



Web-Programmierung WWI21SEB

Aaron Schweig, SAP

Non-Business

Wer bin ich?

Aaron Schweig, 25 Jahre

- Seit 2016 freiberufliche Tätigkeiten im Bereich Webentwicklung
- 2018-2021 Bachelor Wirtschaftsinformatik
- Seit 2021 Fullstack Development @SAP

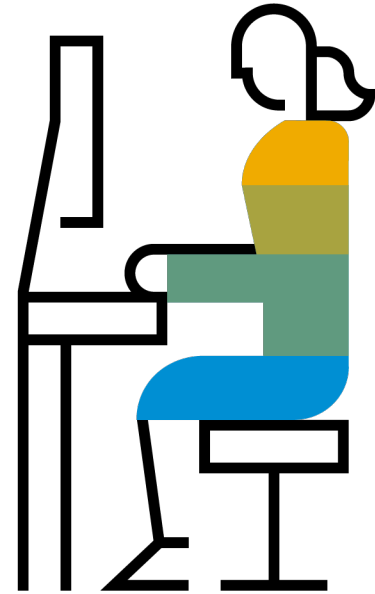
Hobbies:

- Musik (Klavier, Gitarre, etc.)
- Videospiele



Wer seid ihr?

- Name, Alter, Firma
- Hobbies
- Erfahrung mit Web-Development (privat, in der Firma, etc.)
- **Was erhoffe ich mir von der Vorlesung?**



Prüfungsleistung: Portfolio (70 P.)

1. Projekt: Entwicklung einer Webanwendung

- **Abgabe 14.07.2023:** Code als Git-Repository

2. Dokumentation

- Projektidee
- Anforderungen
- Wireframes / Mockups
- Architektur (Technologien, Gründe für die Auswahl von Technologien, Diagramme etc.)
- **Abgabe 14.07.2023:** PDF

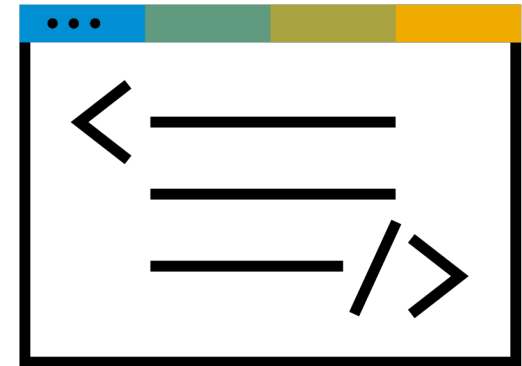
3. Präsentation

- In der letzten Vorlesung (14.07.2023)
- Individuelle Leistungen müssen erkennbar sein
- Gruppengröße: 5 – 6 Studierende



Voraussetzungen

- [GitHub](#) Account
- Entwicklungsumgebung
 - [Visual Studio Code](#) (recommended)
 - WebStorm
- Arbeiten mit [GIT](#) als Version Control System (VCS)
 - <https://github.com/git-guides>
 - <https://docs.github.com/en/get-started/quickstart/hello-world>



Das Vorlesungsrepository klonen

1

```
git clone https://github.com/aaronschweig/wwi21seb.git
```

2

Repository in eurer Entwicklungsumgebung öffnen

Agenda

1

Einführung und Grundlagen

Grundlegende Einführung in die Webentwicklung und das Ökosystem

2

Architektur- und Kommunikationspattern

Muster und Best Practices bei der Entwicklung von Webanwendungen

3

Fortgeschrittene Frontend-Entwicklung

Betrachtung aktueller Frameworks am Beispiel von Svelte

4

Authentifizierungsmechanismen

Mechanismen zur Authentifizierung von Nutzern, Standards wie OAuth2, OIDC, etc.

