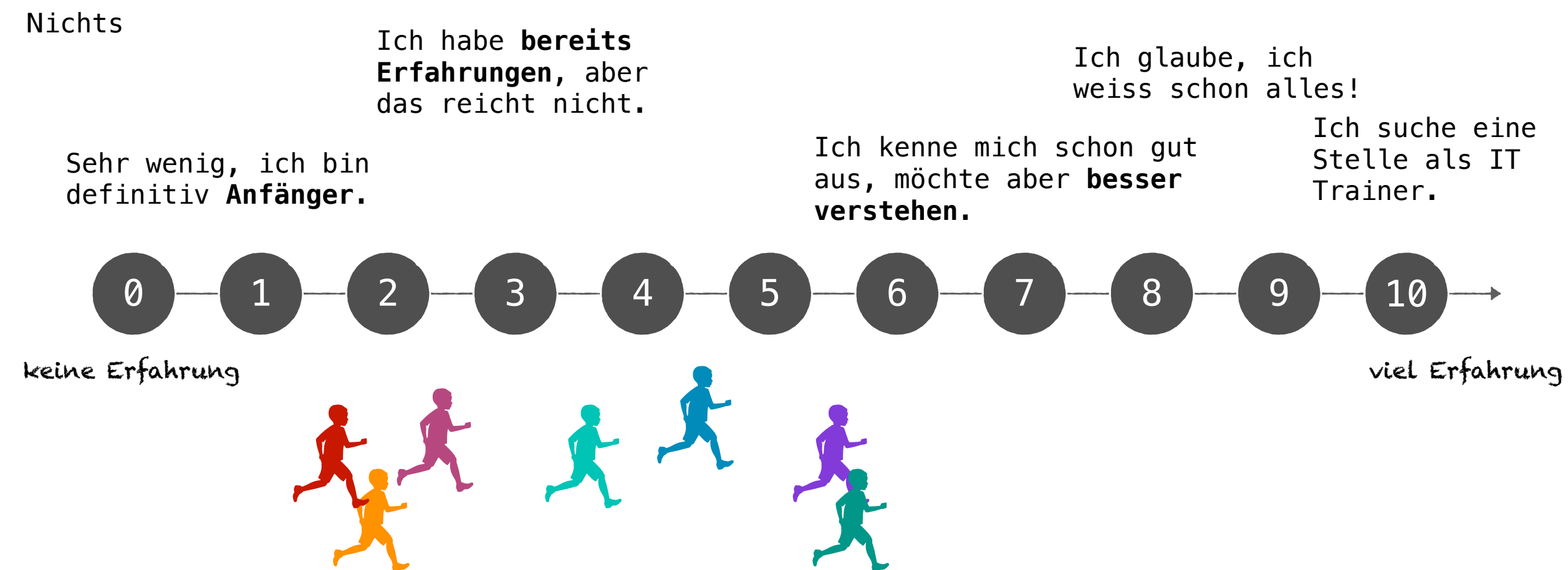
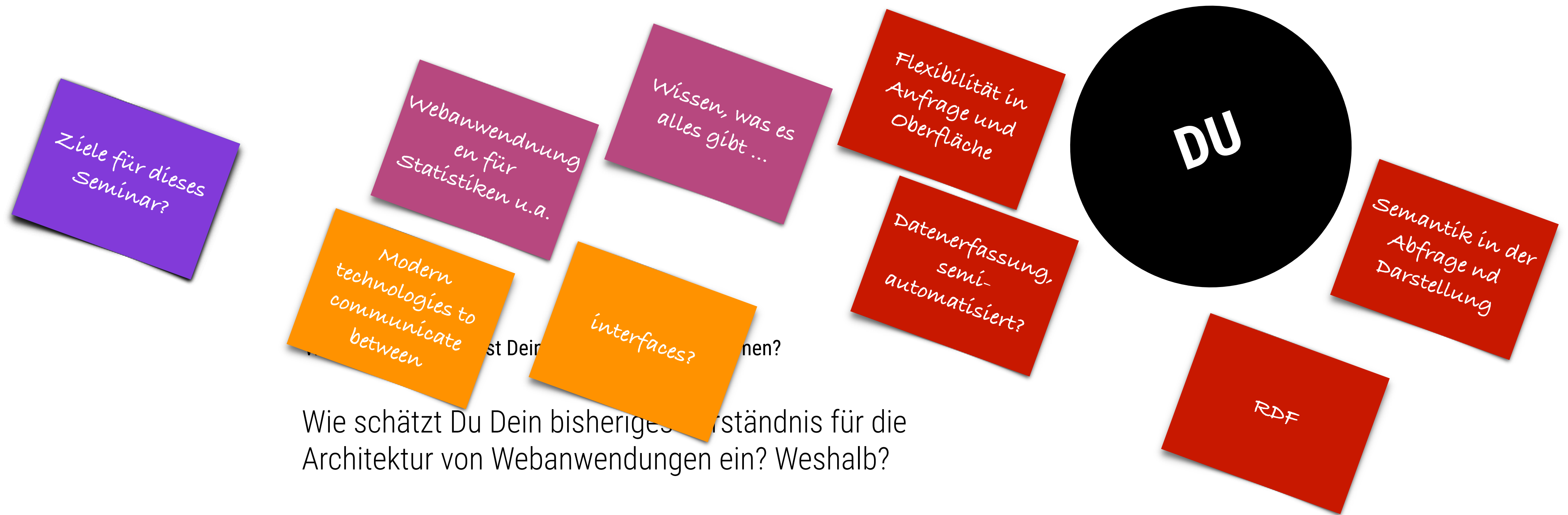


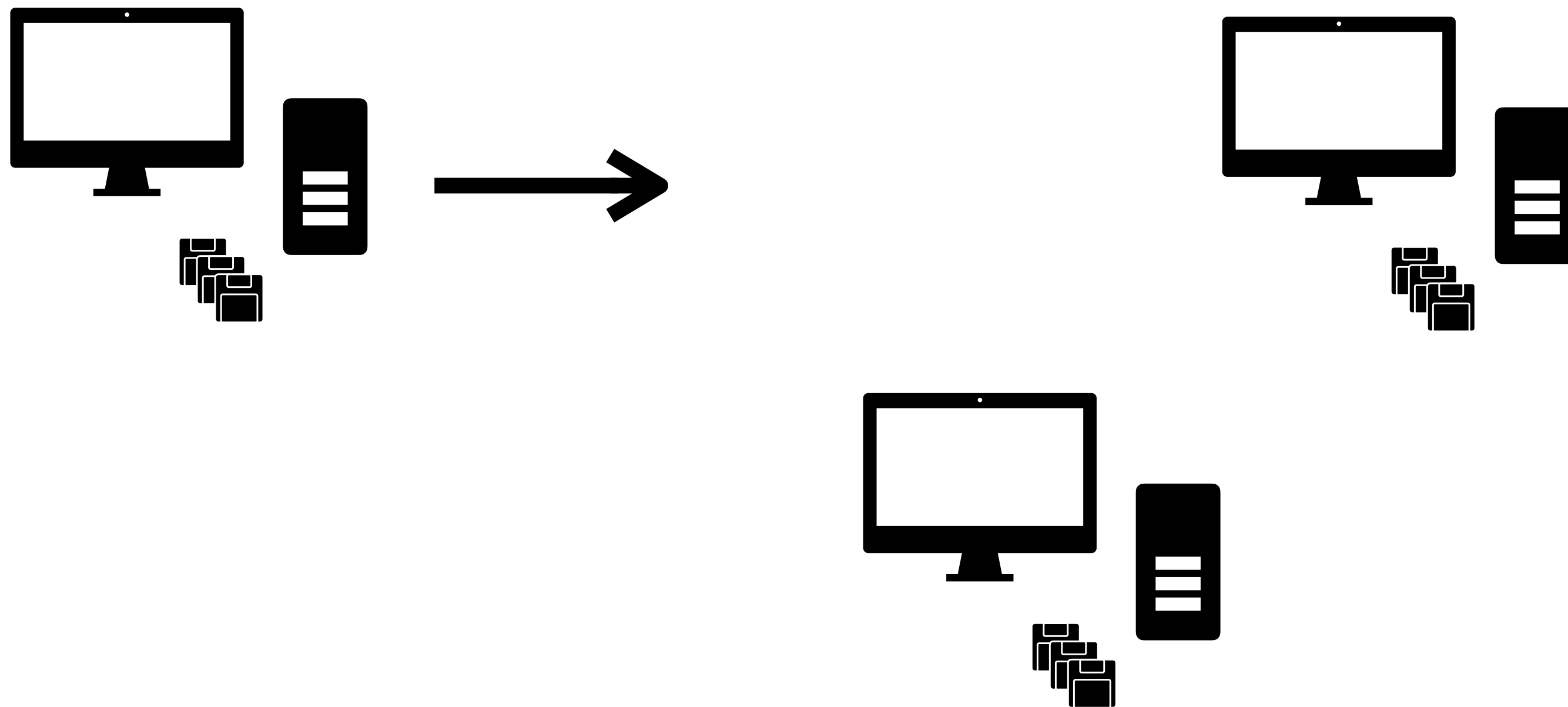
Januar 2023

# ARCHITEKTUR MODERNER WEBANWENDUNGEN

Grundlagen zur Realisierung von Webanwendungen

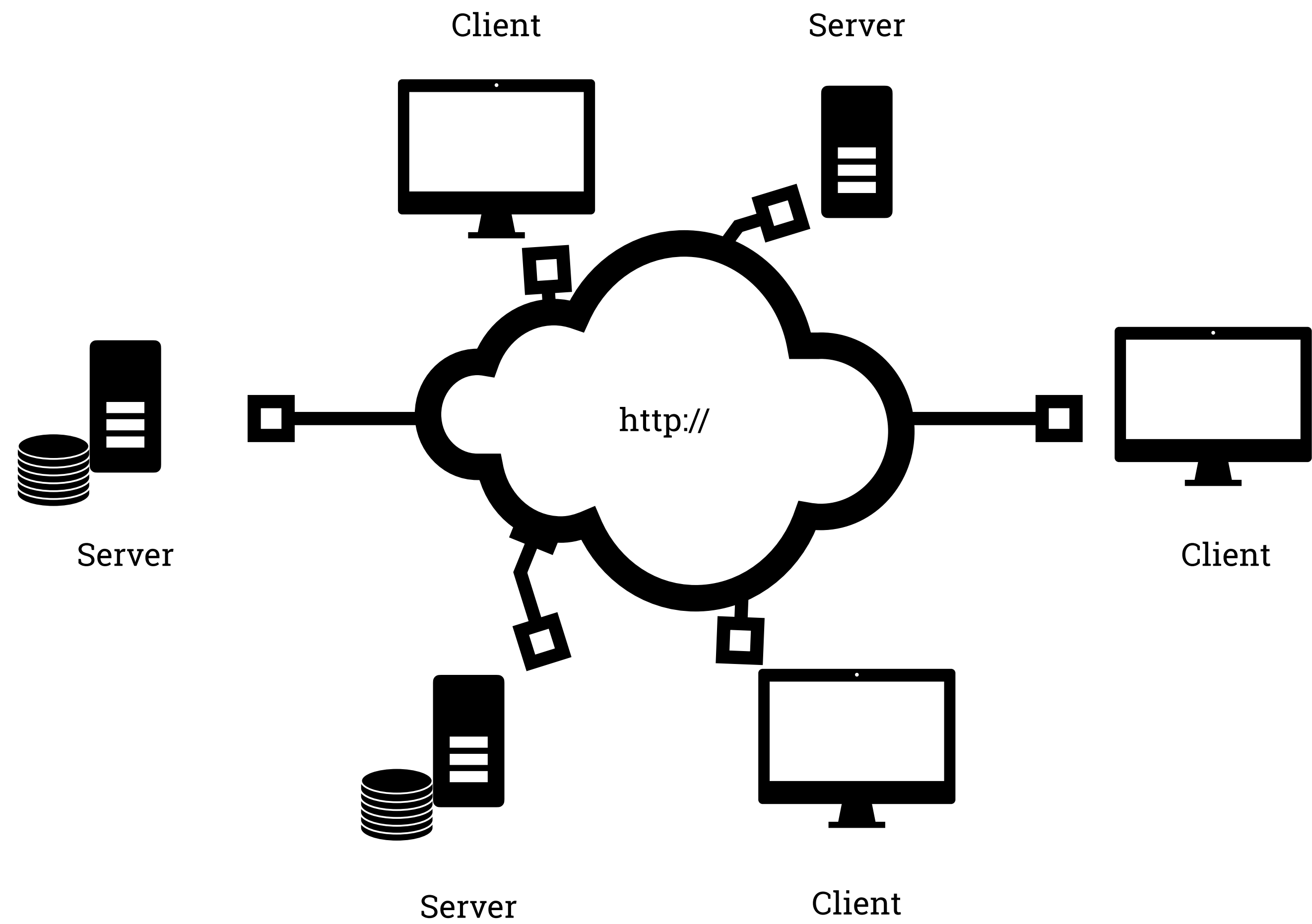


## 1990: Monolithische Software, Installation



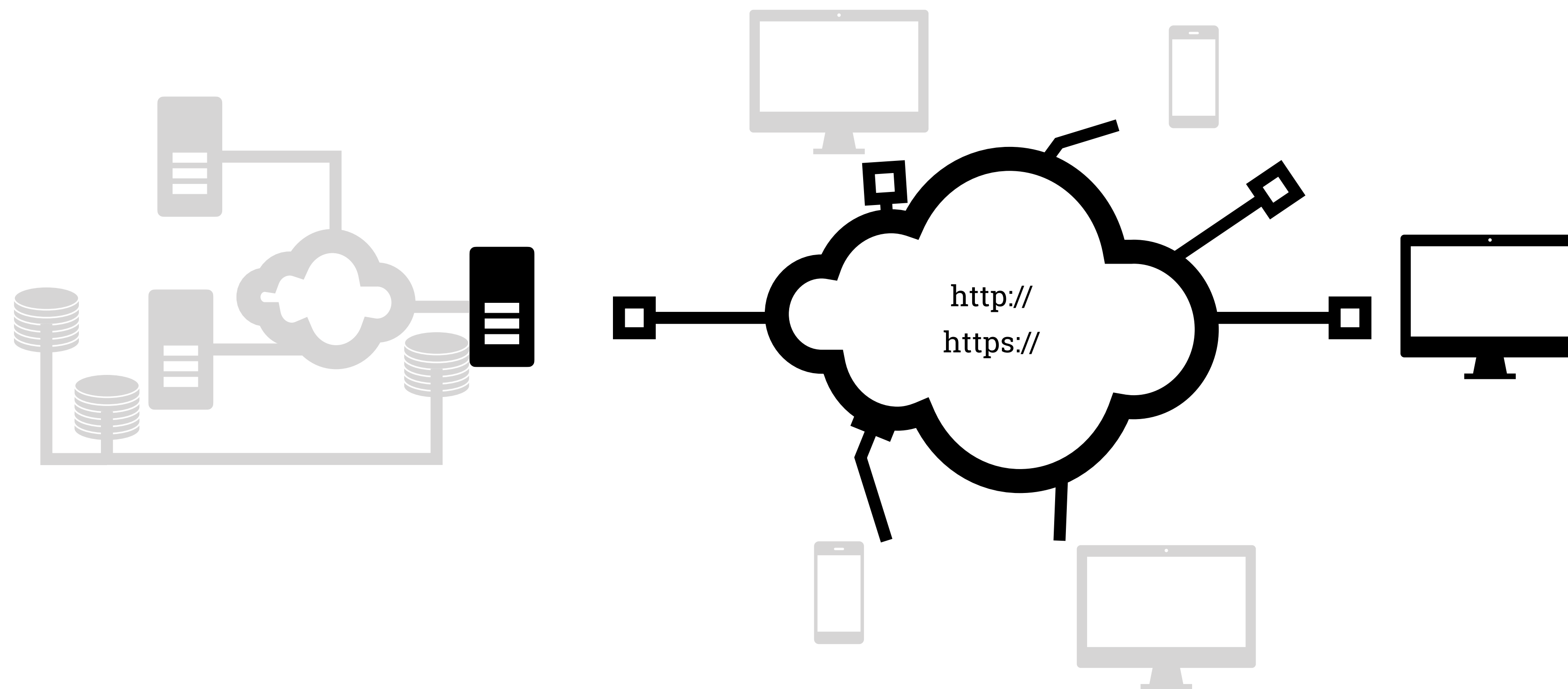
# 1989: INTERNET

---



# WEBSERVER - NET - WEBCLIENT

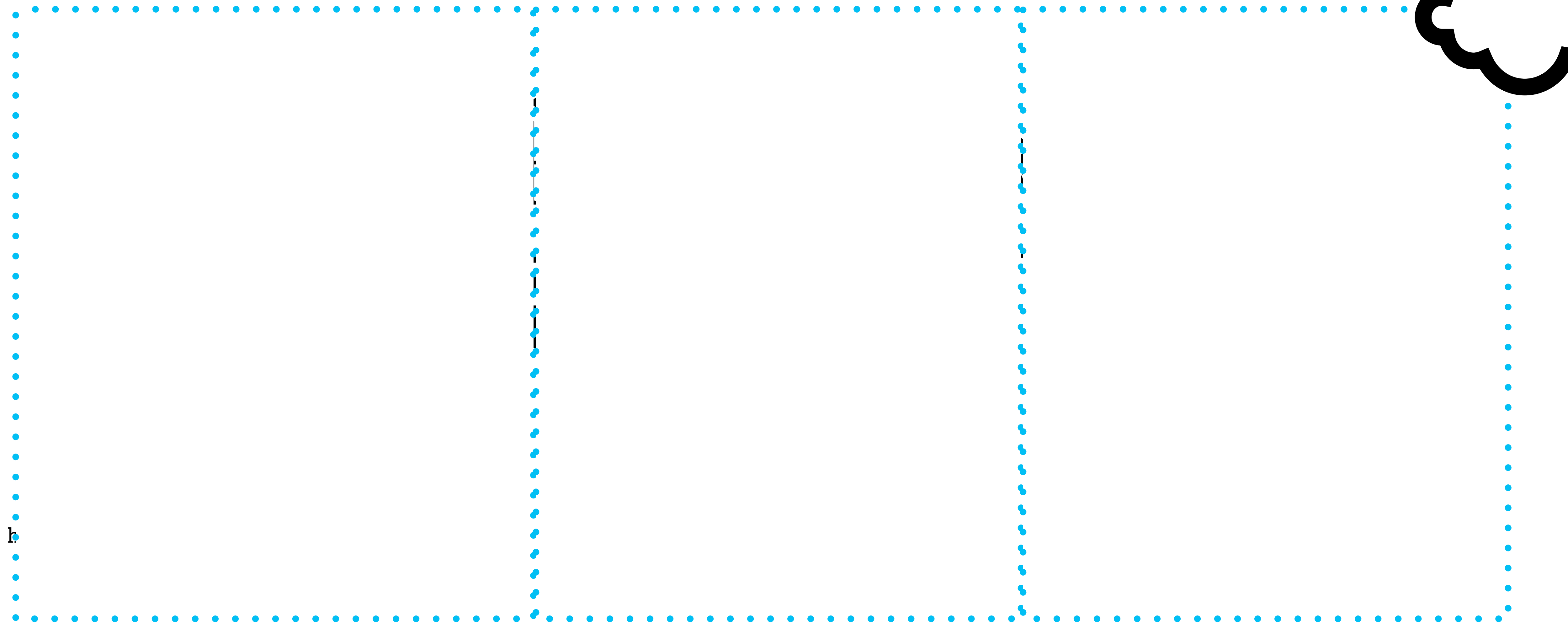
---



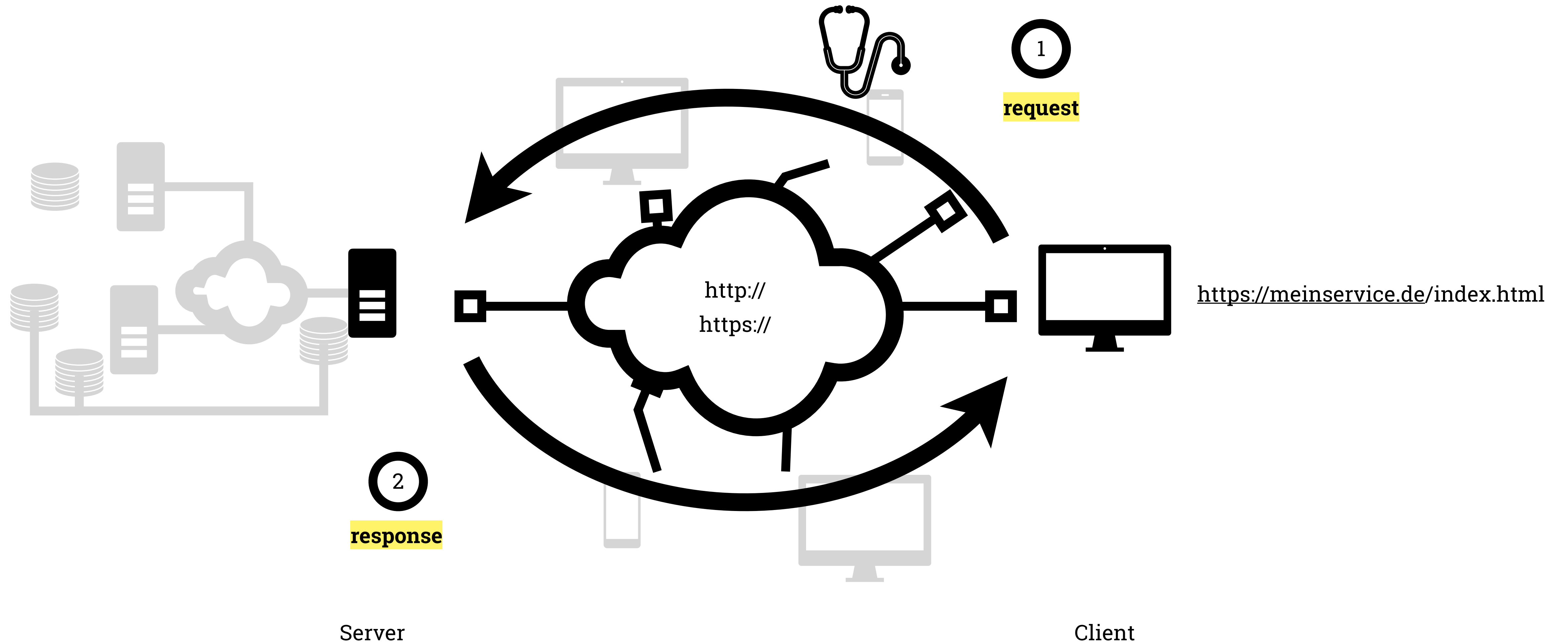
Server

Client

# Web, Web 2.0, Digitalisierung, Industrie 4.0 (Automatisierung), Autonomisierung



# WEBSERVER - INTERNET - WEBCLIENT



- ▶ HTML - Hypertext Markup Language
- ▶ CSS - Cascading Style Sheets
- ▶ Javascript
- ▶ DHTML
- ▶ AJAX
- ▶ Browser-Extensions
- ▶ HTML5
- ▶ SPA, AMP, PWA
- ▶ Web Hook (Middleware?)
- ▶ Webserver
- ▶ HTTP, HTTP2, HTTP3
- ▶ Server-Side Processing
- ▶ Programmiertechnologien
- ▶ Anwendungsarchitekturen (Softwarearchitektur)  
OSI, 4+1 Sichten, Client-Server, 3- und N-Tier-Systeme
- ▶ Firewall
- ▶ Router
- ▶ Grundbegriffe der Sicherheit
- ▶ Authentifizierung und Autorisierung
- ▶ HTTPS / TLS (aka SSL)
- ▶ OWASP
- ▶ RDF
- ▶ XML
- ▶ JSON
- ▶ ODATA



# SOFTWARE ARCHITEKTUR - SICHTEN UND SCHICHTEN

---

- ▶ Anwendungsarchitekturen (Softwarearchitektur)  
OSI, 4+1 Sichten, Client-Server, 3- und N-Tier-Systeme

## 4+1 - SICHTEN-MODELL

---

**logical view** - Funktionalität des Systems für den **Endnutzer**. (Lastenheft-Team Fragen den Endnutzer)

**process view** - sie verdeutlicht die Prozesse des Systems, sowie deren Laufzeitverhalten und Kommunikation. Die Prozesssicht beschreibt Parallelität, Verteilung, Integration, Performance und Skalierbarkeit. (Requirements Engineer)

**development view** (*implementation view*) - beschreibt das System vom Standpunkt eines Entwicklers. (20% Scrummaster als Teamkoordinator)

**physical view** (*deployment view*) beschreibt das System vom Standpunkt des Systemarchitekten. Es beschäftigt sich mit der Verteilung der Softwarekomponenten auf physikalischer Ebene (Hardware) und der Kommunikation zwischen diesen Komponenten.

