

Universal Rendering

Julian Beck

Betreuer: Prof. Dr. rer. nat. Christian Zirpins

Zusammenfassung An dieser Stelle sollte später eine Kurzzusammenfassung stehen.

Inhaltsverzeichnis

| | | |
|-----|--|---|
| 1 | Einleitung | 3 |
| 1.1 | Anforderungen an eine Webanwendung | 3 |
| 1.2 | Terminologie | 3 |
| 2 | Serverseitiges Rendering | 3 |
| 2.1 | Serverseitiges Rendering mit Ajax | 3 |
| 3 | Clientseitiges Rendering | 4 |
| 4 | Universal Rendering | 5 |
| 4.1 | Isomorphic JavaScript | 5 |
| 4.2 | Virtuelles DOM | 5 |
| 4.3 | Clientseitige Hydration | 5 |
| 4.4 | Rendering Ablauf | 5 |
| 4.5 | Vorteile | 6 |
| 4.6 | Nachteile | 6 |
| 4.7 | Alternativen | 6 |
| 5 | Frameworks | 7 |
| 5.1 | React und Next.js | 7 |
| 5.2 | Vue.js und Nuxt.js | 7 |
| 5.3 | Angular Universal | 7 |
| 6 | Universal Rendering in der Praxis | 8 |
| 7 | Fazit und Ausblick | 9 |

1 Einleitung

Seit dem Beginn des Webs funktioniert das Surfen wie folgt: Ein Webbrowser fordert eine bestimmte Seite an, ein Server im Internet bearbeitet die Anfrage und generiert ein HTML (Hypertext Markup Language) Dokument als Antwort. Dies bezeichnet man als serverseitiges rendern. In den Anfängen des Webs stellte dies kein Problem dar, da die Browser nicht leistungsstark waren und die Webseiten aus meist statischen Inhalt bestanden. Später mit HTML5 wurden Webseiten dynamischer und interaktiver für den Nutzer, was dazu führte, dass immer mehr Apps, sogenannte Single Page Applikationen, vollständig im Browser auf einer Seite liefen. Um dies zu ermöglichen wird clientseitiges rendern verwendet. Single Page Applications oder kurz SPAs, bieten Vorteile für den Anwender: Sie reagieren schnell auf Benutzerinteraktionen und können zwischen Seiten navigieren, ohne sie komplett neu zu laden. Gleichzeitig sind SPAs komfortabel zu entwickeln, dank moderner Frameworks. Beide Varianten, serverseitiges und clientseitiges rendern, haben Vor- und Nachteile. Universal Rendering kombiniert die beiden Ansätze und erfüllt alle Anforderungen an eine moderne Webanwendung.

1.1 Anforderungen an eine Webanwendung

1.2 Terminologie

2 Serverseitiges Rendering

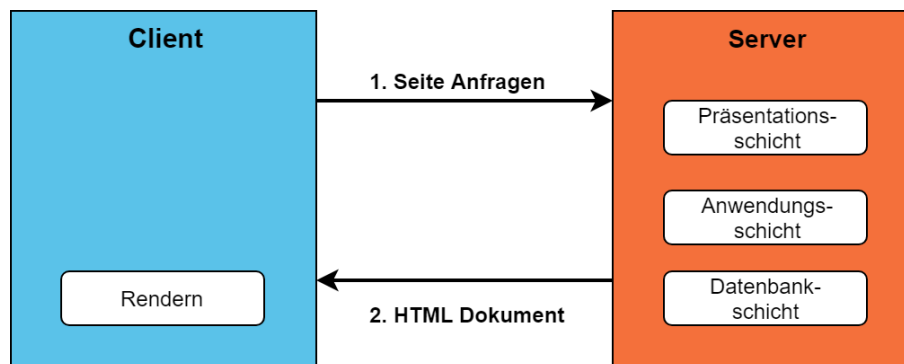


Abbildung 1. HTML Dokument einer React Seite

2.1 Serverseitiges Rendering mit Ajax

Dies ist ein Zitat [BeKR09]. test[Jaso16] test[uJoh18] test[Joel16]

