Softwareentwurf und Anwendungen verteilter Systeme

BA Internet der Dinge – Gestaltung vernetzter Systeme

Semester 3

Hochschule für Gestaltung Schwäbisch Gmünd

Dozent: Yannick Schiele

H f G

Hochschule für Gestaltung
Schwäbisch Gmünd

Arduino

Webservertechnologien

Agenda

- Einführung
- NodeMCU ESP 8266
- Installation
- Programmierung
  - Sensoren
  - Aktoren
  - Webserver
- REST-APIs

Recherche zu Sensor & Aktor

Abgabe an yannick.schiele@hfg.design

Projekt Teams + Sensor & Aktor

- Florian Kiem & Anton Stallbörger
  - Sensor: ICQUANZX Pulsmesser Pulsmesser Modul DIY Für Arduino
  - Aktor: AZDelivery 5 x AZDelivery Hochwertig Vibrationsmotor
  - Eventuell weiterer Aktor: AZDelivery 1,3 Zoll OLED Display I2C SSH1106 Chip

- Jannis Schuler & Alice Sopp
  - Aktor: Servomotor
  - Sensor: Zahlen Pad und evtl Fingerabdrucksensor oder Piezo Sensor

- Michael Kluge & Nils Eller
  - Sensor: Luftqualitätssensor, Temperatursensor und oder Feuchtigkeitssensor
  - Aktor: evtl. einen Motor, oder eine LED als Anzeigeelement

- Christian Licu, Lena Dethloff & Marina Beck
  - Aktor: 2x Steppermotor
  - Sensor: 2x Joystick

- Jakob Finkenzeller & Simon Feldmann
  - Aktor: ESP-32-Wrover-E-Camera
  - Sensor: ESP-32-Wrover-E-Camera

```
void setup(){
 pinMode(LED_BUILTIN, OUTPUT);
 // Initialize the LED_BUILTIN pin as an output
}

// the loop function runs over and over again
 void loop(){
 digitalWrite(LED_BUILTIN, LOW);
 // Turn the LED on (LOW is the voltage level)
 // the LED is on;
 // because it is active on low
 delay(1000);
 // Wait for a second
 digitalWrite(LED_BUILTIN, HIGH);
 // Turn the LED off by making the voltage HIGH
 delay(2000);
 // Wait for two seconds
}
```

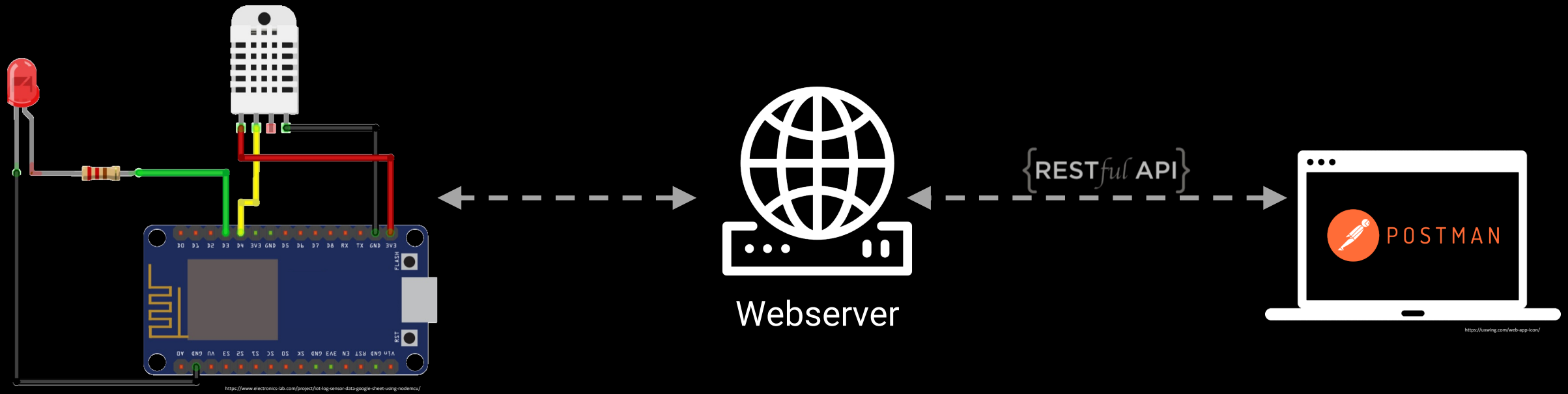
Wiederholung

Skript zum Blinken der eingebauten LED des ESP 8266

Übung (10 min)

1. Projekt

- Arduino mit Sensor, Aktor und eigenem Webserver
  - Sensor Daten werden an Webserver gesendet
  - Webserver wird vom Arduino selbst gehostet und ist über das lokale Netz erreichbar
  - Über Rest APIs werden Sensor Daten abgerufen und der Aktor gesteuert



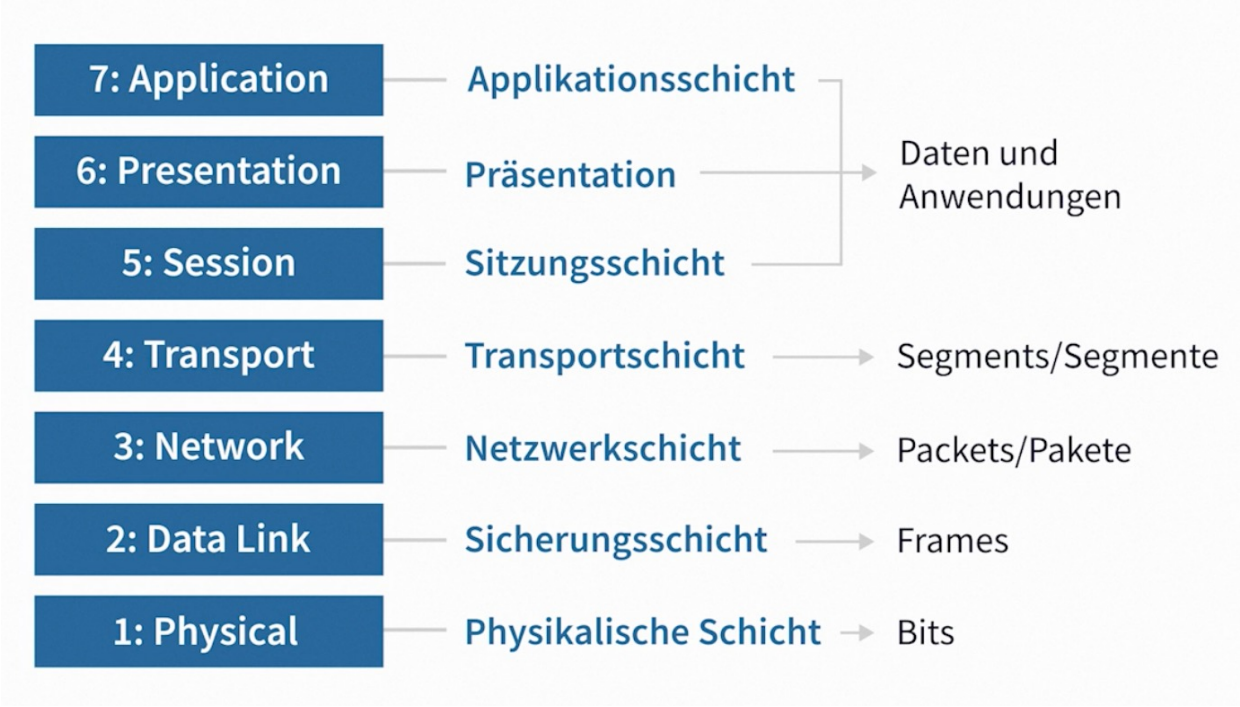
Schichtenmodelle der Netzwerktechnik

OSI- und TCP/IP Schichtenmodell

https://www.linkedin.com/learning/netzwerkgrundlagen-1-die-theorie-fur-die-praxis/das-osi-modell?autoplay=true&u=82266650

