

# WEB-APPLIKATIONEN

## CLIENT UND SERVER



# THEMENÜBERSICHT

1. AUFBAU MODERNER WEB-ANWENDUNGEN
2. MERN-STACK
  - Backend-Server
  - Frontend-Server
  - Datenbank
3. NODE.JS UND EXPRESS



# ZIELSETZUNG

1. BISHER AUSSCHLIESSLICH TECHNOLOGIEN AUF CLIENT-EITE
  - HTML, CSS3, DOM, JavaScript
2. WEB-APPLIKATIONEN SIND ABER VERTEILTE CLIENT-SERVER ANWENDUNGEN
  - Sowohl Client als auch Server übernehmen Teile der Verarbeitung
3. ZIEL: AUSTAUSCHBARKEIT VON TECHNOLOGIEN



# ARCHITEKTUR

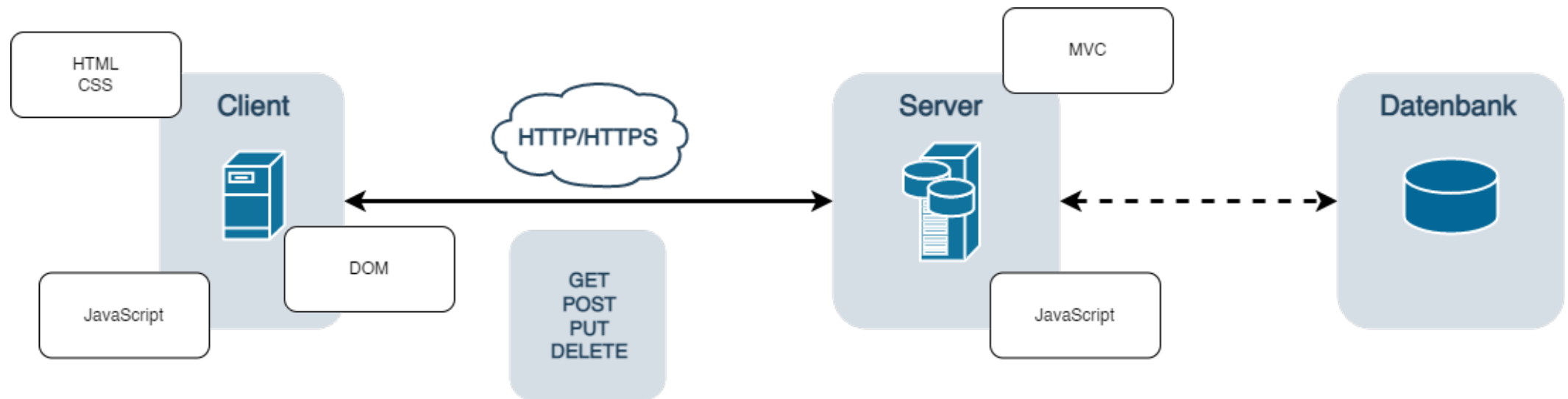
› Sind verteilte Anwendungen

› „Teilen“ sich in:

Backend

Frontend

Datenbank



Moderne Webanwendungen



# BACKEND

- › Teil der Applikation, der nicht sichtbar ist
- › Implementiert Logik
- › Sorgt für Datenzugriffe
- › Stellt Schnittstellen (z.B. REST) bereit
- › Kommunikation erfolgt über Frontend oder REST

