

FOM Hochschule

Web Technologie

(1) Formales & Vorstellung
Orientierungsskript

Dozent: Axel Sierau
Folien: Prof. Dr. Peter Hoffmann

Copyright

**© FOM Hochschule für Oekonomie & Management
gemeinnützige Gesellschaft mbH (FOM), Leimkugelstraße 6, 45141 Essen**

Dieses Werk ist urheberrechtlich geschützt und nur für den persönlichen Gebrauch im Rahmen der Veranstaltungen der FOM bestimmt.

Die durch die Urheberschaft begründeten Rechte (u. a. Vervielfältigung, Verbreitung, Übersetzung, Nachdruck) bleiben dem Urheber vorbehalten.

Das Werk oder Teile daraus dürfen nicht ohne schriftliche Genehmigung des Urhebers / der FOM reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden. Dies schließt auch den Upload in soziale Medien oder andere digitale Plattformen ein.

Kurzvorstellung

Die Studierenden:

- Name ?
- Aus woher ?
- Hobby(ies) / Sportart ?
- Aktueller Job / Arbeitgeber ?
- Erfahrung mit Web Technologie ?
- Wo sehen Sie sich in fünf Jahren ?
- Was war Ihr positives Erlebnis der letzten Woche ?
- Erwartungshaltung an diesen Kurs ?

Kurzvorstellung

Axel Sierau

- E-Mail: sierau.axel@gmail.com
- Tel.: 0177-4459170
- Wolfsburg
- Dannenberg/Elbe
- Abi
- Gejobbt bei Minimal / REWE
- Zivildienst
- USA u.v.a.m.



Kurzvorstellung

Axel Sierau

- Sport- und Wirtschaftswissenschaften in Köln studiert (DSHS und Uni)
- 20 Jahre Marketing(leiter) in der Pharmabranche
- 13 Jahre am Institut für Sportökonomie und Sportmanagement (DSHS)
- 7 Jahre für die Stiftung Stadtgedächtnis in Köln
- 5 Jahre SportMeetsCharity (Stiftung e.V. und GmbH)
- 4 Jahre SPONSORIS – FBIN
- 2 Jahre SC Bayer 05 Uerdingen / Beachclub Krefeld
- Handballer (EHF – 2020 HOK – Handball Online Kongress)

Kurzvorstellung

Axel Sierau

- Seit 2008 SportTreff / #CGN-Sports e.V., Vorstand
- Seit 2010 European Handball Federation – EHF Expert
- Seit 2012 Dozent an verschiedenen privaten Hochschulen, Unis und Akademien
- Seit 2022 Sparringspartner für Startups und Scaleups

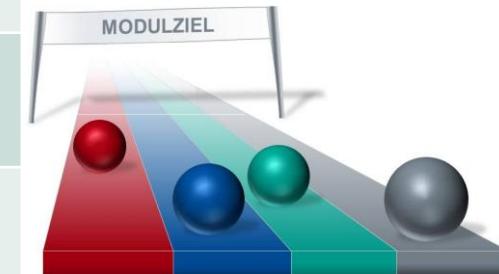
Kurzvorstellung

Axel Sierau

- Veröffentlichungen:
 - „Professionalisierung im Sportmanagement“, Aachen, 1999
Hrsg.: Horch, H.-D. / Heydel, J. / Sierau, A.
 - „Finanzierung des Sports“, Aachen, 2002
Hrsg.: Horch, H.-D. / Heydel, J. / Sierau, A.
 - „Events im Sport. Marketing, Management, Finanzierung“, Köln, 2004
Hrsg.: Horch, H.-D. / Heydel, J. / Sierau, A.

Aufwand & Outcome

Präsenzstd.:	36,0 UE
Eigenstudium:	58,00 ZStd.
Student Consulting/ Praxistransfer:	40,00 ZStd.
Workload:	125,00 ZStd.
ECTS:	5



Die Studierenden können nach erfolgreichem Abschluss des Moduls

- die Entwicklung von Web-Anwendungen planen und selbstständig **Vorgehensmodelle** entwickeln
- die grundlegenden Techniken zur **Programmierung von Webanwendungen** erklären und diese Techniken grundlegend anwenden
- die unterschiedlichen Charakteristika von Hypermedia und Multimedia erklären
- aktuelle **Interaktionsparadigmen** erklären und diese in der Web-Entwicklung anwenden
- sowohl **statische** als auch **Webprojekte** mit client- und serverseitiger **Dynamik** umsetzen
- die jeweiligen Vor- und Nachteile der einzelnen Technologien und Plattformen identifizieren und beurteilen
- Webprojekte veröffentlichen



Die Studierenden werden durch das erworbene Wissen in die Lage versetzt, ...

... mehrschichtige verteilte Softwaresysteme auf Basis von Webtechnologien zu entwickeln.

Die Produktivität von Unternehmensprozessen kann gesteigert werden, durch moderne Softwareplattformen und Nutzungskonzepte. Durch Webtechnologien können Plattformen geschaffen werden, die im Rahmen von Digitalisierungsstrategien vermehrt Anwendung finden.

Das erworbene Wissen kann zum erfolgreichen Umsetzen komplexer IT-Projekte und Prozessoptimierungen beitragen.



Curriculum /Veranstaltungsgliederung

1	Formales, Vorstellung
2	Einführung
2.1	Allgemeine Einführung
2.2	Historie Internet, Web & Web-Technologien
3	(Web) Server – Client Kommunikation
3.1	Protokolle & Architekturen
3.2	Web- & Microservices
3.3	Authentifizierungs- und Authorisierungskonzepte
4	Grundlagen der Mediengestaltung
4.1	Media & Multimedia
4.2	Media & Modalitäten – Interaktion & Wahrnehmung
4.3	Hypermedia
5	Statische Webseiten
5.1	HTML
5.2	CSS
5.3	Medieneinbindung
6	Dynamische Webseiten
6.1	DOM, HTML & CSS
6.2	JavaScript & JSON
6.3	Weitere Konzepte & Sprachen der Webprogrammierung
7	Praxis
7.1	Vorstellung eines (ausgewählten) (JavaScript-) Frameworks
7.2	Vorstellung einer (ausgewählten) Thematik für ein eigenes Projekt
7.3	Betreute, eigenständige Entwicklung des (ausgewählten) Themas
8	Wiederholung (opt.)

Die Modulnote setzt sich folgendermaßen zusammen:

100% Seminararbeit

- Umfang \pm 4.000 Wörter
- In der Seminararbeit wird das gewählte Thema auf die Praxis bezogen und in einem eigenen Gliederungspunkt dargestellt (mind. 25% des Projektarbeitsumfangs)

- **Gruppenarbeit:** Gruppen zu je **zwei bis max. (!) vier** Studenten
 - Als Gruppenarbeit: **EINE** Arbeit (als Projektdokumentation) pro Gruppe mit einem Pendant zu **4000 Wörtern je Student/-in**
 - Ihre Entscheidung, ob Gruppen- oder Einzelnote
 - bei Einzelnoten muss erkennbar sein, wer für welchen Teil verantwortlich ist !!!
 - bei Gruppennoten ist die Gruppe für ALLE Gruppenmitglieder selbst verantwortlich !!!
 - Projektdokumentation ist keine "klassische" Hausarbeit. Hier ein Vorschlag für Aufbau und Inhalte:
 - **Aufgabenbeschreibung**
Was ist das Ziel der Projektarbeit?
Worin bestehen die (wahrscheinlichen) Herausforderungen?
(allg. technisch und auch persönlich)
Welche Techniken/ Technologien sollen eingesetzt werden, um die Aufgabe zu lösen/ realisieren?
Warum sollen gerade diese eingesetzt werden?
Gibt es besondere Anforderungen? (technisch, Benutzer, sonstige)
Wie soll das Ziel erreicht werden (Vorgehen, Architektur)
 - **Herangehensweise**
Erfüllt das Ergebnis die (ursprünglichen) Anforderungen und Erwartungen?
Wenn nein, warum nicht?
Wo lagen besondere Probleme bei der Entwicklung.
 - **Vorstellung des Ergebnisses**
 - **Reflektion**
 - Dies ist nur ein Vorschlag, wie die Dokumentation des Projektes aussehen könnte.
Die einzelnen Punkt können, je nach Projekt unterschiedlich umfangreich sein und auch unterschiedlich aussehen.
 - In den Umfang der Projektdokumentation wird auch die Entwicklungsleistung (z.B. Teile des Sourcecodes, UML-Diagramme, etc.) eingerechnet.
Natürlich sollte die Dokumentation entsprechend umfangreich sein - 500 Wörter werden wohl eher nicht ausreichen ;-)

Themen für Gruppenarbeit:

- Level 1: Spiel "Multimodales Memory" - Ein Spieler, ein Browser
 - nicht nur Bilder "aufdecken", sondern auch Video- oder Musik-Clips
 - z.B.: in Feld A2 liegt das Filmplakat zu "Godzilla vs. Kong"
und in Feld C1 liegt wird der "Godzilla vs. Kong"-Soundtrack abgespielt.
- Level 2: Spiel "Schiffe versenken"
 - Level 2a: Zwei Spieler, ein Browser
 - Level 2b: Zwei Spieler, zwei Browser
- Level 3: 360°-Tour durch das FOM-Studienzentrum München
- Level 4: WebVR-Anwendung (freies Thema)
 - AR.js-Anwendung für die Benutzung im Browser (freies Thema)

Themenbereiche für Einzelarbeiten:

- Non-GU-Interfaces für Web-/ Mobile Web-Anwendungen
(Arten, Gestaltung und Anforderungen, Anwendungsmöglichkeiten, Technologie/ Technik)
- Google WebXR - What is it? And what for?
- Das "Metaverse" - Wohin entwickelt sich das Web?

- Cherny, B: Programmieren in TypeScript: Skalierbare JavaScript-Applikationen entwickeln
ISBN-13: 978-1976818134
Herausgeber: O'Reilly (21. November 2019)
- Takai, D.: Architektur für Websysteme: Serviceorientierte Architektur, Microservices, Domänengebundener Entwurf
ISBN-13: 978-3446450561
Herausgeber: Carl Hanser Verlag GmbH & Co. KG (11. September 2017)
- Tilkov, S.; Eigenbrodt, M.; Schreier, S.; Wolf, O.: REST und HTTP: Entwicklung und Integration nach dem Architekturstil des Web
ISBN-13: 978-3864901201
Herausgeber: dpunkt.verlag GmbH; (1. April 2015)
- Hoffmann, P.: Beyond Hypertext: Hypermedia vs. Multimedia
ISBN-13: 978-3948773182
Herausgeber : bifop-Verlag (10. Juni 2020)
- Hoffmann, P.: Beyond (Multi-) Media: Teil 1: Grundlagen
ISBN-13: 978-3948773168
Herausgeber : bifop-Verlag (7. August 2020)
- Weitere Literatur wird durch den am Standort zuständigen Dozenten bekannt gegeben.
Die Reihenfolge der Auflistung stellt keine Bewertung dar.



FOM Hochschule

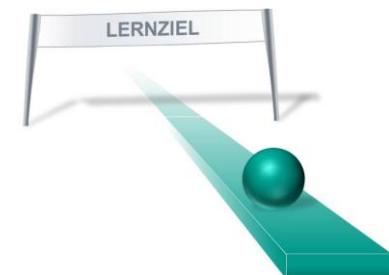
Web Technologie

(2) Einführung: Allgemeines und Historie

1	Formales, Vorstellung
2	Einführung
2.1	Allgemeine Einführung
2.2	Historie Internet, Web & Web-Technologien
3	(Web) Server – Client Kommunikation
3.1	Protokolle & Architekturen
3.2	Web- & Microservices
3.3	Authentifizierungs- und Authorisierungskonzepte
4	Grundlagen der Mediengestaltung
4.1	Media & Multimedia
4.2	Media & Modalitäten – Interaktion & Wahrnehmung
4.3	Hypermedia
5	Statische Webseiten
5.1	HTML
5.2	CSS
5.3	Medieneinbindung
6	Dynamische Webseiten
6.1	DOM, HTML & CSS
6.2	JavaScript & JSON
6.3	Weitere Konzepte & Sprachen der Webprogrammierung
7	Praxis
7.1	Vorstellung eines (ausgewählten) (JavaScript-) Frameworks
7.2	Vorstellung einer (ausgewählten) Thematik für ein eigenes Projekt
7.3	Betreute, eigenständige Entwicklung des (ausgewählten) Themas
8	Wiederholung (opt.)

Im Anschluss an diesen Themenblock sollen Sie:

- Wissen, woher die grundlegenden analogen Ideen des Internet stammen
- Wissen, woher die grundlegenden digitalem Ideen des Internet stammen
- Wissen, wie der technische Stammbaum des heutigen „Web“ im Überblick ausschaut
- Wissen, was „die Themen von morgen“ in der Web Technologie sein werden



Charakteristika des Internet und des Web

- Vernetzung von Information
→ Hypermedia
- Verlassen der klassischen
Inhaltsstruktur
→ Nichtlinearität
- Kombination unterschiedlicher
Medien/ Medienformen
→ Multimedia
- Aktivierung der „Zuschauer“
→ Interaktivität
→ Proaktivität

Charakteristika des Internet und des Web

- Vernetzung von Information
→ Hypermedia
- Verlassen der klassischen Inhaltsstruktur
→ Nichtlinearität
- Kombination unterschiedlicher Medien/ Medienformen
→ Multimedia
- Aktivierung der „Zuschauer“
→ Interaktivität
→ Proaktivität

Gemeinsamkeit

- Das Aufbrechen „klassischer“ Strukturen (inhaltlich, gestalterisch)

Ziel

- Die Immersion des Benutzers in die aufgespannte Informationswelt
→ Das Eintauchen in das Web

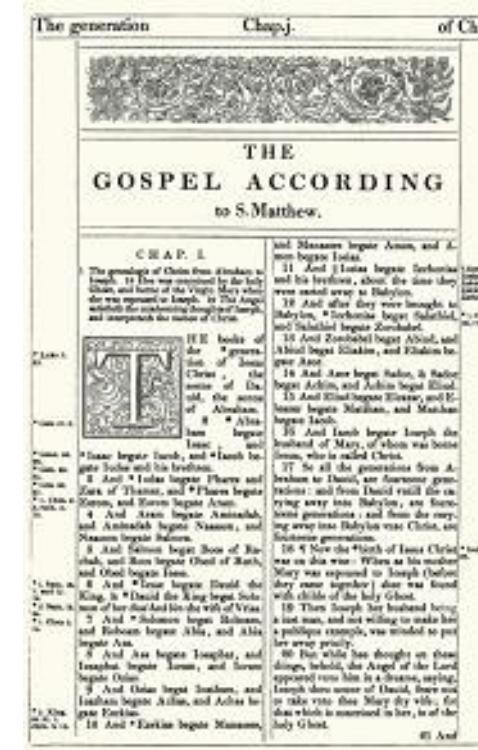
Herausforderungen

- Identifikation passender Modalitäten
- Identifikation passender Mediabilitäten
- Identifikation passender Technik/ Technologie

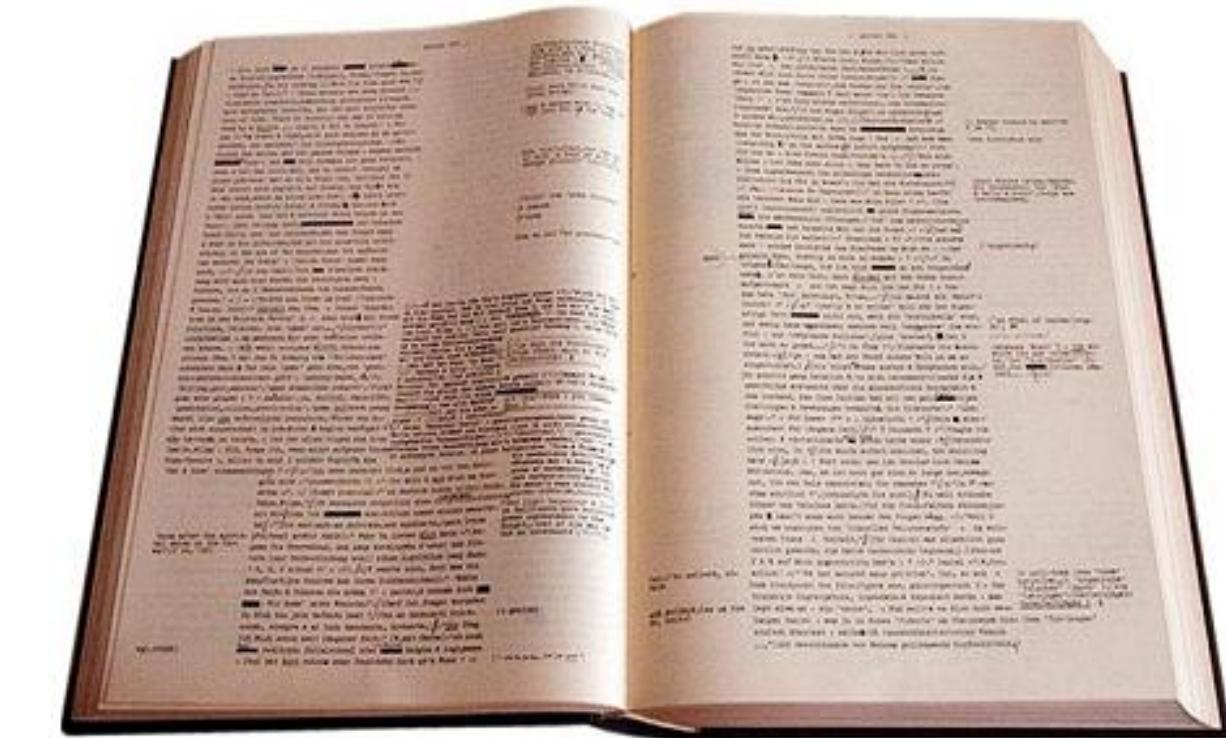
Die Idee von Hypermedia – uralt



Talmud
(mehrere 100 Jahre v. Chr.)
<http://www.judentum.org/geschichte/talmud.htm>



King James Bible
(1611)
https://www.kingjamesbibleonline.org/Mathew-Chapter-1_Original-1611-KJV/



Arno Schmitt: Zettel's Traum
(1970)
<https://www.zinzip.com/observations/2013/who-was-arno-schmidt-and-what-is-zettels-traum-some-evidentiary-fragments/>

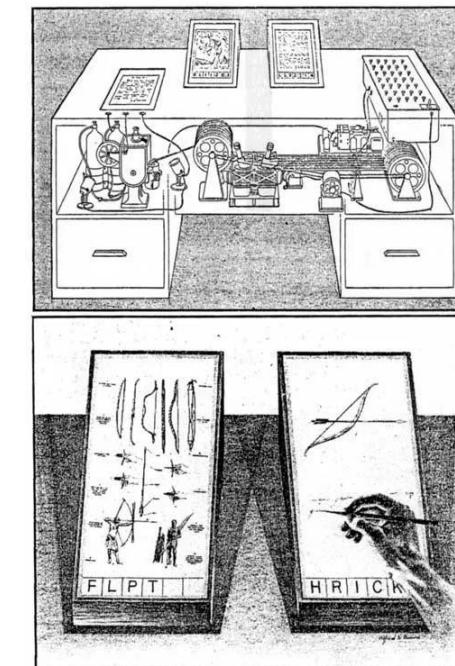
Die Idee von Hypermedia – alt

As We May Think

“Consider a future device ... in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.”

(Vannevar Bush, The Atlantic: July 1945 Issue)

<https://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/303881/>



<https://issuu.com/edavo/docs/bush-as-we-may-think>

<https://www.brainpickings.org/2012/10/11/as-we-may-think-1945/>

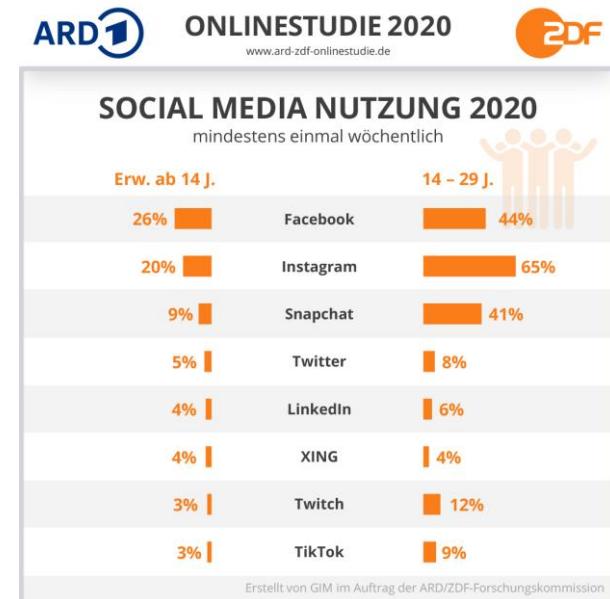
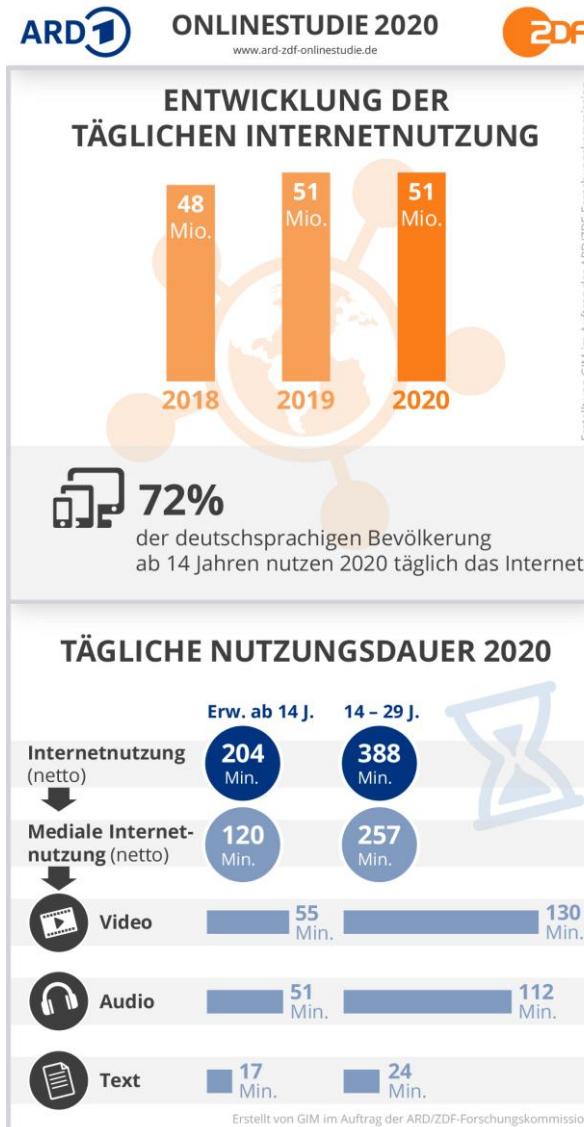
Die Frühzeit

- 1968 „Arpanet“
- 1972 erste E-Mail → (SNDMSG / READMAIL)
- 1974 „Internet“
- 1983 400 vernetzte Rechner (Arpanet)
- 1984 erste E-Mail in Deutschland
- 1989/ 90 Freigabe des Internets (kommerzielle Nutzung)

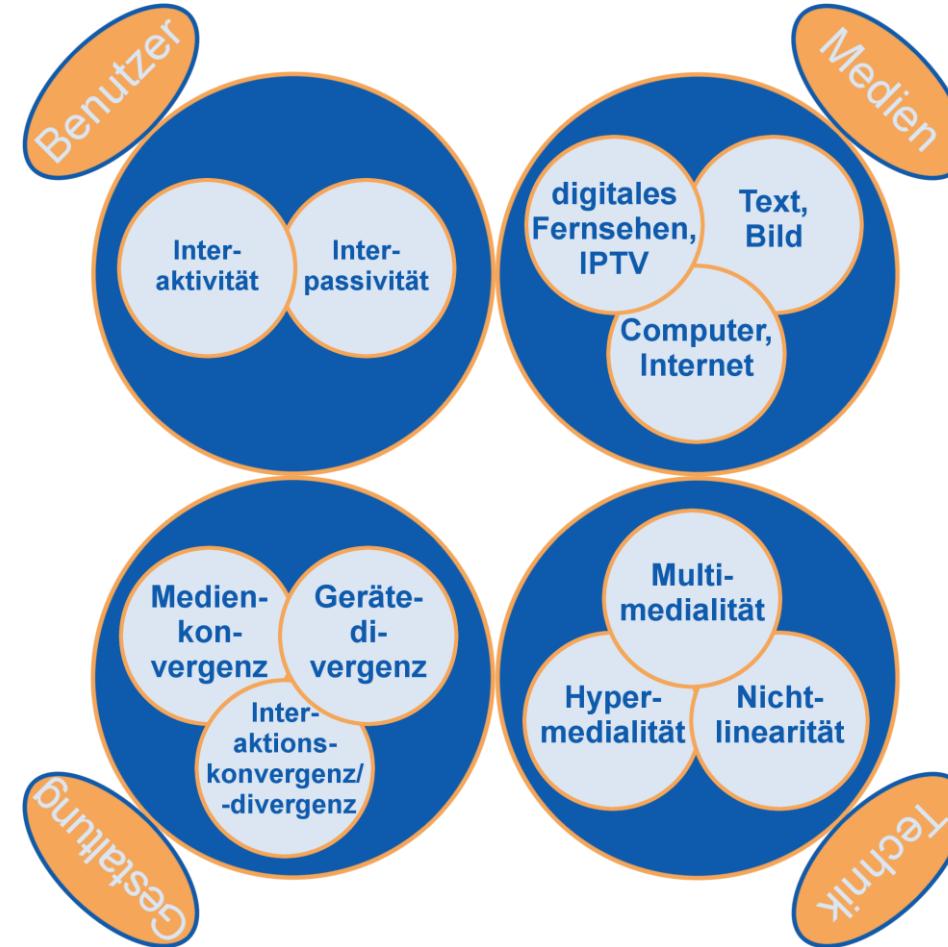


Die Zeit des WWW

1990	„Hypertext-Projekt“ → Tim Berners-Lee	1999	HTML 4.01
1991	WWW startet im CERN	2000	XHTML Platzen der „Dotcom-Blase“
1992	erste Version von „Mosaic“	2004	Beginn von „Web 2.0“
1994	erste WWW-Konferenz (Genf) HTML 2 Gründung „Netscape“ → “Navigator”	2005/ 08	Web 2.0-Boom
1995	Veröffentlichung “Internet Explorer 1.0” Gründung W3C (www.w3c.org)	2009	„Slow Media“ → Post-Web 2.0
1996	Erster Entwurf JavaScript, CSS	2010	„Social Web“
1997	Januar → HTML 3.2 Dezember → HTML 4	2011	HTML 5 (11. Arbeitsentwurf)



Hypermedia und Internet – Ein Zusammenspiel von Technologie, Medien, Gestaltung ... und Mensch



Hoffmann, Beyond Hypertext (2020)

Hypermedia und Internet – ein wenig Theorie



Hypertext Abstract Machine



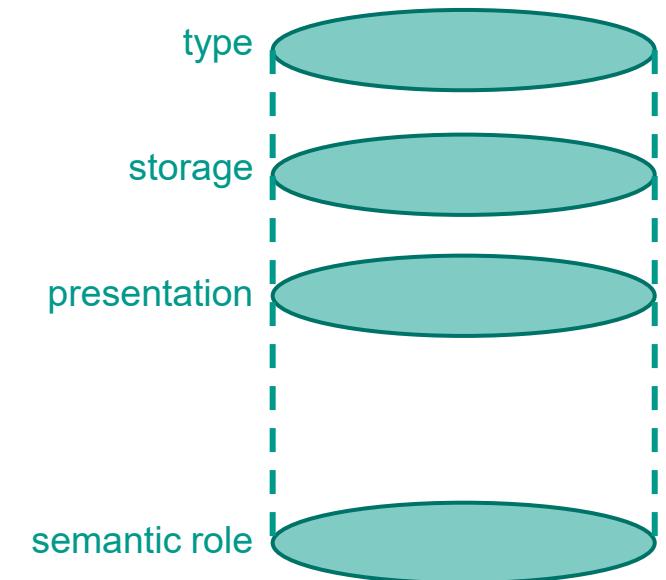
Dexter Hypertext Referencemodel

Hypermedia und Internet – ein wenig Theorie

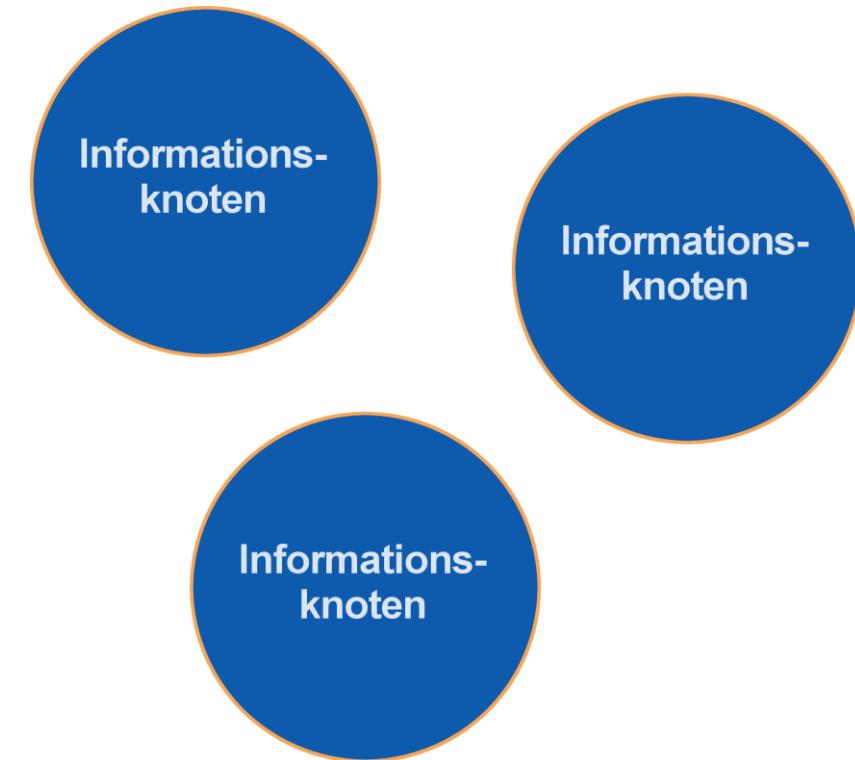
- „Tower-Modell“: De Bra, Houben, Kornatzky (1992): An extensible data model for hyperdocuments

(Proceedings 4th ACM Conference on Hypertext (ECHT'92, Milan, Italy, November 30-December 4, 1992))

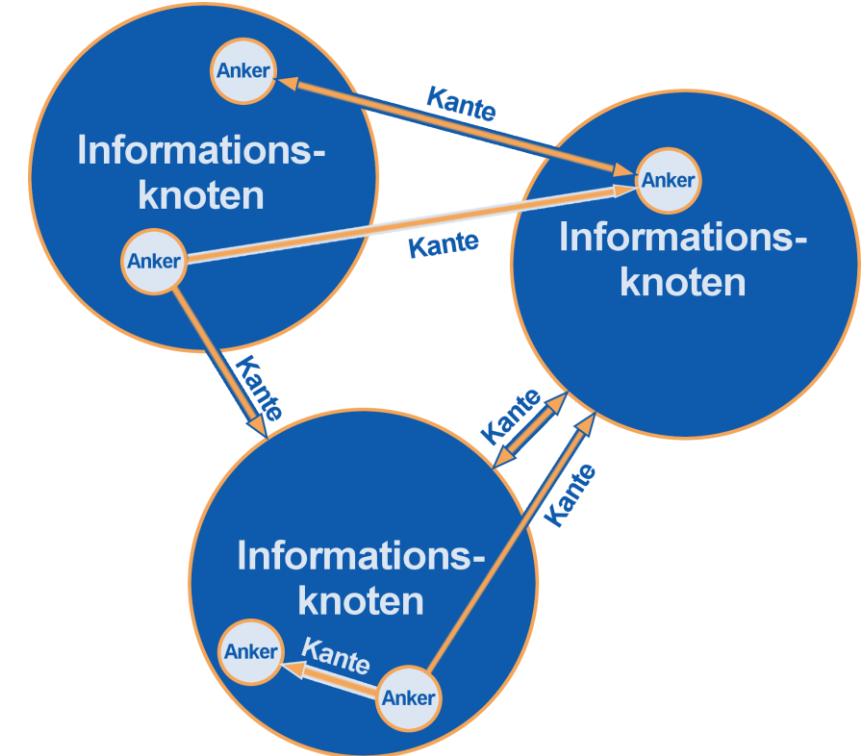
- Strukturelle Basiselemente
 - Knoten, beschrieben durch:
 - Objekt-Identifikation (oid)
 - Wert
 - Links, beschrieben durch:
 - Objekt-Identifikation (oid)
 - Quellanker
 - Zielanker
 - Wert
 - Anker , beschrieben durch:
 - Objekt-Identifikation (oid)
 - Wert und evtl. weitere Angaben (Knoten und Ortskoordinaten des Ankers, o.a.)
- Tower-Objekte, City-Objekte



Hypermedia und Internet – ein wenig Theorie



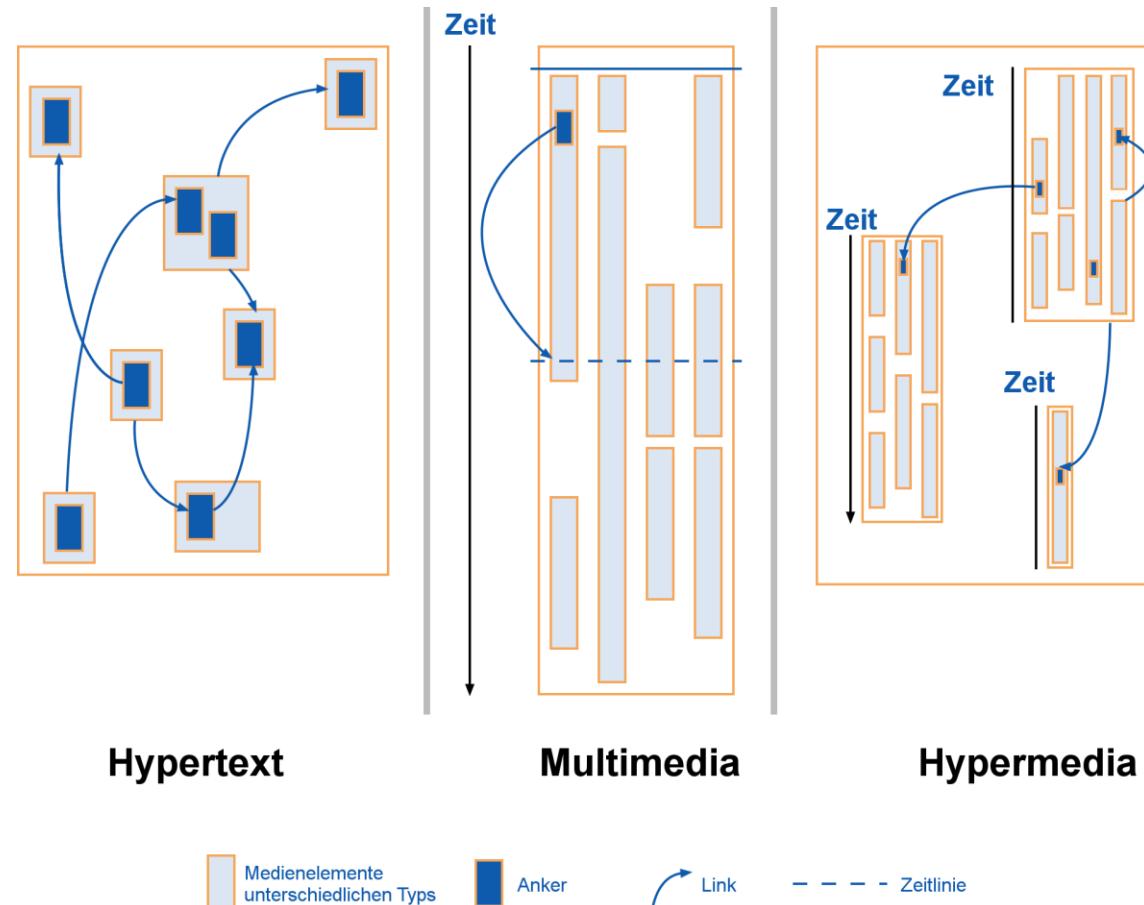
Ausgangslage:
Informationsknoten als
Mediensammlung(en)



Erweiterte Konstruktion
von Hypermedia:
Gerichtete, multiple Kanten

Hypermedia und Internet – ein wenig Theorie

- Hardman, Bulterman (1994): The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model
(Communications of the ACM 37(2):50-62)



Hoffmann, Beyond Hypertext (2020)

Die Idee von Hypermedia und Internet

- Der Versuch einer allgemeinen Definition von Hypermedia:

Hypermedia ist der Oberbegriff für eine Menge an Informationen zu einem Thema, die in verschiedenen Medientypen vorliegen können und untereinander auf verschiedene Weisen vernetzt sind.

- Der Versuch einer ausführlichen Definition von Hypermedia:

Ein Hypermedium ist eine Sammlung von Informationsknoten zu einem Thema.

Jeder Informationsknoten kann aus einer oder mehreren Informationen bestehen, die in verschiedenen Medienformen vorliegen können. Diese Informationsknoten sind untereinander durch Kanten vernetzt. Kanten beginnen und enden mit Ankern innerhalb desselben oder eines anderen Informationsknotens. Sie können gerichtet von einem Anker auf einen anderen Anker deuten oder auch ungerichtet sein.

Anker können an Ausschnitte innerhalb eines Informationsknotens gebunden sein oder an den ganzen Informationsknoten.

Ein Hypermedium, dessen Informationen lediglich in einer Medienform vorliegen oder sich in seiner Repräsentation überwiegend an einer Medienform orientiert, ist ein Spezialfall. Die bedeutendste und bekannteste Spezialisierung ist Hypertext.

(Hoffmann: Beyond Hypertext (2020))

FOM Hochschule

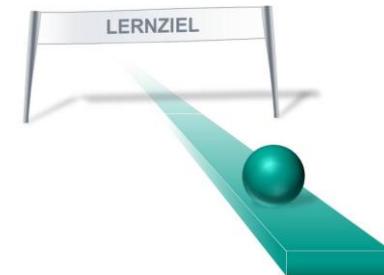
Web Technologie

(3) (Web) Server – Client Kommunikation

1	Formales, Vorstellung
2	Einführung
2.1	Allgemeine Einführung
2.2	Historie Internet, Web & Web-Technologien
3	(Web) Server – Client Kommunikation
3.1	Protokolle & Architekturen
3.2	Web- & Microservices
3.3	Authentifizierungs- und Authorisierungskonzepte
4	Grundlagen der Mediengestaltung
4.1	Media & Multimedia
4.2	Media & Modalitäten – Interaktion & Wahrnehmung
4.3	Hypermedia
5	Statische Webseiten
5.1	HTML
5.2	CSS
5.3	Medieneinbindung
6	Dynamische Webseiten
6.1	DOM, HTML & CSS
6.2	JavaScript & JSON
6.3	Weitere Konzepte & Sprachen der Webprogrammierung
7	Praxis
7.1	Vorstellung eines (ausgewählten) (JavaScript-) Frameworks
7.2	Vorstellung einer (ausgewählten) Thematik für ein eigenes Projekt
7.3	Betreute, eigenständige Entwicklung des (ausgewählten) Themas
8	Wiederholung (opt.)

Im Anschluss an diesen Themenblock sollen Sie:

- Einen Überblick über die Web-Architekturen gewonnen haben
- Die aktuellen Protokolle im Umfeld des Web kennen und voneinander abgrenzen können
- Die sichere Kommunikation in Hinsicht auf Authentifizierung und Authentisierung beschreiben können



Grundlagen

Allgemeine Grundbegriffe

- **Benutzer:** Person, die „mit dem Web“ interagiert
- **Client:** Programm, dass vom Rechner des Benutzers auf entfernte Web-Server zugreift
 - **Browser:** spezielle Art von Client für die Nutzung im Web
 - greift auf Ressourcen zu, die auf Web-Servern gespeichert sind
 - User Agent: Browser greift stellvertretend als Agent des Benutzers
- **Server:** Programm/ Prozess auf einem Rechner, der Funktionalitäten bereitstellt, um Anfragen von Clients zu be-/ verarbeiten
 - **Web-Server:** Spezielle Art von Client für die Nutzung im Web
 - stellt Ressourcen bereit, auf die Web-Clients zugreifen können

Grundlagen

DIE allgemeinen Grundbegriffe

- Internet
 - Gesamtheit aller vernetzten Computer, die auf Basis von Internet-Protokollen kommunizieren
→ **Internet ≠ WWW**
- World Wide Web (WWW)
 - verteiltes Hypermedia-System, das einige Services des Internet nutzt
 - Benennungsdienst DNS
 - verbindungsorientierte Übertragung nach TCP

Grundlagen

Domain Name System (DNS)

- **RFC 1034, RFC 1035:**
→ globaler „Naming Service“ zur Abbildung von DNS-Namen auf IP-Adressen
- Hierarchische Anordnung in Domains
- **Top-Level Domains (TLD)** sind Domains der obersten Ebene dieser Hierarchie
 - Country-Code TLD (ccTLD)
bestehend aus 2 Buchstaben nach ISO 3166
 - Generic TLD (gTLD)
com, net, edu, ...

