

**NILS HARTMANN**

<https://nilshartmann.net>

Slides: <https://react.schule/oose-react19>

# React 19

Die wichtigsten  
Neuerungen

im praktischen

# Überblick

OOSE EVENTS | HAMBURG, 10. SEPTEMBER 2024 | @NILSHARTMANN

# NILS HARTMANN

nils@nilshartmann.net

**Freiberuflicher Entwickler, Architekt, Trainer aus Hamburg**

**Java, Spring, GraphQL, React, TypeScript**



<https://graphql.schule/video-kurs>



<https://reactbuch.de>

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

# NILS HARTMANN

Seminare bei OOSE

Freiberger Software-Architekt, Trainer aus Hamburg

Java, Spring, GraphQL, React

Single-Page-Anwendungen mit React und TypeScript

Webanwendungen mit React - Fortgeschrittene Konzepte

GraphQL-Anwendungen mit Spring Boot

<https://graphql.schule/videokurs>

<https://reactbuch.de>

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

### Release Candidate (seit April!)

BLOG >

# React 19 RC


April 25, 2024 by [The React Team](#)

---


React 19 RC is now available on npm!

- Ihr könnt die Version aber schon installieren und verwenden
  - Siehe dazu: <https://react.dev/blog/2024/04/25/react-19-upgrade-guide>

## Warum nicht stabil?

- "Probleme" mit Suspense
-  React 18 vs. React 19
  - React **18** Workspace ~/develop/javascript/hello-react19-but-actually-18.3
  - React 19 Workspace ~/develop/javascript/hello-react19

## Blocker: "Problem" mit Suspense

-  React 18 vs. React 19
- ansonsten wohl Feature Complete

## React Versionen

- **latest**: die aktuelle stabile Version (zurzeit 18.3)
- **canary**: "Stabil", zur Integration/Anpassung für Library Autoren
- **experimental**: Neue, noch nicht fertige Features
- **rc**: entspricht aktuell dem canary-Release
- Installation mit dem entsprechenden Version-Tag:
  - `pnpm add --save-exact react@canary react-dom@canary`

### Für die Migration: React 18.3

- Als Vorbereitung auf Version 19 könnt ihr React **18.3** installieren
- Darin sind keine neuen Features
- Aber Warnungen etc. die mögliche React 19-Probleme in eurem Code anzeigen



### use-Funktion

- Mit der use-Funktion kann auf Ressourcen zugegriffen werden:
  - Promises
  - Contexte
- Die Funktion ist kein Hook!
- Die use-Funktion unterliegt nicht den "Rules of Hook"
  - Kann also z.B. in if-Abfragen verwendet werden

## use-Funktion mit Promises

- Mit use kann man auf ein Promise warten
- Achtung! Das muss stabil sein (z.B. vom Router)
- Kann auch vom Server kommen! (dazu später mehr)
- 🕵️ 12\_use rendern
- 🕵️ 15\_use rendern conditional

## Vereinfachungen beim Context

- createContext gibt jetzt direkt den Provider zurück
  - MyContext.Provider damit überflüssig
  - MyContext.Consumer gibt es nicht mehr
- Context kann mit der use-Funktion abgefragt werden
- 🕵️ 20\_use\_context

## React Compiler

- Separates Projekt, nicht Teil von React 19
- Fügt Memoisierungen in euren Code ein
- Ergebnis dann ähnlich wie React.memo, useMemo, useCallback
  - Technisch über das Analysieren und Verfolgen von Abhängigkeiten
- Eure Bundle-Size wird dadurch größer
- Ihr müsst die Rules of React befolgen

## React Compiler: Healthcheck

- Bevor ihr den Compiler verwendet, könnt ihr einen "Healthcheck" machen, ob eure Codebasis damit funktioniert
- Es gibt einzelne Bibliotheken die mit dem Compiler nicht kompatibel sind.

```
npx react-compiler-healthcheck@latest
```



### React Compiler: Schrittweise Einführung

- Es gibt ein ESLint Plug-in, dass euch warnt, wenn ihr Code schreibt, der mit dem Compiler nicht kompatibel ist
- Ihr könnt den Compiler auf einzelne Dateien beschränken (z.B. bei Problemen)
- Bestehende useCallback, useMemo etc.-Aufrufe können im Code bleiben
- Es gibt einen Playground zum ausprobieren: <https://playground.react.dev/>
- Die React DevTools bieten auch Support für den Compiler

## React Compiler: Installation


- Der Compiler ist zurzeit ein Babel Plug-in
- Der Kern ist aber unabhängig von Babel
  - Es wird Adapter für andere Buildtools geben
- In Next.js (ab Version 15) könnt ihr den Compiler per Konfiguration ein- und ausschalten

