



the rise of the meta framework









Daniel Roe

daniel@roe.dev

 @danielcroe

 @daniel@roe.dev

hello 🖐️

 nuxt.com •  nitro.unjs.io •  page-speed.dev •  regexp.dev
 fontaine.sh •  elk.zone •  firstcommit.is •  react-to-nuxt.com

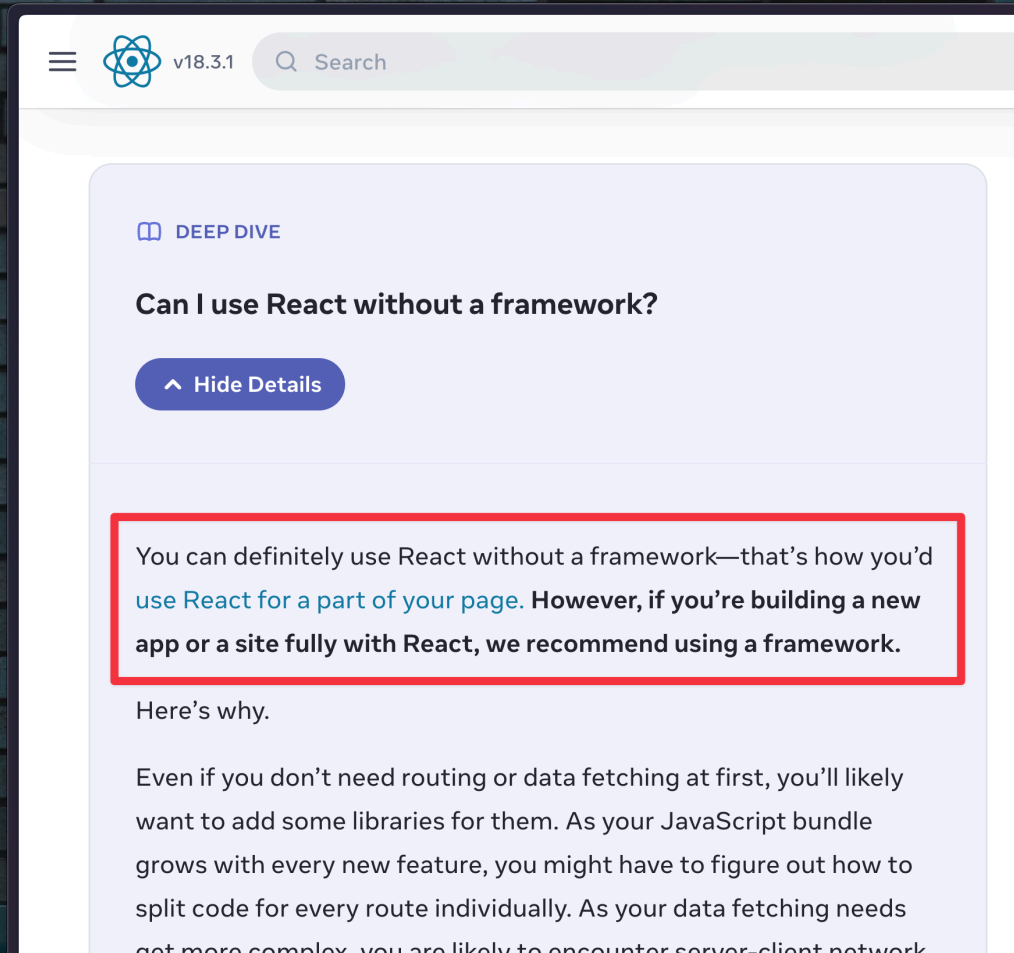





why use a meta-framework?



why use a meta-framework?