Grundlagen

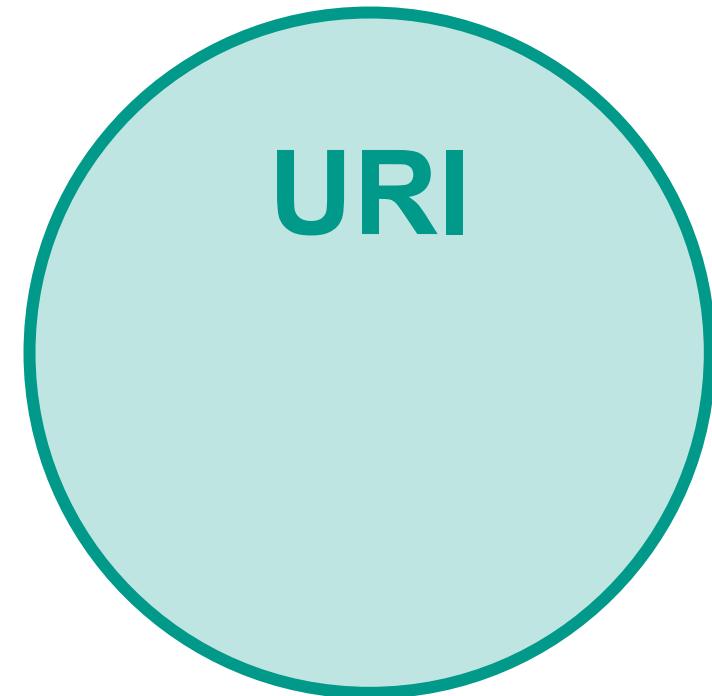
Technische Grundbegriffe

- Uniform Resource Locator
 - [URL](#)
 - Schema zur Benennung von Ressourcen
- HyperText Transfer Protocol
 - [HTTP](#)
 - Übermittlungsprotokoll für Ressourcen
- HyperText Markup Language
 - [HTML](#)
 - Dokumentformat für Hypertext
- eXtensible Markup Language
 - [XML](#)
 - Plattform für benutzerspezifische Dokumentformate

Grundlagen: Technische Grundbegriffe

Uniform Resource Identifier (URI)

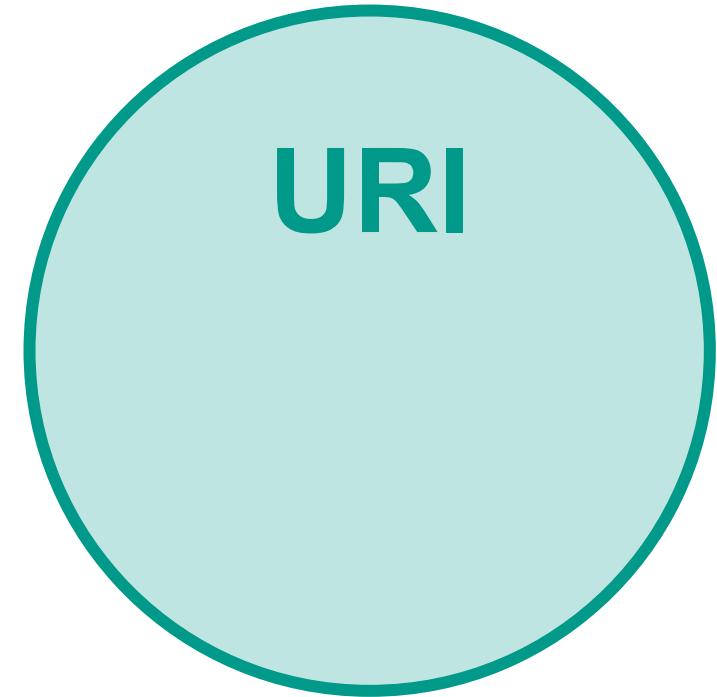
- „Zeichenfolge, die zur Identifizierung einer abstrakten oder physischen Ressource dient“
- Bezeichnung von Ressourcen (vor allem im WWW)
 - Webseiten
 - sonstige Dateien
 - Aufruf von Webservices
 - E-Mail-Empfängern
 - etc.
- Aktuell: RFC 3986 (2016)
- als Zeichenfolge kann jede URI in jedes digitale Dokument aufgenommen werden
→ HTML-Link
- Erweiterung: Internationalized Resource Identifiers (IRI)
→ nur aus druckbaren ASCII-Zeichen bestehenden URIs



Grundlagen: Technische Grundbegriffe

Aufbau eines Uniform Resource Identifier (URI)

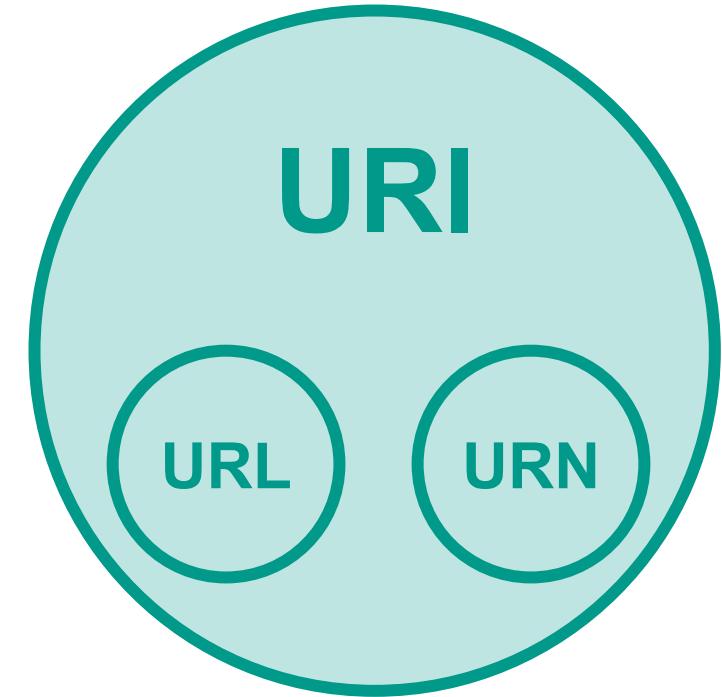
- Syntax
 - URI = Schema : schemaspezifischer Teil
- Schema einer URI
 - bestimmt das Namensschema der URI
 - ftp, http, ...
- schemaspezifischer Teil einer URI
 - enthält die Information über ein bestimmtes Objekt
 - Form vom Schema abhängig



Grundlagen: Technische Grundbegriffe

Name & Location → URN & URL

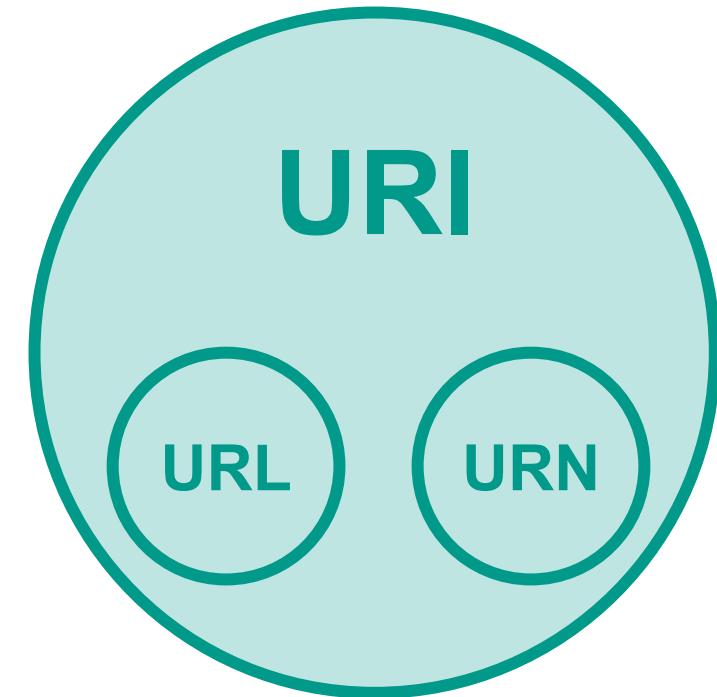
- Ressource
 - alles, was sich eindeutig beschreiben lässt
- Uniform Ressource Name (URN)
 - dauerhaft gültiger Name für eine Ressource
 - dauerhaft gültige eindeutige Identifizierbarkeit
- Uniform Ressource Locator (URL)
 - (dauerhaft) gültige Beschreibung des Ortes, an dem sich eine Ressource befindet
- Vereinfacht:
 - URN → beschreibt, was für eine Art von Ressource etwas ist
 - URL → beschreibt, wo die Ressource ist und wie man dahin kommt
- URL und URN sind URIs
- URI ist entweder URL oder URN



Grundlagen: Technische Grundbegriffe

Aufbau eines Uniform Resource Locator (URL)

- Syntax
 - URL = Schema : schemaspezifischer Teil
- Schema einer URL
 - ftp: File Transfer Protocol (FTP)
 - https: HTTP über Secure Socket Layer (SSL)
 - mailto: E-Mail Adressen
 - telnet: interaktive Dienste über das Telnet-Protokoll
 - ...
- schemaspezifischer Teil einer URL **“//”[User [“:”Password] “@”Host [“:”Port] “/”Path**
 - User: Benutzername (optional, nicht in allen Schemata)
 - Password: Kennwort des Benutzers (optional)
 - Host: Name des Zielrechners (DNS-Name, IP-Adresse)
 - Port: Port des Hosts (optional, Standard: Port 80 bei HTTP)
 - Path: „Rest der URL“ (z.B. Verzeichnis beim Host)



Übung: URI, URN oder URL?

Sind folgende Angaben URI, URN oder URL?

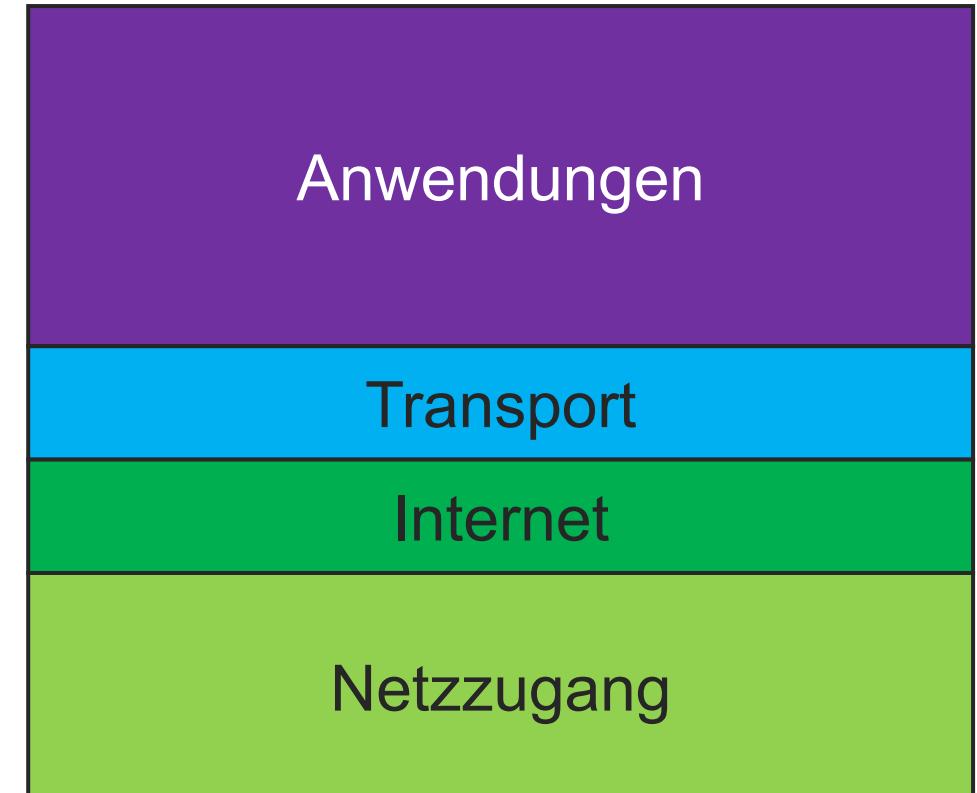
- t3n.de
- http://t3n.de
- ftp://t3n.de
- urn:isbn:9783948773168

Allgemeine Grundbegriffe

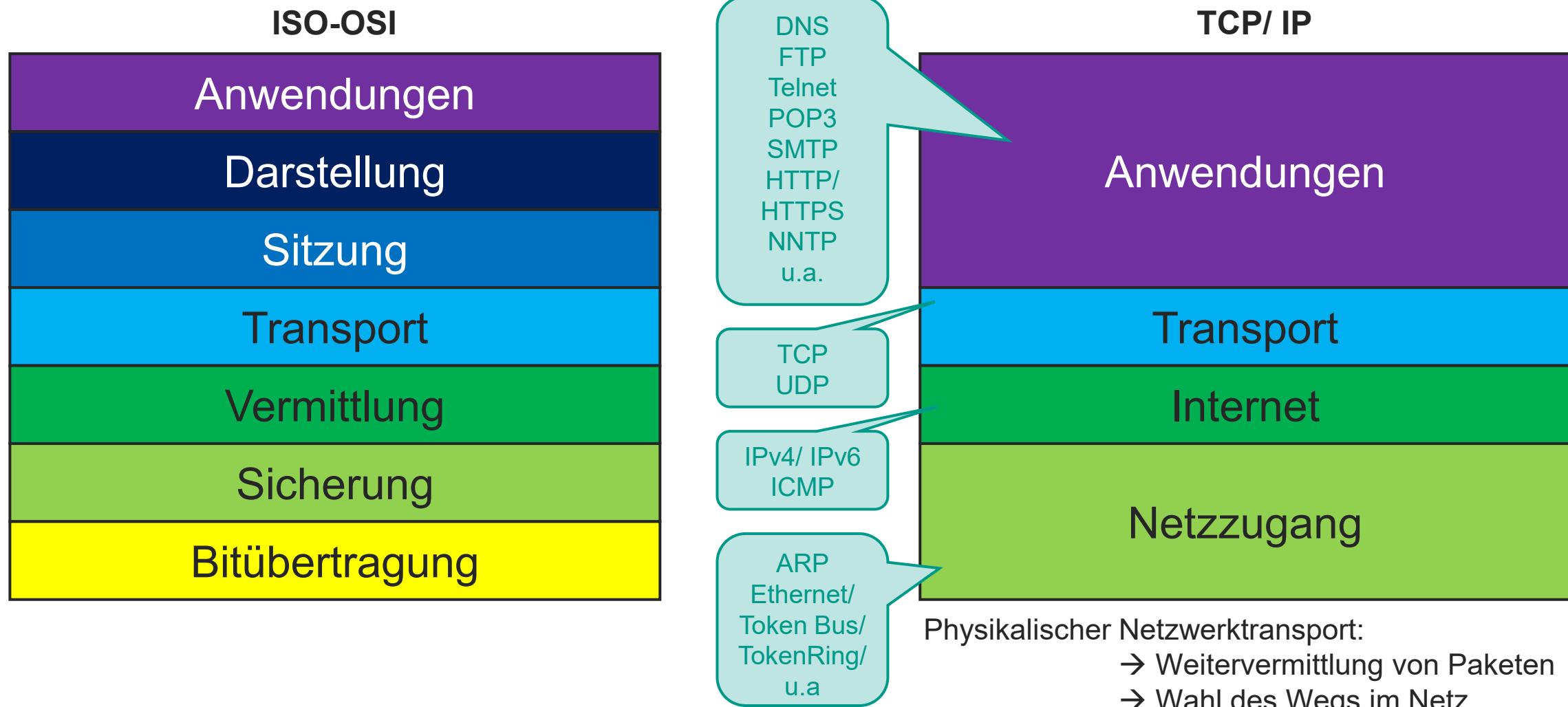
ISO-OSI



TCP/ IP

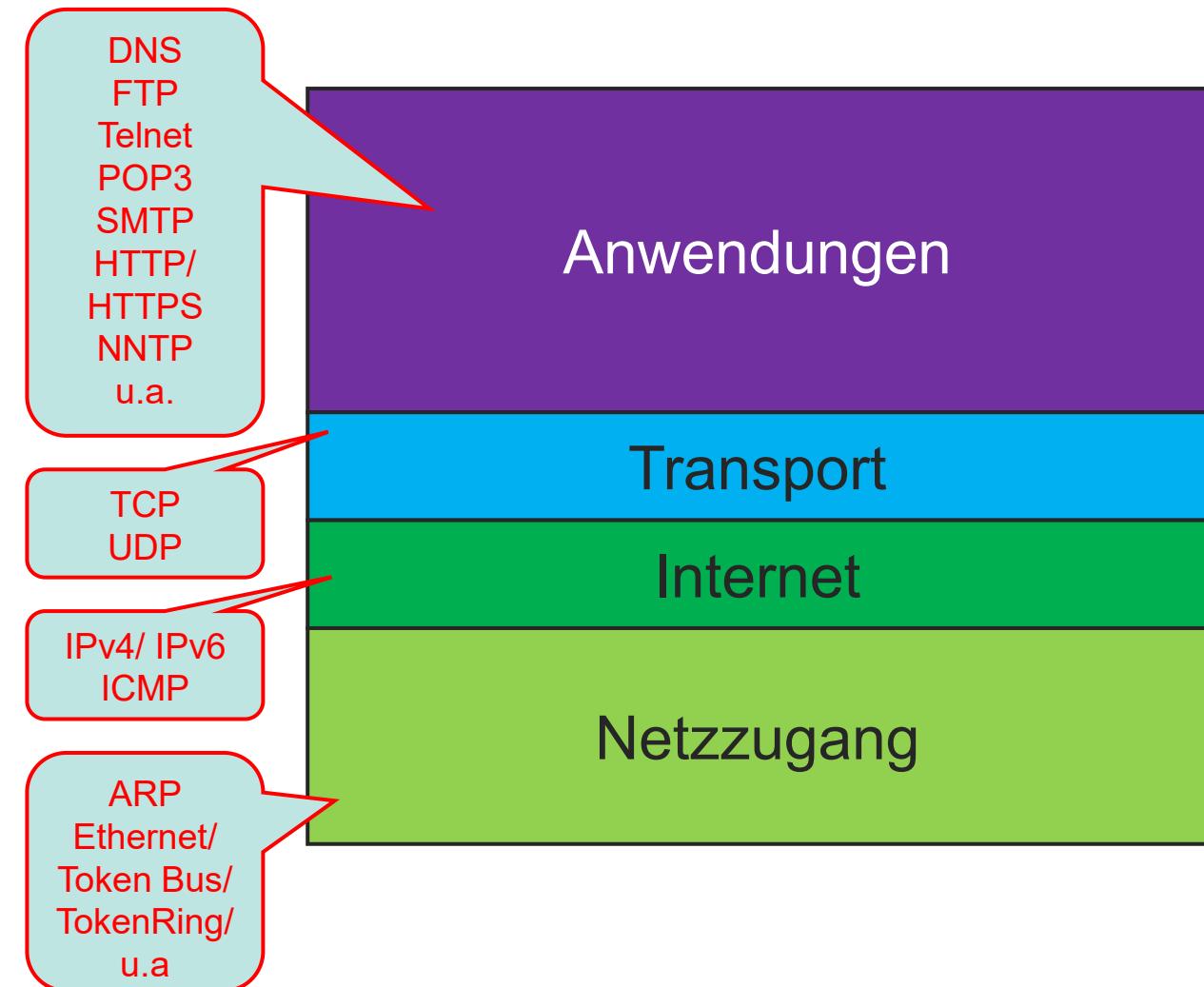


Allgemeine Grundbegriffe



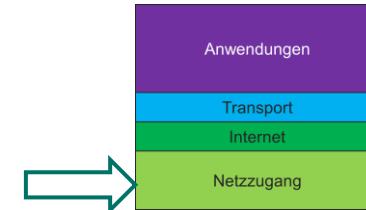
Übung: Protokolle

Wählen Sie aus der Vielzahl der Protokolle eines aus und stellen Sie dieses in der kommenden Veranstaltung kurz vor.



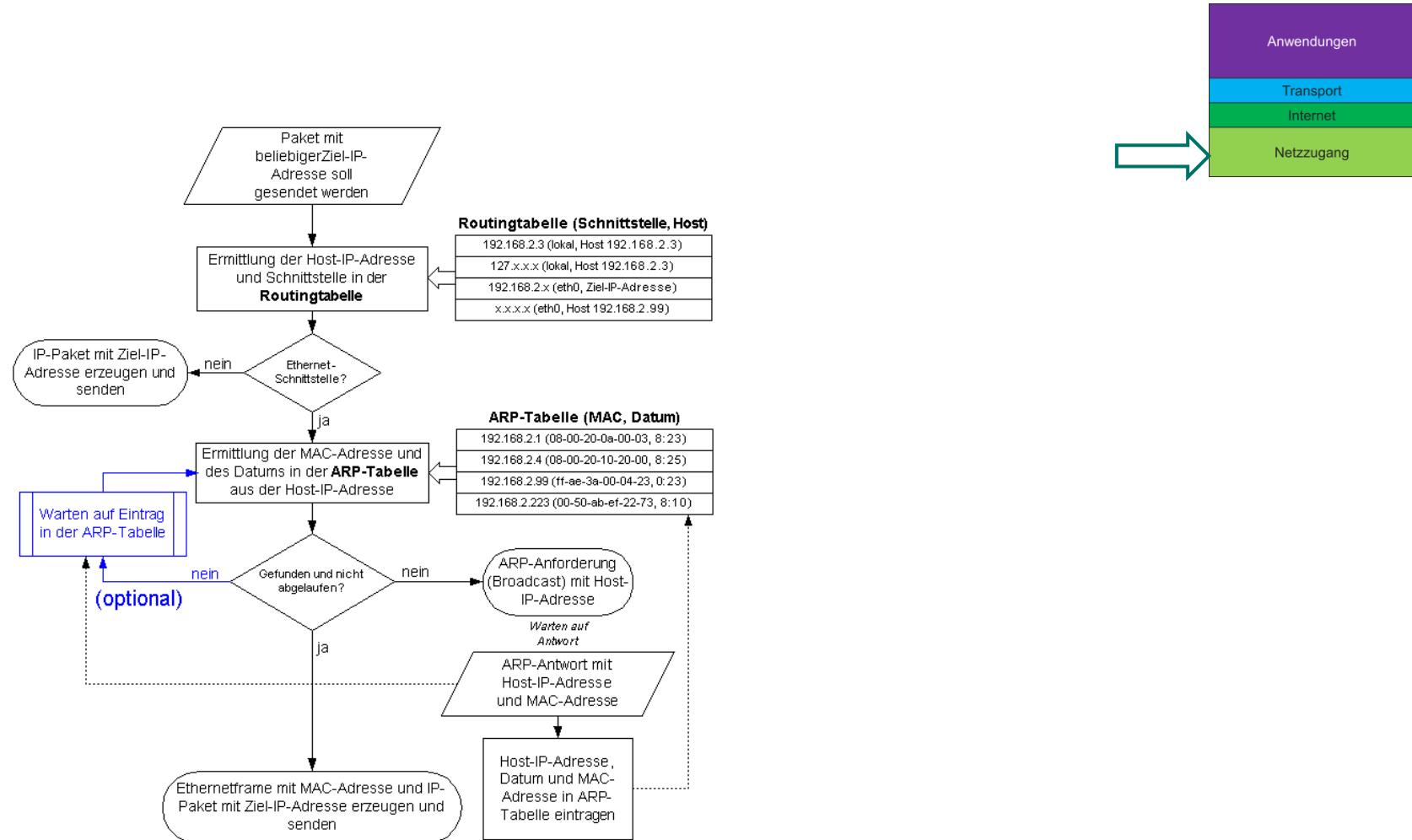
Grundlagen: Protokolle

Adress Resolution Protocol (ARP)



- sucht zu einer Netzwerkadresse (der Internetschicht) die physische Adresse (Hardware-/ MAC-Adresse)
- Hauptsächliche Nutzung in Ethernet-Netzen und bei IPv4
→ bei IPv6: Neighbor Discovery Protocol (NDP)
 - der Host sendet einen ARP-Request mit der MAC-Adresse **FF-FF-FF-FF-FF-FF**
→ MAC-Broadcast an alle Systeme im Netzwerk
 - wird von jedem Netzwerk-Interface entgegengenommen und ausgewertet
 - Host passender IP-Adresse schickt als ARP-Reply seine MAC- und die IP-Adresse an den Sender zurück
 - die gemeldete MAC-Adresse wird im lokalen ARP-Cache des Senders gespeichert
→ schnellere ARP-Adressauflösung
- Wenn die MAC-Adresse bekannt und die IP-Adresse gesucht ist,
kann RARP als Variante von ARP genutzt werden

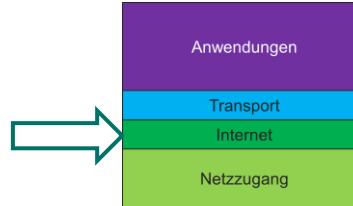
Adress Resolution Protocol (ARP)



Wikipedia: Adress Resolution Protocol

Grundlagen: Protokolle

Internet Control Message Protocol (ICMP)



- Austausch von Informations- und Fehlermeldungen über das Internet-Protokoll
→ Bestandteil von IPv4
→ für IPv6: ICMPv6
- Transport Diagnose-, Kontroll-, Routingdatenpakete
- Typ des ICMP-Pakets steht als 8-Bit-Zahl am Anfang des ICMP-Headers
 - 0 Echo (Antwort)
 - 3 Ziel des Datagramms nicht erreichbar
 - 5 Umleitungsempfehlung zu anderem Gateway desselben Netzwerks mit schnellerer Verbindung
 - ...
 - 11 Zeitüberschreitung
 - 13 Zeitstempel (erleichtert die Zeitsynchronisation)
 - 14 Zeitstempel (Antwort)
 - ...

A screenshot of a Microsoft Windows Command Prompt window titled "Eingabeaufforderung". The window shows the output of a ping command to the IP address 87.190.244.22. The output includes several ICMP echo replies (Echo Reply messages) with details like bytes (32), time (32ms), and TTL (55). Below this, a summary of the ping statistics is provided, stating 4 packets sent, 4 received, 0 lost (0% loss), and round-trip times ranging from 32ms to 33ms with a mean of 32ms.

```
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Programmierung>ping www.fom.de

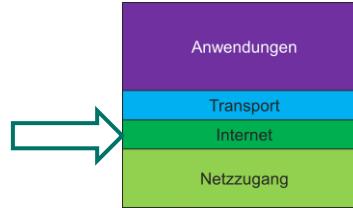
Ping wird ausgeführt für www.fom.de [87.190.244.22] mit 32 Bytes Daten:
Antwort von 87.190.244.22: Bytes=32 Zeit=32ms TTL=55
Antwort von 87.190.244.22: Bytes=32 Zeit=32ms TTL=55
Antwort von 87.190.244.22: Bytes=32 Zeit=33ms TTL=55
Antwort von 87.190.244.22: Bytes=32 Zeit=32ms TTL=55

Ping-Statistik für 87.190.244.22:
Pakete: Gesendet = 4, Empfangen = 4, Verloren = 0
(0% Verlust),
Ca. Zeitangaben in Millisek.:
Minimum = 32ms, Maximum = 33ms, Mittelwert = 32ms

C:\Users\Programmierung>
```

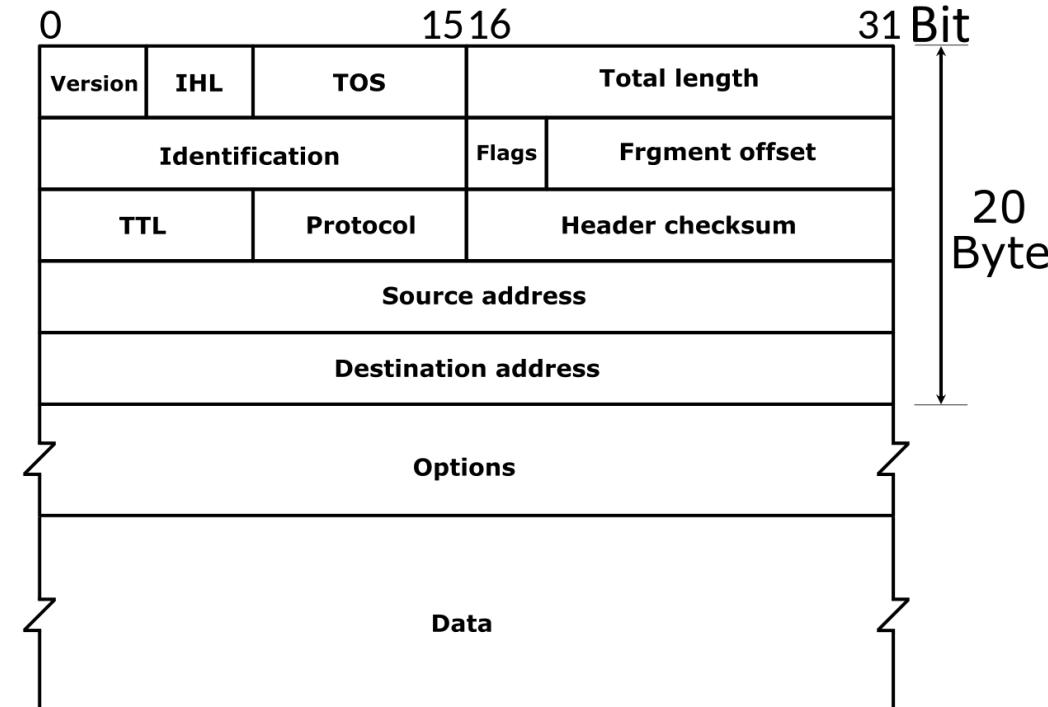
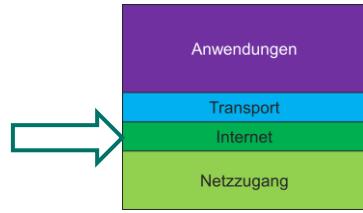
Grundlagen: Protokolle

Internet Protocol



- erste vom Übertragungsmedium unabhängige Schicht der Internetprotokollfamilie
→ mittels IP-Adresse und Subnetzmaske (subnet mask, IPv4) bzw. Präfixlänge (IPv6) werden Geräte innerhalb eines Netzwerkes in logische Einheiten gruppiert
- Ziel: Computer adressieren und ihnen IP-Pakete senden zu können
- Öffentliche IP-Adressen müssen i.d.R. weltweit eindeutig zugeordnet werden können
→ Vergabe durch die Internet Assigned Numbers Authority (IANA) geregelt
- Charakteristika/ Ziele/ Möglichkeiten:
 - IP-Adressierung jeder Netzwerknoten besitzt eine eindeutige Adresse (IP-Adresse)
 - IP-Fragmentierung Datenpakete können bei Bedarf in kleinere Einheiten zerlegt werden
 - IP-Broadcast Unicast: Datenpakete werden an einen bestimmten Host gesandt
 Multicast: Datenpakete können an mehrere Hosts gesandt werden
 - IP-Routing gezieltes Leiten des Datenstroms
 - keine Fehlerkorrektur ausschließlich Transport der Daten

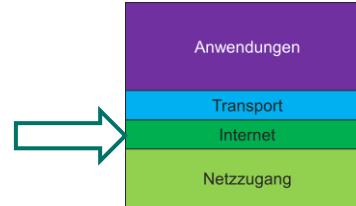
Internet Protocol



Wikipedia: Internet Protocol

Grundlagen: Protokolle

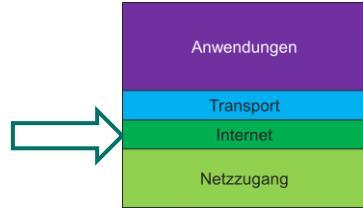
Internet Protocol IPv4



- aufgeteilt in 4 Blöcke (Oktette)
- Wertebereich jedes Blocks: 0 bis 255
- 32 Bit-Adressen ($4 * 1$ Byte)
 - max. 4.294.967.296 eindeutige Adressen
- Netzwerk-Klassen:
 - Class A: Netze 0.0.0.0/8 bis 127.255.255.255
 - Class B: Netze 128.0.0.0/16 bis 191.255.255.255
 - Class C: Netze 192.0.0.0/24 bis 223.255.255.255
 - Class D: Multicast-Gruppen 224.0.0.0/4 bis 239.255.255.255
 - Class E: Reserviert 240.0.0.0/4 bis 255.255.255.255
- 14. Dezember 2012: letzte Zuteilung freier IPv4 Adressen

Grundlagen: Protokolle

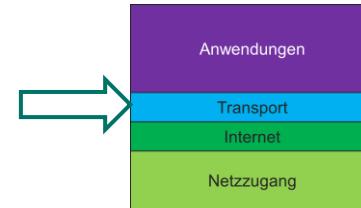
Internet Protocol IPv6



- Beginn der Entwicklung 1995
- Seit 1998 offiziell Nachfolger von IPv4
- Vergrößerung des adressierbaren Raums auf 2^{128} (≈ 340 Sextillionen = $3,4 \cdot 10^{38}$) Adressen
- Unterschiede im Aufbau: **2001:0db8:85a3:08d3:1319:8a2e:0370:7344**
 - Trennung durch „：“ anstelle „.“
 - Hexadezimale anstelle **dezimaler** Darstellung
 - **8 Blöcke zu je 16 Bit** anstelle von **4 Blöcken zu je 8 Bit**
 - „**0**“ oder Blöcke mit „**0000**“ können weggelassen werden
→ ebenfalls eine valide IPv6 Adresse: **2001:db8::1428:57ab**

Grundlagen: Protokolle

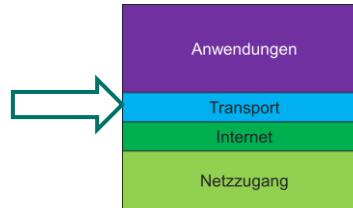
Transmission Control Protocol (TCP)



- „Übertragungssteuerungsprotokoll“
- definiert, wie Daten zwischen Netzwerkkomponenten ausgetauscht werden sollen
 - verbindungsorientiert
 - paketvermittelt
- Ende-zu-Ende-Verbindung im Vollduplexbetrieb
→ jede Verbindung wird eindeutig durch zwei Endpunkte definiert/ identifiziert
 - Verbindung 1: (Lokaler Rechner, Port x, Entfernter Rechner, Port y)**
 - Verbindung 2: (Lokaler Rechner, Port x, Entfernter Rechner, Port z)**
- Portnummern sind 16-Bit-Zahlen → 0 bis 65535
 - Portnummern 0 bis 1023 sind reserviert und werden von der IANA vergeben
 - Port 21 reserviert für FTP
 - Port 80 reserviert für HTTP im WWW
 - Die Nutzung der vordefinierten Ports ist nicht zwingend
→ jeder Administrator kann die Ports anders belegen/ nutzen

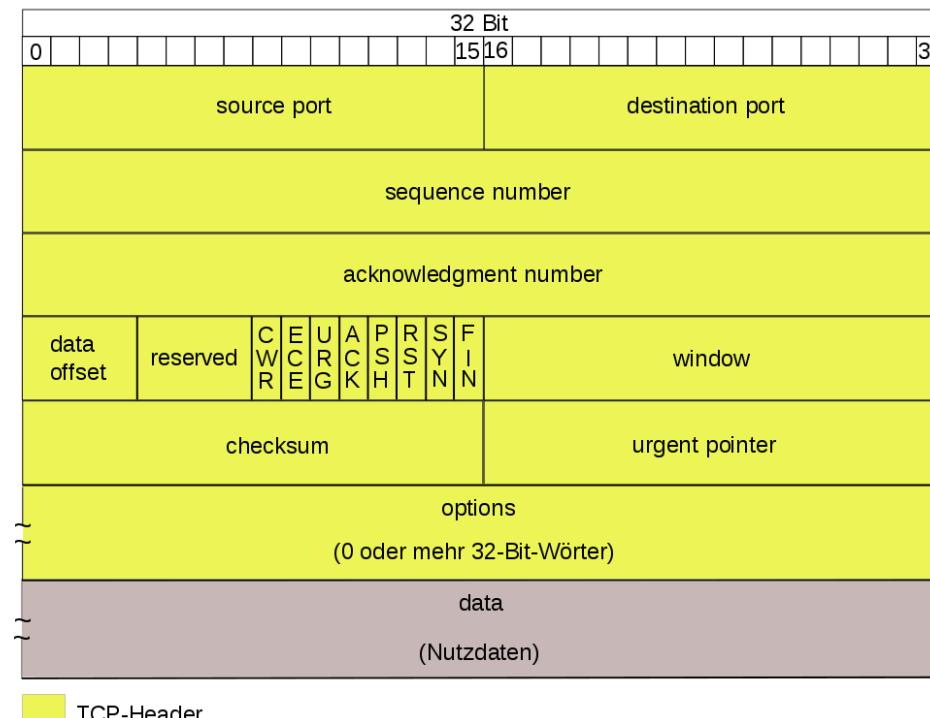
Grundlagen: Protokolle

Transmission Control Protocol (TCP)



- Aufbau TCP-Header
 - Sequence Number
 - Acknowledge Number

Sortierung der Pakete
 Mitteilung, bis zum wievielten Datenpaket alle
 Daten erfolgreich eingetroffen sind

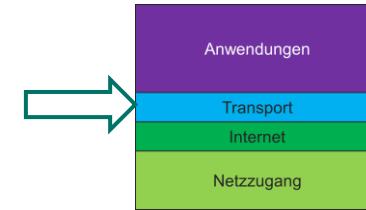
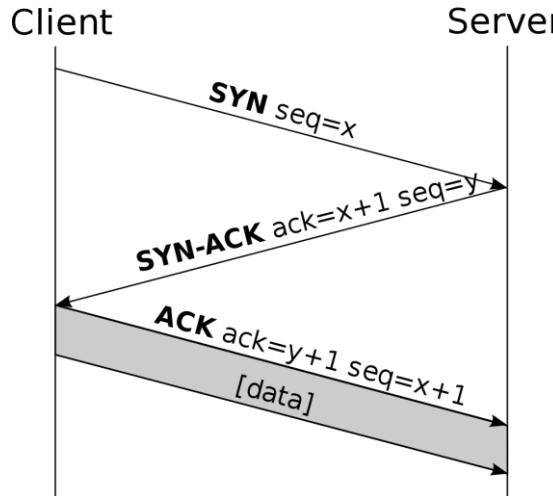


[Wikipedia: Transmission Control Protocol](#)

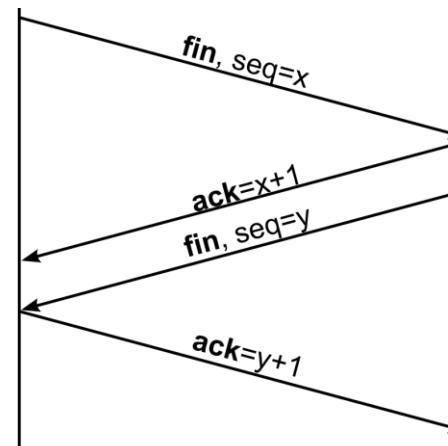
Grundlagen: Protokolle

Transmission Control Protocol (TCP)

- Verbindungsauftbau: TCP-Handshake



- Verbindungsabbau: TCP-Teardown

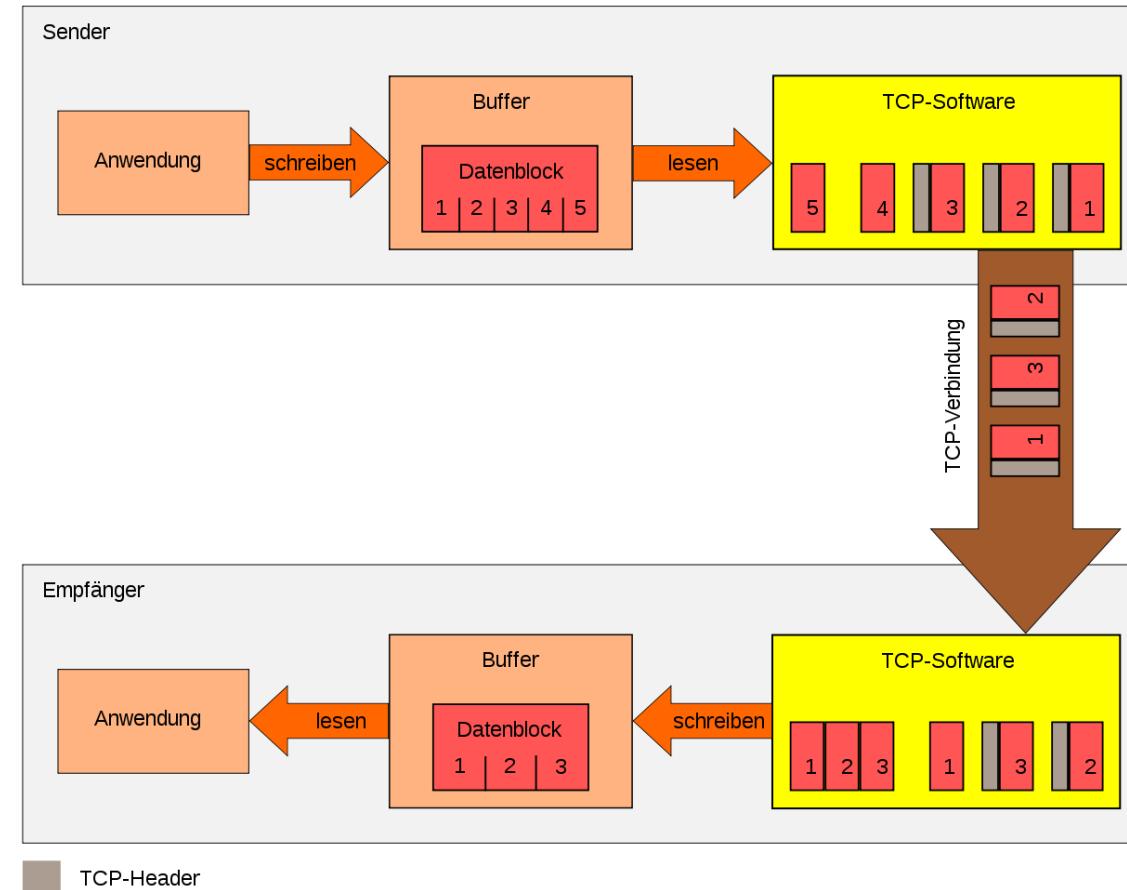


Wikipedia: Transmission Control Protocol

Grundlagen: Protokolle

Transmission Control Protocol (TCP)

