

Was macht moderne (JavaScript) Frontend- Architektur so schwierig?

OOP 2018

Oliver Zeigermann / @DJCordhose

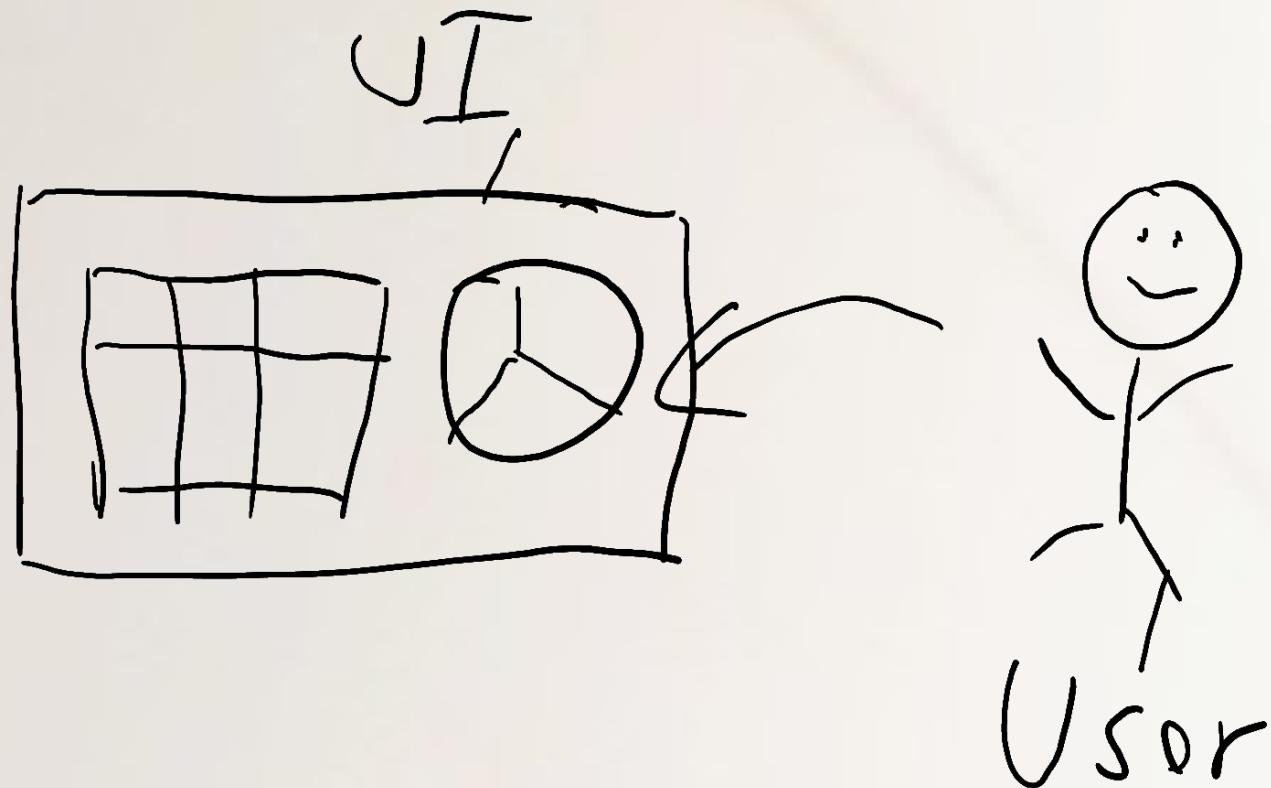
Folien: <http://bit.ly/oop-frontend>

embarc GmbH

Teil 1

Im Spannungsfeld der
Anforderungen

Nutzer möchten eine konsistente, gut bedienbare UI erleben



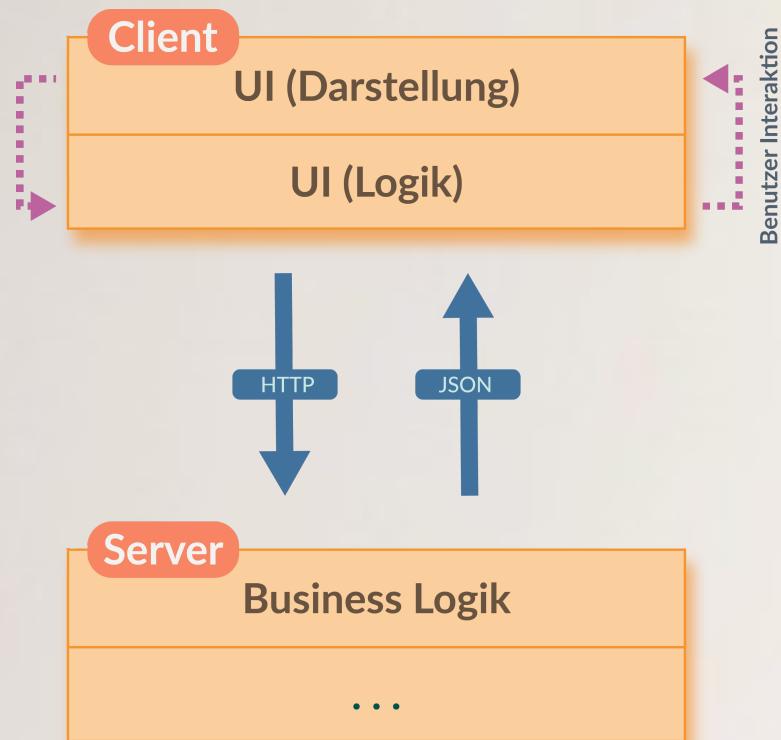
Axiom I

*Beste UI / UX kann nur durch
Single Page Applications (SPA),
Mobile oder Desktop App
erreicht werden*

