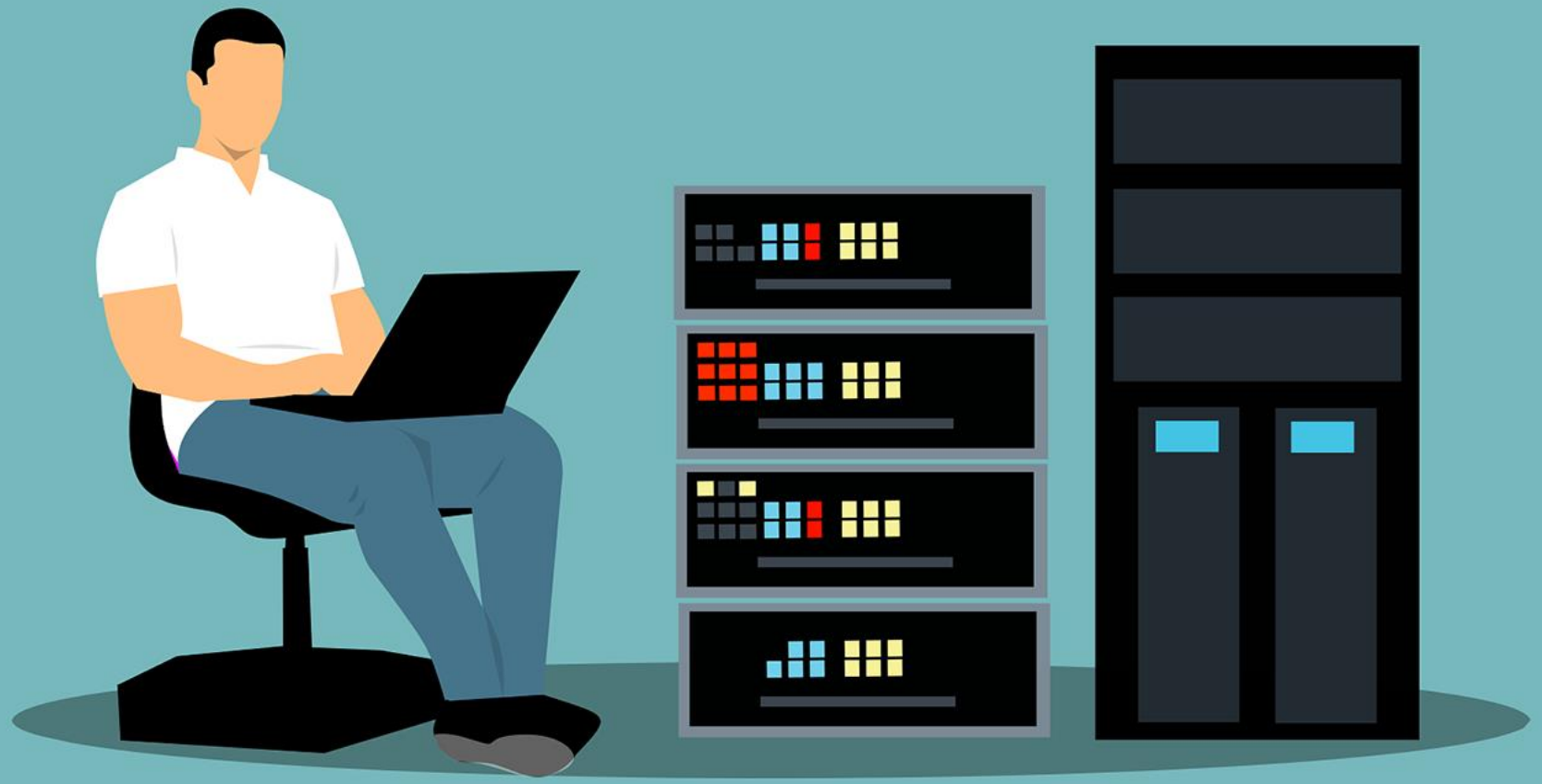
DIHSÜD
Digital Innovation Hub

FH | JOANNEUM
University of Applied Sciences

WebEngineering & ServerTechnologies



Mathias Knoll

pictures without given source are from pixabay

Überblick

Theorie

- Das **Internet** | Geschichte
- **DNS** | Domain Name System
- **Cloud Computing** | Der neue Trend
- **WWW und Webserver** | Das World Wide Web
- **HTML & CSS** | Bausteine von Webseiten
- **Javascript & Typescript** | Dynamik in Webseiten
- **Asynchrones Javascript** | Kerntechnologie Web 2.0
- **Web Frameworks** | Clientseitig, ein Überblick
- **WebServices** | Dienste im WWW
- **PHP** | Beliebteste serverseitige Scriptsprache

Übungen

- SSH | Zugang
- DNS | Namensdienste mit bind9
- Webserver | Apache2, NginX und NodeJS
- Docker | Dienste in Containern
  (Ohne Anleitung)

DIHSÜD
Digital Innovation Hub

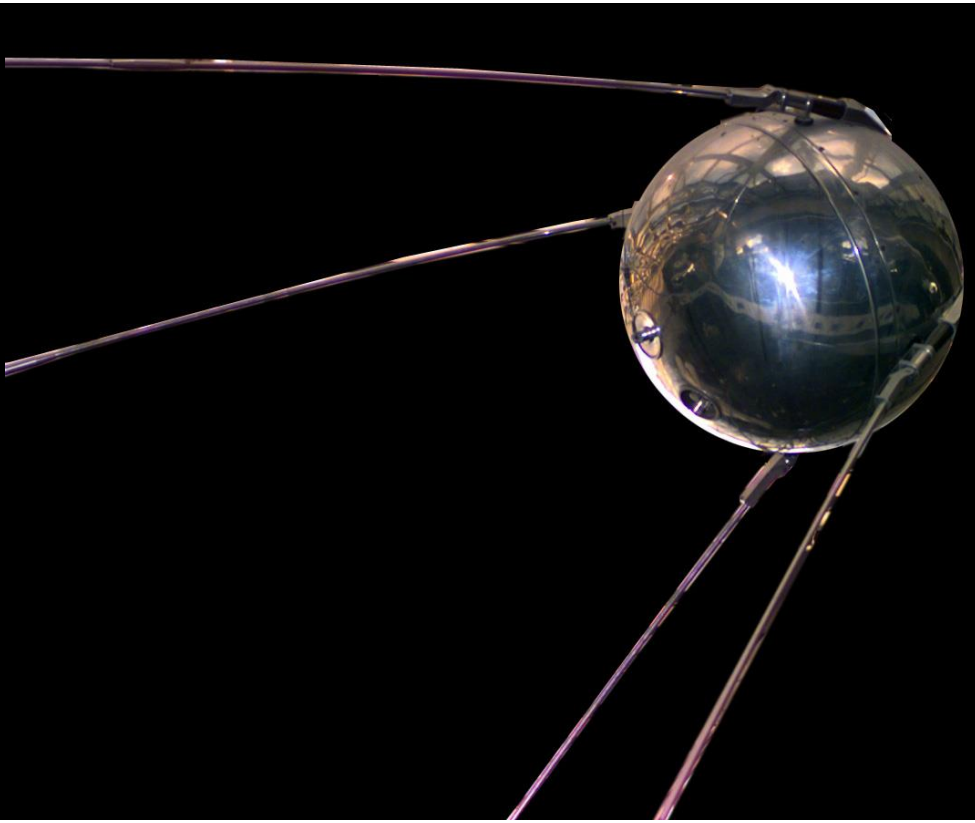
FH | JOANNEUM
University of Applied Sciences

Die Geschichte des Internet

Mathias Knoll

pictures without given source are from pixabay

Sputnik | 1957



urday Evening, October 5, 1957 (UP)—Means United Press Price: Five

Russians Win Race To Launch Earth Satellite

Man On Threshold Of Space Travel

By DANIEL F. GILMORE
United Press Staff Correspondent

LONDON (UP)—The pulsating radio "beep" of the first manmade earth satellite signalled today to the world that man had crossed the threshold into the age of travel through space.

The Soviet Union announced it had won the race into space by launching an earth satellite Friday, a 184-pound, 22-inch globe now orbiting the earth at 18,000 miles an hour, 560 miles up.

Millions of persons throughout the world heard the "beep...beep... beep..." rebroadcast today by local stations and realized that man had taken his first faltering steps into the new era.

Launching of the satellite was a tremendous victory for science. It was a more tremendous victory for Soviet propaganda to be able to trumpet to the world the Russians were the first to break through the frontiers of space.

Bolsters ICBM Claims

— WEATHER —

WEST VIRGINIA—Partly cloudy with highest in the 60s today and Sunday. Lowest tonight 50 west and 40 east portions.

VIRGINIA—Fair with lowest 45 to 50 west and north and 50 to 55 southeast portions tonight, Sunday mostly sunny and a little warmer. Tides on the coast and lower bay will run a foot or two above normal.

How To Spot Satellite

By UNITED PRESS

Here's how to look for the Russian earth satellite which will be whizzing through the sky at 18,000 miles an hour.

The best time to spot it is at dawn or dusk when the sky is semi-dark. There is a chance that it could be seen if it travels across the face of the moon at night.

The best instruments to use are ordinary binoculars or telescopes. Powerful telescopes won't pick it up because of their narrow fields.

Through optical instruments, the satellite will look like the faintest star which can be seen with the naked eye.

Keep a sharp eye out. The satellite travels so fast it may appear on the horizon for only seconds and chances of spotting it have been estimated at one in a hundred.

U. S. May Speed Up Satellite Program

By JOSEPH L. MYL
United Press Staff Corre

WASHINGTON (UP)—A scientists caught glimpse Russia's epic launching of man-made moon, indicate the United States may s its own earth satellite pro

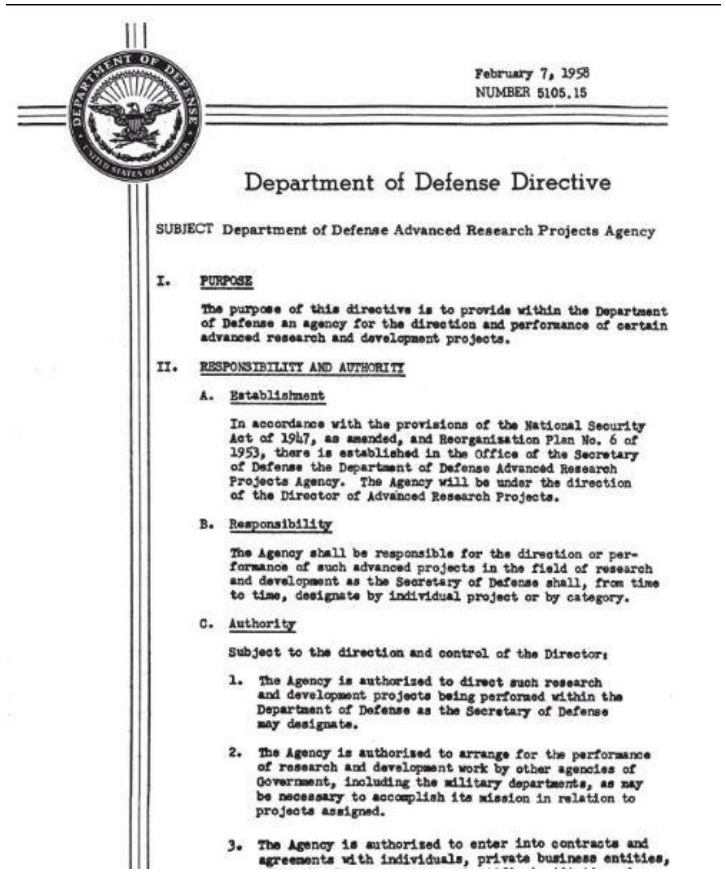
Leaders of the U.S. satel gram also said that it Russia rocketed its hea pound satellite into a gl dling orbit with a rocke to an intercontinental missile.

That could mean Rus only has beaten this count frontiers of space, but also has been called the weapon" for modern day ICBM. This country has tested a successful ICBM

American diplomats

Workshop
WebEngineering & Server Technologies

Die ARPA Direktive | 1958



February 7, 1958
NUMBER 5105.15

Department of Defense Directive

SUBJECT Department of Defense Advanced Research Projects Agency

I. PURPOSE

The purpose of this directive is to provide within the Department of Defense an agency for the direction and performance of certain advanced research and development projects.

II. RESPONSIBILITY AND AUTHORITY

A. Establishment

In accordance with the provisions of the National Security Act of 1947, as amended, and Reorganization Plan No. 6 of 1953, there is established in the Office of the Secretary of Defense the Department of Defense Advanced Research Projects Agency. The Agency will be under the direction of the Director of Advanced Research Projects.

B. Responsibility

The Agency shall be responsible for the direction or performance of such advanced projects in the field of research and development as the Secretary of Defense shall, from time to time, designate by individual project or by category.

C. Authority

Subject to the direction and control of the Director:

1. The Agency is authorized to direct such research and development projects being performed within the Department of Defense as the Secretary of Defense may designate.

2. The Agency is authorized to arrange for the performance of research and development work by other agencies of Government, including the military departments, as may be necessary to accomplish its mission in relation to projects assigned.

3. The Agency is authorized to enter into contracts and agreements with individuals, private business entities,

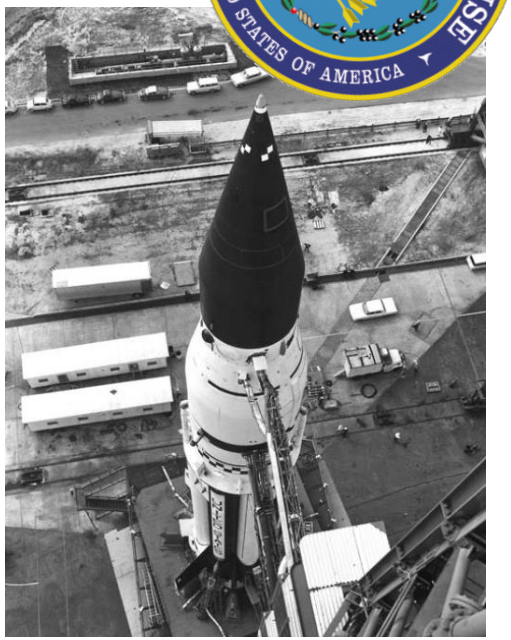
The Advanced Research Projects Agency

Initialer Fokus:

- Raumfahrttechnologie
- Ballistische Raketenverteidigung
- Erkennung von Nuklearwaffentests

The Agency is authorized to enter into contracts and agreements with individuals, private business entities, educational, research or scientific institutions including federal or state institutions.

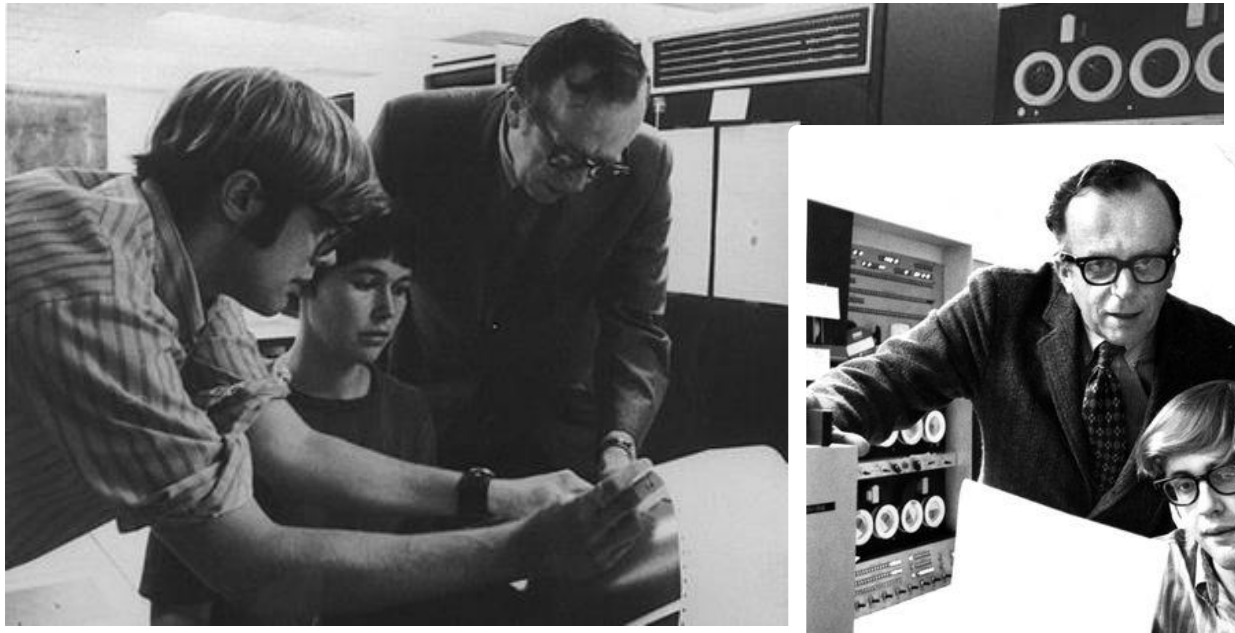
Die Agentur ist **autorisiert**, mit **Individuen, privaten Geschäftseinrichtungen, Institutionen für Lehre, Forschung oder Wissenschaft** aber auch stattlichen Einrichtungen **Verträge abzuschließen.**



Saturn Raketen
mit DARPA Technologie

Information Processing Techniques Office (IPTO) | 1961

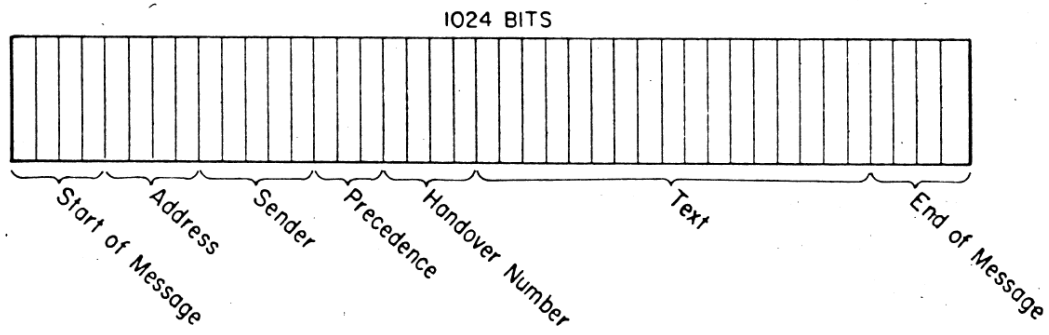
- Abteilung der DARPA
- Erster Direktor:
  Joseph Carl Robnett Licklider
  - Visionär
  - Konzept
    "Intergalactic Computer Network"
- Seine Programme:
  - **Computer Science Departments in Universitäten**,
  - **Time-Sharing Systeme** und
  - **Netzwerke**



Netzwerke | 1962/3 | "Packet Switching" (Paketvermittlung)

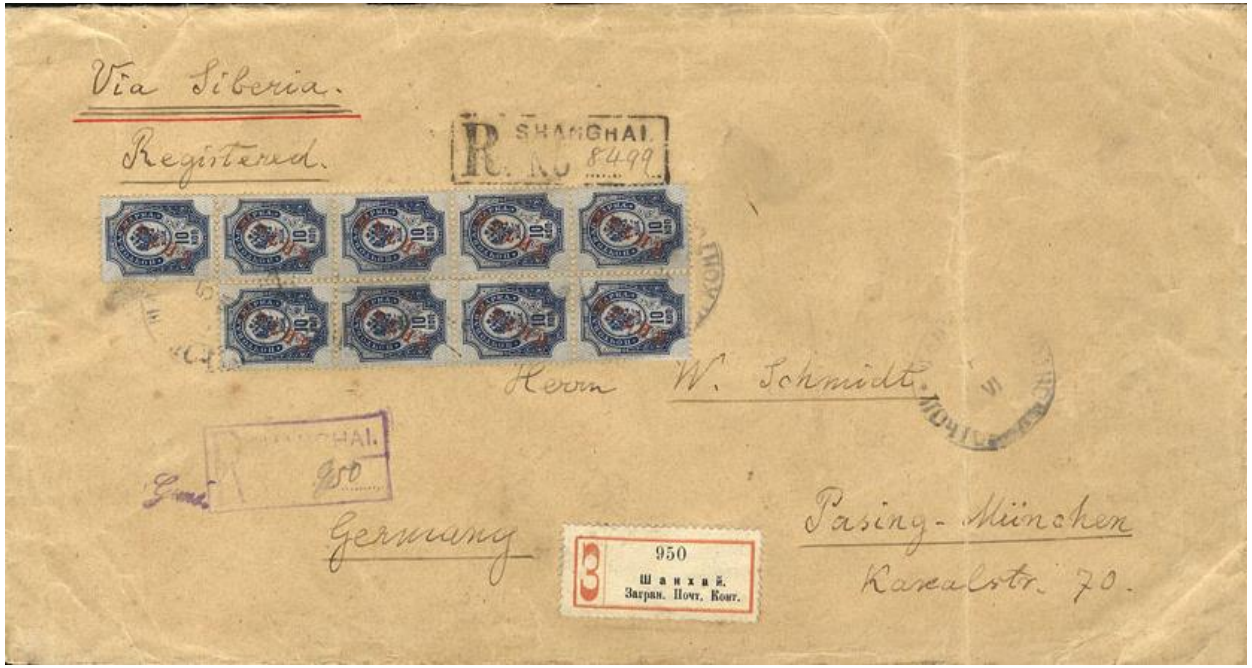
statt Leitungsvermittlung

- Paul Baran (Rand Corp) "On Distributed Communication Networks" (1962-)
- Donald Davies (National Physical Laboratory (United Kingdom) entwickelte den Terminus "Packet" (1963)
- Aus welchen Teilen besteht ein Paket:
  - **Quelladresse**,
  - **Zieladresse**,
  - **Länge des Datenteils**,
  - **Nummer**
  - **Art des Pakets** und
  - der **Payload** (Nutzlast) – die **Daten** (oder Text)



Paul Baran "On Distrtibuted Communication Networks" (1962)

Netzwerke | Packet Switching - Analogie: Briefe



Netzwerktopologie | 1967

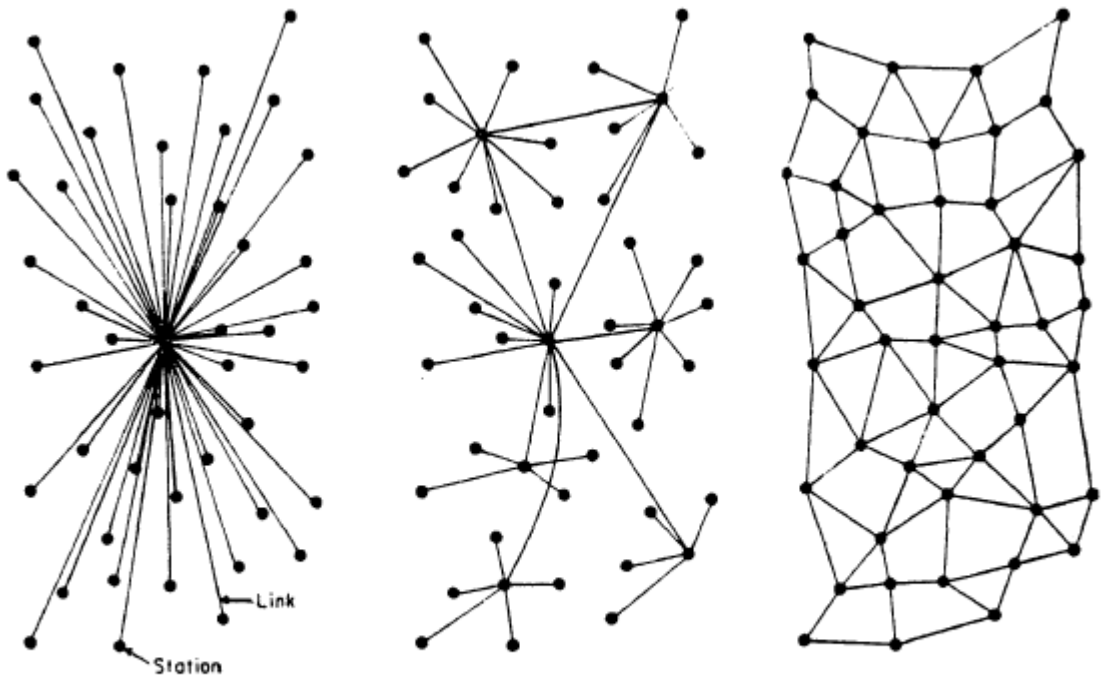
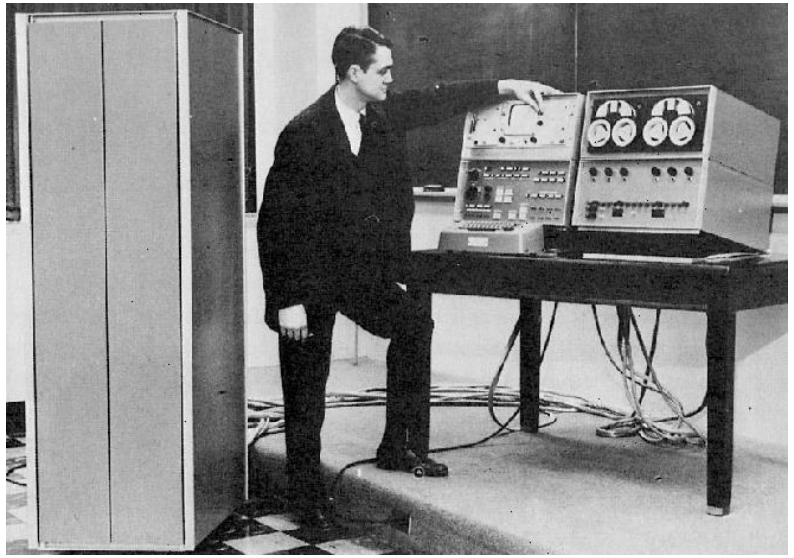


Fig. 1—(a) Centralized. (b) Decentralized. (c) Distributed networks.

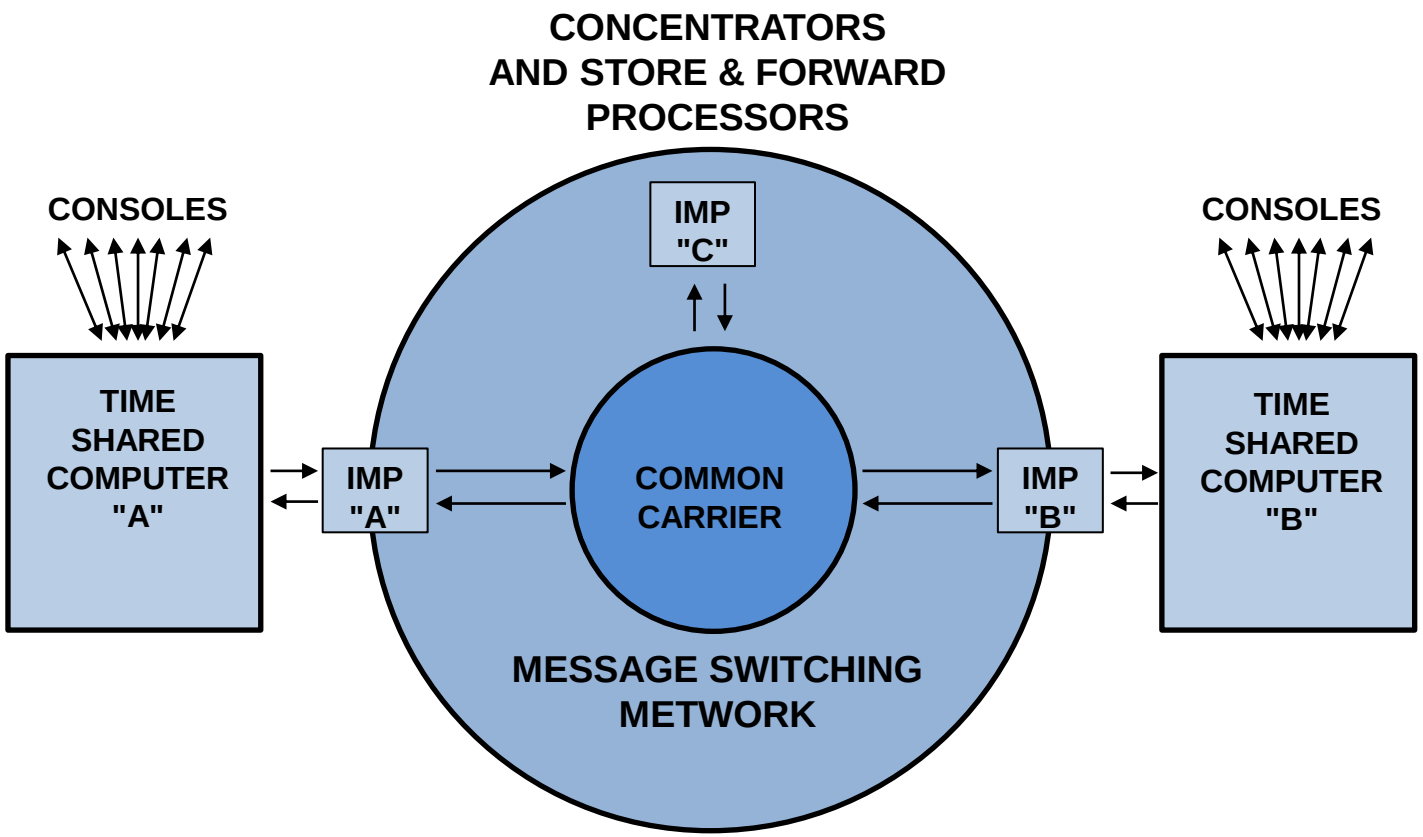
Paul Baran, 1962



Wes Clark, Erfinder des ersten Minicomputers **LINC** (**Laboratory INstrument Computer**) kam **1967** mit der Idee, Minicomputer als Nachrichtenüberbringer einzusetzen- also als **Vermittler zwischen Host und Netzwerk**

Entwurf | 1968

Larry Roberts (2016)
https://www.youtube.com/watch?v=qkD4HVRnGJE



USE OF INTERFACE MESSAGE PROCESSORS (IMP)
FOR ROUTINNG AND CHECKING INTER COMPUTER
COMMUNICATION

Der erste Netzwerkknoten | 1969

Interface Message Prozessor
"Gateway/Router"
Protokoll 1822 *(RFC802)*

SDS Sigma 7
"32Bit Host"

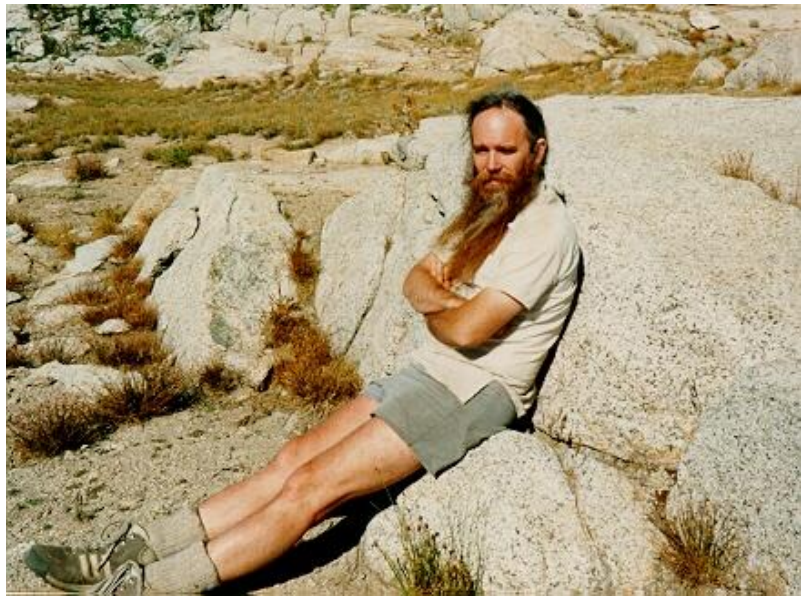
```
   +-+--------------+--------+------+-------------------------+
   |*|   Op Code    |   R    |  X   |    Reference address    |
   +-+--------------+--------+------+-------------------------+
bit 0 1            7 8      1 1    1 1                        3
                            1 2    4 5                        1

Bit   0     indicates indirect address.
Bits  1-7   contain the operation code (opcode)
Bits  8-11  encode a register operand (0:15)
Bits 12-14  encode an index register (1:7). 0 indicates no indexing.
Bits 16-31  encode the address of a memory word.
```


https://en.wikipedia.org/wiki/SDS_Sigma_series

RFCs (Request for Comments) | 1969

"The early R.F.C.'s ranged **from grand visions to mundane details**, although the latter quickly became the most common. Less important than the content of those first documents was that they were available free of charge and anyone could write one. Instead of authority-based decision-making, we relied on a process we called "**rough consensus and running code**." **Everyone was welcome to propose ideas**, and **if enough people liked it and used it, the design became a standard**."

Stephen Crocker, Creator of RFC1, 04/2009
https://www.nytimes.com/2009/04/07/opinion/07crocker.html



RFCs (Request for Comments) | Überblick

- RFCs? (WTF? Request for Comments!)

  "We reject kings, presidents and voting. We believe in rough consensus and running code."

  -Dave Clark, MIT / IAB

  – Keine zentrale Organisation für technische Spezifikationen
  – Die Evolution der Internet Spezifikationen ist ein offener, "unkontrollierter" Prozess
  – Jeder "Netizen" kann dran teilhaben, wenn wer eine Anregung hat, reicht er ein RFC ein
  – Basis: RFCs – Request for Comments (http://www.rfc-editor.org/)
- Über den RFC Editor (und andere Werkezuge) wird der Entwurf kommentiert, eine eindeutige Nummer referenziert darauf
- Einmal akzeptiert kann ein RFC nicht mehr geändert, allerdings fortgeführt und ergänzt werden

- Einträge:

  An Ethernet Address Resolution Protocol (ARP) (RFC826)

  A Standard for the Transmission of IP Datagrams on Avian Carriers (RFC1149)

  The Infinite Monkey Protocol Suite (IMPS) (RFC 2795)

  Electricity over IP (RFC 3251)

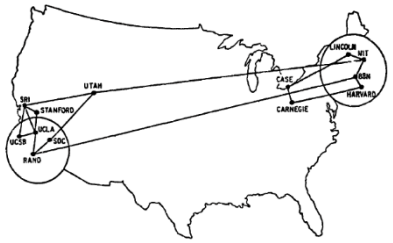
  Management of IP Numbers by Peg-DHCP (RFC2322)

  usw.

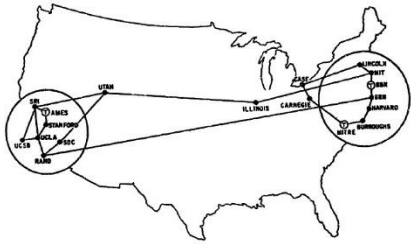
RFC Editor

https://www.rfc-editor.org
https://www.ietf.org/standards/rfcs/

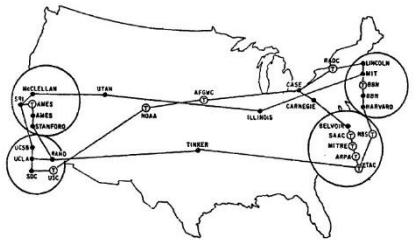
ARPAnet | 1970-1979 | Es skaliert …



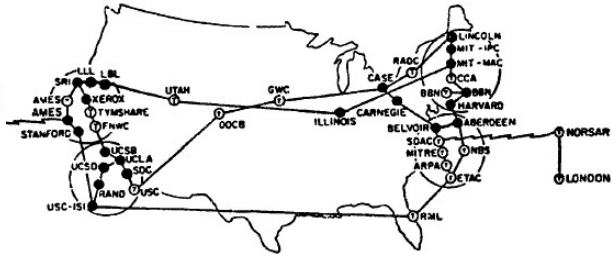
1970



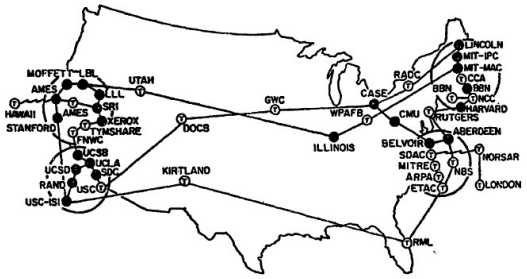
1971



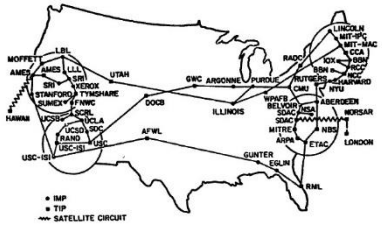
1972



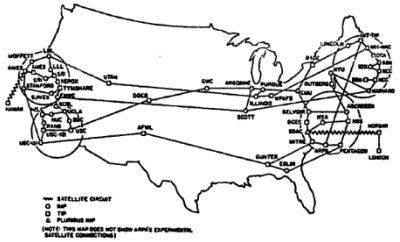
1973



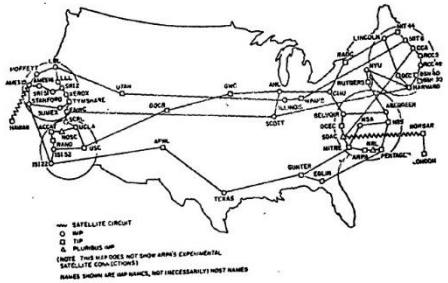
1974



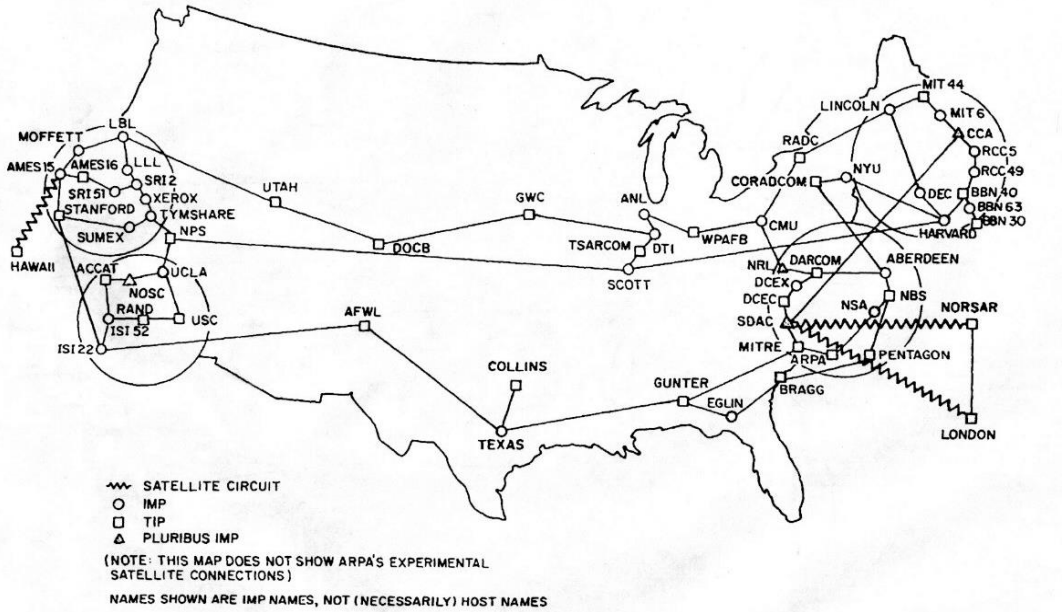
1975



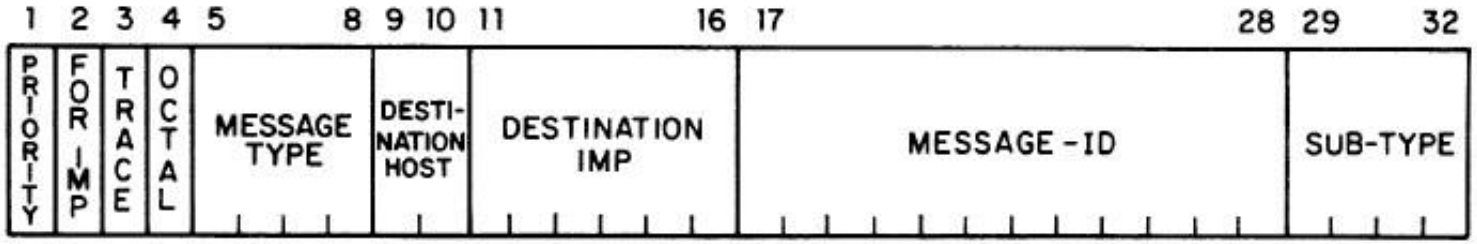
1976



1977

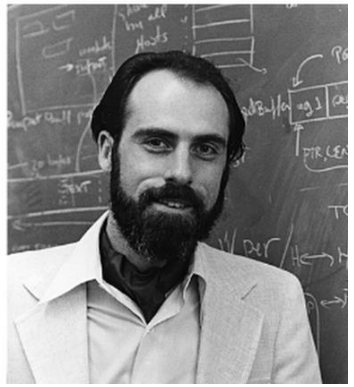


Protokolle | So sprechen Computer übers Netzwerk



Host zu IMP (Teil des **Network Communication Protocols NCP**)

Bob Kahn



Vint Cerf

Bob Kahn und Vint Cerf arbeiten an einem neuen Protokoll, dass die Kommunikation erheblich verbessern soll …

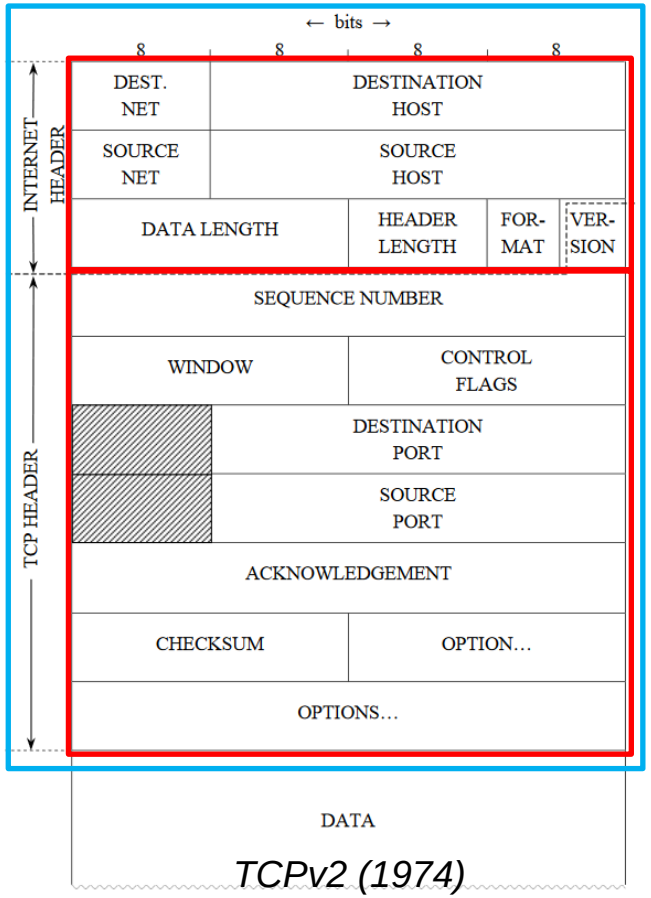
TCP/IP

beide gründen später die

Internet Society

TCP/IP

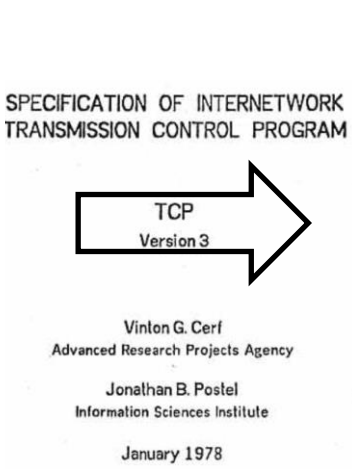
Transmission Control Protocol / Internet Protocol | 1974-1983

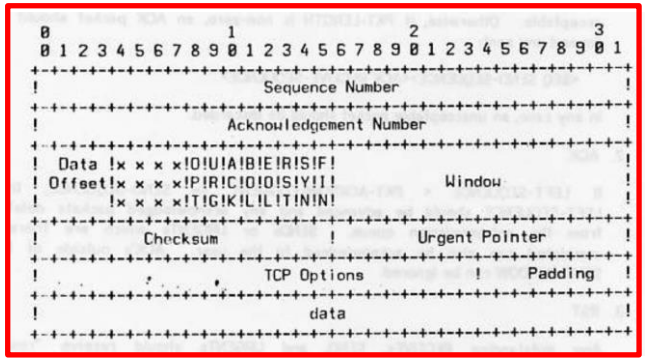


TCPv2 (1974)

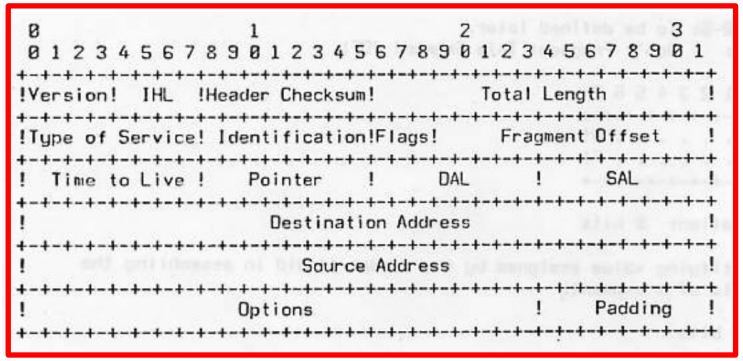
"**We are screwing up** in our design of internet protocols by **violating the principle of layering**. Specifically we are trying to use TCP to do two things: serve as a host level end to end protocol, and to serve as an internet packaging and routing protocol. These two things should be provided in a layered and modular way. I suggest that **a new distinct internetwork protocol** is needed, and that **TCP be used strictly as a host level end to end protocol**."

