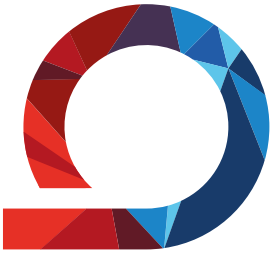


WHZ Westsächsische Hochschule Zwickau
Hochschule für Mobilität

media project

Diplomarbeit

Universelle Applikation auf Basis der React-Bibliothek und des Next.js Frameworks

Westsächsische Hochschule Zwickau (FH)

Fakultät: Physikalische Technik / Informatik

Studiengang: Diplom-Informatiker/in (FH)

eingereicht von:	Yuliya Ivanchenko
Matrikel:	41147
media project Gruppennummer:	M009
geboren am:	15.04.1988 in Charkiw, Ukraine

Betreuer /Einrichtung:	Prof. Dr. W. Golubski, Westsächsische Hochschule Zwickau (FH)
	Prof. Dr. F. Grimm, Westsächsische Hochschule Zwickau (FH)
Abgabedatum:	24.07.2024

Kurzreferat/Abstract

Diese Diplomarbeit untersucht die Entwicklung von isomorphen Webanwendungen unter Verwendung der JavaScript-Bibliothek React und des Frameworks Next.js. In Anbetracht der wachsenden Anforderungen an moderne Webanwendungen bezüglich Nutzerinteraktivität und Suchmaschinenoptimierung (SEO) bieten isomorphe Anwendungen eine vielversprechende Lösung. Sie kombinieren die schnellen Ladezeiten und SEO-Vorteile von Multi-Page-Applications (MPAs) mit der interaktiven Benutzererfahrung von Single-Page-Applications (SPAs) durch serverseitiges Rendering (SSR) und clientseitige Interaktivität.

Zunächst wird die historische Entwicklung der Webtechnologien dargestellt, gefolgt von den technischen Grundlagen und Herausforderungen der isomorphen Anwendungsentwicklung. Ein besonderer Fokus liegt auf dem Vergleich der JavaScript-Frameworks Angular, Vue und React sowie auf der detaillierten Analyse von React und Next.js.

Es wird gezeigt, wie Unternehmen React und Next.js einsetzen, um leistungsfähige, wartbare und SEO-optimierte Webanwendungen zu entwickeln. Diese Technologien werden verwendet, um eine bessere Nutzererfahrung zu bieten.

Abschließend fasst die Arbeit die Forschungsergebnisse zusammen, nimmt eine kritische Bewertung der isomorphen Anwendungen vor und zeigt zukünftige Entwicklungen sowie Forschungspotenziale auf.

Danksagung

Mein besonderer Dank gilt meinem Betreuer, Prof. Dr. W. Golubski, für seine wertvollen Ratschläge, seine kontinuierliche Unterstützung und seine Bereitschaft, all meine Fragen zu beantworten. Seine hilfreichen Tipps und Tricks sowie seine Anleitung zur Strukturierung der Arbeit waren von unschätzbarem Wert.

Übersicht verwendeter Hilfsmittel

ReactJS: Ein JavaScript-Framework zur Erstellung dynamischer Benutzeroberflächen.

Next.js: Ein Framework für serverseitiges Rendering und statische Seitengenerierung in Verbindung mit React.js.

Visual Studio Code: Die integrierte Entwicklungsumgebung (IDE), die für das Schreiben und Testen des Codes verwendet wurde.

ChatGPT: Ein KI-Modell von OpenAI, das verwendet wurde, um Textentwürfe zu verbessern und Fehler zu überprüfen. ChatGPT half dabei, Texte klarer und prägnanter zu formulieren, sprachliche Fehler zu vermeiden, die Präzision zu erhöhen und wertvolle Hinweise für notwendige Anpassungen und weiterführende Überlegungen zu geben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

API: Application Programming Interface

ASP: Active Server Pages

CEO: Chief Executive Officer

CLI: Command Line Interface

CMS: Content Management System

CSS: Cascading Style Sheets

CSP: Content Security Policy

DOM: Document Object Model

ES: ECMAScript

HTML: Hypertext Markup Language

http: Hypertext Transfer Protocol

iOS: iPhone Operating System

ISR: Incremental Static Regeneration

JS: JavaScript

JSON: JavaScript Object Notation

JSX: JavaScript XML

MPA: Multi-Page Application

MVC: Model-View-Controller

MVVM: Model-View-ViewModel

PHP: Hypertext Preprocessor

PPR: Partial Page Reload

PWA: Progressive Web Application

SEO: Search Engine Optimization

SPA: Single-Page Application

SSG: Static Site Generation

SSR: Server-Side Rendering

SSTI: Server-Side Template Injection

TTI: Time to Interactive

UI: User Interface

URI: Uniform Resource Identifier

URL: Uniform Resource Locator

XML: Extensible Markup Language

XSS: Cross-Site Scripting

YUI: Yahoo User Interface

1. Einleitung

1.1. Hintergrund und Motivation

In der heutigen digital vernetzten Welt spielt die Webentwicklung eine zentrale Rolle. Sie beeinflusst maßgeblich, wie Unternehmen operieren, Informationen übermitteln und soziale Interaktionen gestalten. Laut Gartner sollten die weltweiten IT-Ausgaben im Jahr 2023 um 5,1 % steigen. [21] Trotz wirtschaftlicher Unsicherheiten investieren Unternehmen weiterhin stark in digitale Geschäftsinvestitionen, was die zunehmende Bedeutung von Webtechnologien unterstreicht. Diese Technologien verbessern die Effizienz und Benutzerfreundlichkeit von Webanwendungen erheblich.

Ein zentrales Problem vieler Entwickler und Unternehmen ist die Balance zwischen Benutzerinteraktivität und SEO-Freundlichkeit. Trotz der Verbreitung von Frameworks wie React, Angular und Vue bleibt diese Herausforderung bestehen. Hier bietet die isomorphe (universelle) Anwendungsentwicklung eine Lösung, die die Vorteile von Multi-Page-Anwendungen (MPAs) und Single-Page-Anwendungen (SPAs) vereint. Diese Technologie kombiniert die SEO-Vorteile und die schnelle Ladegeschwindigkeit von MPAs mit der Interaktivität und dem nahtlosen Benutzererlebnis von SPAs.

Die Wahl dieses Forschungsthemas basiert auf der Beobachtung, dass viele Entwickler und Unternehmen trotz der breiten Akzeptanz von Frameworks wie React, Angular und Vue Schwierigkeiten haben, ihre Webanwendungen sowohl nutzer- als auch suchmaschinenfreundlich zu gestalten. Isomorphe Anwendungen bieten eine vielversprechende Lösung, indem sie die Vorteile des serverseitigen Renderings mit den dynamischen Möglichkeiten des Clients verbinden.

React.js, von Meta (Facebook) entwickelt, ist ein führendes Werkzeug zur Erstellung dynamischer Benutzeroberflächen. Die Bibliothek ermöglicht Entwicklern, wiederverwendbare UI-Komponenten zu erstellen, was den Aufbau und die Wartung komplexer Benutzeroberflächen vereinfacht. Next.js, ein Framework für React, erweitert diese Möglichkeiten um serverseitiges Rendering und statische Seitengenerierung und ist damit ideal für die Erstellung isomorpher Webanwendungen geeignet.

Neben den technischen Aspekten hat auch die wirtschaftliche Bedeutung von Webanwendungen in den letzten Jahren stark zugenommen. Unternehmen investieren zunehmend in die Entwicklung und Optimierung ihrer Webpräsenz, um im digitalen Wettbewerb bestehen zu können. Dies verdeutlicht die Notwendigkeit, Webanwendungen

zu entwickeln, die nicht nur funktional, sondern auch leistungsfähig und benutzerfreundlich sind.

1.2. Zielsetzung und Aufbau der Arbeit

Der Schwerpunkt dieser Arbeit liegt auf einer umfassenden und detaillierten Untersuchung bestehender Konzepte und Implementierungen. Das Ziel dieser Arbeit besteht nicht in der Entwicklung einer neuen Anwendung, sondern in der Erlangung eines gründlichen Verständnisses und einer sorgfältigen Bewertung bestehender Lösungen. Dieser Ansatz ermöglicht einen fundierten Blick auf das Zusammenspiel und die Effektivität der betrachteten Technologien und trägt damit zur wissenschaftlichen Diskussion des Themas bei.

Die spezifischen Ziele dieser Arbeit sind:

- eine detaillierte Analyse der historischen Entwicklung und der technischen Grundlagen der isomorphen Anwendungsentwicklung.
- ein umfassender Vergleich der JavaScript-Frameworks Angular, Vue und React hinsichtlich ihrer Eignung für die isomorphe Entwicklung.
- eine technische Untersuchung der Herausforderungen und Lösungen bei der Implementierung isomorpher Anwendungen mit besonderem Fokus auf SEO und Leistung.
- die Identifikation von Best Practices und potenziellen Verbesserungen für die Entwicklung und Wartung isomorpher Webanwendungen.

Die Arbeit ist in mehrere Kapitel unterteilt, die jeweils spezifische Aspekte der isomorphen Webentwicklung behandeln:

Einleitung

Darstellung des Themenbereichs, Erläuterung der persönlichen Motivation sowie Aufzeigen von Forschungsdesideraten.

Theoretische Grundlagen

In diesem Kapitel erfolgt eine Erläuterung der historischen Entwicklung sowie der technischen Konzepte der Webtechnologien. Dabei werden zunächst die Ursprünge der Webentwicklung dargelegt, anschließend die Entwicklung von statischen HTML-

Seiten zu modernen Webanwendungen nachgezeichnet und schließlich die Grundlagen von mehrseitigen und einseitigen Anwendungen erläutert.

Begriffserklärung

Im Folgenden erfolgt eine Erläuterung der für das Verständnis der Arbeit relevanten Begriffe und Technologien.

Vergleich der JavaScript-Frameworks

Das vorliegende Kapitel beinhaltet einen detaillierten Vergleich der JavaScript-Frameworks Angular, Vue und React. Dabei werden ihre jeweilige historische Entwicklung, die maßgeblichen Innovationsschritte sowie die technologischen Fundamentaldaten erörtert. Zudem wird der Unterschied zwischen einem Framework und einer Bibliothek dargelegt, um die divergierenden Ansätze der drei Technologien zu verdeutlichen.

Technische Analyse und Herausforderungen

In diesem Kapitel erfolgt eine detaillierte Untersuchung der technischen Aspekte und Herausforderungen bei der Entwicklung isomorpher Anwendungen. Zur Veranschaulichung der theoretischen Erklärungen werden konkrete Beispiele und Code-Snippets präsentiert. Dabei wird ein besonderes Augenmerk auf die Bereiche SEO-Optimierung, Leistung und Unterstützung durch isomorphe Anwendungen und hybride Ansätze gelegt.

Zusätzlicher Abschnitt: Vue.js und Next.js

Im vorliegenden Abschnitt erfolgt eine Erörterung der Integration und Verwendung von Vue.js in Kombination mit Next.js.

Ausblick

Der Ausblick fasst die Forschungsergebnisse zusammen und enthält eine kritische Bewertung der isomorphen Anwendungen. Zudem werden zukünftige Entwicklungen und Forschungspotenziale aufgezeigt, die auf den Ergebnissen dieser Arbeit aufbauen können.

Fazit

Das Fazit bietet eine abschließende Bewertung der Studie und gibt praktische Empfehlungen für die Umsetzung und Optimierung isomorpher Anwendungen.

2. Theoretische Grundlagen

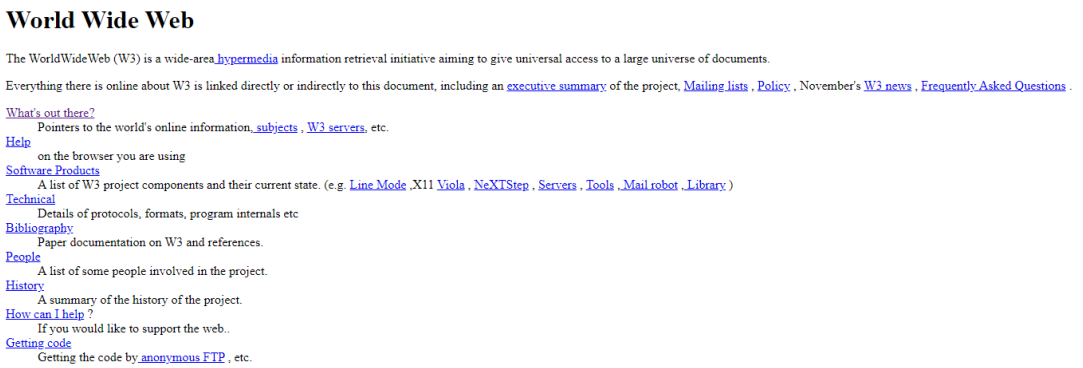
2.1. Webentwicklung: Eine historische Übersicht

Die Entwicklung des Internets und der Web-Technologien hat einen enormen Wandel durchgemacht. Am Anfang waren Webseiten hauptsächlich statische HTML-Seiten. Das heißt, sie waren einfach aufgebaut und hatten kaum interaktive Elemente. Dann kamen JavaScript und serverseitige Skriptsprachen wie PHP und ASP ins Spiel. Mit diesen Technologien konnten Webseiten dynamisch erstellt werden, d. h. sie konnten sich anpassen und auf die Interaktion des Nutzers reagieren. Dadurch wurden Webseiten wesentlich benutzerfreundlicher und interaktiver.

2.1.1. Statische HTML-Seiten (Entstehung 1990er Jahre)

In den frühen 1990er Jahren waren Webseiten statisch. HTML (HyperText Markup Language) wurde verwendet, um einfache Seiten mit Text, Bildern und Links zu erstellen. Jede Änderung erforderte manuelle Bearbeitung des HTML-Codes, was die Benutzererfahrung einfach und wenig interaktiv machte.

1990: Veröffentlichung der ersten Webseite von Tim Berners-Lee.



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

Abbildung 1: Die erste Webseite der Welt. (Quelle: Tim Berners-Lee (20 Dezember 1900): World Wide Web. [Online, Zugriff: 15.06.2024]. Verfügbar unter: https://info.cern.ch/hypertext/WWW/TheProject.html)

Die Vorteile statischer HTML-Seiten lagen in ihrer Einfachheit und den schnellen Ladezeiten.

Jedoch gab es auch Nachteile, wie die fehlende Interaktivität und die Notwendigkeit, jede Änderung durch Bearbeitung und Hochladen neuer Dateien vorzunehmen.

2.1.2. Einführung von JavaScript und serverseitigen Skriptsprachen (Entstehung Mitte 1990er Jahre)

Mit der Einführung von JavaScript Mitte der 1990er Jahre konnten Webseiten interaktiver gestaltet werden. JavaScript ermöglichte die Validierung von Formularen direkt im Browser und das Hinzufügen dynamischer Elemente.

Im Jahr 1995 wurde JavaScript durch Netscape eingeführt, damals hieß die Sprache LiveScript und im Jahr 1996 wurde die erste Version von JavaScript veröffentlicht.

Ebenfalls im Jahr 1995 wurde PHP veröffentlicht, das die Erstellung dynamischer Webseiten ermöglichte.

Ein Jahr später, 1996, führte Microsoft ASP (Active Server Pages) ein.

Beispiele für die Nutzung dieser Technologien sind PHP-basierte Foren und Content-Management-Systeme (CMS) wie WordPress, die es den Benutzern ermöglichten, Inhalte dynamisch zu erstellen und zu verwalten. ASP wurde häufig für dynamische Unternehmenswebsites in Microsoft-Umgebungen verwendet.

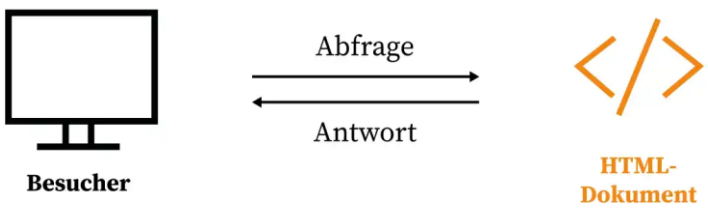


Abbildung 2: Statische HTML-Seite. (Quelle: Lautenschlager (22 Mai 2022): Statische Websites: die simpelsten Websites. [Online, Zugriff: 15.06.2024]. Verfügbar unter: https://www.lautenschlager.de/blog/statische-websites-die-simpelsten-websites/)

2.2. Multi-Page-Anwendungen (MPAs) (Entstehung Späte 1990er bis frühe 2000er Jahre)

Multi-Page Applications (MPAs) stellen eine spezielle Form traditioneller Webanwendungen dar, bei denen jede Benutzeraktion die Generierung und Ladung einer neuen HTML-Seite auf dem Server zur Folge hat. Diese Architektur erweist sich insbesondere im Hinblick auf die Suchmaschinenoptimierung als vorteilhaft, da Suchmaschinen jede Seite problemlos indizieren können. Dies macht MPAs insbesondere für Websites mit einem hohen Anspruch an die Online-Sichtbarkeit, wie etwa E-Commerce-Websites oder Blogs, zu einer idealen Wahl. Die klare Trennung von Front-End und Back-End in MPAs bedingt jedoch verlängerte Entwicklungszeiten und eine Verlangsamung des Benutzererlebnisses, da jede Navigation zwischen Seiten ein vollständiges Neuladen erfordert.

Amazon beispielsweise nutzt die MPA-Architektur, um jede Produktseite separat zu laden, wodurch sich die SEO-Optimierung verbessert und eine detaillierte Analyse des Benutzerverhaltens ermöglicht wird. Auch E-Commerce-Websites wie eBay verlassen sich auf MPAs, um ihre umfangreichen Produktkataloge effizient anzuzeigen und zu verwalten.

Beispielsweise nutzt Amazon die MPA-Architektur, um jede Produktseite separat zu laden, was die SEO-Optimierung verbessert und eine detaillierte Analyse des Nutzerverhaltens ermöglicht. E-Commerce-Websites wie eBay verlassen sich ebenfalls auf MPAs, um ihre umfangreichen Produktkataloge effizient anzuzeigen und zu verwalten.

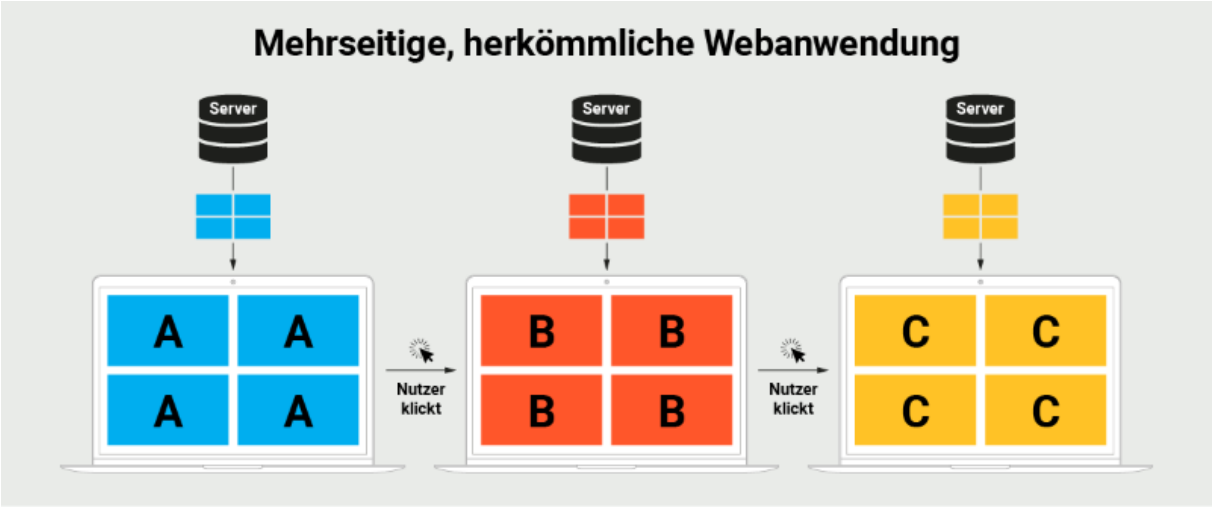


Abbildung 3: Multi-Page-Anwendungen. (Quelle: Jan Schulte (27 Mai 2024): Single Page Applications (SPA): Erklärung, Vorteile und Beispiele. [Online, Zugriff: 15.06.2024]. Verfügbar unter: https://web.archive.org/web/20240101000000*/https://www.magnolia-cms.com/de_DE/blog/all-about-single-page-applications.html)

2.3. Single-Page-Anwendungen (SPAs) (Entstehung Frühe 2000er Jahre)

Bei der initialen Ladung der Anwendung durch SPAs erfolgt eine dynamische Aktualisierung des Inhalts, ohne dass eine erneute Ladung der Seite erforderlich ist. Dies resultiert in einem reibungsloseren und schnelleren Benutzererlebnis nach dem initialen Download. SPAs eignen sich insbesondere für Anwendungen, die eine hohe Interaktivität sowie Echtzeit-Updates erfordern. Diesbezüglich sind insbesondere soziale Netzwerke sowie Kollaborationsplattformen zu nennen. Des Weiteren erweisen sich SPAs als vorteilhaft für mobile Anwendungen, da derselbe Code für Web- und Mobilversionen zum Einsatz kommen kann.

