Webtechnologien

Vorlesung Informatik
Dr. rer. nat. Harald Sack
Institut für Informatik
Friedrich Schiller Universität Jena

Sommersemester 2006

http://www.informatik.uni-jena.de/~sack/SS06/webtechnologien/webtechnologien.htm

Webtechnologien

- **Teil I – Internet und WWW**

Webtechnologien

Teil I: Internet und WWW

1. Internet
2. **World Wide Web**
3. Web-Programmierung
4. Sicherheit im WWW
5. Web-Services
6. WWW-Suchmaschinen

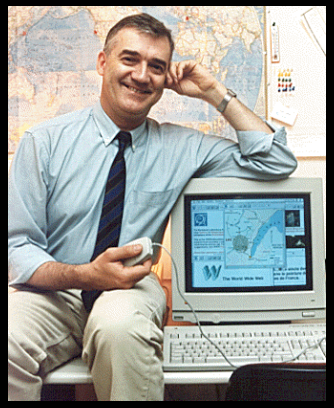
Teil I: Internet und WWW

2. World Wide Web

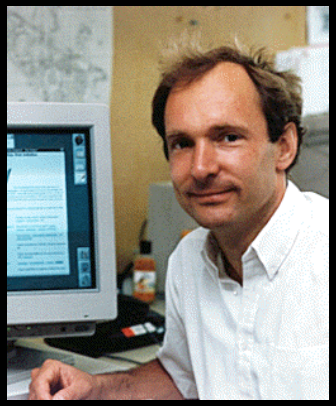
- WWW – Historisches und Grundlagen
- HTML/CSS und http
- WWW-Server
- WWW-Caching
- CGI-Schnittstelle
- Cookies / Web-Bugs
- DOM
- XML und XML-Derivate

2. World Wide Web

- Kurze Geschichte des **WWW**
  - Das WWW wurde **1990** am CERN zur Dokumentenverwaltung entwickelt
  - Der erste WWW-Server ging **1991** online

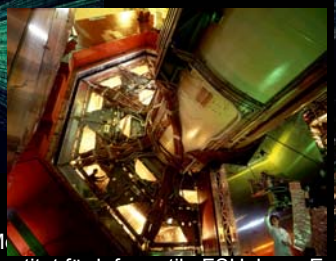


Robert Cailliau



Sir Tim Berners Lee

http://nxoc01.cern.ch



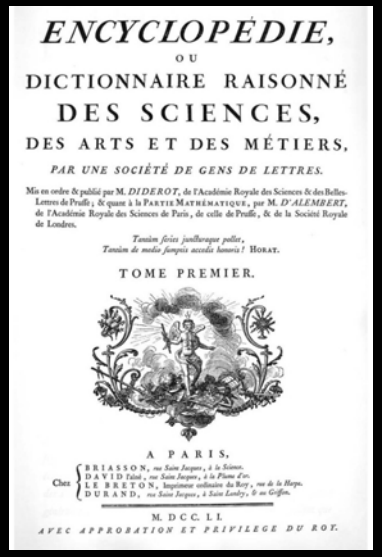
CERN
Conseil Européen
pour la Recherche Nucléaire

2. World Wide Web

- Kurze Geschichte des **WWW**
  - Aber die ursprüngliche Idee reicht viel weiter zurück…

Problem: Benutzerschnittstelle

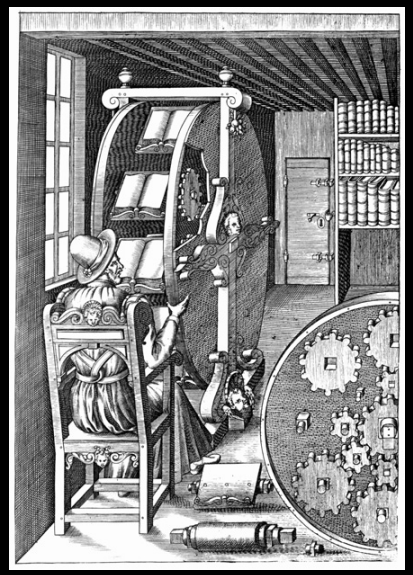
Talmud



Encyclopédie ou Dictionnaire raisonné des sciences, des arts et des métiers (1772)

Jean-Baptiste le Rond d'Alembert (1717-1783)

Denis Diderot (1713-1784)



Agostino Ramelli (1588), *Le diverse et artificiose machine; composte in lingua Italiana et Francese*

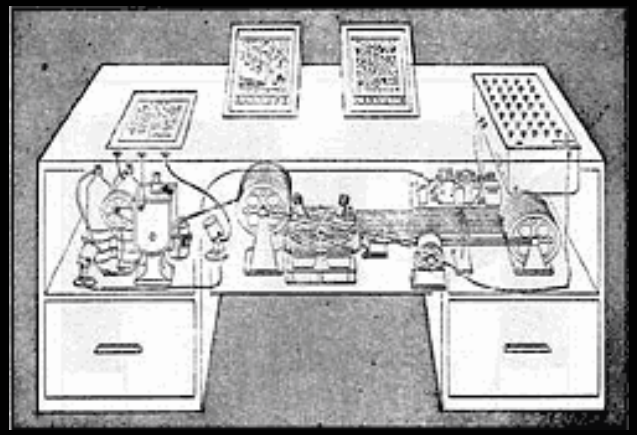
2. World Wide Web

Kurze Geschichte des **WWW**

- Aber die ursprüngliche Idee reicht viel weiter zurück…

Vennevar Bush
(1890-1974)

- Vennevar Bush schlägt **1945** das erste Hypertext-System „**MEMEX**“ vor



2. World Wide Web

Kurze Geschichte des **WWW**

- Aber die ursprüngliche Idee reicht viel weiter zurück…

Douglas C. Engelbart
(*1925)

- Douglas Engelbart entwickelt **1962-1968** den Prototyp eines Hypertextsystems (**NLS** – On-line System) mit
  - fensterbasierter Benutzeroberfläche
  - Maus als Eingabegerät
  - editierbarem Hypertext / Hypermedia
  - Groupware Eigenschaften

erste Computer Maus

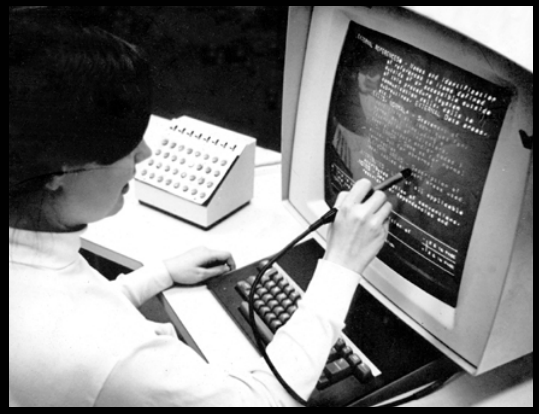
2. World Wide Web

Kurze Geschichte des **WWW**

- Aber die ursprüngliche Idee reicht viel weiter zurück…

Ted Nelson
(*1937)

- Ted Nelson prägt **1965** als erster den Begriff „Hypertext“
  - **Xanadu** - unvollendetes Hypertextsystem Projekt
  - unterstützt Andries v. Dam in der Entwicklung von **HES** – Hypertext Editing System (1968/1969)

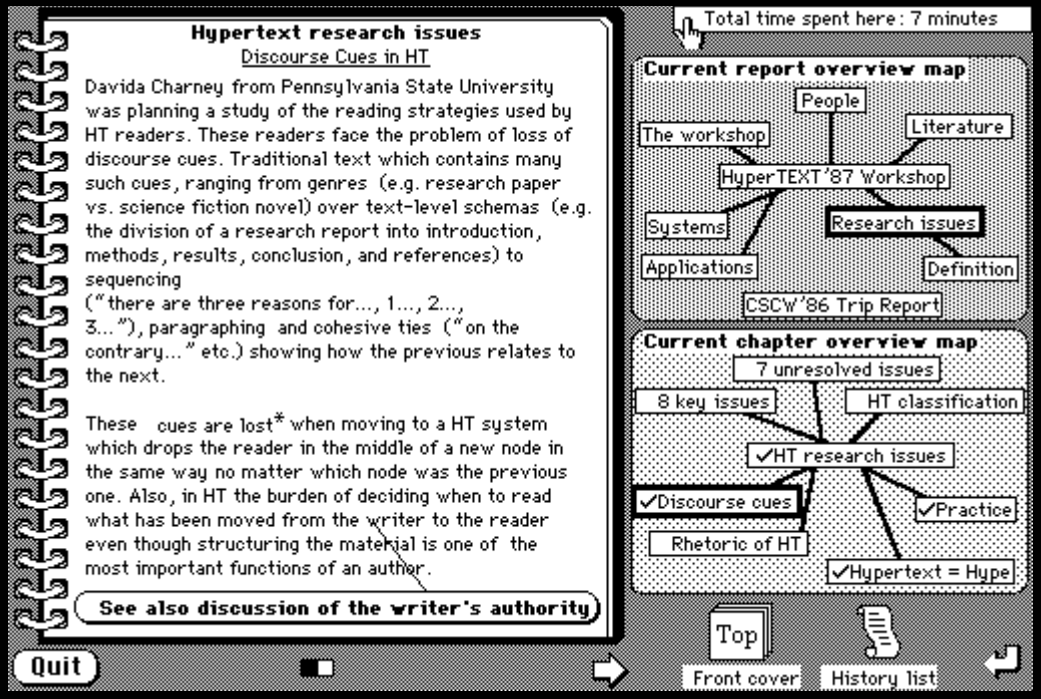


HES – Hypertext Editing System (1969)

2. World Wide Web

Kurze Geschichte des **WWW**

- Aber die ursprüngliche Idee reicht viel weiter zurück…



Apple HyperCard (seit 1987)

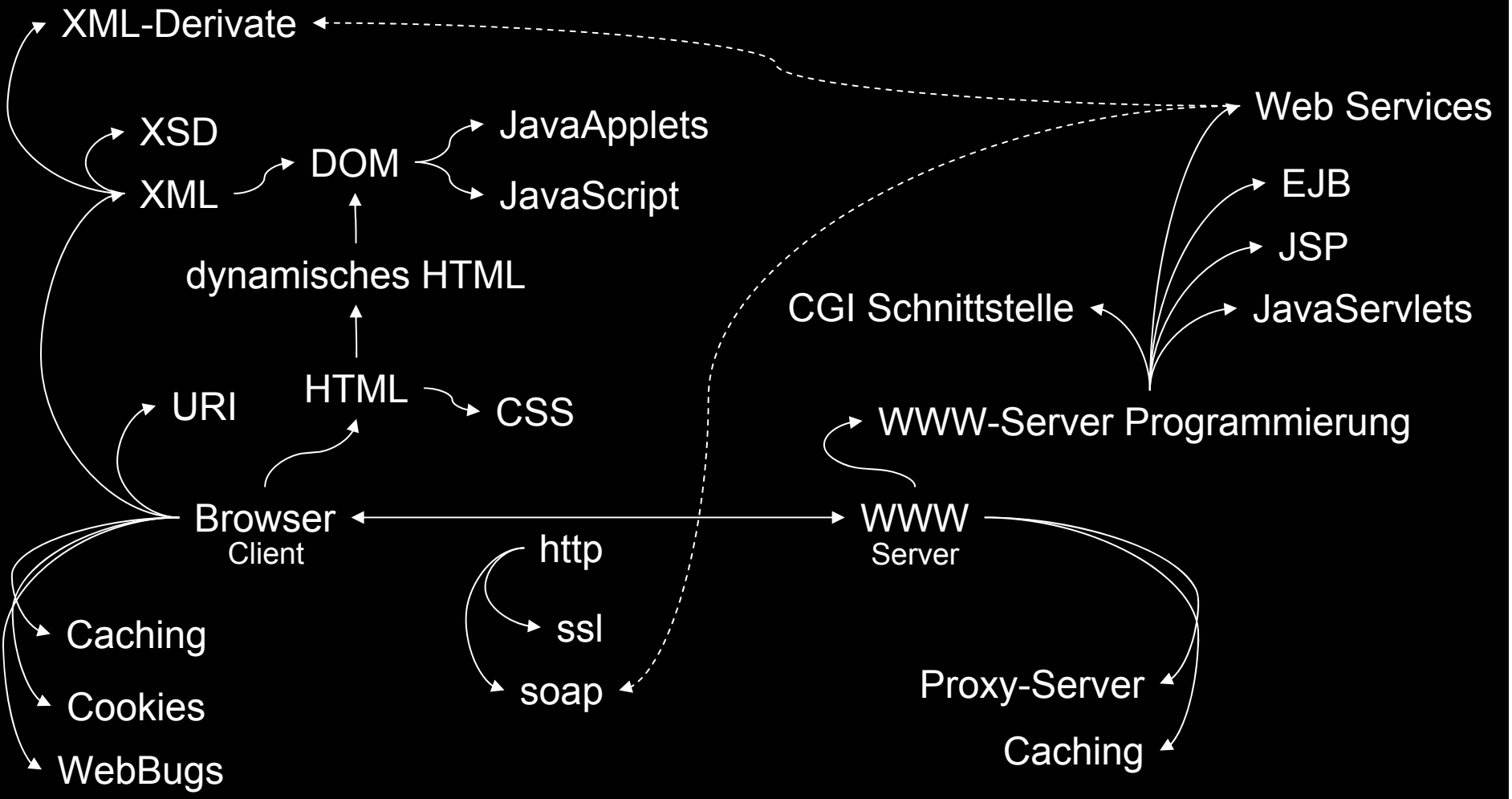
- **1987** erobert Hypertext und Hypermedia mit **Apple's Hypercard System** den Personal Computer

Teil I: Internet und WWW

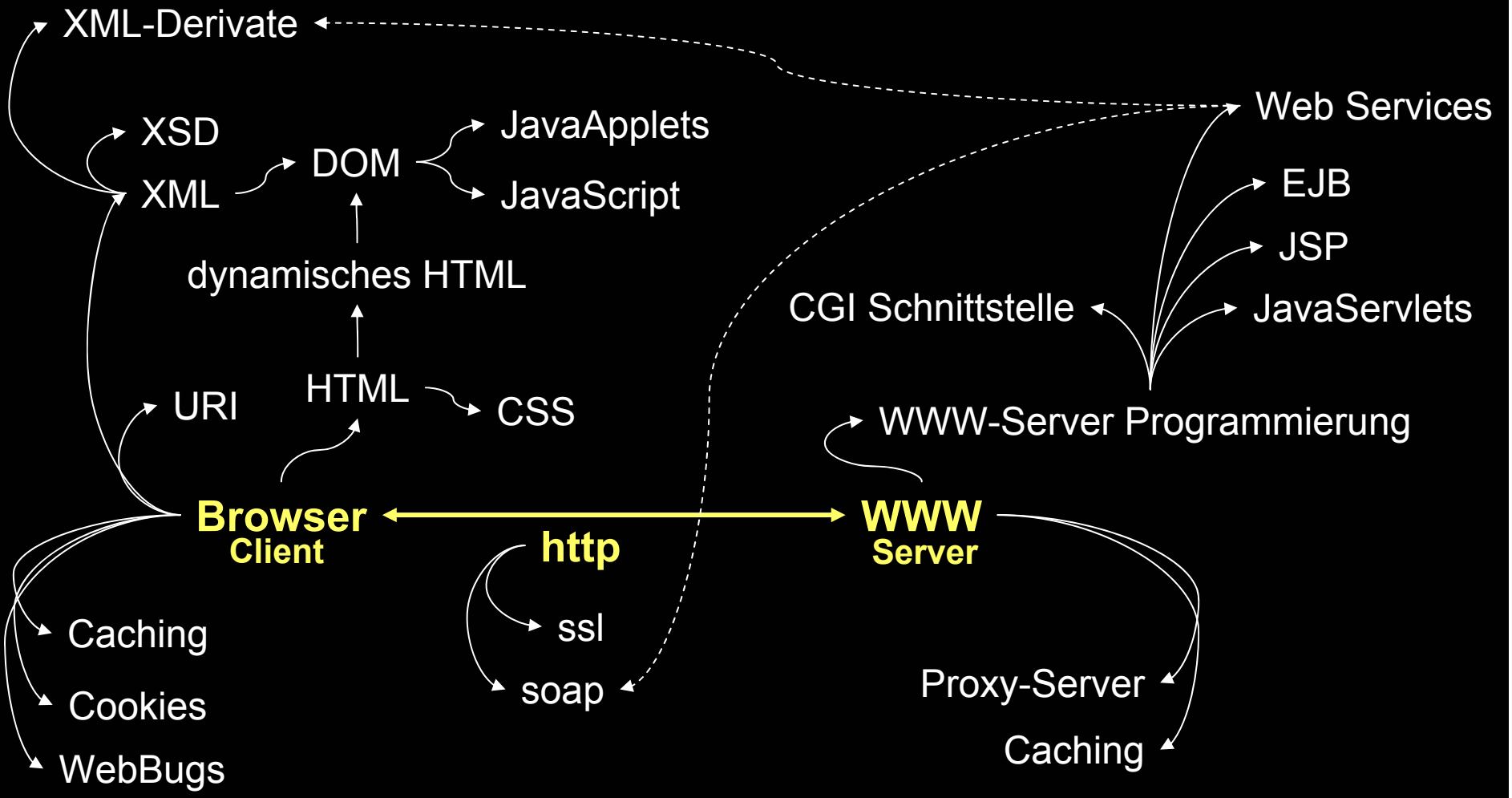
2. World Wide Web

- WWW – Historisches und Grundlagen
- HTML/CSS und http
- WWW-Server
- WWW-Caching
- CGI-Schnittstelle
- Cookies / Web-Bugs
- DOM
- XML und XML-Derivate