Logik muss nahe an Interaktion und Daten müssen nahe an
Logik liegen

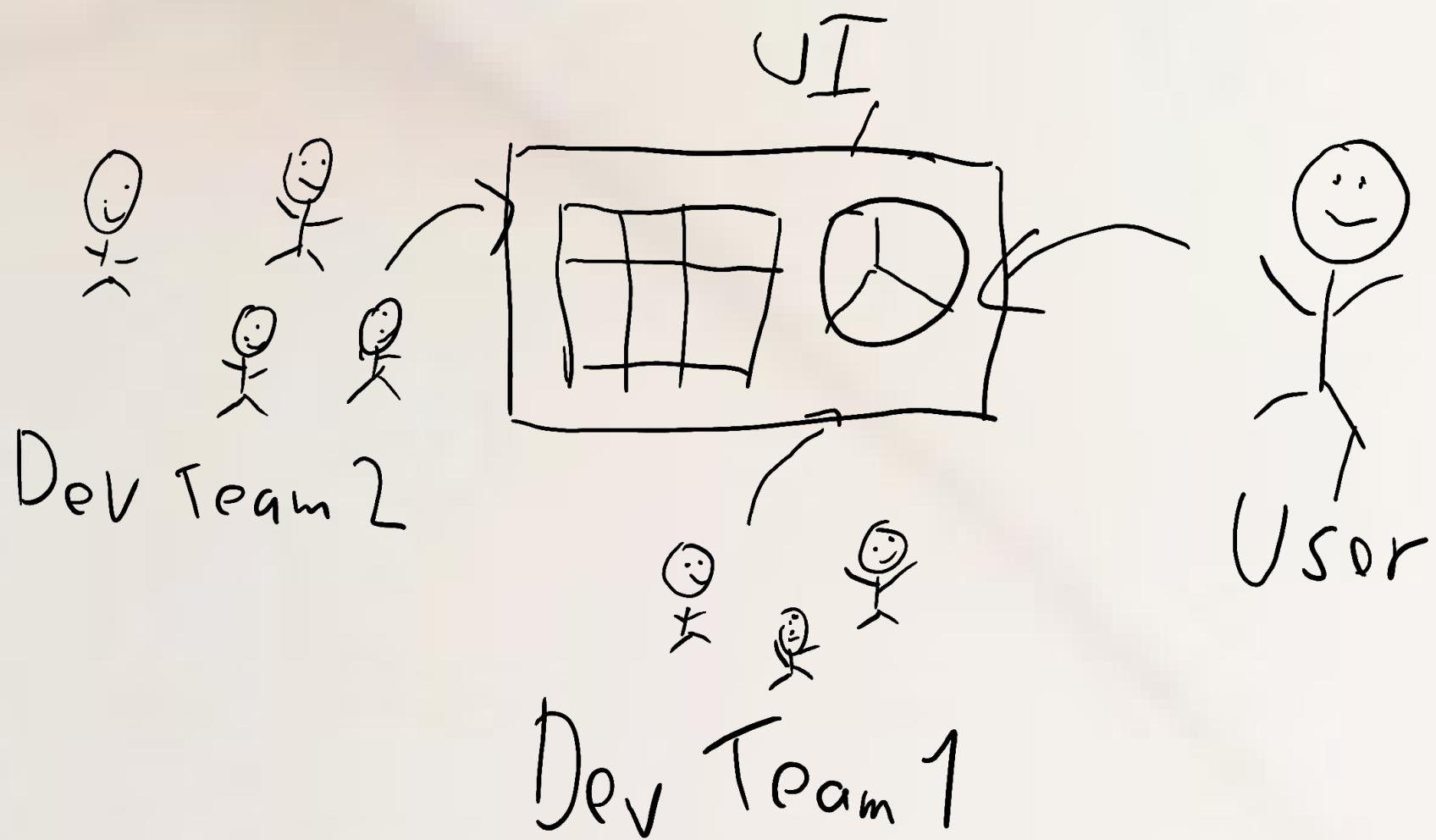
SPA

verschieben daher Hauptteile eurer Anwendung in den Client



- Server ist hauptsächlich Daten-Schnittstelle (JSON/REST API)
- Server kann Business Logic enthalten, diese kann aber komplett im Client liegen
- erlaubt Offline-Betrieb für SPAs (u.a. über Service Workers)

Größere Anwendungen können nicht von einem einzigen Team entwickelt werden



Axiom II

*Wartbarkeit und Skalierung im
größeren Kontext ist nur durch
Modularisierung erreichbar*

Moderne Frontends...