Ein exemplarisches Beispiel für eine SPA ist Gmail, das sich durch eine hohe Geschwindigkeit sowie eine exzellente Benutzererfahrung auszeichnet. Auch Google Mail war einer der ersten, wer im Jahr 2004 SPAs eingeführt hat. Die SPA-Architektur findet bei Netflix Anwendung, um ein unterbrechungsfreies Streaming zu ermöglichen. Dazu werden große Datenmengen effizient gleichzeitig an zahlreiche Benutzer übertragen.

Allerdings weisen SPAs auch Nachteile auf, insbesondere im Hinblick auf die Suchmaschinenoptimierung (SEO). Da Inhalte dynamisch geladen werden, ist es für Suchmaschinen schwierig, alle Inhalte zu indizieren, was die Sichtbarkeit der Website

beeinträchtigen kann. Zudem sind SPAs stark auf JavaScript angewiesen, was bei unsachgemäßer Handhabung ein potenzielles Sicherheitsrisiko darstellen kann.

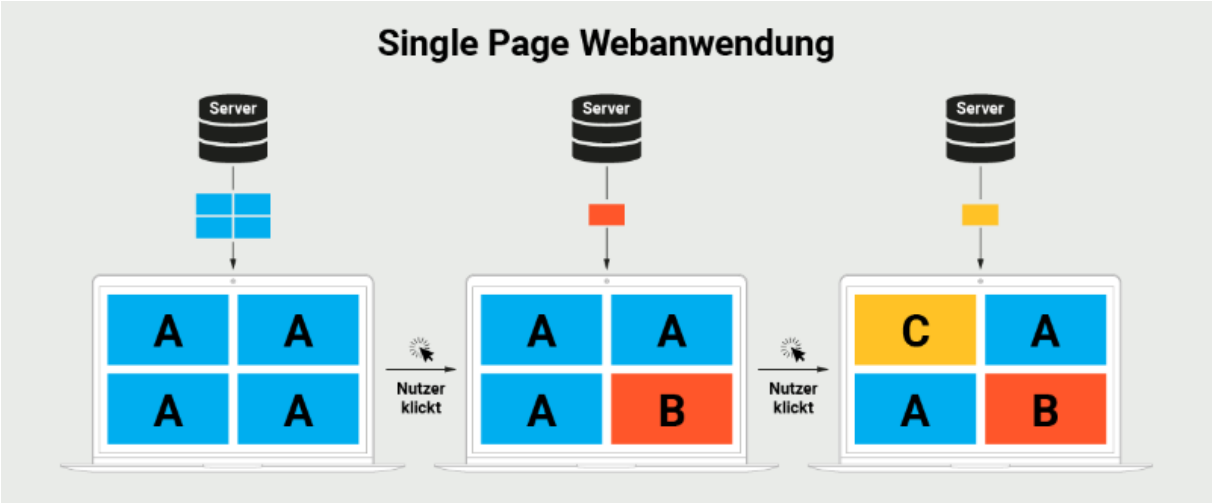


Abbildung 4: Single-Page-Anwendungen. (Quelle: Jan Schulte (27 Mai 2024): Single Page Applications (SPA): Erklärung, Vorteile und Beispiele. [Online, Zugriff: 15.06.2024]. Verfügbar unter: https://web.archive.org/web/20240101000000*/https://www.magnolia-cms.com/de_DE/blog/all-about-single-page-applications.html)

2.4. Isomorphe Anwendungen

Isomorphe Anwendungen kombinieren die Vorteile von Multi-Page-Anwendungen (MPA) und Single-Page-Anwendungen (SPA), indem sie serverseitiges Rendering (SSR) und clientseitige Interaktivität bieten. Der ursprüngliche Seiteninhalt wird auf dem Server gerendert, wodurch sich sowohl die Ladezeiten als auch die SEO-Leistung verbessern. Nach dem initialen Laden übernimmt die Client-Anwendung, was eine hohe Interaktivität und schnelle Benutzerreaktionen ermöglicht.

Ein Beispiel für eine isomorphe Anwendung ist die Verwendung von Next.js für React oder Nuxt.js für Vue.js. Diese Architekturen kombinieren sowohl serverseitiges Rendering als auch statische Seitengenerierung. Plattformen wie Nike nutzen Next.js in Kombination mit React, um ihre Webanwendungen zu optimieren und ein nahtloses Benutzererlebnis zu generieren.

Der Hauptvorteil isomorpher Anwendungen liegt in der verbesserten SEO-Leistung und den schnelleren initialen Ladezeiten, da die Inhalte auf dem Server gerendert werden. Dies resultiert in einer gesteigerten Benutzerzufriedenheit bei gleichzeitiger

Reduktion der Absprungraten. Nach dem initialen Laden übernimmt die Client-Anwendung die Kontrolle, wodurch ein hohes Maß an Interaktivität gewährleistet wird.

Allerdings sind auch Probleme zu verzeichnen. Die Entwicklung und Wartung isomorpher Anwendungen ist mit Schwierigkeiten verbunden, da hierfür Kenntnisse sowohl der Server- als auch der Client-Programmierung erforderlich sind. Außerdem kann die Serverlast steigen, da der Server die initialen Seiten rendert, was zu höheren Betriebskosten führen kann. Es gibt auch potenzielle Sicherheitsrisiken, da mehr serverseitige Logik exponiert wird, und die Fehlersuche kann schwieriger sein, da Fehler sowohl auf der Server- als auch auf der Client-Seite auftreten können

Der erste dokumentierte Einsatz von isomorphen Webanwendungen stammt aus den Jahren 2011-2013:

- 2011: Charlie Robbins, CEO von Nodejitsu, bezeichnete JavaScript als isomorphe Sprache. Dies war ein wichtiger Schritt zur Popularisierung des Konzepts.
- 2013: Spike Brehm von Airbnb machte den Begriff "isomorphes JavaScript" populär und betonte damit die Möglichkeit, denselben Code sowohl auf dem Client als auch auf dem Server auszuführen.
- 2016: Veröffentlichung von Next.js, einem Framework für React.

Dieser Zeitraum markiert den Beginn der Entwicklung und Verwendung isomorpher Webanwendungen, insbesondere mit der Verbreitung von Frameworks wie React, die die Verwendung von JavaScript für server- und clientseitiges Rendering ermöglichen. React selbst wurde im Mai 2013 von Facebook veröffentlicht und ebnete den Weg für isomorphe Anwendungen.

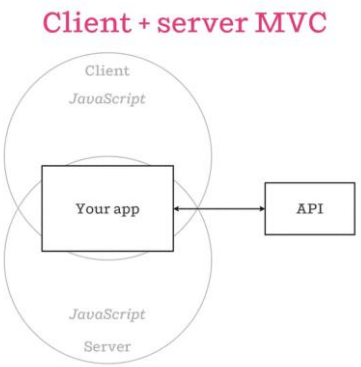


Abbildung 5: Isomorphe Anwendungen. (Quelle: AirbnbEng (19 Mai 2017): Isomorphic JavaScript: The Future of Web Apps. [Online, Zugriff: 15.06.2024]. Verfügbar unter: https://medium.com/airbnb-engineering/isomorphic-javascript-the-future-of-web-apps-10882b7a2ebc)

2.5. Zusammenfassung

Die Entscheidung für eine der Optionen MPA, SPA oder isomorphe Anwendungen ist in erster Linie von den spezifischen Anforderungen und Zielen des Projekts abhängig. MPAs erweisen sich als ideale Wahl für Projekte, bei denen die Suchmaschinenoptimierung (SEO) sowie die Präsentation umfangreicher Inhalte im Mittelpunkt stehen. Dies trifft insbesondere auf E-Commerce-Websites und Blogs zu. SPAs sind für Anwendungen geeignet, die nach dem ersten Laden eine hohe Interaktivität und Geschwindigkeit erfordern. Dies kann beispielsweise bei sozialen Netzwerken oder Echtzeitanwendungen der Fall sein. Isomorphe Apps kombinieren die besten Elemente beider Architekturen und eignen sich für komplexe Webanwendungen, die sowohl SEO-Optimierung als auch dynamische Interaktivität erfordern.

Merkmal	Multi-Page Applications (MPA)	Single-Page Applications (SPA)	Isomorphe Anwendungen
SEO	Hervorragend, da jede Seite separat indiziert wird	Problematisch, da Inhalte dynamisch geladen werden	Sehr gut, da initialer Inhalt serverseitig rendert wird
Ladezeit	Langsamere Ladezeiten, da jede Seite neu geladen wird	Initial langsamer, danach schnell	Schnellere initiale Ladezeiten durch SSR
Interaktivität	Weniger interaktiv, da Seiten neu geladen werden	Sehr interaktiv, keine vollständige Seitenneuladung	Hohe Interaktivität nach dem initialen Laden
Komplexität der Entwicklung	Einfacher, da klare Trennung von Front- und Backend	Mittel, erfordert umfassende JavaScript-Kenntnisse	Höher, erfordert Kenntnisse in server- und clientseitiger Programmierung
Serverlast	Niedrig, da nur Seitenanfragen bearbeitet werden	Niedrig, da die meisten Interaktionen clientseitig sind	Höher, da der Server initiale Seiten rendert
Sicherheitsrisiken	Geringer, traditionellere Sicherheitsmechanismen	Höher, JavaScript-abhängige Sicherheitslücken	Potentiell höher, da mehr serverseitige Logik exponiert wird
Fehlersuche und Debugging	Einfacher, da nur serverseitiger Code	Komplexer, da clientseitig	Sehr komplex, da sowohl server- als auch clientseitiger Code
Beispiele	Amazon, eBay	Gmail, Netflix	Nike (Next.js mit React), Airbnb (Rendr mit React)

Abbildung 6: MPA, SPA, Isomorphe App Merkmale

3. Begriffserklärung

3.1. Isomorphe Applikation

In der Welt der JavaScript-Entwicklung bezieht sich der Begriff "isomorph" (manchmal auch "universal" oder "shared rendering" genannt) auf eine Anwendung, die sowohl auf dem Server als auch auf dem Client ausgeführt werden kann. Isomorphe Anwendungen können ihren Code sowohl auf der Server- als auch auf der Clientseite wiederverwenden, was mehrere Vorteile bietet, insbesondere in Bezug auf die Leistung und die Benutzerfreundlichkeit. Der Begriff "isomorph" kommt daher, dass derselbe Code in unterschiedlichen Umgebungen (Server und Browser) "gleich" funktioniert.

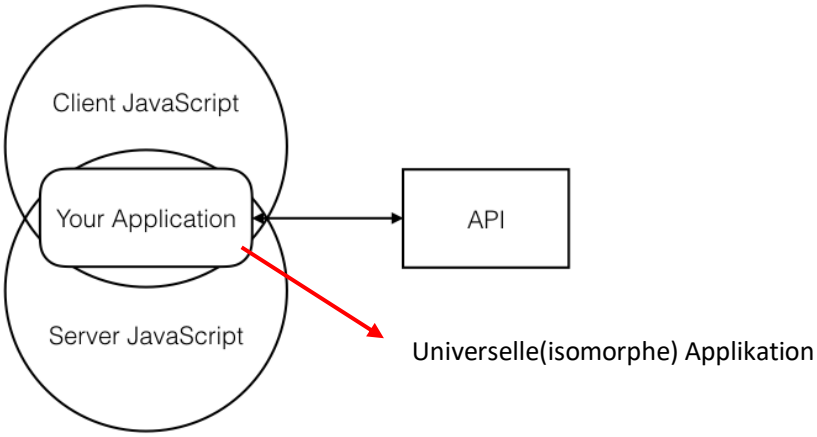


Abbildung 7: Beispiel einer universellen(isomorphen) Anwendung. (Quelle Köbler, Niko (28 April 2016): Isomorphe JavaScript-Webanwendungen mit Java (EE) und React.JS. [Online, Zugriff: 15.06.2024]. Verfügbar unter: https://jaxenter.de/isomorphe-javascript-webanwendungen-mit-java-ee-und-react-js-38266)

3.2. Hydreielogik

Hydrierlogik (auch "Hydration" genannt) bezieht sich auf den Prozess, bei dem ein bereits serverseitig gerendertes HTML-Dokument mit interaktiven Funktionen ausgestattet wird, indem JavaScript auf dem Client ausgeführt wird.

3.3. ReactJS

React ist eine beliebte Open-Source-JavaScript-Bibliothek, die von Facebook entwickelt wurde und für die Erstellung von Benutzeroberflächen, insbesondere für Single-

Page-Anwendungen, verwendet wird. Sie ermöglicht es Entwicklern, wiederverwendbare UI-Komponenten zu erstellen und zu verwalten, was die Entwicklung komplexer, interaktiver Web- und mobiler Anwendungen vereinfacht.

3.4. Next.js

Next.js ist ein beliebtes Open-Source-Framework, das auf React basiert, einer JavaScript-Bibliothek zur Erstellung von Benutzeroberflächen. Es wurde entwickelt, um die Entwicklung von Single-Page-Anwendungen (SPAs), statischen Websites und serverseitig gerenderten Anwendungen zu vereinfachen.

3.5. Framework

Ein Framework stellt eine umfassende Umgebung dar, welche einen strukturierten und einheitlichen Ansatz für die Softwareentwicklung bietet. Es beinhaltet eine Vielzahl an eingebauten Werkzeugen und Funktionen, welche Entwicklern dabei behilflich sind, Anwendungen effizient zu erstellen. Frameworks bieten eine vordefinierte Architektur und folgen bestimmten Entwurfsprinzipien und Mustern, welche die Entwicklung erleichtern und standardisieren.

3.6. Bibliothek

Eine Bibliothek ist eine Sammlung vorgefertigter Funktionen oder Klassen, die bestimmte Aufgaben in einer Anwendung erleichtern. Im Gegensatz zu einem Framework bietet eine Bibliothek keine umfassende Struktur oder Architektur, sondern wird nach Bedarf in die Anwendung integriert. Die Entwickler haben die Freiheit, verschiedene Bibliotheken auszuwählen und zu kombinieren, um bestimmte Anforderungen zu erfüllen.

3.7. Crawler

Ein Suchmaschinen-Crawler, auch als Webcrawler, Bot, Spider oder Searchbot bekannt, ist ein Softwareprogramm, das von Suchmaschinen verwendet wird, um das Internet zu durchsuchen und Webseiteninhalte wie Texte, Bilder und Videos zu

sammeln. Diese Informationen werden dann indexiert und in einer Datenbank gespeichert, damit die Suchmaschine schnell relevante Suchergebnisse liefern kann. Webseiten können dem Crawler mithilfe einer "robots.txt"-Datei Anweisungen geben, welche Bereiche durchsucht werden dürfen und welche nicht.

4. Vergleich der JavaScript-Frameworks: Angular, Vue und React

Die JavaScript-Frameworks Angular 2+, React und Vue.js gehören zu den beliebtesten Frameworks für die Entwicklung von Benutzeroberflächen. Sie werden von unterschiedlichen Communities und Unternehmen unterstützt und weisen jeweils spezifische Stärken und Schwächen auf.

4.1 Geschichte Angular/Vue/React

Angular

Angular wurde ursprünglich im Jahr 2010 unter dem Namen AngularJS von Google entwickelt und veröffentlicht. Es handelte sich hierbei um eines der ersten weit verbreiteten Frameworks für Single Page Applications (SPAs). Im Jahr 2016 erfolgte eine grundlegende Überarbeitung, welche die Veröffentlichung von Angular 2 nach sich zog. Ziel war es, moderne Webstandards und Leistungsanforderungen besser zu unterstützen. Seitdem durchläuft Angular eine kontinuierliche Weiterentwicklung.

Die integrierten Funktionen von Angular, darunter bidirektionale, Dependency Injection und eine umfassende Kommandozeilenschnittstelle (CLI), eignen sich in besonderem Maße für die Umsetzung großer und komplexer Anwendungen.

Die neueste Version von Angular, Version 18, wurde im Mai 2024 veröffentlicht. Diese Version beinhaltet wesentliche Verbesserungen, darunter die experimentelle zonenlose Änderungserkennung, stabile Material-3-Komponenten sowie erweiterte serverseitige Rendering-Funktionen mit i18n-Unterstützung. Darüber hinaus wurden die Leistungsfähigkeit und Funktionalität durch die integrierte Unterstützung für Asynchronität und partielle Hydrationsstrategien weiter optimiert.

Zu den großen Unternehmen, die Angular verwenden, gehören Google, Microsoft, Forbes

React

React wurde im Jahr 2013 von Facebook veröffentlicht und entwickelt. Es handelt sich dabei um eine Bibliothek zur Erstellung dynamischer und interaktiver Benutzeroberflächen, deren Entwicklung darauf abzielt, die damit verbundenen Prozesse zu vereinfachen. Ein wesentlicher Bestandteil von React ist das virtuelle DOM, welches die

Leistung optimiert, indem nur diejenigen Teile der Benutzeroberfläche neu gerendert werden, die sich tatsächlich ändern.

Im Jahr 2017 führte Facebook React Fiber ein, eine komplette Überarbeitung der internen Architektur von React, welche eine signifikante Steigerung der Performance zur Folge hatte.

Die im Jahr 2022 veröffentlichte aktuelle Version von React, React 18, beinhaltet wesentliche Verbesserungen, darunter die Möglichkeit des Concurrent Renderings sowie automatischer Batch-Updates. Dadurch konnten die Leistung und Benutzerfreundlichkeit von React weiter optimiert werden.

Zu den Unternehmen, die React in großem Umfang nutzen, zählen unter anderem Meta (Facebook), Netflix, Airbnb.

Vue.js

Vue wurde von dem ehemaligen Google-Mitarbeiter Evan You entwickelt und erstmals im Februar 2014 veröffentlicht. Der Fokus bei der Entwicklung von Vue lag auf der Kombination der besten Eigenschaften von Angular und React. Gleichzeitig sollte eine einfache Lernkurve und hohe Flexibilität gewährleistet werden.

Die aktuelle Version von Vue 3 wurde im September 2020 veröffentlicht und weist zahlreiche Leistungsverbesserungen sowie eine vollständige TypeScript-Unterstützung auf. Vue verfügt über eine wachsende Community und ist insbesondere in Asien weit verbreitet.

Zu den Unternehmen, die Vue nutzen, gehören unter anderem Alibaba, Baidu, Xiaomi.

4.2 Kernentwicklung

Die Technologien React und Angular erfahren weiterhin Unterstützung durch die beiden größten Technologieunternehmen der Welt, Meta (Facebook) und Google. Diese Unterstützung gewährleistet Stabilität und kontinuierliche Entwicklung. Umfangreiche Teams und Ressourcen sorgen für regelmäßige Updates, eine umfassende Dokumentation und ein breites Ökosystem von Tools und Bibliotheken. Dies gibt Unternehmen und Entwicklern die Gewissheit, dass diese Plattformen langfristig unterstützt werden.

Angular eignet sich in besonderem Maße für die Entwicklung großer und komplexer Unternehmensanwendungen. Es bietet eine Vielzahl an Funktionen sowie unmittelbar einsatzbereite Lösungen, wodurch sich Vorteile für die Softwareentwicklung in großen Teams und in strukturierten Projekten ergeben. Google stellt regelmäßige Updates zur Verfügung und bietet zudem einen starken Community-Support. Darüber hinaus verfügt Angular über eine umfangreiche Anzahl offizieller Komponenten und Bibliotheken, wie Angular Material für UI-Komponenten und NgRx für das Zustandsmanagement.