Frontend  
ist das, was  
Sie sehen, die  
Oberfläche



<https://www.frontend-gmbh.de/blog/frontend-und-backend/>

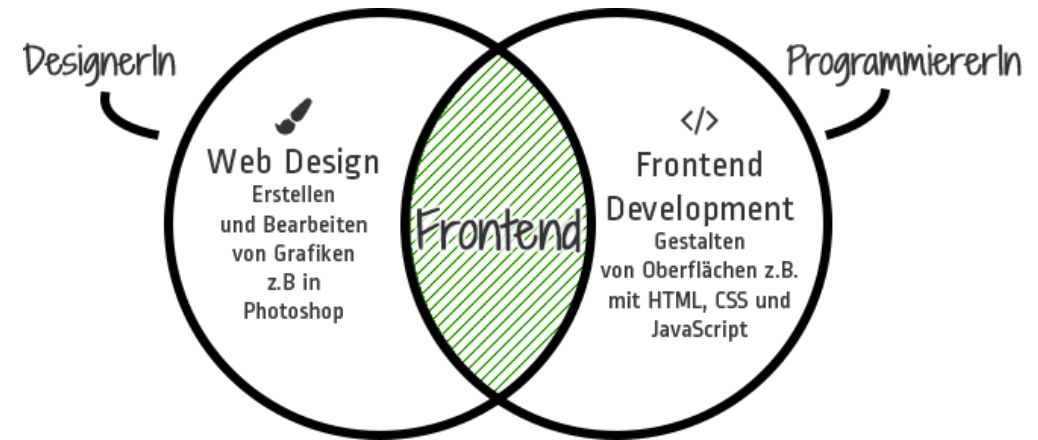


Manuel Richardt, Dominik Rupprecht  
FB **ANGEWANDTE INFORMATIK**



# FRONTEND

- › Präsentationsebene
- › Kann „gesehen“ werden
- › Kommuniziert mit dem Backend
- › Stellt Daten des Backend grafisch dar
- › Nimmt Benutzereingaben entgegen und leitet diese weiter




<https://www.frontend-gmbh.de/blog/frontend-und-backend/>



# SERVER VS CLIENT RENDERING

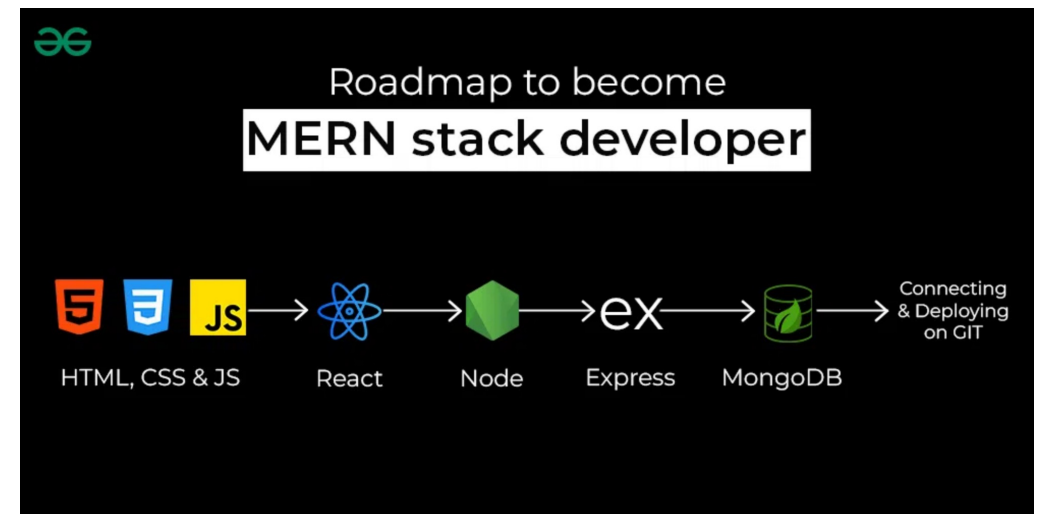
## WER MACHT WAS?