["Comments on Internet Protocol and TCP", JonPostel,1977 https://www.rfc-editor.org/ien/ien2.txt]





TCPv4 (Juni 1978)



IPv4 (Juni 1978)

Netzwerkmodelle

- 4 Schichten Netzwerkmodel (TCP/IP)
  - Seit 1973
  - Implementierung vor Spezifikation
  - Aus ARPAnet Protokollen entstanden
- 7 Schichten Netzwerkmodel (OSI model)
  - Seit 1983
  - Spezifikation vor Implementierung - Referenzmodel
  - Standardisierung durch die Open System Interconnection
  - Wird als heute als Beschreibung der Funktionalitäten verwendet
  - Protokolle wie ARP (RFC 826, 1982) lassen sich nicht eindeutig einordnen

Wer regiert/regelt das Internet? | Heute

Grundsätzlich: Niemand – Verschiedene Institutionen bieten Richtlinien und Verfahrensanweisungen. Die meisten davon sind (meist U.S.) NPOs - Non-Profit Organisationen.

- ICANN (Internet Corporation for Assigned Names and Numbers):
  - Domains und DNS Routing (.com, .at, .net, …)
  - Dienste und „Ports“
- IANA (Internet Assigned Numbers Authority)
  - Untersteht ICANN in Koop. mit IETF
  - IP Adressverteilung
- IETF (Internet Engineering Task Force):
  - Technische Spezifikationen → IAB (Internet Architecture Board) / RFC Editor
  - Loose Organisation mit vielen Arbeitsgruppen für viele Bereiche
- IEEE (Institute of Electrical and Electronics Engineers)
  - Größter technische Berufsverband der Welt

https://www.icann.org/

http://www.iana.org/

https://www.ietf.org/

https://www.ieee.org/

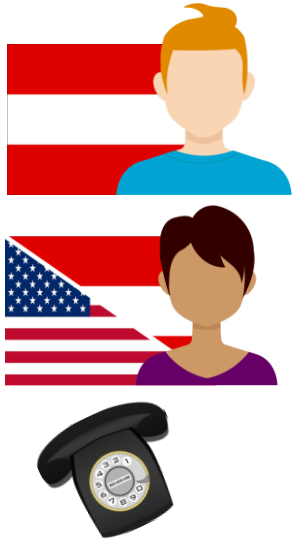
OSI – Open Systems Interconnect

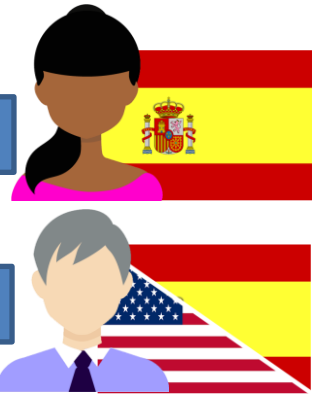
Kommunikation in **sieben Schichten** mit diesen Zielen:

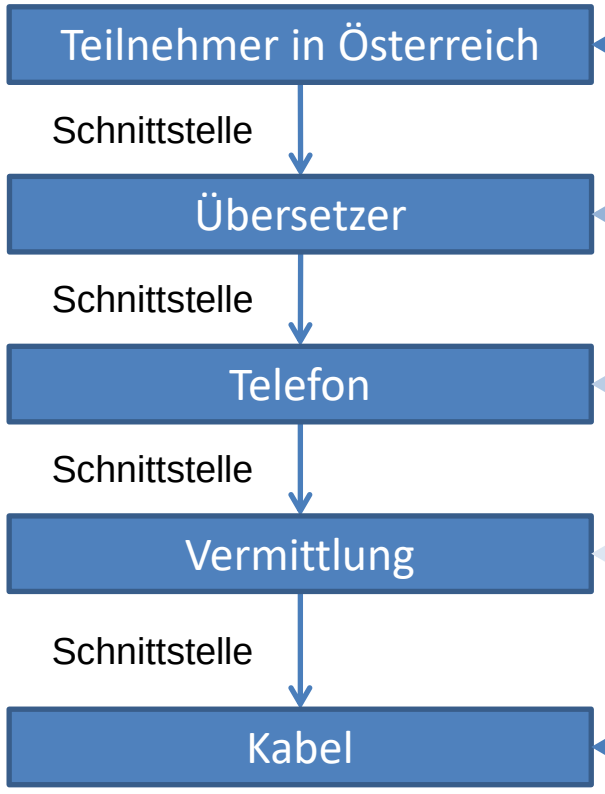
- Eine **Schicht** soll dort erstellt werden, wo ein neuer **Abstraktionsgrad** nötig ist.
- Jede Schicht soll eine **genaue definierte Funktion** erfüllen.
- Bei Funktionswahl sollen international **genormte** Protokolle berücksichtigt sein.
- Die Grenzen zwischen den Schichten sollen so sein, dass der **Informationsfluss** über sie **möglichst gering** bleibt.
- Die Anzahl der Schichten soll eine **handliche Architektur** mit unterschiedlichen Funktionen in unterschiedlichen Schichten sein.

Die glorreichen Sieben

Beispiel – Telefonanruf nach Spanien







Schnittstelle
Übergang zwischen 2 übereinanderliegend en Schichten
Dienste
Leistungen, die eine Schicht der darüber liegenden zur Verfügung stellt

Protokolle
Kommunikationsvorschrif t von Partnern der gleichen Schicht.
Virtuelle Verbindung
Kommunikation zwischen 2 Partnern der gleichen Schicht

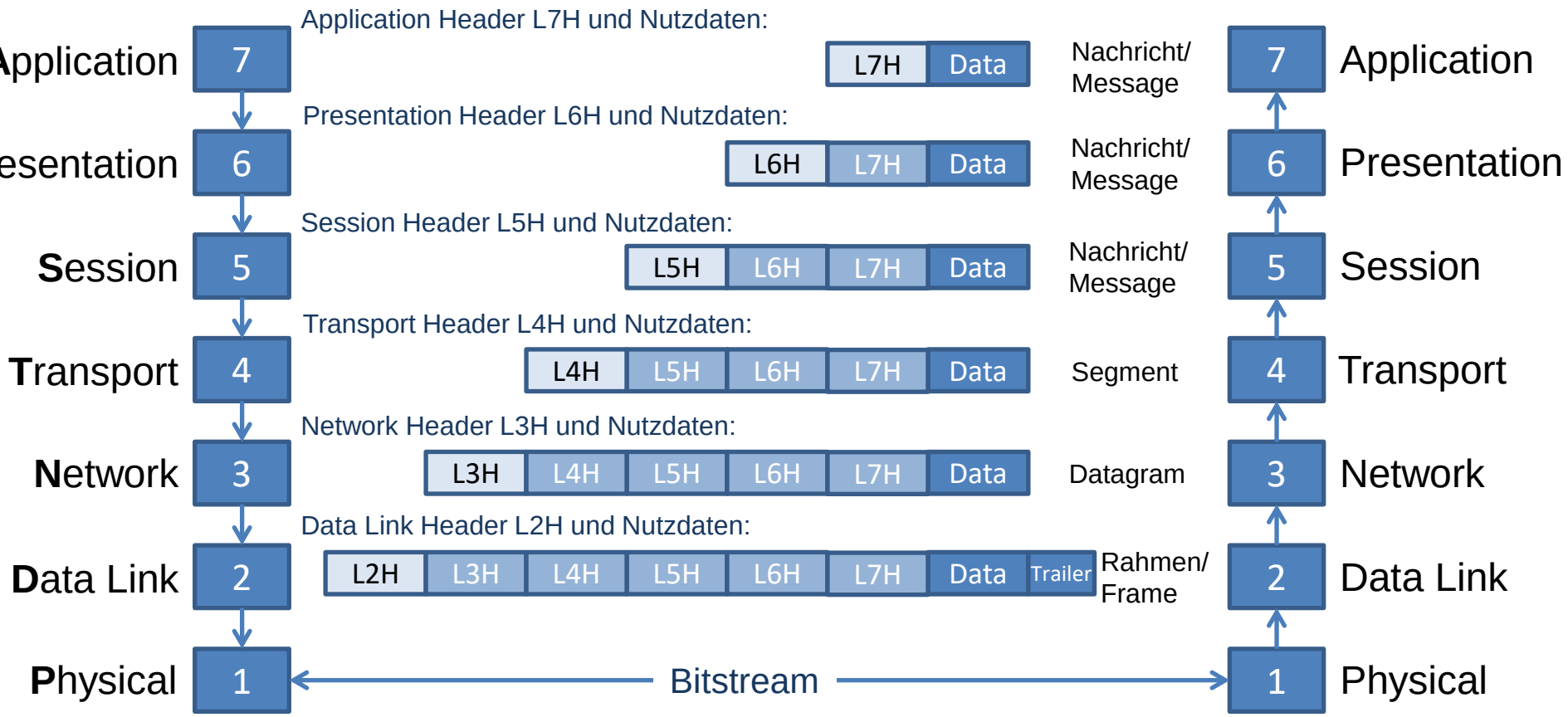
OSI Datenkapselung und -transport

Anwendungsorientiert:

- Anwendungsschicht
- Präsentationsschicht
- Sitzungsschicht

Transportorientiert:

- Transportschicht
- Vermittlungsschicht
- Sicherungsschicht
- Physische Schicht

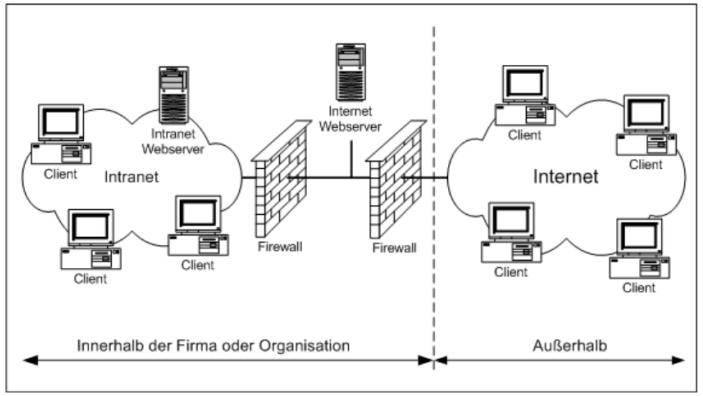


Funktionen des Netzwerkmodells

- **Verbindungslose** und **verbindungsorientierte** Kommunikation
- **Unicast, Broadcast und Multicast**
- **Qualitätskriterien** und **Fehlerbehandlung**
- **Flußkontrolle** (Vermeidung von Staus)
- **Sicherheit**
- **C.I.A. Confidentiality, Intergrity, Accessability**
- **Verschlüsselung**, **Authentifizierung**
- **Accounting** (Kosten, Zeit, Datenrate, …)

Internet

- Dezentrales **Computernetzwerk** von Rechnern
- **Netz aus Subnetzen**
  - Netzwerkknoten vieler ISP zum effizienteren Datenaustausch
  - IXP (Internet Exchange Point), NAP (Network Access Point)
  - Firmen & Organisationen betreiben lokale Netze via Subnetting
- **Keine zentrale Kontrolle**
- **Dienste**
  - **World Wide Web (WWW)** | HTTP+ URLs+ Ressourcen (Hypertext, Hypermedia/Files)
  - **Email** | SMTP, POP, IMAP
  - **Dateitransfer** |FTP, SFTP, NFS, …
  - **Fernzugriff** | SSH, telnet



Fragen …

- Für was steht DNS?
- Warum braucht man DNS?
- Wie arbeitet DNS?
- Aus welcher Notwendigkeit hat sich DNS entwickelt?
- Wie installiere ich einen DNS Server?
- Wie konfiguriere ich einen DNS Server?
- Welche Probleme können auftreten?

Das Domain Name System

- Einfache Definition
  Ein DNS Server übersetzt eine IP Adresse in einen Namen und einen Namen in eine IP
  *D.h. er macht eine **Namensauflösung***

- Warum DNS?
  - *Milliarden IP Adressen werden verwendet und sind schwer mit Firmen, Orten zu assoziieren*
  - *IP Adressen ändern sich regelmäßig – Namen nicht*

Geschichte (das Wesentliche)

- 60s: Advanced Research Projects Agency (**ARPA**)
  - Experimental wide area network for data exchange
- 1973 – **HOSTS.TXT** (RFC 606)
  - Internet Assigned Numbers Authority (IANA)
  - Management of root and top level domains
- 1983 – **DNS erfunden** (RFC 882)
  - TCP/IP and DNS RfCs by ***Mockapetris***
  - Int. Corp. for Assigned Names & Numbers (ICANN)
  - Outsourcing to civil companies
  - Controversy over naming and assignments

Vor DNS

- Namensauflösung per **simpler Textdatei**, welche auf alle Hosts im Netzwerk verteilt wurde.
- ***Probleme***
  - Rasanter Wachstum des ARPAnet
  - Namenskonflikte durch/und
  - Inkonsistente Dateien

HOSTS.TXT

```
NET : 10.0.0.0 : ARPANET :
NET : 128.10.0.0 : PURDUE-CS-NET :
GATEWAY : 10.0.0.77, 18.10.04 :
MIT-GW.ARPA,
MIT-GATEWAY : PDP-11 :
MOS : IP/GW, EGP :
HOST : 26.0.0.73, 10.0.0.51
SRI-NIC.ARPA, SRI-NIC, NIC :
DEC-2060 : TOPS-20 :
TCP/TELNET, TCP/SMTP
TCP/TIME, TCP/FTP
TCP/ECHO, ICMP :
HOST : 10.2.0.11 : SU-TAC.ARPA,
SU-TAC : C/30 : TAC : TCP
```

Nach DNS

- Schnelle Suche (Abfragen)
- Replizierte Inhalte (Ausfallsicherheit)
- Verteilte Kontrolle (Zonen)

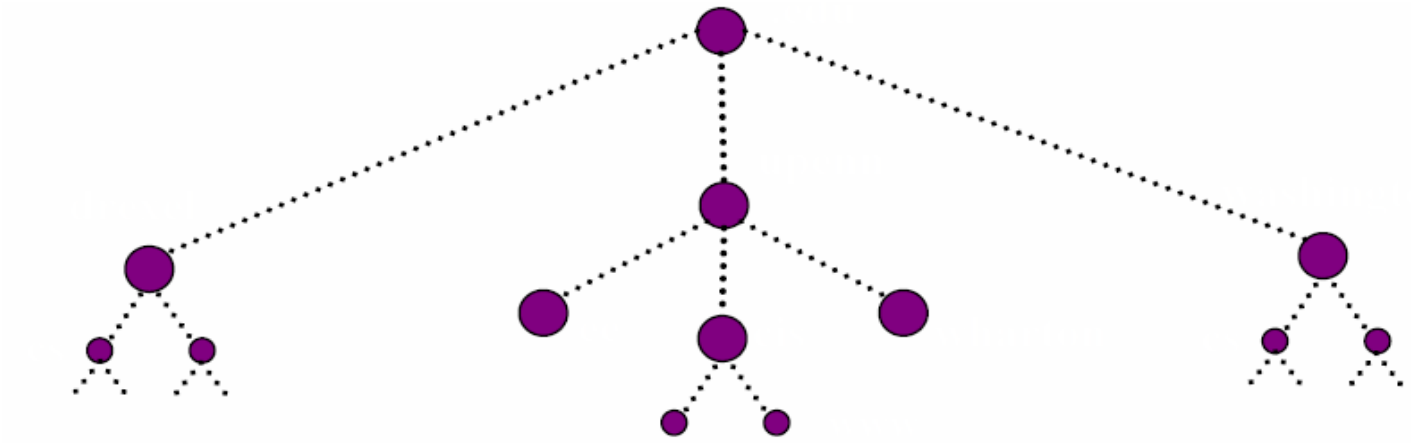
- Der **B**erkeley **I**nternet **N**ame **D**omain Server (BIND) wird entwickelt. Jetzt verwaltet ihn das **I**nternet **S**ystems **C**onsortium (ISC).
- Zahlreiche andere DNS Server mit Option eine Datenbank anzubinden!

DNS Eckdaten

- Anfragen gehen über **UDP Port 53** mit **Maximallänge von 512Byte**
- **Zonentransfers und lange Abfragen/Antworten** gehen über **TCP Port 53**
- EDNS (Extended DNS) erweitert die Funktionalität des Protokolls und ist für DNSSEC (Secure DNS) obligatorisch

Eigenschaften des DNS

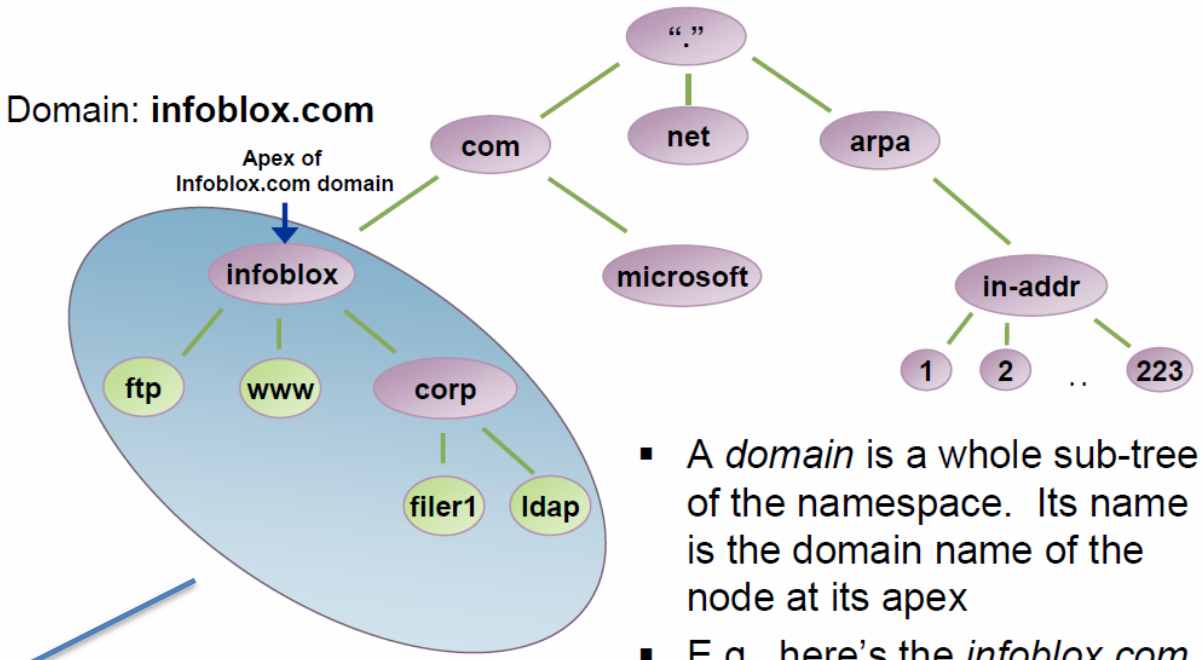
- Es ist hierarchisch aufgebaut.
- Es ist skalierbar.
- Es hat eine dezentralisierte Organisation.
- Es besteht aus Domänennamen, welche die flache Struktur des HOSTS.TXT ersetzen.



Die DNS Hierarchie

- Oberstes Element des DNS Baumes ist der „Root“-Knoten. Er wird durch einen einfachen Punkt („.“) dargestellt.
- Es folgen Top-Level-Domänen wie „at“, „org“ oder „net“.
- Immer mit Punkt getrennte fügen sich weitere Domänennamen an. (Second-Level-Domänene und Subdomänen)
- Domänennamen definieren entweder einen Host oder die Domäne selbst, in der sich Domänen und Hosts befinden!

Domänen, Hierarchie & FQDN



- A *domain* is a whole sub-tree of the namespace. Its name is the domain name of the node at its apex
- E.g., here’s the *infoblox.com* domain

Eine Domäne ist ein ganzer Teil-Baum im Namensraum. Ihr Name ist der Domänenname des obersten Knotens.

In diesem Beispiel ist es infoblox.com.

Wichtig:
FQDN – Fully Qualified Domain Name
filer1.corp.infoblox.com.

Der DNS Namensraum

- Es handelt sich um eine verteilte Datenbank, die über Domänennamen indiziert ist.
- Jeder Domänenname ist ein Pfad im Namensraum (DNS Baum/Hierarchie)
- Jeder Knoten/Name kann bis zu 63 Buchstaben haben. (Ausnahme: Root Ebene hat keinen.)
- Ein FQDN beschreibt einen eindeutigen Pfad im DNS. **Achtung**: Zum FQDN gehört der Punkt am Ende! Beispiel: *filer1.corp.infoblox.com***.**

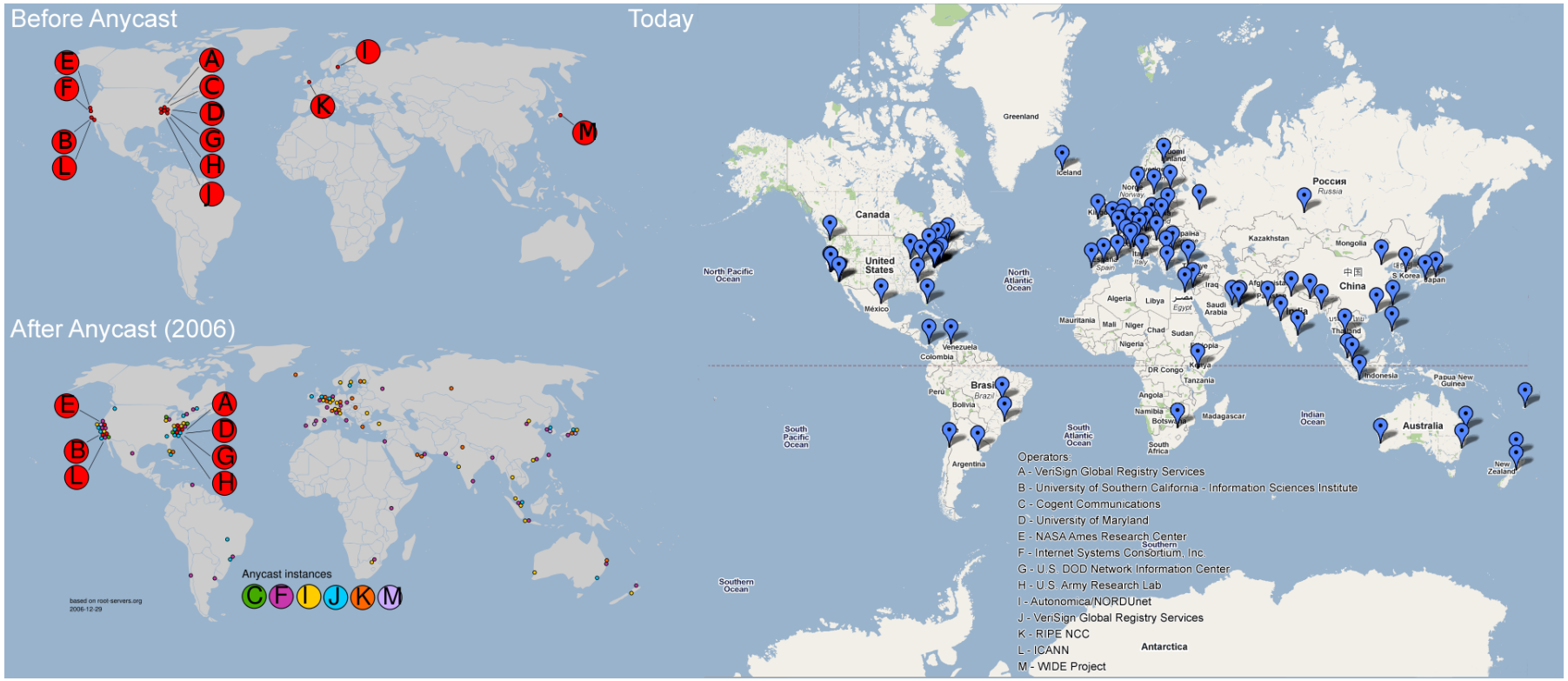
Delegation

- Das DNS hat ein dezentralisiertes Management.
- Eine Organisationseinheit verwaltet eine Domäne, welche sie in eigen-verwaltete Subdomänen teilen kann.
- Die OE bzw. der Betreiber der Rootserver ist die ICANN.
- Für TLD delegiert die ICANN die „Domain Name Registry“ – Organisationen oder Unternehmen.
  - Z.B. nic.at für Österreich

Redundanz

- Pro Zone/Domäne gibt es mehr als einen Namensserver
- Dieser wiederum kann auf mehrere Maschinen repliziert (Master/Slave) werden.
- So gibt es zum Beispiel:
  - 13 Rootserver
  - Ca. 9 Server für die TLD „at“ und
  - Ca. 2 Server für die SLD „fh-joanneum.at“

Die DNS Root Server



Wikipedia Commons

Vormals gab es wirklich nur 13 physikalische Maschinen

Mittlerweile gibt es pro IP eines Root Servers viele Server dahinter. (Anycast, Kürzeste Route gewinnt!)

Top Level Domänen

Unter
http://www.iana.org/domains/root/db/
oder
http://www.bitmedia.com/cc/url1.htm
findet man die vollständige Liste…

Praktisches Beispiel

- *Wie findet man heraus, welche Nameserver für die TLD unseres Landes zuständig sind?*
- *Wie findet man heraus wie viele Nameserver für unsere Universität zuständig sind?*

- Hinweise:
  - Unter Debian wird der Befehl „host“ verwendet.
  - „host“ ist Teil des „dnsutils“ Pakets.

Ergebnis des Beispiels

```
servertec:~# host -t ns at.
at name server ns1.univie.ac.at.
at name server ns9.univie.ac.at.
at name server j.nic.at.
at name server n.nic.at.
at name server ns-uk.nic.at.
at name server ns2.univie.ac.at.
at name server d.nic.at.
```

```
servertec:~# host -t ns fh-joanneum.at.
fh-joanneum.at name server tapa.fh-joanneum.local.
fh-joanneum.at name server ribe.fh-joanneum.local.
fh-joanneum.at name server waco.technikum.fh-joanneum.local.
fh-joanneum.at name server doha.technikum.fh-joanneum.local.
fh-joanneum.at name server mora.technikum.fh-joanneum.local.
fh-joanneum.at name server oita.technikum.fh-joanneum.local.
fh-joanneum.at name server anif.fh-joanneum.local.
```

Funktionen im Überblick

- Vorwärts-Auflösung (Forward Mapping)
  - Name zu Adresse
  - Bsp: **`dakar.fh-joanneum.at`** → **`62.218.221.66`**
- Rückwärts-Auflösung (Reverse Mapping)
  - Adresse zu Name
  - Bsp: **`66.221.218.62.in-addr.arpa`** → **`dakar.fh-joanneum.at`**
  - ***Achtung***: Man beachte, dass es fürs Reverse Mapping eine spezielle Hierarchie gibt, in der die IP umgekehrt inkorporiert ist! An deren Ende steht statt **in-addr.arpa**!
- Klassisches Werkzeug: **`nslookup`**

Wie funktioniert DNS?

Resolver =
Programm oder Bibliothek, die für Namensauflösungen genutzt wird!

1. Anfrage am PC über **Resolver**
2. Ein DNS Server wird gefragt (**rekursiv**)
3. Dieser fragt einen Root Server
4. Dieser gibt einen TLD Server zurück
5. Dieser gibt den Server einer OE zurück
6. Dieser gibt die Adresse zurück
7. Die Adresse wird an den PC zurückgegeben.

Beispiel zu DNS Abfragen

- *Wie kann man sehen, wie viele Ebenen/Nameserver man befragen muss, um an eine IP Adresse zu kommen?*

- **Verwendeter Befehl:**

```
dig +norec +noques +nostats +nocmd <host> @<start-of-search>
```

  - "**dig**" ist ein umfangreiches Abfragewerkzeug.
  - "**+norec**" nicht rekursiv, nur der erste Zweig wird angezeigt.
  - "**<host>**" der gesuchte Host.
  - "**@<start-of-search>**" von welchem DNS Server man sucht.

Ergebnis des Beispiels

```
#dig +norec +noques +nostats +nocmd www.fh-joanneum.at. @a.root-servers.net.

#dig +norec +noques +nostats +nocmd www.fh-joanneum.at. @ns9.univie.ac.at.

#dig +norec +noques +nostats +nocmd www.fh-joanneum.at. @dallas.fh-joanneum.at.

;; ANSWER SECTION:www.fh-joanneum.at.     86400   IN      CNAME   dakar.fh-
joanneum.at.dakar.fh-joanneum.at.   86400   IN      A       62.218.221.66
```

Autoritativ & Nicht Autoritativ

- Autoritative Server
  - Besitzt alle Informationen seiner Zone
  - Beantwortet Anfragen aus der eigenen Datenbank
  - Antworten werden als sicher angesehen.
  - Gibt s.g. autoritative Antworten (AA, mit Authoritative Response Flag)
- Nicht Autoritative Server
  - Beantwortet Anfragen aus dem Cache, der aus Antworten eines autoritativen Servers besteht. (rekursiv oder iterativ)
  - Daten im Cache haben ein Ablaufdatum (TTL = time to live)
  - Antworten werden als nicht gesichert angesehen.

Rekursion & Iteration Nicht autoritative Server

- Rekursiv
  - Recursion **Desired** Flag, Recursion **Available** Flag
  - Ergebnis wird von einem anderen Nameserver geholt
  - Standardeinstellung bei vielen Servern aber bei Root Servern deaktiviert
- Iterativ
  - Verweise auf andere Nameserver oder Ressourcen werden zurückgegeben
  - Einige Resolver können in der Regel nichts damit anfangen

Primär (Master) & Sekundär (Slave)

- Primärer (Master) Server
  - Üblicherweise ein autoritativer Server
- Sekundärer (Slave) Server
  - Synchronisiert sich mit dem primären Server mittels Zonentransfers:
    - Asynchroner, voller Transfer (AXFR)
    - Inkrementeller Transfer (IXFER)

  Anmerkung: Sollten Zonendaten in einer Datenbank sein, werden DB Mechanismen zum Transfer verwendet!

Weitere Arten

- Caching Name Servers
  - Im Falle eines Caching Only Name Server = Hint Name Server
  - Spart Bandbreite
- Forwarding (oder Proxy) Name Server
- Stealth Name Server
  - Nicht sichtbar im Internet, weil nicht in der Datenbank des öffentlichen DNS Servers einer Firma.
  - Interne Hosts werden so versteckt.
- Authoritative Only Name Server
- Split-Horizon Name Server
  - Gibt unterschiedliche Ergebnisse zur selben Anfrage. Grund: **Load Balancing**

Ad Caching Name Server

- Nicht autoritativ
- „Negatives Caching“ (RfC2308) speichert Anfragen zu unbekannten Namen
- „Time to live“ (TTL) gibt an, wie lange die Einträge valide sind.
- Anfragen werden über andere Nameserver erledigt. Er fragt ….
  - Lokaler NS für eine Subdomäne
  - Einen externen NS (sofern bekannt)
  - Einen der Rootserver

Wichtige Dateien

- Hostname des Servers: `#vi` **`/etc/hostname`**
  Nach Änderung:
  `/etc/init.d/hostname.sh` [start|stop]
- Lokale Namensauflösung: **`#vi /etc/hosts`**
- Definition der Domäne und der Nameserver für den Resolver: **`#vi /etc/resolv.conf`**
- DNS Utilities: **`#apt-get install dnsutils`**

RFC1035

```
0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                                               /
/                      NAME                     /
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     CLASS                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TTL                      |
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                   RDLENGTH                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--|
/                     RDATA                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

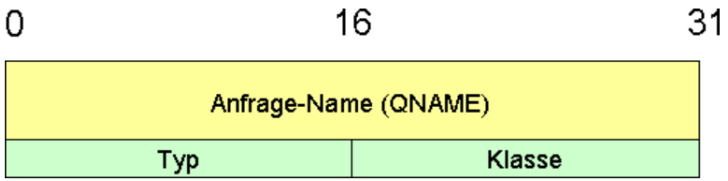
Resource Records

Sind die Einträge in der Zonendatei/-datenbank
Ein Datensatz besteht aus:

- Name (optional)
  Domänenname zu dem der Datensatz gehört
- Gültigkeit (optional)
  Die TTL in Sekunden
- Klasse (optional)
  IN für Internet- andere werden nicht mehr verwendet
- Typ
  Die Typen werden später im Detail durchgenommen
- Ressource Daten
  Beschreiben den Datensatz

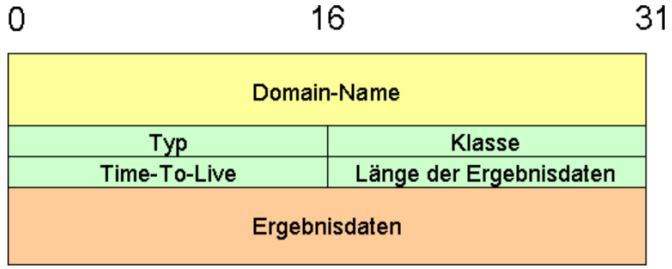
Nachrichtenformat: Anfrage (Request)

```
> Frame 26: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
> Ethernet II, Src: AsustekC_da:58:ea (08:60:6e:da:58:ea), Dst: CiscoInc_ff:fd:2c (00:08:e3:ff:fd:2c)
> Internet Protocol Version 4, Src: 10.52.37.222, Dst: 10.65.1.2
> User Datagram Protocol, Src Port: 63826, Dst Port: 53
v Domain Name System (query)
    [Response In: 27]
    Transaction ID: 0x0004
  v Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v fh-joanneum.at: type A, class IN
        Name: fh-joanneum.at
        [Name Length: 14]
        [Label Count: 2]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
```



Nachrichtenformat: Antwort (Response)

```
> Frame 27: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface 0
> Ethernet II, Src: CiscoInc_ff:fd:2c (00:08:e3:ff:fd:2c), Dst: AsustekC_da:58:ea (08:60:6e:da:58:ea)
> Internet Protocol Version 4, Src: 10.65.1.2, Dst: 10.52.37.222
> User Datagram Protocol, Src Port: 53, Dst Port: 63826
v Domain Name System (response)
    [Request In: 26]
    [Time: 0.000298000 seconds]
    Transaction ID: 0x0004
  v Flags: 0x8580 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .1.. .... .... = Authoritative: Server is an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v fh-joanneum.at: type A, class IN
        Name: fh-joanneum.at
        [Name Length: 14]
        [Label Count: 2]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  v Answers
    v fh-joanneum.at: type A, class IN, addr 91.118.154.85
        Name: fh-joanneum.at
        Type: A (Host Address) (1)
        Class: IN (0x0001)
        Time to live: 1800
        Data length: 4
        Address: 91.118.154.85
```



Beispiel für eine Zone „servertec“

```
$TTL 3D
@                                   IN          SOA         ns.servertec. hostmaster.servertec.
(
                                                                          2010033001 ;serial, date
+ todays serial
                                                                          8H ;refresh of slave‘
info
                                                                          2H ;retry of slave‘ info
                                                                          4W ;expire of slave‘s
info
                                                                          1D ) ;Negative cache
;
@                                           NS                            ns ;Inet Address of name
server
@                                           MX                            10 mail.servertec. ;Prim.
Mail Exchanger
@                                           MX                            20 mail.friend.servertec.
;Sec. Mail Exc.
;
ns                                          A                             192.168.10.1
gw                                          A                             192.168.10.254
mail                                CNAME   ns
```

Beispiel für eine Zone „servertec“

```
$TTL 3D
@                    IN      SOA     ns.servertec. hostmaster.servertec. (
                                          2010033001 ;serial, date + todays serial
                                          8H ;refresh of slave‘ info
                                          2H ;retry of slave‘ info
                                          4W ;expire of slave‘s info
                                          1D ) ;Negative cache
;
@                            NS            ns ;Inet Address of name server
@                            MX            10 mail.servertec. ;Prim. Mail Exchanger
@                            MX            20 mail.friend.servertec. ;Sec. Mail Exc.
;
ns                           A             192.168.10.1
gw                           A             192.168.10.254
mail                 CNAME   ns
```

RRs: Typen 1

- SOA – Start of Authority
  - Administrator der Zone
  - Seriennummer
  - Timer: TTL, Cache, etc..
- NS – Namensserver
  - Der Hostname eines autoritativen Servers in der Zone
- A (Address) resource records
  - Name zu IP Auflösung
  - AAAA für IPv6 Adressen

RRs: Typen 2

- CNAME (Alias)
  - Alias(Kanonischer) Name für eine vorhandene Namensauflösung bzw. einen Host
- MX (Mail Exchanger)
  - Informiert Mailserver an wenn in der Domäne mails gesendet werden sollen.
- PTR (Pointer records)
  - IP auf Name Auflösung

RRs: Anwendungsbeispiele 1

- A