2. World Wide Web



2. World Wide Web



2. World Wide Web

Versuch einer Definition……

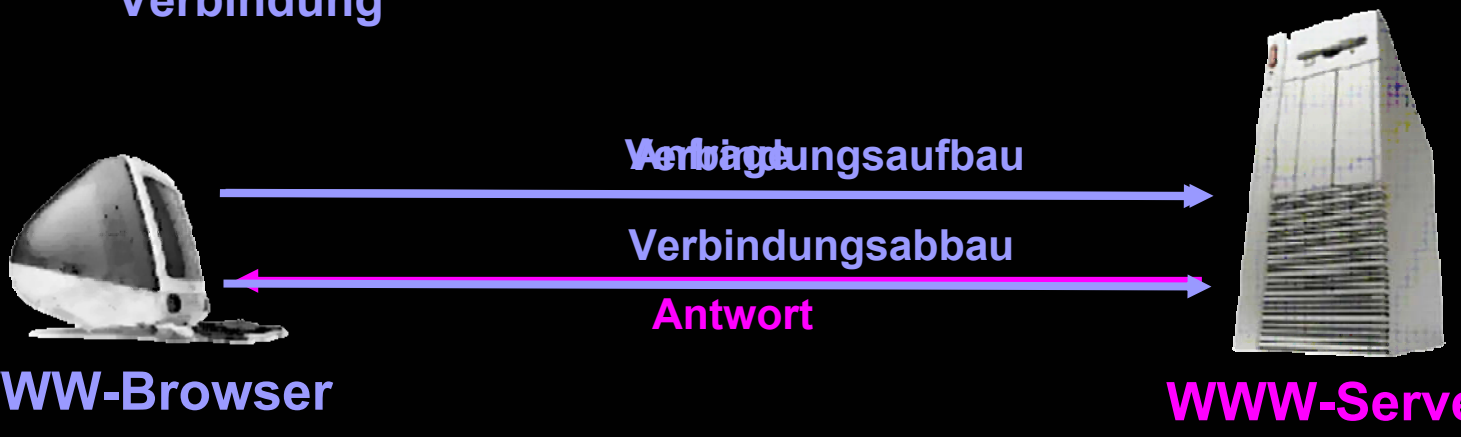
World Wide Web / WEB / **WWW** / W3

- ist ein riesiges Online-Informationslager, auf das mit Hilfe eines interaktiven Anwendungsprogrammes namens „**Browsers**“ zugegriffen werden kann
- Internet-Ressourcen, auf die mit Hilfe des **HTTP-Protokolls** (**Hypertext Transfer Protocol**) zugegriffen werden kann
- Weltweit verteilte, multimediale Informationsressourcen, die untereinander über **Hyperlinks** miteinander verbunden sind

2. World Wide Web

WWW – eine Client/Server-Anwendung

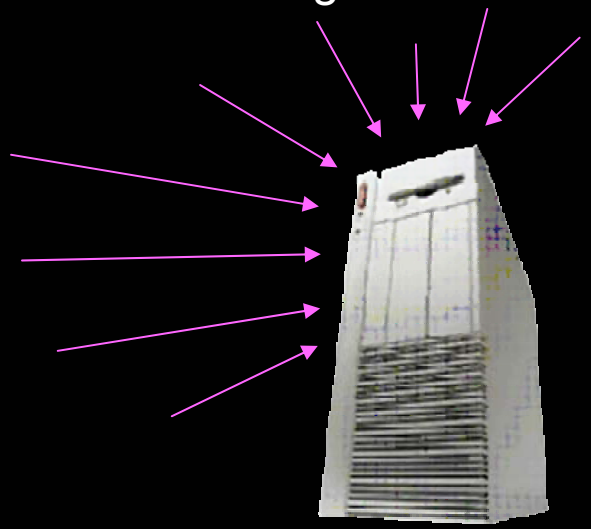
- Verbindung zwischen **WWW-Browser** und **WWW-Server** ist stets nur von kurzer Dauer
  1. Browser **baut** eine **Verbindung** zum Server **auf**
  2. Browser **sendet** eine **Anfrage**
  3. Server **antwortet** mit angefordertem Dokument oder Fehlermeldung
  4. Browser empfängt angefordertes Dokument und **beendet** die **Verbindung**



2. World Wide Web

WWW – Server

- **WWW-Server** sind weniger komplex als Browser
  - **Server** führen wiederholt einfache Aufgaben aus:
    - **Warten** auf Eröffnung einer Verbindung durch Browser und Anforderung eines Dokuments,
    - **Ausgabe** dieses Dokuments bzw. Fehlermeldung und
    - **Schließen** der Verbindung.
    - erneutes **Warten** auf Eröffnung
    - ...

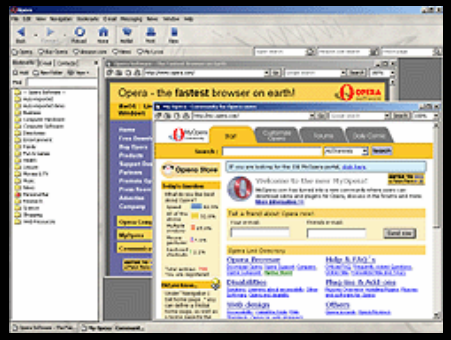


WWW-Server

2. World Wide Web

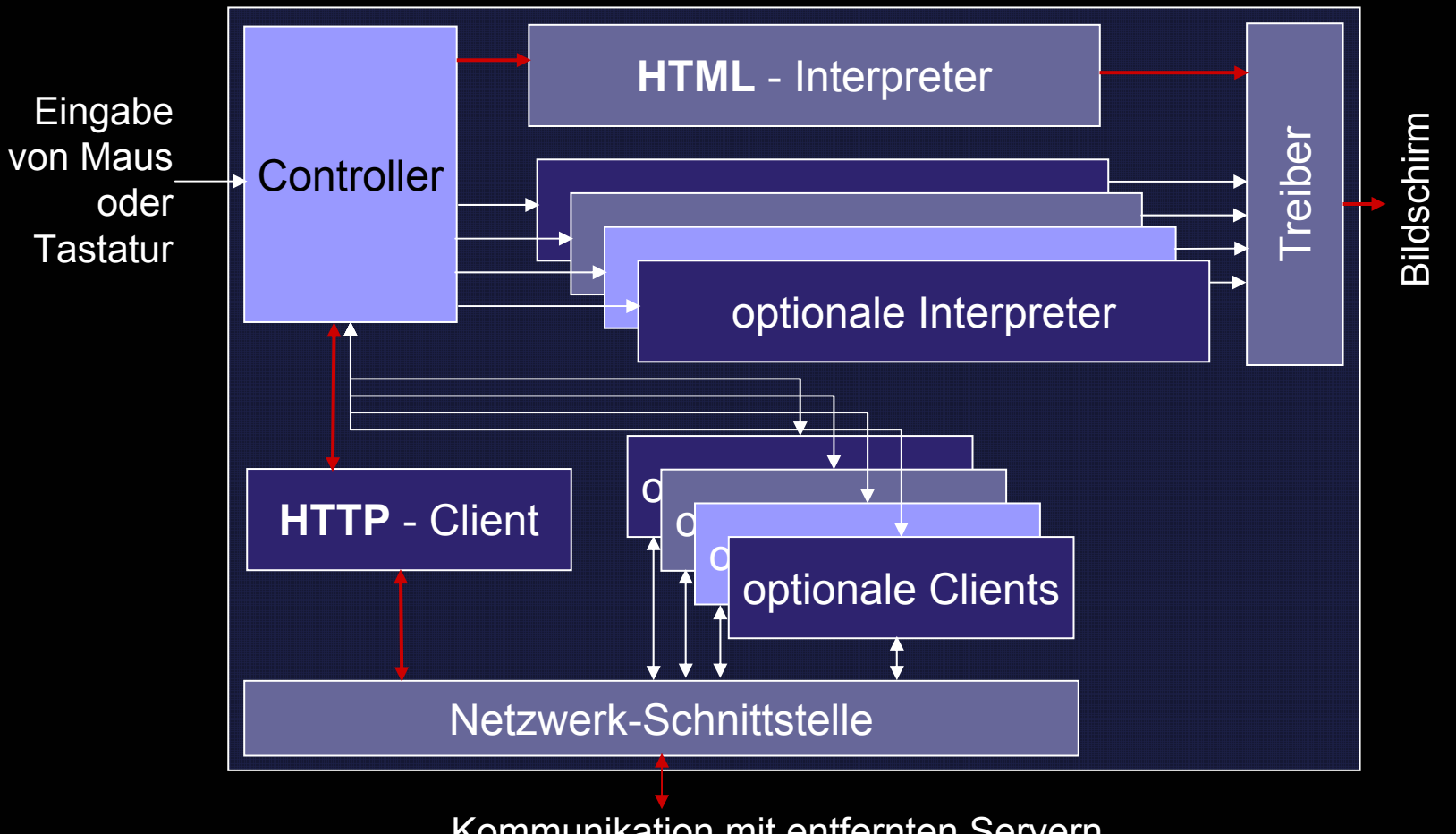
WWW – Browser

- **Browser** haben mehr zu leisten als WWW-Server:
  - **Eröffnung** der Verbindung zum WWW-Server
  - **Anforderung** eines Dokuments
  - **Einlesen** des angeforderten Dokuments
  - **Anzeigen** des empfangenen Dokuments
  - **Reagieren** auf Aktionen des Benutzers bei der Bedienung der grafischen Benutzeroberfläche

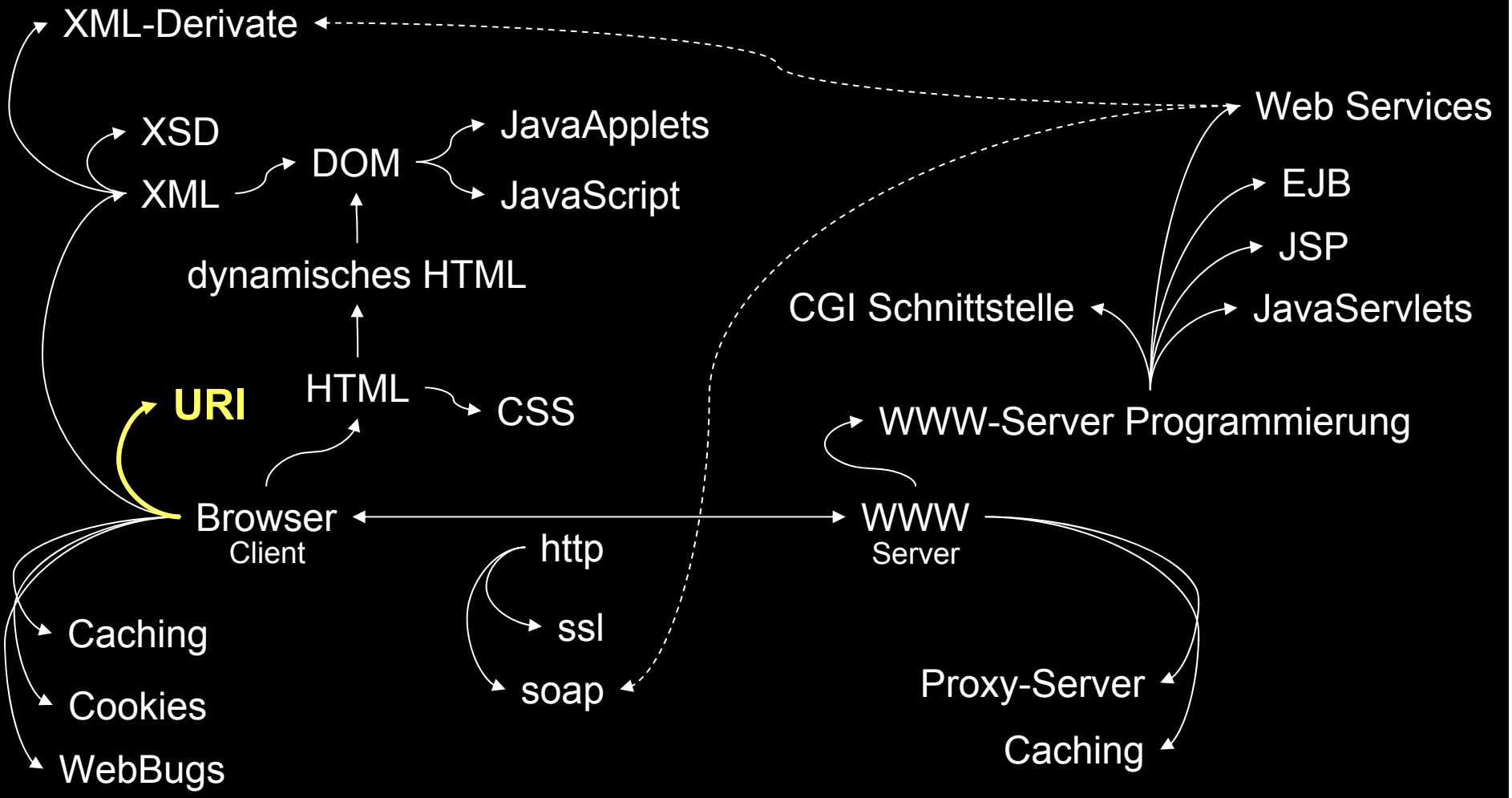


2. World Wide Web

Browserarchitektur



2. World Wide Web



2. World Wide Web

Uniform Resource Identifier (URI)

- WWW-Page muss **eindeutig identifizierbar** sein
  (Speicherort, Name, Darstellungsart, evtl. integrierte Anwendung)

- **Realisierung im WWW:**
  - Kodierung notwendiger Informationen über **eindeutige Zeichenkette**

    Uniform Resource Identifier – URI

  - Ein URI besteht entweder aus einem
    - **URN** (**Uniform Resource Name**) oder einem
    - **URL** (**Uniform Resource Locator**)