Agenda

5

Testen von Webanwendungen

Webentwicklungsprojekte automatisiert testen

6

Deployment / Hosting von Webanwendungen

Überblick über Möglichkeiten zur Bereitstellung von Webanwendungen

7

Projektvorstellung 🎉

Vorstellung der Prüfungsleistung in der Vorlesung



Web-Programmierung Einführung und Grundlagen

Non-Business

Agenda

1.1

**HTML, CSS &
JavaScript**

+ Übung

1.2

NodeJS, (p)npm

1.3

TypeScript

1.4

**Nächste Vorlesung,
nützliche
Dokumentationen/Links
& Fragen**

1.1

HTML, CSS & JavaScript

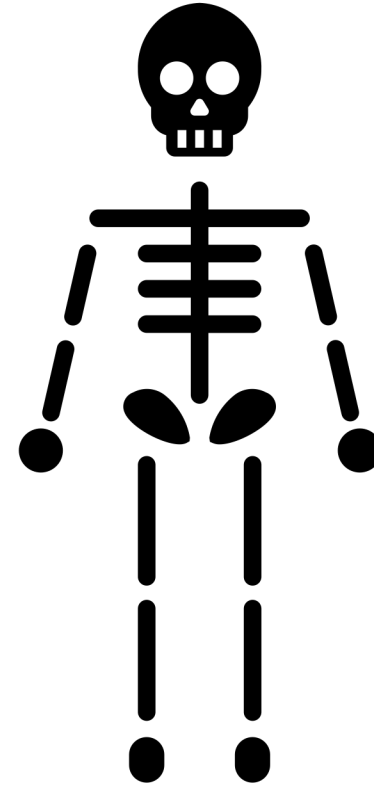
Grundlegende Technologien zur Entwicklung von Webanwendungen

Non-Business

HTML

HyperText Markup Language

- Das *Skelett* einer Website / Webanwendung
- Dient zur **Strukturierung** der Inhalte einer Website
 - Durch die Nutzung von **Tags**
- *Beispieltags:*
 - `<div> </div>`
 - `<h1> </h1>`
 - ...



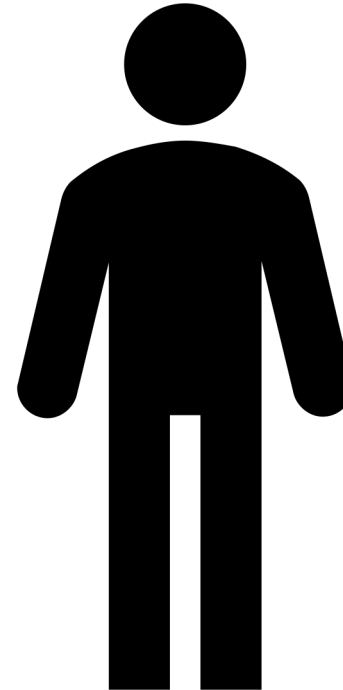
HTML Beispiel

```
<!DOCTYPE html>
<html>
  <head>
    <title>Das ist ein Titel</title>
  </head>
  <body>
    <div>
      <p>Hallo WW21SEB!</p>
    </div>
  </body>
</html>
```

CSS

Cascading Style Sheets

- Die Gestaltung einer Website / Webanwendung
- Dient zur Anpassung der Darstellung des HTML-Skeletts
 - Layout
 - Farben
 - Typographie
- *Beispielklassen:*
 - `display: flex`
 - `background-color: green`
 - ...



CSS Beispiel

```
<!DOCTYPE html>
<html>
  <head>
    <title>Das ist ein Titel</title>
  </head>
  <body>
    <div>
      <p style="color: red">Hallo WW21SEB!</p>
    </div>
  </body>
</html>
```

Inline Styles

```
p {
  color: red;
}
```

Separate CSS Klassen

JavaScript

- Die Programmiersprache für Webentwicklung
 - Wird durch alle Browser unterstützt
- Definiert das Verhalten einer Webanwendung
- Standardisiert als ECMAScript



JavaScript Beispiele

Variablen (nicht typisiert 😬)

```
let y;  
y = 10;  
const z = "this value cannot be reassigned!";
```

Funktionen

```
function foo(n) {  
  return n + 1;  
}
```

Mit einem DOM interagieren

```
const myElem = document.createElement('span');  
  
myElem.classList.add('foo');  
myElem.id = 'bar';  
myElem.setAttribute('data-attr', 'baz');  
  
document.body.appendChild(myElem);  
  
document.querySelector('.class');
```

