NEWS · JANUARY 30, 2026

Stop the Hype! Let's Look at the Facts Before Leaving Next.js in 2026

By Dr. Holger Flick

Here's an interesting fact: When I got deeply involved in the Delphi community back in 2004, **everybody was telling me "Delphi is dead."** The hot new thing was .NET, and conventional wisdom said anyone still writing in Delphi was living in the past.

Guess what? It's 2026, and **Delphi is still in production, still being actively developed, and I'm still making money writing code in Delphi**. In fact, my entire consulting business is built partly on helping companies maintain and modernize their Delphi applications.

I learned something valuable from that experience: **premature obituaries and hype cycles are terrible guides for technology decisions**. Longevity comes from solving real problems reliably, not from being the hottest thing on tech Twitter.

Which brings me to today's situation.

The tech world moves fast—sometimes too fast. In recent weeks, I've watched YouTube video after YouTube video, blog post after blog post, from popular tech personalities proclaiming they're abandoning Next.js for TanStack Start. The hype cycle is in full swing, and I get it: shiny new frameworks are exciting. But as someone who's been in this industry for over three decades, I've learned that decisions about production frameworks need to be based on facts, not feelings.

Let me be clear upfront: **this is my opinion**, formed through experience building and maintaining production systems. I encourage you to research these facts yourself and come to your own conclusion. Same for all the numbers that I collected here: **verify them independently**—don't just take my word for it. I used Google and other public sources to gather the data.

But before you jump ship, let's examine what we're actually comparing here.

The Company Behind Next.js: Vercel

Next.js isn't just some open-source side project—it's backed by Vercel, a company that has proven itself as a serious player in the web development infrastructure space.

Vercel's Track Record:

- Founded in 2015 (originally as ZEIT), rebranded to Vercel in 2020
- Raised over $313 million in funding across multiple rounds, with a valuation exceeding $2.5 billion as of their Series D in 2021
- Serves over 1 million developers and powers websites that handle billions of requests
- Enterprise customers include major brands like McDonald's, The Washington Post, Porsche, Twitch, and Under Armour
- Employs hundreds of people, with a significant engineering team dedicated to Next.js development

Next.js Itself:

- First released in October 2016—that's over 8 years of production use
- Powers some of the world's most trafficked websites
- Has a massive ecosystem: over 5 million weekly npm downloads
- Extensive documentation, tutorials, and community resources
- Regular security updates and patch releases
- Enterprise support options available

When you choose Next.js, you're choosing a framework backed by a well-funded company with a business model built around ensuring the framework's success. Vercel's revenue depends on Next.js being stable, scalable, and production-ready.

The Reality of TanStack Start

Now let's talk about TanStack Start. I want to be fair here—Tanner Linsley has built an impressive suite of libraries with TanStack Query (formerly React Query), TanStack Table, and others. The TanStack ecosystem has proven its value, and I respect the work immensely.

But here are the facts about TanStack Start:

- **No stable release exists.** As of January 2026, TanStack Start is still in Release Candidate status. It has not reached v1.0.
- TanStack is primarily maintained by Tanner Linsley and a smaller community of contributors
- While TanStack libraries are well-adopted, TanStack Start as a full-stack framework is unproven in large-scale production environments
- The documentation, while improving, is still being built out
- The ecosystem of third-party integrations, plugins, and tools is minimal compared to Next.js
- There is no enterprise company backing it with the same resources as Vercel backs Next.js

Let me emphasize: **being in Release Candidate status isn't a criticism—it's a fact.** It means the maintainers themselves don't yet consider it production-ready enough to commit to semantic versioning stability. That's responsible software development, not a shortcoming.

Corporate Liability and Framework Selection

Here's something that doesn't get discussed enough in YouTube videos: **corporate liability and due diligence**.

When you're making technology decisions for a business—whether you're a startup, a consultancy like mine, or an enterprise—you have a responsibility to your stakeholders. If you're building client projects, you have a contractual and ethical obligation to deliver stable, maintainable solutions.

Consider these questions:

- What happens when your production application breaks due to a framework bug?
- Who do you call at 2 AM when your startup's infrastructure is down?
- How do you justify to your CTO or client that you chose an RC-stage framework for a mission-critical application?
- What's your migration path if the framework changes direction or development slows?
- Can you get professional support and indemnification?