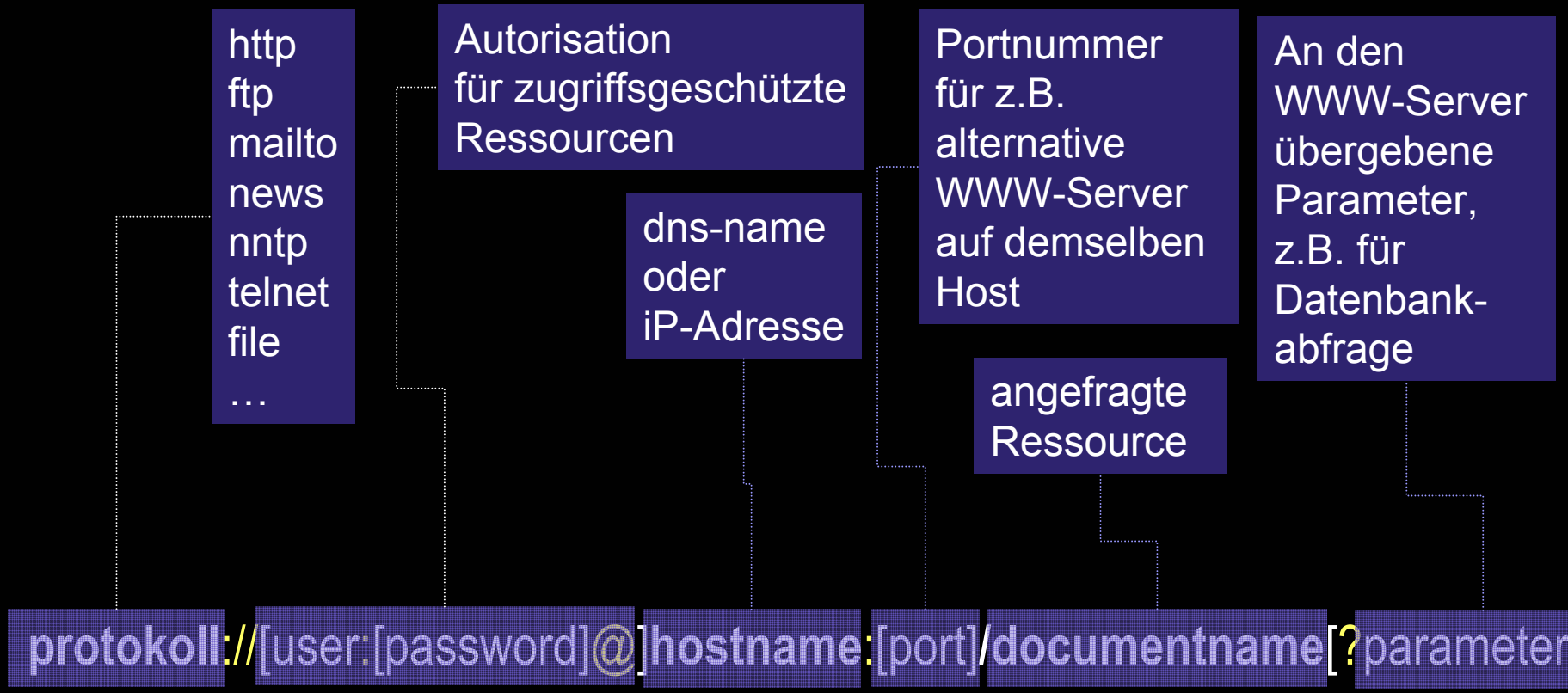
Name → URN

Adresse → URL

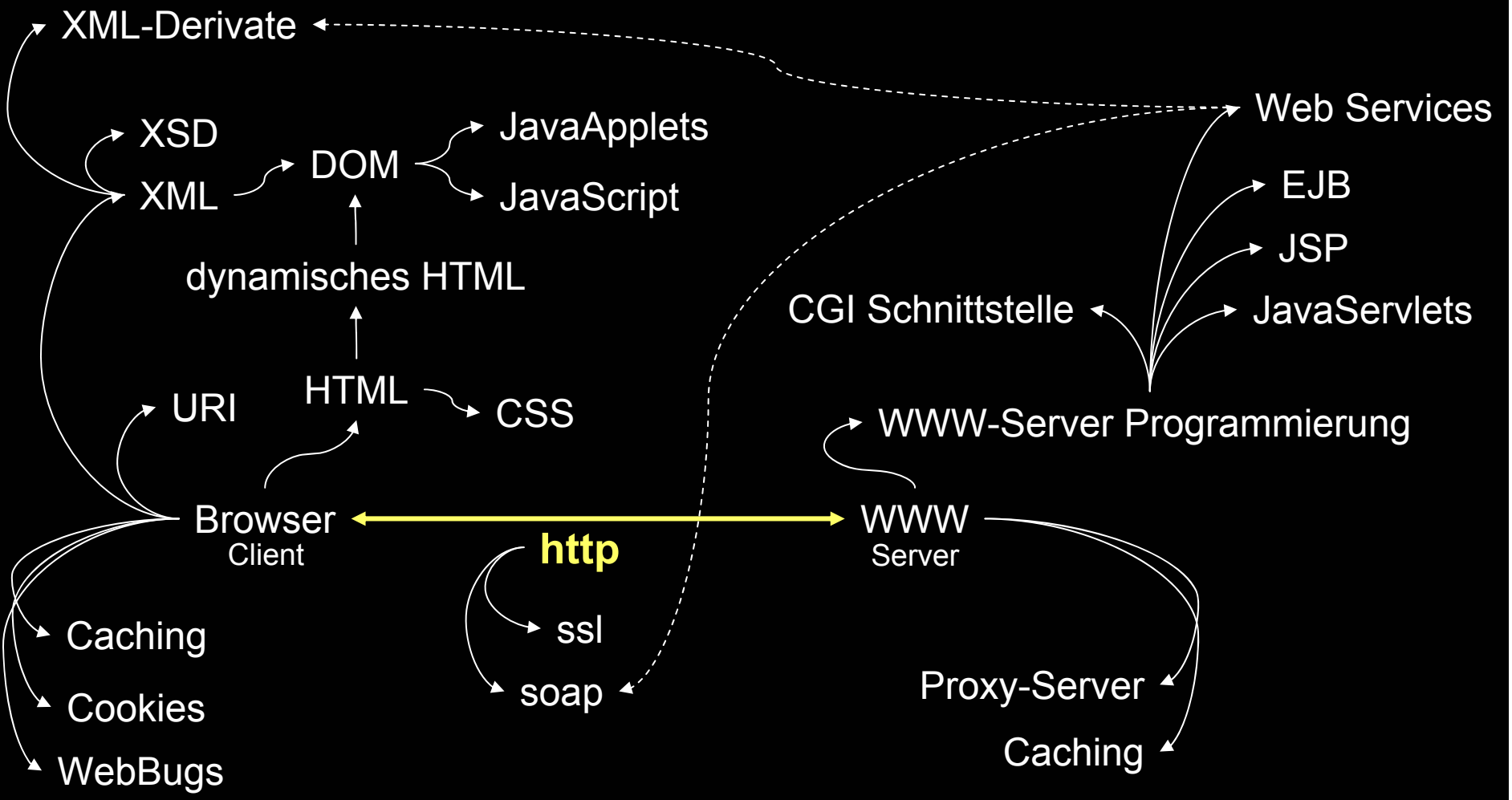
protokoll://[user:[password]@]**hostname**:[port]/**documentname**[?parameter]

2. World Wide Web

Uniform Resource Locator (URL)



2. World Wide Web



2. World Wide Web

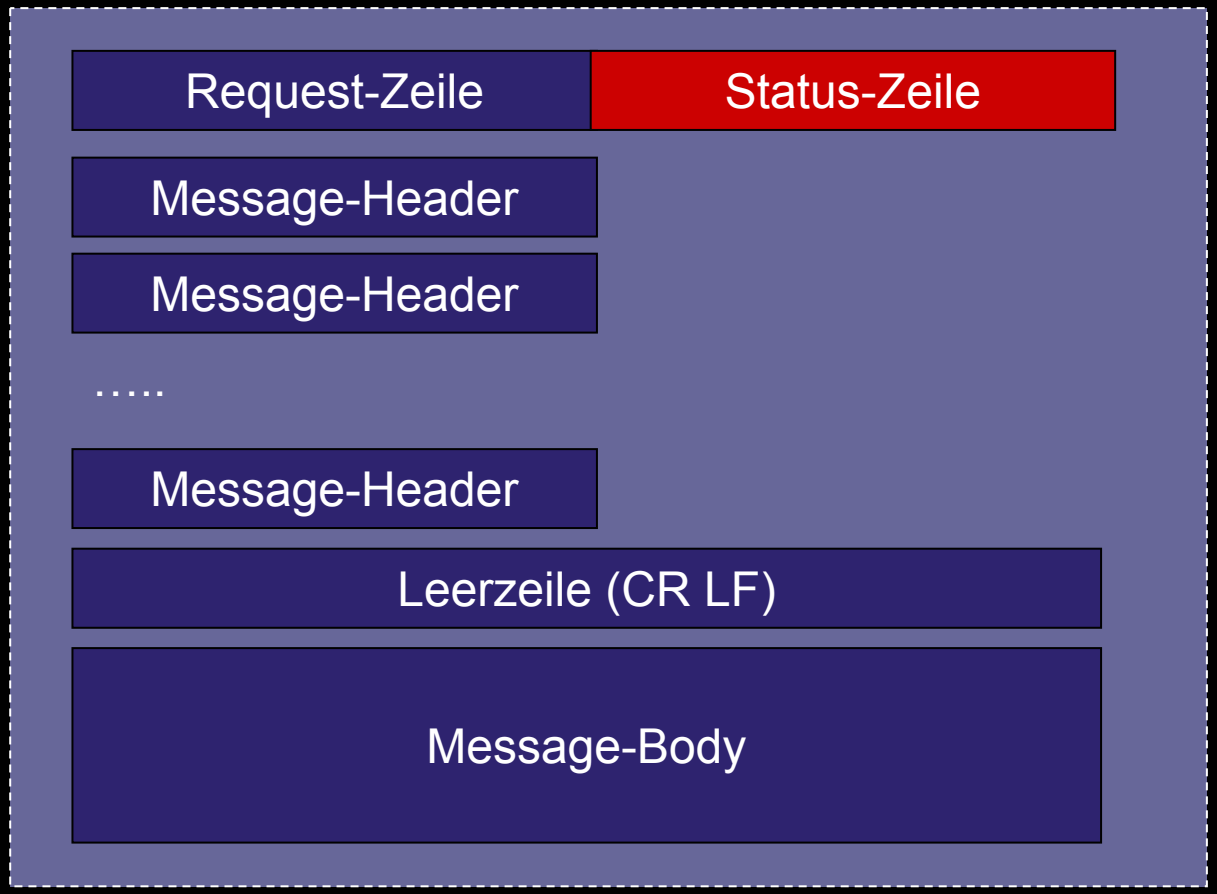
HyperText Transport Protocol

- Die Interaktion zwischen Browser und WWW-Server erfolgt über das **HyperText Transfer Protokoll - HTTP**
  - 1992 – HTTP/0.9
  - 1996 – HTTP/1.0 (RFC 1945)
  - 1997 – HTTP/1.1 (RFC 2068 / 1999 RFC 2616)

→ web-sniffer

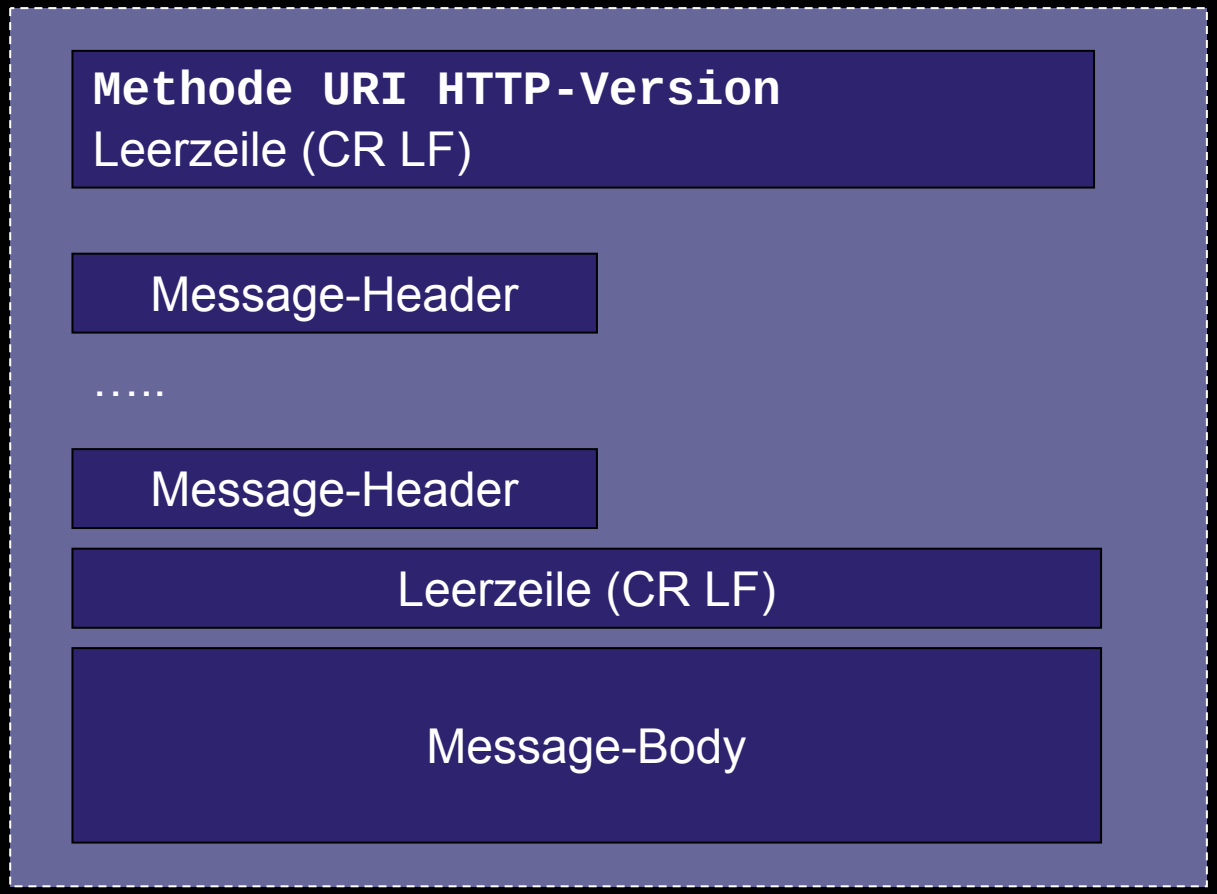
2. World Wide Web

HTTP Nachrichtenformat



2. World Wide Web

HTTP Anfrage (Request)



2. World Wide Web

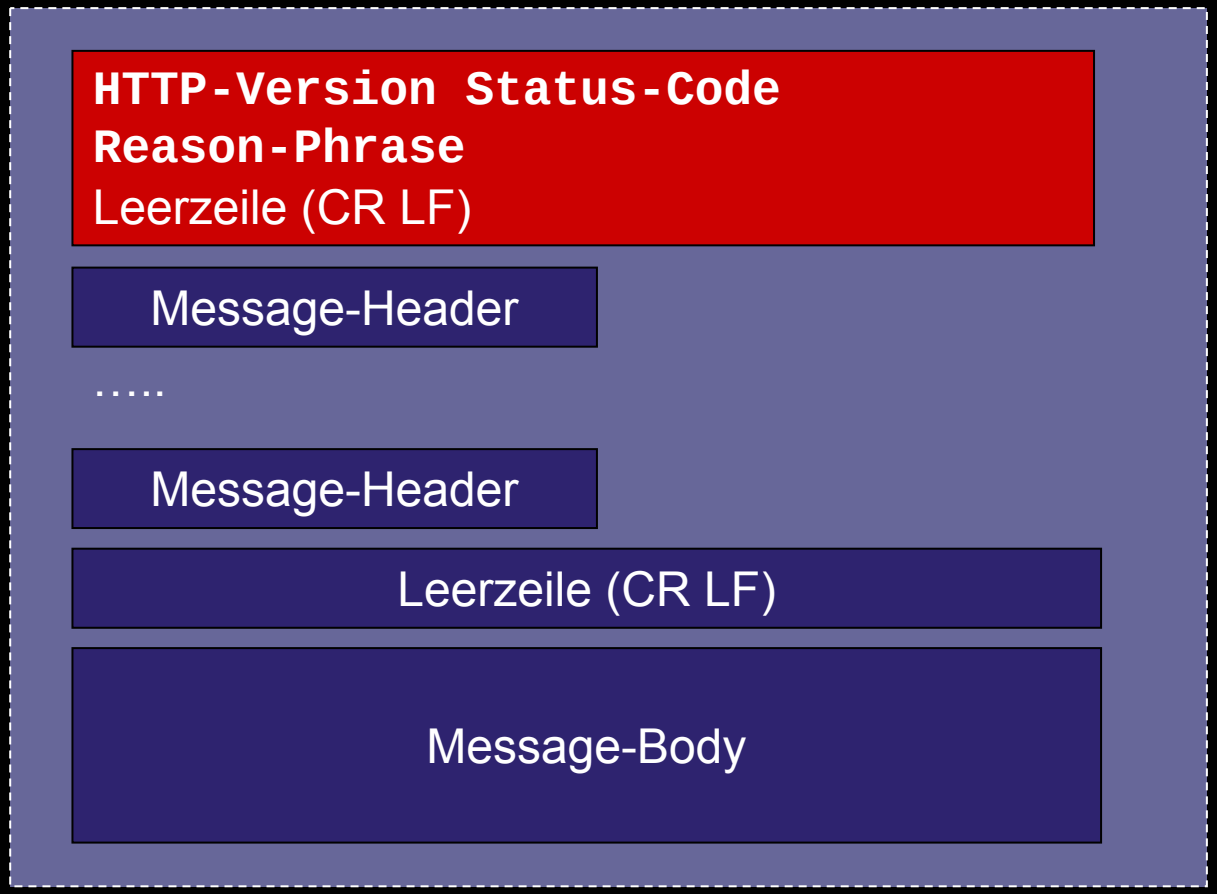
HTTP Anfrage (Request)

- **Methoden**

CONNECT	Verbindungsaufbau zu Proxy-Server
DELETE	Löschen einer Ressource auf dem Server
GET	Anfrage einer durch URI spezifizierten Ressource
HEAD	wie GET, liefert jedoch nur Message Header
OPTIONS	Abfrage von Kommunikationsparametern
POST	Übermittlung von Daten zum in der URI angegegebenen WWW-Server
PUT	wie POST, jedoch wird dabei eine neue Ressource erzeugt
TRACE	Abfrage von Informationen über den Verbindungspfad

2. World Wide Web

HTTP Antwort (Response)



2. World Wide Web

HTTP Status-Codes

- HTTP-Statuscodes dienen zur Kommunikation von Status- oder Fehlermeldungen zwischen Browser und WWW-Server
- Einteilung in Gruppen:

1xx - Informational	vorläufige Antwort, Server hat Anfrage erhalten und bearbeitet diese gerade
2xx - Successful	Anfrage wurde vom Server empfangen, verstanden und akzeptiert
3xx – Redirection	Anfrage konnte nicht (vollständig) vom Server bearbeitet werden, Verweis auf anderen Server
4xx – Client Error	Anfrage konnte nicht bearbeitet werden, da Fehler auf Client-Seite (z.B. falscher URL)
5xx – Server Error	Anfrage konnte nicht bearbeitet werden, da Fehler auf Server-Seite

2. World Wide Web

HTTP Content Negotiation

- angefragte Ressource kann auf Server in unterschiedlichen Varianten vorliegen:
  - sprachspezifische Varianten
  - qualitätspezifische Varianten
  - kodierungsspezifische Varianten
- Browser und WWW-Server verhandeln darüber, welche Variante übertragen wird → **Content Negotiation**

2. World Wide Web

HTTP Content Negotiation

- **Server Driven Content Negotiation**
  - Auswahl der Variante übernimmt der **WWW-Server**
  - Kriterium für die Auswahl sind
    - Informationen in den Headerfeldern des HTTP-Requests
    - Informationen über die am WWW-Server vorliegenden Ressourcen

2. World Wide Web

HTTP Content Negotiation

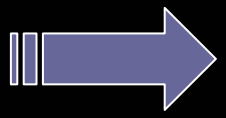
- **Agent Driven Content Negotiation**
  - Auswahl der Variante wird dem **Client (Browser)** überlassen
    - Server liefert auf Client-Anfrage Informationen über die verfügbaren Varianten
    - Client spezifiziert in einer zweiten Anfrage, welche Ressource übertragen werden soll
  - mit Proxy-Server → **Transparent Negotiation**