Stand Juli 2024:
GitHub-Sterne: über 95,3 Tausend [40]
Mitwirkende: über 1930 [40]

React zeichnet sich durch eine hohe Flexibilität aus und wird von einer großen Community unterstützt. Es eignet sich für eine breite Palette von Projekten, von kleinen bis hin zu großen Anwendungen, und ermöglicht Entwicklern die effiziente Entwicklung interaktiver Benutzeroberflächen. Meta (Facebook) fördert aktiv die Weiterentwicklung und organisiert regelmäßig Veranstaltungen wie die React Conf, um die Community zu stärken. Darüber hinaus existiert eine Vielzahl an Bibliotheken von Drittanbietern, welche React erweitern. Dazu zählen beispielsweise Next.js für das serverseitige Rendering sowie React Native für die Entwicklung mobiler Apps.

Stand Juli 2024:
GitHub-Sterne: über 216 Tausend [49]
Mitwirkende: über 1650 [49]

Die Entwicklung von Vue wird hauptsächlich von einem kleinen Team vorangetrieben, wobei die Finanzierung durch eine Gemeinschaft von Entwicklern erfolgt, die sich durch Spenden und Sponsoring finanzieren. Der Schöpfer von Vue.js, Evan You, begann das Projekt als persönliches Nebenprojekt und konnte sich durch die Unterstützung von Plattformen wie Patreon und direkten Sponsoren finanzieren. Dies ermöglichte ihm eine vollständige Konzentration auf die Entwicklung von Vue.js.

Die Entwicklung und Wartung von Vue.js ist in erheblichem Maße von der Unterstützung der Community abhängig. Die Finanzierung der Entwicklung und Wartung von Vue.js erfolgt durch Sponsoren wie Open Collective und GitHub. Die Sponsoren leisten

finanzielle Beiträge, die es den Entwicklern ermöglichen, neue Funktionen zu entwickeln und die Qualität der Software sicherzustellen. Zu den Sponsoren zählen auch größere Unternehmen wie Chrome, FactSet und Frontend Masters, die ebenfalls erhebliche Mittel beitragen. Diese Struktur hat dazu beigetragen, dass Vue eine flexible und leicht zugängliche Lösung für viele Entwickler darstellen kann und gleichzeitig eine kontinuierliche Entwicklung gewährleistet ist.

Stand Juli 2024:
GitHub-Sterne Vue 2: über 207 Tausend [11]
Mitwirkende: über 360 [11]
GitHub-Sterne Vue 3: über 45,9 Tausend [50]
Mitwirkende: über 490 [50]

4.3 Technologie dahinter

Die Technologien Angular, React und Vue.js weisen eine Reihe von wesentlichen Unterschieden auf, die bei der Entscheidung für eine bestimmte Lösung in unterschiedlichen Projekten zu berücksichtigen sind.

Angular ist ein umfassendes Framework, welche von Google entwickelt und gepflegt wird. Die Plattform verfügt über eine Vielzahl integrierter Funktionen, welche die Entwicklung umfangreicher und komplexer Anwendungen signifikant erleichtern. Zu den grundlegenden Technologien und Konzepten, auf denen Angular basiert, gehören:

- TypeScript stellt eine weitere wichtige Technologie in diesem Kontext dar. Die Entwicklung von Angular basiert in erster Linie auf der Programmiersprache TypeScript. Hierbei handelt es sich um eine statisch typisierte, erweiterte Version von JavaScript, welche die Wartbarkeit und Fehlererkennung verbessert.
- Angular verwendet das Model-View-ViewModel (MVVM)-Muster, welches eine Trennung von Logik und Darstellung ermöglicht.
- Dependency Injection. Dieses Muster findet häufig Anwendung, um Abhängigkeiten zwischen Komponenten zu verwalten. Dies führt zu einer verbesserten Testbarkeit und Modularität.
- Angular verwendet Zonen zur Erkennung von Änderungen, sodass Änderungen am Datenmodell automatisch in der Ansicht widergespiegelt werden.

- RxJS ist eine reaktive Programmierbibliothek, die asynchrone Datenströme und Ereignisse effizient verarbeiten kann.
- Angular verwendet standardmäßig bidirektionale Datenbindung, was bedeutet, dass Änderungen an der Benutzeroberfläche automatisch im Datenmodell reflektiert werden und umgekehrt. Angular bietet auch unidirektionale Datenflussmechanismen, aber die bidirektionale Datenbindung ist ein herausragendes Merkmal von Angular. Bidirektionale Datenbindung reduziert den Bedarf an Boilerplate Code für die Synchronisierung von UI und Modell, was es für komplexe Unternehmensanwendungen attraktiv macht. Die intensive Nutzung von TypeScript verbessert auch die Wartbarkeit und Fehlererkennung im Code.

Angular bietet eine umfassende Lösung für die Entwicklung großer Unternehmensanwendungen, mit integrierten Funktionen wie bidirektionaler Datenbindung und Dependency Injection. Allerdings können die steile Lernkurve und die Komplexität für Einsteiger abschreckend sein. Die Vielzahl an eingebauten Funktionen kann zudem zu einer höheren initialen Komplexität führen, was die Einarbeitung erschwert. Ein besonderes Merkmal von Angular ist die vollständige Toolchain, die von Anfang an alle notwendigen Werkzeuge für die Entwicklung komplexer Anwendungen bereitstellt, wie beispielsweise Angular CLI und Angular Material.

React ist eine von Meta (Facebook) entwickelte JavaScript-Bibliothek, die für die Erstellung von Benutzeroberflächen konzipiert ist. Im Gegensatz zu Angular zeichnet sich React durch eine höhere Flexibilität aus, wobei jedoch zusätzliche Bibliotheken erforderlich sind, um bestimmte Funktionen zu implementieren. Zu den Kerntechnologien und -konzepten von React gehören:

- React ist eine vollständig in JavaScript geschriebene Bibliothek, welche die ES6- und höhere verwendet. React unterstützt TypeScript vollständig, was bedeutet, dass Entwickler React-Komponenten mit klarer Typisierung erstellen können.
- React verwendet ein virtuelles DOM, um eine effiziente Änderungsverwaltung zu gewährleisten. Bei einer Zustandsänderung einer Komponente erstellt React eine virtuelle Kopie des DOM und vergleicht sie mit der aktuellen Version. Die Aktualisierung erfolgt lediglich für die Teile des DOM, die sich tatsächlich geändert haben, wodurch eine Leistungssteigerung erzielt wird.

- Die JavaScript-Syntaxerweiterung JSX (JavaScript XML) ermöglicht die Verwendung von HTML-ähnlicher Syntax direkt im JavaScript-Code. Dies verbessert die Lesbarkeit und Verständlichkeit des Codes erheblich, insbesondere bei der Entwicklung von Benutzeroberflächen mit React. JSX erlaubt es Entwicklern, die Struktur von Komponenten klar und intuitiv zu definieren, wodurch der Entwicklungsprozess beschleunigt und vereinfacht wird. Obgleich JSX eine eigenständige Syntax aufweist, können Dateien, die JSX nutzen, dennoch die Endung .js aufweisen. Moderne Build-Tools und Transpiler wie Babel transformieren JSX-Code in Standard-JavaScript-Code, den die Browser interpretieren können. Somit besteht keine Notwendigkeit, eine andere Dateierweiterung zu verwenden. Dennoch kann die Endung .jsx verwendet werden, um eine klarere Zuordnung zu treffen und anzugeben, dass die Datei JSX-Syntax enthält.
- React verwendet ausschließlich einen unidirektionalen Datenfluss. Das bedeutet, dass Daten immer von übergeordneten Komponenten an untergeordnete Komponenten weitergegeben werden. Wenn untergeordnete Komponenten Informationen an die übergeordneten Komponenten zurückgeben müssen, geschieht dies über Callbacks. Diese Architektur erleichtert die Verfolgung des Datenflusses und vereinfacht die Fehlersuche. Der unidirektionale Datenfluss ermöglicht eine klare und vorhersehbare Verwaltung des Anwendungsstatus, was die Wartung und Erweiterung der Anwendung erleichtert.
- Hooks in React stellen Funktionen dar, welche den Zustand sowie weitere React-Merkmale in funktionalen Komponenten nutzen. Vor Hooks war der Zugriff auf die Methoden „state" und „lifecycle" lediglich im Kontext von Klassenkomponenten möglich. Dank der Hooks können funktionale Komponenten nun auf diese Funktionen zugreifen, was den Code häufig einfacher und verständlicher gestaltet.

React's virtuelles DOM und JSX bieten eine hohe Performance und Flexibilität. Die große Community und die Unterstützung durch Facebook sind weitere Pluspunkte. Durch die Verwendung von Hooks kann der Code zudem übersichtlicher und wartungsfreundlicher gestaltet werden. Da React sich hauptsächlich auf die View-Schicht konzentriert, müssen Entwickler zusätzliche Bibliotheken für Routing und State Management verwenden, was zu einer höheren Komplexität führen kann. Diese Modularität erfordert, dass Entwickler selbst entscheiden, welche Bibliotheken und Tools sie verwenden wollen, was die Einstiegshürde erhöhen kann. React's Ökosystem ist sehr

groß und bietet zahlreiche Bibliotheken und Tools, wie beispielsweise Redux für das State Management und Next.js für serverseitiges Rendering

Vue.js wird häufig als progressive JavaScript-Framework bezeichnet, die von Evan You entwickelt wurde und die besten Eigenschaften von Angular und React in sich vereint. Dabei ist er jedoch ebenso einfach zu erlernen und flexibel zu verwenden. Zu den Kerntechnologien und Konzepten von Vue.js gehören:

- Vue zeichnet sich durch eine hohe Flexibilität aus, da es sowohl mit JavaScript als auch mit TypeScript verwendet werden kann. Diese Vielseitigkeit macht es für Entwickler mit unterschiedlichem Hintergrund attraktiv, da sie dasjenige Werkzeug wählen können, das ihren individuellen Anforderungen am besten entspricht.
- Vue verwendet ein reaktives Datenbindungssystem, welches eine automatische Aktualisierung der Benutzeroberfläche bei Änderungen der zugrunde liegenden Daten gewährleistet. Dies ermöglicht eine einfache Synchronisation des Anwendungszustandes mit der Benutzeroberfläche sowie eine Optimierung der Benutzerinteraktion.
- Die Definition von Komponenten in Vue kann in einer einzigen Datei erfolgen, wobei diese als „Single-File Components“ bezeichnet werden. Diese Datei umfasst HTML, JavaScript und CSS. Die Modularität des Systems gewährleistet eine übersichtliche und praktische Struktur, da alle relevanten Komponenten einer Datei zugeordnet werden. Die deklarative Syntax der Vue-Templates erlaubt eine einfache Erstellung und Interpretation von Views. Die Konzentration der Entwickler auf die Anwendungslogik wird dadurch gewährleistet, dass sie sich nicht mit der direkten Manipulation des DOM befassen müssen.
- Die Weitergabe von Daten von übergeordneten zu untergeordneten Komponenten erfolgt in Vue mittels Props. Diese unidirektionale Datenflussarchitektur stellt sicher, dass Änderungen in der übergeordneten Komponente automatisch in der untergeordneten Komponente reflektiert werden, wodurch eine konsistente Datenverwaltung gewährleistet wird.
- Die Vue-Direktive des V-Modells ermöglicht eine einfache und automatische Synchronisation zwischen dem Datenzustand und der Benutzeroberfläche, was insbesondere bei Formularen und Benutzereingaben von Vorteil ist. Diese bidirektionale Datenbindung kann ebenfalls in benutzerdefinierten Komponenten Anwendung

finden, um eine reibungslose Interaktion zwischen Zustand und Benutzeroberfläche zu gewährleisten.

Vue.js zeichnet sich durch seine einfache Lernkurve und hohe Flexibilität aus, was es besonders für kleinere bis mittelgroße Projekte attraktiv macht. Die Single-File Components und die klare Trennung von HTML, CSS und JavaScript innerhalb dieser Komponenten machen es für Entwickler besonders zugänglich. Diese Struktur ermöglicht es Entwicklern, sich schnell in das Framework einzuarbeiten und effiziente Lösungen zu entwickeln.

Allerdings können die kleinere Community und die geringere Unternehmensunterstützung im Vergleich zu Angular und React Einschränkungen darstellen. Vue.js hat weniger offizielle Bibliotheken und Erweiterungen, was die Integration in größere Projekte manchmal erschwert. Dennoch bietet Vue eine progressive Integration, was bedeutet, dass es leicht in bestehende Projekte eingebunden werden kann, ohne diese komplett umzustellen.

Vue.js erweist sich insbesondere für kleinere bis mittelgroße Projekte als überaus nützlich und findet häufig Verwendung in kreativen und persönlichen Projekten.

4.4 Framework gegen Bibliothek

Angular stellt eine vollständige Umgebung zur Verfügung, die eine strukturierte und umfassende Lösung für die Entwicklung von Webanwendungen bietet. Die Umgebung beinhaltet sämtliche erforderliche Werkzeuge und Funktionalitäten zur Erstellung von Anwendungen, darunter Routing, Formularverarbeitung, HTTP-Anfragen und zahlreiche weitere Elemente. Ein Alleinstellungsmerkmal von Angular ist die integrierte Funktionalität, da Entwickler keine zusätzlichen Bibliotheken integrieren müssen, um grundlegende Funktionen zu implementieren. Angular stellt sämtliche benötigten Funktionen bereit, sodass keine weiteren externen Bibliotheken erforderlich sind.

Im Gegensatz zu Angular stellen React und Vue keine vollständigen Frameworks dar, die eine umfassende All-in-One-Lösung für die Webentwicklung bieten. Vielmehr sind sie als spezialisierte Tools zu betrachten, die für die Erstellung und Verwaltung der Benutzeroberfläche (UI) konzipiert sind. Sie bieten Entwicklern die Möglichkeit, je nach Bedarf zusätzliche Tools und Bibliotheken auszuwählen und zu integrieren, je nach

den spezifischen Anforderungen des Projekts. Diese Flexibilität ermöglicht es Ihnen, gezielte Lösungen für verschiedene Aufgaben zu finden und zu nutzen, anstatt sich auf eine vordefinierte Struktur, wie z. B. ein Framework, zu verlassen.

Ein Framework wie Angular stellt eine umfassende und strukturierte Lösung mit integrierten Tools und Funktionen bereit. Es definiert die Architektur und Entwicklungsmethode einer Anwendung und bietet einen klaren, vordefinierten Entwicklungspfad. Bibliotheken wie React und Vue sind hingegen aufgabenorientiert und konzentrieren sich hauptsächlich auf die Erstellung von Benutzeroberflächen. Die Entwickler können frei entscheiden, wie sie die Bibliothek verwenden und welche zusätzlichen Tools sie integrieren wollen. Es gibt keine vordefinierte Architektur, die mehr Flexibilität bietet.

Ein Framework bietet integrierte Lösungen, die ohne zusätzliche Bibliotheken direkt verwendet werden können. Dies beschleunigt die Entwicklung, allerdings ist die Flexibilität begrenzt. Bei der Verwendung von Bibliotheken hingegen können Entwickler je nach den Anforderungen ihres Projekts verschiedene Bibliotheken und Tools auswählen und integrieren. Diese Anpassungsfähigkeit erfordert jedoch mehr Entscheidungen und Planung.

4.5 Zusammenfassung

Die Entscheidung für die Verwendung eines Frameworks oder einer Bibliothek ist von den spezifischen Anforderungen des Projekts sowie dem präferierten Entwicklungsansatz abhängig. Frameworks wie Angular bieten eine umfassende und strukturierte Lösung für die Entwicklung großer und komplexer Anwendungen. Sie stellen alle erforderlichen Tools und Funktionen bereit, sodass keine zusätzlichen Bibliotheken erforderlich sind. Aufgrund des umfassenden Ansatzes eignet sich Angular insbesondere für die Entwicklung großer Unternehmensanwendungen.

Im Gegensatz dazu bieten Bibliotheken wie React und Vue eine höhere Flexibilität und Anpassungsfähigkeit, was insbesondere für kleine und mittlere Projekte von Vorteil sein kann. React, das von Meta (Facebook) unterstützt wird, erfreut sich großer Beliebtheit und verfügt über eine aktive Entwicklergemeinde. Vue hingegen besticht durch seine Einfachheit und Benutzerfreundlichkeit und findet zunehmend Verbreitung, insbesondere bei Anfängern, obwohl es von einem kleinen Team entwickelt wird.

Die Nachfrage nach Entwicklern unterliegt einer variierenden Dynamik, die durch verschiedene Einflussfaktoren auf den relevanten Märkten und in den jeweiligen Regionen determiniert wird. Die weltweit größte Beliebtheit genießt React, gefolgt von Angular und Vue. Die Nachfrage nach React-Entwicklern ist insbesondere in Start-ups und großen Technologieunternehmen hoch, während Angular-Entwickler häufig in Unternehmensanwendungen und komplexen Projekten zum Einsatz kommen. Vue erfreut sich insbesondere in asiatischen Ländern großer Beliebtheit, erfährt jedoch auch im Westen eine zunehmende Verbreitung.

Jede Plattform weist spezifische Vorzüge auf: React bietet eine hohe Flexibilität, eine starke Unterstützung durch die Community sowie eine Vielzahl an Bibliotheken, wodurch es sich ideal für die Entwicklung interaktiver Benutzeroberflächen und plattformübergreifender Anwendungen eignet. Angular bietet umfassende Plattformfunktionen sowie robuste offizielle Bibliotheken und Tools, die sich insbesondere für die Entwicklung großer Unternehmensanwendungen eignen. Vue hingegen besticht durch seine Einfachheit und Flexibilität und ist daher besonders für kleine und mittlere Projekte sowie für Entwicklungsteams, die eine leichtgewichtige Lösung suchen, geeignet.

Merkmal	**Angular**	**Vue.js**	**React**
Entwickler	Google	Evan You	Meta (Facebook)
Erstveröffentlichung	2010 (AngularJS), 2016 (Angular 2+)	2014	2013
Aktuelle Version	18	3	18
Lizenz	MIT	MIT	MIT
Architektur	MVVM	MVVM	V (nur View-Schicht)

Programmier-sprache	TypeScript	JavaScript, Type-Script	JavaScript, Type-Script
Zielgruppe	Große Unterneh-mensanwendun-gen	Kleine bis mittel-große Projekte	Breite Palette von Anwendun-gen
Performance	Gut, optimiert für große Projekte	Sehr gut, leicht-gewichtig	Sehr gut, virtuel-les DOM
Community und Unterstützung	Groß, stark un-terstützt durch Google	Wachsende Community, be-sonders in Asien	Sehr groß, stark unterstützt durch Facebook
Besondere Merkmale	Bidirektionale Datenbindung, Dependency In-jection	Einfache Lern-kurve, hohe Fle-xibilität	Virtuelles DOM, JSX, Hooks
Nachteile	Steile Lern-kurve, komplex	Kleinere Commu-nity, weniger Un-ternehmensun-terstützung	Fokus nur auf die View-Schicht, zu-sätzliche Biblio-theken nötig
Anwendungs-beispiele	Google, Micro-soft, Forbes	Alibaba, Baidu, Xiaomi	Meta (Facebook), Netflix, Airbnb

5. Technische Analyse und Herausforderungen

5.1. Isomorphe Applikation

Die Entwicklung des World Wide Web hat einen bemerkenswerten Wandel erfahren, insbesondere im Hinblick auf die Entwicklung von Webanwendungen. Früher war es üblich, dass ein Webbrowser jede Seite abrief, indem er eine Anfrage an einen Server im Internet richtete, der daraufhin HTML-Code generierte und zurückgab. Dieser Ansatz war angemessen, als Browser noch nicht sehr leistungsfähig waren und HTML-Seiten im Allgemeinen statisch und eigenständig waren.