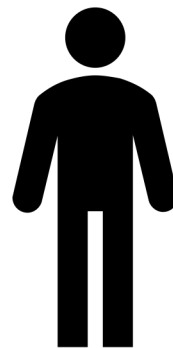
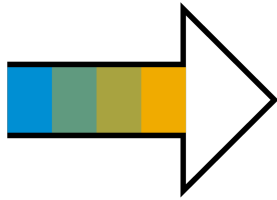
Klassen, Prototyping ...

Übung 1

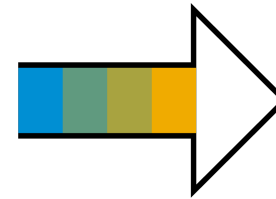
- **Erstellt eine Website über euch selbst.**
 - Name, Alter, Bild (freiwillig)
 - Liste mit Hobbies / Erfahrung mit Webentwicklung
 - Neue Hobbies und Erfahrungen können über Textfeld mit Button hinzugefügt werden



HTML



+ CSS



+ JavaScript

- Website in einem Git-Repository speichern / commiten

1.2

NodeJS, NPM und Bundling

Bereitstellung von Webanwendungen, Einbindung von 3rd Party Code

Non-Business

NodeJS, (P)NPM and Bundling

- plattformübergreifende Open-Source-JavaScript-Laufzeitumgebung
- JavaScript kann außerhalb des Webbrowsers ausgeführt werden
- Basiert auf Googles V8 JavaScript engine
- <https://nodejs.org/en>



(P)NPM

- **N**ode **P**ackage **M**anager
- Paketmanager für die JavaScript-Laufzeitumgebung Node.js
- Erlaubt Veröffentlichung und Konsumierung von 3rd Party JavaScript Paketen im eigenen Code
- <https://www.npmjs.com/>
- <https://pnpm.io/>



Bundling

- Verkleinern und Optimieren von Assets
- Fingerprinting von Assets für besseres Caching
- Erzeugen client-spezifischer Assets
- Unterstützen neuer JS-Features
- Einbinden von Polyfills
- Transpilieren von Code
- Beispiel: <https://vitejs.dev/>



1.3

TypeScript

JavaScript + Typisierung

Non-Business

Was ist TypeScript?

- **Strikt typisierte** Programmiersprache, die auf JavaScript basiert
- Besseres Tooling (IDE Autocompletion, Compile Time Errors, besseres Refactoring, etc)
- Superset von JavaScript → Jeder valide JavaScript Code ist auch valider TypeScript Code
- Danke modernen Tools wie Bundlern und Deno sehr weit verbreitet



1.4

Dokumentation / Links, nächste Vorlesung & Fragen

Non-Business

Nützliche Dokumentation/Links zu Web-Standards

- MDN <https://developer.mozilla.org/en-US/>
- W3Schools <https://www.w3schools.com/tags/default.asp>
- IETF standards <https://datatracker.ietf.org/> (useful in the later parts of the lecture)

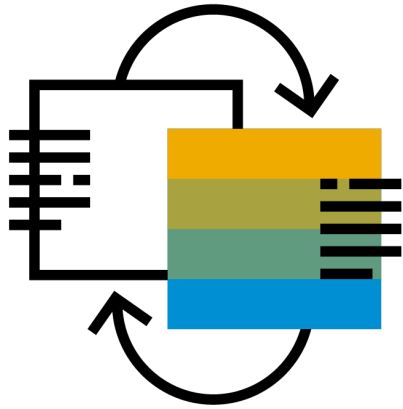
Aufgaben bis zum 26.05.2023

- Gruppen selbstständig festlegen
- Projektidee festhalten → Als Issue im Vorlesungsrepo



Architektur- und Kommunikationspattern

Non-Business

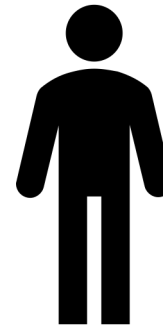
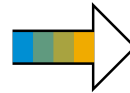


Recap

HTML, CSS, JavaScript
Node.js, (p)npm etc.



