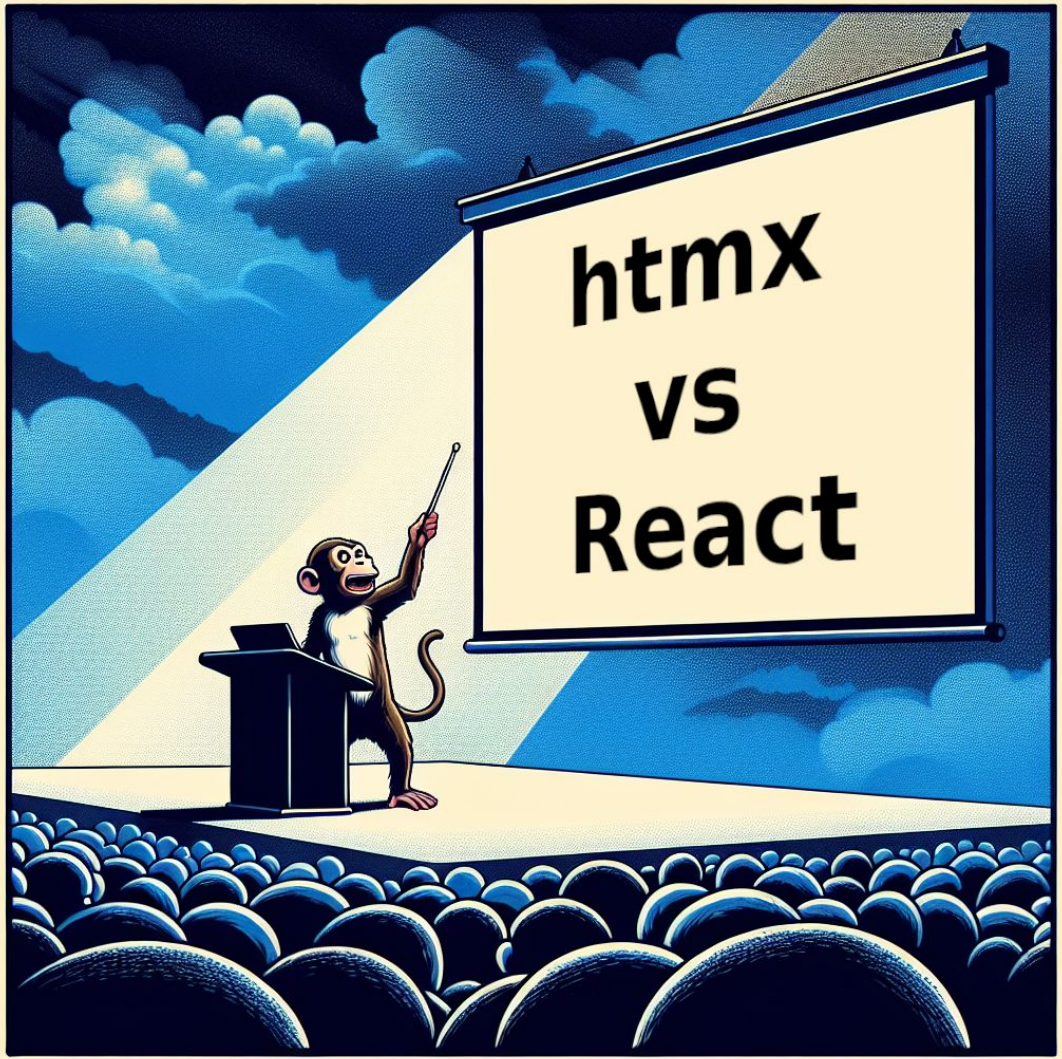
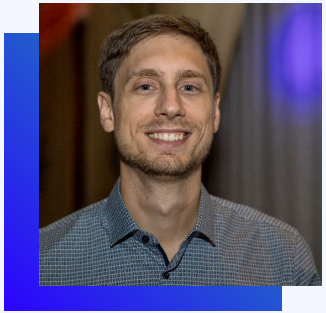


Der ultimative Web App Showdown

inovex

Jonas Kaltenbach



Jonas Kaltenbach

@kajo404

@kajo404

Software Developer @inovex GmbH

- Web development, mostly SPAs
  - Angular, Vue.js
- Accessibility
- Web application performance

inovex

Flight Plan

- Status Quo SPAs
- API / Application Design
- Skaliert das?
- Security
- Accessibility

inovex

Disclaimer

- React steht nur als Platzhalter
- “pro-htmx” Bias
- Inhalte basieren auf essays von htmx.org

inovex

Dieser Talk soll zur Diskussion anregen

inovex

SPA vs MPA

SPA	MPA
Dynamisches Nachladen	Jede Seite wird neu geladen
Nahtlos & Interaktiv	Langsame Seitenübergänge
Komplex	Weniger Komplex

inovex

Hypermedia Driven Applications (HDA)