Die Einführung von JavaScript revolutionierte die Webentwicklung, indem es die Möglichkeit bot, Webseiten dynamischer zu gestalten. Zunächst wurden einfache Interaktionen wie Diashows oder die Auswahl von Daten ermöglicht. Doch im Laufe der Jahre haben Enthusiasten und Entwickler das Potenzial des Webs über seine ursprünglichen Grenzen hinaus erweitert. Heute hat sich das Web zu einer vollständigen Plattform für Anwendungen entwickelt, wobei die rasche Entwicklung von JavaScript und HTML es Entwicklern ermöglicht, leistungsstarke Anwendungen zu erstellen, die früher nur für bestimmte Plattformen geeignet waren.

Ein wichtiger Meilenstein in dieser Entwicklung war der Übergang von rein serverseitigen zu clientseitigen Webanwendungen. Beispielsweise reagiert eine Anwendung wie Gmail (ein klassisches Beispiel für eine einseitige Anwendung) direkt auf Benutzerinteraktionen, ohne ständig Anfragen an den Server zu senden, um statisches HTML zu erhalten.

Bei diesen modernen Webanwendungen befindet sich ein Großteil der Anwendungslogik auf der Client-Seite, wobei spezielle APIs für die Datenkommunikation verwendet werden. Die serverseitige Logik kann in jeder beliebigen Programmiersprache implementiert werden. Sobald die JavaScript-Dateien im Browser geladen sind, initialisiert sich die Client-Anwendung, lädt die Daten über die entsprechende API und stellt den Rest des HTML-Templates dar.

Dieser Ansatz bietet sowohl für Benutzer als auch für Entwickler erhebliche Vorteile. Benutzer profitieren von schnellen Seitenwechseln, ohne die Anwendung nach dem Laden erneut laden zu müssen. Für Entwickler bietet die klare Trennung zwischen Client und Server eine effiziente Entwicklungsumgebung.

Trotz der offensichtlichen Vorteile weist die Single-Page-Anwendungsarchitektur (SPA) einige wesentliche Schwächen auf, die ihren Einsatz in der Praxis erschweren.

Beispiel für isomorphen Code

Eine Wetteranzeige-Website kann so gestaltet werden, dass die Wetterseiten sowohl auf dem Server als auch auf dem Client angezeigt werden. Dies dient dazu, ein schnelles Laden der Seiten und eine gute Suchmaschinenoptimierung zu gewährleisten sowie eine dynamische Interaktivität zu ermöglichen.

In der serverseitigen Renderlogik wird die WetterPage-Komponente auf dem Server gerendert und der resultierende HTML-Code an den Client gesendet. Dies gewährleistet einen schnellen Seitenaufbau und eine gute Suchmaschinenoptimierung.

```
2  import Header from "../Header";
3  import Footer from "../Footer";
4  import styles from "../page.module.css";
5  import WeatherDisplay from "./WeatherDisplay";
6  import { fetchWeatherData } from "../api/weather/route";
7
8  export const metadata = {
9    title: "Wetter",
10   description: "Willkommen auf der Wetter-Seite.",
11 };
12
13 export default async function WetterPage() {
14   let weather = await fetchWeatherData();
15
16   return (
17     <>
18       <Header />
19       <main className={styles.main}>
20         <div>
21           <h1>Willkommen auf isomorphe Wetter Seite</h1>
22           <WeatherDisplay initialWeather={weather} />
23         </div>
24       </main>
25       <Footer />
26     </>
27   );
28 }
```

Abbildung 8: serverseitige Renderlogik

In der clientseitigen Hydrierlogik wird dieselbe WeatherDisplay-Komponente auf dem Client "rehydriert". React nimmt den bereits gerenderten HTML-Code und fügt dynamische Interaktivität hinzu.

```
"use client";

import React, { useState, useEffect } from "react";
import { fetchWeatherData } from "../api/weather/route";

export default function WeatherDisplay({ initialWeather }) {
  const [weather, setWeather] = useState(initialWeather);

  useEffect(() => {
    if (!weather || weather.error) {
      fetchWeatherData().then((data) => {
        if (data && !data.error) {
          setWeather(data);
        }
      });
    }
  }, []);

  if (!weather) {
    return <p>Loading weather data...</p>;
  }

  const temp = Math.round(weather.main.temp);
  const feelsLike = Math.round(weather.main.feels_like);
  const weatherDescription = weather.weather[0].description;

  return (
    <div>
      <p>
        Aktuelles Wetter in Dresden:
        <span className="weather-highlight">
          {temp} °C, {weatherDescription}
        </span>
      </p>
      <p>
        Gefühlt wie:
        <span className="weather-highlight">{feelsLike} °C</span>
      </p>
      <button
        className="update-button"
        onClick={() => fetchWeatherData().then(setWeather)}
      >
        Aktualisieren
      </button>
    </div>
  );
}
```

Abbildung 9: clientseitige Hydrierlogik

Erklärung

Die serverseitige Renderlogik bedeutet, dass die WetterPage-Komponente auf dem Server gerendert wird und der resultierende HTML-Code an den Client gesendet wird. Dies gewährleistet einen schnellen Seitenaufbau und eine gute Suchmaschinenoptimierung. Die WeatherDisplay-Komponente wird dabei mit den initialen Wetterdaten gerendert, die vom Server abgerufen wurden.

Auf dem Client wird dieselbe WeatherDisplay-Komponente "rehydriert". React übernimmt den bereits gerenderten HTML-Code und fügt dynamische Interaktivität hinzu. Durch die Verwendung von isomorphen Anwendungen können Entwickler die Vorteile beider Welten (Client und Server) nutzen und gleichzeitig die Wartbarkeit und Konsistenz des Codes verbessern. Dies macht isomorphe Anwendungen besonders attraktiv für komplexe Webanwendungen, bei denen sowohl Suchmaschinenoptimierung als auch Benutzerfreundlichkeit entscheidend sind.

5.2. SEO

Eine reine Client-Anwendung hat in der Regel Schwierigkeiten, effektiv mit den Crawlern der Suchmaschinen zu interagieren. Das bedeutet, dass Single Page Applications (SPAs) standardmäßig nicht für Suchmaschinenoptimierung (SEO) geeignet sind. Suchmaschinen-Crawler senden Anfragen an einen Webserver und interpretieren die Ergebnisse. Wenn der Server aber nur eine leere Seite zurückgibt, ist das für Suchmaschinen wenig hilfreich. Es gibt Umgehungslösungen, die aber oft Herausfordernd sind.

In SPAs wird ein Großteil des Inhalts durch clientseitiges JavaScript erzeugt. Suchmaschinen-Crawler, die zunächst nur die HTML-Antwort sehen, sehen oft eine leere Seite oder nur eine minimale HTML-Struktur, was die Indexierung erschwert. Der Inhalt erscheint erst, nachdem das JavaScript geladen und ausgeführt wurde, was die Ladezeiten verlängern kann. Obwohl Google und andere Suchmaschinen damit begonnen haben, JavaScript zu rendern, benötigt dieser Prozess Ressourcen und ist nicht immer zuverlässig.

Ein weiteres Problem besteht darin, dass SPAs Inhalte dynamisch aktualisieren, ohne die URL zu ändern. Infolgedessen sind Crawler möglicherweise nicht in der Lage, alle Seiten einer Anwendung zu erkennen und zu indexieren, was die SEO-Leistung

beeinträchtigt.

Beispiel eines HTML-Dokuments einer isomorphen Applikation.

```html
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="stylesheet" href="/_next/static/css/app/layout.css?v=1720903746722"
    data-precedence="next_static/css/app/layout.css" />
    <link rel="stylesheet" href="/_next/static/css/app/wetter/page.css?v=1720903746722"
    data-precedence="next_static/css/app/wetter/page.css"/>
    <link rel="preload" as="script" fetchPriority="low" href="/_next/static/chunks/webpack.js?
    v=1720903746722"/>
    <script type="text/javascript" src="http://gc.kis.v2.scr.kaspersky-labs.com/
    FD126C42-EBFA-4E12-B309-BB3FDD723AC1/main.js?
    attr=hGOgF7OxKj2NkCOPOon02TeIr0JsBAX6Fiv-uddU31IMLYJbD5-Ou2XXm3xRMOW_" charset="UTF-8"></script>
    <link rel="stylesheet" crossorigin="anonymous" href="http://gc.kis.v2.scr.kaspersky-labs.com/
    E3E8934C-235A-4B0E-825A-35A08381A191/abn/main.css?attr=aHR0cDovL2xvY2FsaG9zdDozMDAwL3dldHR1cg"/>
    <script src="/_next/static/chunks/main-app.js?v=1720903746722" async=""></script>
    <script src="/_next/static/chunks/app-pages-internals.js" async=""></script>
    <script src="/_next/static/chunks/app/wetter/page.js" async=""></script>
    <title>Wetter</title>
    <meta name="description" content="Willkommen auf der Wetter-Seite." />
    <link rel="icon" href="/favicon.ico" type="image/x-icon" siz-es="16x16" />
    <script src="/_next/static/chunks/polyfills.js" noMod-ule=""></script>
  </head>
  <body class="__className_aaf875">
    <header>
      <nav>
        <ul>
          <li><a href="/">Startseite</a></li>
          <li><a href="/wetter">Wetter</a></li>
          <li><a href="/about">Über uns</a></li>
        </ul>
      </nav>
    </header>
    <main class="page_main__nw1Wk">
      <div>
        <h1>Willkommen auf isomorphe Wetter Seite</h1>
        <div>
          <p>
            Aktuelles Wetter in Dresden:
            <span class="weather-highlight">19°C, klarer Himmel</span>
          </p>
          <p>
            Gefühlt wie:
            <span class="weather-highlight"> 19°C</span>
          </p>
          <button class="update-button">Aktualisieren</button>
        </div>
      </div>
    </main>
    <footer>
      <p>© 2024 Meine Website</p>
    </footer>
  </body>
</html>
```

Abbildung 10: HTML-Dokument einer isomorphen Applikation

Beispiel eines HTML-Dokuments einer Single-Page-Applikation (SPA)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="/logo192.png" />
    <link rel="manifest" href="/manifest.json" />
    <title>React App</title>
    <script
      type="text/javascript"
      src="http://gc.kis.v2.scr.kaspersky-labs.com/FD126C42-EBFA-4E12-B309-BB3FDD723AC1/
      main.js?attr=2ie1PLVlF1NkvRtfd6ClNFDiBh85AZbVVff_RZHpJ6FZcHmXHB2NTK0oKxieqXMy"
      charset="UTF-8"
    ></script>
    <link
      rel="stylesheet"
      crossorigin="anonymous"
      href="http://gc.kis.v2.scr.kaspersky-labs.com/E3E8934C-235A-4B0E-825A-35A08381A191/
      abn/main.css?attr=aHR0cDovL2xvY2FsaG9zdDozMDAwL3dlYXRoZXI"
    />
    <script defer src="/static/js/bundle.js"></script>
  </head>
  <body>
    <div id="root">Loading</div>
  </body>
</html>
```

Abbildung 11: HTML-Dokument einer Single-Page-Applikation

5.3. Leistung

In einer modernen Webanwendung ist die Leitung ein entscheidender Faktor für die Benutzerfreundlichkeit und -zufriedenheit. Bei Single Page Applications (SPAs) wird der Großteil des Inhalts mit clientseitigem JavaScript generiert, was zu erheblichen Verzögerungen führen kann, da der Benutzer auf den vollständigen Seiteninhalt wartet.

Ein großes Problem bei SPAs ist die Initialisierung. Da die meisten Inhalte über JavaScript geladen werden, sehen Benutzer oft eine leere Seite oder ein Ladesymbol, bis alle erforderlichen Ressourcen vollständig geladen und verarbeitet wurden. Dies kann mehrere Sekunden dauern, insbesondere bei langsamen Internetverbindungen oder weniger leistungsfähigen Geräten.

Auch die Ausführung von JavaScript im Browser kann ressourcenintensiv sein, insbesondere bei komplexen Anwendungen. Dies führt zu längeren Ladezeiten und kann die Reaktionsfähigkeit der Anwendung beeinträchtigen. SPAs aktualisieren Inhalte dynamisch, ohne die gesamte Seite neu zu laden, was die Benutzerinteraktion verbessert, aber auch die Leistung beeinträchtigen kann, wenn viele Daten geladen und verarbeitet werden müssen.

Diese Probleme machen deutlich, dass bei der Entwicklung von SPAs die Performance sorgfältig berücksichtigt werden muss, um eine optimale Benutzerinteraktion zu gewährleisten.

5.4. Unterstützung

Warum ist die Unterstützung isomorpher Anwendungen eine Herausforderung?

Bei der Entwicklung einer isomorphen Anwendung wird derselbe Code sowohl auf dem Server als auch auf dem Client ausgeführt. In der Theorie klingt das gut und einfach, aber in der Praxis gibt es einige Probleme, die die Situation komplizieren können.

Idealfall: Getrennter Code für Client und Server.

Im Idealfall sollte der Code, der auf dem Server läuft, und der Code, der auf dem Client läuft, klar voneinander getrennt sein. Dies würde bedeuten, dass server-spezifische Logik und clientspezifische Logik in getrennten Dateien oder Modulen untergebracht werden. Dies erleichtert die Wartung und Weiterentwicklung der Anwendung, da man genau weiß, welcher Code wo und warum ausgeführt wird.

Praxis: Vermischung der Logik

In der Praxis ist es oft nicht einfach, die Logik vollständig zu trennen. In vielen Fällen wird die gleiche Logik sowohl auf dem Server als auch auf dem Client benötigt. Ein gutes Beispiel hierfür ist die Datenformatierung. Wenn eine Anwendung beispielsweise ein Datum anzeigen soll, muss dieses Datum sowohl auf dem Server (wenn die Seite angezeigt wird) als auch auf dem Client (wenn die Seite aktualisiert wird) im gleichen Format angezeigt werden.

Ein weiteres Beispiel ist die Validierung von Formularen. Wenn ein Benutzer ein Formular ausfüllt, sollte das Formular bereits im Browser (Client) validiert werden, bevor es an den Server gesendet wird. Gleichzeitig muss der Server die Daten erneut

validieren, um sicherzustellen, dass sie korrekt und sicher sind, da man sich nicht darauf verlassen kann, dass der Client immer korrekt arbeitet oder dass der Benutzer die Daten nicht manipuliert hat.

Isomorphie als Lösung

Isomorphe Anwendungen bieten eine vielversprechende Lösung, da sie die Ausführung des gleichen Codes sowohl auf dem Server als auch auf dem Client ermöglichen. Dadurch kann Logik für Aufgaben wie Datenformatierung und Formularvalidierung gemeinsam genutzt und wiederverwendet werden, was die Konsistenz und Wartbarkeit des Codes verbessert.

In der Praxis sehen sich Entwickler bei der Implementierung isomorpher Anwendungen jedoch einer Reihe von Herausforderungen gegenüber. Eine der größten Herausforderungen ist die Komplexität der Implementierung. Die Idee, denselben Code auf beiden Seiten auszuführen, ist zwar verlockend, erfordert aber zusätzliche Tests und Fehlerbehebungen, um sicherzustellen, dass der Code in beiden Umgebungen korrekt funktioniert.

Ein weiteres Problem ist die zusätzliche Belastung des Servers. Da der Server nun auch für die Anzeige der Seiten verantwortlich ist, kann dies zu einer erheblichen Belastung der Serverressourcen führen. Wenn der Server nicht richtig skaliert ist, kann dies zu Leistungseinbußen führen.

Außerdem ist die Synchronisation zwischen Client und Server eine komplexe Aufgabe. Die Logik muss so geschrieben werden, dass sie in beiden Umgebungen funktioniert, was bedeutet, dass bestimmte Funktionen oder Bibliotheken, die nur in einer der beiden Umgebungen verfügbar sind, vermieden werden müssen. Diese Einschränkungen erfordern sorgfältige Planung und Design, um eine effiziente und konsistente Anwendung zu gewährleisten.

Trotz dieser Herausforderungen bieten isomorphe Anwendungen erhebliche Vorteile in Bezug auf Ladezeiten, Suchmaschinenoptimierung und Benutzerfreundlichkeit, was sie zu einer attraktiven Option für moderne Webanwendungen macht.

5.5. Hybrid-Ansatz

Ein hybrider Ansatz für die Webentwicklung kombiniert das Beste aus Server- und Clientseitigem Rendering, um die Vorteile beider Methoden zu nutzen. Dieser Ansatz ermöglicht ein schnelles Laden einer Webseite, indem diese zunächst als fertiges HTML-Dokument vom Server an den Browser gesendet wird. Diese Methode verkürzt nicht nur die Ladezeit, sondern optimiert auch die Suchmaschinenoptimierung (SEO), da der Inhalt sofort für Suchmaschinen sichtbar ist.

Sobald die erste Seite geladen ist, beginnt das clientseitige Rendering, das eine dynamische und interaktive Benutzererfahrung bietet. Die Nutzer profitieren von einer nahtlosen Interaktion mit der Website, da Inhalte aktualisiert werden können, ohne dass die gesamte Seite neu geladen werden muss. Dieses Verfahren bietet eine flexible und reaktionsfähige Schnittstelle, die besonders für komplexe Anwendungen wie Online-Shops oder interaktive Plattformen geeignet ist.

Ein weiterer Vorteil dieses Ansatzes ist die Möglichkeit, denselben JavaScript-Code sowohl auf dem Server als auch im Browser zu verwenden. Dies bedeutet, dass Entwickler nicht zwei getrennte Codebasen pflegen müssen, was den Entwicklungs- und Wartungsaufwand erheblich reduziert. Eine einzige Codebasis vereinfacht auch das Debugging und die Qualitätskontrolle, da Fehlerquellen schneller erkannt und behoben werden können.

Die Hybridtechnologie ermöglicht auch eine effizientere Nutzung von Ressourcen. Indem ein Teil der Datenverarbeitung auf den Server verlagert wird, können Ressourcen auf dem Endgerät eingespart werden, was die Benutzerinteraktion insbesondere für Nutzer mit älteren oder weniger leistungsfähigen Geräten verbessert. Gleichzeitig können Entwickler die Serverleistung optimieren, um die Last auszugleichen und die Gesamtleistung der Anwendung zu verbessern.

Trotz seiner vielen Vorteile erfordert dieser Ansatz ein tiefes technisches Verständnis und eine sorgfältige Architekturplanung. Die Entwickler müssen sicherstellen, dass die Serverleistung auch bei hoher Benutzerlast effizient und stabil bleibt, um eine gleichbleibende Performance zu gewährleisten. Dazu gehört auch ein Lastausgleich zwischen Server und Client, damit die Server nicht überlastet werden und die Anwendung auch bei einer großen Anzahl von Benutzern reibungslos läuft.

Damit bietet der hybride Ansatz eine effektive Lösung für moderne Webanwendungen, die sowohl schnelle Ladezeiten als auch eine hohe Benutzerinteraktivität erfordern. Durch die geschickte Kombination von server- und clientseitigem Rendering kann eine optimale Performance erreicht werden, die sowohl den Erwartungen der Nutzer als auch den Anforderungen der Suchmaschinen gerecht wird.

5.6. Isomorphes JavaScript in der freien Entwicklung

Die Idee ist nicht neu - Nodejitsu hat die Architektur von isomorphem JavaScript bereits im Jahr 2011 ausführlich beschrieben. Allerdings war die Anpassung dieser Technologie damals schwierig. Seitdem sind mehrere isomorphe Frameworks entstanden.

Mojito war das erste bekannte isomorphe Open-Source-Framework. Es ist ein fortgeschrittenes Full-Stack-Framework, das auf Node.js basiert, jedoch stark von YUI (Yahoo User Interface) abhängig war.

Meteor ist heute das wohl bekannteste isomorphe Projekt. Meteor wurde von Grund auf für die Unterstützung von Echtzeitanwendungen erstellt. Das Team entwickelte ein komplettes Ökosystem rund um das Batching-System und die Bereitstellungstools.

Asana, eine Task-Management-App, die von dem Facebook-Mitbegründer Dustin Moskovitz mitbegründet wurde, hat eine interessante isomorphe Geschichte. Ohne Probleme mit der Finanzierung hat Asana jahrelang recherchiert und ihren eigenen geschlossenen Quellcode Luna entwickelt - eines der fortschrittlichsten Beispiele für isomorphes JavaScript.

Luna, ursprünglich für v8cgi (in den Tagen vor Node.js) erstellt, durfte für jede Benutzersitzung eine Kopie der Anwendung auf dem Server ausführen. Für jeden Benutzer wurde ein separater Serverprozess gestartet, der denselben JavaScript-Code ausführt, der auf dem Client ausgeführt wird. Dies brachte alle Vorteile der fortschrittlichen Optimierung mit sich, wie Offline-Widerstandsfähigkeit und schnelle Reaktionszeiten.

Alle diese Projekte sind recht groß, was bedeutet, dass Full-Stack-Frameworks mit ziemlich komplexen Problemen konfrontiert sind. Client und Server sind sehr unterschiedliche Umgebungen. Daher muss eine bestimmte Gruppe von Abstraktionen erstellt werden, die Anwendungslogik von grundlegenden Implementierungen trennt, sodass eine einzige API für Anwendungsentwickler erstellt werden kann.

5.7. Routing

Das Routing in isomorphen Anwendungen spielt eine zentrale Rolle, da es bestimmt, wie Anfragen in der Server- und Client-Umgebung weitergeleitet werden. Für eine effektive Suchmaschinenoptimierung und eine verbesserte Benutzerinteraktion ist es wichtig, dass die Routen so konfiguriert sind, dass sie bestimmte URI-Muster erkennen und diese korrekt an Suchmaschinen-Crawlern weiterleiten.

Beispiel für Routing in Next.js

Weiterleitung

Next.js ermöglicht die Konfiguration von Weiterleitungen direkt in der Datei next.config.mjs. Dies ist nützlich, um alte URLs auf neue Routen umzuleiten und so SEO und Benutzerfreundlichkeit zu verbessern.

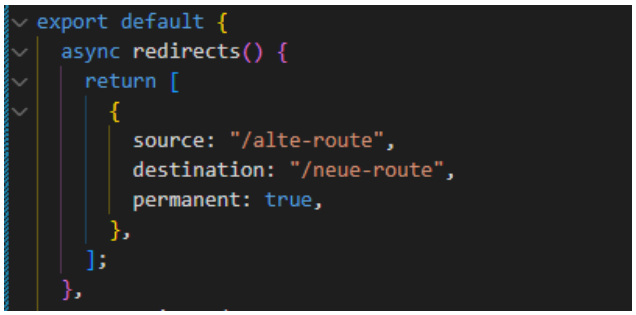


Abbildung 12: Routing Weiterleitungen

source: Dies ist der Pfad der ursprünglichen Anfrage. Beispiel: Wenn jemand versucht, die URL www.meineWebSeite.de/alte-route aufzurufen.

destination: Dies ist das Ziel, an das die Anfrage weitergeleitet wird, z. B. meineWebSeite.de//neue-route.

permanent: Wenn dies auf „true“ gesetzt ist, melden Suchmaschinen, dass die Umleitung permanent ist. Dies hilft, den SEO-Wert der alten Webseite auf die neue Webseite zu übertragen.

Dateibasiertes Routing

Next.js verwendet ein dateibasiertes Routing-System. Jede Datei im Verzeichnis entspricht einer Route in der Anwendung. Dies vereinfacht die Struktur und Verwaltung der Routen erheblich.