HTML



+ CSS



+ JavaScript

Agenda

1.1

**Frontend &
Backend**

1.2

**Single Page App (SPA)
vs.
Server Side Rendered (SSR)**

1.3

**Fetching Data with
APIs**

REST API Hands-on
Session

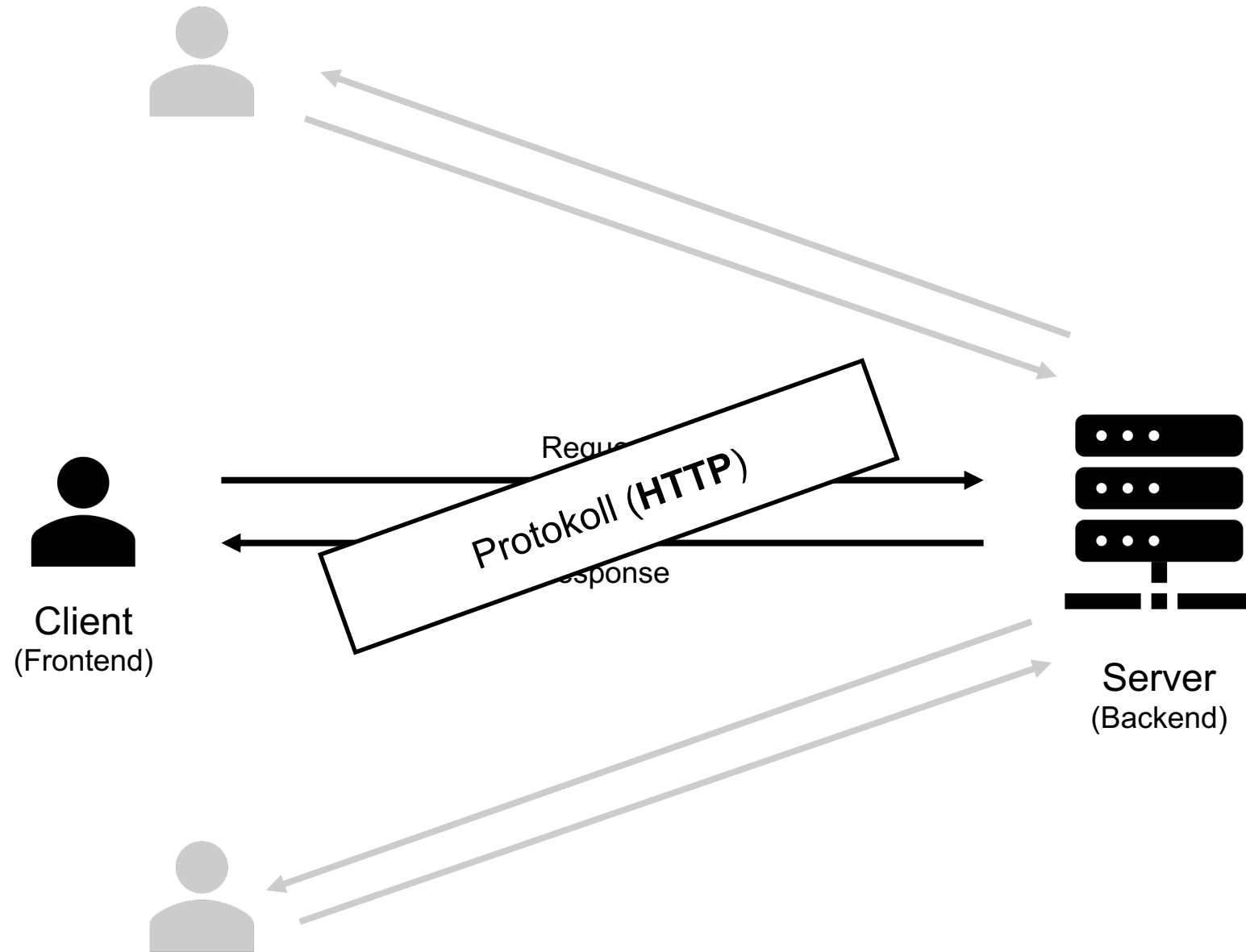
1.4

**Using Server Side
Rendering**

2.1

Frontend & Backend

Wie funktioniert die Kommunikation zwischen Client und Server?