- sollen aus einzelnen **Modulen** zusammensetzbare sein
- müssen zu modernen **Backend-Architekturen** passen
- sollen einzeln zu **deployen** sein
- sollen pro Modul am besten die **freie Wahl von Technologie** erlauben

Das passende Buzzword dazu ist **Micro Frontends** (
<https://www.thoughtworks.com/radar/techniques/micro-frontends>)

Spannung zwischen Axiom I und II

Wenn man die Module für Wartbarkeit entkoppelt (Micro Frontends) muss man sie für ein UI aus einem Guss wieder koppeln

Oder: man muss sich entscheiden, will man:

- die beste Erfahrung für den Nutzer oder
- den besten Entwicklungsprozess



GOOD



CHEAP



FAST

ANIMATION BY

▶ 0:00 / 0:16



COM - © 2017

<https://twitter.com/missingcloudltd/status/826203153934729218>

Es gibt keinen besten Architektur-Ansatz

Daher...

*Man muss die Anforderungen
kennen und die müssen passen.*

- Oliver Zeigermann, embarc

Typische Ziele/Einflüsse

UX, Konsistenz

- Gleichzeitige, synchronisierte Darstellung mehrerer Module
- Schnelle erste Darstellung
- Offline-Fähigkeit
- Unverzögerte Reaktion

Risiko-Minimierung

- schrittweises Ausrollen von technischen Innovationen
- schrittweise Migration, kein Big Bang
- Feature-Stau

Skalierbarkeit der Entwicklung / Entwicklungsgeschwindigkeit

- Anzahl der Teams
- Freie Wahl des Frameworks
- Update der Bibliotheken
- Eignung für vertikale Architektur
- Gemeinsame Bibliothek,
Wiederverwendung
- Freiheitsgrad der Modulgröße
- Migrationspfad von klassischer Web-App

Übung

Unterhalte dich mit deinen Sitznachbarn über dein eigenes (oder ein dir bekanntes) Frontend

Welche Anforderungen gibt es in dem Projekt?

Wie begegnet ihr diesen?

Teil 2

Lösungsansätze

Wir nehmen Frontend-Architektur nicht für voll

Zitate

- *Frontend-Architektur: Ich dachte sowas gibt es gar nicht*
- *Das Frontend kloppen wir am Ende einfach irgendwie drauf*
- *Architektur für ein bisschen CSS-Pixel-Geschubse?*



Yehuda Katz 

@wycats

Following



Front end software development is:

- real-time (instant load, 60fps)
- distributed, incremental (synchronize remote data as needed)
- asynchronous
- reactive (react to user actions in realtime)

Front end is the hardest kind of dev I do. The folks who do it every day are heroes.

4:53 PM - 14 Nov 2017

@wycats : <https://twitter.com/wycats/status/930463710941872128>

Micro Frontend Architektur- Ansätze

Integration...

1. über Links - Codename **Links**
2. zur Build-Zeit - Codename **Majestic Modular Monoliths**
(Kudos an [@axelfontaine](#) für den Namen)
3. im Browser - Codename **Micro Components**

Ansatz 1

Eigenständige Anwendungen über Links verbunden

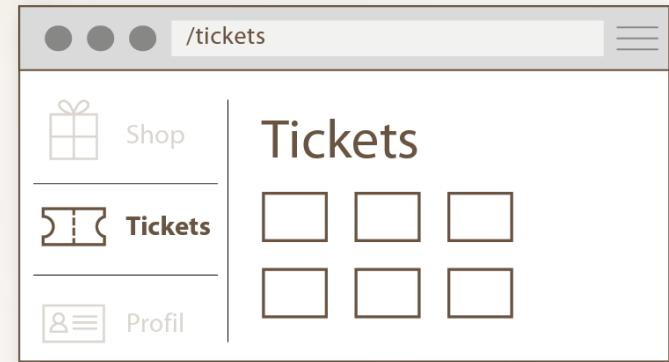
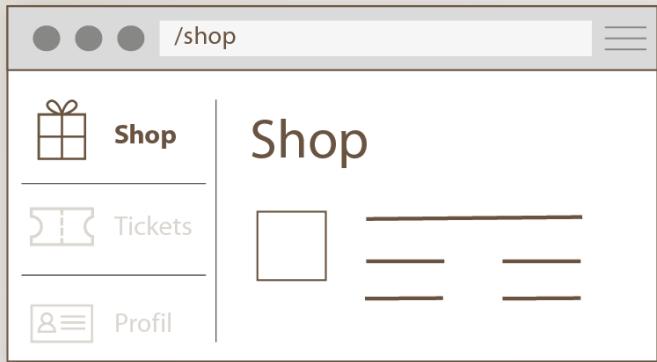
Integration mehrerer Anwendungen über Links, die eine neue Anwendung öffnen und/oder die alte ersetzen

- jedes Modul ist eine eigene (SPA) Anwendung
- erlaubt Vertikalen wie eine klassische Web-App
- Module teilen keinen Zustand, aktualisieren einander nicht

