Web-Engineering

Vorlesung im Sommersemester 2003
für das Studium “Informationstechnik”

Dr. Jürgen Röthig

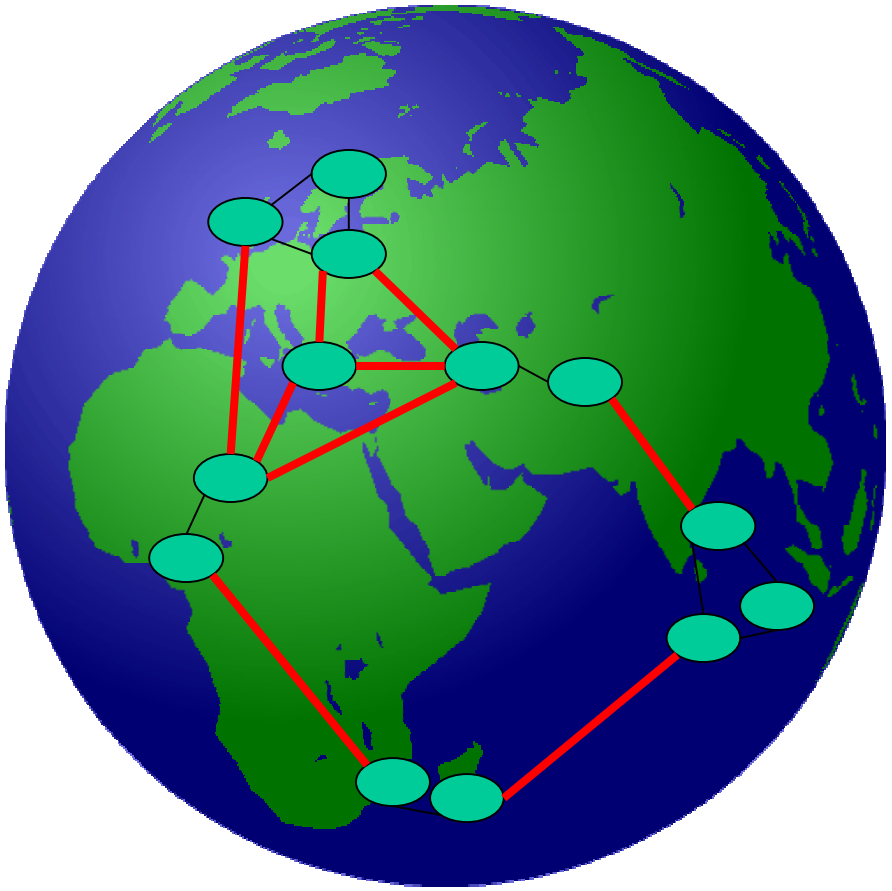
BA

Das Web

- (Kurzform für) World Wide Web
- weltumspannendes Netz von Informationsangeboten, die alle über das Internet auf einheitliche Weise (mit derselben Anwendung) abrufbar sind
- lokationsunabhängig - die Informationen lagern verteilt auf vielen “Servern” (Rechnern)
- medienunabhängig - das Web vereint viele verschiedene Medien-(Darstellungs-)formen
- verbindendes Element ist der “Hyperlink”

Das Internet

- Verknüpfung vieler lokaler Netzwerke (LAN) zu einem weltumspannenden Weitverkehrsnetz (WAN)
- heterogenes Netzwerk
  - unterschiedliche Leitungskapazitäten
  - unterschiedliche Netzwerktechniken
- „Killerapplikation“ WWW



Multimedia

- Medium/Medien
  - lateinisch: das mittlere
  - Duden: Mittler, Kommunikationsmittel
  - Wahrig Deutsches Wörterbuch: Mittler, Lehr- und Lernmittel (Buch, Film, Tonband, Fernsehen)
  - Bünting Deutsches Wörterbuch: Element zur Herstellung von Verbindungen oder Beziehungen, Übermittler von Informationen
  - Übertragungswege: Kupferkabel, Glasfaser, Athmosphäre
  - Nachrichtenkanäle: Zeitung, Rundfunk, Fernsehen
  - Darstellungsformen: Text, Bild, Audio, Video
- multus, Plural: multi
  - lateinisch: viele
- Kombination verschiedener Darstellungsformen

Begriffsbildung

- Web
  - (Kurzform für) World Wide Web
- Engineering
  - ingenieursmäßige Herstellungsweise von Produkten
- Multimedia
  - Verknüpfung verschiedener Darstellungsformen

Motivation

- Multimedia
  - “ein Bild sagt mehr als tausend Worte”
  - viele Dinge lassen sich am besten durch eine Kombination aus Text, Bildern, Audio- und/oder Videoclips beschreiben
- Lokationsunabhängigkeit
  - viele Informationen lagern traditionell an unterschiedlichen Orten (Bibliotheken, Mediatheken, …)
  - EDV-Systeme haben ebenfalls oft eine Spezialisierung auf bestimmte Aufgaben und Themenbereiche
- Hyperlinks
  - Verweise finden sich in vielfältigen Formen im täglichen Leben (und insbesondere in wissenschaftlicher Literatur)

Ziele der Vorlesung

(entsprechend Studienplan):

- Kennenlernen der Techniken und Methoden des Internets als Medium der geschäftsorientierten Kommunikation
- Kennenlernen und Einüben aktueller Web-Programmiersprachen und -methoden

Inhalte der Vorlesung I

(entsprechend Studienplan):

- Adressierungsformate
- Programmierung in Dokumenten
- Anwendung von Plugins
- Programmierung auf Servern
- Aktuelle Weiterentwicklung der Web-Standards
- Weitere Internet-Dienste, Möglichkeiten und Nutzen

Inhalte der Vorlesung II

(im ersten Semester):

- Überblick, Geschichte
- Adressierung:
  - IP-Adressen, URLs …
- Anwendungs-Protokolle (Kommunikationsprotokolle):
  - HTTP, (IP, TCP, UDP,) …
- Dokumentformate:
  - HTML, (GIF, JPG, PNG, MPEG,) …
- Hypertext und Hypermedia:
  - Referenzen/Links, Lokations- und Medienunabhängigkeit, HTML

Zeitleiste - “Wie neu ist neu?”

Telekommunikation & Internet I

- 1838/1844 Samule Morse & Alfred Vail: Telegraph Prinzip/Demonstration
- 1848 erste transatlantische Telegraphenverbindung
- 1876 Alexander Graham Bell: Telefon
- 1907 New York: erste Radioübertragung
- 1945 Vannevar Bush: Artikel “As We May Think” als Vorläufer der Ideen von Hypertext und WWW
- 1964 ANSI: ASCII 7bit-Code zur Textübertragung
- 1965 Theodore Nelson: “Hypertext” als Begriff
- 1965/1967/1969 ARPANET: Projekt Initiierung/Design/erster Prototyp (4 Rechner)
- 1969 RS232C: Standard für Datenaustausch zwischen Computern und Peripherie
- 1971 Ray Tomlinson/Bolt Beranek and Newman: erste Email per Netzwerk

Zeitleiste - “Wie neu ist neu?”

Telekommunikation & Internet II

- 1973 Vinton Cerf: Arbeit an TCP
- 1973 UCL & Royal Radar Establishment (Norwegen): erste internationale Anbindung im Arpanet
- 1973 Robert Metcalfe/Xerox Parc: Beschreibung von Ethernet als modifiziertes Alohanet, erste Implementierung
- 1982 Vint Cerf & Bob Kahn: TCP/IP als IETF RFC
- 1984 Internet: Einführung des DNS
- 1988 CERT: Gründung Computer Emergency Response Team
- 1989 Tim Berners-Lee: Vorschlag an CERN für WWW
- 1990 Tim Berners-Lee/CERN: erster Prototyp für WWW
- 1990 Arpanet: Auflösung
- 1992 Mbone: erste Audio- & Video-Übertragung im Internet
- 1993 Marc Andreessen: NCSA Mosaic Web Browser
- 1994 Netscape: Gründung und erster Browser
- 1995 Sun: Java

Zusammenfassung

- Elektr. Rechenmaschinen gibt es seit etwa 60 Jahren
- Vorformen (“Abacus”) und Grundlagen (“Algorithmen”) teilweise noch viel älter
- Computerkommunikation seit Anbeginn elektronischer Rechner notwendig (zur Anbindung von Peripherie)
- Das Internet entstand ab etwa 1965.
- Das World Wide Web entstand etwa 1990.
- Aber erst mit dem (kommerziellen) Erfolg des WWW (ab etwa 1995) wurde das Internet zum Begriff!

Das Internet

Noch einmal: Was ist das Internet?

- Zusammenschluß vieler lokaler Netze zu einem weltumspannenden Weitverkehrsnetz.

Gemeinsamkeiten?

- Kommunikation ist grundsätzlich zwischen allen Rechnern im Internet möglich.
- verschiedene Anwendungen für gemeinsam genutzte Dienste (z.B. WWW)
- Einheitliches Kommunikations-Protokoll

Das Internet-Protokoll

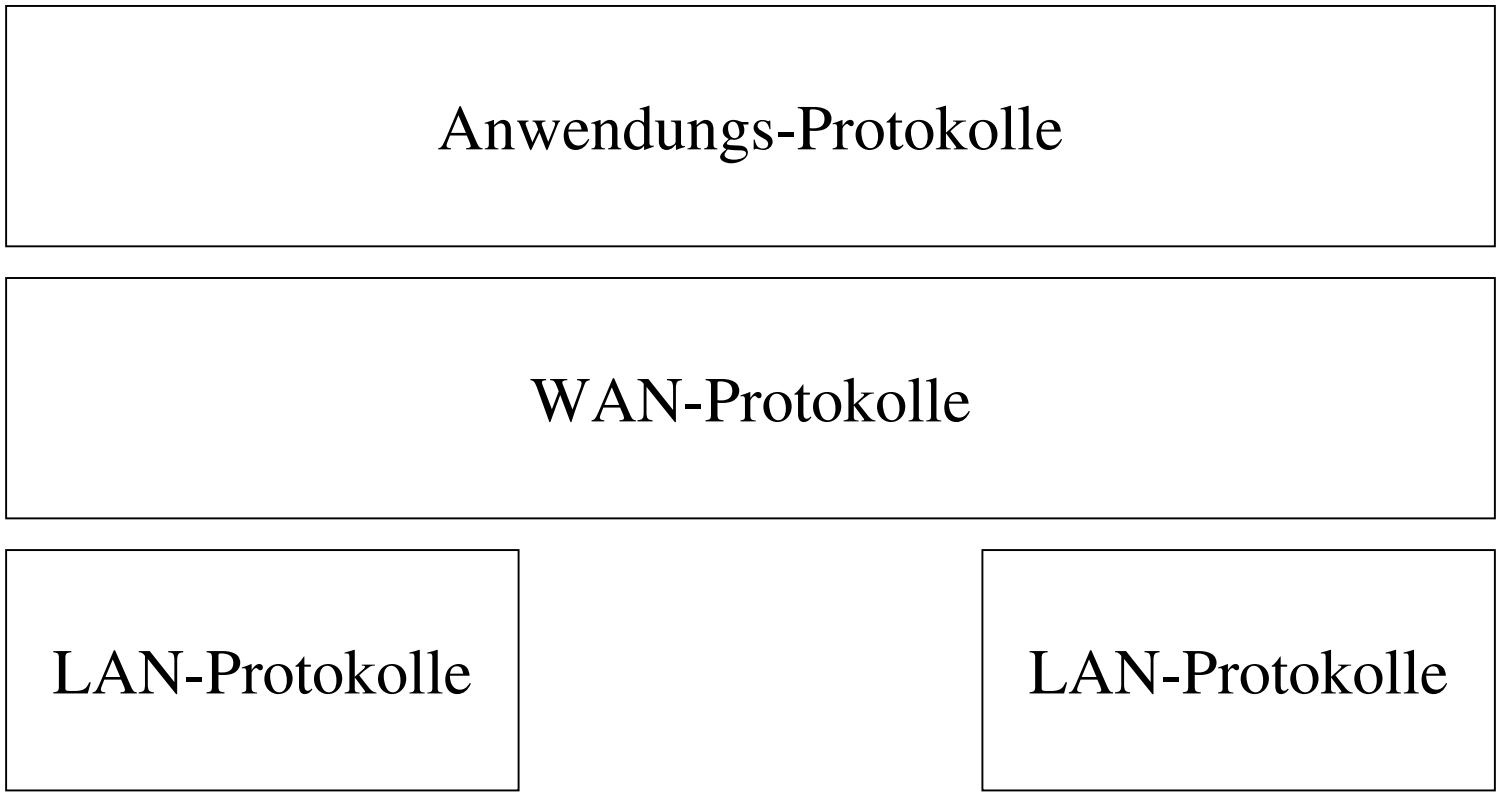
Was ist ein Protokoll?