With Next.js and Vercel, there are clear answers:

- Established support channels, including paid enterprise support
- A company with financial backing and legal accountability
- Years of battle-tested code in production
- Extensive documentation and community knowledge
- Proven scalability and performance characteristics

With TanStack Start right now, you're essentially betting on a promising technology that hasn't yet proven itself at scale. That might be fine for personal projects or experimental work, but it's a different calculation for production systems serving real users and generating real revenue.

The Security Question: Server Actions and Beyond

Let's address the elephant in the room: **yes, Next.js has had security concerns**, particularly around Server Actions.

In 2024 and 2025, several security researchers highlighted potential vulnerabilities in how Server Actions could be exploited if developers weren't careful about authorization checks. These were real concerns that the Next.js team took seriously, and they've implemented multiple layers of protection:

- Built-in CSRF protection for Server Actions
- Clear documentation on authorization best practices
- Tooling to help identify potential security issues
- Regular security audits and rapid patch releases
- A security disclosure program with responsible disclosure practices

The key point: **these issues were identified, disclosed, and addressed** because Next.js has the scrutiny that comes with widespread adoption. Thousands of security researchers and developers are examining Next.js code constantly.

What about TanStack Start?

TanStack Start uses a different approach with its server functions, and initial indications suggest thoughtful security design. But here's the reality: **it hasn't been tested at scale**. Security vulnerabilities often don't emerge until a framework is under real-world stress, handling edge cases that weren't anticipated during development.

This isn't a criticism of TanStack Start's security—it's an acknowledgment that security is proven through time, testing, and the crucible of production use. Next.js has been through that crucible. TanStack Start hasn't yet had the opportunity.

Team Size, Support, and Community

The size and support structure of a framework ecosystem matters enormously for long-term viability.

Next.js/Vercel:

- Dedicated team of full-time engineers working on the framework
- Regular release cycles with clear roadmaps
- Extensive official documentation
- Massive community: thousands of contributors, millions of users
- Abundant third-party resources: courses, tutorials, tools

- Active Discord community, Stack Overflow presence, GitHub discussions
- Multiple books published about Next.js development
- Enterprise support contracts available

TanStack Start:

- Primarily maintained by Tanner Linsley and community contributors
- Smaller community in its early stages
- Growing but limited documentation
- Minimal third-party educational resources currently available
- Community support through Discord and GitHub
- No formal enterprise support structure

Again, I'm not criticizing TanStack Start here—every framework starts small. React started small. Next.js started small. But that's precisely the point: **they started small and proved themselves over years** before becoming the foundation for mission-critical applications.

Deployment Options: Where Will You Host?

Deployment flexibility is crucial for production applications.

Next.js Deployment Options:

- Vercel (optimized, zero-config deployment)
- Self-hosted on any Node.js environment (AWS, Google Cloud, Azure, DigitalOcean, etc.)
- Docker containers
- Serverless platforms (AWS Lambda, Google Cloud Functions, etc.)
- Traditional VPS or dedicated servers
- Kubernetes clusters
- Specialized platforms (Railway, Render, Fly.io, etc.)
- Static export for simple use cases

The ecosystem has matured to the point where you have genuine choice. If you decide Vercel isn't right for you, you're not locked in—you can deploy Next.js anywhere Node.js runs.

TanStack Start Deployment Options:

- Currently more limited, though expanding
- Requires more manual configuration for various platforms
- Documentation for deployment scenarios still being developed
- Less testing of various hosting configurations in production

As TanStack Start matures, this will improve. But right now, if you need to deploy to a specific environment with specific requirements, Next.js has proven solutions and battle-tested configurations.

My Take: Hype vs. Reality

Here's what I believe is happening: **we're in a hype cycle**, driven by the very human desire for novelty and the algorithmic incentives of content creation. Creating content about "Why I'm leaving X for Y" generates clicks, engagement, and revenue for content creators. I don't blame them—it's their business model.

But **your business model is different**. Your job is to deliver reliable, maintainable, scalable applications to your users or clients.

