



---

# Ausblick: React.js

---

*Patrick Oster B. Eng.*

*CEO & Co-Founder of muf:fin*

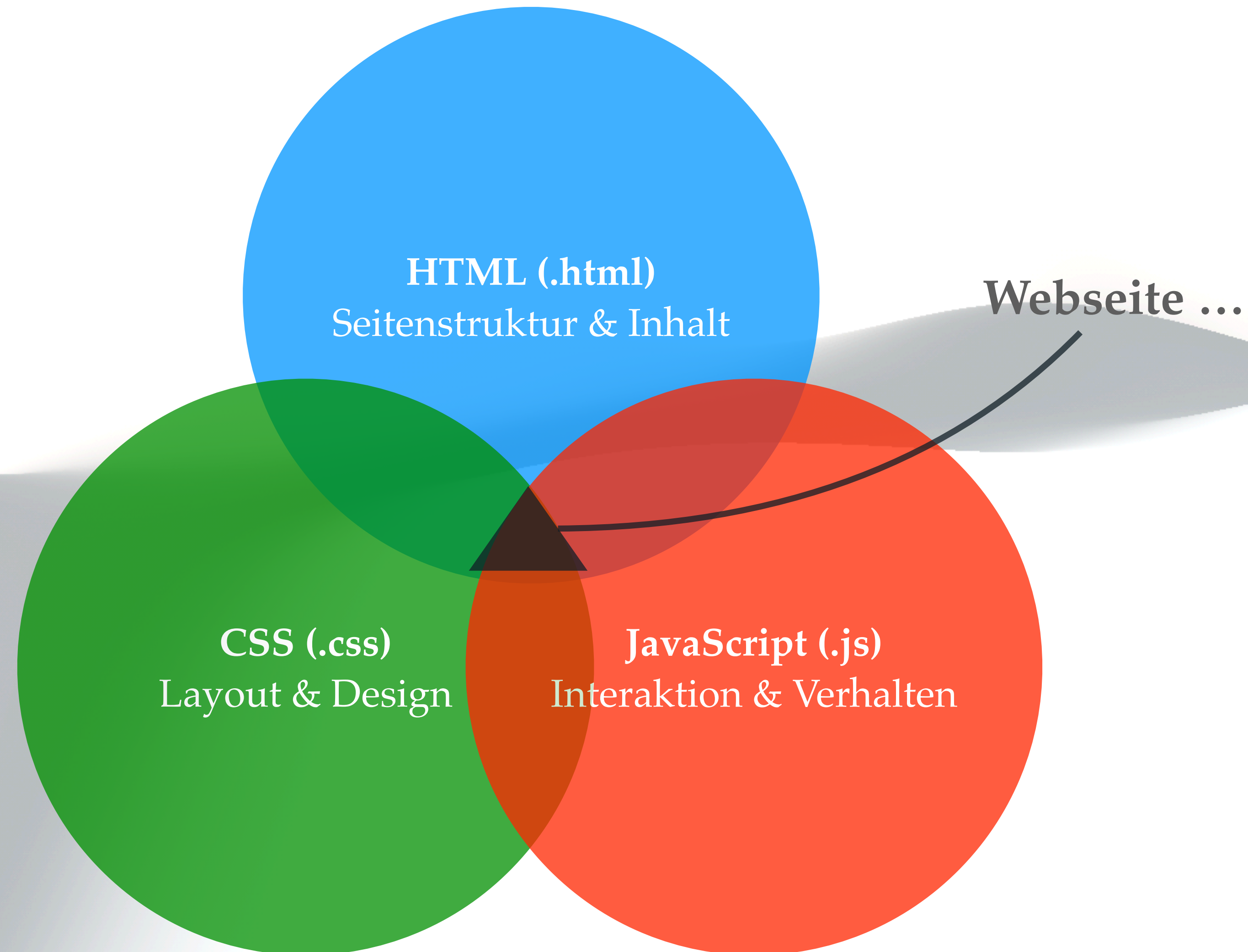
*Professional Trader & Investor*

*Data-Engineer & -Analyst*

*Vocational- & Work-Educator*