HTTP

Hypertext Transfer Protocol

- Zustandsloses application-level Protokoll
- Basis für nahezu gesamte Kommunikation im Internet
- **Request** und **Response** bestehen aus:
 - Header
 - Body
 - Methode
 - Pfad
 - Protokoll (HTTP 1.1/ HTTP 2 / HTTP 3)
 - Response Code



Request

method	path	protocol
GET	/tutorials/other/top-20-mysql-best-practices/	HTTP/1.1
Host: net.tutsplus.com		
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1		
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=		
Accept-Language: en-us,en;q=0.5		
Accept-Encoding: gzip,deflate		
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7		
Keep-Alive: 300		
Connection: keep-alive		
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120		
Pragma: no-cache		
Cache-Control: no-cache		

HTTP headers as Name: Value

Response

protocol	status code
HTTP/1.x	200 OK
Transfer-Encoding: chunked	
Date: Sat, 28 Nov 2009 04:36:25 GMT	
Server: LiteSpeed	
Connection: close	
X-Powered-By: W3 Total Cache/0.8	
Pragma: public	
Expires: Sat, 28 Nov 2009 05:36:25 GMT	
Etag: "pub1259380237;gz"	
Cache-Control: max-age=3600, public	
Content-Type: text/html; charset=UTF-8	
Last-Modified: Sat, 28 Nov 2009 03:50:37 GMT	
X-Pingback: http://net.tutsplus.com/xmlrpc.php	
Content-Encoding: gzip	
Vary: Accept-Encoding, Cookie, User-Agent	

HTTP headers as Name: Value

HTTPS

HTTP, but *secure* 🔒

- **TLS/SSL** verschlüsselt
- Nutzt Zertifikate

Default Ports:

- HTTP: 80
- HTTPS: 443



http

https

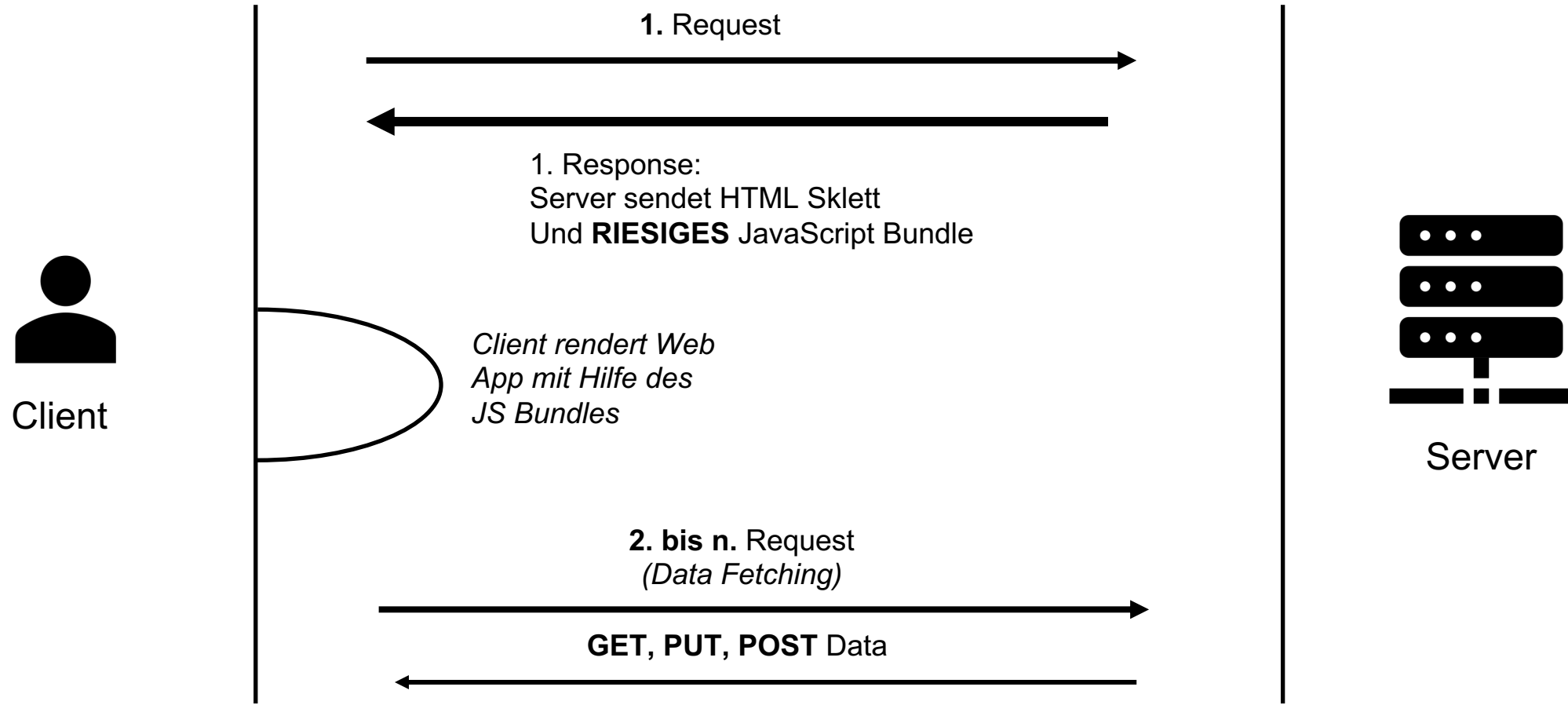
2.2

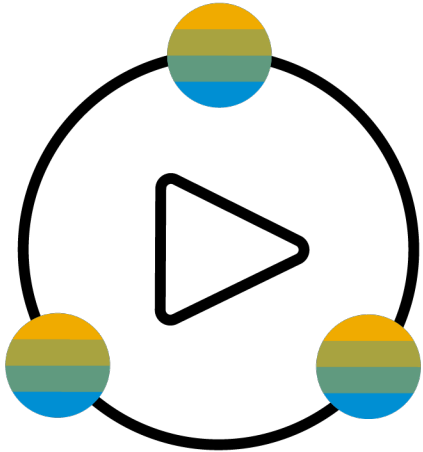
Single Page App vs. Server Side Rendered

Was ist das überhaupt? Und was sind die Unterschiede?

SPA

Single Page Web Application





Single Page Web Application

Beispiel

SSR

Server Side Rendering



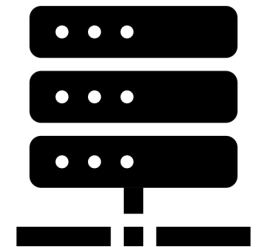
Client

1. Request

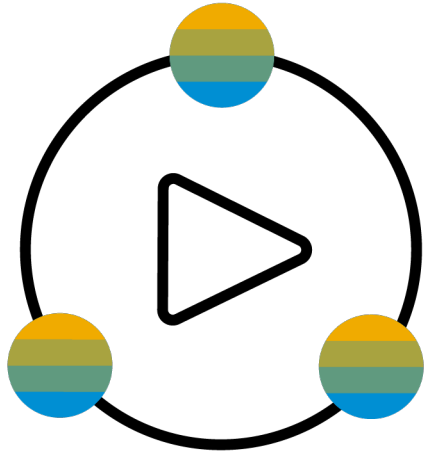
1. Response:
Server sendet fertiges HTML und kein
bis wenig JavaScript

2. bis n. Request
(HTML wird direkt von Server
konstruiert)

GET HTML



Client



Server Side Rendered Web Application

Beispiel

2.3

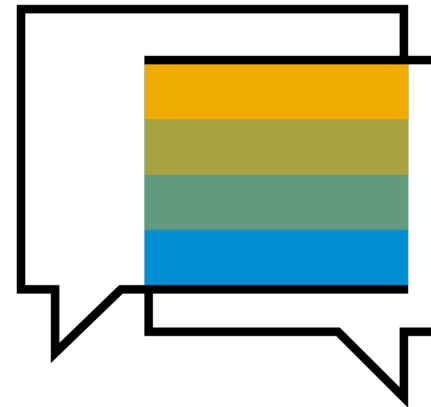
Fetching Data with APIs

Wie erhält der Client Daten vom Server?