**scenarios**: Die fünfte Sicht zeigt wichtige Anwendungsfälle oder Anwendungsszenarien. Sie beschreiben Abläufe zwischen Komponenten bzw. Prozessen. Sie helfen, Architekturelemente zu identifizieren, zu veranschaulichen und die Architektur zu überprüfen.

Sie dienen auch als Startpunkt für erste Architekturtests bzw. Implementierungsentwürfe.

## 4+1 - SICHTEN-MODELL

---

***logical view*** - Funktionalität des Systems für den **Endnutzer**.

- *Beispiel Abrechnungssystem:*
- *Eingabe von Dokumenten*
- *Zuordnung*
- *Verarbeitung*
- *ggf. Verteilung etc.*

## Sicht des Anwenders

UML-Diagrammformen:

Klassendiagramm,

Kommunikationsdiagramm,

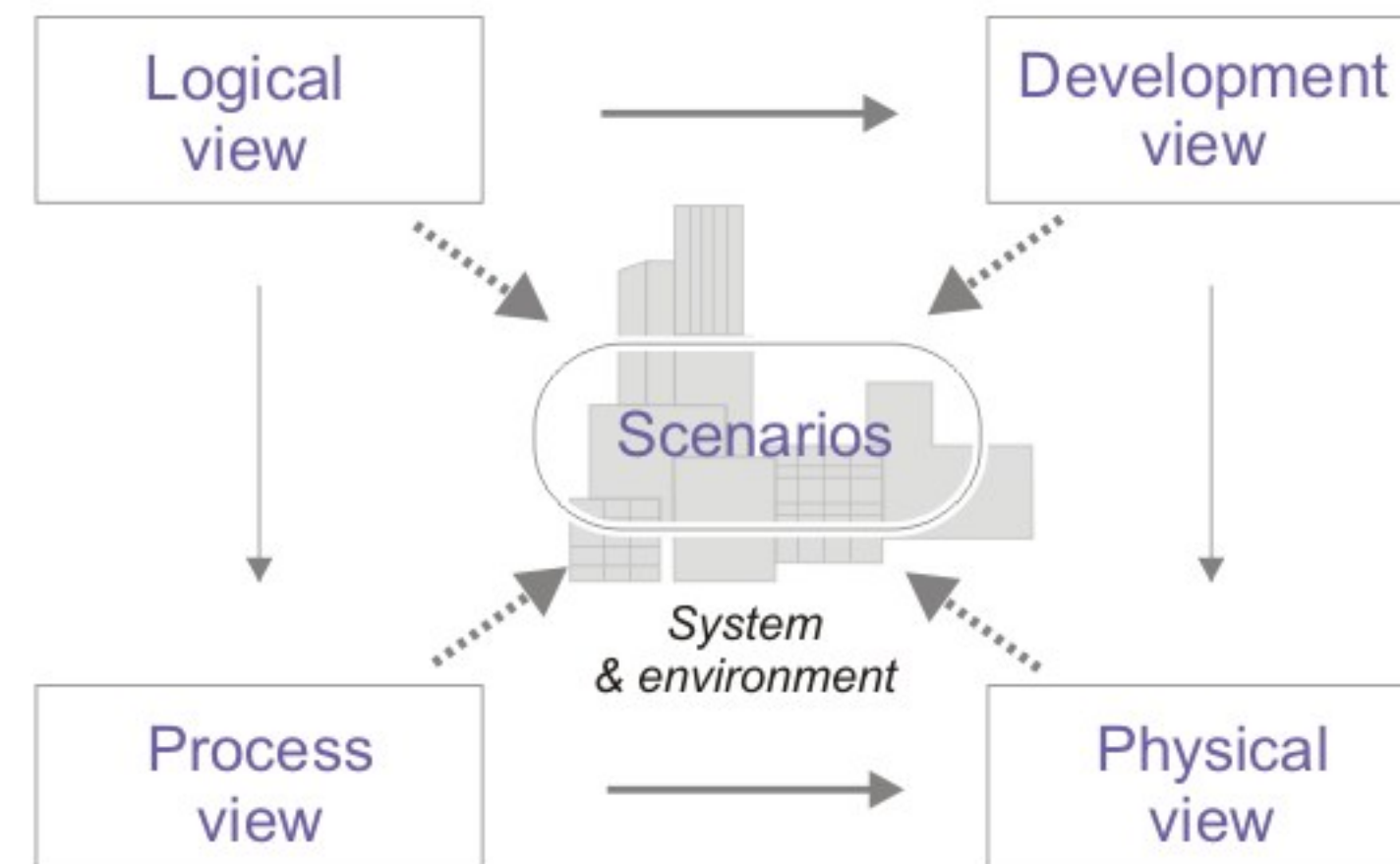
Sequenzdiagramm

u.a.

Dynamik des Systems;  
Laufzeitverhalten

UML-Diagrammform:

Anwendungsfalldiagramm



## Sicht des Entwicklers

UML-Diagrammformen:

Komponentendiagramm,

Paketdiagramm

Sicht des  
Systemarchitekten

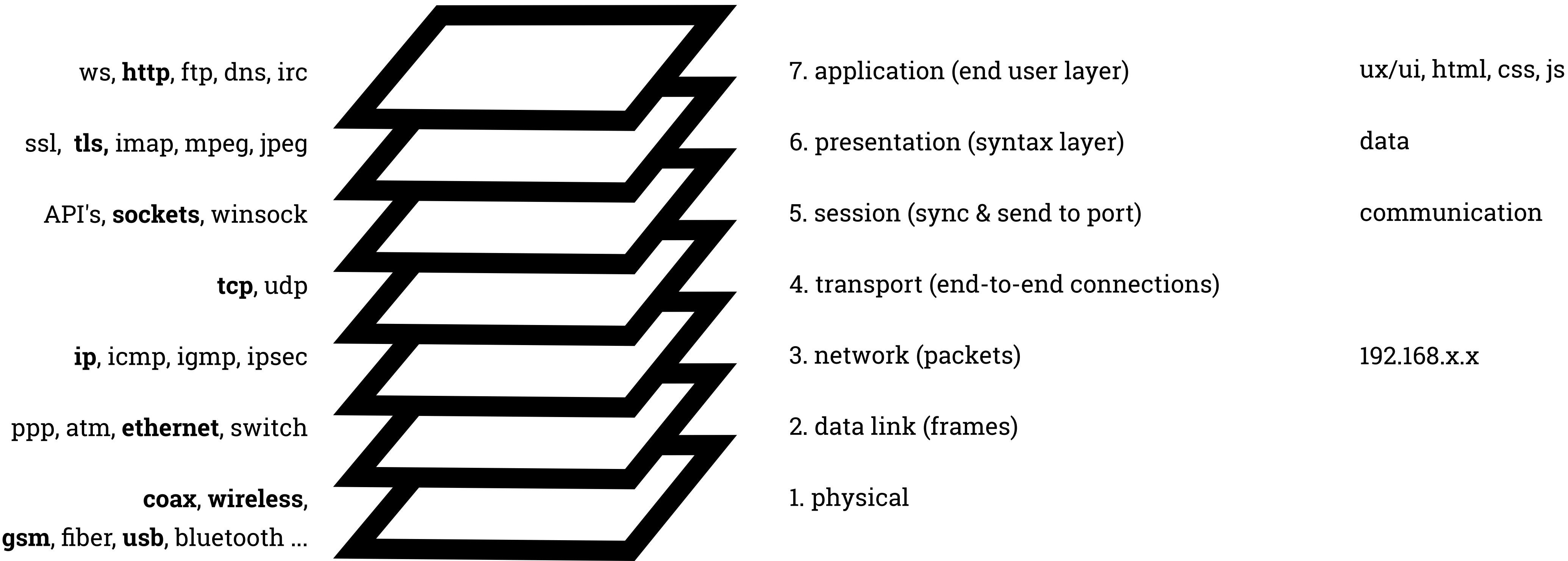
UML-Diagrammform:

Verteilungsdiagramm

# OSI - OPEN SYSTEMS INTERCONNECTION

seven tiers of OSI

**https:443//192.168.x.x/**



# OSI - OPEN SYSTEMS INTERCONNECTION

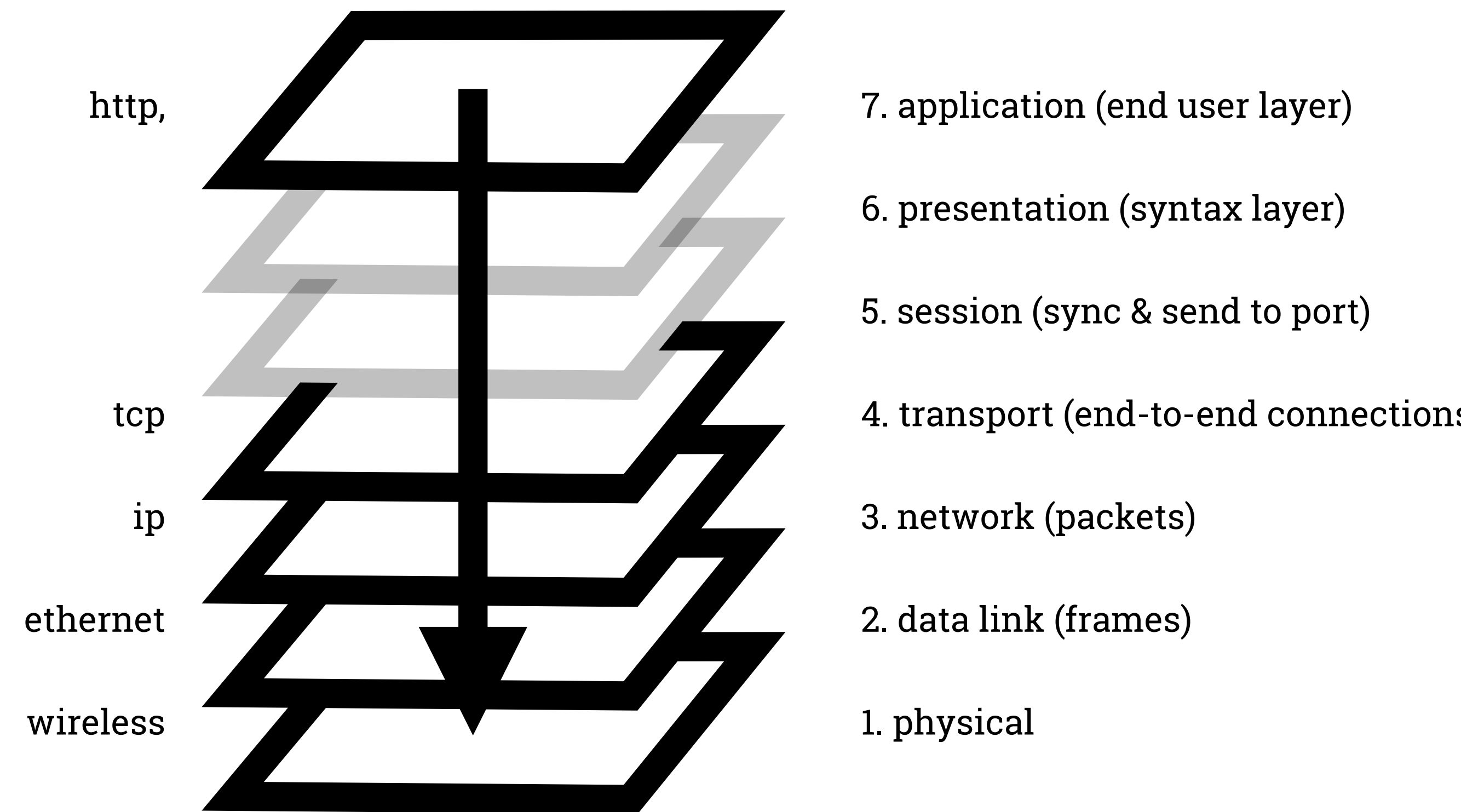
---

If you use your web browser to navigate to **http://www.gfu.net**, this communication uses the following protocols from each layer, starting at layer 7:

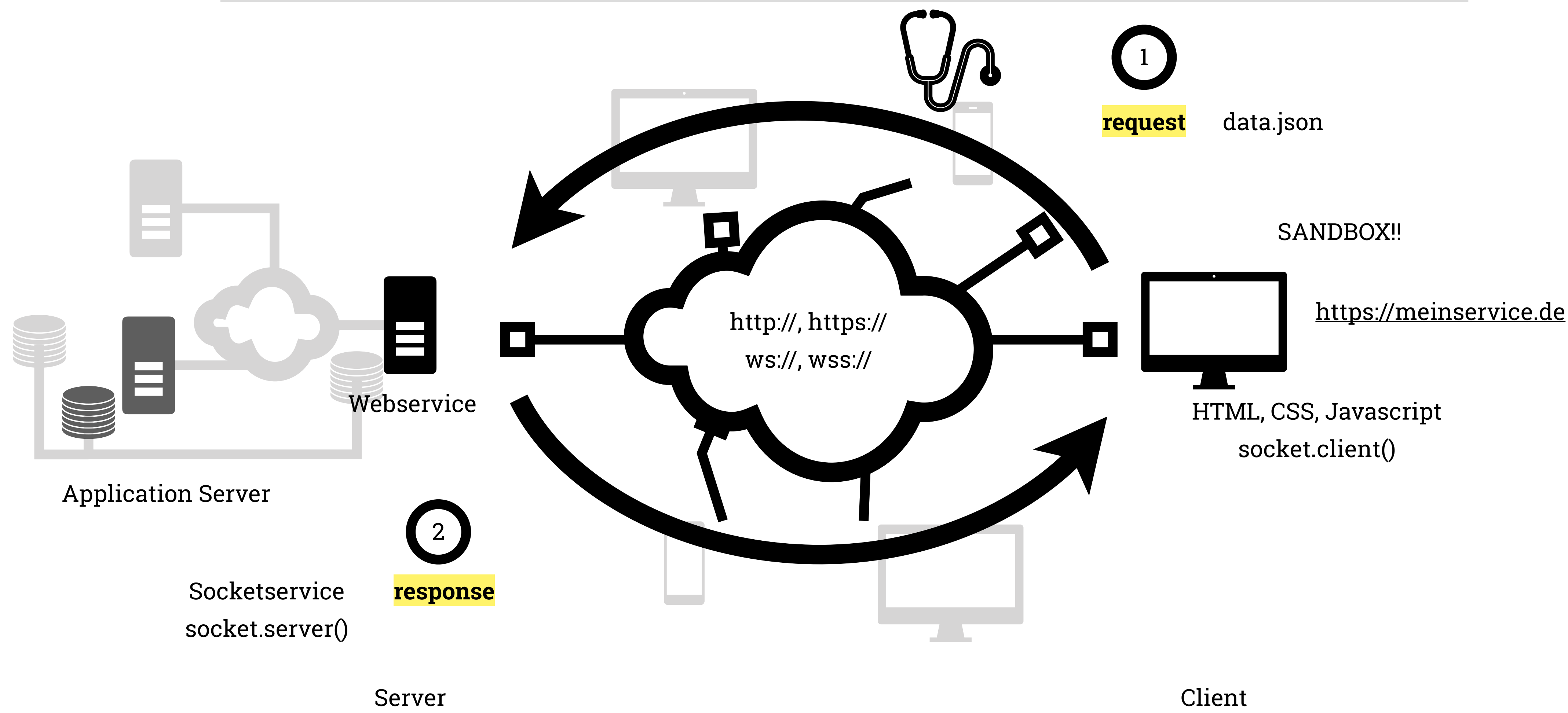
**HTTP → TCP → IP → Ethernet.**

On the other hand, entering **https://www.gfu.net** would use

**HTTP → SSL → TCP → IP → Ethernet.**



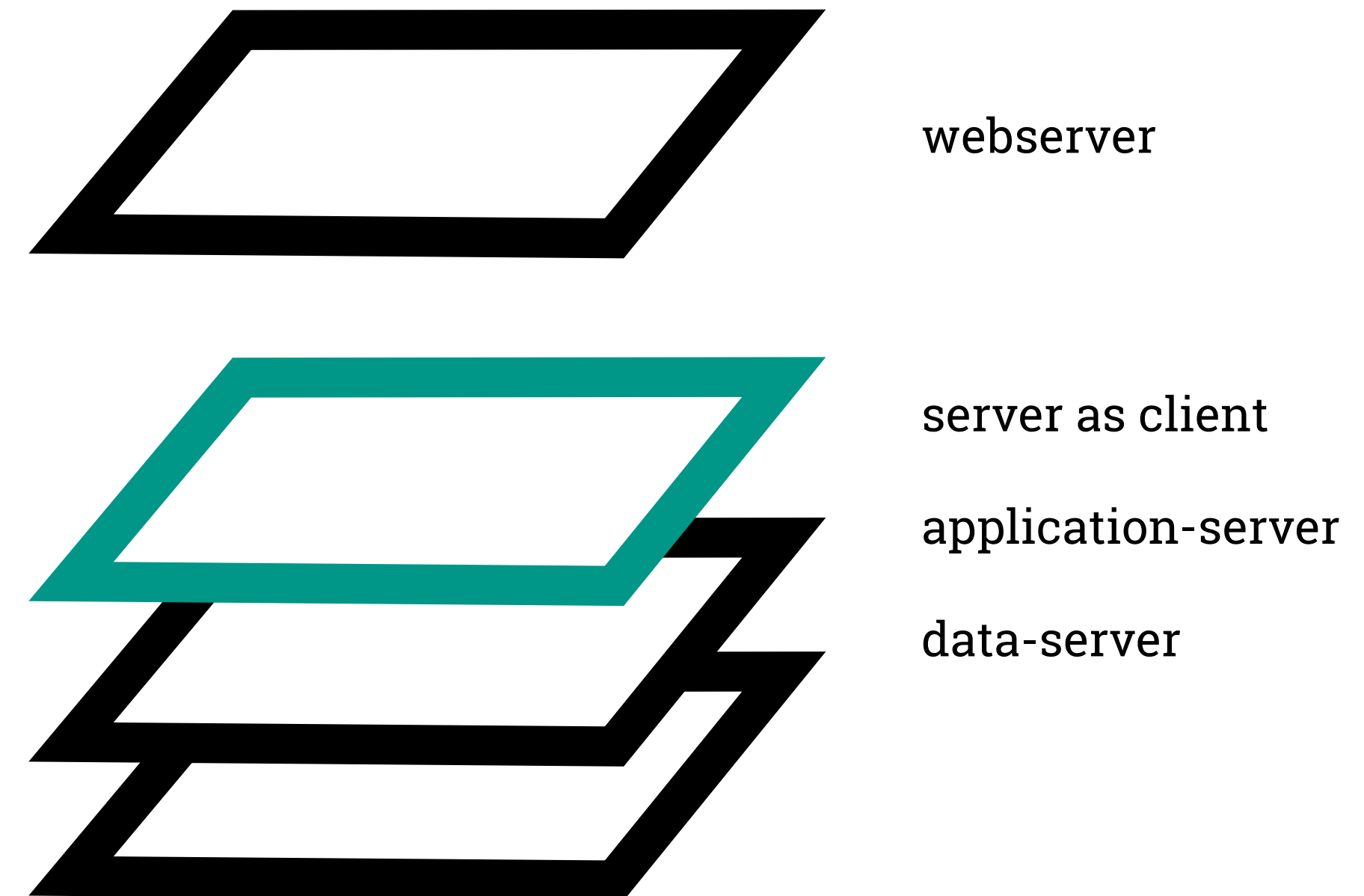
# WEBSERVER - INTERNET - WEBCLIENT



# DREI-SCHICHTEN-ARCHITEKTUR

three tier architecture

Im Gegensatz zur zweischichtigen Architektur gibt es bei der dreischichtigen Architektur noch Logikschicht, die die Datenverarbeitung vornimmt.