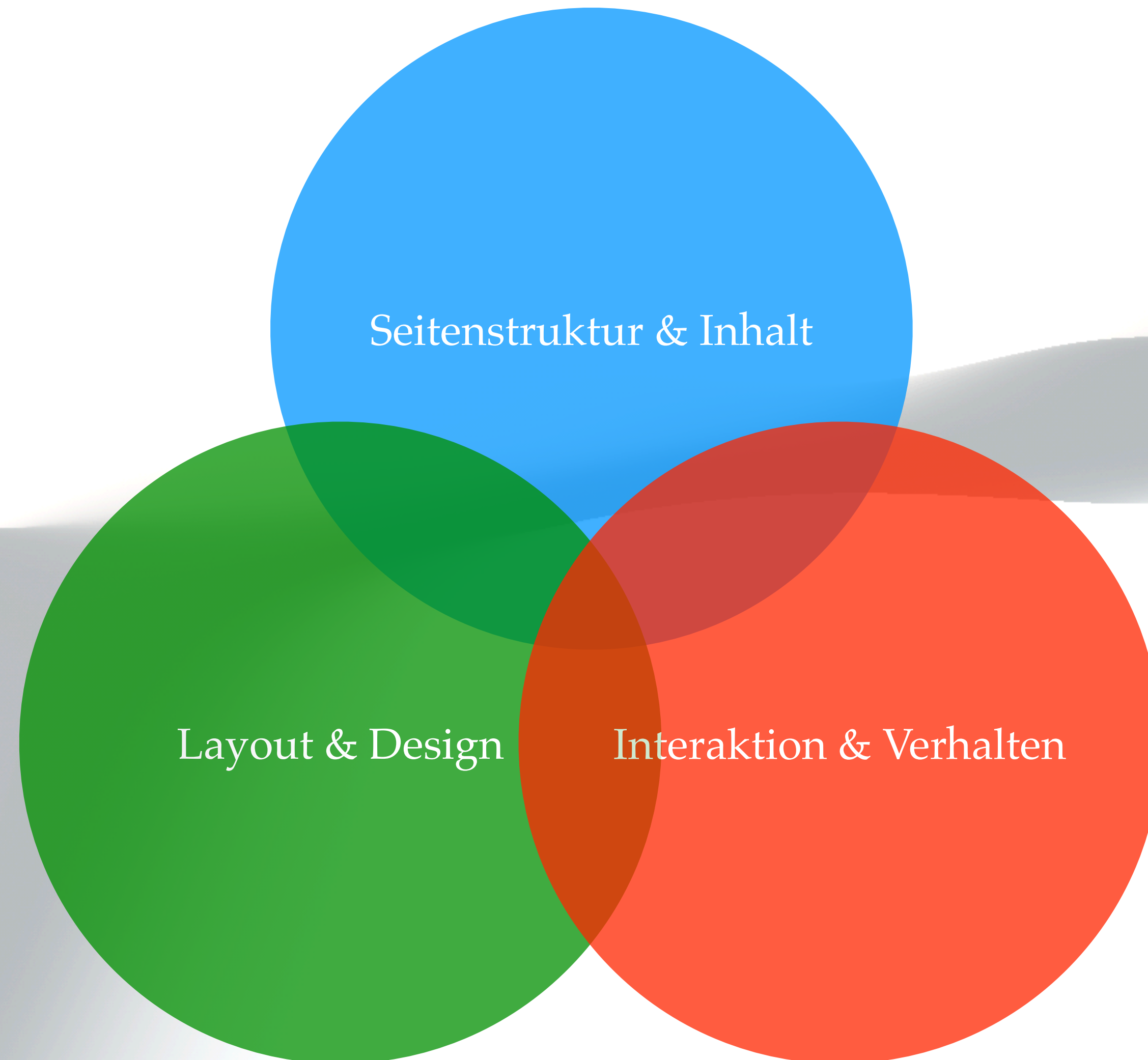
---

# Exkurs: Webentwicklung ...

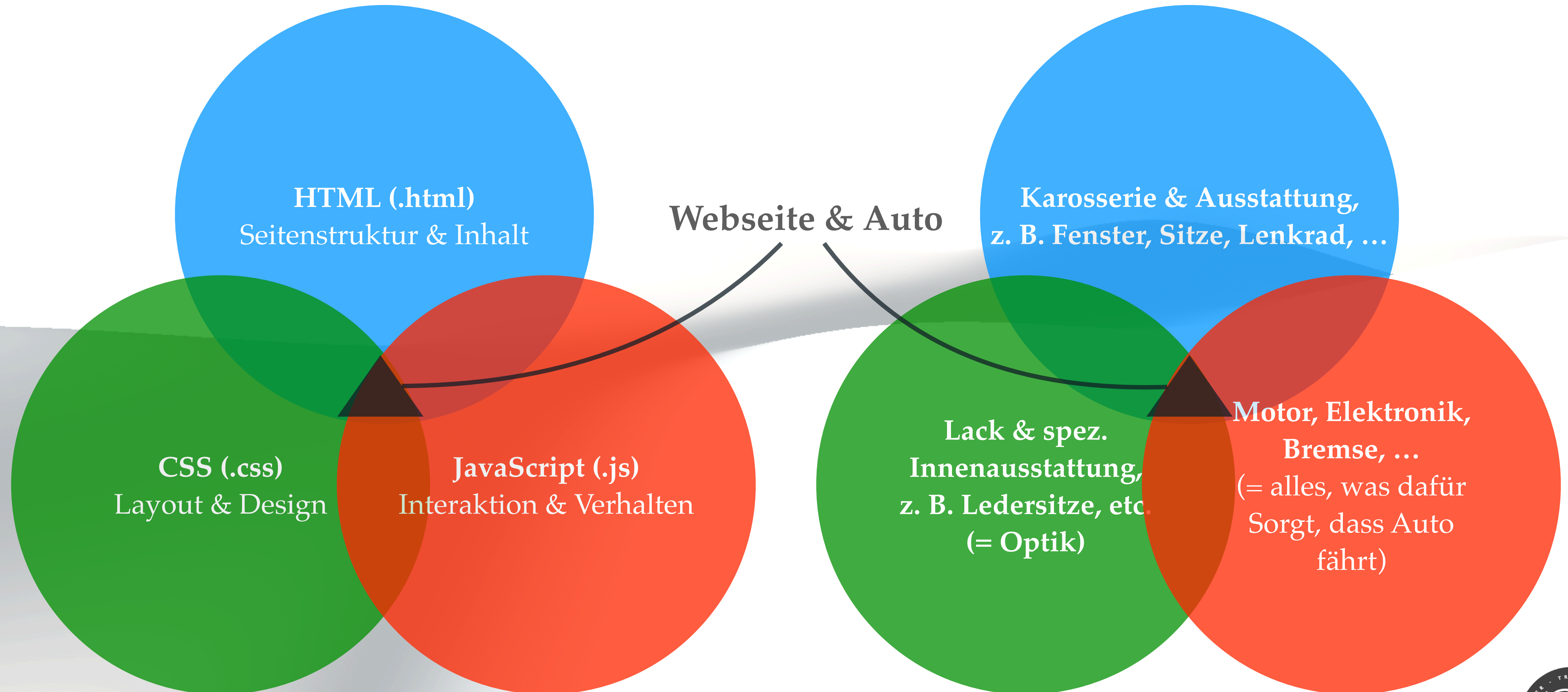




# Exkurs: Webentwicklung ...

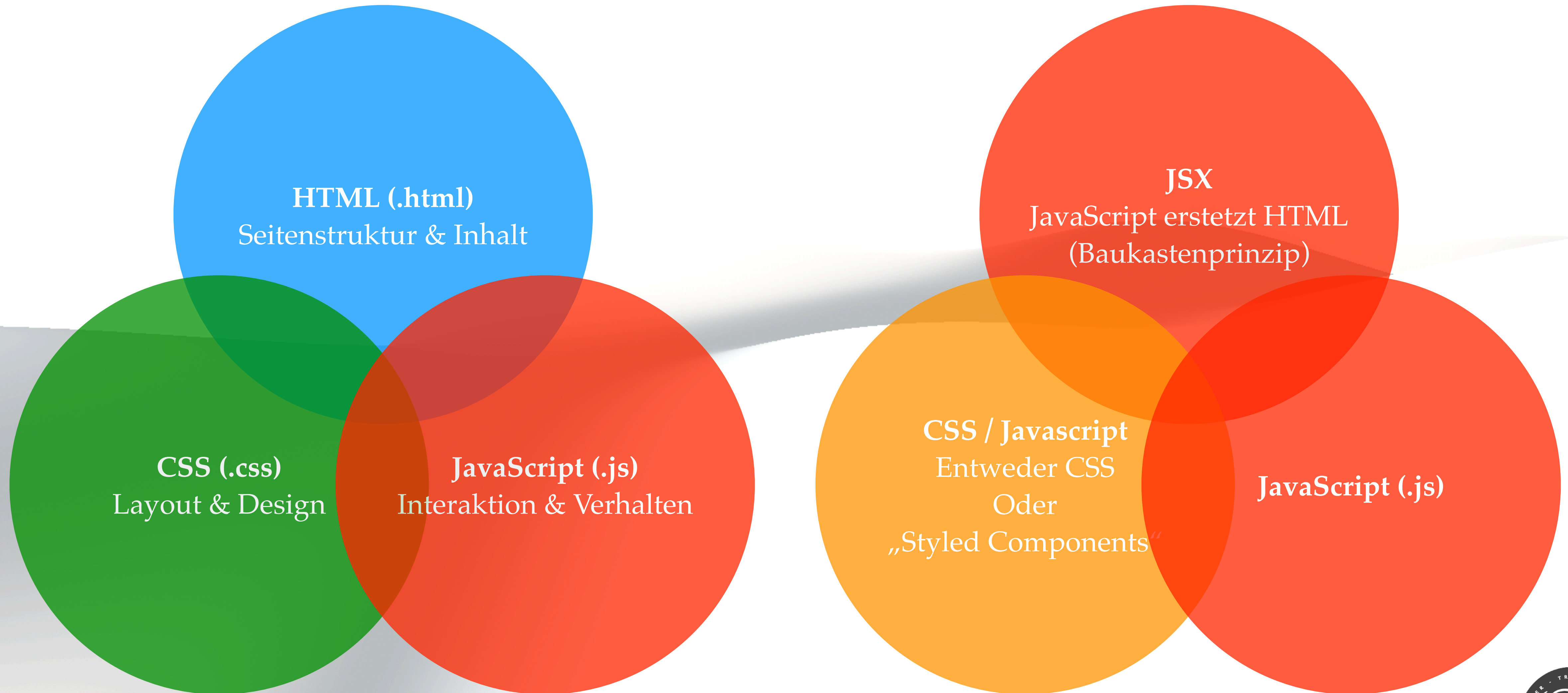


# Analogie ...

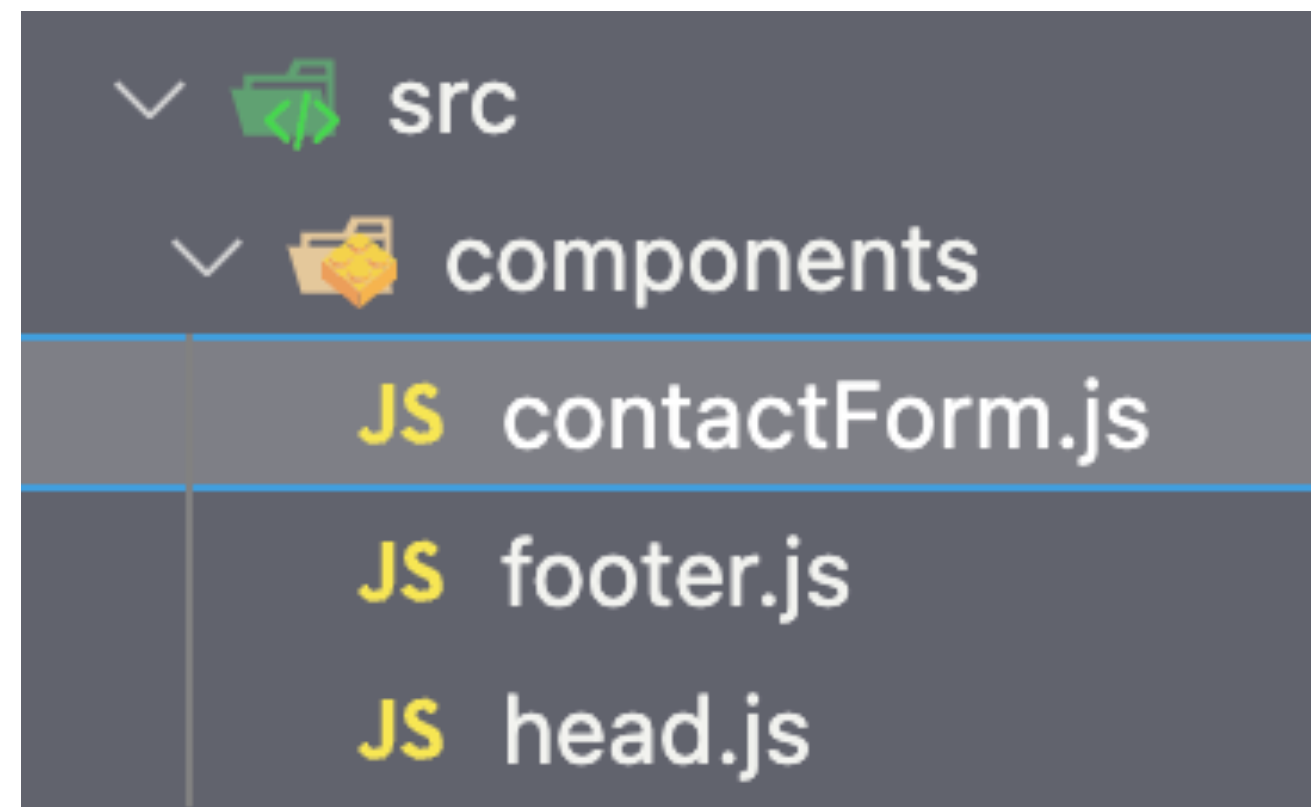