2. World Wide Web

HTTP Sicherheit

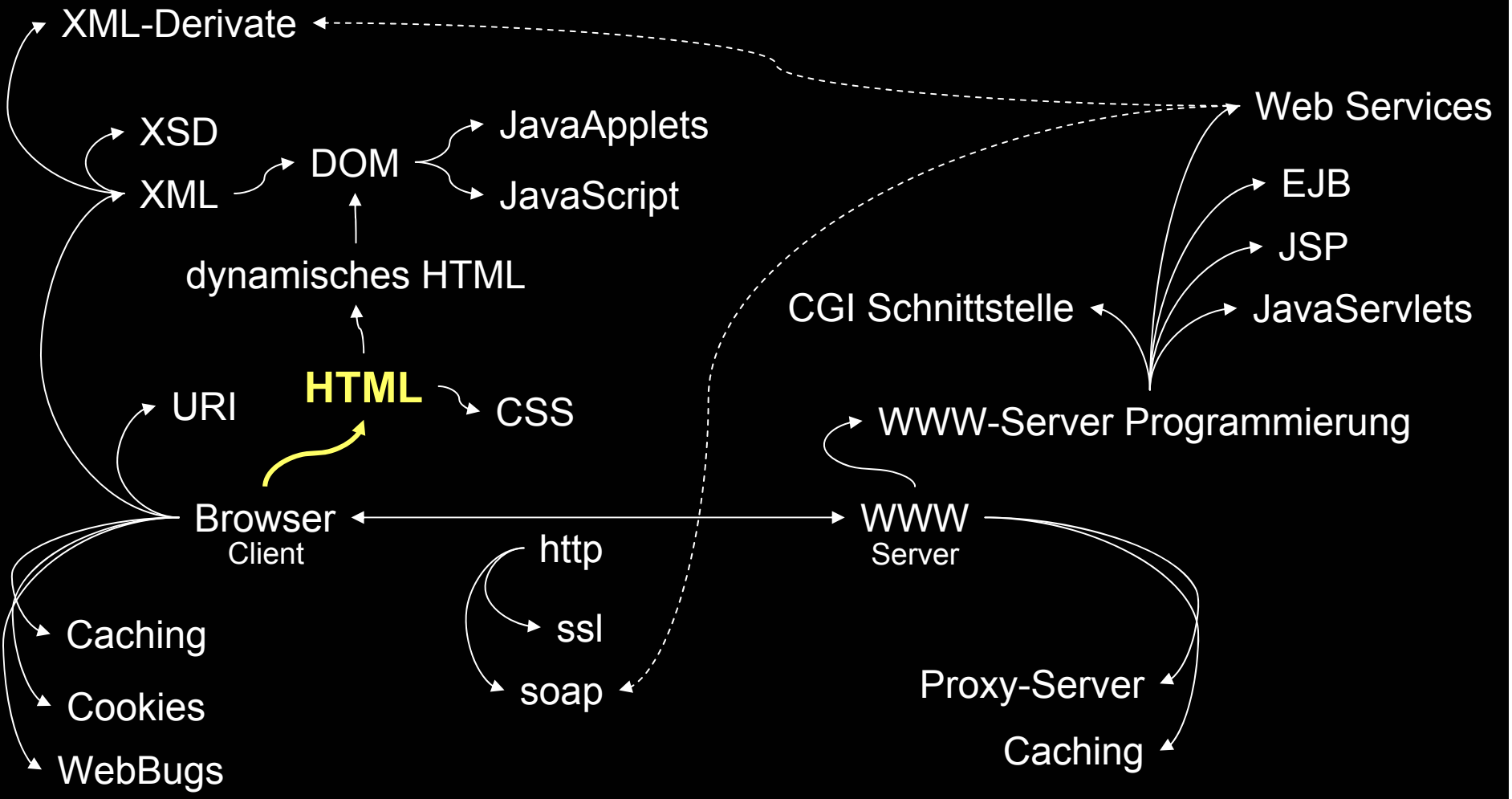
- **Warum notwendig?**
  - Client greift auf sicherheitsrelevante Informationen zu
  - Client übermittelt sicherheitsrelevante Informationen an Server

  - **Identifikation**
    - wer ist der Kommunikationspartner ?
  - **Authentifikation**
    - ist der Kommunikationspartner tatsächlich der, der er vorgibt zu sein?
  - **Autorisation**
    - ist der Kommunikationspartner zum Zugriff befugt?



Basic Authentication
Digest Authentication

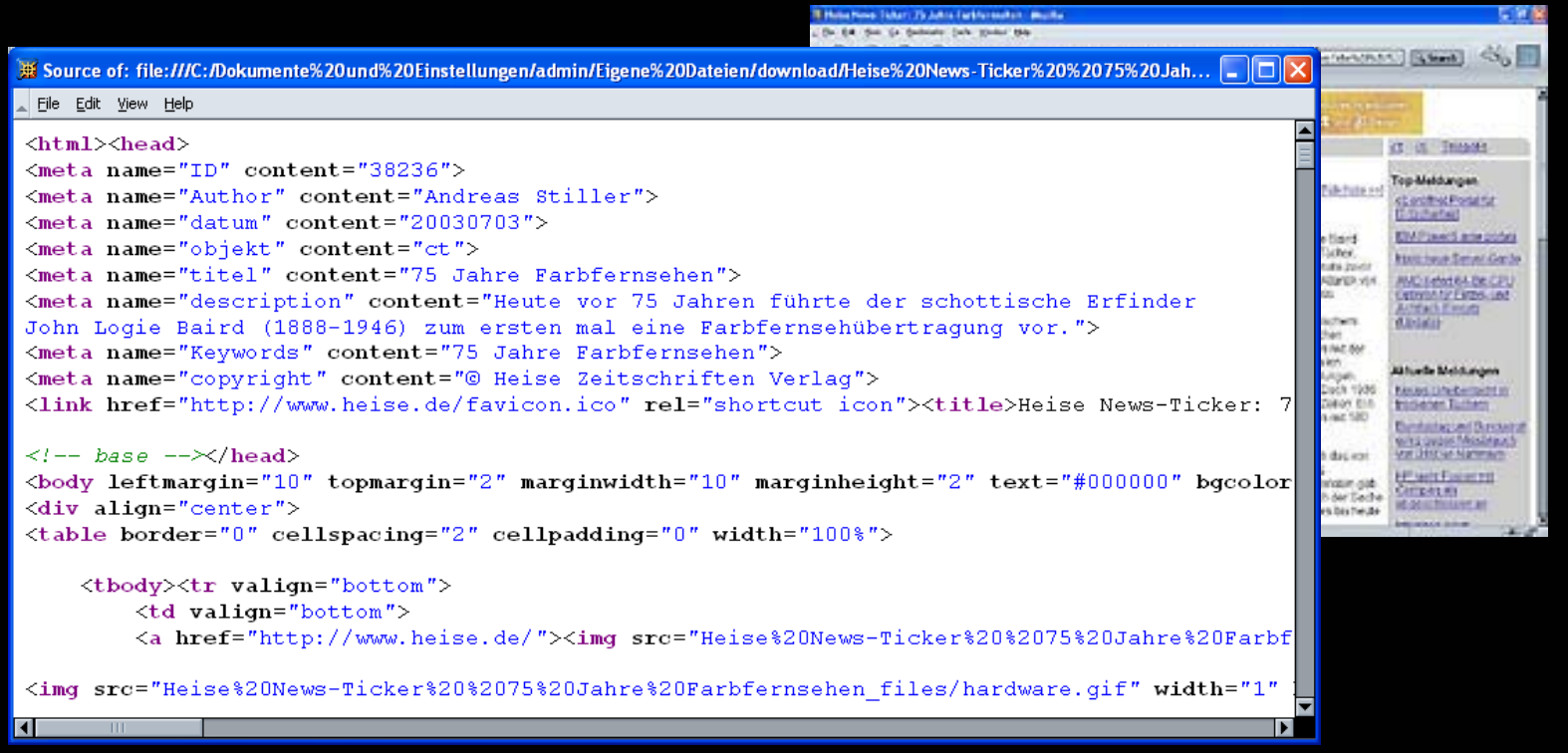
2. World Wide Web



2. World Wide Web

HyperText Markup Language

- Hypermedia-Dokumente werden im WWW in der HyperText Markup Language (HTML) kodiert



2. World Wide Web

HyperText Markup Language

- „**Markup**“ bedeutet **Auszeichnung/Kennzeichnung** und meint damit das Kennzeichnen von bestimmten Abschnitten mit gewissen Attributen oder Anweisungen
- will man einen Textabschnitt **fett** darstellen, müssen Anfang und Ende des Abschnittes mit dem "**Fett-Markup**" gekennzeichnet werden

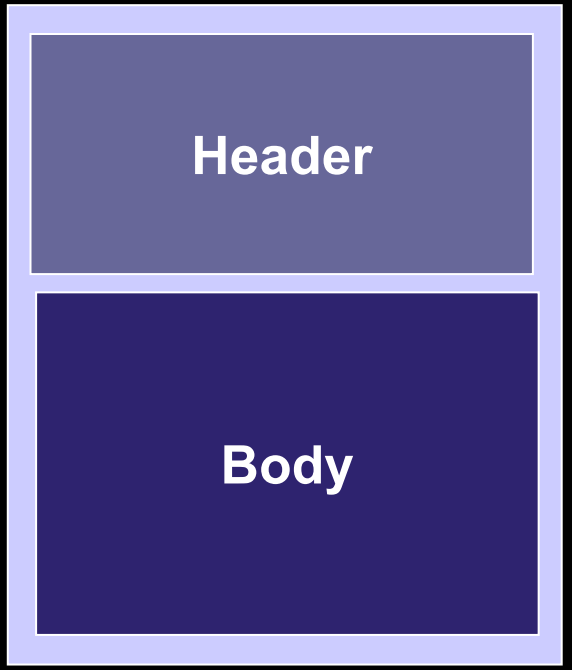
z.B. `<b> fett </b>`

- die einzelnen Markup-Elemente heißen "**Tags**„
- normalerweise enthält Markup **keine ausführlichen Formatierungs-anweisungen**
- es werden nur allgemeine Richtlinien zur **Strukturierung** der Dokumente beschrieben
- Einzelheiten der Formatierung werden dem Browser überlassen

2. World Wide Web

HyperText Markup Language

- HTML-Dokument besteht aus:
  - dem Kopf - "**Header**“
    enthält Informationen über das Dokument, z.B. **Titel**, oder **Stichwörter** über den Dokumenteninhalt usw. und
  - dem Rumpf - "**Body**“
    enthält eigentliche Informationen, die der Autor hier über Markups/Tags mit einer Struktur und Formathinweisen versehen kann
- HTML erlaubt z.B. die Auszeichnung von
  - **Überschriften,**
  - **Listen,**
  - **Grafiken,**
  - **Links …**

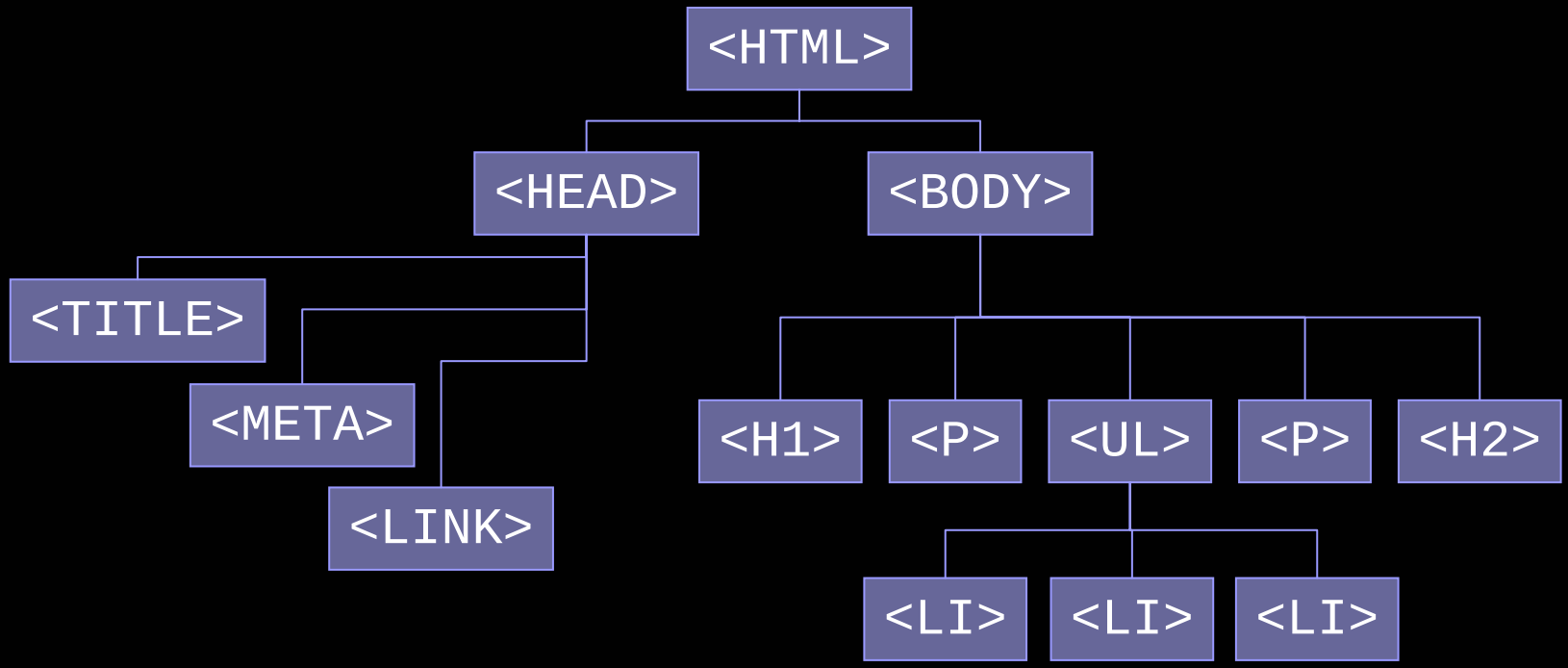


HTML-Dokument

2. World Wide Web

HyperText Markup Language

- **Aufbau eines HTML-Dokuments**

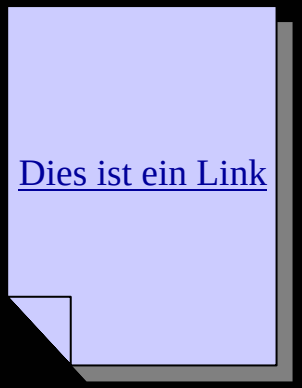


2. World Wide Web

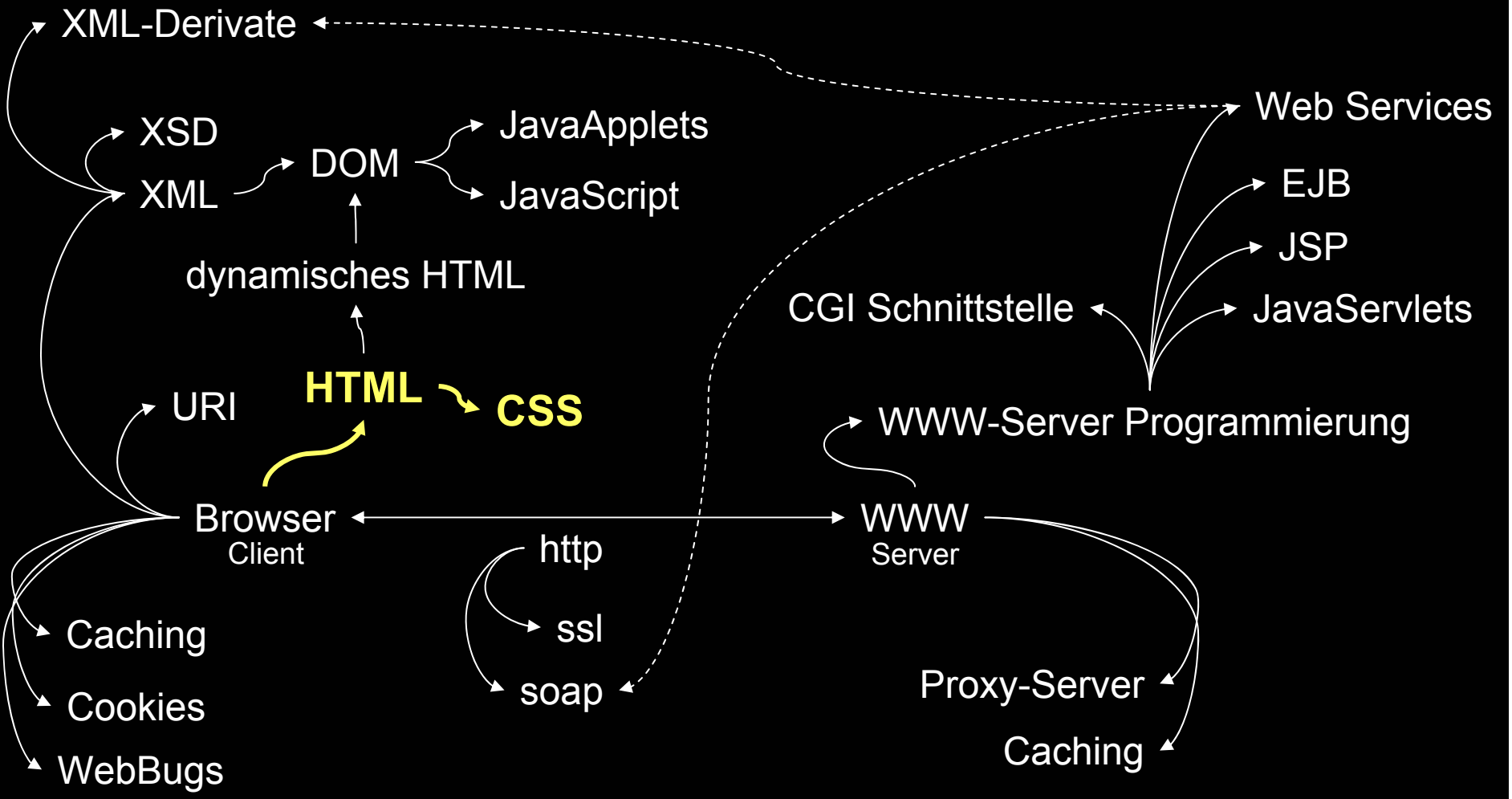
HyperText Markup Language

- **HTML – Hyperlinks**
  - Hypermedia-Referenzen werden im Browser als auswählbares (**anklickbares**) HTML-Element dargestellt
  - **jedes HTML-Element** (Wort, Satz, Absatz, Bild) kann als Hypertext-Referenz verwendet werden
  - das HTML-Element wird dazu zusammen mit dem URL des verlinkten Dokuments von zwei "Ankern" (**Anchor**) - den Tags **<a>** und **** - eingerahmt

```
<a href=„http://www.uni-jena.de/~sack/index.html“>
 Dies ist ein Link
</a>
```