OSI-Referenzmodell

- Open System Interconnection (Offenes System für Kommunikationsverbindungen)
- beschreibt die herstellerunabhängige Netzwerkkommunikation über verschiedene technische Systeme
- Sieben Schichten Modell
- Jeder Schicht sind unterschiedliche Netzwerkstandards zuzuordnen

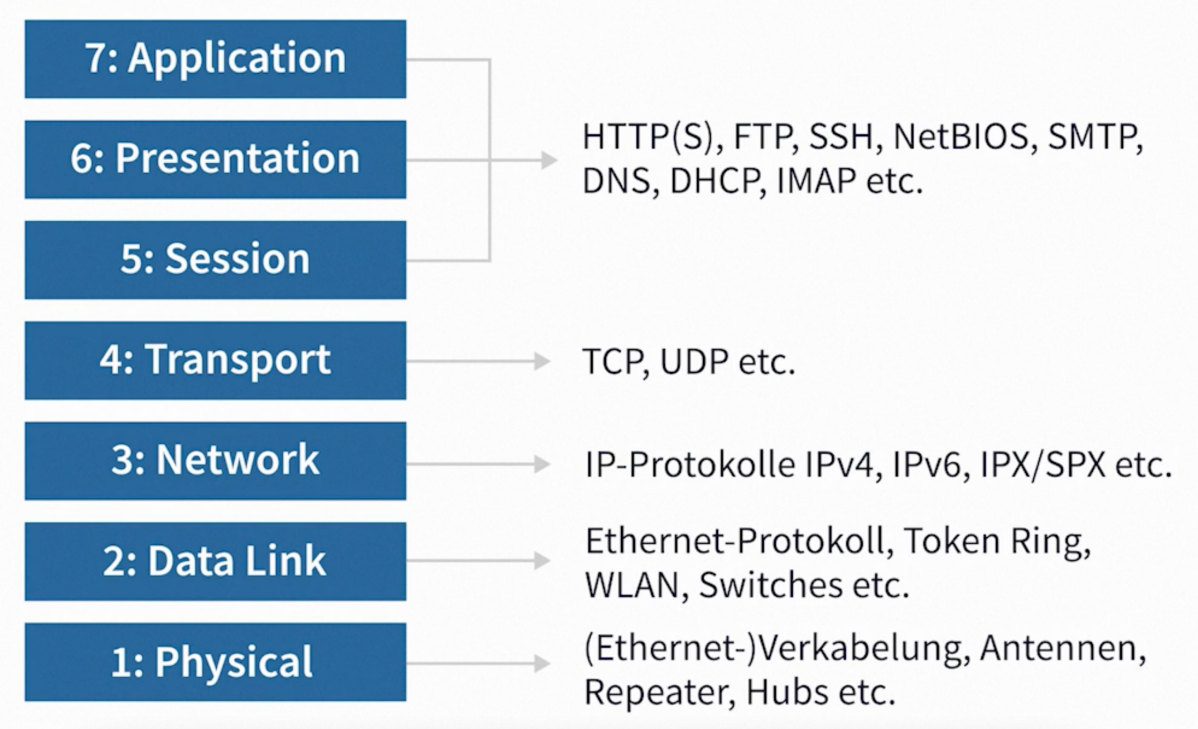


https://www.linkedin.com/learning/netzwerkgrundlagen-1-die-theorie-fur-die-praxis/das-osi-modell?autoplay=true&u=82266650

OSI-Referenzmodell

- Weitere Unterteilung:
  - anwendungsorientierten Schichten 5 bis 7
  - transportorientierten Schichten von 1 bis 4

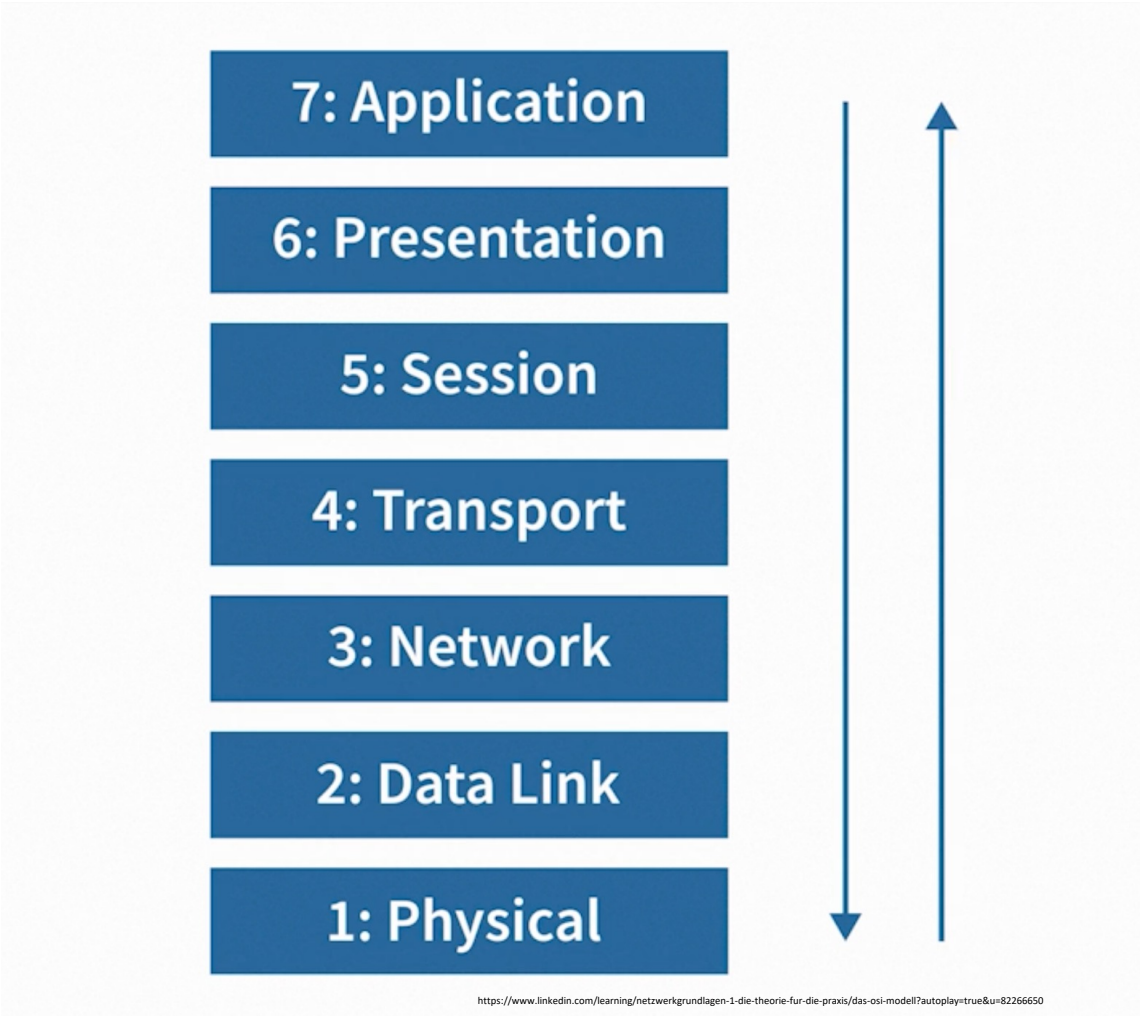
- Die 7 Schichten sind voneinander in ihrer Funktion abgegrenzt
- Dadurch kann jede Schicht einen anderen Standard verwenden, ohne die anderen Schichten zu beeinflussen



https://www.linkedin.com/learning/netzwerkgrundlagen-1-die-theorie-fur-die-praxis/das-osi-modell?autoplay=true&u=82266650

