

# **Weniger Komplexität, mehr Effizienz**

## **Moderne Web-Apps mit Server Side Rendering**

Raphael Zimmermann

Ergon Informatik AG

05.12.2023

**I ❤️ the Web**

I  the "modern" Web

# Registration

Name:

Email:

Phone:

Address:

Submit

```
1 <form method="POST" action="/submit-form">
2   <label for="name">Name:</label>
3   <input type="text" id="name" name="name" required>
4
5   <label for="email">Email:</label>
6   <input type="email" id="email" name="email" required>
7
8   <label for="phone">Phone:</label>
9   <input type="tel" id="phone" name="phone" pattern="[0-9]{10}" required>
10
11  <label for="address">Address:</label>
12  <textarea id="address" name="address" required></textarea>
13
14  <input type="submit" value="Submit">
15 </form>
```

```

1 import React, { useState, FormEvent } from 'react';
2
3 function CustomerForm() {
4   const [name, setName] = useState<string>('');
5   const [email, setEmail] = useState<string>('');
6   const [phone, setPhone] = useState<string>('');
7   const [address, setAddress] = useState<string>('');
8   const [outcome, setOutcome] = useState<string>('');
9
10  const handleSubmit = async (event: FormEvent<HTMLFormElement>): Promise<void> => {
11    event.preventDefault();
12
13    const formData = {
14      name,
15      email,
16      phone,
17      address,
18    };
19
20    const response = await fetch('/submit-form', {
21      method: 'POST',
22      headers: {
23        'Content-Type': 'application/json',
24      },
25      body: JSON.stringify(formData),
26    });
27
28    if (response.ok) {
29      setOutcome('Form submitted successfully!');
30    } else {
31      setOutcome('Error submitting form.');
```

Browser Feature neu implementiert

Browser Feature neu implementiert

Identisches "HTML" + Event-Handler

```

32  }
33  }
34  }
35  return (
36    <div>
37      <outcome && <p>{outcome}</p>>
38      <form onSubmit={handleSubmit}>
39        <label htmlFor="name">Name:</label>
40        <input type="text" id="name" name="name" value={name} onChange={(event) => setName(event.target.value)} required />
41
42        <label htmlFor="email">Email:</label>
43        <input type="email" id="email" name="email" value={email} onChange={(event) => setEmail(event.target.value)} required />
44
45        <label htmlFor="phone">Phone:</label>
46        <input type="tel" id="phone" name="phone" pattern="[0-9]{10}" value={phone} onChange={(event) => setPhone(event.target.value)} required />
47
48        <label htmlFor="address">Address:</label>
49        <textarea id="address" name="address" value={address} onChange={(event) => setAddress(event.target.value)} required</textarea>
50
51        <button type="submit">Submit</button>
52      </form>
53    </div>
54  );
55  }
56  export default CustomerForm;
57
```

Server Side Rendering =  
weniger Komplexität  
schnellere Entwicklung  
gleichwertige UX

# Agenda

- Wie sind wir überhaupt da gelandet?
- Server vs. Client-Generated Views
- HTML over the Wire Ansatz
- Zusammenfassung



# Das Web um ~2007

- Weniger Dynamik
- Dynamische Inhalte mit Flash
- Full Page Reload
- Problem: Browser Kompatibilität

The screenshot shows the MySpace homepage with a navigation bar at the top containing links like Home, Browse, Search, Invite, Film, Mail, Blog, Favorites, Forum, Groups, Events, Videos, Music, Comedy, and Classifieds. The main content area is divided into several sections:

- Cool New Videos:** A section titled "50,620 uploaded today!" featuring four video thumbnails: "Valentines" by Steve Rosswick, "Secret Lives" by Black 20, "For Singles" by Danny, and "Working Cupid" by Cupid.
- Member Login:** A blue box with input fields for "E-Mail" and "Password", a "Remember Me" checkbox, and "LOGIN" and "SIGN UP!" buttons. A link for "Forgot your password?" is at the bottom.
- Cool New People:** A section with three profile thumbnails for "Alejandro", "Renee", and "Milo".
- Videos:** A section titled "Love On Myspace" with a video thumbnail and the text "What goes through a man's mind online looking for love on Myspace." and a "Watch It Now!" link.
- MySpace Music:** A section for "Comeback Kid" (Hardcore / Punk, Winnipeg, Canada) with a photo and a red "EXCLUSIVE" badge. Text below reads: "Since forming in 2002, Winnipeg's Comeback Kid has become what AMP Magazine called 'a wildly successful scene institution.' Watch for the release their third album, 'Broadcasting...' out February 20" and a "Listen Now" link.
- MySpace Specials:** A section for "Kokua Festival Exclusive Presale Feb 15!" with a colorful poster and text: "Add Kokua Festival to get an early crack at tickets to Jack Johnson's Earth Day celebration in Honolulu on April 21-22. This year's lineup includes Jack Johnson, Eddie Vedder and Boom Gaspar of Pearl Jam, Matt Costa and more." and a "Check it out!" link.