- Sammlung von verbindlichen Regeln, wie die Kommunikation zwischen mehreren Teilnehmern (Rechnern) abläuft

Man unterscheidet (ganz grob) zwischen Netzwerk-Protokollen (und dabei zwischen den Protokollen im jeweiligen LAN sowie denen im WAN) und Anwendungsprotokollen

- Protokoll im lokalen Netzwerk abhängig von jeweiliger Netzwerktechnologie (Ethernet, Token Ring, …)
- Anwendungsprotokoll ist abhängig von der jeweiligen Anwendung
- Die Netzwerkprotokolle “dazwischen” (im Internet) sind einheitlich: Internet Protocol (IP) sowie Transport Control Protocol (TCP) bzw. User Datagram Protocol (UDP)

Kommunikations-Protokolle



Die Anwendungsprotokolle

- Jede Anwendung nutzt ein spezielles (auf die Anforderungen zugeschnittenes) Protokoll
- alle Anwendungsprotokolle nutzen die gemeinsamen Protokolle TCP/UDP/IP

Zwei Varianten:

- 1. Alle Kommunikationspartner sind gleichberechtigt.
- 2. Anwender (Client) nutzt den Dienst eines Dienstleisters (Server)

Anwendungen im Internet: Das World Wide Web

Bedeutung des World Wide Web (WWW)

- Die Killer-Applikation schlechthin im Internet
- Aber: das “World Wide Web” und das “Internet” sind nicht dasselbe!
- WWW ist nur eine Applikation (neben vielen), welche das Internet nutzt.
- Erst mit dem Aufkommen des WWW (um 1990) wurde das Internet weithin bekannt (etwa um 1995)
- Seither gibt es einen (nach wie vor anhaltenden) Internet-Boom.

Begriffsbildung

Das “Web”?

- Sammlung vieler Informationsangebote im Internet
- lokationsunabhängig - die Informationen lagern verteilt auf vielen “Servern” (Rechnern)
- medienunabhängig - das Web vereint viele verschiedene Medien-(Darstellungs-)formen
- verbindendes Element ist der “Hyperlink”

Multimedia-Anwendungen

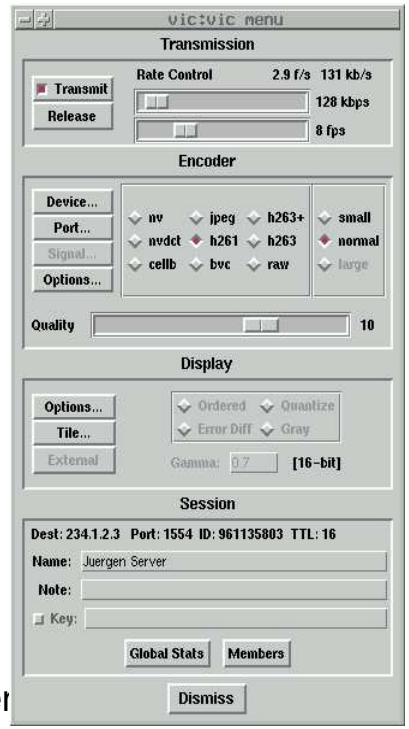
- Mbone
  - vic (Video)
  - vat (Audio)
  - wb (virtuelle Wandtafel)
- diverse
  - RealPlayer (Audio und Video)
  - NetMeeting (Audio, Video, gemeinsames Nutzen beliebiger Anwendungen/Application Sharing)

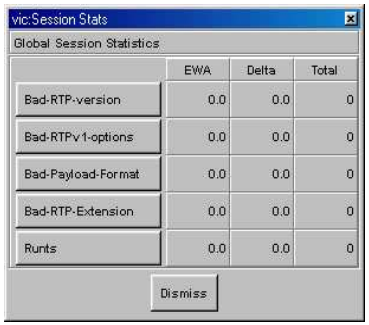
➔ meist doch nicht so „multimedial“, sondern auf ein oder zwei Medienformen (z.B. Audio/Video) festgelegt

- World Wide Web
  - integriert HTML (Text und Bilder) mit verschiedensten anderen Anwendungen über einen einheitlichen Adressierungs-Mechanismus

Mbone vic

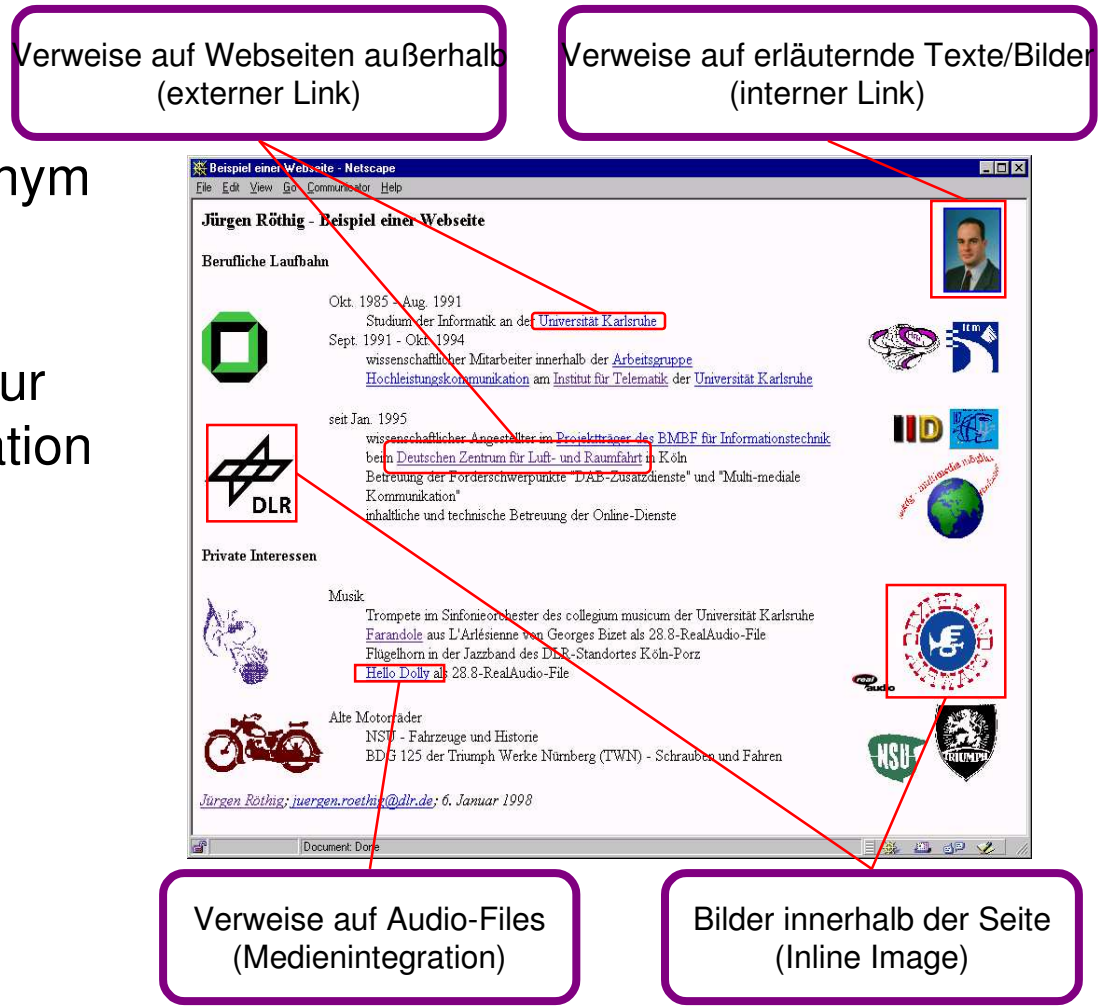
- Mbone:
  - Multicast Backbone
  - ursprünglich akademisches Netzwerk zur Übertragung von IETF-Meetings, wissenschaftl. Konferenzen, …
- vic:
  - video conferencing
  - ursprünglich am LBL entwickelt
  - Open Source
  - multicast (mehrere Sender und Empfänger gleichzeitig)
  - auch unicast (1:1-Verbindung) möglich
  - verwendet RTP/UDP/IP





Anwendung WWW

- WWW: World Wide Web
- eine der häufigsten Anwendungen und Synonym für Internet
- grundlegende Idee: HyperText/HyperMedia zur Verknüpfung von Information
- Integration von verschiedenen Medien (Texte, Bilder/Grafiken, Musik/Sprache, Videos)
- Beispiel



Geschichte des WWW

- Das World Wide Web
  - Tim Berners-Lee 1989/1990: erste Konzeption und erster Prototyp, entstanden am Centre Européen de la Recherche Nucléaire (CERN)
- Vorformen:
  - gopher
  - HyperCard (Apple Macintosh)

➔ Alle diese Ansätze basieren auf den Ideen von HyperText!

  - Theodore Nelson, 1965: “HyperText” als Begriff
  - Vannevar Bush: As We May Think, Atlantic Monthly, Juli 1945
    - Maschine “Memex” zur Vernetzung von verschiedenen (konventionellen) Informationen mit Hilfe von mechanischen Zusätzen (Mikrofilm, komplizierte Mechanik zur Auswahl der Dokumente)

Adressierung im WWW

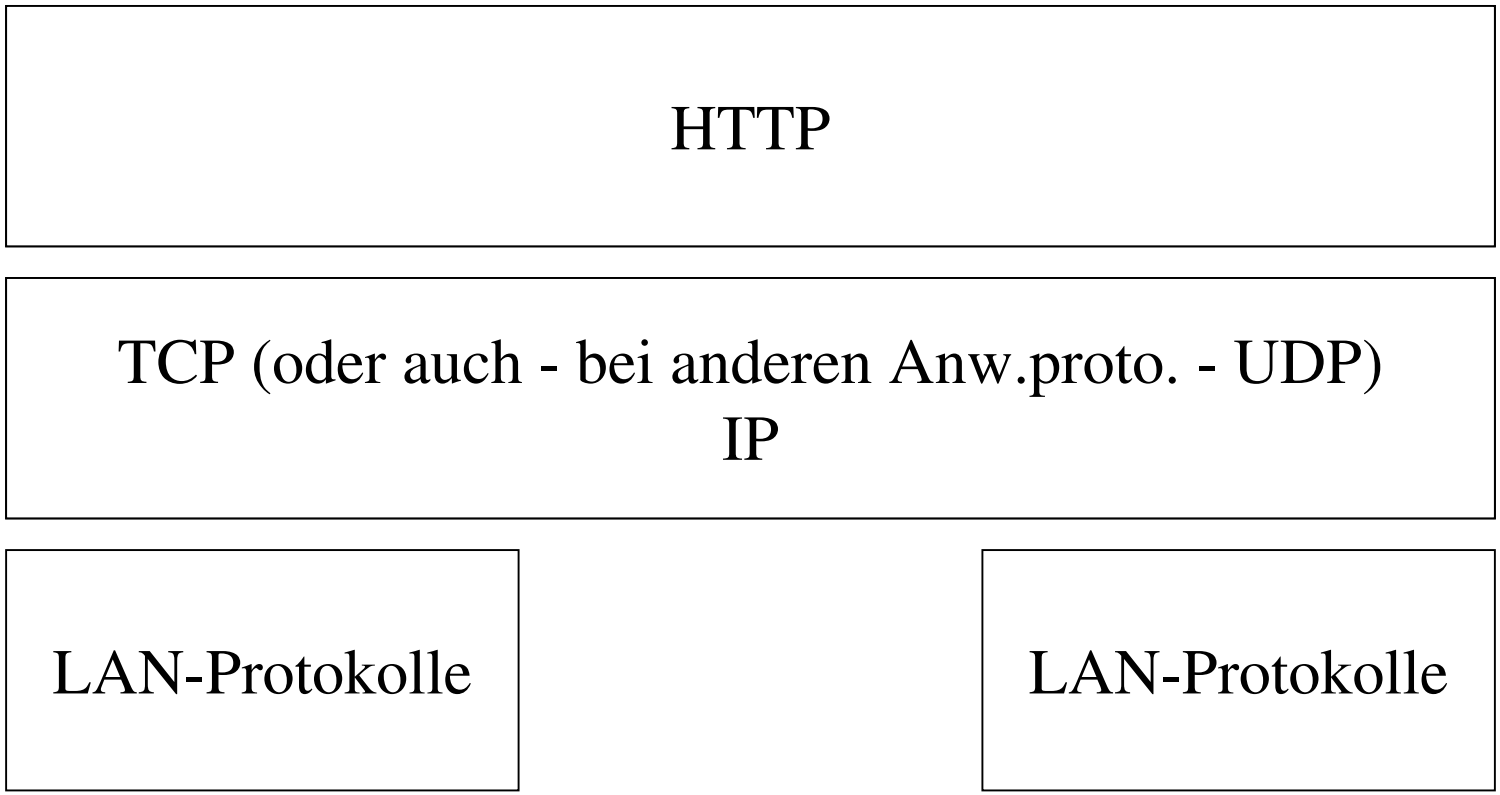
- URL: Uniform Resource Locator
- Aufbau/Bestandteile:
  - protocol://rechner[:port]/verzeichnis/[datei.dateiendung]
- protocol:
  - Protokoll der „Anwendung“, also oberhalb IP, TCP, UDP, ...
  - http (HyperText Transfer Protocol)
  - ftp (File Transfer Protocol)
  - proprietäre Protokolle (z.B. pnm/Progressive Networks Protocol)
  - „pseudo-Protokolle“ (mailto, telnet, news)
- Rechner:
  - IP-Adresse a.b.c.d, wobei a,b,c,d $\in$ [0..255]
  - Name, welcher über DNS (Domain Name Service) auf eine IP-Adresse abgebildet wird, beispielsweise
    - www.ba-karlsruhe.de $\rightarrow$ 193.196.6.88

Ablauf des Zugriffs auf den Webserver

- Im Webbrowser wird eine URL eingegeben.
  - http://name.des.webservers:port/pfad/name.html
- oder
  - http://name.des.webservers:port/pfad/
- oder
  - http://name.des.webservers:port/pfad
- Vom Webserver wird das entsprechende Dokument (meist HTML) geliefert.