# Klassisch vs. React ...



# JSX-Components ...



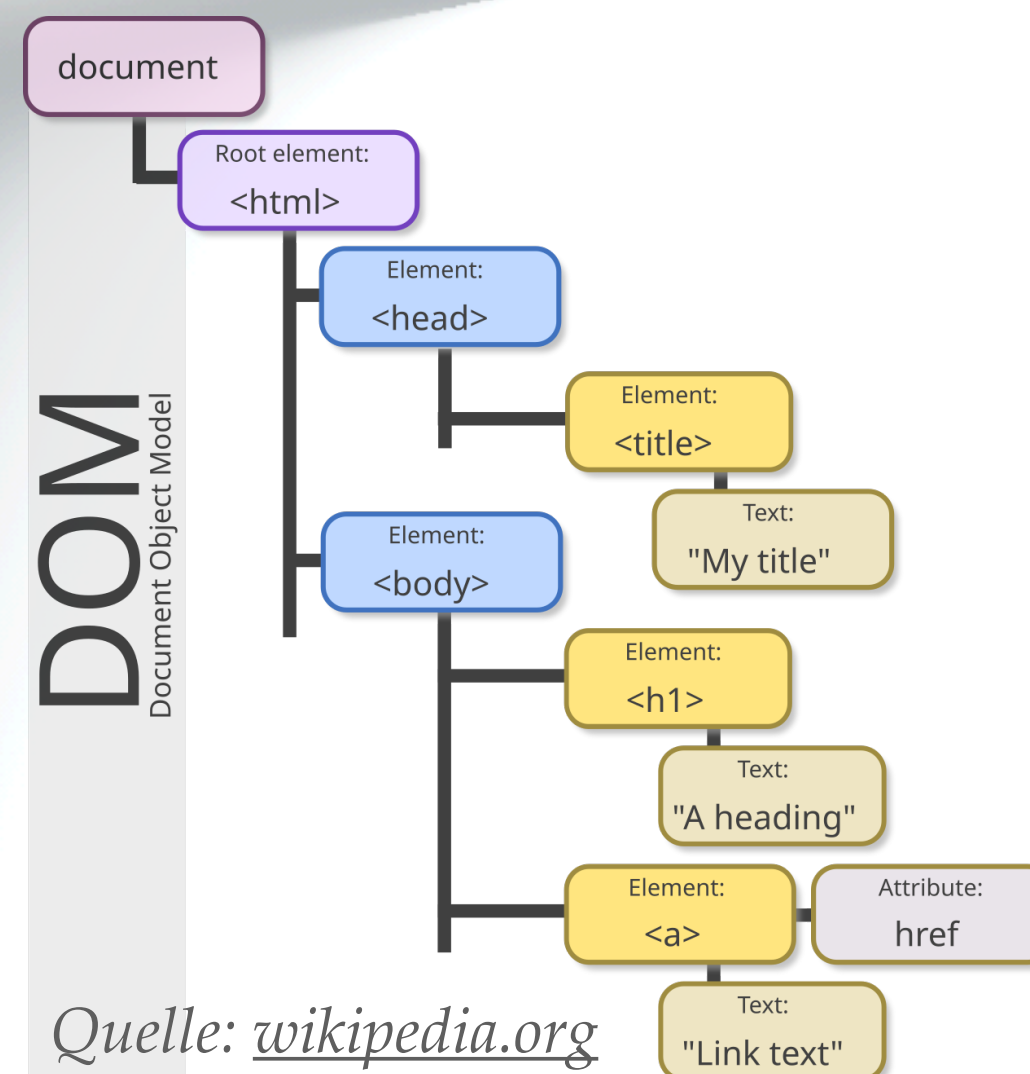
Wiederverwendbare Komponenten ...

```
38 <ContactForm
39   geil=[
40     "Warum der Dozent so schöne Augen hat:",
41     "Was mir an unserem Dozenten / dieser Lichtgestalt so gefällt:",
42     "Alter:",
43     "Warum der Dozent so eine sympathische Stimme hat:",
44   ]
45 >/>
46 <ContactForm
47   geil=[
48     "Was wollen Sie bestellen:",
49     "Wie lautet Ihre Adresse:",
50   ]
51 >/>
```