Links

Variante: App Shell hat JavaScript Rahmen, weniger flexibel, kann aber u.a. Zusand halten

Eigenständige Anwendungen in Eigenständiger Darstellung



Jede Anwendung wird komplett neu aufgebaut

Prominentes Beispiel: Outlook Online

The screenshot shows the Microsoft Outlook Online interface. At the top, there are three tabs: "E-Mail – Oliver Zeigermann", "Fotos – Oliver Zeigermann", and "Kalender – oliver.zeigermann". The main title bar says "Outlook" and has a search bar with "Suchen". Below the title bar, there's a button for "Neue Nachricht" and a toggle switch for "Outlook (Beta)". On the left, a sidebar menu is open, showing "Favoriten" and "Posteingang" selected. Under "Posteingang", there are links for "Entwürfe", "Archiv", "Ordner", "Posteingang", "Junk-E-Mail", "Entwürfe", "Gesendete Elemente", "Gelöschte Elemente", "Archiv", "Conversation History", and "Neuer Ordner". The main content area is mostly empty, featuring a large blue trophy icon and the text "Ihr Posteingang mit Relevanz wurde gelöscht." (Your inbox with relevance was deleted.)

E-Mail – Oliver Zeigermann

Secure | https://outlook.live.com/mail/#/inbox

Oliver

Outlook Suchen

+ Neue Nachricht Outlook (Beta)

Favoriten

Posteingang

Entwürfe

Archiv

Ordner

Posteingang

Junk-E-Mail

Entwürfe

Gesendete Elemente

Gelöschte Elemente

Archiv

Conversation History

Neuer Ordner

Ihr Posteingang mit Relevanz wurde gelöscht.

0:00 / 0:55



A screenshot of the Microsoft Outlook web interface showing the inbox. The sidebar on the left lists various folders like 'Posteingang', 'Entwürfe', and 'Gesendete Elemente'. The main area displays a message stating 'Ihr Posteingang mit Relevanz wurde gelöscht.' (Your inbox with relevance was deleted.) accompanied by a trophy icon. At the bottom, there's a media control bar with a play button, a progress bar at 0:00 / 0:55, volume controls, and download icons.

UX-Schwächen, unterschiedliche Technologien, jede App wird anders dargestellt
Wechsel der App dauert (Service Workers können die Zeit ab 2. Aufruf vermindern)

Technik

Kommunikation zwischen getrennten Anwendungen

Ansätze: Links

Bedient Anforderungen: **nicht gleichzeitige, aber**
synchronisierte Darstellung mehrerer Module

Real-Time

- **Long Polling:** Erfordert Server Verbindung, kann viel Server Ressourcen fressen
- **Web Sockets:** Erfordert Server Verbindung, geht immer noch nicht überall
- **PWA Push:** Erfordert Service Worker und Server

Parameter-Übergabe

- **Cookies:** gut für Session Id, Tokens
- **URL Parameter:** begrenzte Länge, nicht alles möchte man in der URL haben, Bookmark und verschickbarer Link möglich
- **Session Storage:** Key/Value, wird gelöscht wenn Tab geschlossen wird
- **Local Storage:** Key/Value, bleibt bestehen bis zur expliziten Löschung

