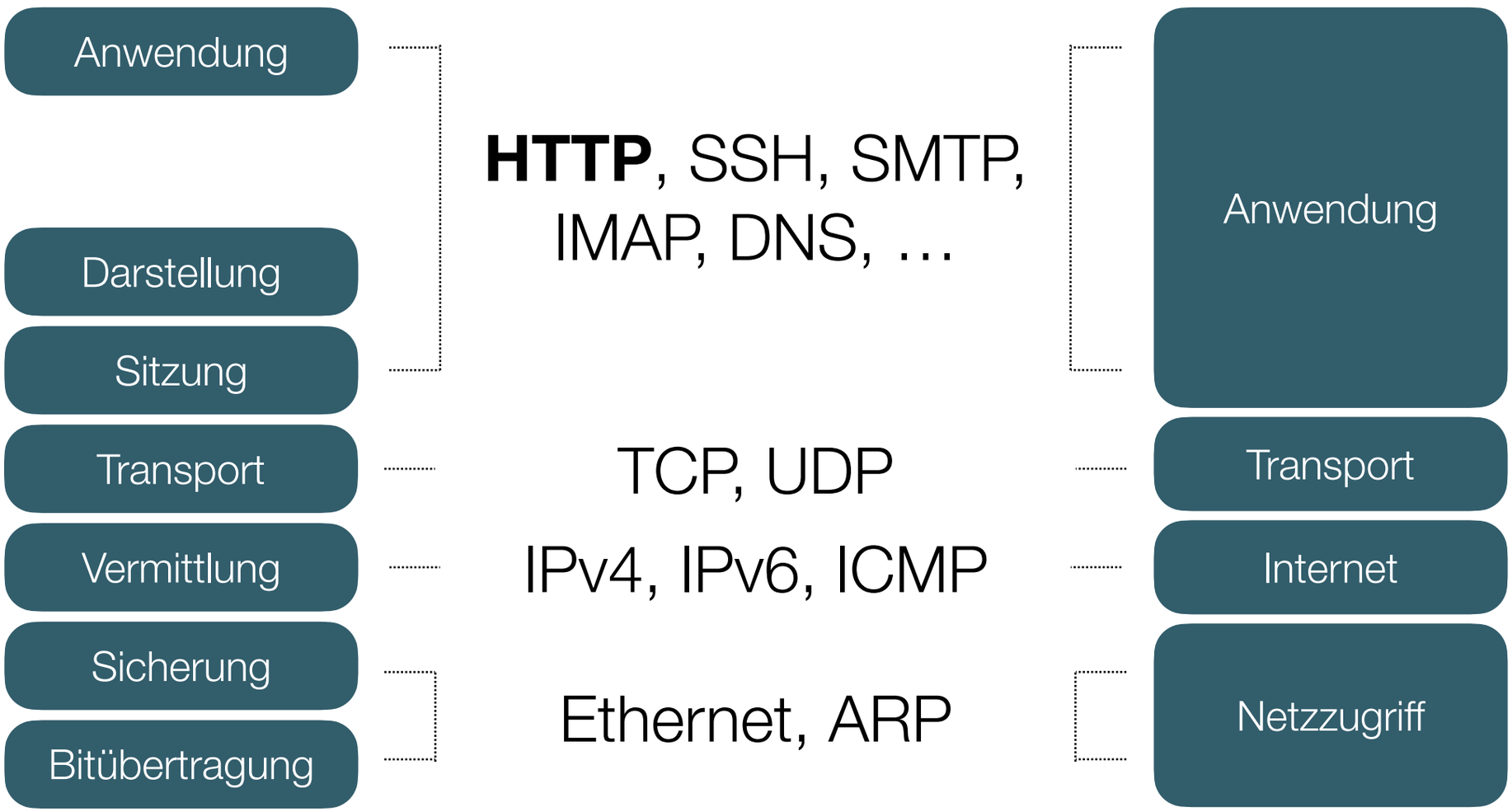
Modul Verteilte Systeme

HyperText Transfer Protocol

Peter Tröger
Beuth Hochschule für Technik Berlin
Sommersemester 2020
(Version 1)

Netzwerke heute



*„Well, I found it frustrating that in those days, there was different information on different computers, but you **had to log on to different computers to get at it**. Also, sometimes you had to **learn a different program on each computer**. So finding out how things worked was really difficult. …*

*Because people at CERN came from universities all over the world, they brought with them **all types of computers**. Not just Unix, Mac and PC: there were all kinds of big mainframe computer and medium sized computers running all sorts of software.*