At the bottom, there are four blue boxes with promotional text and links:

- Get Started On MySpace!** Join for free, and view profiles, connect with others, blog, rank music, and much more! [» Learn More](#)
- Create Your Profile!** Tell us about yourself, upload your pictures, and start adding friends to your network. [» Start Now](#)
- Browse Through Profiles!** Read through millions of profiles on MySpace! See pics, read blogs, and more! [» Browse Now](#)
- Invite Your Friends!** Invite your friends, and as they invite their friends your network will grow even larger! [» Invite Friends Now](#)

Bild Quelle: [webdesignmuseum.org](http://webdesignmuseum.org)

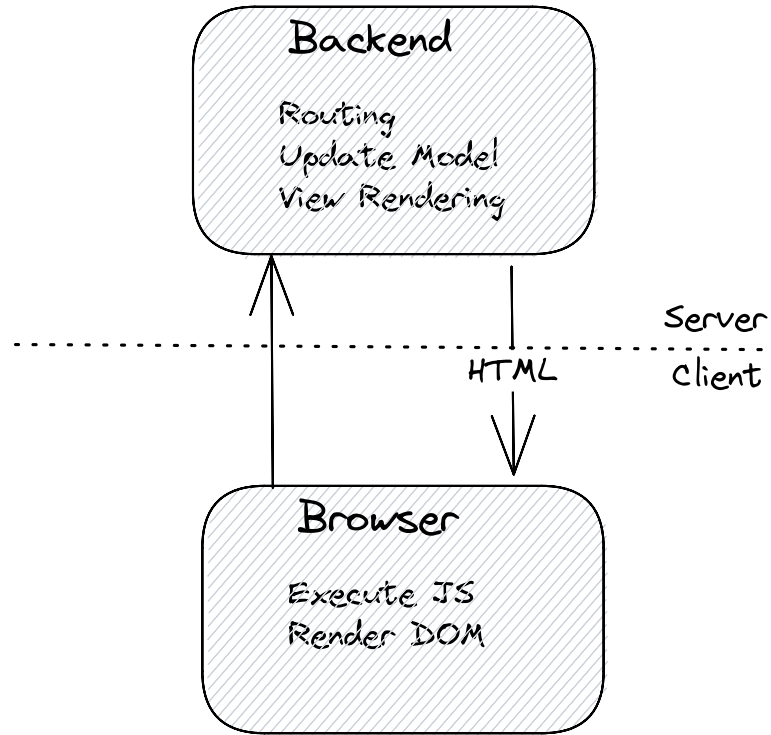
# Neue Generation von Webanwendungen

- Mehr Dynamik erwünscht
- Insbesondere Partial Page Updates
- Ajax wird Populär
- Neue Probleme 🍷

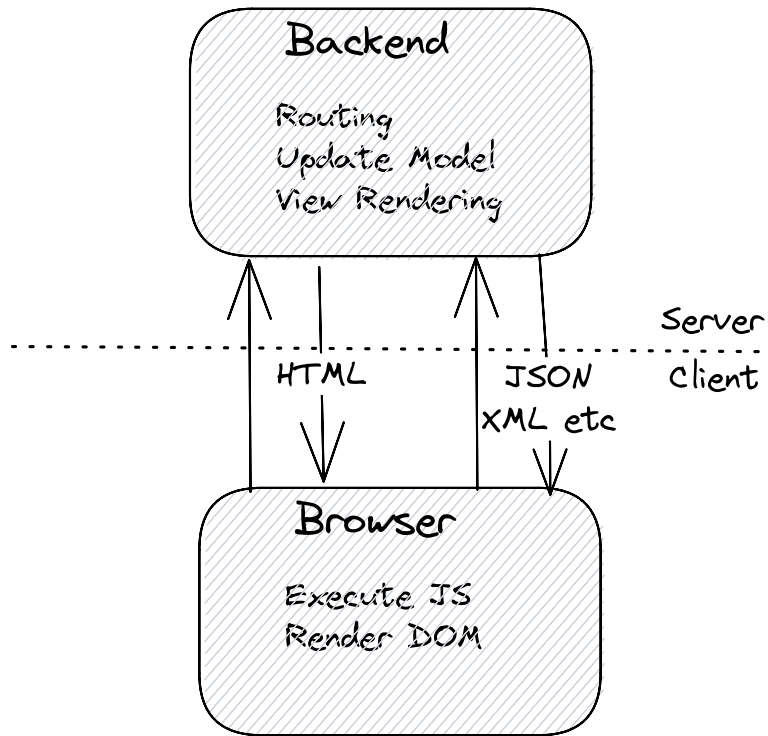
The screenshot displays the Gmail web interface for the user frank@gmail.com. At the top, there are navigation links for Mail, Calendar, Documents, Photos, Groups, Web, and more. A search bar is present with a search button. The main content area shows a list of emails, including messages from Google Calendar, Philipp, me (2), The Box Team, alert, and various Google Blogscoped Forum entries. On the left side, there is a sidebar with navigation options like Compose Mail, Inbox (228), Starred, Chats, Sent Mail, Drafts (1), All Mail, Spam, and Trash. Below these are Contacts and Labels, with a search bar for contacts and a list of labels including Family, Gmail Drive (1), Job (1), and sent to cellphone. An 'Invite a friend' button is also visible.

