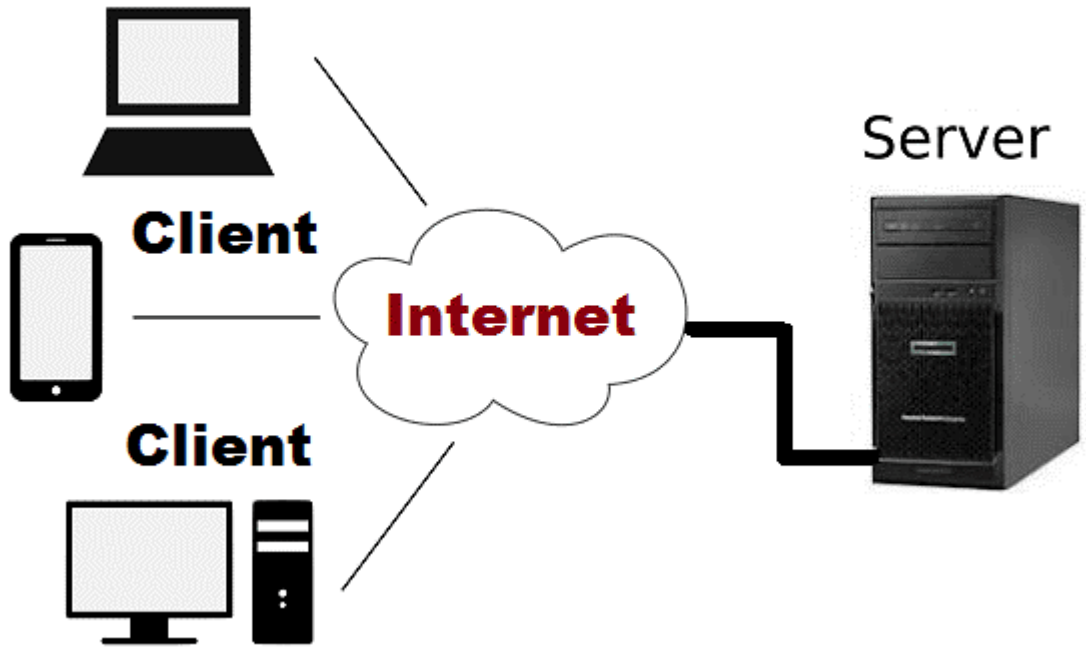
LU03a - POST & GET

Einleitung

In modernen Webanwendungen spielt die Kommunikation zwischen Client und Server eine zentrale Rolle. Dabei kommen standardisierte HTTP-Methoden zum Einsatz, von denen GET und POST zu den wichtigsten gehören. Sie definieren, wie Daten zwischen einem Client – etwa einem Webbrowser mit JavaScript – und einem Server ausgetauscht werden. Ohne diese Mechanismen wäre das Internet im Grunde nur eine statische Ansammlung von Dokumenten.

1. Grundidee: Client und Server

Nachfolgend sehen Sie das Client-Servermodell, die zwei Bestandteile, auf die Informationsaustausch im www möglich macht.



In einer CS-Umgebung bedeutet das meistens:

- Client = Browser oder Frontend mit JavaScript
- Server = Webserver / API, die Anfragen entgegennimmt und Antworten zurückgibt

Wenn JavaScript mit einem Server kommuniziert, passiert das oft per fetch()-Methode.

Theorie: HTTP-Methoden GET & POST

In der Webentwicklung ist das Hypertext Transfer Protocol (HTTP) die Sprache, in der Client (Browser) und Server miteinander kommunizieren. Die beiden wichtigsten Befehle („Verben“) dieser Sprache

sind GET und POST.

Das nachfolgende Youtube-Kurzvideo (https://www.youtube.com/shorts/7x2xhOffOdQ) erklärt POST und GET einfach und verständlich.

GET-POST-Kurzvideo: https://www.youtube.com/shorts/7x2xhOffOdQ

1. GET: Daten abrufen

Die GET-Methode wird verwendet, um Informationen vom Server anzufordern. Sie ist vergleichbar mit dem Aufschlagen einer Seite in einem Lexikon.

- **Sichtbarkeit:** Parameter werden direkt an die URL angehängt (z.B. search?term=javascript).
- **Lesezeichen:** Da die Daten in der URL stehen, können GET-Anfragen als Lesezeichen gespeichert werden.
- **Caching:** Browser speichern GET-Antworten oft im Zwischenspeicher, um sie beim nächsten Mal schneller zu laden.
- **Sicherheit: Nicht** für sensible Daten (Passwörter) geeignet, da die Parameter im Browserverlauf und in Server-Logs auftauchen.

```
// Eine einfache Abfrage von Daten
fetch('https://api.beispiel.de/users')
  .then(response => response.json())
  .then(data => console.log(data));
```

2. POST: Daten senden

Die POST-Methode wird verwendet, um Daten an den Server zu senden, damit dieser sie verarbeitet (z.B. in eine Datenbank schreibt). Dies ist vergleichbar mit dem Einwerfen eines ausgefüllten Formulars in einen Briefkasten.

- **Sichtbarkeit:** Die Daten werden im Body (Körper) der HTTP-Anfrage übertragen. Sie sind nicht in der URL sichtbar.
- **Kapazität:** POST hat keine strikte Längenbeschränkung (ideal für Datei-Uploads oder lange Texte).
- **Sicherheit:** Sicherer als GET für Logins, da Daten nicht im Verlauf erscheinen. Dennoch ist für echte Sicherheit eine HTTPS-Verschlüsselung zwingend erforderlich.

```
// im Gegensatz zu GET, werden hier die Daten nicht in der URL, sondern im
Body transportiert.
fetch('https://api.beispiel.de/users', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ name: "Max Mustermann", beruf: "Informatiker" })
});
```

3. Direkter Vergleich

Merkmal	GET	POST
Hauptzweck	Daten abrufen	Daten übermitteln/erstellen
Datenposition	URL (Query String)	Request Body
Sichtbarkeit	Für jeden in der Adresszeile sichtbar	In der Adresszeile unsichtbar
Lesezeichen	Ja	Nein