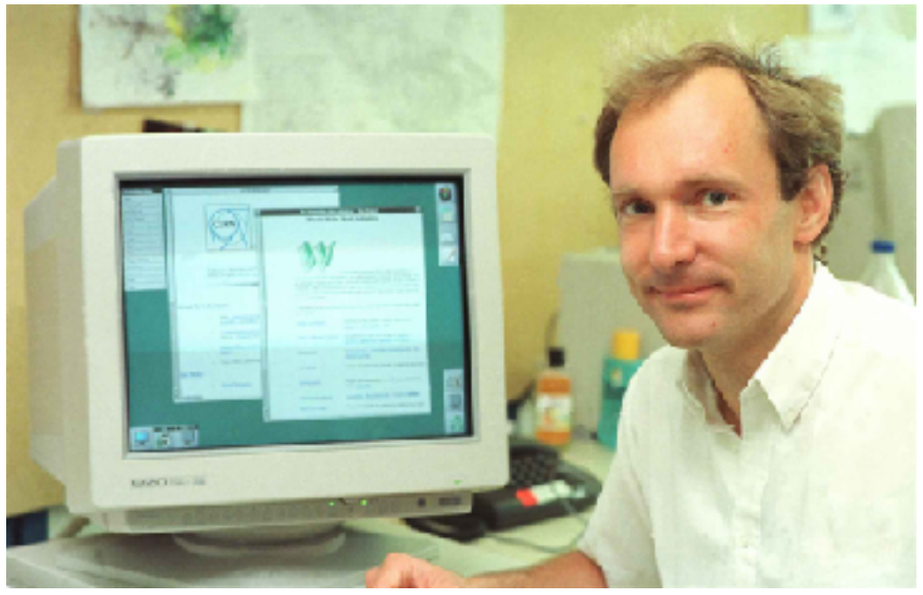
*… "Isn't there a better way? Can't we just fix this problem for good?" That became "**Can't we convert every information system so that it looks like part of some imaginary information system which everyone can read?**" And that became the WWW.“*

–Tim Berners Lee

(https://www.w3.org/People/Berners-Lee/Kids.html)

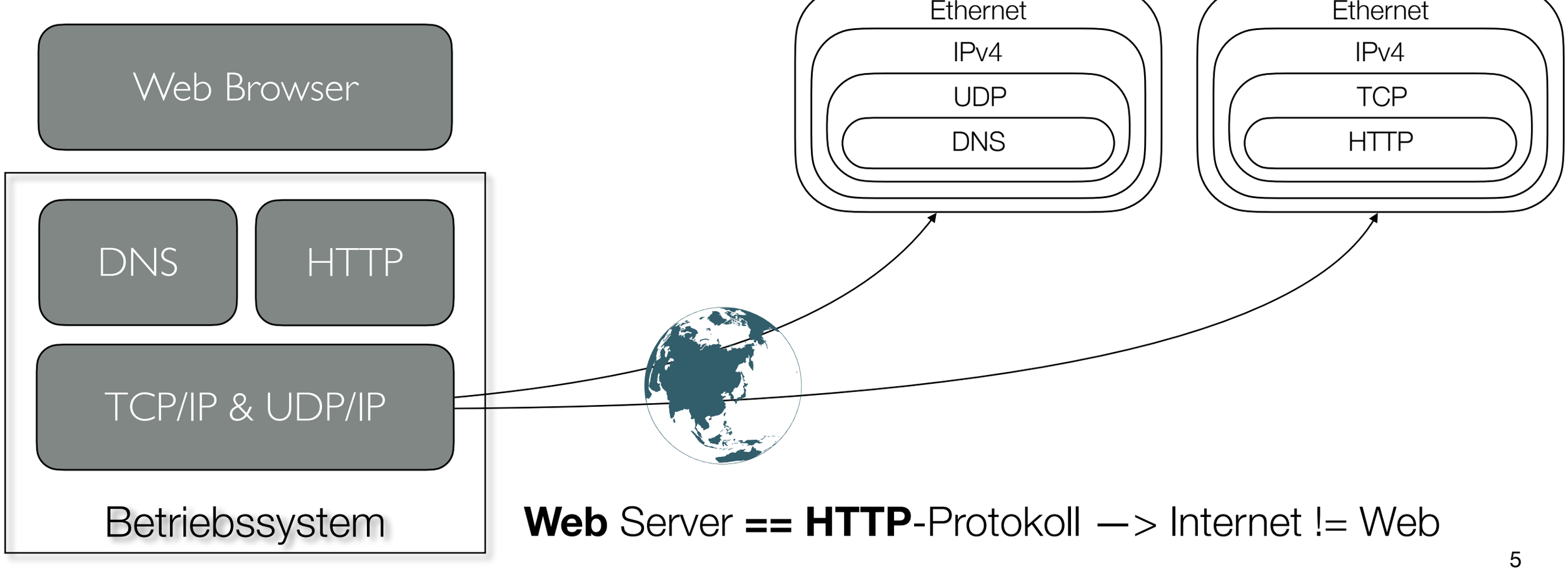
World Wide Web

- Informationsraum zum Austausch von Dokumenten
- Idee von Tim Berners-Lee am CERN in den 90er Jahren
- Nutzte etablierte Internet-Technologien (TCP/IP, DNS) als Ausgangspunkt
- Drei wesentliche Grundkonzepte
  - ***Hypertext Transfer Protocol (HTTP)***
  - ***Uniform Resource Identifier (URI)***
  - *HyperText Markup Language (HTML)*