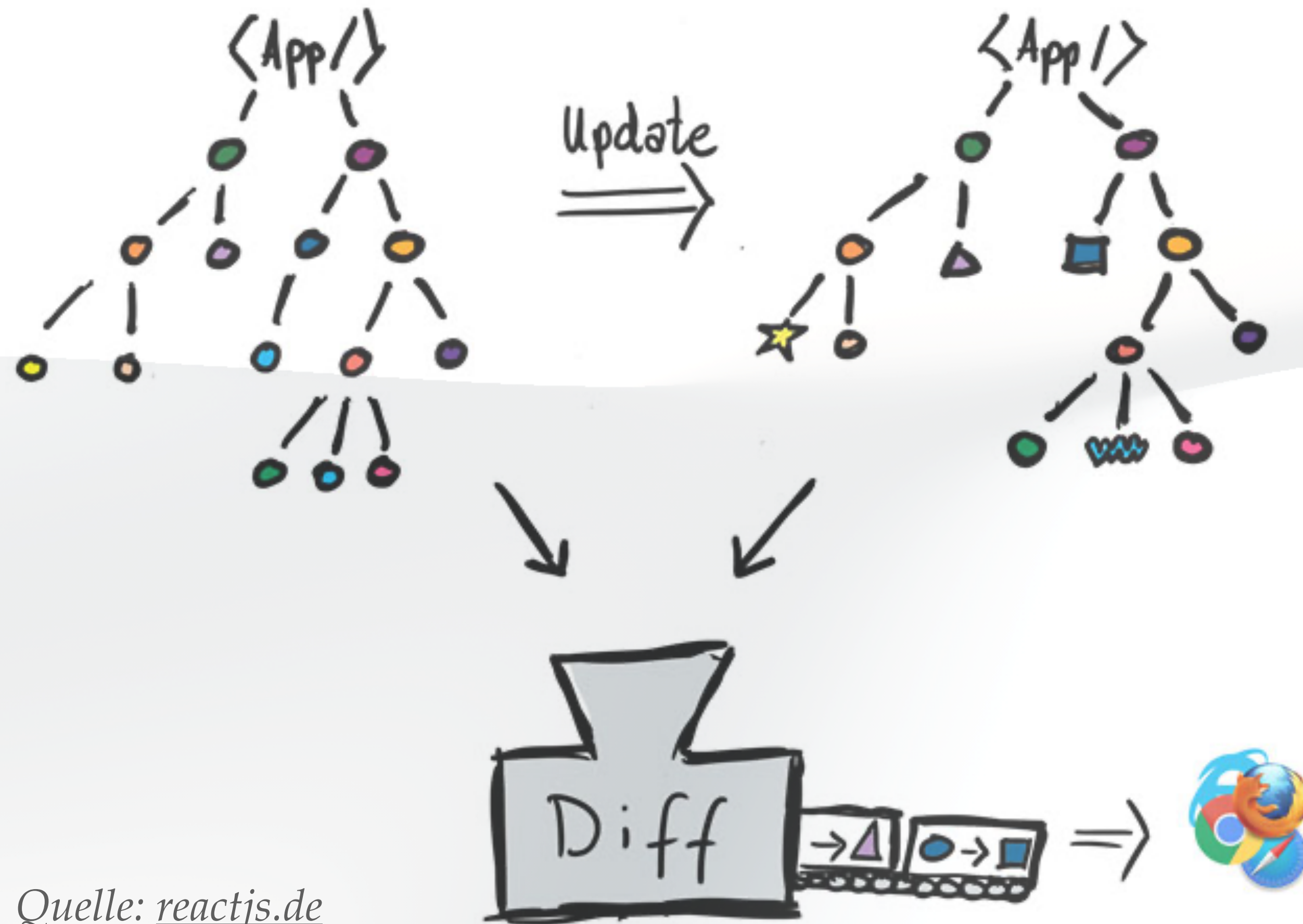
# DOM ...

**DOM** steht für *Document Object Model*. Es handelt sich dabei um eine standardisierte Programmierschnittstelle, die die Struktur eines HTML- oder XML-Dokuments als hierarchischen Baum darstellt. Jeder Teil des Dokuments (Elemente, Attribute, Texte) wird als Objekt im DOM-Baum repräsentiert, und JavaScript kann auf diese Objekte zugreifen und sie verändern.