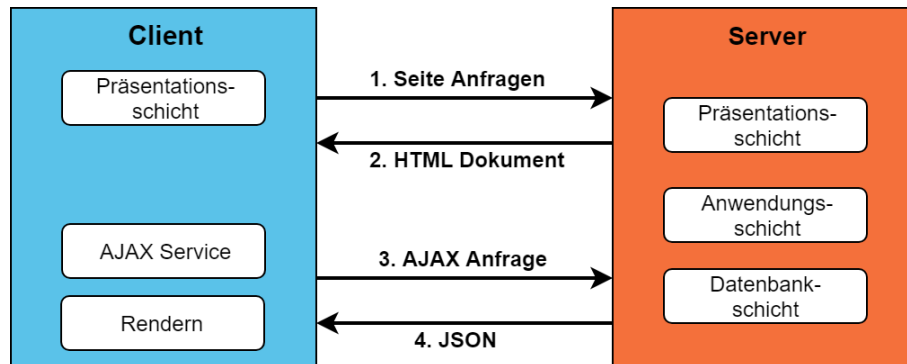


Abbildung 2. HTML Dokument einer React Seite

3 Clientseitiges Rendering

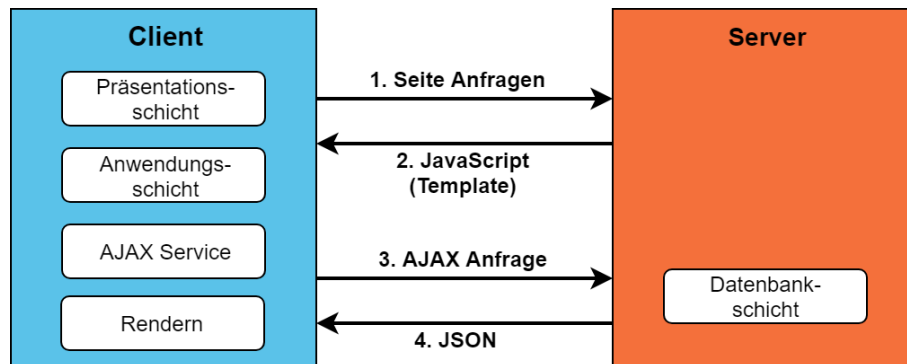
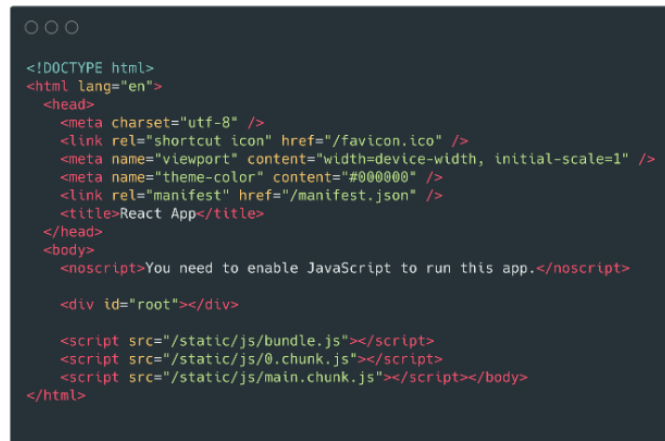


Abbildung 3. HTML Dokument einer React Seite



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="shortcut icon" href="/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <link rel="manifest" href="/manifest.json" />
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