Kombiniert die Einfachheit & Flexibilität von MPAs

mit der besseren UX von SPAs

Die Standard SPA

UI

Aufgebaut durch JavaScript

...z.B. mit React

Backend

JSON API vor einer Datenbank

inovex

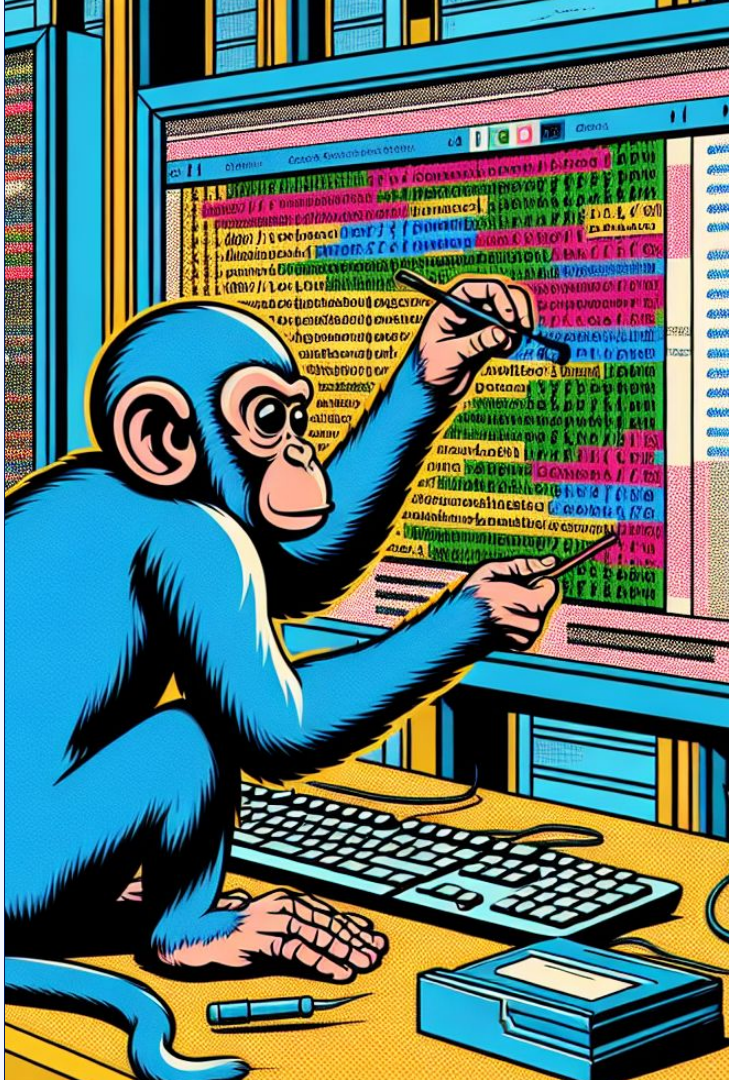
UI - Bundle Bizness

Viel Code führt zu großen Bundles

Wir splitten Bundles, um initiale Loads zu reduzieren

Teils mehrere Roundtrips nötig

Was passiert mit langen Nutzer-Sessions?