	Server				Browser
					
	Server Rendering	"Static SSR"	SSR with (Re)hydration	CSR with Prerendering	Full CSR
Overview:	An application where input is navigation requests and the output is HTML in response to them.	Built as a Single Page App, but all pages prerendered to static HTML as a build step, and the JS is <b>removed</b> .	Built as a Single Page App. The server prerenders pages, but the full app is also booted on the client.	A Single Page App, where the initial shell/skeleton is prerendered to static HTML at build time.	A Single Page App. All logic, rendering and booting is done on the client. HTML is essentially just script & style tags.
Authoring:	Entirely server-side <small>(request-response, HTML)</small>	Built as if client-side <small>(components, DOM*, fetch)</small>	Built as client-side	Client-side	Client-side
Rendering:	Dynamic HTML	Static HTML	Dynamic HTML <b>and</b> JS/DOM	Partial static HTML, then JS/DOM	Entirely JS/DOM
Server role:	Controls all aspects. <small>(thin client)</small>	Delivers static HTML	Renders pages <small>(navigation requests)</small>	Delivers static HTML	Delivers static HTML
Pros:	👍 TTI = FCP 👍 Fully streaming	👍 Fast TTFB 👍 TTI = FCP 👍 Fully streaming	👍 Flexible	👍 Flexible 👍 Fast TTFB	👍 Flexible 👍 Fast TTFB
Cons:	👎 Slow TTFB 👎 Inflexible	👎 Inflexible 👎 Leads to hydration	👎 Slow TTFB 👎 TTI >>> FCP 👎 Usually buffered	👎 TTI > FCP 👎 Limited streaming	👎 TTI >>> FCP 👎 No streaming
Scales via:	Infra size / cost	build/deploy size	Infra size & JS size	JS size	JS size
Examples:	Gmail HTML, Hacker News	Docusaurus, Netflix*	<a href="#">Next.js</a> , <a href="#">Razzle</a> , etc	Gatsby, Vuepress, etc	Most apps



# MERN-STACK

- › Webentwicklung bezieht sich auf die Erstellung, den Aufbau und die Pflege von Websites. Sie umfasst Aspekte wie Webdesign, Webpublishing, Webprogrammierung und Datenbankmanagement. Einer der bekanntesten Ansätze, der für die Webentwicklung verwendet wird, ist der MERN-Ansatz. Dieser Ansatz bietet einen durchgängigen Rahmen für die Arbeit der Entwickler, und jede dieser Technologien spielt eine große Rolle bei der Entwicklung von Webanwendungen.
- › MERN Stack ist ein JavaScript-Stack, der für die einfachere und schnellere Bereitstellung von Full-Stack-Webanwendungen verwendet wird. MERN Stack besteht aus 4 Technologien, nämlich: **MongoDB, Express, React und Node.js**. Er wurde entwickelt, um den Entwicklungsprozess reibungsloser und einfacher zu gestalten.