Surfen im World Wide Web

„http://www.beuth-hochschule.de“



Web Server **== HTTP**-Protokoll —> Internet != Web

Surfen im World Wide Web

- Aufruf einer Webseite im Browser umfasst verschiedene Aktionen
  - Ggf. Rundruf mit **ARP** Protokoll, um MAC-Adresse des Routers zu erhalten
  - Ggf. Anfrage an Namensdienst, um IP-Adresse für diese Domain zu erhalten —> **DNS Protokoll**, basierend auf UDP/IP
  - Anfrage an Web Server, um Inhalt der Webseite zu erhalten —> **HTTP Protokoll**, basierend auf TCP/IP
- DNS und HTTP arbeiten auf Grundlage der Transportschicht
- Protokolle auf Layer 1 / 2 nicht sichtbar - WLAN, Ethernet, USB, Mobilfunk, …

Hypertext Transfer Protocol

- Aktuell üblich: Version 1.1 (RFC 2068 + Ergänzungen, seit 1997)
- Client-Server Protokoll
  - Client versendet eine Anfragenachricht an den Server (***HTTP request***)
    - Client hier als *User Agent* bezeichnet
    - Beispiele: Web Browser, Google Bot, Apps, Download-Tool, …
  - Server sendet eine Antwortnachricht (***HTTP response***)
- Nachricht: Mischung aus ASCII-Text und binären Daten
- Unterteilung in Nachrichtenkopf (*header*) und Nachrichtenkörper (*body*)

Demo: HTTP / HTTPS Wireshark, Browser

- Header vs. Body
- HTML Quelltext im Browser

HTTP Anfrage

- *Zeile 1*: Anfragemethode, gewünschte **Resource**, Protokollversion
- *Zeilen 2 bis n*: Weitere Informationen im Nachrichtenkopf
- *Zeile n+1*: leer
- Danach Anfragekörper (***request body***), ggf. leer

- Unverschlüsselte TCP/IP - Verbindungen für HTTP auf Port 80
- Verschlüsselte TCP/IP - Verbindung für HTTPS auf Port 443
- Bestehende TCP/IP-Verbindung für mehrere HTTP-Anfragen nutzbar

Wi-Fi: en0

http

No.	Time	Source	Destination	Protocol	Length	Info
534	14.451057	192.168.178.62	193.141.3.68	HTTP	469	GET / HTTP/1.1
577	15.147686	193.141.3.68	192.168.178.62	HTTP	1505	HTTP/1.1 200 OK (text/html)
581	15.248199	192.168.178.62	193.141.3.68	HTTP	447	GET /plugins/content/xtypo/assets/scr
604	15.271685	192.168.178.62	193.141.3.68	HTTP	458	GET /media/djextensions/picturefill/p
605	15.271915	192.168.178.62	193.141.3.68	HTTP	462	GET /media/djextensions/jquery-easing/
608	15.272430	192.168.178.62	193.141.3.68	HTTP	479	GET /components/com_djmediatools/layou
611	15.288925	193.141.3.68	192.168.178.62	HTTP	1155	HTTP/1.1 200 OK (application/javascr

```
▶ Frame 534: 469 bytes on wire (3752 bits), 469 bytes captured (3752 bits) on interface 0
▶ Ethernet II, Src: Apple_30:b6:20 (1c:36:bb:30:b6:20), Dst: AvmAudio_65:07:ff (c8:0e:14:65:07:ff)
▶ Internet Protocol Version 4, Src: 192.168.178.62, Dst: 193.141.3.68
▶ Transmission Control Protocol, Src Port: 54829, Dst Port: 80, Seq: 1, Ack: 1, Len: 403
▼ Hypertext Transfer Protocol
  ▶ GET / HTTP/1.1\r\n
    Host: der-flix.de\r\n
    User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:67.0) Gecko/20100101 Firefox/67.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,de;q=0.7,en;q=0.3\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
  ▶ Cookie: c4d0945e02287043809c77c23c0d18ae=si70u74h1jpq9livaqhfif5c64\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
    [Full request URI: http://der-flix.de/]
    [HTTP request 1/2]
    [Response in frame: 577]
    [Next request in frame: 581]
```