inovex

UI - Server Side Rendering

Initial liefern wir blank.html aus

Gefüllt wird das durch JS/React

Wir rendern die views vorher server-seitig

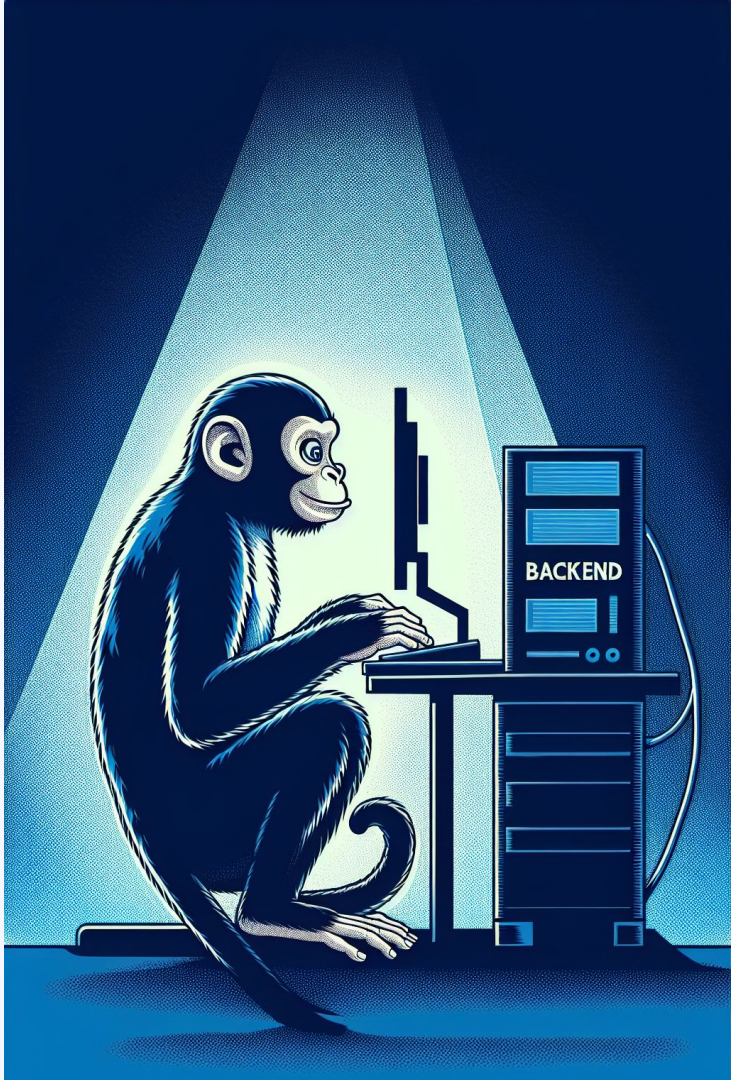
Interaktion ist sichtbar, aber nicht möglich

Backend - Generische APIs

Datenbank -> JSON REST API

Eine API bedient Web, App, ...

Oft mehrere requests pro view

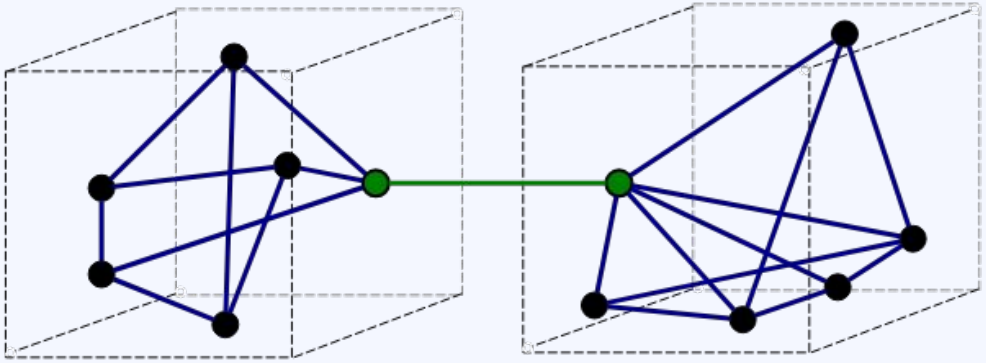


inovex

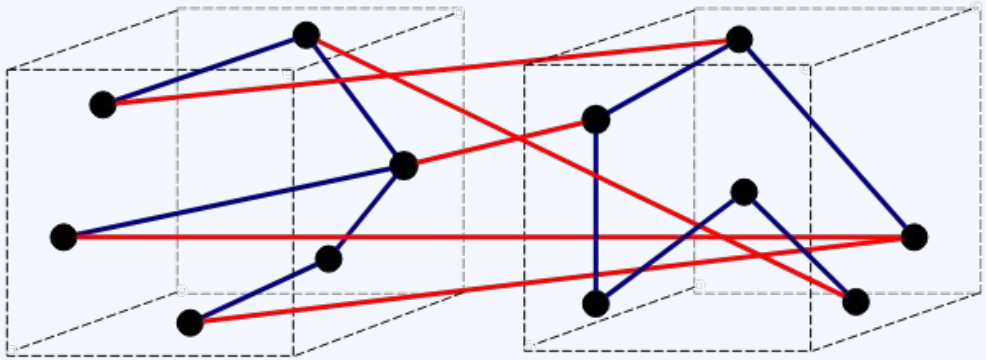
API Design

inovex

Coupling vs Cohesion



a) Good (loose coupling, high cohesion)



b) Bad (high coupling, low cohesion)

inovex

Decoupling auf Applikationsebene

Defeat

A: *Wir haben ne View, zeigt X, Y, Z an. Kann ich dafür nen Endpunkt haben*

B: *Ok ich bau dir nen Endpunkt*

Mit htmx haben wir eine Application API

Und damit Decoupling auf Systemebene

Die Vorteile von htmx

Alle Vorteile einer REST API

- Einfachheit, Zuverlässigkeit, ...

Alle Vorteile von SSR

- Caching, SQL tuning, ...