Non-Business

Paradigmen zur Kommunikation zwischen Client und Server

- *Request Driven (Pull)*
 - **RESTful API**
 - graphql
- *Event Driven (Push)*
 - Server Sent Events (SSE)
 - WebSockets (bidirektionale Kommunikation)



REST APIs

Representational State Transfer

- Paradigma zur Übertragung von Ressourcen (Daten) zwischen Client und Server
- Ressourcen werden über URIs identifiziert
- Nutzung von HTTP Request Methods zur Abbildung von CRUD Operationen
 - Create → POST
 - Read → GET
 - Update → PUT/PATCH
 - Delete → DELETE
- Nutzung von HTTP Statuscodes zur Abbildung des Ergebnisses einer Operation
 - 200 OK, 404 Not Found etc.

Beispiel: <https://tasks.moritzmoe.de/api>

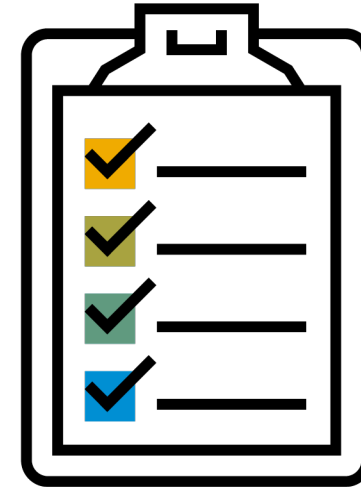
Referenz: <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design>

Übung 2

- **Erstellt eine Website mit der sich eine Aufgabeliste erstellen und verwalten lässt.**
 - Aufgaben hinzufügen
 - Aufgaben als erledigt markieren
 - Aufgaben löschen

Bonus:

- Aufgaben bearbeiten
- Aufgaben suchen
- Pagination



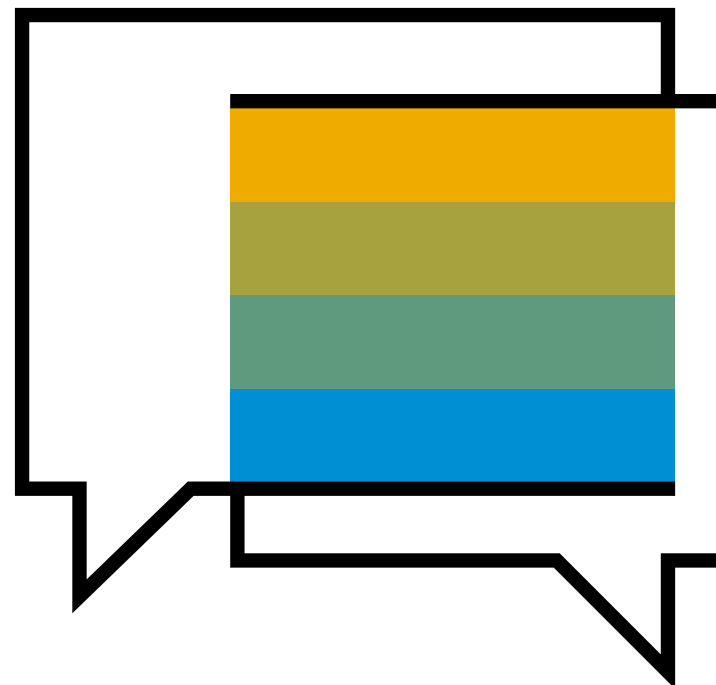
2.4

Using Server Side Rendering

Live-Coding 

Non-Business

Q & A



Vielen Dank.

Contact information:

Aaron Schweig
aaron.schweig@sap.com

