

# PESPA On The Edge

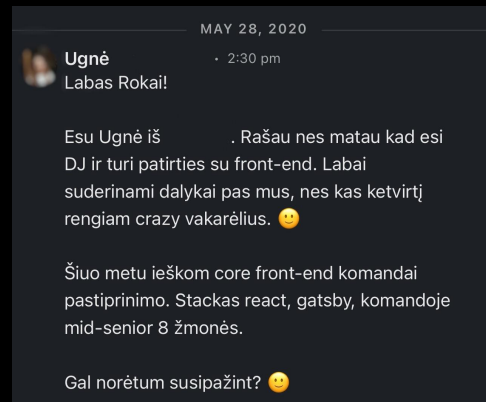
Rokas Muningis

23 January 26, Vilnius

# Me

I've been working as Frontend Engineer for the past 6 years, of which past half year I've been working in InsurTech field with Sapiens where I tech-lead small team of 3 engineers.

And just like everyone else, I've received messages from recruiters on LinkedIn for something else, however, it's not always *that bad*.



# Edge

Child of CDN and Load Balancer or as per CloudFlare's explanation [1]:

*Edge computing is a networking philosophy focused on bringing computing as close to the source of data as possible in order to reduce latency and bandwidth use.*

# PESPA

Progressively

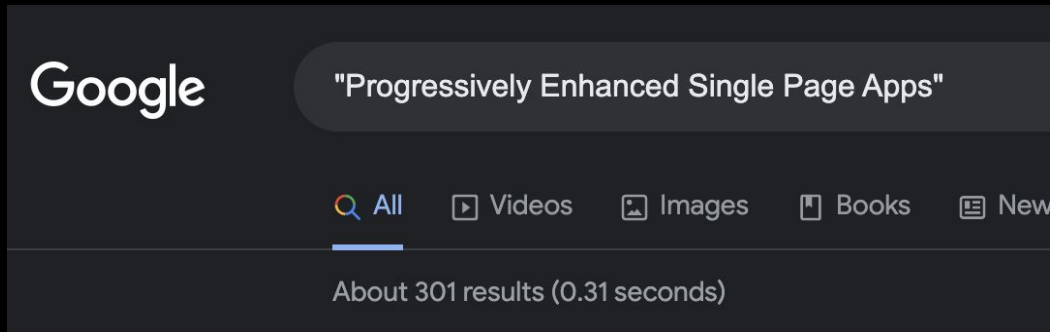
Enhanced

Single

Page

Application

# Progressively Enhanced Single Page Apps



Kent C. Dodds  
@kentcdodds

I'm excited to introduce the web's next transformation:

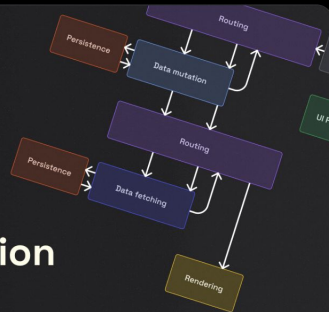
✨ Progressively Enhanced Single Page Apps ✨

This transition has already started. Let's take a bit to talk about the history of architecture for web apps and explore its future as well.



EpicWeb.Dev

## The Web's Next Transition



epicweb.dev

### The Web's Next Transition

Web is made up of technologies that got started over 25 years ago. Now, we are transitioning to a new and improved architecture for building web applications.

6:18 PM · Oct 11, 2022

# Page Applications

MPA - **M**ulti **P**age **A**pps

PEMPA - **P**rogressively **E**nhanced **M**ulti **P**age **A**pps

SPA - **S**ingle **P**age **A**pps

Honorable Mention - **P**rogressive **W**eb **A**pps

# Brief differences

MPA - No javascript, client handled only link clicks and form submissions

PEMPA - ***Optional javascript*** to enhance experience, but most things still handled by server

SPA - ***Javascript required***, as routing, data-loading, rendering, you name it are handled by client.



## Not enough memory to open this page

Try closing other tabs or programs to free up memory.

Error code: Out of Memory

[Learn more](#)

[Send feedback](#)



# Brief differences

MPA - No javascript, client handled only link clicks and form submissions

PEMPA - ***Optional javascript*** to enhance experience, but most things still handled by server

SPA - ***Javascript required***, as routing, data-loading, rendering, you name it are handled by client.

<https://my.awesome.website> · [Translate this page](#) ⋮

[my.awesome.website](#)

# Server Side Rendering

- Next.JS
- Nuxt
- Angular Universe
- Gatsby
- Remix
- SvelteKit

# Progressive Enhancement

## Progressive Enhancement and the Future of Web Design

Published: March 21, 2003

Editor's Note: For more on Progressive Enhancement as a web design strategy and its application and adoption, read [Progressive Enhancement on Wikipedia](#).

Reprinted from Webmonkey  
by Steve Champeon, Chief Technical Officer

# Progressive Enhancement



```
1 <details>
2   <summary>Title of accordion</summary>
3   <article>Put contents of your accordion here!</article>
4 </detail>
5
```

# Progressive Enhancement



```
1 details[open] {  
2   box-shadow: 1px 1px 0 0 rgba(0, 0, 0, .5)  
3 }
```

# Progressively Enhanced Apps

1. Works fine without Javascript on client
2. Works fine without CSS on client

# Brief differences

MPA - No javascript, client handled only link clicks and form submissions

PEMPA - ***Optional javascript*** to enhance experience, but most things still handled by server

SPA - ***Javascript required***, as routing, data-loading, rendering, you name it are handled by client.