The screenshot shows the React v18.3.1 documentation page. At the top, there is a navigation bar with the React logo, the version number 'v18.3.1', and a search bar. Below the navigation bar, the page title 'Can I use React without a framework?' is displayed. A 'DEEP DIVE' icon is visible to the left of the title. A blue button labeled 'Hide Details' is positioned below the title. The main content area contains a paragraph that is highlighted with a red border. The paragraph states: 'You can definitely use React without a framework—that's how you'd use React for a part of your page. However, if you're building a new app or a site fully with React, we recommend using a framework.' Below this paragraph, the text 'Here's why.' is followed by a paragraph explaining the reasons for this recommendation.

☰  v18.3.1

DEEP DIVE

Can I use React without a framework?

[^ Hide Details](#)

You can definitely use React without a framework—that's how you'd use React for a part of your page. However, if you're building a new app or a site fully with React, we recommend using a framework.

Here's why.

Even if you don't need routing or data fetching at first, you'll likely want to add some libraries for them. As your JavaScript bundle grows with every new feature, you might have to figure out how to split code for every route individually. As your data fetching needs get more complex, you are likely to encounter server-client network

why use a meta-framework?

Use a framework to build React Native apps

June 25, 2024 · 4 min read



Nicola Corti

Software Engineer at Meta

At [React Conf](#), we updated our guidance on the best tool to get started building React Native apps: a **React Native framework** - a toolbox with all the necessary APIs to let you build production-ready apps.

Using React Native frameworks, such as Expo, is now the **recommended** approach to create new apps.

In this blogpost we want to walk you through what they are in detail and what they mean for you as a React Native developer starting a new project.

What is a React Native framework?

why use a meta-framework?

The image shows a screenshot of the Vue.js documentation website. The background is a dark brick wall. The website interface is dark-themed. At the top, there is a navigation bar with the Vue.js logo, a search bar, and links for Docs, API, Playground, Ecosystem, and About. Below the navigation bar, there is a sidebar on the left with a table of contents. The main content area on the right is titled 'Fullstack / SSR' and contains text explaining the benefits of SSR. A red box highlights a specific paragraph in the text.

Vue.js Search K

Docs API Playground Ecosystem About

API Preference

Options Composition

Getting Started

- Introduction
- Quick Start

Essentials

- Creating an Application
- Template Syntax
- Reactivity Fundamentals
- Computed Properties
- Class and Style Bindings
- Conditional Rendering
- List Rendering
- Event Handling
- Form Input Bindings
- Lifecycle Hooks
- Watchers
- Template Refs
- Components Basics

Fullstack / SSR

Pure client-side SPAs are problematic when the app is served from a server. This is because the browser will receive a large amount of content to wait until the JavaScript is loaded before rendering anything.

Vue provides first-class APIs to "render" a Vue app into HTML. This allows the server to send back already-rendered HTML, so the content immediately while the JavaScript is being downloaded. This is called "hydrate" the application on the client side to make it interactive. This is called **Side Rendering (SSR)** and it greatly improves Core Web Vitals like **Contentful Paint (LCP)**.

There are higher-level Vue-based frameworks built on top of Vue, such as **Nuxt**, which allow you to develop a fullstack application with ease.

JAMStack / SSG





EXPRESS EXPRESS EXPRESS
EXPRESS EXPRESS EXPRESS
EXPRESS EXPRESS EXPRESS
EXPRESS EXPRESS EXPRESS

EXPRESS EXPRESS EXPRESS
EXPRESS EXPRESS EXPRESS
EXPRESS EXPRESS EXPRESS
EXPRESS EXPRESS EXPRESS



6th SMASH YEAR!

PRETTY WOMAN
THE MUSICAL

SOME FOLK ARE AWAY

2011

DO YOU WANT TO BUY A HOUSE? WITH TRYST

2011

DO YOU WANT TO BUY A HOUSE? WITH TRYST

DO YOU WANT TO BUY A HOUSE? WITH TRYST

MARRIOTT MARQUIS
AIRFRANCE
FRANCE IS IN THE AIR
PUTTING YOU AWAY

AN INTERACTIVELY
INTERACTIVE CANDY
WONDERLAND
CANDYTOPAL.COM

Mobile
RECORDED

Give real.