OSI-Referenzmodell

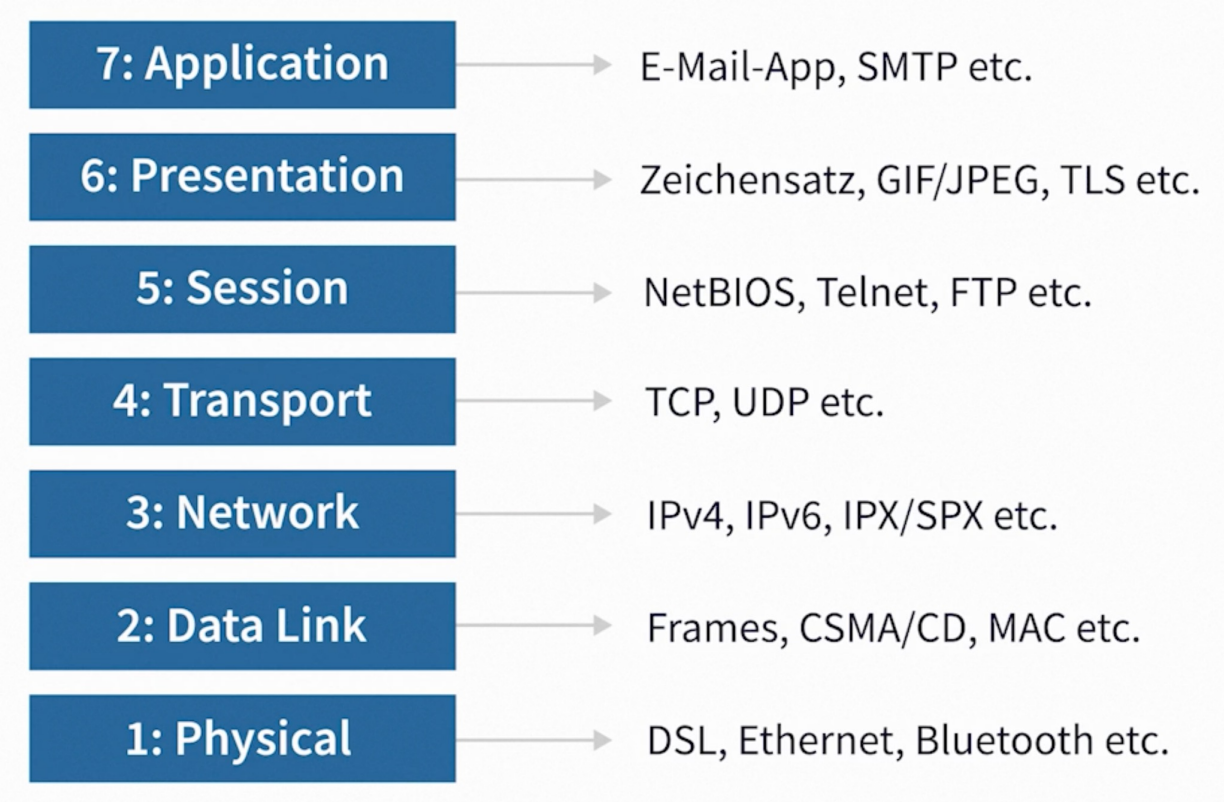
1. Schicht Bitübertragungsschicht / Physical Layer
   Umwandlung der Bits in ein zum Medium passendes Signal und physikalische Übertragung.
2. Schicht Sicherungsschicht / Data Link Layer
   Segmentierung der Pakete in Frames und Hinzufügen von Prüfsummen.
3. Schicht Vermittlungsschicht / Network Layer
   Routing der Datenpakete zum nächsten Knoten.
4. Schicht Transportschicht / Transport Layer
   Zuordnung der Datenpakete zu einer Anwendung.
5. Schicht Kommunikationsschicht / Session Layer
   Steuerung der Verbindungen und des Datenaustauschs.
6. Schicht Darstellungsschicht / Presentation Layer
   Umwandlung der systemabhängigen Daten in ein unabhängiges Format
7. Schicht Anwendungsschicht / Application Layer
   Funktionen für Anwendungen, sowie die Dateneingabe und -ausgabe.



https://www.linkedin.com/learning/netzwerkgrundlagen-1-die-theorie-fur-die-praxis/das-osi-modell?autoplay=true&u=82266650

OSI-Referenzmodell

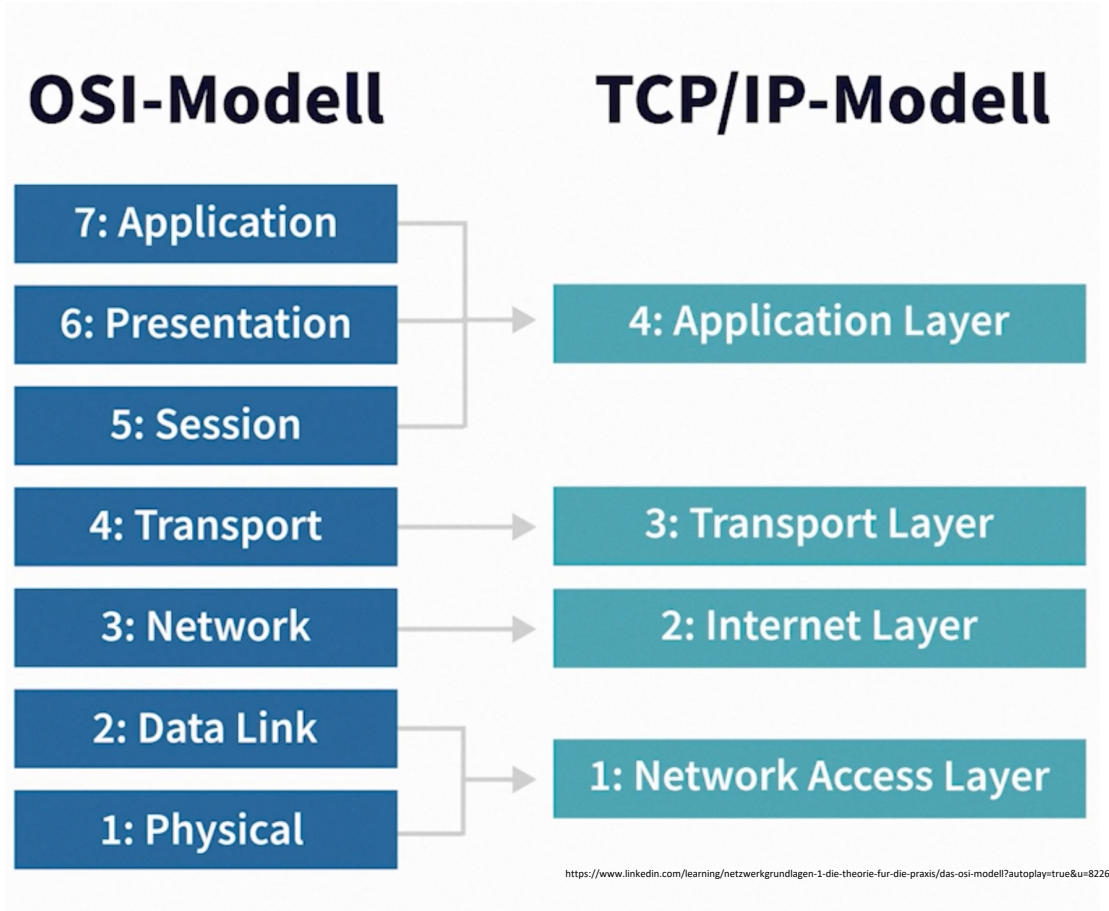
- Die Kommunikation verläuft vom Absender von der obersten Applikationsschicht herunter bis zu Schicht 1
- Beim Empfänger ist es genau umgekehrt. Der empfängt die Daten auf der untersten Schicht und die nehmen ihren Weg bis in die oberste Schicht 7



https://www.linkedin.com/learning/netzwerkgrundlagen-1-die-theorie-fur-die-praxis/das-osi-modell?autoplay=true&u=82266650