- Doch was passiert alles dazwischen?

Kommunikations-Protokolle



Aufbau einer Verbindung zum Webserver

- Kommunikation zwischen Browser und Server erfolgt über HTTP (HyperText Transfer Protocol).
- HTTP basiert auf einer TCP-Verbindung.

- Auflösung des Namens des Webservers in seine IP-Adresse mittels DNS (Domain Name Service).
- Adressierung der Anwendung mittels des Port.
- Aufbau der TCP-Verbindung.
- Versendung der Daten per IP.

- Welche Informationen muß dann eine HTTP-Anforderung enthalten?

HTTP 0.9 anno 1991

- Zwei Typen von HTTP-Datenpaketen:
  - Request
  - Response
- Request:
  - enthält Teil des URLs ohne “http”, Name des Servers, Port
  - GET pfad/name.html
- Response:
  - enthält HTML-Dokument
  - kein Status, keine Header
  - Fehlermeldungen werden als HTML-Dokument zurückgeliefert
  - keine Unterscheidungsmöglichkeit zwischen Erfolg und Fehler
  - keine Möglichkeit zur Übertragung anderer Medienformen

HTTP 1.0 (ab 1992)

- abwärtskompatibel zum ursprünglichen HTTP
- Request in den meisten Fällen unverändert (GET)
  - aber zusätzliche Typen (POST)
- Response erfährt die wichtigste Erweiterung
- Response:
  - beginnt mit einer Status Line:
    <http version> <status code> <reason string> <CrLf>
  - enthält verschiedene Header-Informationen
    Content-Length, Content-Type, Date, Last-Modified, Expires, Content-Language

- schön, aber reicht das?

Große Webserver I

- Anzahl der Web-Adressen (und damit der “logischen Webserver) nimmt seit zehn Jahren rasend schnell zu.
- Heutige Webserver-Hardware (“physikalische Webserver”) bedient oft mehrere Webauftritte
  - Beispiel: große Webhoster wie Strato, 1&1/Puretec

- Aber: Wie können mehrere Webauftritte von derselben Maschine bedient werden?

Große Webserver II

- Einfache Lösung:
  - jeder Webauftritt mit einem eigenen Bereich auf einem existierenden Webserver unter dessen Adresse
    - Homepage auf einem Unix-Rechner: http://www.rechner.de/~user/
    - viele Anbieter von “Free Webspace”: http://www.rechner.de/members/user/
  - nicht akzeptabel für einen “professionellen Webauftritt”
- häufig genutzte Lösung:
  - jeder Webauftritt ist unter einer eigenen Domain erreichbar
    - http://www.firma-eins.de/
    - http://www.firma-zwei.com/

Große Webserver III

- Problem 1: Die Software
  - Muß für jeden Webauftritt eine eigene Instanz der Webserver-Software gestartet werden und laufen?
  - Kostet zusätzliche Ressourcen (Speicher, CPU-Zeit)
  - Gängige Webserver-Software (z.B. Apache) kann mehrere Webauftritte (“virtuelle Server”, “virtual host”) parallel bedienen.

  - Definition eines virtuellen Servers:

<VirtualHost>

…

</VirtualHost>

Große Webserver IV

- Problem 2: Die Adressierung
  - Adressierung findet im Internet (beim Internet-Protokoll) über die IP-Adresse statt.
  - Kann ein Netzwerk-Interface mehrere IP-Adressen gleichzeitig haben?
  - Ja. Beispiel Ethernet-Interface unter Linux: eth0, eth0:1, eth0:2
  - Definition eines virtuellen Servers, der auf der IP-Adresse 1.2.3.4 hört:

<VirtualHost 1.2.3.4>

…

</VirtualHost>

Große Webserver V

- Problem 2: Die Adressierung
  - Aber: Benötigt jeder Webauftritt auch eine eigene IP-Adresse?
  - Nein !?!

Hintergrund: Die IP-Adreßknappheit I

- IP-Adressen bestehen aus 32 bit
- werden normalerweise als 4-Tupel à 8 bit gruppiert
- ergibt theoretisch gut 4 Mrd. IP-Adressen
- praktisch sind nicht alle Adressen verfügbar:
  - reservierte IP-Adreßbereiche für private Netzwerke
    - 10.0.0.0 - 10.255.255.255 (10/8)
    - 172.16.0.0 - 172.31.255.255 (172.16/12)
    - 192.168.0.0 - 192.168.255.255 (192.168/16)
  - reservierter IP-Adreßbereich für Multicast
    - 224.0.0.0 - 239.255.255.255 (224/4)
  - Netzwerkadressen und Broadcast-Adressen
    - Bsp: 195.168.48.0 und 195.168.48.255 bei 195.168.48/24

Hintergrund: Die IP-Adreßknappheit II

- IP-Adressen eigentlich (noch) nicht knapp.
  - Aber: Mehrzahl (etwa drei Viertel) aller IP-Adressen ist an Organisationen in den USA vergeben (historische Gründe).
- Abhilfe:
  - dynamische Adreßzuteilung (bei der Einwahl ins Internet)
  - NAT (Network Address Translation, ein privates Netzwerk hat nur eine öffentliche IP-Adresse)
- Aber:
  - Wenn tatsächlich eine Vielzahl von Geräten (Heim-PC, Telefon, Kaffeemaschine, …) eine eigene feste IP-Adresse erhalten soll, reichen die derzeitigen IP-Adressen bei weitem nicht aus.
- Echte Abhilfe:
  - IPv6, Adreßlänge 128 bit
  - ausreichend für mehrere Mio Adressen pro m^2

Große Webserver VI

- Adressierung
  - Unterscheidung nach Portnummern.

  - Definition eines virtuellen Servers, der auf Port 8080 hört:

```
Listen 80
Listen 8080
<VirtualHost 1.2.3.4:8080>
…
</VirtualHost>
```

Große Webserver VII

- Adressierung
  - Unterscheidung nach Domainnamen.
  - Festlegung daß auf der IP-Adresse 1.2.3.4 mehrere virtuelle Server hören:

    NameVirtualHost 1.2.3.4
  - Adressiert wird der Rechner (für die TCP-Verbindung) nur per IP-Adresse.
  - Woher weiß dann der Webserver, welcher Webauftritt gemeint ist?

HTTP 1.1 (seit 1997)

- Request enthält wichtige Erweiterung:
  - Host Header
  - enthält den Namen des addressierten Webservers
  - Host Header ist Pflicht bei HTTP 1.1!
- Damit werden virtuelle Server anhand des Domain Name problemlos möglich!

- Außerdem:
  - Unterstützung von persistenten Verbindungen
  - Nutzen einer TCP-Verbindung für mehrere aufeinanderfolgende http-Anforderungen

Übertragung von Dateien im WWW

- HTTP
  - HyperText Transfer Protocol
  - Protokoll zur Kommunikation zwischen Web-Browser und Web-Server
  - Browser sendet Anfrage (Request) und erhält Antwort (Response)
  - zusätzlich ergibt jede Anfrage einen Status Code
    - 200 ok
    - 404 not found
  - HTTP basiert auf TCP, d.h. vor der Kommunikation zwischen Browser und Server muß eine Verbindung aufgebaut werden
  - HTTP/1.1 bietet Unterstützung für “multi-homed Web Server” (mehrere Web-Server auf einer IP-Adresse)

HTTP Return Codes

- Successful Client Requests
  - 200 OK
  - 201 Created
  - 202 Accepted
  - 203 Non-Authorative Information
  - 204 No Content
  - 205 Reset Content
  - 206 Partial Content
- Client Request Redirected
  - 300 Multiple Choices
  - 301 Moved Permanently
  - 302 Moved Temporarily
  - 303 See Other
  - 304 Not Modified
  - 305 Use Proxy
- Server Errors
  - 500 Internal Server Error
  - 501 Not Implemented
  - 502 Bad Gateway
  - 503 Service Unavailable
  - 504 Gateway Timeout
  - 505 HTTP Version Not Supported
- Client Request Errors
  - 400 Bad Request
  - 401 Authorization Required
  - 402 Payment Required (not used yet)
  - 403 Forbidden
  - 404 Not Found
  - 405 Method Not Allowed
  - 406 Not Acceptable (encoding)
  - 407 Proxy Authentication Required
  - 408 Request Timed Out
  - 409 Conflicting Request
  - 410 Gone
  - 411 Content Length Required
  - 412 Precondition Failed
  - 413 Request Entity Too Long
  - 414 Request URI Too Long
  - 415 Unsupported Media Type

Dokumentbeschreibungssprachen

- proprietäre Dokumentformate für WYSI(N)WYG-Anwendungen
  - “What You See Is (Never) What You Get”
  - Beispiel: “.doc”-Format von Microsoft Word
  - Attributierung von Textpassagen mit ihrem jeweiligen Aussehen (Schriftart, -größe, Fett-, Kursivschrift)
  - binär kodiertes, undokumentiertes, nicht frei verfügbares Format
- Strukturorientierte Sprachen mit (S)GML
  - “(Standard) Generalized Markup Language”
  - Beispiel: XHTML zur Darstellung von Dokumenten im Web
  - Attributierung von Textpassagen mit ihrer jeweiligen Funktion (Überschrift, Absatz, Fußnote)
  - rein textorientiertes (im Klartext les- und editierbares) standardisiertes Format

HyperText

Bisher:

- verschiedene Medienformen (MIME-Types)

➔ Medienunabhängigkeit

- Adressierung (per URL)

➔ Lokationsunabhängigkeit

Aber:

- Wie wird das ganze nun in Web-Dokumenten verwendet?

➔ HyperText/HyperMedia!

HyperText-Prinzip I

- In wissenschaftlicher Literatur sind Verweise ein gängiges Prinzip:
  - andere Stellen im Text: “Kapitel 3.7 zur pädagogischen Leistungsfähigkeit von BA-Dozenten”
  - Abbildungen: “in Bild 4.37 wird ein typischer BA-Hörsaal gezeigt”
  - Tabellen: “Tabelle 15.33 auf Seite 999 zeigt die Entwicklung der Studierendenzahlen an der BA Karlsruhe”
  - Fußnoten: “das Studium an Hochschulen[1] ist von wissenschaftlicher Arbeitsweise geprägt”
  - Literaturverweise/Literaturverzeichnis: “wie in [BA-Studienführer 2003] erläutert”
- Diesen Verweisen muß immer “manuell” nachgegangen werden.

HyperText-Prinzip II

- Prinzip von HyperText:
  - einzelne Textpassagen werden mit Verweisen verknüpft
  - Automatismus hilft beim Auflösen des Verweises
- Der unterliegende Text bildet eine Beschreibung des Verweisinhaltes.
- Das (technische) Ziel des Verweises ist nicht unbedingt offensichtlich!