AMERICAN EAGLE OUTFITTERS

wafels & dinges

BINGO

in the beginning...

HTML



django




spring

Express





A single, smooth, white egg is centered in the frame against a light, neutral background. The egg is oriented vertically. The word "capability" is printed in a bold, black, sans-serif font across the middle of the egg's surface. The lighting is soft and even, highlighting the egg's smooth texture and casting a subtle shadow on the surface it rests upon.

capability

BILLY

... and necessity



keepers of the config

- ✦ developer experience
- ✦ server-side rendering

NEXT.js

ember

 Angular

METEOR

 Nuxt

CLOUDFLARE RADAR

year in review 2023

Website Technologies

Most popular technologies across top websites

Modern web sites are complex productions, relying on a mix of frameworks, platforms, services, and more. Using Radar's [URL Scanner](#), we analyzed web sites associated with the [top 5000 domains](#) to identify the most popular technologies in use across a number of categories.

Note that the data was collected across a subset of the top 5000 domains, as not all have active websites.

Advertising

Analytics

A/B testing

CMS

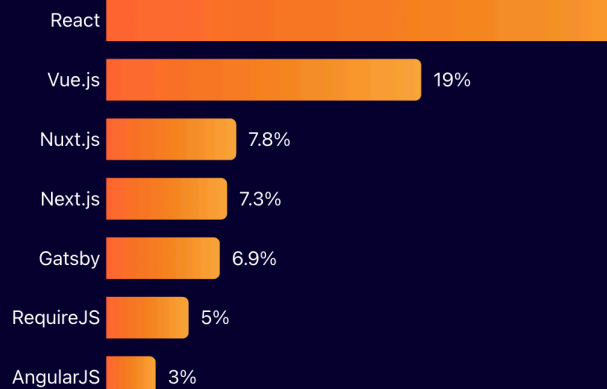
Cookie compliance

JavaScript frameworks

JavaScript libraries

Marketing automation

PaaS



analog



2

astro



21

deno fresh



3

eleventy



5

gatsby



10

next.js



21

nuxt



66

quasar



27

qwik



8

remix



2

solid start



4

sveltekit



19



roe.dev/feedback

52

- ◆ **Preference for Nuxt over Next**: Many users seem to prefer Nuxt over Next, calling it "king" and praising its simplicity and configurability, as well as the speed of development it facilitates.
- ◆ **Appreciation for Meta Frameworks**: There's a general positive feeling towards meta frameworks with some users even calling them the "new" frameworks, noting their capability, ease of use, and the convenience they offer in terms of developing apps.
- ◆ **Concerns about Complexity and Control**: Some users expressed concerns about the added complexity layer of meta frameworks and the potential lack of control, citing issues with server-side rendering (SSR) and error handling.
- ◆ **Suggestions for Improvement**: Users highlighted areas for improvement, such as better documentation for newcomers, more flexibility in choice of language (one user still preferred Java) and a need for less tech debts on handover.
- ◆ **Critiques and Praise for Specific Features**: Users gave mixed feedback on specific features such as documentation quality, default frameworks, speed of development and out-of-the-box functionality, while some mentioned a lack of confidence in performance and scalability in production.

In summary, the feedback was generally positive, with many users expressing proficient use and preference for Nuxt as a front-end meta framework; however, they also aired concerns about added complexities and control, indicating room for improvements in these areas.

deeper bundler integrations

unlocking features

- ✦ multi-app and app federation
- ✦ partial hydration and resumability
- ✦ ... and more