OSI-Referenzmodell

Die verwendeten Protokolle der einzelnen Schichten beim Versenden einer E-Mail per Mail Client wie bspw. Outlook

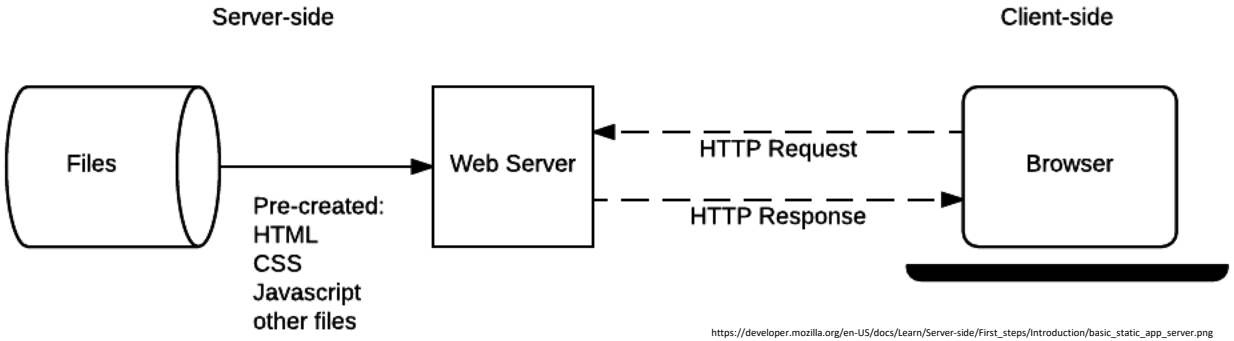


https://www.linkedin.com/learning/netzwerkgrundlagen-1-die-theorie-fur-die-praxis/das-osi-modell?autoplay=true&u=82266650

TCP IP Referenzmodell

- Dieses Modell ist kein abstraktes und neutrales, wie das OSI-Modell, sondern bezieht sich explizit auf die TCP/IP- Protokolle
- Gegenüber dem OSI-Modell definiert das TCP/IP-Modell statt 7 nur 4 Schichten

Webserver Technologien



https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction/basic_static_app_server.png

Webserver

- Als Webserver bezeichnet man jene Server, die zur Verbreitung von Webinhalten im Inter- oder Intranet dienen
- Als Teil eines Rechnernetzwerks übertragen sie Dokumente an sogenannte Clients– beispielsweise eine Webseite an einen Webbrowser
- Für die Übermittlung wird das Übertragungsprotokoll HTTP (OSI Layer 6) genutzt das auf den Netzwerkprotokollen IP und TCP beruht

https://www.ionos.de/digitalguide/server/knowhow/webserver-definition-hintergruende-software-tipps/

```
const http = require('http‘);
const hostname = 'localhost‘;
const port = 8080;

const server = http.createServer((req, res) => {
  console.log(req.headers);
  res.statusCode = 200;
  res.end('<html><body><h1>Hello, World!</h1></body></html>‘);
 })

server.listen(port, hostname);
```

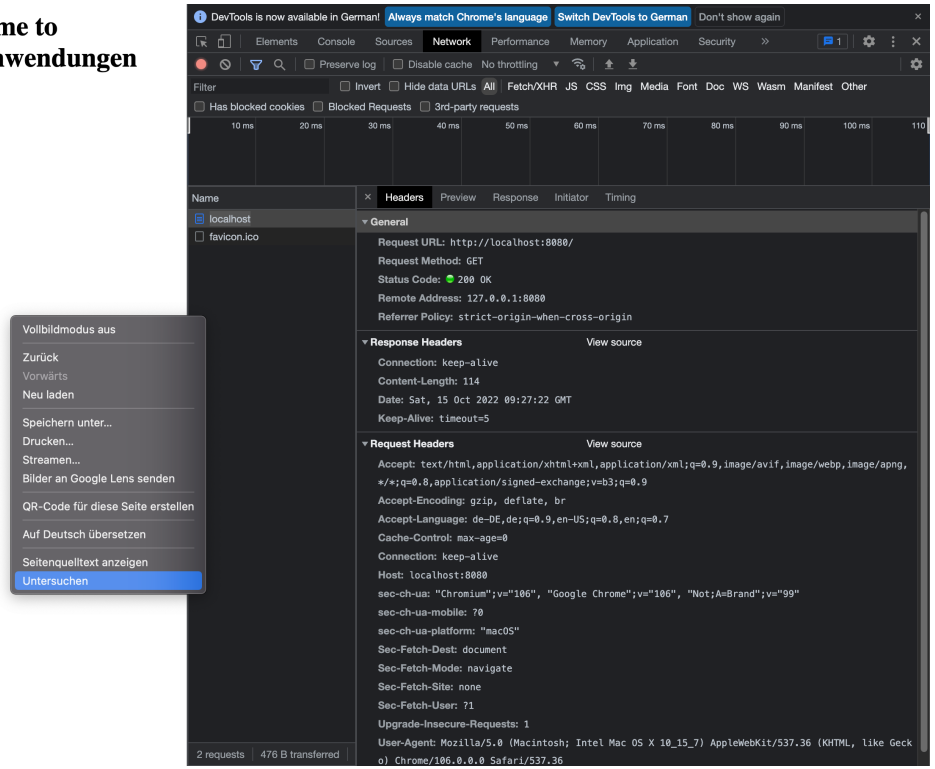
HTTP-Webserver in Node.js

Praktische Zwischenübung (15 min):

- Installation von Node.js & npm notwendig
- Erstellen einer Node.js Datei mit Localhost Webserver Code
- Dann im Terminal folgende Befehle ausführen:

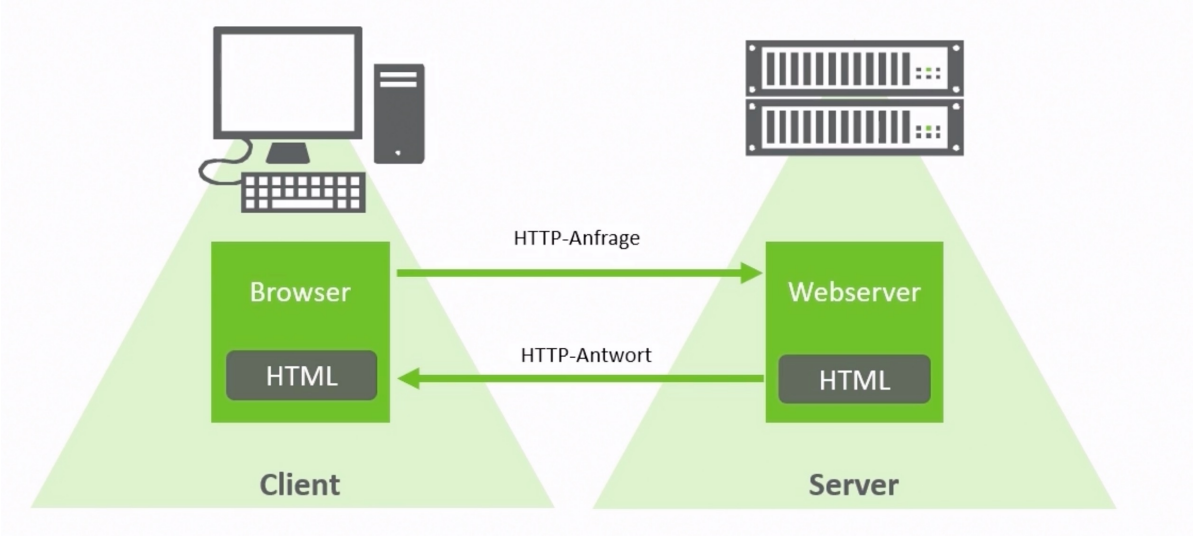
```
node {yourfilename.js}
curl localhost:8080
```

Hello, World and Welcome to Softwareentwurf und Anwendungen verteilter Systeme!