Alle Ansätze halten auch bei Reload den Zustand

Ansatz 2

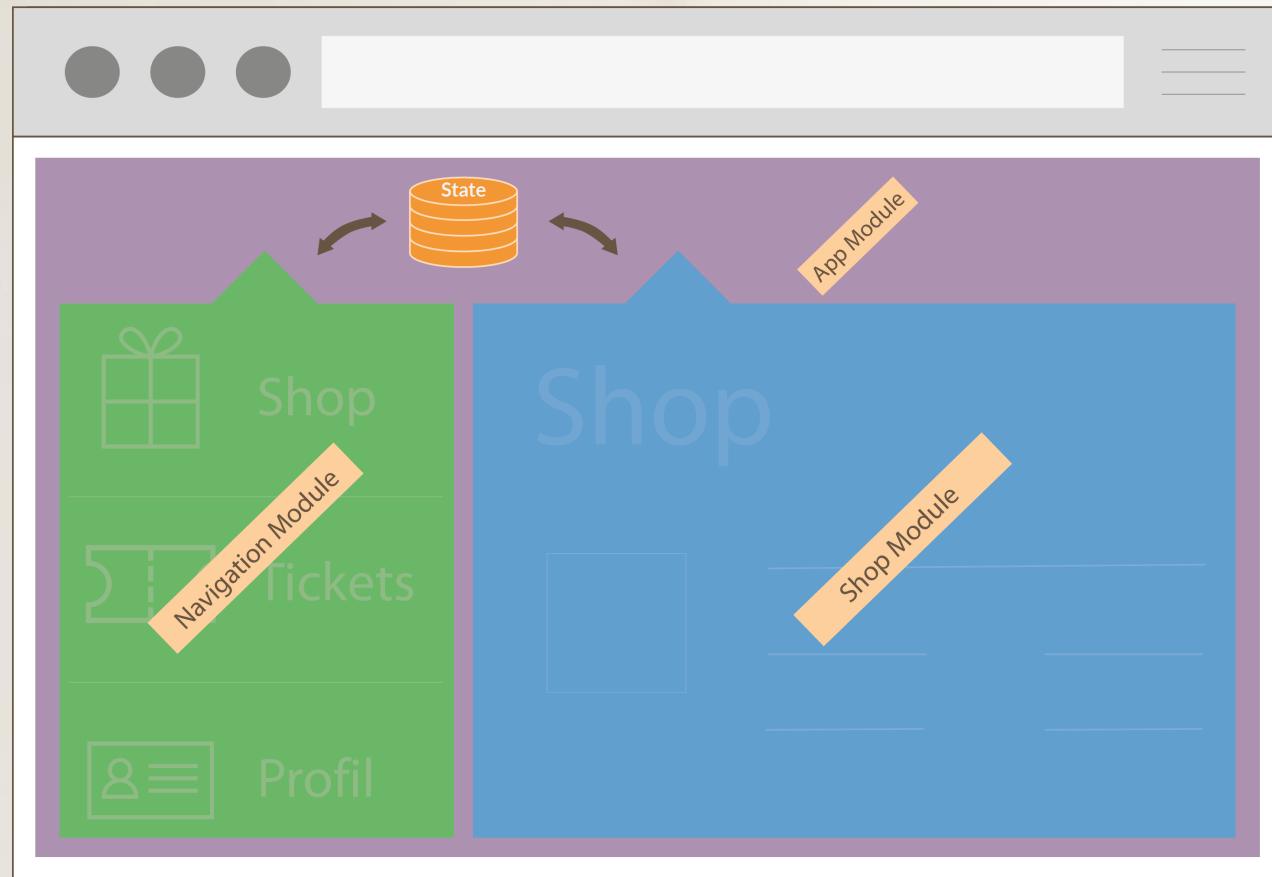
Single Page Application

Eine einzelne Single Page Application mit statischem Build

- Jedes Team entwickelt eigenes Modul
- Statischer Build integriert komplette Anwendung
- Zur Laufzeit Lazy-Loading möglich
- Anwendung nutzt ein Framework mit einer Version
- Gemeinsame Bibliothek sinnvoll
- Export einzelner Komponenten möglich

