. Tools like as Prettier, EsLint, and several test libraries such as Jest, Mocha, Storybook, Cypress, and Enzyme have shown to improve the quality of software development.

Then I question if the MVC (Model-View-Controller) design is appropriate for UI Components, or if it makes more sense to utilize the Flux pattern instead of MVP, MV*, with distinct View and Data layers. Then, maybe, there should be libraries that offer View construction.

React, Vue libraries arose. DOM rendering was carried out on **Virtual DOM**. Shadow DOM existed; **Polymer** used it, but it was not as popular as **React and Vue**.

Until now, each library had its own data libraries for accessing model data in line with the Flux Pattern: **Redux, MobX, RxJS, NgRx, VueRx.**

Then, as the components became more complex, data transit mechanisms like as High Order Components, ContextProvider, and so on were discovered, but these could not connect the data as flexibly as required. Libraries need further structural adjustments. React, the ClassComponent FunctionComponent and Hook library, and the Vue Composition API, rather than Vue UI component structure extend or mixin, making binding and state transfer events as flexible as feasible.

Meanwhile, a library known as **Svelte** had formed. **VirtualDOM** said there was no need. At the same time, a separate library for React State management called **Recoil.JS** was forming. New State management solutions such as Jotai, Signal, and Zustand continue to develop. Instead of using RSC (React Server Component) to manage state, certain components are designed to be very near to the data and do the rendering process on the server.

If these View libraries existed, many UI Layout libraries, UI component libraries, Chart libraries, **React, Angular, Vue libraries, SemanticUI, KendoUI, AtlassionUI Kit**, **Rechart, etc...** many component libraries that implement **Metarial Design** could be created on them.

With module loading (require.JS), async libraries, FP (functional programming) libraries, and AJAX/http libraries (Axios) becoming standardized at least in ES6, ES7, ES8, and ESNext, things are becoming increasingly hard for us frontend devs.

For the request, we used normal **JSON and RESTful**, however this was insufficient on the backend. Rest should be abstracted, and a method should be established to provide the JSON you want in accordance with your dynamic queries, to the extent that your schema permits based on the intended request. Hop appeared **GraphQL**. In databases, we may use dynamic queries instead of conventional queries, much like we can with **ES (ElasticSearch)**.

Meanwhile, NodeJS libraries began to gain a slice of the backend server market from Ruby on Rails, Python,.Net, and Servlet by providing Rest API with Express.JS and allowing you to design apps that can establish backend APIs.

There was a blog and a CMS (Content Management System) sector that created static pages. **Wordpress, Joomla, Drupal**, and other similar platforms use the **LAMP Stack (Linux, Apache, MySQL, PHP)** instead of a headless Node.On **Jekyll, Hugo, Nuxt, Next, Gatsby,** and Server Side Rendering frameworks, js CMS Ghost and **JAM Stack** (JavaScript, APIs, and Markup) evolved. Now, JS Backend is becoming more popular.

We will enable developers to utilize Angular and Ember as a Framework for Routing, SEO, and other purposes, similar to how React, Vue, and Svelte are View libraries.

- **React → Next, Remix, Gatsby**
- **Vue → Nuxt,**
- Frameworks such as **Svelte → Sapper** started to be created.

Of course, BaaS and MBaaS methods evolved if there were a set of ready-made tools in the backend of the apps, if we could conduct the deployment directly. We could now easily create backends using configuration and very little code. First, Google Firebase was released.

- Authentication,
- Authorization,

- DB, Log,
- Analytics,
- Notification Sending

as easy as possible for the Frontend developer to control and manage. In order

- Amazon → Amplify(**AppSync, Cognito, S3, DynamoDB, SNS..**)**
- Microsoft → Azura Static Web Apps.**
- **Heroku** other as PaaS
- AWS, Google, MS are easier to use, big companies caught a gap in this area and started to fill it immediately.
- In Static Page Deploy, **Netflix, Vercel** became one of the hosting services that suddenly became popular.

Recently, we began to hear **Deno** from Node developers, which they created based on their regrets when creating Node in compliance with today's features. Microservices began to influence the frontend, and now Micro Frontends are being discussed.

Conclusion

others of the libraries I mentioned above are in **bold italic**, others are new, and some are rarely or never utilized.