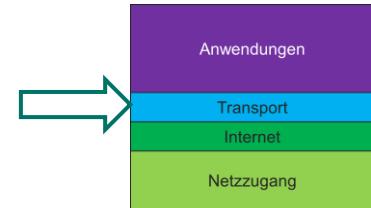
- Datenübertragung:



Wikipedia: Transmission Control Protocol

Grundlagen: Protokolle

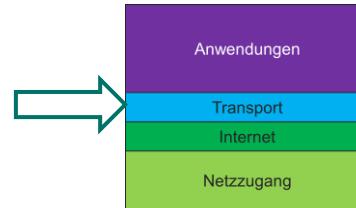
User Datagram Protocol (UDP)



- verbindungsloses, nicht-zuverlässiges, ungesichertes und ungeschütztes Übertragungsprotokoll
- keine Garantie, dass ...
 - ... ein gesendetes Paket ankommt
 - ... Pakete in der gleichen Reihenfolge beim Empfänger ankommen, in der sie gesendet wurden
 - ... ein Paket nur einmal beim Empfänger eintrifft
 - ... dass die Daten unverfälscht oder unzugänglich für Dritte beim Empfänger eintreffen
- UDP-Anwendung müssen ...
 - ... gegenüber verlorengegangenen und unsortierten Paketen unempfindlich sein oder ...
 - ... selbst entsprechende Korrektur-/ Sicherungsmaßnahmen implementieren
- kein Verbindungsaufbau vor Übertragungsbeginn
→ schneller Start des Datenaustausches
→ ideal für Anwendungen, die nur kleine Datenmengen austauschen (z.B. DNS)

Grundlagen: Protokolle

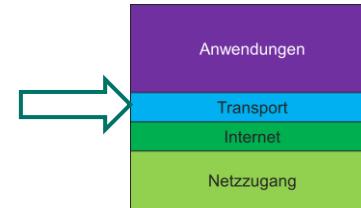
User Datagram Protocol (UDP)



- „Datagram“
 - in sich geschlossene, unabhängige Dateneinheit
 - wird ohne weitere Verbindungssicherung zwischen zwei Endpunkten verschickt
→ Peer-to-Peer
- besondere Variante: Lightweight User Datagram Protocol (UDP-Lite)
 - RFC 3828
 - Übertragung von Daten, bei denen geringe Verzögerung wichtig ist
 - kleinere Fehler werden dabei toleriert
 - Beispiel: Liveaudio-/ -videoübertragungen

Grundlagen: Protokolle

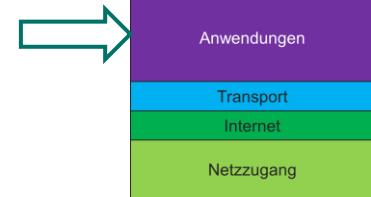
Weitere Protokolle auf der Transportebene



- **RIP** Routing Information Protocol
→ automatische Erstellung der Routingtabellen von Routern
- **TFTP** Trivial File Transfer Protocol
→ ausschließliche Unterstützung des Lesens und Schreibens von Dateien
→ keine Rechtevergabe, keine Anzeigen vorhandener Dateien, keine Benutzeroauthentifizierung, etc.
- **SNMP** Simple Network Management Protocol
→ koordiniert die Kommunikation zwischen überwachten Netzwerkelementen wie Routern, Servern, Switches, Druckern, Computern, etc.
- **SIP** Session Initiation Protocol
→ Aufbau, Steuerung und Abbau von Kommunikationssitzungen zwischen zwei und mehr Teilnehmern

Grundlagen: Protokolle

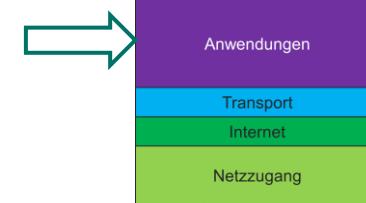
File Transfer Protocol (FTP)



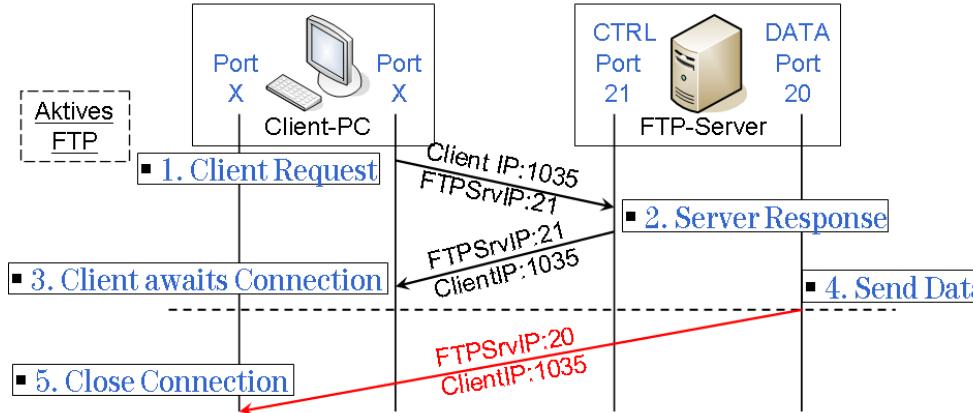
- „Dateiübertragungsprotokoll“
→ zustandsbehaftetes Netzwerkprotokoll zur Übertragung von Dateien über IP-Netzwerke
- Anwendung:
 - „Hochladen“ von Dateien vom Client zum Server
 - „Runterladen“ von Dateien vom Server zum Client
 - Clientseitige Steuerung der Datenübertragung zwischen zwei Servern
 - Anlegen, Auslesen, Umbenennen und Löschen von Dateien und Verzeichnissen
- Separate Verbindungen für die Steuerung der Datenübertragung und für die eigentliche Datenübertragung
- Beginn einer FTP-Sitzung:
 - Aufbau einer TCP-Verbindung vom Client zum Control Port des Servers (i.d.R. Port 21)
 - Kanal für die Befehle zum Server
 - diese werden mit einer Statusnachricht beantwortet, an die oft ein erklärender Text angehängt ist
 - Befehle sind i.d.R. der erst nach erfolgreicher Authentifizierung zulässig

Grundlagen: Protokolle

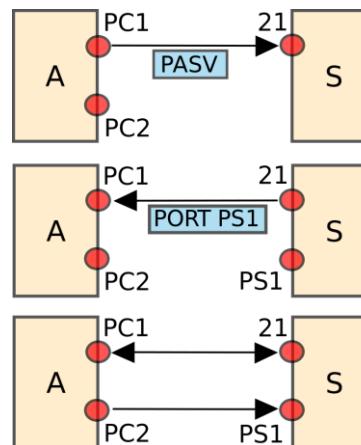
File Transfer Protocol (FTP)



- Modus: Aktives FTP



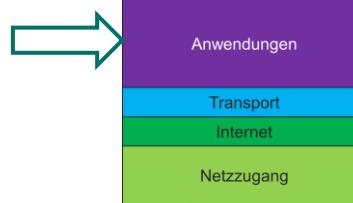
- Modus: Passives FTP



Wikipedia: File Transfer Protocol

Grundlagen: Protokolle

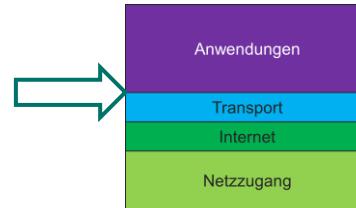
File Transfer Protocol (FTP)



- Webbrowser hatten zum Teil einen eigenen FTP-Client integriert
Standardisierung der Syntax der FTP-Adressierung seit RFC 1738
ftp://[ftp_username[:ftp_PWD]@]Servername[:Port]
In Google Chrome und Firefox standardmäßig deaktiviert
 - WebFTP Dienst von FTP-Servern, der zum Zugriff auf Webserver mittels Webbrowsern per HTTP
 - Terminal Kommandozeilen-/ Terminal-/ Shell-Implementierung von FTP
 - Dateimanager Einbindung von FTP in die Dateimanager der Betriebssysteme (z.B. Einbindung von FTP-Servern als lokales Laufwerk)
 - Proprietäre Programme z.B. FileZilla o.ä.
 - Weitere Varianten/ Integrationen

Grundlagen: Protokolle

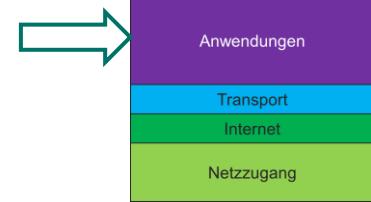
FTP over SSL (FTPS)



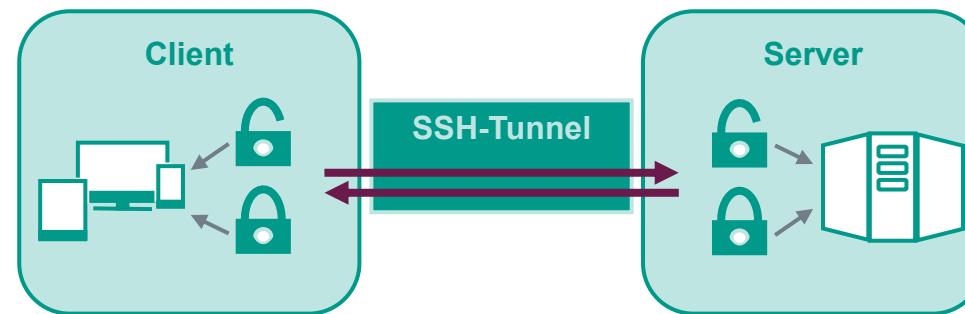
- FTP-Kommunikation über Transport Layer Security (TLS)
- Ablauf:
 - Aufbau eines unverschlüsselten Kontrollkanals zum Server vom Client
 - Client sendet die Aufforderung **AUTH TLS**
Ohne diese Aufforderung kann der Server alle weiteren Aufforderungen ablehnen
 - Ankündigung des Wechsels zu TLS durch die die Antwort **234** vom Server
Ohne diese Antwort kann der Client unverschlüsselt weitermachen
 - Nach dem Wechsel zu TLS auf dem Kontrollkanal kann der Client zusätzlich TLS für die Nutzdatenkanäle anfordern
 - Client kann zu jeder Zeit die Rückkehr zur unverschlüsselten Verbindung fordern
 - Rückkehr zur unverschlüsselten Verbindung kann vom Server verweigert werden

Grundlagen: Protokolle

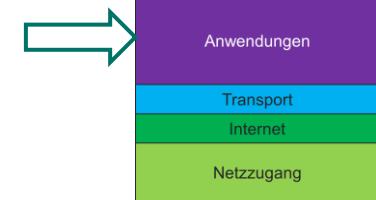
Secure File Transfer Protocol (SFTP)



- verschlüsselte Datenübertragung zwischen Clients und Servern
- nur eine Verbindung
- Daten der Verbindungsaunahme und übertragene Daten werden mit den Schlüssel aus dem SSH-Protokoll verschlüsselt.



Network News Transfer Protocol (NNTP)

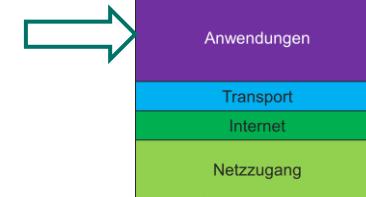


- Übertragungsprotokoll für Nachrichten in Newsgroups
→ Usenet
- spezifiziert im RFC 977 aus dem Februar 1986
- Textbasiertes Protokoll mit abwechselnden Anfragen des Clients und Antworten des Servers

Grundlagen: Protokolle

Simple Mail Transfer Protocol (SMTP)

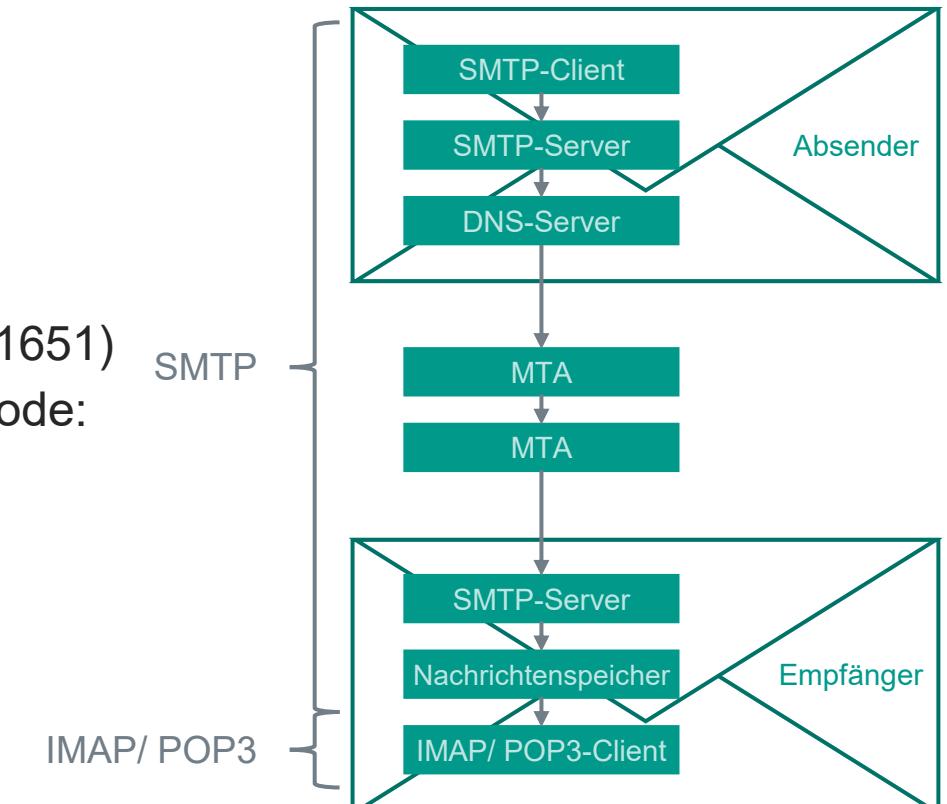
Extended Simple Mail Transfer Protocol (ESMTP)



- SMTP:
 - RFC 821 vom Februar 1982
 - textbasiertes TCP-Protokoll
 - Austausch von E-Mails in Netzwerken
 - Keine Authentifizierung, daher auftretendes Spam-Problem

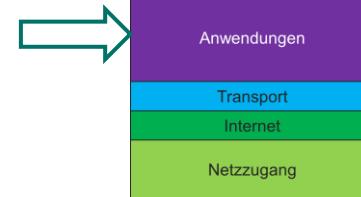
- ESMTP:
 - RFC 1869 vom November 1995 (auf Basis von RFC 1425/ RFC 1651)
 - Ergänzung von SMTP um weitere Kommandos im 8-Bit-ASCII-Code:
→ neue Funktionen zur Einsparung von Bandbreite und zum Schutz von Servern

- Authentifizierung des Absenders
- SSL-Verschlüsselungen von E-Mails
- Anhängen von Multimedia-Dateien an E-Mails
- Größenbeschränkungen von E-Mails gemäß Server-Vorgaben
- Gleichzeitiger Versand an mehrere Empfänger
- Standardisierte Fehlermeldungen bei Unzustellbarkeit



Grundlagen: Protokolle

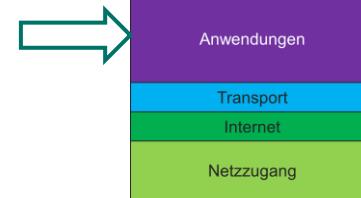
Telnet (Teletype Network)



- zeichenorientierter Datenaustausch über TCP
- erlaubt die Fernsteuerung von Computern mittels textbasierter Ein- und Ausgaben
 - Client-Server-Verbindung über TCP auf Port 23
 - das ferngesteuerte Gerät dient als Server und wartet auf Befehle von der Clientseite
→ (Telnet-) Client ist die steuernde Instanz
- kann auch verwendet werden, um Anwendungen zu steuern, die über keine grafische Oberfläche haben
- Zugriff auf Datenbanken
- Interaktion mit Anwendungen auf Applikationsservern
- Administration von Netzwerken und Servern
- auf vielen (netzwerkfähigen) Betriebssystemen standardmäßig installiert
→ muss jedoch meist explizit eingerichtet/ freigeschaltet werden

Grundlagen: Protokolle

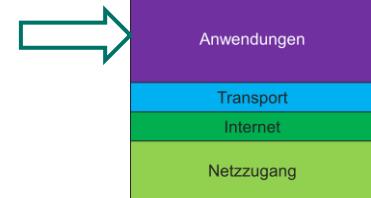
Hypertext Transfer Protocol (HTTP)



- Verbindungs- und statusloses Protokoll:
Client und Server nehmen keine besondere Zustände an
 - Nach jeder Kommunikation/ jedem Kommando wird die Kommunikation vollständig beendet
 - Erfolgreich
 - Fehlermeldung
 - Die Reaktion auf die jeweilige Antwort obliegt dem Kommunikationspartner
- Aktuell: HTTP/2 nach RFC 7540 vom Mai 2015

Grundlagen: Protokolle

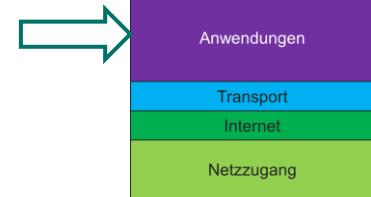
Hypertext Transfer Protocol (HTTP)



- Textbasiertes (ASCII) Protokoll
- Verbindungs- und statuslos:
Client und Server nehmen keine besondere Zustände an
 - Nach jeder Kommunikation/ jedem Kommando wird die Kommunikation vollständig beendet
 - Erfolgreich
 - Fehlermeldung
 - Die Reaktion auf die jeweilige Antwort obliegt dem Kommunikationspartner
- Aktuell: HTTP/2 nach RFC 7540 vom Mai 2015

Grundlagen: Protokolle

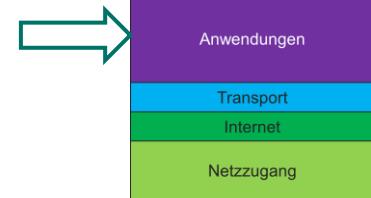
Hypertext Transfer Protocol (HTTP)



- HTTP-Kommunikation zwischen Client und Server basiert auf zwei Arten von Nachrichten
 - Anfrage (Request) vom Client an den Server
 - Antwort (Response) Reaktion vom Server zum Client
- Nachrichten bestehen stets aus zwei Teilen
 - Nachrichtenkopf (Message Header/ HTTP-Header)
→ Informationen über den Nachrichtenrumpf, z.B.:
 - verwendete Kodierungen
 - Inhaltstyp (um vom Empfänger richtig interpretiert werden zu können)
 - Nachrichtenrumpf (Message Body).
→ die Nutzdaten

Grundlagen: Protokolle

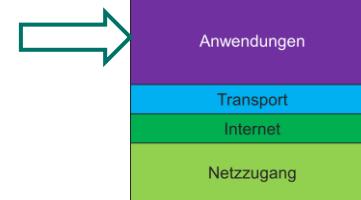
Hypertext Transfer Protocol (HTTP)



- HTTP-Kommunikation zwischen Client und Server basiert auf zwei Arten von Nachrichten
 - Anfrage (Request) vom Client an den Server
 - Antwort (Response) Reaktion vom Server zum Client
- Standardmäßig läuft die Kommunikation über Port 80
- Nachrichten bestehen stets aus zwei Teilen
 - Nachrichtenkopf (Message Header/ HTTP-Header)
→ Informationen über den Nachrichtenrumpf, z.B.:
 - verwendete Kodierungen
 - Inhaltstyp (um vom Empfänger richtig interpretiert werden zu können)
 - Nachrichtenrumpf (Message Body).
→ die Nutzdaten

Grundlagen: Protokolle

Hypertext Transfer Protocol (HTTP)



- Anfrage:
GET /winf.html HTTP/1.1
Host: www.fom.de

→ die Ressource **winf.html** soll vom Host mit dem Namen **www.fom.de** angefordert werden

- **www.fom.de** wird über DNS in eine IP-Adresse übersetzt
- Übertragung wird über den TCP-Standard-Port 80 des HTTP-Servers mit einer HTTP-GET-Anforderung angefordert

- Antwort:
HTTP/1.1 200 OK
Server: Apache/1.3.29 (Unix) PHP/4.3.4
Content-Length: 123456
Content-Language: de
Connection: close
Content-Type: text/html
(Inhalt von winf.html)

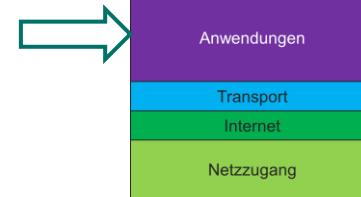
(Größe von **winf.html** in Byte)

(nach RFC 3282 sowie RFC 1766)

- Dateien werden normalerweise in „Seitenbeschreibungssprachen“ wie HTML übertragen
- Zusätzliche Inhalte können Bilder, Stylesheets, Skripte usw. sein
- jede Datei kann in jedem beliebigen Format übertragen werden

Grundlagen: Protokolle

Hypertext Transfer Protocol (HTTP)

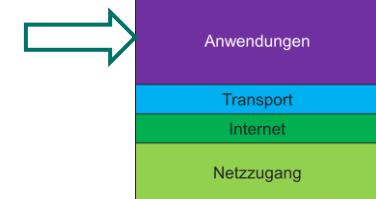


■ HTTP-Anfragen

- **GET** Anforderung des Clients von Ressourcen vom Server unter Angabe der URI
- **POST** Senden von Daten zur Verarbeitung vom Client an den Server
→ z.B. Name-Wert-Paare aus HTML-Formularen (Vertiefung in Kap. 6: Dyn. Webseiten)
- **HEAD** Anweisung für den Server, den gleichen HTTP-Header wie bei GET zu senden, aber einen anderen Nachrichtenrumpf
→ schnell Prüfung auf Dateien im Browser-Cache
- **PUT** Anweisung, eine Ressource unter Angabe der Ziel-URI auf einen Webserver hochzuladen
→ existiert schon eine Ressource unter der Ziel-URI, so wird sie ersetzt ansonsten neu erzeugt
- **PATCH** Anweisung zur Änderung einer bestehenden Ressource
- **DELETE** Anweisung zum Löschen einer Ressource vom Server
- **TRACE** liefert die Anfrage unverändert vom Server zurück
→ Überprüfung, ob/ wie eine Anfrage auf dem Weg zum Server verändert wurde
- **OPTIONS** Anforderung einer Liste der vom Server unterstützten Methoden und Merkmale
- **CONNECT** für den Aufbau von SSL-Tunnels

Grundlagen: Protokolle

Hypertext Transfer Protocol (HTTP)



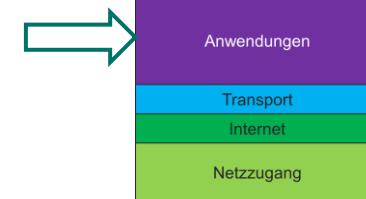
- Anfrage:
`GET /winf.html HTTP/1.1`
`Host: www.fom.de`
- Erweiterung von GET-Anfragen
→ Ergänzung durch Name-Wert-Paare

`GET /studies/Spezial:Suche?suche=Winf&med=Videos HTTP/1.1`
`Host: www.fom.de`

Name/ Argument	Wert
suche	Winf
med	Videos

Grundlagen: Protokolle

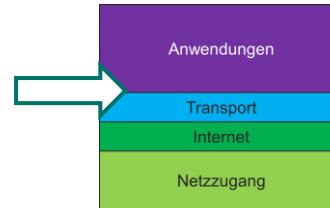
Hypertext Transfer Protocol (HTTP)



- HTTP-Statuscodes
 - **1xx** Informationen
 - Bearbeitung der Anfrage dauert trotz Rückmeldung noch
 - kann notwendig sein, weil Clients oft nach einer bestimmten Zeit annehmen, dass ein Fehler bei der Übertragung oder Verarbeitung der Anfrage aufgetreten ist
 - **2xx** Erfolgreiche Operation
 - Anfrage wurde bearbeitet und an den Anfragesteller zurückgesendet
 - **3xx** Umleitung
 - Für eine erfolgreiche Bearbeitung der Anfrage sind weitere Schritte seitens des Clients erforderlich
 - **4xx** Client-Fehler
 - Bei der Bearbeitung der Anfrage durch den Server ist ein Fehler aufgetreten, für den der Client verantwortlich ist:
 - 404**, wenn ein Dokument angefragt wurde, das nicht existiert
 - 403**, wenn es dem Client nicht erlaubt ist, das entsprechende Dokument abzurufen
 - **5xx** – Server-Fehler
 - Fehler auf Seiten des Servers

Grundlagen: Protokolle

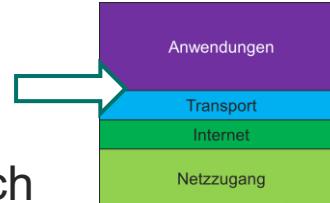
Hypertext Transfer Protocol Secure (HTTPS)



- Kommunikationsprotokoll für die „abhörsichere“ Datenübertragung im WWW
→ „Transportverschlüsselung“
- Erste Implementierung 1994 im damaligen Netscape Navigator
- Ziel: Vertraulichkeit und Integrität in der Kommunikation von Webserver und Webbrowser:
 - Verschlüsselung
 - Authentifizierung (später in diesem Abschnitt)
- HTTPS ist in seiner Syntax identisch mit HTTP
- Die Verschlüsselung der Daten geschieht mittels SSL/ TLS
- Standard-Port für HTTPS-Verbindungen ist Port **443**

Grundlagen: Protokolle

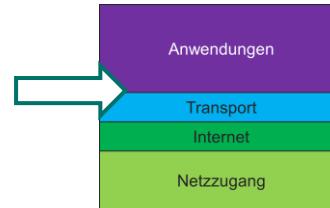
Hypertext Transfer Protocol Secure (HTTPS)



- Entscheidung, ob eine HTTPS- oder eine HTTP-Verbindung genutzt wird, ist unterschiedlich möglich
 - Serverseitig:
 - es wird ausschließlich HTTPS zugelassen
→ Online-Banking
 - eventuelle automatische Weiterleitung von HTTP- auf HTTPS-Adresse
 - Login wird über HTTPS erzwungen:
 - Setzen eines HTTP-Cookies im Browser
→ die weitere Kommunikation erfolgt unverschlüsselt
 - Clientseitig durch HTTP Strict Transport Security (HSTS)
 - Wenn Server nur HTTPS zulassen, kann der Browser dies speichern
→ alle zukünftigen Verbindungen werden dann immer über HTTPS aufgebaut

Grundlagen: Protokolle

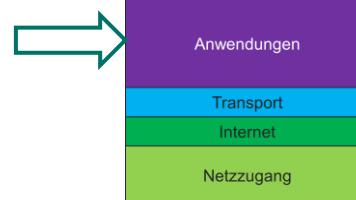
HTTP/2 vs. HTTP/3: Gemeinsamkeiten und Unterschiede



- Unterschiede:
 - HTTP/3 baut auf UDP aus, HTTP/2 auf TCP
 - HTTP/3 integriert TLS-1.3-Verschlüsselung
 - keine zusätzlichen Verschlüsselungsanfragen (Handshakes) auf TLS-Ebene
 - ausschließlich verschlüsselte Verbindungen.
- Gemeinsamkeiten:
 - Beide Protokolle nutzen Header-Kompression
 - die HTTP/2 HPACK-Kompression wird in HTTP/3 QPack abgelöst
 - beide Protokolle unterstützen das beschleunigte Senden von CSS- und JavaScript-Daten
 - beide Protokolle nutzen Anfrage-/ Antwort-Multiplexing
 - Stream-Priorisierung, um Seiteninhalte priorisiert laden zu können

Grundlagen: Protokolle

Websockets



- TCP basiertes Netzwerkprotokoll für bidirektionale Verbindungen zwischen Webanwendung und Webserver
→ Webserver muss dazu WebSockets unterstützen
- HTTP: jede Aktion des Servers bedarf einer vorhergehenden Anfrage des Clients
- Websockets: der Client kann die Verbindung öffnen und dann „offen stehen lassen“
der Server kann auf dieser Verbindung jederzeit neue Daten übertragen

Clientanfrage:



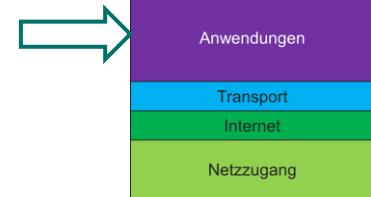
Serverantwort:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzhZBbK+xOo=
Sec-WebSocket-Protocol: chat
```

Grundlagen: Protokolle

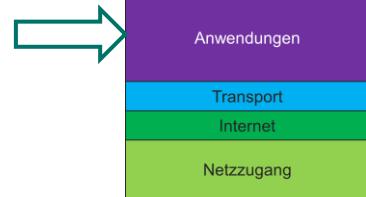
Web-based Distributed Authoring and Versioning (WebDAV)



- HTTP/1.1-basiertes Netzwerkprotokoll zur Bereitstellung von Dateien über das Internet
 - Übertragung ganzer Verzeichnisse
 - Spezifikation einer Versionskontrolle
- WebDAV erweitert HTTP um neue Methoden und Header-Attribute:
 - **PROPFIND** Abfragen von in XML beschriebenen Eigenschaften einer Ressource
 - **PROPPATCH** Ändern/ Löschen mehrerer Eigenschaften einer Ressource mit einer Anfrage
 - **MKCOL** Erstellen eines Verzeichnisses
 - **COPY** Kopieren einer Ressource (Dateinamen werden als URI angegeben)
 - **MOVE** Verschieben einer Ressource
 - **DELETE** Löschen einer Ressource
 - **LOCK** Anweisung, eine Ressource zu sperren (Zugriffskontrolle)
 - **UNLOCK** Anweisung, eine Ressource zu entsperren

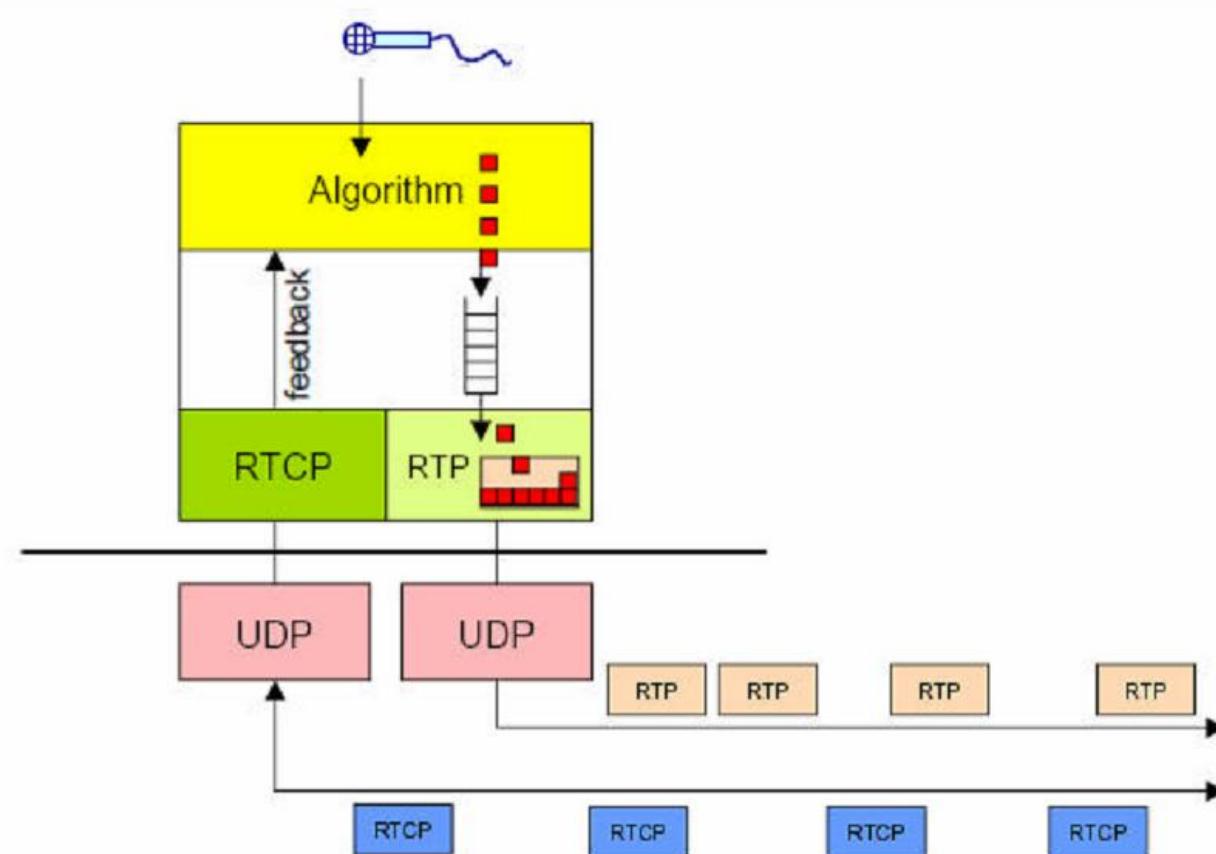
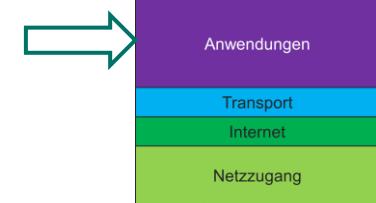
Grundlagen: Protokolle

Real-Time Transport Protocol (RTP)



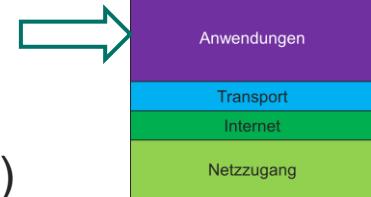
- Protokoll zur kontinuierlichen Übertragung audiovisueller Daten über IP-basierte Netzwerke
→ Streams
- RTP kümmert sich hauptsächlich um die Übertragung von Datenströmen, die „Echtzeit“ benötigen
→ Steuerung und Kontrolle der Datenübertragung übernimmt Real-Time Streaming Protocol (RTSP)
→ Quality-of-Service (QoS) wird durch RealTime Control Protocol (RTCP) beschrieben
- Transport von „Multimedia“-Datenströmen (Audio, Video, Text, etc.) über Netzwerke
 - Kodieren der Daten
 - „Paketieren“ der Daten → paketbasiert
 - Versenden der Daten
- Anwendungsbeispiel: IP-Telefonie nach den Standards H.323 und SIP
→ Übertragung der Audio- und Videoströme des Gesprächs

Real-Time Transport Protocol (RTP)



Grundlagen: Protokolle

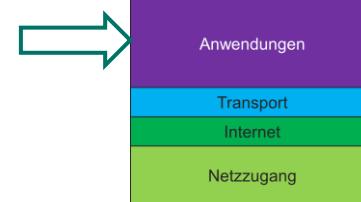
RealTime Control Protocol (RTCP)



- Protokoll zur Aushandlung und Einhaltung von Parametern der Dienstqualität (Quality of Service)
→ periodischer Austausch von Steuernachrichten zwischen Sender und Empfänger
 - Rückmeldung der bisherigen QoS
→ mögliche Anpassung der Datenübertragungsrate
 - Identifikation aller Sitzungsteilnehmer
→ Synchronisierung semantisch zusammenhängender, aber getrennt gesendeter Medienströme
 - Steuerung der für RTCP-Pakete verwendeten Datenübertragungsrate
→ Vermeiden, dass der Austausch von RTCP-Nachrichten nicht die Übertragung behindert
- In der Regel UDP-basierte Implementierung
 - verwendete Portnummern sind nicht statisch fixiert,
sondern werden für jede RTP-Session neu bestimmt
→ RTP nutzt (i.d.R.) eine gerade, RTCP die folgende ungerade Portnummer

Grundlagen: Protokolle

Real-Time Streaming Protocol (RTSP)



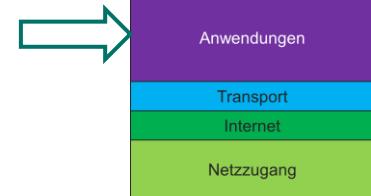
- Textbasiertes Netzwerkprotokoll zur Steuerung der kontinuierlichen Übertragung audiovisueller Daten über IP-basierte Netzwerke, dass in Aufbau und Verhalten HTTP ähnelt
→ Steuerung der Session zwischen Empfänger und Server

- | | |
|----------------------------|--|
| • Aggregate Control | Steuerung mehrerer Streams im Media-Server durch eine Zeitachse
Zur gemeinsamen Steuerung kann der Client einen Befehl für beide senden „Mehrplatz-Multimedia-Präsentation“ |
| • Conference | empfängt kontinuierlich Informationen vom Server |
| • Client | Netzwerkverbindung mit dem Ziel des Informationsaustauschs |
| • Connection | Datei, die Informationen für Media-Streams liefert |
| • Container File | Audio- und Videosignale müssen in gleicher zeitlicher Reihenfolge beim Client abgespielt werden wie beim Server |
| • Continuous Media | Summe der Transportinformationen die Client(s) und Server(n) |
| • Entity | Datentyp/ Codec, Zeitangaben, Farbsignale, etc. |
| • Media Initialization | Server, der den/ die Media-Streams bereitstellt. |
| • Media Server | Umleitung eines Clients zu einem anderen Media-Server |
| • Media Server Indirection | Instanz eines Audio- oder Video-Streams, o.ä. |
| • (Media) Stream | Befehl zum Verbindungsauftbau. |
| • Message | Teilnehmer einer Conference |
| • Participant | |

Grundlagen: Protokolle

Network Time Protocol (NTP)

Precision Time Protocol (PTP)



- NTP:

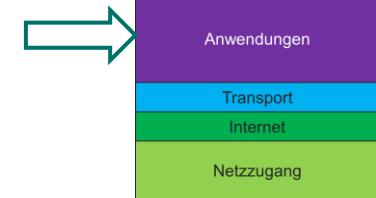
- Standard zur Synchronisierung von Uhren der Computersysteme in paketbasierten Kommunikationsnetzen
→ Ziel: zuverlässige Zeitangabe in Netzwerken mit variabler Paketlaufzeit
- Basiert auf
 - UDP (verbindungslos)
 - TCP (verbindungsbezogen)
- Vereinfachte Version: Simple Network Time Protocol (SNTP)

- PTP:

- höhere Genauigkeit in lokal begrenzten Netzen
 - in Hardware-Ausführung Genauigkeiten im Bereich von Nanosekunden
 - In Software-Implementierung im Bereich unter einer Mikrosekunde

Grundlagen: Protokolle

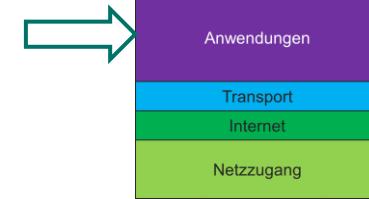
Microsoft Media Server Protocol (MMS)



- Von Microsoft entwickeltes Protokoll zur Übertragung von Streaming Media
→ proprietärer Dialekt von RTSP
- Die Dateien sind zwar für die Nutzung im Windows Media Player beschrieben zugeschnitten, können aber auch von anderen/ eigenen Anwendungen be-/ verarbeitet werden
- Um MMS-Streams hören zu können, werden proprietäre Codecs benötigt
→ MMS überträgt über TCP oder UDP dann Audio- und Videodaten im ASF-Format
→ ASF ist wiederum ein Container für WMA- oder WMV-Dateien

Grundlagen: Protokolle

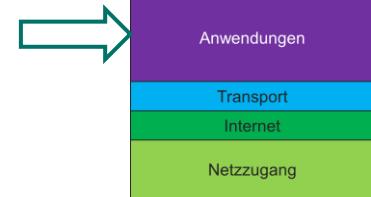
Web Real-Time Communication (WebRTC)



- „Web-Echtzeitkommunikation“
- Sammlung von Kommunikationsprotokollen und Programmierschnittstellen, die „Echtzeit“kommunikation über Rechner-Rechner-Verbindungen ermöglichen soll
- Ziel: Webclients (Browser) können nicht nur Datenressourcen von Servern abrufen, sondern auch Echtzeitinformationen von Clients anderer Benutzer
 - Videokonferenz
 - Dateitransfer/ Datenübertragung
 - Chat
 - Desktopsharing
- Offener Standard/ Framework, basierend auf HTML5 und JavaScript
 - Übertragung erfolgt mit RTP/ SRTP über eine Peer-to-Peer-Verbindung
 - Diese basiert auf dem JavaScript Session Establishment Protocol (JSEP)

Grundlagen: Architekturen

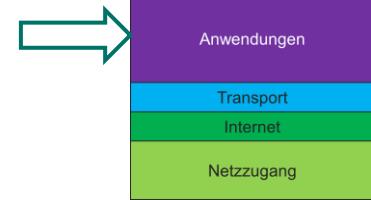
REST - REpresentational State Transfer, API für Application Programming Interface



- Ansatz für die Kommunikation zwischen Client und Server in Netzwerken
- Programmierschnittstelle, die sich an den Paradigmen und Verhalten des WWW orientiert
→ Grundlage: Maschine-Maschine-Kommunikation
- Alternative zu anderen Schnittstellen wie SOAP, WSDL oder RPC
(werden später in diesem Abschnitt kurz angesprochen)
- REST ist weder Protokoll noch Standard:
Als „RESTful“ charakterisierte Implementierungen bedienen sich standardisierter Verfahren
 - HTTP/S
 - URI
 - JSON ([siehe hierzu auch Abschnitt 6](#))
 - XML
 - (und auch weitere)
- Entwicklung: als Dissertation 2000, parallel zu HTTP1.1

Grundlagen: Architekturen

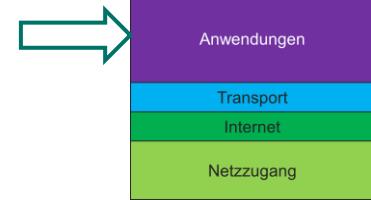
REST - 6 Architekturprinzipien (1)



- Client-Server-Modell
 - Trennung von Nutzerinterface und Datenhaltung
 - „leichtere“ Portierung von Clients auf andere Plattformen
 - Vereinfachung der Skalierung von Serverkomponenten
- Zustandslosigkeit
 - zustandslose Kommunikation von Client und Server
 - keine Sitzungsverwaltung
 - jede REST-Nachricht beinhaltet alle Informationen, die für die Verarbeitung benötigt werden
 - Server können auf keinen gespeicherten Kontext zurückgreifen
 - Vorteil: verbesserte Visibilität, Zuverlässigkeit, Skalierbarkeit, Lastverteilung
 - Nacht(e): schlechtere Performanz und
keine Kontrolle des Servers über das konsistente Verhalten der Client-Anwendung

Grundlagen: Architekturen

REST - 6 Architekturprinzipien (2)

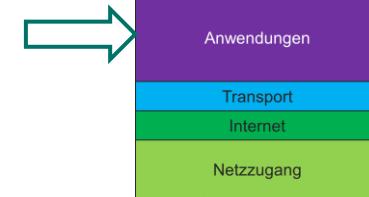


▪ Caching

- Clients können vom Server gesendete Antworten speichern
→ bei gleichartigen Requests können die Antworten später erneut verwendet
- Informationen müssen als „cacheable“ oder „non-cacheable“ gekennzeichnet werden
- Vorteile: „responsivere“ Anwendungen, höhere Effizienz und Skalierbarkeit
- Nachteile/ Risiken: Clients greifen evtl. auf veraltete Daten aus dem Cache zurück

Grundlagen: Architekturen

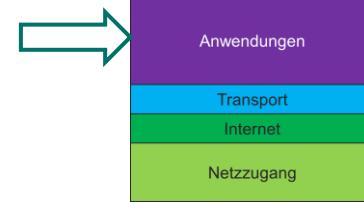
REST - 6 Architekturprinzipien (3)



- Einheitliche Schnittstelle
 - eine einheitliche, allgemeine und vom implementierten Dienst entkoppelte Schnittstelle
 - Ziele: vereinfachte Architektur und erhöhte Visibilität von Interaktionen
 - Nachteile: schlechtere Effizienz durch die Umwandlung von Informationen in ein standardisiertes Format und evtl. nicht für die Anwendungen geeignetes Format
- Vier Eigenschaften der Einheitlichkeit:
 - Adressierbarkeit von Ressourcen
 - jede Information, die über einen URI kenntlich gemacht wurde, ist eine Ressource
 - jeder REST-konforme Dienst hat eine eindeutige Adresse (URL)
 - Repräsentationen zur Veränderung von Ressourcen
 - unter einer Adresse zugängliche Dienste können unterschiedliche Darstellungsformen haben
 - z.B. verschiedene Sprachen, oder verschiedene Formate, wie z.B. HTML, JSON oder XML
 - Selbstbeschreibende Nachrichten
 - „Hypermedia as the Engine of Application State“ (HATEOAS)
 - Client einer REST-Schnittstelle navigiert ausschließlich über URLs, die vom Server bereitgestellt werden

Grundlagen: Architekturen

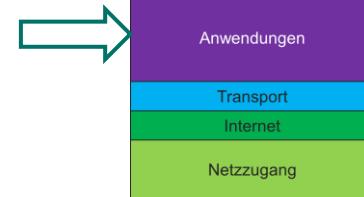
REST - 6 Architekturprinzipien (4)



- Layered System
 - mehrschichtiger, hierarchischer Aufbau, bei dem jede Komponente nur die jeweils direkt angrenzenden Schichten sehen.
 - Legacy-Anwendungen
 - Load Balancer als agierende Vermittler („Intermediaries“) zur Verbesserung der Skalierbarkeit verbessern
 - Nachteile: zusätzlicher Overhead und erhöhte Latenzen
- Code-On-Demand
 - Funktionen von Clients können über nachlad- und ausführbare Programmteile erweitert werden
 - Diese Option kann jedoch deaktiviert werden/ sein

Grundlagen: Architekturen

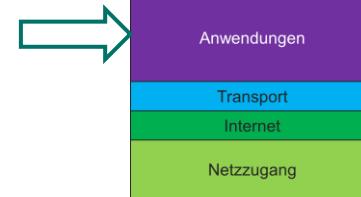
REST - Umsetzung per HTTP



- Realisierung/ Umsetzung des REST-Paradigmas erfolgt in der Praxis bevorzugt per HTTP/S
- Services werden per URL/ URI angesprochen
- HTTP-Methoden (**GET**, **POST**, **PUT**,...) geben an, welche Operation ein Dienst ausführen soll
- Für erste (eigene) Versuche:
Fake Online REST API for Testing and Prototyping.
<https://jsonplaceholder.typicode.com/> (letztmalig abgerufen am 22.12.2020)
 - Zugriff auf miteinander in Beziehung stehender Datensätze (100 Posts, 500 Kommentare, 100 Alben, 5.000 Fotos, 200 Todo-Listen und 10 Nutzer)
 - gewünschte Ressource sowie eventuelle Parameter werden jeweils nach dem Slash an die URL angehängt
 - **GET** fordert Daten vom Server an
 - **POST** übermittelt Daten an den Server
 - **PUT/ PATCH** ändern bestehende Daten auf dem Server
 - **DELETE** löscht bestehende Daten auf dem Server

Grundlagen: Architekturen

REST - Umsetzung per HTTP



- HTTP- Methoden

- **GET** fordert Ressourcen vom Server an
- **POST** übermittelt Ressourcen an den Server
- **PUT** erzeugt neue Ressourcen auf dem Server
- **PATCH** ändert bestehende Ressourcen auf dem Server
- **DELETE** löscht bestehende Ressourcen auf dem Server
- **HEAD** forderte Metadaten vom Server an
- **OPTIONS** erlaubte Methoden auf einer Ressource
- **CONNECT** Weiterleitung der Anfrage durch einen TCP-Tunnel
- **TRACE** gibt die Anfrage so zurück, wie der Server sie empfängt

Grundlagen: Architekturen

Service Oriented Architecture (SOA)

- industrieller Standard des World Wide Web Consortiums
- Ursprünglich: Simple Object Access Protocol
- Netzwerkprotokoll zum Austausch von Daten zwischen Systemen und zur Ausführung von Remote Procedure Calls (RPCs)
- stellt Regeln für das Nachrichtendesign auf
 - wie sind Daten in der Nachricht abzubilden
 - wie sind Daten zu interpretieren
 - Konvention für RPCs mittels SOAP-Nachrichten vor
- keine Vorschriften zur Semantik applikationsspezifischer Daten
- zum Senden von Nachrichten können beliebige Transportprotokolle verwendet werden
 - FTP, SMTP, HTTP, JMS
 - in der Praxis aus Gründen der Kompatibilität meist HTTP

Grundlagen: Architekturen

Service Oriented Architecture (SOA)

SOAP-ENV: Envelope

SOAP-ENV: Header

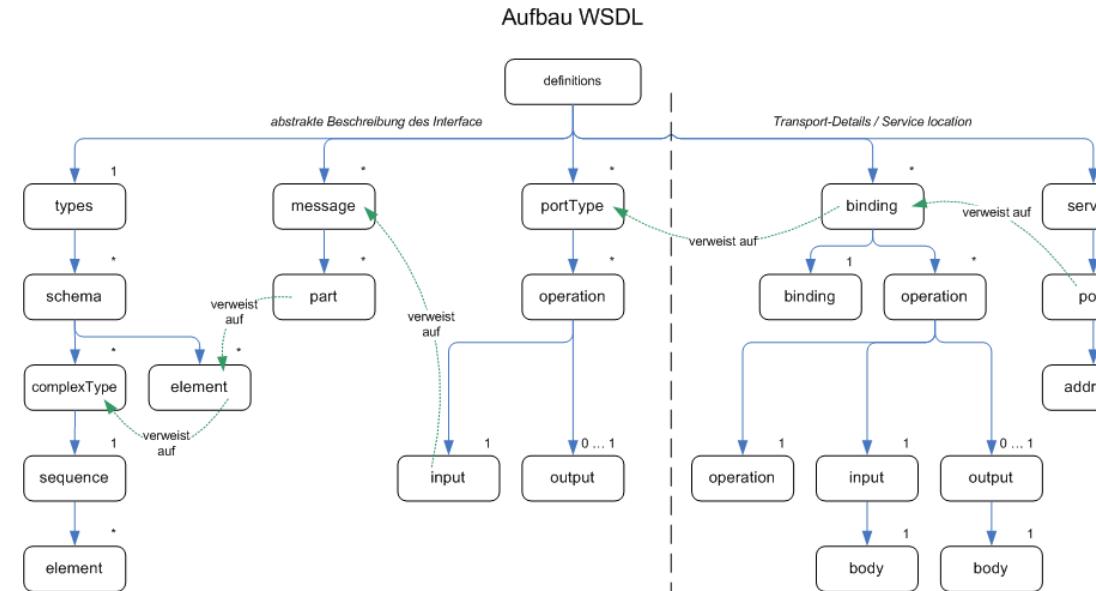
SOAP-ENV: Body

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
    <s:Header>
        <m:RequestID xmlns:m="http://www.lecture-db.de/soap">
            a3f5c109b<
        </m:RequestID>
    </s:Header>
    <s:Body>
        <m:DbResponse xmlns:m="http://www.lecture-db.de/soap">
            <m:title value="DOM, SAX und SOAP">
                <m:Choice value="1">
                    Arbeitsbericht Informatik
                </m:Choice>
                <m:Choice value="2">
                    Seminar XML und Datenbanken
                </m:Choice>
            </m:title>
        </m:DbResponse>
    </s:Body>
</s:Envelope>
```

Grundlagen: Architekturen

Web Services Description Language (WSDL)

- industrieller Standard des World Wide Web Consortiums (W3C)
- plattform-, programmiersprachen- und protokollunabhängige Beschreibungssprache für Netzwerkdienste (Webservices)
→ Austausch von Nachrichten auf Basis von XML

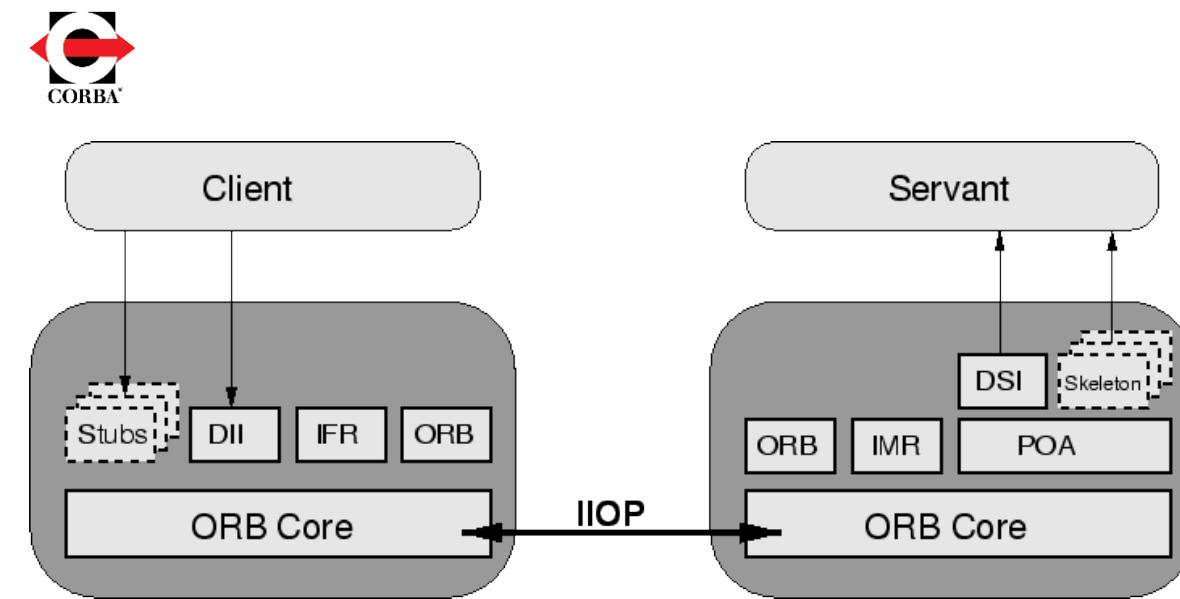


Wikipedia: WSDL

Grundlagen: Architekturen

Remote Procedure Call (RPC)

- Technik zur Realisierung von Interprozesskommunikation
- ermöglicht den Aufruf von Funktionen (Procedures) in anderen Adressräumen/
→ im Normalfall bedeutet dies, dass die aufgerufenen Funktionen auf einem anderen Computer ausgeführt werden
- Unterschiedlichste Implementierungen
 - Java RMI (Remote Method Invocation)
 - CORBA (Common Object Broker Architecture)
 - und viele mehr



Web-Services

- Webservices stellen Dienste über das Internet zur Verfügung
→ Maschine-Maschine-Kommunikation
- Zwei zentrale Eigenschaften:
 - Plattformunabhängigkeit
 - Client und Server müssen nicht die gleichen Konfigurationen haben, um kommunizieren zu können
 - Der Webservice schafft die gemeinsame Ebene für eine Kommunikation
 - Verteiltheit:
 - Webservices stehen i.d.R. nicht nur einem Client zur Verfügung
 - Über das Internet können unterschiedliche Clients auf den Webservice zugreifen
- Webservices werden über einen eindeutigen Uniform Resource Identifier (URI) angesprochen

Web-Services

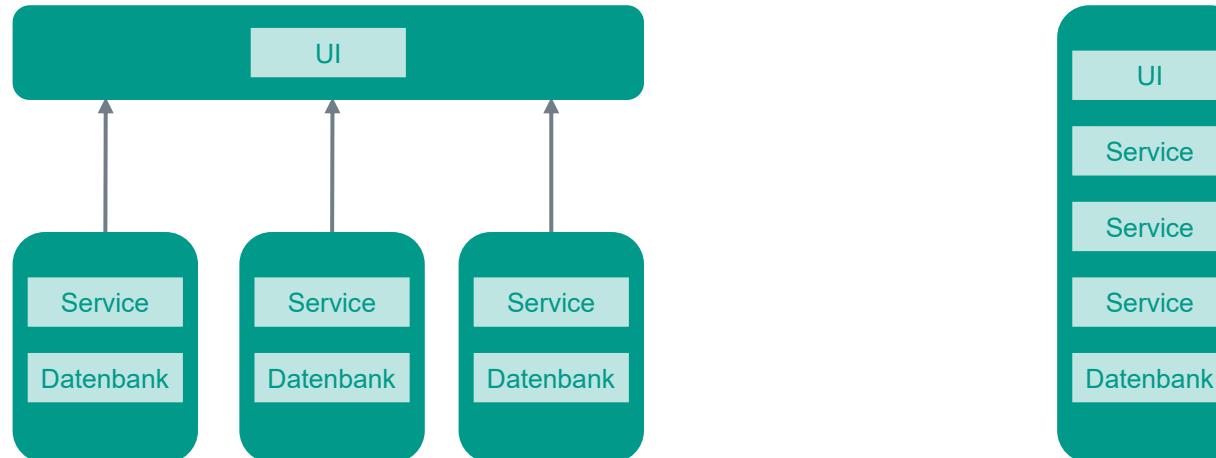
- Vorteil(e) von Webservices
 - die Kommunikation läuft plattformunabhängig ab
 - Client und Server müssen für eine funktionierende Kommunikation kaum Gemeinsamkeiten aufweisen
 - Dazu nutzen Webservices auf standardisierte Formate, die (nahezu) alle Systeme verstehen
- Nachteil(e)
 - XML ist ein „unhandliches“ und „sperriges“ Format, das zu großen Datenpaketen führt
 - Dies kann zu Problemen bei langsamem Netzwerkverbindungen führen
 - Alternative für die Kommunikation zwischen zwei Systemen: Web-APIs

Grundlagen: Architekturen

Micro-Services

- Architekturmuster mit dem Ziel, komplexe Anwendungen aus unabhängigen Prozessen aufzubauen
- Kommunikation über sprachenunabhängige Programmierschnittstellen
- Microservices sollen weitgehend entkoppelt arbeiten
- Microservice-Architektur

Monolithische Architektur



Micro-Services

- Vorteile

- Unabhängige Entwicklung durch unabhängige Teams
- Möglichkeit zur Implementierung in unterschiedlichen Programmiersprachen/ Technologien
- Überschaubare Größe der einzelnen Microservices
- Gesamt-Architektur bleibt auch bei Wartung (z.B. Austausch/ Update) einzelner Microservices erhalten
→ Grund: Implementierung auf Basis der eingesetzten API
- Langfristige Nutzung durch stabile Architektur
- Unabhängige Skalierbarkeit der Microservices
- Größere Robustheit des Gesamtsystems

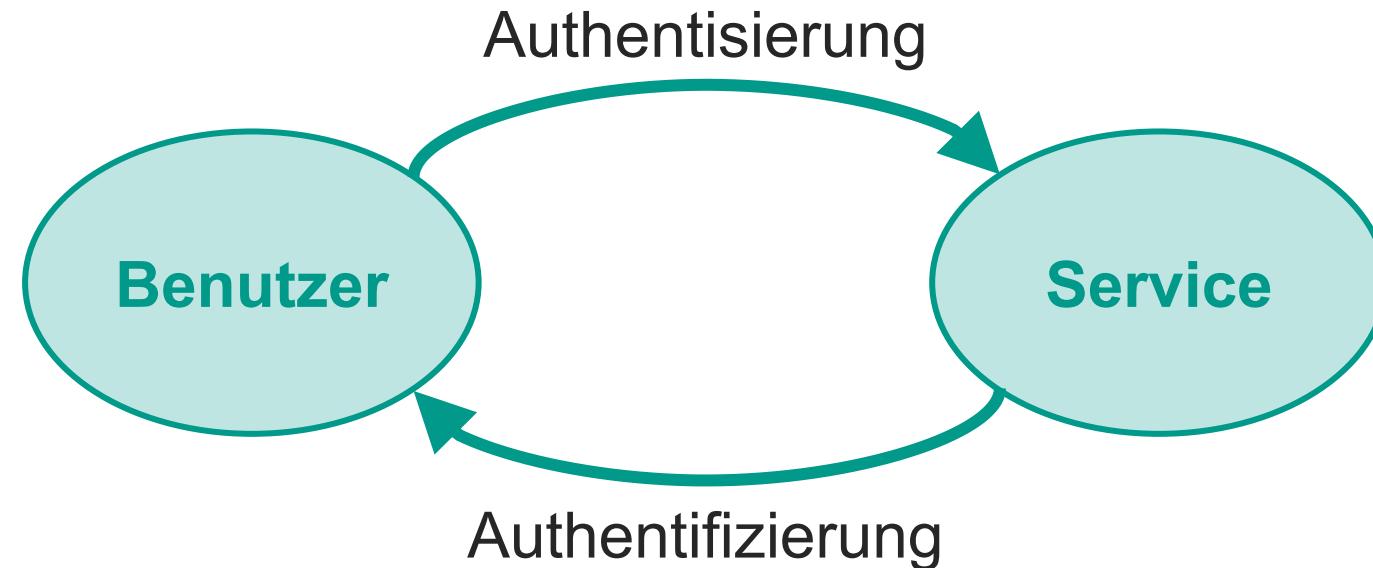
Micro-Services

- Nachteile
 - Größere Komplexität der Gesamtanwendung
 - Größere Komplexität des Testvorgangs
 - Logging und Monitoring ist durch die Verteilung schwieriger
 - Abhängigkeit von (Netzwerk-) Kommunikation
 - Je größer die Zahl der Microservices in einer Anwendung, desto höher die Wahrscheinlichkeit des Ausfalls eines Microservices
 - eventuell Notwendigkeit zur Zeitsynchronisierung

Grundlagen: Authentifizierung, Authentisierung & Autorisierung

Authentifizierung ↔ Authentisierung

- Authentifizierung bedeutet, den Nachweis über eine behauptete Eigenschaft einer Entität zu erbringen
→ Entität == Device, Service, Dokument, Mensch
- Authentisieren bedeutet im Gegenzug, dass die Entität die behauptete Eigenschaft belegen kann.



Grundlagen: Authentifizierung, Authentisierung & Autorisierung

Authentisierung

- Nachweis einer Entität, dass sie eine Eigenschaft tatsächlich aufweist von der sie behauptet, diese zu besitzen oder zu sein
- Die Entität legt einen Nachweis vor, die die behauptete Eigenschaft bestätigt
- Authentisierungsansätze
 - Identifizierungsinformation → **Wissen**: vorlegen geheimer Informationen, die die Eigenschaft bestätigen (z.B. Passwort)
 - Identifizierungsgegenstand → **Besitztum**: vorlegen eines akzeptierten Gegenstands/ Dokuments (z.B. Personalausweis)
 - Identifizierungsobjekt → **Gegenwart**: vorlegen einer untrennbar mit der Entität verbundenen Information (z.B. biometrische Merkmale wie Fingerabdruck)
- Beispiel:
 - Der FC-Bayer-Fan Thomas möchte das Pokalfinale live in Berlin sehen.
 - Er hat sich ein personalisiertes Ticket beim Veranstalter gekauft.
 - Am Finaltag wird am Stadion nicht nur geprüft, ob er ein gültiges Ticket hat, sondern auch, ob der auf dem Ticket aufgedruckte Name sein Name ist.
 - Mit dem Vorzeigen des Personalausweises behauptet er, dieser „Thomas“ zu sein.

Authentifizierung

- Prüfung einer behaupteten Authentisierung
- Prüfende Entität prüft die Angaben auf Echtheit
→ zeitlich bedeutet dies, dass die „Authentifizierung“ nach der „Authentisierung“ stattfindet
- Beispiel:
 - Nach der Authentisierung überprüft und bestätigt der Sicherheitsdienst die Identität von Thomas:
Der Sicherheitsmitarbeiter schaut sich den Personalausweis an und prüft, ob Bild und Name im Ausweis mit dem Thomas und dem Namen auf dem Ticket übereinstimmen.

Grundlagen: Authentifizierung, Authentisierung & Autorisierung

Autorisierung

- das Einräumen spezieller Rechte, z.B. Zugriff auf besondere Services oder Informationen
- Beispiel: Thomas darf nach erfolgreicher Authentifizierung, also seiner „Identifizierung“, ins Stadion.
Er wird autorisiert, dass Stadion betreten und auf seinem Sitz Platz nehmen zu dürfen.
Er wird nicht autorisiert, das Spiel aus der VIP-Lounge zu verfolgen.

Beispiel nach. <https://www.dr-datenschutz.de/authentisierung-authentifizierung-und-autorisierung/>

Grundlagen: Authentifizierung, Authentisierung & Autorisierung

Methoden der Authentisierung

	Wissen	Besitz	Gegenwart
Charakteristika	<ul style="list-style-type: none"> • kann vergessen werden • kann dupliziert, verteilt, weitergegeben oder verraten werden • kann erraten werden • die Preisgabe kann kompromittiert werden • die Mitführung erfordert keine Hilfsmittel 	<ul style="list-style-type: none"> • Kosten für die Erstellung des Merkmals • Verwaltung des Besitzes kann unsicher und mit Aufwand verbunden sein • Merkmal kann verloren gehen • Merkmal kann gestohlen, übergeben, weitergereicht oder dupliziert werden • Merkmal kann ersetzt werden • Merkmal kann sich selbst schützen (aktive Änderung) 	<ul style="list-style-type: none"> • Merkmal wird immer mitgeführt • kann nicht weitergegeben werden • spezielle Vorrichtung zur Erkennung notwendig • als Vergleich mit Referenzmuster i.A. nicht sicher (Wahrscheinlichkeit < 1) • fälschliche Akzeptanz/ Zurückweisung möglich • evtl. „Lebenderkennung“ notwendig • Merkmal kann sich verändern • kann nicht ersetzt werden • Probleme beim Datenschutz
Beispiele	<ul style="list-style-type: none"> • Passwort • PIN • Antwort auf eine Sicherheitsfrage • Zero-Knowledge-Beweis 	<ul style="list-style-type: none"> • Chip-/ Smartcard • RFID • physischer Schlüssel • Schlüssel-Code 	<ul style="list-style-type: none"> • Fingerabdruck • Gesichtserkennung • Stimmerkennung • Iriserkenntnung • Handschrift

Methoden der Authentisierung

- Wahl der jeweiligen Authentisierungsmethode(n) sollte in Abhängigkeit des Anwendungsgebietes erfolgen
- je nach Anwendungsgebiet verschiedene Vor- und Nachteile der gleichen Methode
 - Praktikabilität im Alltag
 - Sicherheitsbedarf des zu schützenden Guts
- Eine sorgfältige Abwägung vor Umsetzung und Inbetriebnahme gewährleistet das tatsächlich erreichbare Sicherheitsniveau.

FOM Hochschule

Web Technologie

(4) Grundlagen der Mediengestaltung

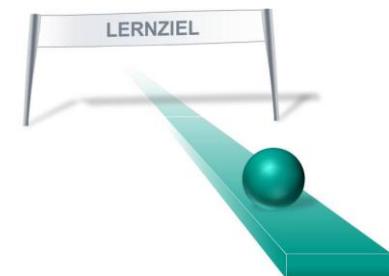
1	Formales, Vorstellung
2	Einführung
2.1	Allgemeine Einführung
2.2	Historie Internet, Web & Web-Technologien
3	(Web) Server – Client Kommunikation
3.1	Protokolle & Architekturen
3.2	Web- & Microservices
3.3	Authentifizierungs- und Authorisierungskonzepte
4	Grundlagen der Mediengestaltung
4.1	Media & Multimedia
4.2	Media & Modalitäten – Interaktion & Wahrnehmung
4.3	Hypermedia
5	Statische Webseiten
5.1	HTML
5.2	CSS
5.3	Medieneinbindung
6	Dynamische Webseiten
6.1	DOM, HTML & CSS
6.2	JavaScript & JSON
6.3	Weitere Konzepte & Sprachen der Webprogrammierung
7	Praxis
7.1	Vorstellung eines (ausgewählten) (JavaScript-) Frameworks
7.2	Vorstellung einer (ausgewählten) Thematik für ein eigenes Projekt
7.3	Betreute, eigenständige Entwicklung des (ausgewählten) Themas
8	Wiederholung (opt.)

Im Anschluss an diesen Themenblock sollen Sie:

- Die Bereiche „Medien/ Media“, „Multimedia“ und „Hypermedia“ definieren und voneinander abgrenzen können,
- Multimedia auf die menschlichen Modalitäten abbilden können,
- Die Herausforderungen der Kombination von Hypermedia und Multimedia verstehen können.

Dieser Themenblock hat nicht das Ziel, in die klassische Mediengestaltung einzuführen, denn:

- Die klassische Mediengestaltung ist zu umfangreich für eine kurze Darstellung innerhalb dieses Moduls.
- Die klassische Mediengestaltung ist nur peripher mit der Web-Technologie verwandt.



Multi... → mehr als eine Medienform

→ Ein Multimedium setzt sich also aus verschiedenen Medien(typen) zusammen.

Eine radikale Herangehensweise an eine Definition von Multimedia:

Sogar das klassische „Buch“ ist ein Multimedium.

Schließlich sind in nahezu allen Büchern neben Texten auch Bilder oder bildhafte Inhalte zu finden.

Folge:

Auch ein Film wäre dann ein Multimedium!

Auch hier steht in der Regel das Bild nicht alleine sondern wird von auditiven Zusätzen begleitet.

Kritik:

Bücher und Filme verstehen sich als eigenständige Medien. Für beide Medien gibt es unzählige theoretische Grundlagen und Gestaltungsprinzipien, die diese Kritik unterstreichen.

Multi... → mehr als eine Medienform

→ Ein Multimedium setzt sich also aus verschiedenen Medien(typen) zusammen.

Eine nicht ganz so radikale Definition (häufig in Internetquellen zu finden):

Jegliche Kombination aus Bildern, Videoclips, Texten, Ton und Grafik ist Multimedia.

(z.B. InfoWissWiki, 2006)

Kritik:

Es sieht aus, als ob nur diese und keine weiteren Medien(formen) zur Konstruktion eines Multimedias genutzt werden können.

(Mittlerweile hat InfoWissWiki seine Definition stark überarbeitet und folgt im wesentlichen der Definition nach Steinmetz.)

Betrachtung der Charakteristika der jeweils einzelnen Submedien:

Sowohl Text als auch Bild im Medium Buch als auch die visuellen und auditiven Anteile im Medium Film weisen jeweils gleiche Zeitverhalten auf.

Multi... → mehr als eine Medienform

→ Ein Multimedium setzt sich also aus verschiedenen Medien(typen) zusammen.

Die Definition nach Steinmetz:

Steinmetz greift die Zeitverhalten der einzelnen Medien auf.

Nach seiner Definition zeichnet sich „*ein Multimediasystem [...] durch die rechnergestützte, integrierte Erzeugung, Manipulation, Darstellung, Speicherung und Kommunikation von unabhängigen Informationen ... [aus], die in mindestens einem kontinuierlichen und einem diskreten Medium kodiert sind*

.

(Steinmetz: Multimedia-Technologie: Grundlagen, Komponenten und Systeme; Berlin, Heidelberg: Springer, 2000)

Kritik 1:

Steinmetz stellt die Technik in den Mittelpunkt.

Kritik 2:

Auch Steinmetz ist in seiner Definition radikal: Nach ihm muss ein Multimedium zwingend aus mindestens einem statischen und mindestens einem dynamischen Medium bestehen.

→ Dieser Definition folgend wäre ein Film ebenso wenig Multimedia wie ein Buch.

Generelle Kritik an den bisherigen Definitionsversuchen:**→ Wo bleiben dabei der Benutzer und seine Interaktion?**

- Benutzer erwarten mittlerweile zumindest minimalen Einfluss auf ein dynamisches Medium
 - (mindestens) Starten und Stoppen
 - zusätzlich vor- und zurück“spulen“
- Individuelle Verknüpfungsmöglichkeiten, die aus dem verwandten „Hypermedia“ stammen, entfallen.
- Dem Benutzer wird keine Möglichkeit angeboten, ...
 - ... die Abfolge der Präsentation und ...
 - ... den Handlungsverlauf zu beeinflussen.

Der Versuch einer ausführlichen Definition von Multimedia:

Multimedia im Sinne einer (medien-) informatischen Anwendung meint die einzelne oder kombinierte Darstellung verschiedener Medien, die sowohl zeitabhängig als auch zeitunabhängig sein können.

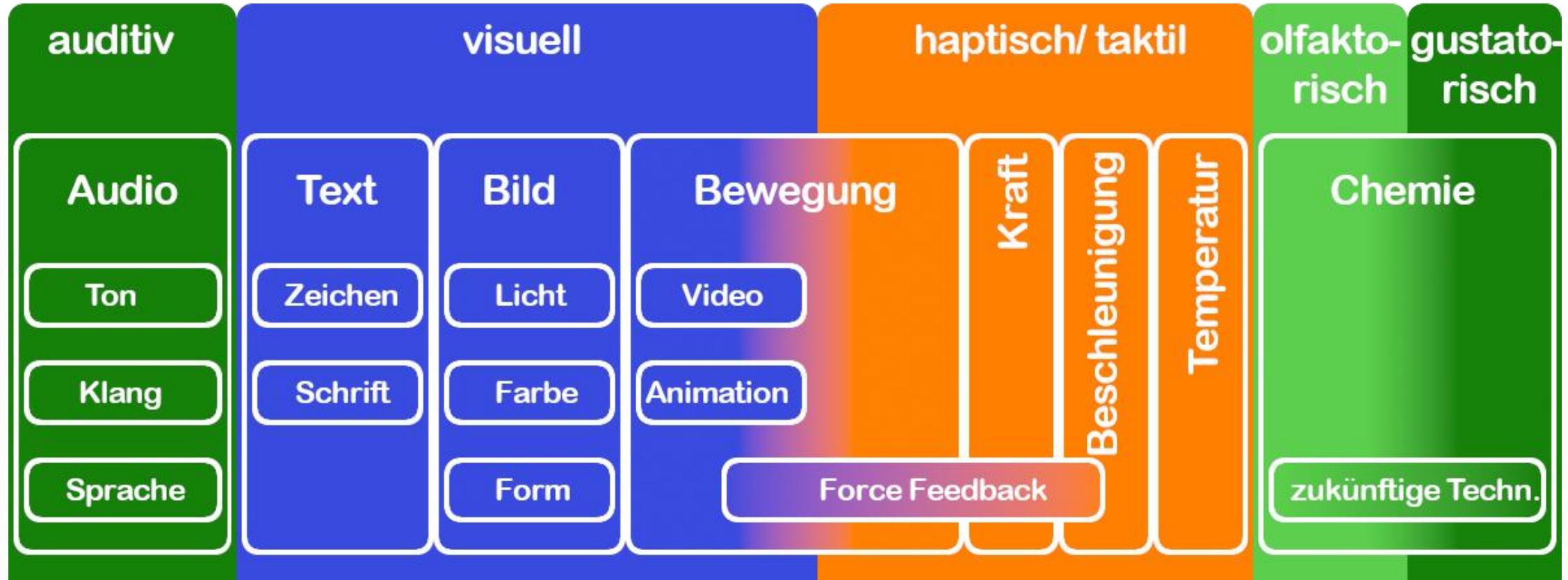
Diese Darstellung kann ein einzelnes Medium als auch jede mögliche Kombination zu einem singulären Zeitpunkt oder eine zeitliche Abfolge solcher Kombinationen in einem kontextbezogenen Rahmen sein.

Multimedia kann die Möglichkeit zur direkten und indirekten Einflussnahme durch einen Benutzer auf die Kombination oder auf den zeitlichen Ablauf der einzelnen Medien beinhalten.

(Hoffmann: Beyond Hypertext (2020))

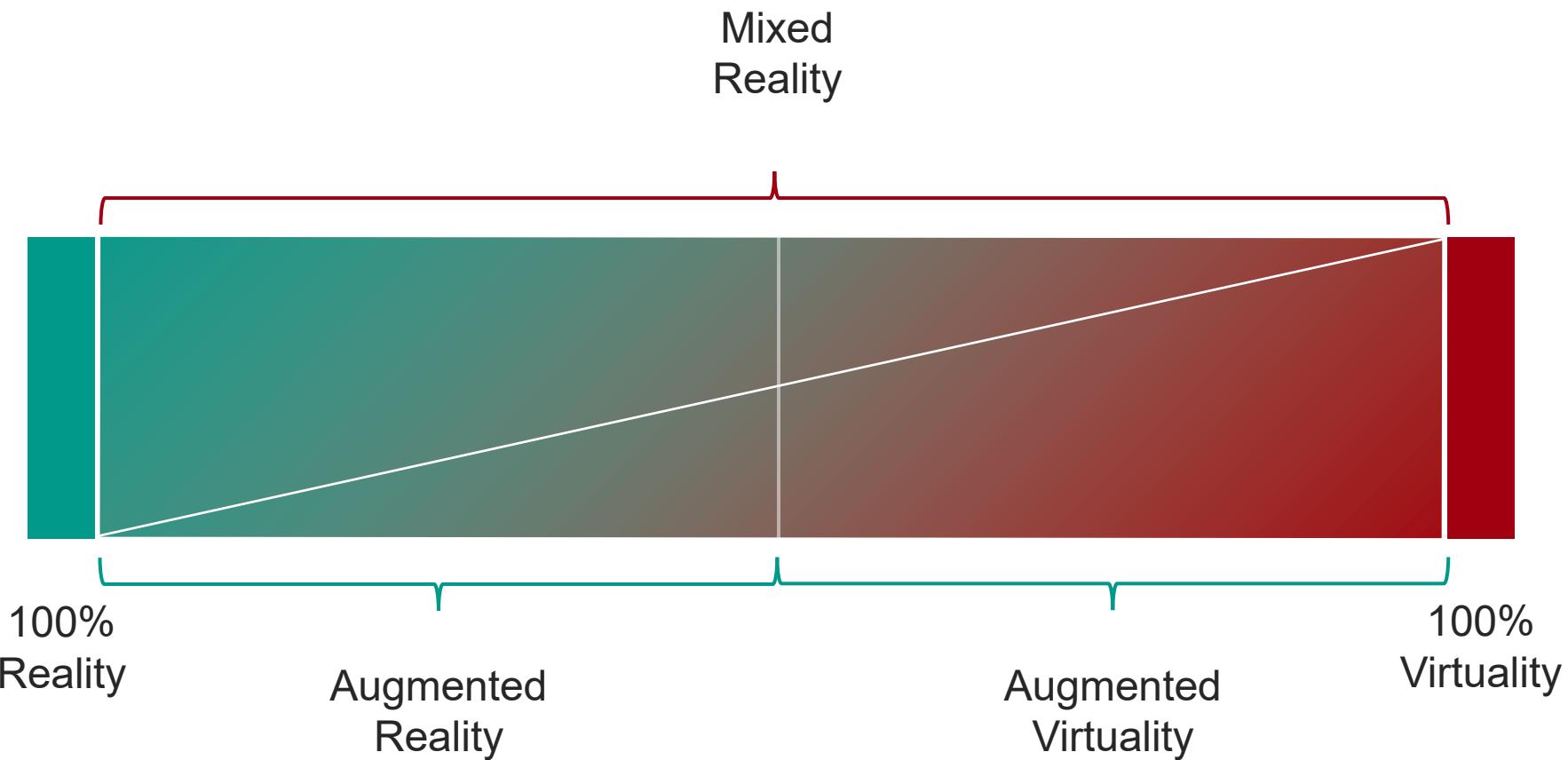
Medien vs. Modalitäten

- Modalität
 - genauer: **Sinnesmodalität** = „Empfindungskanal“
 - „physikochemische“ Reize werden durch Nervenfasern in spezifischen Nervenbahnen bis an die für die Auslösung der Empfindung zuständigen Hirnneuronengruppen geleitet.
 - einfacher ausgedrückt: reservierte Übertragungs-/ Kommunikationskanäle, die jeweils festgelegte bestimmte Adressen im Gehirn haben
 - „klassisch“: Sehen, Hören, Riechen, Schmecken, Fühlen
 - Erweiterung nach Ayres:
 - vestibulär: Gleichgewichtssinn
 - propriozeptiv Tiefensensibilität
 - taktil Oberflächensensibilität

Von den Modalitäten zu technischen Entsprechungen

Modalitäten und Synästhesie

- Synästhesie oder auch simultane Synthese:
Die einzelnen medialen Kanäle eines Multimediasystems verschmelzen zu einer ganzheitlichen Gesamtwahrnehmung
 - Voraussetzung:
 - geeignet viele sensorische und motorische Kanäle
 - synchrone Präsentationen
 - „Echtzeitverarbeitung“ (einige Millisekunden)
 - naturalistische Modellierung
 - Synästhesie kann **Immersion** erzeugen
→ der Eindruck einer vollständigen körperlichen Einbezogenheit in eine Welt

**Medialität, Modalitäten und Realität:
Das Realitäts-Virtualitäts-Kontinuum (Milgram et. al., 1994)**

Interaktion

- Ziel von Interaktion in Multimedia-Anwendungen ist es, mehrere Medien, die zunächst voneinander unabhängig sind, gemeinsam anzusprechen, um diese zu einem bestimmten Verhalten zu bewegen.
 - Ansprechen der Medien durch eine Anwendung(-slogik) Maschine-Medien-Interaktion
 - Ansprechen der Medien durch einen Anwender Mensch-Medien-Interaktion
 - Ansprechen der Medien durch (andere) Medien Medien-Medien-Interaktion
 - Media: **eine** Interaktion beeinflusst **ein** Medium
 - Multimedia: **eine** Interaktion beeinflusst **mehrere** Medien
- hoher technischer Aufwand bei „hoher Multimedialität“ (Synchronisierung)
→ großer Aufwand bei komplexen Operationen
→ hohe Belastung der Wahrnehmung bei „hoher Multimedialität“

Interaktive multimediale Systeme ...