```
name            IN          A           adresse
www             IN          A           192.168.1.1
```

- CNAME

```
name            IN          CNAME  ziel
ftp             IN          CNAME  www
```

- NS

```
domain          IN          NS          server
@               IN          NS          ns.servertec.
servertec.IN            NS          ns.servertec.
```

- MX

```
domain          IN          MX          priority server
@               IN          MX          10 mail.servertec.
servertec.IN            MX          10 mail.servertec.
```

RRs: Anwendungsbeispiele 2

- PTR *(Reverse Lookup! *IP in umgekehrter Reihenfolge!)*

```
ip*          IN          PTR         name
1            IN          PTR         www.servertec.
```

- SOA

```
zone         IN          SOA         primary
                                                 mailaddresse
                                                 seriennummer
                                                 refresh retry expire
                                                 TTL
@            IN          SOA         ns.servertec.
                                                 root.servertec.
                                                 2013020201
                                                 3600 1800 604800
                                                 600
```

Erweiterungen

- Dynamisches DNS
  - Trotz wechselnder IP Adresse soll ein Host immer den gleichen Namen haben.
  - TTL muss klein sein
  - IP Update muss zum DynDNS Server übermittelt werden (DynDNS Client)
- EDNS
  - Extended DNS
- DDNS
  - Aktualisierungsmechanismus am Server
  - „nsupdate“ unter BIND9 (ab V9.3)

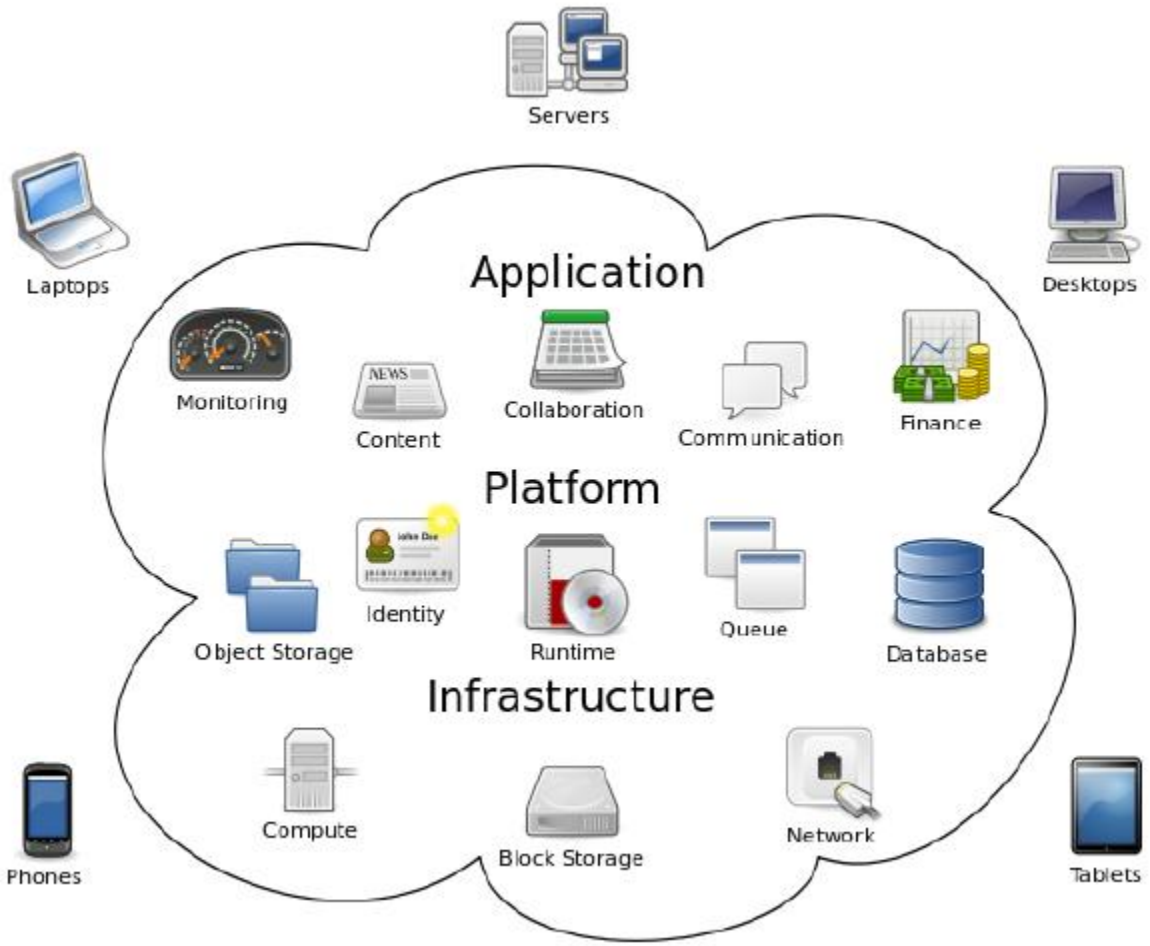
DIHSÜD

FH JOANNEUM
University of Applied Sciences

Cloud Computing

Workshop
WebEngineering & Server Technologies

Übersicht


Servers
Laptops
Desktops
Application
Monitoring
Content
Collaboration
Communication
Finance
Platform
Object Storage
Identity
Runtime
Queue
Database
Infrastructure
Compute
Block Storage
Network
Phones
Tablets

Geschichte

- 1960 | Grundlegende Idee zum "Cloud Computing" entstand. Ziel war das Bereitstellen von IT Ressourcen für Geld
- 1999 | "Salesforce.com" macht Standardanwendungen wie z.B. das Customer Relation Management (CRM) über Browser verfügbar
- 2002 | Amazon's Web Services ermöglichen die Online Speicherung von Daten
- 2006 | Amazon_s "Elastic Computing 2" und "Simple Storage 3" bieten Rechenleistung und Speicher im Internet an
- 2006 | Mit "Google Docs" kann man Online Office Anwendungen nutzen
- 2011 | Mit der Apple iCloud sind zahlreiche Dienste im Internet verbunden

Abstract der NIST Definiton für Cloud Computing

U.S. National Institute for Standards and Technology

Netzwerkzugang

Gemeinsamer Fundus an Mitteln

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be
rapidly provisioned and released with minimal management effort or service provider interaction."

minimales Management

schnell zuzuweisen

https://csrc.nist.gov/publications/detail/sp/800-145/final

Alternative Definition ISO//IEC 17700:20145

"Cloud Computing is a paradigm for enabling network access to a scalable and elastic pool of shareable physical or virtual resources with self-service provisioning and administration on-demand."

Eine Menge an Ressourcen wie Rechenleistung und Speicher werden in einen Fundus geworfen

Kunden fragen anhand ihrer Bedarfe: Z.B.: 8 Prozessoren, 16GB Speicher …

In der Cloud werden diese Ressourcen zugeordnet, welch über das Netzwerk verwendet werden können

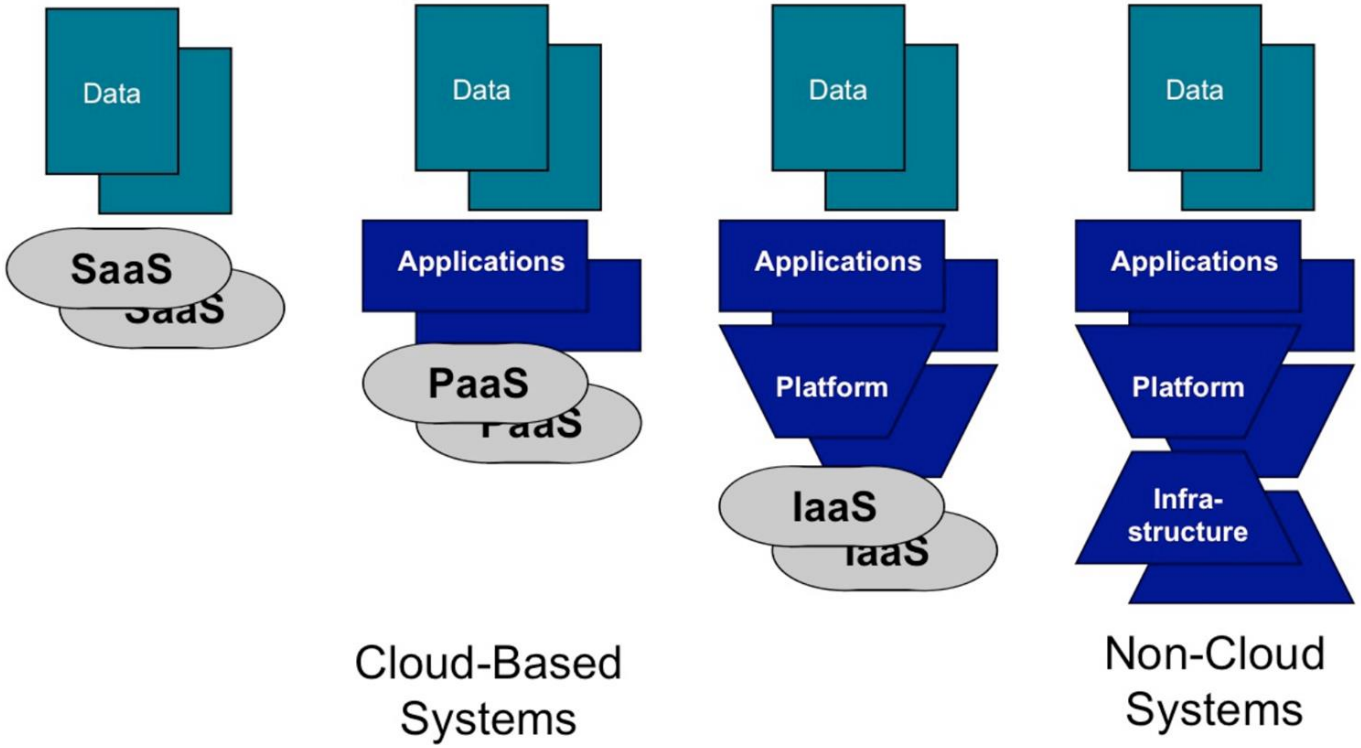
Wenn ein Kunde die Ressourcen nicht mehr braucht, werden diese wieder in den Fundus zurückgegeben – für andere Kunden

Charakteristiken (nach NIST)

- On-Demand Self-Service
  - Automatische Provision von Ressourcen
- Breitbandzugriff
  - Alle Möglichkeiten sollen über das Internet verfügbar sein
  - Verschiedene Plattformen sollen unterstützt werden
- Ressourcen-Pool
  - Ressourcen sind nicht dezidiert sondern werden automatisch zugewiesen
- Rapid Elasticity
  - Ressourcen können je nach Nachfrage leicht provisioniert werden
- Measured service
  - Die Verwendung von Ressourcen wird überwacht, kontrolliert und für eine Abrechnung / Transparenz gezählt

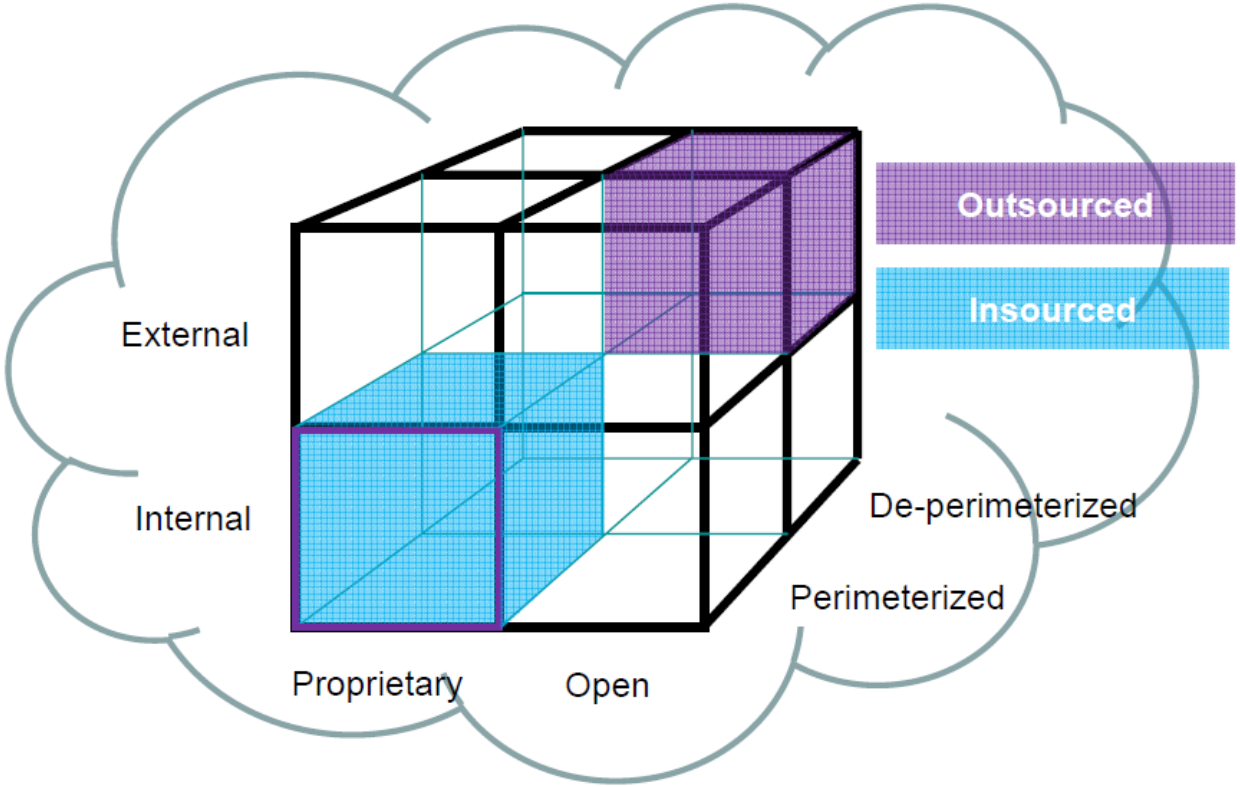
Service Modelle (lt. NIST)

- Software as a Service (SaaS)
  - Auswahl von Software, die Online verwendet werden kann
  - Google Firbase, Oracle On Demand, salesforce.com, SQL Azure
- Platform as a Service (PaaS)
  - Umgebungen mit eigenen Rechen- und Datendiensten, in welchen eigene Software entwickelt werden kann
  - Google AppEngine, force.com, Windows Azure Platform
- Infrastrukture as a Service (IaaS)
  - Virtualisierte Computer, Speicher und Netzwerke
  - Amazon Elastic Compute Cloud, Eucalyptus, GoGrid, OpenStack



Opengroup.com/G135.pdf/18.07.2013

OpenGroup Jericho Cube Model (Cloud Cube Model)



- Physische Dimension
  - Intern
  - Extern
- Besitzende Dimension
  - Proprietär
  - Offen
- Sicherheitsdimension
  - Eingegrenzt (Permieterized)
  - Nicht eingegrenzt (De-Perimterized)
- Quelldimension
  - Innerhalb (In.Sourced)
  - Außerhalb (Outsourced)

 www.opengroup.org/jericho/cloud_cube_model_v1.0.pdf

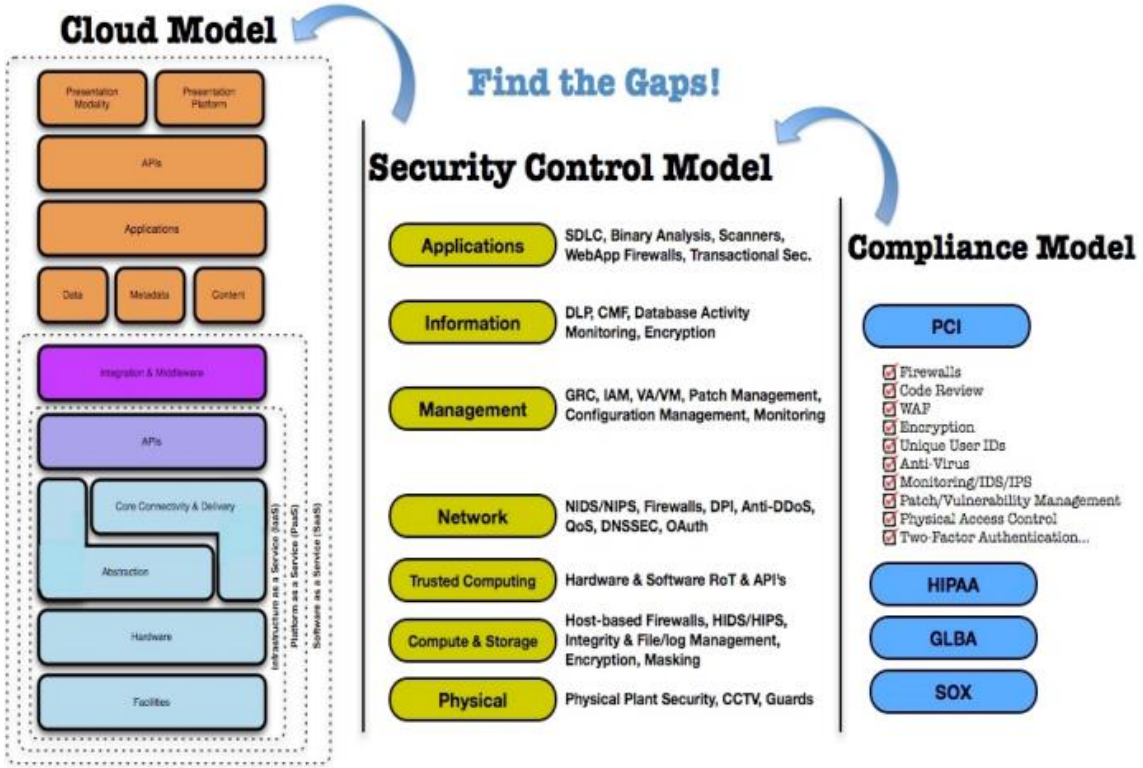
Vorteile des Cloud Computing

- **Niedrige Kosten** bei der **Infrastruktur**
- **Einfache Anwendung** (z.B.: Keine Hardware, keine Lizenzen)
- **Servicequalität (Garantien durch Dienstanbieter)**
- **Externes IT Management** (Keine Lohnkosten)
- **Einfache Wartung und Aktualisierungen** (durch zentralisiertes Managementsystem)
- **Geringe Einstiegshürde**

Mögliche Nachteile des Cloud Computing

- **Abhängigkeit vom Provider** (Nur bestimmte Dienste und eventuell unflexibel)
- Netzwerk – **Latenzen und WAN Verbindungen** (Mehr als im LAN)
- **Privatsphäre / Sicherheit** (Daten werden außerhalb der Firma gespeichert…)
- **Richtlinien** (**Compliance**) (Internationalisierung bedeutet multiple Jurisdiktionen und Standards

Cloud Security Reference Model & Top Sicherheitsrisiken



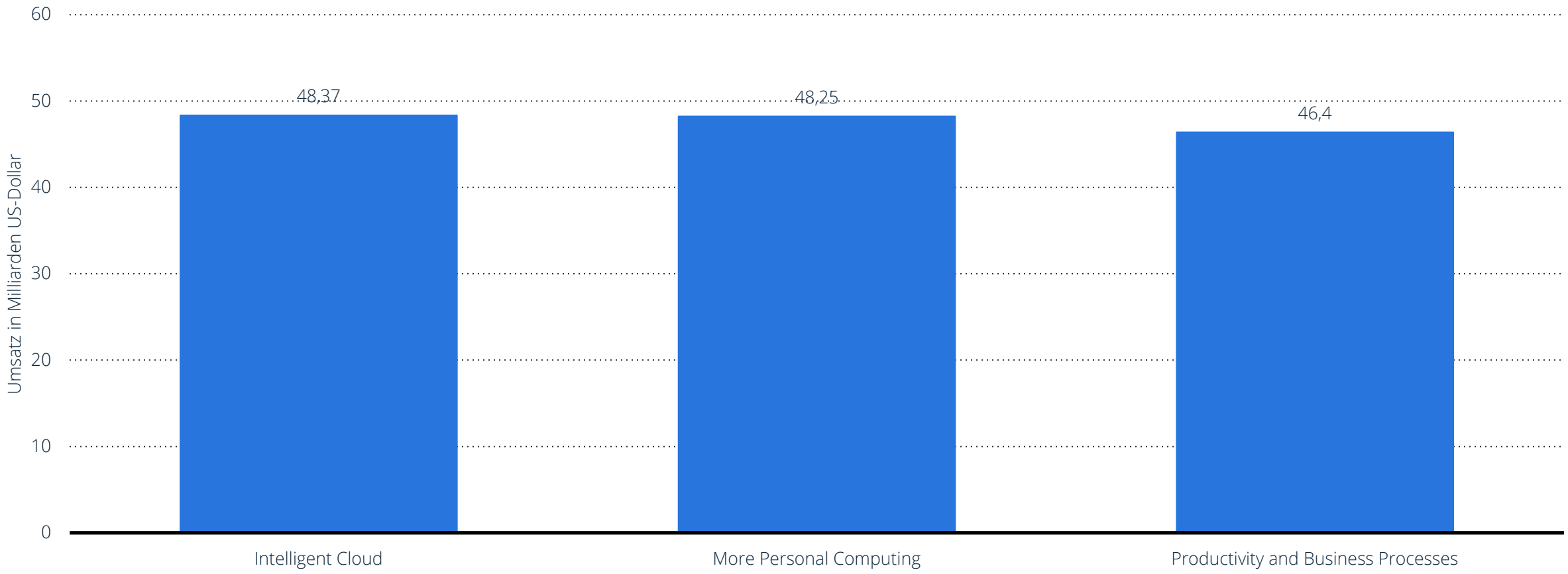
- SLAs beinhalten u.U. nicht alle Sicherheitsaspekte
- Eingegrenzte Portabilität von Daten, Anwendungen und Diensten (Lock-In)
- Ressourcen könnten für andere sichtbar sein (Isolation Failure)
- Richtlinien könnten nicht eingehalten werden (Compliance Risk – Verlust einer Zertifizierung)
- Daten könnten unsachgemäß behandelt werden und gglfs. nicht endgültig gelöscht werden

Source: https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf

Cloud Anbieter | **Auswahl**

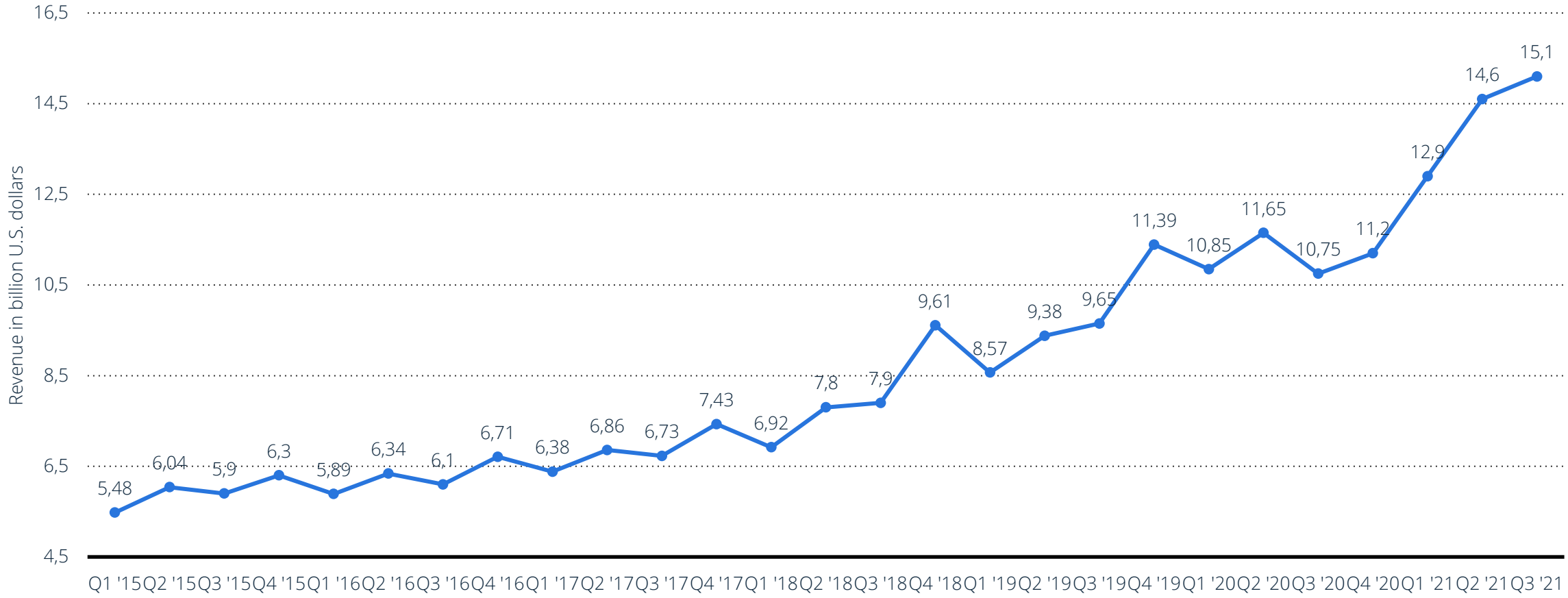
- Microsoft Azure
- Google Cloud | Firebase
- Amazon Web Services
- Hewlett Packard Enterprise
- Digital Ocean
- Timewarp
- Oracle Cloud
- IBM Cloud
- Alibaba Cloud

Microsoft Azure | **Umsatz von Microsoft 2020 nach Bereichen**



https://de.statista.com/statistik/daten/studie/192095/umfrage/umsatz-von-microsoft-nach-businessbereichen/

Microsoft Azure | **Umsatz "Intelligent Cloud" Segment**



https://www.statista.com/statistics/508993/worldwide-microsoft-intelligent-cloud-revenue-by-quarter/

Microsoft Azure | **Cloud Dienste**

- Microsoft Azure
  - Berechnungen/Programme (Compute)
  - Speicher (Storage)
- Microsoft 365
  - Office
  - Exchange
  - Sharepount
  - One Drive Teams
- Dynamics 365
  - Customer Relation Management (CRM) und
  - Enterprise Ressource Planning (ERP) Lösungen

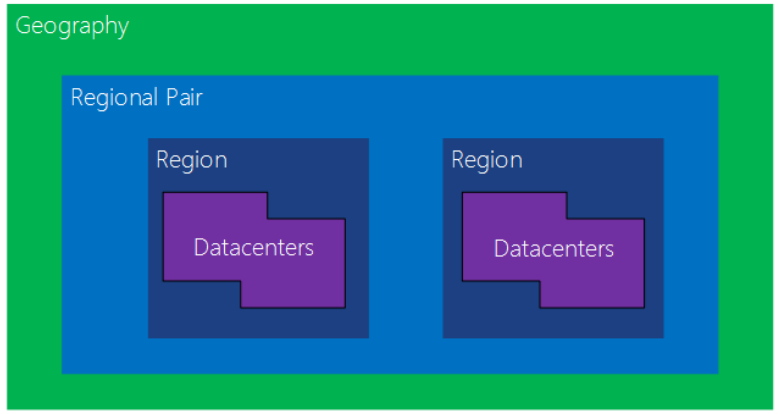
Mehr als 100 individuelle Dienste wie Rechner, Container, Datenbanken, Netzwerke, Speicher, Internet of Things (IoT), …

54 geografische "Azure" Regionen wie Westeuropa, Frankreich, Deutschland, westliche U.S., Japan, …

In 140 Ländern verfügbar

Microsoft Azure | **Infrastruktur**

- Regionen
  - Datacenter mit einem durch Latenz definierten Perimeter
  - Verbunden durch regionale Netzwerke mit geringen Latenze
- Geografisch
  - Diskreter Markt mit üblicherweise 2 Regionen (Redundanz)
  - Erhält Datenhoheit und Compliance-Grenzen
- Verfügbarkeit
  - Mehr als ein Datacenter mit unabhängiger Stromversorgung, Kühlung und Netzwerk
  - Physikalisch getrennte Orte innerhalb einer Region
  - 99,99% Uptime für Dienste durch 2 oder mehrere virtuelle Maschinen in 2 oder mehreren Zonen derselben Region



https://docs.microsoft.com/en-us/azure/best-practices-availability-paired-regions

Microsoft Azure | **Übersicht Dienste**

- Machine Learning
  - Bots, text-to-speech, search, ...
- Analyse
  - Log analytcs, data warehouse, Apache Hadoop/Spark, ...
- Compute und Container
  - Virtual machines, container, Kubernetes, SQL server, ...
- Datenbanken
  - MS SQL, PostgreSQL, Redis, ...
- Developer Tools und DevOps
  - Visual Studio, SDKs, blockchain, ...

- Identity
  - Active Directory Synchronizaton , Multifactor Authentcation, ...
- Integration
  - Service bus, ...
- IoT
  - Data hubs, central device management, ...
- Netzwerke
  - Content delivery, load balancer, VPN, ...
- Speicher
  - Disks, fle, BLOB, ...

Microsoft Azure | **Übersicht Preise**

Category	Instance	vCPU	RAM	Storage	Price/Month
General purpose	B1S	1	1 GiB	4 GiB	~€6,75
General purpose	D64a v4	64	256 GiB	1.600 GiB	~€3.319,05
Compute optimised	F72s v2	72	144 GiB	576 GiB	~€3.312,21
Memory optimised	M128ms	64	3.892 GiB	4.096 GiB	~€16.816,31
Storage optimised	L80s v2	80	640 GiB	10 x 1,9 TB	~€5.182,98
HPC (RDMA)	H16mr	16	224 GiB	2.000 GiB	~€1.662,66

Source: https://azure.microsoft.com/de-de/pricing/details/virtual-machines/windows/

DIHSÜD

FH JOANNEUM
University of Applied Sciences

Das World Wide Web (WWW)

Workshop
WebEngineering & Server Technologies

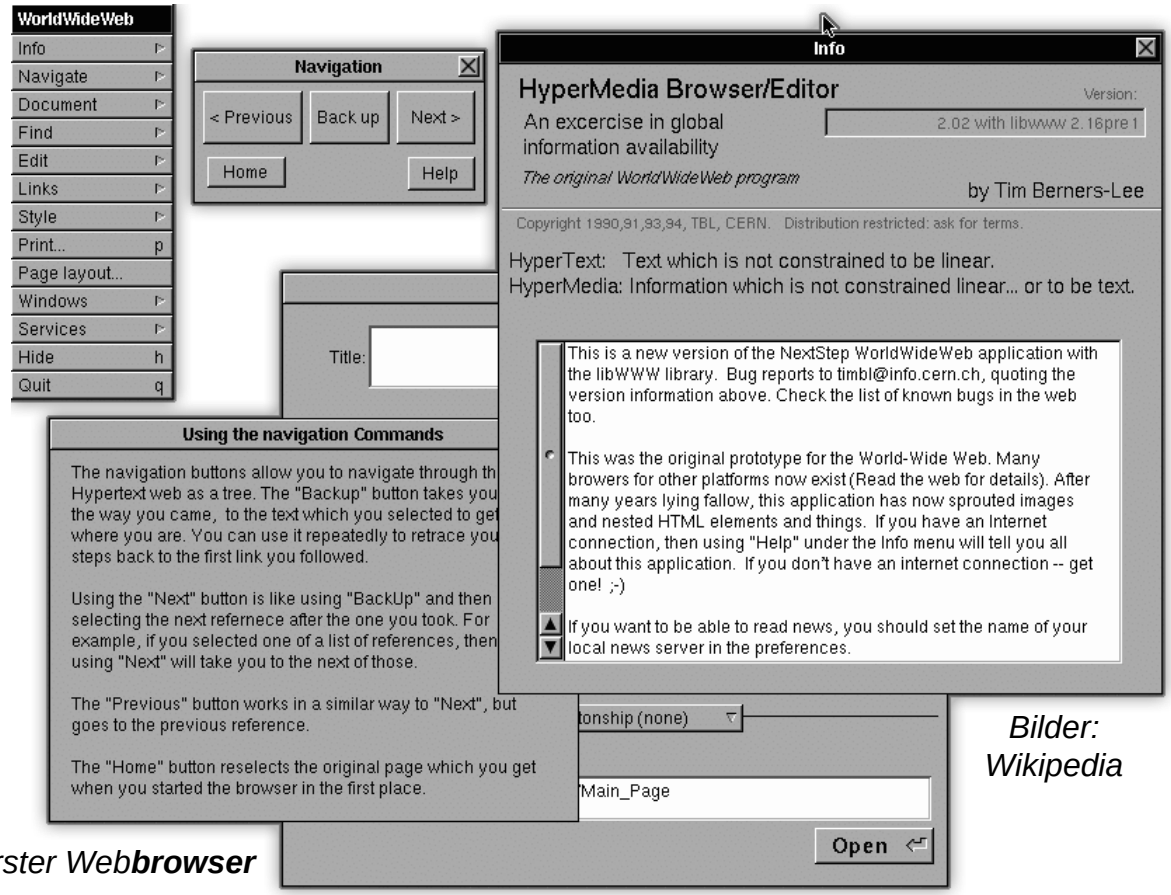
Das **World Wide Web (WWW)**

Historisches Logo des WWW von Robert Cailliau

*Erster Web**server***

*Erstes **Bild** über das WWW*

- Das **WWW – World Wide Web** ist einer der wichtigsten Dienste, der das Internet erst populär gemacht hat
- Es wurde 1988 bis 1991 von **Tim Berners-Lee** (Cern) entwickelt
- Das WWW ist **NICHT** das Internet !!



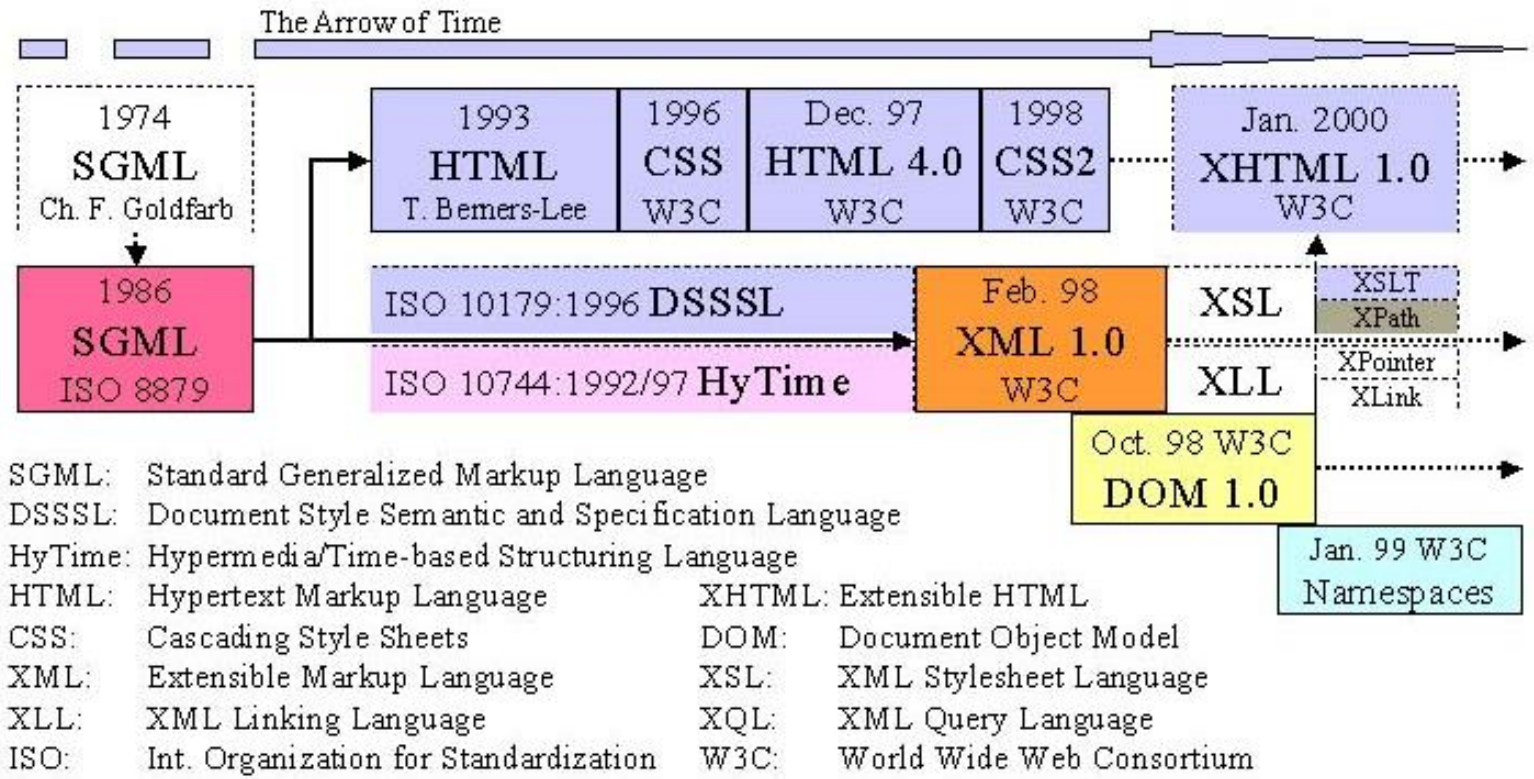
*Erster Web**browser***

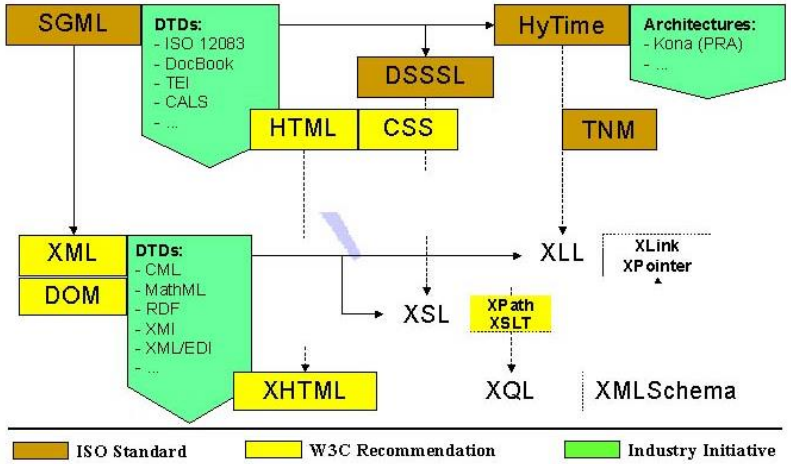
Bilder: Wikipedia

Kerntechnologien

- Drei Kerntechnologien
  - **Hyper Text Transfer Protocol** (HTTP, aktuelle Version 2.0)
  - **Hypertext Markup Language** (HTML, aktuell in der Version 5)
  - **Unified Resource Identificators und –Locators** (URI,URL)
- Spätere Zusätze
  - **Cascading Styleheets** (CSS)
  - **Javascript** (aktuelle Version ist ECMAScript 2020)
  - **Document Object Model** (DOM, die Baumstruktur einer HTML Seite)
- Bilder im Web
  - Graphics Interchange Format (GIF), Portable Network Graphics (PNG), GIF, PNG und Joint Photographic Experts Group (JPEG) und andere

Standards





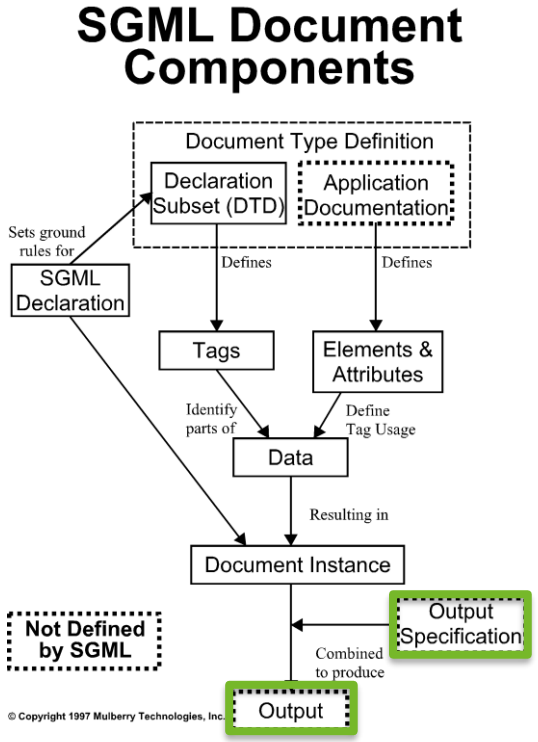
Quelle: https://www.jmir.org/2000/2/e12/

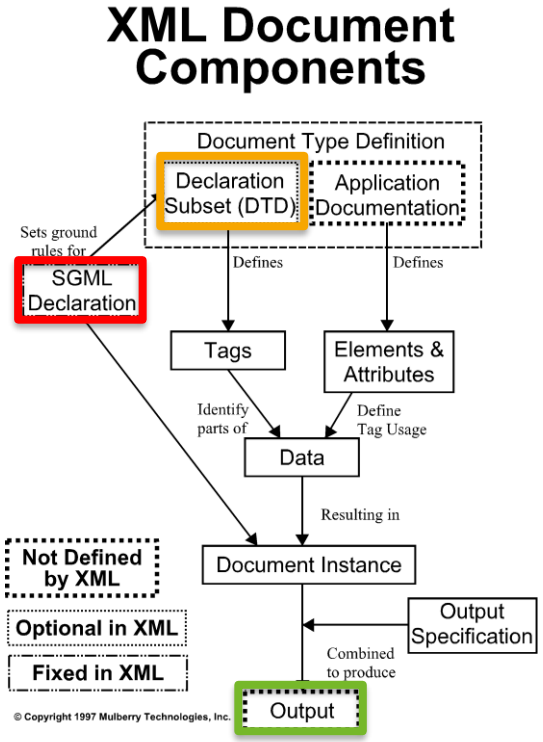
Bekannte Sprachen im Vergleich

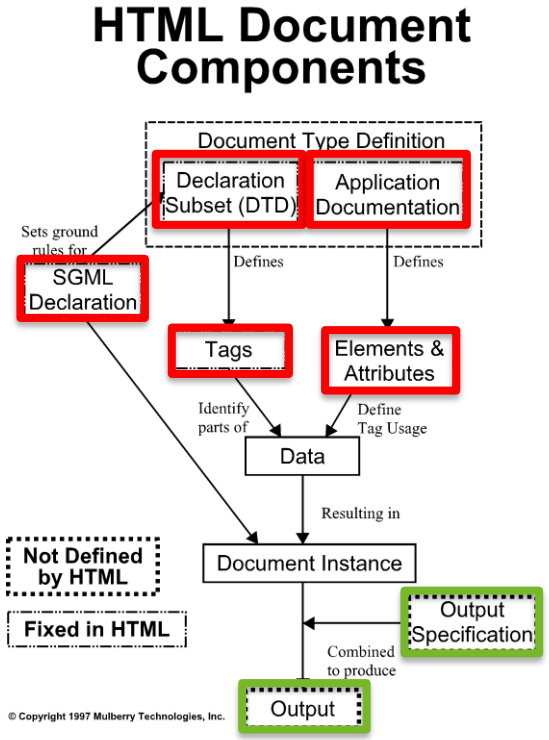
Beschreibung, wie eine Sprache definiert wird

Einfacher gemachte Teilmenge von SGML

Angewandtes SGML







SGML Deklaration: Regeln der Syntax

DTD Deklaration: Regeln der Struktur

Quelle: http://www.psyhyde.com/faq/index.php?action=artikel&cat=3&id=5&artlang=en-us

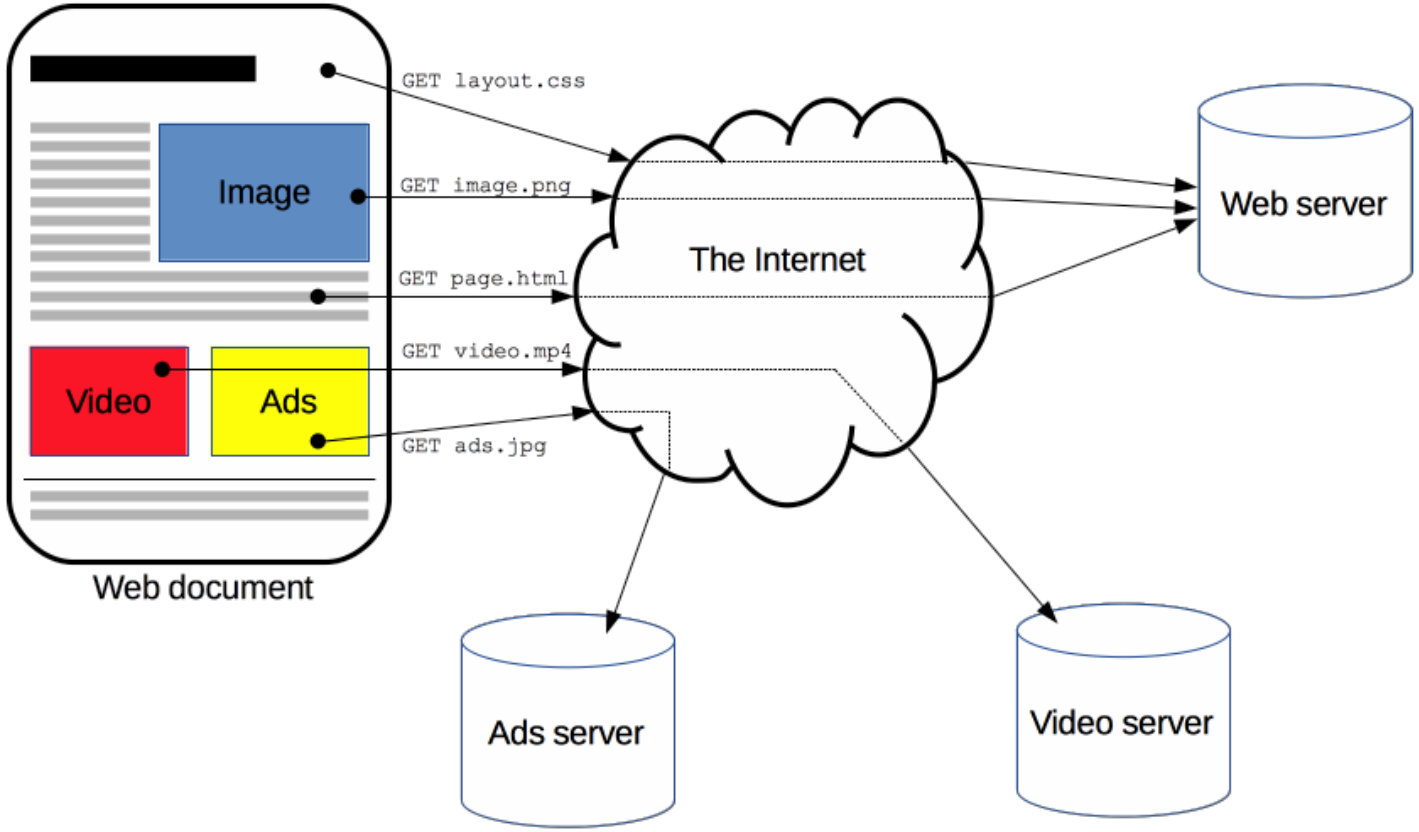
Organisationen

- **W3C** (World Wide Web Consortium)
  - Seit 1994
  - Definition von WWW-Standards (HTML, CSS, XML, SVG uvm.)
  - Ablauf zum Standard (= W3C-Recommendation)

    Working Draft → Last Working Draft → Candidate Recommendation → Proposed Recommendation → Recommendation

    Finanzierung über Mitgliederorgansationen & Spenden
- **WHATWG** (Web Hypertext Application Technology Working Group)
  - Browserhersteller: Apple, Mozilla, Google, Microsoft
  - Etablierung aufgrund langsamer Recommendation-Prozesse des W3C
  - HTML5 bzw. HTML Living Standard

Hyper Text Transfer Protocol (HTTP) | 1989



https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

URI & URL

- **URI** (Uniform Resource Identifier) `<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"`
  - RFC3986
  - Eine „compact sequence of characters that identifies an abstract or physical resource“
  - muss nicht physisch existieren, aber eindeutig sein
  - sieht oftmals aus wie URL

- **URL** (Uniform Resource Locator) https://elearning.fh-joanneum.at/my/
  - Identifikation von Online-Inhalten
  - Ist eine Spezialform eines URI
  - Aufbau:
    `<Protocol>://[<Username>:<Password>@]<Host>[:<Port>][/<Path>[?<Query>[#Fragment]]]`

URL Encoding

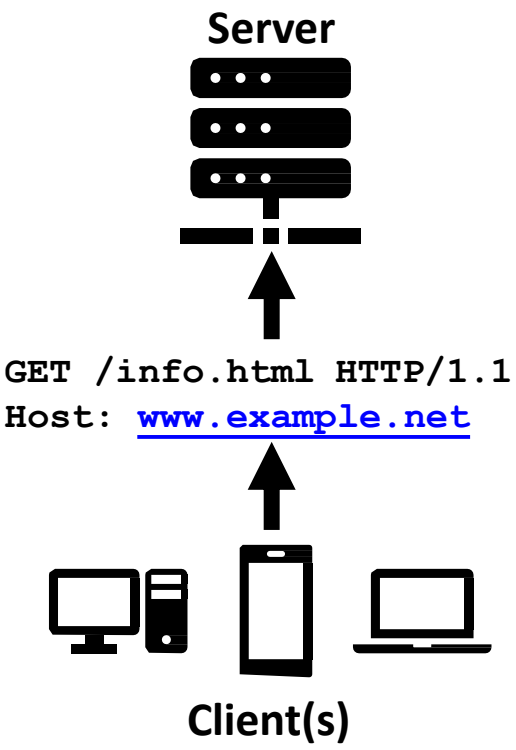
- ASCII-Basiszeichensatz
- Reservierte Zeichen
- Lösung:
  - Percent-Encoding z.B. Leerzeichen à%20 (vgl. RFC3986)
  - Alternativ internationale URLs (vgl. RFC3987)
  - Tipp: Vermeiden Sie Sonderzeichen in URLs und bilden Sie Leerzeichen als "-" ab. (Nicht "_")

Einführung | **Hypertext Transfer Protocol (HTTP)**

- Übertragungvon Inhalten über das Netzwerk
- Beispiel: Ein Client fordert über das Internet von einem Webserver eine Website an
- Zustandslos
  - Requests stehen für sich alleine
  - Sitzungsidentifikationmittels Session-Ids über z.B. Cookies
- Aufbau:
  - Header: Metainformationen
  - Body: Payload (HTML,…)

Site | Die gesamte Webpräsenz unter einer Domain
Webseite | Eine einzelne Seite unter einer URL
Homepage | Die Startseite einer Website

HTTP | **Flow**

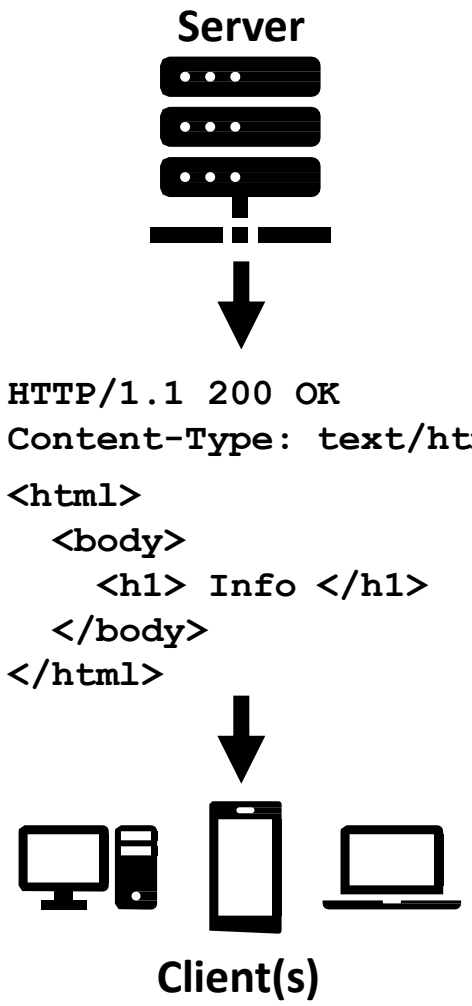


Client Request

- Browser auf irgendeinem Gerät (PC, Laptop, Smartphone)
- sendet eine (HTTP-) Anfrage
- an den (Web-) Server.

Server Response

- Server (Virtuelle Maschine, physisch in einem Rack)
- schickt eine (HTTP-) Antwort
- mit einem (HTML) Dokument
- an den (Web-) Client.



Client

rendert das (HTML-) **Dokument am Bildschirm**

Client

HTTP | **Protokoll**

Request

Response

Pfad

Protokoll Version

Methode