HTTP-Webserver in Node.js

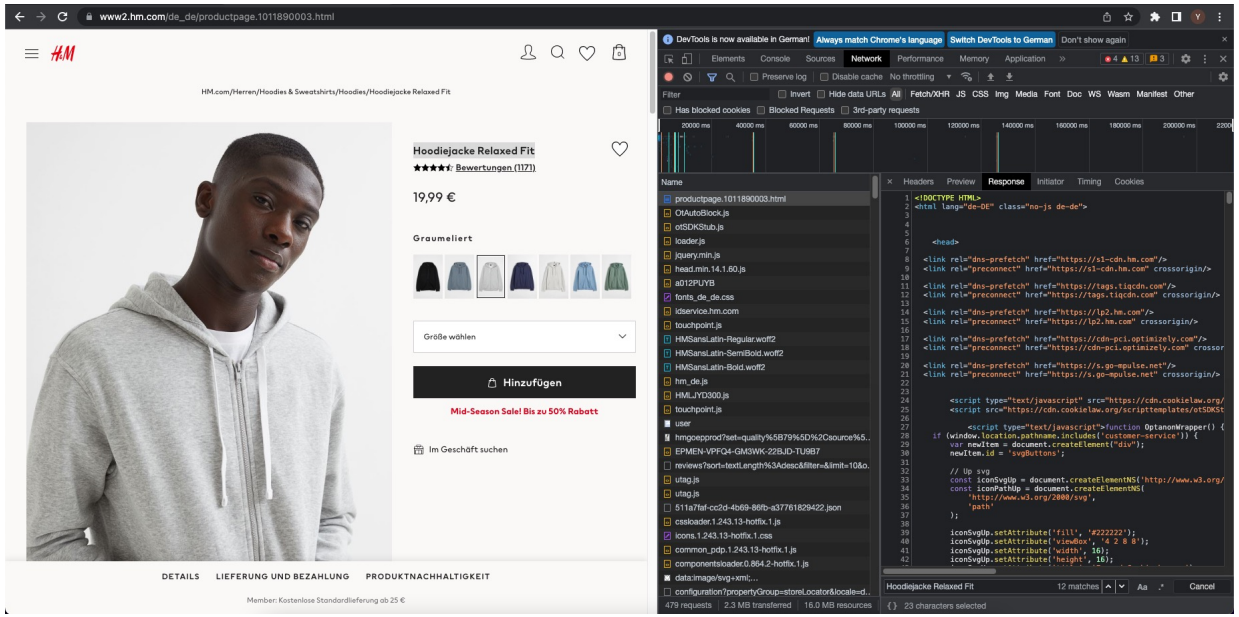
Google Chrome Developer Tools zeigen HTT GET Request und HTTP Response mit Statuscode 200



https://www.linkedin.com/learning/search?keywords=client%20server&language=de_DE&u=82266650

Client - Server - Architektur

- Herzstück der Webtechnologie
- Man unterscheidet in „clientseitige“ und “serverseitige“ Technologien
- Client = Browser des Anwenders
- clientseitige Software:
  - Sendet Anfrage an Server
  - Kümmert sich um Darstellung der HTML Seite
- Serverseitige Software:
  - Nimmt Anfrage entgegen
  - Packt Inhalt in Response und schickt sie an den Client

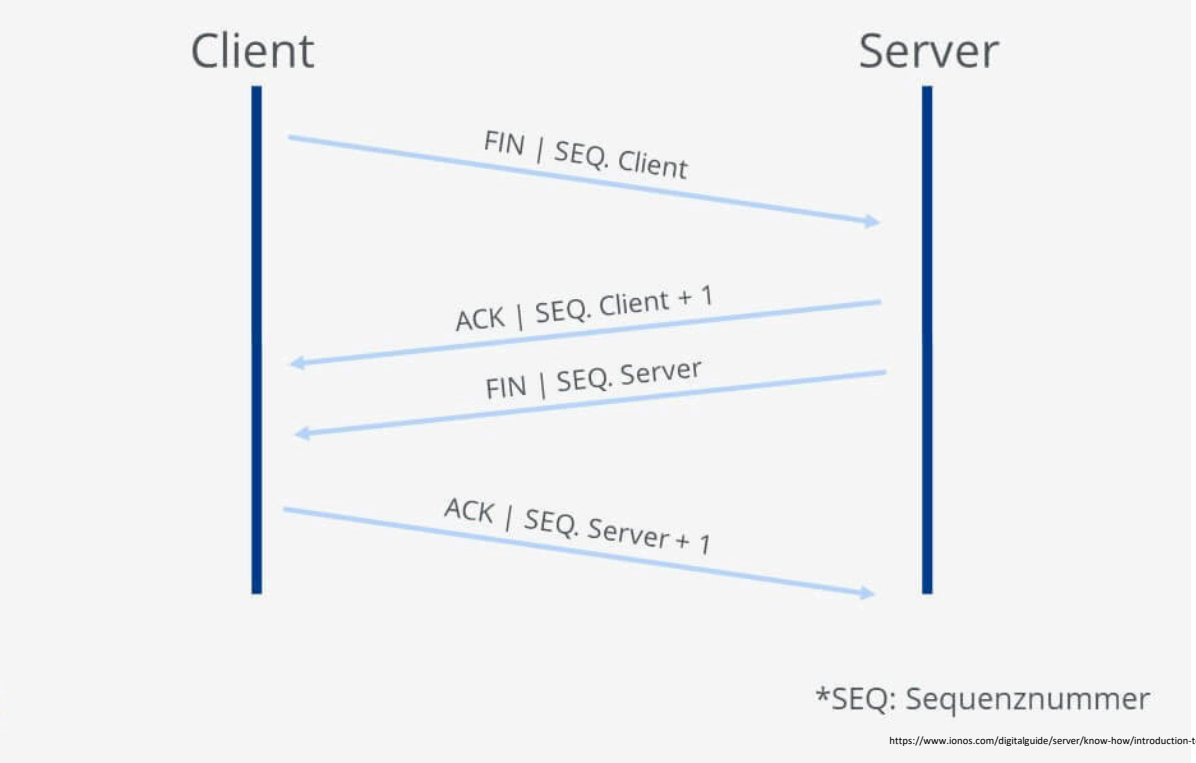


Client - Server

- Praktisches Beispiel (10 min): eigene Netzwerkanalyse einer Shop-Website (bspw. H&M)
- Hier sieht man, welche Kommunikation zwischen Client, also Browser und Server erfolgt
- Sehr viele Dateien werden durch HTTP Anfragen und Antworten ausgetauscht
  - Jedes einzelne Bild und Stylesheet
- Zentrale HTML Datei enthält Grundsatzinformationen über die Website
  - GET https://www2.hm.com/de_de/productpage.1011890003.html

Netzwerkprotokolle

TCP, HTTP, MQTT, etc.