PESPA - ***Optional javascript***, and most things handled both on client and server while running same code.



*Not the technology, but rather way to develop websites*

# Progressively Enhanced Apps

1. Use what is provided by spec. (e.g. `<details />`)
2. Need to submit the data? Use `<form />`
3. Need to cache the data? Use Cache headers

# Sacrifices and Compromises

In engineering, we're quite often in position, where have sacrifice something for something else.

One, which has been an issue for years, and to some extent is still an issue to this day - SPA and SEO.

At some point we started using SPAs even for SEO-dependant public-facing websites, yet crawlers were unable to crawl, index and rank them - thus hurting SEO rankings. Server Side Rendering was middle point between SPAs and PESPAs.

## Brief Summary of Web Development History

Born in 1990 November, first webpages had no javascript, and only `<a>` tags for navigation, and `<form>` for submissions. These, now are known as Multi Page Applications.

In 1995 December 4th, engineering world saw worst disaster of all or greatest gift one could imagine (it's still huge ongoing debate) known as JavaScript. Back then, it would give you some trails around cursor, hover effects or snow. Quite shortly, in 1999 XMLHttpRequest saw shine of light, which allowed loading or submit data without reloading the page. Though, back then it was used quite conservatively and remained known as Progressively Enchanted Multi Page Applications.

## Brief Summary of Web Development History

At some point around 20xx, we ditched server side rendered HTML, and did most of things like rendering, data fetching, persistence on Client - loading page once, was enough to access whole application. And what a blessing it was, or so we thought. Removing any kind of server-side rendering, for few years, completely damaged SEOs but unrelated to that fact, these websites there known as Single Page Applications

## Brief Summary of Web Development History

Eventually, we started rendering our javascript's templates in backend. Some saw it as progress, others who could not use internet while talking on phone *with a wire* saw it as a glimpse to the past, yet it was not perfect and coincidentally it had quite an unpleasant acronym - USSR (Universal Server Side Rendering).

However, as time past by, some smart people realised, that we might not need Javascript on client side at all, and hence, just like it's ancestor PEMPA, Progressively Enchanted Single Page Application was born.

## Fun fact time!

Even though XMLHttpRequest was initially released in 1999, it was not until 2016 until it was standardized!

# Popular implementations

- Remix
- Next.JS
- SvelteKit

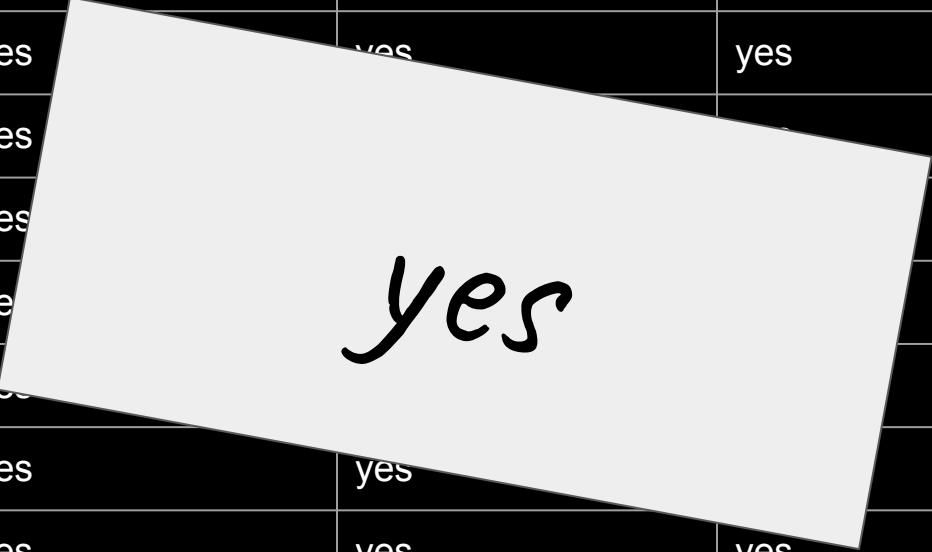


# Comparison

	SvelteKit	Remix	Next.js
Type	Compiler*	Compiler*	Framework
Based on	Svelte	react-router**	react
Routing	File-based, Nested	File-based, Nested	File-based, Nested (beta)
Pre-rendering	Yes	No***	Yes

# Supported run-times

	SvelteKit	Remix	Next.JS
node.js	yes	yes	yes
deno	yes	yes	yes
Cloudflare Pages	yes	yes	yes
Cloudflare Workers	yes	yes	yes
Netlify	yes	yes	yes
Vercel	yes	yes	yes
AWS Lambda	yes	yes	yes



*(DEMO) PESPA on the Edge*

Q&A

## Links

[1] [What is a CDN edge server? | Cloudflare](#)

[2] [The Web's Next Transition | Epic Web Dev by Kent C. Dodds](#)

[3] [Progressive Enhancement](#)