```
GET /info.html HTTP/1.1
Host: www.example.net
```

Header

```
HTTP/1.1 200 OK
Content-Type: text/html
```

Body

```
<html>
  <body>
    <h1> Info </h1>
  </body>
</html>
```

HTTP | **Methoden**

"Sicher"
Keine Aktion am Server

GET	Abrufen einer Ressource
HEAD	Inspizieren der Ressourcen Header

Nachrichten mit "Body"
Daten zum Server senden

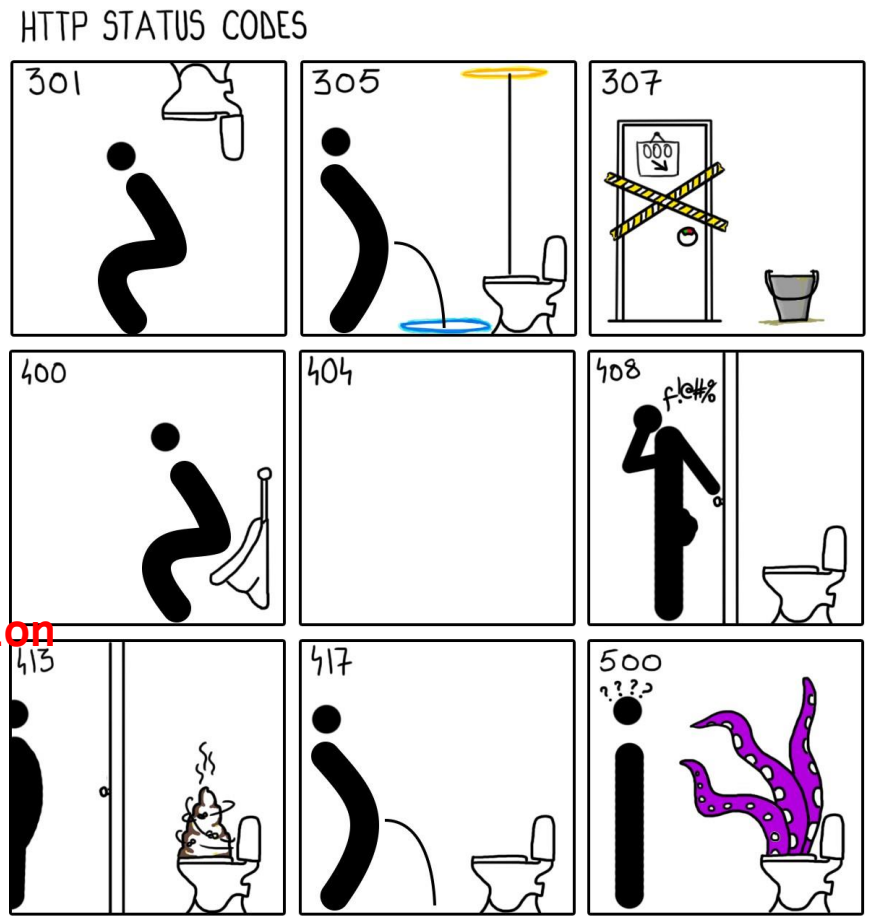
POST	Senden von Daten zur Verarbeitung
PUT	Ablegen von Daten am Server
PATCH	Teilweise Modifikation von Ressourcen

TRACE	Zurücksenden der erhaltenen Nachricht
OPTIONS	Abrufen der Server Eigenschaften
DELETE	Löschen einer Ressource

HTTP | **Status Codes**

2xx Success	200	OK
3xx Redirection	301	Moved Permanently
	305	Use Proxy
	307	Temporary Redirect
4xx Client Error	400	Bad Request
	404	Not Found
	408	Request Timeout
	413	Payload Too Large Expectation
	417	Failed
5xx Server Error	500	Internal Server Error

*1xx Informational Response



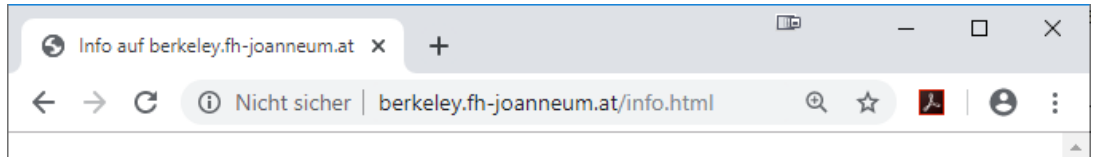
Fallbeispiel: **Client - Server Kommunikation**

- **Client**
  Web Browser
  z.B. Google Chrome)

- **sendet**
  über das Netzwerk eine
  HTTP*- **Anfrage**

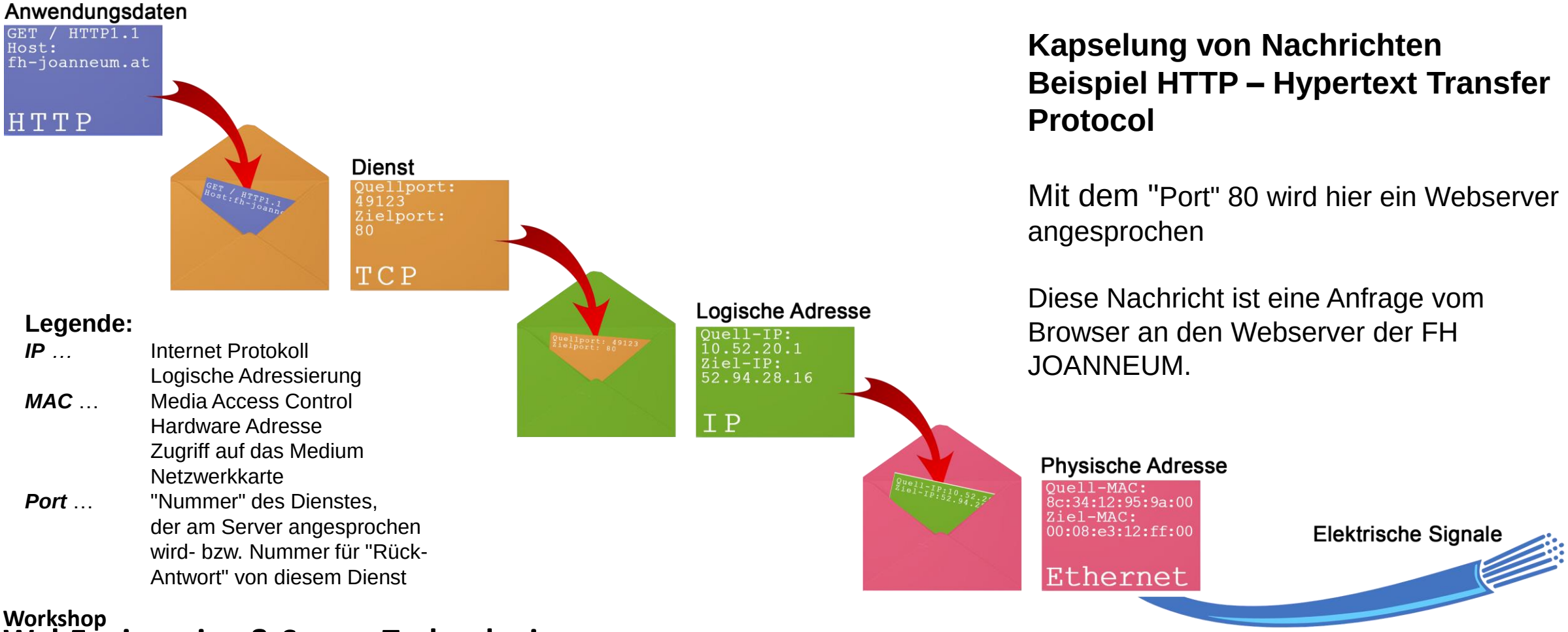
- an den
  Web-**Server**
  z.B. Apache 2

*(*HTTP = Hypertext Transfer Protocol)*




```
GET /info.html HTTP/1.1
Host: berkeley.fh-joanneum.at
```

Fallbeispiel: **Nachrichten an den Server | Diagramm**



Legende:

IP … Internet Protokoll
Logische Adressierung

MAC … Media Access Control
Hardware Adresse
Zugriff auf das Medium
Netzwerkkarte

Port … "Nummer" des Dienstes, der am Server angesprochen wird- bzw. Nummer für "Rück-Antwort" von diesem Dienst

Kapselung von Nachrichten
Beispiel HTTP – Hypertext Transfer Protocol

Mit dem "Port" 80 wird hier ein Webserver angesprochen

Diese Nachricht ist eine Anfrage vom Browser an den Webserver der FH JOANNEUM.

Fallbeispiel: **Nachricht an den Server | Wireshark**

von Client (Browser)

von
Hardware Adresse
Client PC

an
Hardware Adresse
Gateway

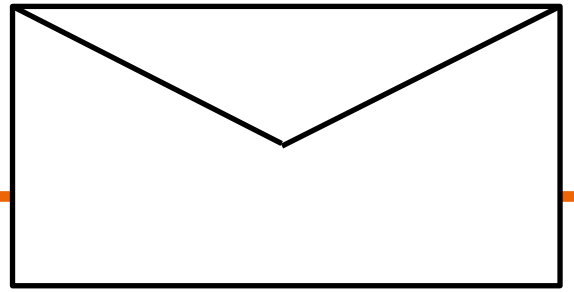
No.	Time	Source	Destination	Protocol	Length	Info
3382	22.991350	10.52.200.132	91.229.57.56	HTTP	698	GET /info.html HTTP/1.1

```
Frame 3382: 698 bytes on wire (5584 bits), 698 bytes captured (5584 bits) on interface 0
Ethernet II, Src: 40:b0:76:a2:42:07, Dst: 00:08:e3:ff:fd:2c
Internet Protocol Version 4, Src: 10.52.200.132, Dst: 91.229.57.56
Transmission Control Protocol, Src Port: 53638, Dst Port: 80, Seq: 1, Ack: 1, Len: 644
Hypertext Transfer Protocol
  GET /info.html HTTP/1.1\r\n
  Host: berkeley.fh-joanneum.at\r\n
  Connection: keep-alive\r\n
  Cache-Control: max-age=0\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: de-DE,de;q=0.9,en-US;q=0.8,en;q=0.7\r\n
  Cookie: _ga=GA1.2.1115739603.1574071763; _gid=GA1.2.764581842.1574071763\r\n
  If-None-Match: "9b-5979e1fe538ec-gzip"\r\n
  If-Modified-Since: Mon, 18 Nov 2019 12:31:19 GMT\r\n
```

Port für **HTTP**

von
Logische Adresse
Client PC

an
Logische Adresse
Webserver



Aus "ipconfig /all"
Netzwerkkarte von Client:
Physische Adresse : 40-B0-76-A2-42-07
IPv4-Adresse . . . . : 10.52.200.132
Standardgateway : **10.52.1.254**

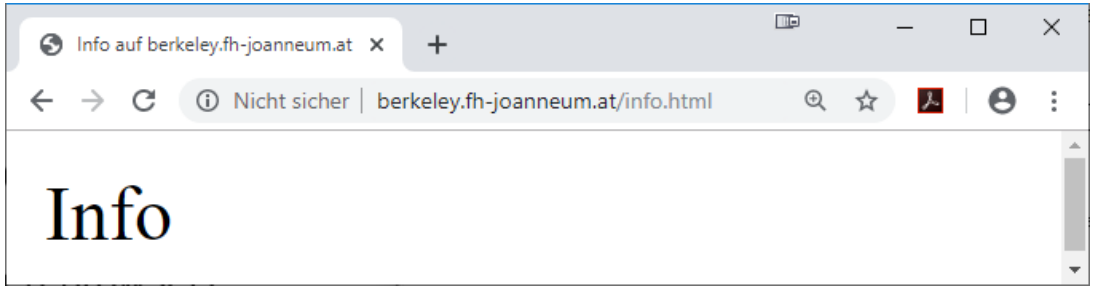
Aus "arp –a":
Schnittstelle: 10.52.200.132
Internetadresse Physische Adresse **10.52.1.254** **00-08-e3-ff-fd-2c**

Fallbeispiel: **Server – Client Kommunikation**

- Der **Server** sendet eine (HTTP-) **Antwort** an den **Client** (=Browser) zurück

```
HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Info auf berkeley.fh-joanneum.at</title>
  </head>
  <body>
        Info
  </body>
</html>
```

- und deren Inhalt zeigt ein Browser als (HTML) **Dokument** gerendert an



Fallbeispiel: **Server – Client Kommunikation | Wireshark**

von Server

von Hardware Adresse **Gateway**

an Hardware Adresse **Client PC**

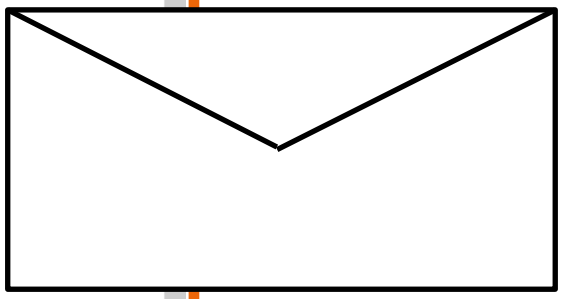
No.	Time	Source	Destination	Protocol	Length	Info
3483	13.250701	91.229.57.56	10.52.200.132	HTTP	559	HTTP/1.1 200 OK (text/html)

```
> Frame 3483: 559 bytes on wire (4472 bits), 559 bytes captured (4472 bits) on interface 0
> Ethernet II, Src: 00:08:e3:ff:fd:2c, Dst: 40:b0:76:a2:42:07
> Internet Protocol Version 4, Src: 91.229.57.56, Dst: 10.52.200.132
> Transmission Control Protocol, Src Port: 80, Dst Port: 54074, Seq: 1, Ack: 619, Len: 505
v Hypertext Transfer Protocol
  > HTTP/1.1 200 OK\r\n
    Date: Mon, 18 Nov 2019 13:20:42 GMT\r\n
    Server: Apache/2.4.25 (Debian)\r\n
    Last-Modified: Mon, 18 Nov 2019 12:31:19 GMT\r\n
    ETag: "9b-5979e1fe538ec-gzip"\r\n
    Accept-Ranges: bytes\r\n
    Vary: Accept-Encoding\r\n
    Content-Encoding: gzip\r\n
    Access-Control-Allow-Origin: *\r\n
  > Content-Length: 137\r\n
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.004144000 seconds]
    [Request in frame: 3481]
    [Request URI: http://berkeley.fh-joanneum.at/info.html]
    Content-encoded entity body (gzip): 137 bytes -> 155 bytes
    File Data: 155 bytes
v Line-based text data: text/html (11 lines)
```

Port für **HTTP**

von Logische Adresse **Webserver**

an Logische Adresse **Client PC**



DIHSÜD

FH JOANNEUM
University of Applied Sciences

APACHE
HTTP SERVER PROJECT

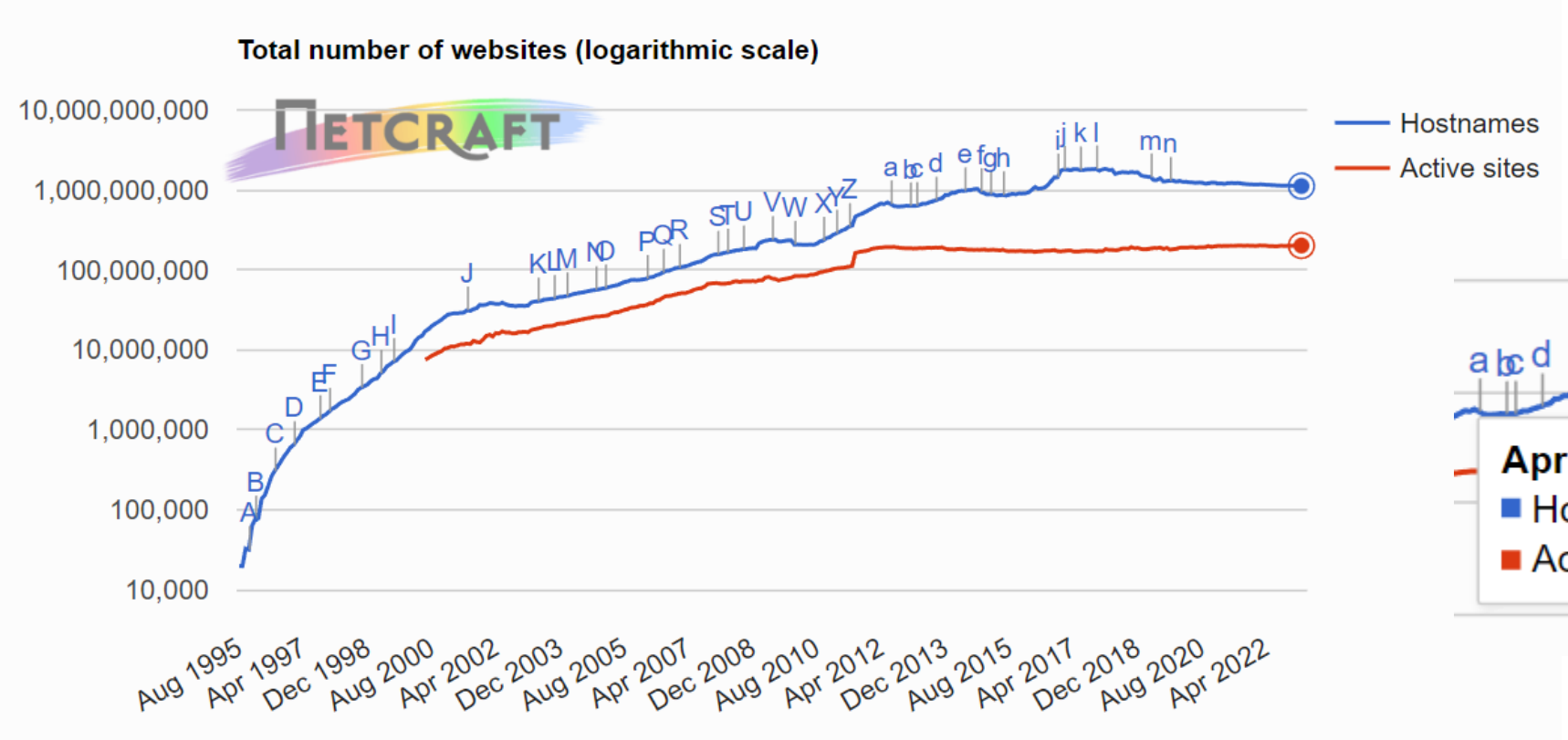
node
JS

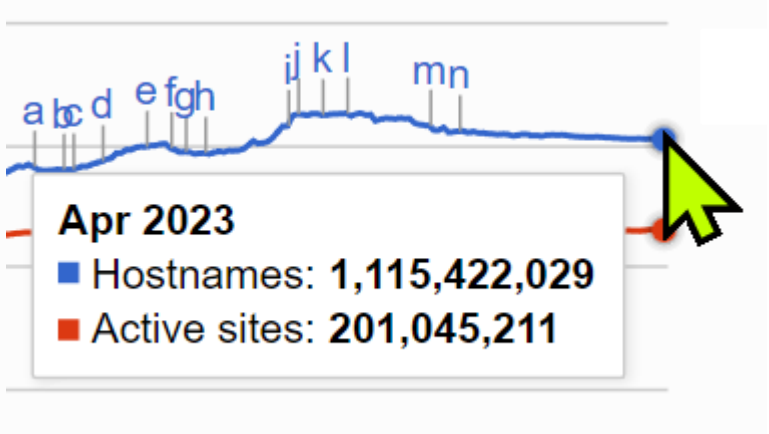
NGINX

Einige gern verwendete Webserver

Workshop
WebEngineering & Server Technologies

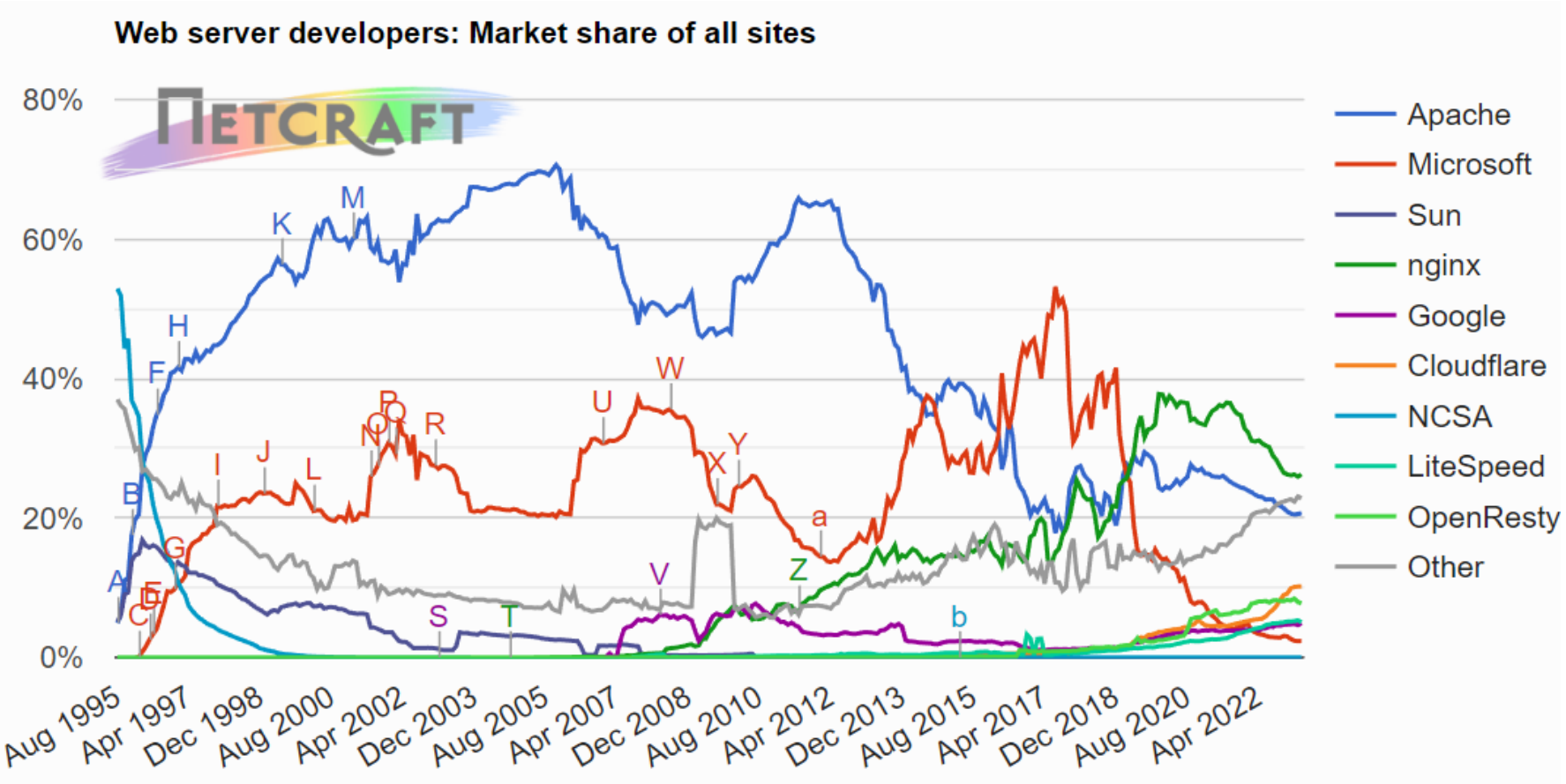
Websites | April 23

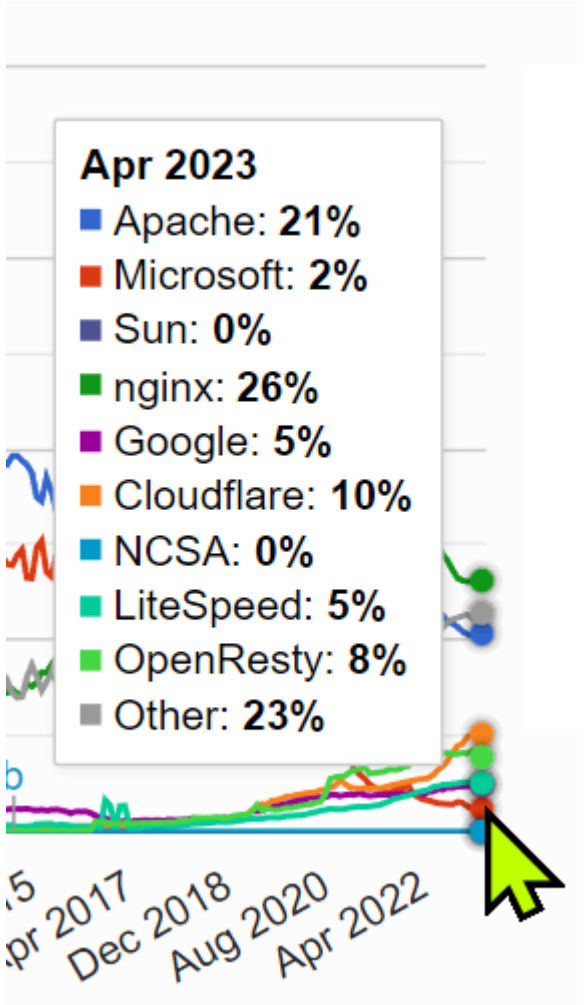




https://news.netcraft.com/archives/2023/04/27/april-2023-web-server-survey.html

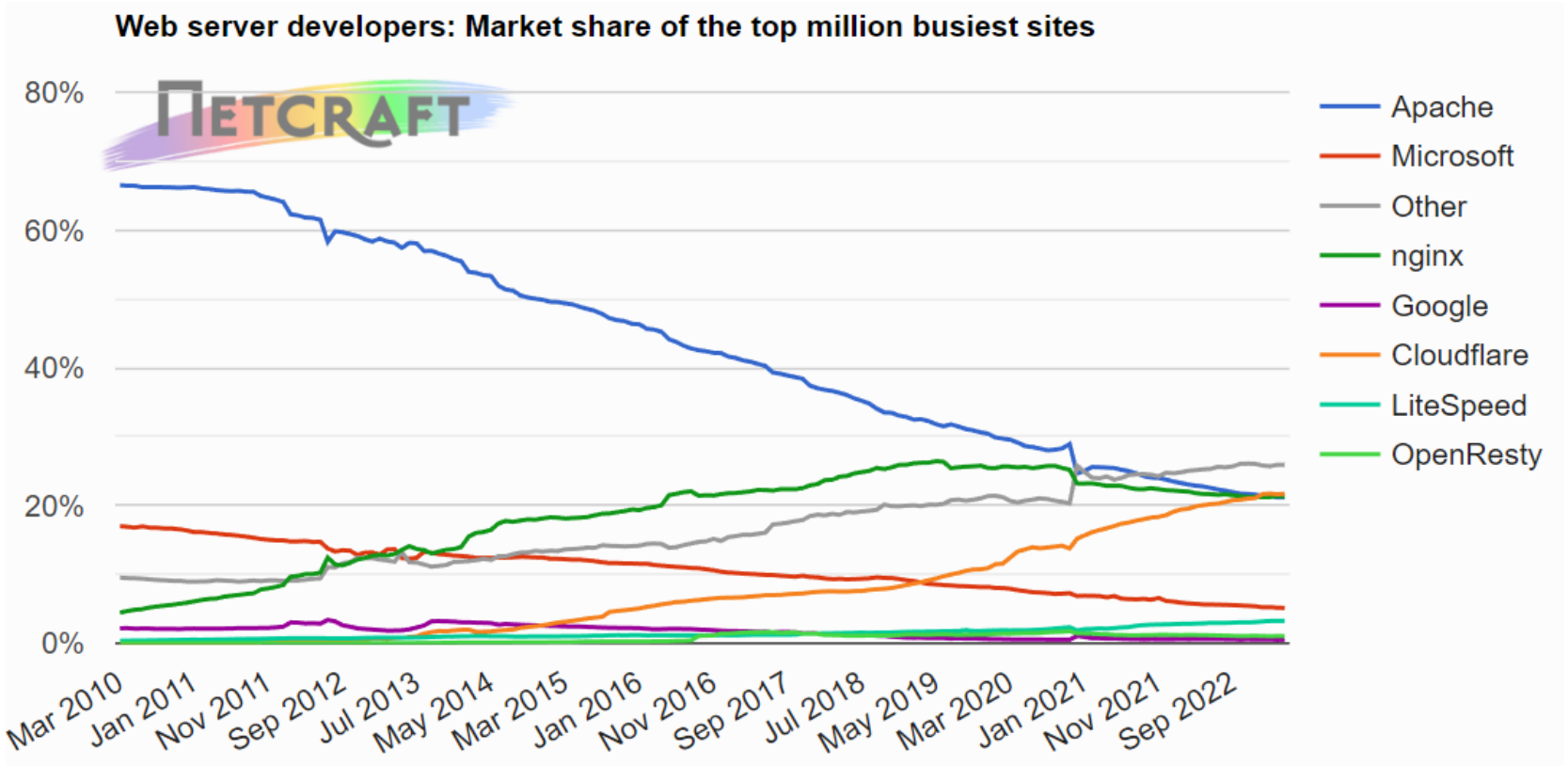
Webservers | April 23

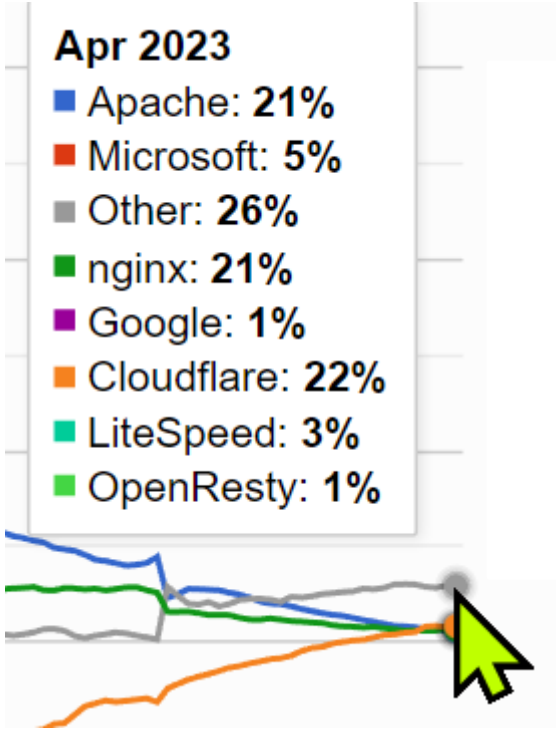




https://news.netcraft.com/archives/2023/04/27/april-2023-web-server-survey.html

Busiest Sites | April 23





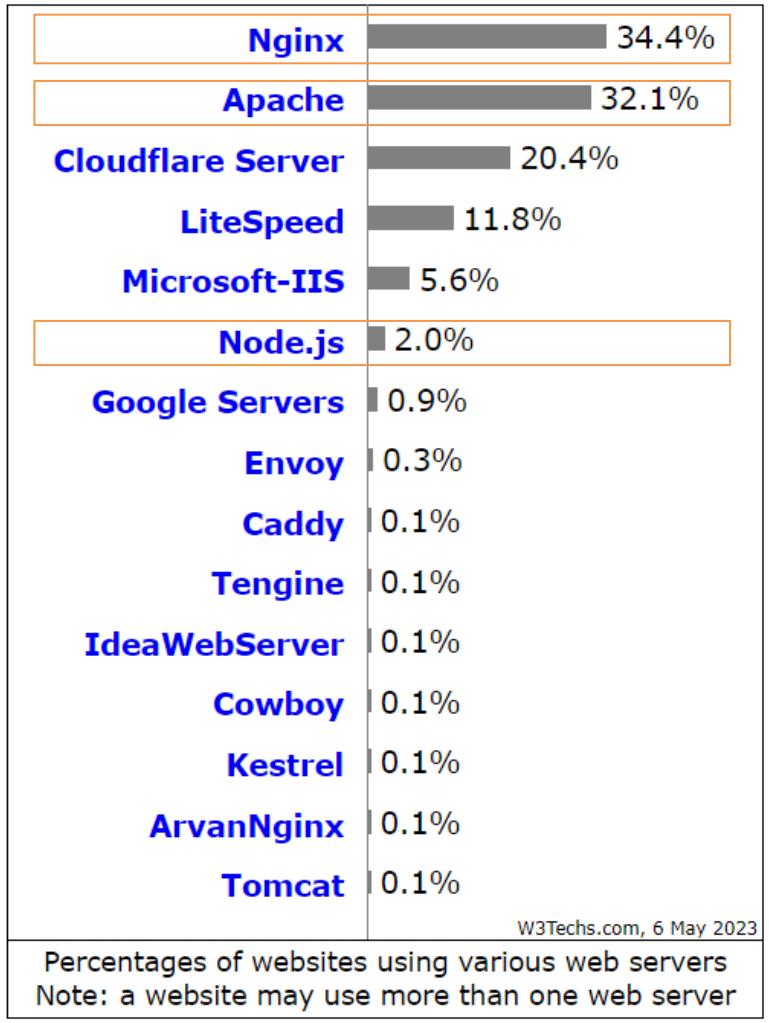
https://news.netcraft.com/archives/2023/04/27/april-2023-web-server-survey.html

Webserver | Top 2023

nginX +
- Performance! Streaming…!
- Statische Dateien!

Apache +
- .htaccess
- Viele Funktionen!

Beide zusammen? …. ;-

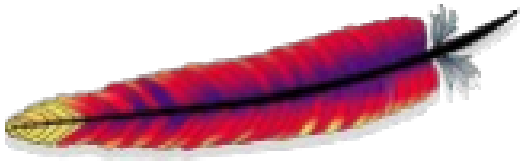


https://wpengine.com/whats-the-internet-built-on/

Nginx

- http://nginx.org
- BSD-License
- HTTP, E-Mail Proxy
- Russland
- Fast, stable, scalable
- Module: Load Balancing,...
- Linux, BSD, Solaris, Mac OSX, Windows XP, Server 2003

Apache



- http://www.apache.org
- Open Source (Apache Licence)
- Ursprünglich basierend auf NCSA HTTPd
- Unterstützt Perl, Python, Tcl, and PHP
- Unterstützt viele weitere Technologien als Plugins/Module
- Load Balancing
- Proxying

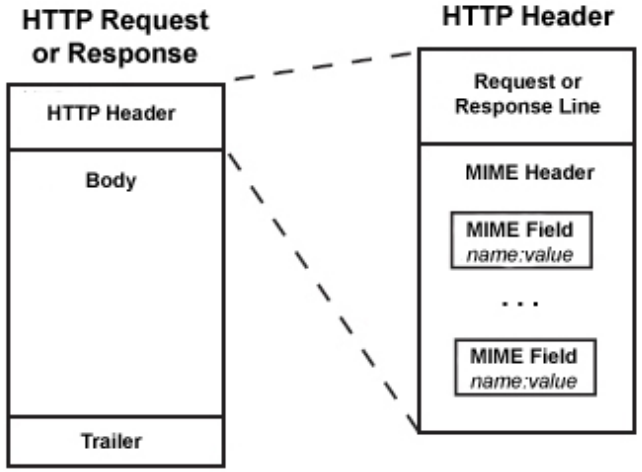
Node

- https://nodejs.org/en
- Open Source (MIT Licence)
- Non Blocking I/O und asynchrones Request Handling
- Weniger Arbeitsspeicher wegen Thread Pool
- Unterstützt viele weitere Technologien als Module
- Am meisten verwendete Technologie bei Webanwendungen