    <script src="/static/js/bundle.js"></script>
    <script src="/static/js/0.chunk.js"></script>
    <script src="/static/js/main.chunk.js"></script></body>
</html>
```

Abbildung 4. HTML Dokument einer React Seite

4 Universal Rendering

4.1 Isomorphic JavaScript

4.2 Virtuelles DOM

4.3 Clientseitige Hydration

4.4 Rendering Ablauf

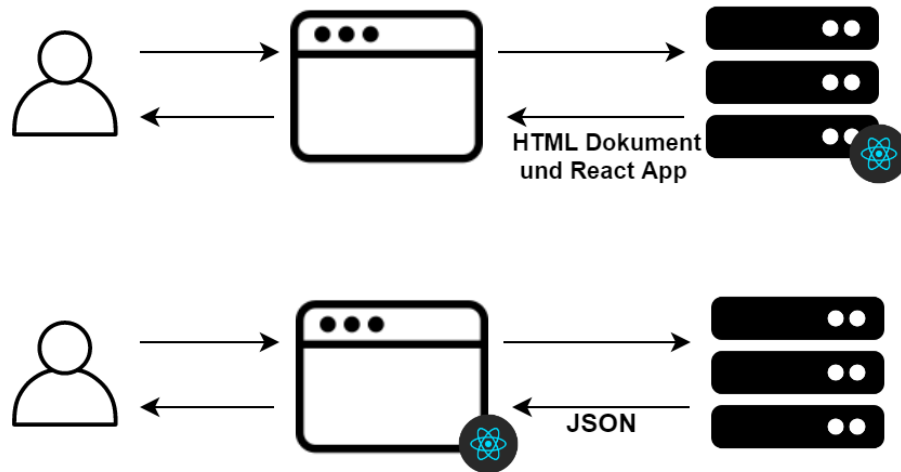


Abbildung 5. HTML Dokument einer React Seite

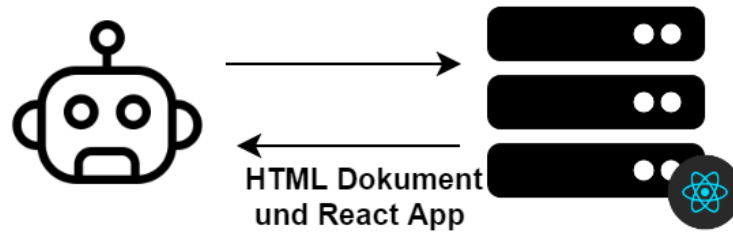


Abbildung 6. HTML Dokument einer React Seite

4.5 Vorteile

4.6 Nachteile

4.7 Alternativen



Abbildung 7. HTML Dokument einer React Seite

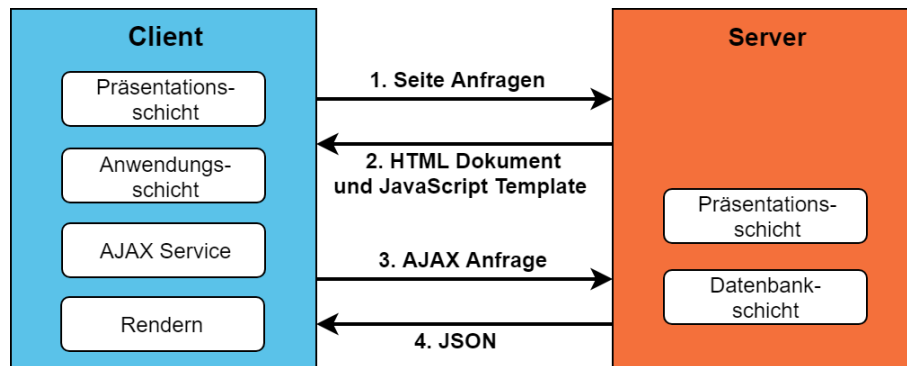


Abbildung 8. HTML Dokument einer React Seite

5 Frameworks

5.1 React und Next.js

5.2 Vue.js und Nuxt.js

5.3 Angular Universal

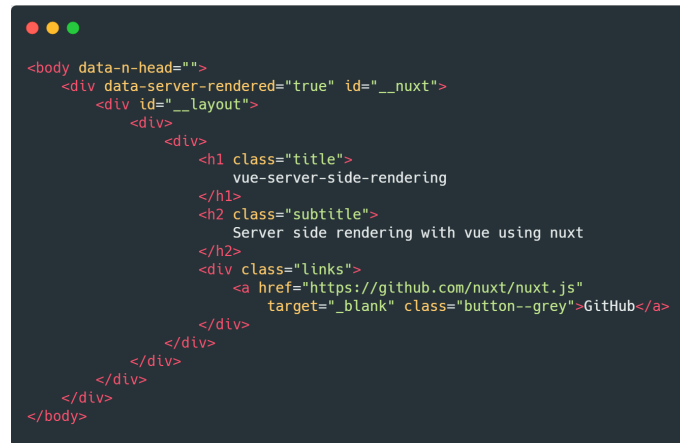


Abbildung 9. HTML Dokument einer React Seite