Quelle: [blogscoped.com](http://blogscoped.com)

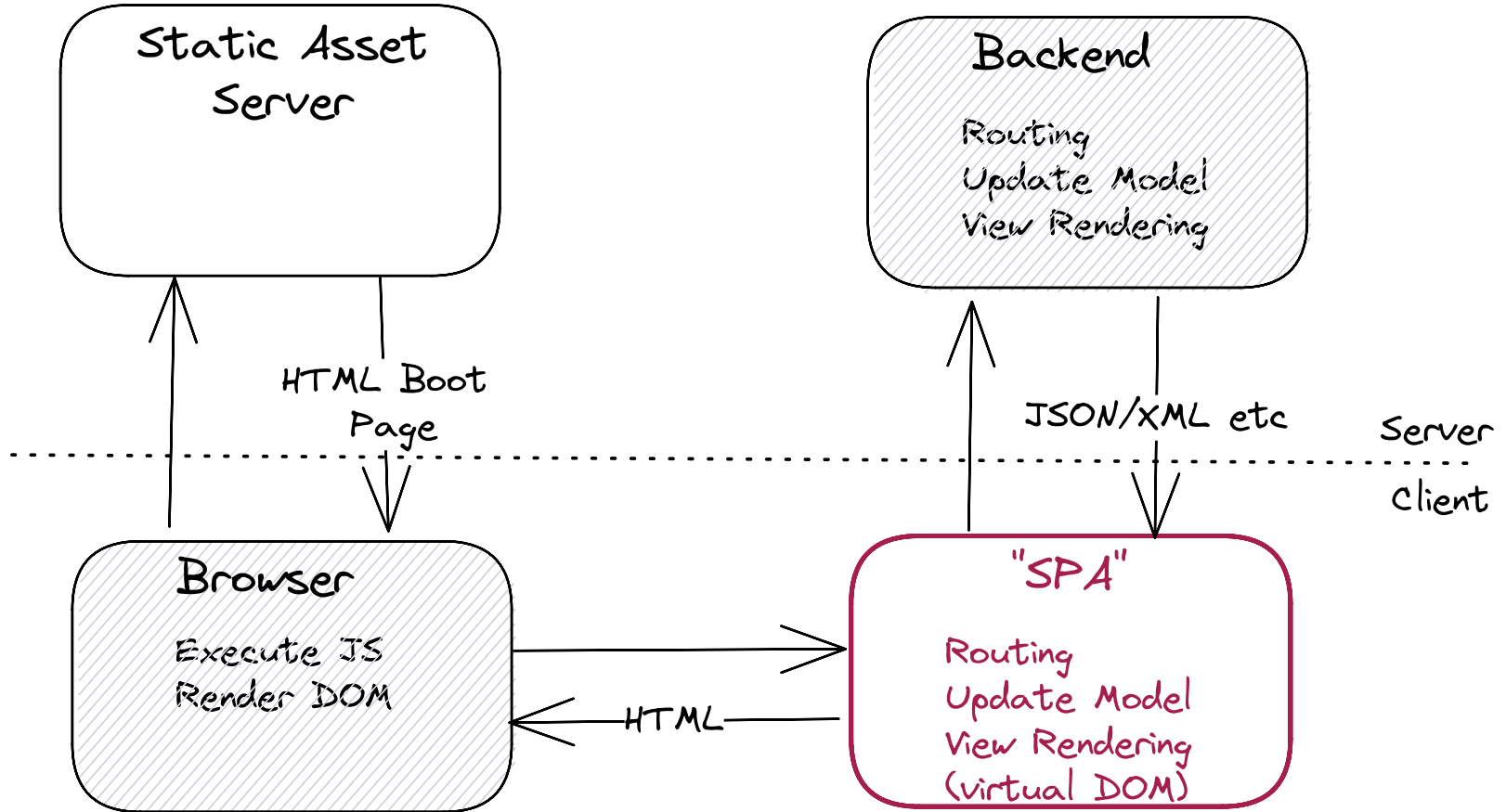
# Server-Generated Views



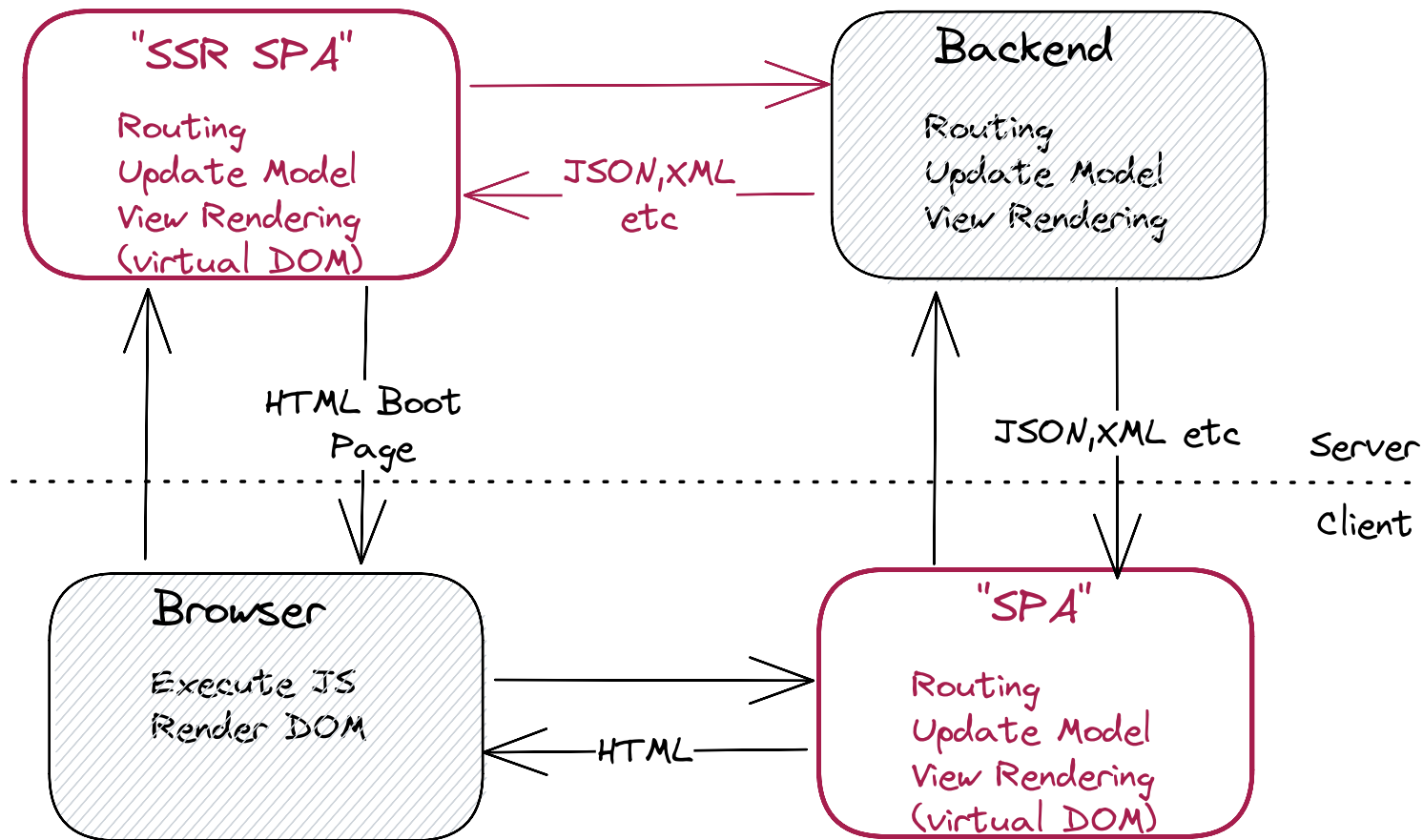
## Server-Generated Views + Ajax



# Client-Generated Views



# Pre-Rendered Client-Generated Views



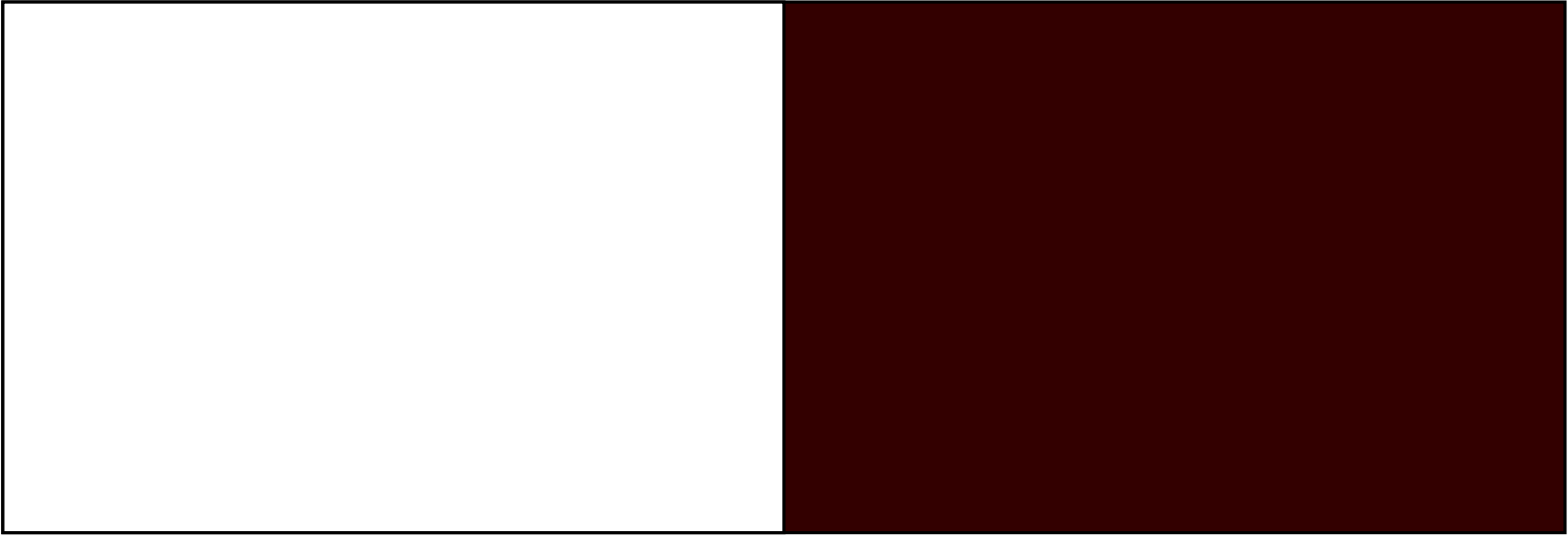
# Herausforderungen von Client-Generated Views

- Initial Page Load ist langsam
- SEO nach wie vor nicht so gut
- Benötigt viel spezifisches Know-How
  - um Routing, Forms, Reactivity "richtig" zu machen
- "Wasserfall"-Netzwerk calls
- Viel Last auf Client Geräte
- **Frontend Cache**

# Zwei Codebasen

- 2 Deployments
- 2 Sprachen / Ökosysteme
- Fördert Trennung von Frontend und Backend
- Definition von API-Contracts
- Benötigt mehr Entwickler
- Kostet alles Zeit und Geld 📦