- ... werden oft lediglich wegen ihrer Attraktivität statt wegen ihrer Effizienz eingesetzt.
- ... werden oft auch trotz ihrer Ineffizienz bzw. schlechten Performance eingesetzt!
→ „Klickibunti“

Interaktivität in Multimedia fördert, neben allen positiven Eigenschaften, auch Belastungen und Beanspruchungen beim Menschen!

- hoher mentaler Aufwand zur Gesamtproblemlösung
- hohe Anforderungen an die Konzentrationsfähigkeit
- hohe Belastung der Sensorik (meist der visuellen Informationsaufnahme)
- hohe Belastung des Gedächtnisses (vor allem des Kurzzeitgedächtnisses)

Hypermedia != Multimedia

- Definition Hypermedia: [siehe Abschnitt 2 - Einführung: Allgemeines und Historie](#)
 - Hypermedia kann ohne **Multi**media existieren
→ dann ist es eine spezielle Form von Hypermedia wie z.B. Hypertext
(wie im Folgenden noch gezeigt wird)
- Definition Multimedia: [siehe Abschnitt 5 - Grundlagen der Mediengestaltung \(dieser Abschnitt\)](#)
 - Multimedia kann ohne Hypermedia existieren
→ dann ist es eben Multimedia
- Hypermedia und Multimedia können miteinander verknüpft werden
 - Leider gibt es dafür keine feststehende, definierte Bezeichnung:
interaktives Multimedia, multimediales Hypermedia, hypermediales Multimedia, etc.

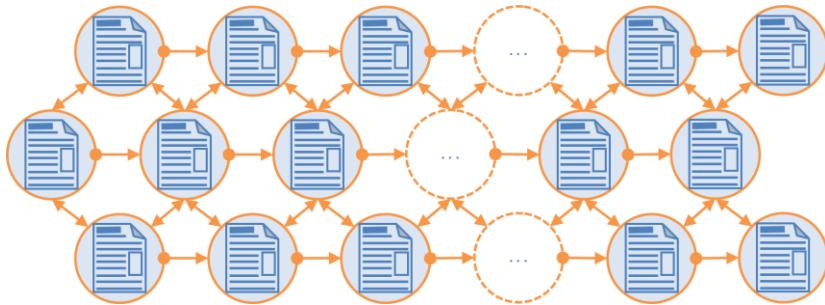
**Mit Hypermedia wird das Ziel verfolgt, ...
... zusammengehörige Informationen zu strukturieren und zu vernetzen.**

- Vergleich mit der einführenden Definition von Hypermedia
- Zudem werden verschiedene Unterziele verfolgt:
 - Aufbrechen „klassischer“ linearer Inhaltsstrukturen
 - Verknüpfung von verschiedenen Interaktionsknoten durch Einbettung von Links
 - Wandel, weg vom (passiven) Leser hin zum (Interaktiven) Benutzer(
 - Stärkere Auseinandersetzung der Benutzer mit dem Inhalt durch die Aktivierung
 - Höhere Immersion in die aufgespannte Informationswelt

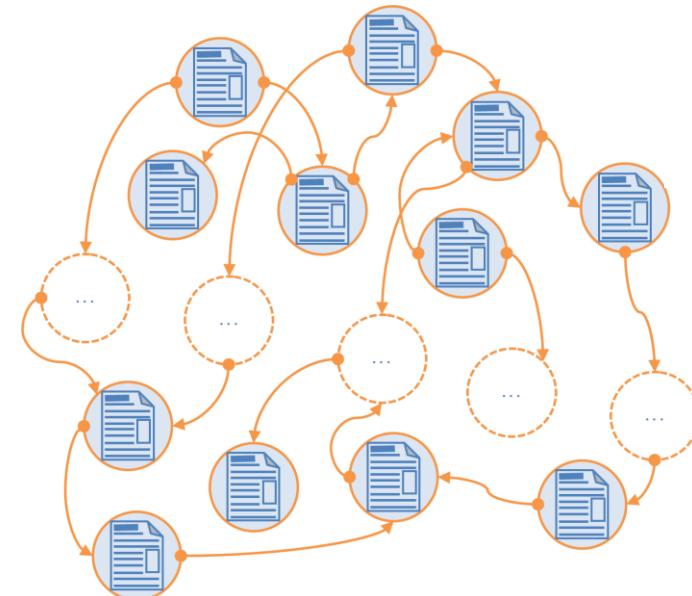
Inhaltsstrukturen in Hypermedia (Beispiele)



Einfache lineare Struktur



Parallele lineare Struktur



Nichtlineare Netzstruktur

Hoffmann, Beyond Hypertext (2020)

Hypermedia

Ausprägungen von Hypermedia

- **Statische Hypermedien:** sind solche Hypermedien, deren Führungsmedium sich während des Zeitraums des Betrachtens oder Bearbeitens nicht ändert.
- **Dynamische Hypermedien:** sind solche Hypermedien, deren Führungsmedium sich während des Zeitraums des Betrachtens oder Bearbeitens ändern können.
- **Kombinierte Hypermedien:** sind solche Hypermedien, die sich sowohl aus sich zeitlich ändernden als auch aus sich zeitlich nicht ändernden Medien bilden, ohne das es ein eigenliches erkennbares Führungsmedium gibt.
- Als **Führungsmedium** im Umfeld von Multi-/ Hypermedia kann das Medium bezeichnet werden, an dem der Benutzer sich auf seinem Weg durch die vernetzten Informationen orientieren kann und die durch den Ablauf der Präsentation führt.
In dieses Medium sind alle anderen Medien eingebettet, führen aber stets zu dem Führungsmedium zurück

Hypermedia

Statisches Hypermedia

▪ Hypertext

- „Links“ sind als integrierter Teil in den Text eingebaut
- Der Link kann durch Angaben zu Farbe und Format und Farbe erweitert werden

```
<html>
  <head>
  </head>
  <body>
    <h1>Gliederungsstufe 1</h1>
    <p>
      Lorem ipsum <a href="www.fom.de">dolor</a> sit
      amet, consetetur sadipscing elitr, sed diam
      nonumy <a href="foo.html">eirmod</a> tempor
      invidunt ut labore et dolore magna aliquyam
      erat, sed diam voluptua.
    </p>
  </body>
</html>
```

▪ Hyperimage

- bekannteste Implementierung: [HTML-Imagemap](#)
- in HTML-Imagemaps wird der Link durch Angaben der Position und der Form erweitert

• Probleme:

- Der Link wird in HTML-Imagemaps nicht dargestellt
- Wird die Größe des Bildes verändert, muss die HTML-Imagemap händisch angepasst werden

• Lösung:

- Responsive Imagemaps mit SVG (→ Abschnitt 5)

```

<map name="Voegel">
  <area shape="circle" coords="375,65,25" href="www.zoo.de" >
</map>
```

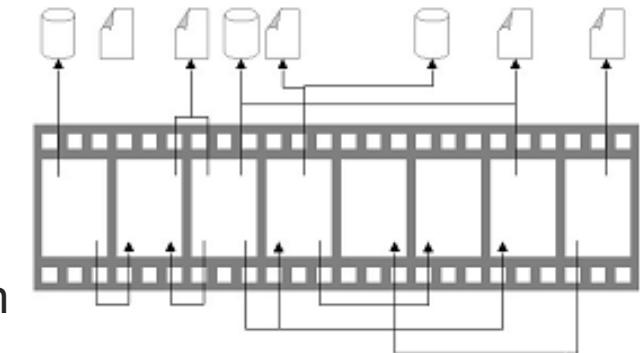


Hoffmann, Beyond Hypertext (2020)

Dynamisches Hypermedia

▪ Hypervideo

- Links werden in Filme/ Videoclips eingebettet
- Derzeit keine anerkannten Standards in Programmier-/ Gestaltungssprachen
→ Proprietäre-/ Einzellösungen
- Ein Link wird in mehr als einem Frame angezeigt
 - Wann erscheint er? Startzeit/ Startframe!
 - Wie lange „lebt“ er? Endzeit/ Dauer?
- Ein Link bezieht sich oftmals nicht auf das gesamte Frame sondern nur auf einen Ausschnitt
 - Auf welches Objekt/ welche Person bezieht er sich?
 - Wie verhält sich der Link, wenn das Objekt, auf den sich der Link bezieht, seine Position oder Form verändert?
- Ein Link sollte nicht versteckt sein und sich vom Hintergrund abheben
 - Wie sieht der Link aus?
 - Form und Farbe

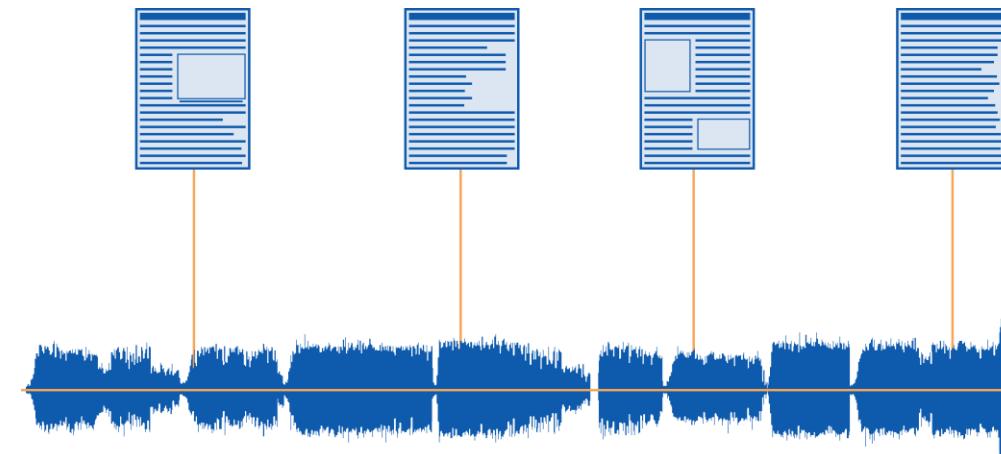


Hoffmann, Beyond Hypertext (2020)

Dynamisches Hypermedia

▪ Hyperaudio

- Sonderfall
- Derzeit keine anerkannten Standards in Programmier-/ Gestaltungssprachen
 - Proprietäre-/ Einzellösungen
 - Beispiel: Donker, H. & Blenn, N., (2007). Gestaltung von Hyperlinks in einer Hyperaudio-Enzyklopädie. In: Gross, T. (Hrsg.), Mensch und Computer 2007: Interaktion im Plural. München: Oldenbourg Verlag. (S. 139-148).
- Besondere Herausforderung: Gestaltung der Interaktion mit den „Links“



Hoffmann, Beyond Hypertext (2020)

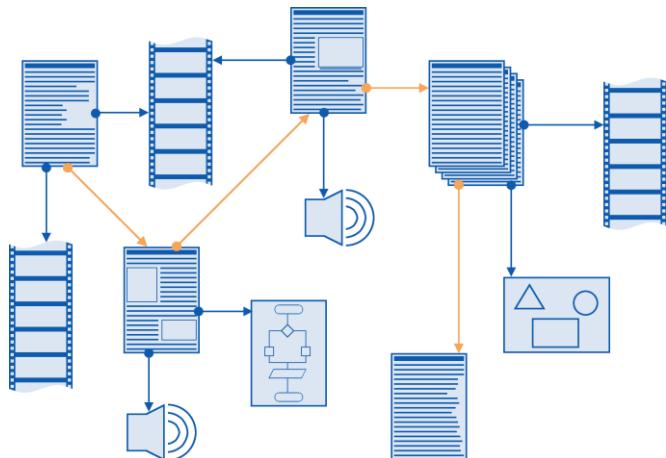
Kombiniertes Hypermedia

Mit Führungsmedium

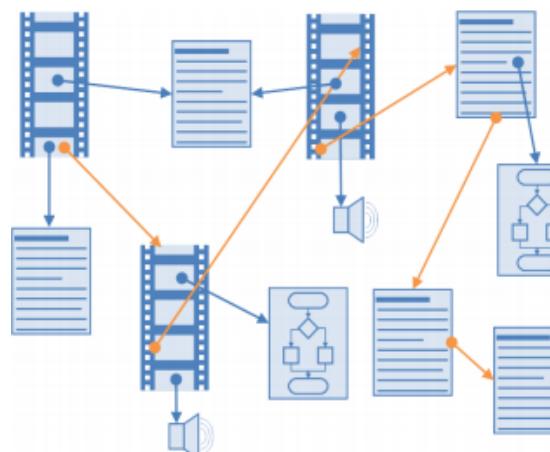
- Links, die einen Handlungs-/ Inhaltsstrang vorantreiben, befinden sich alle im gleichen Medientyp
→ Führungsmedium

Ohne Führungsmedium

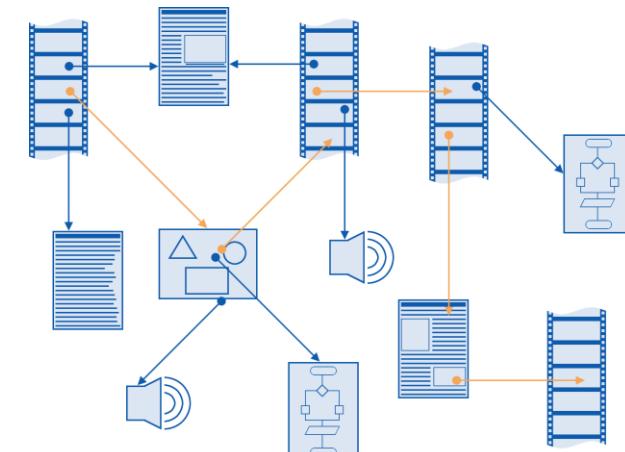
- Links, die einen Handlungs-/ Inhaltsstrang vorantreiben, können sich in unterschiedlichen Medientypen befinden



Kombiniertes Hypermedia
(mit Text als Führungsmedium)

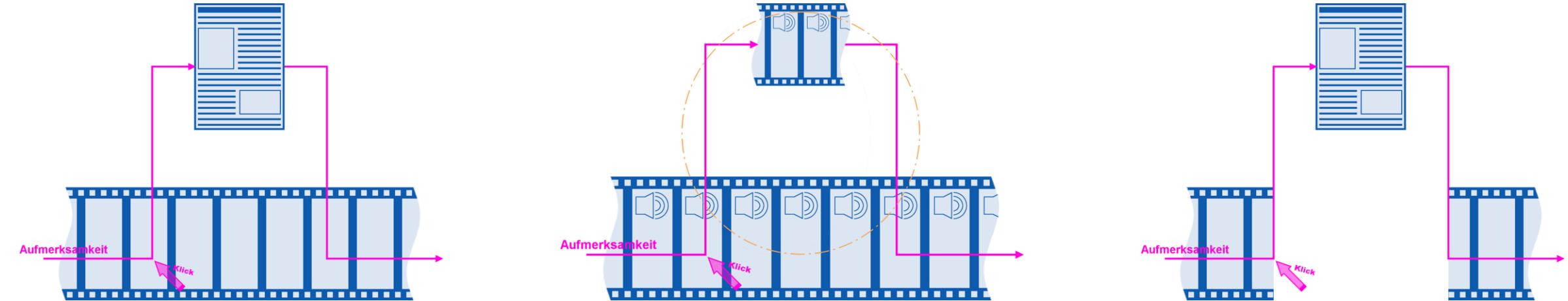


Kombiniertes Hypermedia
(mit Film als Führungsmedium)



Kombiniertes Hypermedia
(ohne Führungsmedium)

Besonderes Problem bei dynamischem und kombiniertem Hypermedia: Aufmerksamkeitsfokus des Benutzers



Hoffmann, Beyond Hypertext (2020)

- 1. Grenzen Sie Hypermedia von Multimedia ab.**

- 2. Beschreiben Sie die Problembereiche, die bei der Gestaltung einer multimedialen (Web-) Anwendung berücksichtigt werden müssen.**

- 3. Welche Probleme können sich auf Seiten des Benutzers durch ein hohes Maß von**
 - Multimedia**
 - Hypermedia****einstellen?**

Mensch-Maschine-System (MMS)

Mensch-Maschine-Interaktion (MMI)

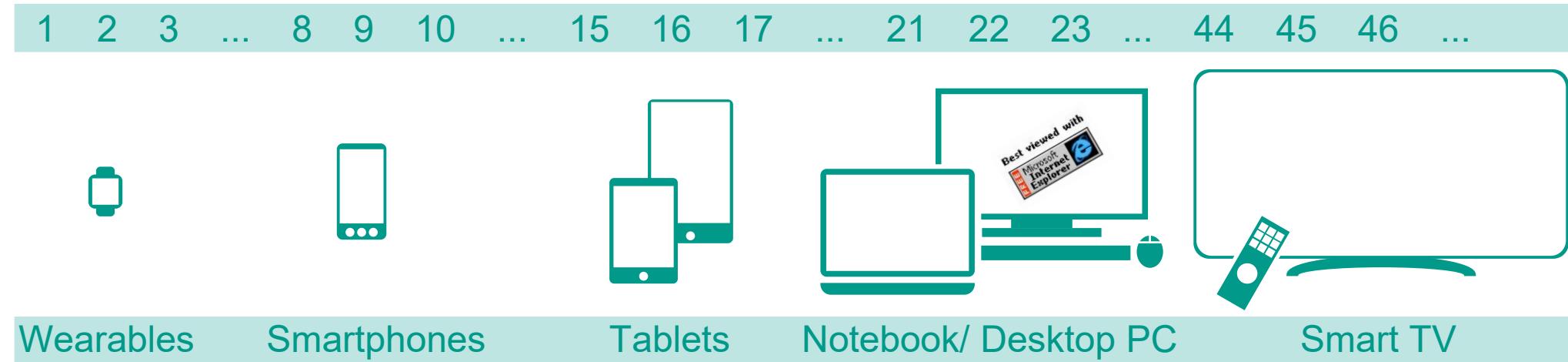
- MMI liegt immer vor, wenn menschliche Arbeit zur Benutzung einer Maschine notwendig ist
- → Arbeit = zielgerichtetes Handeln
 - vorgegebene Aufgabenstellung und
 - daraus resultierende Aufgabenerfüllung
 - → **MMI ist somit die Verbindung zwischen Aufgabenstellung und -erfüllung**

Human Computer Interaction (HCI)

- "Spezialfall" der MMI: die Interaktion zwischen Mensch und Computer
- bei HCI muss bei dem Begriff "Interaktion" unbedingt die gegenseitige Beeinflussung der Interaktionspartner betrachtet und berücksichtigt werden
- Die Interaktion zwischen Mensch und Computer wird aus verschiedenen Perspektiven betrachtet
 - Perspektive der Informatik
 - Perspektive (Kognitions-) Psychologie/ -wissenschaft
 - Perspektive der Soziologie
- HCI gliedert sich in drei Teile
 - der Anwender: ein einzelnes Individuum oder eine Gruppe, die mit einer Aufgabe beschäftigt ist und diese mit Hilfe der Technologie erledigt
 - der Computer: das vom Anwender benutzte Arbeitsmittel
 - die Interaktion: jede Kommunikation zwischen dem Anwender und dem Computer
 - direkte Interaktion: z.B. Dialog mit Feedback
 - indirekte Interaktion: z.B. eine angestoßene Hintergrundverarbeitung

Human Computer Interaction (HCI)

- Auswahl und Gestaltung der Interaktion ist abhängig von dem zum Einsatz kommenden technischen Arbeitsmittel

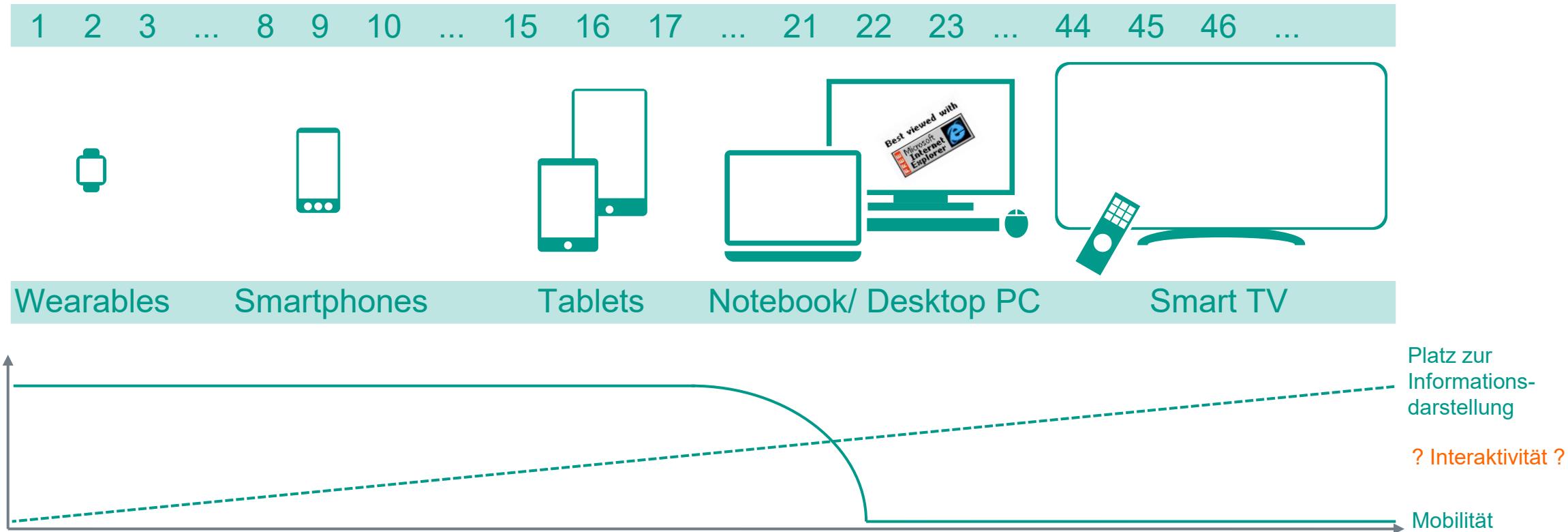


- Internet of Things (IoT)?



Human Computer Interaction (HCI)

- Auswahl der zum Einsatz kommenden technischen Arbeitsmittel hat unmittelbaren Einfluss auf die Möglichkeiten zur Gestaltung der Interaktion



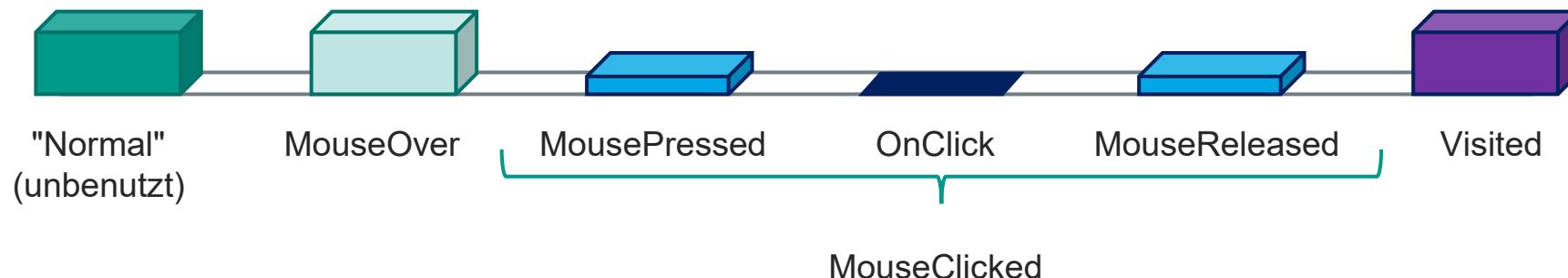
Interaktion: Paradigmen der Human Computer Interaction (HCI)

Art der Interaktion	Wichtigste Vorteile	Wichtigste Nachteile	Anwendungsbeispiele
Direkte Manipulation	<ul style="list-style-type: none"> • schnelle Interaktion • leicht erlernbar 	<ul style="list-style-type: none"> • Komplexe Implementierung • klare Beziehung zwischen dargestellten Elementen/ Informationen 	<ul style="list-style-type: none"> • Videospiele • CAD-Systeme • (industrielle) MSR-Anwendungen
Menüauswahl	<ul style="list-style-type: none"> • verhindert Fehler • wenig tippen 	<ul style="list-style-type: none"> • (für erfahrene Benutzer) Langsam 	<ul style="list-style-type: none"> • Viele allgemeine Systeme
Eingabemasken	<ul style="list-style-type: none"> • einfache Eingabe • leicht erlernbar • überprüfbar 	<ul style="list-style-type: none"> • viel benötigter Platz auf dem Bildschirm 	<ul style="list-style-type: none"> • Bestandskontrollen • Banken
Befehlssprache	<ul style="list-style-type: none"> • leistungsfähig • (bedingt) flexibel 	<ul style="list-style-type: none"> • schwer zu erlernen • Fehlermanagement 	<ul style="list-style-type: none"> • Betriebssysteme • (industrielle) MSR-Anwendungen
Natürliche Sprache	<ul style="list-style-type: none"> • Zugänglichkeit • leicht erlernbar 	<ul style="list-style-type: none"> • Fehlerkorrektur • Zuverlässigkeit 	<ul style="list-style-type: none"> • Informationsabruf

Interaktion

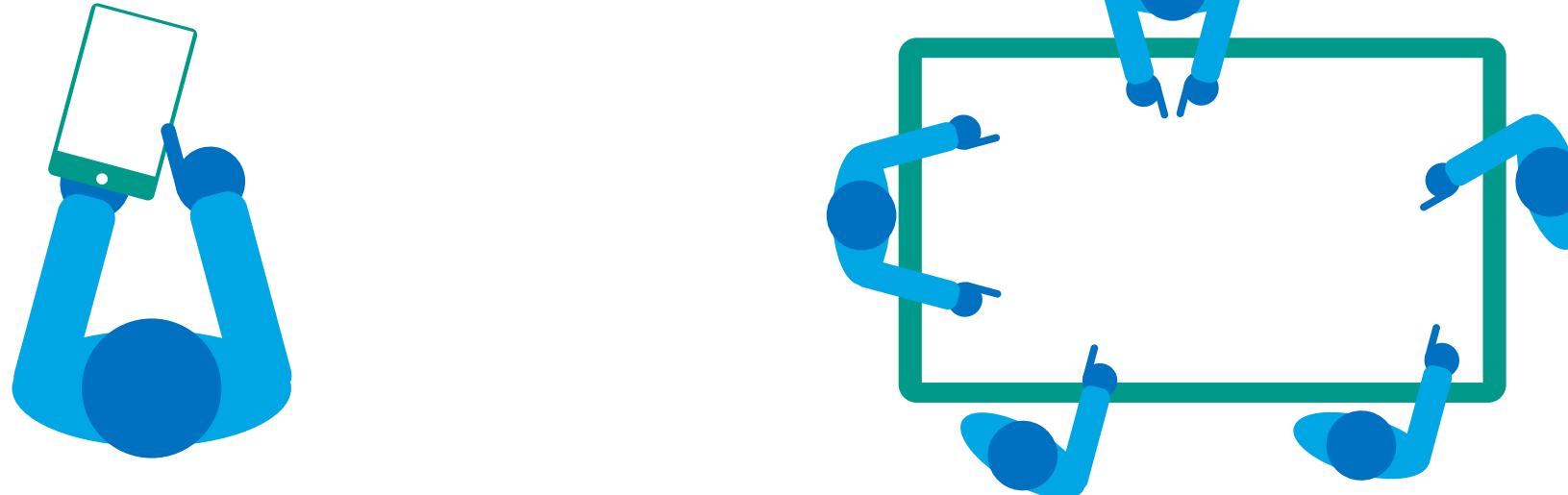
Technische Realisierung der Interaktion

- Die technische Realisierung von Interaktion ist abhängig von
 - der Wahl des Interaktionsparadigmas
 - der Wahl der eingesetzten Technik/ Geräte
 - den Anwendern/ Benutzern
- Beispiel: Mouse vs. Touch
 - beide Interaktionstechniken sind Abbildungen von Zeigegesten
 - der "Touch" stellt sich jedoch technisch anders dar als der Mausklick



Technische Realisierung der Interaktion

- Die technische Realisierung von Interaktion ist abhängig von
 - der Wahl des Interaktionsparadigmas
 - der Wahl der eingesetzten Technik/ Geräte
 - den Anwendern/ Benutzern
- Beispiel: Multitouch ist nicht gleich Multitouch
 - Multitouch auf kleinen Geräten wird i.d.R. anders genutzt als auf großen Geräten
→ Multiuser



FOM Hochschule

Web Technologie

(5) Statische Webseiten

1	Formales, Vorstellung
2	Einführung
2.1	Allgemeine Einführung
2.2	Historie Internet, Web & Web-Technologien
3	(Web) Server – Client Kommunikation
3.1	Protokolle & Architekturen
3.2	Web- & Microservices
3.3	Authentifizierungs- und Authorisierungskonzepte
4	Grundlagen der Mediengestaltung
4.1	Media & Multimedia
4.2	Media & Modalitäten – Interaktion & Wahrnehmung
4.3	Hypermedia
5	Statische Webseiten
5.1	HTML
5.2	CSS
5.3	Medieneinbindung
6	Dynamische Webseiten
6.1	DOM, HTML & CSS
6.2	JavaScript & JSON
6.3	Weitere Konzepte & Sprachen der Webprogrammierung
7	Praxis
7.1	Vorstellung eines (ausgewählten) (JavaScript-) Frameworks
7.2	Vorstellung einer (ausgewählten) Thematik für ein eigenes Projekt
7.3	Betreute, eigenständige Entwicklung des (ausgewählten) Themas
8	Wiederholung (opt.)

Im Anschluss an diesen Themenblock sollen Sie ...

- ... definieren können, was eine statische Webseite ist.
- ... HTML und CSS als die technischen Grundlagen für die Gestaltung statischer Webseiten kennengelernt haben und ...
- ... diese grundlegend anwenden können.
- ... Medien in (statische) Webseiten einbinden können.



Eine Website mit 1 Million Besuchern ...

- ... hat einen Marktanteil > 0,1% (eher noch deutlich kleiner)
- Ein Besucher bleibt durchschnittlich zwischen 25 bis 35 Sek. auf einer „Homepage“
- 60% aller Besucher gehen nicht weiter als zur ersten Seite einer Website

Der Betrachter **sollte** die Website

- ... schnell, ...
- ... leicht und ...
- ... verständlich ...
- ... **bedienen** können.



Der Betrachter **muss** die Website

- ... schnell, ...
- ... leicht und ...
- ... verständlich ...
- ... **verstehen und bedienen** können!

Eine Website mit 1 Million Besuchern ...

- Neben der Medien- und Interaktionsgestaltung wichtig: Die Gestaltung der **Inhaltsstruktur**
→ **Intuitive, narrative Navigation**
 - **Seite 1**
 - Worum geht es?
 - Wie funktioniert es?
 - Wie ist der Aufbau?
 - → Das Gefühl vermitteln, dass es sich lohnt, „zu bleiben“.
 - **Und grundsätzlich**
 - keine zu tiefen Hierarchien
 - Faustformel: „Mit 3 Klicks zum Ziel“

Seitenstruktur & Bestandteile

The screenshot shows the homepage of the FOM Hochschule website. The layout includes:

- Logo:** FOM Hochschule logo in the top right corner.
- Hintergrund:** A large dark blue vertical bar on the left side.
- Subnavigation 1 (variabel):** A sidebar on the left containing links like "Die FOM", "Studieninteressierte", "Studiengänge", etc., and accreditation information from FIBAA and WR.
- Header (Hauptnavigation):** A horizontal navigation bar at the top with links for FAQ, Kontakt, Presse, Mediathek, Newsletter, Impressum, Sitemap, and search.
- Hier studiere ich.:** A large headline with a portrait of a woman.
- Bachelor-Studium neben dem Beruf.:** Text about part-time study.
- Präsenzstudium für:** Categories: Berufstätige, Berufstätige mit Hochschulabschluss, (Fach-) Abiturienten und Schüler.
- Hauptinhalt (variabel):** The main content area displaying study programs like Bachelor of Arts (B.A.) in Banking & Finance, Business Administration, etc., and Master's programs like M.A. in Business Administration (MBA), SPEZIA, etc.
- Subnavigation 2 (meta):** A sidebar on the right listing study locations across Germany and Luxembourg.

Document Object Model

DOM - Document Object Model

- beschreibt alle Objekte einer HTML-Seite
 - Hierarchie der Objekte
 - Zugriff auf die Objekte
- vereinheitlicht als Standard des [W3C](#)
 - **Eigenschaften:**
Zugriff auf bestimmte Elemente der Seite
(als Unterobjekte des document-Objekts)
 - **Methoden**
Zugriff, Be-/ Verarbeitung und Veränderung der Objekte und ihrer Eigenschaften
→ wichtigste Methode: „`getElementByID`“
(wird im Abschnitt „Dynamische Webseiten“ im Detail betrachtet)

DOM - Objekthierarchie

window



document



node

```
alert( "foo");  
window.alert( "foo" );  
  
window.resizeTo( x, y )  
  
window.event.clientX  
window.event.clientY  
  
window.event.PageX  
window.event.PageY
```

```
document.write("<p>Hallo!<br>" );  
document.documentElement.clientHeight
```

„A node is the generic name for any type of object in the DOM hierarchy. An element is a specific type of node. It can be directly specified in HTML with an HTML-tag and it can have properties like an id or a class.“

Seitenstruktur & Positionen

- Window-Fixed
 - alle Elemente liegen auf einem Layer an jeweils eigenen Positionen
→ beim Scrollen bewegen sich alle Elemente parallel
- Position-Fixed
 - Elemente wie z.B. Header, Navigationen, o.ä. sind auf dem obersten Layer an einer festen Position fixiert
→ bewegen sich beim Scrollen nicht
 - Inhalte sind auf tieferen Layern angeordnet
→ scrollbar „hinter“ den Inhalten der höheren Layer
- „Blogging-Design“
 - „Neue Inhalte vor alten Inhalten“
 - „intuitive“ Sortierung (Online-Shops, „News“-Inhalte, etc.)

Seitenstruktur & Positionen

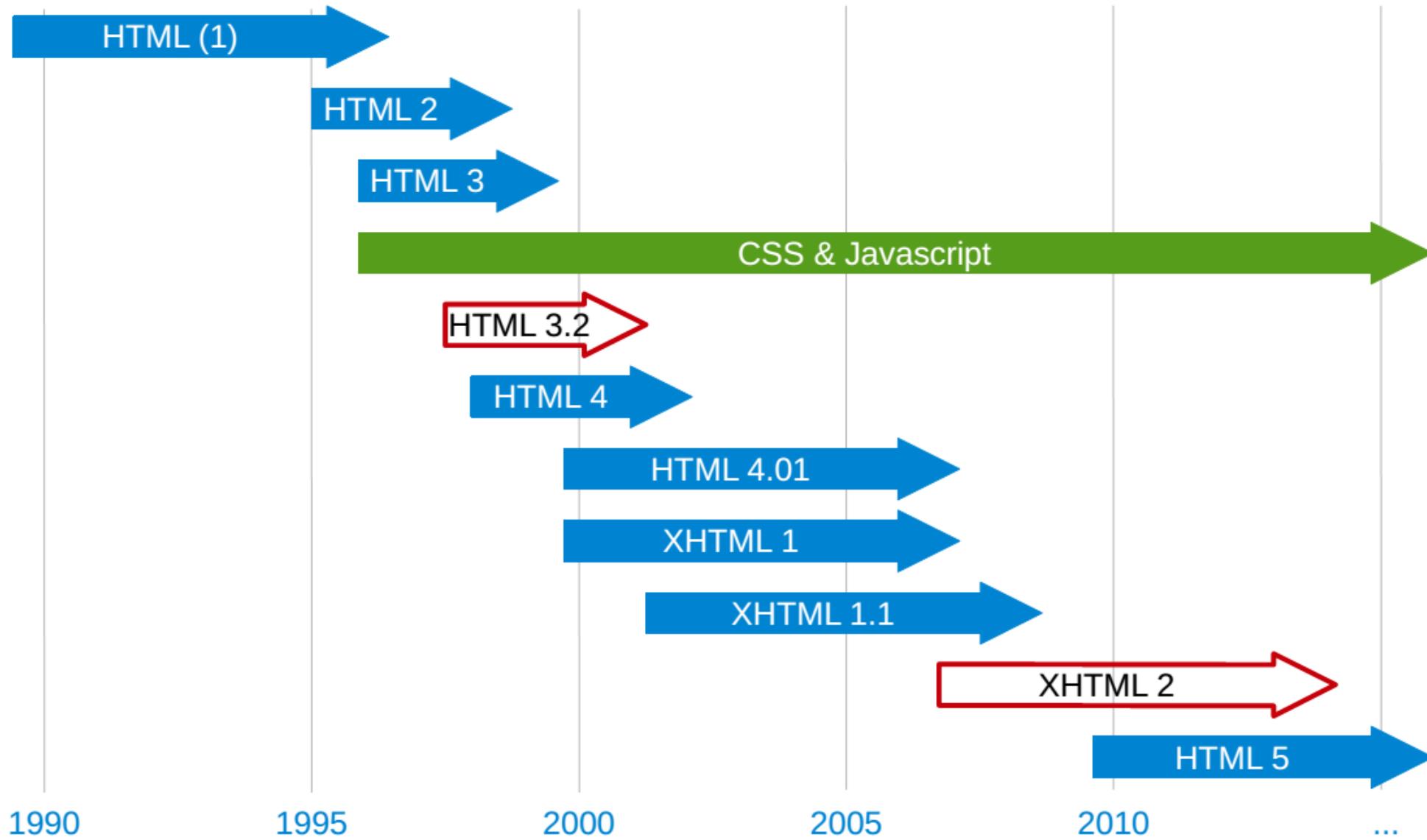
- Responsive Design
 - Anordnung der Elemente passen sich in Position (und Layer) an die Geräte“größe“ an
 - Aufgeben des Prinzips „schwächstes Glied“, bei dem die kleinsten Geräte die Implementierung vorgeben
 - Einfachste (alte, bekannte) Implementierung: Webweiche mit HTML und CSS

HTML:

```
<link rel="stylesheet" type="text/css" href="css/main.css" />
<link rel="stylesheet" type="text/css" media="all and ( max-device-width: 320px )"
      href="css/small.css" />
```

CSS:

```
a{
    color: #0000ff; }
@media( max-device-width:320px; ){
    a{
        color: #00ff00; } }
```



HTML 3.x → HTML 4.1 → XHTML → HTML 5

- Und jetzt?
„Nur“ noch **HTML5** lernen?
- „Historische“ Techniken (und Tags) sind noch weit verbreitet !
 - Wichtig ist: Bei allen Beispielen im Internet auf die jeweilige Version achten!
 - **gestern empfehlenswert** → **heute veraltet (oder falsch)**

HTML - Syntax

Auszeichnungen („Tags“)

- Tags sind keine Befehle
 - Auszeichnungen zur Interpretation der Inhalte/ Informationen
 - „Anreicherung der Information mit Erklärungen zur Verarbeitung“
 - Allgemeine Syntax: <tag>...</tag>
Kurzform: <tag />

HTML 4

Ein **Satz** mit *unterschiedlichen* Auszeichnungen.

Ein **Satz** mit *unterschiedlichen* Auszeichnungen.