HTML

- HTML
  - HyperTextMarkup Language
  - für Textdokumente im WWW verwendete Auszeichnungssprache
  - rein textorientierte Dokumentenbeschreibungsspache
  - basiert auf GML-Grundprinzip (Generalized Markup Language) im Gegensatz zu proprietären Dokumentformaten à la Microsoft Word
  - standardisiert vom W3C (WWW Consortium)
- gut geeignet zur Darstellung statischer (feststehender) Inhalte (beispielsweise wissenschaftliche Aufsätze)

Historie von HTML

- Um 1970: erste GML-basierende Sprachen
- 1992: ursprüngliches HTML
  - Überschriften, Paragraphen, Listen, Verweise
- 1995: HTML 2.0
  - Bilder, Textgestaltung, Forms
- 1996: HTML 3.2
  - Tabellen, Applets, Textfluß um Grafiken
- 1997: HTML 4.0
  - StyleSheets, Frames
- 2000: XHTML 1.0
  - Neuformulierung von HTML als XML-konforme Sprache

Aufbau von (X)HTML-Dokumenten I

- HTML-Dokumente bestehen aus Klartext
- Text wird normalerweise direkt im Browser dargestellt.
- Trennzeichen (Leerzeichen, Zeilenumbruch) dienen zur Abgrenzung von Wörtern.
- Zusätzlicher “Whitespace” wird ignoriert/mehrere aufeinanderfolgende Trennzeichen werden wie ein Trennzeichen behandelt.
- Speziell markierte Kommentare werden ignoriert:
  <!-- Das ist ein Kommentar -->

Aufbau von (X)HTML-Dokumenten II

- Nicht im 7-bit-ASCII-Code darstellbare Zeichen (nationale Sonderzeichen) müssen/können umschrieben werden:
  - ä, ö, ü: ä, ö, ü
  - Ä, Ö, Ü: Ä, Ö, Ü
  - ß: ß
  - é, è, ê: é, è, ê
- bestimmte Zeichen müssen umschrieben werden, da sie spezielle Funktionen bei HTML erfüllen:
  - <: <
  - >: >
  - &: &
  - ”: "

Tags I

- Die Funktion von Textbestandteilen (das “Markup”) wird durch sogenannte “Tags” beschrieben.
- Ein Tag besteht aus einer Folge von (kleinen) Buchstaben.
- Ein Tag steht in spitzen Klammern (“<…>”).
- Ursprünglich (bei HTML) waren Tags unabhängig von der Groß- und Kleinschreibung.
- Tags bei XML sind dagegen “case-sensitive”.

➔ Alle Tags bei XHTML bestehen aus Kleinbuchstaben.

Tags II

- Ein “nichtleerer” Tag bezieht sich auf eine Textpassage.
- Ein “nichtleerer” Tag begrenzt den betreffenden Text durch einen Start- und einen End-Tag.
- Der entsprechende End-Tag besteht aus derselben Buchstabenfolge wie der Start-Tag, aber mit einem vorangestellten Schrägstrich (“</…>”).
- Bei HTML war bei manchen nichtleeren Tags kein End-Tag notwendig
  - Bsp: <p> - der nächste Paragraph beendete automatisch den vorhergehenden Paragraph
- Bei XHTML sind die End-Tags obligatorisch!

Tags III

- Ein “leerer” Tag bezieht sich nicht auf einen bestimmten Text.
- Eigentlich (und ursprünglich) ist hier daher kein separater End-Tag notwendig.
- Bei XML muß zu jedem Start-Tag ein zugehöriger End-Tag existieren.

➔ Zwei Möglichkeiten zur Realisierung bei XHTML:

  – Ein separater End-Tag folgt direkt dem Start-Tag.
  – Der Start-Tag enthält einen abschließenden Schrägstrich und wird damit gleichzeitig zum Ende-Tag (“<… />”).

Tags IV

- Tags können (mehrere) Attribute haben
- Attribute haben einen Namen und einen Wert
- das Attribut wird durch ein Leerzeichen vom Tag-Namen getrennt, der Wert folgt nach einem Gleichheits-Zeichen (“=”)
- <tag attribut1=“wert1” attribut2=“wert2”>

Schachtelung

- Die meisten (nichtleeren) Tags können Behälter (Container) nicht nur für Text, sondern auch für weitere Tags sein.
- Die korrekte Schachtelung der Tags ist wichtig!

Grundaufbau eines Dokuments

- Ein HTML-Dokument besteht aus Kopf und Rumpf.

```
<html>
<head>
<title>Das erste HTML-Dokument</title>
</head>
<body>
Hier steht das erste HTML-Dokument.
</body>
```

Erkennung des Dokumenttyps

- Woran erkennt man (der Browser), um welche HTML-Variante es sich bei einem Dokument handelt?

➔ Am Dokument selbst!

Deklaration des Dokumenttyps

- Referenz auf die DTD (Document Type Definition) vor dem HTML-Quelltext:
  - <!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN”
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
  - <!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN”
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
  - <!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN”
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
- <html>-Tag enthält Referenz auf den Namespace:
  - <html xmlns="http://www.w3.org/1999/xhtml”
    xml:lang="en" lang="en">

Dokument-Rumpf

- Enthält das eigentliche Dokument
  - Überschriften
  - Absätze
  - sonstige Gliederung
  - Verweise
  - Bilder
  - Listen
  - Tabellen

Überschriften

- Überschriften dienen zur Strukturierung des Textes.
- Sie sind verfügbar auf verschiedenen Ebenen: 1-6.
- Tags:
  - <h1>Überschrift der obersten Kategorie</h1>
  - <h6>Überschrift der niedrigsten Kategorie</h6>

Absätze

- Absätze dienen zur Aufnahme des Textes und zu dessen blockweiser Gliederung.
- Fließtext kann aber auch “einfach so” ohne Nutzung der Absatz-Tags eingegeben werden (falls keine weitere Untergliederung gewünscht ist).
- Tags:
  - Dies ist ein Paragraph. Er enthält viele Sätze. Innerhalb eines Paragraphen steht also unmittelbar zusammengehöriger Text.

Weitere Gliederungsmöglichkeiten

- Kennzeichnung eines Bereichs (meist am Ende einer HTML-Seite) von im Browser sichtbarer Meta-Informationen über das HTML-Dokument:
  - <address>Urheber-Information</address>
- Tags zur Auszeichnung von Text als (irgendwie) zusammengehörig (beispielsweise um sie mittels CSS in der gleichen Art zu formatieren):
  - Für größere Bereiche (über ein oder mehrere Absätze): Text
  - Für einzelne Sätze oder Wörter (innerhalb eines Absatzes): Text

Verweise

- Verweise ermöglichen Verknüpfungen zu anderen Seiten und Dokumenten im Web.
- Tags:
  - <a href="http://www.ba-karlsruhe.de/">
- Attribute
  - href: Verweis auf das Objekt
- Es gibt verschiedene Formen von URLs:
  - absolute URL: komplette Angabe der URL
  - relative URL: einige Angaben (Protocol/Method, Network Location, Path) können fehlen

Warum relative URLs?

- Dokumente sollen über verschiedene Zugriffsmethoden (http, ftp) erreichbar sein.
- Dokumente sollen an unterschiedlichsten Lokationen gelagert werden können.
- Dokumente sollen einfach kopiert ("gespiegelt") werden können.

➔ Unabhängigkeit von zusammengehörigen Dokumentbäumen von der jeweiligen physikalischen Lokation und Anbindung ist gewünscht!

Bilder

- Bilder fügen graphische Darstellungen direkt in den Text ein.
- Bildformate:
  - gif
  - jpeg,
  - png
- Tags:
  - <img src="bilddatei.jpg" alt="Hier sollte eigentlich ein Bild zu sehen sein">
- Attribute:
  - src: Verweis auf die Lokation der Bilddatei
  - alt: alternativer (beschreibender) Text, der statt des Bildes angezeigt werden kann

Listen

- Listen dienen zur Aufzählung.
- Es gibt drei verschiedene Typen:
  - einfache Listen
  - geordnete (numerierte) Listen
  - Definitionslisten
- Tags:
  - erstes Element einer einfachen Liste
zweites Element einer einfachen Liste
  - erstes Element einer geordneten Liste
zweites Element einer geordneten Liste
  - <dl><dt>Tag</dt><dd>Bezeichnung eines speziellen Schlüsselwortes zur Markierung von Textpassagen</dd></dl>

Tabellen

- Tabellen sind unterteilt in Zeilen und Spalten.
- Tabellen werden als Folge von Zeilen eingegeben, wobei jede Zeile die Inhalte der jeweilig zugehorigen Spaltenwerte enthält.
- Tags:
  - <table>
    <tr><td>erste Zeile erste Spalte</td>zweite Spalte</td>
    <td>zweite Zeile erste Spalte</td>zweite Spalte</td>
    </tr>
- Mehrere Zellen aus nebeneinanterliegenden Spalten bzw. untereinander liegenden Zeilen können zusammengefaßt werden
  - <td colspan="3">Zelle über drei Spalten</td>
  - <td rowspan="4">Zelle über vier Zeilen</td>

Einige “Kleinigkeiten”

- Zeilenumbruch dient zum Erzwingen des Beginns einer neuen Zeile.
- Tag:
  -
- Horizontal Rule zeichnet eine horizontale Linie quer über das Browserfenster.
- Tag:
  - <hr />
- Der Adressbereich ist ein Teil des Dokumentrumpfes, liegt meist am Ende und ist dafür da, Informationen zum Urheber und ähnliches abzulegen.
- Tag:
  - <address>BA Karlsruhe</address>

Dokument-Kopf

- Der Kopf des Dokuments enthält beschreibende Elemente, die nicht direkt darzustellende Texte darstellen, sondern das Dokument beschreiben:
  - <title>Titel des Dokuments</title> (Pflichtbestandteil)
  - <base href="URL"> (Lokation des Dokuments)
  - <link rel="Beziehung" href="URL" type="Typ"> (Verweis auf weitere Informationen)
  - <meta http-equiv="Header" CONTENT="irgendwas"> (HTTP-Header-Informationen)
  - <meta name="Name" CONTENT="Wert"> (Meta-Informationen
- Beispiel: Einbindung eines Stylesheets:
  <link rel="stylesheet" href="my.css" type="text/css">

Metadaten

- Ein möglicher Ausweg aus dem Chaos und eine Hilfe zur Beschreibung von und bei der Suche nach Objekten in einem großen Datenbestand:

➔ Indexierung

- Verknüpfung von Objekten/Dateien mit diese beschreibenden Metadaten

- Aber:
- hoher Aufwand zur Erstellung der Metadaten
- Wie sehen die Metadaten konkret aus?
- Wohin mit den Metadaten?

Metadaten bei HTML-Dateien

- META-Tag im Header der HTML-Datei
- enthält den Namen einer Eigenschaft und dessen Wert

➔ <META name="property" content="value">

- einige Eigenschaften sind "üblich"

➔ author, date, keywords, ...

- Attribut "lang" ermöglicht Angabe der Sprache
- Attribut "scheme" ermöglicht Angabe des Formats (Bsp. Datum)
  - <META name="author" content="J.W. von Goethe">
  - <META name="keywords" lang="de" content="Literatur, Klassik, Tragödie, Faust">
  - <META name="date" scheme="Month-Day-Year" content="04-01-03">
- "Sonderfall" http-equiv

➔ Angabe von http-Headern (z.B. Verfallsdatum, Weiterleitung auf andere URL ohne JavaScript)

  - <META http-equiv="Expires" content="Tue, 18 Mar 2003 14:00:00 GMT">

➔ Profile ermöglichen Verweis auf die Definition der Eigenschaften

  - <head profile="http://dublincore.org/documents/dces/">

Metadaten-Standards

- Problem: Beschreibung von Objekten mit wahllose beliebigen Texten verschiebt das Problem “nur eine Ebene höher”

➔ ein genau festgelegter Standard, was mit welchen Stichworten beschrieben werden kann und muß, ist notwendig

- Beispiel: DublinCore-Metadatenstandard

Dublin Core I

- Ziel:

➔ “The Dublin Core Metadata Initiative is an open forum engaged in the development of interoperable online metadata standards that support a broad range of purposes and business models.”

➔ http://dublincore.org/

- Definition der möglichen Metadaten-Eigenschaften:

➔ Dublin Core Metadata Element Set, Version 1.1: Reference Description

➔ http://dublincore.org/documents/dces/

- http://www.ietf.org/rfc/rfc2413.txt

Dublin Core II

- Einteilung in drei Kategorien:
  - “(1) elements related mainly to the Content of the resource”
  - “(2) elements related mainly to the resource when viewed as Intellectual Property”
  - “(3) elements related mainly to the Instantiation of the resource”

- Content
  - Title
  - Subject
  - Description
  - Type
  - Source
  - Relation
  - Coverage
- Intellectual Property
  - Creator
  - Publisher
  - Contributor
  - Rights
- Instantiation
  - Date
  - Format
  - Identifier
  - Language

Cascading Style Sheets

- Bislang:
  - Nur “funktionale Struktur” mittels HTML-Tags
  - Aussehen von beispielsweise Überschriften wird nur durch den Browser bestimmt.
- Mittels CSS:
  - Bestimmung des grundsätzlichen Aussehens des Textes und von bestimmten Tags
  - Zusätzlich Definition von speziellen Klassen und IDs, die eigenes (anderes) Aussehen für die jeweiligen Texte vorgeben.
- CSS:
  - Version 1: 1996
  - Version 2: 1998 (aktuell)
  - Version 3: in Vorbereitung/Entwicklung

CSS: Angabe des Styles

- 1. Möglichkeit:
  - Direkt beim jeweiligen Tag.
  - <h1 style=“color: blue”>
  - kein großer Unterschied (kein Vorteil) zu vorher (ohne CSS)
- 2. Möglichkeit:
  - Im Head des HTML-Dokuments
  - <style type="text/css"> H1 { color: blue; } </style>
- 3. Möglichkeit:
  - Verweis auf ein externes Style-Dokument
  - <link href=”mystyle.css" rel="stylesheet" type="text/css" />
- 4. Möglichkeit:
  - Einbinden eines anderen innerhalb eines Stylesheets
  - @import "mystyle.css";

CSS: Aufbau eines Stylesheets

- Ein Stylesheet besteht aus Regeln.
- Eine Regel (Rule) besteht aus Selector und Declaration:
  - Selector { Declaration }
- Der Selector gibt an, auf welche Tags sich die Regel bezieht.
- Die Declaration besteht aus Eigenschaft (Property) und Wert (Value):
  - Property: Value
- Mehrere Selectoren, auf die sich dieselbe Declaration beziehen soll, werden durch Kommata getrennt.
- Mehrere Declarations für denselben Selector können durch Semikolon getrennt angegeben werden.
- Mehrere Values, die sich auf dieselbe Property beziehen, werden durch Kommata getrennt.