```
const http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {
    'Content-Type': 'text/html'
  });
  res.write('Hallo Welt');
  res.end();
}).listen(3000);
```

Protokoll HTTP

- **Stateles**
- **Request – Response**
- **Header – Body**
- **Request Methoden**
  (Die wichtigen sind …)
  - **GET**
  - **POST**
- **Protokoll Versionen**
  - HTTP/1.0 (RFC1945) – Pro Anfrage eine neue TCP Verbindung
  - HTTP/1.1 (RFC2616) – Persistente Verbindung möglich
  - HTTP/2.0 (RFC7540/1) – Anfragen zusammenfassen, Kompression, Binärdaten übertragen, Push-Verfahren (Server initiiert Komm.)



Methoden:

```
Method          = "OPTIONS"                ;
                | "GET"                    ;
                | "HEAD"                   ;
                | "POST"                   ;
                | "PUT"                    ;
                | "DELETE"                 ;
                | "TRACE"                  ;
                | "CONNECT"                ;
                | extension-method
extension-method = token
```

HTTP Methoden

- **GET**
  - Fordert unter Angabe einer URL eine Ressource vom Server.
  - Daten werden in der URL übertragen.
- **POST**
  - Sendet beliebige Datenmenge an Server im Inhalt (Body) einer Nachricht
- **HEAD**
  - Verlangt nur den Kopf (Header) einer Nachricht vom Server
- **PUT**
  - Zum Hochladen einer Ressource unter Angabe der Ziel-URL
  - Damit können Ressourcen auch modifiziert werden
- **DELETE**
  - Löscht eine Ressource am Server
- **TRACE**
  - Lieferte einen Request zurück, wie ihn der Server bekommen hat.
  - Zum Debuggen verwendet- aber Sicherheitsrisiko!
- **OPTIONS**
  - Liefert vom Server unterstützte Methoden und Merkmale
- **CONNECT**
  - Für HTTP Tunnel mittels Proxy- Leitet die TCP Verbindung weiter-
  - ab dann nur mehr TCP und kein HTTP mehr

Features

- HTTP **Cache**
  - **Zwischenspeichern** von Ressourcen am Client
  - Feld: "Cache-Control:" Feld
    - Kein Caching (no-store)
    - Ablaufzeit (max-age=xxxx)
    - Validierung mit Cache (must-revalidate)
    - Privat und Öffentlich (private, public)
      Einzelbenutzer vs. Proxy

- **Same Origin Policy**
  - **Schutz vor Angriffen-** von einer Site können nur **Anfragen an die gleiche Herkunft** (*Protokoll, Domain, Port, URL*) gestellt werden
  - Ausnahme sind Subdomains
  - Feld "**Access-Control-Allow-Origin**":
    - *
    - <origin>

CORS - Cross Origin Resource Sharing

`Access-Control-Allow-Origin: https://developer.mozilla.org`

Protokollbeispiel 1

GET Request

```
GET /test?name=Mathias HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64;
rv:45.0) Gecko/20100101 Firefox/45.0
Accept:
text/html,application/xhtml+xml,application/xml;
q=0.9,*/*;q=0.8
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate



Connection: keep-alive
```

POST Request

```
POST /test HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64;
rv:45.0) Gecko/20100101 Firefox/45.0
Accept:
text/html,application/xhtml+xml,application/xml;
q=0.9,*/*;q=0.8
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 12
Connection: keep-alive

name=Mathias
```

Protokollbeispiel 2

OPTIONS Request

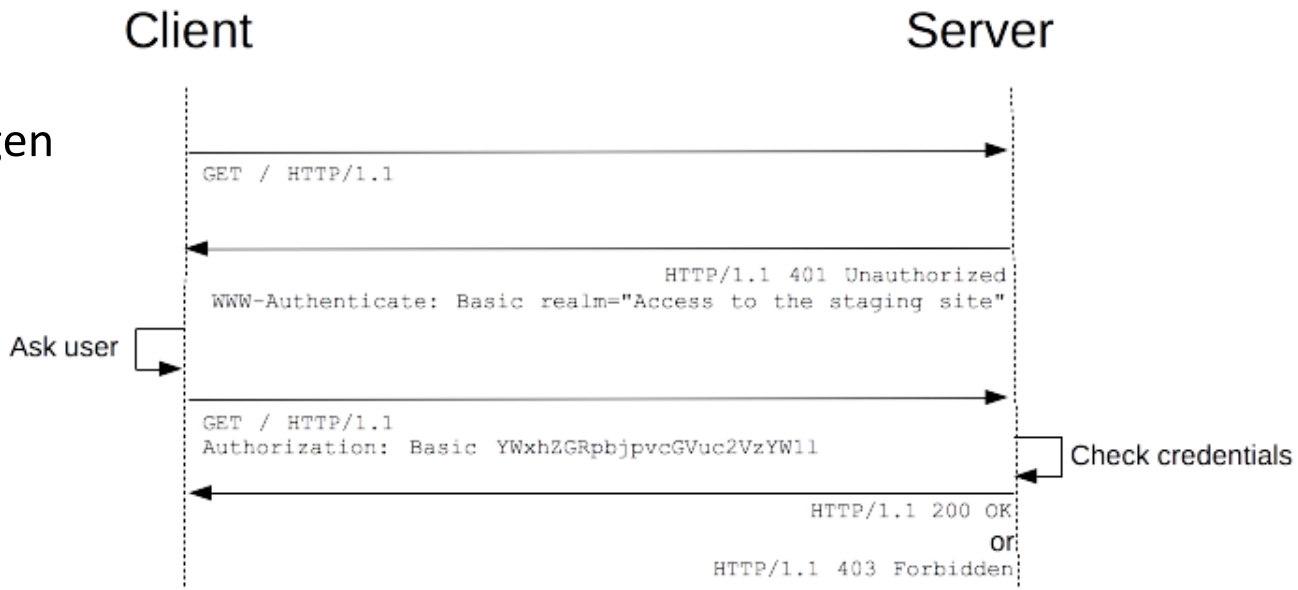
```
OPTIONS /test/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64;
rv:45.0) Gecko/20100101 Firefox/45.0
Accept:
text/html,application/xhtml+xml,application/xml;
q=0.9,*/*;q=0.8
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

OPTIONS Response

```
HTTP/1.1 200 OK
Date: Fri, 12 May 2015 11:35:43 GMT
Server: Apache/2.4.18 (Win32) OpenSSL/1.0.2e
PHP/7.0.4
Allow: GET,HEAD,POST,OPTIONS,TRACE
Content-Length: 0
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8
```

Authentifizierung & Autorisierung 1

- Feld "WWW-Autenticate <Typ> <Bereich>":
  - **Basic**
    Klartext – Verschlüsselung muss über HTTPS erfolgen
    Base64 kodiert
  - **Digest Access**
    Prüfsumme (Hashcode) aus Daten des Headers
    Name, Kennwort, Enthaltene Zeichenfolgen,
    Methode, usw. Mit MD5 als Standard unsicher
  - **Bearer**
    Oauth Bearer Token (IT & Mobile Security Thema)
  - **und andere**!
- Feld "Authorization <Typ> <Name:Kennwort>":
  - Name:Kennwort ist unter Basic BASE64 kodiert



Cookies

- Feld "Set-Cookie <name>=<wert>":
  - Für Session Management, Personalisierung, Tracking
  - Direktiven definieren u.a. Ablaufzeit ("Expires")

- Feld "Cookie <Liste Name Wert Zuweisungen>:

```
Set-Cookie: PHPSESSID=298zf09hf012fh2;
            Expires=Wed, 21 Oct 2019 07:28:00 GMT

Cookie: PHPSESSID=298zf09hf012fh2;
```

Virtual Hosting (VH)

Es gibt

Namensbasiertes

Portbasiertes

IP-basiertes

Virtual Hosting um …

- … Hosting multipler Domänen auf einem Server zu betreiben
- … Ressourcen effizient einzusetzen
- … für viele Eventualitäten Sites zu konfigurieren

Namensbasiertes VH

- **Unterschiedliche Domänennamen** verweisen auf **unterschiedliche Verzeichnisse auf einem Server**
- Der **Port ist immer der gleiche**

```
# Ensure that Apache listens on port 80
Listen 80

# Listen for virtual host requests on all IP addresses
NameVirtualHost *:80

<VirtualHost *:80>
          DocumentRoot /www/example1
          ServerName www.example.com

# Other directives here

</VirtualHost>

<VirtualHost *:80>
          DocumentRoot /www/example2
           ServerName www.example.org

# Other directives here

</VirtualHost>
```

http://httpd.apache.org/docs/2.2/vhosts/examples.html

Port-basiertes VH

- **Unterschiedliche Ports** verweisen auf **unterschiedliche Verzeichnisse auf einem Server**, können aber den gleichen Domänennamen haben
- Der **Port ist unterschiedlich**

```
Listen 80
Listen 8080

NameVirtualHost 172.20.30.40:80
NameVirtualHost 172.20.30.40:8080

<VirtualHost 172.20.30.40:80>
ServerName www.example.com
DocumentRoot /www/domain-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
ServerName www.example.com
DocumentRoot /www/domain-8080
</VirtualHost>

<VirtualHost 172.20.30.40:80>
ServerName www.example.org
DocumentRoot /www/otherdomain-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
ServerName www.example.org
DocumentRoot /www/otherdomain-8080
</VirtualHost>
```

Hier im Beispiel eine Kombination aus namensbasiertem und portbasierten VH

http://httpd.apache.org/docs/2.2/vhosts/examples.html

IP-basiertes VH

- **Unterschiedliche IPs auf unterschiedliche Verzeichnisse auf einem Server**, können aber den gleichen Domänennamen haben
- Beide werden am selben Server gehostet, haben aber unterschiedliche IPs

```
Listen 80

<VirtualHost 172.20.30.40>
   DocumentRoot /www/example1
   ServerName www.example.com
</VirtualHost>

<VirtualHost 172.20.30.50>
   DocumentRoot /www/example2
   ServerName www.example.org
</VirtualHost>
```

http://httpd.apache.org/docs/2.2/vhosts/examples.html

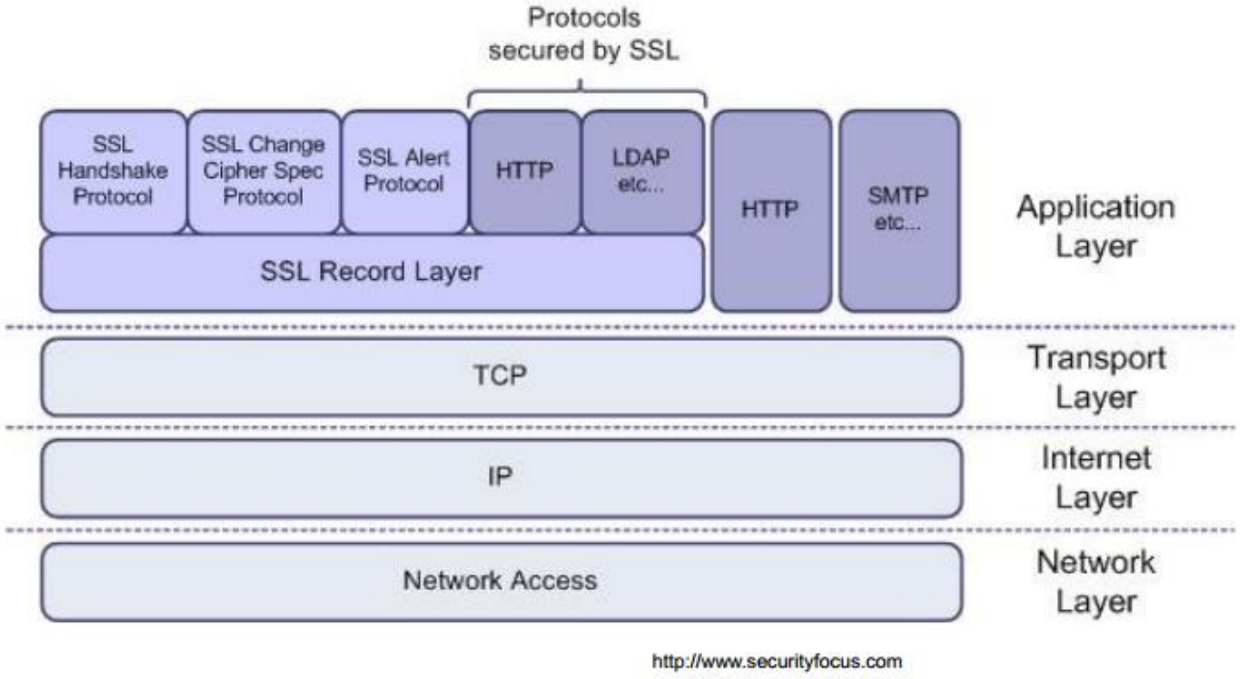
Transport Layer Security (TLS)

- Secure Socket Layer (alt)
- Transport Layer Security (TLS)
- HTTPS Port 443
- Sichere Kommunikation
  - Authentication
  - Encryption

- State full Connection
- Handshake Prozedur
  - Client sendet Liste unterstützter “Cipher” Suiten
  - Server wählt die stärkste
  - Server sendet Identifikation (Digitales Zertifikat)
    Server name, CA, public key
  - Client kann die CA kontaktieren, um die Validität zu überprüfen
  - “Key exchange” mit “Public key cryptography”

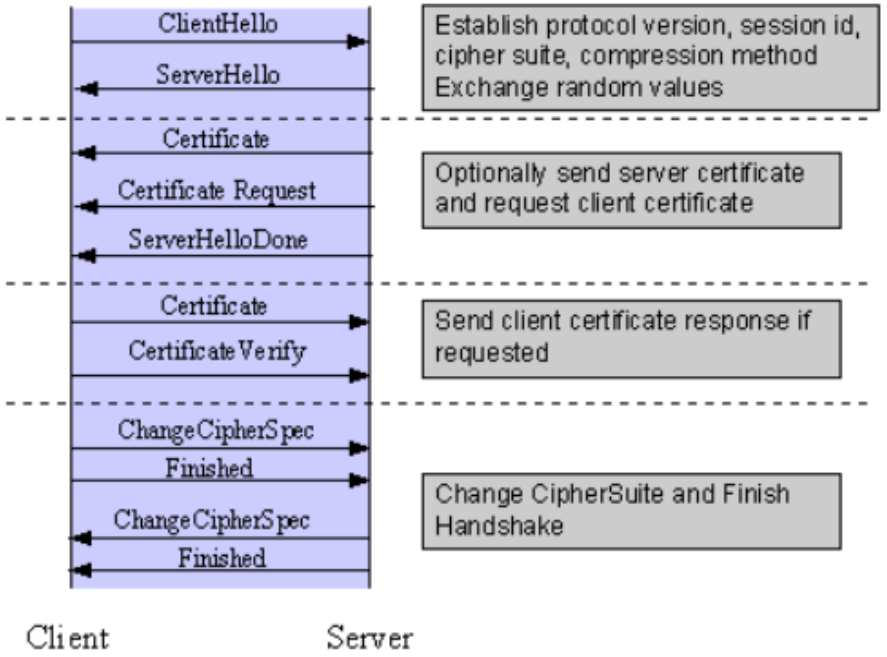
Transport Layer Security (TLS) | **OSI & Handschake**

TLS im OSI Referenzmodel



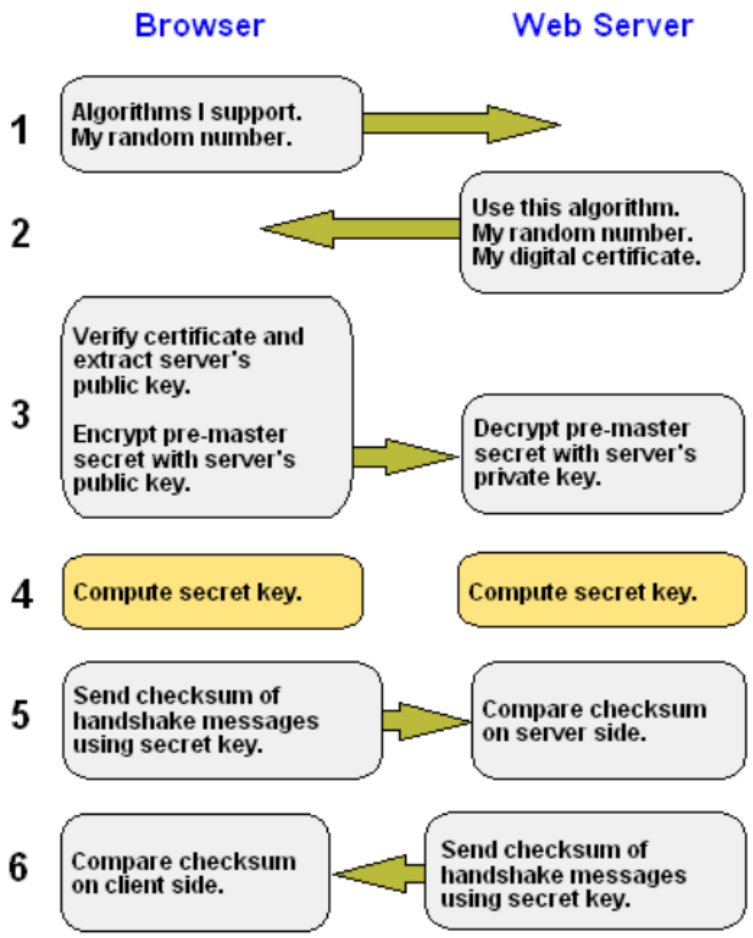
TLS Handshake Protocol

Identifikation & Authentifizierung
Algorithmen und Schlüssel werden ausgetauscht
4 Phasen



Transport Layer Security (TLS) | **Handshake**

1. Der Client sendet ein "Hello" an den Server – zusätzlich werden die Fähigkeiten des Clients übertragen
2. Der Server antwortet mit seinen Fähigkeiten und seinem Zertifikat, dass von einem Root-Zertifikat signiert wurde
3. Der Client überprüft das Zertifikat und sendet einen Vorab-Schlüssel, verschlüsselt mit dem öffentlichen Schlüssel des Servers
4. Beide generieren den gemeinsamen Schlüssel
5. Beide vergleichen die Nachrichten mittels Prüfsummen auf beiden Seiten (6)



DIHSÜD

FH JOANNEUM
University of Applied Sciences

Hypertext Markup Language (HTML)

Cascading Stylesheets (CSS)

Workshop
WebEngineering & Server Technologies

Werkzeuge

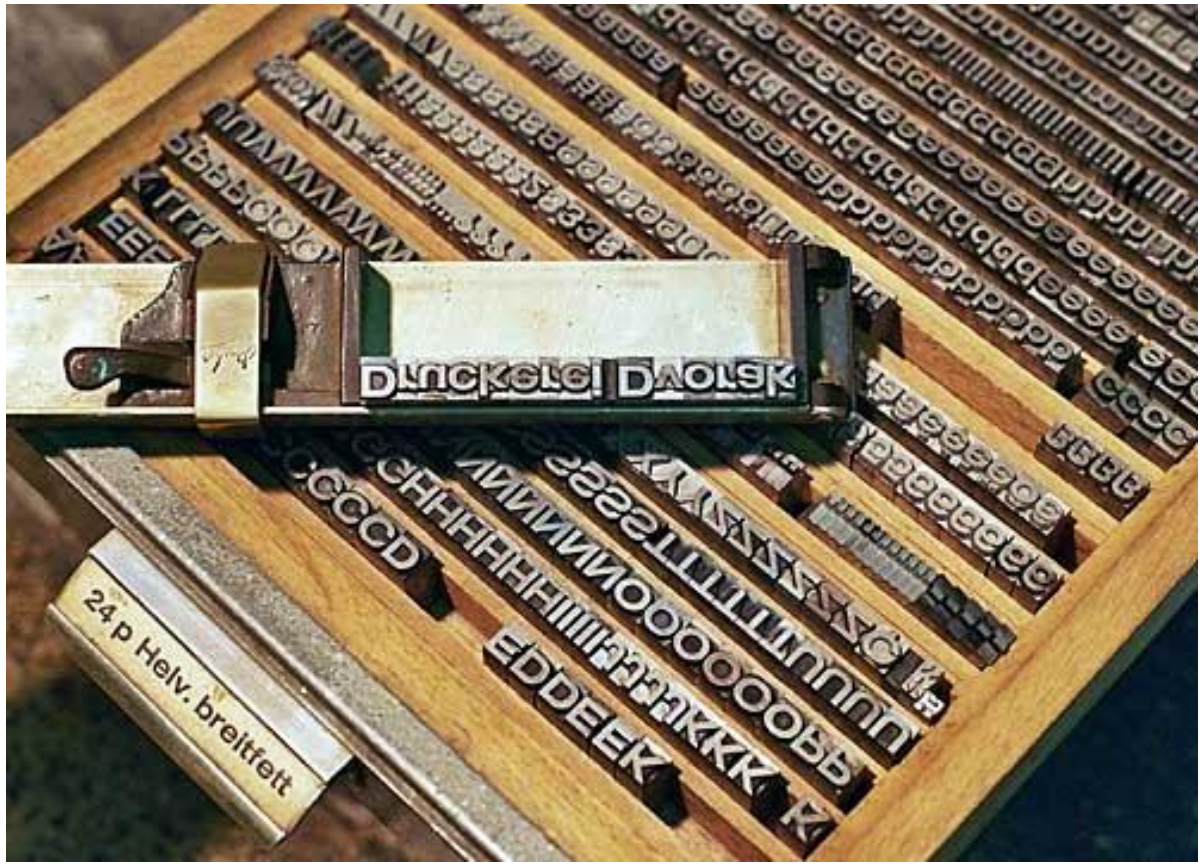
- Webstorm oder
  - https://www.jetbrains.com/webstorm/
- Visual Studio Code
  - https://code.visualstudio.com/download
- NodeJS
  - https://nodejs.org/
  - Version anzeigen
    `node -v`
  - Installierte Pakete
    `npm list -g -depth=0`
  - Installieren/Aktualisieren von Paketen
    `npm install -g <package>`

```
$ node -v
v16.3.0

$ npm list -g -depth=0
npm notice
npm notice New minor version of npm available! 7.15.1 -> 7.20.5
npm notice Changelog: https://github.com/npm/cli/releases/tag/v7.20.5
npm notice Run npm install -g npm@7.20.5 to update!
npm notice
/development/npm
+-- cordova@10.0.0
+-- ionic@5.4.16
+-- native-run@1.3.0
`-- npm@6.14.11
```

Hypertext

- **Keine sequentielle Präsentationsreihenfolge**
- **Dokument** als Menge **verlinkter Ressourcen**
  - Format der Ressourcen
  - Format der Links
- **Erste** Systeme waren rein **textbasiert**
- **Zunehmende Bedeutung des Betrachters**
  - keine eindeutige Reihenfolge vorgegeben
  - aktive **Navigation** statt passiver **Präsentation**
- Wenn auch **Multimedia-Dokumente** (nicht rein textbasiert: Film, Animation, Audio) eingesetzt werden, spricht man auch von **Hypermedia**

Hypertext Markup Language (HTML)




```
<HEADER>
<TITLE>The World Wide Web project</TITLE>
<NEXTID N="55">
</HEADER>
<BODY>
<H1>World Wide Web</H1>The WorldWideWeb (W3) is a wide-area<A
NAME=0 HREF="WhatIs.html">
hypermedia</A> information retrieval
initiative aiming to give universal
access to a large universe of documents.<P>
Everything there is online about
W3 is linked directly or indirectly
```

HTML1.0

Beschreibt den Inhalt von Webseiten. HTML ist das Format, in dem die Text- und Hypertext-Informationen im WWW gespeichert und übertragen werden und ist eine Anwendung von SGML (Standard Generalized Mark up Language)

Versionen

- HTML 1.0
- HTML 4.0 Externe Stilangaben über eigene Sprache möglich (CSS)
- XHTML 1.0/1.1 und **HTML 5.0**

https://www.w3schools.com/html/default.asp

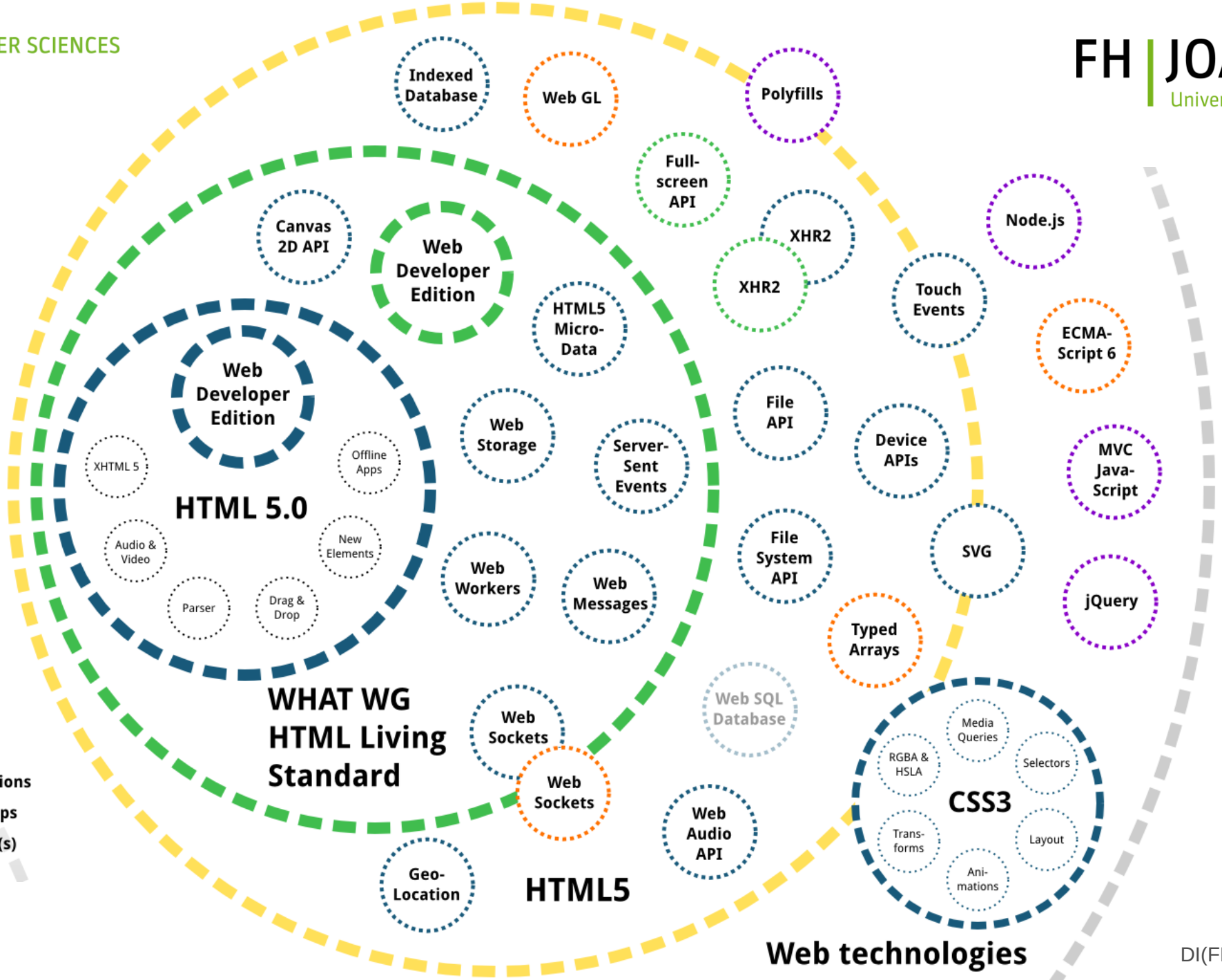
Hypertext Markup Language - HTML

- Hypertext-Dokumentformat, Auszeichnungssprache
- Wird in Dateien gespeichert oder mittels Programmen dynamisch generiert
- Enthält <tags>, welche einerseits Strukturanweisungen, Formatanweisungen oder Verweise auf andere Ressourcen (Links) enthalten.
- Ressourcen können Dokumente jeglicher Art sein:
  - Andere HTML Dokumente
  - Grafiken und Bilder (gif, jpeg, png)
  - Video und Sounddateien
- Über Links können auch andere Programme aufgerufen werden (Telnet, FTP, Mail, …).
- Kann auch Formulare enthalten, welche Benutzereingaben erwarten und diese zur Bearbeitung an den Server schicken.
- **WICHTIG**: Trennung von Inhalt und Darstellung

HTML5

- Entwicklung
  - Web Hypertext Application Technology Working Group (WHATWG)
  - Referenz: https://www.w3.org/TR/html50/
- Neue Features
  - Strukturell (Neue Bereiche)
  - Formulare (Elemente)
  - Audio, Video & Grafik (2d/3D)
- URLs
  - https://en.wikipedia.org/wiki/Comparison_of_browser_engines_(HTML_support)#HTML5
  - http://html5test.com/
  - https://caniuse.com/
  - https://www.w3.org/TR/2011/WD-html5-20110525/content-models.html#content-models

HTML



W3C Specifications
WHATWG Specifications
Other working groups
Community project(s)
Common use

https://github.com/SirPepe/SpecGraph

HTML | **Grundsätzlicher Aufbau**

```
<html>
        <head>
                Weitere Tags
        </head>
        <body>
                Weitere Tags oder Text!
        </body>
</html>
```

- Verschachtelung von Tags bildet eine Baumstruktur: **DOM (Document Object Model)**
- Tags müssen / sollten immer einen **Abschlusstag** haben!
- Wenn ein **Tag keinen Inhalt** hat, kann man den Abschluss gleich im Anfangstag einbauen: `<br />`

https://www.w3schools.com/html/html_basic.asp

HTML | **Anatomie einer Seite**

Typ

```
<!DOCTYPE html>

<html lang="en">
```

Struktur

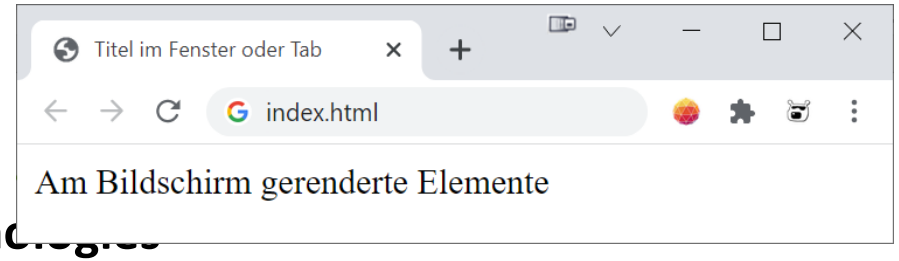
```
  <head>
    <meta charset="UTF-8">
    <title>Titel im Fenster oder Tab</title>
  </head>
  <body>
    <header data-meinAttribut="xyz">
      Am Bildschirm gerenderte Elemente
    </header>
  </body>
</html>
```

Dokumenttypdefinitionen (DTD):

- HTML5 hat keine DTD mehr- weil dynamischer z.B. selbstbestimmte Elemente wie eigene Attribute: `data-mathias="meine Eigenschaft"`
- Alle Vorgänger hatten ein genaues Set an Regeln.
  z.B.:`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`

"Kopf"
mit Metainformationen
zur Seite

"Körper"
mit Inhalten



Kein SGML mehr …. Weil nicht per DTD abbildbar

HTML | **Der Tag / Das Element**

- **Tags** sind **Element der Seite**.

- Tags **beschreiben** ein Element, werden selbst **aber nicht angezeigt**!