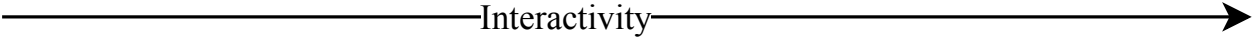


All  
Server-  
Generated

All  
Client-  
Generated



All  
Server-  
Generated



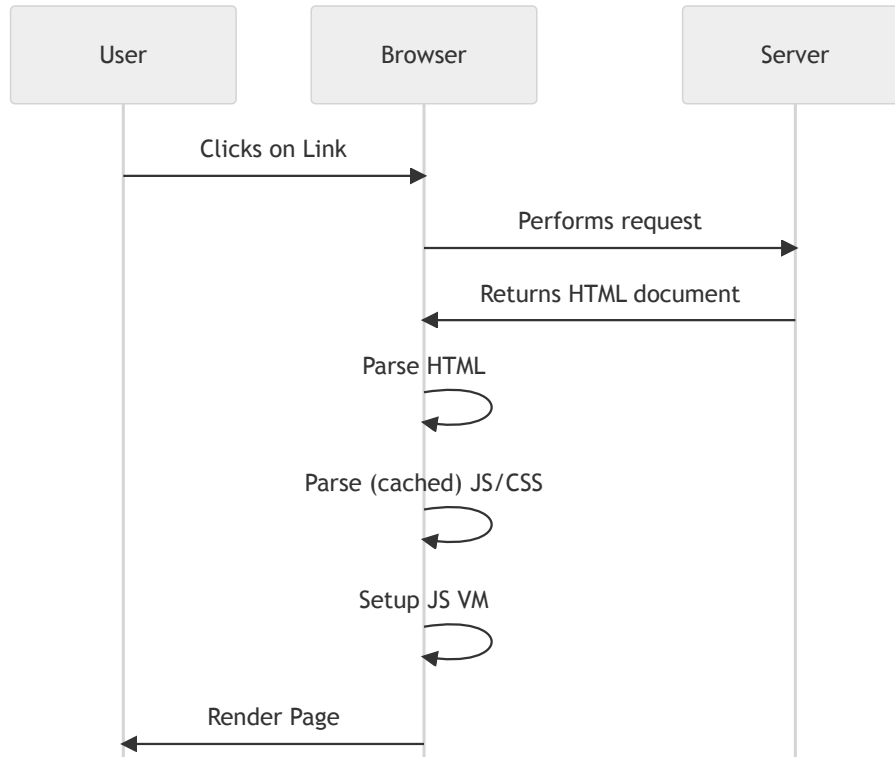
All  
Client-  
Generated

# HTML Over The Wire

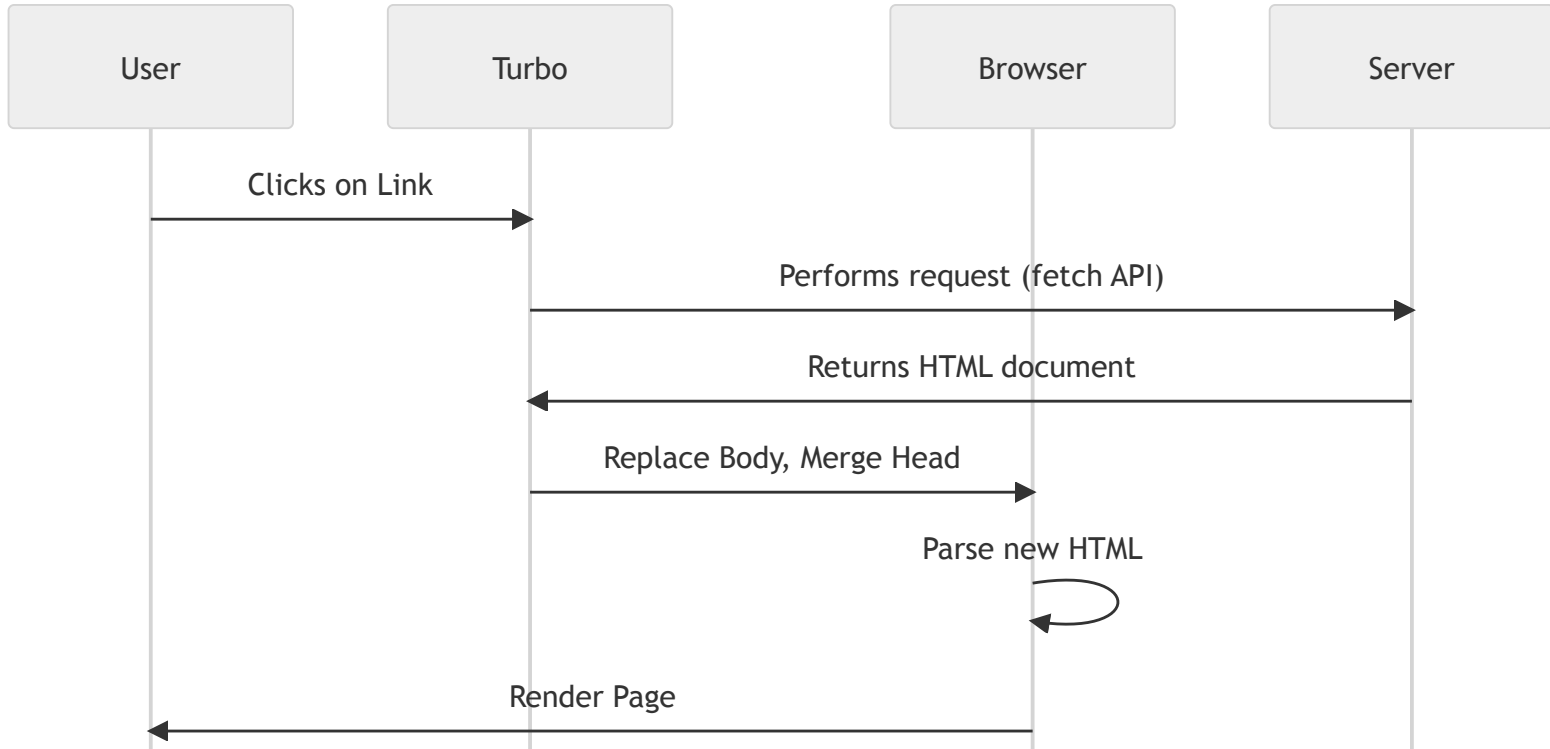
- Wir generieren all unser HTML auf dem Server
- senden HTML anstelle von JSON/XML zum Client

# Hotwired Turbo

# Navigation without Turbo Drive (Full Page Reload)



# Navigation with Turbo Drive




# Demo

# Turbo Frames

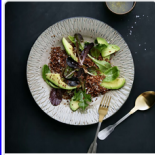
Kochbuch [Recipes](#) [Neues Rezept](#)