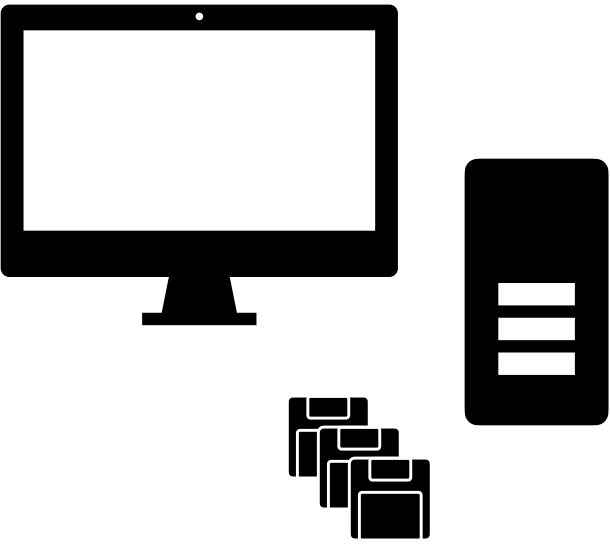




# SOFTWAREARCHITEKTUR

---

Die strukturierte oder hierarchische Anordnung der Systemkomponenten sowie Beschreibung ihrer Beziehungen.



# ZWEI-SCHICHTEN-ARCHITEKTUR

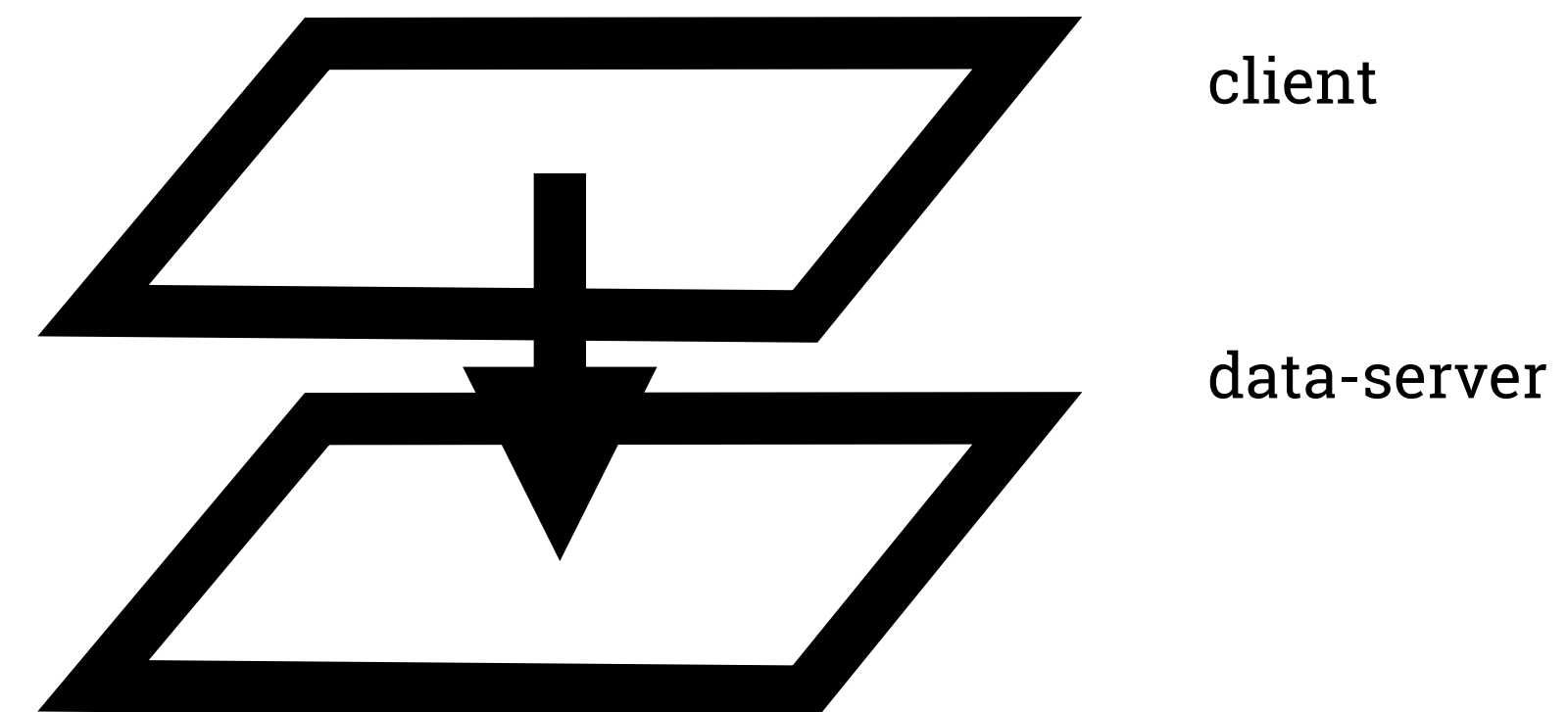
---

two tier architecture

*client-server-architektur.*

Die niedrigere Schicht ist ein Dienstanbieter (englisch Server) der höheren.

Die höhere Schicht (client) darf auf die niedrigere Schicht (server) zugreifen.



# ZWEI-SCHICHTEN-ARCHITEKTUR

---

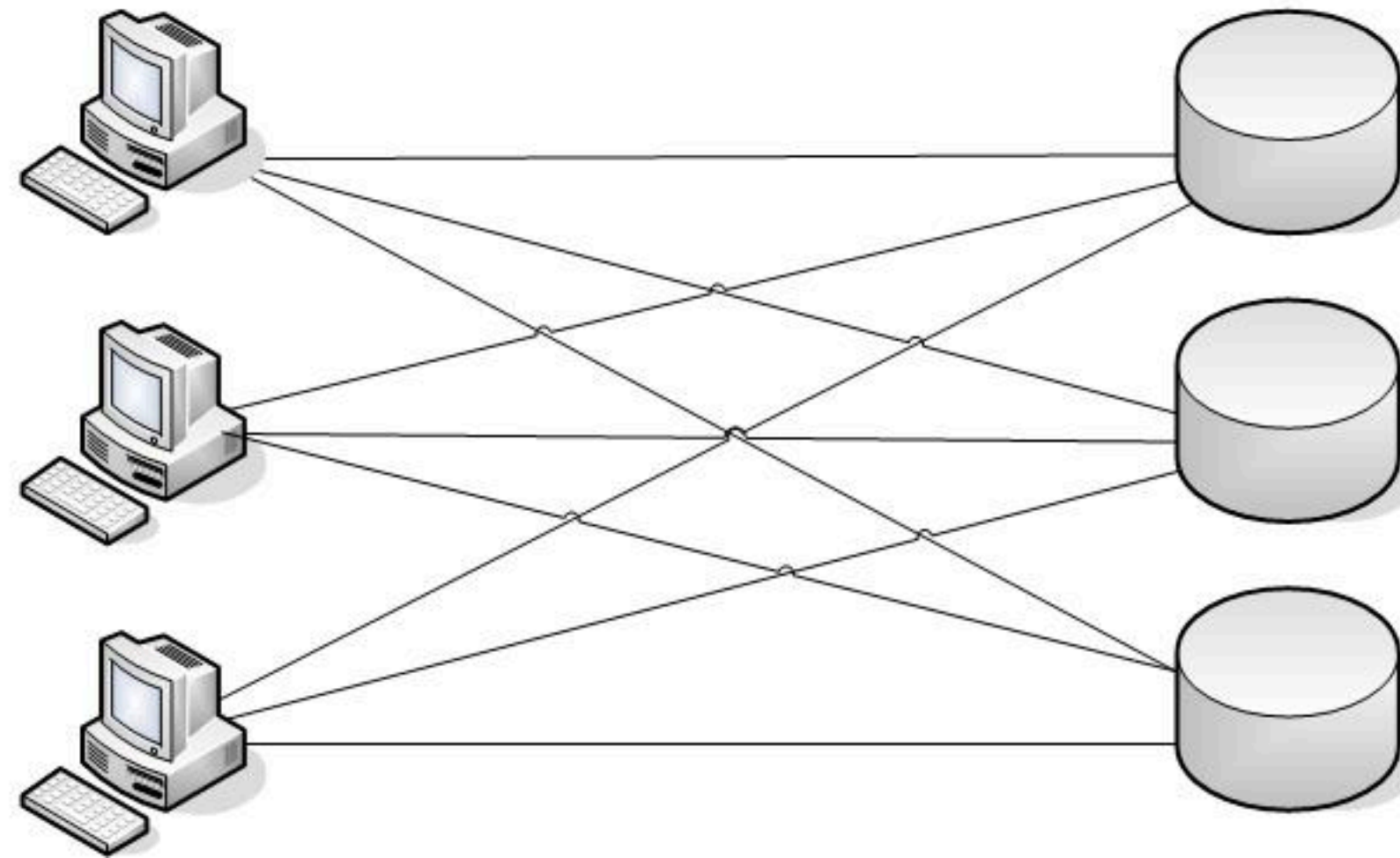
Traditionell kommen ein *fat client* und ein *fat server* zum Einsatz.

Auf dem Server läuft eine Datenbank Anwendung. Die Clients übernehmen Logik und Benutzerschnittstelle.

Client-Server-Architekturen können aber auch als eine 2-Module-Anwendung verstanden werden, in der ein Modul auf ein zweites Modul auf demselben Rechner, meist innerhalb derselben Anwendung zugreift.

Anwendungsschicht

Datenschicht



client

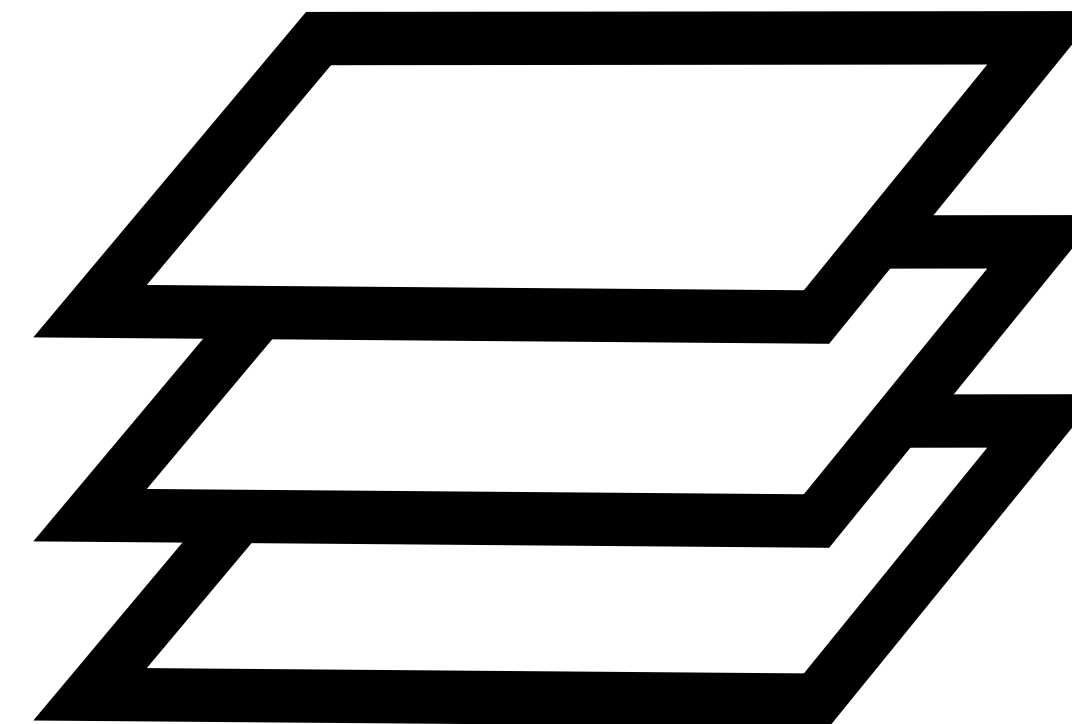
data-server

# DREI-SCHICHTEN-ARCHITEKTUR

---

three tier architecture

Im Gegensatz zur zweischichtigen Architektur gibt es bei der dreischichtigen Architektur noch Logikschicht, die die Datenverarbeitung vornimmt.



client-services

application-services

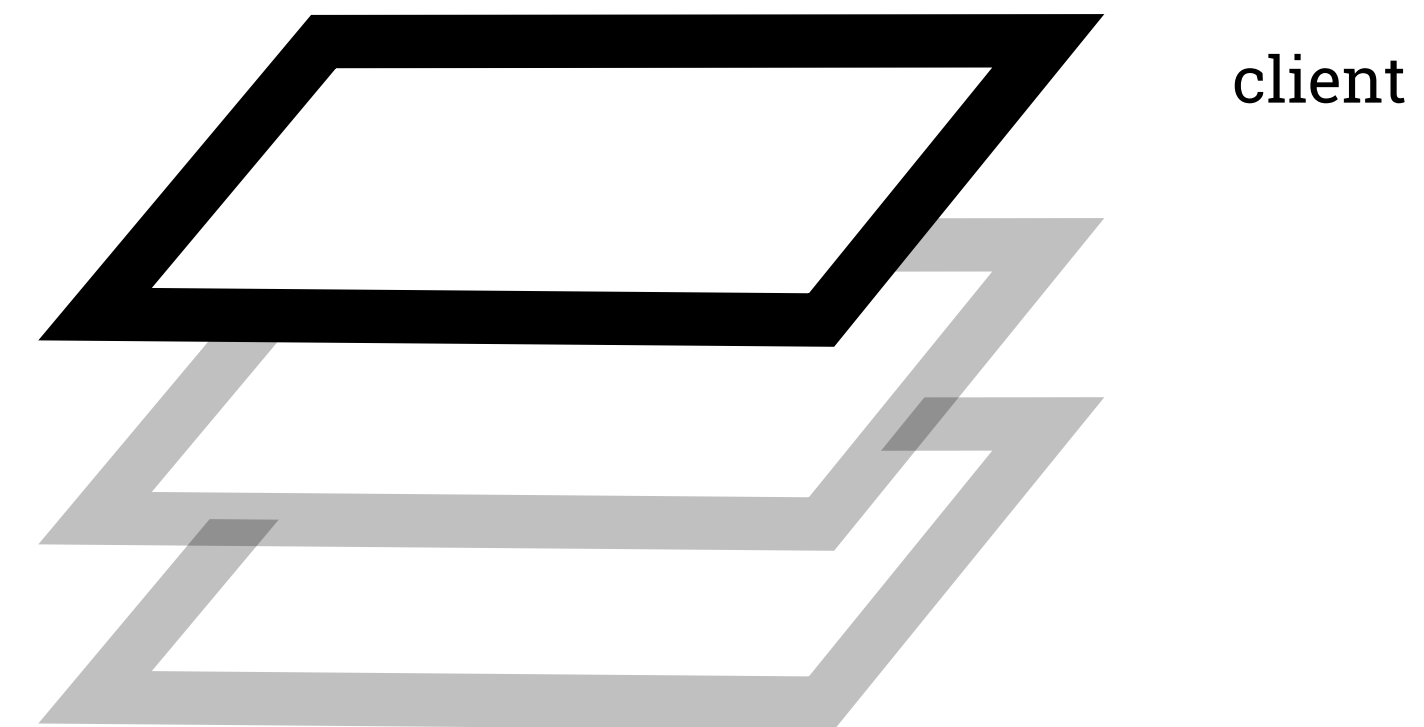
data-services

# PRÄSENTATIONSSCHICHT

---

*client tier, front end (thin client)*

Sie ist für die Repräsentation der Daten verantwortlich und bildet die **Schnittstelle zum Benutzer**, insbesondere für die Benutzereingaben.

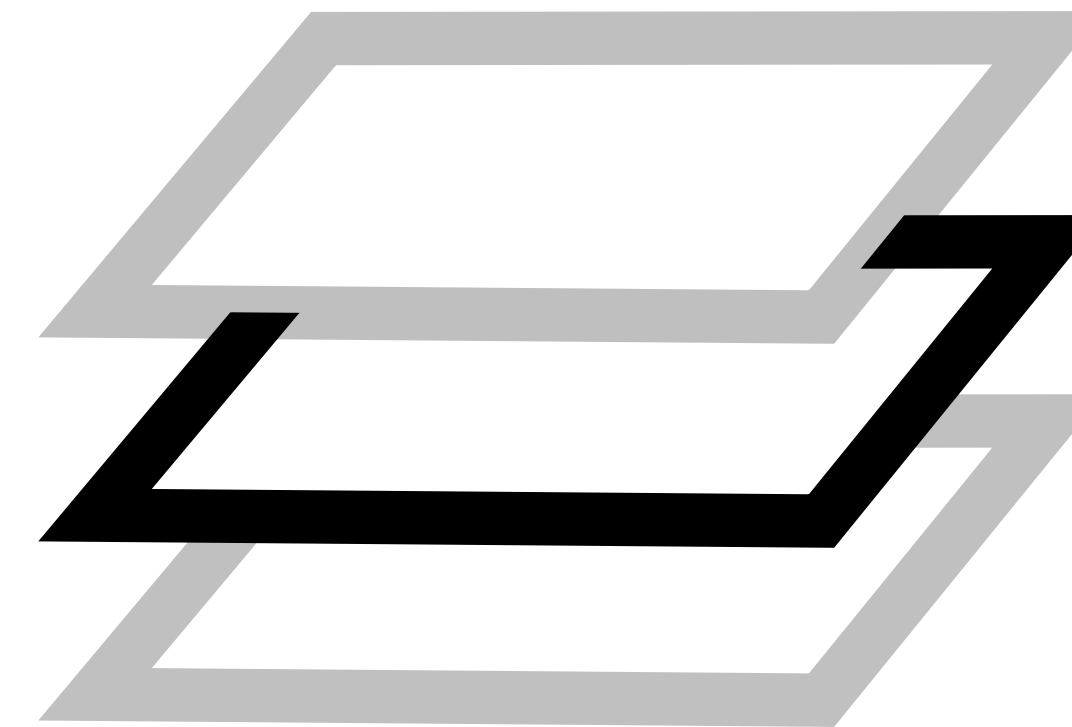


# LOGIKSCHICHT

---

*application-server tier, business tier,  
middle tier, enterprise tier*

Sie beinhaltet alle  
Verarbeitungsmechanismen. Hier ist die  
**Anwendungslogik** vereint.