CSS: Beispiele für Properties I

- Schriften:
  - font-style: normal | italic | oblique
  - font-variant: normal | small-caps
  - font-weight: normal | bold | bolder | lighter
  - font-size: small | medium | large | larger | smaller | 16pt | 120% (absolute, relative, length, percentage)
  - font-family: “New Century Schoolbook” | Arial | serif | sans-serif | cursive | fantasy | monospace
- Listen:
  - list-style-image: URL
  - list-style-type: disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-alpha | lower-latin | upper-alpha | upper-latin

CSS: Beispiele für Properties II

- Farben:
  - Vordergrund/Schriftfarbe (für fast alle Tags)
  - color: red | rgb(255,0,0)
- Hintergrund:
  - background-color: <color> | transparent
  - background-image: URL
  - background-repeat: repeat | repeat-x | repeat-y | no-repeat
  - background-attachment: scroll | fixed
  - background-position: horizontal | vertical (length oder percentage) bzw. [left | center | right] || [top | center | bottom]

CSS: Gültigkeit von Regeln I

- Properties werden vererbt:
  - Beispiel: von <body> nach <p>
- Selectors können in einen Kontext gesetzt werden und gelten nur in diesem:
  - Beispiel: p em {}
  - ➔ nur falls der -Tag innerhalb eines <p> auftritt, tritt die Regel (bzw. dessen Declaration) in Kraft.

CSS: Vererbung

- Document Tree
  - Baum von Elementen im HTML-Quelltext
  - jedes Element hat genau ein Elternelement (außer der Wurzel)
  - Eigenschaften von Elternelementen gehen auf die Kinder über

CSS: Klassen

- Regeln können für spezielle Klassen definiert werden.
- Regeln gelten dann nur für Elemente, welche dieser Klasse angehören.
  - Bsp:
    - p.myclass{} im Stylesheet
    - <p class="myclass> … </p> im HTML-Text
- Es existieren einige vordefinierte Pseudo-Klassen.
  - Bsp Links:
    - A:link { color: red } /* unvisited links */
    - A:visited { color: blue } /* visited links */
    - A:hover { color: yellow } /* user hovers */
    - A:active { color: lime } /* active links */

CSS: Gültigkeit von Regeln II

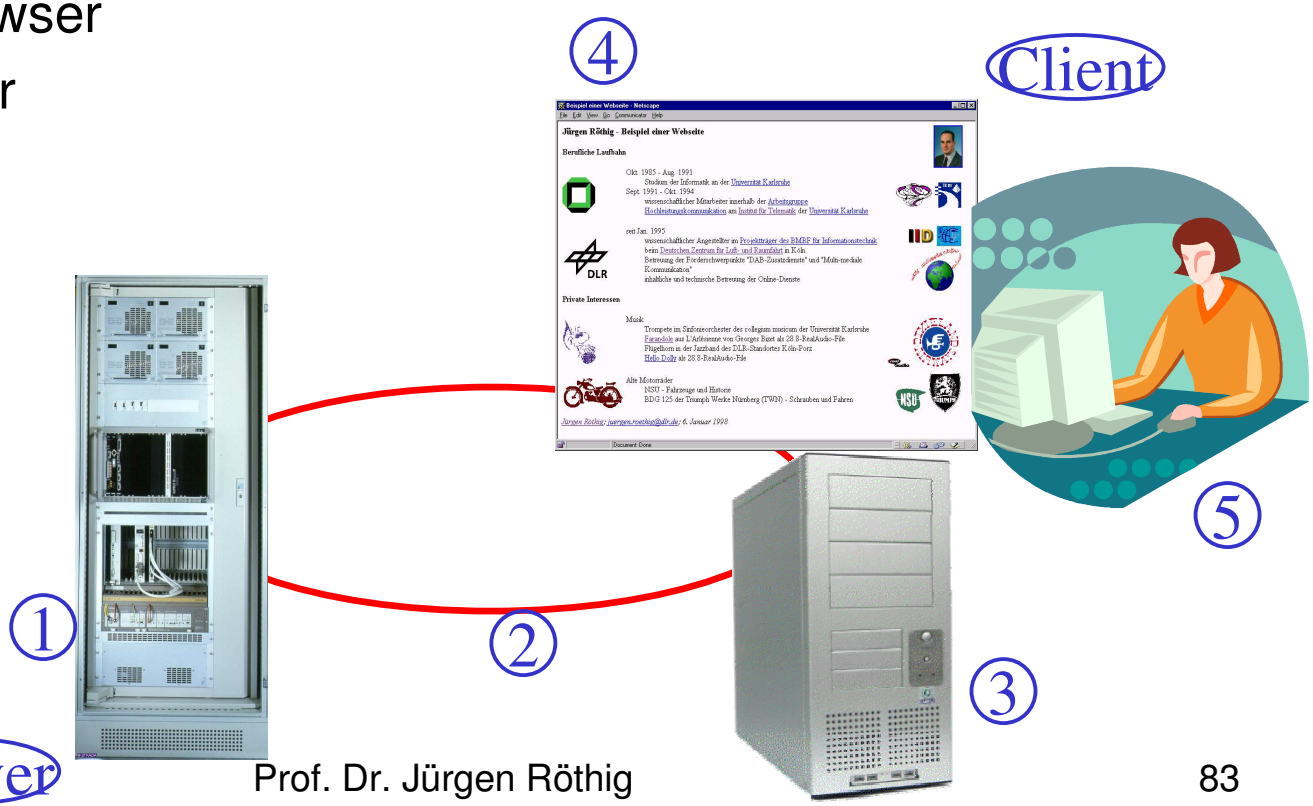
- Kaskade:
  - Stylesheets haben verschiedene Ursprungsmöglichkeiten:
    - Author
    - User
    - User Agent
  - Gewichtung der Regeln nimmt von Author über User nach User Agent ab.
  - Höher spezifizierte Regeln haben eine höhere Gewichtung als allgemeine Regeln (id>class>element name)
    - #myid{}
    - p.myclass{}
    - p {}
  - Mit “!important” gekennzeichnete Regeln haben eine höhere Gewichtung als andere.

Motivation für dynamische Inhalte

- Trennung von Gehalt und Gestalt
  - der Inhalt wird dynamisch (on the fly) mit einem Rahmen versehen und als HTML-Seite ausgeliefert
  - Rahmen enthält Menue, Navigationselemente
  - einfache Anpassung des Designs einer gesamten Website durch Änderung des Rahmens
- Darreichung von Inhalten, die nicht nur für das Web vorgesehen sind
  - Vorhalten derselben Texte für Web, Print, … kostet unnötig Speicherplatz
- Anbindung an Datenbanken
  - Abfrage der umfangreichen Datenbankinhalte über einfach zu bedienendes Web-Frontend

Dynamische Inhalte im Web

- Fluß der Informationen:
  - ① Webserver
  - ② Netzwerk/Internet
  - ③ PC
  - ④ Webbrowser
  - ⑤ Benutzer



Dynamische Inhalte im Web (Client)

- JavaScript:
  - direkte Reaktion auf Benutzeraktionen
  - Anpassung der Inhalte von Listen abhängig von einer Menüauswahl
  - einfache Spiele
  - rein clientseitige Dynamisierung - Server liefert statische Inhalte
- Java:
  - Ausführung von Java-Applets in einer “Sandbox” auf dem Client
  - kompliziertere Abläufe als mit JavaScript möglich
  - über spezielle Kommunikationskanäle auch Datenaustausch mit dem Server möglich
  - Java-Applet selbst ist “statisch” (alle Variabilitäten müssen von vornherein einprogrammiert sein) und läuft als Programm auf dem Client

Dynamische Inhalte im Web (Server)

- SSI (Server Side Includes):
  - Einbindung von Kommandos, die beim Abruf der HTML-Seite vom WebServer ausgeführt werden
- CGI (Common Gateway Interface):
  - Aufruf wie eine “normale” HTML-Seite
  - Schnittstelle zur Rückmeldung von Benutzereingaben vom Client an den Webserver
  - führt beliebige Aktionen/Programme auf dem Webserver aus
  - generiert HTML-Code, der zum Client übertragen und vom Browser angezeigt wird
- PHP (PHP: Hypertext Preprocessor):
  - verbreitete Skriptsprache für Webserver
- ASP (Active Server Pages):
  - Microsofts Lösung für dynamisch generierte Webseiten

Dynamische Inhalte im Web (Daten)

- Datenbanken
  - enthalten nicht-aufbereitete Texte, Grafiken, …
  - Abfrage über Web-Frontend als HTML-Formular
  - Übermittlung der Benutzereingaben über CGI
  - Formulierung als Datenbankabfrage
  - Aufbereitung der gelieferten Daten als HTML-Seite
  - Auslieferung der Ergebnis-HTML-Seite an den Web-Browser
- Häufig genutzte Kombination:
  - “LAMP”
    - Linux (Betriebssystem)
    - Apache (WebServer)
    - MySQL (relationale Datenbank)
    - PHP (Skriptsprache)

Dynamische Inhalte im Web (Content)

- CMS:
  - Content Management System
  - Verwaltung von Benutzerrechten (wer darf welche Inhalte wo einstellen?)
  - Verwaltung der eingestellten Inhalte in einer Datenbank
  - dynamische Generierung und Auslieferung der HTML-Seiten entsprechend vorgegebenem Rahmen und Design
  - Verwaltung der Beziehungen der Dokumente untereinander (“interne Links”)
  - zentrale Verwaltung externer Links

Dynamische Web-Seiten-Generierung I

- Dynamische Generierung von Web-Seiten bedeutet Ausführung von Programmen auf dem Web-Server

➔ Performance-Problem

➔ Sicherheitsproblem

- Ausführung von Programmen beschränkt auf bestimmte einzelne Verzeichnisse (“cgi-bin”)
- Programme werden als user “nobody” (ohne spezielle Rechte) ausgeführt

Ausführung eines CGI-Scripts

- Ausführbares Programm:
  - formuliert in beliebiger Programmiersprache
  - ausführbar, d.h. compiliert und gelinkt
- direkt interpretierbares Script:
  - Shell-Script, Perl-Script
  - Script-Interpreter muß sich auf dem Web-Server befinden
- lokalisiert in speziellen freigegebenen Verzeichnissen (“cgi-bin”) abhängig von der Web-Server-Konfiguration
- eventuell versehen mit spezieller File-Extension (“cgi”) abhängig von der Web-Server-Konfiguration

Dynamische Web-Seiten-Generierung II

- Dynamische Generierung von Web-Seiten bedeutet Eingabe von Parametern durch den Benutzer

➔ normalerweise nur Abfrage von Seiten vom Server durch den Client

➔ zusätzlicher Datenfluß vom Client zum Server notwendig

- Common Gateway Interface definiert Möglichkeiten der Übermittlung von Daten an den Server
- Eingabeparameter werden über Umgebungsvariablen an das Programm übermittelt

Umgebungsvariablen für CGI-Scripts

- Umgebungsvariablen, welche das CGI-Script auswerten kann:
  - SERVER_NAME
  - SERVER_SOFTWARE
  - SERVER_PROTOCOL
  - REQUEST_METHOD
    - Art des Aufrufs der Seite/des Scripts: GET, POST
  - QUERY_STRING
    - an die URL nach einem “?” angefügte Information (GET)
  - CONTENT_TYPE
    - MIME type der angefügten Daten, beispielsweise "text/html".
  - CONTENT_LENGTH
    - Länge der angefügten Daten (POST)
  - GATEWAY_INTERFACE
  - HTTP_USER_AGENT
  - HTTP_REFERER
  - DOCUMENT_ROOT

Dynamische Web-Seiten-Generierung III

- Dynamische Generierung von Web-Seiten bedeutet: Programm muß den HTML-Text der Web-Seite bzw. Inhalt des Web-Dokuments erzeugen