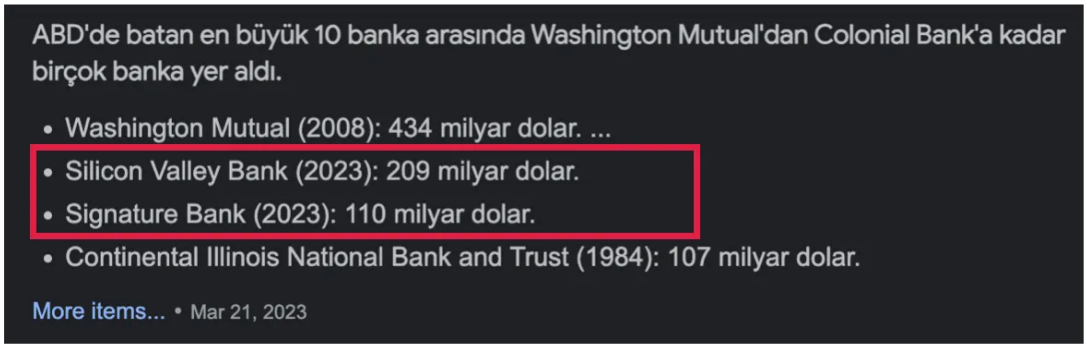
Technology and library production are currently moving at breakneck pace. Every day, life becomes more complicated because we are the ones who utilize technology, not those who create it.

Because they work on low-level standards, I believe folks who create technology are dealing with more algorithms and library logics. Even if we, as developers who use this technology, wanted to prevent this transition, it does not appear to be possible.

All firms are striving hard to catch up with these innovations as the industry rises on this excitement. Finally, we all find ourselves drawn into this whirlwind. People who do not understand these technologies cannot find jobs, and those who do are expected to work to this standard. Every day, the climate becomes hotter and more difficult for developers.

2.2 Pressure from the Financial Recession

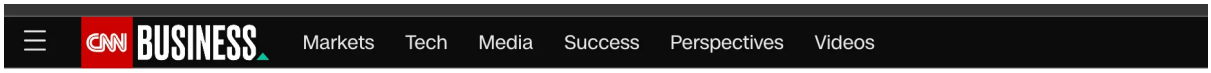
In 2023, the financial industry is not as supportive to startups and software firms as it was in 2022. During this time, Silicon Valley Bank, where many software businesses stored their funds, declared bankruptcy. According to March 21, two banks in America will fail in 2023, totalling over $ 320 billion.



- Silicon Valley Bank (2023) - $209 billion
- Signature Bank (2023) - $110 billion

Although the American government is helping to avoid this, the software industry is working hard to keep the sector's negative aspects from worsening.

Every day, large corporations get laid off. For example, today (19.04.2023), the company formerly known as Facebook, now renamed as Meta, stated that it will lay off more employees. The severity of the problem may be seen in the article below (layoffs Meta).



New York (CNN) — Facebook parent Meta on Wednesday began its latest round of layoffs focusing on technical workers, who are often thought of as more immune to job cuts in Silicon Valley.

Meta spokesperson Nkechi Nneji confirmed to CNN that some previously announced layoffs were taking place Wednesday, and pointed to CEO Mark Zuckerberg's March announcement that the company would cut another 10,000 employees in the coming months.

Zuckerberg's notice said that restructurings and layoffs in Meta's tech groups would take place in April. Among those affected by Wednesday's layoffs were members of the company's sustainability, well-being, user experience, news feed and messaging teams, according to public LinkedIn posts.

Meta reportedly told North American employees to work from home on Wednesday in anticipation of the layoffs. (CNN has not independently confirmed that.)

Members of Meta's recruiting team were notified of additional layoffs last month, and cuts to the company's business groups are expected to take place in late May.

The 10,000 job reductions mark the second recent round of significant job cuts at Meta. The company said in November that it was eliminating approximately 13% of its workforce, or 11,000 jobs, in the single largest round of cuts in its history.

You can check the links below to understand the magnitude of this situation.

- https://layoffs.fyi/
- A comprehensive list of 2023 tech layoffs
- Tech layoffs in 2023: A timeline
- 2023 Layoff Tracker: Latest Meta Cuts Could Hit 4,000

According to the figures I got from https://layoffs.fyi/. As it can be seen, there is almost twice as many layoffs in 2022, this year in 2023, this figure reached 171K.

The aforementioned cutbacks are putting enormous strain and pressure on the market's software engineers. Thousands of job seekers have lost their employment,

and many software engineers are concerned about when they will lose their positions.