<https://www.geeksforgeeks.org/mern-stack/>

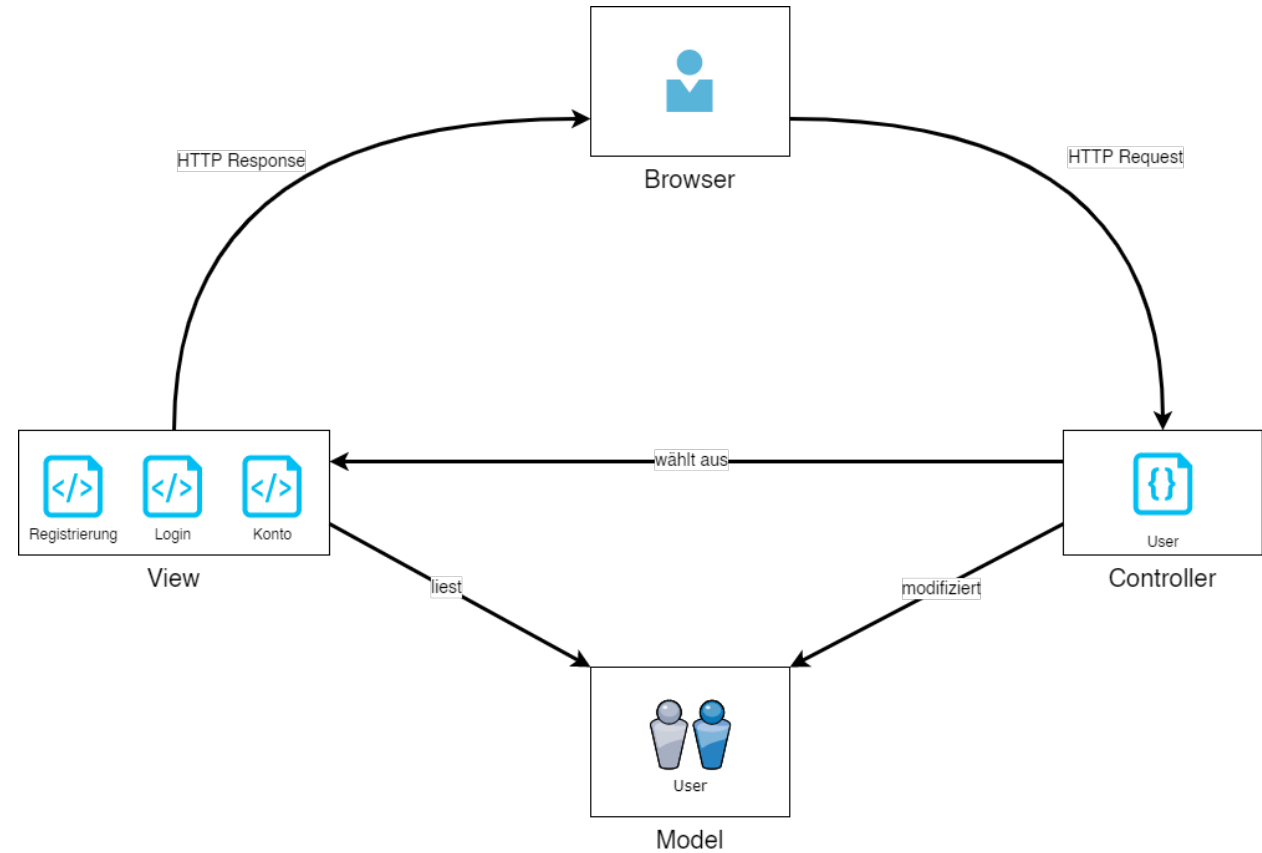


Manuel Richardt, Dominik Rupprecht  
FB **ANGEWANDTE INFORMATIK**



# MVC-ANSATZ

- › Model (Modell)
- › View (Präsentation)
- › Controller (Steuerung)



# WERKZEUGE

## 1. NODE.JS (SERVER)

- <https://nodejs.org/en/>

## 2. NPM (PACKAGE MANAGER)

- <https://www.npmjs.com/>

## 3. EXPRESS (WEB-FRAMEWORK)

- <https://expressjs.com/de/>

## 4. VISUAL STUDIO CODE (PROGRAMMIERUMGEBUNG)

- Erweiterungen: Nodemon, ThunderClient, Bootstrap5, EJS language support, Live-Share



# WEB-APPLIKATIONEN

## EXPRESS



# WEB-PROGRAMMIERUNG

## SCHRITT 0: VORBEREITUNG

- › Auswahl des Namens für die WebApp

In diesem Beispiel myApp

- › Eingabeaufforderung öffnen (Windows)

Windowstaste und cmd eingeben

- › Terminal öffnen (MAC/Linux)

Command+Leertaste -> Terminal suchen



# SCHRITT 1: VERZEICHNIS ERSTELLEN

## ANLEGEN EINES VERZEICHNIS FÜR DIE WEBAPP

› Mit `cd` in das Arbeitsverzeichnis wechseln

z.B. `cd Desktop`

› Mit `mkdir` (make directory) das Anwendungsverzeichnis erstellen

`mkdir myApp`



# **SCHRITT 2: VERZEICHNIS WECHSELN**

## WECHSELN INS VERZEICHNIS DER ANWENDUNG

› In der Eingabeaufforderung das Arbeitsverzeichnis wechseln

z.B. `cd myApp`





# SCHRITT 3: NODE

## TESTEN, OB NODE INSTALLIERT IST

› Die JavaScript Umgebung node ist die Basis für die Entwicklung der serverseitigen Webanwendung

› Überprüfung mit dem Befehl:

```
node --version
```



## SCHRITT 4: NPM

### TESTEN, OB NPM INSTALLIERT IST

› Der node package manager npm installiert Software für node

› Überprüfung mit dem Befehl:

```
npm --version
```



# SCHRITT 5: EXPRESS GENERATOR

## GLOBALE INSTALLATION DES EXPRESS-GENERATOR(S)

- › Mithilfe des express generators wird es möglich, das Grundgerüst einer laufenden Webanwendung zu generieren

```
npm install express-generator -g
```