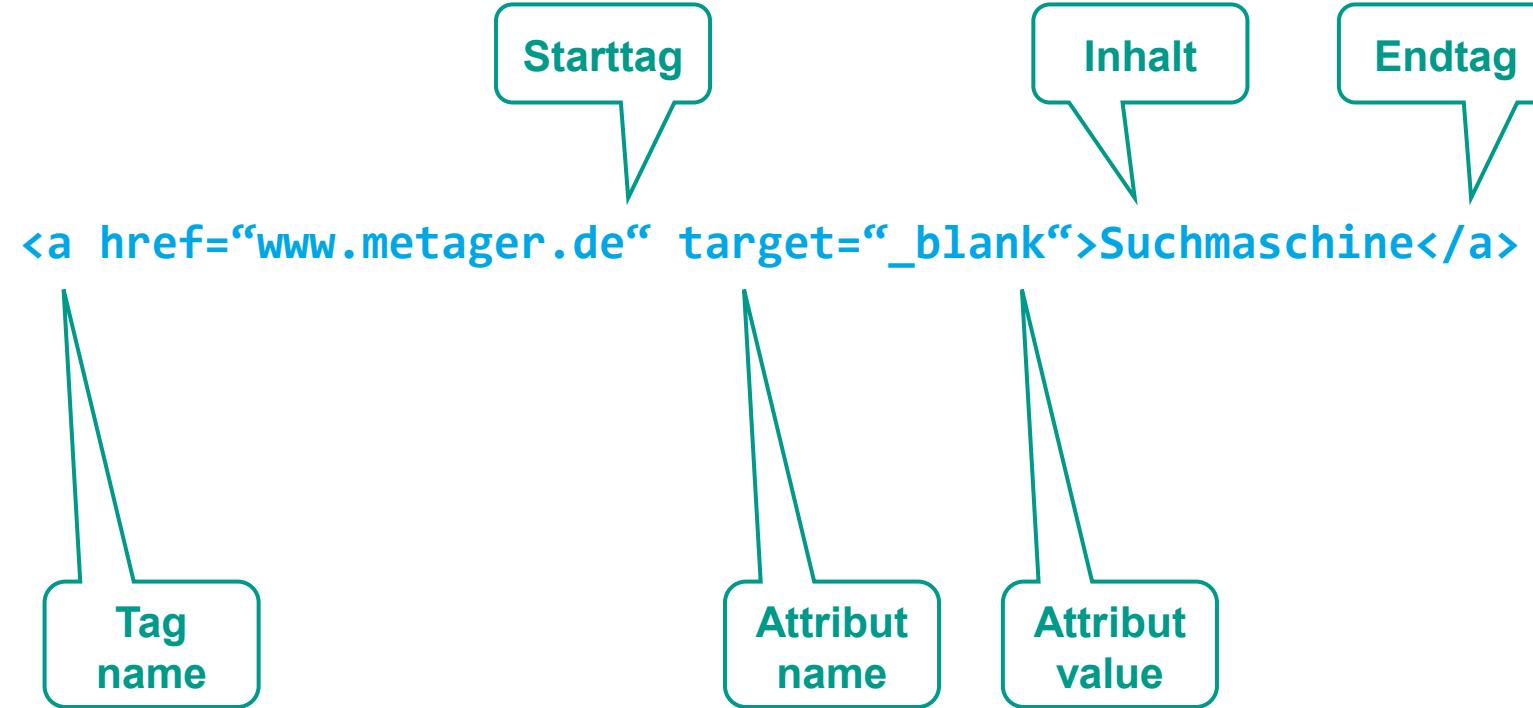
HTML 5

Ein **Satz** mit *unterschiedlichen* Auszeichnungen.

Grundgerüst (der Seitenstruktur)

```
<html>  
  
  <head>  
    <!-- Meta-Informationen zur Seite -->  
    <title>...</title>  
  
  </head>  
  
  <body>  
    Hallo World!  
  
  </body>  
  
</html>
```

Tags, Attribute und Werte



Übersichten, Dokumentationen, Playgrounds

- SelfHTML: <https://wiki.selfhtml.org/>
- W3Schools: <https://www.w3schools.com/html>
- HTML5rocks
→ Web Fundamentals <https://developers.google.com/web>

HTML - Vorüberlegungen

Werkzeuge

- Keine Internetverbindung notwendig
 - HTML-Dateien sind einfache Textdateien
 - Texteditor reicht
- Browser
 - zum Rendering

Planung !!!

- wie bei jedem Projekt!
 - Was soll in das Webprojekt (oder dessen Ergebnis) dem Surfenden anbieten`?
 - Texte, Bilder, Filme, Tonaufnahmen ...
 - ... Struktur!
 - Zielgruppe!

```
<!doctype html>
<html lang="de">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Beschreibung</title>
  </head>
  <body>
    <p>Hallo Welt!</p>
  </body>
</html>
```

- Doctype: bei HTML5 eigentlich nicht mehr notwendig, in HTML4 gab es noch mehrere andere Doctypes
- Head
 - Zeichencodierung: üblicherweise UTF-8
 - Viewport: das Browserfenster
hier: Angabe für die Nutzung der Bildschirmbreite ohne Skalierung
- Body: der Seiteninhalt

HTML – Kopf Inhalte

- Informationen für nichtmenschliche Besucher einer Internetseite
- erlauben die korrekte Darstellung des sichtbaren Teils des Dokuments
- Elemente, die ausschließlich im Kopfbereich erlaubt sind:
 - **base** Basis für relative Verweise
 - **title** Seitentitel
 - **link** Verlinkung zu anderen Dokumenten im gleichen Web-Space
 - **meta** zusätzliche Angaben, z.B. Dublin-Core
- Elemente, die sowohl Kopfbereich als auch im body erlaubt sind:
 - **style** CSS-/ Stylesheet-Angaben
 - **script** Skripte und Skript-Anweisungen
 - **noscript** Alternative, falls ein Skript nicht ausgeführt werden darf/ kann

HTML – Seitenstruktur

- **body** gesamter Inhaltsbereich
- **main** Hauptinhalt
- **header** einleitende Inhalte, Orientierungshilfen
- **footer** informative Inhalte etwa Impressum, Copyright, Autor
- **nav** Seitennavigation
- **article** einzelne in sich geschlossene Inhaltsbereiche
- **section** thematische Gruppierung von Inhalten
- **aside** ergänzende Hinweise, „Randbemerkungen“
- **address** Kontaktinformationen

HTML – Seitenstruktur

Tags, Attribute und Werte → Einfluss auf die Seitenstruktur

Viel Text ist schwierig zu lesen. Nicht nur im Netz, sondern auch in *<h1>Büchern</h1>* werden deshalb Überschriften benutzt, um lange Textstücke zu gliedern und übersichtlicher zu machen!

<h1>Das ist sinnvoll</h1>

Gegliederte Texte werden nicht nur inhaltlich, sondern auch visuell erinnert!

Allerdings lässt es sich manchmal nicht vermeiden, dass Textstücke lang bis sehr lang werden, wenn sie sich innerhalb eines Themenkontextes bewegen.

Überschriften sind dann oftmals nicht hilfreich, sinnvoll oder logisch.

<p>Weitaus besser sind dann Absätze geeignet, um Textwüsten zu vermeiden und lange Texte zu gliedern.</p>

Manchmal sind allerdings auch Absätze ungeeignet zur Gliederung.

Dann kann eventuell ein Zeilenumbruch hilfreich sein.

Insbesondere in der deutschen Sprache gibt es ein besonderes Problem durch die

Möglichkeit zur Wortneubildung: der Wortverknüpfungs-

fetischismus. Solche Wortneuundlangschöpfungskonstruktionen durch-

brechen den ästhetischen Textfluss, der von Browsern versucht wird zu implementieren.

<body>

Foto & Video Hammelmeir

Willkommen im Fotostudio Hammelmeir!

Wir sind seit 17 Jahren darauf spezialisiert,
alle Kundenwünsche zu erfüllen. In unseren
Studios produzieren wir selbst - alle Fotos
und Filme nach Ihren Wünschen.

Unsere Leistungen:

Fotos nach Wunsch

Bewerbung

Hochzeit

Portrait

Filme nach Wunsch

Aufbereitung von Reisefilmen

Werbefilme

Lehrfilme

Multimedia

Kommen Sie vorbei - es wird bunt!

</body>

- Überschriften:

6 Gliederungsebenen h1, h2, ... h6

<h1> ... </h1>

- „normaler“ Text

funktioniert i.A. auch ohne besondere Auszeichnung
üblicherweise aber als Absatz (= Paragraph)

<p> ... </p>

- p
- pre
- blockquote
- figure
- figcaption

Textabsätze

präformatierter Text

umfangreiche Zitate

grafische Ergänzungen, z.B. Bilder,
Diagramme, Code-Beispiele, usw.

Beschreibung des Inhalts eines
figure-Elements

- hr

thematischer Bruch
(durch trennende Linie)

- div

gruppierendes Element ohne
semantische Bedeutung → CSS

<body>

Foto & Video Hammelmeir

Willkommen im Fotostudio Hammelmeir!

Wir sind seit 17 Jahren darauf spezialisiert,
alle Kundenwünsche zu erfüllen. In unseren
Studios produzieren wir selbst - alle Fotos
und Filme nach Ihren Wünschen.

Unsere Leistungen:

Fotos nach Wunsch

Bewerbung

Hochzeit

Portrait

Filme nach Wunsch

Aufbereitung von Reisefilmen

Werbefilme

Lehrfilme

Multimedia

Kommen Sie vorbei - es wird bunt!

</body>

- Aufzählungen/ Listen

nummerierte und nicht nummerierte Listen

- Unordered list: ...

- Ordered list: ...

- Jeder Listeneintrag ist ein Item: ...

- Alle Listen können ineinander verschachtelt werden

- „Beschreibungs“listen: <dl> ... </dl>

- **dt** Ausdruck, der in erläutert werden soll

- **dd** die Erläuterung

1. Übertragen Sie die Seite von Foto Hammelmeir in „richtiges“ HTML

<body>

Foto & Video Hammelmeir

Willkommen im Fotostudio Hammelmeir!

Wir sind seit 17 Jahren darauf spezialisiert,
alle Kundenwünsche zu erfüllen. In unseren
Studios produzieren wir selbst - alle Fotos
und Filme nach Ihren Wünschen.

Unsere Leistungen:

Fotos nach Wunsch

Bewerbung

Hochzeit

Portrait

Filme nach Wunsch

Aufbereitung von Reisefilmen

Werbefilme

Lehrfilme

Multimedia

Kommen Sie vorbei - es wird bunt!

</body>

```
<body>
```

Foto & Video Hammelmeir

Unsere Preise:

Bewerbungsfotos

3 Abzüge von je zwei Positionen: 48€

2 Abzüge von je drei Positionen: 64€

4 Abzüge von einer Position: 32€

Hochzeitsfotos

Begleitung halber Tag und 10 Fotos: 240€

Begleitung ganzer Tag und 60 Fotos: 640€

Portrait

3 Abzüge in unterschiedlicher Größe: 100€

3 Abzüge in gleicher Größe: 80€

2 Abzüge auf kleiner Leinwand: 100€

je 1 Abzug auf kleiner und großer
Leinwand: 120€

Kommen Sie vorbei – es wird bunt!

```
</body>
```

- Das Grundgerüst einer Tabelle

```
<table>
```

```
  <thead>
```

```
    <tr>
```

```
      <th>...</th>
```

```
    ...
```

```
  </tr>
```

```
</thead>
```

```
<tr>
```

```
  <td>...</td>
```

```
  ...
```

```
</tr>
```

```
<tfoot>
```

```
  <tr>
```

```
    <td>...</td>
```

```
  </tr>
```

```
</tfoot>
```

```
</table>
```

<body>

Foto & Video Hammelmeir

Unsere Preise:

Bewerbungsfotos

3 Abzüge von je zwei Positionen: 48€

2 Abzüge von je drei Positionen: 64€

4 Abzüge von einer Position: 32€

Hochzeitsfotos

Begleitung halber Tag und 10 Fotos: 240€

Begleitung ganzer Tag und 60 Fotos: 640€

Portrait

3 Abzüge in unterschiedlicher Größe: 100€

3 Abzüge in gleicher Größe: 80€

2 Abzüge auf kleiner Leinwand: 100€

je 1 Abzug auf kleiner und großer
Leinwand: 120€

Kommen Sie vorbei – es wird bunt!

</body>

- Erweiterung der Auszeichnung von Tabellen

- Rahmen:

<table border="1">

- Zusammenführen von ...

- ... Reihen:

<td rowspan="2">

- ... Zeilen:

<td colspan="2">

- Breite von Zellen/ Spalten

<td width="25%">

→ Achtung: HTML4 vs. HTML5

- Textausrichtung in Zellen

- Horizontal:

<td align="left">

<td align="center">

<td align="right">

<td valign="top">

<td valign="middle">

<td valign="bottom">

- Vertikal:

- Viele Auszeichnungen werden üblicherweise im CSS definiert

→ CSS vs. Quick-and-dirty

Einfluss auf die Seitenstruktur: Tabellen sind **keine Strukturierungselemente!**

```
<table border="1">
  <thead>
    <tr>
      <td>Head</td>
      ...
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td>Foot</td>
      ...
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td>Inhalt</td>
      <td><table border="1"><tr><td>...</td></tr></table></td>
      ...
    </tr>
  </tbody>
</table>
```

Tabelle 1 Head	Tabelle 1 Head	Tabelle 1 Head
Tabelle 1 Inhalt	Tabelle 2 Tabelle 2 Tabelle 2 Tabelle 2 Tabelle 2 Tabelle 2	Tabelle 1 Inhalt
Tabelle 1 Foot	Tabelle 1 Foot	Tabelle 1 Foot

1. Übertragen Sie die Seite von Foto Hammelmeir in „richtiges“ HTML

<body>

Foto & Video Hammelmeir

Unsere Preise:

Bewerbungsfotos

3 Abzüge von je zwei Positionen: 48€

2 Abzüge von je drei Positionen: 64€

4 Abzüge von einer Position: 32€

Hochzeitsfotos

Begleitung halber Tag und 10 Fotos: 240€

Begleitung ganzer Tag und 60 Fotos: 640€

Portrait

3 Abzüge in unterschiedlicher Größe: 100€

3 Abzüge in gleicher Größe: 80€

2 Abzüge auf kleiner Leinwand: 100€

je 1 Abzug auf kleiner und großer
Leinwand: 120€

Kommen Sie vorbei - es wird bunt!

</body>

<body>

Foto & Video Hammelmeir

Unsere Preise:

Bewerbungsfotos

3 Abzüge von je zwei Positionen: 48€

2 Abzüge von je drei Positionen: 64€

4 Abzüge von einer Position: 32€

Hochzeitsfotos

Begleitung halber Tag und 10 Fotos: 240€

Begleitung ganzer Tag und 60 Fotos: 640€

Portrait

3 Abzüge in unterschiedlicher Größe: 100€

3 Abzüge in gleicher Größe: 80€

2 Abzüge auf kleiner Leinwand: 100€

je 1 Abzug auf kleiner und großer
Leinwand: 120€

Kommen Sie vorbei - es wird bunt!

</body>

- Erweiterung der Auszeichnung von Tabellen

- Tabellenüberschrift `<caption>...</caption>`
→ Position wird in CSS definiert

- Beschriftung von Tabellen /Objekten
`<figure>`
`[Object]`
`<figcaption>Beschriftung<figcaption>`
`</figure>`

- Zusammenfassung von Tabellen-/ Zelleninhalten
`<th abbr="In jeder">Zeile</th>`
`<td abbr="befinden sich">Zellen</td>>`
→ nützlich u.a. für Screenreader

- Verbindungen/ Referenzen innerhalb von Tabellen
`<th id=„name“>Name</th>`
...
`<td headers=„name“>Hammelmeir</td>`
→ nützlich u.a. für Screenreader: „Der Name lautet
Hammelmeir“

HTML – Textauszeichnung: Textstellen mit „besonderer“ Bedeutung

- Texthervorhebungen → CSS

- **b** üblicherweise fett
- **em**
- **i** üblicherweise kursiv
- **mark** aus Referenzgründen hervorgehobener Text
- **s** „irrelevante“ Inhalte
- **small** „nebensächliche“ Informationen
- **u** üblicherweise unterstrichen
- **sub** tiefgestellter Text
- **sup** hochgestellter Text
- **cite** Titel einer Publikation
- **q** Zitat im Fließtext
- **dfn** Definition bzw. zu definierender Begriff
- **abbr** Abkürzung

- Texthervorhebungen

- **code** Programm- oder sonstige Quelltexte
- **var** Variable, Konstante, Größe
- **samp** Ausgabe eines Programmes
- **data** maschinenlesbare Daten (keine Zeitangaben)
- **time** Zeitangabe, Datum, Uhrzeit
- **ruby** Aussprachehinweise (hauptsächlich für asiatische Schriften)
- **rt** Text, der zu einem ruby-Hinweis gehört
- **rp** Klammern um rp
- **bdi** Textteile mit anderer Textrichtung
- **bdo** Textrichtung überschreiben
- **br** expliziter Zeilenumbruch
- **wbr** Stelle für einen impliziten Zeilenumbruch
- **del** gelöschte Inhalte
- **ins** neu eingefügte Inhalte
- **span** auszeichnendes Element ohne semantische Bedeutung → CSS

- `Tatort`

- Attribute:

• <code>href</code>	Verweisziel
• <code>#</code>	Anker innerhalb einer Seite
• <code>mailto</code>	Aufruf des lokalen Mail-Clients
• <code>tel:</code>	Aufruf des lokalen Telefon-Clients
• <code>target</code>	Zielfenster
• <code>_blank</code>	neues Fenster/ Tab
• <code>_self</code>	aktueller Fenster/ Tab
• <code>_parent</code>	Elternfenster
• <code>_top</code>	oberstes Fenster
• <code><name></code>	Fenster mit eigenem Namen
• <code>title</code>	Tooltip
• <code>lang/ hreflang</code>	Sprachangabe für Start- und Zielfäilen
• <code>ping</code>	Aufruf einer zusätzlichen URL
• <code>rel</code>	logische Beziehung zwischen Verweis und Verweisziel
• <code>accesskey</code>	Ansteuerung des Verweises über Tastaturkürzel
• <code>tabindex</code>	Ansteuerung des Verweises über Tabulator-Taste

``
``
``

``

``

``

CSS – Cascading Style Sheets

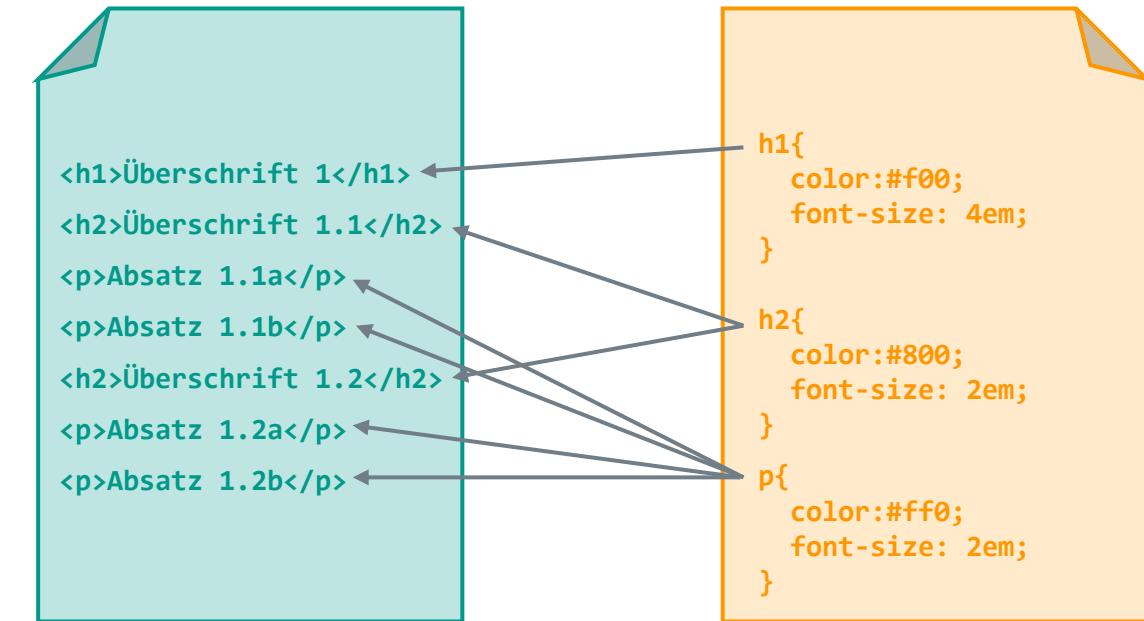
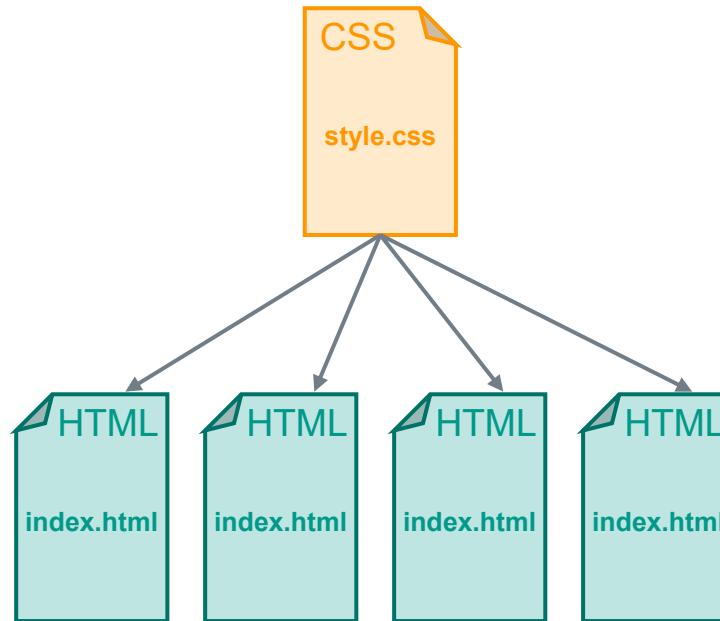
- 1994 Vorläufer Hakon Wium Lie: Cascading HTML Style Sheets (CHSS)
- 1996 CSS Level 1 W3C
- 1998 CSS 2 = CSS Level 2, W3C
(zu Beginn) keine Unterstützung durch damalige Browser
- 2000 Beginn der Entwicklung von CSS 3
- 2011 CSS 2.1 W3C, keine Funktionserweiterung ggü. CSS 2
- 2012 Beginn der Entwicklung von CSS 4
„There will never be a CSS 4“
→ wegen des modularen Aufbaus von CSS 3
- 2016 CSS 2.2 W3C, Arbeitsentwurf
- CSS 3 kein festes „Veröffentlichungsdatum“
→ modularer Aufbau mit separater (Weiter-)Entwicklung der einzelnen Module
 - CSS Color Level 3
 - CSS Namespaces
 - Selectors Level 3
 - Media Queries
 - weitere Module wie Layout-Modul, Grafikfilter, usw.

Ansprechendes und aufgabenorientiertes Layout

- Navigation
- Übersichtliche Gliederung der Seiten
- einheitliches (Firmen-) Layout → CI
- klare Formulare
- professionelles Erscheinen
- In reinem HTML schwierig zu erreichen und nicht sinnvoll
 - HTML-Seiten neigen zur Unübersichtlichkeit
 - HTML-Seiten schwer zu pflegen
 - HTML-Seiten sind statisch

Warum CSS?

- Trennung von Inhalt und Design
→ hohe Flexibilität, wenn das Layout oder die Struktur einer „Seite“ geändert werden muss
- „Am Anfang ohne Layout“
→ Fokus bei der Gestaltung erst auf der Seitenstruktur und den Inhalten



CSS – Einbinden

- Einbinden mit dem style-Attribut
 - direkte Zuweisung von Eigenschaften zu einem Element

```
<p style="text-align: center; color: green;">  
    Dieser Absatz wird über ein style-Attribut formatiert.  
</p>
```

- Einbinden mit dem HTML-Element style
 - Zentrale Festlegung von Eigenschaften im Kopfbereich der HTML-Datei

```
<html>  
    <head>  
        <style>  
            h1{  
                background-color: green;  
                color: white;  
            }  
            ...
```

CSS – Einbinden

- Einbinden externer Style-Sheet-Dateien
 - Normalfall
 - externes Style-Sheet-Dokument mit der Endung .css wird mit dem HTML-Element link direkt im Kopf eingebunden

```
<head>
  <link rel="stylesheet" href="stylesheet.css">
  ...

```

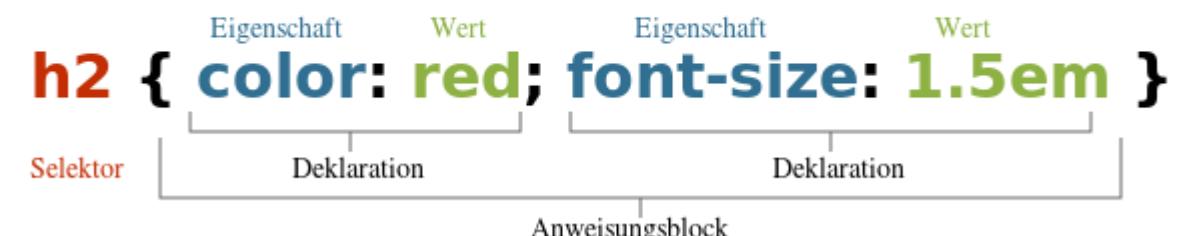
- link-Element ist ein Kindelement des head-Elements
 - Attribute
 - **rel** definiert, dass ein Element vom Typ „Stylesheet“ eingebunden werden soll.
 - **href** URL der Style-Sheetd-Datei
 - Zusätzliche Attribute
 - **type** Medientyp der einzubindenden Datei, für CSS also „text/css“ (optional, ohne Auswirkung)
 - **media** kann festlegen, dass ein Stylesheet nur benutzt werden soll, wenn das Dokument von einem bestimmten Ausgabemedium ausgegeben wird (screen, print, all)
 - **title** Definition von „Titeln“ für einzelne Style Sheets, wenn mehrere Style Sheets eingebunden werden

CSS – Syntax

- CSS Regeln und Regelsätze organisieren die zuzuweisenden Eigenschaften
- Regelsätze bestehen aus
 - dem Selektor/ einer Gruppe von Selektoren
 - durch geschweifte Klammern begrenzten Bereichen, in denen
 - Eine/ mehrere Deklarationen stehen, in denen Eigenschaften ein Wert zugewiesen wird.
- Eine Deklaration benötigt zwischen Eigenschaft und Wert einen Doppelpunkt
- Zwischen zwei Deklarationen steht ein Semikolon

```
h1{  
    color: red;  
}  
  
h1, h2{  
    background-color: #ccc;  
    border-radius: .5em;  
}
```

CSS-Regelsatz



Quelle: Selfhtml.org

CSS – Syntax

- „Shorthand“-Eigenschaften
 - Zuweisen mehrerer Werte für eine Eigenschaft
 - Trennung durch Leerzeichen

```
h1{  
    color: red;  
    font: 300% cursive;  
}
```

- Mehrfaches Vorkommen einer Eigenschaft
 - Fallback für ältere Browser

```
body {  
    background: skyblue;  
    background: linear-gradient(white, skyblue);  
}
```

Hier wird die untere Angabe von älteren Browsern ignoriert

- Kommentare

```
/* Dies ist ein  
mehrzeiliger Kommentar!*/
```

- Gleichheitszeichen bei Wertzuweisungen

```
h1{ color=red; }  
h2{ color:red; }
```

- das beliebte Semikolon ;-)

```
h1{ color:red background-color:yellow; }  
h2{ color:red; background-color:yellow; }
```

- die beliebten Klammern ;-)

```
h1{ color: red; background-color: yellow; }  
h2{ color: red; background-color: yellow; }
```

- Groß- und Kleinschreibung

```
#meldungen{ background: lightgreen; }
```

...

```
<body>  
<div id="Meldungen">  
...
```

- Leerzeichen
 - Semantische Bedeutung
 - Ungültige Werte

```
h2 span{  
    color: red;  
}  
h2, span{  
    background: #ccc;  
    border: 1px solid;  
}
```

- nur dem span-Element innerhalb von h2 wird Rot als Schriftfarbe zugewiesen

<h2>Leerzeichen sind nicht Nichts!</h2>

- der zweite Regelsatz fasst alle h2- und span-Elemente zusammen:
grauer Hintergrund und einen Rand in Schriftfarbe

- Selektoren dienen der exakten Auswahl und Formatierung von Elementen auf einer Webseite

- Typselektor

- weist allen Elementen eines Typs Eigenschaften zu

```
h2{ color: red; }
```

- id

- Wenn nur Teile einer Seite/ eines Textes formatiert werden sollen
 - jedem HTML-Element kann eine eindeutige id zugewiesen werden, über die das Element angesprochen wird

```
#warnung{ color: red; }
```

```
...
```

```
<div id="warnung">  
  <h2>Achtung!</h2>
```

- Klassen
 - Wenn zwar mehrere, aber nicht alle Teile einer Seite/ eines Textes formatiert werden sollen, ist der **class**-Selektor besser geeignet als **id**

```
.warnung{  
    color: red;  
}  
p.warnung{  
    border: 1px solid red;  
    background-color: peachpuff;  
}  
...  
<h2 class="warnung">  
    Dies sind einige Möglichkeiten, die Klasse <code>warnung</code>  
    auf verschiedene HTML-Elemente anzuwenden.  
</h2>  
<p class="warnung">Bitte die AGB lesen!</p>  
<p><strong class="warnung">Achtung:</strong> Bitte die AGB lesen!</p>
```

- Pseudoklassen
 - häufig genutzt zur Ansprache von Elementen aufgrund ihrer Position innerhalb der Seite (und nicht aufgrund der Semantik)
 - Ansprechen des ersten Elementes (z.B. bei Listen)

```
li{  
    background-color: lightgrey;  
}  
li:first-child{  
    background-color: yellow;  
}
```

- Ansprechen eines beliebigen Elementes (z.B. bei Listen)

```
li:nth-child(2){  
    background-color: grey;  
}  
li:nth-child(3){  
    background-color: darkgrey;  
}
```

CSS – Selektoren

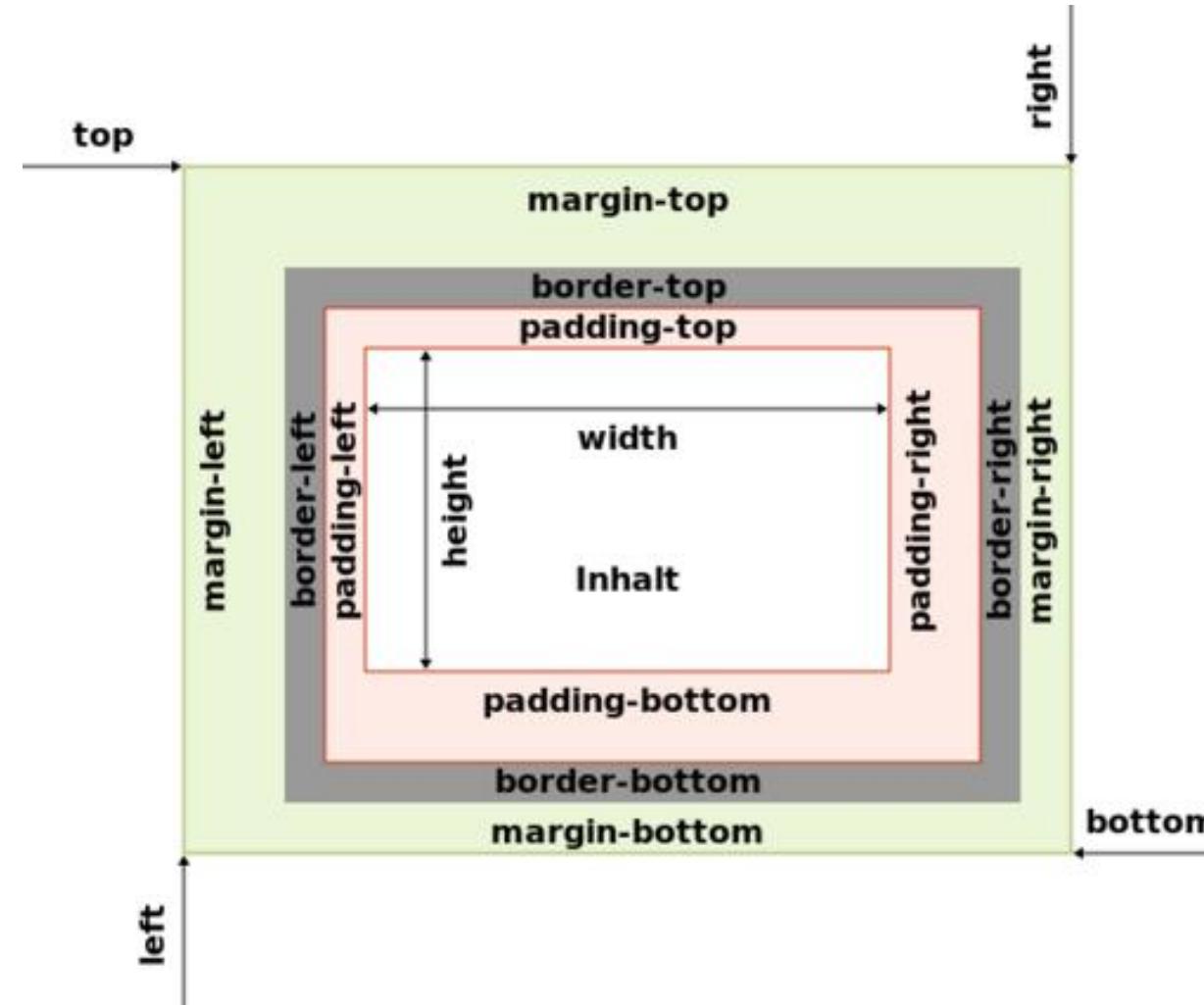
- Dynamische Pseudoklassen
 - Selektion und Formatierung von Elementen aufgrund von Benutzeraktionen
 - **:hover** Elemente, die mit dem Mauszeiger berührt werden
 - **:focus** fokussierte Elemente (z.B. durch die Tabulatortaste)
 - **:active** angeklickte Elemente

```
a { padding: .2em; }
a:focus { background-color: red; color: black; }
a:hover { background-color: blue; color: white; }
a:active{ background-color: yellow; color: black; }

...
<ul>
  <li><a href="https://hallo-welt.org">In die Welt</a></li>
  <li><a href="https://forum.hallo-welt.org">Zum Forum</a></li>
  <li><a href="https://wiki.hallo-welt.org">Zum Wiki</a></li>
</ul>
```

- Potenzielles Problem 1: Reihenfolge der definierten dynamischen Pseudoklassen muss beachtet werden
- Potenzielles Problem 2: Mobilgeräte, die i.d.R. keine Maus haben (→ Touch-Interaktion)

- Aufbau des klassischen Box-Modells



Quelle: Selfhtml.org

CSS – Box-Modell

- Jedes Block-Element bildet eine rechteckige Box, die frei formatiert werden kann
 - Textabsätze **p**
 - (Text-) Abschnitte **div, article, main, aside, ...**
 - Überschriften **h1, h2, ...**

```
h1{  
    color: midnightblue;  
    background-color: yellow;  
    border: 1px solid;  
    margin: 0;  
    padding: 0;  
}  
  
p{  
    width: 200px;  
    color: green;  
    background-color: lightyellow;  
    border: 2px solid red;  
    border-radius: 0 1em 1em 1em;  
    margin: 20px auto;  
    padding: 2em;  
}
```

Überschrift in einer rechteckigen Box

Sie können die Überschrift und auch den Textabsatz mit CSS beliebig gestalten.

Quelle: Selfhtml.org

CSS – Box-Modell

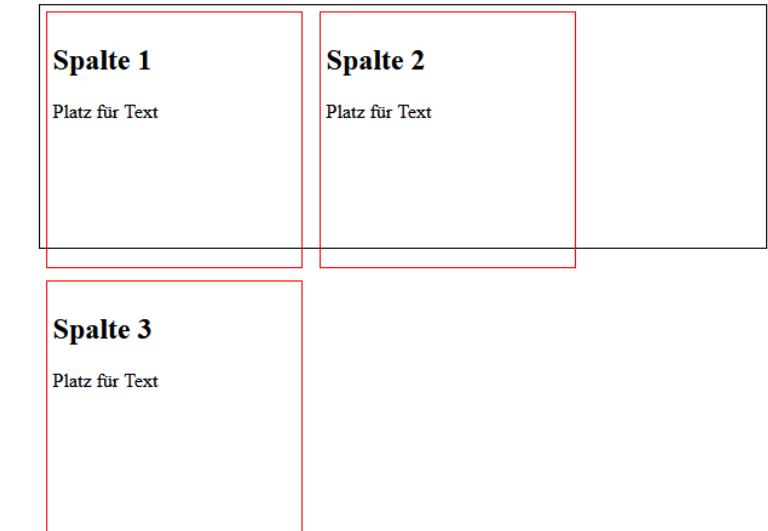
- Elemente in einem Spalten-Layout anordnen (mit fester Pixel-Breite)

```
.container{  
    width: 600px;  
    height: 400px;  
    border: 1px solid;  
    padding: 0;  
}
```

```
.spalte {  
    width: 200px;  
    height: 200px;  
    border: 1px solid red;  
    margin: 5px;  
    padding: 5px;  
    display: inline-block;  
}
```

```
<h1>pixelgenaues Mehrspalten-Layout</h1>  
<main>  
    <div class="container">  
        <section class="spalte">  
            <h2>Spalte 1</h2>  
            <p>Platz für Text</p>  
        </section>  
        <section class="spalte">  
            <h2>Spalte 2</h2>  
            <p>Platz für Text</p>  
        </section>  
        <section class="spalte">  
            <h2>Spalte 3</h2>  
            <p>Platz für Text</p>  
        </section>  
    </div>  
</main>
```

pixelgenaues Mehrspalten-Layout



Quelle: Selfhtml.org

- Elemente in einem Spalten-Layout anordnen (mit fester Pixel-Breite)



The screenshot shows the Chrome DevTools inspecting an element with the class "spalte". The "Layout" tab is selected. The "Gitter" section displays the message: "Es wird kein CSS-Gitter auf dieser Seite verwendet." (No CSS grid is used on this page). The "Box-Modell" section provides a detailed breakdown of the box dimensions:

Außenabstand	5					
Rand	1					
Innenabstand	5					
5	1	5	200×200	5	1	5
5	1	5	5	1	5	

Quelle: Selfhtml.org

CSS – Box-Modell

- Layout-Probleme mit festen Pixel-Angaben lassen sich durch andere Box-Modells evtl. umgehen
- Mathematisch durch Nutzung der calc()-Funktion

```
section, aside{  
    width: calc(33.3% - (2px + 10px + 10px));  
    height: calc(200px - (2px + 10px ));  
    ...  
}
```

- Alternatives Box-Modell durch box-sizing

```
*{ box-sizing: border-box; }
```

Innenabstand und Randlinie werden vom Browser bei der Breitenberechnung berücksichtigt

CSS – Box-Modell

- Layout-Probleme mit festen Pixel-Angaben lassen sich durch andere Box-Modells evtl. umgehen

- Flexibles Layout durch Flexbox

```
article{  
  ...  
  display: flex;  
}  
  
section, aside{  
  ...  
  flex: 1;  
}
```

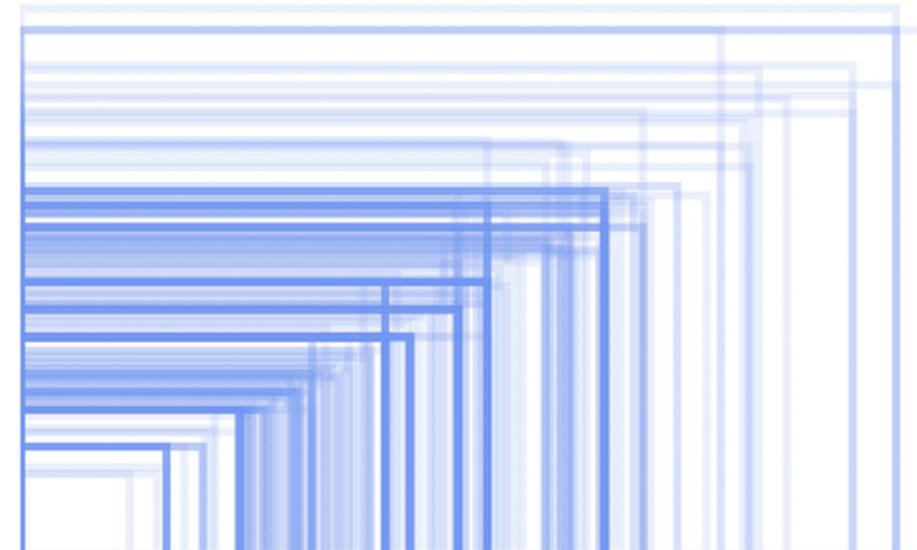
- Flexibles Layout durch Grid Layout

```
article {  
  border: 1px solid;  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr;  
  grid-gap: 1em;  
}
```

Responsives Webdesign Als responsives Webdesign versteht man das an das Ausgabemedium, vor allem eben auf die Bildschirmgröße, angepasste Verhalten der Webseite.	Grid Layout Mit dem Grid Layout Modul ist es möglich, responsive zweidimensionale Layouts zu erstellen. Dabei wird ein Raster angelegt, indem sich die Kindelemente ohne feste Größenangaben und weitere CSS-Einstellungen wie position, float oder clear bequem und flexibel positionieren (lassen).	Links • Grid Layout • Einstieg in Grid Layout
---	--	--

Quelle: Selfhtml.org

- Eine der Anforderungen u.a. aus dem Responsive Web-Design:
die Anpassung der Darstellung von Dokumenten an das Ausgabemedium
- Gleiche Inhalte können speziell auf die unterschiedlichen Viewports formatiert werden
 - Ausgabe im Druck
 - Ausgabe auf dem Bildschirm der Workstation
 - Ausgabe auf dem Display des Smartphones
 - usw.