## Rezepte


Suche  Suche




**Tofu-Stir-Fry**




**Quinoasalat mit Avocado**




**Krautstiel-Pasta mit Kräutersauce**




**Quinoa-Sauerampfer-Salat mit Speck**




**Rucola mit Formaggini und Croûtons**



**Pikanter Hackfleischsalat mit Mais**



**Papaya-Kichererbsen-Salat**

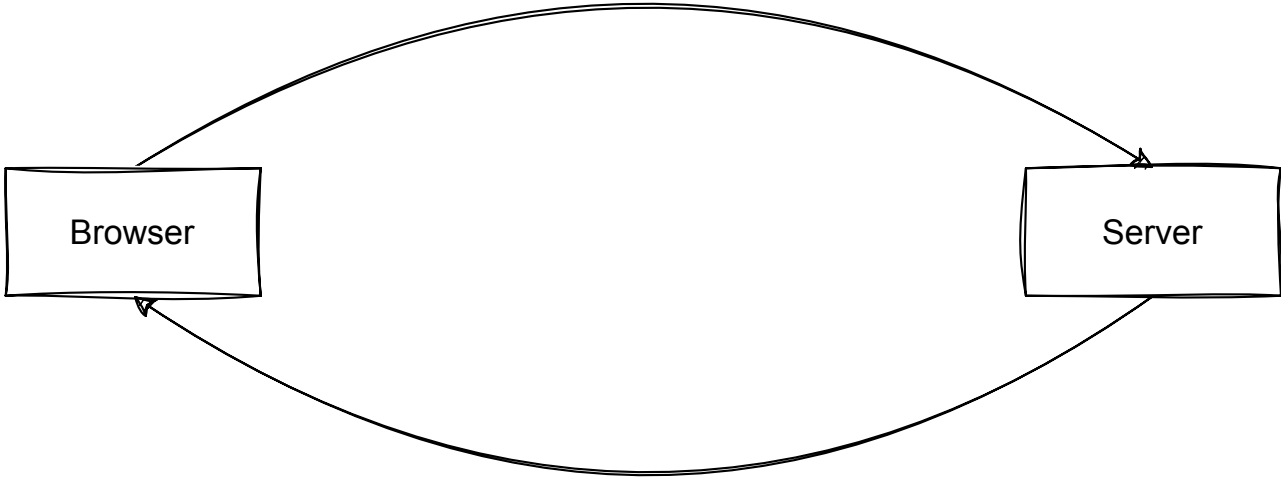


**Poulet-Avocadosalat**

1 2 3 4 5 6 7 [Nächste Seite >](#)



Quinoa Suche



Three recipe cards are displayed in a row, each with a photo and a title:

- Quinoasalat mit Avocado**
- Quinoa-Sauerampfer-Salat mit Speck**
- Quinoa-Gemüse-Pfanne**

Below the cards is a small black square containing the number 1.

# It's just HTML

```
1 <turbo-frame id="recipe-list">
2   <div>...</div>
3 </turbo-frame>
4 <turbo-frame id="recipe-42">
5   
6
7 </turbo-frame>
```

# Demo

# Server Push mit Turbo Streams

## Client

```
1  const eventSource = new EventSource("/events");  
2  Turbo.session.connectStreamSource(eventSource);
```

## Server

```
1  sseService.send(event.sseEmitterId, renderHtml())
```

## HTML

```
1  <turbo-stream action="replace" target="recipe-42">  
2    <template>  
3        
4      ...  
5    </template>  
6  </turbo-stream>
```

# Demo

# Interaktivität

## Beispiel: Password Visibility

Password

# Stimulus.js: HTML

```
1 <div data-controller="password-visibility">
2   <input type="password" data-password-visibility-target="input" spellcheck="false" />
3
4   <button type="button" data-action="click->password-visibility#toggle">
5     <span data-password-visibility-target="icon">Eye</span>
6     <span data-password-visibility-target="icon" class="hidden">Eye Slash</span>
7   </button>
8 </div>
```

Quelle: [stimulus-components.com](https://stimulus-components.com)



# Stimulus.js: JavaScript

```
1 // <... data-controller="password-visibility" ...>
2 stimulus.register("password-visibility", class extends Controller {
3   static targets = [
4     "input", // <... data-password-visibility-target="input" ...>
5     "icon",  // <... data-password-visibility-target="icon" ...>
6   ];
7
8   connect() {
9     this.hidden = this.inputTarget.type === "password";
10  }
11
12  toggle(e) { // <... data-action="password-visibility#toggle" ...>
13    e.preventDefault();
14
15    this.inputTarget.type = this.hidden ? "text" : "password";
16    this.hidden = !this.hidden;
17
18    this.iconTargets.forEach((icon) => icon.classList.toggle("hidden"));
19  }
20 }
21 );
```