➔ Ausgabe des Programms muß gültigen HTML-Text generieren

➔ einige notwendige (HTTP-)Header müssen ebenfalls zusätzlich ausgegeben werden

- erhöhter Aufwand zur Generierung einer HTML-Seite
- kein direkter Check der HTML-Syntax möglich

Header eines CGI-Scripts

- Mögliche Header:
  - Content-type, z.B. “Content-type: text/html”

  ➔ Angabe des MIME-Types

  - Location, z.B. “Location: /pfad/document.html”

  ➔ direkte Weiterleitung auf eine andere Seite

  - Status, z.B. “Status: 404 Not found”

  ➔ Rückgabe des Status

Beispiel für Hello World in C

```
#include <stdio.h>

main()
{
    printf ("Content-type: text/html; charset=utf-8\n\n");
    printf ("<html>\n");
    printf ("<head>\n");
    printf ("<title>Hello World - C</title>\n");
    printf ("</head>\n");
    printf ("<body>\n");
    printf ("<h1>Hello World - C</h1>\n");
    printf ("<p>\n");
    printf ("Welcome to this small cgi script written as a C
   program!\n");
    printf ("</p>\n");
    printf ("</body>\n");
    printf ("</html>\n");
}
```

Beispiel für Hello World als Shell Script

```
#!/bin/bash
echo "Content-type: text/html; charset=utf-8"
echo ""
echo "<html>"
echo "<head>"
echo "<title>Hello World - bash</title>"
echo "</head>"
echo "<body>"
echo "<h1>Hello World - bash</h1>"
echo "<p>"
echo "Welcome to this small cgi script written as a bash script!"
echo "</p>"
echo "</body>"
echo "</html>"
```

Beispiel für Hello World als Perl Script

```
#!/usr/bin/perl
print "Content-type: text/html; charset=utf-8\n\n";
print "<html>\n";
print "<head>\n";
print "<title>Hello World - Perl</title>\n";
print "</head>\n";
print "<body>\n";
print "<h1>Hello World - Perl</h1>\n";
print "<p>\n";
print "Welcome to this small cgi script written as a Perl script!\n";
print "</p>\n";
print "</body>\n";
print "</html>\n";
```

Umgebungsvariablen bei CGI-Scripts

```
#!/bin/bash
echo "Content-type: text/html; charset=utf-8"
echo ""
echo "<html>"
echo "<head>"
echo "<title>Umgebungsvariablen</title>"
echo "</head>"
echo "<body>"
echo "<ul>"
echo "<li>QUERY_STRING $QUERY_STRING</li>"
echo "<li>REQUEST_METHOD $REQUEST_METHOD</li>"
echo "<li>REMOTE_ADDR $REMOTE_ADDR</li>"
echo "<li>REMOTE_HOST $REMOTE_HOST</li>"
echo "<li>SCRIPT_NAME $SCRIPT_NAME</li>"
echo "<li>QUERY_STRING $QUERY_STRING</li>"
echo "</ul>"
echo "</body>"
echo "</html>"
```

Verwendung des Headers

- Header für Ausgabe anderer Medientypen:

```
#!/bin/bash
echo "Content-type: image/gif"
echo ""
cat graphic.gif
```

- Header zur Umleitung auf andere Seiten:

```
#!/bin/bash
echo "Location: http://www.jroethig.de/"
echo ""

#!/bin/bash
echo "Location: /bilder/einbild.jpg"
echo ""
```

Nachteile bei CGI-Scripten

- CGI-Scripte können nicht an beliebigen Orten auf dem Web-Server gelagert werden
- cgi-bin-Verzeichnis kann oft nicht zur Lagerung von normalen HTML-Seiten oder Grafik-Dateien genutzt werden

➔ CGI-Script und “Inhalt” an getrennten Stellen

- höherer Aufwand zum Schreiben des HTML-Codes

➔ erhöhte Fehleranfälligkeit

Generierung dynamischen Web-Contents: Server Side Includes (SSI)

- Der einfachste Ansatz, um HTML-Seiten “dynamisch” zu erzeugen:

➔ Server Side Include (SSI)

- standardisiert seit NCSA HTTPd (bis Mitte der 90er-Jahre die am häufigsten eingesetzte WebServer-Software)
- unterstützt von gängigen WebServern (Apache, IIS)
- zusätzliches Parsen jeder Seite bedeutet zusätzliche Belastung der WebServer-Hardware
- Mehraufwand relativ gering verglichen mit PHP oder CGI

Server Side Includes (SSI) I

- Dateien, die auf SSI-Kommandos geparst werden sollen, haben den Filetype “.shtml”.
- Der größte Teil einer SSI-Seite besteht aus “normalem” HTML-Code.
- SSI-Komandos sind HTML-Kommentare (erscheinen nicht im WebBrowser, auch falls sie nicht geparst werden).
- Sie haben das Format
  <!--#command tag1="value1" tag2="value2" -->

Server Side Includes (SSI) II

- Mögliche SSI-Kommandos:
  - config
    verändert Einstellungen (Fehlerseite, Datumsformat, Größenangabe)
  - include
    fügt Inhalt einer anderen Datei in die Ausgabe ein
  - echo
    gibt Variablenwerte aus
  - fsize
    gibt Größe einer Datei aus
  - flastmod
    gibt Datum der letzten Modifikation einer Datei aus
  - exec
    führt ein Kommando aus

Server Side Includes (SSI) III

- Variablen sind alle der CGI-Environment-Variablen:
  - siehe Folie 153
- einige zusätzliche Variable:
  - DOCUMENT_NAME
  - DOCUMENT_URI
  - QUERY_STRING_UNESCAPED
  - DATE_LOCAL
  - DATE_GMT
  - LAST_MODIFIED

Server Side Includes (SSI) IV

- Beispiel: Ausgabe des letzten Änderungsdatums der aktuellen Datei
  - <!--#flastmod file="$DOCUMENT_NAME" -->

  oder
  - <!--#echo var="LAST_MODIFIED" -->
- Beispiel: Ausgabe des aktuellen Pfads
  - <!--#echo var="DOCUMENT_URI" -->

  oder
  - <!--#echo var="SCRIPT_NAME" -->

Server Side Includes (SSI) V

- Beispiel: Ausgabe des Servernames
  - <!--#echo var="SERVER_NAME" -->
- Beispiel: Ausgabe der Adresse des Aufrufers
  - <!--#echo var="REMOTE_ADDR" -->
- Beispiel: Ausgabe des aktuellen Datums und Zeit
  - <!--#echo var="DATE_LOCAL" -->
- Beispiel: Ausgabe des aktuellen Verzeichnislistings
  - <!--#exec cmd="ls -la" -->
- Beispiel: Einfügen einer Datei
  - <!--#include file=”menu.inc" -->

Anwendungsbeispiele SSI

- Automatische Ausgabe einer Statuszeile (letzte Änderung, Dateinamen, …)
- Einfügen feststehender Seitenbestandteile (beispielsweise von Navigationselementen)

➔ leichte Änderung des Websitedesigns

- Aufruf externer Kommandos zur Ausgabe von Elementen, die nicht per HTML und SSI direkt abrufbar sind

PHP-Scripte I

- “Gemischtes” Konzept
  - Web-Seite enthält an beliebigen Stellen PHP-Programmcode
  - oftmals Teil des HTML-Quelltextes
  - PHP-Code wird direkt auf und von dem Web-Server interpretiert
- PHP speziell für Nutzung im Web-Server konzipiert
  - spezielle Funktionen/Anweisungen, z.B. zum Inhalt des HTTP-Headers, zum Abruf von MySQL-Datenbanken, …
- PHP-Sequenzen werden gekennzeichnet mittels spezieller Begrenzer
  - <?php … php-Code … ?>

PHP-Scripte II

- PHP-Code enthaltende Dateien haben spezielle File Extension “php”
- Web-Server muß speziell für die Ausführung von PHP-Scripts konfiguriert sein

- PHP-Dateien haben keinen speziellen MIME-Type!
- Das Ergebnis der Ausführung von PHP-Scripts wird mit dem “gewöhnlichen” MIME-Type ausgeliefert.
- meist text/html

Beispiel für PHP-Seiten I

- Abfrage der Fähigkeiten und Konfiguration der PHP-Web-Server-Installation sowie Information zu den Umgebungsvariablen beim aktuellen Aufruf:

```
<?php phpinfo(); ?>
```

- ergibt umfangreiche Ausgabe

- Beispielausgabe

Beispiel für PHP-Seiten II

- Auslieferung von anderen Dateitypen:

```
<?php
Header( "Content-type: image/jpeg");
readfile('mypicture.jpg');
?>
```

- ergibt “mypicture.jpg” als Ausgabe

Auswertung von Forms mit PHP-Seiten I

- Besonders elegante Behandlung von per POST übermittelten Formulareingaben
- gewöhnliche Formulierung des Formulars als HTML-Datei
- Beispielformular

```
<HTML>
<HEAD>
<TITLE>A simple form</TITLE>
</HEAD>
<BODY>
<FORM method="POST" action="form.php">
```

Auswertung von Forms mit PHP-Seiten II

```
Fill in the text box: <INPUT type="text" name="mytextbox"
   size="20"><BR>
Check me or not: <INPUT type="checkbox" name="mycheckbox[]"
   value="1"><BR>
Check me or not: <INPUT type="checkbox" name="mycheckbox[]"
   value="2"><BR>
Check me or not: <INPUT type="checkbox" name="mycheckbox[]"
   value="3"><BR>
Choose one:
Blue <INPUT type="radio" name="myradio" value="blue" CHECKED>
Green <INPUT type="radio" name="myradio" value="green">
Yellow <INPUT type="radio" name="myradio" value="yellow"><BR>
Select something from this list:
<SELECT name="myselectbox">
<OPTION value="dog">Dog</OPTION>
<OPTION value="cat">Cat</OPTION>
<OPTION value="pig">Pig</OPTION>
</SELECT><BR>
<TEXTAREA name="mytextarea" rows="4" cols="30"></TEXTAREA><BR>
<INPUT type="submit" name="mybutton" value="Click me!">
</FORM>
</BODY>
</HTML>
```

Auswertung von Forms mit PHP-Seiten III

- “Ziel” (<form action=“…”>) verweist auf PHP-Datei

```
<html>
<head>
<title>Doing something with the form</title>
</head>
<body>
<p>
<?
if(isset($HTTP_POST_VARS['mybutton'])) {
```

Auswertung von Forms mit PHP-Seiten IV

```
echo "Great! You clicked the button!\n";
    echo "<br>You typed <em>" . $HTTP_POST_VARS['mytextbox'] . "</em>
   in the textbox.\n";
    if(isset($HTTP_POST_VARS['mycheckbox'])) {
        echo "<br>You checked the checkbox.\n";
    foreach($HTTP_POST_VARS['mycheckbox'] as $value) {
        echo "<br>You clicked checkbox number <em>" . $value .
   "</em>.\n";
    }
    } else {
        echo "<br>You didn't check the checkbox.\n";
    }
    echo "<br>The color you picked was <em>" .
   $HTTP_POST_VARS['myradio'] . "</em>.\n";
    echo "<br><em>" . $HTTP_POST_VARS['myselectbox'] . "</em> was
   chosen from the select list.\n";
    echo "<br>In the textarea, you typed: <em>" .
   $HTTP_POST_VARS['mytextarea'] . "</em>\n";
```

Auswertung von Forms mit PHP-Seiten V

```
} else {
    echo "Hmm...you must have come to this script without clicking the
   button.\n";
}
?>
</p>
<p>
<?php phpinfo(); ?>
</p>
</body>
</html>
```

Beispielausgabe

Vergleich SSI, PHP, CGI

	SSI	PHP	CGI
Parsendurch Web-Server	Ja	Ja	Nein
Aufwand/ Serverbelastung	Niedrig	Mittel	Hoch
Flexibilität	Niedrig	Hoch	Mittel
Mächtigkeit	Niedrig	Hoch	Hoch
HTML-Parsenim Editormöglich	Ja	Ja	Nein

Dynamische Inhalte im Web (XML)

- XML:
  - eXtensible Markup Language
  - universelle Auszeichnungssprache
  - “universal format for structured documents and data on the Web”
  - Definition von Sprachen mit Document Type Definitions (global für das gesamte Dokument) oder XML Namespaces (für einzelne Elemente)
  - Beispiel: XHTML als Nachfolger von HTML
  - Beschreibung der Darstellung eines XML-Dokuments mit XSL (eXtensible Stylesheet Language)
  - Umwandlung zwischen verschiedenen Formaten mitttels XSLT (XSL Transformation)

XML

- HTML verfügt nur über ein beschränktes Sortiment an Elementen (Tags).
- Inhalt und Struktur sind bei HTML nicht völlig getrennt, sondern konkurrieren miteinander (vs. <b> oder).
- Bei XML können beliebig strukturierte Informationen in einem textbasierten Format abgelegt werden.
- XML ist eine Metasprache, mit der sich jeder Anwender seine eigene Sprache erzeugen kann.
- Dokumente müssen wohlgeformt und gültig sein.