# DOM vs. Virtual DOM ...

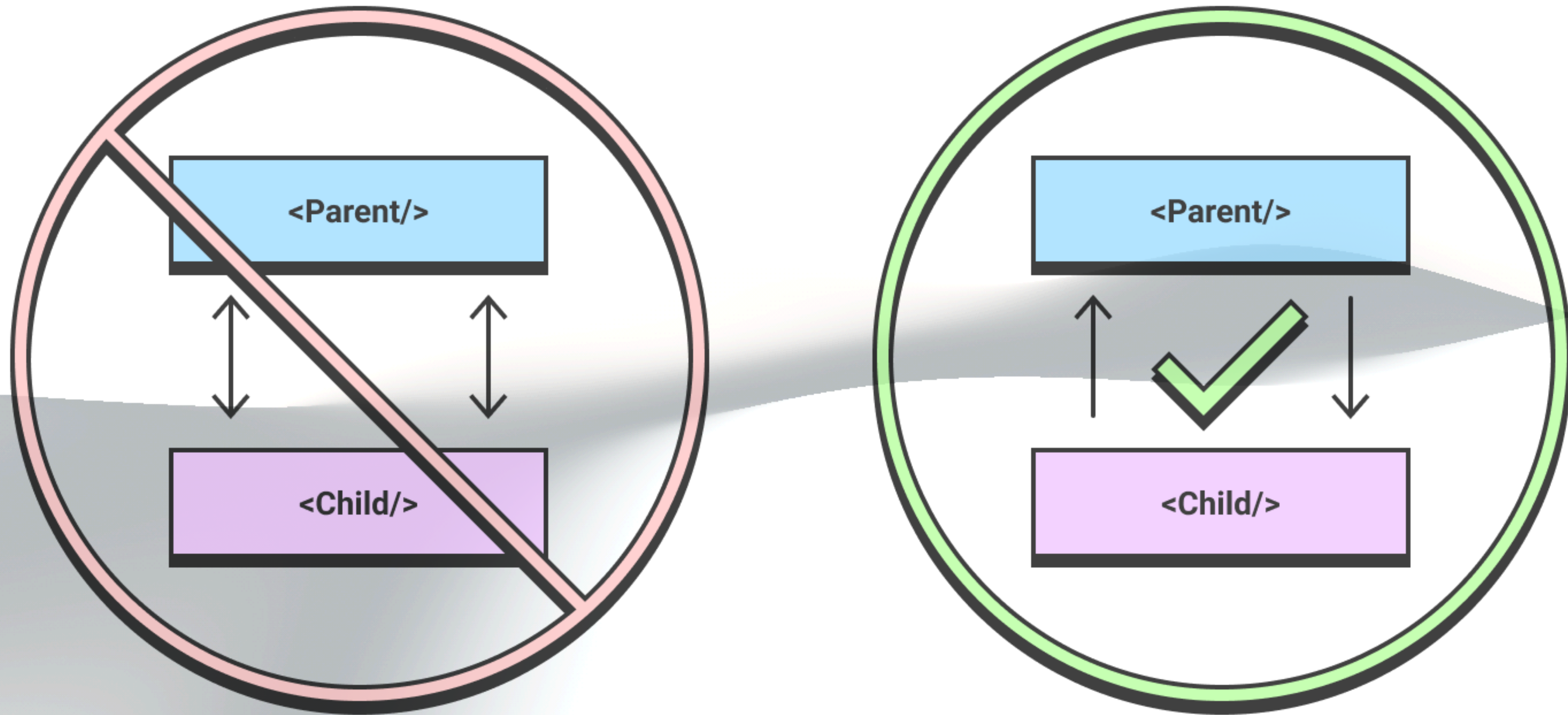


Quelle: [reactjs.de](https://reactjs.de)

React verwendet ein **virtuelles DOM**, um Änderungen effizient zu verfolgen und nur die betroffenen Teile der Seite neu zu rendern, was zu einer besseren Performance führt.



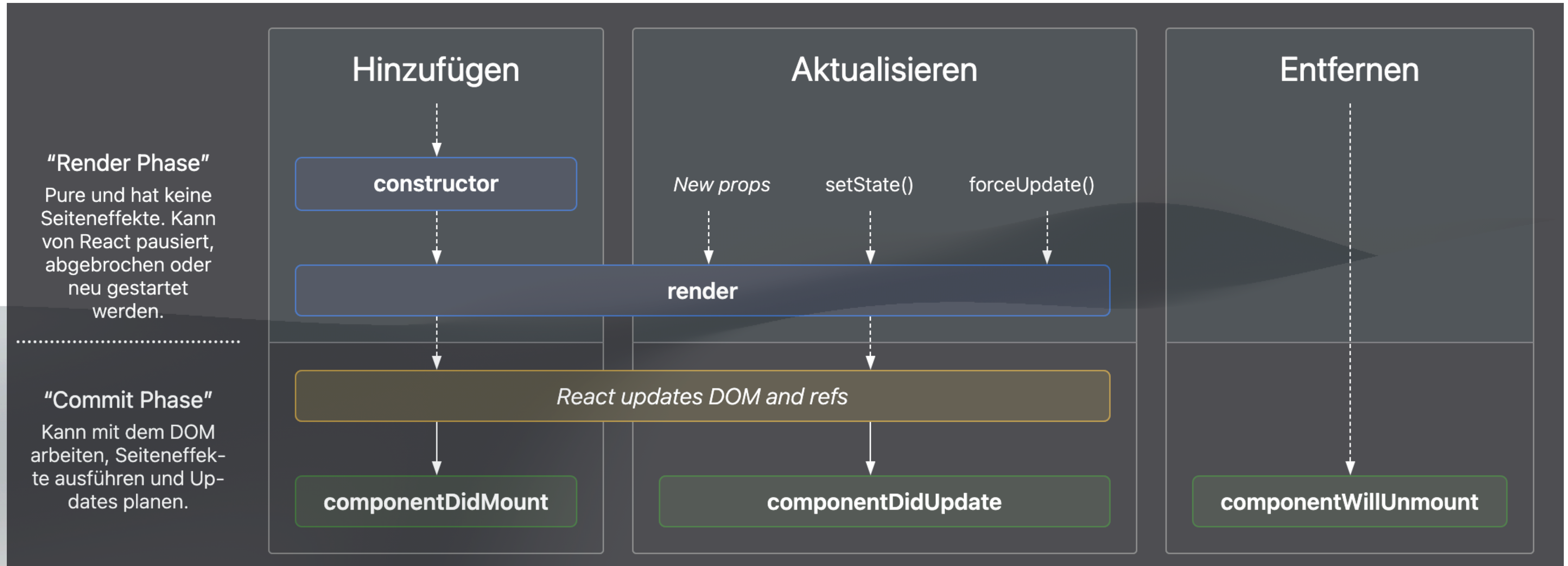
# Unidirektionaler Datenfluss ...



Quelle: [coderpad.io](https://coderpad.io)

# React Lifecycle ...

Quelle: <https://projects.wojtekmaj.pl/>





# Praxisnahes Beispiel ...

```
src > components > JS newsPost.js > ...
1  import React, { Component } from "react"
2
3  export default class NewsPost extends Component {
4    render() {
5      return (
6        <div>
7          <h1>{this.props.text.title}</h1>
8          <p>{this.props.text.message}</p>
9        </div>
10     )
11   }
12 }
13
```

1. constructor() -> Initialisierung des States
2. componentDidMount() -> Aufruf von fetchNews()
3. fetchNews() -> API-Request und Datenverarbeitung
4. this.setState({ news }) -> Update des States mit den abgerufenen Nachrichten
5. render() -> Überprüfung des States und Rendering der Komponenten

Hinweis: Auch andere Lifecycle-Methoden verwendbar (immer überlegen, WAS Sie WANN machen wollen!)

```
src > pages > JS newsArea.js > ...
1  import React, { Component } from "react"
2  import NewsPost from "../components/newsPost" // Selbst erstellte Komponente für einzelne News
3
4  class NewsArea extends Component {
5    constructor(props) {
6      super(props)
7      this.state = { news: [] } // State initialisieren mit leerem News-Array
8    }
9
10   // Nachrichten über eine API laden
11   async fetchNews() {
12     try {
13       const response = await fetch("https://jsonplaceholder.typicode.com/posts") // Beispiel-API für Testdaten
14       const data = await response.json() // Daten als JSON parsen
15
16       // Wir nehmen hier nur die ersten 5 Posts und simulieren die Nachrichten
17       const news = data
18         .slice(0, 5)
19         .map(post => ({ title: post.title, message: post.body }))
20
21       // Setze die Nachrichten in den State
22       this.setState({ news })
23     } catch (error) {
24       console.error("Fehler beim Laden der Nachrichten:", error)
25     }
26   }
27
28   // Daten werden hier nach dem ersten Rendern geladen (empfohlener Weg)
29   async componentDidMount() {
30     await this.fetchNews() // Nachrichten laden
31   }
32
33   // DOM wird aktualisiert und neue NewsPost-Komponenten werden für jedes Element im news-Array erstellt
34   render() {
35     return (
36       <div>
37         {this.state.news.length > 0 ? (
38           this.state.news.map((newsText, index) => (
39             <NewsPost text={newsText} /> // Für jedes News-Element eine NewsPost-Komponente
40           ))
41         ) : (
42           <p>Loading news...</p> // Ladeanzeige, während die Nachrichten geladen werden
43         )}
44       </div>
45     )
46   }
47 }
48
49 export default NewsArea
```














