nuqs 2.5

Debounce, Standard Schema, TanStack Router, Key isolation, Global defaults, and more...

François Best
@francoisbest.com

22 August 2025

nuqs@2.5.0 is available, try it now:

npm pnpm yarn bun

```
npm install nuqs@latest
```

It's a big release full of long-awaited features, bug fixes & improvements, including:

- ⏱️ Debounce: only send network requests once users stopped typing in search inputs
- ☑️ Standard Schema: connect validation & type inference to external tools (eg: tRPC)
- ⚡ Key isolation: only re-render components when their part of the URL changes
- 🏝️ TanStack Router support with type-safe routing *(🧪 experimental)*

Debounce

While nuqs has always had a throttling system in place to adapt to browers rate-limiting URL updates, this system wasn't ideal for **high frequency inputs**, like `<input type="search">`, or `<input type="range">` sliders.

For those cases where the *final value* is what matters, **debouncing** makes more sense than **throttling**.

Do I need debounce?

Debounce only makes sense for **server-side data fetching** (RSCs & loaders, when combined with `shallow: false`), to control when requests are made to the server. For example: it lets you avoid sending the first character on its own when typing in a search input, by waiting for the user to finish typing.

The state returned by the hooks is always updated **immediately**: only the network requests sent to the server are debounced.

If you are **fetching client-side** (eg: with TanStack Query), you'll want to debounce the returned state instead (using a 3rd party `useDebounce` utility hook).

You can now specify a new option, `limitUrlUpdates`, that replaces `throttleMs` and declares either a debouncing or throttling behaviour:

```
import { debounce, useQueryState } from 'nuqs'

function DebouncedSearchInput() {
  // Send updates to the server after 250ms of inactivity
  const [search, setSearch] = useQueryState('search', {
    defaultValue: '',
    shallow: false,
    limitUrlUpdates: debounce(250)
  })

  // You can still use controlled components:
  // the local state updates instantly.
  return (
    <input
      type="search"
      value={search}
      onChange={e => setSearch(e.target.value)}
    />
  )
}
```

Read the complete documentation for the API, explanations of what it does, and a list of tips when working with search inputs *(you might not want to **always** debounce)*.

Standard Schema

You can now use your **search params definitions objects** (the ones you feed to `useQueryStates`, `createLoader` and `createSerializer`) to derive a **Standard Schema validator**, that you can use for basic runtime validation and type inference with other tools, like:

- tRPC, to validate procedure inputs when feeding them your URL state
- TanStack Router's `validateSearch` (see below)

```
import {
  createStandardSchemaV1,
  parseAsInteger,
  parseAsString,
} from 'nuqs'

// 1. Define your search params as usual
export const searchParams = {
  searchTerm: parseAsString.withDefault(''),
  maxResults: parseAsInteger.withDefault(10)
}

// 2. Create a Standard Schema compatible validator
export const validateSearchParams = createStandardSchemaV1(searchParams)

// 3. Use it with other tools, like tRPC:
router({
  search: publicProcedure.input(validateSearchParams).query(...)
})
```

Read the complete documentation for the options you can pass in.

Key isolation

Also known as *fine grained subscriptions*, and pioneered by TanStack Router, key isolation is the idea that components listening to a search param key in the URL should **only re-**

render when the value for that key changes.

Without key isolation, any change of the URL re-renders every component listening to it.

Take a look at those two counter buttons, and how clicking one re-renders both, *without key isolation*:

Increment "a": 0 Increment "b": 0

And this is what happens **with key isolation**:

Increment "c": 0 Increment "d": 0

Key isolation is now built-in for the following adapters:

- React SPA
- React Router (v6 & v7)
- Remix
- TanStack Router

No Next.js? 😢

Unfortunately, Next.js uses a single Context to carry a `URLSearchParams` object, which changes reference whenever any search params change, and therefore re-renders every `useSearchParams` call site.

I'm working with the Next.js team to find solutions to this issue to improve performance for everyone (not just nuqs users).

TanStack Router

We've added experimental support for TanStack Router, so you can load and use nuqsenabled components from NPM, or shared between different frameworks in a monorepo.

TanStack Router already has great APIs for type-safe URL state management, and we **encourage you to use those in your application code**. This adapter serves mainly as a compatibility layer.

This also includes *limited* support for connecting nuqs search params definitions to TSR's type-safe routing, via the Standard Schema interface.

Refer to the complete documentation for what is supported.

Other changes

Global defaults for options

You can now specify different defaults for some options, at the adapter level:

```
<NuqsAdapter
  defaultOptions={{
    shallow: false,                 // Always send network requests on updates
    scroll: true,                   // Always scroll to the top of the page on updates
    clearOnDefault: false,          // Keep default values in the URL
    limitUrlUpdates: throttle(250), // Increase global throttle
  }}
>
  {children}
</NuqsAdapter>
```

Preview support for Next.js 15.5 typed routes

Type-safe routing is now available as an option in Next.js 15.5.