```jsx
// app/about/page.js
import Layout from "../layout";
import Header from "../Header";
import Footer from "../Footer";
import styles from "../page.module.css";

export const metadata = {
  title: "Über uns",
  description: "Willkommen auf meiner Über uns Seite.",
};

export default function About() {
  return (
    <>
      <Header />
      <main className={styles.main}>
        <div>
          <h1>{metadata.title}</h1>
          <p>{metadata.description}</p>
          <h2>Meine Mission</h2>
        </div>
      </main>
      <Footer />
    </>
  );
}
```

Abbildung 13: Dateibasiertes Routing

Die Datei app/about/page.js, About-Komponente definiert eine Route /about, die eine statische Seite mit Informationen enthält.

API-Routen

Next.js unterstützt auch API-Routen, die es ermöglichen, serverseitige Endpunkte in einer einzigen Anwendung zu definieren. Diese können für verschiedene Zwecken verwendet werden, z. B. zum Abrufen von Daten, zur Authentifizierung

```jsx
//api/weather/route.js
import fetch from "node-fetch";

export async function fetchWeatherData() {
  const apiUrl = `https://api.openweathermap.org/data/2.5/weather?q=Dresden,de&appid=9263e4527fa4851adcef1f4faa5cbb7b&units=metric&lang=de&${new Date().getTime()}`;

  try {
    const response = await fetch(apiUrl);

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    const data = await response.json();
    return data;
  } catch (error) {
    console.error("Fetch error:", error);
    return null;
  }
}
```

Abbildung 14: API-Routen

Die Datei app/api/weather/route.js definiert eine API-Route /api/weather, die Wetterdaten von einer externen API abruft und als JSON-Antwort zurückgibt.

Clientseitige Navigation

Für die clientseitige Navigation bietet Next.js eine Link-Komponente im Modul next/link. Damit kann zwischen Seiten navigiert werden, ohne dass die Seite komplett neu geladen werden muss.

```
2   import Link from "next/link";
3   import "./globals.css";
4
5   export default function Header() {
6     return (
7       <header>
8         <nav>
9           <ul>
10            <li>
11              <Link href="/">Startseite</Link>
12            </li>
13            <li>
14              <Link href="/wetter">Wetter</Link>
15            </li>
16            <li>
17              <Link href="/about">Über uns</Link>
18            </li>
19          </ul>
20        </nav>
21      </header>
22    );
23  }
```

Abbildung 15: Clientseitige Navigation

Der Header enthält Navigationslinks zu verschiedenen Seiten der Anwendung, wobei Link verwendet wird, um die Navigation zu ermöglichen.

Durch die Kombination dieser Techniken bietet Next.js eine leistungsfähige und flexible Routing-Implementierung, die sowohl server- als auch clientseitige Anforderungen erfüllt. Dies verbessert die Benutzerfreundlichkeit, die Suchmaschinenoptimierung und die Wartbarkeit der Anwendung.

5.8. Seitengenerierung

In der modernen Webentwicklung steht die Seitengenerierung im Mittelpunkt vieler technischer Fragen, insbesondere wenn es um Benutzerfreundlichkeit und Performance geht. Man hat die Wahl zwischen manueller Manipulation des DOM, der Verwendung von HTML-Templates oder der Verwendung von Komponentenbibliotheken, die das DOM abstrahieren. Unabhängig von der gewählten Methode ist es wichtig, dass das Markup sowohl auf dem Server als auch auf dem Client effizient und konsistent erzeugt wird. Durch diese Flexibilität kann die Darstellung der Inhalte an die spezifischen Anforderungen der Anwendung angepasst werden.

Ein fortgeschrittener Ansatz ist die Verwendung von Frameworks wie React, Angular oder Vue.js, die die Wiederverwendung von Komponenten und die Konsistenz des Codes zwischen Server und Frontend fördern. Diese Technologien verbessern nicht nur die Wartbarkeit des Codes, sondern auch die Entwicklungsgeschwindigkeit und die Anwendungsperformance durch optimierte Rendering-Zyklen und effiziente Ressourcennutzung.

5.9. Server-Rendering

Bibliotheken bzw. Frameworks wie React, Vue und Angular unterstützen das serverseitige Rendering, das wesentlich zur Suchmaschinenoptimierung und zur initialen Ladegeschwindigkeit beiträgt. Das serverseitige Rendering von Seiteninhalten macht den zunächst sichtbaren Inhalt für Suchmaschinen-Crawler verfügbar und verbessert so die Indizierung. Dieser Ansatz ermöglicht auch schnelle Ladezeiten, da die Inhalte den Nutzern sofort angezeigt werden, was insbesondere für mobile Nutzer von Vorteil ist.

Eine Herausforderung beim serverseitigen Rendering ist jedoch der Umgang mit asynchronen Anfragen. Traditionell erwartet das synchrone serverseitige Rendering keine asynchronen Operationen, was zum Verlust von dynamisch geladenen Daten führen kann. Moderne Frameworks bieten jedoch Lösungen an, wie z.B. Next für React, um diese Daten während des Renderings zu berücksichtigen und sicherzustellen, dass alle Seiteninhalte korrekt generiert und angezeigt werden.

Durch den Einsatz fortschrittlicher Caching-Strategien und die Optimierung der Serverkonfiguration können Server-Rendering-Probleme effektiv angegangen werden, um eine hohe Performance und Benutzerfreundlichkeit zu gewährleisten.

5.10. Code-Aufteilung

Um die Leistung von Webanwendungen zu optimieren, ist es wichtig, den erforderlichen JavaScript-Code für jede Seite einzeln zu laden, anstatt die gesamte Anwendung auf einmal zu laden. Diese als „Code-Splitting“ bekannte Technik verkürzt die Ladezeiten und optimiert den Ressourcenverbrauch, indem die ursprünglich geladene Datenmenge minimiert wird.

Frameworks wie Next.js haben sich bei der Umsetzung dieser Technik als besonders effektiv erwiesen. Im Gegensatz zu herkömmlichen Single-Page-Applikationen (SPAs), die oft eine manuelle Konfiguration erfordern, um den Code effizient aufzuteilen, bietet Next.js eine eingebaute automatische Lösung. Sie verwendet einen dynamischen Import, der es Entwicklern ermöglicht, Komponenten und Bibliotheken nur dann zu laden, wenn sie tatsächlich benötigt werden. Dies reduziert die initiale Last und beschleunigt den Seitenaufbau.

Beim Wechsel von einer Seite zur anderen erfolgt lediglich die Ladung des zusätzlichen Codes, welcher spezifisch für die neue Seite erforderlich ist. Dieser Ansatz minimiert die Anzahl erforderlicher Datenübertragungen und beschleunigt das Laden der Seite erheblich. Dies ist insbesondere für Benutzer mit langsamen Internetverbindungen oder mobilen Geräten von Vorteil. Im Vergleich zu traditionellen SPAs, bei denen in der Regel das gesamte Paket neu geladen werden muss, resultiert dies in einer signifikanten Steigerung der Leistung.

Des Weiteren bietet Next.js Unterstützung für prädiktives Laden. Während der Benutzer durch die Anwendung navigiert, erkennt Next.js potenzielle zukünftige Navigationsschritte und lädt erforderliche Codepakete im Hintergrund vor. Diese proaktive Ladestrategie optimiert die Benutzerinteraktion, indem sie die Wartezeiten beim Laden neuer Seiten minimiert und eine nahezu sofortige Reaktion der Anwendung ermöglicht.

Die fortschrittlichen Techniken der Code-Aufteilung und des prädiktiven Ladens sind von entscheidender Bedeutung für die Entwicklung nicht nur funktionaler und ansprechender, sondern auch effizienter und reaktionsschneller Webanwendungen. Next.js

implementiert diese Konzepte nahtlos und automatisiert deren Ausführung. Damit stellt es ein wertvolles Werkzeug für Entwickler dar, die eine leistungsstarke und benutzerfreundliche Webarchitektur benötigen, welche die Interaktion der Benutzer mit einer Anwendung so reibungslos und angenehm wie möglich gestaltet.

5.11. Best Practices für die Entwicklung isomorpher Anwendungen

Die Identifizierung und Implementierung von Best Practices erweist sich für die effiziente und erfolgreiche Entwicklung, Wartung und Optimierung isomorpher Anwendungen als von entscheidender Bedeutung. Auf Basis einer Analyse gegenwärtiger Technologien und Frameworks wie React und Next.js lassen sich folgende Best Practices identifizieren: Zunächst ist eine klare Trennung zwischen Geschäfts- und Präsentationslogik erforderlich. Dies vereinfacht die Wartung und ermöglicht die Wiederverwendung von Komponenten. Beispielsweise sollten Datenabruffunktionen in separaten Servicedateien abgelegt und Präsentationskomponenten ausschließlich für die Präsentation verwendet werden.

Ein weiteres wesentliches Prinzip ist die Implementierung eines effizienten Routing-Systems, welches Anfragen sowohl auf Server- als auch auf Clientseite korrekt verarbeitet. Next.js eignet sich in besonderem Maße für die Umsetzung dieser Anforderungen, da es die serverseitige Seitengenerierung und die clientseitige Navigation durch die Link-Komponente unterstützt. Um die SEO-Leistung zu optimieren, ist die Verwendung von Server-Side-Rendering (SSR) empfehlenswert, um die Inhalte für Suchmaschinen-Crawler zugänglich zu machen. Next.js kann genutzt werden, um Seiteninhalte auf dem Server zu rendern, bevor sie an den Client gesendet werden.

Des Weiteren wird die Verwendung von Code-Splitting und Lazy Loading empfohlen, um die Ladezeit zu minimieren und die Anwendungsleistung zu optimieren. Dies kann durch den dynamischen Import von Komponenten in Next.js erreicht werden, was den initialen JavaScript-Download reduziert.

Ein weiterer wichtiger Aspekt zur Gewährleistung der Anwendungszuverlässigkeit ist die Durchführung automatisierter Tests. Hierfür können Testbibliotheken wie Jest für Unit-Tests und Cypress für End-to-End-Tests verwendet werden.

Die Hydrierlogik muss zudem so gestaltet werden, dass sie eine effiziente Interaktivität auf Client-Seite nach dem initialen Laden gewährleistet. Dies kann durch die

Verwendung von React-Hooks und State-Management-Lösungen wie Redux erreicht werden, um den Zustand der Anwendung konsistent zu halten.

Sicherheitsmaßnahmen sind von essentieller Bedeutung, um Angriffe sowohl auf Server- als auch auf Clientseite zu verhindern. Zu diesen Maßnahmen zählt die Validierung von Eingaben sowohl auf dem Server als auch auf dem Client sowie die Verwendung sicherer Authentifizierungsverfahren.

Schließlich ist die Verwendung von Tools zur Überwachung und Optimierung der Anwendungsleistung zu empfehlen. Die Integration von Leistungsüberwachungstools wie Lighthouse und Sentry ermöglicht die Analyse und Optimierung von Ladezeiten sowie die Behebung von Fehlern.

5.12. Gründe für ReactJS und dessen Funktionalität

React hat sich schnell unter JavaScript-Entwicklern etabliert und erfreut sich nach wie vor großer Beliebtheit. Dies lässt sich auf mehrere Faktoren zurückführen:

Entwicklung durch Meta (Facebook): React wurde von Meta (ehemals Facebook) entwickelt, einem renommierten Unternehmen mit großem Einfluss in der Tech-Community. Diese Herkunft hat dazu beigetragen, dass React von Anfang an viel Aufmerksamkeit und Vertrauen erhielt. Seit seiner Einführung wird React kontinuierlich weiterentwickelt, wobei Fehler behoben und neue Funktionen hinzugefügt werden.

Unidirektionaler Datenfluss: React unterscheidet sich durch seinen Ansatz im Vergleich zu anderen Frameworks wie Angular oder Vue. Während viele Frameworks bidirektionales Datenbinding unterstützen, nutzt React einen unidirektionalen Datenfluss. Dies bedeutet, dass Daten von übergeordneten Komponenten zu untergeordneten Komponenten fließen und Änderungen durch Events ausgelöst werden. Diese Struktur sorgt für eine klare Trennung und erleichtert die Wartung und Skalierung von Anwendungen.

Virtual DOM: Eine der Kerninnovationen von React ist das Virtual DOM. Anstatt den tatsächlichen DOM direkt zu manipulieren, erstellt React eine virtuelle Kopie des DOM im Speicher. Änderungen werden zuerst im virtuellen DOM vorgenommen und dann effizient auf den realen DOM angewendet. Dies führt zu einer verbesserten Performance, insbesondere bei umfangreichen Updates der Benutzeroberfläche.

Kernfunktionen von React

Komponentenbasierte Architektur: React basiert auf einer komponentenbasierten Architektur, bei der jede Komponente eine eigenständige Einheit der Benutzeroberfläche darstellt. Dies fördert die Wiederverwendbarkeit und erleichtert die Wartung und Weiterentwicklung der Anwendung.

Deklarative UI-Beschreibung: Mit React beschreiben Entwickler den gewünschten Zustand der Benutzeroberfläche, und React kümmert sich um die effiziente Aktualisierung und Darstellung der Komponenten, wenn sich die Daten ändern. Dies vereinfacht das Debugging und die Optimierung der Benutzeroberfläche.

Effiziente DOM-Aktualisierung: Das Virtual DOM ermöglicht es React, Updates effizient durchzuführen, indem nur die tatsächlich geänderten Teile des DOM aktualisiert werden. Dies verbessert die Performance und reduziert die Ladezeiten.

JSX: React verwendet JSX, eine Syntaxerweiterung für JavaScript, die es ermöglicht, HTML-ähnliche Strukturen direkt in JavaScript einzufügen. Dies macht den Code lesbarer und einfacher zu schreiben.

Unidirektionaler Datenfluss: React folgt einem unidirektionalen Datenfluss, bei dem Daten von übergeordneten Komponenten zu untergeordneten Komponenten weitergegeben werden. Dies vereinfacht die Kontrolle über den Datenfluss und reduziert die Komplexität der Anwendung.

React Hooks: React Hooks ist eine Erweiterung von React, die es Entwicklern ermöglicht, Zustände und andere React-Funktionen in Funktionskomponenten zu verwenden. Hooks haben den Umgang mit Zuständen und Nebenwirkungen vereinfacht und den Code modularer und wiederverwendbarer gemacht.

Server Components: Server Components sind eine experimentelle Funktion, die es ermöglicht, Komponenten auf dem Server zu rendern und an den Client zu senden. Dies kann die Performance und Ladezeiten weiter verbessern, insbesondere für Anwendungen, die eine schnelle initiale Ladezeit erfordern.

Anwendungsfälle:

Web- und mobile Anwendungen: React wird sowohl für Web- als auch für mobile Anwendungen genutzt. Mit React Native, einer Erweiterung von React, können Entwickler native mobile Anwendungen für iOS und Android erstellen.

Dynamische Webseiten: React eignet sich besonders gut für dynamische Webseiten, deren Inhalte sich häufig ändern. Durch die effiziente Aktualisierung des DOMs werden nur die Teile aktualisiert, die tatsächlich geändert werden müssen.

Progressive Web Apps (PWAs): React unterstützt die Erstellung von PWAs, die offline funktionieren und eine schnelle, zuverlässige Performance bieten.

Integration in CMS-Systeme: React ermöglicht eine reaktionsschnelle und benutzerfreundliche Verwaltung von Inhalten mit benutzerdefinierten Editoren und Vorschau-Tools.

React bleibt aufgrund seiner Flexibilität, Effizienz und der starken Community weiterhin eine der führenden Bibliotheken für die Frontend-Entwicklung. Die kontinuierliche Weiterentwicklung und die Einführung neuer Funktionen tragen dazu bei, dass React stets am Puls der Zeit bleibt und sich den Anforderungen moderner Web- und Mobilanwendungen anpasst.

5.13. Next.js

Next.js ist ein populäres Open-Source-Framework, das auf React basiert und für die Entwicklung von Single-Page-Anwendungen (SPAs), statischen Websites und serverseitig gerenderten Anwendungen verwendet wird. Es bietet zahlreiche Vorteile und Funktionen, die es für moderne Webprojekte besonders attraktiv machen.

Server-Side Rendering (SSR): Next.js ermöglicht das serverseitige Rendern von Webseiten, bei dem der HTML-Inhalt einer Seite auf dem Server generiert und an den Browser gesendet wird. Dies verbessert die Ladezeiten und die Suchmaschinenoptimierung (SEO), da Suchmaschinen vollständig gerenderte Seiten leichter indizieren können.

Statische Seitengenerierung (SSG): Next.js kann statische Websites generieren, bei denen jede Seite zur Build-Zeit als HTML-Datei vorgefertigt wird. Diese Seiten können dann blitzschnell über ein CDN ausgeliefert werden, was erhebliche Performancevorteile bietet und die Serverlast reduziert.

Incremental Static Regeneration (ISR): ISR ermöglicht es, statische Seiten nach ihrer Bereitstellung zu aktualisieren. Entwickler können statische Seiten mit dynamischen Inhalten kombinieren, die regelmäßig aktualisiert werden, ohne dass die gesamte Seite neu gebaut werden muss.

Automatisches Code-Splitting: Next.js teilt den JavaScript-Code automatisch in kleinere Pakete auf, die bei Bedarf geladen werden. Dies verbessert die anfängliche Ladezeit der Webseite, da nur der Code geladen wird, der für die anfängliche Darstellung benötigt wird.

API-Routen: Mit Next.js können Entwickler API-Endpunkte direkt im Projekt definieren. Diese API-Routen laufen auf demselben Server wie die Next.js-Anwendung, was die Entwicklung und den Betrieb von Full-Stack-Anwendungen vereinfacht.

Integrierte Bildoptimierung: Next.js bietet eine integrierte Bildoptimierung, die Bilder automatisch lädt und basierend auf der Bildschirmgröße des Benutzers und anderen Faktoren optimiert. Dies reduziert die Ladezeiten und verbessert die Benutzererfahrung.

Unterstützung für Server Components: Server Components ermöglichen es, Komponenten auf dem Server zu rendern und an den Client zu senden. Dies kann die Performance und Ladezeiten weiter verbessern, insbesondere für Anwendungen, die eine schnelle initiale Ladezeit erfordern.

Partial Prerendering (PPR): Partial Prerendering kombiniert statische und dynamische Rendering-Techniken auf derselben Seite. Dies ermöglicht es, eine statische HTML-Hülle sofort zu liefern und die dynamischen Teile im selben HTTP-Request nachzuladen.

Turbopack (Beta): Turbopack ist der neue, hochperformante JavaScript-Bundler von Vercel, der schneller als Webpack ist und eine verbesserte Entwicklungs- und Produktionsperformance bietet.

5.14. Fallstudien

Fallstudie: Nike

Nike setzt React und Next.js ein, um die Performance und Benutzerfreundlichkeit seiner E-Commerce-Plattform zu optimieren. Diese Technologien bieten mehrere technische und betriebliche Vorteile:

React ermöglicht die Entwicklung von hochgradig interaktiven und dynamischen Benutzeroberflächen. Empirische Studien zeigen, dass interaktive und ansprechende Benutzeroberflächen die Verweildauer und Konversionsraten signifikant erhöhen können

. Durch die Verwendung von Komponenten, die wiederverwendbar und leicht wartbar sind, kann Nike eine konsistente und ansprechende User Experience sicherstellen.

Next.js übernimmt das serverseitige Rendering, was zu schnelleren Ladezeiten führt und die Suchmaschinenoptimierung (SEO) verbessert. Serverseitig gerenderte Seiten können von Suchmaschinen besser indexiert werden, was zu einer höheren Sichtbarkeit und besseren Platzierungen in den Suchergebnissen führt. Laut einer Studie von Google erhöht sich die Wahrscheinlichkeit, dass Nutzer auf einer Seite bleiben, um 32%, wenn diese innerhalb von drei Sekunden geladen wird. [25]

Durch die verbesserte SEO wird die Sichtbarkeit der Produkte in Suchmaschinen erhöht, was wiederum zu mehr Traffic und höheren Umsätzen führt. Empirische Untersuchungen belegen, dass eine Verbesserung der SEO die organischen Suchzugriffe um bis zu 50% steigern kann. [37][48][61]

Ein zentrales Anwendungsgebiet bei Nike ist die Optimierung der Produktpräsentation, Nutzerbewertungen und des Kaufprozesses. Besonders bei Marketingkampagnen, wie Produkteinführungen und Sonderangeboten, sind die schnelleren Ladezeiten und SEO-Verbesserungen von großem Vorteil. Next.js ermöglicht Nike, effizient Landing Pages zu erstellen, die sowohl für Nutzer als auch für Suchmaschinen optimiert sind.

Fallstudie: Ticketmaster

Ticketmaster verwendet React und Next.js zur Optimierung seiner Ticketing-Plattform und erzielt dadurch erhebliche Verbesserungen in der Geschwindigkeit und Benutzerfreundlichkeit der Webanwendung:

Die Verwendung von React ermöglicht die Erstellung reaktiver und intuitiver Benutzeroberflächen. Studien zeigen, dass reaktive Interfaces die Nutzerzufriedenheit erhöhen und die Wahrscheinlichkeit eines Ticketkaufs steigern können.

Durch Next.js werden die Seitenladezeiten drastisch reduziert, was zu einer verbesserten Nutzererfahrung führt. Laut Google geben 53% der mobilen Nutzer eine Website auf, die länger als drei Sekunden zum Laden benötig. [15][25][58][63]

> ***"The average time it takes to fully load a mobile landing page is 22 seconds, according to a new analysis. Yet 53% of visits are abandoned if a mobile site takes longer than three seconds to load."*** *(Daniel An, Februar 2017, https://www.thinkwithgoogle.com/intl/en-emea/marketing-strategies/app-and-mobile/mobile-page-speed-new-industry-benchmarks/)*

Das serverseitige Rendering verbessert sowohl die Ladezeiten als auch die SEO, wodurch die Sichtbarkeit in Suchmaschinen steigt und mehr organische Besucher generiert werden. Eine um 10% bessere SEO kann den organischen Traffic um bis zu 20% steigern. [9][23][38][60]

Ticketmaster profitiert insbesondere bei der Darstellung von Veranstaltungen, Nutzerbewertungen und im gesamten Ticketverkaufsprozess von diesen Verbesserungen. Next.js ermöglicht die effiziente Erstellung von Landing Pages, die sowohl für Nutzer als auch für Suchmaschinen optimiert sind.

Fallstudie: Netflix Jobs

Netflix nutzt React und Next.js zur Optimierung seiner Karriereseite und bietet dadurch eine dynamische und benutzerfreundliche Webanwendung:

React ermöglicht die Erstellung dynamischer und interaktiver Benutzeroberflächen, die Bewerbungsprozesse und Unternehmensinformationen ansprechend präsentieren. Empirische Untersuchungen zeigen, dass eine gut gestaltete Karriereseite die Anzahl der Bewerbungen erhöhen und die Qualität der Bewerber verbessern kann.

Next.js sorgt für schnellere Ladezeiten und eine verbesserte SEO, was die Sichtbarkeit der Stellenangebote in Suchmaschinen erhöht.

Schnelle Ladezeiten und eine gute SEO sind für Rekrutierungskampagnen besonders wichtig. Next.js ermöglicht es Netflix, effizient Landing Pages zu erstellen, die sowohl für Nutzer als auch für Suchmaschinen optimiert sind. Dies führt zu einer höheren Anzahl und Qualität der eingehenden Bewerbungen

6. Zusätzlicher Abschnitt: Vue.js und Next.js

Inkompatibilität von Vue.js und Next.js

Next.js ist speziell für React entwickelt worden und nutzt dessen Ökosystem und Paradigmen. Aufgrund der fundamentalen Unterschiede in der Architektur und den Entwicklungsansätzen von React und Vue.js sind Next.js und Vue.js nicht direkt kompatibel.

Rendering-Mechanismen:

Next.js verwendet Reacts JSX (JavaScript XML) und Virtual DOM (Document Object Model). JSX erlaubt es Entwicklern, HTML direkt in JavaScript zu schreiben, was eine reibungslose Integration von Logik und Layout ermöglicht. Der Virtual DOM von React aktualisiert nur die Teile der Benutzeroberfläche, die sich tatsächlich geändert haben, was die Performance verbessert.

Vue.js verwendet eine eigene Template-Syntax, die HTML-ähnlich ist, und ein Reaktivitätssystem, das Datenänderungen effizient im DOM widerspiegelt. Vue's Reaktivitätssystem basiert auf „Reaktiven Objekten", die automatisch erkennen, wann sich Daten ändern, und die Benutzeroberfläche entsprechend aktualisieren. Diese Unterschiede in der Handhabung von UI-Komponenten und der DOM-Manipulation machen eine direkte Kompatibilität zwischen Vue.js und Next.js unmöglich.

Ökosystem und Bibliotheken:

Next.js ist tief in das React-Ökosystem integriert. Es verwendet viele React-spezifische Bibliotheken und Tools wie React Router für das Routing und Redux für das State Management. Diese Tools sind speziell auf die Arbeitsweise von React zugeschnitten und funktionieren nicht ohne weiteres mit Vue.js

Vue.js hat sein eigenes Set an Tools und Bibliotheken, wie Vue Router für das Routing und Vuex für das State Management. Diese sind speziell auf die Bedürfnisse und die Struktur von Vue.js-Anwendungen abgestimmt und sind nicht kompatibel mit der Art und Weise, wie React und Next.js arbeiten.

Serverseitiges Rendering (SSR):

Next.js unterstützt serverseitiges Rendering (SSR) nativ. Es rendert die HTML-Inhalte auf dem Server und sendet diese an den Client. Dies verbessert die initiale Ladezeit und SEO, erfordert aber eine spezielle Infrastruktur und Implementierung.

Vue.js kann ebenfalls SSR unterstützen, jedoch ist dies nicht nativ integriert und erfordert zusätzliche Konfiguration und Bibliotheken wie Nuxt.js, um eine ähnliche Funktionalität wie Next.js zu bieten. Die Implementierung von SSR in Vue.js und Next.js unterscheidet sich erheblich in Bezug auf die zugrunde liegenden Mechanismen und die Art der Umsetzung

Alternativen und Lösungen

Nuxt.js ist ein Framework, das speziell für Vue.js entwickelt wurde und ähnliche Funktionen wie Next.js bietet, jedoch vollständig auf Vue.js zugeschnitten ist.

Nuxt.js bietet SSR für Vue.js-Anwendungen, was die Performance und SEO erheblich verbessert. Es übernimmt die serverseitige Ausführung und Client-Code-Verteilung, wodurch Entwickler sich auf die Erstellung der Anwendung konzentrieren können.

Ähnlich wie Next.js kann Nuxt.js statische Seiten zur Build-Zeit generieren, was zu schnellen Ladezeiten und besserer SEO führt.

Nuxt.js unterstützt eine modulare Architektur, die eine einfache Integration und Erweiterung ermöglicht. Es bietet automatische Importe von Komponenten und Modulen, was die Entwicklungszeit verkürzt und den Prozess effizienter gestaltet

Zusammenfassend bietet Nuxt.js eine robuste Alternative für Entwickler, die Vue.js verwenden und ähnliche Funktionen wie Next.js benötigen, jedoch vollständig in das Vue-Ökosystem integriert bleiben möchten.

7. Ausblick

7.1. Zusammenfassung der Forschungsergebnisse

Die isomorphe Webentwicklung bietet eine Hybridlösung, das serverseitige Rendering (SSR) mit clientseitiger Interaktivität kombiniert. Diese Methode ermöglicht es Entwicklern, die Vorteile von Multi-Page-Anwendungen (MPA) und Single-Page-Anwendungen (SPA) zu kombinieren, um die Benutzerfreundlichkeit und die Suchmaschinenoptimierung (SEO) zu verbessern.

Das bereitgestellte Code zeigt, wie es umgesetzt sein könnte:

```
// pages/wetter.js
import Header from "../Header";
import Footer from "../Footer";
import styles from "../page.module.css";
import WeatherDisplay from "./WeatherDisplay";
import { fetchWeatherData } from "../api/weather/route";

export const metadata = {
  title: "Wetter",
  description: "Willkommen auf der Wetter-Seite.",
};

export const dynamic = "force-dynamic";

export default async function WetterPage() {
  let weather = await fetchWeatherData();
  if (!weather || weather.error) {
    weather = await fetchWeatherData();
  }

  if (!weather) {
    return (
      <>
        <Header />
        <main className={styles.main}>
          <div>
            <p>Fehler beim Laden der Wetterdaten.</p>
          </div>
        </main>
        <Footer />
      </>
    );
  }

  return (
    <>
      <Header />
      <main className={styles.main}>
        <div>
          <h1>Willkommen auf isomorphe Wetter Seite</h1>
          <WeatherDisplay initialWeather={weather} />
        </div>
```

```
      </main>
      <Footer />
    </>
  );
}
```

Das serverseitige Rendering sorgt dafür, dass die Seite mit den bereits geladenen Daten an den Browser gesendet wird. Dadurch wird nicht nur die Ladezeit verkürzt, sondern auch sichergestellt, dass der Inhalt von Suchmaschinen beim Crawlen der Seite erkannt wird, was für die SEO-Optimierung entscheidend ist.

Die API spielt eine zentrale Rolle bei der Bereitstellung von dynamischen Echtzeitdaten. Diese Daten werden sowohl auf der Serverseite geladen, wenn die Seite zum ersten Mal angezeigt wird, als auch auf der Client-Seite, wenn der Benutzer interagiert.

```
//api/weather/route.js
import fetch from "node-fetch";

export async function fetchWeatherData() {
  const apiUrl = `https://api.openweathermap.org/data/2.5/weather?q=Dres-
den,de&appid=9263e4527fa4851adcef1f4faa5cbb7b&units=metric&lang=de&${new
Date().getTime()}`;

  try {
    const response = await fetch(apiUrl);

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    const data = await response.json();
    return data;
  } catch (error) {
    console.error("Fetch error:", error);
    return null;
  }
}
```

Der asynchrone API-Aufruf stellt sicher, dass stets aktuelle Wetterdaten geladen werden, was die Relevanz und den Nutzen der Anwendung erhöht. Die Verwendung der Fetch-Funktion ermöglicht einen effizienten und schnellen Abruf der Daten, sowohl für die initiale, initiale Server-Side-Rendering als auch für spätere Aktualisierungen durch den Benutzer.

Nach dem initialen serverseitigen Laden der Seite wird eine clientseitige Logik verwendet, um die Anwendung interaktiv zu machen. Dies geschieht mit Hilfe der Hydration-Logik in React, die es dem Seiteninhalt ermöglicht, nach dem Laden auf Benutzeraktionen zu reagieren.

```
// WeatherDisplay.js
"use client";

import React, { useState, useEffect } from "react";
import { fetchWeatherData } from "../api/weather/route";

export default function WeatherDisplay({ initialWeather }) {
  const [weather, setWeather] = useState(initialWeather);

  useEffect(() => {
    if (!weather || weather.error) {
      fetchWeatherData().then((data) => {
        if (data && !data.error) {
          setWeather(data);
        }
      });
    }
  }, []);

  if (!weather) {
    return <p>Loading weather data...</p>;
  }

  const temp = Math.round(weather.main.temp);
  const feelsLike = Math.round(weather.main.feels_like);
  const weatherDescription = weather.weather[0].description;

  return (
    <div>
      <p>
        Aktuelles Wetter in Dresden:
        <span className="weather-highlight">
          {temp} °C, {weatherDescription}
        </span>
      </p>
      <p>
        Gefühlt wie:
        <span className="weather-highlight">{feelsLike} °C</span>
      </p>
      <button
        className="update-button"
        onClick={() => fetchWeatherData().then(setWeather)}
      >
        Aktualisieren
      </button>
    </div>
  );
}
```

Durch die Verwendung von useState und useEffect können Wetterdaten nach dem Laden der Seite aktualisiert werden, ohne dass die Seite neu geladen werden muss. Dies verbessert die Benutzerinteraktion erheblich, da die Anwendung dynamisch und reaktionsfähig bleibt, ohne die Leistung zu beeinträchtigen.

Das HTML-Dokument wird vollständig auf der Serverseite gerendert und an den Client gesendet. Dadurch ist der gesamte Inhalt sofort verfügbar, was die Suchmaschinen-optimierung verbessert, da die Crawler der Suchmaschinen den gesamten Inhalt der Seite sehen und indizieren können.

```
<!DOCTYPE html>
<html lang="de">
    <head>
        <meta charSet="utf-8"/>
        <meta name="viewport" content="width=device-width, initial-scale=1"/>
        <link rel="stylesheet" href="/_next/static/css/app/lay-
out.css?v=1720903746722" data-precedence="next_static/css/app/layout.css"/>
        <link rel="stylesheet" href="/_next/static/css/app/wet-
ter/page.css?v=1720903746722" data-precedence="next_static/css/app/wet-
ter/page.css"/>
        <link rel="preload" as="script" fetchPriority="low"
href="/_next/static/chunks/webpack.js?v=1720903746722"/>
        <script type="text/javascript" src="http://gc.kis.v2.scr.kaspersky-
labs.com/FD126C42-EBFA-4E12-B309-BB3FDD723AC1/main.js?attr=hGOgF7OxKj2NkCO-
POon02TeIr0JsBAX6Fiv-uddU3lIMLYJbD5-Ou2XXm3xRMOW_" charset="UTF-8"></script>
        <link rel="stylesheet" crossorigin="anonymous"
href="http://gc.kis.v2.scr.kaspersky-labs.com/E3E8934C-235A-4B0E-825A-
35A08381A191/abn/main.css?attr=aHR0cDovL2xvY2FsaG9zdDozMDAwL3dldHRlcg"/>
        <script src="/_next/static/chunks/main-app.js?v=1720903746722"
async=""></script><script src="/_next/static/chunks/app-pages-internals.js"
async=""></script><script src="/_next/static/chunks/app/wetter/page.js"
async=""></script>
        <title>Wetter</title>
        <meta name="description" content="Willkommen auf der Wetter-Seite."/>
        <link rel="icon" href="/favicon.ico" type="image/x-icon"
sizes="16x16"/>
        <script src="/_next/static/chunks/polyfills.js" noModule=""></script>
    </head>
    <body class="__className_aaf875">
        <header>
            <nav>
                <ul>
                    <li><a href="/">Startseite</a></li>
                    <li><a href="/wetter">Wetter</a></li>
                    <li><a href="/about">Über uns</a></li>
                </ul>
            </nav>
        </header>
        <main class="page_main__nw1Wk">
```

```
            <div>
                <h1>Willkommen auf isomorphe Wetter Seite</h1>
                <div>
                    <p>
                        Aktuelles Wetter in Dresden:
                        <span class="weather-highlight">
                            19<!-- --> °C, <!-- -->Klarer Himmel
                        </span>
                    </p>
                    <p>
                        Gefühlt wie:
                        <span class="weather-highlight">
                            19<!-- --> °C
                        </span>
                    </p>
                    <button class="update-button">Aktualisieren</button>
                </div>
            </div>
        </main>
        <footer>
            <p>© 2024 Meine Website</p>
        </footer>
    </body>
</html>
```

<meta charset="utf-8"/>: Stellt sicher, dass die Seite korrekt angezeigt wird und alle Zeichen richtig interpretiert werden.

<meta name="viewport" content="width=device-width, initial-scale=1"/>: Optimiert die Darstellung der Seite auf mobilen Geräten.

<meta name="description" content="Willkommen auf Wetter Seite."/>: Liefert eine kurze Beschreibung des Seiteninhalts, die in den Suchergebnissen angezeigt wird und die Relevanz der Seite erhöht.

<link rel="preload" as="script">: Lädt wichtige JavaScript-Dateien vorab, um die Ladezeiten zu verkürzen und die Benutzererfahrung zu verbessern.

<script src="/_next/static/chunks/main-app.js?v=1720903746722" async="">: Asynchrone Skripte sorgen dafür, dass diese Skripte parallel zum Dokument geladen werden, ohne das Rendering zu blockieren. Dies hilft, die interaktive Zeit der Seite zu reduzieren.

<script src="/_next/static/chunks/polyfills.js" noModule=""></script>: Keine Blockierung durch Polyfills. Das Skript wird nur geladen, wenn die erforderlichen Module nicht unterstützt werden, was die Performance für die meisten modernen Browser verbessert.

Die Verwendung von semantischen Elementen wie <**header**>, <**nav**>, <**main**> und <**footer**> hilft den Suchmaschinen, die Struktur der Seite besser zu verstehen und die Relevanz der Inhalte besser einzuschätzen.

Die klare Verwendung von Überschriften wie <**h1**> für den Titel der Hauptseite und detaillierte Informationen in <**p**>-Tags verbessern die Lesbarkeit und damit die Suchmaschinenoptimierung.

In einer rein client-seitigen Anwendung, wie z.B. React-App, wird der Inhalt der Seite vollständig durch JavaScript im Browser des Benutzers gerendert. Dies führt zu mehreren potenziellen SEO-Problemen

In client-seitigen Anwendungen wird der sichtbare Inhalt erst im Browser durch JavaScript erzeugt. Dies kann zu erheblichen Verzögerungen führen.

Das bereitgestellte Code zeigt, wie es umgesetzt sein könnte:

```
import React from "react";
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
import "./App.css";
import Header from "./Header";
import Footer from "./Footer";
import Weather from "./Weather";
import About from "./About";

function App() {
  return (
    <Router>
      <div className="App">
        <Header />
        <main>
          <Routes>
            <Route path="/" element={<h1>Willkommen auf der Startseite</h1>}
/>
            <Route path="/weather" element={<Weather />} />
            <Route path="/about" element={<About />} />
          </Routes>
        </main>
        <Footer />
      </div>
    </Router>
  );
}

export default App;
```

```
import React, { useEffect, useState } from "react";
import "./App.css";

const Weather = () => {
  const [weather, setWeather] = useState(null);
  const [error, setError] = useState(null);

  const fetchWeather = async () => {
    try {
      const res = await fetch(
        `https://api.openweathermap.org/data/2.5/weather?q=Dresden,de&ap-
pid=9263e4527fa4851adcef1f4faa5cbb7b&units=metric&lang=de&${new Date().get-
Time()}`
      );

      if (!res.ok) {
        throw new Error(`HTTP error! status: ${res.status}`);
      }

      const data = await res.json();
      setWeather(data);
    } catch (error) {
      setError(error.message);
    }
  };

  useEffect(() => {
    fetchWeather();
  }, []);

  if (error) {
    return <div>Fehler beim Laden der Wetterdaten: {error}</div>;
  }

  if (!weather) {
    return <div>Lädt...</div>;
  }

  const temp = Math.round(weather.main.temp);
  const feelsLike = Math.round(weather.main.feels_like);
  const description = weather.weather[0].description;

  return (
    <div>
      <h1>Willkommen auf der Wetterseite React</h1>
      <div className="weather-info">
        <p>
          Aktuelles Wetter in Dresden:{" "}
          <span className="weather-highlight">
            {temp} °C, {description}
          </span>
        </p>
        <p>
          Gefühlt wie:
          <span className="weather-highlight">{feelsLike} °C</span>
```

```
        </p>
        <button onClick={fetchWeather} className="update-button">
          Aktualisieren
        </button>
      </div>
    </div>
  );
};

export default Weather;
```

Die gesamte Seite, einschließlich der Navigationswege und Inhalte, wird dynamisch durch JavaScript generiert. Dies bedeutet, dass die Zeit bis zur ersten interaktiven Anzeige erheblich sein kann, insbesondere wenn das JavaScript groß ist oder die Netzwerkbedingungen schlecht sind. Nutzer sehen möglicherweise eine leere Seite oder nur "Loading", bis das Skript vollständig geladen und ausgeführt wurde.

Die Anwendung ist erst interaktiv, nachdem das gesamte JavaScript geladen und ausgeführt wurde, was die unmittelbare Benutzerinteraktion mit der Seite verhindert. Dies kann besonders störend sein, wenn der Benutzer schnell Informationen abrufen oder eine Aktion ausführen möchte.

Das HTML-Snippet zeigt deutlich die Herausforderungen für die Suchmaschinenoptimierung:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="/logo192.png" />
    <link rel="manifest" href="/manifest.json" />
    <title>React App</title>
```

```
    <script
      type="text/javascript"
      src="http://gc.kis.v2.scr.kaspersky-labs.com/FD126C42-EBFA-4E12-B309-
BB3FDD723AC1/main.js?attr=2ie1PLVlF1NkvRtfd6ClNFDiBh85AZ-
bVVff_RZHpJ6FZcHmXHB2NTK0oKxieqXMy"
      charset="UTF-8"
    ></script>
    <link
      rel="stylesheet"
      crossorigin="anonymous"
      href="http://gc.kis.v2.scr.kaspersky-labs.com/E3E8934C-235A-4B0E-825A-
35A08381A191/abn/main.css?attr=aHR0cDovL2xvY2FsaG9zdDozMDAwL3dlYXRoZXI"
    />
    <script defer src="/static/js/bundle.js"></script>
  </head>
  <body>
    <div id="root">Loading</div>
  </body>
</html>
```

Fehlender Inhalt im HTML beim initialen Laden. Suchmaschinen, die die Ausführung von JavaScript nicht simulieren, sehen nur diese minimale Struktur ohne kontextuellen Inhalt. Dies beeinträchtigt das Ranking der Seite in Suchmaschinen, da kein Textinhalt indiziert werden kann.
Die Meta-Beschreibung ist nicht nur generisch, sondern enthält auch keine spezifischen Informationen über die Seite oder ihren Inhalt, was die Klickrate in den Suchergebnissen verringern kann.

Die bereitgestellten Code-Beispiele und dazugehörigen HTML-Scripts illustrieren eindrucksvoll, wie isomorphe Webentwicklung die Vorteile des serverseitigen Renderings (SSR) mit der clientseitigen Interaktivität vereint. Diese Technik verbessert sowohl die Benutzerfreundlichkeit als auch die Suchmaschinenoptimierung, indem sie die Inhalte der Seite sowohl für Nutzer als auch für Suchmaschinen-Crawler sofort zugänglich macht. Auf diese Weise wird sichergestellt, dass die Inhalte bereits beim ersten Laden der Seite verfügbar sind und von Suchmaschinen effektiv indiziert werden können, was die Sichtbarkeit der Webseite erheblich steigert.

In der Diskussion über isomorphe Webentwicklung und deren Einfluss auf die Benutzerfreundlichkeit und Suchmaschinenoptimierung (SEO) lässt sich das spezifische Ladeverhalten von serverseitigem Rendering (SSR) und client-seitigem Rendering (SPA) anhand von Next.js und React illustrieren

Die Verwendung von Next.js für serverseitiges Rendering bietet entscheidende Vorteile, indem die Seite bereits mit den erforderlichen Inhalten auf dem Server gerendert und als statisches HTML an den Browser gesendet wird. Dies erfolgt unabhängig von der Geschwindigkeit der Internetverbindung des Benutzers. Der im HTML eingebettete Inhalt, einschließlich der durch serverseitige API-Aufrufe bereitgestellten Daten, wird direkt geladen. Der Inhalt ist somit fast sofort nach dem Seitenaufruf sichtbar

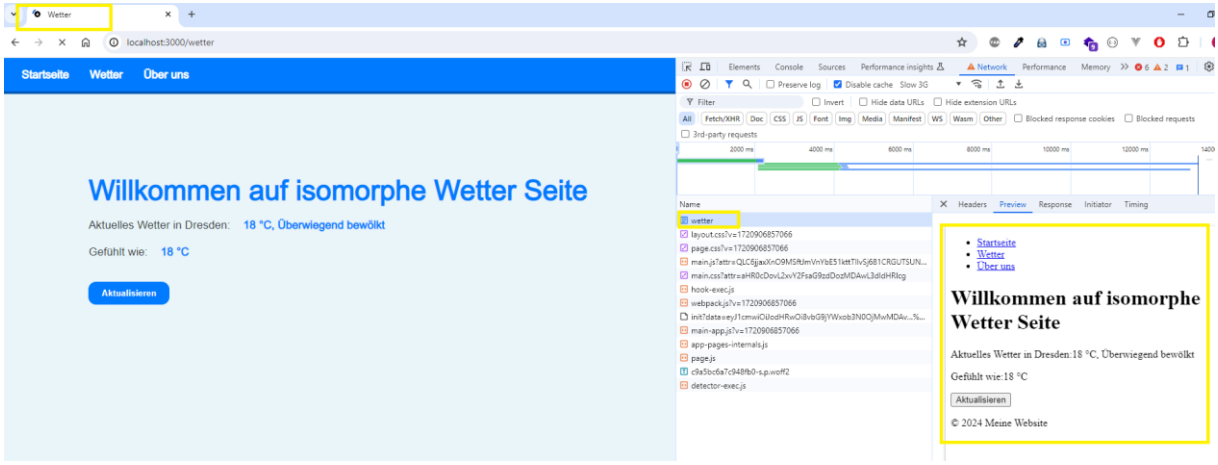


Abbildung 16: isomorphe App noch beim Laden

Sobald die zusätzlichen clientseitigen Skripte und Stylesheets nachgeladen sind, erreicht die Seite ihre volle Interaktivität

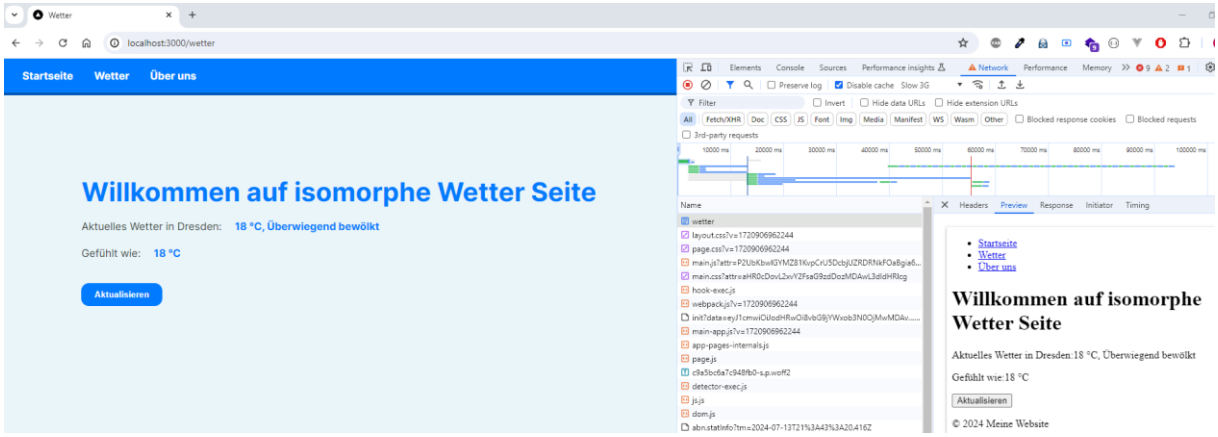


Abbildung 17: isomorphe App bereits geladen

Im Gegensatz dazu zeigt eine React-basierte Single Page Application (SPA) ein anderes Ladeverhalten. Beim client-seitigen Rendering wird initial nur ein minimales HTML-Gerüst geladen, das hauptsächlich Links zu JavaScript enthält. Der gesamte Content wird im Browser durch das Ausführen von JavaScript dynamisch generiert. Bei langsamen Verbindungen führt dies zu einer initialen Anzeige einer leeren Seite mit einem "Loading"-Hinweis

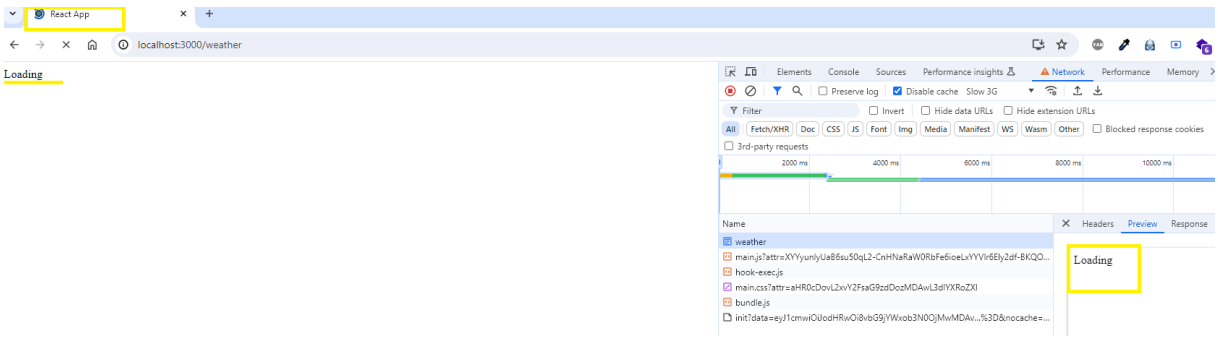


Abbildung 18: SPA beim Laden

Erst nachdem alle JavaScript-Pakete heruntergeladen und ausgeführt wurden, wird der Content sichtbar und interaktiv

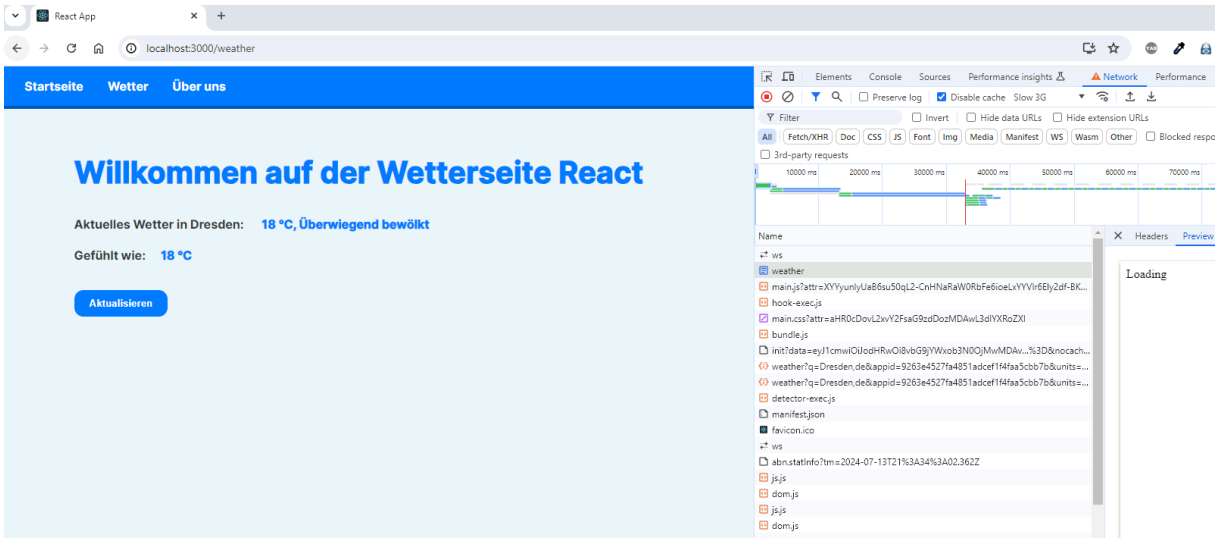


Abbildung 19: SPA bereits geladen

Diese spezifischen Beispiele verdeutlichen die Vorteile der isomorphen Webentwicklung. Durch das serverseitige Rendering wird nicht nur die Ladezeit verkürzt, sondern auch sichergestellt, dass der Inhalt von Suchmaschinen beim Crawlen der Seite effektiv erkannt und indiziert wird. Dies verbessert die Sichtbarkeit und Auffindbarkeit der

Webseite in Suchmaschinen. Der client-seitige SPA-Ansatz bietet zwar mehr Flexibilität und Dynamik für interaktive Anwendungen, setzt jedoch eine robustere Internetverbindung voraus und kann bei langsamen Verbindungen zu einer suboptimalen Nutzererfahrung führen.

7.2. Kritische Bewertung der isomorphen Anwendungen

Isomorphe Anwendungen bieten eine vielversprechende Lösung für moderne Webanwendungen, indem sie sowohl serverseitiges Rendering (SSR) als auch clientseitige Interaktivität kombinieren. Dies führt zu einer verbesserten Suchmaschinenoptimierung (SEO) und einer schnelleren initialen Ladezeit, was die Benutzererfahrung erheblich verbessert. Dennoch gibt es einige Herausforderungen und kritische Punkte, die berücksichtigt werden müssen.

Ein wesentlicher Vorteil der isomorphen Anwendungen liegt in der SEO-Optimierung. Da der HTML-Inhalt auf dem Server gerendert wird, können Suchmaschinen den Inhalt leichter durchsuchen und indexieren, was zu einer besseren Sichtbarkeit und einem höheren Ranking in den Suchergebnissen führt. Zudem sorgt das serverseitige Rendering für eine schnellere Ladezeit, da der Benutzer den Inhalt sofort sehen kann, ohne auf das Laden und Ausführen von JavaScript warten zu müssen. Eine robuste Architektur ist ein weiterer Vorteil, da SSR eine stabile Struktur bietet, die die Verwaltung und Skalierung komplexer Anwendungen vereinfacht. Der Server übernimmt den Großteil der Rendering-Arbeit, während der Client für die Interaktivität zuständig ist.

Jedoch bringt die Implementierung isomorpher Anwendungen auch Herausforderungen mit sich. Die Entwicklung ist komplex und erfordert ein tiefes Verständnis sowohl der server- als auch der clientseitigen Technologien. Der Code muss in beiden Umgebungen funktionieren, was zusätzlichen Entwicklungs- und Testaufwand bedeutet. Darüber hinaus kann die erhöhte Serverlast ein Problem darstellen, da der Server für das Rendern jeder Seite verantwortlich ist. Dies erfordert eine gut skalierbare Infrastruktur und effiziente Caching-Strategien, um die Leistung aufrechtzuerhalten. Die Synchronisation zwischen Client und Server stellt ebenfalls eine Herausforderung dar, da die Logik so geschrieben werden muss, dass sie in beiden Umgebungen funktioniert. Dies kann komplex sein und erfordert sorgfältige Planung und Design, um eine effiziente und konsistente Anwendung zu gewährleisten.

Ein weiteres Risiko bei der Entwicklung isomorpher Anwendungen sind die Sicherheitsaspekte. Obwohl SSR die Sicherheitslage verbessert, indem weniger clientseitiges JavaScript ausgeführt wird, muss dennoch darauf geachtet werden, dass sowohl serverseitige als auch clientseitige Sicherheitsmaßnahmen implementiert werden. Serverseitiges Rendering kann Sicherheitsvorteile bieten, aber es muss sichergestellt werden, dass serverseitige Eingaben validiert und bereinigt werden, um potenzielle Schwachstellen wie SQL-Injection oder Server-Side Template Injection (SSTI) zu verhindern. SSTI ist eine Sicherheitslücke, die auftritt, wenn unsichere Benutzereingaben direkt in Templates eingebettet werden, was es Angreifern ermöglicht, beliebigen Code auf dem Server auszuführen. Auf der Client-Seite müssen Mechanismen wie Content Security Policy (CSP) und Cross-Site Scripting (XSS) Schutzmaßnahmen implementiert werden, um potenzielle Angriffe zu verhindern. Eine sorgfältige Validierung und Bereinigung aller Eingaben ist dabei von zentraler Bedeutung.

7.3. Zukünftige Entwicklungen und Forschungspotentiale

Die isomorphe Webentwicklung steckt noch in der Entwicklung Stadium und es gibt noch viel Raum für zukünftige Entwicklungen. Dazu gehören verbesserte Werkzeuge und Plattformen, um die Entwicklung zu vereinfachen, sowie weitere Forschung im Bereich der Leistungs- und Suchmaschinenoptimierung.

In Zukunft könnten verbesserte Werkzeuge und Plattformen die Implementierung und Unterstützung isomorpher Anwendungen erleichtern. Dies kann die Entwicklung neuer Werkzeuge und Bibliotheken beinhalten, die speziell auf die Bedürfnisse isomorpher Anwendungen zugeschnitten sind. Diese Werkzeuge können Debugging-Prozesse verbessern, Build-Prozesse vereinfachen und die Integration neuer Technologien erleichtern.

Ein weiterer Forschungsbereich ist die Performance-Optimierung. Dazu gehören die Optimierung der Rendergeschwindigkeit und die Verringerung der Serverlast durch verbesserte Algorithmen und eine effizientere Nutzung der Ressourcen. Auch fortgeschrittene Caching-Strategien können eingesetzt werden, um die Serverlast zu verringern und die Antwortzeiten zu verbessern. Dies kann sowohl serverseitiges Caching als auch clientseitiges Caching umfassen, um Inhalte effizient im Browser-Cache zu speichern und so die Ladezeiten weiter zu verkürzen.

Auch im Bereich der Suchmaschinenoptimierung besteht ein großes Potenzial für zukünftige Forschung. Weitere Entwicklungen könnten SEO-Techniken umfassen, die speziell auf die Bedürfnisse isomorpher Anwendungen zugeschnitten sind und deren Sichtbarkeit und Platzierung in Suchmaschinen weiter verbessern. Dies würde auch eine effiziente Verwaltung und Präsentation von Inhalten beinhalten, um die Relevanz und Qualität der indexierten Seiten zu maximieren.

Die Optimierung der Infrastruktur spielt ebenfalls eine wichtige Rolle. Der Einsatz von Cloud-Lösungen und Lastausgleichsmechanismen kann dazu beitragen, dass die Infrastruktur skalierbar ist und hohen Belastungen standhält. Durch automatisierte Tests kann eine robuste Testumgebung geschaffen werden, die sowohl Server- als auch Client-Komponenten umfasst und die Qualität und Zuverlässigkeit der Anwendung sicherstellt.