2. World Wide Web

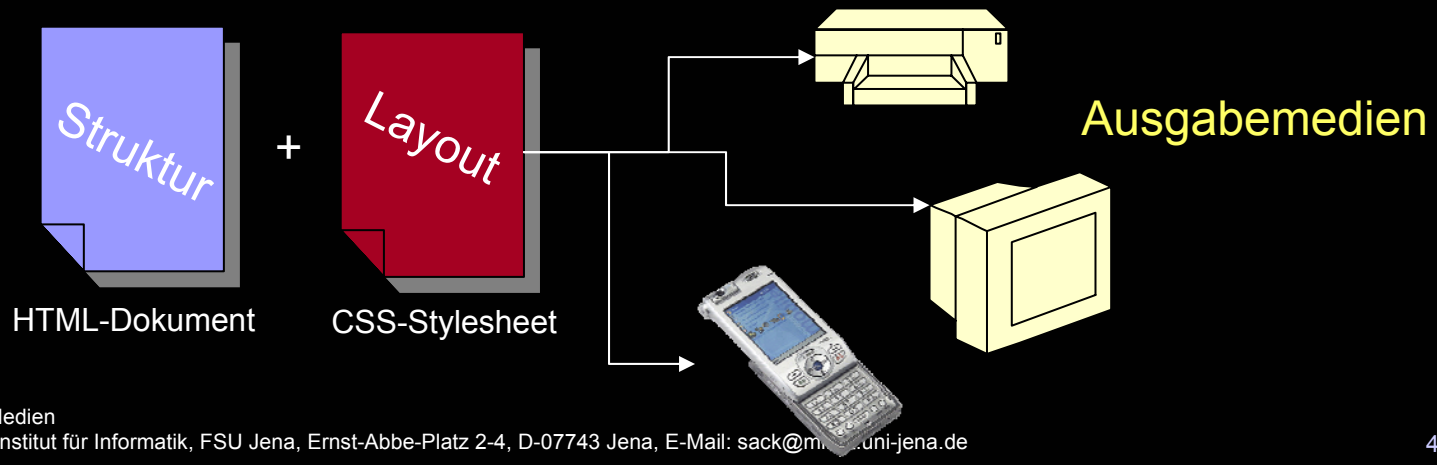


2. World Wide Web

Cascading Style Sheets

- **HTML und Formatierung mit CSS**
  - HTML legt nur die **Dokumentenstruktur** fest (idealerweise)
  - Zur **Formatierung** der Darstellung der einzelnen HTML-Strukturelemente (Überschriften, Listen, Tabellen, etc.) dient eine spezielle Formatierungssprache:

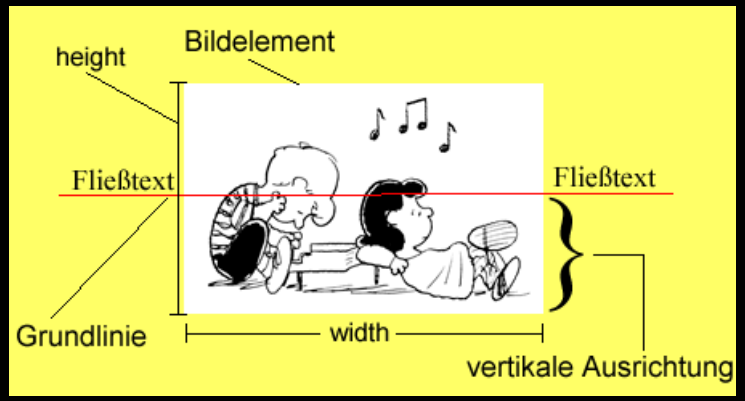
    Cascading Style Sheets (CSS)
  - Das CSS legt das Layout eines HTML-Dokuments für ein bestimmtes Ausgabemedium (Bildschirm, Drucker, etc.) fest.

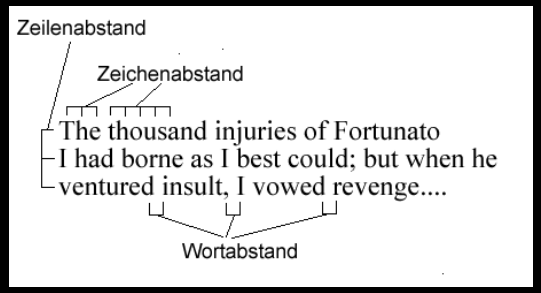


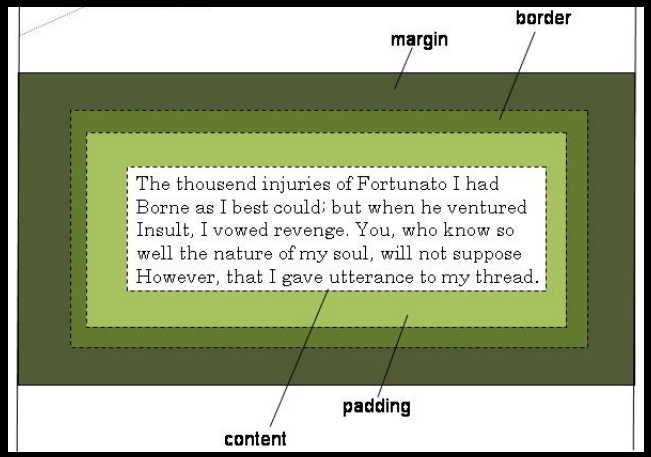
2. World Wide Web

Cascading Style Sheets

- CSS-Stylesheets bestimmen
  - Schriftattribute
  - Bemaßungen
  - Objektausrichtung
  - …





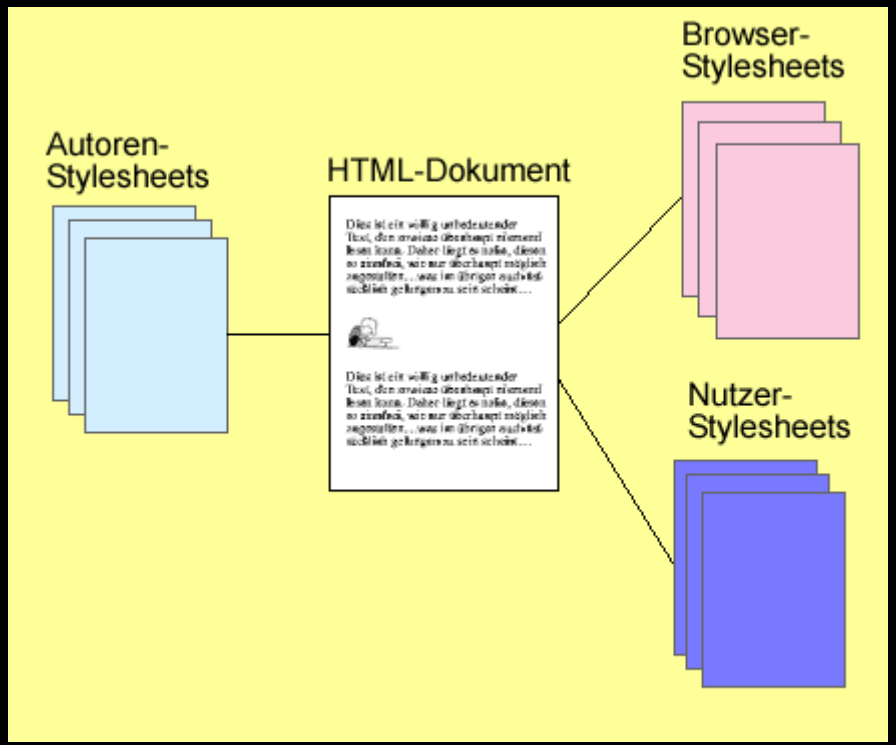


2. World Wide Web

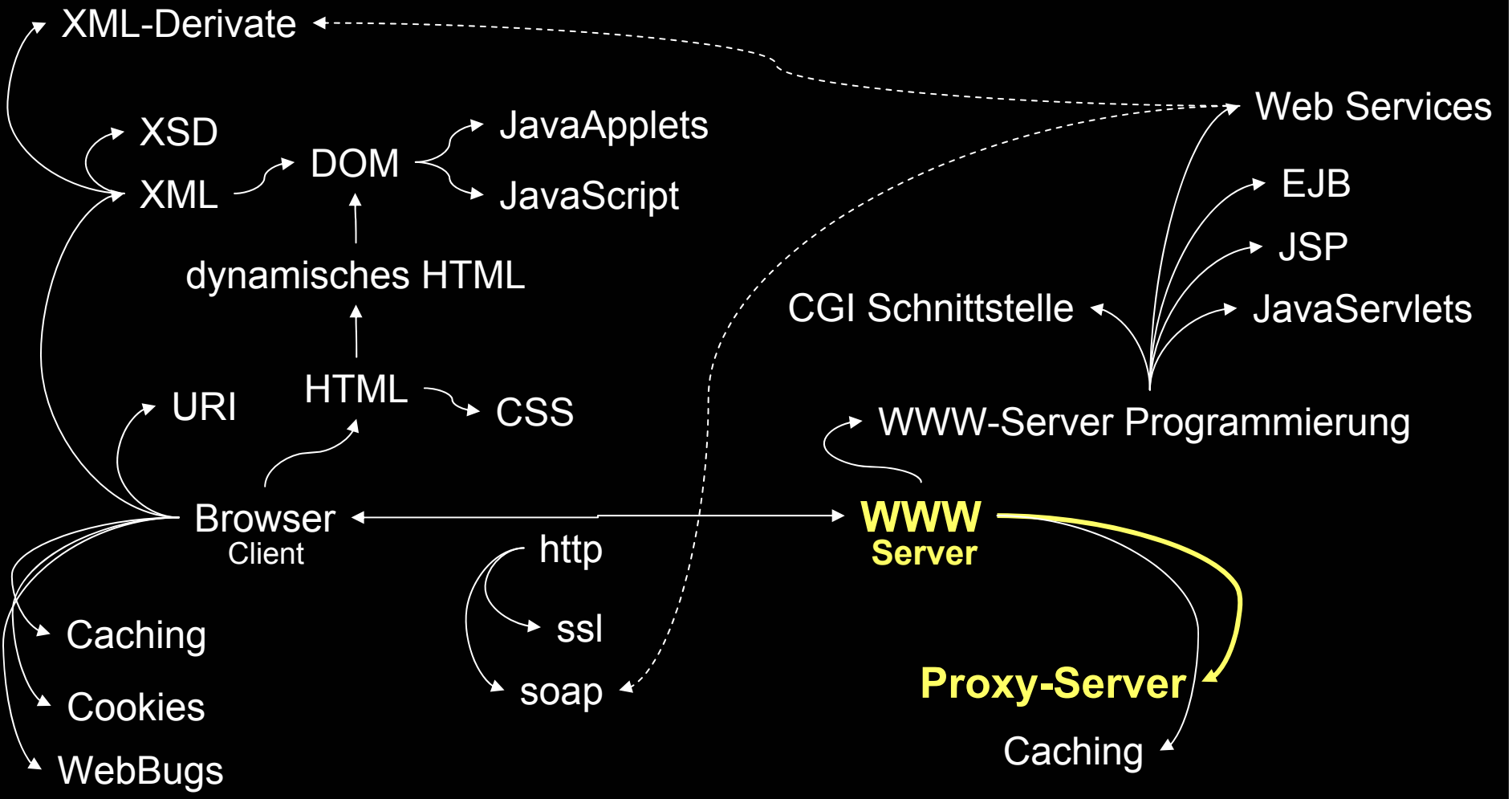
Cascading Style Sheets

- Für ein HTML-Dokument können **verschiedene, konkurrierende CSS-Stylesheets** angegeben werden

- unterscheide
  - **Autoren**stylesheet
  - **Browser**stylesheet
  - **Nutzer**stylesheet



2. World Wide Web

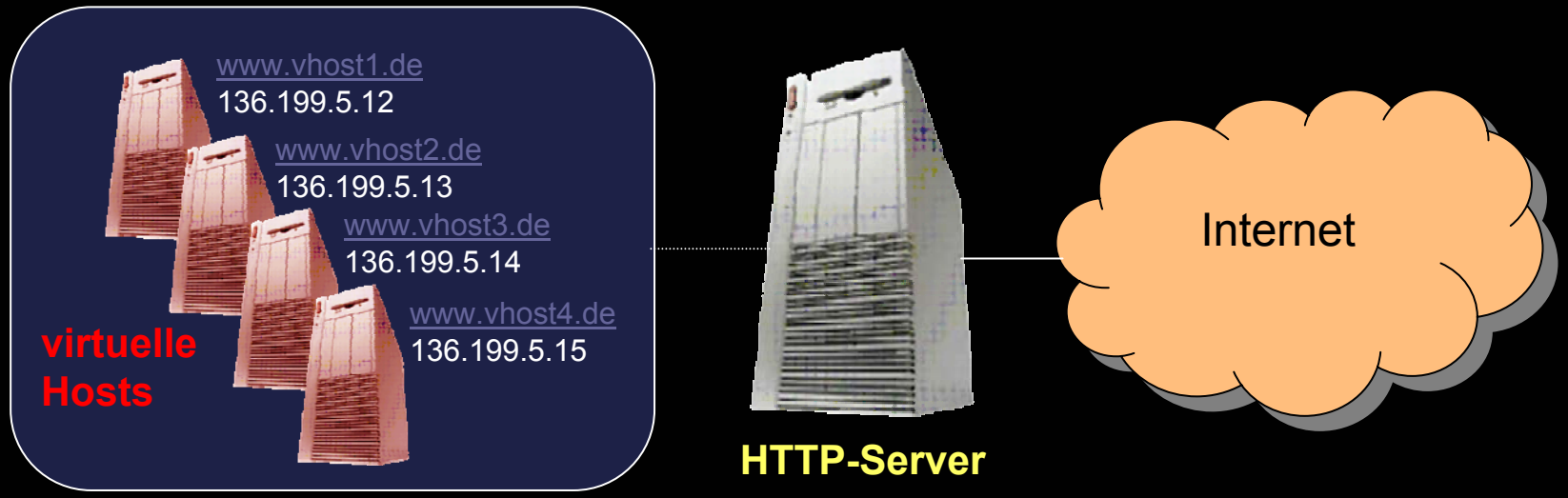


2. World Wide Web

WWW-Server

- Server-Prozess überwacht standardgemäß Port 80
- Server kann mehrere virtuelle Hosts gleichzeitig betreiben

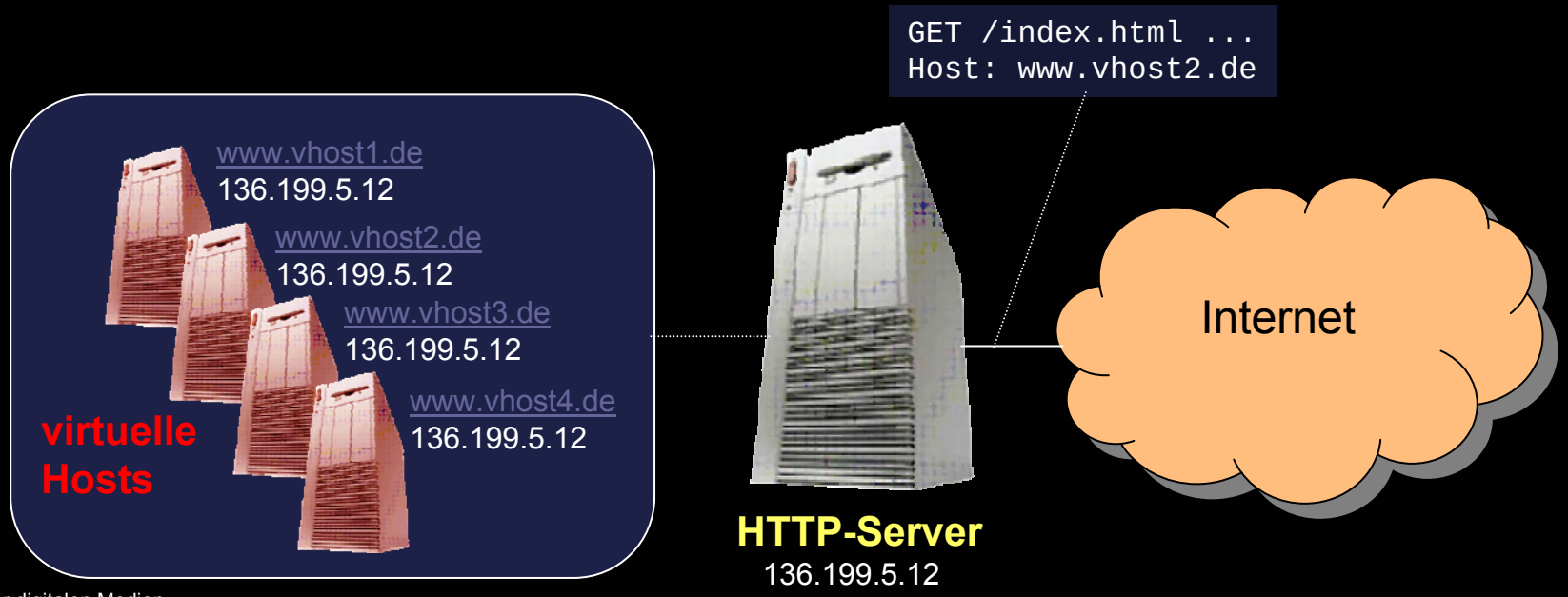
- **IP-basierte Hosts**
  - Jeder Host verfügt über eigene IP-Adresse und DNS-Eintrag