HTTP Antwort

- *Zeile 1*: Protokollversion, Status-Code
- *Zeilen 2 bis n*: Weitere Informationen im Nachrichtenkopf
- *Zeile n+1*: leer
- Danach Antwortkörper (***response body***), ggf. leer
- Auch bei Nutzung der gleichen TCP/IP-Verbindung wird jede HTTP-Anfrage vom Server separat behandelt
- Wichtige Grundlage für Lastbalanzierung

Capturing from Wi-Fi: en0

http

No.	Time	Source	Destination	Protocol	Length	Info
17	2.664598	192.168.178.62	193.141.3.68	HTTP	469	GET / HTTP/1.1
46	3.315050	193.141.3.68	192.168.178.62	HTTP	1505	HTTP/1.1 200 OK (text/html)
48	3.365409	192.168.178.62	193.141.3.68	HTTP	493	GET /media/djmediatools/css/slideshow_12610ed8
57	3.389941	193.141.3.68	192.168.178.62	HTTP	200	HTTP/1.1 200 OK (text/css)
67	3.424706	192.168.178.62	193.141.3.68	HTTP	526	GET /templates/der_flix_1/modules.js HTTP/1.1
74	3.446217	193.141.3.68	192.168.178.62	HTTP	378	HTTP/1.1 200 OK

```
▶ Frame 46: 1505 bytes on wire (12040 bits), 1505 bytes captured (12040 bits) on interface 0
▶ Ethernet II, Src: AvmAudio_65:07:ff (c8:0e:14:65:07:ff), Dst: Apple_30:b6:20 (1c:36:bb:30:b6:20)
▶ Internet Protocol Version 4, Src: 193.141.3.68, Dst: 192.168.178.62
▶ Transmission Control Protocol, Src Port: 80, Dst Port: 53759, Seq: 18278, Ack: 404, Len: 1439
▶ [16 Reassembled TCP Segments (19716 bytes): #21(1440), #22(1440), #23(1440), #24(61), #29(1440), #30(1379), #32(1440), #33(1
▼ Hypertext Transfer Protocol
  ▶ HTTP/1.1 200 OK\r\n
    Date: Sat, 22 Jun 2019 15:49:10 GMT\r\n
    Server: Apache/2.4.39 (Unix)\r\n
    X-Powered-By: PHP/7.0.33\r\n
    Expires: Wed, 17 Aug 2005 00:00:00 GMT\r\n
    Last-Modified: Sat, 22 Jun 2019 15:49:11 GMT\r\n
    Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0\r\n
    Pragma: no-cache\r\n
    Vary: User-Agent\r\n
    Keep-Alive: timeout=3, max=100\r\n
    Connection: Keep-Alive\r\n
    Transfer-Encoding: chunked\r\n
    Content-Type: text/html; charset=utf-8\r\n
    \r\n
    [HTTP response 1/2]
    [Time since request: 0.650452000 seconds]
    [Request in frame: 17]
0000  48 54 54 50 2f 31 2e 31  20 32 30 30 20 4f 4b 0d   HTTP/1.1  200 OK.
0010  0a 44 61 74 65 3a 20 53  61 74 2c 20 32 32 20 4a   .Date: S at, 22 J
```

Ressourcen

- Ein HTTP-Server bietet **Ressourcen** an
  - Daten eines bestimmten Typs
  - Bekannte Beispiele: HTML-Text, Bilddatei, Videodatei, Programm, …
    - Empfang einer Ressource umgangssprachlich als „Download“ bezeichnet
    - Senden einer Ressource umgangssprachlich als „Upload“ bezeichnet
- Der Clou: HTTP ist es egal, welchen Typ die Ressource hat
- Gleiche Idee wie bei einer Datei in einem Betriebssystem

Uniform Resource Locator (URL)

- URL erlaubt die Referenzierung einer bestimmten Ressource
- Somit Teil jeder HTTP-Anfrage
- *Uniform Resource Identifier (URI)*: Eindeutiger Name für eine Ressource

```
urn:isbn:0451450523
```

- *Uniform Resource Locator (URL)*: URI, welche ein Zugriffsschema enthält

```
http://www.beuth-hochschule.de/fileadmin/tmpl/favicon.ico
```

- Der Informationsraum WWW wird durch URLs strukturiert
- Sollten langlebig und unveränderlich sein