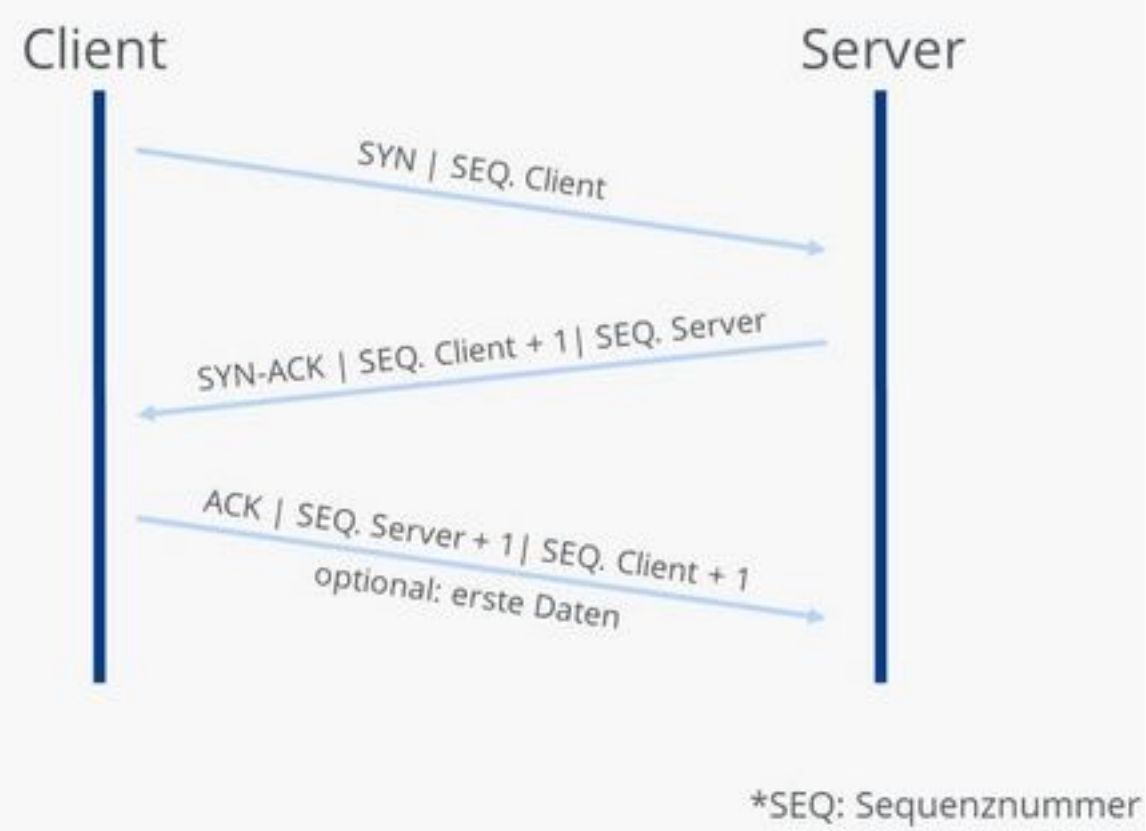


https://www.ionos.com/digitalguide/server/know-how/introduction-to-tcp/

TCP

- Das Transmission Control Protocol, ist eine standardisierte Vereinbarung zur Datenübertragung zwischen verschiedenen Teilnehmern eines Computernetzwerks
- TCP erlaubt es zwei Endpunkten in einem gemeinsamen Computernetz, eine Verbindung aufzubauen, die eine beidseitige Datenübertragung erlaubt
- Dabei werden jegliche Datenverluste erkannt und automatisch behoben, weshalb man es auch als zuverlässiges Protokoll bezeichnet
- Transportschicht Layer 4 OSI Modell

https://www.ionos.com/digitalguide/server/know-how/introduction-to-tcp/

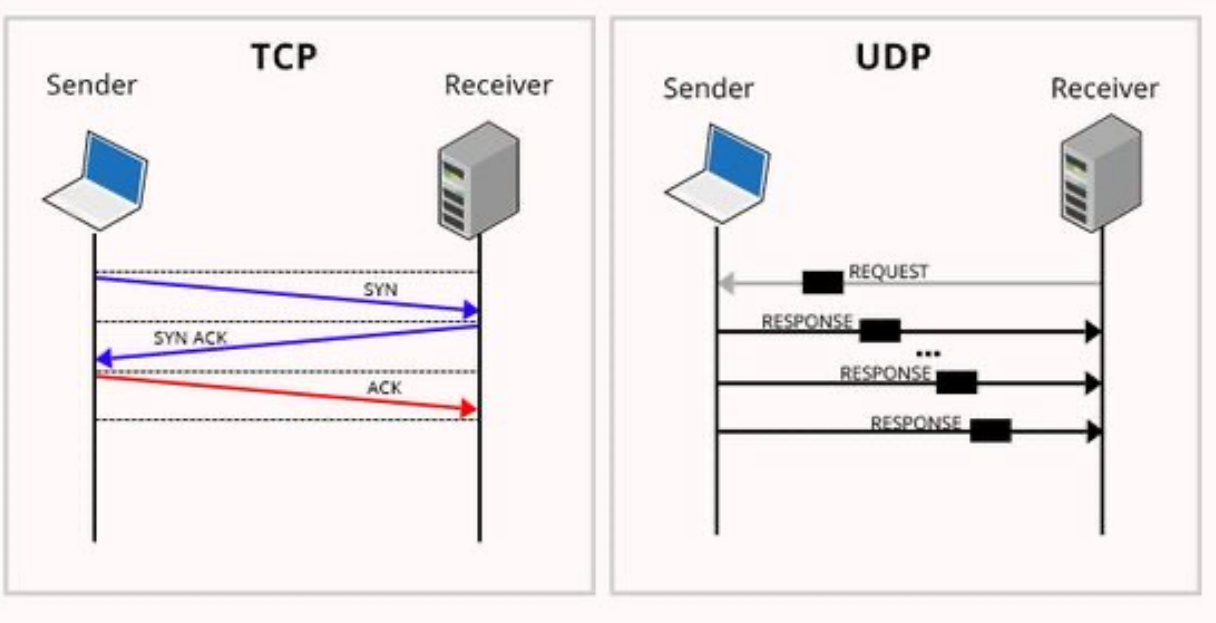


https://www.ionos.com/digitalguide/server/know-how/introduction-to-tcp/

TCP Verbindungen

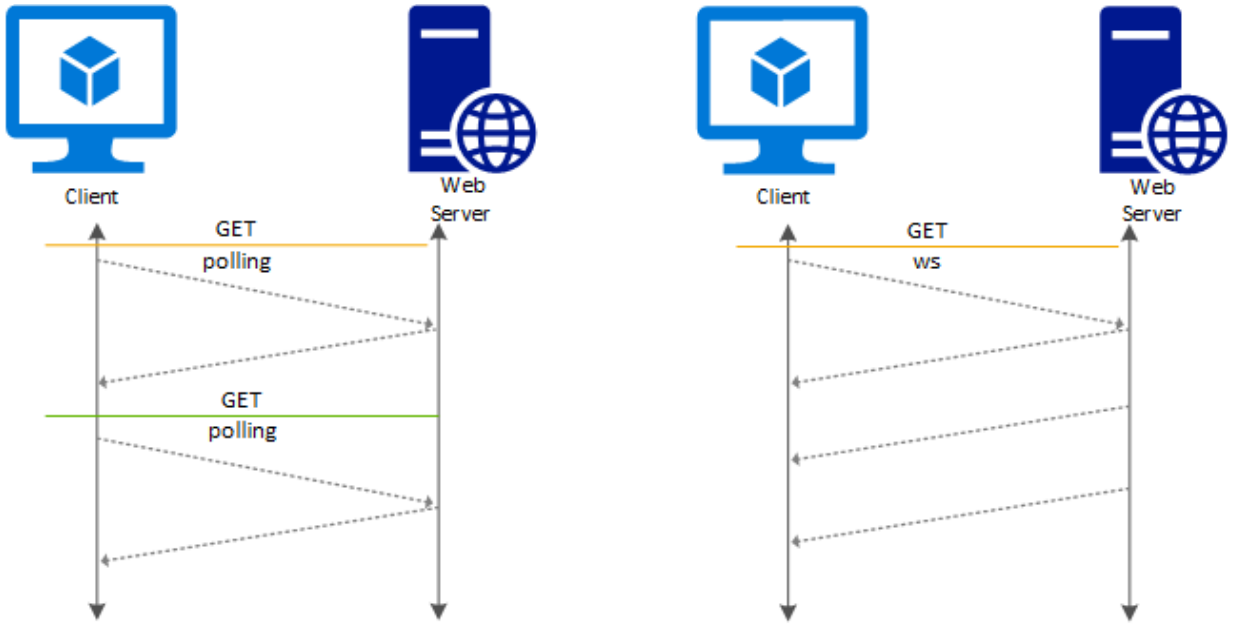
- Übertragung von Informationen in beide Richtungen möglich (vgl. Telefon-Gespräch)
- Ende-zu-Ende-Verbindungen
- jede Verbindung muss immer durch zwei klar definierte Endpunkte (Client und Server) identifiziert werden
- TCP-Verbindungsaufbau erfolgt über den „Drei-Wege-Handshake“
  - SYN-Paket: zufällige Sequenznummer
  - SYN-ACK-Paket: um 1 erhöhten Sequenznummer + eigene Sequenznummer
  - ACK-Paket: um 1 erhöhte Sequenznummer des Servers