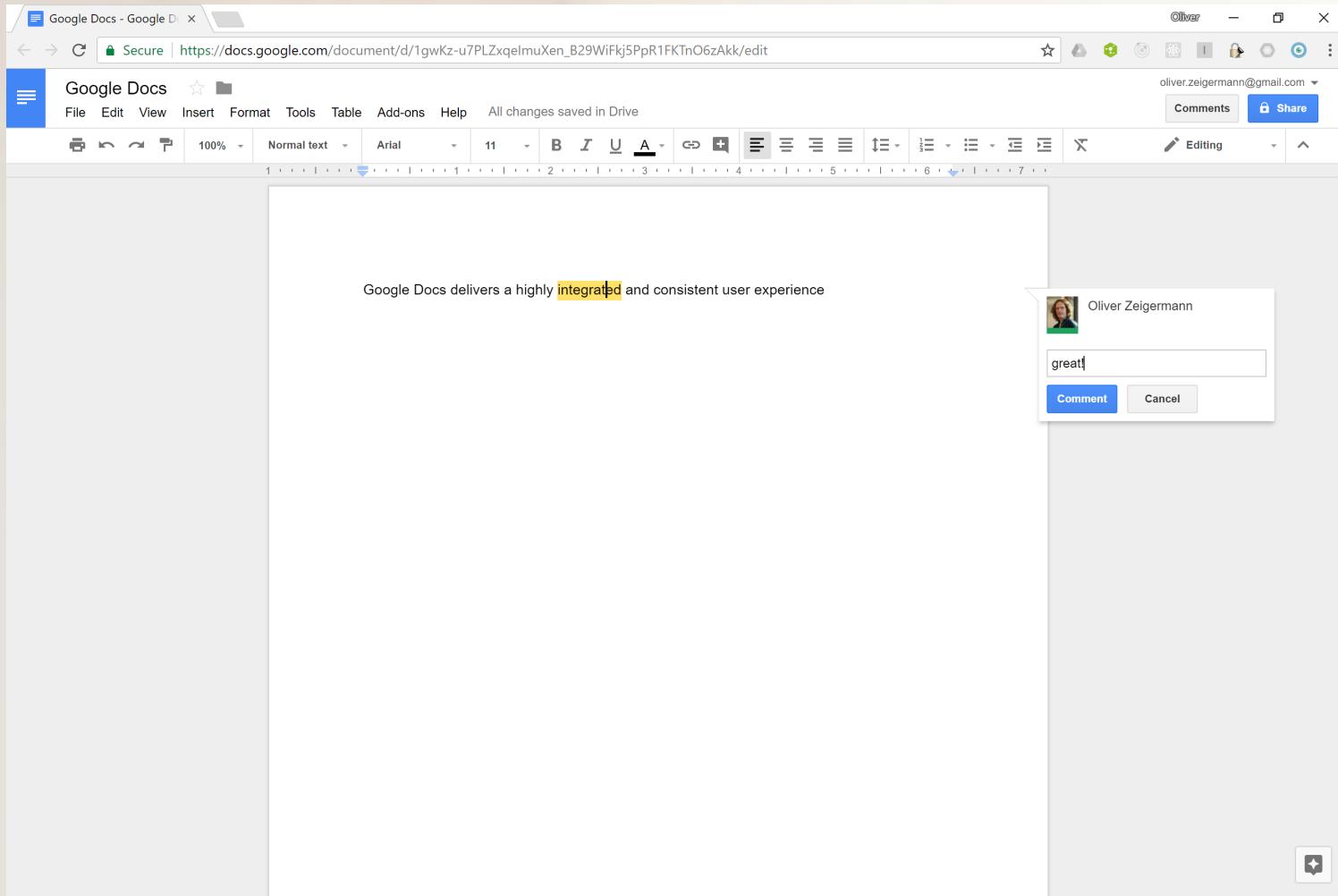
Majestic Modular Monoliths

Gemeinsame Anwendungen in gemeinsamer Darstellung



Server-Zugriff nur für Daten oder einmalig für Code

Prominentes Beispiel: Google Docs



Keine systematischen UX Probleme

Universal Rendering

Mit demselben Code sowohl auf dem Server vorrendern als
auch im Client interaktiv sein

Ansätze: Majestic Modular Monoliths

Bedient Anforderungen: SEO, schnelle erste Page-Impression

Ansatz 3

Zusammensetzen der Anwendung im Browser

- Anwendung ist in logische Module aufgeteilt, die als ganze Anwendung zusammen laufen
- Frameworks frei wählbar pro Modul
- Gesamtanwendung wahrscheinlich groß