Uniform Resource Locator

Ressourcen

- HTTP-Standard definiert nur das Protokoll für den Zugriff auf Ressourcen
- Verarbeitung und Darstellung ist Aufgabe der beteiligten Anwendungen
- Generische Zugriffsoperationen, auch als Verben (*verbs*) bezeichnet
  - **GET**: Empfang von Ressource + Zusatzinformationen (*Header*) vom Server
  - **HEAD**: Wie GET, aber nur der *Header* wird heruntergeladen
  - **POST / PUT**: Geänderte bzw. neue Fassung der Ressource hochladen
  - Weiterhin: OPTIONS, DELETE, TRACE, PATCH, CONNECT
- Anwendungen können neue Verben erfinden (Bsp. WebDAV)

HTTP GET

- Standardmethode bei der Anfrage von Webseiten
- Idempotente Methode
  - Download verändert die Eigenschaften der Ressource nicht
  - Einmaliger oder mehrmaliger Aufruf macht keinen Unterschied
  - Durch den Standard festgelegte Konvention
  - Grundlage u.a. für Suchmaschinen und Caching
  - Abweichende Implementierungen haben unerwünschte Effekte (Beispiel: `http://www.example.com/users/4736/delete`)

SPIEGEL ONLINE - Aktuelle Nac

www.spiegel.de

Beuth Moodle RTM

SPIEGEL ONLINE DER SPIEGEL SPIEGEL TV Anmelden

Menü | Politik Meinung Wirtschaft Panorama Sport Kultur Netzwelt Wissenschaft mehr

25. Mai 2018 Schlagzeilen | Wetter | DAX 12.878,49 | TV-Programm | Abo

FUSSBALL Liveticker SPIX Tippspiel

Importaufschlag

Welche Staaten Trumps Autozölle besonders treffen würden

Inspektor Konsole Debugger Stilbearbeitung Laufzeitanalyse Speicher Netzwerkanalyse Web-Speicher

Adressen durchsuchen | Alles HTML CSS JS XHR Schriften Grafiken Medien WebSockets Sonstiges Logs nicht

Status	Methode	Datei	Host	Urspr...	Typ	Übertragen	Größe	0 ms	639 m
200	GET	image-1290887-280_poster_16x9-htid-1290887.jpg	cdn4.spiegel.de	img	jpeg	15,95 KB	15,48 KB		
200	GET	image-878072-280_poster_16x9-vuyy-878072.jpg	cdn3.spiegel.de	img	jpeg	11,70 KB	11,24 KB		
200	GET	image-1143294-280_poster_16x9-tkjf-1143294.jpg	cdn1.spiegel.de	img	jpeg	13,55 KB	13,08 KB		
200	GET	image-1283841-640_panofree-tquc-1283841.jpg	cdn2.spiegel.de	img	jpeg	12,77 KB	12,31 KB		
200	GET	image-1292190-640_panofree-lddx-1292190.jpg	cdn1.spiegel.de	img	jpeg	69,18 KB	68,72 KB		
200	GET	image-1292213-280_poster_16x9-bjyo-1292213.jpg	cdn4.spiegel.de	img	jpeg	19,67 KB	19,20 KB		
200	GET	image-888442-280_poster_16x9-ncqm-888442.jpg	cdn3.spiegel.de	img	jpeg	17,81 KB	17,34 KB		
200	GET	image-1277493-280_poster_16x9-fzci-1277493.jpg	cdn4.spiegel.de	img	jpeg	11,71 KB	11,25 KB		
200	GET	390x85_Otto.jpg	www.spiegel.de	img	jpeg	11,43 KB	11,05 KB		
200	GET	390x85_Mytoysnew.jpg	www.spiegel.de	img	jpeg	8,10 KB	7,72 KB		
200	GET	Media.jpg	www.spiegel.de	img	jpeg	6,13 KB	5,75 KB		
200	GET	390x85_pull&bear.jpg	www.spiegel.de	img	jpeg	7,41 KB	7,03 KB		
200	GET	btn_refresh.png	www.spiegel.de	img	png	813 B	436 B		
200	GET	live.js	www.spiegel.de	script	js	34,43 KB	82,56 KB		
200	GET	a				40,92 KB	161,45 KB		
200	GET	/				787 B	494 B		
200	GET	a				796 B	333 B		
200	GET	5				704 B	154 B		
404	GET	image-51530-thumbbiga-rkabcfkba-1-76053.jpg	cdn2.spiegel.de	img	html	255 B	0 B		

111 Anfragen | 2,85 MB / 1,63 MB übertragen | Beendet: 3,11 s | DOMContentLoaded: 848 ms | load: 2,91 s

SPIEGEL ONLINE - Aktuelle Nac

www.spiegel.de

Beuth Moodle RTM

SPIEGEL ONLINE DER SPIEGEL SPIEGEL TV Anmelden

Menü | Politik Meinung Wirtschaft Panorama Sport Kultur Netzwelt Wissenschaft mehr