Just like how "Delphi is dead" was wrong in 2004 and remains wrong in 2026, **today's hot takes about Next.js being obsolete are likely premature**. The frameworks that survive aren't always the ones generating the most YouTube views—they're the ones solving real problems for real businesses.

Let me be absolutely clear about my position:

For Personal Projects and Learning:

- Absolutely experiment with TanStack Start
- Learn its patterns and approaches
- Contribute to the community
- Build side projects and share your experiences
- This is how frameworks mature and improve

For Production Applications:

- Stick with proven technology like Next.js
- Wait for TanStack Start to reach v1.0 at minimum
- Wait for real-world production case studies to emerge
- Wait for the ecosystem to mature
- Let others work through the growing pains

Yes, times change faster now. The pace of innovation in web development is breathtaking. But **faster doesn't mean we should abandon wisdom**. Next.js has proven itself over **more than 8 years** of production use, across countless applications, serving billions of requests, weathering security challenges, and evolving with the needs of developers.

That proof matters. That track record matters. That stability matters.

The Path Forward

My recommendation is simple:

1. **Keep building production applications with Next.js** while staying informed about alternatives
2. **Experiment with TanStack Start** in non-critical contexts to understand its strengths and approach
3. **Watch for TanStack Start v1.0** and the production case studies that will follow
4. **Evaluate your specific needs** against the maturity and capabilities of each framework
5. **Make decisions based on facts**, not hype cycles or what's trending on YouTube

In a year or two, this conversation might be completely different. TanStack Start might have proven itself with thousands of production deployments, a robust ecosystem, and clear advantages over Next.js. If that happens, I'll be among the first to reevaluate my position based on the new facts.

But today, in January 2026, those facts don't exist yet.

Conclusion

I respect Tanner Linsley's work enormously. I think TanStack Start looks promising, and I'm excited to see where it goes. I also respect the tech creators making content about their framework preferences—they're entitled to their opinions and approaches.

But I've been in this industry long enough to see hype cycles come and go. I remember when everyone was leaving jQuery for Backbone. Then everyone left Backbone for Angular. Then everyone left Angular for React. Some of those transitions were warranted; some were premature.

And I remember when "Delphi is dead" was the conventional wisdom. Yet here we are, 22 years later, and I'm still writing profitable Delphi code alongside my modern web development work.

The question isn't whether TanStack Start is good—it appears to be very promising. The question is: **has it proven itself sufficiently for production use?**

For me, the answer is: not yet.

Give it time. Let it reach v1.0. Let companies deploy it at scale. Let the security researchers examine it. Let the community build the ecosystem. Let the documentation mature. Let the edge cases emerge and get resolved.

Proven technology isn't boring—it's responsible.

And responsibility is what our users, our clients, and our businesses deserve.

About the Author: I'm Holger, CEO of FlixEngineering LLC and a developer with over 30 years of experience. I help companies navigate technology transitions, including migrations from legacy systems to modern web platforms. My opinions are my own, formed through decades of building and maintaining production systems—and yes, I still write Delphi code profitably in 2026.

INDEPENDENT · AD-FREE · READER-SUPPORTED

Free to read. Not free to make.

Every tutorial, article, and line of example code on Holger's Code is published without a paywall and without ads. This page is about what keeps it that way.

Professional work, published free

The content here is held to the same standard as professional client work: every tutorial is built as a real project, tested against current versions, documented step by step, and revisited when the frameworks move on. A single in-depth tutorial routinely takes days — and that's before the bills behind it: servers, storage, build infrastructure, licenses, and the hardware everything is verified on.

If an article or tutorial here has saved you an afternoon of debugging, taught you a technique, or helped you ship, it created real value for you. Supporting it isn't charity — it's paying for professional work you found useful, at whatever price you choose. And if you can't, keep reading anyway. That's what the free part means.

Ways to support

KO-FI

Buy me a coffee

A one-time tip — no account, no subscription, any amount.

BOOKS & COURSES

Buy the paid work

The paid tier of everything published free here — and you get something lasting in return.

SPREAD THE WORD

Share an article

Send a tutorial to a colleague, cite an article, subscribe to the RSS feed.

Nothing here goes behind a paywall, whether you chip in or not. But if something on this site earned its keep today, this is where to say so.

Support on Ko-fi —