XML-Dateien

```
<folie>
<titel>XML</titel>
<bullet_list>
   <bullet_point>HTML verfügt ... (Tags).</bullet_point>
   <bullet_point> Inhalt ... &lt;strong&gt;).</bullet_point>
   <bullet_point> Bei XML ... werden.</bullet_point>
   <bullet_point> XML ist ... kann.</bullet_point>
</bullet_list>
</folie>
```

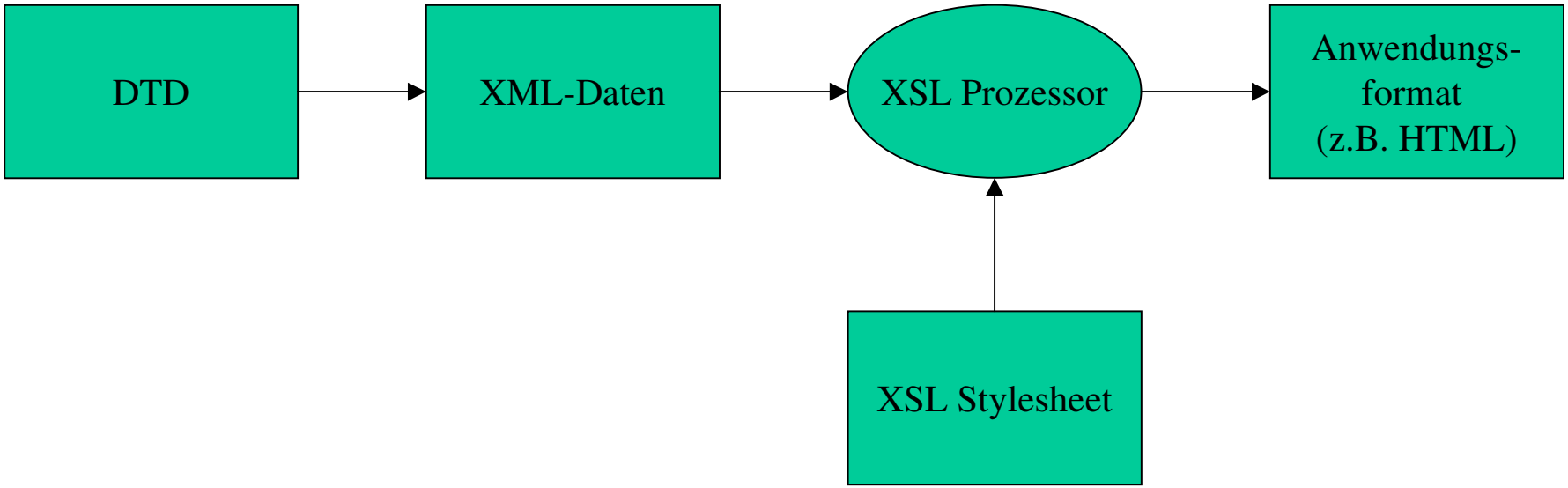
Document Type Definitions (DTD)

- Die Bauvorschriften für XML-Dateien werden in DTD-Dateien abgelegt.
- DTDs werden sehr viel seltener geändert als Dokumente.
- Beispiel:

```
<!-- Präsentation Version 0 -->
<!ELEMENT folie (titel, bullet_list)>
<!ELEMENT titel (#PCDATA)>
<!ELEMENT bullet_list (bullet_point+)>
<!ELEMENT bullet_point (#PCDATA)>
```

- Einige DTDs werden standardisiert und sind frei verfügbar.

Arbeitsprozess mit XML



Was bringt das Ganze?

- Daten werden in standardisiertem Format abgelegt.
- Jedes Programm ist in der Lage, die Daten zu lesen, zu interpretieren und zu schreiben.
- Programme können unterschiedliche Werkzeuge zur Bearbeitung der Dateien zur Verfügung stellen.
- Daten lassen sich einfacher zwischen verschiedenen Systemen austauschen.
- Mehrere Konvertierungsprozesse können hintereinander geschaltet werden.
- Schnittstelle zu Datenbanken

Wo geht es hin?

- Standardisierte DTDs für:
  - XHTML (Hypertext)
  - MathML (Mathematik)
  - Chemical Markup Language (Chemie)
  - Synchronized Multimedia Integration Language (SMIL)
  - Scalable Vector Graphics (SVG)
- Verbindung von mehreren Applikationen über XML-Dokumente
  - Datenbanken
  - SAP-Systeme
  - Skriptsprachen
  - Webserver

XSL

- Extensible Stylesheet Language
- besteht aus
  - XSLT: XSL Transformations
  - XPath: XML Path Language
  - XML-FO: XML Formatting Objects
- im folgenden:
  - XSLT zur Umwandlung von XML-Dokumente in andere XML-Dokumente (oder HTML)
  - XPath zur Auswahl von Elementen aus XML-Dokumenten innerhalb XSLT

XSLT

- Wo findet die Transformation statt?
- Client-side:
  - XML-fähiger Web-Browser
  - Mozilla (Version 1)
  - Internet Explorer (Version 6)
- Server-side:
  - Erweiterung des Web-Servers
  - Apache-Projekt Cocoon
  - Perl-Modul AxKit
- Standalone-Tool:
  - XSL processor
  - Apache: xalan
  - James Clark: xt
  - Michael Kay: saxon

XSL-Dokument-Rahmen

- <?xml version="1.0" encoding="ISO-8859-1"?>

  <xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <!-- Code für Transformationen (Templates) -->

  </xsl:stylesheet>

- XSL-Dokument ist selbst ein XML-Dokument
- wohlgeformt (Start-Tags und Ende-Tags)!

XML-Dokument-Rahmen

- <?xml version="1.0" encoding="ISO-8859-1"?>
  <?xml-stylesheet type="text/xsl" href="mystyle.xsl">

- Verweis auf das XSL-Dokument in "mystyle.xsl"

XSL: Pfad-Ausdrücke

- “/”
  - Dokumentwurzel
- “//”
  - irgendwo im Dokument
- “.”
  - aktuelles Element
- “..”
  - übergeordnetes Element

XSL: Template

- <xsl:template match=“ ”>
  <!-- Ausgabe-Code -->
  </xsl:template>
- Führt das Template für alle Elemente durch, auf die “match” zutrifft. Beispiel:
  - match=“/”
    - Dokument
  - match=“tag”
    - Element “Tag”

- rekursiver Aufruf:
- <xsl:apply-templates />

XSL: Ausgaben

- <xsl:text>Einzufügender Text</xsl:text>
- Ausgabe von “Einzufügender Text”

- <xsl:value-of select=“Element” />
- Ausgabe des Wertes (Inhaltes) des selektierten Elements

XSL: iterierte Ausführung

- <xsl:for-each select="Element"> <xsl:value-of"." /> </xsl:for-each>
- Ausführung für jedes "Element" im XML-Dokument

- <xsl:for-each select="Element[attribut='wert']"> <xsl:value-of"." /> </xsl:for-each>
- Filtern nach den Elementen, bei denen das Attribut "attribut" den Wert "Wert" hat
  - = gleich
  - != ungleich
  - < kleiner als
  - > größer als

XSL: bedingte Ausführung

- <xsl:if test="bedingung"> Ausgaben </xsl:if>
- Ausführung für jedes "Element" im XML-Dokument
- Aber: Es gibt keinen "else"-Zweig!

- <xsl:choose>
  <xsl:when test="Bedingung"> Irgendwas </xsl:when>
  <xsl:when test="Bedingung2"> Anderes </xsl:when>
  <xsl:otherwise> Ausgabe </xsl:otherwise>
  </xsl:choose>

Was haben wir? I

Bislang:

- (X)HTML zur Creation von Web-Seiten mit vorgegebener Struktur/funktionalen Mark-Ups
- CSS zur individuellen Gestaltung der Mark-Ups
- XML/DTD zur Creation von (Web-)Seiten mit selbst definierter logischer Struktur (eigenen Mark-Ups)
- XSL zur Umwandlung von XML-Seiten nach HTML zur Darstellung im Web-Browser
- diverse Bitmap-Bildformate zur direkten Einbindung in Web-Seiten
- Verweise zur Darstellung beliebiger Medienformen in Helper Applications

Was haben wir? II

Angesprochen, aber (noch) nicht näher behandelt:

- Objekte zur Einbindung von Medienformen in Web-Seiten mittels PlugIn
- Applets zur Programmierung dynamischer (programmierter) Vorgänge und Abläufe
- SVG zur Beschreibung von Vektor-Graphiken

Was fehlt?

- Medienformen/-darstellungen sind unabhängig voneinander
- programmierter Ablauf verschiedener Darstellungen nicht oder nur schwer möglich
  - Frames
  - JavaScript
  - Problem: schlechte Standardisierung (Jscript, VBScript, …)
- spezielle Sprache für Ablauf von Multimedia-Darstellungen wäre sinnvoll!

➔ SMIL (ausgesprochen wie engl. “smile”)

SMIL

- Synchronized Multimedia Integration Language
- basiert auf XML
- textbasiert
- Erstellung mit jedem Text-Editor möglich
  - diverse Authoring-Tools verfügbar
- zum Abspielen existieren verschiedene Tools:
  - Oratrix' s GRiNSPlayer (kommerziell, kostenpflichtig)
  - Internet Explorer
  - RealPlayer/RealOne
  - Apple Quicktime Player

Geschichte von SMIL

- 10/1996: W3C Workshop
- 03/1997: Working Group on synchronized multimedia
- 06/1998: SMIL 1.0 als W3C Recommendation
- 02/1999: Synchronized Multimedia Working Group (SYMM)
- 08/2001: SMIL 2.0 als W3C Recommendation

Möglichkeiten von SMIL 1.0

- describe the temporal behavior of the presentation
- describe the layout of the presentation on a screen
- associate hyperlinks with media objects

- basiert auf XML 1.0
- Spezifikation mittels DTD

Erweiterungen bei SMIL 2.0

- Neues MarkUp:
  - transition effects
  - animation
  - author-defined windows
  - improved event-handling (events can come from UI or from network)
  - hierarchical layout
  - authoring adaptive content (e.g. to have the same document display on a PC and on a mobile device).
- SMIL 2.0 is defined as a set of reusable markup modules
  - Mobile Phones
  - Animation in SVG
- Define a set of language profiles that incorporate the SMIL 2.0 modules
  - Bsp: XHTML+SMIL

Grundaufbau einer SMIL-Datei

- <smil>
  <head>
  <layout>
  <!-- Aufteilung der Darstellungsfäche -->
  </layout>
  <body>
  <!-- darzustellende Objekte und zeitliche Abfolge -->
  </body>
  </smil>

Layout

- <root-Layout width=“w” height=“h” background-color=“c” />
- gibt die Größe der gesamten Präsentation an
- <region id=”id" left=”x" top=”y” width=”w" height=”h" />
- definiert Bereiche innerhalb der Präsentation

Layout Größenangaben

- Absolute Angaben (Pixel):
  - n
- Relative Angaben:
  - xx%

Layout Überlappung

- Normalerweise wird die letzte Region vollständig dargestellt.
- Überlappung kann aber auch gezielt beeinflusst werden.
  - Dritte Dimension: z-index
  - Region mit größerem z-index liegt über Region mit kleinerem z-index

Layout weiteres

- Medienobjekte werden innerhalb einer Region mit ihrer normalen Größe dargestellt.
- Mittels des Attributes fit im region-Tag kann erreicht werden, daß der Bereich anders ausgefüllt wird.
  - Ausdehnung auf kompletten Bereich: fit="fill"
  - maximale maßstabsgerechte Ausdehnung: fit="meet"
  - komplettes maßstabsgerechtes Ausfüllen: fit="slice"
  - Scrollbalken bei normaler Größe: fit="scroll"

Medienobjekte

- Einfache Medienobjekte
  - generisches Element ref
  - typenspezifische Elemente
    - animation
    - audio
    - img
    - video
    - text
    - textstream

Medienobjekte Attribute

- Attribute:
  - Ort der Mediendatei: src
  - MIME-Type der Mediendatei: type
  - Verweis auf darzustellenden Bereich (entsprechende ID): region
  - Clipping: clipBegin, clipEnd

Medienobjekte Timing

- Dauer der Präsentation: dur
- Verzögerung: begin
- Ende: end
- Ausschnitt: clip-begin, clip-end

- Zeitangabe in Sekunden: begin=“4s”
- Zeitangabe relativ zu einem anderen Element: begin=“id(anderesElement) (4s)”

- Anzahl der Wiederholungen: repeat

Komplexe Medienobjekte

- Sequenz (zeitliche Abfolge)
- <seq>
  <!-- Abfolge von Medienobjekten -->
  </seq>

- Parallelität (Gleichzeitigkeit)
- <par>
  <!-- gleichzeitig dargestellte Medienobjekte -->
  </par>