25. Mai 2018 Schlagzeilen | Wetter | DAX 12.902,99 | TV-Programm | Abo

FUSSBALL Liveticker SPIX Tippspiel

Importaufschlag

Welche Staaten Trumps Autozölle besonders treffen würden

Inspektor Konsole Debugger Stilbearbeitung Laufzeitanalyse Speicher Netzwerkanalyse Web-Speicher

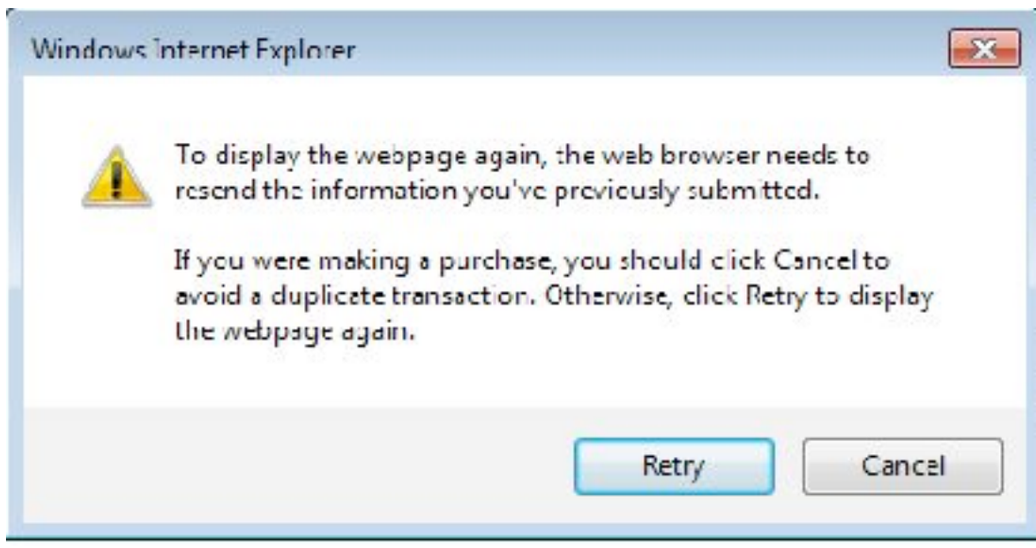
Adressen durchsuchen | Alles HTML CSS JS XHR Schriften Grafiken Medien WebSockets Sonstiges Logs nicht

Status	Methode	Datei	Host	Urspr...	Typ	Übertragen	Größe	0 ms
200	GET	/	www.spiegel.de	document	html	63,36 KB	301,14 KB	→ 73 ms
200	GET	style-V8-45.css	www.spiegel.de	stylesheet	css	Aus Cache	355,47 KB	
200	GET	global-V8-45.js	www.spiegel.de	script	js	Aus Cache	332 B	
200	GET	javascript-V8-45.js	www.spiegel.de	script	js	Aus Cache	0 B	
200	GET	interface-V8-45.js	www.spiegel.de	script	js	Aus Cache	0 B	
200	GET	netmind-V8-45.js	www.spiegel.de	script	js	Aus Cache	0 B	
200	GET	spaab-V8-45.js	www.spiegel.de	script	js	Aus Cache	0 B	
200	GET	live.css	www.spiegel.de	stylesheet	css	Aus Cache	20 B	
200	GET	xml-66885.json	www.spiegel.de	xhr	json	1,78 KB	4,52 KB	→ 20 ms
200	GET	ani-loader_small.gif	www.spiegel.de	img	gif	Aus Cache	2,49 KB	
200	GET	pixel.gif	www.spiegel.de	subdocument	gif	Aus Cache	43 B	
200	GET	spiegel-daily.png	www.spiegel.de	img	png	Aus Cache	5,37 KB	
304	GET	live.js	www.spiegel.de	script	js	Aus Cache	82,56 KB	→ 20 ms
200	GET	/status/	optout.adalliance.io	subdocument	html	Aus Cache	494 B	
200	GET							
200	GET							
200	GET							102 ms
404	GET							25 ms
200	GET	alltops.json	www.spiegel.de	xhr	json	493 B	184 B	

19 Anfragen | 753,07 KB / 101,28 KB übertragen | Beendet: 30,78 s | DOMContentLoaded: 480 ms | load: 1,10 s

HTTP POST

- Nicht idempotent, einmaliger vs. mehrmaliger Aufruf macht Unterschied
- Ändert Zustand der Ressourcen auf dem Web Server, daher kein Caching
- Gedacht zum Speichern von Daten
  - Typische Anwendung im WWW: Abschicken von Formulardaten
  - PUT-Methode: Zustand ändern, aber idempotent
- *Reload* einer Web-Seite im Browser nach POST-Anfrage führt deshalb üblicherweise zu einer Warnung
- Daten stehen im Anfragekörper