This implies that employees in the software business are under increased pressure to perform and be more productive on the job.

This is a circumstance that worsens developers' mental and physical health. Because software development is already a difficult job, we keep a lot of business logic, language logic, and system logic in mind, and on the other hand, it is a period when we are expected to increase needs, rapid development requests, and efficiency, which are unbalanced by the financial recession situation in the market that develops outside of us.

Looking for flaws with these structures and raising productivity via them will not fix the problem if the firm is established on the incorrect structures from the start, if it is run on dreams and inflated numbers rather than a solid revenue structure.

These constructions must be demolished and replaced with sturdy ones. This is exactly what startups and large corporations face. In response to these realities, businesses may opt to close down specific initiatives or enterprises. In other words, they embrace flawed structures from the start and attempt to forge new paths.

2.3 Health Issues and the Pressures They Cause

When I checked the internet, I discovered that when developers work under high levels of stress, they might have a variety of health issues, and when these health issues are combined with the current intensity and time restrictions, the pressure on developers grows tremendously.

What kinds of health issues are we dealing with?

- **Mental health:** Workplace stress may cause anxiety, despair, and burnout, affecting a developer's general mental health. One of them is impostor syndrome. This is when you assume that, despite your success, it is a fluke, and you feel uneducated and worthless in the face of changing and developing technologies.
- **Physical health problems:** Long periods of computer use can result in a variety of physical health issues such as neck discomfort, back pain, carpal tunnel syndrome, and eye strain.
- **Poor sleep quality:** Working in a stressful atmosphere might have an impact on a developer's sleep quality, resulting in insomnia or other sleep problems.

- **Cardiovascular problems:** Chronic stress can raise the risk of cardiovascular disorders such as hypertension, heart disease, and stroke.
- **Weakened immune system:** Because stress weakens the immune system, individuals are more vulnerable to illnesses and disorders.

2.4 Artificial Intelligence (AI) Pressures

With the advent of ChatGPT 3 and eventually ChatGPT 4, artificial intelligence has drastically decreased even our Google searches, and we are now attempting to answer our curiosities or difficulties with ChatGPT.

We see and utilize artificial intelligence products that have reached a certain level of maturity everywhere, from education to code development, from content creation to answering queries, from translation to image creation.

- ChatGPT
- Copilot
- DeepL
- DALL-E

This is a major issue for the occupations of employees who execute a variety of intermediate activities. It compels us to alter our present business practices and take on new roles.

This is because most businesses would rather deal with cheaper and faster robots and AI (artificial intelligence) goods than deal with people for automated employment. Many people's existing jobs will be terminated as a result of this.

In the circumstances I described above, it is undeniable that software engineers place a great deal of pressure on those who work in this industry.

3. How Do We Deal With Change and Stress?

> Excerpt from Jim Collins From Good to Great Company
> When I was presenting this book at the conference, one person

> stood up and asked, **"Do your findings fit the new economy?"**

> And he said, "Yes, the world continues to change and will continue to change, but that doesn't mean that we should stop **looking for eternal principles**. I like to think that while engineering is constantly changing, the laws of physics are constantly doing what we do, eternal principles, something that will always be permanent no matter how much the world changes, I like to think of it as looking for the physics of perfect companies. Yes, the practice in a certain field will change (engineering) but the immutable laws of human performance (physics) will always remain the same.

We may apply the above principle to this article perfectly. Yes, the technology and environment around us are changing at a rapid pace, but by adopting basic principles and habits, we may adjust to these changes swiftly or continue on our course with little impact.

I'll try to describe the issue here by drawing conclusions from my own experiences or what I observe around me.

3.1 English

English is one of the most significant subjects; there are many resources on technology available in other countries; in order to profit from these resources, you must comprehend the blog post, podcast, or video you watch. When required, you must be able to ask inquiries and speak with others.

3.2 Mental Model

The fact that you comprehend anything in Turkish does not imply that you understand the concepts and issues, because your brain need real-world examples to absorb such notions.

You only need a setting in which to experiment with sandbox and playground investigations. Attempting to better comprehend the concepts through tiny independent examples and projects. Mental representations of these must be developed.

3.3 Real Learning (Deep Learning)

We have learnt a lot of things after we graduate from university, for example.