- Tags können weitere **beschreibende Attribute** besitzen:
  `<img src="logo.png" width="20px" height="30px" />`

- Zeichen, die zum Beschreiben der Seite verwendet werden, müssen selbst beschrieben werden. Man spricht hier von „**Entitäten**“:
  „**<**“ wird mit „**`<`**“ , „**>**“ wird mit „**`>`**“ „maskiert.

https://www.w3schools.com/html/html_elements.asp

HTML Elemente | **Verweise (Links)**

„**target**“ öffnet den Verweis mit …
… „_blank“ auf einer neuen Seite
… „_self“ in der aktuellen Seite

- **Interne** Verweise

```
<a href="seite.html" target="_self"> Zur Seite </a>
```

- **Externe** Verweise

```
<a href="http://extern.at/seite.html"> Zur externen Seite </a
```

- **Anker** auf einer Seite

```
[http://external.at/beispiel.html]
<p>
   Das <a name="beispiel">Beispiel</a> ist mit einem Anker belegt. Dieser Text ist in der Seite
   beispiel.html unter http://external.at erreichbar!
<p>

[index.html]
<p>
   <a href="http://external.at/beispiel.html#beispiel">Zum Beispiel</a>
</p>
```

https://www.w3schools.com/html/html_links.asp

HTML Elemente | **Formulare**

- **Benutzereingabe** aufnehmen und versenden.
- **Tag**: <form> ...Elemente... </form>
- **Attribute** des Formulars:
  - **action**="<URL oder relative Angabe zu einer Seite>" - verarebeitende Seite, an welche die Eingaben übergeben werden
  - **method**="<get|post>"
    - **get** → Formulardaten als Parameter nach dem Verweis von „action“
    - **post** → speziellles versenden – bei vielen Formulardaten zu verwenden!
  - **accept-charset**="UTF-8" - Zeichenkodierung der Daten
- **Elemente**
  - <input>, <textarea>, <select>, <button>

https://www.w3schools.com/html/html_forms.asp

HTML Elemente | **Formularbeispiel**

Formular

Texteingabefeld

Auswahl

Textfeld

Buttons

```
<form action="http://mysite.at/cgi-bin/test.pl">
 <table border="0" cellpadding="5" cellspacing="0" bgcolor="#E0E0E0">
   <tr>
     <td align="right">Name:</td>
     <td><input name="Name" type="text" size="30" maxlength="30"></td>
   </tr>
   <tr>
     <td align="right">Ort:</td>
     <td>
       <select name="Ort" size="3">
           <option>Wien</option>
           <option>Paris</option>
       </select>
     </td>
   </tr>
   <tr>
     <td align="right" valign="top">Kommentar:</td>
     <td><textarea name="Text" rows="10" cols="50"></textarea></td>
   </tr>
   <tr>
     <td align="right">Formular:</td>
     <td>
       <input type="submit" value=" Absenden ">
       <input type="reset" value=" Abbrechen">
     </td>
   </tr>
 </table>
</form>
```

HTML Elemente | **Zusammenfassung**

- Der „**Tag**“
  - <tag> Irgendetwas </tag>
  - Irgendetwas ist Text oder wieder ein weiterer „Tag“.
  - „Tags“ können Attribute haben. Beispiel: <tag width=“100“></tag>
  - „Tags“ beschreiben ein Element werden aber selbst nicht angezeigt.
  - Beispiel: fett nicht fett

https://www.w3schools.com/html/html_entities.asp

- Die „**Entität**“ - „maskierte“ Zeichen
  - &Kürzel des Zeichens;
  - Beispiele: & ... & > ... > < ... <   … „ „

https://www.w3schools.com/html/html_entities.asp

Cascading Style Sheets (CSS)

- Entwicklung
  - World Wide Web Consortium
  - Referenz: https://www.w3.org/TR/CSS/#css
- Einige Features
  - Media Queries
  - Layouts (Grid-, Box-, Flex- usw.)
  - Animationen, Webfonts, usw.
- URLs
  - https://enjoycss.com
  - http://css3test.com/
  - http://www.csszengarden.com/
  - https://en.wikipedia.org/wiki/Comparison_of_browser_engines_(CSS_support)

Syntactically **A**wesome **S**tyle **S**heets

CSS pre-processor
Scriptsprache, welche
SassScript in CSS
transpiliert.
https://sass-lang.com/

CSS | **Formatierung**

- Formatangaben für (X)HTML Elemente
  - Farben, Größen, Positionen, Bilder, Abstände, Sichtbarkeit, …
- Einbindung in HTML
  - Innerhalb des Seitenkopfes (head)
    `<style type="text/css"> … </style>`
  - In einer oder mehreren externen Dateien
    `<link rel="stylesheet" type="text/css" href="formate.css" />`
  - Innerhalb eines Tags (style-Attribut)
    `<p style=".....">  … </p>`
- Syntax:
  `[Selektor] {[Eigenschaft] : [Wert];}`
  - Selektor:
    - Bezug zu dem gewünschten Element
    - Zum Beispiel der Tagname p
  - Eigenschaft:
    - Formatangabe, welche gesetzt werden soll
    - Zum Beispiel eine Farbe: #DDEE00

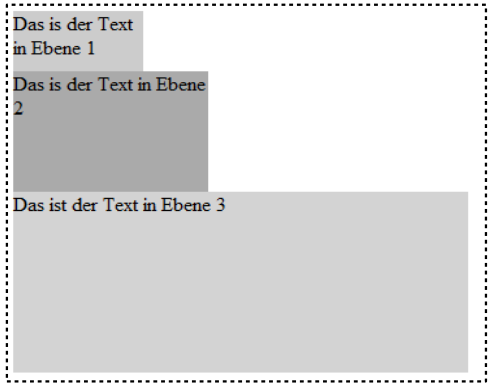
css/layout.css

```
/* In Ebene 1 ist eine Klasse definiert */
div.ebene1 {
    width: 100px; height: 50px;
    background-color: #CCCCCC;
}
/* In Ebene 2 ist eine Id  definiert */
#ebene2 {
    width: 150px; height: 100px;
    background-color: #AAAAAA;
}
```

beispiel.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8" />
        <title>CompanyName - PageName</title>
        <link rel="stylesheet" type="text/css" href="css/layout.css" />
    </head>
    <body>
        <div class="ebene1">
            Das is der Text in Ebene 1
        </div>
        <div id="ebene2">
            Das is der Text in Ebene 2
        </div>
        <div style="width:350px;height:150px;background-color:lightgrey;">
            Das ist der Text in Ebene 3
        </div>
    </body>
</html>
```

Im Browser:



https://www.w3schools.com/html/html_css.asp

CSS | **Struktur**

Prioritäten

- Nach Element Id: #[Element-Id]
  - Element: `<p id="spezial">Test</p>`
  - Styleangabe: `#spezial { color:red; }`
- Nach Klassen: [Tagname].[Klassenname]
  - Element: `<p class="spezial">Test</p>`
  - Styleangabe: `p.spezial { color:red; }`
- Pseudoklasse: [Tagname]:[Pseudoklasse]
  - Für Elementeigenschaften, die nicht eindeutig durch ein HTML Element beschreiben lassen: Typischfür Links: a:link, a:visited, a:hover

Verschachtelung

- Angabe eines Stile für mehrere Elemente: Selektoren mit Beistrich getrennt
- Direktes Kindelement ansprechen: Selektoren mit Leerzeichen getrennt:
  Beispiel: `td p { color:red; }`
- Irgendein Kindelement darunter ansprechen: Selektoren mit * getrennt
  Beispiel: `table * p { color:red; }`

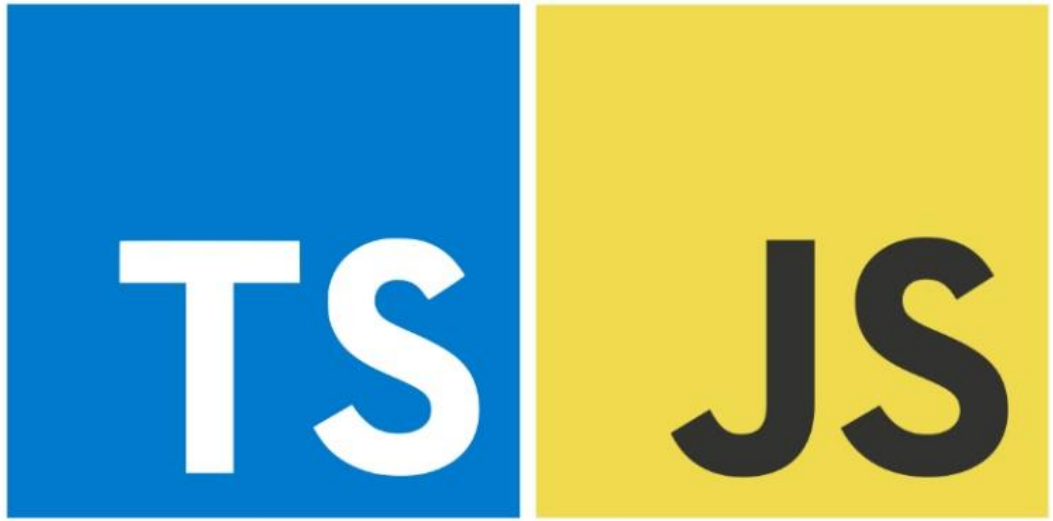
https://www.w3schools.com/css/default.asp

Wichtige Webseiten

- http://www.**w3schools.com/**
  - Anleitungen zum Erstellen von Webseiten
- http://**de.selfhtml.org/**
  - DAS Kompendium über
  - Auszeichnungssprachen im Web
- http://www.**csszengarden.com/**
  - Was alles mit Stilangaben machbar ist
- http://www.**w3.org/**
  - Gremium zur Standardisierung der das
  - World Wide Web (WWW) betreffenden Techniken
  - http://validator.w3.org/
  - http://jigsaw.w3.org/css-validator/

DIHSÜD

FH JOANNEUM
University of Applied Sciences



JavaScript & TypeScript

Workshop
WebEngineering & Server Technologies

Anwendungsgebiete

- **Webseiten benutzerfreundlich** und **funktionell** gestalten (wie Desktop-Applikationen)
- **Plausibilitätsprüfung** (Validierung) von Formular-eingaben vor dem Absenden
- **Browsererkennung** (Browserweiche)
- **Multimedia** (Slide Shows, Games, etc.)
- Dynamische Manipulation über **DOM**
- Senden und Empfangen von Daten, ohne dass der Browser die Seite neu laden muss mit **AJAX / Fetch**

Tutorials: https://www.w3schools.com/js/default.asp

Was ist JavaScript?

- **Soll wie Java aussehen**
- **Skriptsprache vornehmlich für Clients** (Browser)
- ECMA Standard (seit 1997)
  heute ECMAScript 2018
  https://kangax.github.io/compat-table/es2016plus
- Sandboxumgebung *
  - Zugriff auf Browser (-Objekte) beschränkt
  - Kein Zugriff auf Dateisystem
  - Kein Lesen oder Schreiben von Dateien

  (* Umgehung über Sicherheitsbeschränkungen im Browser u.U. möglich)

- Syntax ist Java und C ähnlich
- Programmiersprache ist
  Prozedural – Objektorientiert - Funktional
- Dynamische Typisierung
- Einsatzgebiet: Client- u. Serverseitig

Serverseitiges Javascript: Node.js

ECMA
European Computer Manufacturers Association

https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

Was ist **TypeScript**

- Von Anders Hejlsberg,Microsoft, 2021 entwickelt
- Sprache für **typisiertes JavaScript**
  auch "überlagertes JavaScript" genannt
- Wird **in Standard-JavaScript "transpiliert"** und braucht dazu ein eigenes Programm, einen "**Transpiler**"
- Üblicherweise haben **TS Dateien *.ts als Erweiterung** und es werden *.js Dateien daraus erzeugt
- **JavaScript Bibliotheken und –Frameworks** können durch Deklarationsdateien **ermöglicht** werden
- **Neuere JavaScript Funktionen** werden unterstützt (z.B.: async)

"TypeScript is **JavaScript with syntax for types.**"
-- https://www.typescriptlang.org/

JavaScript **ideal** für kleine Projekte
- Schwach typisiert
- Mehrere Paradigmen
- siehe Websprachen!

TypeScript **ideal** für große Projekte
- Superset von JavaScript - transpiliert zu JavaScript
- Typen ("TypeSafe") & Interfaces
- Prototype Pattern

Javascript | **Einbettung in HTML: Scriptbereich**

```
<html>
        <head>
                <title>Eingebaut</title>
                        <script type="text/javascript">
                                <!--
                                        alert(„JavaScript at work!");
                                -->
                        </script>
        </head>
        <body>
                Javascript Beispiel
        </body>
</html>
```

Ältere Browser, die JS nicht kennen ignorieren Code unter Kommentaren und interpretieren ihn nicht irrtümlich als Text innerhalb der HTML-Datei! – kaum mehr der Fall...

https://www.w3schools.com/js/js_whereto.asp

Javascript | **Einbettung in HTML: Datei**

```
<html>
        <head>
                <title>Separate Datei</title>
                        <script src="alert.js" type="text/javascript"></script>
        </head>
        <body></body>
</html>
```

Inhalt von alert.js:

```
// Kommentar
alert(„Externe JavaScript-Datei at Work!");
```

https://www.w3schools.com/js/js_whereto.asp

Javascript | **Einbettung in HTML: Events**

```
<html>
        <head>
                <title>Als Event</title>
        </head>
        <body>
                <img
                        src="button1.gif"
Events                  onmouseover="alert('Hallo');" />
                <img
                        src="button2.gif"
                        onclick="javascript: alert('Alert mit Sprachdefinition');" />
        </body>
</html>
```

https://www.w3schools.com/js/js_events.asp

Javascript | **Event Handler**

Maus-Ereignisse

- **onmouseover**
- **onmouseout**
- **onmousemove** Mauszeiger bewegen
- **onmousedown**
- **onmouseup**
- **onclick**
- **ondblclick**

Tastatur-Ereignisse

- **onkeydown**
- **onkeyup**
- **onkeypress**

Formularereignisse

- **onsubmit**
- **onreset**
- **onselect**
  Inhalt eines Formularelements mit der Maus markieren
- **onchange**
  Ändern des Inhaltes eines select-Feldes

Weitere Ereignisse

- **onload**
  HTML-Seite wurde komplett(!) geladen
- **onunload**
  HTML-Seite wurde verlassen
- **onabort**
  Das Laden der HTML-Seite wurde abgebrochen

Allgemeine Ereignisse

- **onfocus**
  Wenn ein Element aktiviert wird
- **onscroll**
  Wenn ein Element mit der Maus gescrollt werden kann
- **onblur**
  Wenn ein zuvor aktiviertes Element verlassen wird

Es gibt sicher noch ein paar mehr …

Javascript | **Überblick Variablen**

- Zuweisung

```
a = 2;          // Global ("window" Objekt Attribut)
var b = "x"; // Lokal innerhalb einer Funktion
let c = "y"; // Lokal innerhalb von Blöcken
                // Global, wenn außerhalb Funktion
```

- Schwach, dynamisch typisiert

```
typeof(a);      // number
typeof(b);      // string
```

und *boolean, function, object, undefined, null, symbol*

- **Number**
  In der Regel Double Precision Float
  bei Bitoperationen 32Bit Integer
- **String**
  Unicode/UTF16 – einmal erstellt nicht veränderbar

Achtung!
Globale variablen können auch in Funktionen deklariert werden!

https://www.w3schools.com/js/js_datatypes.asp

TypeScript | **Überblick Typen**

- **`boolean`**
- **`number`**
- **`string`**
- **`number[]`** oder **`Array <number>`**
- Tupel: Arrays mit unterschiedlichen Typen: **`[string, number]`**
- Mehrere Konstanten (enums):
  `enum Color {red=1, green, blue}`
  `let color:Color = Color.green; // 2`
- **`any`** (unbestimmter Typ); **`any[]`**
- **`void`** (leer) ist **`null`** oder **`undefined`**
  Letztere sind auch definiert
- **`never`** (bei Funktionen: nie ein return)
- **`object`** (kann auch null sein)

```
function helloworld(degree_programme:string){
    return "Hello" + degree_programme;
}

let dp = {};

document.body.innerHTML = helloworld(dp);
```

Error:(7, 38) TS2345: Argument of type '{}' is not assignable to parameter of type 'string'.

Überprüfung (Assertion):

```
let zeichen:number = (<string> wert).length;
let zeichen:number = (wert as string).length;
```

(Keine Auswirkungen beim transpilieren.)

TypeScript | **Eigenen Datentypen deklarieren**

Eigene Typen vs. Standardtypen

- Einfach oder kombiniert mit '|' (oder)

```
type meinSpezifischerTyp = number;
type meinUnbestimmterTyp = any;
```

```
type meinSpezifischerTyp = string | number;
```

- Generisch

```
type meinGenerischerTyp<Type> = Type;
```

- Objekte (Schlüsselwort 'type' oder 'interface' kann verwendet werden!)

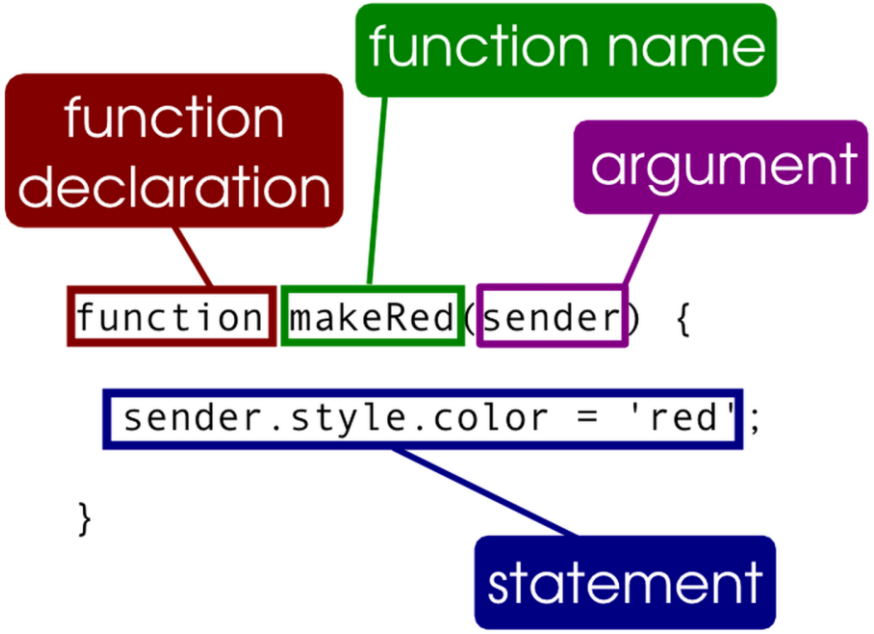
```
type Pizza = {
    name: string;
    typ: string;
    durchmesser:number
};
```

```
interface Pizza2 extends Pizza = {
    notiz?: string;
    readonly rating:number
};
```

Man kann Typen auch erweitern ('extends')
Optionale Eigenschaften können mit '?' markiert werden!
Eigenschaften, die nicht mehr geändert werden können mit 'readonly' markiert werden (in der IDE beim Kompilieren relevant)

Javascript | **Funktionen in Javascript**

- Funktionen werden **wie Objekte** behandelt (*First-Class-Functions*) und merken sich ihren **Erstellungskontext** (*Closure*)
- Funktionen sind **Eigenschaften von Objekten**!
- **Funktionen können Funktionen enthalten**
- Funktionen können **als Parameter Funktionen** erhalten



https://www.w3schools.com/js/js_functions.asp

Javascript | **Funktionen *(einfach)***

Funktionen selbst definieren in HTML

```
<script src="alert.js" type="text/javascript">
            function simple(zahl) { … }
</script>

<input type="button" value="Simple"  onclick= "simple(22307)"/>
```

Funktionen mit Rückgabewert

```
function simple(zahl) {
            var ergebnis = zahl; // Lokale Variable
            return ergebnis;
}

var resultat = simple(22307); // Globale Var.
```

Javascript | **Funktionen *(komplex)***

```
function makeSandwich(){
        var ingredient = 'bread';
        function make(filling){
                return ingredient + ' and ' + filling;
        }
        return make('ham');
}

makeSandwich(); // Gibt „bread and ham“ zurück
```

Eine Funktion in der Funktion …

… und diese wird zurückgegeben ;-)

Adaption des unteren Teils des Snippets möglich:

```
        return make;
}

var f = makeSandwich();
f('ham'); // Gibt „bread and ham“ zurück
```

Die Funktion wird in eine variable gespeichert …

Javascript | **Objekte (`object`)**

- Container für **Schlüssel-Wert-Paare**
- Der **Schlüssel** bezeichnet die **Eigenschaft** oder eine **Methode** des Objekts
- Der **Wert** kann ein **Literal**, eine **Funktion** oder ein anderes **Objekt** sein
- Objekte können auf verschiedene Weisen erstellt werden
  - **Konstruktorfunktion**
  - **Objekt-Literal-Schreibweise**
  - Methode "**create**" des Objekts "**Object**"
- Der Zugriff auf Eigenschaften des Objekts erfolgt mittels **"." Notation** (wie in Java) oder die **"[]" Notation**

https://www.w3schools.com/js/js_objects.asp

Javascript | **Objekte erstellen & darauf zugreifen**

- Fest implementiert
  - **`String`**, **`Number`**, **`Array`** oder **`Math`**
- Host Objekte (Seitenstruktur)
  - **`window`**, **`document`** oder **`form`**
- Benutzerdefinierte Objekte

```
myobject = new Object();
```

oder

```
myobject = {};
```

Letztere Schreibweise wird auch bei JSON verwendet - JavaScript Object Notation

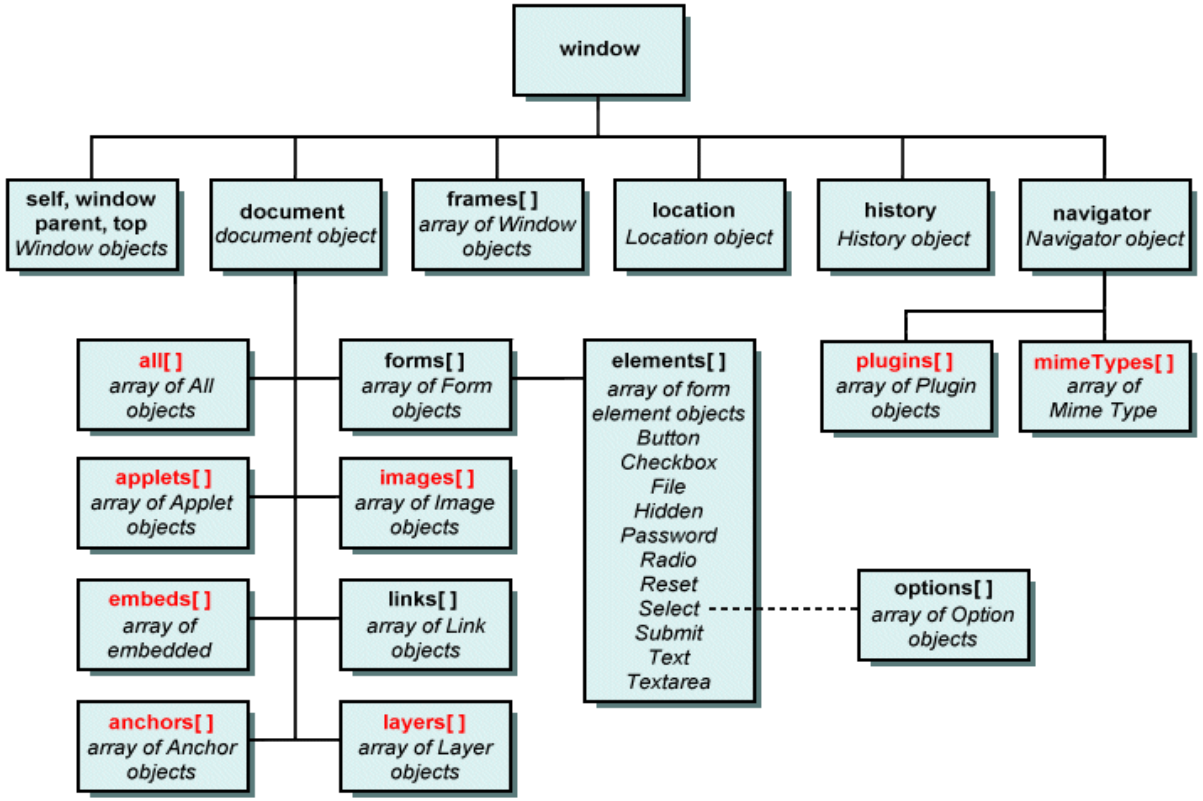
```
pizza = {
        name : "Diabolo",
        preis : 7.5,
        zutaten : ["Käse","Tomaten","Salami","Pfefferoni"]
};
```

```
pizza = new Object();
pizza.name = "Diabolo";
pizza.preis = 7.5;
pizza.zutaten = ["Käse","Tomaten","Salami","Pfefferoni"];
```

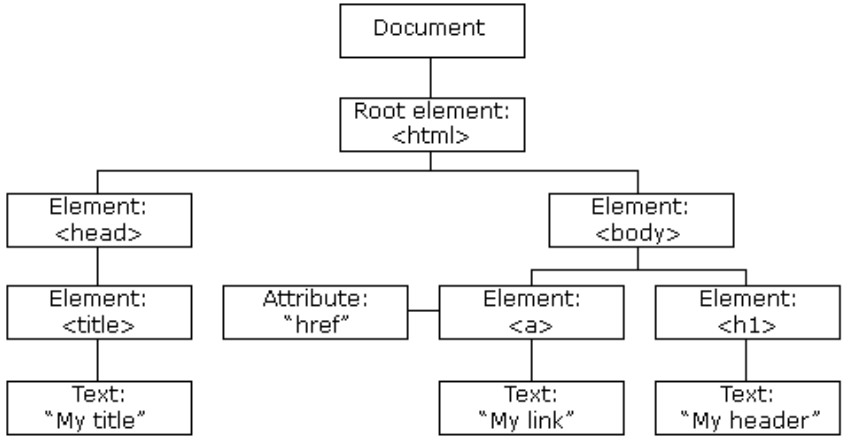
```
pizza = new Object();
pizza["name"] = "Diabolo";
pizza["preis"] = 7.5;
pizza["zutaten"] = ["Käse","Tomaten","Salami","Pfefferoni"];

alert(pizza["name"]); // alert(pizza.name);     // "Diabolo"
alert(pizza["preis"]); // alert(pizza.preis);   // 7.5
alert(pizza["zutaten"][0]); // alert(pizza.zutaten[0]);  // "Käse"
alert(pizza["zutaten"][1]); // alert(pizza.zutaten[1]);  // "Tomaten"
```

https://www.w3schools.com/js/js_object_properties.asp

Javascript | **Überblick der Host Objekte**



Achtung!
u.u. browserabhängig, was hier verfügbar ist!



https://www.w3schools.com/js/js_htmldom.asp

Javascript | **Arrays (`array`)**

- Sind **Objekte** und werden über "**new Array()**" oder die **Literal-Kurzschreibweise** instanziiert
- Wichtige Funktionen:
  - **`concat()`** … Elemente an bestehende Arrays anhängen
  - **`forEach()`** … Wendet eine übergebene Funktion an jedes Element im Array an
  - **`pop()`** … entfernt das letzte Element
  - **`push()`** … fügt ein neues Element hinzu
  - **`reverse()`** … kehrt die Reihenfolge um
  - **`sort()`** … sortiert auf Basis einer übergebenen Vergleichsfunktion

https://www.w3schools.com/js/js_arrays.asp

Javascript | **Vorteile assoziativer Arrays**

Einfaches Durchlaufen der Eigenschaften
und Funktionen!

```
pizza = {
          name : "Diabolo",
          preis : 7.5,
          getPreis : function(){ return this.preis; }
};
var output = "";
for ( x in pizza ){
          output += x + " || " + pizza[x] + "\n";
}
alert(output);
/*
name || Diabolo
preis || 7.5
getPreis || function(){ return this.preis; }
*/
```

Arrays in Javascript:

Regular:

```
var myCars=new Array();
myCars[0]="Saab";
myCars[1]="Volvo";
myCars[2]="BMW";
```

Condensed:

```
var myCars=new
Array("Saab","Volvo","BMW");
```

Literal:

```
var myCars=["Saab","Volvo","BMW"];
```

Siehe http://www.w3schools.com/js/js_obj_array.asp

JavaScript | **Unobtrusive**

- Grundsätze
  - Abgrenzung zwischen **Verhalten**(Behavior), **Präsentation** (Presentation) und **Inhalt** (Content)
    Beispiel: Event Handling mit und ohne Event-Attribut in HTML
  - Site soll auch ohne JavaScript funktionieren (😐)
  - Mit JavaScript soll nur die Benutzerfreundlichkeit erhöht werden
  - "Never trust JavaScript" – also keine kritischen Operationen damit durchführen …

- URL
  - Siehe: https://www.w3.org/wiki/The_principles_of_unobtrusive_JavaScript

Javascript | **APIs & Bibliotheken**

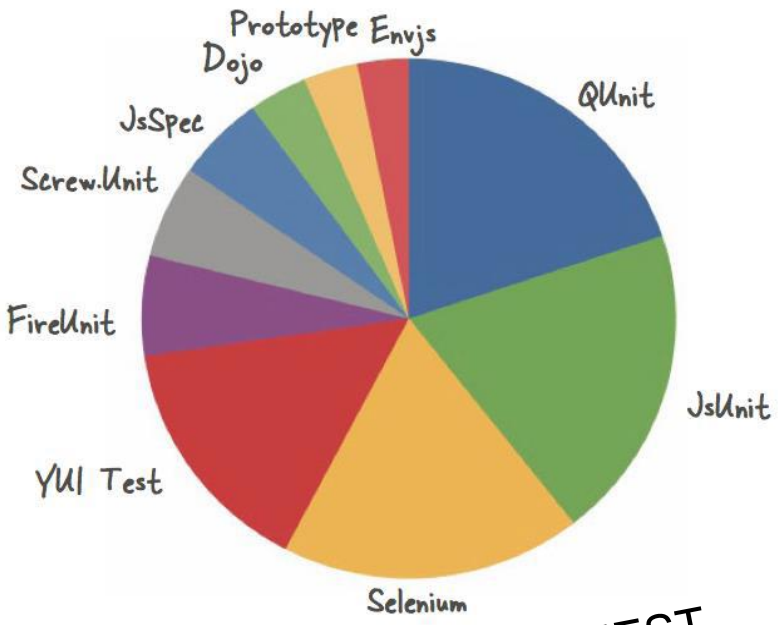
- APIs
  - ***List***: https://developer.mozilla.org/en-US/docs/Web/API
  - AJaX & Fetch
  - Webworker & Service Worker
  - Canvas (Animation)
  - WebSocket
  - WebGL
- Bibliotheken
  - ***List:*** https://en.wikipedia.org/wiki/List_of_JavaScript_libraries
  - jQuery
  - Chart.js
  - D3.js

Welche API kennen Sie bereits?

Welche Bibliothek verwenden Sie am meisten?

JavaScript | **Wekzeuge**

- Logging
- Debugging
- Testing
  - https://jsfiddle.net/
  - https://qunitjs.com/
  - https://www.selenium.dev/
  - https://clarle.github.io/yui3/y



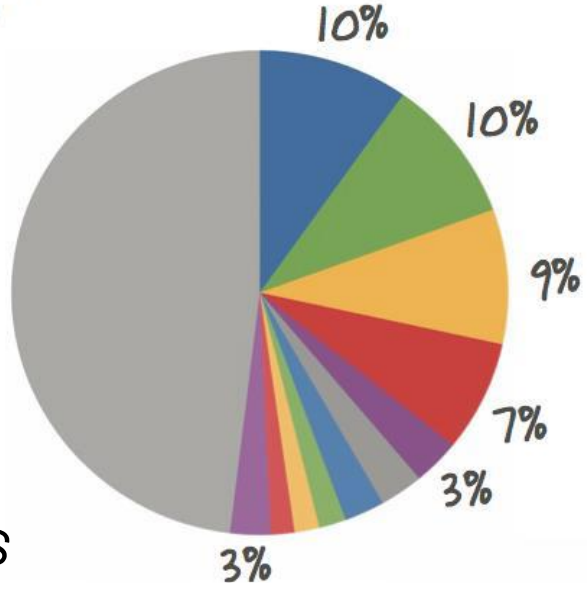
Cypress

JEST

Almost half of JavaScript devs don't test!

48%

MochaJS



Pupeteer

Secrets of the JavaScript Ninja

Modernes Web Design

- Responsives Web Design (RWD)
  - Reaktionsfähiges Layout
  - Der Aufbau eines Webauftritts wird für alle erdenklichen Fälle optimiert
  - Ziel ist für jeden Bildschirm und jedes Gerät einen fließenden Übergang zu ermöglichen
  - Rein clientseitig

- Adaptives Web Design (AWD)
  - Bietet verschiedene Lösungen für bestimmte Bildschirme und Geräte
  - Maßgeschneidertes Layout (nicht eines für alle Eventualitäten sondern eines für genau diese Eventualität)
  - Serverseitige Funktionalitäten notwendig

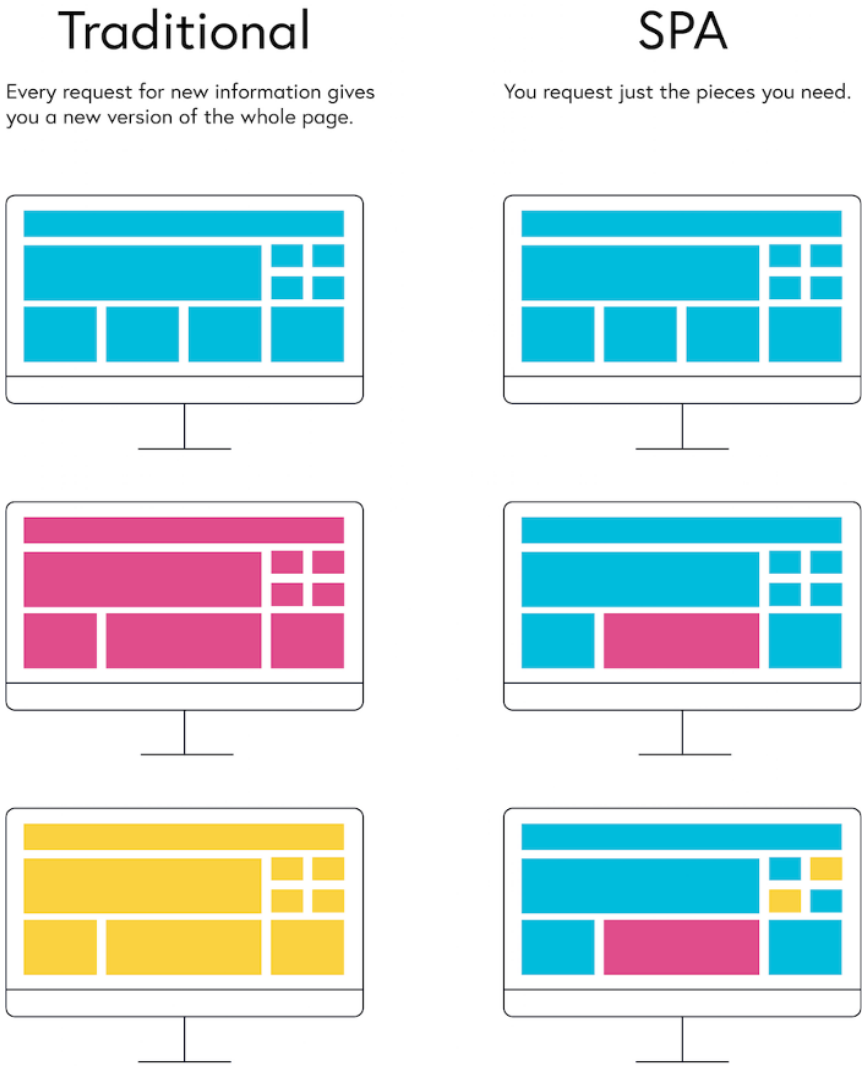
Single Page Application (SPA)

Definition

"An SPA (Single-page application) is a web app implementation that loads only a single web document, and then updates the body content of that single document via JavaScript APIs such as XMLHttpRequest and Fetch when different content is to be shown."

URLs

- https://www.awwwards.com/websites/single-page/
- https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html (Bild rechts)



Single Page Application (SPA) | Vor- und Nachteile

- Vorteile
  - Nur einmal wird eine Datei geladen
  - Benutzerfreundlichkeit (Antwortzeiten, Dynamik, etc.)
  - Entwicklung/Debugging ist einfacher
  - Backend kann auch für andere Apps verwendet werden
  - Effektives Cachen in beliebigem lokalen Speicher

- **Nachteile**
  - Initiales Laden dauert
  - Suchmaschinenoptimierung (SEO)
    - ***Wenig eindeutige Links***
    - ***Semantik schwer analysierbar***
  - Analysen
  - Geteilte Links zu Bereichen der App shwierig
  - Weniger Sicherheit, weil viel Funktion am Client ist

Siehe auch
https://www.bloomreach.com/en/blog/2018/07/what-is-a-single-page-application.html#single-page-applications-advantages

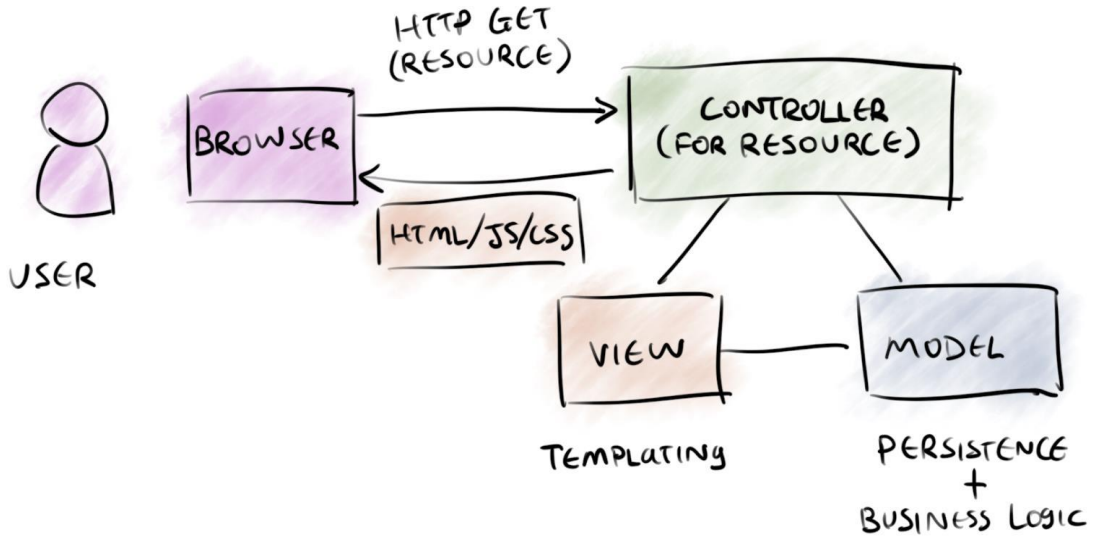
Single Page Application (SPA) | Frameworks

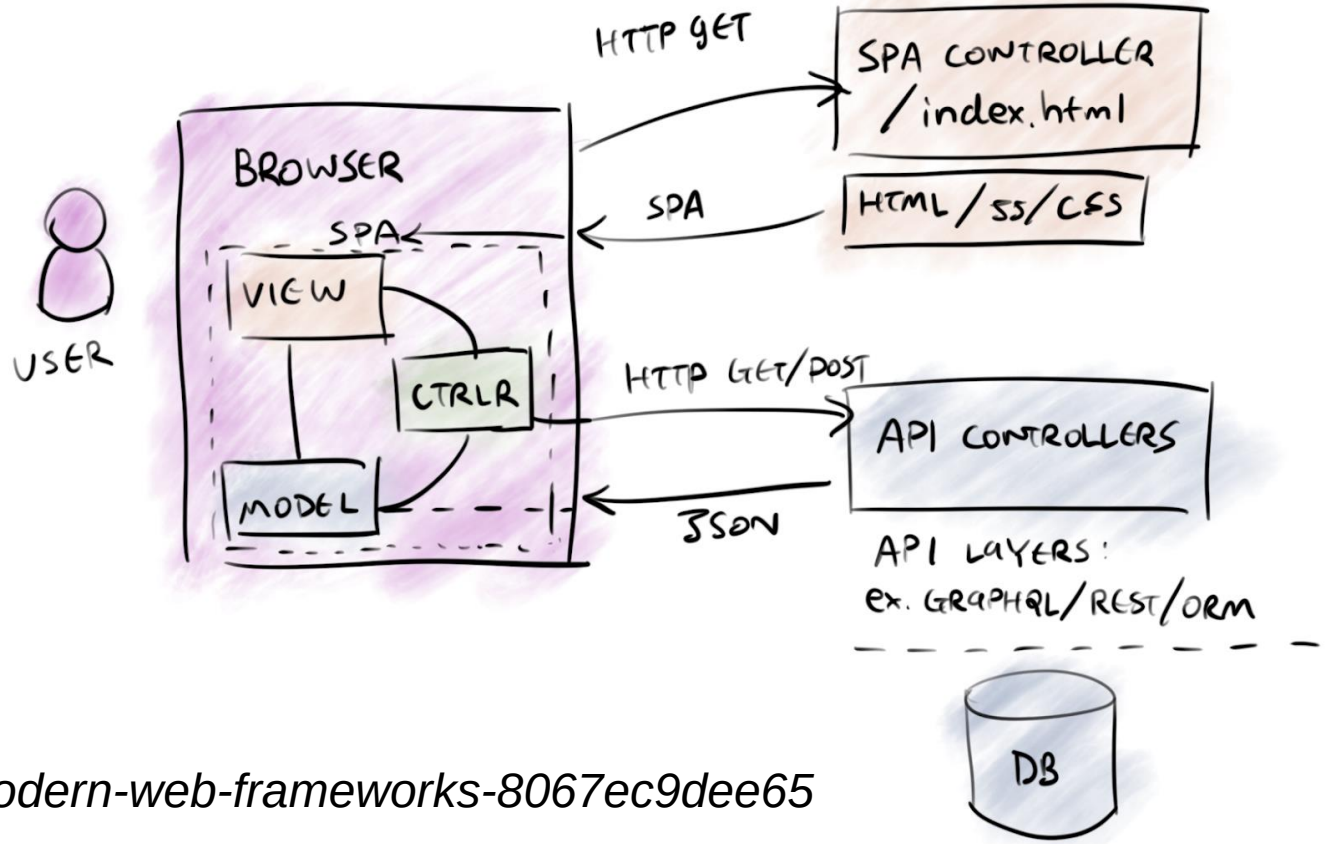
- Frameworks
  - VanillaJS
  - Angular
  - React
  - Vue
  - Aurelia
- Komponentenbasierte Web-Entwicklung
  - Fokus auf wiederverwendbare Komponenten
  - Komponenten haben Eltern und Kinder
  - Daten werden von Eltern an Kinder weitergereicht

https://www.droptica.com/blog/component-based-design/

Single Page Application (SPA) | Moderne Web Frameworks

Model View Controller (MVC)





https://hackernoon.com/from-mvc-to-modern-web-frameworks-8067ec9dee65

DIHSÜD

FH JOANNEUM
University of Applied Sciences

Asynchronous JavaScript and XML (AJaX)

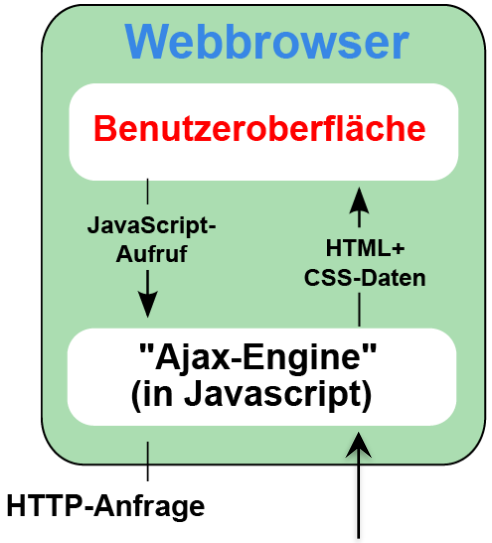
Workshop
WebEngineering & Server Technologies

Asynchronous JavaScript and XML

- **Asynchrone Datenübertragung** zwischen Server und Browser
- Seite **muss nicht neu geladen** werden
- Schlüsseltechnik für Web2.0
- Kombination von Funktionalitäten mehrerer Technologien
  - **(X)HTML** (und CSS) zur Darstellung auf der Webseite
  - **DOM** zur dynamischen Repräsentation und Interaktion der Daten / Inhalte im Dokument
  - **XMLHttpRequest** zur asynchronen Kommunikation mit dem Webserver
  - **JavaScript** als Schnittstelle zwischen diesen Technologien
- **XHR - XmlHttpRequest**
  - Das XMLHttpRequest-Objekt wird zum Austausch von Daten zwischen Nutzer und Server verwendet.
  - Man kann damit ...
    - eine Webseite aktualisieren, ohne sie neu laden zu müssen
    - Daten vom Server anfordern, nachdem die Seite geladen ist
    - Daten zu einem Server im Hintergrund senden

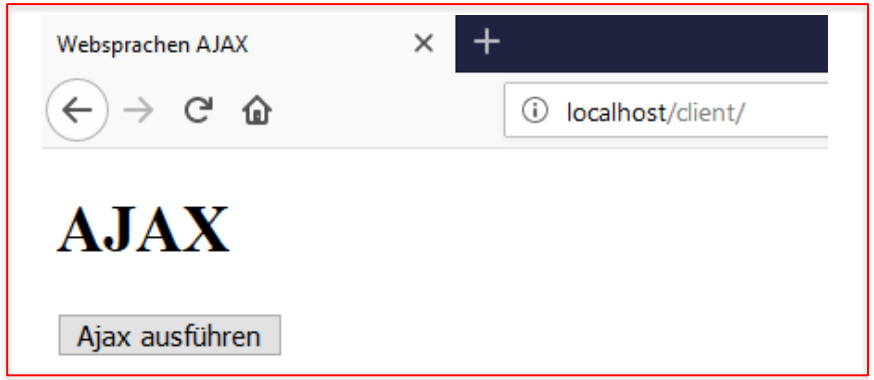
https://www.w3schools.com/js/js_ajax_intro.asp

AJaX - Ablauf | **Benutzeroberfläche**

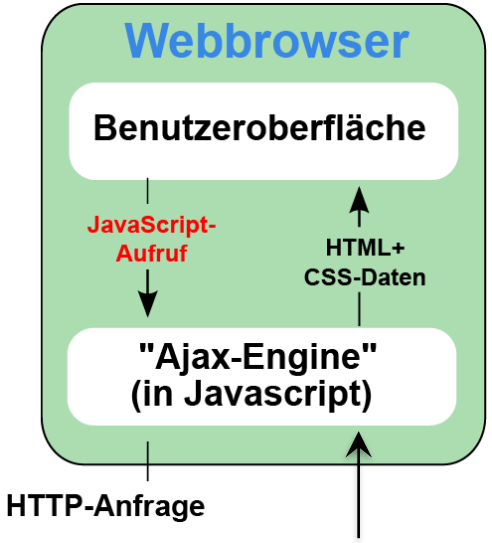


index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Websprachen AJAX</title>
    <script src="js/ajax.js"></script>
  </head>
  <body>
    <h1>AJAX</h1>
    <button onclick="doAjax()">
          Ajax ausführen
    </button>
    <div id="result"></div>
  </body>
</html>
```