application-server



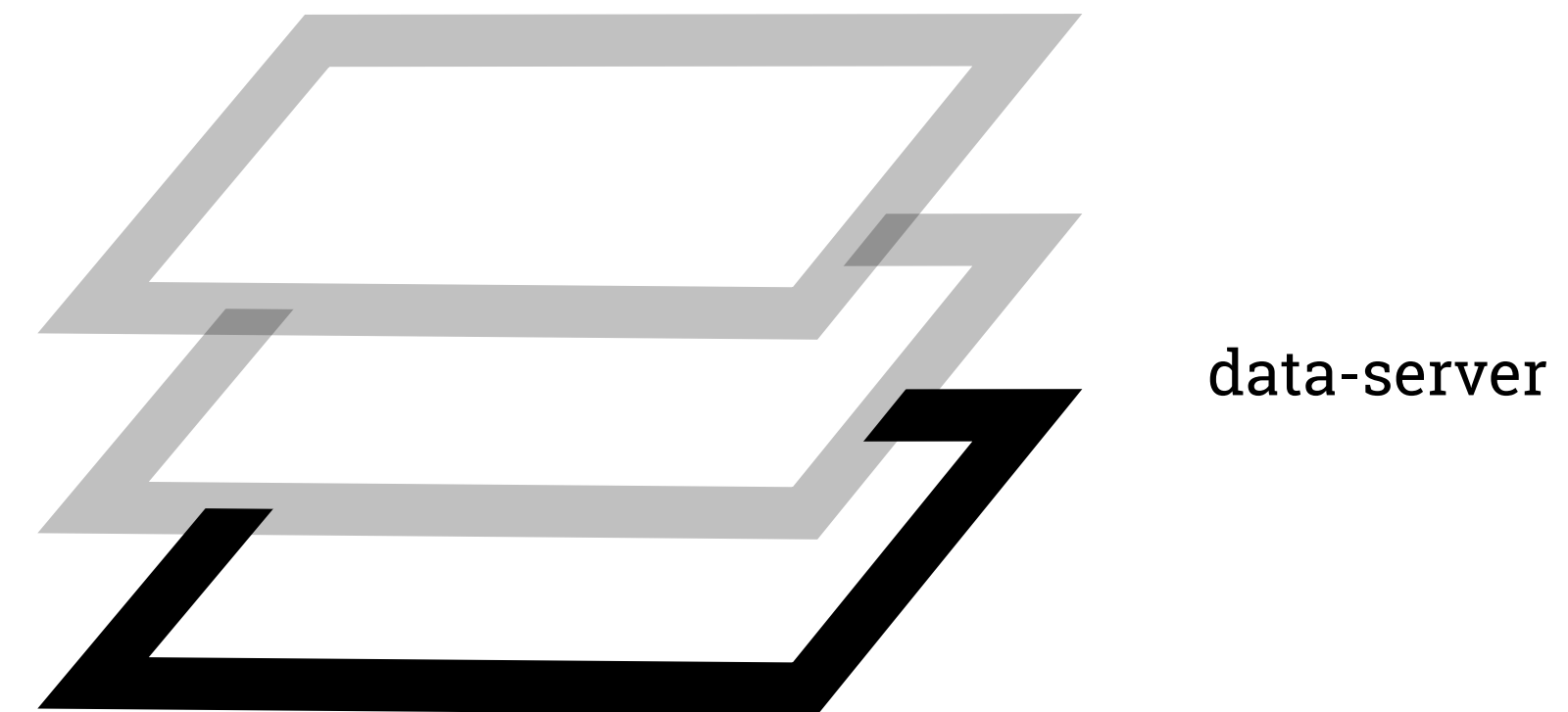
# DATENSCHICHT

---

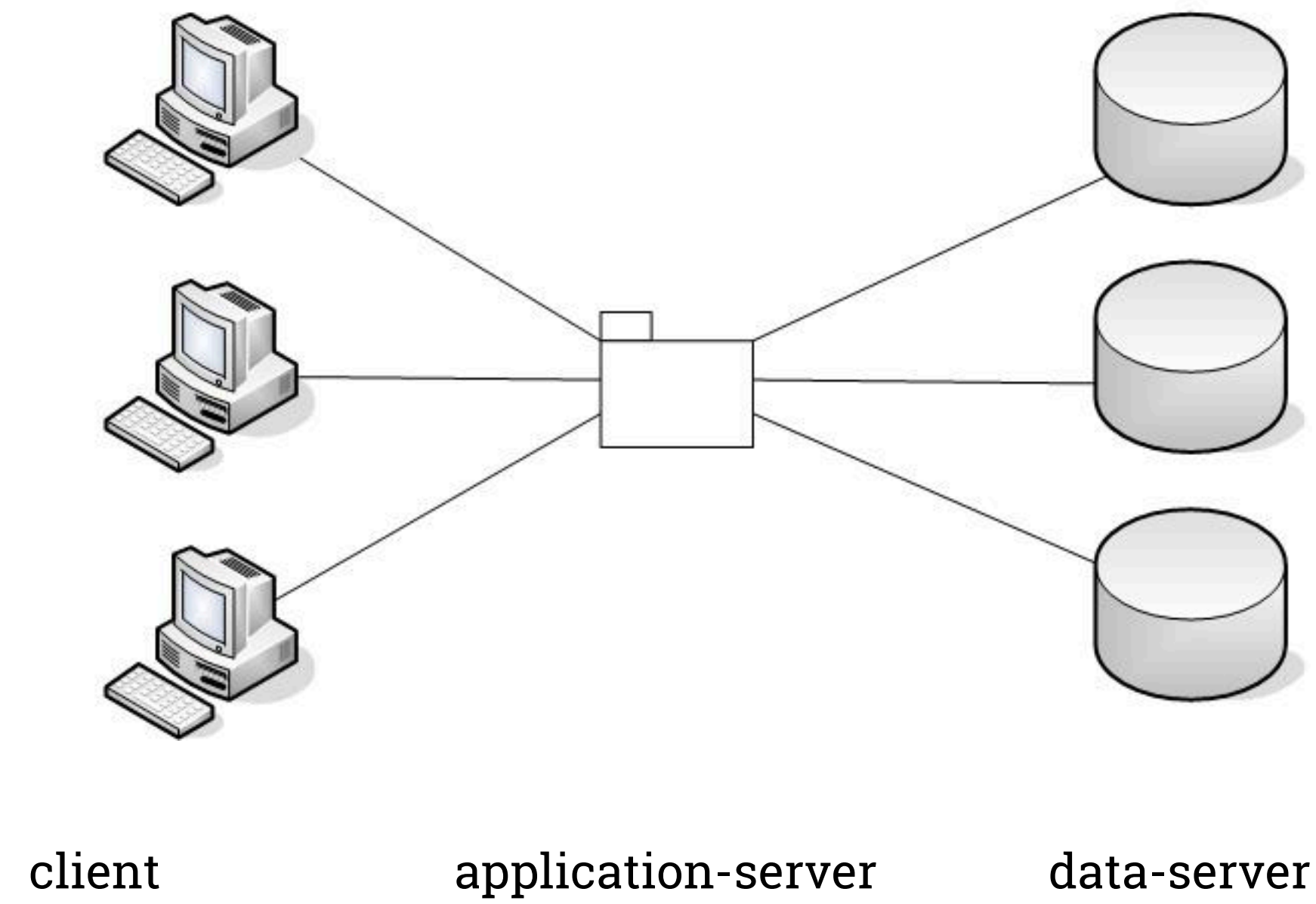
*data-server tier, back end*

Sie enthält die Datenbank und ist verantwortlich für das Speichern und Laden von Daten.

SQL Daten, Big Data, Data Lake, XML, JSON



Anwendungsschicht    Domänenschicht    Datenschicht



**Domäne** (von lateinisch *dominium* über französisch *domaine* „Herrschaft, Herrschaftsbereich“)

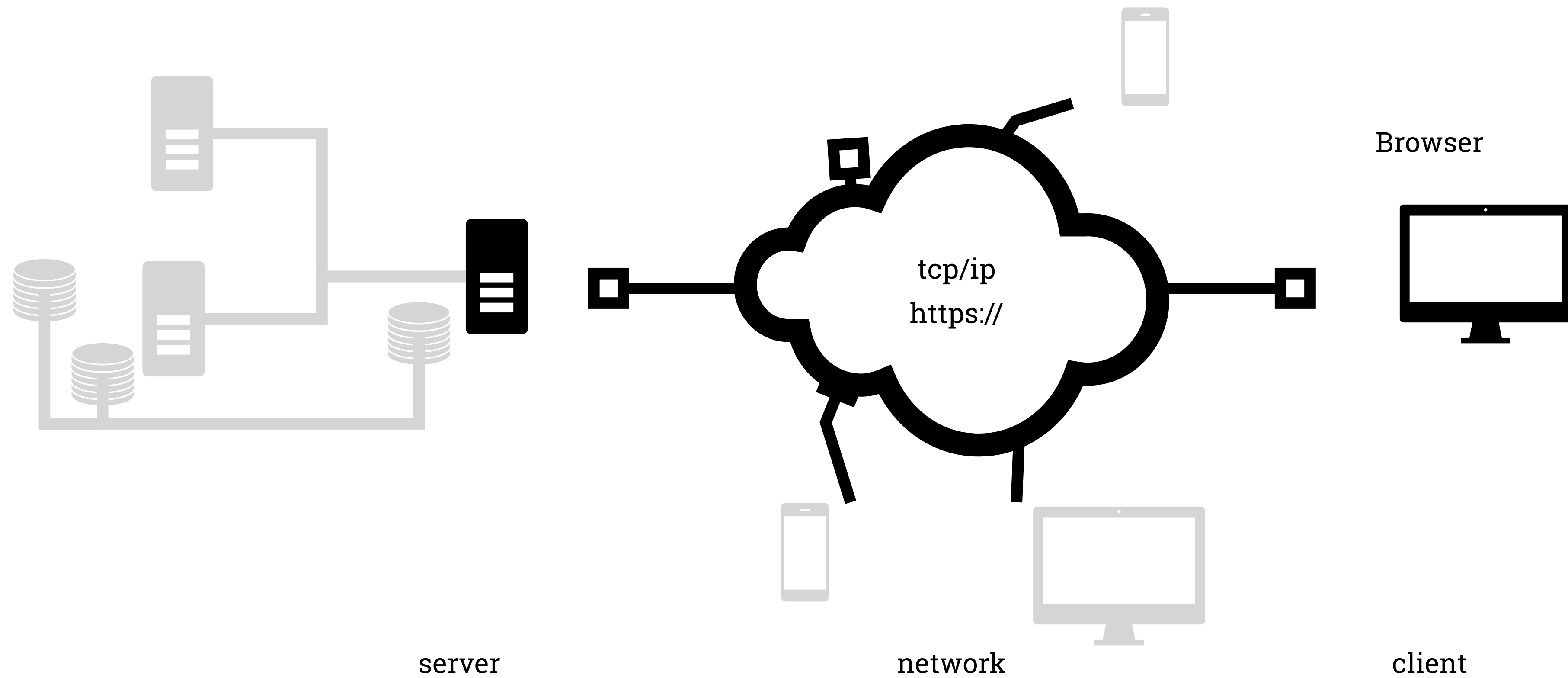
# WEB-CLIENT TECHNOLOGIEN

---

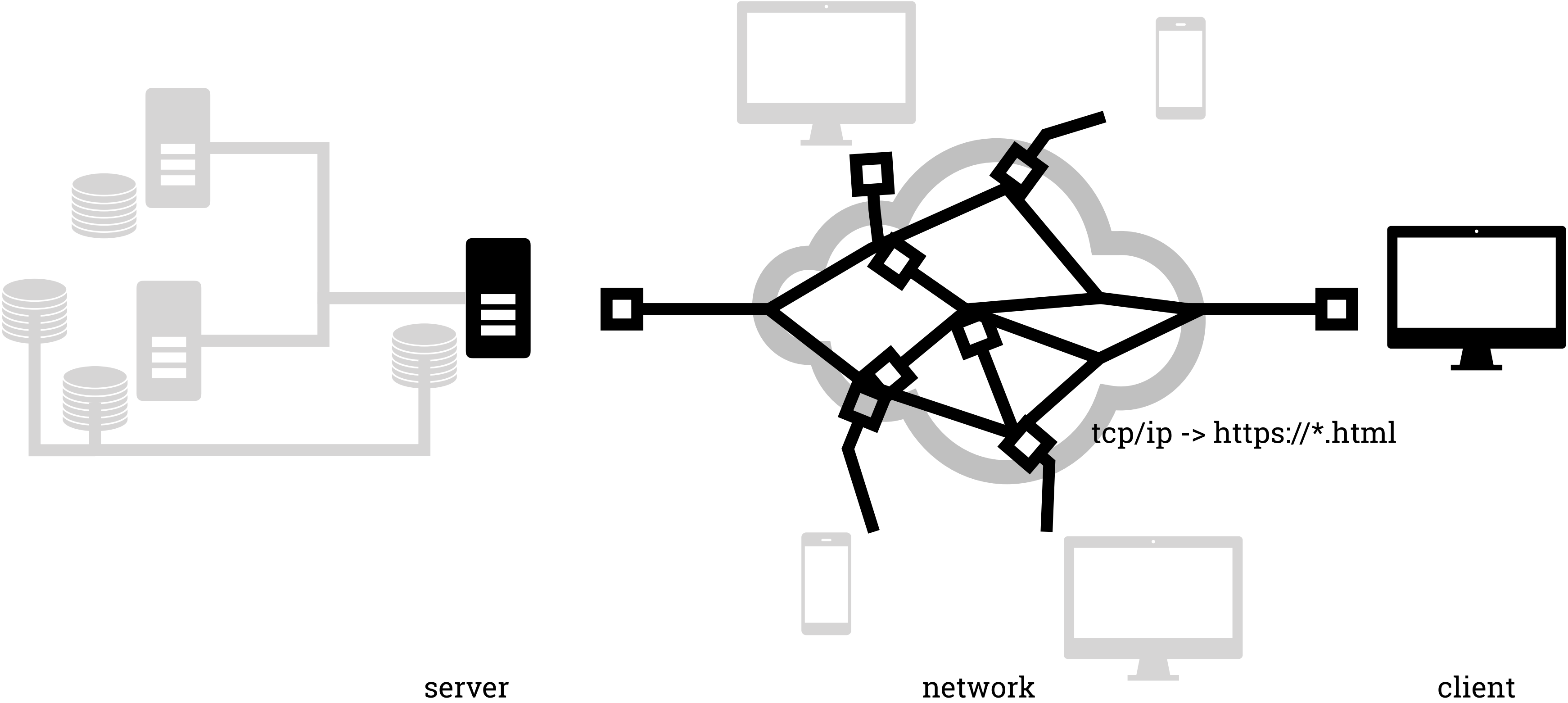
- ▶ HTML - Hypertext Markup Language
- ▶ CSS - Cascading Style Sheets
- ▶ Javascript
- ▶ DHTML
- ▶ Ajax
- ▶ HTML5
- ▶ SPA, AMP, PWA

# WEBSERVER - INTERNET - WEBCLIENT

---

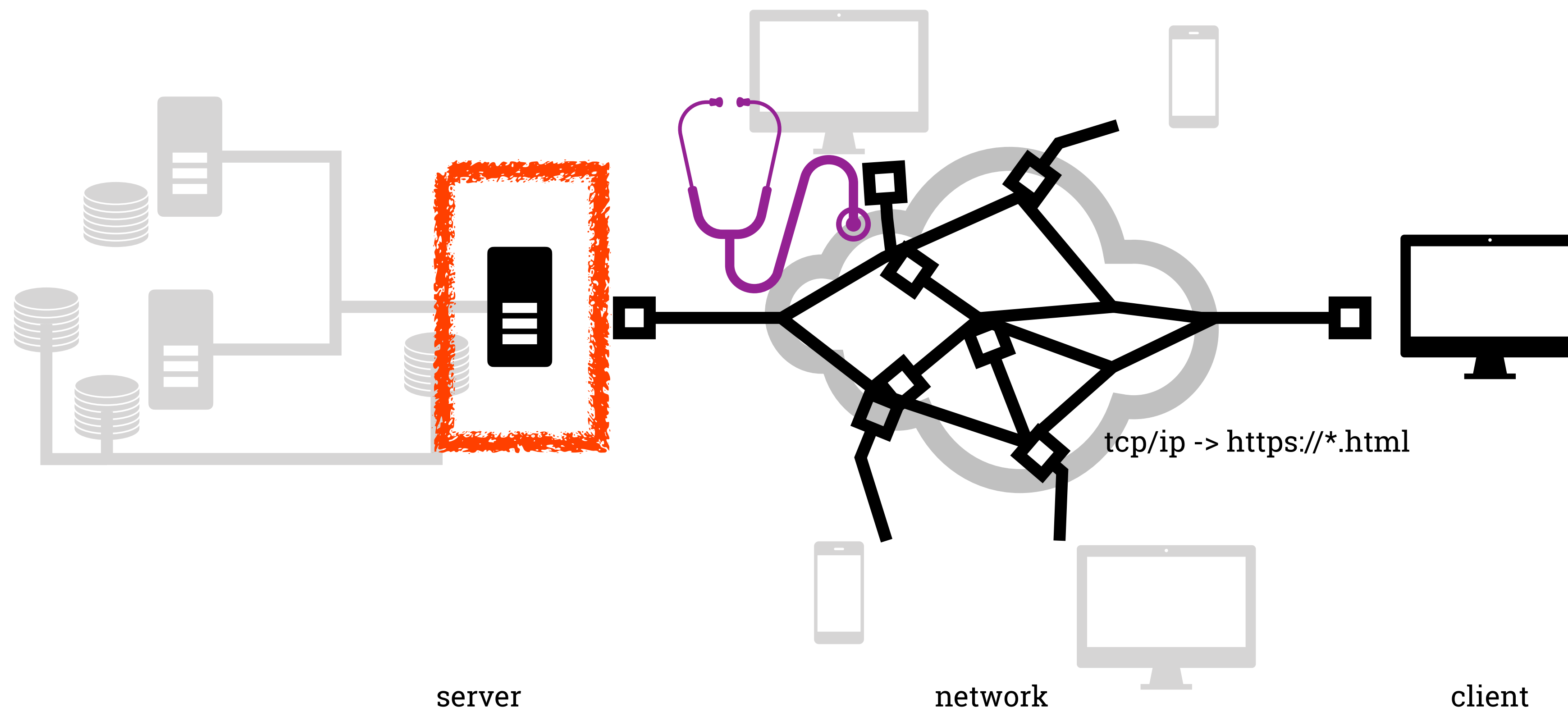


# WEBSERVER



# HTTPS, TLS, SSL

---



# HTML - HYPERTEXT MARKUP LANGUAGE

---

Die Hypertext Markup Language (HTML, englisch für Hypertext-Auszeichnungssprache) ist eine textbasierte Auszeichnungssprache zur Strukturierung elektronischer Dokumente wie Texte mit Hyperlinks, Bildern und anderen Inhalten.

HTML-Dokumente sind die Grundlage des World Wide Web und werden von Webbrowsern dargestellt.

# HTML5

---

Das Paket HTML5 ist eine seit 2010/2014 existierende Bündelung von HTML, CSS und Javascript.

Es bildet die Grundlage für das Umsetzen von beliebigen Applikationsinterfaces.

Für den Browser erscheinen mehr und mehr Bibliotheksobjekte, die sämtliche Funktionalitäten einer Anwendungsoberfläche bereitstellen.

... websockets, local storage, audio/video, 2D/3D Grafik, bluetooth, usb devices ... sind nur einige der neuen Funktionalitäten.



# CONTENT: HTML - HYPERTEXT MARKUP LANGUAGE

---

```
<html>
<head>
  <title>The Document Title</title>
</head>
<body>
<header><span id="logo"></span><nav><ul><li><a href>link</a></li></ul></header> (footer ...)
<main>
  <h1>Headline Ipsum Dolor Sit Amet</h1>
  <p>A paragraph consectetur ad piscit ...</p>
  
  <a href="other-file.html">to the other file</a>
</main>
<aside>
  <form><label>Email</label><input id="my-field" tabindex="1" ...><button type="submit">log in</button></form>
</aside>
  <script src="behavior.js"></script>
</body>
</html>
```

# HTML - HYPERTEXT MARKUP LANGUAGE

---

HTML hat die Aufgabe, eine Seite oder einen Inhalt semantisch zu strukturieren.

Eine Webseite soll kein CSS oder Javascript direkt enthalten, sondern sich den Inhalt samt seiner inhaltlichen Struktur begrenzen.

Dies macht Inhalte maschinenlesbar und durch Programme und assistive Systeme verarbeitbar.

# CSS - CASCADING STYLE SHEETS

---

Cascading Style Sheets (für gestufte Gestaltungsbögen; kurz: CSS) ist eine Stylesheet-Sprache für elektronische Dokumente und zusammen mit HTML und JavaScript eine der Kernsprachen des World Wide Webs.

# CSS - CASCADING STYLE SHEETS

---

CSS schreibt Regeln für die Visualisierung und das dynamisch-visuelle Verhalten von HTML Elementen fest.

Jede sichtbare Eigenschaft besitzt eine CSS Regeln und kann verändert werden

Der Browser selbst benutzt ein rein auf Semantik abzielendes CSS.

Eine Webapplikation kann durch CSS beliebig und in jedem Aspekt umgestaltet werden, ohne das der Inhalt seine Struktur verliert.

U. a. dies macht HTML/CSS zu einem universellen Oberflächenwerkzeug für Applikationen.

# CSS - CASCADING STYLE SHEETS

---

```
/* my-styles.css */  
html * { box-sizing: border-box;}  
body { background-color: white; color: black;}  
.responsive-image { height: auto; max-width: 100%}  
a[href]:hover { color: red; }
```

# HTML + CSS

---

```
<html>

<head>

  <title>The Document Title</title>

  <link type="stylesheet" href="my-styles.css"

</head>

<body>

  <h1>Headline Ipsum Dolor Sit Amet</h1>

  <p>A paragraph consectetur ad psicit ...</p>

  <a href="other-file.html">to the other file</a>

</body>

</html>
```

# JAVASCRIPT

---

JavaScript (kurz JS) ist eine Skriptsprache, die für dynamisches HTML in Webbrowsern entwickelt wurde, um Interaktionen des Benutzers auszuwerten, Inhalte zu verändern, nachzuladen oder zu generieren und so die Möglichkeiten von HTML und CSS zu erweitern.

Heute findet JavaScript auch außerhalb von Browsern Anwendung, so etwa auf Servern und in Microcontrollern.

Javascript ist im Grunde eine im Browser implementierte Bibliothek auf Basis von ECMA Script (ES).

Nodejs ist eine ES-Variante, die als Serversprache implementiert wurde.

Landläufig wird beides Javascript genannt.

# JAVASCRIPT

---

Javascript ist eine im Kern funktionale Sprache, die auf Objekten basiert und Objekte handhabt.

Javascript besitzt eine eigene Objektnotation namens JSON, die sich auch außerhalb von Javascript als Datenformat durchgesetzt hat.

Javascript beherrscht auch Klassen und Instanzenbildung, besitzt aber eine nur schwache Typisierung und keine Sichtbarkeiten.

Javascript wird im Browser kompiliert.



# JAVASCRIPT

---

```
/* my-script.js */  
  
let anchor = document.querySelector('a[href]');  
anchor.addEventListener('click', onMyAnchorClick);  
  
function onMyAnchorClick(event){  
    event.preventDefault();  
    loadData();  
}  
  
function loadData () { ... }
```

# HTML + CSS + JAVASCRIPT

---

```
<html>

<head>

  <title>The Document Title</title>

  <link type="stylesheet" href="my-styles.css"

</head>

<body>

  <h1>Headline Ipsum Dolor Sit Amet</h1>

  <p>A paragraph consectetur ad psicit ...</p>

  <a href="other-file.html">to the other file</a>

  <script src="any-library-or-framework.js">

  <script src="my-script.js">

</body>

</html>
```

# DHTML - DYNAMIC HTML

---

Die Begriffe DHTML (englisch dynamic HTML) oder auch DOM-Scripting bezeichnen bestimmte Webdesign-Methoden, bei denen während der Anzeige einer Webseite diese selbst, ausgelöst durch Benutzereingaben, verändert wird.

Der Begriff „dynamisch“ bezieht sich dabei auf die Idee, dass diese Änderungen von Ereignissen bedingt werden, die auch mehrfach beim Anzeigen einer Seite auftreten können.

Der Begriff DHTML ist veraltet, obwohl das Konzept immer noch gilt und die Basis moderner Webapplikationen bildet.