Viewportunterschiede, Quelle: Selfhtml.org / opensignal.com

- Verschiedene Ansätze: von Desktop First über Mobile First bis zu Flexible Layout

```
nav li{  
    display: block;  
}  
  
@media( min-width: 35em ){  
    /* Festlegungen */  
    nav li{  
        display: block;  
        width: 50%;  
    }  
  
    @media( min-width: 50em ){  
        nav li{  
            display: inline;  
            width: 50%;  
        }  
    }  
}
```

CSS – Media Queries

- Media Queries berücksichtigen nicht nur die Größe der Ausgabemedien, sondern auch deren Auflösung (Pixeldichte)
→ besonders für Anwendungen im (professionellen) Grafikbereich relevant

```
@media  
( -webkit-min-device-pixel-ratio: 2 ),  
( min-resolution: 192dpi ){  
    /* Retina-spezifische Angaben */  
}
```

- Media Queries können auch Einfluss auf die Inhalte nehmen

```
...  
if( window.matchMedia( "( min-width: 35em )" ).matches ){  
    // Der Viewport ist mindestens 35em breit  
    // also wäre ein klein(er)es Bild empfehlenswert)  
}  
else{  
    // Das Medienmerkmal trifft nicht zu, alle anderen Viewports  
    // hier kann auch ein grosses Bild eingefügt werden  
}
```

1. Wieviele und welche Fehler beinhaltet folgender Code? (Quelle: selfhtml.org)

```
.medaille {  
    color: # 000;  
    background-color: rgb (255, 204, 0);  
    font-size: 50 px;  
    margin: 1em auto;  
    padding: 1em;  
    width: 1em;  
    height: 1em;  
    text-align: center;  
    border: 1px solid;  
    border-radius: 50%;  
}
```

- Grafiken
 - **img**
 - verweisensitive Grafiken mit (responsiven) Imagemaps
 - **picture**
- Audio und Video
 - **audio**
 - **video**
- „aktive“ Inhalte
 - **iframe**
 - **embed**
 - **object**

HTML – Medieneinbindung : Grafik

- Grafiken mit dem Image-Tag einbinden
 - ``
 - **src** Quell-/ Pfadangabe (Pflichtangabe)
 - **srcset** Referenzierung von Bildern in verschiedenen Größen/ Auflösungen
`<img src = "hallowelt-320.jpg"
srcset = "hallowelt-640.jpg 2x, hallowelt-960.jpg 3x, hallowelt-1280.jpg 4x" >`
 - **alt** alternativer Text, falls die Bilddatei nicht gefunden wird
→ Barrierefreiheit → Screenreader
 - **width**
height Größenangaben in Pixeln
→ **Achtung: mögliche Verzerrung, wenn beide angewandt werden!**
→ in HTML5 Angabe in Pixeln ohne Einheit
→ in HTML4 Angabe in Pixeln (**pix**) oder Prozent (%) mit der entsprechenden Einheit
 - **title** Tooltip
 - Für die Freunde der Sicherheit: Attribut **crossorigin** für CORS-Anfragen

HTML – Medieneinbindung : Grafik

- Verweisensitive Grafiken
 - Imagemaps
→ siehe Abschnitt 4
incl. der Probleme durch die pixelbasierten Angaben
in den HTML-Imagemaps

```

<map name="Voegel">
  <area shape="circle" coords="375,65,25" href="www.zoo.de" >
</map>
```



Hoffmann, Beyond Hypertext (2020)

- Verweisensitive Grafiken

- Responsive Imagemaps mit SVG

```
<figure id="imagemap">
  <svg viewBox="0 0 1536 1024" >
    <defs>
      <style>
        rect:hover{
          fill: white;
          opacity:0.5; }
      </style>
    </defs>

    <image width="1536" height="1024" xlink:href="Voegel.jpg">
      <title>Basstölpel im Zoo Bremerhaven</title>
    </image>
    <a xlink:href="https://de.wikipedia.org/wiki/Basstölpel">
      <rect x="300" y="125" opacity="0" width="250" height="300" />
    </a>
  </svg>
</figure>
```

HTML – Medieneinbindung : Audio

- Audio-Dateien können mit dem Audio-Tag ohne Plugin oder Add-On im Browser eingebunden werden

```
<audio src="hallo-welt-rap.mp3" controls />
```

oder:

```
<audio id="embedded_audio_with_controls" controls>
  <source src="hallo-welt-rap.mp3" type="audio/mp3" />
  <source src="hallo-welt-rap.ogg" type="audio/ogg" />
  Sie würden hier den Hallo-Welt-Rap hören.<br>
  Ihr Browser kann diese Datei aber nicht abspielen.<br>
  Schade.<br>
  Aber Sie können ihn unter <a href="#">hallo-welt-rap.mp3</a> runterladen.
</audio>
```

HTML – Medieneinbindung : Audio

```
<audio src="hallo-welt-rap.mp3" controls>
```

...

- Attribute zur gezielten Einbindung

- **autoplay** Clip startet sofort
- **buffered** liest aus, wieviel schon zwischengespeichert worden ist (TimeRanges-Object)
- **controls** Werkzeugleiste zur Steuerung
- **loop** startet nach Ende automatisch von vorne
- **muted** Audio ist zu Beginn stumm geschaltet
- **played** zeigt an, wie viel schon abgespielt wurde (TimeRanges-Object)
- **preload** bestimmt, wie die Datei beim Laden der Seite in den Browser geladen werden soll, mögliche Werte:
 - **auto** gesamte Datei wird geladen
 - **none** Datei wird nicht vorgeladen
 - **metadata** nur die Metadaten werden geladen
- **src** URL der Quelle

HTML – Medieneinbindung : Video

- Videos können mit dem Video-Tag ohne Plugin oder Add-On im Browser eingebunden werden

```
<video src="hallo-welt.mp4" controls />
```

oder:

```
<audio id=„embedded_audio_with_controls“ controls>
  <source src="hallo-welt-rap.mp3" type="audio/mp3" />
  <source src="hallo-welt-rap.ogg" type="audio/ogg" />
  Sie würden hier den Hallo-Welt-Clip sehen.<br>
  Ihr Browser kann diese Datei aber nicht zeigen.<br>
  Schade.<br>
  Aber Sie können ihn unter <a href="hallo-welt.mp4">hallo-welt.mp4</a> runterladen.
</audio>
```

HTML – Medieneinbindung : Video

```
<video src="hallo-welt.mp4" controls />
```

...

- Attribute zur gezielten Einbindung

- **autoplay** Clip startet sofort
- **buffered** liest aus, wieviel schon zwischengespeichert worden ist (TimeRanges-Object)
- **controls** Werkzeugleiste zur Steuerung
- **height** Höhe des Players (ohne dieses Attribut wird diese automatisch erkannt)
- **width** Breite des Players (ohne dieses Attribut wird diese automatisch erkannt)
- **loop** startet nach Ende automatisch von vorne
- **muted** Audio ist zu Beginn stumm geschaltet
- **played** zeigt an, wie viel schon abgespielt wurde (TimeRanges-Object)
- **preload** bestimmt, wie die Datei beim Laden der Seite in den Browser geladen werden soll, mögliche Werte:
 - **auto** Verhalten entsprechend der Browsetreinstellungen
 - **none** Datei wird nicht vorgeladen
 - **metadata** nur die Metadaten werden geladen
- **src** URL der Quelle
- **poster** Vorschaubild

```
<video src="hallo-welt.mp4" controls />
```

...

- Unterstützte Formate

- MP4       mit H264 Video Codec und AAC Audio Codec
- WebM     mit VP8/ VP9 Video Codec und Vorbis Audio Codec
- Ogg     mit Theora Video Codec und Vorbis Audio Codec

```
<video poster="hallo-welt.png" controls>  
  <source src="hallo-welt.webm" type="video/webm" />  
  <source src="hallo-welt.ogg" type="video/ogg" />  
</video>
```

HTML – Medieneinbindung : Video

- Einbinden von Textdateien in Videos mit **track**-Element
 - als Untertitel oder als Inhaltsbeschreibung (→ Barrierefreiheit)

```
<video width="320" height="240" controls>
  <source src="hallo-welt.webm" type="video/webm" />
  <source src="hallo-welt.ogg" type="video/ogg" />
  <track src="subtitles_de.vtt" kind="subtitles" srclang="de" label="Deutsch" default>
  <track src="subtitles_en.vtt" kind="subtitles" srclang="en" label="English">
</video>
```

- Attribute für das **track**-Element
 - **default** gibt den Standardwert an
 - **kind** die Art der Texteinblendung
 - **captions** Bildunterschriften
 - **chapters** Kapitel
 - **descriptions** Beschreibungen
 - **subtitles** Untertitel
 - **label** Beschreibung des Text-Tracks
 - **src** URL der Textquelle (im .vtt-Format)
 - **srclang** spezifiziert Sprache

HTML – Medieneinbindung : „aktive“ Inhalte

- Einbinden von Dokumenten und anderen Web-Seiten mit dem **iframe**-Element
 - Achtung: Sicherheit → Inhalte aus fremden Quellen!

```
<p>Lorem ipsum dolor usw.</p>
<iframe src="www.hallo.welt" width="75%" height="512" name="Hallo Welt">
  <p>Ihr Browser kann leider keine eingebetteten Frames anzeigen</p>
</iframe>
```

- Attribute für das **iframe**-Element
 - **default** gibt den Standardwert an
 - **height** Höhe
 - **width** Breite
 - **name** Name des **iframe**
 - **src** URL der Quelle
 - **srcdoc** ermöglicht die Angabe von HTML, das dann als Inhalt des **iframe** dargestellt wird
 - **seamless** lässt das **iframe** als Teil der Seite erscheinen (randlos)

HTML – Medieneinbindung : „aktive“ Inhalte

- Einbinden von Dokumenten und anderen Web-Seiten mit dem **iframe**-Element
 - Achtung: Sicherheit → Inhalte aus fremden Quellen!

```
<p>Lorem ipsum dolor usw.</p>
<iframe src="www.hallo.welt" width="75%" height="512" name="Hallo Welt">
  <p>Ihr Browser kann leider keine eingebetteten Frames anzeigen</p>
</iframe>
```

- Besonderes Attribut für das **iframe**-Element
 - **sandbox** lässt das **iframe** in einer sandbox mit erhöhten **Sicherheitsrestriktionen** laufen
bestimmte Aktivitäten der eingebundenen Webseite können unterdrückt werden
 - **kein Wert** alle Restriktionen gelten
 - **allow-forms** erlaubt das Absenden von Formularen
 - **allow-pointer-lock** ermöglicht API-Zugriffe
 - **allow-popups** erlaubt Popups
 - **allow-same-origin** erlaubt es, den Inhalt des **iframe** als **same-origin** zu behandeln
 - **allow-scripts** erlaubt Scripting
 - **allow-top-navigation** erlaubt es, dass der Inhalt des **iframe** andere Seiten aufrufen darf

HTML – Medieneinbindung : „aktive“ Inhalte

- Einbinden von „Nicht-HTML“-Dokumenten mit dem **embed**-Element

```
<embed width="600" height="400"  
       data="hallo-welt.swf"  
       type="application/x-shockwave-flash"  
       src="hallo-welt.swf">
```



- Attribute für das **embed**-Element
 - **height** Höhe
 - **width** Breite
 - **src** URL der Quelle
 - **type** Typ des zu ladenden Plugins

HTML – Medieneinbindung : „aktive“ Inhalte

- Einbinden von „Nicht-HTML“-Dokumenten mit dem **object**-Element

```
<object width="600" height="400"
        data="hallo-welt.svg"
        type="image/svg+xml" >
    <param name="src" value="/local/hallo-welt.svg" />
    <!-- Fallback -->
    
</object>
```

- Attribute für das **object**-Element
 - **height** Höhe
 - **width** Breite
 - **data** URL der Quelle
 - **name** Name des Objekts
 - **type** MIME-Type
 - **usemap** Name des map-Elementes, mit dem es in Verbindung stehen kann
(siehe verweisensitive Grafiken)
- **parameter**-Objekt zur Zuweisung/ Anwendung weiterer (medienbezogener) Parameter als Name-Wert-Paar

HTML – Medieneinbindung : „aktive“ Inhalte

- Einbinden von „Nicht-HTML“-Dokumenten mit dem **object**-Element

```
<object width="600" height="400"
        data="hallo-welt.svg"
        type="image/svg+xml" >
    <param name="src" value="/local/hallo-welt.svg">
    <!-- Fallback -->
    
</object>
```

- Das **parameter**-Objekt und seine Attribute
 - **width** Breite
 - **data** URL der Quelle
 - **name** Name des Objekts
 - **type** MIME-Type
 - **usemap** Name des map-Elementes, mit dem es in Verbindung stehen kann
(siehe verweisensitive Grafiken)

1. Übungsprojekte

- **Informations- und Kontaktseite einer Naturschutzorganisation**
→ Seiten-Layout mit externem CSS
- Konzipieren und entwickeln Sie eine
 - Landingpage für ein Start-Up
 - Informationsseite mit Möglichkeit zur Kontaktaufnahme

FOM Hochschule

Web Technologie

(6) Dynamische Webseiten

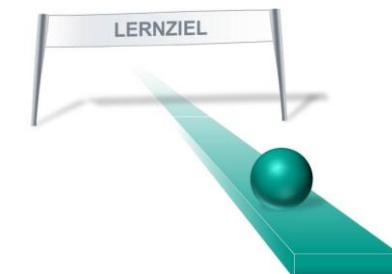
1	Formales, Vorstellung
2	Einführung
2.1	Allgemeine Einführung
2.2	Historie Internet, Web & Web-Technologien
3	(Web) Server – Client Kommunikation
3.1	Protokolle & Architekturen
3.2	Web- & Microservices
3.3	Authentifizierungs- und Authorisierungskonzepte
4	Grundlagen der Mediengestaltung
4.1	Media & Multimedia
4.2	Media & Modalitäten – Interaktion & Wahrnehmung
4.3	Hypermedia
5	Statische Webseiten
5.1	HTML
5.2	CSS
5.3	Medieneinbindung
6	Dynamische Webseiten
6.1	DOM, HTML & CSS
6.2	JavaScript & JSON
6.3	Weitere Konzepte & Sprachen der Webprogrammierung
7	Praxis
7.1	Vorstellung eines (ausgewählten) (JavaScript-) Frameworks
7.2	Vorstellung einer (ausgewählten) Thematik für ein eigenes Projekt
7.3	Betreute, eigenständige Entwicklung des (ausgewählten) Themas
8	Wiederholung (opt.)

Im Anschluss an diesen Themenblock sollen Sie

- ... definieren können, was eine dynamische Webseite ist und wie sich Dynamik im Web ausprägen kann.
- ... die Grundlagen der Programmierung im Web am Beispiel von JavaScript und TypeScript kennengelernt haben und ...
- ... diese grundlegend anwenden können.
- ... Entwurfsmuster wie Web-MVC erläutern und anwenden können.

Je nach noch zur Verfügung stehender Zeit und Vorwissen bei den Studenten könnte am Ende dieses Themenblocks eine weitere Vertiefung von Konzepten von Programmiersprachen stattfinden:

- Prototypen
- Callback-Funktionen
- usw.



Dynamische Webseiten?

Ein Definitionsversuch – und eine Diskussion!

Neben statischen Webseiten finden sich zahlreiche **dynamische Webseiten** im Internet.

Im Gegensatz zu statischen Webseiten bieten dynamische Webseiten ein hohes Maß an **Flexibilität**, da sie erst dann erzeugt werden, wenn sie aufgerufen werden.

Dabei werden die Inhalte [...] abhängig von der Benutzereingabe, dem Datum oder der Uhrzeit aus einer Datenbank oder Textdatei ausgelesen und [es wird] ein HTML-Dokument generiert. [...]

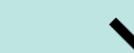
Damit ermöglichen dynamische Webseiten dem Betreiber, Inhalte von größtmöglicher Aktualität anzubieten.

Ein weiterer Vorteil von dynamischen Webseiten besteht darin, dass der **Inhalt vom Format getrennt** und mittels eines **Redaktionssystems** ohne HTML-Kenntnisse geändert bzw. aktualisiert werden kann.

Außerdem können sich wiederholende Inhalte ausgelagert werden, um sie bei Bedarf an anderer Stelle einzubinden.

(Quelle: <https://onlinemarketingfans.de>)

Check

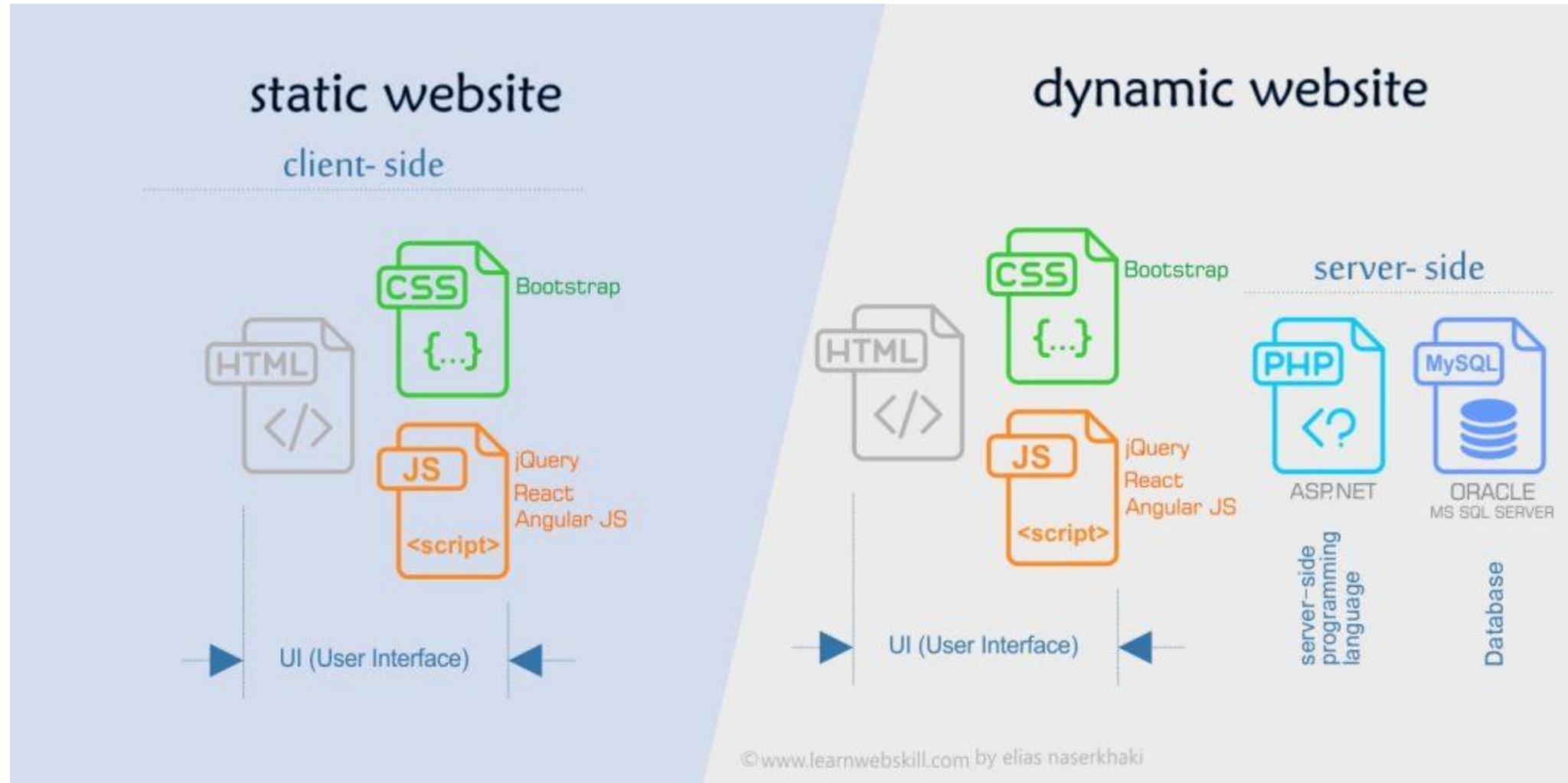


Diese Charakteristiken beschreiben allerdings nur eine externe und von Serverseite betriebene Dynamik!

Dynamik kann jedoch auch durch Interaktion auf der Seite selbst oder mit Elementen auf der Seite sowie durch Skripte auf der Seite erzeugt werden. Dies ist dann interne Dynamik und diese ist serverunabhängig!

Das geht auch bei statischen Webseiten

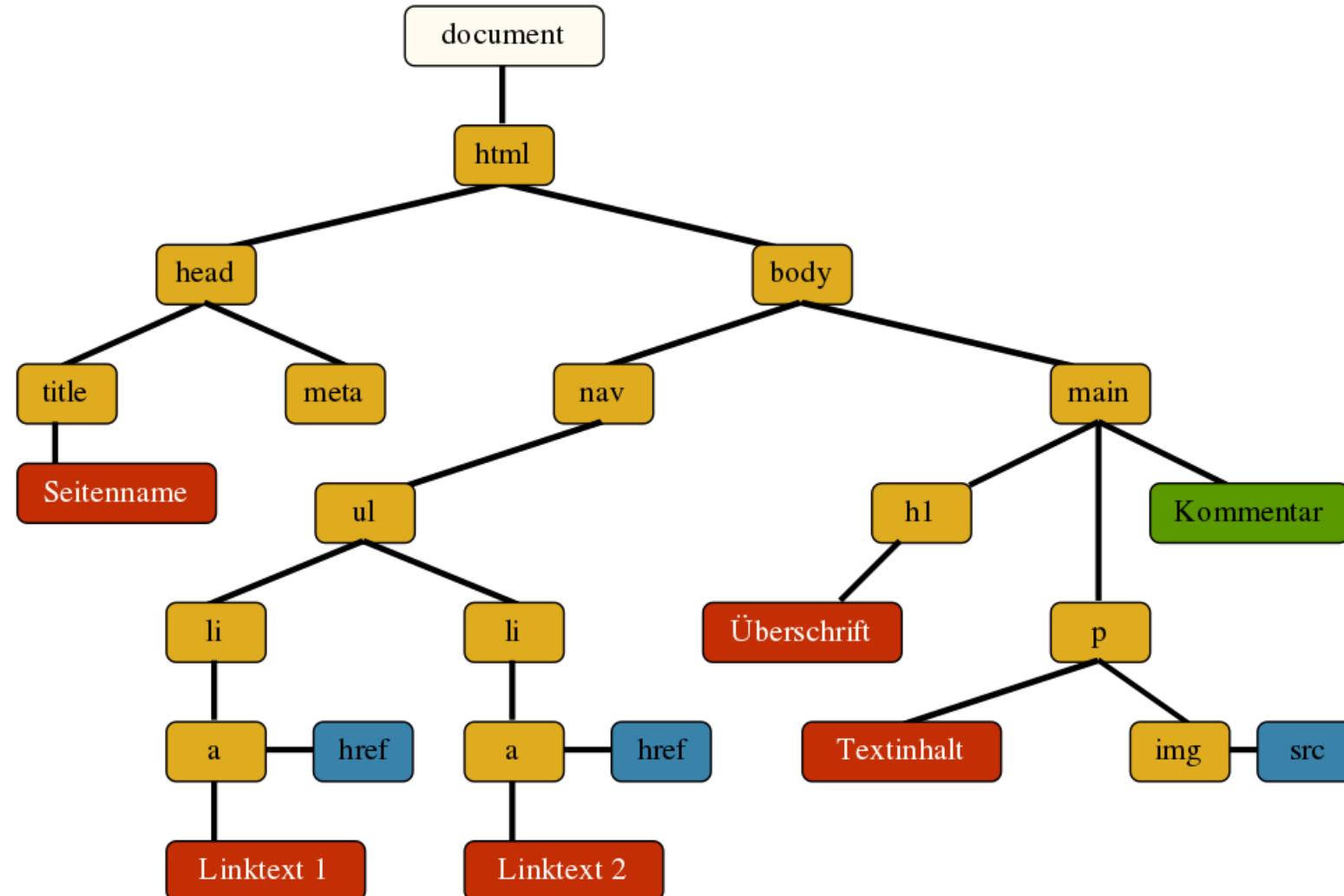
Dynamische Webseiten?



Quelle: learnwebskill.com

Document Object Model

DOM - Objekthierarchie



Quelle: Selfhtml.org

DOM - Objekthierarchie

window



document



node

```
alert( "foo");  
window.alert( "foo" );  
  
window.resizeTo( x, y )  
  
window.event.clientX  
window.event.clientY  
  
window.event.PageX  
window.event.PageY
```

```
document.write("<p>Hallo!<br>" );  
document.documentElement.clientHeight
```

A node is the generic name for any type of object in the DOM hierarchy.
An element is a specific type of node.
It can be directly specified in HTML with an HTML-tag and it can have properties like an id or a class.

(Quelle: stackoverflow.com)

Document Object Model

DOM - Objekthierarchie

- Jeder Node kann einzeln und gezielt angesprochen und die Werte seiner Eigenschaften verändert werden.
- In der Regel geschieht dies durch Aufrufe aus Skripten, die in die Seiten eingebunden sind.
- Auch aus Style Sheets können Elemente gezielt angesprochen werden.
- Nodes können (aus Skripten heraus) zur Laufzeit erzeugt und in das DOM eingefügt werden.
- Nodes können (aus Skripten heraus) zur Laufzeit aus dem DOM herausgenommen und auch gelöscht werden.

```
<html>
  ...
  <div id="foo"></div>
  ...
```

```
<script>
  document.getElementById('foo').innerText = "Hallo Welt";
</script>
```

Das DOM wird nach der Einführung in CSS und JavaScript noch einmal aufgegriffen.

HTML – Erweiterung für die dynamische Nutzung

Interaktion als Teil/ Grundlage der Dynamik - Formulare

<form>

Bitte geben Sie Ihren Namen ein:

<input type="text" value="?" size="15" name="user">

Bitte geben Sie Ihre Meinung hier ein:

<textarea cols="15" rows="4" name="adresse">

Ihre Nachricht?

</textarea>

<p>

Ich interessiere mich für:

<input type="checkbox" name="tv">TV

<input type="checkbox" name="internet" checked>internet

<input type="checkbox" name="radio">Radio

</p>

<p>

Ich nutze das Medium:

<input type="checkbox" name="nutzen" value="1" checked>beruflich

<input type="checkbox" name="nutzen" value="2">privat

...

Bitte geben Sie Ihren Namen ein:

?

Bitte geben Sie Ihren Namen ein:

Ihre Nachricht?

Ich interessiere mich für:

Internet



Ich nutze das Medium:

beruflich

privat

beides

gar nicht

HTML – Erweiterung für die dynamische Nutzung

Interaktion als Teil/ Grundlage der Dynamik – Formulare und Barrierefreiheit

```
form action="http://in.der.tiefe.des.raumes/cgi-bin/foo method=get">
<p>
  Ich interessiere mich für:<br>
  <option value="tv" tabindex="1">TV
  <option value="internet" tabindex="2" selected>internet
  <option value="radio" tabindex="3">Radio
</p>
<p>
  Ich nutze das Medium:<br>
  <option value="beruflich" tabindex="4" accesskey="G">beruflich
  <option value="privat" tabindex="5" accesskey="P">privat
  <option value="beides" tabindex="6" accesskey="B">beides
  <option value="garnicht" tabindex="7" accesskey="N">garnicht
  <option value="malsomalso" tabindex="8" accesskey="M">mal so, mal so
</p>
<p>
  <button name="fertig" value="1" tabindex="4">beruflich
...

```

HTML – Erweiterung für die dynamische Nutzung

Interaktion als Teil/ Grundlage der Dynamik – Formulare und Datenübermittlung

- CGI – Common Gateway Interface
 - Übertragung Anforderung Client → Server
 - Start des CGI-Programms
 - Ablauf des CGI-Programms
 - (Rück-) Übertragung Antwort Server → Client
- Methoden:
 - **get** Formulardaten werden in einem String gespeichert und dann verarbeitet **QUERY_STRING**
 - **Post** Verarbeitung der Formulardaten wie Benutzereingaben in Kommandozeilen
Allerdings: kein **EoF** → **CONTENT_LENGTH**

```
<form action="mailto:guenter@in.der.tiefe.des.raumes" method=post>
<form action="http://in.der.tiefe.des.raumes/cgi-bin/foo method=get>
```

CSS-Animation als Teil der Dynamik

- „Instinktive“ Reaktion des menschlichen Auges auf Bewegung
→ Animationen als Möglichkeit:
 - Interesse zu wecken
 - Aufmerksamkeit auf wichtige Bereiche zu lenken
- Animieren = lat. „zum Leben erwecken“
- CSS bietet (zunächst) drei Möglichkeiten der Animation
 - **transition** weiche Übergänge zwischen zwei Werten
 - **animation** komplexe Abfolgen und Bewegungsmuster
 - **offset** Animationen entlang von Pfaden
- CSS-Animations sind die Grundlage der Web Animations API (WAAPI)
→ animieren von CSS-Eigenschaften und deren Steuerung mit **JavaScript**

CSS-Animation als Teil der Dynamik - transition

- Transition = Übergang zwischen zwei Zuständen eines HTML-Elements
- **transition** ändert Werte von CSS-Eigenschaften allmählich oder schrittweise durch Interpolation der dazwischenliegenden Werte
- Transitionen werden (häufig) durch Benutzeraktionen angestoßen
- Beispiel für „einfliegende Bildunterschriften“ (Quelle: selfhtml.org):

```
figure h2{  
    transform: translateY(-200px);  
    transition: all 0.3s ease-in-out;  
}  
figure:hover h2 {  
    transform: translateY(0px);  
    transition-delay: 0.4s;  
}  
figure p {  
    transform: translateX(200px) rotate(90deg);  
    transition: all 0.4s ease-in-out;  
}  
...
```

CSS-Animation als Teil der Dynamik - animation

- Animation von HTML-Elementen durch die Zuweisung veränderter Werte zu den Element-Eigenschaften
- Unterschied zu **transition**: Möglichkeit zu mehrstufigen Animationsfolgen
→ **@keyframes**
- **@keyframes**-Regeln legen durch „Wegpunkte“ einzelne Schritte einer CSS-Animationssequenz fest
- **@keyframes**-Regeln bestehen aus
 - einem Namen zum Aufrufen der Animation
 - Wegpunkten
 - Start- und Endpunkt mit den Schlüsselwörtern **from** und **to**
 - mehrere Wegpunkte mit Prozentwerten
 - Wertangaben für die zu animierenden CSS-Eigenschaften

```
h2{ animation: textwelle 3s; }

@keyframes textwelle{
    0% { transform: scale(0.1); opacity: 0; }
    70% { transform: scale(1.25); opacity: 1; }
    100%{ transform: scale(1); opacity: 1; }
}
```

CSS-Animation als Teil der Dynamik - offset

- Mit offset sind auch nicht-lineare Animationen möglich, also z.B. Bewegungen entlang
 - freier Kurven
 - Schleifen
 - Ellipsen
 - usw.

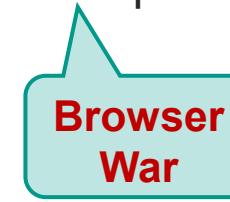
```
.path{  
  offset-path: path( 'M300,100h300v100h-300z' );  
  animation: run 6s linear infinite;  
}  
.reverse{  
  offset-path: path('M300,100h300v100h-300z');  
  animation: run 6s linear reverse infinite;  
}  
 @keyframes run{  
  0% { offset-distance: 0; }  
  to { offset-distance: 100%; }  
}
```

Historie

1996, März	JavaScript 1.	
1996, August	JavaScript 1.1.0	
1997	JavaScript 1.2.0	ECMAScript 1
1998	JavaScript 1.3.0	ECMAScript 2
1999	JavaScript 1.4.0 (nur für Netscape Server)	ECMAScript 3
2000	JavaScript 1.5.0	
2005	JavaScript 1.6.0	
2006	JavaScript 1.7.0	
2008	JavaScript 1.8.0	
2009		ECMAScript 5
2010	JavaScript 1.8.5 (ECMAScript 5 Compliance)	
2011		ECMAScript 5.1
2015		ECMAScript 2015
2016		ECMAScript 2016
2017		ECMAScript 2017
2018		ECMAScript 2018
2019		ECMAScript 2019
2020		ECMAScript 2020

Java, JavaScript, ECMAScript, JScript?

- **Java ≠ JavaScript !**
- „**JavaScript** bezeichnet genau genommen verschiedene Versionen der Sprache zugleich, da sich sowohl die Implementationen in Browsern und Frameworks wie Node.js.“
(Quelle: vollzeitblogger.de)
- „**ECMAScript** beschreibt die standardisierte Form der Sprache. Derzeit wird in aktuellen Browsern ECMAScript 5 unterstützt und in manchen Browsern sind schon experimentelle Implementationen von ECMAScript 6 Funktionen bereit gestellt.“
(Quelle: vollzeitblogger.de)
- „**JScript** war eine Microsoft-eigene, proprietäre Entwicklung einer Skriptsprache für Webbrowser und baut auf ECMAScript als zentralem Bestandteil auf. JScript erweitert JavaScript um zusätzliche, ursprünglich Microsoft-spezifische Features.
Als einziger Webbrowser unterstützt der Microsoft Internet Explorer JScript vollständig.“
(Quelle: wikipedia.de)



Browser
War

JavaScript - Einbinden

- Direktes Einbinden in HTML

```
<html>
  <head>
    <script>
      alert("Hallo Welt!");
    </script>
  ...

```

- Einbinden externer JavaScript-Dateien

```
<body>
  <input type="number" id="Eingabe" value="0">
  <button type="button" id="los">Quadrieren</button>
  <script src="quadrat.js"></script>
</body>
```

- Einbinden externer JavaScript-Dateien im Head

```
<head>
  <script type="text/javascript" language="JavaScript" src="foo.js"></script>
```

```
function Quadrat() {
  var Ein = document.getElementById('Eingabe');
  var Erg = Eingabe.value * Eingabe.value;
  alert("Quadrat von " + Ein.value + " = " + Erg);
  Eingabe.value = 0;
}

var los = document.getElementById('los');
los.addEventListener ('click', Quadrat, true);
```

Datei quadrat.js

Mächtig, mächtig!

TensorFlow.js

KI (von Google)

Schach

<https://ichi.pro/de/erstellen-sie-eine-einfache-schach-ki-in-javascript-2384596745009>
<https://zeyu2001.github.io/chess-ai/>

Bildbearbeitung

[https://ichi.pro/de/10-javascript-bildbearbeitungsbibliotheken-für-2020-26937466213807](https://ichi.pro/de/10-javascript-bildbearbeitungsbibliotheken-fur-2020-26937466213807)

Diagramme

Charts.js (<https://www.chartjs.org/>)

Unbegrenzte Welten

<https://kinsta.com/de/blog/javascript-bibliotheken/>

Sensordaten

mit Tricks

- Variablen

```
var counter = 0;
```

globale Variable

...

```
function init(){
```

```
    let myCounter = 0; lokale Variablen (seit ECMAScript 6)
```

...

- „strenger“ Modus mit **strict** (seit ECMAScript 5)

- aktiviert ein standardisiertes, um Fehlerquellen bereinigtes, Subset von JavaScript
- zwingt zu „sauberer“ Programmierung
→ fehlerhaft geschriebene Methoden oder Variablen
- Verbot von
 - undeclareden Variablen
 - Anweisungen ohne abschließendes Semikolon

```
use strict;
```

...

```
let counter = 0;
```

...

- Kontrollausgabe mit der Konsole

- Funktionstaste F12
- über die jeweiligen Menüs des Browsers
- Tastenkombinationen der Browser

The screenshot shows the Firebug interface in Firefox. The tabs at the top are Inspektor, Konsole (selected), Debugger, and Netzwerkanalyse. Below the tabs, the file 'JS-tut-04.html' is open. The code editor displays the following JavaScript:

```
<head>
<meta charset="utf-8">
<title>4. Eingabe von Namen und Alter</title>
</head>
<body>
<script>
  'use strict';
  var name = '';
  var alter = '';
  const erwachsen = 18;
  name = prompt('Bitte geben Sie Ihren Namen ein!', name);
  alter = prompt('Bitte geben Sie Ihr Alter ein!', alter);

  if (alter < erwachsen) {
    console.log('Du kommst hier net rein!');
  }

  else {
    var ausgabe = `Hallo ${name}!`;
    console.log(ausgabe);
  }
</script>
```

Blue arrows on the left margin indicate the current execution context for each line of code.

Firebug-Konsole im Firefox Version 40

Quelle: Selfhtml.org

JavaScript - Kontrollstrukturen

- Verzweigung mit **if ... else** und mit **switch**

...

```
alter = prompt( 'Bitte geben Sie Ihr Alter ein!', alter );
if( isNaN( alter ) )
    eingabe = 'dumm';
else if( alter < erwachsen )
    eingabe = 'jung';
else
    eingabe = 'ok';

switch( eingabe ){
    case 'jung':
        text ='Du bist leider zu jung!';
        break;
    case 'dumm':
        text ='Zahlen einzugeben ist schon schwierig!';
        break;
    default:
        ...
}
```

JavaScript – Strukturierung des Programmcodes

- Programmstruktur durch Funktionen

```
'use strict';
let eingabe,
    ergebnis,
    text;

// Hilfsfunktion, die Wurzel aus 'zahl' zieht und als 'ergebnis' zurueckgibt
function wurzelAus( zahl ){
    zahl      = parseFloat( zahl );
    ergebnis = Math.sqrt( zahl );
    return ergebnis;
}

eingabe = prompt( 'Bitte geben Sie eine Zahl ein!', eingabe );

wurzelAus( eingabe );
text = 'Die Wurzel von ${ eingabe } ist ${ ergebnis }.';
console.log( text );
...
```

JavaScript – Strukturierung des Programmcodes

- Programm-/ Datenstruktur durch Objekte
- Statisch erstellt Objekte

```
let order = {  
    orderNo: 2346598,  
    name: 'Hammelmeir, Jupp',  
    adress: 'Gehweg 23, 42555 Astadt',  
    items: [ 23578, 65879, 69873, 69877 ]  
};
```

- Dynamisch erstellte Objekte

```
let nextOrder = new Object();  
  
nextOrder.id      = orderCount;  
nextOrder.name    = 'Mustermann, Michael';  
nextOrder.adress  = 'Postweg 307, 42655 Solingen';  
nextOrder.items   = [ 64586, 36887 ];
```

- Einführung in OOP mit JavaScript:

https://developer.mozilla.org/de/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript

- Datenformat, um Informationen wie Objekte, Arrays und sonstige Variablen in lesbarer Form zu speichern
- An JavaScript angelehnte, leichtgewichtige Alternative zu XML

```
{  
  "name": "Georg",  
  "alter": 47,  
  "verheiratet": false,  
  "beruf": null,  
  "kinder": [  
    {  
      "name": "Lukas",  
      "alter": 19,  
      "schulabschluss": "Abitur"  
    },  
    {  
      "name": "Lisa",  
      "alter": 14,  
      "schulabschluss": null  
    }  

```

- Schlüssel-Werte-Paare
 - mehrere Paare werden durch Kommata getrennt
 - Schlüssel und Werte werden jeweils durch einen Doppelpunkt getrennt
- Eigenschaftnamen (Schlüssel)
 - werden in doppelten Anführungszeichen notiert
- Werte
 - Führende Kommata in Objekten und Arrays sind verboten.
 - Strings müssen durch doppelte Anführungszeichen begrenzt sein.
 - Führende Nullen bei Zahlen sind verboten.
 - Einem Dezimalpunkt muss mindestens eine Ziffer folgen.

- Datenformat, um Informationen wie Objekte, Arrays und sonstige Variablen in lesbarer Form zu speichern
An JavaScript angelehnte, leichtgewichtige Alternative zu XML

```
{  
  "name": "Georg",  
  "alter": 47,  
  "verheiratet": false,  
  "beruf": null,  
  "kinder": [  
    {  
      "name": "Lukas",  
      "alter": 19,  
      "schulabschluss": "Abitur"  
    },  
    {  
      "name": "Lisa",  
      "alter": 14,  
      "schulabschluss": null  
    }  

```

- Variablentypen

- **Null** Bezeichnung, dass sich kein Wert in der Variable befindet
- **Boolean** Zuweisung von booleschen Werten
- **Zahl** Zuweisung von Zahlenwerten werden
Die Zahl darf nur aus den Ziffern 0 - 9, einem (optionalen) Vorzeichen und einem Dezimalpunkt bestehen
Optional kann die Zahl um einen Exponenten erweitert werden (**1.3e+4**)
- **String** Zuweisung von Zeichenketten
- **Array** Deklaration/ Zuweisung von Wertefeldern
Das Array wird in eckigen Klammern notiert
Jeden Eintrag im Array kann man einen beliebigen zugelassenen Typ besitzen
- **Objekt** Objekte werden in geschweiften Klammern notiert

- Direktes Einbinden von JSON in den Programmcode

```
var name = {  
    "name": „Schorsch“,  
    ...  
}  
...  
...
```

- Einbinden externer JSON-Dateien

```
<html>  
...  
<script src="data.js"></script>  
<script src="javascript.js"></script>  
...
```

JavaScript Object Notation - JSON

- Laden externer JSON-Dateien mit Callback-Funktion (unter Nutzung von **XMLHttpRequest**)

```
function loadJSON( file, callback ){
    var xobj = new XMLHttpRequest();
    xobj.overrideMimeType( 'application/json' );
    xobj.open( 'GET', file, true );
    xobj.onreadystatechange = function(){
        if( xobj.readyState == 4 && xobj.status == '200' ){
            // Required use of an anonymous callback as .open will NOT return a value
            // but simply returns undefined in asynchronous mode
            callback( xobj.responseText );
        }
    };
    xobj.send( null );
}
```

- Anwendung

```
loadJSON('/Users/Documents/workspace/data.json', function( text ){
    var data = JSON.parse(text);
    console.log( data );
});
```

- bisher: statische Betrachtung
→ unmittelbares Ausführen der (Beispiel-) Skripte nach dem Laden der Seite
- Einführung von Dynamik durch „Event-Handling“
 - Dokument laden
 - Anhängen von Event-Handlen an (DOM-) Elementknoten mit JavaScript
 - Reaktion des Dokumentes/ der Skripte bei/ durch Bedienung durch den Anwender
- veraltet (deprecated): HTML-Eventhandling
 - <html>
 - ...
 - <button onclick="alert('Hallo Welt!');">Drück mich!</button>
 - ...