Quelle: [stimulus-components.com](https://stimulus-components.com)

# Server-Generated Views

+ HTML rendern ist einfach

+ Eine Codebasis

+ Weniger Komplexität

+ **einfaches Ausführungs-  
Modell**

👉 **Erhöht Produktivität**

— Fehlende (Component) Libraries

👉 CSS Frameworks

— Libraries müssen auch gut verstanden werden

— Fehlende Framework Integrationen

# Wann soll ich jetzt was einsetzen?

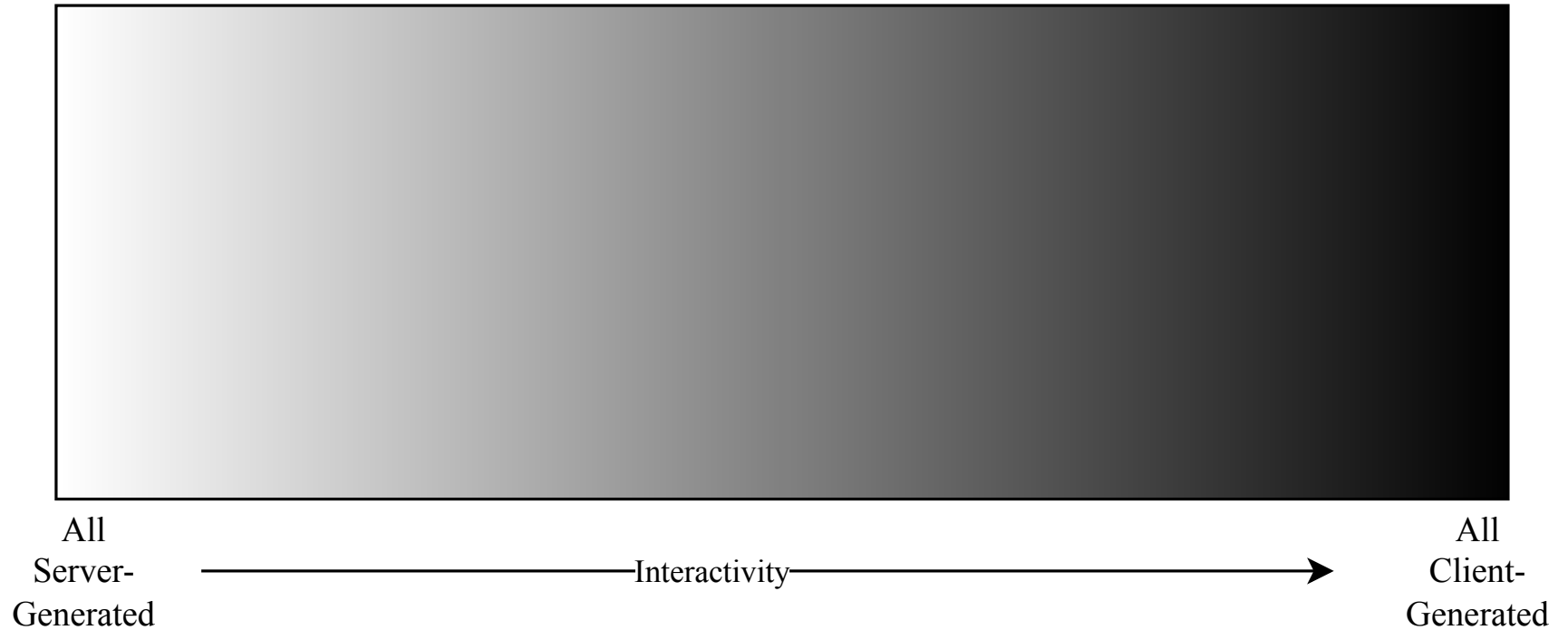
## Server-Generated

- Default!
- Auch wenn API vorhanden (Bsp. Mobile Apps)

## Client-Generated

- E2E Encryption (Bsp. Threema, Whatsapp etc.)
- 100% Offline fähig
- Sehr sehr komplexe Komponenten (Bsp. Figma)
- Organisatorische Trennung von Frontend und Backend

# Zusammenfassung



hi@raphael.li



raphael.zimmermann@ergon.ch



raphiz

# Weiterführende Ressourcen

- [Hotwire - Offizielle Webseite](#)
- [Demo Anwendung auf GitHub](#)
- [RailsConf 2016 - Turbolinks 5: I Can't Believe It's Not Native! by Sam Stephenson](#)
- [Hotwire Turbo Chat with Micronaut Views](#)
- [Hotwire Weekly Newsletter](#)
- [Resource-oriented Client Architecture - ROCA](#)
- [Hypermedia Systems](#)
- [Modern web apps without JavaScript bundling or transpiling](#)
- [JavaScript modules - Performance recommendations \(v8.dev\)](#)