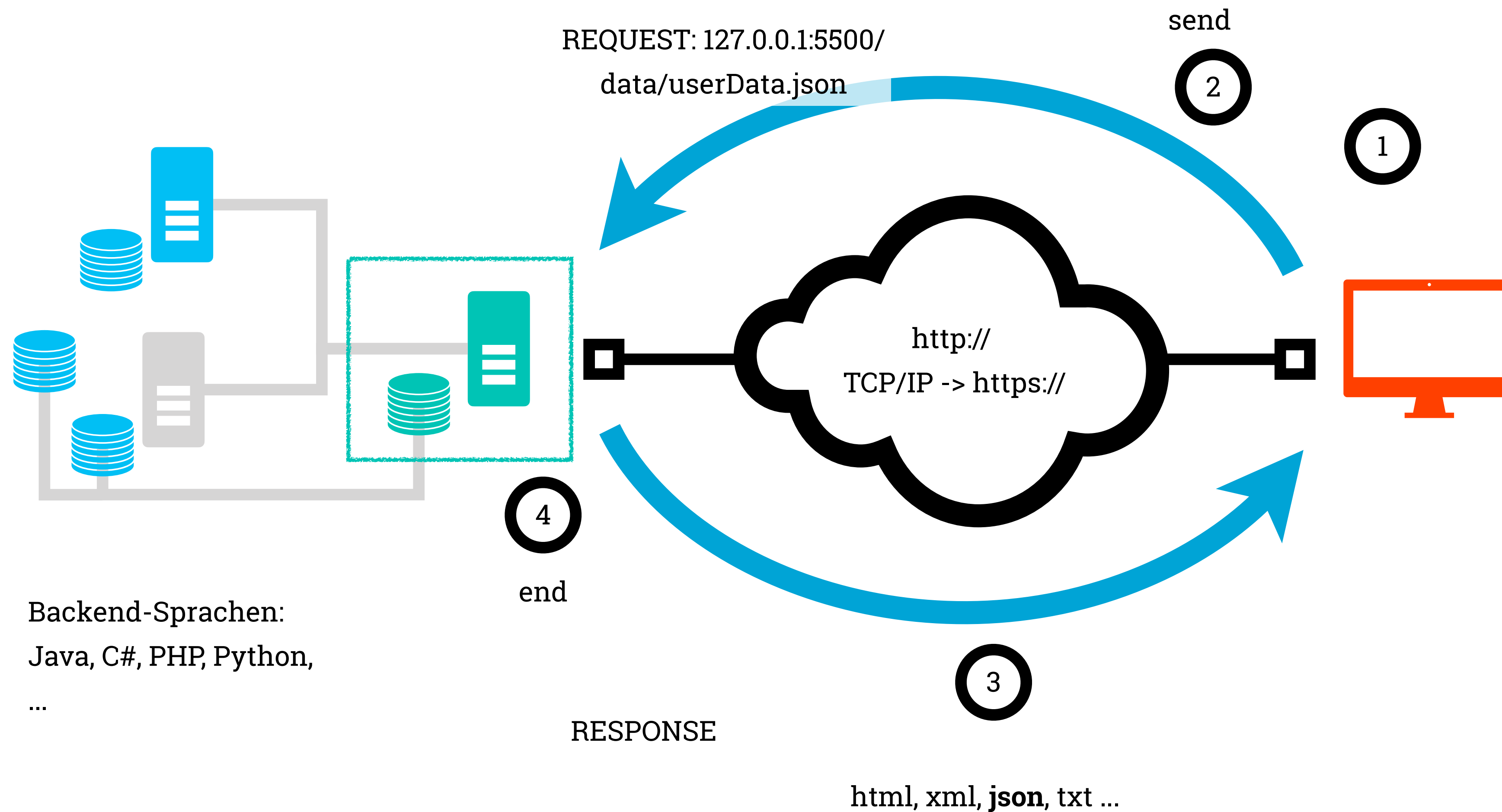
# AJAX - ASYNCHRONOUS JAVASCRIPT AND XML

---

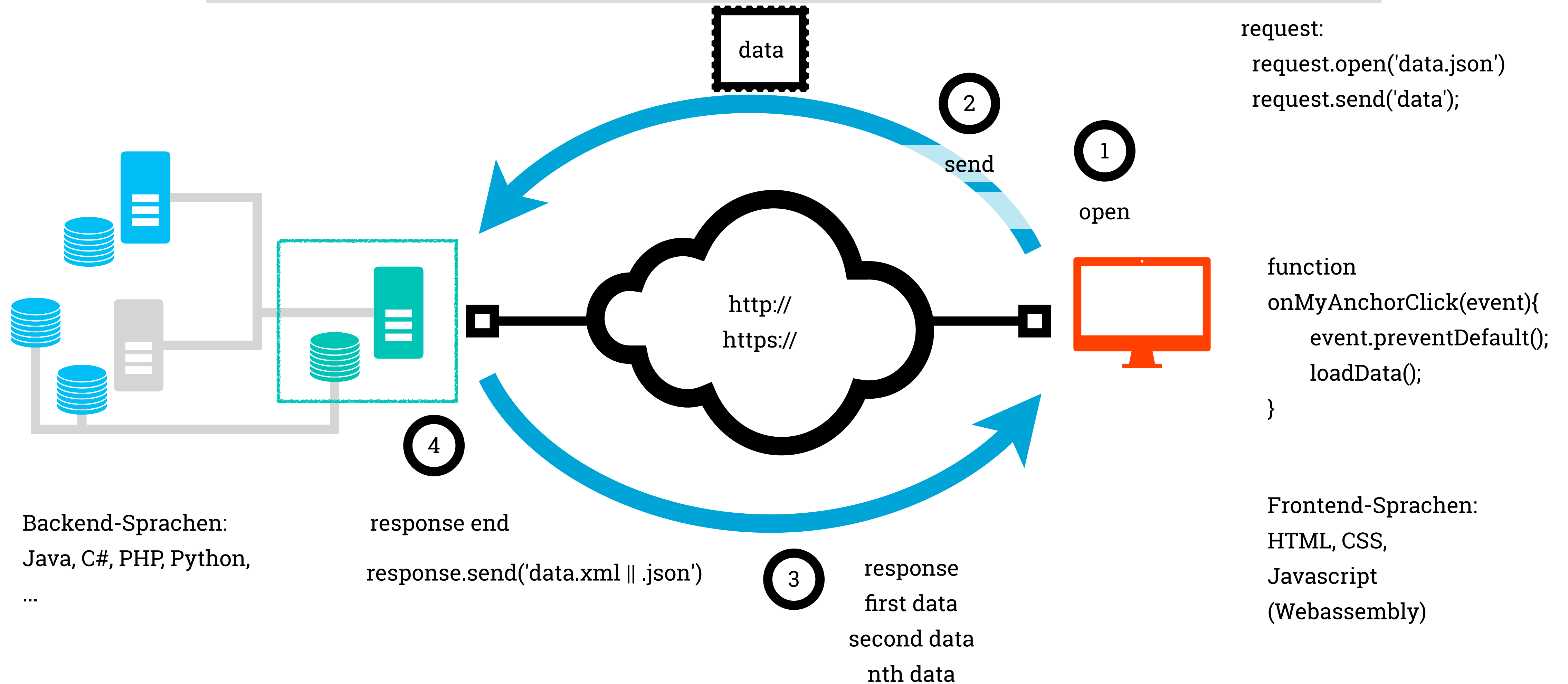
AJAX (Asynchronous JavaScript and XML) bezeichnet ein Konzept der asynchronen **Datenübertragung** per Javascript zwischen einem Browser und dem Server.

Es ermöglicht es, HTTP-Anfragen durchzuführen, während eine HTML-Seite angezeigt wird, und die Seite zu verändern, ohne sie komplett neu zu laden.

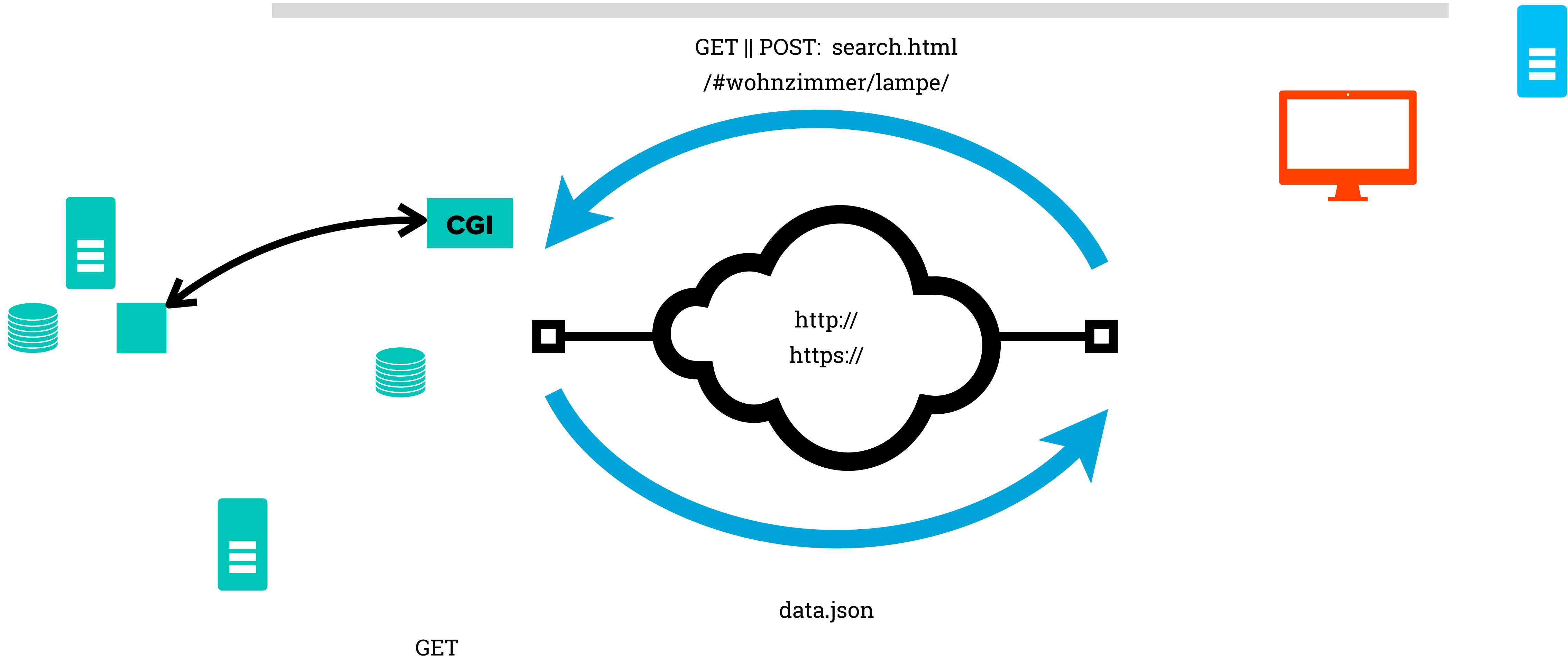
# ASYNCHRONOUS REQUEST - RESPONSE PHASEN



# ASYNCHRONOUS REQUEST - RESPONSE PHASEN



# ASYNCHRONOUS REQUEST - RESPONSE PHASES



# ASYNCHRONER REQUEST (XHR)

old-fashioned style

---

```
xhr = new XMLHttpRequest();
xhr.addEventListener('readystatechange', onReadyStateChange);
xhr.open("GET", 'data.xml'); // Request
xhr.send();

function onReadyStateChange() {
    switch (xhr.readyState) {
        case 0: console.log('there is no request'); break;
        case 1: console.log('request opened'); break;
        case 2: console.log('request sent');    break;
        case 3: console.log('response first part ...'); break;
        case 4: console.log('response more parts and finished!'); break;
    }
}

document.createElement('p') -> p.addChild('data')
```

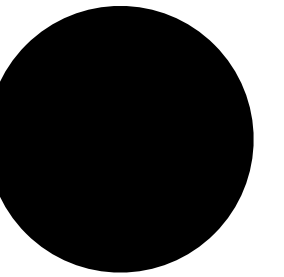


# ASYNCHRONER REQUEST MIT EINEM PROMISE

ES6+

---

```
fetch( 'http://domain.de/data.json' )  
  .then(function (response) {  
    if (response.ok)  
      return response.json();  
    else  
      throw new Error('Daten konnten nicht geladen werden');  
  })  
  .then(function (json) { // Code zum Verarbeiten der  
Daten })  
  .catch(function (err) { // Hier Fehlerbehandlung });
```



# WEBSOCKET - ECHTZEITDATEN

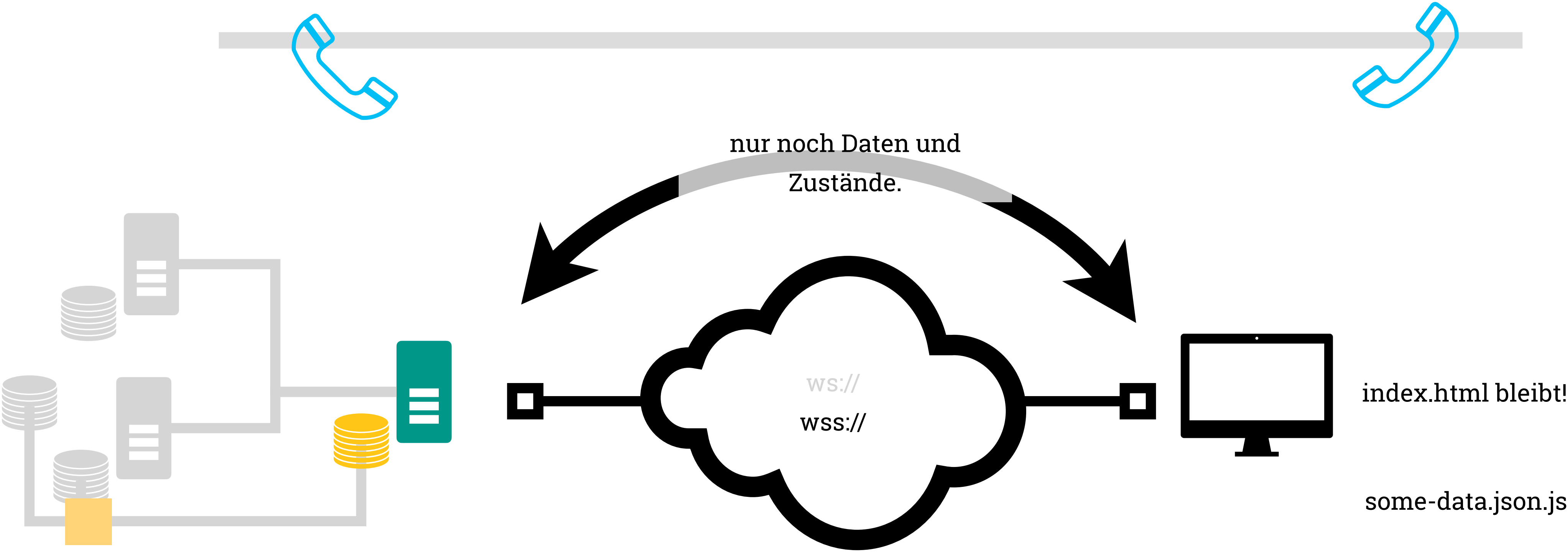
---

Über Websockets kann eine Echtzeitverbindung zwischen Client und Server aufgebaut werden.

Eine WebSocketverbindung ist konsistent und verzichtet auf die Einschränkungen von http.

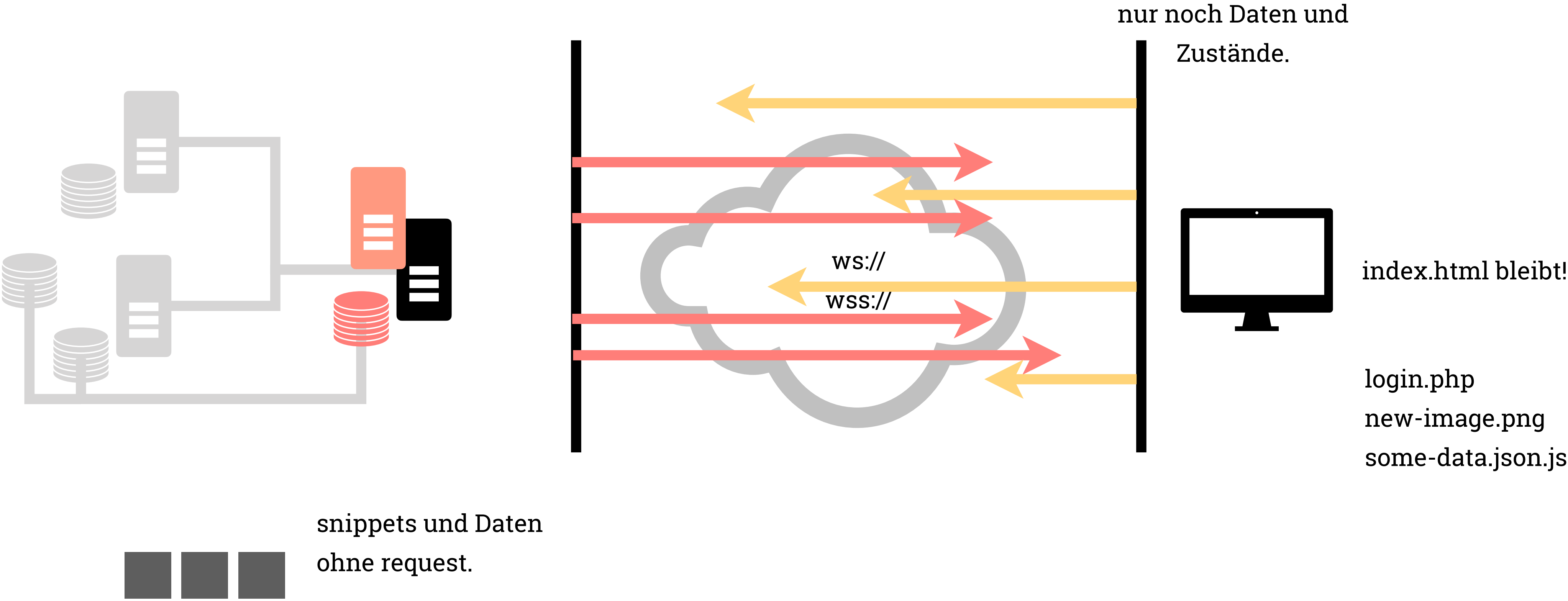
Über das Protokoll ws:// (wss://) können Daten beliebig hin und her geschickt werden.

# WEBSOCKET VERBINDUNGEN



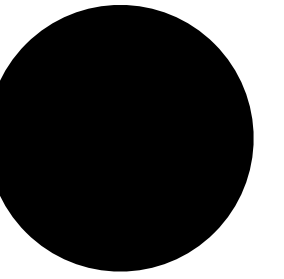
■ ■ snippets und Daten  
ohne request.

# WEBSOCKET VERBINDUNGEN



# SPA - SINGLE PAGE APPLICATION

---

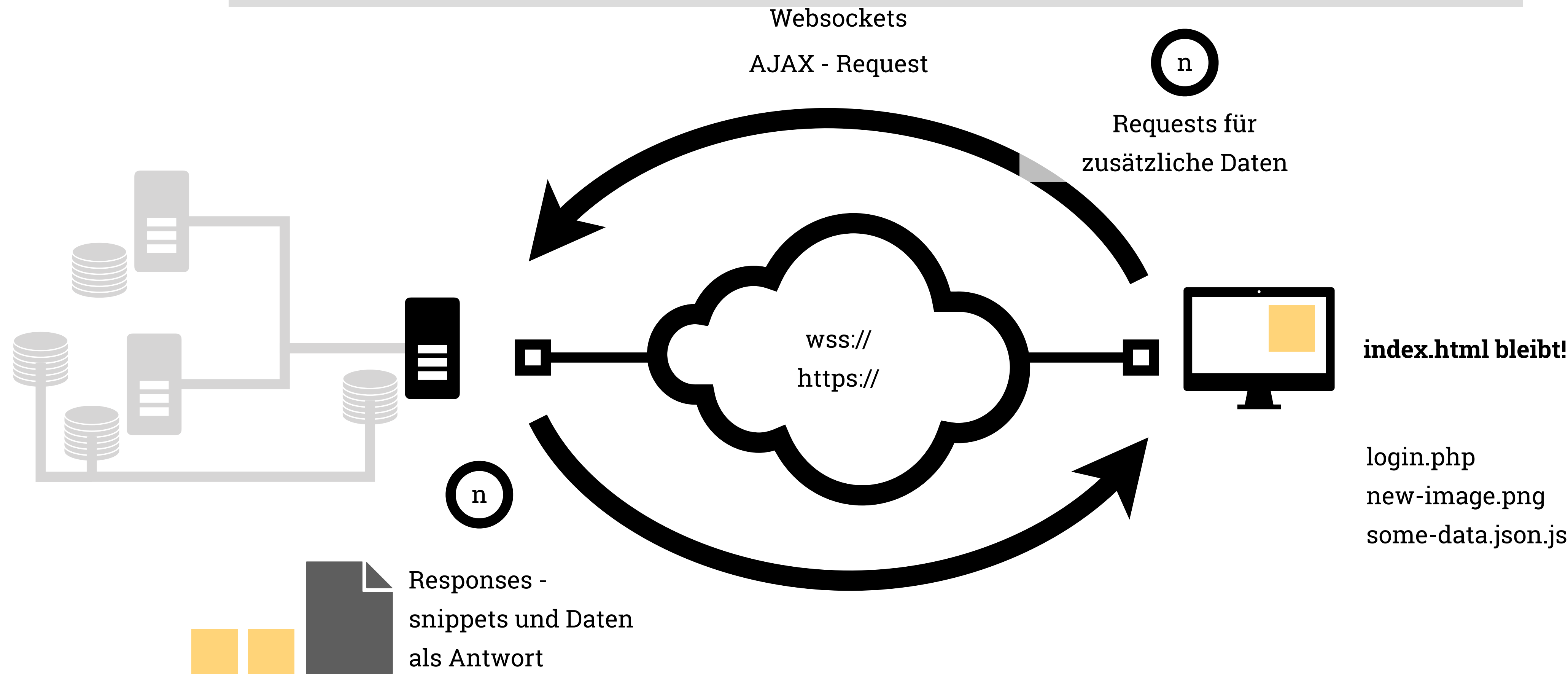


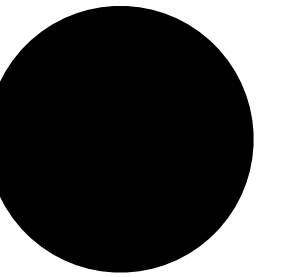
Sie besteht aus **einem einzigen HTML-Dokument**. Weitere Inhalte werden dynamisch nachgeladen.