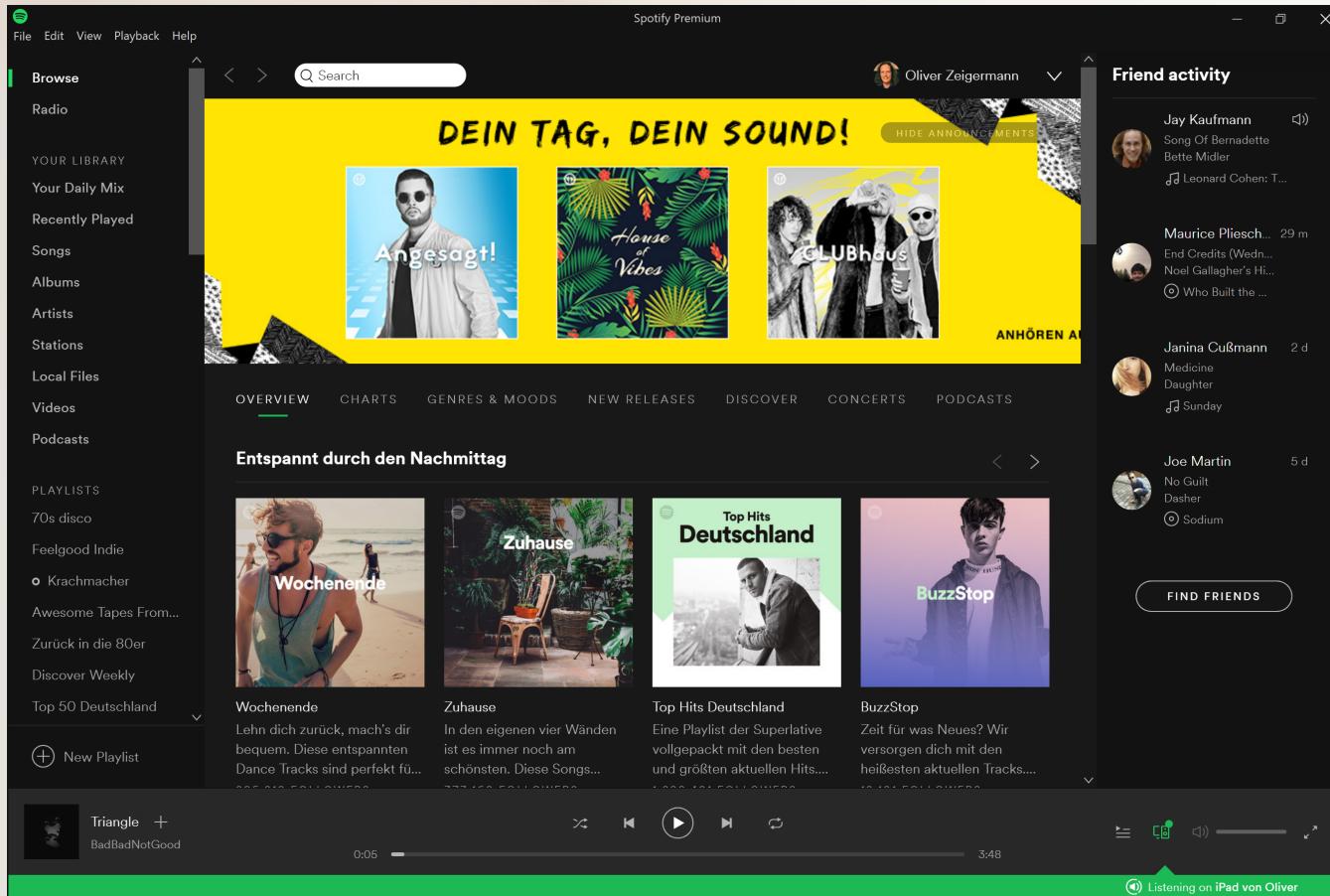
Micro Components

Eigenständige Anwendungen in gemeinsamer Darstellung



Jede Anwendung kann komplett eigenen Stack haben
Kommunikation über EventBus, direkte Listener oder
gemeinsamer Zustand

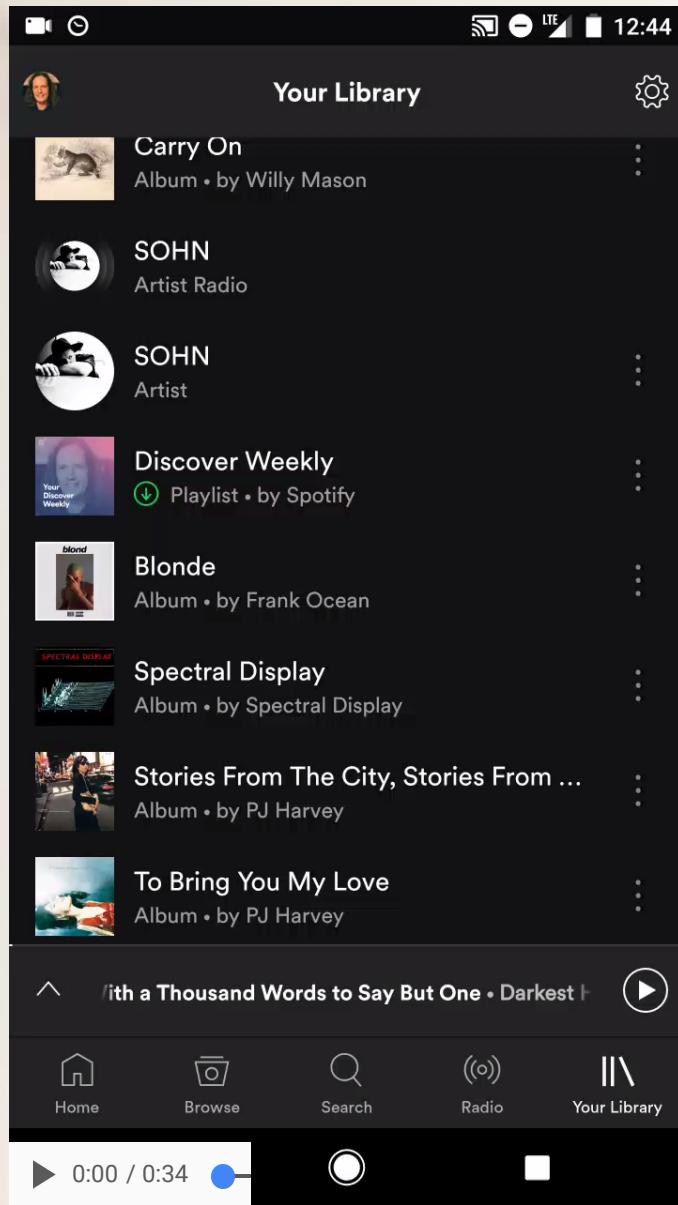
Prominentes Beispiel: Spotify



<https://www.quora.com/What-is-the-technology-stack-behind-the-Spotify-web-client/answer/Andreas-Blixt>

<https://www.quora.com/How-are-Spotify-Web-Player-components-developed>

https://www.reddit.com/r/reactjs/comments/5sgkro/til_spotify_is_using_reactredux_in_their_web_app/



überraschende UX Schwächen, Apps nur gleichzeitig dargestellt, aber nicht komplett integriert

Migrationsszario

Angular (Übersicht) nach React (Spielfeld und Ticker),
Kommunikation über Long Polling

The screenshot shows a live football match interface. At the top, a header indicates the match is between 1. FC Köln and Bor. Mönchengladbach. Below the header is a navigation bar with links for the home page, search, and user profile.

The main content area features a green football pitch diagram with a ball in the center. A callout points to the ball with the text "M'GLADBACH Angriff". The "sport1" logo is visible at the bottom left of the pitch area.

Below the pitch, a black banner displays the text "69. ☺ TOOR FÜR BOR. MÖNCHENGLADBACH, 1:1 DURCH RAFFAEL" in white. To the right of this banner is a yellow box containing the text "data by sport1.de".

Further down, another black banner shows "68. █ GELBE KARTE FÜR SIMON TERODDE (1. FC KÖLN)" in white. Below these banners, a third black banner displays "68. ← AUSWECHSLUNG BEI BOR. MÖNCHENGLADBACH" in white, followed by the note "Mickael Cuisance für Christoph Kramer".