https://www.ionos.com/digitalguide/server/know-how/introduction-to-tcp/



UDP

- User Datagram Protocol, ermöglicht den verbindungslosen Versand von Datagrammen in IP-basierten Netzwerken
- Unterschied zu TCP:
  - eine Anwendung kann sehr schnell Informationen versenden, da weder eine Verbindung zum Adressaten aufgebaut noch eine Antwort abgewartet wird
  - Es gibt allerdings keinerlei Garantie dafür, dass Pakete vollständig und in der gleichen Reihenfolge, in der sie gesendet wurden, ankommen.
- Transportschicht Layer 4 OSI Modell



https://learn.microsoft.com/de-de/azure/application-gateway/media/application-gateway-websocket/websocket.png

Websockets

- Das WebSocket-Protokoll basiert auf TCP
- Um eine WebSocket-Verbindung einzurichten, wird ein spezifischer HTTP-basierter Handshake zwischen dem Client und dem Server ausgetauscht
- Im Erfolgsfall wird ein „Upgrade“ des Protokolls der Anwendungsschicht von HTTP auf WebSockets ausgeführt.
- Sobald dies erfolgt ist, spielt HTTP keinerlei Rolle mehr
- Daten können von beiden Endpunkten gesendet oder empfangen werden, bis die Verbindung geschlossen wird
- kommt immer rann zum Einsatz, wenn schnellen Aufbau von Verbindungen benötigt wird (bspw. Live-Chat)

```
// Importing the required modules
const WebSocketServer = require('ws‘);

// Creating a new websocket server
const wss = new WebSocketServer.Server({ port: 8080 })

// Creating connection using websocket
wss.on("connection", ws => {
  console.log("new client connected");

  // sending message
  ws.on("message", data => {
          console.log(`Client has sent us: ${data}`) });

   // handling what to do when clients disconnects from server
  ws.on("close", () => {
          console.log("the client has connected"); });

  //handling client connection error
  ws.onerror = function () {
          console.log("Some Error occurred")
  }
});

console.log("The WebSocket server is running on port 8080");
```

https://www.piesocket.com/blog/nodejs-websocket

Websocket Tutorial Node.js

Websocket Server Code in Node.js zur Übung (15 min)

Zum ausführen:

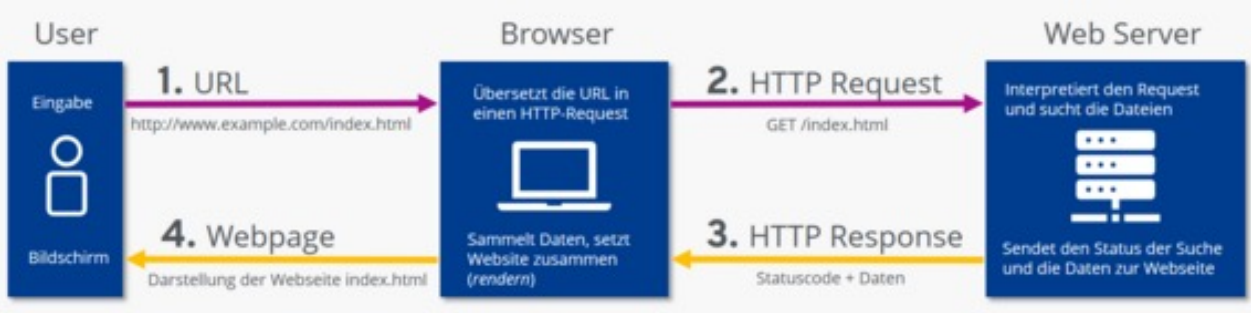
```
npm install ws
node main.js
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>NodeJS WebSocket Server</title>
</head>
<body>
<h1>Hello world</h1>
<script>
const ws = new WebSocket("ws://localhost:8080");
ws.addEventListener("open", () =>{
console.log("We are connected");
ws.send("How are you?");
});
ws.addEventListener('message', function (event) {
console.log(event.data);
});
</script>
</body>
</html>
```

Websocket Tutorial Node.js

- Client HTML Code

- Auch für den ESP8266/32 verfügbar:
  - https://www.arduino.cc/reference/en/libraries/websockets/

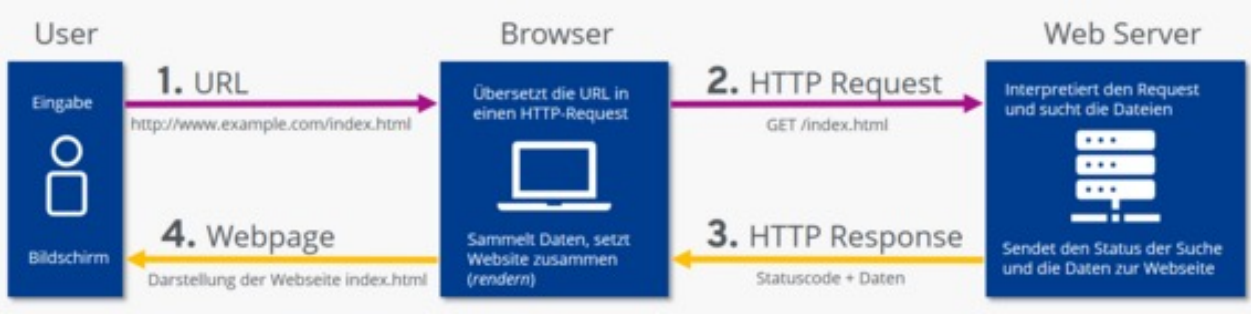


https://www.ionos.de/digitalguide/hosting/hosting-technik/was-ist-http/

HTTP

Das „Hypertext Transfer Protocol“ von Tim Berners-Lee zusammen mit anderen Konzepten als Grundlage für das World Wibe Web entwickelt

HTTP regelt, wie Ressourcen vom Server zum Client übertragen werden



https://www.ionos.de/digitalguide/hosting/hosting-technik/was-ist-http/

Funktionsweise HTTP

1. Der Nutzer tippt in die Adresszeile seines Internet-Browsers example.com ein.
2. Der Browser sendet den HTTP-Request, an den zuständigen Webserver, der die Domäne example.com verwaltet.
3. Der Webserver empfängt den HTTP-Request, sucht die gewünschte Datei und sendet als Erstes den Header, der dem anfragenden Client durch einen Status-Code das Resultat seiner Suche mitteilt. Wenn die Datei gefunden wurde und der Client sie tatsächlich zugesendet haben will, sendet der Server nach dem Header den Message Body, also den eigentlichen Inhalt. In unserem Beispiel ist dies die Datei index.html.
4. Der Browser empfängt die Datei und stellt sie als Webseite dar.