Einen Prüfungszeitraum hinzufügen

5. System-Sicherheit und Zuverlässigkeit Übg., B-TI/7/Zug 1, Notenliste 1

Einen Prüfungszeitraum hinzufügen

6. Verteilte Systeme, B-MI/3/Zug 1, Notenliste 1

2. Prüfungszeitraum	2018-09-24 10:00:00	D 139 L	bearbeiten	lösch
1. Prüfungszeitraum	2018-07-23 10:00:00	D 139 L	bearbeiten	lösch

Einen Prüfungszeitraum hinzufügen

7. Verteilte Systeme Übg., B-MI/3/Zug 1, Notenliste 1

Einen Prüfungszeitraum hinzufügen

8. Verteilte Systeme Übg., B-MI/3/Zug 1, Notenliste 2

Einen Prüfungszeitraum hinzufügen

Zustand

- GET-Anfragen sind idempotent, jede Anfrage steht für sich selbst
- „Wiedererkennung“ von anfragenden Nutzern eigentlich nicht möglich, IP-Adresse des Client kann sich bspw. ändern
- Wie lässt sich ein Einkaufskorb realisieren?
- Server muss beim Client einen Identifikator hinterlassen
- Bei der nächsten Anfrage an den gleichen Server wird der mitgeliefert
- Bekannt als **Cookies**
- Solche Informationen sind Teil des **Nachrichtenkopf**

HTTP Nachrichtenkopf

- Nachrichtenkopf (*header*) enthält **Felder** als Schlüssel-Wert-Paar
- **Anfragefelder** werden mit einer HTTP-Anfrage übertragen
  - Beispiele: *Authorization, Accept-Charset, Cookie*
- **Antwortfelder** werden mit einer HTTP-Antwort übertragen
  - Beispiele: *Content-Length, Expires, Date, Set-Cookie*
- Nicht-standardisierte Felder sind erlaubt, der jeweilige Empfänger (Client oder Server) darf diese dann ignorieren

Header: *Cookies*

- Server bittet den anfragenden Client, sich einen Wert zu merken
- Bei jedem späteren Aufruf der gleichen Domäne + des gleiche Pfads wird das Cookie automatisch als Teil der Anfrage im Header übermittelt
- Teilweise Missbrauch für Nachverfolgung von Nutzern bei Online-Werbung

```
GET /cgi/suche.py?q=cookie+aufbau HTTP/1.0
```

```
HTTP/1.0 200 OK
Set-Cookie: letzteSuche=Y29va2llIGF1ZmJhdQ==;
            expires=Tue, 29-Mar-2014 19:30:42 GMT;
            Max-Age=2592000;
            Path=/cgi/suche.py
```

Demo: Cookies

Header: *Content-Type*

- Verschiedene Arten von Daten (**= Ressourcen**) per HTTP übertragbar
- Header enthält deshalb Angaben zum Datentyp (***Content-Type***) des Body
  - Textdaten: *text/html* (Standard), *text/rtf, text/xml, text/csv, text/javascript, …*
  - Anwendungsdaten: *application/msword, application/pdf, …*
  - Multimediadaten: *image/png, video/mp4, audio/mpeg, …*
- IANA verwaltet zentral die Liste der bekannten Formate für Inhalte (https://www.iana.org/assignments/media-types/media-types.xhtml)
- Körper der Antwort enthält dann die eigentlichen Daten ohne weitere Kodierung
- Nicht unbedingt Textdaten —> ***Content-Length*** Eintrag im Header der Antwort

Header: *Accept*

- Client kann Wunsch für den Datentyp mit ***Accept*** - Header formulieren

- Beispiel:
  `Accept: text/html, application/xhtml+xml, application/xml;q=0.9, */*;q=0.8`

- Server wählt ein Format aus und teilt es via `Content-Type` - Header mit

- Gleicher Ansatz für Sprache und Textkodierung

Header: *Transfer Encoding*

- Teil des *Response Header*, als Reaktion auf entsprechende Anfrage
- `Transfer-Encoding: chunked`
  - Server teilt mit, dass die Daten in mehreren Teilen geliefert werden
    - Gesamtgröße ist dabei unbekannt (Bsp.: Komplexe Datenbankanfragen)
- `Transfer-Encoding: gzip`
  - Server liefert Daten gezippt aus, um Bandbreite zu sparen
  - Reaktion auf Anfrage mit `Accept-Encoding` Header vom Client