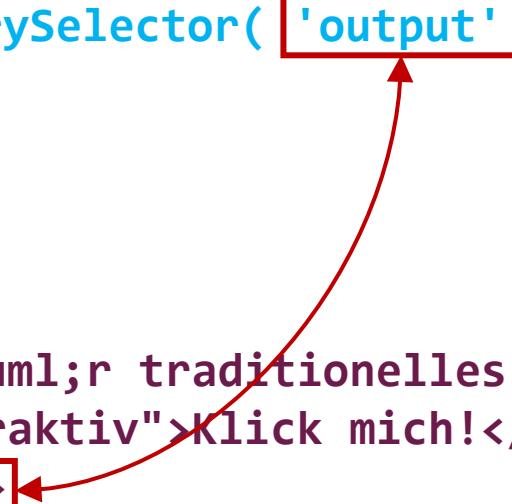
Back to the DOM: Eventhandling

- Sarten von Skripten erst nach dem Laden des Dokuments (traditionelles Event-Handling)

```
<html>
<head>
  <script>
    window.onload = start;

    function start(){ document.querySelector( '#interaktiv' ).onclick = klickFunktion; }
    function klickFunktion(){
      document.querySelector( 'output' ).innerText += 'Neu eingefügter Text.';
    }
  </script>
</head>

<body>
  <h1>Beispiel für traditionelles Event-Handling</h1>
  <button id="interaktiv">Klick mich!</button>
  <output></output>
</body>
</html>
```



Back to the DOM: Eventhandling

- Aktuelles „Event-Handling“:
 - HTML-Dokument durch Browser laden, parsen, rendern
 - Bei Einbindung von Skripten (JavaScript):
 - Phase 1 Dokument empfangen und parsen
→ JavaScript wird erstmals ausgeführt
 - Definition von Objekten und Funktionen für die spätere Nutzung
 - Es werden nicht alle notierten Funktionen unmittelbar aufgerufen
 - Skript hat noch keinen vollständigen Zugriff auf das Dokument
→ Das DOM ist noch nicht vollständig
 - Phase 2 Dokument fertig geladen
→ vollständiger Zugriff auf das Dokument über das DOM ist erst jetzt möglich (!)
 - Registrieren von Event-Handlern
→ an die jetzt vorhandenen Elementknoten werden Event-Handler mit Handler-Funktionen angehängt
 - Inhalt oder Darstellung von Elementen kann verändert werden
 - Phase 3 Bedienung des Dokuments durch den Anwender
→ Reaktion der Skripte auch Aktivitäten des Benutzers:
aktiviert der Benutzer (z.B. durch einen Mausklick auf ein Element) die überwachten Ereignisse, werden die Handler-Funktionen ausgeführt.

Back to the DOM: Eventhandling

- Aktuelles „Event-Handling“:
 - Registrieren von Event-Handlern mit `addEventListener`
 - Entfernen von Event-Handlern mit `removeEventListener`

```
document.addEventListener( 'DOMContentLoaded', function(){

    document.querySelector( '#interaktiv' ).addEventListener( 'click', klickFunktion );
    document.querySelector( '#entferne' ).addEventListener( 'click', entferneklickFunktion );

    function klickverarbeitung(){
        document.querySelector( 'output' ).innerText += 'Neu eingefügter Text.\n';
        document.querySelector( '#entferne' ).setAttribute( 'aria-hidden', false );
    }

    function entferneKlickverarbeitung(){
        document.querySelector( '#interaktiv' ).removeEventListener( 'click', klickFunktion );
        document.querySelector( 'output' ).innerText += ' Event-Handler entfernt. \n';
    }
});
```

- Aktuelles „Event-Handling“:

- Reaktion auf Benutzerinteraktion

```
<html>
...
<button id="button">Drück mich!</button>
<output></output>
```

...

```
...
...
document.addEventListener( 'DOMContentLoaded', function(){
    let anzahl = 0, text = '';
    document.getElementById( 'button' ).addEventListener( 'click', gedrueckt );

    function gedrueckt(){
        anzahl = anzahl + 1;
        if( anzahl < 10 )
            text = 'Der Button wurde ' + anzahl + ',-mal gedrückt!';
        else if( anzahl == 15 )
            text = 'Die Maximalzahl der Klicks für diesen Button wurde erreicht';
        document.querySelector( 'output' ).innerText = text;
    }
});
```

JavaScript-Event-Handler:

<https://wiki.selfhtml.org/wiki/JavaScript/DOM/Event/%C3%9Cbersicht>

HTML-Event-Handler:

<https://html.spec.whatwg.org/multipage/webappapis.html#event-handler-attributes>

Back to the DOM: DOM-Manipulation

- Hinzufügen von Elementen zum DOM

```
<html>
```

```
...
```

```
<button id="button">Drück mich!</button>
```

```
<div id="textblock">
```

```
  <p>Hier können dynamisch weitere Absätze angehängt werden.</p>
```

```
</div>
```

```
...
```

```
...
```

```
document.addEventListener( 'DOMContentLoaded', function(){
```

```
  var text = 'Dies ist ein neuer Absatz';
```

```
  document.getElementById( 'button' ).addEventListener( 'click', createElement );
```

```
function createElement(){
```

```
  var container = document.getElementById( 'textblock' );
```

```
  var newElm = document.createElement( 'p' );
```

```
  newElm.innerText = text;
```

```
  container.appendChild( newElm );
```

```
}
```

```
} );
```

Back to the DOM: DOM-Manipulation

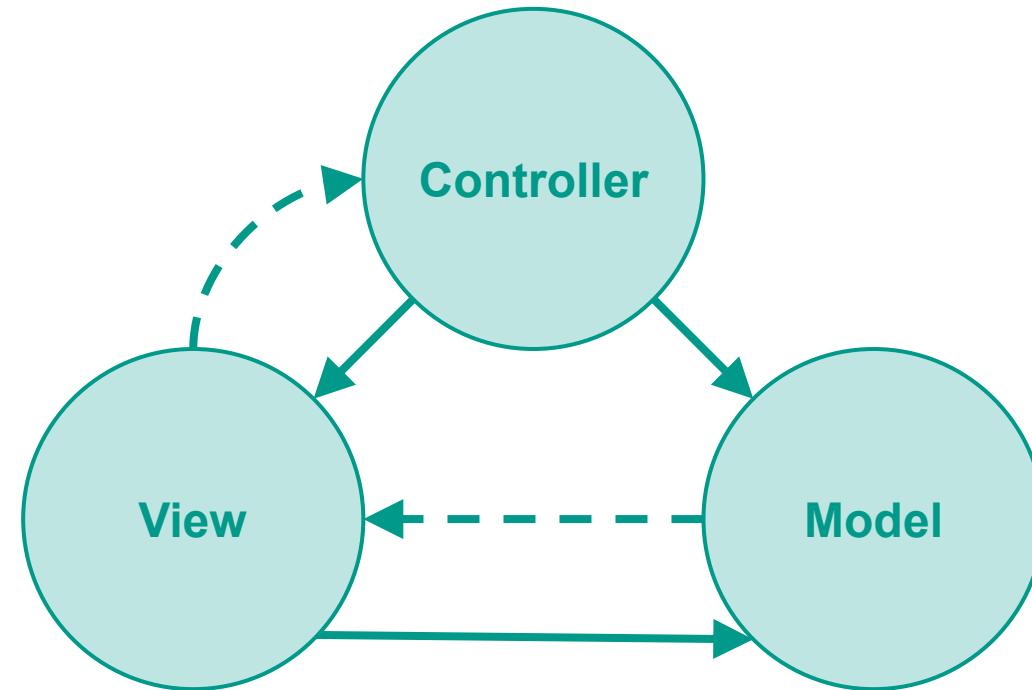
- Entfernen von Elementen aus dem DOM
→ zunächst muss der Elternknoten gefunden werden, um dann dessen Kind(er) zu entfernen

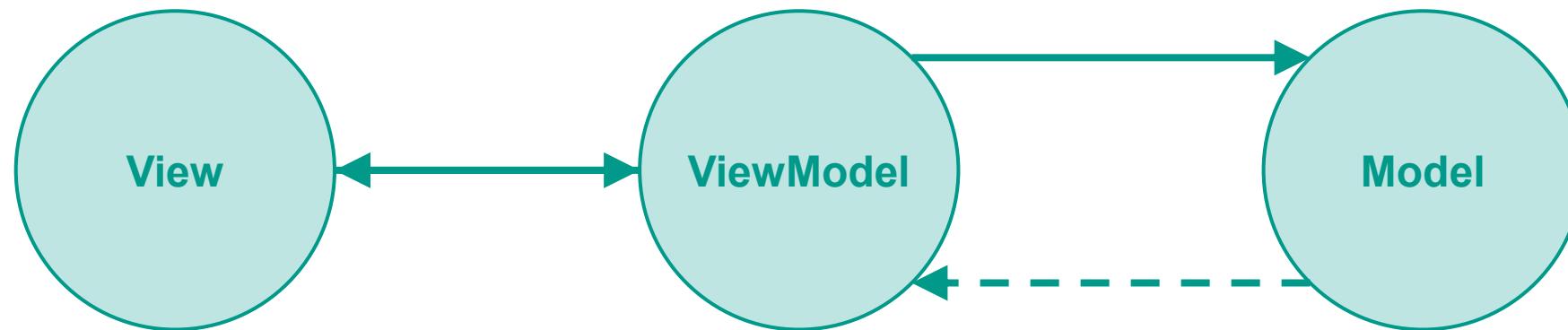
...

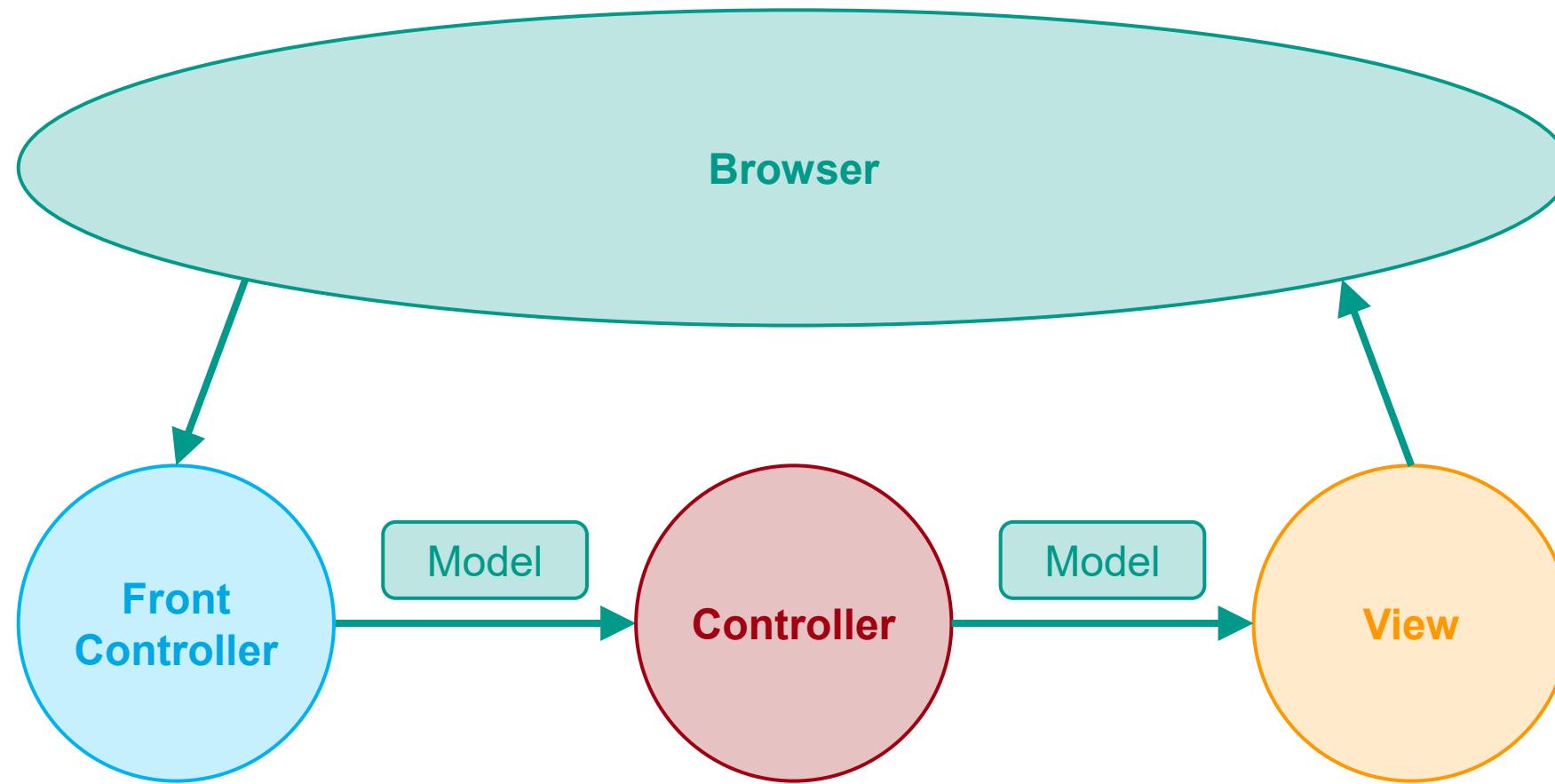
```
document.addEventListener( 'DOMContentLoaded', function(){
    document.addEventListener( 'click', removeElement );

    function removeElement( e ){
        var elem = e.target;
        var main = document.querySelector( 'main' );
        if( main != elem ){
            var parent = elem.parentNode;
            parent.removeChild( elem );
            return false;
        }
    }
});
```

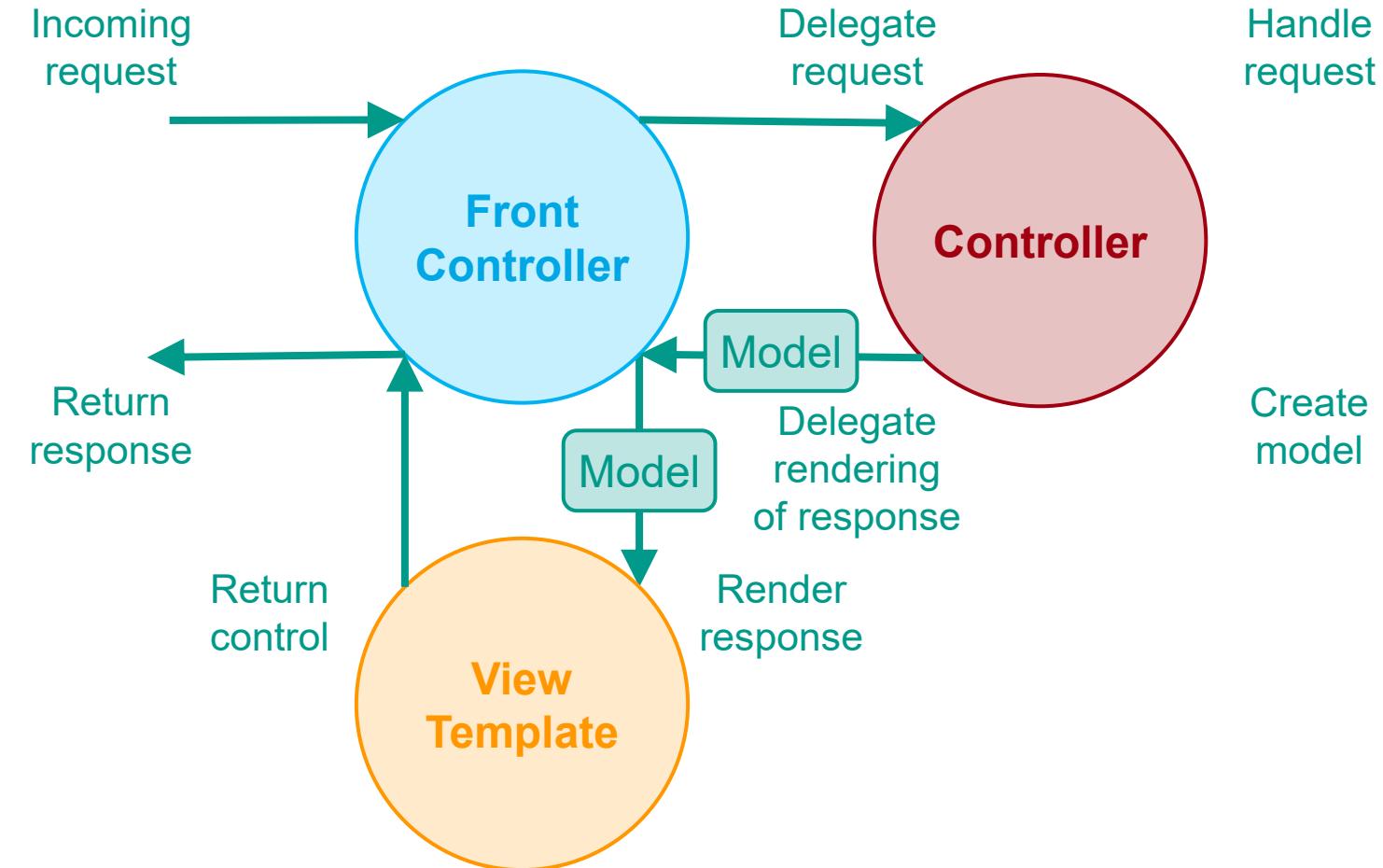
1. „Der beliebte Taschenrechner“ ;-)
Bauen Sie im Zusammenspiel mit HTML, CSS und JavaScript einen (simplen) Taschenrechner auf einer Webseite.
2. **Entwickeln Sie eine Webseite in einem mehrspaltigen Design (>= 3 Spalten), in dem zunächst alle Spalten gleiche Breite haben sollen.**
Wird der Mauscursor über eine der Spalten geführt, so soll die Breite dieser Spalte auf die Hälfte der Fensterbreite vergrößert und die anderen Spalten entsprechend verkleinert werden.
Verlässt der Mauscursor die Spalte wieder, so soll wieder zum ursprünglichen Zustand zurückgekehrt werden.



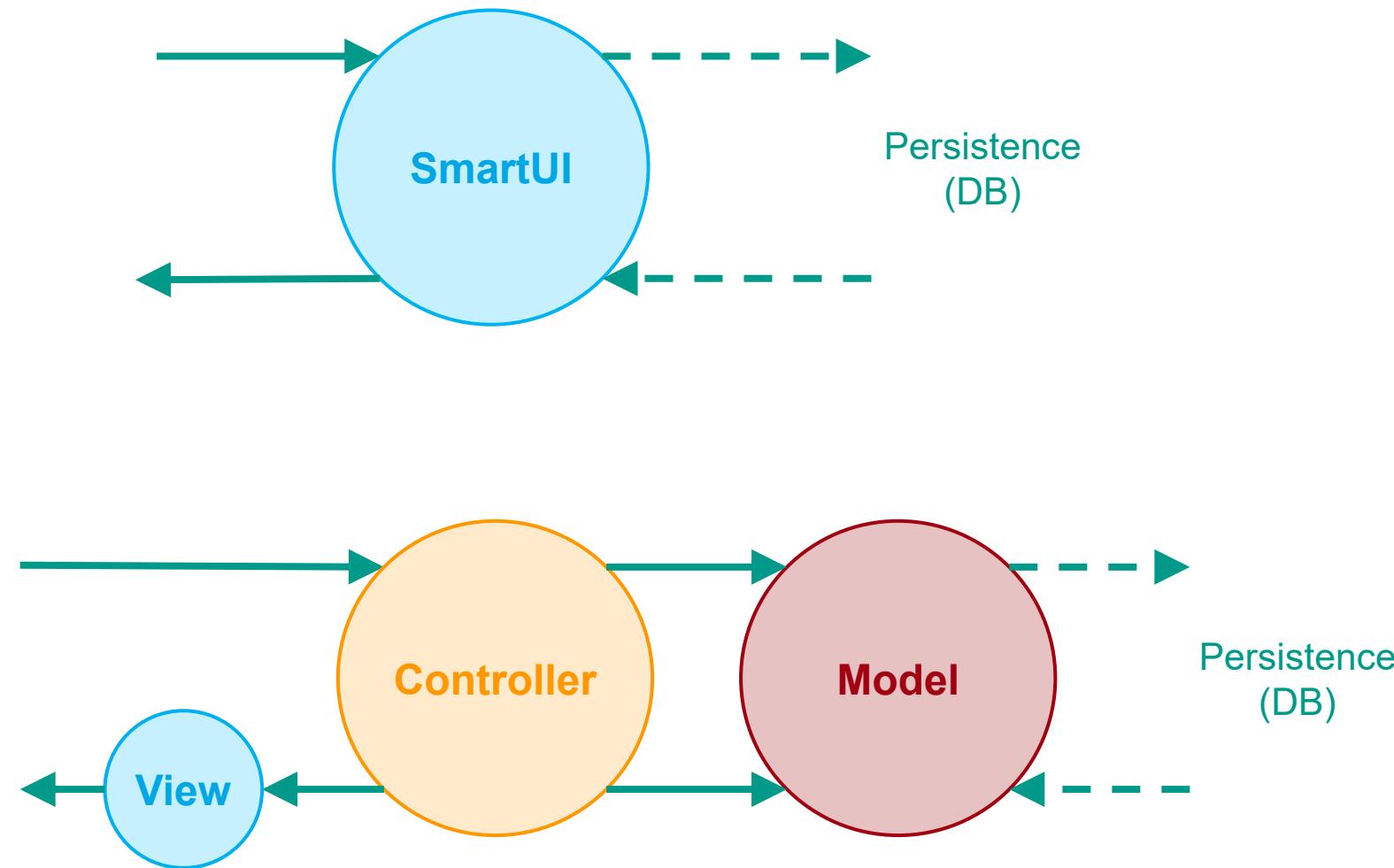




Nach: <https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html>



Nach: <https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html>



Nach: Gao, R.: Smart-UI architectural pattern

1. Übertragen Sie HTML, CSS und JavaScript (das später noch angesprochen wird) auf das Model-View-Controller-Konzept.

TypeScript: Allgemeines

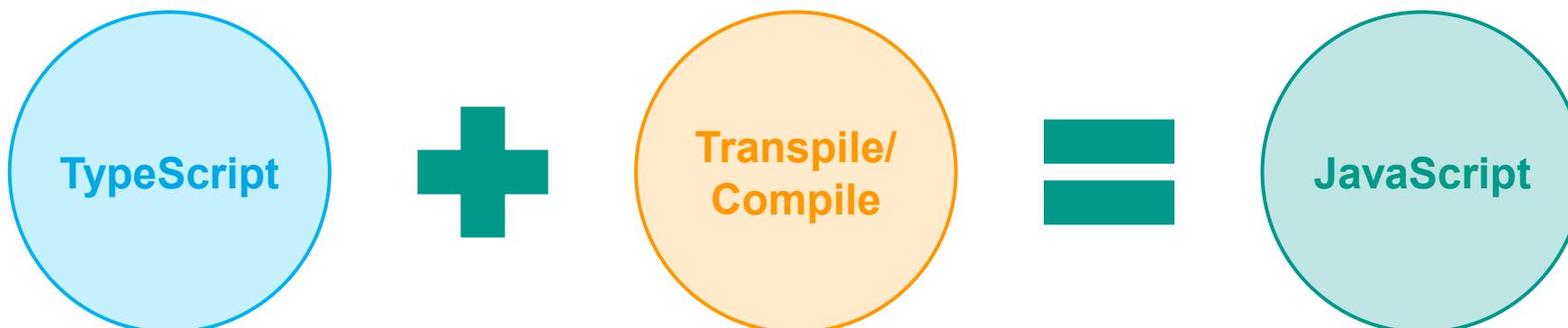
- Nachteile von JavaScript
 - verleitet häufig zu „unsauberem“ Programmcode
 - Unsicherheit durch schwache/ nicht vorhandene Typisierung
 - schlecht umgesetzte Objektorientierung
 - aufwändige/ schwierige Fehlersuche
- TypeScript versucht diese Nachteile zu mindern
 - Obermenge von JavaScript/ ECMAScript
 - Erweiterung von JavaScript durch bessere Umsetzung der Objektorientierung oder stärkerer Typisierung
 - → zunächst muss der Elternknoten gefunden werden, um dann dessen Kind(er) zu entfernen
- 2012 Vorstellung der Version 0.8 von Microsoft
- 2013 Plugins für Eclipse (zur Unterstützung der Entwickler)
TypeScript 0.9 → Unterstützung generischer Typen
- 2014 TypeScript 1.0 (zudem Integration in Visual Studio)
- 2016 TypeScript 2.0 (Funktionen zur besseren/ sichereren Unterstützung der Typisierung)
- 2018 TypeScript 3.0 (u.a. neuer (primitiver) Datentyp **unkown**)
- 2020 **TypeScript 3.9**

TypeScript: Allgemeines

- TypeScript erweitert JavaScript vor allem um statische Typisierung
→ optionale Typisierung
- strukturelle Typisierung
- Typinferenz
- Klassen, klassenbasierte Vererbung und Dekoratoren
- Typparameter (Generics)
- abstrakte Schnittstellentypen (Interface)
- zusätzliche Typen
 - `any`, `void`, `unknown`
 - `tuple`
 - Funktionstypen
 - Vereinigungstypen: `union`
 - Kreuzungstypen: `intersection`
 - Aufzählungstypen: `enum`
- Mehrfachvererbung
- Typaliase
- Namensräume

TypeScript: Allgemeines

- TypeScript ist keine statisch typisierte Sprache
→ TypeScript ermöglicht eine (optionale) starke Typisierung
- Variablen und Methoden können typisiert werden, müssen aber nicht
- Ziel: weniger Laufzeitfehler
- TypeScript und JavaScript sind stets 100 % kompatibel
→ JavaScript-Code ist automatisch gültiger TypeScript-Code
→ JavaScript-Quelldateien können durch TypeScript-Compiler verarbeitet werden
→ jede JavaScript-Bibliothek kann mit TypeScript verwendet werden



- Zum Testen (und Spielen): [TypeScriptlang.org → Playground](https://www.typescriptlang.org/play/) (<https://www.typescriptlang.org/play/>)

TypeScript: Einrichtung

- Um TypeScript-Quellcode in JavaScript zu übersetzen, ist der TypeScript-Compiler notwendig
- Dieser benötigt Node.js
→ <https://nodejs.org/>
- Einrichtung von TypeScript:
→ zunächst Node.js (und Packetmanager npm) installieren
→ danach mit Hilfe von npm TypeScript installieren

```
>node -version
v10.15.0
>npm -version
v6.4.1
>npm imstall - typescript
>tsc -version
V3.2.2
```

- Compilierung von JavaScript nach TypeScript:

```
>tsc helloworld.js
```

TypeScript: Einrichtung

- Beispiel: von TypeScript zu JavaScript

```
function ShowTime( toDayDate: Date ){  
}
```

timeBeispiel.ts

```
>ts timeBeispiel.ts
```

```
function ShowTime( toDayDate ){  
}
```

timeBeispiel.js

TypeScript: Feature: Typisierung

- primitive Datentypen
 - **Boolean**
 - **Number**
 - **String**
 - **Null**
 - **undefined**
 - **Symbol**
 - **void** wenn auf das „Nichtvorhandensein“ eines Typs hingewiesen werden soll
 - **any** bei unbekanntem Datentyp

```
let betrag: number    = 42.23;
let kunde: string     = "Mueller";
let bezahlt: Boolean = false;
```

TypeScript: Feature: Typisierung

- Arrays

```
let list: number[] = [ 1, 2, 3 ];
```

oder:

```
let list: Array< number > = [ 1, 2, 3 ];
```

- Tupel

```
let tupel: [ number, number ];
```

oder auch mit verschiedenen Datentypen, z.B.:

```
let tupel: [ string, number ];
```

- Aufzählungen

```
enum Color{ Red, Green, Blue };
```

...

```
let c: Color = Color.Green;
```

- eigene nichtprimitive Datentypen auf Basis von **object** als Kombination aus Schlüssel-Wert-Paaren

```
let myTyp = {  
    myStatus: true,  
    lastName: "Hammelmeir"  
};
```

TypeScript: Feature: Typisierung

- nichtprimitive Datentypen: auf Basis von **object**
→ Kombination aus Schlüssel-Wert-Paaren

```
let myTyp = {  
    myStatus: true,  
    lastName: "Hammelmeir"  
};
```

- nichtprimitive Datentypen als Klassen (wie in C++ oder Java)

```
class Person{
    static status: boolean = true;
    // Felder
    protected name: string;
    protected lastname: string;
    // Konstruktor
    constructor( _name: string ){
        this.name = _name;
    }
    // Methoden
    check(){
        ...
    }
    print(){
        ...
    }
}
...
let person1 = new Person( "Klaus" );
...
```

```
class Student extends Person{
    // Felder
    private matrikelnr: Number;
    print(){
        super.print();
    }
}
...
```

TypeScript: Feature: Typisierung

- Zum Vergleich die TypeScript-Codebeispiele hier als JavaScript (im Auszug)

```
var __extends = (this && this.__extends) || (function () {
    var extendStatics = function (d, b) {
        extendStatics = Object.setPrototypeOf ||
            ({ __proto__: [] } instanceof Array && function (d, b) { d.__proto__ = b; }) ||
            function (d, b) { for (var p in b) if (b.hasOwnProperty(p)) d[p] = b[p]; };
        return extendStatics(d, b);
    };
    return function (d, b) {
        extendStatics(d, b);
        function __() { this.constructor = d; }
        d.prototype = b === null ? Object.create(b) : (__.prototype = b.prototype, new __());
    };
})();
var Person = /** @class */ (function () {
    // Konstruktor
    function Person(_name) {
        this.name = _name;
    }
    // Methoden
    Person.prototype.check = function () {
        ...
    }
});
```

TypeScript: Feature: Typisierung

- Interfaces

```
interface Rocket{  
    liftOff();  
}  
...  
  
class saturnV implements Rocket{  
    ...  
    liftOff(){  
        // code here for a moon mission  
        ...  
    }  
}
```

TypeScript: Weitere Features

- (logische) Gruppierung des Sourcecodes mit Namensräumen

```
namespace MyNamespace {  
...  
}  
...
```

- (logische) Gruppierung des Sourcecodes mit (Internen und externen) Modulen
→ mittlerweile abgelöst durch die Nutzung von **namespace**

```
export interface Rocket{  
    liftOff();  
}
```

rocket.ts

```
import theRocket = require( 'Rocket' )  
class saturnV implements theRocket.Rocket{  
    ...  
    liftOff(){  
        // code here for a moon mission  
        ...  
    }  
}
```

saturnV.ts

FOM Hochschule

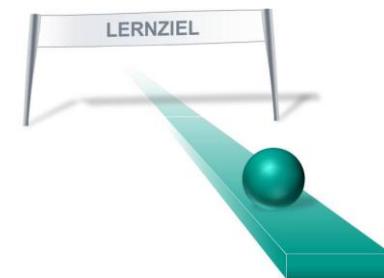
Web Technologie

(7) Praxis

1	Formales, Vorstellung
2	Einführung
2.1	Allgemeine Einführung
2.2	Historie Internet, Web & Web-Technologien
3	(Web) Server – Client Kommunikation
3.1	Protokolle & Architekturen
3.2	Web- & Microservices
3.3	Authentifizierungs- und Authorisierungskonzepte
4	Grundlagen der Mediengestaltung
4.1	Media & Multimedia
4.2	Media & Modalitäten – Interaktion & Wahrnehmung
4.3	Hypermedia
5	Statische Webseiten
5.1	HTML
5.2	CSS
5.3	Medieneinbindung
6	Dynamische Webseiten
6.1	DOM, HTML & CSS
6.2	JavaScript & JSON
6.3	Weitere Konzepte & Sprachen der Webprogrammierung
7	Praxis
7.1	Vorstellung eines (ausgewählten) (JavaScript-) Frameworks
7.2	Vorstellung einer (ausgewählten) Thematik für ein eigenes Projekt
7.3	Betreute, eigenständige Entwicklung des (ausgewählten) Themas
8	Wiederholung (opt.)

Im Anschluss an diesen Themenblock sollen Sie:

- Einen Überblick über aktuelle (JavaScript-) Entwicklungsframeworks und deren Ziele und Aufbau erworben haben.
- Den Umgang mit (mindestens) einem (JavaScript-) Framework zumindest grundlegend beherrschen.
- Ein (angemessen umfangreiches) eigenes Webprojekt realisiert haben.



- Bibliotheken („Programm“bibliotheken)
 - Enthalten Unterprogramme mit Hilfsfunktionen zur Erleichterung der Programmierung
 - i.d.R. entwickelt für ein Thema oder ein Verwendungsziel, z.B. Grafikbibliotheken
→ D3.js für die Datenvisualisierung mit Tabellen, Diagrammen, usw.
 - werden in ein Programm eingebunden, um auf entsprechenden Funktionen einer Programmbibliothek zuzugreifen, sobald das Programm sie benötigt
→ Bibliotheken „arbeiten“ nur innerhalb eines Programms und werden nicht selbstständig ausgeführt
- Framework („Rahmenstruktur“)
 - spezielle Form einer Klassenbibliothek zur Entwicklung neuer, eigenständiger Anwendungen
 - stellt die Architektur einer Anwendung als deren „Grundgerüst“ dar und bestimmt damit auch den Entwicklungsprozess
 - bietet Entwurfsmuster (Design Patterns) mit verschiedenen Funktionen für die Implementierung an
 - Beispiele: Zend-Framework für PHP, AngularJS für JavaScript, usw.

Ein Framework stellt eine wiederverwendbare, gemeinsame Struktur für Anwendungen zur Verfügung:

Entwickler binden ein Framework in die eigene Anwendung ein und erweitern es so, dass es ihre Anforderungen erfüllt.

Johnson, Foote (1988): Designing Reusable Classes

- Inversion of Control
 - Bei Bibliotheken wird der Code von Programmierern über die Programmierschnittstelle einer Software aufgerufen.
 - Frameworks vollziehen hingegen eine Umkehrung der Steuerung („Inversion of Control“): Der Code wird in festgelegte Strukturen des Frameworks eingebettet und aufgerufen, sobald er benötigt wird.
 - Zusammenfassend kann man sagen, dass Bibliotheken von dem Programm aufgerufen werden, wohingegen ein Framework dem Programm Vorgaben macht.

- Quelloffene Bibliothek mit vielen Plug-ins
- Bestandteil vieler Content-Management-Systeme (z.B. WordPress, Drupal, Joomla!, usw.)
- dient als „komfortable“ Schnittstelle zum DOM und bietet dazu diverse Funktionen
 - CSS3-Selektoren zur unkomplizierten Auswahl und Manipulation von Webseiten-Elementen
 - Einfaches Einbinden von Ajax-Anfragen

Vorteile jQuery	Nachteile jQuery
Ajax-Support	im Vergleich zu CSS häufig sehr langsam
Hohe Popularität	Neigung zu „Spaghetti-Code“
Einfach zu nutzen	neuere JavaScript-Funktionen machen jQuery oft überflüssig
Großer Umfang	
Zahlreiche Plug-ins	

- (freie) Erweiterung für jQuery zur einfachen Gestaltung von Benutzeroberflächen
- Schwerpunkte:
 - einfache Gestaltung grafischer Effekte (z.B. Zooming von Elementen)
 - leichte Implementierung von flexiblen Benutzerinteraktionen (z.B. Drag-and-Drop)
 - Animationen
 - Widgets (z.B. Autocomplete, Slider, Datepicker etc.).
- Eigener Grafikeditor „ThemeRoller“ (z.B. für die Entwicklung eigener Themes)

Vorteile jQuery UI	Nachteile jQuery UI
einfache Handhabung	nur langsame Weiterentwicklung
Eigener Grafikeditor	Nicht eigenständig, sondern erfordert jQuery

- Bibliothek für die Erstellung von Benutzeroberflächen
- Client- und serverseitiger Einsatz möglich und für App-Entwicklung nutzbar
- virtuelles DOM (erleichtert (auch) das Testen von Webanwendungen)
- One-direction-data-flow:
Änderungen in hierarchisch tieferliegendem Code können den höherstehenden nicht beeinflussen
- Erste Nutzung im Facebook-Newsfeed (2011)

Vorteile React	Nachteile React
Virtual DOM	Hohe Komplexität, daher zu erlernen
One-way-data-flow	
Serverseitiges Rendering	
Für Mobile-Apps einsetzbar	

- Suite aus vier verschiedenen, unabhängigen Bibliotheken
 - EaselJS Grafik und Interaktivität
 - TweenJS Animationen
 - SoundJS Audio-Funktionen
 - PreloadJS Preloading von Daten
- es müssen nicht alle vier Bibliotheken in einem Projekt implementiert werden, wenn nur benötigt wird
- Schwerpunkt ist die Entwicklung von **HTML5**- und **Flash-Anwendungen**

Vorteile CreateJS	Nachteile CreateJS
Unabhängige Bibliotheken	Kleine Community, daher relativ unbekannt
Verfügbare Entwicklungswerkzeuge	
Integration in Adobe Animate	

- Von Google entwickeltes Framework mit großer Community
- basiert auf jQuery Lite (abgespeckte jQuery-Variante)
- dient zur Erstellung von Single-Page-Webanwendungen
- Verfolgt den MVVM-Ansatz
- Clientseitiges Rendering der Webanwendung
- Nachfolger von AngularJS: Angular (basiert auf TypeScript und nicht mehr auf JavaScript)

Vorteile AngularJS	Nachteile AngularJS
Sehr große Community	inzwischen durch Angular ersetzt
Teil des MEAN-Software-Stack	hohe Komplexität, daher zu erlernen

Vorteile Angular	Nachteile Angular
Mehr Möglichkeiten durch TypeScript	Keine einfache Migration von AngularJS zu Angular
Einfacheres Cross-Platform-Development	noch komplexer als Angular JS

- Einfaches Framework, das bewusst so entworfen wurde, dass auch Anfänger leicht in die (Web-) Entwicklung einsteigen können
- Möglichkeit zur Einbindung von Templates (in HTML)

Vorteile Vue.js	Nachteile Vue.js
Schnell zu erlernen	kleine Community
Unterstützt HTML & CSS	

1. Auswahl eines (JavaScript-) Frameworks für die Durchführung eines Projekts.

Auswahl des zu nutzenden Frameworks (nach Vorgaben des Dozenten)

1. Übungsprojekt(e)

Vorstellung des Themas/ Inhalts eines Übungsprojekts

FOM Hochschule

Web Technologie

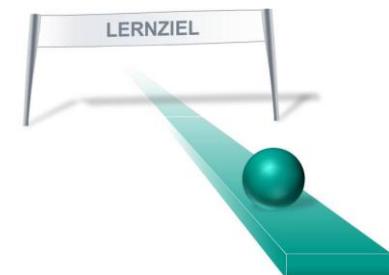
(8) Wiederholung

1	Formales, Vorstellung
2	Einführung
2.1	Allgemeine Einführung
2.2	Historie Internet, Web & Web-Technologien
3	(Web) Server – Client Kommunikation
3.1	Protokolle & Architekturen
3.2	Web- & Microservices
3.3	Authentifizierungs- und Authorisierungskonzepte
4	Grundlagen der Mediengestaltung
4.1	Media & Multimedia
4.2	Media & Modalitäten – Interaktion & Wahrnehmung
4.3	Hypermedia
5	Statische Webseiten
5.1	HTML
5.2	CSS
5.3	Medieneinbindung
6	Dynamische Webseiten
6.1	DOM, HTML & CSS
6.2	JavaScript & JSON
6.3	Weitere Konzepte & Sprachen der Webprogrammierung
7	Praxis
7.1	Vorstellung eines (ausgewählten) (JavaScript-) Frameworks
7.2	Vorstellung einer (ausgewählten) Thematik für ein eigenes Projekt
7.3	Betreute, eigenständige Entwicklung des (ausgewählten) Themas
8	Wiederholung (opt.)

Im Anschluss an diesen Themenblock sollen Sie:

- Die in diesem Modul besprochenen Inhalte reflektieren und selbständig erklären können.

Die Wiederholung ist optional.



FOM
Hochschule