https://miro.medium.com/max/920/1*w_iicbG7L3xEQTArjHUS6g.jpeg

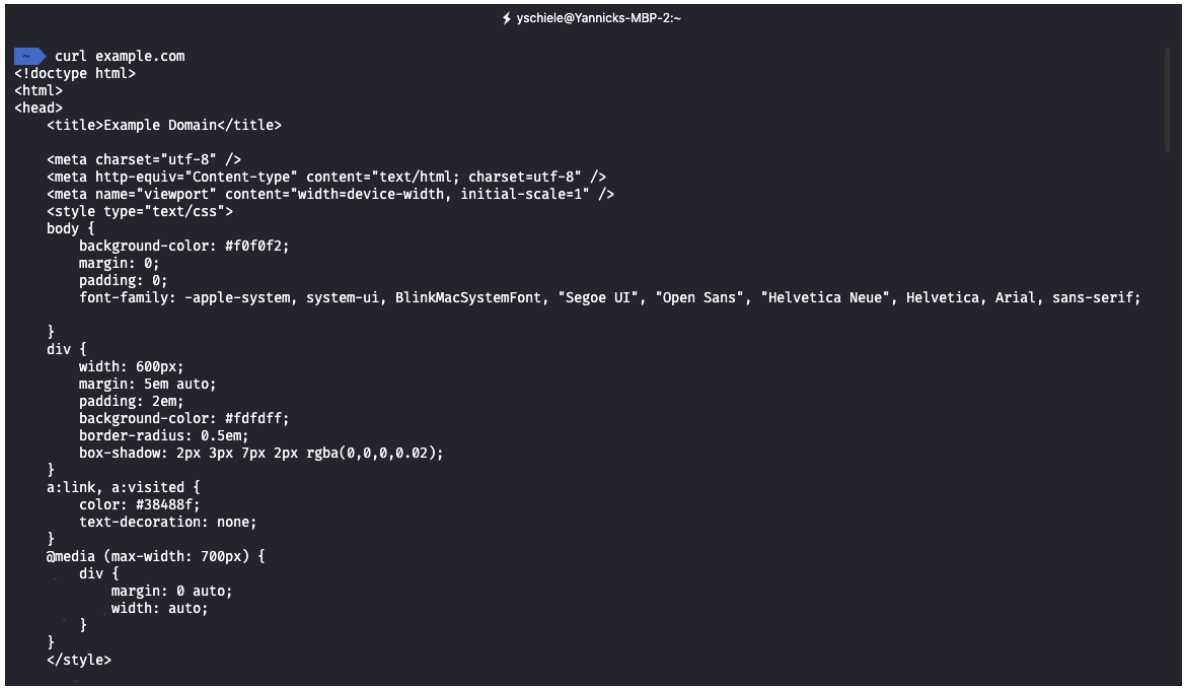
HTTP Status Codes

- Anfragen des Clients werden mit HTTP-Status-Codes in Form einer dreistelligen Zahlenfolge beantwortet
- Durch diese Meldung teilt der Webserver dem Browser mit, ob eine Anfrage erfolgreich bearbeitet wurde, ein Fehler vorliegt oder eine Authentifizierung erforderlich ist
- Der HTTP-Status-Code ist somit ein essenzieller Teil der vom Webserver übermittelten Rückmeldung
- Er wird vom Server automatisch in den Header einer jeden HTTP-Antwort eingefügt

https://miro.medium.com/max/920/1*w_iicbG7L3xEQTArjHUS6g.jpeg

HTTP Status Codes

- Klasse 1xx – Informativ:
  - Hier meldet der Server dem Client, dass die aktuelle Anfrage noch andauert. Diese Klasse fasst Codes zusammen, die Informationen zur Bearbeitung liefern und während der Anfrage gesendet werden.
- Klasse 2xx – Erfolg:
  - Ein 2xx-Code meldet eine erfolgreiche Operation. Dementsprechend werden 2xx-Codes vom Server oft gleichzeitig mit den gewünschten Webseiten-Daten versendet.
- Klasse 3xx – Umleitung:
  - Ein 3xx-Code zeigt an, dass die Anfrage vom Server empfangen wurde. Um eine erfolgreiche Bearbeitung sicherzustellen, sind jedoch weitere Schritte seitens des Clients erforderlich. 3xx-Codes treten vor allem bei Um- und Weiterleitungen auf.
- Klasse 4xx – Client-Fehler:
  - Wird ein 4xx-Code ausgespielt, liegt ein Client-Fehler vor. Der Server hat die Anfrage erhalten, kann diese jedoch nicht ausführen. Der Grund dafür ist in der Regel eine fehlerhafte Anfrage.
- Klasse 5xx – Server-Fehler:
  - Mit einem 5xx-Code weist der Server auf einen Fehler hin, der im Verantwortungsbereich des Servers zu verorten ist. Solche Server-Fehlercodes melden, dass die entsprechende Anfrage vorübergehend nicht ausführbar oder gar unmöglich ist.

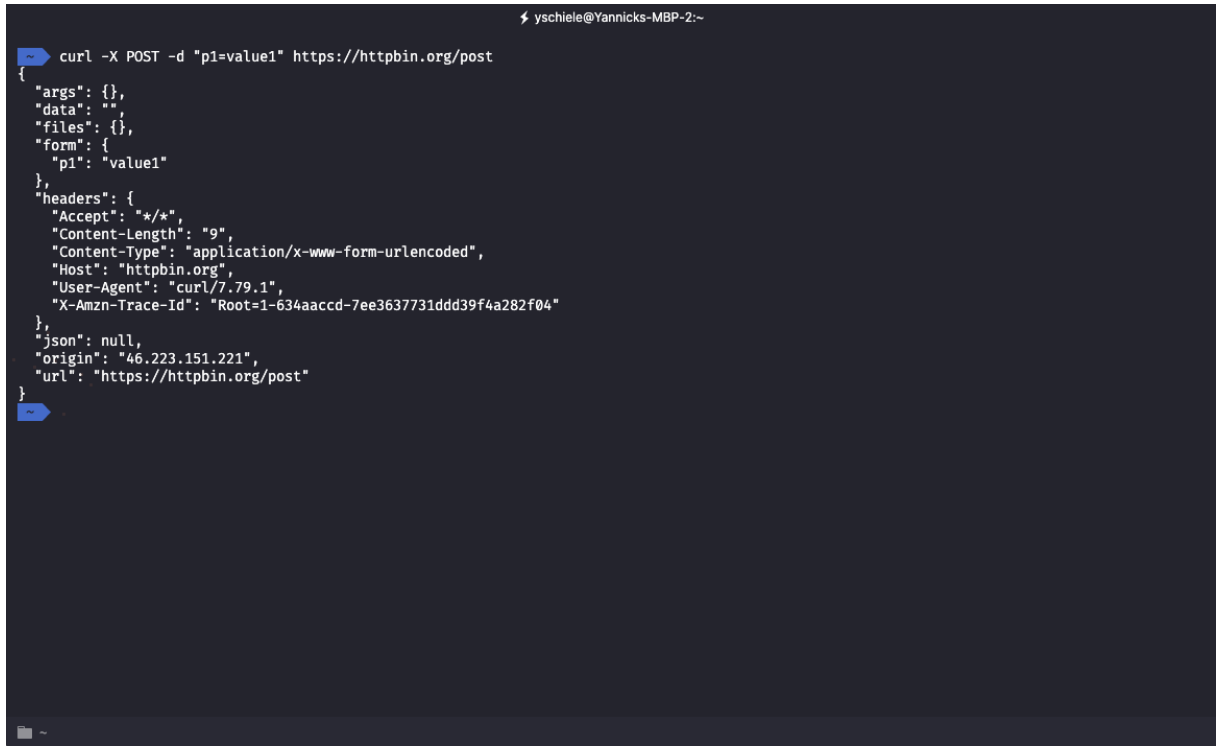


HTTP – curl Tutorial

- `curl example.com`
- `curl -i example.com`
  - Gibt den Header Output mit an
- `curl -I example.com`
  - Gibt nur den Header Output zurück
- `curl -v example.com`
  - Gibt die Kommunikation zwischen cURL und dem Server zurück

https://bytexd.com/basics-http-requests-curl-tutorial/

15 Minuten



HTTP – curl Tutorial

POST Befehl:

```
curl -X POST -d "p1=value1"
```

Mit diesem Befehl wird der Server aufgefordert, die Daten p1=value1 zu übermitteln.

Bei p1 kann es sich um einen beliebigen Parameter mit einem beliebigen Wert handeln. Die URI wird hier nur als Beispiel verwendet.

Fragen?

Projektarbeit