To the right of the pitch diagram, there is a yellow box with the text "Alle Begegnungen im KONFERENZ TICKER". Below this is a section titled "KONFERENZTICKER" which lists several recent matches:

Team 1	Score	Team 2
FC Bayern München	1:3	Borussia Dortmund
RB Leipzig	1:1	FC Schalke 04
FC Augsburg	1:0	FC Bayern München
FC Ingolstadt	1:1	FC Augsburg
FC Ingolstadt	1:0	FC Schalke 04
FC Augsburg	3:2	FC Bayern München
FC Bayern München	3:1	FC Schalke 04
FC Bayern München	1:0	FC Schalke 04

The last row of the table is highlighted with a red box and includes the text "Heute 18:00".

At the bottom right, there is a section titled "AUFPSTELLUNG" with a green box containing the text "1. FC Köln".

<http://www.sport1.de/fussball/bundesliga/live-ticker>

Technik

Client-Seitige Integration

Ansätze: Micro Components

Bedient Anforderungen: Gleichzeitige, synchronisierte
Darstellung mehrerer Module

- einfache *script* Tags und unterschiedliche Mount Points (wie bei Sport1)
- Web Components
- iFrames

Web Components

Web Component ummantelt Modul

Beispiel:

```
<script type="module" src=".my-element.js"></script>
<my-element greeting="Hello"></my-element>
```

Keine Komplette Isolation, Funktioniert in der Praxis nicht gut mit beliebigem Framework

iFrame

Beispiel:

<http://djcordable.github.io/architecture/code/integration/>

Lädt komplett isolierte Anwendung

```
<iframe height="300" width="500"  
src="http://djcordable.github.io/architecture/code/integration/iframe"  
sandbox="allow-scripts allow-top-navigation allow-same-origin"></iframe>
```

- Kein geteiltes Styling
- Einbetten mit richtiger Größe ein Albtraum
- Kommunikation mit Außenwelt lästig

Problem: Ladezeiten, wenn jedes Modul sein eigenes Framework mitbringt

- Frameworks wie React oder Angular bringen von sich aus immer mehr als 100KB mit
- (UI-)Bibliotheken haben oft dieselbe Größe oder mehr
- Bei mehreren Modulen ist man schnell im MB-Bereich:
Langsamer Aufbau der Seite

Lösung: k(l)eine Frameworks und Bibliotheken nutzen

- Preact kann Anwendungen unter 10KB bringen
- Web-Components sind Teil der Browser API (noch nicht überall)

Ansätze sind nicht wechselseitig exklusiv

- Seite einer Link-App kann eine Micro Components App sein
- Teil einer Micro Components kann ein Majestic Modular Monolith sein
- Seite einer Link-App kann ein Majestic Modular Monolith sein (Google Drive vs Google Docs)

Ein Quiz

The screenshot shows a web browser window with the XING homepage loaded. The URL in the address bar is <https://www.xing.com/communities>. The page features a teal header with the XING logo and a search bar. Below the header, a large teal section contains the text: "Information, discussions and networking – find your favourite groups on XING!". A search input field says "Enter topic, interest or group". To the left of the main content is a sidebar with icons for basic navigation: Home, People, Groups, Events, Posts, and Groups (highlighted). The main content area shows a summary of group activity: "88,299 specialist groups on XING" and "2,188 upcoming group events". It also displays a list of groups the user is a member of: "React Native" (20+ new posts), "React JS" (20+ new posts, 4 new classifieds), and "Vue.js Experten". Each group entry includes a thumbnail, the group name, the number of new posts, and a recent post snippet. On the right side, there is a vertical advertisement for a laptop. A green speech bubble icon is visible in the bottom right corner of the main content area.

Fragen

Welcher Ansatz / Technik kommt hier **nicht** vor?

1. React
2. Links
3. Majestic Modular
Monolith
4. Cookies
5. Micro Components
6. Web Sockets

Urteil

- **Links: bester Entwicklungsprozess**
 - beste Story für Modularisierung, Migration, Deployment
 - schwächste Integration von UI/UX
- **Majestic Modular Monoliths: beste Erfahrung für den Nutzer**
 - schwächste Story für unabhängige Teams
 - bestes und konsistentestes UX/UI, starke Stories für State Management und Universal Web Apps
- **Micro Components: Kompromiss**
 - migration von Klassischer Wep-App schwer denkbar
 - UX/UI gut für Intranet oder Desktop

Empfehlung

Anwendung komplett trennen und mit Links verbinden wo möglich

Wahrscheinlich sind mehr Trennungen möglich als ihr denkt

Getrennte Anwendungen als Majestic Modular Monoliths
außer wenn starke Anforderung an Unabhängigkeit

Vielen Dank!

Es gibt keine perfekte Architektur für jeden Satz von Anforderungen

Du musst deine Anforderungen kennen und dabei ehrlich zu
dir sein

Oliver Zeigermann / @DJCordhose / embarc GmbH



<http://bit.ly/oop-frontend>