# Vorteile von React ...

- **Komponentenbasiertes Design:** Wiederverwendbare und modulare Komponenten ermöglichen eine effizientere Strukturierung und Wartung des Codes.
- **JSX (JavaScript + HTML):** Erlaubt die Kombination von JavaScript-Logik und HTML-ähnlicher Syntax, was den Code lesbarer und dynamischer macht.
- **Virtuelles DOM:** React verwendet ein virtuelles DOM, um Änderungen effizient zu verfolgen und nur die betroffenen Teile der Seite neu zu rendern, was zu einer besseren Performance führt.
- **Einfaches State-Management:** React bietet ein integriertes State-Management, mit dem man den Zustand der Anwendung einfach steuern und auf Änderungen reagieren kann.
- **Unidirektionaler Datenfluss:** Besser verständlicher Datenfluss (von Eltern- zu Kind-Komponenten), der das Debugging und die Datenkontrolle erleichtert.
- **Erweiterbarkeit:** React lässt sich einfach mit Libraries und Tools erweitern, um komplexere Features zu ermöglichen.
- **Große Community & Ökosystem:** Starke Community und eine große Auswahl an Open-Source-Bibliotheken und Tools.
- **Cross-Platform-Entwicklung:** React kann durch React Native für die Entwicklung von mobilen Apps verwendet werden.