Dies ist die Grundlage für sogenannte Rich-Clients.

Eine verstärkte clientseitige Ausführung der Webanwendung ermöglicht eine Reduzierung der Serverlast sowie die Umsetzung von selbstständigen Webclients, die beispielsweise eine Offline-Unterstützung anbieten.

# SINGLE PAGE APPLICATION





# AMP - ACCELERATED MOBILE PAGES

---

Accelerated Mobile Pages ist ein speziell für die Erstellung von Websites für mobile Endgeräte (v. a. Smartphones) entwickeltes Derivat von HTML.

AMP wurde im Herbst 2015 vom AMP Project unter Federführung Googles herausgebracht.

**Ziel von AMP ist es, Content schneller zu laden.** Dazu werden insbesondere für Bilder und Mediaelemente eigene Tags verwendet, um deren Laufzeitkontrolle zu verbessern.

Eigene JavaScript müssen in WebWorker-Threads laufen.

# AMP - ACCELERATED MOBILE PAGES

---

```
<!doctype html>
```

```
<html amp lang="en">
```

```
<head>
```

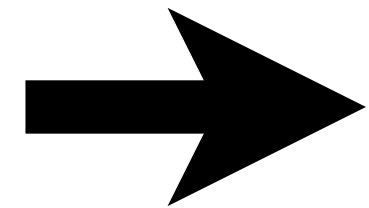
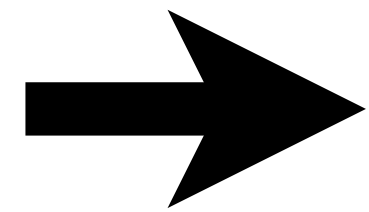
```
  <meta charset="utf-8">
```

```
  <script async src="https://cdn.ampproject.org/v0.js"></script>
```

```
  <title>Hello, AMPs</title>
```

```
  <link rel="canonical" href="https://amp.dev/documentation/
  guides-and-tutorials/start/create/basic_markup/">
```

```
  <meta name="viewport" content="width=device-width,
  minimum-scale=1,initial-scale=1">
```





## AMP - BOILERPLATE

---

```
<script type="application/ld+json">
  {
    "@context": "http://schema.org",
    "@type": "NewsArticle",
    "headline": "Open-source framework for
                publishing content",
    "datePublished": "2015-10-07T12:02:41Z",
    "image": [ "logo.jpg" ]
  }
</script>
```

# AMP - BOILERPLATE

---

```
<style amp-boilerplate>
body{
  animation:-amp-start 8s steps(1,end) 0s 1 normal both}
  @keyframes -amp-start{from{visibility:hidden}
to{visibility:visible}}
</style>

...



</head>
```

# AMP - CONTENT

---

```
<body>
```

```
  <h1>Welcome to the mobile web</h1>
```

```
  <amp-img alt="A view of the sea"
    
    src="/static/inline-examples/images/sea.jpg"
    width="900" height="675" layout="responsive">
    
  </amp-img>
```

```
  <button on="tap:hello.hide">Hide</button>
    
```

```
  <button on="tap:hello.show">Show</button>
    
```

```
  <button on="tap:hello.toggleVisibility">Toggle</button>
    
```

```
</body>
```

```
</html>
```

# AMP - ACCELERATED MOBILE PAGES

---

<https://amp.dev/documentation/>

# PWA - PROGRESSIVE WEB APPS

---

Eine Progressive Web App (PWA) ist eine responsive Webseite mit Funktionen, die (bisher) einer App vorbehalten waren.

**Sie wird HTML5, CSS3 und JavaScript erstellt.** Über Service Worker werden **Offline-Funktionalitäten** via Caching bereitgestellt.

Zur Kommunikation zwischen Webclient und Webserver ist das HTTPS-Protokoll vorgeschrieben.

# PWA - PROGRESSIVE WEB APPS

---

Apps auf dem Smartphone können z. B. auf die Kamera zugreifen, Meldungen auf dem Homescreen platzieren, externe Eingabegeräte wie Gamepads ansprechen und vieles mehr.

HTML5 besitzt (mittlerweile) auch API's für den Zugriff auf Geräte und Komponenten.

Progressive Web Apps können auch den Browser samt Notification-Leiste vollständig ausblenden und die Apps im Vollbildmodus ausführen. So wird der Eindruck einer Applikation erzeugt.

# PWA - WEB-APP-MANIFEST

<https://www.domain.de/manifest.webmanifest>

```
{
  "dir": "ltr",
  "lang": "de-DE",
  "name": "PWA",
  "short_name": "PWA",
  "description": "Eine Beispielanwendung als Progressive Web App",
  "icons": [{
    "src": "img/icon256.png",
    "type": "image/png",
    "sizes": "256x256"
  }],
  "background_color": "white",
  "theme_color": "#f4f4f4",
  "start_url": "app.domainxyz.de",
  "scope": "app.domainxyz.de",
  "display": "standalone",
  "orientation": "portrait-primary"
}
```

# PWA - SOURCECODE

```
<doctype html>

<html>

<head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, minimum-scale=1,initial-scale=1">

    <title>The Document Title</title>

    <link rel="manifest" href="/manifest.webmanifest">
    ████████████████████████████████████████████████████████████████
</head>

<body>

    <h1>Headline Ipsum Dolor Sit Amet</h1>

    <p>A paragraph consectetur ad psicit ...</p>

    <a href="other-file.html">to the other file</a>

</body>

</html>
```



# PWA - TUTORIAL

---

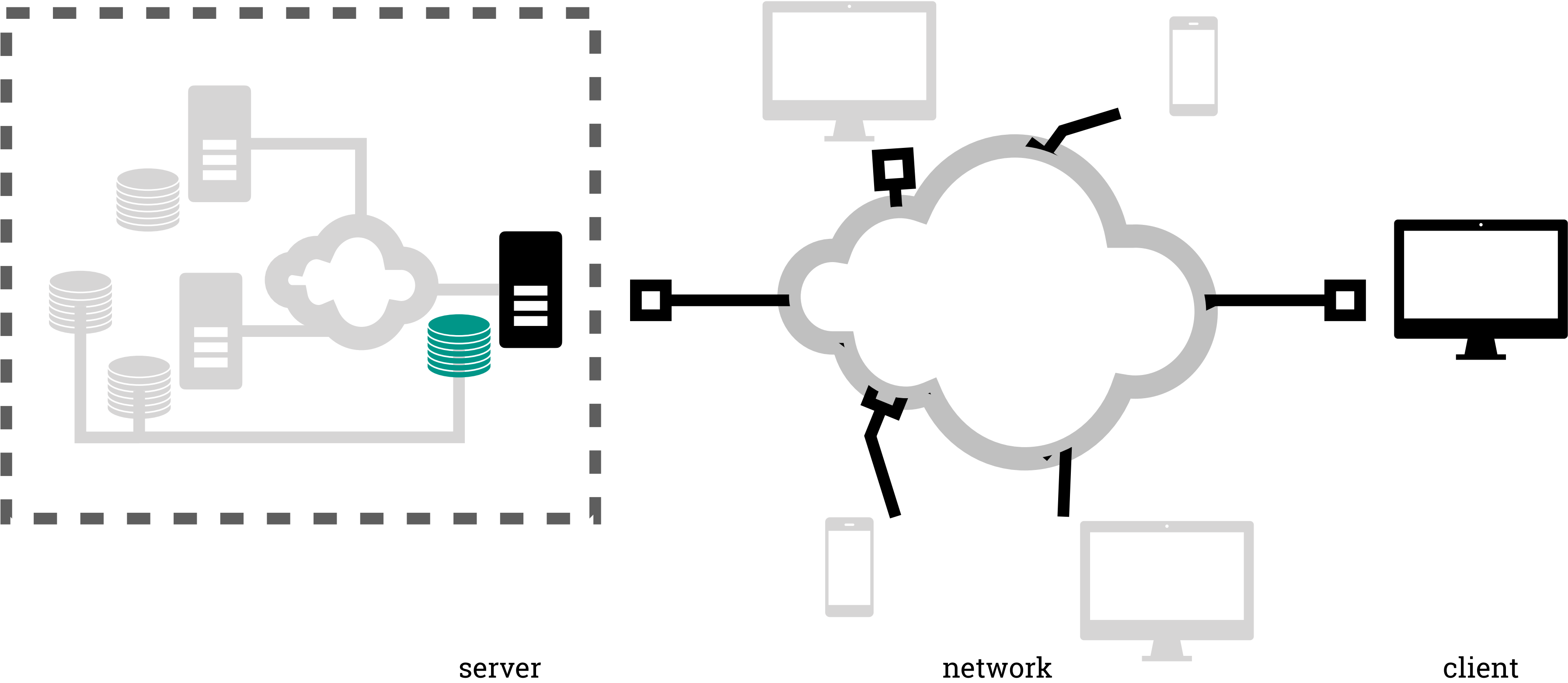
<https://medium.com/james-johnson/a-simple-progressive-web-app-tutorial-f9708e5f2605>

# WEB-SERVER TECHNOLOGIEN

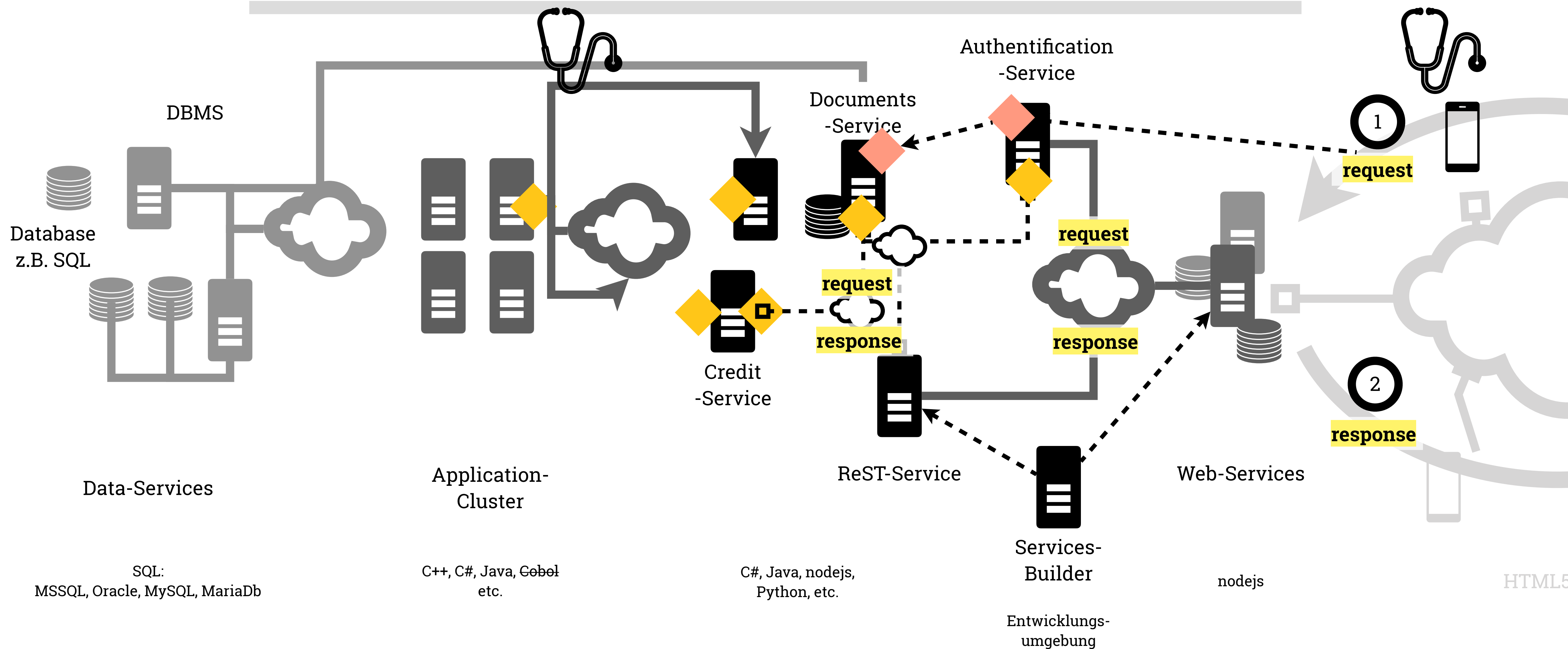
---

- ▶ HTTP, HTTP2, HTTP3
- ▶ Server-Side Processing
- ▶ Programmiertechnologien

# WEBSERVER



# SERVICE - SERVICE - SERVICE



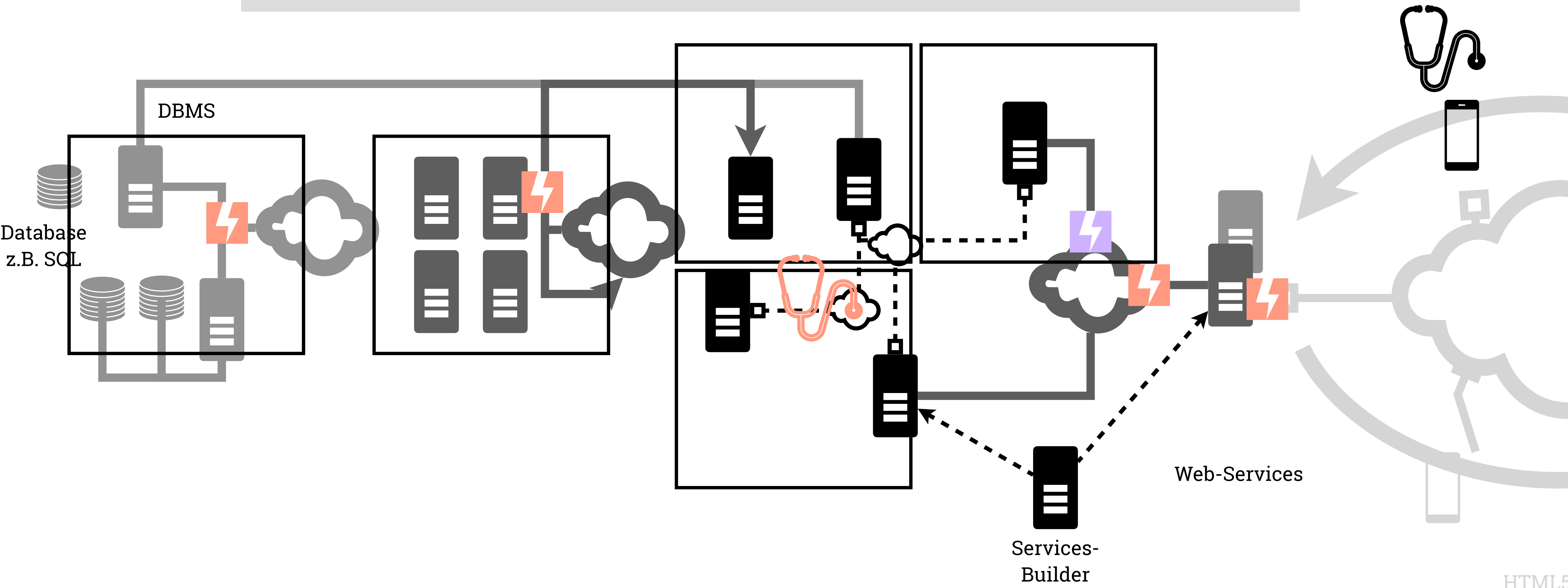
# SERVICE - SERVICE - SERVICE



Firewall



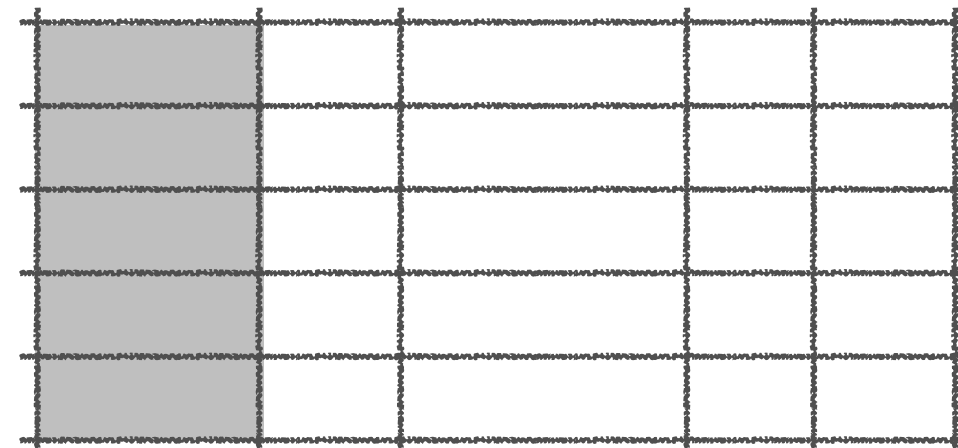
Router



# SQL vs. Not only SQL - noSQL

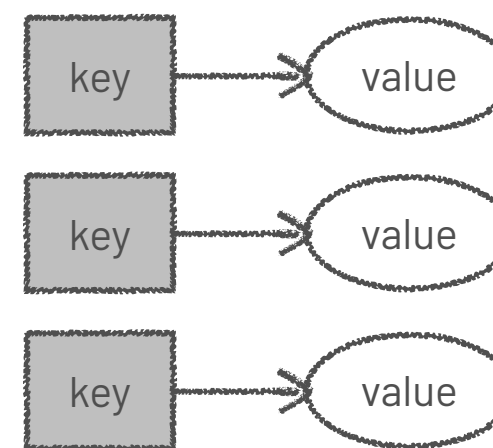
SQL

Relational

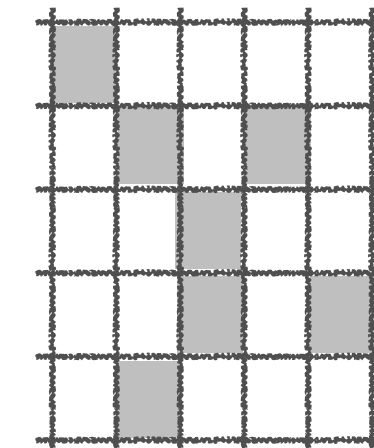


noSQL

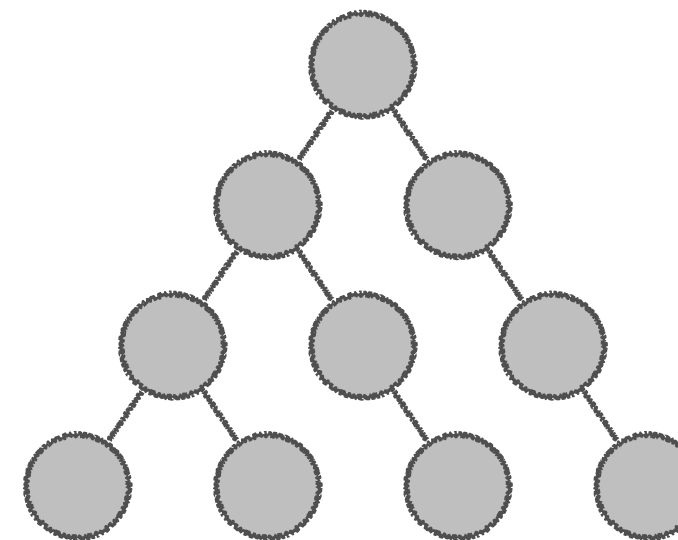
key-value  
(i.e. localStorage)



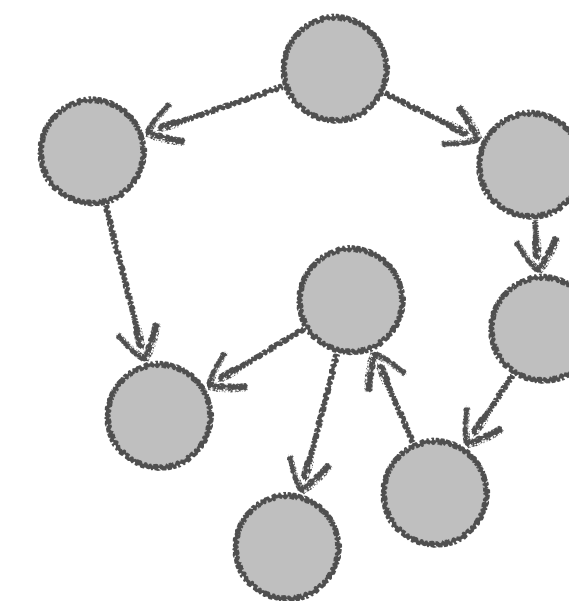
wide-column,  
i.e. Azure tables



**document store,  
i. e. Mongoose**



graph,  
i.e. Neo4j



# HTTP - HYPERTEXT TRANSFER PROTOCOL

---

Das Hypertext Transfer Protocol (HTTP, englisch für Hypertext-Übertragungsprotokoll) ist ein zustandsloses Protokoll zur Übertragung von Daten auf der Anwendungsschicht über ein Rechnernetz. Es wird hauptsächlich eingesetzt, um Webseiten (Hypertext-Dokumente) aus dem World Wide Web (WWW) in einen Webbrowser zu laden. Es ist jedoch nicht prinzipiell darauf beschränkt und auch als allgemeines Dateiübertragungsprotokoll sehr verbreitet.

HTTP wurde von der Internet Engineering Task Force (IETF) und dem World Wide Web Consortium (W3C) standardisiert. Aktuelle Version ist HTTP/2, welche als RFC 7540 am 15. Mai 2015 veröffentlicht wurde. Die Weiterentwicklung wird von der HTTP-Arbeitsgruppe der IETF (HTTPbis) organisiert. Es gibt zu HTTP ergänzende und darauf aufbauende Standards wie HTTPS für die Verschlüsselung übertragener Inhalte oder das Übertragungsprotokoll WebDAV.

# HTTP/1.0

---

Die im Mai 1996 als RFC 1945 (Request for Comments Nr. 1945) publizierte Anforderung ist als HTTP/1.0 bekannt geworden. Bei HTTP/1.0 wird vor jeder Anfrage eine neue TCP-Verbindung aufgebaut und nach Übertragung der Antwort standardmäßig vom Server wieder geschlossen. Sind in ein HTML-Dokument beispielsweise zehn Bilder eingebettet, so werden insgesamt elf TCP-Verbindungen benötigt, um die Seite auf einem grafikfähigen Browser aufzubauen.