deeper bundler integrations

powered by faster bundlers

✦ rspack



✦ rolldown



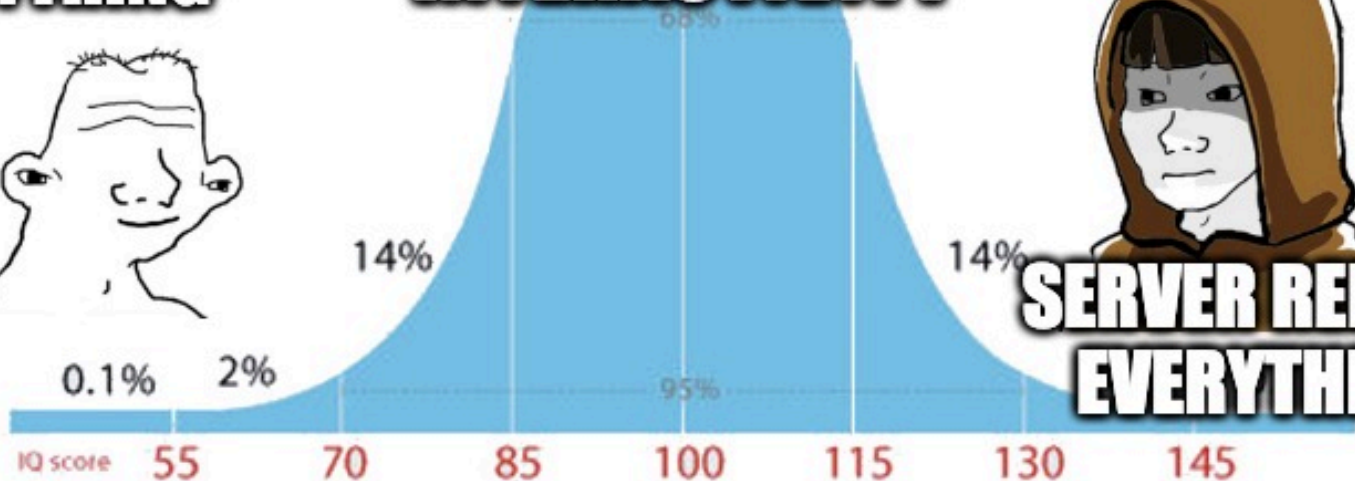
**SERVER RENDER
EVERYTHING**



**CLIENT-SIDE
INTERACTIVITY**



**SERVER RENDER
EVERYTHING**



tighter vertical integrations

deployment platforms



tighter vertical integrations

platform primitives

- ✦ key-value storage
- ✦ image optimisation
- ✦ direct database access
- ✦ websockets/streaming
- ✦ LLM integrations

tighter vertical integrations

two approaches

1. platform-centric
2. framework-centric

NEXT.js

 **Nuxt**

ROUTING

- ✦ vue-router
- ✦ universal routing

DEPLOY

- ✦ alwaysdata
- ✦ aws lambda
- ✦ aws amplify
- ✦ azure
- ✦ cleavr
- ✦ cloudflare
- ✦ deno deploy
- ✦ digitalocean
- ✦ edgio
- ✦ firebase
- ✦ flightcontrol
- ✦ github pages
- ✦ heroku
- ✦ iis

- ✦ koyeb

- ✦ netlify
- ✦ platform.sh
- ✦ render.com
- ✦ stormkit
- ✦ vercel
- ✦ zeabur

TESTING

- ✦ jest
- ✦ vitest
- ✦ playwright
- ✦ cypress
- ✦ cucumber

STORIES

- ✦ storybook
- ✦ histoire

KV AND CACHE

- ✦ azure
- ✦ browser
- ✦ capacitor
- ✦ preferencestore
- ✦ cloudinary
- ✦ filesystem
- ✦ github
- ✦ http
- ✦ lru cache
- ✦ memory
- ✦ mongodb
- ✦ netlify blobs
- ✦ overlay
- ✦ planetscale

- ✦ redis

- ✦ vercel kv

DATABASES

- ✦ bun sqlite
- ✦ cloudflare d1
- ✦ libsql
- ✦ sqlite
- ✦ turso
- ✦ vercel

IMAGES

- ✦ aliyun
- ✦ aws amplify
- ✦ bunny
- ✦ caisy
- ✦ cloudflare
- ✦ cloudimage
- ✦ cloudinary
- ✦ contentful
- ✦ directus
- ✦ edgio
- ✦ fastly
- ✦ glide
- ✦ gumlet
- ✦ hygraph
- ✦ imageengine
- ✦ imagekit
- ✦ imgix
- ✦ ipx