## React Compiler: Demo

- Verzeichnis: 25\_compiler
  - Renderzyklen analysieren
  - Wie könnten wir optimieren? 
  - Compiler in vite-Konfiguration einschalten
  - Was passiert? 
  - Dev Tools "Auto Memo"
  - Eventuell Promise für JSX-Element



### Optimierungen beim Arbeiten mit externen Ressourcen

- Unterstützung für preload, preinit etc. vom Browser
- Dazu könnt ihr neue Funktionen nutzen
  - preInit(...): Laden und Ausführen von JS- oder CSS-Dateien
  - preLoad(...): Vor-laden von Schriften- oder CSS-Dateien
  - prefetchDNS und preconnect: Frühzeitig Verbindungen aufbauen
- Generiert werden <link rel="preload" /> etc. Elemente
-  29\_asset\_loading, preinit


## Stylesheets und Scripte

- Unterstützung für externe Stylesheets und JavaScript-Dateien in euren Komponenten
  - link-Element im Header
  - script-Element
- React stellt sicher, dass die Elemente nur einmal eingebunden werden (auch bei Verwendung in mehreren Komponenten)
- JS-Code kann inline oder extern angegeben werden
- Das funktioniert laut Doku mit Suspense – hat bei mir aber nicht funktioniert wird
- 🕵️ 30\_stylesheets\_und\_links


## Dokument Meta-Daten

- HTML-Elemente, die sich im head-Bereich der Seite befinden können gesetzt werden
  - title, meta, link
- Das "Mergen" von Elementen funktioniert allerdings nicht
  - Man wird also auch künftig auf Bibliotheken wie react-helmet setzen
  - In Next.JS gibt es schon weiteren Support dafür
- 🕵️ 32\_meta\_title


### Transitions

- Mit React 19 können mit `useTransition` asynchrone Funktionen verwendet werden
- Damit kann man typische Muster beim Arbeiten mit asynchronen Daten vereinfachen
- Fehlerbehandlung erfolgt dann zum Beispiel über Error Boundaries
-  40\_transition


### Optimistische Updates

- Mit `useOptimistic` kann ein "optimistischer" State gesetzt werden
- Dieser ist so lange gültig, wie eine Transition läuft
- Damit kann man das vermutete Ergebnis einer Änderung frühzeitig darstellen
-  40\_transition


## Zustandsverwaltung mit `useActionState`

- `useActionState` bietet einen Ersatz bzw. Alternative zu `useState`
- Eine action-Funktion bekommt den Formular-Inhalt beim Submit
- Die Funktion liefert einen State zurück (z.B. eine Fehlermeldung)
- Das Formular bekommt die Information, ob das Submit gerade läuft
- Mit Server-Framework kann man damit Formulare bauen, die zur Laufzeit ohne JS auskommen
-  50\_formulare

## Zustandsverwaltung mit useFormStatus


- Mit useFormStatus kann man den Status eines Formulars abfragen
- Zum Beispiel: ob das Formular gerade submitted wird und welche Action ausgeführt wurde
- Damit kann man zum Beispiel wiederverwendbare Buttons bauen
- Wohl in erster Linie für Bibliotheken
- Der Hook muss in einer Kind-Komponente unterhalb des Formulars verwendet werden
-  50\_formulare

## Direktiven


- React Komponenten können auf dem Server und Client gerendert werden
- Das hängt von deren Verwendung ab
- React Server Components werden nur im Server gerendert
- Um Code-Stellen zu markieren, die für den Server bzw. den Client bestimmt sind, gibt es die "use client" und "use server"-Direktiven
- Zur Verwendung braucht ihr ein Server-Framework (Next.js)
-  hello-react19-nextjs



## React Server Komponenten

- React Server Komponenten werden im Server (oder Build) gerendert, nie auf dem Client
- Der Client bekommt nur die fertig gerenderten Komponenten
  - Dadurch wird JavaScript-Code eingespart
- Server Komponenten können asynchron sein
  - Deswegen kann man darin direkt Data Fetching machen
-  hello-react19-nextjs

## Server Actions

- "Server Actions" sind Funktionen, die auf dem Server laufen
  - Diese können aus euren Komponenten im Client (!) aufgerufen werden
  - Sie sind damit eine Art Remote-Procedure-Call (RPC)
- Ihr könnt sie zum Beispiel einem Formular als action übergeben
- Das Server-Framework stellt dafür automatisch einen Endpunkt zur Verfügung
-  hello-react19-nextjs

**NILS HARTMANN**

<https://nilshartmann.net>

<https://reactbuch.de>



# Vielen Dank!

Slides: <https://react.schule/oose-react19>

Source-Code: <https://github.com/nilshartmann/neues-in-react-19>

Fragen & Kontakt: [nils@nilshartmann.net](mailto:nils@nilshartmann.net)

Twitter: [@nilshartmann](https://twitter.com/nilshartmann)