Header: *Cache-Control*

- Verschiedene Werte für Client und Server möglich
- Art des erlaubten Caching wird durch den Server in der Antwort mitgeteilt
  - `public, private, no-cache, no-store, must-revalidate, …`
- Server kann mitteilen, wie lange die Ressource gültig ist (`max-age`)
- Client kann ebenfalls Forderungen an den Cache stellen
  - `no-cache, no-store, …`

Header: *ETag*

- Eintrag in HTTP-Antwort, welcher die Version der Ressource beschreibt
- Vermeidet Konflikte bei parallelen Änderungen
  - Client kann per `If-Match` Header bestimmtes ETag im POST angeben
    - Änderung wird nur durchgeführt, wenn ETag beim Server übereinstimmt, sonst Rückgabewert 412 („Precondition Failed“)
- Optimierte Nutzung von Bandbreite
  - Client kann per `If-None-Match` - Header bestimmtes ETag in GET angeben
  - Server liefert Ressource nur aus, wenn ETag nicht übereinstimmt, sonst 304

HTTP Status Code

- HTTP-Antwort enthält immer Status-Code
  - **1xx:** Anfrage erhalten und verstanden, noch in Bearbeitung
  - **2xx:** Anfrage erhalten, verstanden und akzeptiert
  - **3xx:** Umleitung der Anfrage (*redirection*), Client muss woanders nachfragen
  - **4xx:** Fehler des Client, Anfrage ungültig
  - **5xx:** Fehler beim Server, Anfrage kann nicht beantwortet werden
- Standardisierte Codes in RFC7231, eigene Codes durch Produkte möglich
- **https://en.wikipedia.org/wiki/List_of_HTTP_status_codes**

HTTP Status Codes

- Übliche Codes bei der Web-Entwicklung
  - 200 „OK“: Anfrage erfolgreich
  - 301 „Moved Permanently“: Diese und alle zukünftigen Anfragen an andere URI umleiten
  - 302 „Found“: Diese Anfrage einmalig an andere URI umleiten
  - 304 „Not modified“: Resource hat sich nicht geändert, erneute Auslieferung nicht nötig
  - 400 „Bad request“: Server versteht die Anfrage nicht
  - 404 „Not Found“: Die Ressource ist nicht verfügbar
  - 403 „Forbidden“: Server hat die Anfrage verstanden, autorisiert sie aber nicht
  - 405 „Method Not Allowed“: HTTP-Verb ist nicht gestattet
  - 500 „Internal Server Error“: Unspezifisches Problem auf dem Server

Beispiel: 301

- Status Code 301:
  „Moved Permanently“
  - Umleitung einer angefragten URI zu neuer URI
  - Neues Ziel im Antwort-*Header*
  - Client darf das Ergebnis zwischenspeichern
  - Zusätzlich *Body* mit menschenlesbarer Erklärung, falls *User Agent* den Code nicht versteht

```
macmini:~ troeger$ telnet www.heise.de 80
Trying 193.99.144.85...
Connected to www.heise.de.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.heise.de

HTTP/1.1 301 Moved Permanently
Server: nginx
Content-Type: text/html
Location: https://www.heise.de/
Last-Modified: Fri, 25 May 2018 15:26:08 GMT
Cache-Control: public,
Content-Length: 178
Accept-Ranges: bytes
Date: Fri, 25 May 2018 15:26:15 GMT
Age: 7
Connection: keep-alive
Strict-Transport-Security: max-age=86400
X-Frame-Options: DENY
Vary: X-Forwarded-Proto,User-Agent,X-Export-Format
Export-Agent

<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

World Wide Web Consortium (W3C)

- Entwicklung und Pflege offener Standards für das Web (HTML, HTTP, …)
  - Dezentral: Kein Erlaubnis erforderlich, um Information zu veröffentlichen
  - Neutral: Kosten für den Zugang haben keinen Einfluss auf die Art der Information, die man erhalten kann
  - Universell: Informationsaustausch unabhängig von Technologien (Hardware, Betriebssystem, Browser-Hersteller, …)
  - Konsens: Standards werden gemeinsam entwickelt
- Viele dieser Prinzipien sind in letzter Zeit gefährdet …

Zusammenfassung

- HTTP ist Grundlage des World Wide Web
- Anfänglich nur für Auslieferung von HTML
- Heutzutage Universalprotokoll für verschiedenste Zwecke (Bilder, Streaming-Dienste, Telefonieprotokolle, Chat, …)
- Flexibilität durch einfache Konzepte: URI, Ressourcen, Verben, Header
- HTTP/1.1 noch immer Standard, aktuellere Fassungen wenig verbreitet
- Moderne Nutzung für entfernt nutzbare Programmierschnittstellen
  —> REST-API