Async State Management mit TanStack Query und Serverless Deployments in Next.js

Einleitung

Dieses Handout bietet einen Überblick über zwei Konzepte in der modernen Webentwicklung mit NextJS:

- Asynchrones State Management mit TanStack Query und
- Serverless Deployments mit Next.js (Theoretisch).

Inhaltsverzeichnis

1. Einführung in Next.js

Next.js ist ein leistungsstarkes React-Framework, das sich auf die Entwicklung moderner Webanwendungen konzentriert. Im Gegensatz zu reinen React-Anwendungen bietet Next.js zusätzliche Funktionen und Vorteile, die insbesondere für Entwickler von Angular-Anwendungen interessant sein könnten.

Überblick und Vorteile von Next.js

- **Serverseitiges Rendering (SSR)**: Next.js ermöglicht serverseitiges Rendering, wodurch Seiten schneller geladen und Suchmaschinenoptimierung verbessert werden (SEO Optimization).
- **Statische Seitengenerierung (SSG)**: Mit Next.js können statische Seiten vorab generiert werden, was zu verbesserten Ladezeiten und geringerem Serveraufwand führt.

  Beispiel:

```
   Creating an optimized production build ...
 ✓ Compiled successfully
   Linting and checking validity of types ...
   Collecting page data ...
 ⚠ Using edge runtime on a page currently disables static generation for that page
   Generating static pages (0/6) ...
   Generating static pages (1/6)
   Generating static pages (2/6)
   Generating static pages (4/6)
 ✓ Generating static pages (6/6)
   Finalizing page optimization ...
   Collecting build traces ...

Route (app)                              Size     First Load JS
┌ ○ /                                    71.9 kB         219 kB
├ ○ /_not-found                          885 B          85.1 kB
├ ƒ /api/beta/join                       0 B                0 B
└ ○ /beta                                7.66 kB         138 kB
+ First Load JS shared by all            84.2 kB
  ├ chunks/69-0d964a44524983ba.js        28.9 kB
  ├ chunks/fd9d1056-1a0090bc0ecca89d.js  53.4 kB
  └ other shared chunks (total)          1.94 kB


○  (Static)        prerendered as static content
ƒ  (Edge Runtime)  server-rendered on demand using the Edge Runtime
```

- **Eingebaute Routenverwaltung**: Next.js bietet eine integrierte Routenverwaltung, die die Entwicklung von Single-Page-Anwendungen vereinfacht.
- **Optimierung für Entwicklerproduktivität**: Next.js enthält integrierte Entwicklerwerkzeuge wie Hot Reloading und automatisches Code-Splitting, um die Entwicklungszeit zu verkürzen.

Warum Next.js für Angular-Entwickler?

- **Ähnliche Konzepte**: Next.js und Angular teilen viele Konzepte wie Komponentenbasierte Architektur, Routing und Serverseitiges Rendering, was den Einstieg erleichtert.
- **Reiche Ökosystem**: Next.js verfügt über ein reichhaltiges Ökosystem an Bibliotheken, Plugins und Tools, das die Entwicklung von anspruchsvollen Webanwendungen erleichtert.

2. Async State Management mit TanStack Query

TanStack Query (früher bekannt als React Query) ist eine mächtige Bibliothek zum Abrufen, Caching und Synchronisieren von serverseitigen Daten in React-Anwendungen.

- **Vorteile und Anwendungsfälle**: Automatisches Caching, Datenvorababruf (Prefetching), Synchronisierung im Hintergrund, einfache Fehlerbehandlung.
- **Codebeispiele**:

```
import { QueryClient, QueryClientProvider, useQuery } from 'react-query';

const queryClient = new QueryClient();

function MyApp({ Component, pageProps }) {
  return (
    <QueryClientProvider client={queryClient}>
      <Component {...pageProps} />
    </QueryClientProvider>
  );
}

export default MyApp;

function FetchData() {
  const { data, error, isLoading } = useQuery('fetchData', fetchDataFromAPI);

  if (isLoading) return 'Loading...';
  if (error) return 'An error occurred';

  return <div>{JSON.stringify(data)}</div>;
}
```

3. Serverless Deployments

- Serverless bedeutet, dass der Code ohne Serververwaltung ausgeführt wird. Dienste wie AWS Lambda, Vercel Functions und Netlify Functions machen dies möglich.
- **Vorteile von Serverless Deployments**: Kosteneffizienz, automatische Skalierung, reduzierte Serververwaltung.
- **Integration in Next.js**: Nutzung von API-Routen in Next.js, um serverlose Funktionen zu erstellen.
- **Beispiele und Best Practices**:

```
// Beispiel: Einfache serverlose API-Route in Next.js
export default function handler(req, res) {
  res.status(200).json({ message: 'Hello from Serverless Function!' });
}
```

5. Fazit und weiterführende Ressourcen

- **Zusammenfassung**: Die Kombination von TanStack Query und Serverless Deployments in Next.js ermöglicht die Erstellung moderner, skalierbarer und effizienter Webanwendungen.
- **Weiterführende Ressourcen**:
  - Offizielle Next.js-Dokumentation
  - TanStack Query-Dokumentation
  - Serverless Framework
  - Vercel