```

import cowsay from 'cowsay-browser';

function CowsayHi() {
  return <pre>{cowsay.say({ text: 'hi there!' })}</pre>;
}

export default CowsayHi;

```

Abbildung 10. HTML Dokument einer React Seite

6 Universal Rendering in der Praxis


```
import Link from 'next/link';

function Header() {
  return (
    <nav>
      <ul>
        <li>
          <Link prefetch href="/">
            <a>Home</a>
          </Link>
        </li>
        <li>
          <Link prefetch href="/about">
            <a>About</a>
          </Link>
        </li>
        <li>
          <Link prefetch href="/contact">
            <a>Contact</a>
          </Link>
        </li>
      </ul>
    </nav>
  );
}

export default Header;
```

Abbildung 11. HTML Dokument einer React Seite

7 Fazit und Ausblick

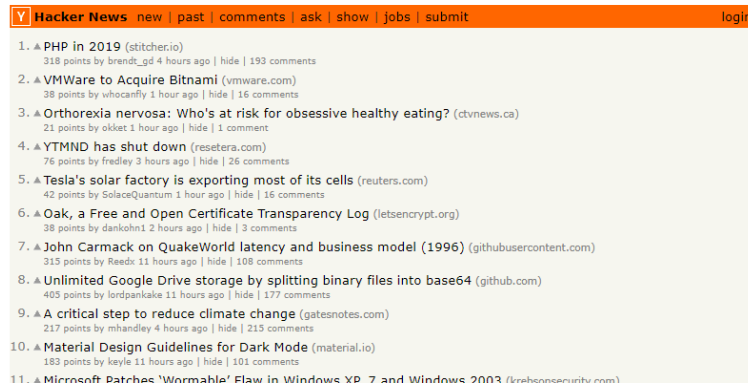


Abbildung 12. HTML Dokument einer React Seite

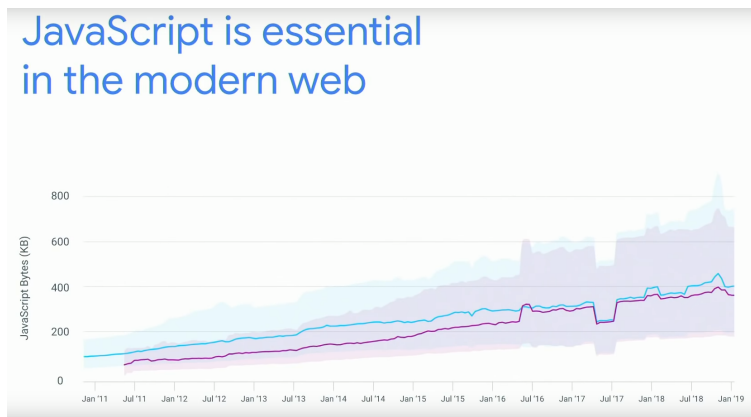


Abbildung 13. HTML Dokument einer React Seite

Literatur

- BeKR09. Steffen Becker, Heiko Koziolk und Ralf Reussner. The Palladio Component Model for Model-driven Performance Prediction. *Journal of Systems and Software*, Band 82, 2009, S. 3–22.
- Jaso16. Maxime Najim Jason Strimpel. *Building Isomorphic JavaScript Apps*. O'Reilly. 2. Auflage, 2016.
- Joel16. Chen Joel. ReactJS SSR Profiling and Caching, Sep 2016.
- uJoh18. Tom Greenaway und John Müller. Deliver search-friendly JavaScript-powered websites. 2018.