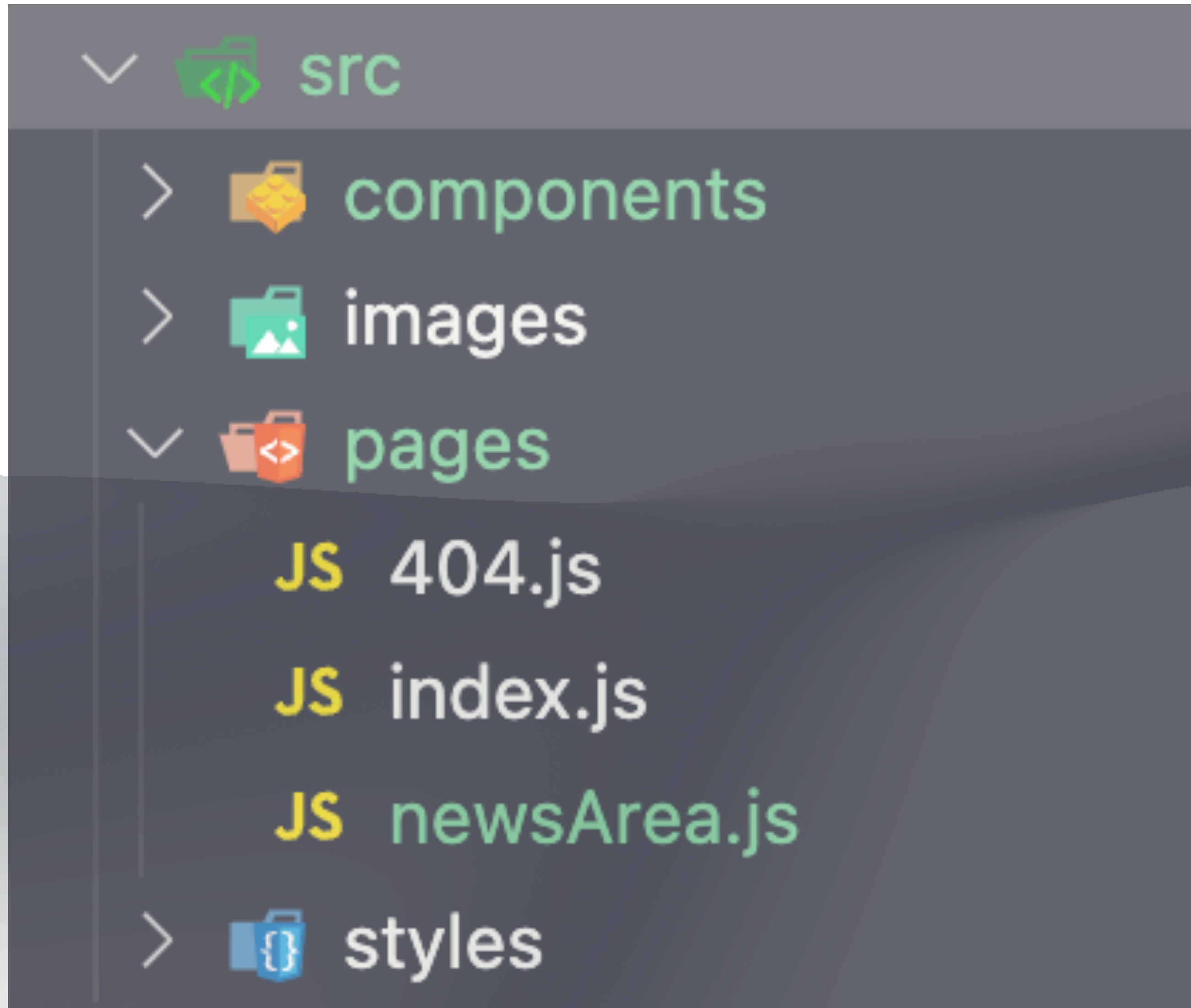


# Ordnerstruktur ...

>  .cache	→	Verbesserung der Build-Zeit durch Speicherung wiederkehrender Daten ...
>  node_modules	→	Speicherung aller notwendigen Packages (Libraries, Extensions, ...) ...
>  public	→	Alle Dateien, die auf dem Server gehostet werden (= Deployment-Ordner) ...
>  src	→	<b>Ihr Quellcode!!!</b>
 .gitignore	→	Verweist auf alle Daten im Ordner, die durch Git ignoriert werden sollen ...
 .prettierignore		
 .prettierrc		
 package-lock.json	→	Auflistung aller Versionen der Abhängigkeiten um sicherzustellen, dass Code auf allen Systemen lauffähig ... mit „npm install“ werden alle install.
 package.json	→	Informationen über verwendete Pakete und Versionen ...
 README.md	→	Projektbeschreibung ...
 yarn.lock	→	Ähnlich wie „package-lock.json“, aber für diejenigen interessant, die „yarn“ Als Package-Manager verwenden (nicht „npm“) ...



# „src“-Ordner ...



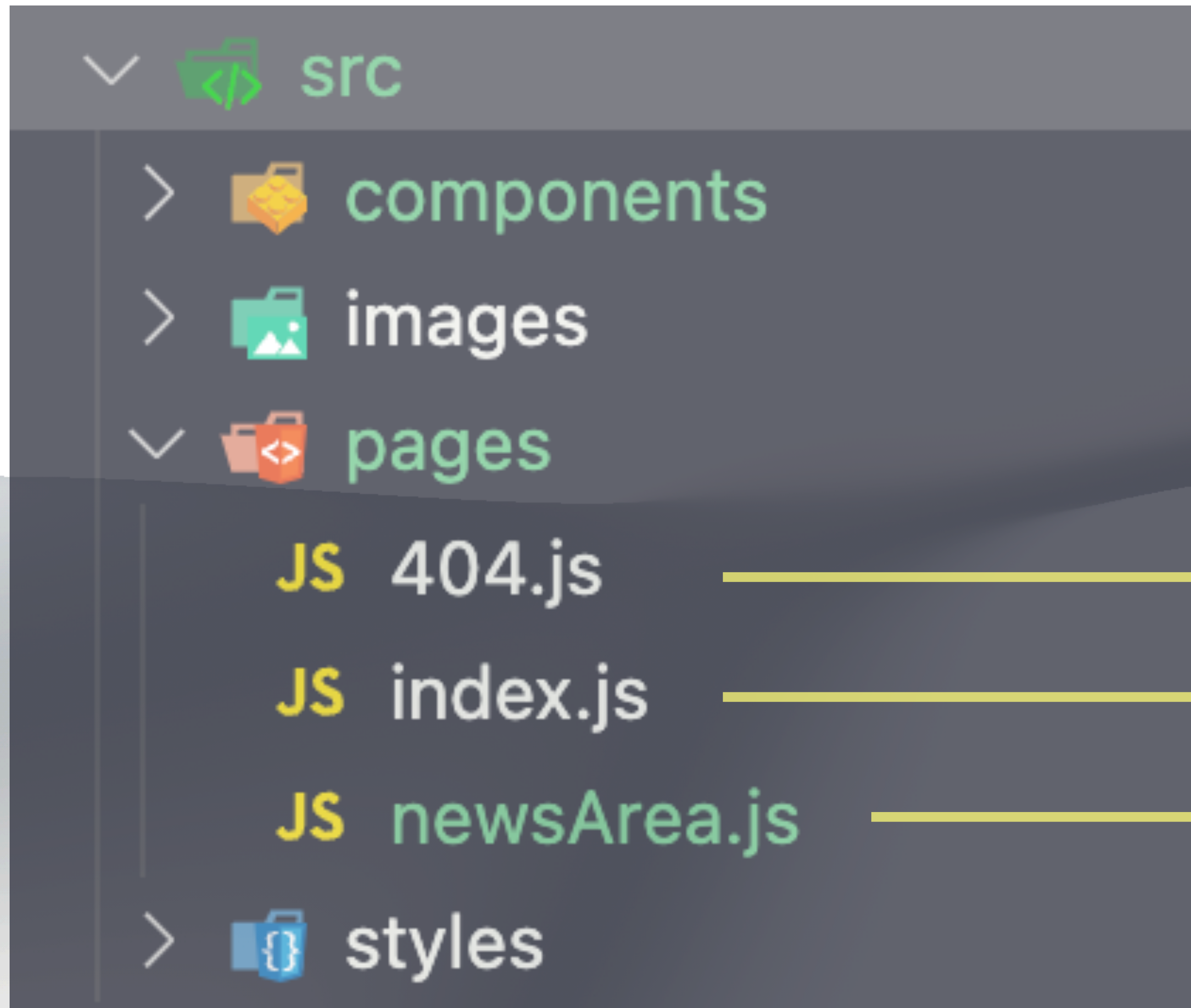
In der Regel trifft man in der Praxis häufig auf eine ähnliche Struktur. Das macht die Arbeit für den Entwickler deutlich leichter, weil man schon ohne viel Hintergrundwissen über den Quellcode in etwa weiß, wonach man zu suchen hat ...

Wichtig: Bei „Gatsby.js“ verwendet man den „pages“-Ordner ... 2 Pages sind dabei immer vonnöten: „index.js“ und „404.js“ ...





# „pages“-Ordner ...



**404: Not Found**

Lachs bereits leblos 🐟

meine-website.de

meine-website.de/newsArea

# Tipp für später: IHK-Projektarbeit ...

*Hier finden Sie Beispiele für sehr gut benotete Abschlussarbeiten:*

<https://it-berufe-podcast.de/vorbereitung-auf-die-ihk-abschlusspruefung-der-it-berufe/beispiele-fuer-ihk-abschlussprojekte-in-den-it-berufen/>