8. Fazit

Die Analyse der isomorphen Webentwicklung hat gezeigt, dass diese Technologie ein bedeutendes Potenzial zur Optimierung moderner Webanwendungen bietet. Durch die Kombination von serverseitigem Rendering (SSR) und clientseitiger Interaktivität können sowohl die Benutzerfreundlichkeit als auch die Suchmaschinenoptimierung (SEO) erheblich verbessert werden. Dies führt zu schnelleren Ladezeiten und einer besseren Sichtbarkeit in Suchmaschinen.

Isomorphe Anwendungen sind besonders vorteilhaft für E-Commerce-Websites, die auf eine hohe Sichtbarkeit und schnelle Ladezeiten angewiesen sind, um potenzielle Kunden anzuziehen und die Conversion-Raten zu steigern. Nachrichtenportale und Blogs profitieren ebenfalls, da die schnelle Bereitstellung von Inhalten die Leserbindung erhöht und die Aktualität der Informationen gewährleistet. Soziale Netzwerke und interaktive Anwendungen profitieren von der reaktionsschnellen und nahtlosen Benutzererfahrung, die isomorphe Anwendungen bieten.

Dennoch ist der Einsatz isomorpher Anwendungen nicht immer die beste Wahl. In Szenarien, in denen SEO keine Rolle spielt und die Anwendung hauptsächlich von einer geschlossenen Benutzergruppe genutzt wird, wie bei internen Unternehmensanwendungen oder komplexen Dashboard-Tools, könnte der zusätzliche Entwicklungsaufwand und die erhöhte Serverlast nicht gerechtfertigt sein. In solchen Fällen können Single-Page-Applications (SPAs) oder traditionelle Multi-Page-Applications (MPAs) die geeigneteren Alternativen sein, um die Komplexität und Ressourcenanforderungen zu minimieren.

Die Implementierung isomorpher Anwendungen erfordert ein tiefes technisches Verständnis und eine sorgfältige Planung. Die Herausforderungen, einschließlich der erhöhten Serverlast und der Sicherheitsrisiken, müssen durch geeignete Maßnahmen und Strategien adressiert werden. Eine fortlaufende Weiterentwicklung und Optimierung der Technologien und Werkzeuge, die isomorphe Anwendungen unterstützen, wird entscheidend sein, um deren Effizienz und Benutzerfreundlichkeit weiter zu steigern.

Zukünftige Forschungs- und Entwicklungsarbeiten sollten sich darauf konzentrieren, die bestehenden Werkzeuge und Plattformen zu verbessern, um die Implementierung isomorpher Anwendungen zu erleichtern. Performance-Optimierungen und

fortschrittliche Caching-Strategien können dazu beitragen, die Serverlast zu reduzieren und die Ladezeiten weiter zu verkürzen. Die Weiterentwicklung von SEO-Techniken, die speziell auf isomorphe Anwendungen zugeschnitten sind, wird ebenfalls von großer Bedeutung sein.

Insgesamt haben isomorphe Anwendungen das Potenzial, die Webentwicklung grundlegend zu transformieren. Sie bieten eine effektive Methode zur Erstellung leistungsstarker und benutzerfreundlicher Webanwendungen. Mit kontinuierlicher Forschung und technologischen Innovationen können die Vorteile dieser Technologie vollständig genutzt werden, was neue Maßstäbe in der Webentwicklung setzen wird.

9. Literaturverzeichnis

[1] Alexandra Mendes (22 Januar 2022): Single page applications - the future of web applications [Online, Zugriff: 21.07.2024]. Verfügbar unter: https://www.imaginarycloud.com/blog/single-page-applications/

[2] Aris Pattakos (26 November 2011): Angular vs React vs Vue: Which Framework Is Better? 2024. [Online, Zugriff: 08.06.2024]. Verfügbar unter: https://athemes.com/guides/angular-vs-react-vs-vue/

[3] Aurelio De Rosa (25 Februar 2015): Isomorphic JavaScript Applications — the Future of the Web? [Online, Zugriff: 21.07.2024]. Verfügbar unter: https://www.sitepoint.com/isomorphic-javascript-applications/

[4] Balázs Orbán, Delba de Oliveira, DongYoon Kang, Jiachi Liu, JJ Kasper, Lee Robinson, Maia Teegarden, Sebastian Markbåge, Shu Ding, Steven und Tim Neutkens (25 Oktober 2022): Next.js 13 | Next.js [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://nextjs.org/blog/next-13#server-components

[5] Benjamin Jung, Homepage-Webhilfe (29 Dezember 2013): Einführung - ASP.NET - Homepage-Webhilfe. [Online, Zugriff: 08.06.2024]. Verfügbar unter: https://www.homepage-webhilfe.de/ASP-NET/

[6] Benjamin Jung, Homepage-Webhilfe (29 Dezember 2013): Einführung - JavaScript - Homepage-Webhilfe. [Online, Zugriff: 07.06.2024]. Verfügbar unter: https://www.homepage-webhil-fe.de/JavaScript/#:~:text=Die%20erste%20Version%20von%20JavaScript,vom%20Microsoft%20Internet%20Explorer%20unterst%C3%BCtzt.

[7] Bhaval Patel (17 Juli 2023): SPA vs MPA: 8 Key Comparisons for Your Web Development [Online, Zugriff: 21.07.2024]. Verfügbar unter: https://www.spaceotechnologies.com/blog/single-page-application-vs-multi-page-application/

[8] Bornfight (17 Oktober 2019): Nuxt.js over Vue.js: when should you use it and why [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://www.bornfight.com/blog/nuxt-js-over-vue-js-when-should-you-use-it-and-why/

[9] Brian Dean (30 Juli 2019): 17 Ways to Improve SEO Rankings [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://backlinko.com/improve-your-seo

[10] Chimezie Innocent (10 April 2024): SvelteKit vs. Next.js: Which Should You Choose in 2024? [Online, Zugriff: 18.07.2024]. Verfügbar unter: https://prismic.io/blog/sveltekit-vs-nextjs

[11] Chris Fritz und Evan You (01 August 2016): GitHub - vuejs/vue: This is the repo for Vue 2. For Vue 3, go to https://github.com/vuejs/core [Online, Zugriff: 21.07.2024]. Verfügbar unter: https://github.com/vuejs/vue

[12] Christian Liebel (28 Februar 2017): App-Entwicklung mit JavaScript, Teil 4: Eine universelle Sprache. [Online, Zugriff: 15.06.2024]. Verfügbar unter: https://www.heise.de/blog/App-Entwicklung-mit-JavaScript-Teil-4-Eine-universelle-Sprache-3637035.html

[13] Cloudpso (09 Mai 2024): Angular vs React vs Vue: Which Framework to Choose in 2024? [Online, Zugriff: 08.06.2024]. Verfügbar unter: https://cloudpso.com/angular-vs-react-vs-vue-which-framework-to-choose-in-2024/

[14] Conductor Team (9 Dezember 2023): Crawler [Online, Zugriff: 23.07.2024]. Verfügbar unter: https://www.conductor.com/de/academy/glossar/crawler/

[15] Daniel An (Februar 2017): Find out how you stack up to new industry benchmarks for mobile page speed [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://www.thinkwithgoogle.com/intl/en-emea/marketing-strategies/app-and-mobile/mobile-page-speed-new-industry-benchmarks/

[16] di Matthäus Lirosi (1 April 2024): Es geschah heute – 1. April 2004: Google startet Gmail, den E-Mail-Dienst, der die Welt erobert hat. [Online, Zugriff: 15.06.2024]. Verfügbar unter: https://www.firstonline.info/de/accadde-oggi-1-aprile-2004-google-lancia-gmail-il-servizio-di-posta-elettronica-che-ha-conquistato-il-mondo/

[17] Droptica (20 Juni 2023): Next.js and React. How to Connect Them, and Why it May Be Beneficial? [Online, Zugriff: 15.02.2024]. Verfügbar unter: https://droptica.medium.com/next-js-and-react-how-to-connect-them-and-why-it-may-be-beneficial-86dec0d907a2/

[18] Eduard (28 Dezember 2017): What is an isomorphic application? [Online, Zugriff: 10.März.2024]. Verfügbar unter: https://stackoverflow.com/questions/48008502/what-is-an-isomorphic-application

[19] Fabian Heihoff (9 Juni 2022): Next.js - Das React Framework für moderne Webanwendungen. [Online, Zugriff: 15.02.2024]. Verfügbar unter: https://www.udemy.com/course/nextjs-react/

[20] Freddy Montes (7 Februar 2021): https://fmontes.com/blog/difference-between-reactjs-and-nextjs [Online, Zugriff: 02.06.2024]. Verfügbar unter: https://fmontes.com/blog/difference-between-reactjs-and-nextjs

[21] Gartner (10 Oktober 2022): Gartner Forecasts Worldwide IT Spending to Grow 5.1% in 2023 [Online, Zugriff: 21.07.2024]. Verfügbar unter: https://www.gartner.com/en/newsroom/press-releases/2022-10-19-gartner-forecasts-worldwide-it-spending-to-grow-5-percent-in-2023

[22] Internet Archive (20 Oktober 2001): Wayback Machine [Online, Zugriff: 23.07.2024]. Verfügbar unter: https://web.archive.org/

[23] Ivana V. (06 Februar 2024): 70+ SEO Statistics to Help You Land on the First Page in 2024 [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://www.smallbizgenius.net/by-the-numbers/seo-statistics/

[24] Jan Schulte (27 Mai 2013): Single Page Applications (SPA): Erklärung, Vorteile und Beispiele. [Online, Zugriff: 15.06.2024]. Verfügbar unter: https://www.magnolia-cms.com/de_DE/blog/all-about-single-page-applications.html

[25] Jay Castro (13 Oktober 2016): Find out how you stack up to new industry benchmarks for mobile page speed [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://blog.google/products/admanager/increase-speed-of-your-mobile-site-wi/

[26] Jenna Thorne (22 März 2023): Single Page Applications (SPA) Vs. Multi-Page Applications (MPA) [Online, Zugriff: 21.07.2024]. Verfügbar unter: https://blog.openreplay.com/single-page-apps-vs-multiple-page-apps/

[27] Jerzy Kozieł (11 Juli 2023): Angular vs. React? Welche Technologie ist für größere Unternehmen die langfristigere Lösung? [Online, Zugriff: 20.07.2024].

Verfügbar unter: https://vmsoftwarehouse.de/angular-vs-react-welche-technologie-ist-fur-grosere-unternehmen-die-langfristigere-losung

[28] Jigar Shah (21 März 2022): Angular vs React vs Vue – Navigating the Best Fit for You. [Online, Zugriff: 08.06.2024]. Verfügbar unter: https://wpwebinfotech.com/blog/angular-vs-react-vs-vue/

[29] Jonathan Creamer (21 April 2015): React To The Future With Isomorphic Apps [Online, Zugriff: 21.07.2024]. Verfügbar unter: https://www.smashingmagazine.com/2015/04/react-to-the-future-with-isomorphic-apps/

[30] Jordan Irabor (03 Februar 2020): Build an isomorphic application with Nuxt.js and Node [Online, Zugriff: 21.07.2024]. Verfügbar unter: https://blog.logrocket.com/build-an-isomorphic-application-with-nuxt-js-and-node/

[31] Jude Miracle (21 März 2024): Next.js vs. Nuxt.js: Ultimate guide [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://blog.logrocket.com/next-js-vs-nuxt-js

[32] Komal Rathi (28 September 2023): Next.js vs Vue.js: Choosing the Right Framework for Your Project [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://www.reveation.io/blog/nextjs-vs-vuejs/

[33] Kyle Man und Shab Hadi (21 März 2024): The impact of website design on user engagement and conversion rates [Online, Zugriff: 15.02.2024]. Verfügbar unter: https://owdt.com/article/the-impact-of-website-design-on-user-engagement-and-conversion-rates/

[34] Lee Robinson und Tim Neutkens (26 Oktober 2023): Next.js 14 [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://nextjs.org/blog/next-14#partial-prerendering

[35] Margarita Loktionova (10 April 2024): 96 Content Marketing Statistics You Need to Know for 2024 [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://www.semrush.com/blog/content-marketing-statistics/

[36] Matt G. Southern (28 Februar 2017): Google: New Industry Benchmarks for Mobile Page Speed [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://www.searchenginejournal.com/google-new-industry-benchmarks-mobile-page-speed/187777/

[37] Matthew Howells-Barby (29 Januar 2016): How We Increased Organic Traffic by Over 50% Using Technical SEO Updates [Online, Zugriff: 13.06.2024]. Verfügbar unter: https://blog.hubspot.com/marketing/technical-seo

[38] Matteo Duò (03 März 2022): 60 SEO Tips to Grow Your Organic Traffic by 280% [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://kinsta.com/blog/seo-tips/

[39] Md Whaiduzzaman, Adnan Sakib, Nisha Jaman Khan, Sudipto Chaki, Labiba Shahrier, Sudipto Ghosh, Md. Saifur Rahman, Md. Julkar Nayeen Mahi, Alistair Barros, Colin Fidge, Scott Thompson-Whiteside und Tony Jan (11 November 2023): Concept to Reality: An Integrated Approach to Testing Software User Interfaces [Online, Zugriff: 13.06.2024]. Verfügbar unter: https://www.mdpi.com/2076-3417/13/21/11997

[40] Miško Hevery (01 Oktober 2014): GitHub - angular/angular: Deliver web apps with confidence [Online, Zugriff: 21.07.2024]. Verfügbar unter: https://github.com/angular/angular

[41] MK Usmaan (29 Januar 2024): Angular vs. React vs. Vue: Ein detaillierter Vergleich im Jahr 2024. [Online, Zugriff: 08.06.2024]. Verfügbar unter: https://techlasi.com/savvy/angular-vs-react-vs-vue/

[42] Natalia Venditto (04 Januar 2023): JavaScript Isomorphism [Online, Zugriff: 03.März.2024]. Verfügbar unter: https://microfrontend.dev/frameworks/javascript-isomorphism/

[43] Nadine Noack (29 Februar 2024): React, Angular und Vue.js: Eine Gegenüberstellung von Frontend-Frameworks [Online, Zugriff: 18 April 2024]. Verfügbar unter: https://www.dotnetpro.de/frontend/user-interface/react-angular-vuejs-gegenueberstellung-frontend-frameworks-2910346.html

[44] Next.js Team (31 August 2011): The React Framework for the Web [Online, Zugriff: 21.07.2024]. Verfügbar unter: https://nextjs.org/

[45] Nicholas Mendez (15 Januar 2021): Understanding Rendering in Web Apps: SPA vs MPA [Online, Zugriff: 14.07.2024]. Verfügbar unter: https://dev.to/snickdx/understanding-rendering-in-web-apps-spa-vs-mpa-49ef

[46] Niko Köbler (28 April 2016): Isomorphe JavaScript-Webanwendungen mit Java (EE) und React.JS. [Online, Zugriff: 15.02.2024]. Verfügbar unter: https://jaxenter.de/isomorphe-javascript-webanwendungen-mit-java-ee-und-react-js-38266

[47] Nils Hartmann (12 Mai 2022): React 18 unter der Lupe: Concurrent Rendering und weitere Neuigkeiten [Online, Zugriff: 20.07.2024]. Verfügbar unter: https://entwickler.de/react/react-18-neue-features#:~:text=Durch%20das%20Concurrent%20Rendering%20kann,entspricht%20also%20dem%20bisherigen%20Verhalten

[48] OnPrintShop Team (05 Juni 2023): 50% Increased Website Visits for PrintAlert, after Collaboration with OnPrintShop [Online, Zugriff: 14.06.2024]. Verfügbar unter: https://onprintshop.com/success-stories/printalerts-website-visits-surge-by-50-percent-post-onprintshop-collaboration

[49] Paul O'Shannessy (29 Mai 2013): GitHub - facebook/react: The library for web and native user interfaces. [Online, Zugriff: 21.07.2024]. Verfügbar unter: https://github.com/facebook/react

[50] Pick (01 Juli 2020): GitHub - vuejs/core: Vue.js is a progressive, incrementally-adoptable JavaScript framework for building UI on the web. [Online, Zugriff: 21.07.2024]. Verfügbar unter: https://github.com/vuejs/core

[51] Probir Sarkar (12 April 2021): Best Next.js Libraries and Tools in 2024 [Online, Zugriff: 19.07.2024]. Verfügbar unter: https://dev.to/probir-sarkar/best-nextjs-libraries-and-tools-in-2024-4j4b

[52] Rana Hasnain Khan (30 December 2021): What Is Isomorphic React [Online, Zugriff: 21.07.2024]. Verfügbar unter: https://www.delftstack.com/howto/react/isomorphic-react/

[53] reactjs (26 Januar 2023): React. The library for web and native user interfaces [Online, Zugriff: 19.07.2024]. Verfügbar unter: https://react.dev/

[54] React Native (25 Februar 2020): React Native. Learn once, write anywhere. [Online, Zugriff: 20.07.2024]. Verfügbar unter: https://reactnative.dev/

[55] Redaktion Deutschland (17 Dezember 2015): Wann und wo ging die erste Website der Welt im World Wide Web online? Ein deutscher Forscher erzählt

vom Beginn des Internets. [Online, Zugriff: 05.06.2024]. Verfügbar unter: https://www.deutschland.de/de/topic/wirtschaft/innovation-technik/die-erste-website-der-welt#:~:text=%E2%80%9C%20Am%2020.,Lee%20gilt%20als%20sein%20Begr%C3%BCnder

[56] Rollbar Editorial Team (13 November 2023): Next.js or Vite.js: Which Framework is Better, and When? [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://rollbar.com/blog/nextjs-vs-vitejs/

[57] Sebastian Markbåge und Tim Neutkens (04 Mai 2023): Next.js 13.4 [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://nextjs.org/blog/next-13-4#turbopack-beta

[58] ScientiaMobile Team (27 März 2017): 53% of Mobile Site Visitors Abandon if it Takes More Than 3 Seconds to Load Page [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://www.scientiamobile.com/mobile-site-visitors-abandon-more-than-3-seconds/

[59] Shannon Jackson-Barnes (16 November 2023): SPA Vs. MPA Explained: Differences Between Single-Page Applications and Multi-Page Applications [Online, Zugriff: 21.07.2024]. Verfügbar unter: https://www.orientsoftware.com/blog/spa-vs-mpa/

[60] Si Quan Ong (18 März 2024): 124 SEO Statistics for 2024 [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://ahrefs.com/blog/seo-statistics/

[61] Srdjan Stojadinovic (29 März 2024): Fintech SEO Case Study: How We Grew Fintech's Organic Traffic by 73.9% in Just 3 Months [Online, Zugriff: 13.06.2024]. Verfügbar unter: https://www.omnius.so/blog/fintech-seo-case-study

[62] The PHP Group (20 Juni 2029): PHP: Die Geschichte von PHP - Manual. [Online, Zugriff: 10.06.2024]. Verfügbar unter: https://www.php.net/manual/de/history.php.php

[63] Think with Google (September 2016): The need for mobile speed: How mobile latency impacts publisher revenue [Online, Zugriff: 22.07.2024]. Verfügbar unter: https://www.thinkwithgoogle.com/intl/en-emea/marketing-strategies/app-

and-mobile/need-mobile-speed-how-mobile-latency-impacts-publisher-revenue/

[64] Vercel Ship (23 Mai 2024): The web framework for when it matters [Online, Zugriff: 14.07.2024]. Verfügbar unter: https://nextjs.org/showcase

[65] Wikipedia (15 Juni 2017): Isomorphic JavaScript [Online, Zugriff: 03.März.2024]. Verfügbar unter: https://en.wikipedia.org/wiki/Isomorphic_JavaScript

[66] Yocum Technology Group (23 Februar 2024): The Power of Interactive Elements: How to Boost User Engagement in UX Design [Online, Zugriff: 22.02.2024]. Verfügbar unter: https://www.ytg.io/blog/boost-ux-engagement

[67] Ларичев, Антон (14 Mai 2021): The impact of website design on user engagement and conversion rates [Online, Zugriff: 28.02.2024]. Verfügbar unter: https://www.udemy.com/course/react-nextjs/