# HTTP/1.1

---

1999 wurde eine zweite Anforderung als RFC 2616 publiziert, die den HTTP/1.1-Standard wiedergibt.[2] **Bei HTTP/1.1 kann ein Client durch einen zusätzlichen Headereintrag (Keepalive) den Wunsch äußern, keinen Verbindungsabbau durchzuführen**, um die Verbindung erneut nutzen zu können (persistent connection). Die Unterstützung auf Serverseite ist jedoch optional und kann in Verbindung mit Proxys Probleme bereiten. Mittels HTTP-Pipelining können in der Version 1.1 mehrere Anfragen und Antworten pro TCP-Verbindung gesendet werden. Für das HTML-Dokument mit zehn Bildern wird so nur eine TCP-Verbindung benötigt. Da die Geschwindigkeit von TCP-Verbindungen zu Beginn durch Verwendung des Slow-Start-Algorithmus recht gering ist, wird so die Ladezeit für die gesamte Seite signifikant verkürzt. Zusätzlich können bei HTTP/1.1 abgebrochene Übertragungen fortgesetzt werden.

Eine Möglichkeit zum Einsatz von HTTP/1.1 in Chats ist die Verwendung des MIME-Typs multipart/replace, bei dem der Browser nach Senden eines Boundary-Codes und eines neuerlichen Content-Length-Headerfeldes sowie eines neuen Content-Type-Headerfeldes den Inhalt des Browserfensters neu aufbaut.

Mit HTTP/1.1 ist es neben dem Abholen von Daten auch möglich, Daten zum Server zu übertragen. Mit Hilfe der PUT-Methode können so Webdesigner ihre Seiten direkt über den Webserver per WebDAV publizieren und mit der DELETE-Methode ist es ihnen möglich, Daten vom Server zu löschen. Außerdem bietet HTTP/1.1 eine TRACE-Methode, mit der der Weg zum Webserver verfolgt und überprüft werden kann, ob die Daten korrekt dorthin übertragen werden. Damit ergibt sich die Möglichkeit, den Weg zum Webserver über die verschiedenen Proxys hinweg zu ermitteln, ein traceroute auf Anwendungsebene.

Aufgrund von Unstimmigkeiten und Unklarheiten wurde 2007 eine Arbeitsgruppe gestartet, die die Spezifikation verbessern sollte. Ziel war hier lediglich eine klarere Formulierung, neue Funktionen wurden nicht eingebaut. Dieser Prozess wurde 2014 beendet und führte zu sechs neuen RFCs:

RFC 7230 – HTTP/1.1: Message Syntax and Routing

RFC 7231 – HTTP/1.1: Semantics and Content

RFC 7232 – HTTP/1.1: Conditional Requests

RFC 7233 – HTTP/1.1: Range Requests

RFC 7234 – HTTP/1.1: Caching

RFC 7235 – HTTP/1.1: Authentication

# HTTP/2

---

Im Mai 2015 wurde von der IETF HTTP/2 als Nachfolger von HTTP/1.1 verabschiedet. Der Standard ist durch RFC 7540 und RFC 7541 spezifiziert.[3] Die Entwicklung war maßgeblich von Google (SPDY) und Microsoft (HTTP Speed+Mobility)[4] mit jeweils eigenen Vorschlägen vorangetrieben worden. Ein erster Entwurf, der sich weitgehend an SPDY anlehnte, war im November 2012 publiziert und seither in mehreren Schritten angepasst worden.

Mit HTTP/2 soll die Übertragung beschleunigt und optimiert werden. [5] Dabei soll der neue Standard jedoch vollständig abwärtskompatibel zu HTTP/1.1 sein.

Wichtige neue Möglichkeiten sind  
die Möglichkeit des Zusammenfassens mehrerer Anfragen,  
weitergehende Datenkompressionsmöglichkeiten,  
die binär kodierte Übertragung von Inhalten und  
Server-initiierte Datenübertragungen (push-Verfahren),  
einzelne Streams lassen sich priorisieren.

Eine Beschleunigung ergibt sich hauptsächlich aus der neuen Möglichkeit des Zusammenfassens (Multiplex) mehrerer Anfragen, um sie über eine Verbindung abwickeln zu können.

Die Datenkompression kann nun (mittels neuem Spezialalgorithmus namens HPACK[6]) auch Kopfdaten mit einschließen. Inhalte können binär kodiert übertragen werden. Um nicht auf serverseitig vorhersehbare Folgeanforderungen vom Client warten zu müssen, können Datenübertragungen teilweise vom Server initiiert werden (push-Verfahren). Durch die Verwendung von HTTP/2 können Webseitenbetreiber die Latenz zwischen Client und Server verringern und die Netzwerkhardware entlasten.[7]

Die ursprünglich geplante Option, dass HTTP/2 standardmäßig TLS nutzt, wurde nicht umgesetzt. Allerdings kündigten die Browser-Hersteller Google und Mozilla an, dass ihre Webbrowser nur verschlüsselte HTTP/2-Verbindungen unterstützen werden. Dafür ist ALPN eine Verschlüsselungs-Erweiterung, die TLSv1.2 oder neuer braucht[8].

HTTP/2 wird von den meisten Browsern unterstützt, darunter Google Chrome (inkl. Chrome unter iOS und Android) ab Version 41, Mozilla Firefox ab Version 36,[9] Internet Explorer 11 unter Windows 10, Opera ab Version 28 (sowie Opera Mobile ab Version 24) und Safari ab Version 8.

# HTTP/3

Im November 2018 wurde von der IETF beschlossen, dass aus Googles "HTTP-over-QUIC" HTTP/3 werden solle.[10]

## GET

ist die gebräuchlichste Methode. Mit ihr wird eine Ressource (zum Beispiel eine Datei) unter Angabe eines URI vom Server angefordert. Als Argumente in dem URI können auch Inhalte zum Server übertragen werden, allerdings soll laut Standard eine GET-Anfrage nur Daten abrufen und sonst keine Auswirkungen haben (wie Datenänderungen auf dem Server oder ausloggen). (siehe unten)

## POST

schickt unbegrenzte, je nach physischer Ausstattung des eingesetzten Servers, Mengen an Daten zur weiteren Verarbeitung zum Server, diese werden als Inhalt der Nachricht übertragen und können beispielsweise aus Name-Wert-Paaren bestehen, die aus einem HTML-Formular stammen. Es können so neue Ressourcen auf dem Server entstehen oder bestehende modifiziert werden. POST-Daten werden im Allgemeinen nicht von Caches zwischengespeichert. Zusätzlich können bei dieser Art der Übermittlung auch Daten wie in der GET-Methode an den URI gehängt werden. (siehe unten)

## HEAD

weist den Server an, die gleichen HTTP-Header wie bei GET, nicht jedoch den Nachrichtenrumpf mit dem eigentlichen Dokumentinhalt zu senden. So kann zum Beispiel schnell die Gültigkeit einer Datei im Browser-Cache geprüft werden.

## PUT

dient dazu, eine Ressource (zum Beispiel eine Datei) unter Angabe des Ziel-URIs auf einen Webserver hochzuladen. Besteht unter der angegebenen Ziel-URI bereits eine Ressource, wird diese ersetzt, ansonsten neu erstellt.

## PATCH

Ändert ein bestehendes Dokument ohne dieses wie bei PUT vollständig zu ersetzen. Wurde durch RFC 5789 spezifiziert.

## DELETE

löscht die angegebene Ressource auf dem Server.

## TRACE

liefert die Anfrage so zurück, wie der Server sie empfangen hat. So kann überprüft werden, ob und wie die Anfrage auf dem Weg zum Server verändert worden ist – sinnvoll für das Debugging von Verbindungen.

## OPTIONS

liefert eine Liste der vom Server unterstützten Methoden und Merkmale.

## CONNECT

wird von Proxyservern implementiert, die in der Lage sind, SSL-Tunnel zur Verfügung zu stellen.

RESTful Web Services verwenden die unterschiedlichen Anfragemethoden zur Realisierung von Webservices. Insbesondere werden dafür die HTTP-Anfragemethoden GET, POST, PUT/PATCH und DELETE verwendet.

WebDAV fügt die

Methoden PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK und UNLOCK zu HTTP hinzu.

# DATENAUSTAUSCH TECHNOLOGIEN

---

- ▶ RDF
- ▶ XML
- ▶ JSON
- ▶ ODATA

# RDF - RESOURCE DESCRIPTION FRAMEWORK

---

Das Resource Description Framework formuliert logische Aussagen über beliebige Dinge (Ressourcen).

Ursprünglich wurde RDF vom World Wide Web Consortium (W3C) als Standard zur Beschreibung von Metadaten konzipiert.

## RDFA - RDF IN ATTRIBUTEN IM HTML KONTEXT

---

```
<article vocab="http://schema.org/" typeof="Product">
  <p>Kaufen Sie den
    <span property="name">Staubsauger XF704</span>
    jetzt im Sonderangebot!
    
  </p>
</article>
```

# RDF - RESOURCE DESCRIPTION FRAMEWORK

---

<https://www.w3.org/RDF/>

# XML - EXTENSIBLE MARKUP LANGUAGE

---

XML ist eine **Auszeichnungssprache** zur Darstellung hierarchisch strukturierter Daten im Format einer Textdatei, die sowohl von Menschen als auch von Maschinen lesbar ist.

XML wird als **Austauschformat von Daten** zwischen unterschiedlichen Systemen eingesetzt.

In XML können durch eine Document Type Definition (DTD) oder über ein XML Schema beliebige Datenstrukturen festgelegt werden. Daraus ergeben sich "XML-Sprachen".

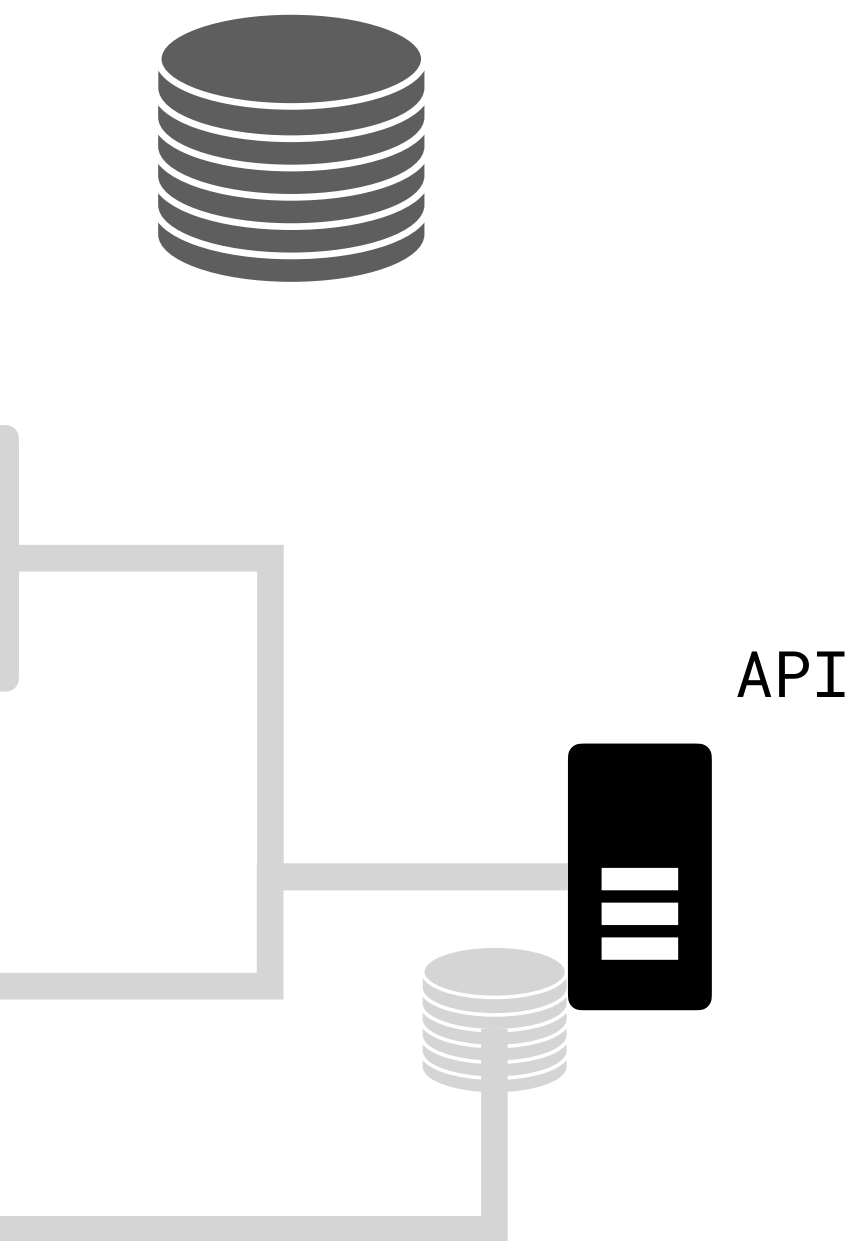
Beispiel dafür sind: RSS, MathML, GraphML, XHTML, XAML, Scalable Vector Graphics (SVG), GPX, aber auch das XML-Schema selbst.



# XML - BEISPIEL



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Kaufmannsladen>
  <Waren>
    <Suesswaren>
      <Lolli preis="5,00" gewicht="100g" />
      <Zuckerstange preis="2,00" />
    </Suesswaren>
    <Weisswaren>
      <Milch />
      <Joghurt />
    </Weisswaren>
  </Waren>
  <Personal>
    <Mitarbeiter id ="2">
      <Name>Dobermann</Name>
      <Anschrift>Beispielweg</Anschrift>
      <Telefon>515248</Telefon>
    </Mitarbeiter>
    <Mitarbeiter id="3">
      <Name>Meier</Name>
      <Anschrift>Knutallee</Anschrift>
      <Telefon>22222</Telefon>
    </Mitarbeiter>
  </Personal>
</Kaufmannsladen>
```



# XML - EXTENSIBLE MARKUP LANGUAGE

---

[https://wiki.selfhtml.org/wiki/XML/Regeln/  
Tags,\\_Attribute,\\_Wertzuweisungen\\_und\\_Ko  
mmentare](https://wiki.selfhtml.org/wiki/XML/Regeln/Tags,_Attribute,_Wertzuweisungen_und_Kommentare)

# JSON - JAVASCRIPT OBJECT NOTATION

---

Die JavaScript Object Notation ist ein kompaktes Datenformat in einer einfach lesbaren Textform und dient dem Zweck des Datenaustausches zwischen Anwendungen.

JSON ist von der Programmiersprache unabhängig. Parser und Generatoren existieren in allen verbreiteten Sprachen.

JSON wurde ursprünglich von Douglas Crockford spezifiziert. Stand Ende 2017 wird es durch zwei unterschiedliche, inhaltlich aber gleiche, Standards spezifiziert – RFC 8259[1] sowie ECMA-404.[2]

# JSON - BEISPIEL



```
{
  "name": "Daniel",      <- key value pairs
  "alter": 47,
  "verheiratet": false,
  "rights": "login, delete",