In einer eigenen API

Aber es gibt doch GraphQL

Generalisierter Zugriff auf Daten

Wir wollen “SQL Zugriff im Frontend”

With great power comes great responsibility

Wem können wir vertrauen?

Why not use content negotiation

Hypermedia API

session-cookie zur Authentifizierung

Getrieben durch die Anforderungen der Applikation

Data API

token-based Authentifizierung

Getrieben durch Konsumenten

versioniert, stabil

rate limited

Skaliert das?

inovex

Scaling Application Performance

- Should be stateless
  - nur Session-Cookie 👍
- Should support horizontal scalability
  - Hypermedia 👍
- Features should be independent
  - entkoppelte Endpunkte 👍
- Performance should be observable
  - per View Analysis 👍
- Should utilize caching
  - generell mit ausgereiften Backend Frameworks 👍

Scaling with Features

- Bisschen abhängig von der Architektur

- Generell skalieren HDAs aber sehr gut mit der Anzahl an Features

Scaling with Complexity

- Server-side Complexity: Yes
  - z.B. Chatbots

- Client-side Complexity: It depends
  - Viele schnelle Events oder inter-UI Abhängigkeiten können problematisch werden

Security

inovex

Grundlegende Regeln

- Route Control
- Auto-Escaping Template Engines
- Care for user generated content
- Cookie Settings

Only calls routes you control

```
DO
<button hx-get="/events">Search events</button>

DO NOT
<button hx-get="https://google.com/search?q=events">Search events</button>
```


inovex

Only call routes you control

- Eingebette <script> Tags würden sofort ausgeführt werden

- **Same Origin!**
  - Relative URLs
  - Keine CORS Sorgen
  - Nur mein eigenes Backend :)

Always use an auto-escaping template engine

```
<p>
{{ user.bio }}
</p>
```

User generierter Content muss escaped sein!

Zum Glück übernehmen das gute templating engines

Always use an auto-escaping template engine

Language	Template Engine	Escapes HTML by default?
JavaScript	Nunjucks	Yes
JavaScript	EJS	Yes, with `<%= %>`
Python	DTL	Yes
Python	Jinja	**Sometimes** (Yes, in Flask)
Ruby	ERB	Yes, with `<%= %>`
PHP	Blade	Yes
Go	html/template	Yes
Java	Thymeleaf	Yes
Rust	Tera	Yes

Serving User generated content

```
<!-- Don't include inside script tags -->
<script>
  const userName = {{ user.name }}
</script>

<!-- Don't include inside CSS tags -->
<style>
  h1 { color: {{ user.favorite_color }} }
</style>
<!-- Don't allow user-defined tag names -->
<{{ user.tag }}></{{ user.tag }}>

<!-- Don't allow user-defined attributes -->
<a {{ user.attribute }}></a>

<!-- User-defined attribute VALUES are sometimes okay, it depends -->
<a class="{{ user.class }}"></a>
```

hx-disable

Verhindert htmx processing auf dem Element und allen Kind Elementen.

Als Fallback für Teile, die Nutzer generierten Content enthalten.

Schützt die Kekse

- Secure:
  - Kein senden über HTTP
- HttpOnly:
  - Kein Zugriff über JS mit document.cookie
- SameSite=Lax:
  - Andere Seiten dürfen cookies nicht für requests verwenden

Accessibility

inovex

htmx & Accessibility

Eigentlich keine Probleme, aber...

```
<div hx-post="/mouse_entered" hx-trigger="mouseenter">
    [Here Mouse, Mouse!]
</div>
```


inovex

Wann htmx?

inovex

Wann htmx?

Yes

UI ist Bild & Text

UI ist viel CRUD

UI ist verschachtelt

Deep Links & First Render Performance

No

UI interdependencies

Offline Features

Häufige UI Updates

Team Support

“No One Ever Got Fired For Using React”

- Jake Lazaroff

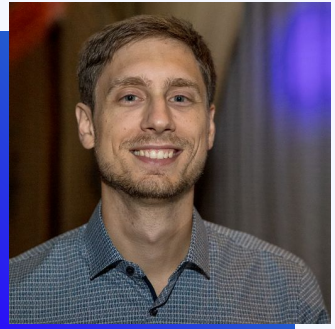
inovex

Vielen Dank!

inovex ist ein innovations- und qualitätsgetriebenes IT-Projekthaus mit dem Leistungsschwerpunkt „Digitale Transformation“.

- gegründet 1999
- 500+ Mitarbeitende
- 8 Standorte in ganz Deutschland

www.inovex.de



Jonas Kaltenbach

Software Developer

jonas.kaltenbach@inovex.de

inovex