



NILS HARTMANN

<https://nilshartmann.net>

Moderne

React

Anwendungen

mit TanStack

GEDOPLAN EXPERTENKREIS | ONLINE | 3. JULI 2025 | [HTTPS://REACT.SCHULE](https://react.schule)

NILS HARTMANN

nils@nilshartmann.net

Freiberuflicher Software-Entwickler, –Architekt, Coach, Trainer

Java, React, TypeScript



<https://graphql.schule>



<https://react.schule>

[HTTPS://NILSHARTMANN.NET](https://nilshartmann.net)

Unser

TanStack

heute

Data Fetching

TanStack Query

aka React Query

Data Fetching

TanStack Query

aka React Query

Routing

TanStack Router

TECHNOLOGIE STACK

Data Fetching

TanStack Query

aka React Query

Routing

TanStack Router

Fullstack

TanStack Start

TECHNOLOGIE STACK

Data Fetching

TanStack Query

aka React Query

Routing

TanStack Router

Fullstack

TanStack Start

Experimental!

Demo



Fazit 

- **Schnelle erste Darstellung**

- Lazy Loading
- SSR
- Server Functions (Daten auf dem Server laden)
- Suspense (nicht auf niedrig priorisierte Daten warten)

- **Performante Darstellung**

- Prefetching (serverseitige Daten möglichst früh laden)
- Vermeiden von Request-Wasserfällen (insb. zwischen Client und Server)
- So wenig wie möglich neu rendern
- Cache Updates im Hintergrund

- **Konsistente Darstellung**

- Globaler State (alle Komponenten bedienen sich aus derselben Datenquelle)
- Cache (für serverseitige Daten)

Anhang

Bibliotheken

und

Frameworks

- **TanStack Router: Typsicheres Routing**

- <https://tanstack.com/router/latest>
- Routen werden aus Dateien und Verzeichnissen generiert
- Alle Routing-Informationen sind typsicher (inklusive Search Parameter)
- Effizientes Arbeiten mit Search Parametern
- Gute Integration mit TanStack Query
- Generator für neue Router-Apps: <https://www.npmjs.com/package/create-tsrouter-app>
- Alternative: React Router (<https://reactrouter.com/>)
 - Überzeugt mich nicht (mehr)

- **TanStack Query: Vollständige Data-Fetching-Lösung**

- <https://tanstack.com/query/latest>
- ehemals bekannt als "React Query"
- Flexibler Cache für serverseitige Daten
- Hohe Typsicherheit
- Keine Notwendigkeit für useEffect mehr (für Data-Fetching)

- **zod: Laufzeitvalidierung**

- <https://zod.dev/>
- Typen- und Wertebereiche werden mit zod API beschrieben
- TypeScript-Typen werden aus der Beschreibung generiert
- zod validiert Objekte zur Laufzeit und leitet TypeScript-Typen ab
- Sehr gute Integration in die TanStack Bibliotheken
- Alternative: ArkType (<https://arktype.io/>)
 - Noch sehr neu
 - "Spezielle" API

- **ky: Alternative zu fetch und axios**

- <https://github.com/sindresorhus/ky>
- Kleiner Wrapper um die fetch API
- Spezialisierte Methoden für die einzelnen HTTP-Methoden
- Automatische JSON-Konvertierung inklusive zugehöriger Header
- Vereinfachte Verarbeitung des Payloads
- HTTP Status Codes, die Fehler anzeigen führen zu Errors

- **TanStack Start: Fullstack-Framework für React**

- <https://tanstack.com/start/latest>
- Bietet SSR und Server Functions
- (Noch) kein Support für React Server Components (RSC)
- Basiert auf dem TanStack Router
- Gute Integration von TanStack Query (optional)
- Alternativen:
 - React Router (vergleichbarer Ansatz)
 - Next.js (eher Server-first Ansatz)

Experimental!

NILS HARTMANN

<https://nilshartmann.net>

Vielen Dank

Code & Slides: <https://react.schule/gedoplan-tanstack>

Fragen und Kontakt: nils@nilshartmann.net