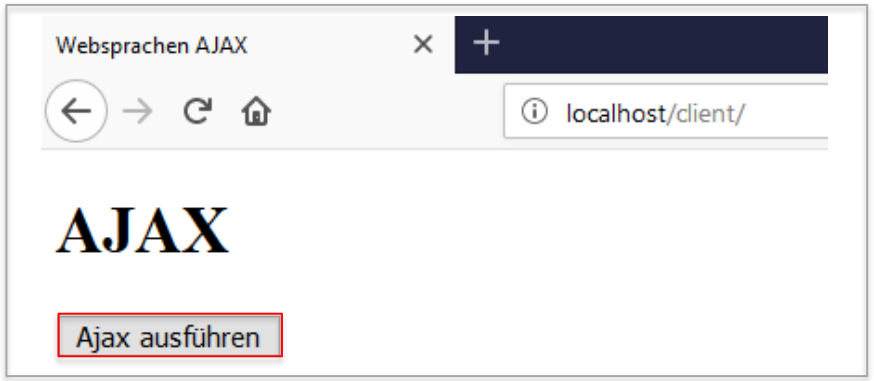


AJaX - Ablauf | **JavaScript aufrufen**

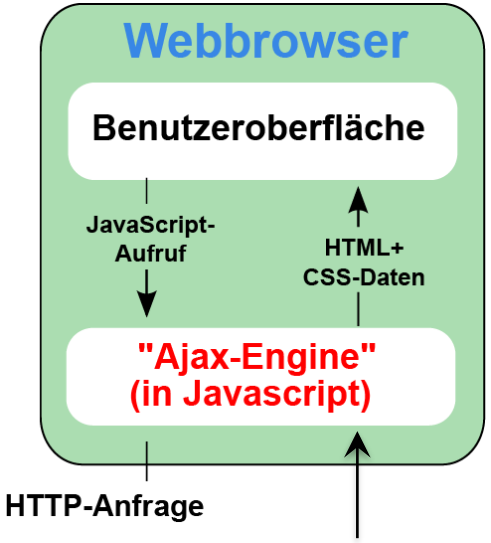


index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Websprachen AJAX</title>
    <script src="js/ajax.js"></script>
  </head>
  <body>
    <h1>AJAX</h1>
    <button onclick="doAjax()">
          Ajax ausführen
    </button>
    <div id="result"></div>
  </body>
</html>
```



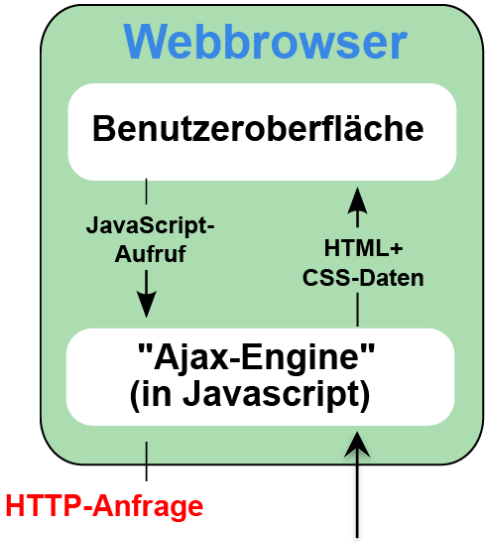
AJaX - Ablauf | **AJAX Engine**



ajax.js

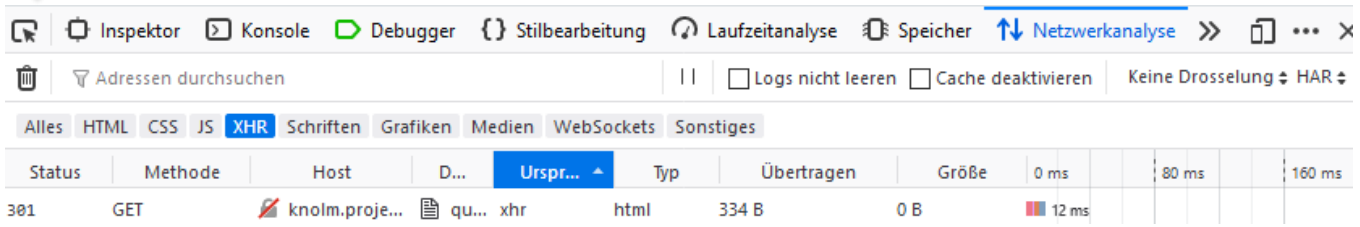
```
function doAjax() {
    let ajaxObj = new XMLHttpRequest();
    ajaxObj.open(
      "GET",
      "quotes.php");
    ajaxObj.onreadystatechange = function () {
        if (ajaxObj.readyState==4 && ajaxObj.status==200) {
            let elemResult = document.getElementById('result');
            elemResult.innerHTML = '';
            let quotesObj = JSON.parse(ajaxObj.responseText);
            quotesObj.quotes.forEach(function (item) {
                 elemResult.innerHTML += item.text;
            });
        }
    };
    ajaxObj.send(null);
}
```

AJaX - Ablauf | **HTTP Anfrage**

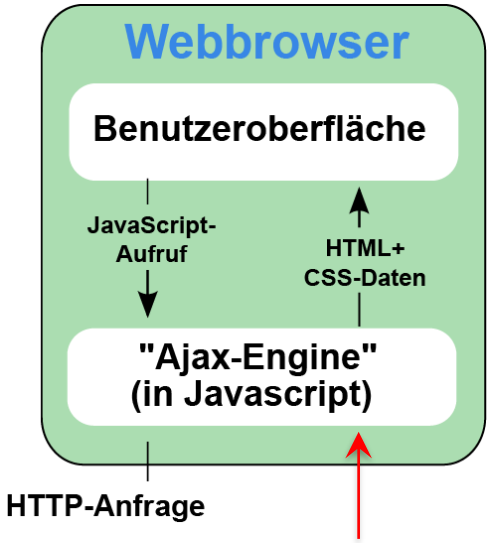


ajax.js

```
function doAjax() {
    let ajaxObj = new XMLHttpRequest();
    ajaxObj.open(
        "GET",
        "http://knolm.projekt-itm.fh-joanneum.at/api/quotes");
    ajaxObj.onreadystatechange = function () {
        if (ajaxObj.readyState==4 && ajaxObj.status==200) {
            let elemResult = document.getElementById('result');
            elemResult.innerHTML = '';
            let quotesObj = JSON.parse(ajaxObj.responseText);
            quotesObj.quotes.forEach(function (item) {
                elemResult.innerHTML += item+'<br/>';
            });
        }
    };

    ajaxObj.send(null);
}
```

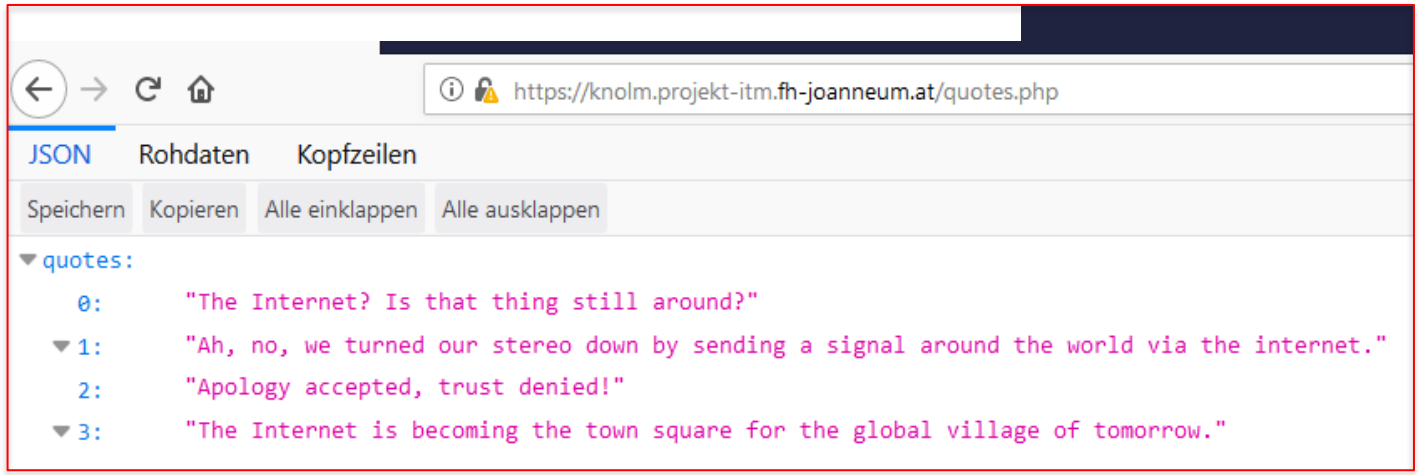


AJaX - Ablauf | **HTTP Antwort**

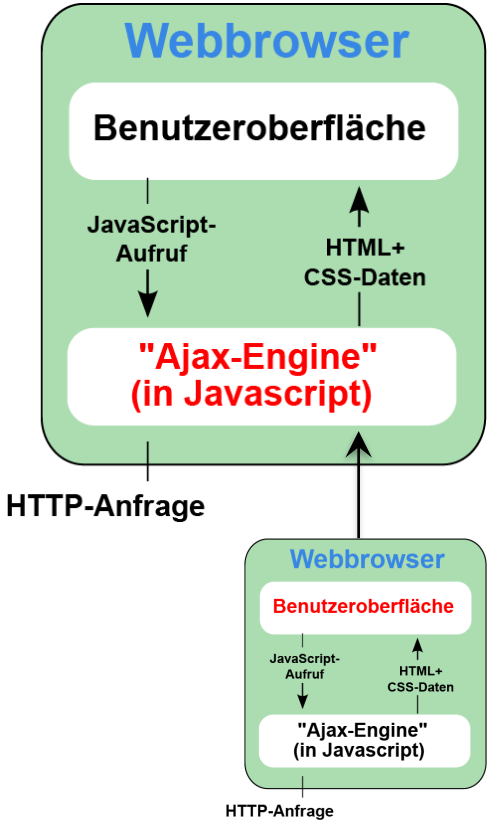


HTTP Response im JSON Format

```
{
  "quotes": [
    "The Internet? Is that thing still around?",
    "Ah, no, we turned our stereo down by sending a signal
     around the world via the internet.",
    "Apology accepted, trust denied!",
    "The Internet is becoming the town square for the
     global village of tomorrow."
  ]
}
```



AJaX - Ablauf | **AJAX Engine & Anzeige im DOM**



ajax.js

```
function doAjax() {
    let ajaxObj = new XMLHttpRequest();
    ajaxObj.open(
        "GET",
        "http://knolm.projekt-itm.fh-joanneum.at/quotes.php");
    ajaxObj.onreadystatechange = function () {
        let elemResult = document.getElementById('result');
        elemResult.innerHTML = '';
        let quotesObj = JSON.parse(ajaxObj.responseText);
        quotesObj.quotes.forEach(function (item) {
          elemResult.innerHTML += +item+'<br/>';
        });
    };
    ajaxObj.send(null);
}
```

AJAX

Ajax ausführen

The Internet? Is that thing still around?
Ah, no, we turned our stereo down by sending a signal around the world via t
Apology accepted, trust denied!
The Internet is becoming the town square for the global village of tomorrow.

XHR Methode | **open()**

Diese Methode legt die Anfrage-Methode und -URL fest. Außerdem legt sie Anfrage-Benutzername und –Kennwort fest, und ob die Anfrage asynchron abgearbeitet wird.

Argumente:

- **method** die zu verwendende **HTTP Methode**, wie "GET", "POST", "PUT", "DELETE", etc.
- **url**: URL, an den die Anfrage geschickt werden soll.
- **async**: (*optional*) boole'scher Parameter
  - ***true***: (Standardwert) gibt an, das Operation **asynchron** ausgeführt werden soll. Der Programmablauf wird nicht blockiert und die Benachrichtigung über die vollendete Transaktion erfolgt mittels Events.
  - ***false*** **blockiert** an der send()-Methode, bis die Antwort vollständig empfangen worden ist.
- **user**: (*optional*) Benutzername zum Zweck der **Authentisierung**; ohne Angabe ist dies ein leerer String.
- **password** (*optional*) zum Zweck der **Authentisierung**; ohne Angabe ist dies ein leerer String.

Beispiel:

HTTP GET und Aufruf einer Datei im gleichen Verzeichnis:

```
ajaxObj.open( "GET", "quotes.php");
```

```
void function open(
    string method,
    string url
    [, boolean async = true
    [, string user = null
    [, string password = null]]])
```

XHR Methode | **setRequestHeader()**

Diese Methode setzt bestimmte Eigenschaften für die Ressourcen-Anfrage. Wenn der gleiche Header mehrmals gesetzt wird, werden die Werte kombiniert

Argumente:

- **header**: HTTP-Header-Feldname
- **value**: Wert für das Header-Feld

```
void function setRequestHeader(
    string header,
    string value
)
```

Beispiele:

- Setzt den Inhaltstyp auf **JSON**:
  `ajaxObj.setRequestHeader('Content-Type', 'application/json');`
- Setzt den Inhaltstyp auf **URL-kodierte Formulardaten**:
  `ajaxObj.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded')`

XHR Eigenschaft | **readyState**

Diese Eigenschaft gibt über den Zustand des XHR Objekts Auskunft. Wenn sich diese Eigenschaft im Verlauf einer AJAX Abfrage ändert, wird ein "readystatechange"-Event ausgelöst.

Zustände:

- **UNSENT** = **0**
  Objekt erstellt, open() wurde noch nicht aufgerufen
- **OPENED** = **1**
  open() wurde aufgerufen
- **HEADERS_RECEIVED** = **2**
  send() wurde aufgerufen. Header und Status sind verfügbar
- **LOADING** = **3**
  Die Antwort vom Server wird gerade heruntergeladen
- **DONE** = **4**
  Die Operation ist fertig

XHR Eigenschaft | **onreadystatechange**

- Hinter dieser Eigenschaft verbirgt sich ein EventHandler. Mit jeder Statusänderung (readyState) wird diese Callback-Funktion aufgerufen
- **Beispiel**:

```
ajaxObj.onreadystatechange = function () {
    console.log(
        "readyState=" + getState(ajaxObj.readyState) +
        "; status=" + ajaxObj.status +
'('+ajaxObj.statusText+')');
};
```

XHR Methode | **send()**

```
void function send(
    [data=null]
)
```

Diese Methode schickt die Anfrage los.

Argument:

data: (optional) Daten, die im Körper der Anfrage gesendet werden.

Achtung:

Damit der Webserver die Daten korrekt interpretiert, muss man den "**Content-Type**" mittels "setRequestHeaders()" angeben!

XHR Alternative | **window.fetch**

Fetch abstrahiert die Hauptkomponenten von **HTTP**

```
function doFetch(){
    fetch('quotes.php')
        .then(function(response) {
            if (response.ok)
                return response.json();
            else
                throw new Error('Zitate konnten nicht geladen werden');
        })
        .then(function(json) {
            let elemResult = document.getElementById('result');
            elemResult.innerHTML = '<b>Quotes:</b><br />';
            json.quotes.forEach(function (item) {
                elemResult.innerHTML += item+'<br/>';
            });
        })
        .catch(function(err) {
            console.error("Fehler",err);
        });
}
```

Verwendet Promises!

https://www.w3schools.com/js/js_api_fetch.asp

Promises

- Ein Promise is `new Promise(function(resolve, reject) { ... });` ichen API kapselt.
- Beim Erzeugen wird eine Funktion als Parameter übergeben, der so genannte **Exekutor**
- Dieser bekommt seinerseits **zwei Callback-Funktionen** als Parameter
  - den Erfüller (**resolve**) und
  - den Zurückweiser (**reject**).
- Erfüller und Zurückweiser erwarten **jeweils einen Parameter (Erfüllungswert und Zurückweisungsgrund**), die vom Promise später zur Verfügung gestellt werden
- **Zustände:**
  - **Pending** (Wartend)
  - **Fulfilled** (Erfüllt)
  - **Rejected** (Abgelehnt)

https://www.w3schools.com/js/js_promise.asp

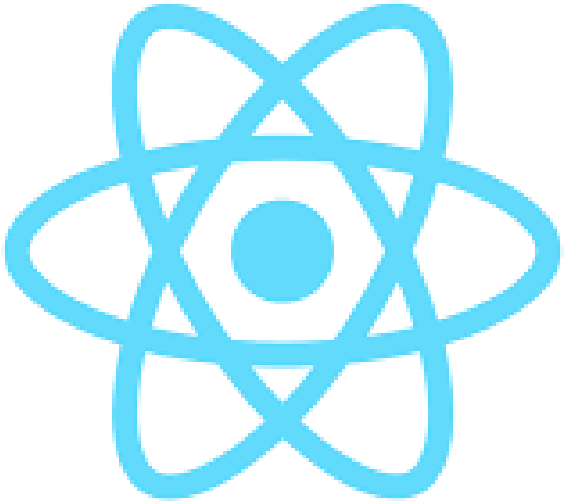
Promises | **Callbacks & Beispiel**

- Methode: **then**(*resolve*, *reject*)
  - Kann selbst aber auch wieder ein **Promise Objekt zurückgeben!** (Kettenbildung…)
- Methode: **catch**(*error*)
  - Wird zur Fehlerbehandlung zurückgegeben

```
function asyncFunction(test){
    const promise = new Promise(
        (resolve, reject) => {
            if(test){
                resolve("OK");
            }else{
                reject("NOK");
            }
        }
    );
    return promise;
}
function callAsyncFunction(test){
    asyncFunction(test).then(function (data_resolve) {
        alert(data_resolve)
    }, function (data_reject) {
        alert(data_reject)
    });
}
```

DIHSÜD

FH | JOANNEUM
University of Applied Sciences

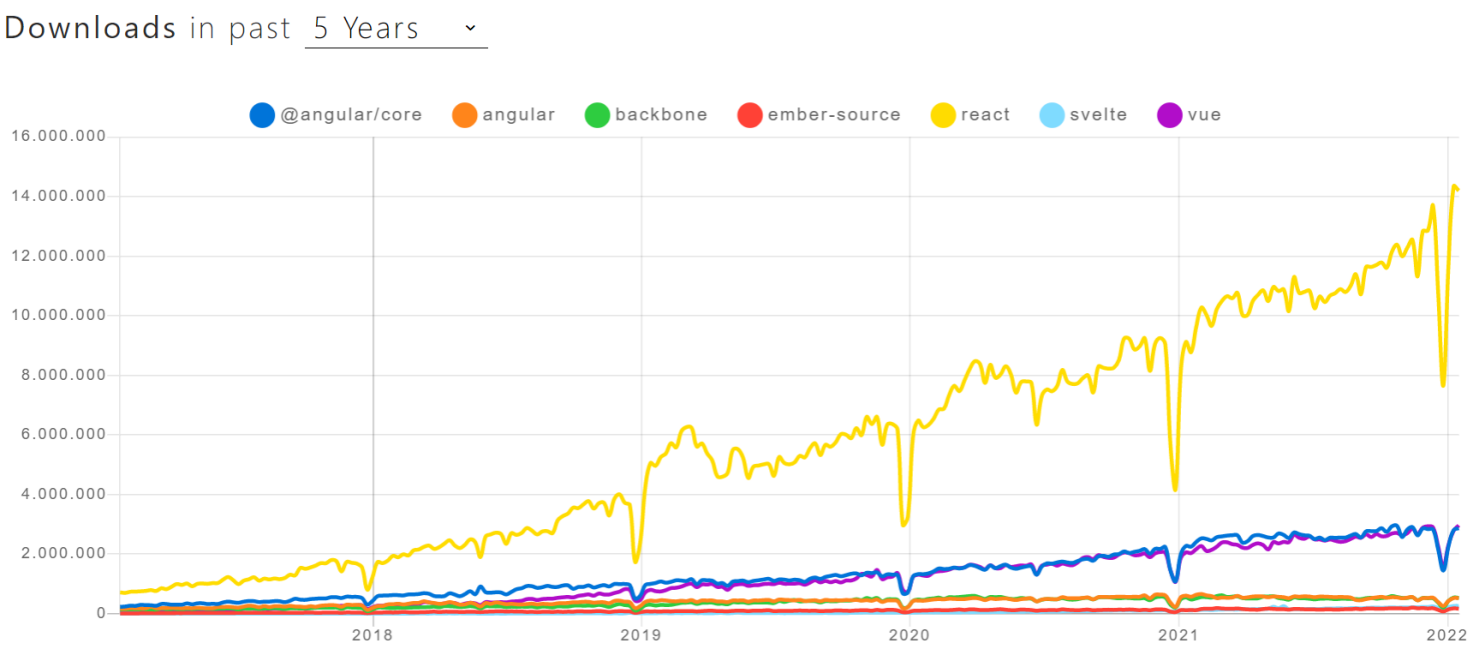


JS & TS Frameworks

Workshop
WebEngineering & Server Technologies

Web Frameworks

- Kernfunktionen
  - **DOM Management -**
  - **Data Binding**
  - **Rendering**
- Termini
  - **Komponenten und Templating**
  - **Direktiven**
  - **Routing**
  - **Reaktive Architektur (RxJS)**
  - **Testing**
  - **Modularisierung**



https://www.npmtrends.com/@angular/core-vs-angular-vs-backbone-vs-ember-source-vs-react-vs-svelte-vs-vue

npm-Trends zu Web Frameworks 2

Stats

	Stars	Issues	Version	Updated	Created	Size
@angular/core	-	-	13.2.0	17 hours ago	6 years ago	minzipped size 73.4 KB
angular	59.588	464	1.8.2	a year ago	10 years ago	minzipped size 62.3 KB
backbone	27.813	96	1.4.0	3 years ago	11 years ago	bundlephobia 429
ember-source	22.131	432	4.1.0	a month ago	5 years ago	minzipped size 401.6 KB
react	181.297	920	17.0.2	10 months ago	10 years ago	minzipped size 2.8 KB
svelte	55.189	642	3.46.3	21 hours ago	5 years ago	minzipped size 1.6 KB
vue	192.537	553	2.6.14	8 months ago	8 years ago	minzipped size 22.9 KB

https://www.npmtrends.com/@angular/core-vs-angular-vs-backbone-vs-ember-source-vs-react-vs-svelte-vs-vue

Angular
https://angular.io/

- Startk komponentenbasiert
- Clientseitiger Framework für SPAs
- Tendenz zur Platform (z.B. mit API)
- Klare Vorgaben zur Struktur
- Fokus auf
  - Dependency Injection
  - Test Driven Development

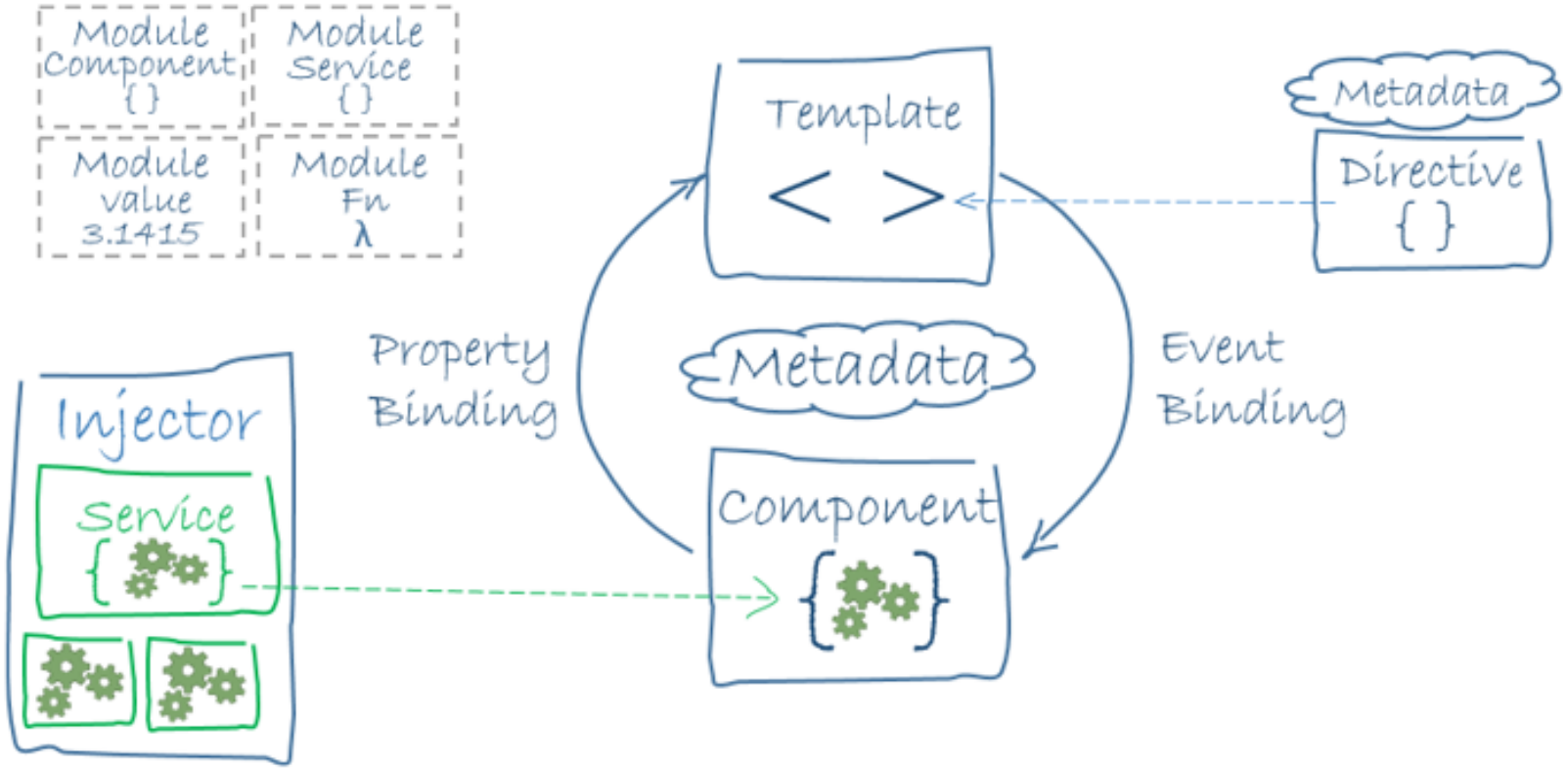
React
https://reactjs.org/

- **Minimal komponentenbasiert, was eine eigene Struktur ermöglicht/erfordert**
- **Viele Module bieten ein breites Spektrum für Lösungen**
- **Höherer Aufwand an Integration & Wartung**

Vue
https://vuejs.org/

- **Zwischen React & Angular**

Angular | **Architektur**



- Module …
  - … bieten verschiedene Funktionen
  - … werden nach Bedarf geladen
  - … haben ein "**Root Module**", mit welchem die **App startet ('AppModule')**
- Komponenten …
  - … bestehen aus Modulen
  - … definieren Views (Templates)
  - … benutzen Dienste (Services)
- Templates
  - … kombinieren HTML & Angular Markup
  - … beinhalten Direktiven für Logik
  - … beinhalten Bindings, um Daten der App mit dem DOM zu verbinden
- Servives & Dependency Injection
  - … helfen Daten zwischen Views zu teilen
  - … helfen beim Routing (Navigation)

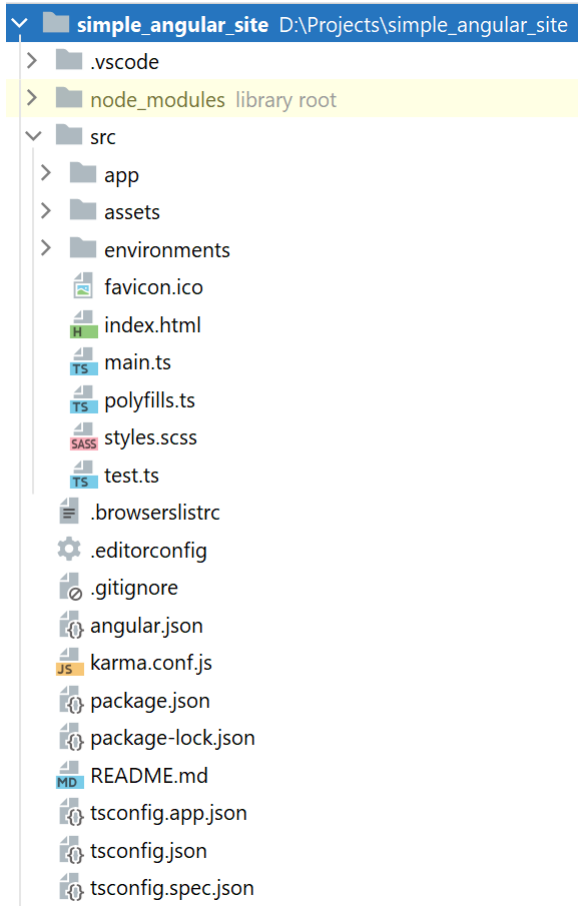
https://angular.io/guide/architecture

Angular | **Erste Schritte**

- `npm install -g @angular/cli`
  - Angular Command Line Interface global installieren
- `ng new simple_angular_site; cd simple_angular_site`
  - Damit wird ihr Angular Projekt erstellt
  - Dialogoptionen fragen nach Routing$_{ja}$ und dem gescripteten CSS Format$_{SCSS}$
- `ng serve --open`
  - Mit diesem Befehl wird ein Server und die Überwachung von Dateiänderungen gestartet
  - Unter http://localhost:4200/ ist die App verfügbar
- Tour of Heroes
  - Tutorial unter https://angular.io/tutorial
  - Bringt alle erwähnten Konzepte näher …
- Online Testen
  - Über die URL https://stackblitz.com/ hat man ein komplette Angular Projekt als Spielwiese zur Verfügung

https://angular.io/guide/setup-local

Angular | **Projektstruktur**



- src
  - app … Komponente
  - assets … Bilder und Dateien als Ressource
  - environments … Konfigurationen zum Erstellen z.B.: Produktiv oder Test
  - main.ts … Haupteinstiegspunkt über Modul "AppModule"
  - styles.scss … Geskriptetes CSS für das Projekt
  - test.ts … Haupteinstiegspunkt für das Teste
- angular.json
  - CLI Konfiguration für die Angular Arbeitsumgebung
- karma.conf.js
  - Basis für Testing
  - Siehe http://karma-runner.github.io/6.3/index.html
- tsconfig.json
  - Typescript Konfiguration
- Projektabhängig können die Inhalte variieren….
  - Zu beachten: Sie haben ein NodeJS Projekt mitsamt den üblichen Inhalten (node_modules, package.json,…)

https://angular.io/guide/file-structure

Angular | **Module**

- Eine App startet mit dem 'AppModule'
- Erstellen: `ng generate module <module_name>`
- Klasse mit dem `@NgModule` Decorator
- Beispiel für verfügbare Module mit eigenen APIs:
  - FormsModule
  - ReactiveFormsModule
  - CommonsModule
  - HttpClientModule
  - RouterModule
- Best Practices
  - Modulalisierung: Funktion kapseln
  - Klein bleiben → **Lazy Loading**

app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})

export class AppModule { }
```

Komponenten, die zum Modul gehören

Andere Module, die benötigt werden z.B. für das Routing

Services auf Modulebene
Start mit Hauptansicht (nur in Root Modul)

Angular | **Services**

- In Verbindung mit 'Dependency Injection" wird eine Klasse in die App eingefügt
- Erstellen: `ng generate service <service_name>`
- Klasse mit dem `@Injectable` Decorator
- Varianten
  - Als "**Singleton**" (nur eine Instanz) mittels **'providedIn'**
  - **Multiple Instanzen** ohne diese Eigenschaft und mittels Angabe (**'providers'**) in der Komponente (*siehe nächste Folie*)
- Bsp.: Logger

siehe *https://angular.io/guide/architecture-services*

test.service.ts

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})

export class TestService {

  constructor() { }
}
```

Angular | **Komponenten**

- Komponenten bestehen aus
  - HTML Template `<component-name>.component.html`
  - Typescript Klasse `<component-name>.component.ts`
  - CSS-Selektor und -Stile `<component-name>.component.css`
  - Test Spezifikation `<component-name>.component.spec.ts`
- Erstellen: **`ng generate component <component-name>`**
- Klasse mit dem `@Component` Decorator
  - **selector**: Komponente **einfügen**, wo im Template **entsprechender Tag** verwendet wird
  - **templateUrl**: **Vorlage** relativ zum Modul
  - **styleUrls**: **Gescriptete Stile** relativ zum Modul
  - **providers**: Service(s) für die Komponente

app.component.ts