Medienobjekte Auswahl

- <switch>
  <-- Medienobjekt mit Test-Attribut -->
  </switch>
- Erstes Medienobjekt, für das der Test wahr ist, wird dargestellt.
- Ein Medienobjekt ohne Test ist immer wahr.
- Mögliche Tests:
  - Auflösung: system-screen-size
  - Farbtiefe: system-screen-depth
  - verfügbare Datenrate: system-bitrate
  - Sprache: system-language
  - verfügbare MIME-Types: type

Anwendungen im Internet: remote login

Eine der grundlegendsten und frühesten Anwendungen:

- Nutzen von Rechnerressourcen über das Netz (wie direkt am lokalen Rechner)
- Kommunikation zwischen Terminal und Host war eine der ersten Anwendungen von Rechnerkommunikation

➔ gleichzeitige Nutzung von teurer Rechner-Hardware durch mehrere Benutzer

- heute standardisierter Zugang mittels telnet (Sicherheitsproblem) bzw. ssh (secure shell, verschlüsselte Kommunikation)

Anwendungen im Internet: Email I

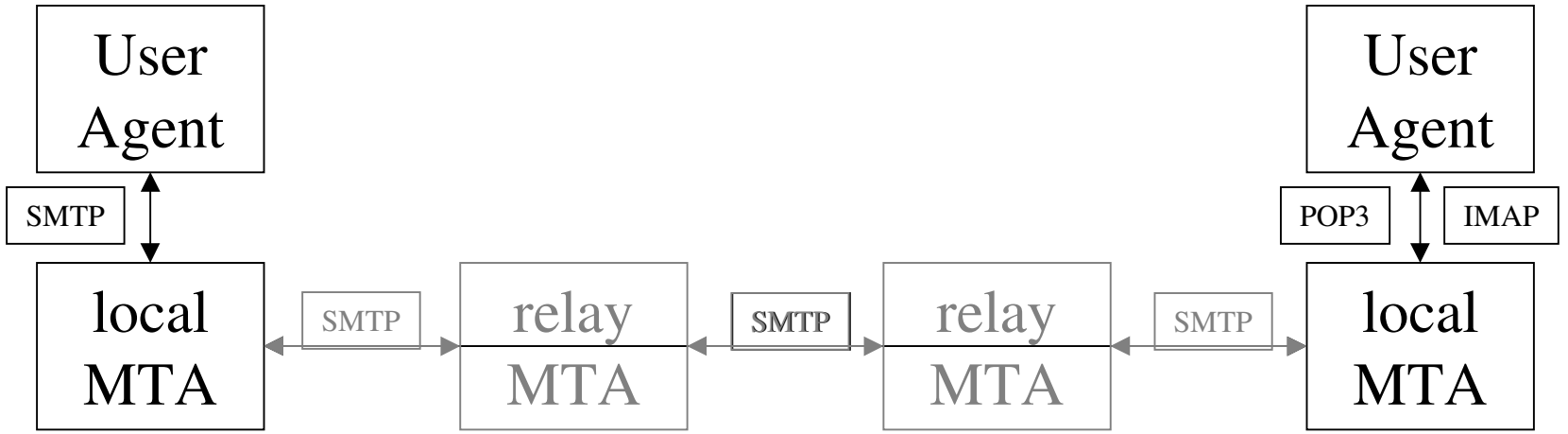
Die einfachste Anwendung:

- Austausch von Nachrichten, “Verschicken von Briefen”
- electronic Mail, EMail
- Wie genau funktioniert das Verschicken einer (elektronischen) Nachricht?
  - Möglichkeit: direkte Kontaktaufnahme des Senders mit dem Empfänger, direkte Auslieferung
  - alle Rechner im Internet sind (potentiell) miteinander verbunden
  - aber: die meisten privat betriebenen Rechner sind nicht ständig im Netz und sind nicht einmal ständig eingeschaltet.
- Wie funktioniert das Verschicken einer (elektronischen) Nachricht trotzdem?

Anwendungen im Internet: Email II

- Wie genau funktioniert das Verschicken eines (herkömmlichen) Briefes?
  - Schreiben des Briefes
  - Einwurf in den Sammelbriefkasten/Abgabe bei der Post
  - Transport durch die Post (über mehrere Zwischenstationen)
  - Verteilung durch Briefträger/Einwurf in persönlichen Briefkasten
  - Entnahme aus Briefkasten
  - Lesen des Briefs

Anwendungen im Internet: Email III



MTA: Message Transfer Agent
SMTP: Simple Mail Transfer Protocol
POP3: Post Office Protocol
IMAP: Interactive Mail Access Protocol

Anwendungen im Internet: Email IV

- Möglichkeiten von Email:
  - ursprünglich ausschließliche Übertragung von Textnachrichten (vgl. 1971 in Zeitleiste)
  - Standardisierung im RFC 822 (bzw. STD 11) von 1982
  - Beschränkung auf reine Textnachrichten war zu restriktiv
  - direkte Übertragung binärer (8bit-kodierter) Daten allerdings oft unmöglich
- erste Lösung zur Übertragung beliebiger Daten:
  - uuencode (Unix-to-Unix encoding) (in BSD 4 von 1980)
  - Kodierung von binären Daten (8bit) als ASCII-Daten (7bit)
  - Übertragung als “normale Textmail”
  - kein Automatismus zur Erkennung multimedialer Daten
  - aus jeweils 45 Zeichen werden 62 Zeichen
  - erhöhtes Datenvolumen!

Anwendungen im Internet: Email V

- Aktuelle Lösung zur Übertragung beliebiger Daten:
  - MIME-Type (korrekt: Media/Content Type)
- MIME: „Multipurpose Internet Mail Extensions“
  - ursprünglich (im Juni 1992 als RFC 1341 und 1342) zum Versand von Nicht-Text-EMails entwickelt
  - heute universelle Nutzung des Content Type, insbesondere auch im WWW (im HTTP-Header)
- aktuelle Definition von „MIME-Types“ in RFC 2046
  - top-level media types: text, image, audio, video, application
  - sub-level media types detaillieren die Angabe, beispielsweise: text/html, image/jpeg, audio/basic, video/mpeg, application/x-tar

Anwendungen im Internet: Email VI

- Probleme bei Email:
  - Phänomen der “Spam-Mails”
  - unerwünschte/unangeforderte Email
    - Werbung für (zweifelhafte) Produkte
    - Versuch der Installation von unerwünschten Dialern beim Empfänger
  - Versender behaupten oft, man habe sich für eine entsprechende Mailing-Liste angemedet (“subscribe”, “opt-in”)
  - laut amerikanischer Gesetzgebung ist es legal, Werbung per Email zu versenden, sofern es die Möglichkeit gibt, Werbung dieses Absenders abzubestellen (“opt-out”)
  - echter Absender oftmals nicht ausfindig und haftbar zu machen
  - mangelnde Authentifizierung beim Email-Versand-Protokoll smtp

Anwendungen im Internet: Email VII

- Woher bekommen die Versender von Spam-Mails die Email-Adressen ihrer Opfer?
  - Absuchen von Web-Seiten nach Email-Adressen
  - Gewinnung von Email-Adressen aus Newsgroup-Postings
  - Bestätigung, daß Email-Adresse wirklich genutzt wird, aus “opt-out”-Anforderungen
- Was kann man dagegen tun?
  - Benutzung von speziellen (entweder gar nicht gelesenen oder nur temporär gültigen) Email-Adressen in Newsgroups
  - keine Verwendung von direkt verwendbaren “mailto:”-Links auf öffentlichen Webseiten
  - keine Benutzung von “opt-out”-Mechanismen!

Anwendungen im Internet: News I

Was sind “News” und “Newsgroups”?

- “schwarzes Brett” im Netz
- Vorläufer:
  - “Bulletin Boards”
  - entstanden in den 80er Jahren des letzten Jahrhunderts, also zu Zeiten der ersten Akustikkoppler und Modems, mit denen man von zu Hause aus Datenübertragung über das Telefonnetz zu anderen Rechnern durchführen konnte
  - Privatleute betrieben spezielle Einwahlrechner, die rund um die Uhr oder zu bestimmten Uhrzeiten über Telefonleitungen direkt angewählt werden konnten
  - dort konnte neben Software auch Nachrichten abgerufen und Diskussionen geführt werden

Anwendungen im Internet: News II

Usenet News:

- Netz von verteilten News-Servern, die untereinander die Nachrichten an die anderen Server verteilen
- Die Nachrichtenbretter werden in (themenbezogene) Gruppen (groups) unterteilt.
- Gruppen sind hierarchisch gegliedert.
- Ein Benutzer aboniert eine Gruppe bei einem (“seinem”) News-Server und kann dann alle in dieser Gruppe geschriebenen Nachrichten lesen.
- Ein für eine Gruppe geschriebener Beitrag wird an alle anderen News-Server, welche diese Gruppe führen, weiterverteilt.

Anwendungen im Internet: News III

Probleme von News:

- Jeder News-Server muß alle Nachrichten aller Gruppen, die er führt, bereithalten.
- Die Speicherung dieser Nachrichten kann zu einem großen Dateivolumen führen.

➔ Die Haltezeit der Nachrichten ist beschränkt (zwischen mehreren Stunden und einigen Monaten).

➔ Die meisten News-Server führen nur einen Teil aller existierenden Gruppen.

Anwendungen im Internet: Chat I

Der ursprüngliche Chat im Internet:

- Internet Relay Chat (IRC)
- konzipiert 1988 vom finnischen Studenten Jarkko Oikarinen zur Kommunikation im lokalen Uni-Netz
- Netz von verteilten IRC-Servern, die untereinander die aktuellen Chat-Daten (Texte) austauschen
- Benutzer melden sich mit Hilfe einer IRC-Anwendung (z.B. mIRC) unter einem eindeutigen Nickname bei einem IRC-Server an und können über diesen Server dann mit allen an irgendeinem Server desselben IRC-Netzes angemeldeten Benutzern kommunizieren
- Kommunikation erfolgt in Kanälen (Channels): Alle in einem Kanal getippten Texte erscheinen (quasi gleichzeitig) bei allen anderen im selben Kanal eingeloggten Benutzern

Anwendungen im Internet: Chat II

Probleme von IRC:

- Skalierbarkeit: Nicknames der Benutzer müssen im IRC-Netz eindeutig sein, weshalb jeder IRC-Server alle Nicknames im Netz kennen muß
- Lag: Die Informationen (Texte) breiten sich nur mit einer gewissen Verzögerung im Netz der IRC-Server aus
- Net Split: Falls einzelne Verbindungen zwischen IRC-Servern zusammenbrechen, kann ein IRC-Netz in mehrere Teilnetze auseinanderbrechen

➔ mittlerweile gibt es viele verschiedene IRC-Netze

Anwendungen im Internet: Chat III

Web-Chat :

- Chat-Client im Web-Browser
- Grundsätzlich zwei Möglichkeiten:

  – Web-Client (also Anwendung im Web-Browser) für IRC

  ➔ nur ein spezieller Zugang zu einem der IRC-Netze

  – spezielle Anwendung, die auf einem Web-Server einen (lokalen) Chat implementieren und diesen über einen Web-Server den Benutzern bereitstellen

  ➔ (meist) proprietäre Lösung, also nicht erweiterbar und nicht interoperabel

Vergleich von Chat und News

- Die Kommunikation geschieht bei IRC in Realzeit, bei News dagegen zeitversetzt.
- Kommunikation bei IRC ist “direkter”.
- Bei IRC ist auch das Eröffnen und Nutzen von privaten Kanälen möglich.
- Nutzung von News ist auch offline möglich; bei IRC ist dies nicht möglich.
- Beide wurden ursprünglich zur reinen textbasierten Kommunikation konzipiert.
- Beide wurden um Möglichkeiten zum Austausch beliebiger (binärer) Dateien erweitert.

Anwendungen im Internet: Dateiabruf

Abruf von Dokumenten im Web

- Internet wurde und wird genutzt, um Informationen für andere Nutzer bereitzustellen.

➔ wissenschaftliche Texte, Programme, …

- Aber wie?
- FTP (File Transfer Protocol)
- Bereitstellen der Dokumente auf einem FTP-Server
- Anmelden am FTP-Server (mit Benutzernamen und Passwort oder anonym)
- Abruf der Dokumente

Anwendungen im Internet: Telefonie

IP-Telefonie/”Voice Over IP (VOIP)”

- Sprachkommunikation über IP-basierte Netzwerke
- Abbildung der Funktionalität des Telefonnetzes auf das Internet.
- Es gibt (noch) keine etablierte Internet-Telefonie-Anwendung und kein eigenständiges Internet-Telefonie-Netz.
- Der Protokoll-Standard H.323 beschreibt neben den Funktionalitäten eines VOIP-Systems vor allem eine Schnittstelle zum herkömmlichen Telefonsystem.
- IP-Telefonie existiert bislang vor allem in Insellösungen.