  "kinder": [
    {
      "name": "Lukas",
      "alter": 19,
      "schulabschluss": "Gymnasium"
    },
    {
      "name": "Lisa",
      "alter": 14,
      "schulabschluss": null
    }
  ]
}
```

# JSON - JAVASCRIPT OBJECT NOTATION

---

<https://www.json.org/json-de.html>

# ODATA - OPEN DATA PROTOCOL

---

Das Open Data Protocol (OData) ist ein von Microsoft veröffentlichtes HTTP-basiertes Protokoll für den Datenzugriff zwischen Systemen, um in diesen CRUD-Operationen zu ermöglichen.

Aufbauend auf älteren Protokollen wie ODBC und JDBC kann OData u. a. innerhalb von Cloud-Diensten, wie Azure, oder MySQL, Java und Rails eingebunden werden und ist in der Lage, in der Client-Server-Kommunikation eine einheitliche Semantik für den Datenaustausch zur Verfügung zu stellen.

„Das Open Data Protocol (OData) ermöglicht das Erstellen von **ReST-basierten Datendiensten**, welche es erlauben, Ressourcen, die über Uniform Resource Identifiers (URIs) identifiziert werden und in einem Datenmodell definiert sind, mittels der Verwendung von HTTP-Nachrichten durch Web-Clients zu veröffentlichen und zu bearbeiten.“

# ODATA - OPEN DATA PROTOCOL

---

<https://www.odata.org/>

# SYSTEMINTEGRATION TECHNOLOGIEN

---

- ▶ SOA
- ▶ Webservice
- ▶ ReST
- ▶ GraphQL
- ▶ WOA
- ▶ MSA



# SOA - SERVICE-ORIENTED ARCHITECTURE

---

Serviceorientierte Architektur, auch dienstorientierte Architektur, ist ein Architekturmuster der Informationstechnik aus dem Bereich der verteilten Systeme, um **Dienste** von IT-Systemen zu strukturieren und zu nutzen.

Eine besondere Rolle spielt dabei die Orientierung an Geschäftsprozessen, deren Abstraktionsebenen die Grundlage für konkrete Serviceimplementierungen sind: „Vergib einen Kredit“ ist beispielsweise auf einer hohen Ebene angesiedelt, dahinter verbirgt sich bei einem Bankunternehmen ein Geschäftsprozess mit mehreren beteiligten Personen und informationstechnischen Systemen („Eröffnen der Geschäftsbeziehung“,

„Eröffnen eines oder mehrerer Konten“, „Kreditvertrag...“ und so weiter), während „Trage den Kunden ins Kundenverzeichnis ein“ ein Dienst auf einer niedrigeren Ebene ist.

Durch Zusammensetzen (Orchestrierung) von Services niedriger Abstraktionsebenen können so recht flexibel und unter Ermöglichung größtmöglicher Wiederverwendbarkeit Services höherer Abstraktionsebenen geschaffen werden.

# MICROSERVICES

---

Microservices sind ein Architekturmuster der Informationstechnik, bei dem komplexe Anwendungssoftware aus unabhängigen Prozessen generiert wird, die untereinander mit sprachunabhängigen Programmierschnittstellen kommunizieren.

Die Dienste sind weitgehend entkoppelt und erledigen eine kleine Aufgabe. So ermöglichen sie einen modularen Aufbau von Anwendungssoftware.

Microservices werden gegenüber anderen Services isoliert.

Die Schnittstellen verstecken Implementierungsdetails.

Es werden dabei bevorzugt Standardverfahren mit geringem Overhead, wie REST, eingesetzt.

Es sollte nicht ersichtlich sein, mit welcher Architektur ein Microservice selbst implementiert wurde.

# WEBSERVICES

---

Ein Webservice (auch Webdienst) stellt eine Schnittstelle für die Maschine-zu-Maschine- oder Anwendungs-Kommunikation über Rechnernetze wie das Internet zur Verfügung. Dabei werden Daten ausgetauscht und auf entfernten Computern (Servern) Funktionen aufgerufen.

Jeder **Webservice besitzt einen Uniform Resource Identifier** (URI), über den er eindeutig identifizierbar ist, sowie je nach Implementierung eine Schnittstellenbeschreibung in maschinenlesbarem Format (als XML-Artefakt, z. B. WSDL), die definiert, wie mit dem Webservice zu interagieren ist.

Die Kommunikation kann über Protokolle aus dem Internetkontext wie HTTP oder HTTPS erfolgen; über diese Protokolle wiederum kann beispielsweise XML oder JSON übertragen werden. Ein Webservice ist plattformunabhängig und steht in der Regel mehreren Programmen zum Aufrufen bereit.

# REST - REPRESENTATIONAL STATE TRANSFER

---

Representational State Transfer (ReST) ist ein Paradigma für die Softwarearchitektur von verteilten Systemen, insbesondere für Webservices.

ReST ist eine Abstraktion der Struktur und des Verhaltens des World Wide Web. REST hat das Ziel, einen Architekturstil zu schaffen, der die Anforderungen des modernen Web besser darstellt. Dabei unterscheidet sich ReST vor allem in der Forderung nach einer einheitlichen Schnittstelle von anderen Architekturstilen.

Der Zweck von ReST liegt schwerpunktmäßig auf der Maschine-zu-Maschine-Kommunikation. ReST stellt eine einfache Alternative zu ähnlichen Verfahren wie SOAP und WSDL und dem verwandten Verfahren RPC dar.

ReST nutzt die im WWW vorhandene Infrastruktur (z. B. Web- und Application-Server, HTTP-fähige Clients, HTML- und XML-Parser, Sicherheitsmechanismen). So sind viele Web-Dienste per se ReST-konform.

Die Bezeichnung „Representational State Transfer“ soll den Übergang vom aktuellen Zustand zum nächsten Zustand (state) einer Applikation verbildlichen. Dieser Zustandsübergang erfolgt durch den Transfer der Daten, die den nächsten Zustand repräsentieren.

# GRAPHQL

---

GraphQL ist eine Open-Source-Datenabfrage- und Manipulationssprache und ein Laufzeitsystem zum Beantworten von Abfragen mit vorhandenen Daten.

Es bietet eine Alternative zu SQL, ganz im Sinne von ReST und Ad-hoc-Webservice-Architekturen.

Als eine zustandslose Abfragesprache ermöglicht es Clients, die genaue Struktur der benötigten Daten zu definieren. Durch Parametrisierung wird vermieden – ganz im Gegensatz zu vielen anderen implementierten ReST-Schnittstellen –, bei jeder Anfrage unnötig große Datenmengen zu übermitteln.

GraphQL unterstützt das Lesen, Schreiben und Abonnieren von Datenänderungen (Echtzeit-Updates).

Zu den wichtigsten GraphQL-Clients gehören Apollo Client und Relay.

GraphQL-Server sind für mehrere Sprachen verfügbar, einschließlich Haskell, JavaScript, Python, Ruby, Java, C#, Scala, Go, Elixir, Erlang, PHP, R und Clojure.

<https://www.ionos.de/digitalguide/websites/web-entwicklung/graphql/>

# WOA - WEB ORIENTED ARCHITECTURE

---

Web Oriented Architecture, ein  
Rahmenkonzept für verteilte, unabhängige  
und via Hyperlinks vernetzte Daten- und  
Prozeßelemente, angelehnt an  
Dienstorientierte Architektur.

# MSA - MOBILE SERVICE ARCHITECTURE

---

Mobile Service Architecture, eine  
Spezifikation der Java-Plattform auf  
mobilen Endgeräten

# SICHERHEIT IN DER WEB-ANWENDUNGSENTWICKLUNG

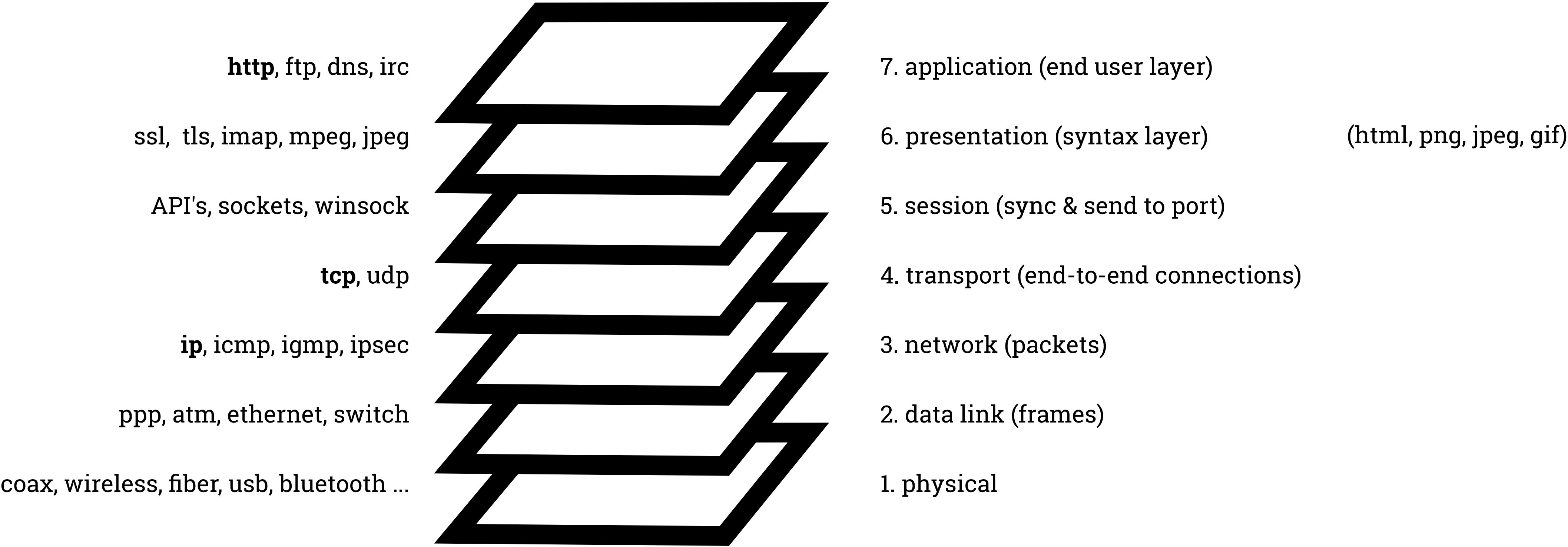
---

- ▶ Grundbegriffe der Sicherheit
- ▶ Authentifizierung und Autorisierung
- ▶ HTTPS / SSL
- ▶ OWASP



# OSI - OPEN SYSTEMS INTERCONNECTION

seven tiers of OSI



# GRUNDBEGRIFFE - SOFTWAREVERSION UND UPDATES

---

Die Entwicklung von Software ist ein Prozess, der prinzipiell kein Ende kennt.  
**Regelmäßige Updates liefern einerseits neue Funktion, schließen andererseits aber Sicherheitslücken, die durch Fehler in der Programmierung entstanden sind.**

Für die IT-Sicherheit ist es von zentraler Bedeutung, Software auf dem aktuellen Stand zu halten und Sicherheitspatches einzuspielen.

Stellt der Hersteller den Support von Software ein und liefert keine Updates mehr aus, ist von der Verwendung der Software abzuraten.

# GRUNDBEGRIFFE - COMMON VULNERABILITIES AND EXPOSURES

---

Um einen Überblick über entdeckte Schwachstellen in Software zu ermöglichen, dient der Industriestandard Common Vulnerabilities and Exposures (CVE).

Die einheitliche Namenskonvention ermöglicht klare Identifizierung der Sicherheitslücken und einen Austausch zwischen verschiedenen Systemen und Institutionen.

Gesammelt und verwaltet wird die Liste aller CVEs von der MITRE Corporation.

# GRUNDBEGRIFFE - ANGRIFFSVEKTOREN

---

Als Angriffsvektor wird eine Technik oder eine Kombination von Methoden bezeichnet, mit der sich ein Angreifer Zugang zu IT-Systemen verschafft oder sie außer Gefecht setzt.

# GRUNDBEGRIFFE DER SICHERHEIT

---

## Exploit

Als Exploit wird das Ausnutzen einer **Sicherheitslücke im Code** einer Software oder der Hardware bezeichnet, um sich Zugang zu einem IT-System zu verschaffen. Zero-Day-Exploits sind Sicherheitslücken, die entdeckt und ausgenutzt werden, bevor der Hersteller mit einem Patch reagieren kann.

# GRUNDBEGRIFFE - SOCIAL ENGINEERING

---

Als **Social Engineering** werden Versuche von Angreifen bezeichnet bei ihren Opfern ein bestimmtes Verhalten auszulösen, damit diese Daten preisgeben und Zugriffe ermöglichen. Dabei setzen Hacker auf menschliche Schwächen wie Neugier, Angst oder Naivität.

# GRUNDBEGRIFFE - PHISHING

---

**Phishing** (eine Wortschöpfung aus „Password“ und „Fishing“) bezeichnet Social Engineering-Methoden, die darauf abzielen, dass das Opfer unwissentlich geheime Zugangsdaten preisgibt. Es lässt sich zum Beispiel durch manipulierte E-Mails und Webseiten umsetzen.

# GRUNDBEGRIFFE - BRUTEFORCE

---

**Bruteforce** bezeichnet allgemein das Lösen von Rechenproblemen durch „reine Gewalt“. Konkret heißt das durch massenhaftes Ausprobieren. Bruteforce-Attacken zielen darauf, zufällige oder aus Dictionaries stammende Kombinationen von Benutzernamen und Passwörtern auszuprobieren. Entsprechende Programme können so innerhalb von einer Minute unzählige Versuche starten.



# GRUNDBEGRIFFE - MAN-IN-THE-MIDDLE

---

Eine **Man-in-the-Middle-Attacke** verfolgt das Ziel, sich unbemerkt in Kommunikation von zwei oder mehr Kommunikationsteilnehmern zwischenzuschalten, um Daten abzufangen oder zu manipulieren. Dazu leitet der Angreifer die Verbindungsanfrage des Senders auf sich um und baut dann eine Verbindung zum eigentlichen Empfänger auf. Cyber-Kriminelle können sich so zum Beispiel zwischen den Besucher einer Webseite und den Server schalten und Zahlungsdaten mitlesen oder umleiten.

# GRUNDBEGRIFFE - SQL INJECTION

---

Bei einer **SQL Injection** versucht ein Angreifer eigene Datenbankbefehle in eine SQL-Datenbank einzuschleusen, um Daten auszuspähen oder die Kontrolle über das System zu erlangen. Dazu werden Eingabefelder der Webseite genutzt. Durch proaktive Maßnahmen lässt sich eine SQL Injection verhindern.

# GRUNDBEGRIFFE - CROSS-SITE-SCRIPTING

---

Bei **Cross Site Scripting** (XSS) wird HTML/JavaScript/CSS unvalidiert in eine Webseite eingebettet. So lassen sich Nutzersessions abfischen und zum Beispiel Zahlungsdaten mitschneiden.

# GRUNDBEGRIFFE - DDOS-ATTACKE

---

Ein Distributed Denial of Service (DDoS) verfolgt das Ziel, IT-Systeme **durch Überlastung außer Gefecht** zu setzen.

Dazu kommen meist Botnetze zum Einsatz, die massenhaft einen Server oder eine andere Netzkomponente ansteuern. Die Botnetze bestehen häufig aus wiederum mit einem Trojaner infizierte Computer, deren Besitzer keine Kenntnis darüber besitzen, Teil eines Botnetzes zu sein.

# SCHADSOFTWARE, MALWARE, VIREN, WÜRMER, TROJANER

---

Schadsoftware oder Malware ist der Oberbegriff für alle Programme, die das Ziel verfolgen, schädliche Funktionen auszuführen. Beispiele hierfür sind Viren, Würmer und Trojaner. Als **Computer-Virus** wird Schadsoftware (meist in Form einer .exe) bezeichnet, die sich selbst verbreitet und unterschiedliche Schadfunktionen in sich tragen kann. Ein Wurm ist eine Unterkategorie eines Virus, der die Fähigkeit besitzt, sich über vorhandene Übertragungsfunktionen auszubreiten. Trojaner sind keine Viren. Sie

zielen darauf ab, eine Hintertür auf dem System zu öffnen, um es dadurch fernsteuern zu können. Sie bilden etwa die Grundlage von Bot-Netzen.

# GRUNDBEGRIFFE RANSOMWARE

---

Ransomware ist eine bestimmte Kategorie von Schadsoftware, die das Ziel verfolgt, IT-Systeme lahmzulegen und Zugriffe zu verunmöglichen. Dazu verschlüsseln sie die Festplatten ganzer IT-Infrastrukturen. Die Cyberkriminellen versprechen gegen die Zahlung eines Lösegeldes, den Zugriff wieder freizugeben.

# GRUNDBEGRIFFE - ANTIVIRENSOFTWARE (AV), VIRENSCANNER

---

**Antivirensoftware** erkennt bekannte Schadsoftware, blockiert sie und kann sie gegebenenfalls beseitigen oder isolieren. Virens Scanner sind reaktive Verfahren, die ausschließlich bekannte Malware erkennen können. Dazu lädt die Software in regelmäßigen Abständen aus Malware-Datenbanken die Signaturen der sich neu im Umlauf befindenden Schadsoftware herunter.

# GRUNDBEGRIFFE - FIREWALL



Eine Firewall ist ein Sicherungssystem, das ein Rechnernetz oder einen einzelnen Computer vor unerwünschten Netzwerkzugriffen schützt. Weiter gefasst ist eine Firewall auch ein Teilaspekt eines Sicherheitskonzepts. Jedes Firewall-Sicherungssystem basiert auf einer Softwarekomponente.



# GRUNDBEGRIFFE - FIREWALL UND UTM

---

Eine **Firewall** ist zwischen Server oder Client und Internet geschaltet. Sie entscheidet, ob der Versuch eines Verbindungsaufbaus von außen zulässig ist. Dabei fungiert die Firewall wie eine Ampel, die entweder grün oder rot gibt. Sie schaut dabei aber nicht in die Autos rein. Das heißt: Sie prüft nicht den Inhalt der Datenpakete.

Sogenannte NextGen-Firewalls oder Unified Threat Management (UTM)-Systeme erweitern die Funktionsweise der Firewall. Sie bieten beispielsweise Contentfilter, Spam-Schutz, VPN oder auch eine Deep Packet Inspection.

# GRUNDBEGRIFFE - DEEP PACKET INSPECTION

---

Deep Packet Inspection (DPI) ist ein Verfahren, um Datenpakete zu überwachen und zu prüfen. Neben dem Headerteil des Datenpakets wird auch der Inhalt einer Prüfung unterzogen. So können Netzwerkattacken und unrechtmäßige Datenströme aufgedeckt werden.

Eine Deep Packet Inspection lässt sich an mehreren Stellen in der IT-Infrastruktur platzieren. Entweder zentral als Teil einer erweiterten Firewall (bspw. UTM-System) oder aber dezentral an Switches, Servern und Clients.

# GRUNDBEGRIFFE - VPN UND PROXY

---

Der Einsatz von Virtual Private Networks (VPN) und Proxy-Servern dient der Verschleierung der Identität im Internet. Anstatt eine Zieladresse direkt aufzurufen, wird eine Instanz zwischengeschaltet, die zunächst kontaktiert wird und dann die Zieladresse aufruft.

Neben der Verschleierung der eigenen IP-Adresse kann VPN auch zum Einsatz kommen, um den Zugriff auf bestimmte Dienste einzuschränken. Lässt sich beispielsweise die Administrator-Oberfläche einer Webseite nur von einer IP-Adresse aus aufrufen, wird der Zugang für Hacker stark erschwert.

# GRUNDBEGRIFFE - IT-MONITORING

---

Unter IT-Monitoring werden alle Aktivitäten zusammengefasst, die durch die Überwachung der Systeme einen reibungslosen Betrieb der IT gewährleisten. Auftretende Probleme sollen schnell erkannt werden, um entsprechende Maßnahmen einzuleiten. Im Idealfall erkennt der IT-Verantwortliche Probleme dadurch bereits bevor der Nutzer Kenntnis von ihnen nimmt.

# GRUNDBEGRIFFE - SIEM

---

Security Information and Event Management (SIEM) ist der Oberbegriff für Technologien, die es durch eine Echtzeitanalyse sämtlicher für die IT-Sicherheit relevanter Daten ermöglichen, einen ganzheitlichen Blick auf den Sicherheitszustand der IT-Systeme zu erlangen. Die Daten können darüber hinaus hilfreich sein, einen erfolgreichen Angriff zu rekonstruieren. Dazu sammeln die SIEM-Systeme selbst Daten und/oder aggregieren Daten von anderen Systemen.

# GRUNDBEGRIFFE - RISIKOMANAGEMENT

---

Als Risikomanagement werden alle organisatorischen wie technischen Aktivitäten bezeichnet, um Risiken abzuschätzen, zu steuern und zu kontrollieren. In der IT-Sicherheit spielt ein geeignetes Risikomanagement eine wichtige Rolle, um die richtigen Maßnahmen zur Steigerung des Sicherheitsniveaus erreichen zu können.

# GRUNDBEGRIFFE - INVENTARISIERUNG UND VISUALISIERUNG

---

Unter Inventarisierung wird im IT-Management die Sammlung eingesetzter Systeme verstanden. Dabei kann es sich ebenso um Software wie Hardware handeln. Inventarisierung spielt für die IT-Sicherheit eine große Rolle, da eine vorhandene Übersicht Grundlage für eine Gefahreneinschätzung darstellt. Mit einer Visualisierung der IT-Infrastruktur wird der reinen Sammlung der Daten ein grafisches Element hinzugefügt, sodass Abhängigkeiten und Zusammenhänge sichtbar werden.

# GRUNDBEGRIFFE - PENETRATIONSTEST, PENTEST

---

Ein Penetrationstest (kurz: Pentest) ist ein umfassender Sicherheitstest von IT-Systemen. Dabei nimmt der Tester die Position eines Hackers ein und untersucht die Systeme nach möglichen Einstiegspunkten. Ziel eines Pentestes ist eine Sammlung von Sicherheitslücken, die es zu beheben gilt, um das System zu härten. Neben in Auftrag gegebenen manuellen Pentests lassen sich Penetrationstest auch automatisiert durchführen.



# GRUNDBEGRIFFE - AUTHENTIFIZIERUNG

---

Durch Authentifizierungsverfahren können Systeme und Benutzer gegenüber anderen Systemen ihre Identität nachweisen. Dies wird in der Regel durch ein gemeinsames Geheimnis gelöst, etwa die Kombination von Passwort und Kennwort.

Authentifizierungsverfahren stellen sicher, dass nur berechtigte Personen oder Systeme auf die entsprechenden Daten und Funktionen zugreifen können.

# GRUNDBEGRIFFE - VERSCHLÜSSELUNG, ENCRYPTION

---

Durch den Einsatz von Verschlüsselungsverfahren lassen sich Daten so verändern, dass sie für Unbefugte nicht mehr lesbar sind. Verschlüsselt werden können sowohl Daten, die auf einem Datenspeicher liegen als auch die Kommunikation zwischen zwei Systemen. Um die Verbindung zwischen Webseitenbesucher und Server abzusichern kommt beispielsweise SSL/TLS zum Einsatz.

# GRUNDBEGRIFFE - DATENSCHUTZ-GRUNDVERORDNUNG (DSGVO)

---

Die Datenschutz-Grundverordnung (DSVGO) ist eine Verordnung der Europäischen Union und regelt den Umgang und die Verarbeitung mit personenbezogenen Daten. Sie legt fest, dass Verantwortliche neben organisatorischen Maßnahmen auch technische Maßnahmen ergreifen müssen, um den Schutz der Daten zu gewährleisten. Dazu gehört „ein Verfahren zur regelmäßigen Überprüfung, Bewertung und Evaluierung der Wirksamkeit“ dieser Maßnahmen (DSVGO Art. 32). Ein rechtlich

konformer Datenschutz ist daher ohne IT-Sicherheit nicht umzusetzen.

# AUTHENTISIERUNG , AUTHENTIFIZIERUNG, AUTORISIERUNG

---

## **Schritt 1 – die Authentisierung**

Die Authentisierung bezeichnet das Vorlegen eines Nachweises für die Identität des Nutzers gegenüber dem IT-System oder der IT-Ressource, an der er sich anmelden möchte.

Dieser Nachweis kann beispielsweise durch eine Information, die nur der Nutzer kennt (Passwort, PIN), etwas, was er ist (Fingerabdruck, Iris-Scanner), etwas, das er besitzt (Smartcard, Token, Badge) oder einer Kombination der Genannten.

## **Schritt 2 – die Authentifizierung**

Die Authentifizierung ist die Überprüfung der Identität samt dem Ergebnis. Das System gleicht die hinterlegten Informationen mit der behaupteten Identität ab. Alternativ kann eine dritte, autorisierte Stelle befragt.

## **Schritt 3 – die Autorisierung**

Dann erfolgt die „Autorisierung“. Damit werden „bestimmte Rechte“ erteilt.

# AAA - AM BANKAUTOMAT

---

Ein klassisches Beispiel hierfür ist das Abheben von Bargeld an einem Bankautomaten.

1. Der Kunde authentisiert sich mit seiner Kombination aus EC-Karte (etwas, was er besitzt) und seiner PIN (etwas, was er weiß).

2. Der Bankautomat authentifiziert den Kunden, bei Übereinstimmung der Informationen, als den rechtmäßigen Benutzer des Bankkontos.

3. Nun ist der Bankkunde autorisiert, einen Betrag von seinem Konto abzuheben. Bei einem Übersteigen des Limits würde der Vorgang aufgrund fehlender Autorisierung abgebrochen werden.

# HTTPS - HYPERTEXT TRANSFER PROTOCOL SECURE

---

mit TLS

HTTPS ist ein Kommunikationsprotokoll im World Wide Web, mit dem Daten abhörsicher übertragen werden können. Es stellt eine Transportverschlüsselung dar.

Technisch definiert wurde es als URI-Schema, eine zusätzliche Schicht zwischen HTTP und TCP.



# TLS - TRANSPORT LAYER SECURITY

---

Früher: Secure Sockets Layer (SSL)

TLS ist ein Verschlüsselungsprotokoll zur sicheren Datenübertragung im Internet.

Es besteht aus den beiden Hauptkomponenten TLS Handshake und TLS Record.

Im TLS Handshake findet ein sicherer Schlüsselaustausch und eine Authentisierung statt.

TLS Record verwendet dann den im TLS Handshake ausgehandelten symmetrischen Schlüssel für eine sichere Datenübertragung – die Daten werden verschlüsselt und gegen Veränderungen geschützt übertragen.



# OWASP - OPEN WEB APPLICATION SECURITY PROJECT

---

Das Open Web Application Security Project (OWASP) ist eine Non-Profit-Organisation mit dem **Ziel, die Sicherheit von Anwendungen und Diensten im World Wide Web zu verbessern.**

Durch Schaffung von Transparenz sollen Endanwender und Organisationen fundierte Entscheidungen über wirkliche Sicherheitsrisiken in Software treffen können.

An der OWASP-Community sind Firmen, Bildungseinrichtungen und Einzelpersonen aus aller Welt beteiligt. Innerhalb der Gemeinschaft werden frei verfügbare Informationsmaterialien, Methoden, Werkzeuge und Technologien erarbeitet.

Das OWASP steht nicht mit Technologiefirmen in Verbindung, obgleich es den bedachten Einsatz von Sicherheitstechnologie unterstützt. Die Verbindungen werden vermieden, um frei von organisationsseitigen Zwängen zu sein. Dadurch wird es leichter, unvoreingenommene, praxisnahe und wirtschaftliche Informationen über Applikationssicherheit bereitzustellen.

Das OWASP verfolgt den Ansatz, Informationssicherheit unter Berücksichtigung der Beteiligten, Abläufe und Ausmaße der Technologie zu erreichen.

# OWASP - THE WEB SECURITY TESTING GUIDE PROJECT.

---

<https://owasp.org/www-project-web-security-testing-guide/stable>

0. Foreword by Eoin Keary

1. Frontispiece

2. Introduction

3. The OWASP Testing Framework

4. Web Application Security Testing

4.0 Introduction and Objectives

4.1 Information Gathering

4.2 Configuration and Deployment  
Management Testing

4.3 Identity Management Testing

4.4 Authentication Testing

4.5 Authorization Testing

4.6 Session Management Testing

4.7 Input Validation Testing

4.8 Testing for Error Handling

4.9 Testing for Weak Cryptography

4.10 Business Logic Testing

4.11 Client Side Testing

5. Reporting

# OWASP - BEISPIEL FÜR SQL INJECTION

---

[https://owasp.org/www-project-web-security-testing-guide/stable/4-Web\\_Application\\_Security\\_Testing/07-Input\\_Validation\\_Testing/05.2-Testing\\_for\\_MySQL.html](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05.2-Testing_for_MySQL.html)

# ASYNCHRONOUS REQUEST - RESPONSE PHASEN