```
import { Component } from '@angular/core';
import { TestService } from "./test.service";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss'],
  providers: [ TestService ]
})

export class AppComponent {
  title = 'simple_angular_site';
}
```

index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>SimpleAngularSite</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Angular | **Templates (Vorlagen)**

- Text Interpolation
  - Eigene Strings einfach im Template integrieren: `{{ <variable> }}`
  - Beispiel: `<p><img src="{{heroImageUrl}}"> is the <i>interpolated</i> image.</p>`
  - Berechnungen, Funktionen, etc. können ebenso eingebunden werden.
  - siehe https://angular.io/guide/interpolation
- Property Binding
  - Wird für Eigenschaften von Elementen verwendet
  - Beispiel: `<img [src]="itemImageUrl">`
  - Siehe https://angular.io/guide/property-binding
- Template Statements
  - Werden für Events verwendet
  - Beispiel: `<button (click)="deleteHero()">Delete hero</button>`
  - Events warden dabei in geschwungenen Klammern definiert
- Event Binding

Data Binding Kategorien

"One-Way" von Datenquelle zu Ziel
`{{Ausdruck}}`
`[Ziel]="Ausdruck"`

"One-Way" von Ziel(View) zu Datenquelle
`(Ziel)="Statement`

"Two Way" – in beide Richtungen
`[(Ziel)]="Ausdruck"`

https://angular.io/guide/binding-syntax

Angular | **Routing**

https://angular.io/guide/router

- Angular Router
- Erstellen einer App mit Routing:

  `ng new routing-app --routing --defaults`
- Schritte für das Routing
  - **Importieren der Komponenten** in 'AppRoutingModule'
  - **Routen in 'routes'** hinzufügen (siehe Optionen)
  - **'AppRoutingModule' in AppModule** einbinden
- Optionen
  - Pfad: **path und component** für Pfad und anzuzeigende Komponente
  - Redirecting: **redirectTo und pathMatch**
  - Wildcards: **** für keinen bestimmten Pfad
  - Hierachien: **children**, um Subpfade anzugeben!
  - **Parameter**: **Pfad/:<parameter> und ngOnInit Interface**

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

const routes: Routes = [];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Siehe https://angular.io/guide/router

Web Services

https://www.realisable.co.uk/connect/web-services/

Definiton

- **"Ein Webservice ist eine im Internet veröffentlichte Schnittstelle, welche über ein offenes Protokoll zugreifbar ist"**
- Technisch gesehen (siehe auch http://www.w3.org/2002/ws) ist damit die **automatisierte Kommunikation zwischen Applikationen über Netzwerke (z.B. Internet) gemeint.**
- **Technologien**
  - **SOAP (Simple Object Access Protocol)**
  - **ReST (Repräsentational State Transfer)**
  - **GraphQL (Graphical Query Language)**

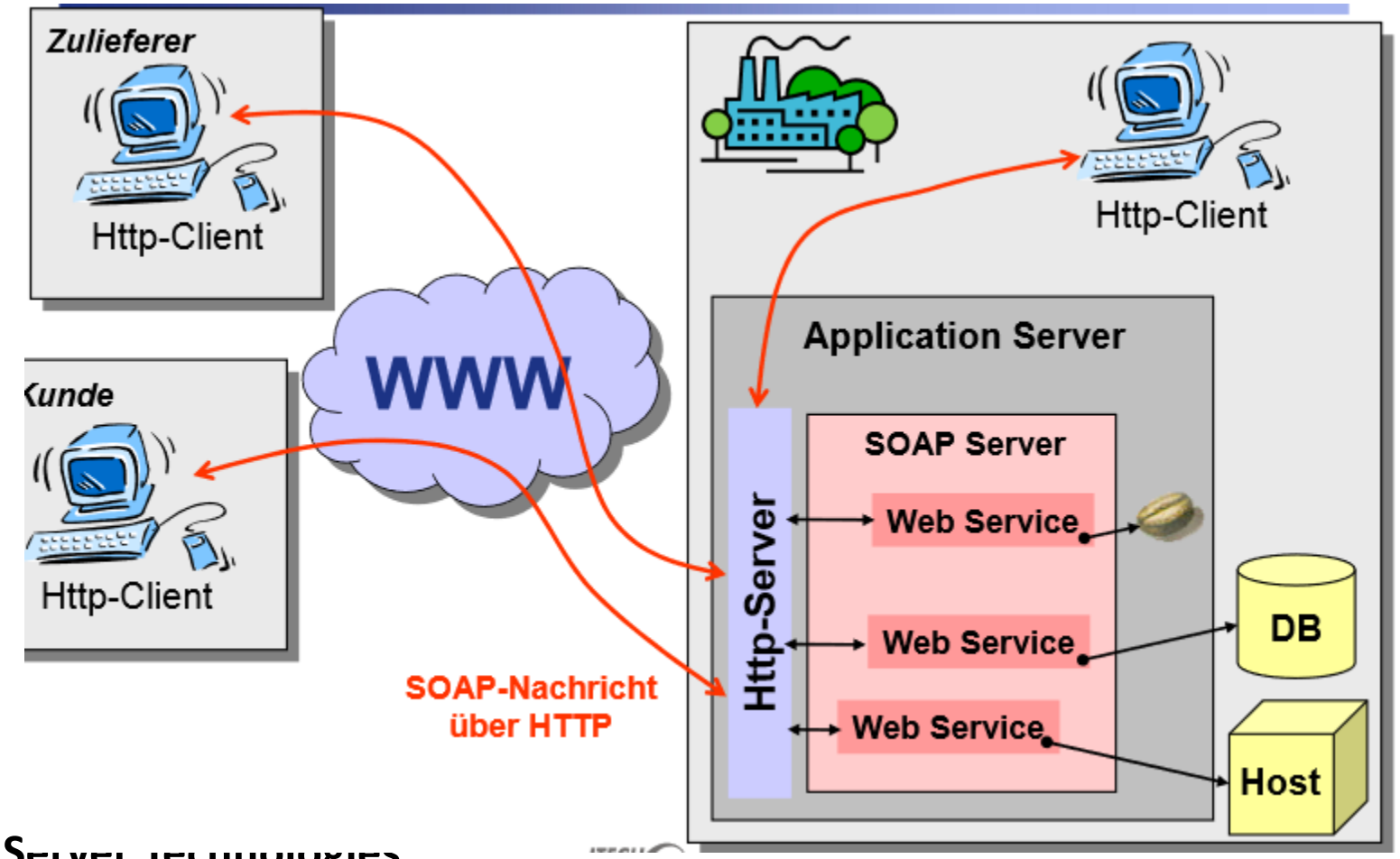
Anwendungsbeispiele

- Amazon
  - Ermitteln strukturierter Daten (Produktname, Hersteller, Preis, etc.)
  - Produkte über Schlagworte suchen
- Google
  - Integration der Suchmaschine in eigene Anwendungen
  - Integration von Karten in eigenen Anwendungen
- Webshops
  - Kommunikation mit mobilen Geräten
  - Kommunikation von anderen Diensten/Anwendungen mit dem Shop

SOA – Service Orientierte Architektur

- Software Architektur, bei der die grundlegende Infrastruktur/Funktionalität in Forem von Diensten organisiert ist:
  - Geschäftsprozesse werden dynamisch aus Diensten erstellt
  - Definierte Schnittstellen der Dienste
- Nicht standardisiert
- WebServices sind eine konkrete Implementierung davon
- Architekturstile
  - **Schnittstellenorientiert -** (z.B. SOAP)
  - **Nachrichtenorientiert -** (z.B. SOAP)
  - **Ressourcenorientiert -** (z.B. ReST, GraphQL) → Ressourcenorientierte Architektur

Funktionsweise



Vergleich herkömmlicher Kommunikation mit Webservices

Technology	Connections	Partnering	Network	Data transactions	Connectivity dimension	Governing/ standard bodies
File transfer	One to one	Trusted partners	Virtual private network (VPN)	flat or CVS file exchanges through FTP	Trade documents	Not governed
Electronic data interchange	One to one partners	Trusted partners	VPN	ANSI X12 or Edifact for inter- and intra-industry	Trade documents	Specific industry consortium
ASP networks	One to many	Trusted partners	Proprietary	Proprietary standards	Trade documents	Vendor
Web scraping	One to many	Desired partners	Internet	Text copied from a Web site	Browser to application	Not governed
Web services	Many to many	Dynamically joined partners	Internet	Simple Object Access Protocol (SOAP)	Application to application	W3C (World Wide Web Consortium), Oasis (Organization for the Advancement of Structured Information Standards), and WS-I (Web Services Interoperability)

Web services: E-commerce partner integration
Article in IT Professional · April 2005

DIHSÜD

FH | JOANNEUM
University of Applied Sciences

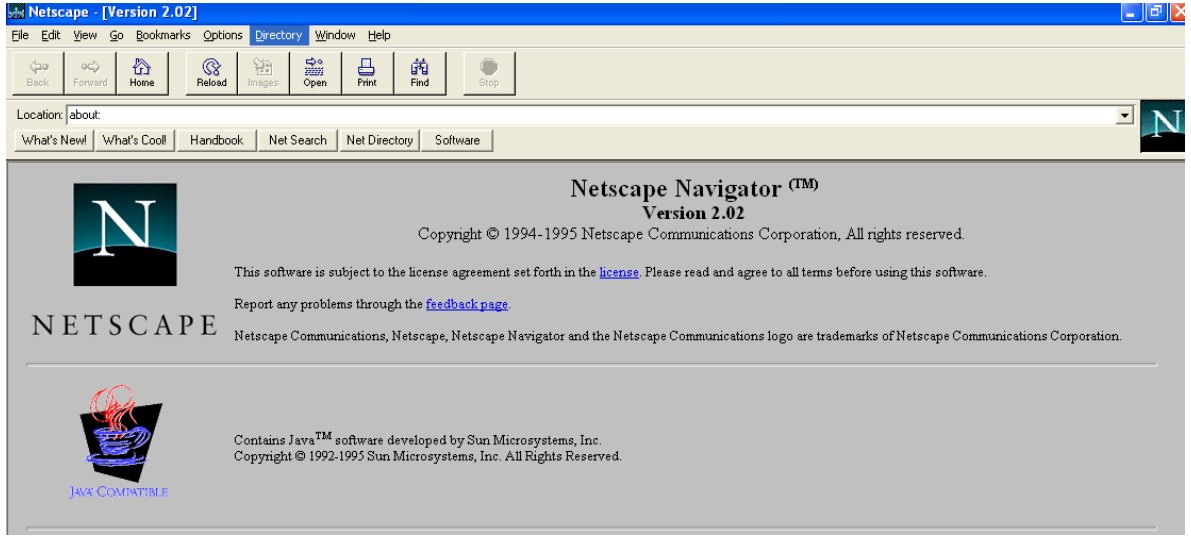
1996-

λ Self Java JavaScript

"It was all within six months from May till December (1995) that it was Mocha and then LiveScript. And then in early December, Netscape and Sun did a license agreement and it became ***JavaScript*** *" --Brendan Eich/2008*

1996*- W3C® XML

*"**Extensible Markup Language**, abbreviated XML, describes a class of data objects stored on computers and partially describes the behavior of programs which process these objects. Such objects are called XML documents. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language [ISO 8879].(*)*



* https://www.w3.org/TR/WD-xml-961114.html

Was ist JavaScript?

- **Soll wie Java aussehen**
- **Skriptsprache vornehmlich für Clients** (Browser)
- ECMA Standard (seit 1997)
  heute ECMAScript 2018
  https://kangax.github.io/compat-table/es2016plus
- Sandboxumgebung *
  - Zugriff auf Browser (-Objekte) beschränkt
  - Kein Zugriff auf Dateisystem
  - Kein Lesen oder Schreiben von Dateien

  (* Umgehung über Sicherheitsbeschränkungen im Browser u.U. möglich)

- Syntax ist Java und C ähnlich
- Programmiersprache ist
  Prozedural – Objektorientiert - Funktional
- Dynamische Typisierung
- Einsatzgebiet: Client- u. Serverseitig

Serverseitiges Javascript: Node.js

ECMA
European Computer Manufacturers Association

https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

JavaScript | **Objekte erstellen & darauf zugreifen**

- Benutzerdefinierte Objekte

```
myobject = new Object();
```

oder

```
myobject = {};
```

Letztere Schreibweise wird auch bei JSON verwendet - JavaScript Object Notation

```
pizza = {
        name : "Diabolo",
        preis : 7.5,
        zutaten : ["Käse","Tomaten","Salami","Pfefferoni"]
};

pizza = new Object();
pizza.name = "Diabolo";
pizza.preis = 7.5;
pizza.zutaten = ["Käse","Tomaten","Salami","Pfefferoni"];

pizza = new Object();
pizza["name"] = "Diabolo";
pizza["preis"] = 7.5;
pizza["zutaten"] = ["Käse","Tomaten","Salami","Pfefferoni"];

alert(pizza["name"]); // alert(pizza.name);     // "Diabolo"
alert(pizza["preis"]); // alert(pizza.preis);   // 7.5
alert(pizza["zutaten"][0]); // alert(pizza.zutaten[0]);  // "Käse"
alert(pizza["zutaten"][1]); // alert(pizza.zutaten[1]);  // "Tomaten"
```


https://www.w3schools.com/js/js_object_properties.asp

JavaScript Objekt Notation (JSON) Beispiel

http://berkeley.fh-joanneum.at/api/heroes
http://berkeley.fh-joanneum.at/api/heroes.php
Zeigt Ihnen eine Liste von HeldInnen im JSON Format! Jede/r Comic-Held/in hat eine id und einen Namen.
Beispielaufruf

Einen Helden mittels Id aufrufen (GET&POST möglich)

http://berkeley.fh-joanneum.at/api/heroes/<id>
http://berkeley.fh-joanneum.at/api/heroes.php?id=X
Zeigt Ihnen eine/n der HeldInnen im JSON Format an! Jede/r Comic-Held/in hat eine id, einen Namen, eine Beschreibung und möglicherweise ein Bild.
Beispielaufruf für id=1

Demo zur Einbindung!

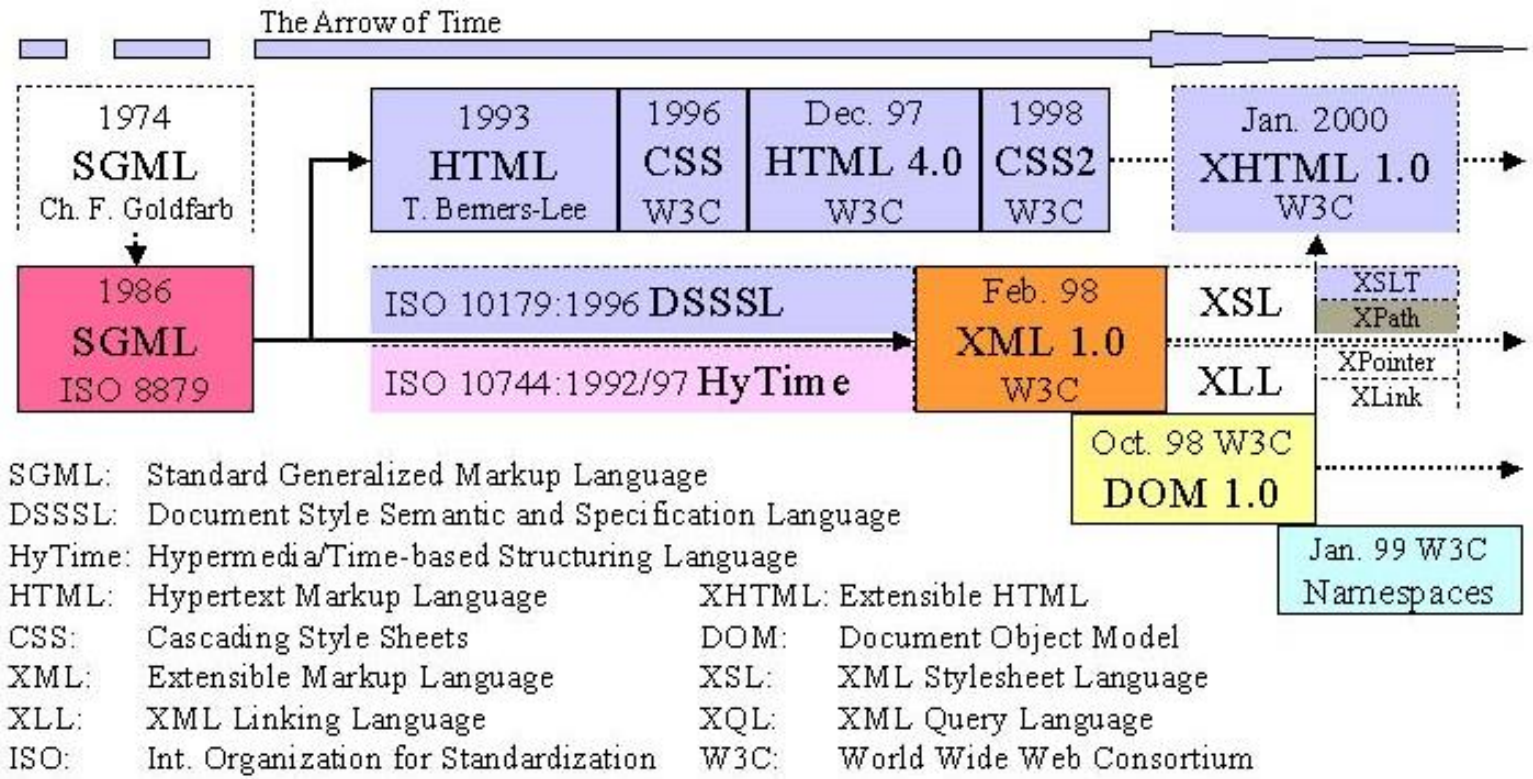
http://berkeley.fh-joanneum.at/api/heroes

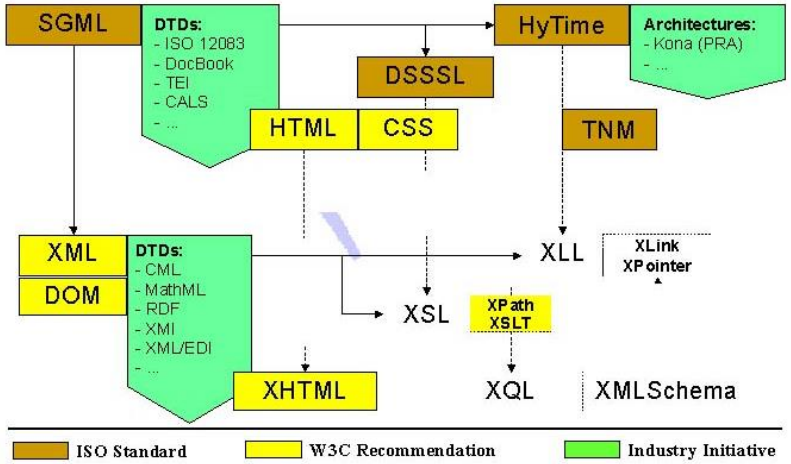

```
{
  "heroes": [
    {
      "id": 1,
      "name": "Ant-Man"
    },
    {
      "id": 2,
      "name": "Aquaman"
    },
```


http://berkeley.fh-joanneum.at/api/heroes/1


```
{
  "id": 1,
  "name": "Ant-Man",
  "description": "Ant-Man and the Wasp, comic strip superheroes created
for Marvel Comics by Stan Lee and Jack Kirby. Ant-Man debuted in Tales to
Astonish no. 27 (January 1962), and the Wasp first appeared in Tales to
Astonish no. 44 (June 1963).",
  "image":
```

XML | **Web Standards**





Quelle: https://www.jmir.org/2000/2/e12/

XML | **Aufbau "in a Nutshell"**

```
<?xml version="1.0" encoding="utf-8"?>
<students>
      <student>
            <name>Mathias</name>
            <persnr>0815123</persnr>
            <note>5</note>
      </student>
      <student>
            <name>Elmar</name>
            <persnr>0815124</persnr>
            <note>1</note>
      </student>
</students>
```

-Prolog-

```
<wurzelknoten>
   <element>
      <unterelement>
         .....
      </unterlelement>
   </element>
</wurzelknoten>
```

XML | **Das Element**

- Der **Text zwischen Start-Tag und End-Tag** (ohne die Tags selbst) ist der Inhalt.
- Die **Document Type Definition** (DTD) legt fest, was genau als Inhalt erlaubt ist
- Grundsätzlich kann ein Element folgendes enthalten:
  - Reiner Text
  - Andere Elemente
  - Mischung aus Text und Elementen
  - Kein Inhalt

Textknoten – Reiner Text

```
<name>Mathias</name>
```

Andere Element und deren Inhalte

```
<name><vorname>Mathias</vornamename></name>
```

Mischung aus Text und Elementen
(nach besonderen Regeln)

```
<p>Hallo <em>Du</em>!</p>
```

Ohne Inhalt
Wird implizit geschlossen

```
<br />
```

XML | **Die Attribute**

- Attribute enthalten **Zusatzinformationen** für Elemente in Starttags von Elementen
- Zahl und Art der Attribute kann (in der DTD) festgelegt werden
- Die **Reihenfolge** im Tag ist **willkürlich**
- Die Attributwerte stehen **immer in Anführungszeichen**
- Der Attributbezeichner ist **Case Sensitive**
- **Keine Unterstrukturierung** im Vergleich zu Elementen

```
<p id="1" typ="Einleitung" >
    Dies ist zufällig der erste Abschnitt
    eines Textes.
</p>
```

XML | **Anwendungsgebiete**

- Darstellung und Speicherung von Daten in hierarchisch strukturierter Form (**XML Datenbanken**).
- Für die **Definition von Austauschformaten zwischen verschiedenen Anwendungen** in heterogenen Netzen. Hierzu dienen die Standards SOAP und XML-RPCs .
- für die **Entwicklung von verschiedenen Beschreibungsprachen** verwendet. Vorhandene Beispiele dafür sind z. B. XHTML, SVG und MathML.
- **Transformation** in verschiedene Formate mittels **XSL(T)** Werkzeugen!

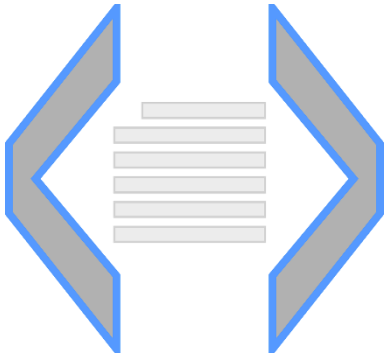
XML "Dialekte"

RDF
Resource Description Framework
WML
Wireless Markup Language
GML
Geography Markup Language
KML
Keyhole Markup Language
SVG
Scalable Vector Graphics (2D-Grafiken)
MathML
Mathematical Markup Language
SMIL
Synchronized Multimedia Integration Lang.
RSS
Really Simple Syndication

Für und Wider

- **Anwendungsspezifische** Sprachen
- Auszeichnungselemente haben **semantische Bedeutung**
- **Eindeutige** Regeln und Strukturen
- **Einfache Verarbeitung durch Anwendungsprogramme**
- **Trennung von Inhalt und Darstellung** Unterschiedliche Darstellungen, geräteabhängige Visualisierungen mit XSL(T)
- Gut **strukturierte Textdokumente-**Damit sind sie sowohl von **menschlichen Lesern** als auch von **Anwendungsprogrammen** einfach zu interpretieren und zu verarbeiten! (xPath, xQuery,…)
- Strenge Fehlerüberprüfung (**Validierung**)

- Mehr **textueller Overhead** und sich **wiederholende Elemente** als in vergleichbaren Formaten (z.B. JSON)
- Dadurch auch **mehr Kosten** in Speicher und Transport der Daten



Simple Object Access Protocol (SOAP)

- Standardisiertes Nachrichtenprotokoll (http://www.w3.org/TR/soap)
- Remote-Prozeduraufruf-Mechanismus mit XML als Nachrichtenformat
- Verschiedene Transportprotokolle möglich (JMS, SMTP,…)

SOAP | Vor- und Nachteile

- Vorteile
  - Standardisierung (http://www.w3.org/TR/soap/)
  - Plattformunabhängigkeit
  - Offenheit
  - Robustheit
  - Skalierbarkeit
- Nachteile
  - Viel Overhead (wegen Nachrichtenformat XML) und dadurch
  - geringe Performance

SOAP | Nachrichtenformat

- Der XML-Teil besteht hauptsächlich aus dem so genannten "**Envelope**"-XML-Element. Dieses wiederum enthält die beiden XML-Elemente "Header" und "Body", wobei das "Header"-Element auch entfallen kann.

- Das "Body"-Element muss enthalten sein. Hierin wird der eigentliche Inhalt platziert, also die Daten, eine Meldung, eventuell eine Fehlermeldung oder ein RPC-Funktionsaufruf.

```
+-------------------------------------------------HTTP-Header--+
| POST /realtimequotes/ncrouter HTTP/1.1                       |
| Host: mysoapserver                                           |
| Content-Type: text/xml; charset=utf-8                        |
| ...                                                          |
+--------------------------------------------------------------+

  <?xml version="1.0" encoding="UTF-8"?>
+-----------------------------------------------SOAP-Envelope--+
| <SOAP-ENV:Envelope                                           |
|  xmlns:SOAP-ENV="http://..."                                 |
|  ...                                                         |
| +-----------------------------------SOAP-Header (optional)--+ |
| | <SOAP-ENV:Header>                                          | |
| |   <t:transaction                                           | |
| |    xmlns:t="..." ...                                       | |
| | </SOAP-ENV:Header>                                         | |
| +------------------------------------------------------------+ |
| +-----------------------------------------------SOAP-Body--+ |
| | <SOAP-ENV:Body>                                            | |
| |   <m:getLastTradePrice xmlns:m="trading-uri">              | |
| |     <ticker>SUNW</ticker>                                  | |
| |   </m:getLastTradePrice>                                   | |
| | </SOAP-ENV:Body>                                           | |
| +------------------------------------------------------------+ |
| </SOAP-ENV:Envelope>                                         |
+--------------------------------------------------------------+
```

SOAP | Beispielanfrage

```
POST /perl/soaplite.cgi HTTP/1.0
Host: services.xmethods.net:80
Content-Type: text/xml; charset=utf-8
Content-Length: 546
SOAPAction: ""

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:BabelFish xmlns:ns1="urn:xmethodsBabelFish"
                   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <translationmode xsi:type="xsd:string">de_en</translationmode>
      <sourcedata xsi:type="xsd:string">Hallo Welt, Guten Tag</sourcedata>
    </ns1:BabelFish>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP | Beispielantwort

```
HTTP/1.1 200 OK
Date: Thu, 05 Sep 2002 08:01:13 GMT
Server: Apache/1.3.22 (Unix) Enhydra-Director
SOAPServer: SOAP::Lite/Perl/0.52
Content-Length: 546
Connection: close
Content-Type: text/xml; charset=utf-8

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
                   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <namesp1:BabelFishResponse xmlns:namesp1="urn:xmethodsBabelFish">
      <return xsi:type="xsd:string">hello world, good day</return>
    </namesp1:BabelFishResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Representational State Transfer (ReST)

- Im Jahr 2000 von Roy Thomas Fielding geprägt und definiert allgemeine Grundlagen eines Architekturstils
- Eine Ressource kann dabei über verschiedene Medientypen dargestellt werden, auch Repräsentation der Ressource genannt.
- Die Bezeichnung „Representational State Transfer“ soll den Übergang vom aktuellen Zustand zum nächsten Zustand (state) einer Applikation verbildlichen.

ReST | **CRUD Beispiele**

REST-konforme Verwendung von GET, PUT, POST und DELETE sowie Assoziationen zu CRUD

HTTP	CRUD	Beispiel-URL und -Bedeutung
GET	Read	http://xyz.de/Artikel/Buecher --> Liste aller Bücher; http://xyz.de/Artikel/Buecher/4711 --> Informationen zu dem per ID ausgewählten Buch; http://xyz.de/Artikel/Buecher?isbn=1234567890 --> Informationen zu dem per Suchkriterium ausgewählten Buch
PUT	Update, Create	http://xyz.de/Artikel/Buecher/4711 --> Update (oder Create) des per ID identifizierten Artikels
POST	Create	http://xyz.de/Artikel/Buecher --> Neuen Artikel hinzufügen (mit neuer ID) (dabei wird üblicherweise die automatisch vergebene ID returniert)
DELETE	Delete	http://xyz.de/Artikel/Buecher/4711 --> Diesen per ID identifizierten Artikel löschen

Vergleich | ReST versus SOAP

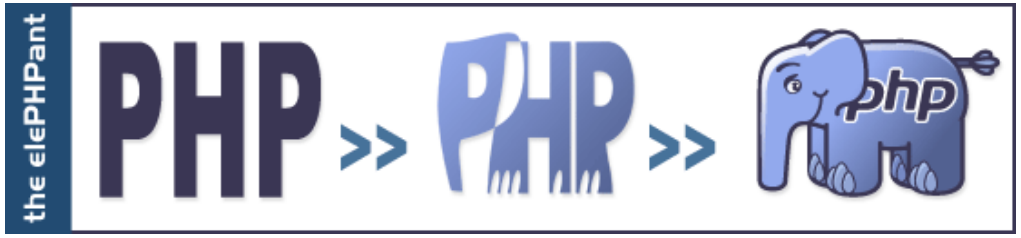
Attribute / Features	RESTful Web Services	SOAP Web Services
Basis-Standard	REST	W3C u.a.
Architekturstil	"ROA" (Resource Oriented Architecture), ressourcenorientiert mit generischer uniformer Schnittstelle (GET, PUT, POST, DELETE)	eher "SOA" (Service Oriented Architecture), schnittstellen-/nachrichtenorientiert
Bevorzugtes Anwendungsgebiet	datenorientierte synchrone kurz laufende Services	sowohl datenorientierte als auch lang laufende prozessorientierte Services, synchron und asynchron
Serverseitiger Zustand/Status	zustandslos	eher zustandslos, kann aber auch zustandsbehaftet sein
Formale syntaktische Schnittstellenbeschreibung	nur zum Teil standardisiert: Nachrichtenformate können durch XML Schema Definition und WADL beschrieben werden	vollständig durch WSDL
Nachrichtenformat, Repräsentation	Text, HTML, XML, JSON, binär, ...	XML (plus Attachments)
Nachrichtenprotokoll, Anwendungsprotokoll	REST, HTTP	SOAP
Transportprotokoll	HTTP	HTTP, SMTP, JMS, ...
Asynchrone Kommunikation	nicht direkt (nur über Umwege simuliert, z.B. über Atom-Feeds im Atom Syndication Format)	ja, per JMS und WS-Notification (WSN)
Operationsabhängiger Zugriffsschutz	einfach, per Webserver oder Firewall	per WS-Security (WSS)
Bookmarks/Links	ja	nein
Caching	einfach	schwierig
Skalierbarkeit	optimal	kann schwieriger sein
Performance	gut	eher schlechter
Lose Kopplung, Interoperabilität, Plattformunabhängigkeit, Internetfähigkeit	ja	ja

DIHSÜD

FH JOANNEUM
University of Applied Sciences

Pre Hypertext Processing (PHP)
Serverseitiges Scripting

Workshop
WebEngineering & Server Technologies



- PHP steht für
  - **P**HP: **H**ypertext **P**reprocessor oder logischer verdreht
  - **P**re **H**ypertext **P**rocessing/or und historisch
  - **P**ersonal **H**ompage **T**ools (PHP1)
- Es ist als **serverseitige Skriptsprache** konzipiert
- Scripts werden am Server **interpretiert**
- Es muss **am Webserver installiert und konfiguriert** sein, um in Webseiten eingebunden zu werden
- Die Homepage ist auf **http://php.net**

https://www.w3schools.com/php/default.asp

Ausführung

- Die Konfigurationsdatei für PHP nennt sich **php.ini**
- Kernkompetenz ist die **serverseitige Scripting** für Webseiten und Webapplikationen
- **PHP Scripts** in der Shell sind möglich

```
#!/usr/bin/php
echo "Hello World";
```

- Möglich aber kaum verwendet ist die Verwendung als **Desktop-Anwendung** mittels PHP-GTK GUIs
  (**G**imp **T**ool**k**it oder auch Graphical Toolkit)

Grundlegende Syntax

- Scriptblock innerhalb HTML

```
<?php /* Code */ ?>
```

- Beispiel

```
<?php echo "Hello World"; ?>
```

- Abschließen von Befehlen mit Strichpunkt (;)
- Kommentare
  - Einzeilig
  - Mehrzeilig:

```
// Eine Zeile,
/* oder mehrere
   Zeilen lang */
```

https://www.w3schools.com/php/php_syntax.asp

Variablen | **Grundlagen**

- Beginnen mit **$**
  - **Case Sensitive**
  - Name **beginnt mit Buchstaben oder _**
  - **Alphanumerische Zeichen und _** können verwendet werden
- **Schwache Typisierung**
  - **Typ** wird **durch** den **zugwiesenen Wert bestimmt**
- Referenzen/Zeiger auf Variablen durch **&**

https://www.w3schools.com/php/php_variables.asp
https://www.php.net/manual/de/language.variables.basics.php

```
$var_name = "Mathias";
$name = "Max";
$nr = 12;
```

Tip: Schnellausgabe

```
<?=$var_name?>
```

```
$ref_name = &$var_name;
echo $ref_name
```

Ausgabe: `Mathias`

Variablen | **Standardtypen**

- Boolean `<?php $isTrue = True; // oder False`
- Integer `<?php $iNumber = 20;  //  oder -20  dezimal`
  `oder 024  oktal`
  `oder 0x14  hexadez.`
  `oder 0b10100  binär`
- Float `<?php $fNumber = 1.2;  //  oder 1.2e1  Exponential`
- String `<?php $sHello = 'Hello';// oder "Hello"`
  - Bei einfachen Hochkommas (') funktionieren **Maskierungen** und **Escapesequenzen NICHT**
  - Bei doppelten Hochkommas (") **funktionieren Maskierungen** und **Escapesequenzen**

https://www.w3schools.com/php/php_datatypes.asp
https://www.php.net/manual/de/language.types.php

Alternative „heredoc“

```
$bar = <<<MEINTEXTHIER
$sHello World
MEINTEXTHIER;
```

Ausgabe: `Hello World`

ähnlich wie in Linux

Alternative „nowdoc“

```
$bar = <<<'OHNEESCAPING'
$sHello World
OHNEESCAPING;
```

Ausgabe: `$sHello World`

Variablen | **Spezialtypen**

- **Variable ohne Wert**
  Wenn der Variable **nichts oder explizit `Null`** zugewiesen oder mit `unset($value)` gelöscht wurde
- **Not a Number**
  Wenn das Ergebnis einer Berechnung **nicht repräsentierbar** oder **undefiniert** ist
- **Resource** (siehe http://php.net/manual/de/resource.php)
- **Welcher Typ?** Mit der Funktion `gettype ( mixed $var )` kann man es herausfinden!

NULL

```
$value = NULL; is_null($value);
```

NaN

```
$value = NaN; is_nan($value);
```

Variablen | **Gültigkeit**

- **Lokal**
  Wenn innerhalb von Funktionen definiert und gültig
- **Global**
  Außerhalb von Funktionen definiert
  Kann **in** Funktionen nur über Schlüsselwort verwendet werden: `global $varGlobal;`
  Werden im Array `$GLOBALS` gespeichert!
- **Statisch**
  Damit geht ein Wert in einer Funktion nicht verloren sondern bleibt bis zum nächsten Funktionsaufruf `static $varStatic;`

https://www.w3schools.com/php/php_variables_scope.asp
https://www.php.net/manual/de/language.variables.scope.php

Variablen | **Gültigkeit - Beispiel**

```php
<?php
        $varGlobal = "Statische Variable: ";
        function funktionTest(){
                global $varGlobal;
                static $varStatic = 0;
                $sUmbruch = "<br>"; // Lokal
                $varStatic++;
                echo $varGlobal . $varStatic . $sUmbruch;
        }
        funktionTest();
        funktionTest();
```

Ausgabe:

```
Statische Variable: 1
Statische Variable: 2
```

Wichtig! In Javascript ist die Gültigkeit etwas anders!

Arrays

Ist eigentlich eine **geordnete Zuweisung oder Ordered Map**

```
    array (
    Schlüssel => Wert,
    Schlüssel => Wert
  )
```

Schlüssel: String oder Integer

Wert: Jeder Typ

Besondere Variablen presenting ...

https://www.w3schools.com/php/php_arrays.asp
https://www.php.net/manual/de/book.array.php

Arrays | **Beispiele**

```
<?php $arrTest01 = array(1,2,3,4); ?>

<?php $arrTest02 =
            array( 1=>"Eins", 2=>"Zwei"
      ); ?>
<?php $arrTest03 =
            array("alphabet" =>
                                array("a"=>1,"b"=>2)
      ); ?>
<?php var_dump($arrTest03); // Anzeigen
```

Arrays | **Zugriff**

```
echo $arrTest01[0]; // Ordinate
```

Ausgabe: **1**

```
echo $arrTest02[1]; // Index
```

Ausgabe: **Eins**

```
echo $arrTest03["alphabet"]["a"];
```

Ausgabe: **1**

```
foreach($arrTest03 as $index => $wert){
    echo $index . " " . $wert["a"];
}
```

Ausgabe: **alphabet 1**

```
array(1,2,3,4);
array( 1=>"Eins", 2=>"Zwei");
array("alphabet" =>array("a"=>1,"b"=>2))
```

Arrays | **Bearbeitung**

Hinzufügen ohne Ordinate mit []
(Ordinate wird automatisch angelegt)

```
array( 1=>"Eins", 2=>"Zwei");
```

Hinzufügen mit Ordinate oder Zeichenkette

```
$arrTest02[] = "Drei";
```

Entfernen eines (unerwünschten) Elements oder **ganzen Array leeren**

```
$arrTest02[3] = "Drei";
$arrTest02["xy"] = „test";
unset($arrTest02["xy"]);
unset($arrTest02);
```

Vordefinierte Variablen

Alles Arrays!

`$GLOBALS`	Referenziert alle Variablen, die im globalen Gültigkeitsbereich vorhanden sind
`$_SERVER`	Informationen über Server und Ausführungsumgebung
`$_GET`	HTTP GET-Variablen
`$_POST`	HTTP POST-Variablen
`$_FILES`	HTTP Dateiupload-Variablen
`$_REQUEST`	HTTP Request-Variablen
`$_SESSION`	Sessionvariablen
`$_ENV`	Umgebungsvariablen
`$_COOKIE`	HTTP Cookies
`$argc`	Die Anzahl der an das Skript übergebenen Argumente
`$argv`	Array der an das Skript übergebenen Argumente

Eine Auswahl-
mehr auf http://php.net

Variable Variablen(namen)

Mit einem weiteren **$** vor dem Variablennamen, wird auf den Inhalt der Variablen als Name referenziert

```
$meinvariablenname = "vorname";
$$meinvariablenname = "Mathias"; // $vorname = "Mathias";
echo "$meinvariablenname ${$meinvariablenname}";
echo "$meinvariablenname $vorname";
```

Ausgabe in beiden Fällen: `vorname Mathias`

Anmerkung: Das lässt sich beliebig fortsetzen – zum Beispiel: **`$$$$$variable`**

Falls man mal so einen Anwendungsfall hat …

Konstanten

Selbstdefiniert
Beispiel:

```
<?php define("FOOBAR", "irgendwas");
```

https://www.w3schools.com/php/php_constants.asp

„Magische“ Konstanten vom System
Beispiele:

```
__LINE__ // Aktuelle Zeilennummer oder
__DIR__ // Aktuelles Verzeichnis
```

https://www.php.net/manual/de/language.constants.php
https://www.php.net/manual/de/language.constants.predefined.php

Funktionen:

```
bool defined ( string $name );
// ob eine Konstante definiert ist
```

Funktionen

- Funktionen werden mit dem **Schlüsselwort** „function“ definiert: `function funktion(){ …`
- Es können **Parameter** und Referenzen/Zeiger übergeben werden: `function funktion(&$a){ …`
- Es können **Vorgabeparamter** definiert werden: `function funktion($a,$b="Baehh"){ … //nur am Ende`
- Eine variable Anzahl von Parametern wird auch unterstützt!

- **Rückgabewerte** mit "return": `return $x;` Standardmäßig (ohne `return`) wird `NULL` zurückgegeben

- **Variablenfunktion**
  Wenn man an eine Variable `()` anhängt, versucht der Interpreter eine Funktion mit dem Namen des Variablenwertes aufzurufen

```
function test(){ echo "TEST"; }
$meinefunktion = "test";
$meinefunktion();
```

Ausgabe: Test

https://www.w3schools.com/php/php_functions.asp
https://www.php.net/manual/de/language.functions.php

Dateien einbinden

include bindet eine angegebene Datei ein und führt sie aus.

require ist ähnlich, wirft aber im Fehlerfall einen E_COMPILE_ERROR Fehler und beendet so die Programmausführung während include nur eine Warnung (E_WARNING) generiert.

include_once, require_once überprüft, ob die Datei schon eingebunden wurde.

Beispiel

```
<html>
    <body>
        <?php include("header.php"); ?>
        <h1>Welcome to my home page!</h1>
        <p>Some text.</p>
    </body>
</html>
```

https://www.w3schools.com/php/php_includes.asp

Parameterübergabe

- Übergabe von Parametern via HTTP GET und POST
- Parameter über URL

```
http://localhost/welcome.php?name=Mathias
echo $_GET["name"];
```

- Parameter werden im „body“ eingeschlossen

```
<form action="welcome.php" method="post">
   Name: <input type="text" name="name" />
echo $_POST["name"];
```

https://www.w3schools.com/php/php_forms.asp

PHP Beispiel

welcome.html

```
.
<form action="welcome.php" method="post">
        Name: <input type="text" name="username" />
        Alter: <input type="text" name="age" />
       <input type="submit" />
</form>
.
```

welcome.php

```
.
Welcome <?php echo $_POST["username"]; ?>!<br/>
You are <?php echo $_POST["age"]; ?> years old.
.
```

Dateizugriff

- Dateien als Speicher von Daten (auch XML)
- Rechte beachten !!!
- Auch entfernte Dateien/URLs können geladen werden:
  ```
  $datei = fopen ("http://www.fh-joanneum.at/", "r");
  ```
- Achtung bei gleichzeitigem Zugriff
- ```
  <?php
          $file = fopen("welcome.txt", "r");
          // do something
      fclose($file);
  ?>
  ```
- File Modes – *siehe http://at1.php.net/manual/de/function.fopen.php*
  - r, r+, w, w+, a, a+, x, x+

https://www.w3schools.com/php/php_file.asp
  ```

Sessions & Cookies

HTTP ist „stateless“

Oft ist aber eine Beobachtung des Zustands einer Webanwendung notwendig. Z.B.:

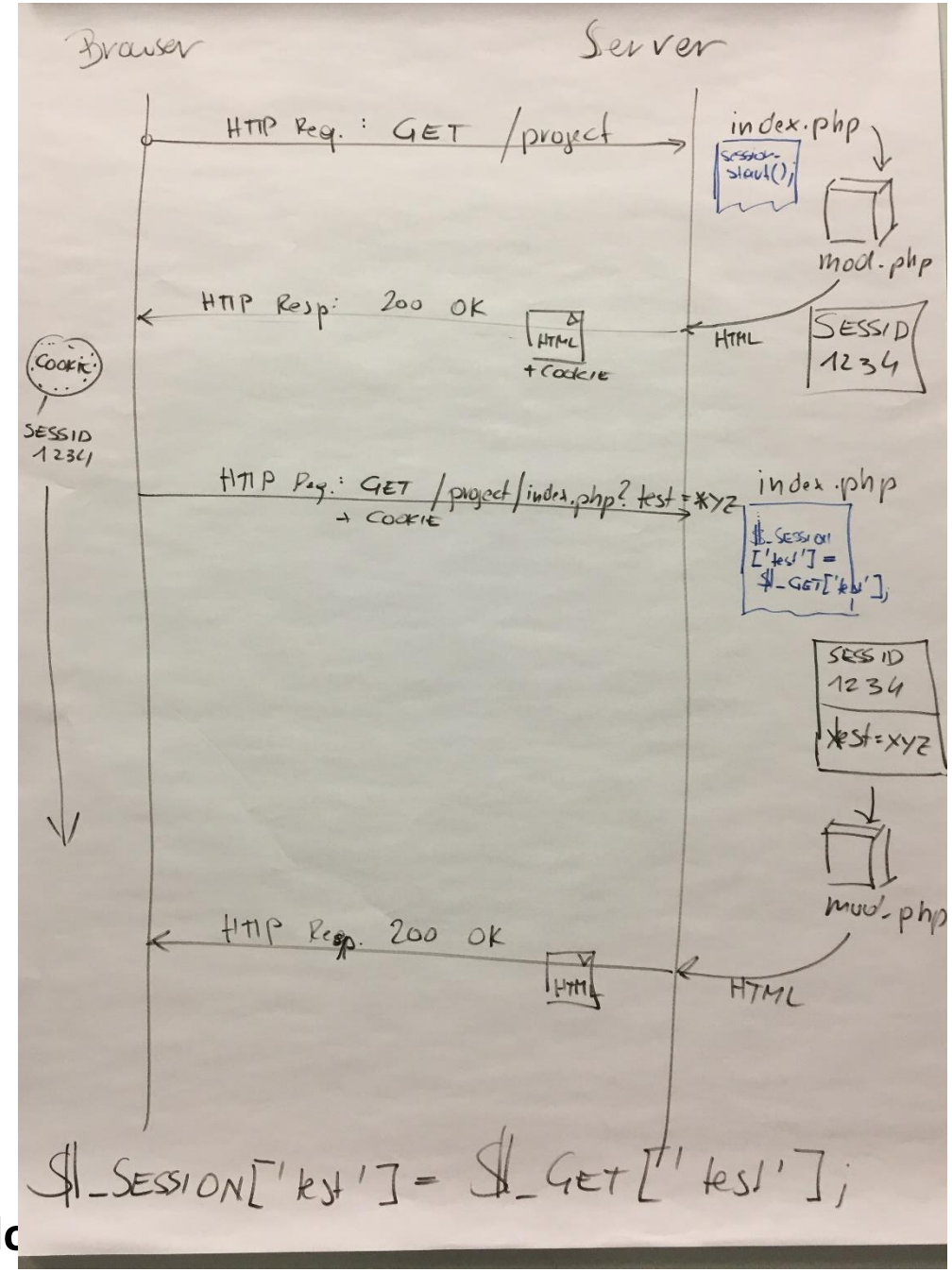
- Eingeloggt bleiben
- Online Einkauf - Warenkorb

Sessions (Variablen die den Zustand eines einzelnen Users speichern)

```
session_start(); $_SESSION['Sessionname']="Hallo";
// Code
echo "Name=". $_SESSION['Sessionname'];
```

Cookies (Vom Server an den Client geschickte Strings dort gespeichert)

```
setcookie(name, value, expire, path, domain);
// Code
echo $_COOKIE["user"];
```

https://www.php.net/manual/de/book.session.php
https://www.w3schools.com/php/php_sessions.asp
https://www.w3schools.com/php/php_cookies.asp

Sessions in PHP