# SCHRITT 6: AUFRUFPFAD EXPRESS

- › Die Installation erfolgt global und eine Verknüpfung (link) wird erstellt
- › In des HS Fulda üblicherweise unter  
C:\Users\studentai\AppData\Roaming\npm\express
- › Mit `express` startet man die Generierung



# SCHRITT 7: GENERIERUNG STARTEN

## › Im Verzeichnis der Anwendung express starten

In der Eingabeaufforderung muss man im Verzeichnis der Anwendung sein

Mit dem Befehl startet man die Generierung

```
C:\Users\studentai\AppData\Roaming\npm\express --view=ejs
```

Ggf. reicht `express --view=ejs` aus

Die Generierung beginnt und es werden eine Reihe von Verzeichnissen und Dateien erzeugt

## › Alternativ:

```
C:\Users\studentai\AppData\Roaming\npm\express --view=ejs AppName
```

Erzeugt im aktuellen Verzeichnis einen Ordner „AppName“ sowie darin alle notwendigen Verzeichnisse und Dateien



# SCHRITT 8: ABHÄNGIGKEITEN

## ANWENDUNGSABHÄNGIGKEITEN INSTALLIEREN

- › Zum ordnungsgemäßen Funktionieren benötigt die Anwendung weitere Softwarepakete (siehe package.json)

Versionen von Express (4.18.2) und EJS (3.1.9) überprüfen und ggf. ändern

- › Mit npm können diese installiert werden

```
npm install
```

- › Das System lädt die Module herunter und speichert sie lokal in der Anwendung





# SCHRITT 8: ANWENDUNG STARTEN

## ANWENDUNG STARTEN

- › Jetzt steht eine vollständig konfigurierte Webanwendung zur Verfügung. Diese startet man mit dem Befehl

```
npm start
```

- › Öffnen Sie den Browser und laden die Seite

```
localhost:3000
```

- › Die Anwendung zeigt sich mit der bekannten Standardseite



# SCHRITT 10: PORT

## ANPASSUNG DES PORTS

- › Jetzt steht eine vollständig konfigurierte Webanwendung zur Verfügung. Diese startet man mit dem Befehl

```
npm start
```

- › Standardmäßig startet die Anwendung auf Port 3000
- › Port kann in der Datei `bin/www` geändert werden
- › Server muss gestoppt und neu gestartet werden

```
Strg+c (Control+C)
```


```
npm start
```



# PACKAGE.JSON

## VERÄNDERUNGEN GEGENÜBER DEM STANDARD SIND HIER

- › Unter „scripts“ der Parameter „dev“ mit dem Wert „nodemon ./bin/www“
- › „devDependencies“ mit dem Parameter „nodemon“ und dem Wert „^2.0.22“
- › Ersteres ermöglicht es den Server über „npm run dev“ mittels nodemon zu starten.
- › Zweiteres installiert nodemon über npm (npm install nodemon)

```
{  
  "name": "appname",  
  "version": "0.0.0",  
  "private": true,  
   Debug  
  "scripts": {  
    "start": "node ./bin/www",  
    "dev": "nodemon ./bin/www"  
  },  
  "dependencies": {  
    "cookie-parser": "~1.4.4",  
    "debug": "~2.6.9",  
    "ejs": "~3.1.9",  
    "express": "~4.18.2",  
    "http-errors": "~1.6.3",  
    "morgan": "~1.9.1"  
  },  
  "devDependencies": {  
    "nodemon": "^2.0.22"  
  }  
}
```



# ZUSAMMENFASSUNG

- › Web-Applikationen sind verteilte Client-Server Anwendungen
- › Mit node, npm und express lassen sich schnell und einfach Webapplikationen erzeugen
  - Node verwendet JavaScript auf der Serverseite
  - Express generiert die Grundstruktur der Webanwendung und
  - Verwendet eine leistungsstarke Template-Engine