- Algorithm
- Database
- Network,
- Linear Algebra,
- Logic,
- Operating System,
- Data Structures,
- Object Oriented Programming,
- Programming Languages ,
- Computer Graphics,
- Design Patterns
- etc.

We cannot truly understand these concepts no matter how much we engage in these courses until we use them in real-world initiatives within the team. In this sense, I believe the following remark is essential.

> Real learning is not memorizing knowledge. It's understanding and knowing how to use and find knowledge. Learning is what you do with knowledge, how you integrate it, how you talk to your family, friends and classmates about it. That's what learning is. (Dennis Littky)

Memorizing information;

- doesn't get you anywhere,

the information you receive in the courses

- unless you go out and use it,
- if you are not testing it in your projects,

It is useless knowledge, and because it has no foundation, you do not know when to use it and in what situation.

- Information that you do not know what and where to use is not learned. Then experience the topics in the trainings we receive in real projects in order to lay solid foundations.

What I mean by fundamental subjects;

- Mathematics, Physics
- Algorithm, Data Structures
- Programming Languages, Paradigms
- Basic concepts in the field you want to specialize in

You must be able to analyze a variety of problems rationally and come up with innovative solutions to them in order for us to eventually participate in the industry. To build these muscles, that is, your problem-solving muscles, you must focus on them at all times, reinforce the foundations, and practice.

3.4 Working with Good Teams

The point where I consider myself lucky is that I had the opportunity to work in good teams in Turkey. This is a very important issue. Especially when you are a new graduate, it is very important to have elders who guide you or to have challenging jobs that challenge you in projects. These jobs will cook and develop you.

3.5 Create Your Own Methods

Try to tackle difficulties on your own, and come up with your own problem-solving style. Problem resolution should not be done by copying and pasting from stack overflow or blog posts; you can do it if you are short on time, but you should also strive to investigate and comprehend the underlying technology, topic, and concepts.

Your ability to react quickly to your surroundings will offer you an edge for a while, but after a while (many years), you may feel empty.

3.6 General Knowledge with a Specialty

Changing the sector/domain on a regular basis and constantly performing things at the same level, To use an example from my own experience

- Military sector,

- The game,
- Social Media,
- Telecommunications,
- Public, Bank, Insurance,
- Decision Support Systems,
- R&D (NLP, AI),
- I have worked in many areas including Observability.

My good fortune has been that no matter how many various fields I have worked in, I have always worked on UI and Application development. In addition, I seldom join in initiatives from the start, that is, at zero time.

You have no chance of learning these infrastructures if you stay behind the abstraction developed by the Infra (infrastructure), (SRE & DevOps) team and are constantly coding Business/Domain behind these abstractions, if you are making definitions on an engine (game engine, rule engine) with Workflow and Rule script.

Despite the fact that organizations seek to produce quicker and less error-prone code by abstracting it, always ask your firm to explain the infrastructures to you, and even request to develop code on these infrastructures on occasion.

3.7 Fundamental Operating Principles

- Concentrate on a single issue and attempt to study all the subtleties and delve deeper into it.
- Make an effort to study the standards and fundamental mechanics.
- I discussed changing technology before, yet there are some UI fundamentals that remain constant. If you can master the fundamental working principles and mechanics of your profession, you will find it quite easy to adapt to any new technology.
- For example, the following UI-related issues are unrelated to developing technologies. Learn the **similar unchangeable fundamental commonalities** in every business.
    - Standards (HTTP, HTML, CSS, ... etc)
    - UI working Mechanics (Get screen data, Validate, Save, etc.)
    - Color Matching, Layout structures etc..

- Understanding the requirements to work in different environments (Desktop, Mobile, Tablet ...)
- Design Patterns

3.8 Understanding the Cause and Effect Relationship as well as its History

Try to grasp the technology's history to learn more about it. Prior to the invention of this technology

- which technologies came out in order
- which library, which feature, why it came out,
- which problem it came out to meet.

I stress this in my essays, for example, and I try to remember it by forming links in this way. There is a historical link. In summation, if you can establish the Cause - Effect link, if you understand why these changes were made, you will be able to adapt to technology faster and embrace it sooner.

3.9 Understanding Artificial Intelligence (AI)

As you can see from the examples above, artificial intelligence will have a significant impact on our lives. Instead of opposing it, we could learn to harness artificial intelligence and adopt a different production structure, be able to do other occupations, and speed up our work.

Attempt to accelerate and certify your work with new tools by incorporating developing technology into your own work.