2. World Wide Web

WWW-Server

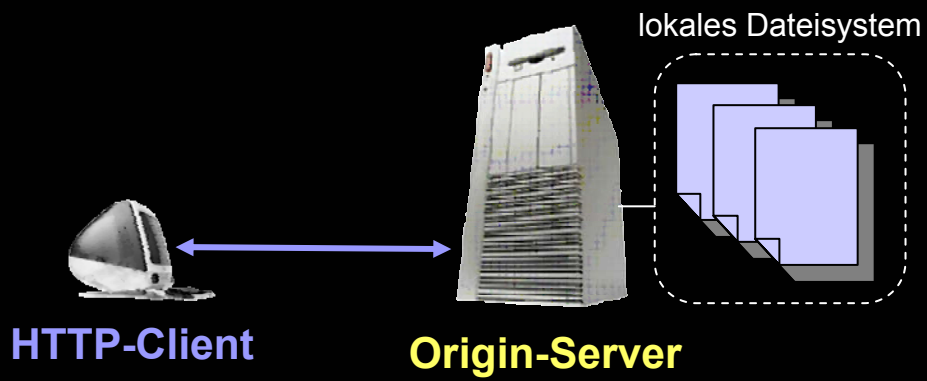
- **Nicht-IP-basierte Hosts**
  - Alle Hosts haben dieselbe IP-Adresse, DNS-Einträge zeigen alle auf dieselbe IP-Adresse
  - Host-Feld des HTTP/1.1-Headers dient zur Identifikation eines bestimmten virtuellen Hosts



2. World Wide Web

Origin-Server

- WWW-Server beantwortet Anfrage **selbst** und leitet diese nicht weiter
  - Ressourcen aus dem **lokalen Dateisystem** des Servers müssen auf **URLs** abgebildet werden



URL ⟷ Pfadangabe im lokalen Dateisystem

2. World Wide Web

Origin-Server

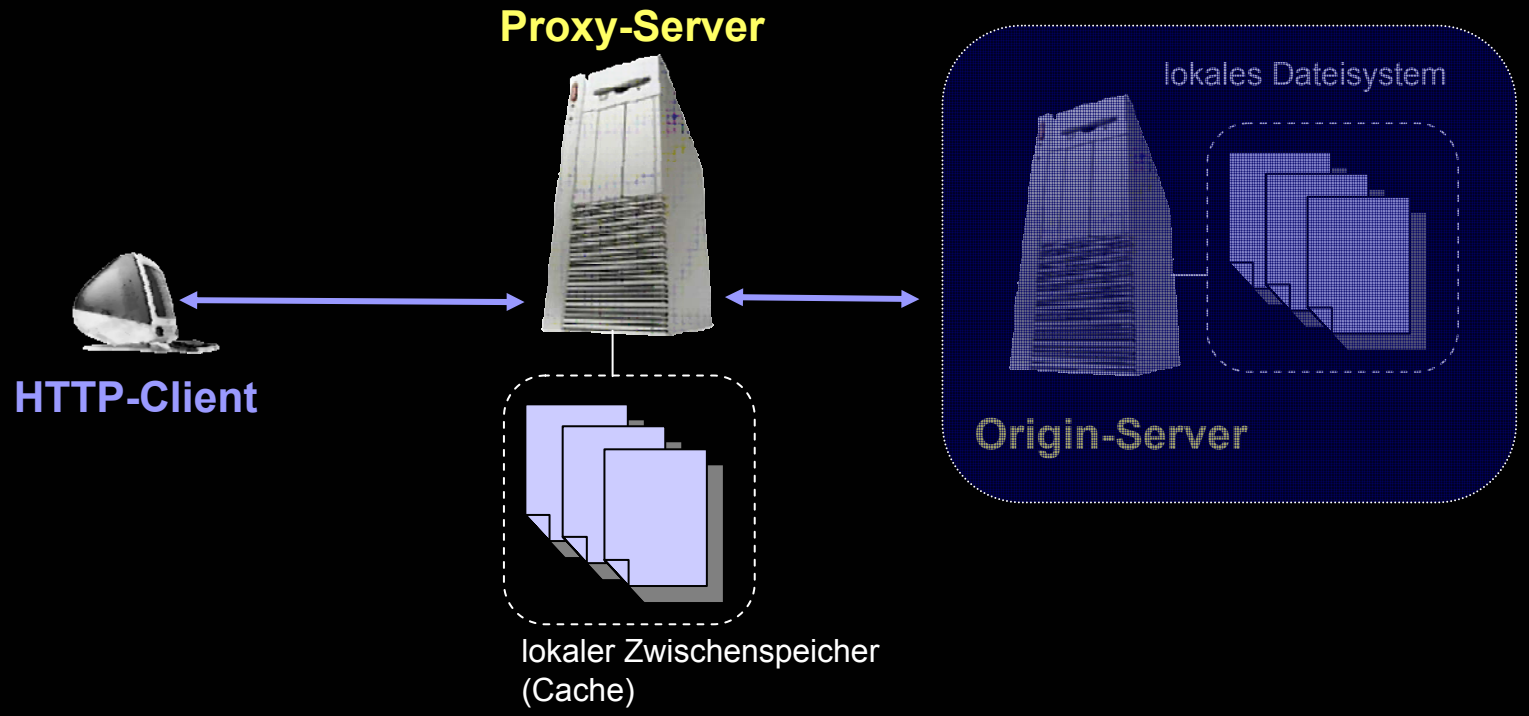
- Ressourcen aus dem **lokalen Dateisystem** des Servers müssen auf **URLs** abgebildet werden
  - statische Ressourcen
  - dynamische Ressourcen
  - nutzerbezogene Resourcen

- **Fehlerbehandlung:**
  - Fehlerhafter URL: 404 – not found
  - Veralteter URL: 410 – gone
    301 – moved permanently (Redirect)
    302 – moved temporarily (Redirect)

2. World Wide Web

Proxy-Server

- WWW-Server beantwortet Anfrage **nicht selbst**, sondern
  - **leitet** die Anfrage des Clients **nur weiter** und
  - **speichert** die Antwort des Origin-Servers zwischenzeitlich



2. World Wide Web

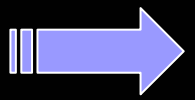
Proxy-Server

- **Kritisch**: Konfiguration des lokalen Zwischenspeichers
  - Größe des Cache
  - Platzierung des Cache
  - Caching Strategien
    - Lebensdauer ?
    - Refresh ?
    - etc.
  - Handhabung sicherheitsrelevanter Daten
    - Zwischenspeicherung ?
    - Verschlüsselung oder Klartext ?
  - Proxy-Server als **Firewall**

2. World Wide Web

Logfiles

- Aufzeichnung von
  - Kontroll- und Statusinformationen
  - Kommunikation des WWW-Servers mit dem Internet
- Auskunft über
  - Zugegriffene Ressourcen
  - Popularität/Effizienz von Ressourcen
  - Clickstream der User
  - Automatische Auswertung erfordert einheitliches Format



Usability

- Statistiken
- Kennzahlen
- Leistungsziffern

Common-Logfile-Format (CLF)

2. World Wide Web

Logfiles – Comon Logfile Format (CLF)

```
remotehost rname authuser [date] “request“ status bytes
```

Erweiterungen (**Expanded CLF**)

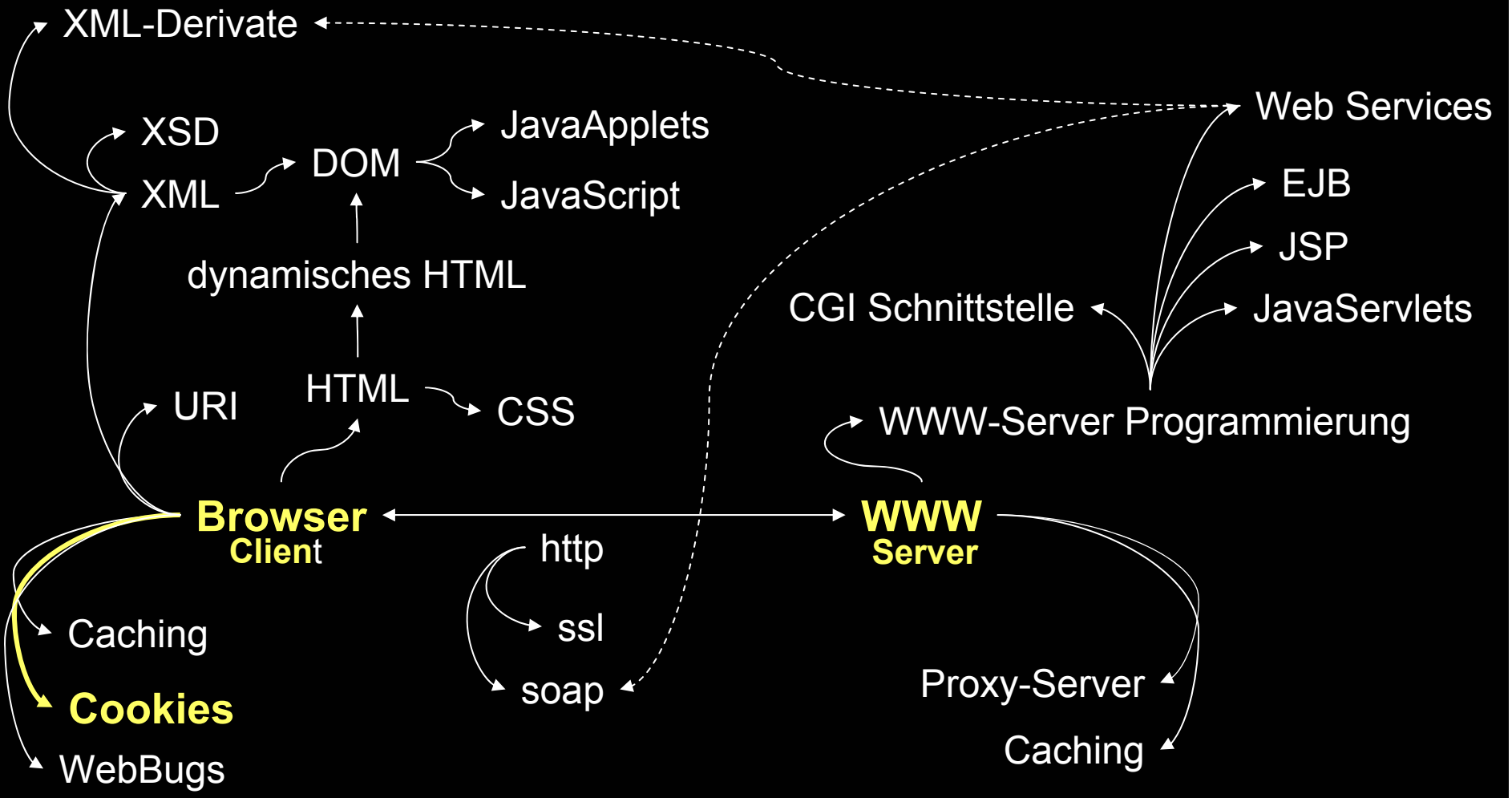
- Referrer
- Agent

HTTP-AccessLog

```
62.224.213.53 - - [17/Jun/2004:02:30:44 +0200]
"GET /~sack/WS0304/seminar/arbeiten/schwartze/ausarb.pdf HTTP/1.1"
200 65536 "-" "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:0.9.8) Gecko/20020204„

62.224.213.53 - - [17/Jun/2004:02:32:57 +0200]
"GET /~sack/WS0304/seminar/arbeiten/schwartze/ausarb.pdf HTTP/1.1"
206 605231 "-" "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:0.9.8) Gecko/20020204„
…
```

2. World Wide Web



2. World Wide Web

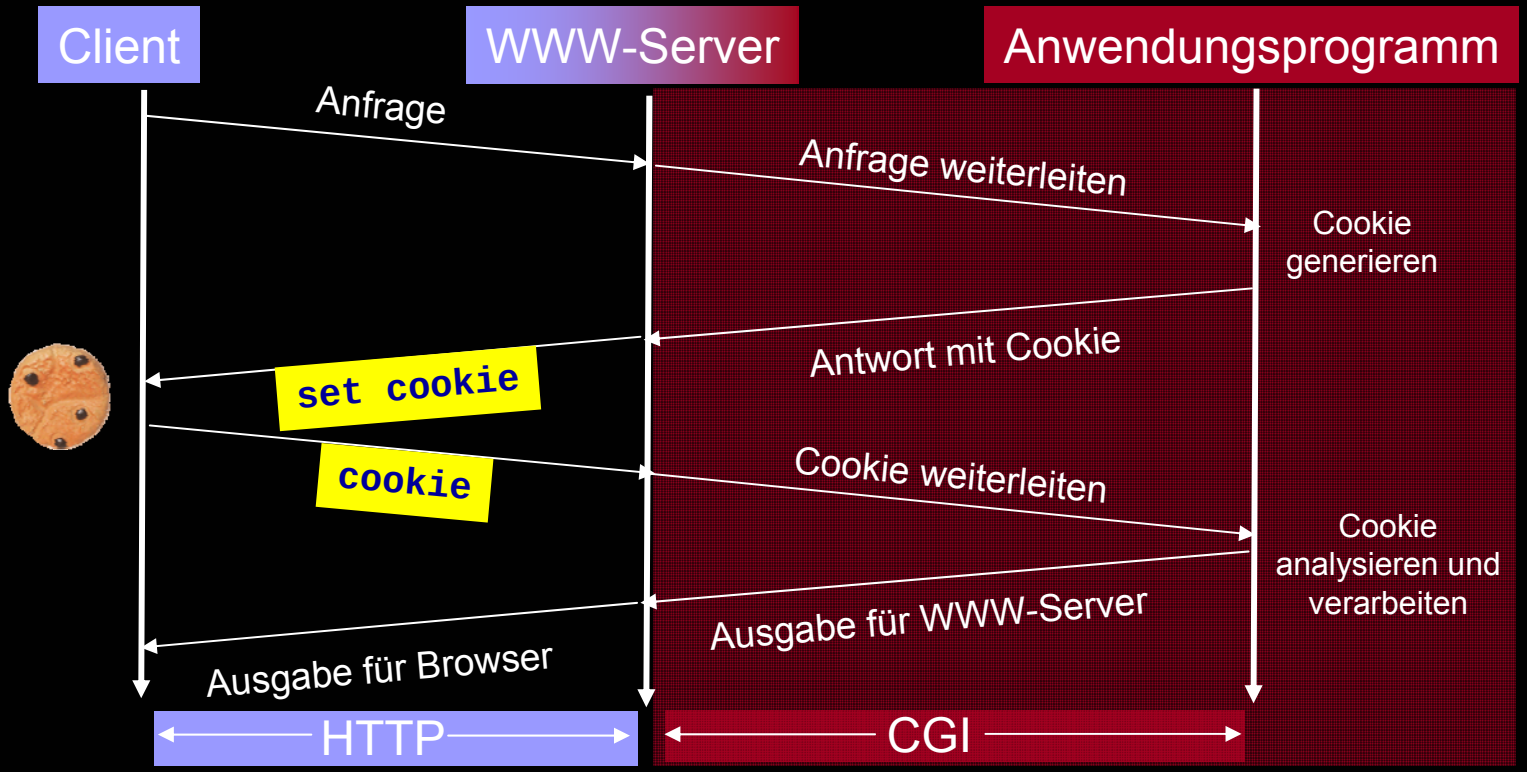
HTTP – Cookies

- HTTP ist ein **zustandsloses Protokoll**
  - über früher stattgefundene Transaktionen hat HTTP keine Information
- Informationen über **früher stattgefundene Transaktionen** sind aber wichtig z.B. für Shopping-Portale (vgl. Warenkorb)
- diese Information muss daher bei Client und Anwendungsprogramm abgespeichert werden
- die beim Client gespeicherte Information wird als **Cookie** bezeichnet

2. World Wide Web

HTTP – Cookies

- Ablauf



2. World Wide Web

HTTP – Cookies

RFC 2109

- Was kann man mit Cookies alles machen?
  - **Identitätsüberprüfung** des Nutzers (Session-ID)
  - Feststellen, ob der Nutzer den WWW-Server bereits schon einmal besucht hat
  - Erstellen von **Kunden-** und **Nutzer-Profilen**
  - Führen von **Warenkörben**
- Was genau ist ein Cookie?
  - **Zeichenkette** (<4KB), die zwischen Client und WWW-Server kommuniziert wird
  - zum Austausch von Informationen, die nicht durch das HTTP-Protokoll ausgetauscht werden können
  - **Session Cookies** und **Persistent Cookies**