- ✦ netlify

- ✦ prepr
- ✦ prismic
- ✦ sanity
- ✦ sirv
- ✦ storyblok
- ✦ strapi
- ✦ twicpics
- ✦ unsplash
- ✦ uploadcare
- ✦ vercel
- ✦ weserv

FONTS

- ✦ google
- ✦ adobe
- ✦ bunny
- ✦ fontshare
- ✦ fontsource
- ✦ local

90+
PROVIDERS

tighter vertical integrations

provider patterns



developer experience

intuition, convention + best practices

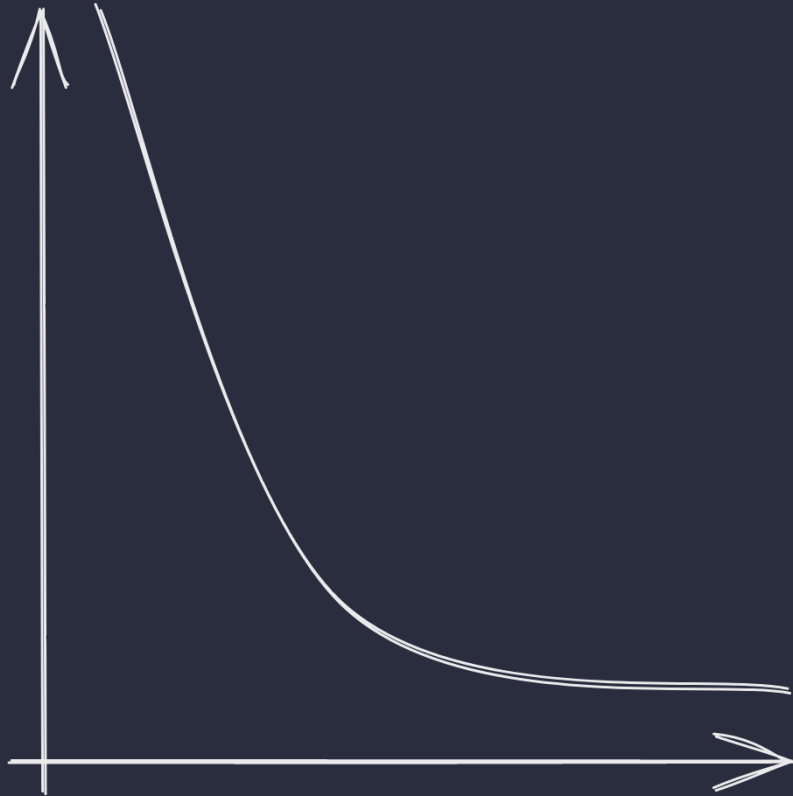
1. an unadulterated good
2. a vicious virtuous circle



a question

when have you felt the most productive?



Creativity



Constraint

developer experience

intuition, convention + best practices


- ▶ accessibility
- ▶ performance optimisations
- ▶ nuxt/a11y  ANTFU · DANIELROE
- ▶ nuxt/hints  BAROSHEM · DANIELROE
- ▶ nuxt/scripts  HARLAN-ZW · HUANG-JULIEN · FLASHDESIGNORY · HOUSSEINDJIRDEH

developer experience

intuition, convention + best practices

- ✦ optimised to be as **minimal** and **light** as possible
- ✦ **type-enabled** to reduce error and reduce context-switching

Daniel Roe
daniel@roe.dev

 @danielcroe

 @daniel@roe.dev

Interested ... ?

- ◆ check out the docs on nuxt.com and nitro.unjs.io
- ◆ follow [@nuxt_js](https://twitter.com/nuxt_js) on Twitter or [@nuxt@webtoo.ls](https://mastodon.social/@nuxt@webtoo.ls) on Mastodon
- ◆ join chat.nuxt.dev to discuss
- ◆ and a final offer 🙏

roe.dev/chat

