React Performance

author: Christoph Schmalhofer

BOB 2021

==========

InNuce Solutions (Hamburg)

- Fuhrparkmanagement Software seit 2004
- Seit 2019:
- React Material-UI
- Apollo GraphQL Client
- React Hooks

==========

Erfahrung React Performance

- IO Performance unkritisch
- große Wertebereiche (Select Box, Autocomplete) problematisch
- Formulare mit vielen komplexen Controls werden langsam

==========

React Programmiermodelle

- Performance Charakteristik
- Optimierungsmöglichkeiten

==========

Vorweg

- Production Build
- React Dev Tools
- O(viewsize)
- Form State (React Hook Form)

==========

Demo State, Controlled/Uncontrolled Components

- keine direkte Geschwister Kommunikation
- State wandert nach oben

==========

Reacts Standard Programmiermodell (Hooks)

Funktionales Rendering

- funktionale Berechnung der View (VDOM)
- State (und Props) Änderungen sind in der Renderphase nicht zulässig
- Standard Mode: Kind Komponenten werden auch berechnet

Effekt- und Eventhandler

- Updates werden gebatched, Updatefunktionen bevorzugt

explizite Datenabhängigkeiten

- Lifecycle implizit

==========

Demo Cache/Memo

- React Developer Tools

==========

Standardoptimierung: Komponenten Caching

- React.memo Higher Order Component
- Default: Vergleich aller Props
- explizite Vergleichsfunktion möglich
- Alternativ: React.useMemo (Dependency Array)

- für Komponenten und allgemeine Berechnungen geeignet

==========

Komponenten Caching II

- Berechnung ohne Seiteneffekte
- Key Attribut für Listeneinträge
- möglichst schlanke Props (Kübelpattern vermeiden)
- Objektliterale vermeiden
- useCallback oder useReducer für Funktionsobjekte
- automatische Memoisierung immer wieder mal diskutiert

==========

Demo Store (useRef)

- manuelle Kontrolle des Renderings
- Concurrent Mode unsafe

==========

Demo Jotai (Reactivity)

- Datenabhängigkeiten werden automatisch getracked
- Atome, Atomfamilien, Abgeleitete Atome
- Ähnlichkeiten zu Recoil und MobX
- Concurrent Mode safe

==========

Demo Jotai Reaktive React Komponenten

- gute Performance auch bei Insert/Delete
- intelligentes Caching von React Komponenten

==========

Programmiermodelle

- React State
- Store / Session
- Fine Grained Reactivity

==========

Backup: Concurrent Mode (experimental) / React Fiber

- React kann Kompontenrendering flexibel steuern
- Suspense und Transition
- Priorisierung, Bail out, Wiederholung
- keine unkontrollierten Seiteneffekte

==========

Backup: React Fiber / Hooks / Algebraische Effekte

Abramov: Now, if algebraic effects were coming to JS I think we'd happily change Hooks to use them.

UI Runtime (React)	Language Runtime
function component	function
React Context	dynamic scope
Fiber	Stack frame
Component Instance	delimited continuation (multi shot)
React State	state effect
React Runtime	Effect Handler
VDOM Diffing	Partial Evalulation

==========

Backup: React Server Components (experimental)

- Serverkomponenten laufen auf dem Server (nicht nur initiales Rendering)
- können asynchron zum Client übertragen/serialisiert werden
- Optimierung der Datenübertragung zum Client
- können Daten an Clientkomponenten durchreichen

- benötigen Concurrent Mode

==========

Referenzen

Application State Management with React

- Gute Einführung und pragmatische Performance Hinweise

A (Mostly) Complete Guide to React Rendering Behavior

- der Beitrag verdient eine Referenz in der React Dokumentation

React as a UI Runtime

- die Konzepte hinter React

==========

Michel Weststrate — MobX and the unique symbiosis of predictability and speed

- Michel Westrates Beiträge sind allesamt empfehlenswert

Jotai

- Daishi Katos weitere Aktivitäten im Bereich State Management/Concurrent Mode sind auch interessant

React Hook Form

- ignoriert Reacts Statemanagement
- gute Performancedemos

==========

React Server Components

Continuation-based partial evaluation

- bekannte Optimierung - Korrespondenz zur React Runtime

Prepack

- "Prepack is a partial evaluator for JavaScript"