2. World Wide Web

HTTP – Cookies

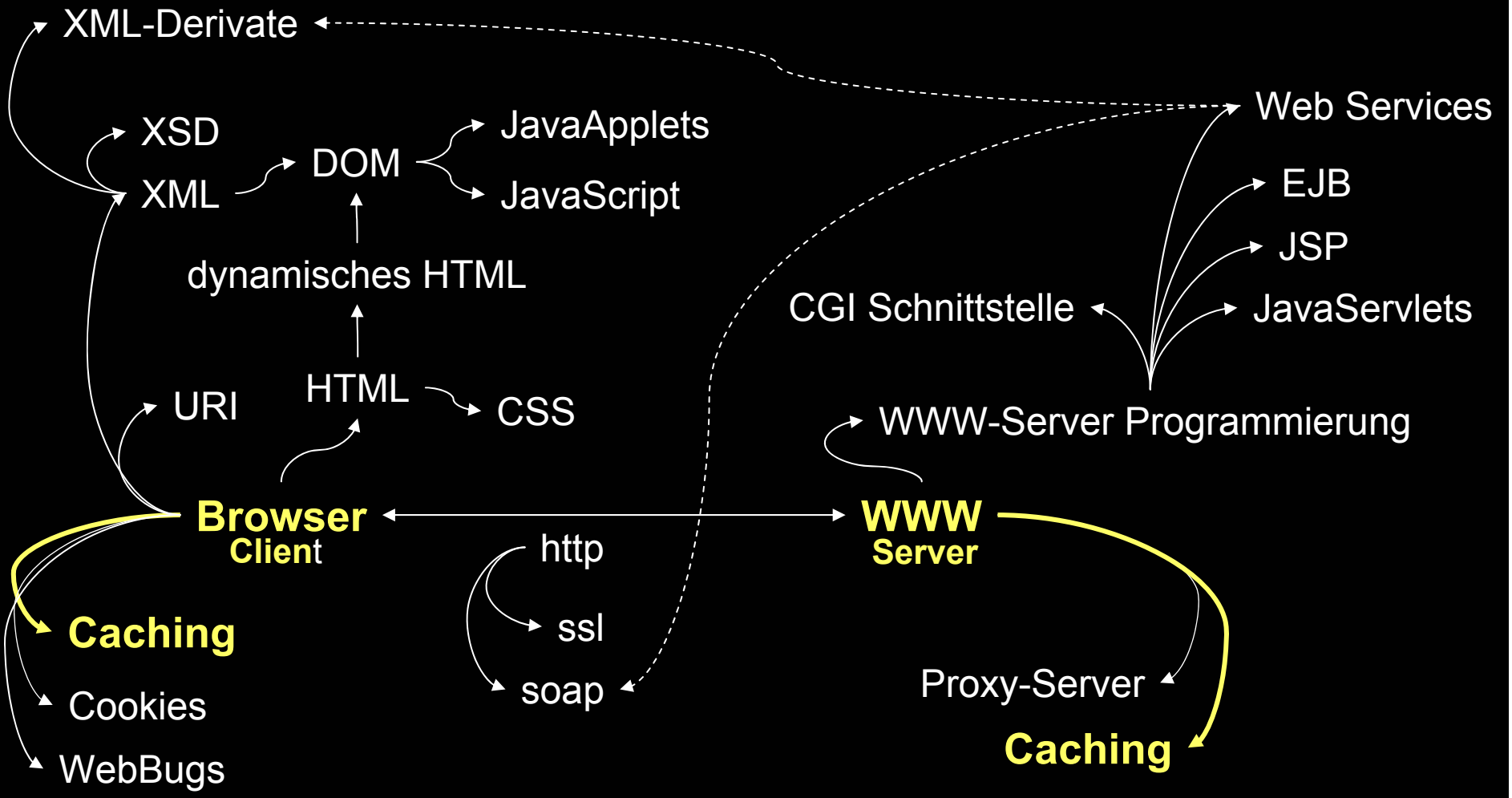
- **Cookie – Parameter:**
  - **`Comment`** Verwendungszweck (optional)
  - **`Domain`** Gültigkeitsbereich (optional)
  - **`Max-Age`** Cookie-Lebensdauer (optional)
  - **`Path`** Einschränkung auf bestimmte Ressourcen (optional)
  - **`Secure`** oft zur sicheren Übertragung notwendiger Schlüssel (nicht genau spezifiziert, optional)
  - **`Version`**

2. World Wide Web

HTTP – Cookies

- Sind Cookies ein **Sicherheitsrisiko**?
  - **WerbeBanner** von Marketingfirmen verfolgen Nutzer über mehrere WebSites hinweg…
  - Sperren von **persistenten Cookies** bzw. nur nach Rückfrage
  - **Session Cookies** erlauben
  - Selektives Sperren von einzelnen Cookies
  - **Lebensdauer** von Cookies beschränken (Mozilla) bzw. persistente Cookies öfters „aufräumen“

2. World Wide Web



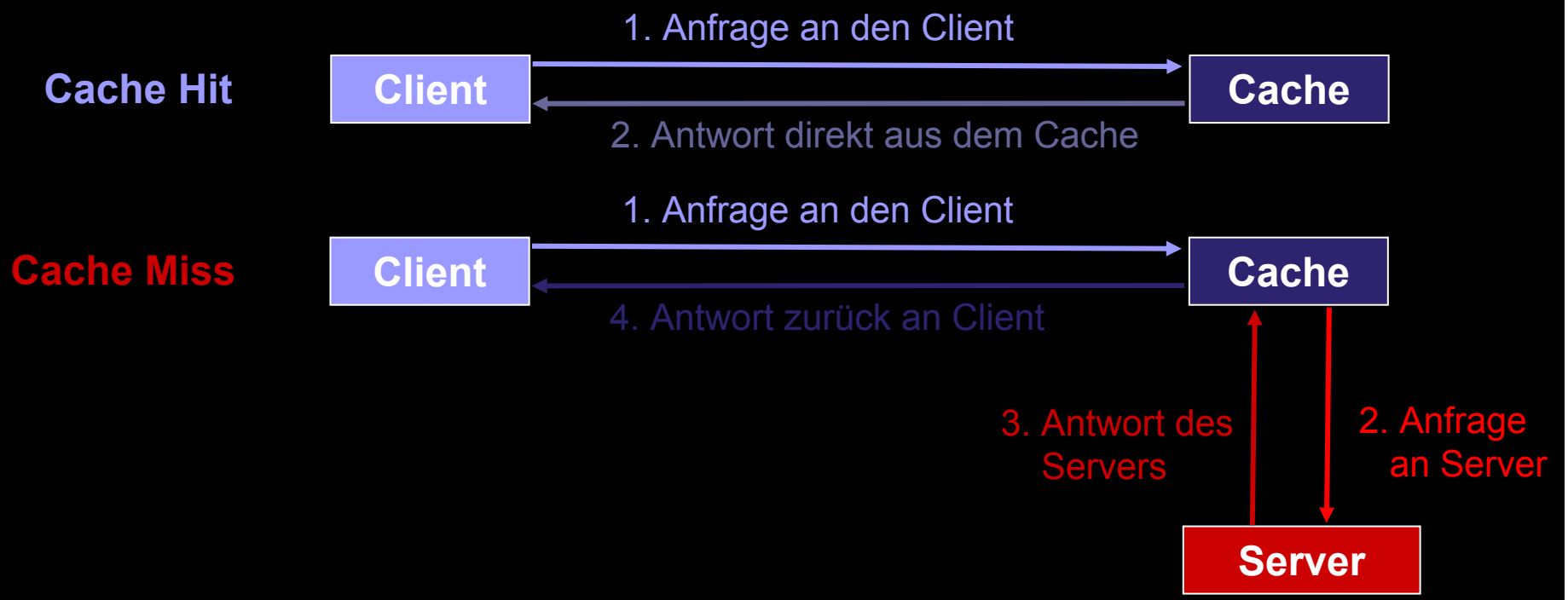
2. World Wide Web

HTTP Caching

- Um unnötige Datenübertragungen zu vermeiden, verfügt der Browser über einen **intelligenten Zwischenspeicher** (**Cache**) für bereits empfangene Dokumente
- Ist ein anzuforderndes Dokument bereits im Cache des Browsers vorhanden, kann dieses direkt aus dem Cache bezogen werden, **ohne den Server zu kontaktieren**
- Cache-Speicher können **unterschiedlich** realisiert werden:
  - auf **Client-Seite**
  - **eigenständig**
  - auf **Server-Seite**
  - mit unterschiedlichen **Caching-Strategien**

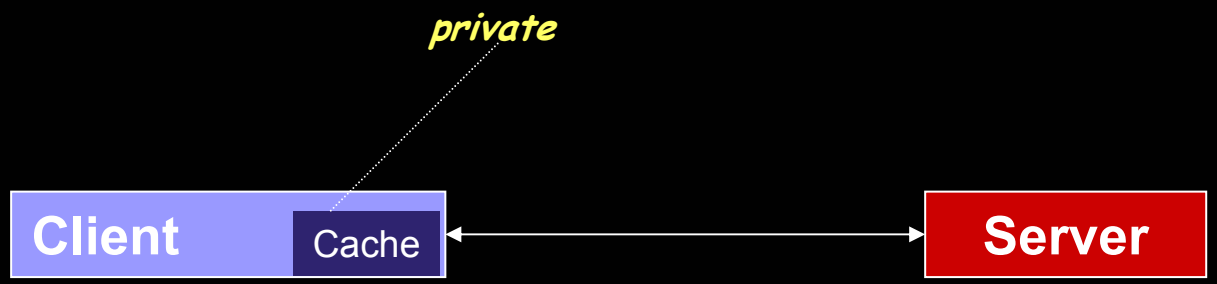
2. World Wide Web

Cache Operationen



2. World Wide Web

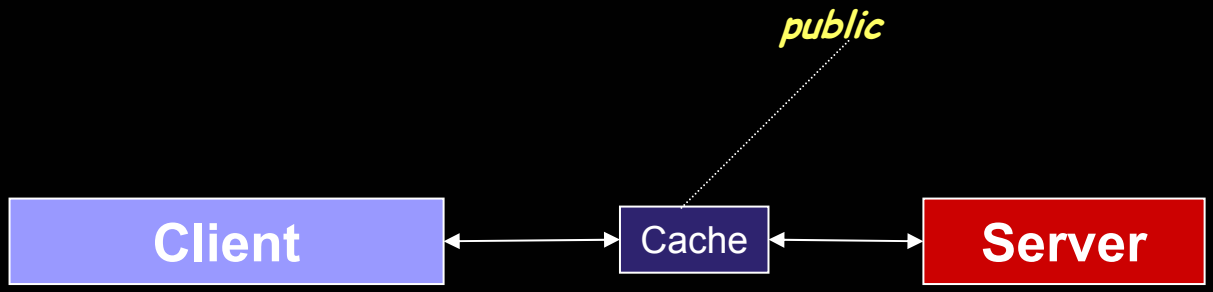
Clientseitiger Cache



- Cache liegt beim Client
- speichert jedes übertragene Dokument
- →History-Funktion (Zurück) des Browsers

2. World Wide Web

Eigenständiger Cache



- Cache ist **zwischen** Client und Server platziert
  z.B. auf Proxy-Server
- kann (optional) mit Client sogar gemeinsam denselben Rechner nutzen, ist aber logisch eigenständig
- erfordert spezielle Client-Konfiguration
- Cache-Hierarchie möglich
  (Weiterleitung an den nächsten Cache)

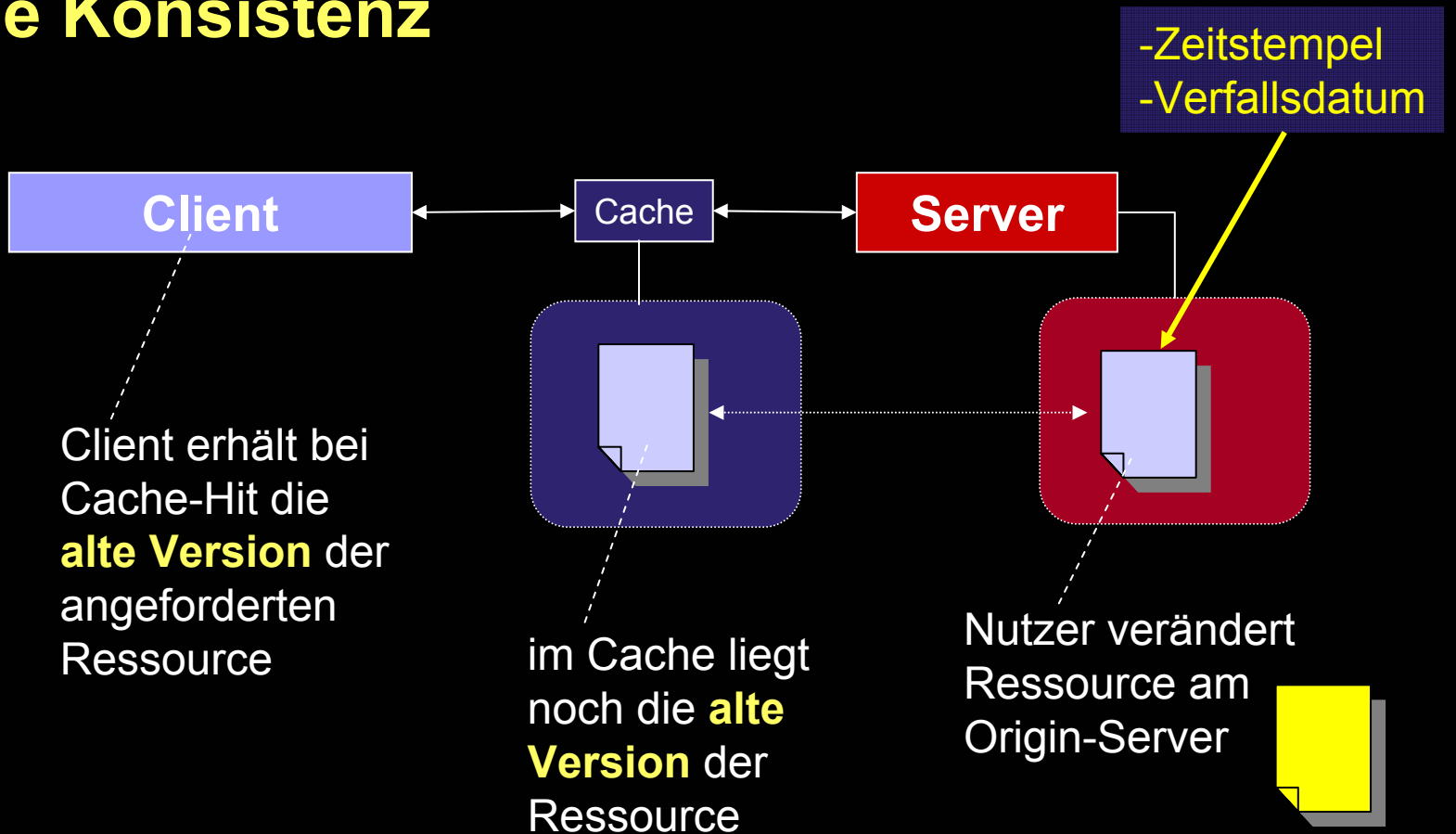
2. World Wide Web

Serverseitiger Cache

- Cache ist direkt beim Server platziert
- **Wann sinnvoll?**
  - speichert Antworten, die der Server erst auf Anfrage hin aufwändig berechnen muss bzw.
  - die von Clients besonders häufig angefragt werden

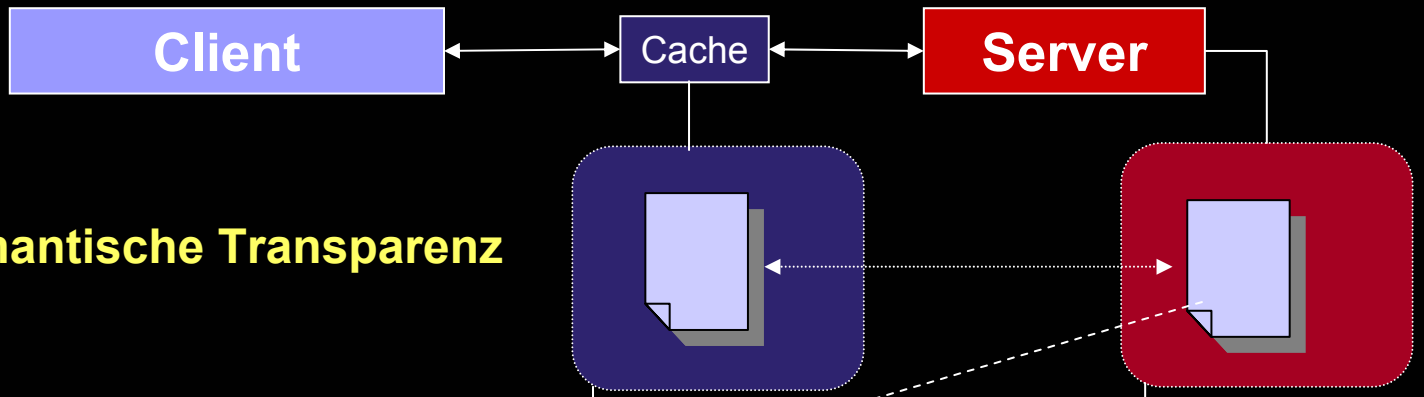
2. World Wide Web

Cache Konsistenz



2. World Wide Web

HTTP/1.0 Caching



semantische Transparenz

```
<html>
<head>
  <meta http-equiv="expires"
        content="Sat, 15 Dec 2005 12:00:00 GMT ">
  <!-- ...  -->
</head>
<body>
…
```

HTML-Dokument

Expires
No-Cache
(Cache Busting)