While I'm still working on designing an API to support this elegantly, a little change to the serializer types can allow you to experiment with it in userland, using a copy-pastable utility function:

src/typed-links.ts

```
// Copy this in your codebase
import { Route } from 'next'
import {
  createSerializer,
  type CreateSerializerOptions,
  type ParserMap
} from 'nuqs/server'

export function createTypedLink<Parsers extends ParserMap>(
  route: Route,
  parsers: Parsers,
  options: CreateSerializerOptions<Parsers> = {}
) {
  const serialize = createSerializer<Parsers, Route, Route>(parsers, options)
  return serialize.bind(null, route)
}
```

Usage:

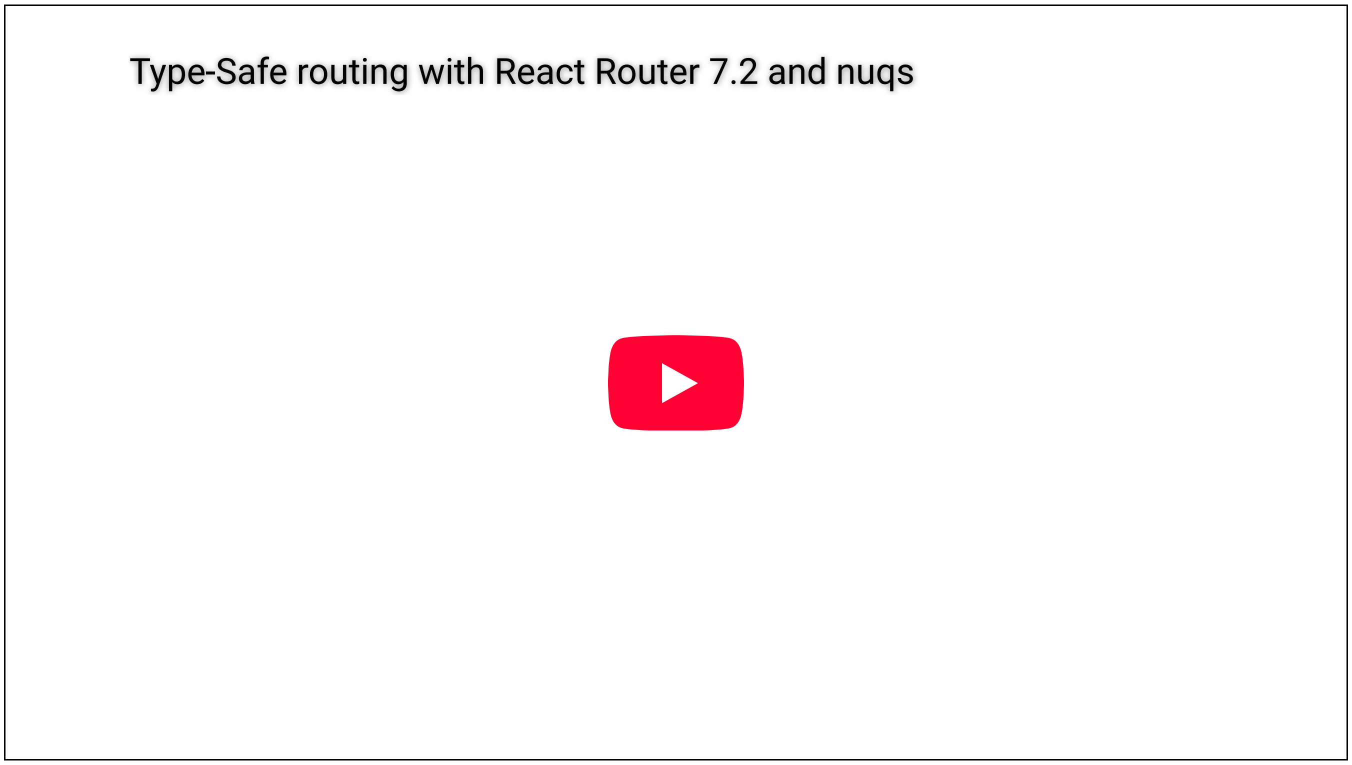
```
import { createTypedLink } from '@/src/typed-links'
import { parseAsFloat, parseAsIsoDate, parseAsString, type UrlKeys } from 'nuqs'

// Reuse your search params definitions objects & urlKeys:
const searchParams = {
  latitude: parseAsFloat.withDefault(0),
  longitude: parseAsFloat.withDefault(0),
}
const urlKeys: UrlKeys<typeof searchParams> = {
  // Define shorthands in the URL
  latitude: 'lat',
  longitude: 'lng'
}

// This is a function bound to /map, with those search params & mapping:
const getMapLink = createTypedLink('/map', searchParams, { urlKeys })

function MapLinks() {
  return (
    <Link
      href={
        getMapLink({ latitude: 48.86, longitude: 2.35 })
        // → /map?lat=48.86&lng=2.35
      }
    >
      Paris, France
    </Link>
```

This is based on the same technique I used on React Router's type-safe `href` utility in this video:



I'll open an RFC discussion soon to define the API, with the goals in mind that:

- It should support both Next.js & React Router typed routes *(if we could connect to TSR too that'd be nice 👀)*
- It should handle static, dynamic & catch-all routes, with type-safe pathname params, search params, and hash.

Dependencies & bundle size

nuqs is now a **zero runtime dependencies** library! 🙌

While this release packed a lot of new features, we kept it under **5.5kB** (minified + gzipped).

Full changelog

Features

- #855 Key isolation franky47
- #900 Debounce franky47
- #953 Add support for TanStack Router ahmedrowaihi
- #965 Add Standard Schema interface franky47
- #1038 Add `const` modifier to literal parsers to auto-infer their arguments as literals neefrehman
- #1062 Export ./package.json in exports field for Module Federation support AfeefRazick
- #1066 defaultOptions for NuqsAdapter TkDodo
- #1079 Add support for more global default options at the adapter level franky47
- #1083 Allow specifying a different base type for the serializer franky47

Bug fixes

- #996 Replace require by default conditional export field stefan-schubert-sbb (helps with ESM/CJS interop)
- #1057 Type inference for defaultValue of object syntax TkDodo
- #1063 Remove esm-only on TanStack Router export franky47
- #1073 Handle JSON in TanStack Router franky47

Documentation

- #787 Add inertia community adapter Joehoel → Read the docs
- #976 Add blog section franky47
- #1000 Vercel OSS program franky47
- #1004 Add code.store & oxom as sponsors 💖 franky47
- #1005 The URL type-safety iceberg franky47

- #1017 Fix non-null assertions franky47
- #1021 Add Deploy on Vercel button franky47
- #1025 Extend next-app example to include more features I-3B
- #1027 Fix mobile navbar collapse & sticky franky47
- #1032 Fix 500 error on Vercel ISR franky47
- #1037 Add Aurora Scharff as a sponsor 💖 franky47
- #1041 Fix transition docs to not call parser as function phelma
- #1043 Title is hidden behind headers on mobile awosky
- #1046 Update NUQS-404.md dmytro-palaniichuk
- #1051 add effect schema parser page ethanniser
- #1052 Debounce docs edits franky47
- #1056 Migrate docs to Fumadocs 15, Tailwind CSS v4 fuma-nama
- #1058 Fix default value for shallow in React Router disclaimer franky47
- #1070 prevent NaN appearing in pagination example 87xie
- #1082 Add nuqs 2.5 release blog post franky47

Other changes

- #985 Use React Compiler RC franky47
- #990 Replace tsup with tsdown franky47
- #1011 Add RSS feed auto-discovery franky47
- #1029 Fix API stability test franky47
- #1033 Linting PR titles franky47
- #1065 Add TanStack Router to og:images franky47
- #1067 Improve stats page franky47
- #1074 Configure MDX types remcohaszing
- #1077 Track beta versions adoption in the stats page franky47
- #1078 Update dependencies franky47
- #1080 Pass children as config in createElement to avoid ts-expect-error franky47

- #1081 Reduce the bundle size franky47
- #1086 Add publint step to CI workflow Amirmohammad-Bashiri *Congrats on your first OSS contribution! 🙌*

What’s next?

Long standing issues and feature requests include:

- Support for **native arrays** by repeating keys in the URL (eg: `?foo=bar&foo=egg` gives you `['bar', 'egg']`)
- **Runtime validation** with Standard Schema (Zod, Valibot, ArkType etc), to validate what TypeScript can’t represent (like number ranges & string formats).
- Support for **typed links** in Next.js 15.5 and React Router’s `href` utility.

Thanks

I want to thank sponsors, contributors and people who raised issues, discussions and reviewed PRs on GitHub, Bluesky and X/Twitter. You are the growing community that drives this project forward, and I couldn’t be happier with the response.

Sponsors

VERCEL INC. // 2025
OPEN SOURCE SOFTWARE PROGRAM

Thanks to these amazing people and companies, I’m able to dedicate more time to this project and make it better for everyone. Join them on 💖 GitHub Sponsors!

Contributors

Huge thanks to @87xie, @AfeefRazick, @ahmedrowaihi, @Amirmohammad-Bashiri, @AmruthPillai, @an-h2, @anhskohbo, @awosky, @brandanking-decently, @devhasson, @didemkkaslan, @dinogit, @dmytro-palaniichuk, @Elya29, @ericwang401, @ethanniser, @fuma-nama, @gensmusic, @I-3B, @jaberamin9, @Joehoel, @Kavan72, @krisnaw, @Manjit2003, @neefrehman, @phelma, @remcohaszing, @SeanCassiere, @snelsi, @stefan-schubert-sbb, @thewebartisan7, @TkDodo, @vanquishkuso, and @Willem-Jaap for helping!

Edit on GitHub • Comment on Bluesky

?n=u&q=s

Made by **François Best** and **contributors**

Quick Links

Documentation

Playground

Blog

Project Stats

Social

GitHub

YouTube

Bluesky