Last-Modified
If-Modified-Since

2. World Wide Web

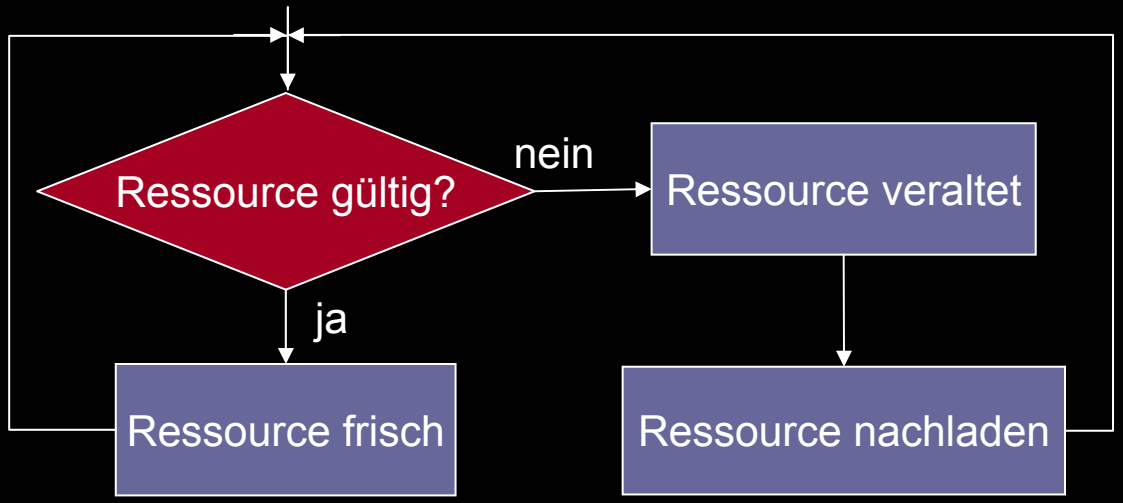
HTTP/1.1 Caching

- **Hauptziele:**
  - Nutzer kann Cache-Verhalten detailliert kontrollieren (**Direktiven**)
  - Bestimmte **Mindestanforderungen** werden garantiert
  - Kann eine Mindestanforderung nicht erfüllt werden, wird der Nutzer darüber **informiert**

  - Cache überwacht aktiv Gültigkeit die gespeicherten Ressourcen
    ⇒ **Cache-Expiration-Model**
  - Cache überprüft Gültigkeit der Ressourcen beim Origin-Server
    ⇒ **Cache-Validation-Model**
  - Cache überwacht die Einhaltung der Nutzerdirektiven
    ⇒ **Cache-Control**

2. World Wide Web

HTTP/1.1 Cache Expiration Modell



Server Specified Expiration

- Headerfeld `Expires`
- Cache-Control `max-age`
  `must-revalidate`

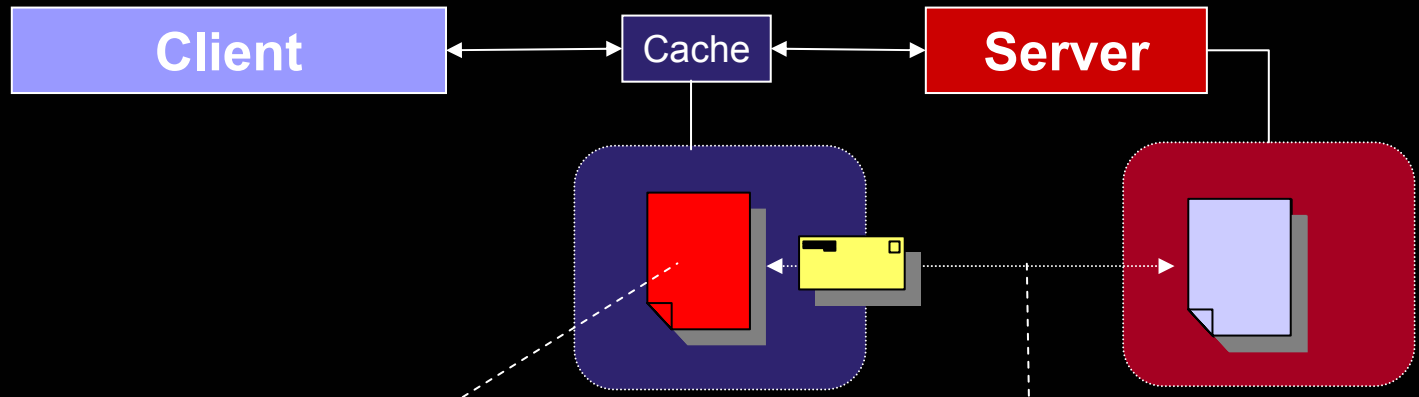
Heuristic Expiration

- Headerfeld `Last Modified`
  `Age`
  `Date`

2. World Wide Web

HTTP/1.1 Cache Validierung

EntityTag (ETag)
zur eindeutigen Identifikation der betreffenden Ressource
→ HTTP `if-match`



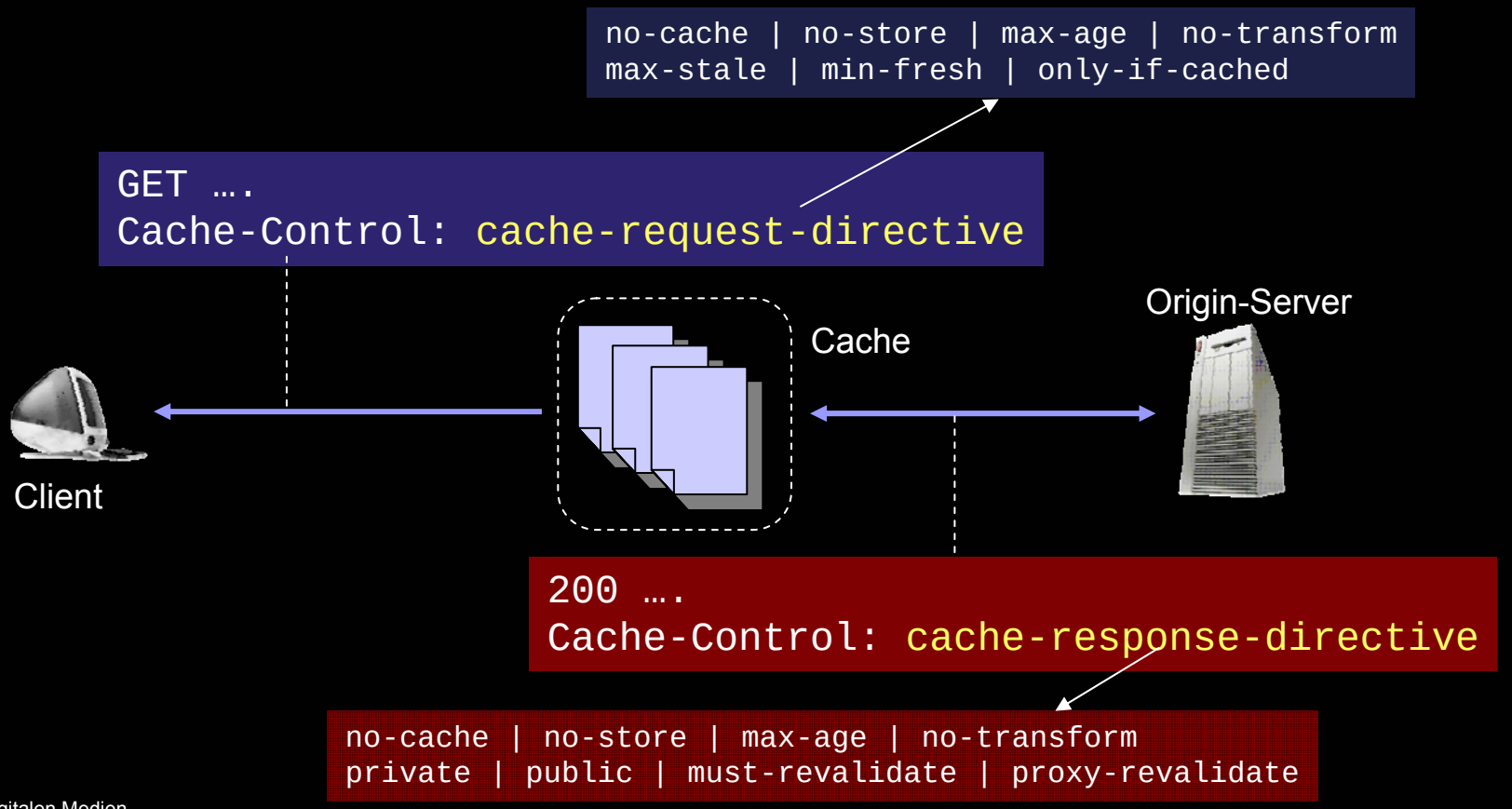
als ungültig gekennzeichnete Ressource im Cache kann trotzdem noch gültig sein

Validierungsanfrage an Origin-Server
→ Bestätigung (positive Antwort) oder Ressource neu übertragen

2. World Wide Web

HTTP/1.1 Cache Kontrolle

- HTTP **Cache-Control Headerfeld** gibt Direktiven zum Cache-Verhalten an



Dr.rer.nat. Harald Sack, Institut für Informatik, FSU Jena, Ernst-Abbe-Platz 2-4, D-07743 Jena, E-Mail: sack@minet.uni-jena.de

2. World Wide Web

HTTP/1.1 Cache Kontrolle

- **Cache-Control Headerfeld** gibt Direktiven zum Cache-Verhalten an:
  - welche Art von Ressourcen dürfen in den Cache aufgenommen werden?
    - `public / private / no-cache`
  - Einschränkungen, was im Cache gespeichert werden darf
    - `no-cache / no-store`
  - Festlegung von Expirationsmodellen
    - `max-age / min-fresh / max-stale`
  - Festlegung von Validierungsmodellen
    - `no-cache / must-ravalidate / proxy-revalidate / only-if-cached`
  - Kontrolle über Veränderung der gespeicherten Ressourcen
    - `no-transform`

2. World Wide Web

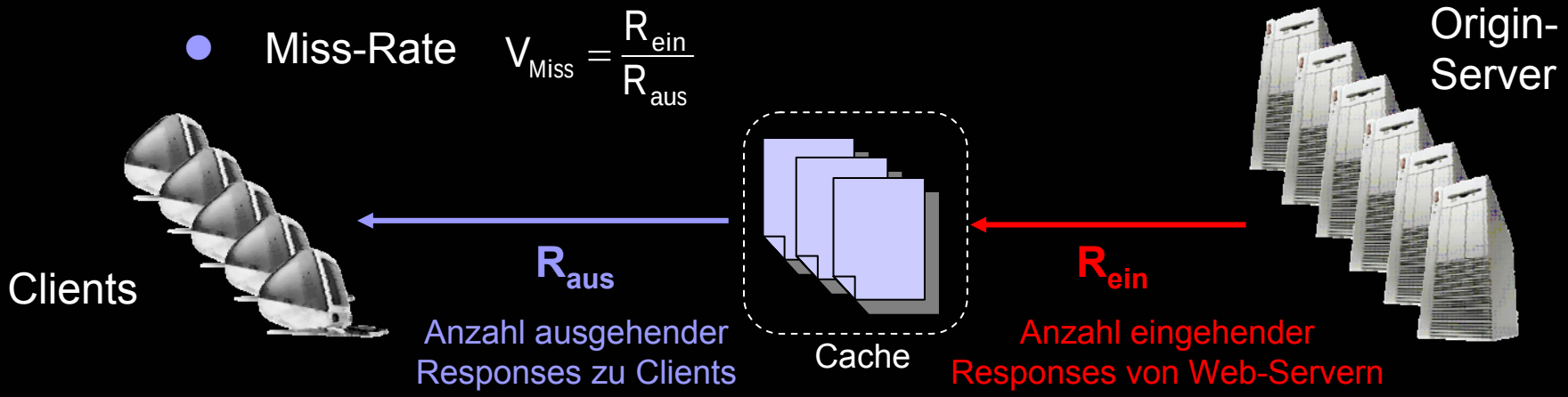
Cache Effizienz

- Je häufiger Cache-Hits bei Anfragen an den Cache auftreten, desto höher ist dessen Effizienz

- Kenngrößen:

  - Hit-Rate $V_{Hit} = \frac{(R_{aus} - R_{ein})}{R_{aus}} = 1 - \frac{R_{ein}}{R_{aus}}$

  - Miss-Rate $V_{Miss} = \frac{R_{ein}}{R_{aus}}$

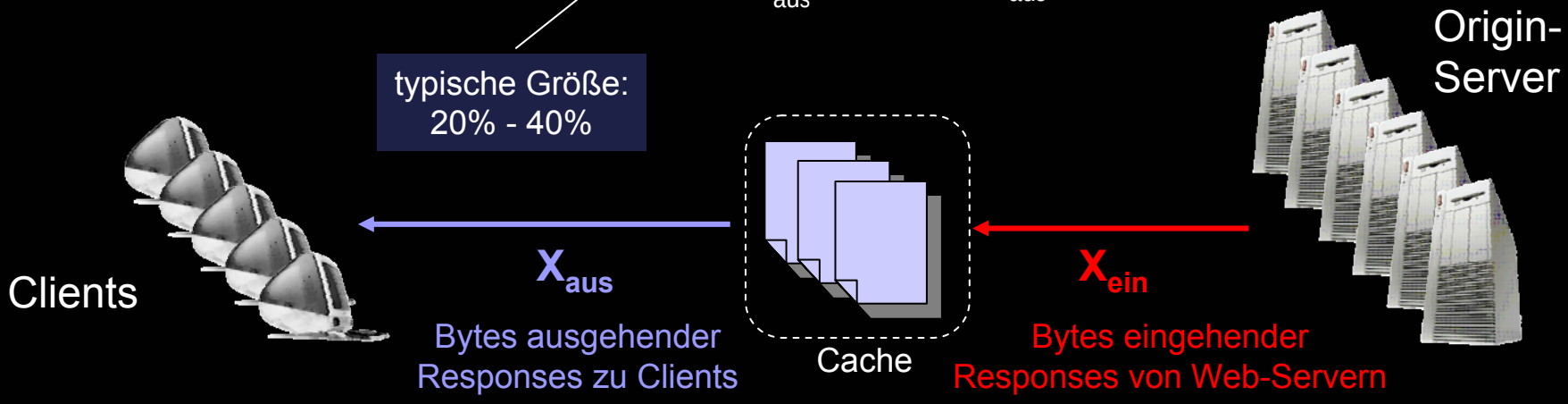


2. World Wide Web

Cache Effizienz

- Je größer die Datenmenge ist, die ein Cache direkt aus seinem Speicher an die Web-Clients ausliefern kann, desto höher ist seine Effizienz

- Kenngrößen:
  - Cache-Effizienz $N = \frac{(X_{aus} - X_{ein})}{X_{aus}} = 1 - \frac{X_{ein}}{X_{aus}}$



2. World Wide Web

Cache Effizienz

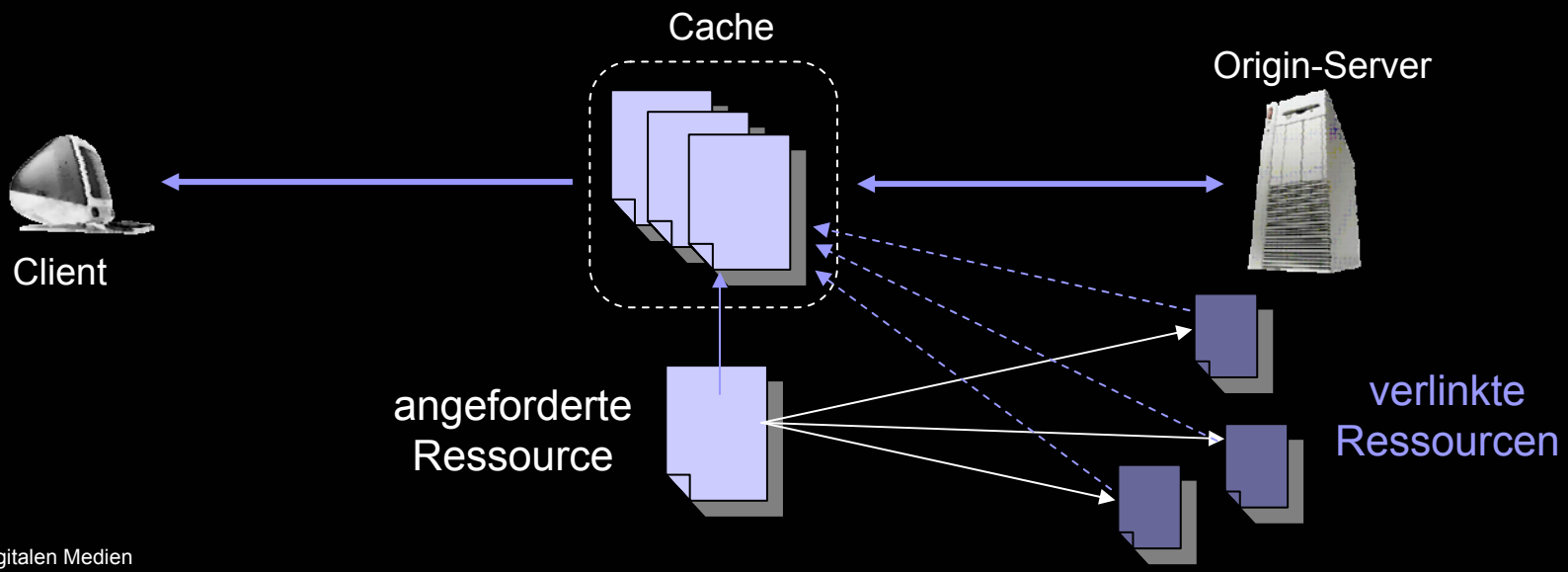
- Hit-Rate/Cache-Effizienz abhängig vom zur Verfügung stehenden Speicher
- Experimentell wurde gezeigt:
  - Hit-Rate wächst etwa logarithmisch mit der Anzahl der zugreifenden Clients
  - → Hit-Rate wächst etwa logarithmisch mit Anzahl der Anfragen
- Je größer Cache-Speicher, desto länger wird die mittlere Reaktionszeit (→ I/O-Leistung maßgeblich)
- Weitere wichtige Kenngröße:
  - Cache-Durchsatz:
    - max. Anzahl der verarbeiteten Responses pro Zeiteinheit

2. World Wide Web

Cache Effizienz

- Cache Effizienz kann weiter erhöht werden durch **Web-Pre-Caching**

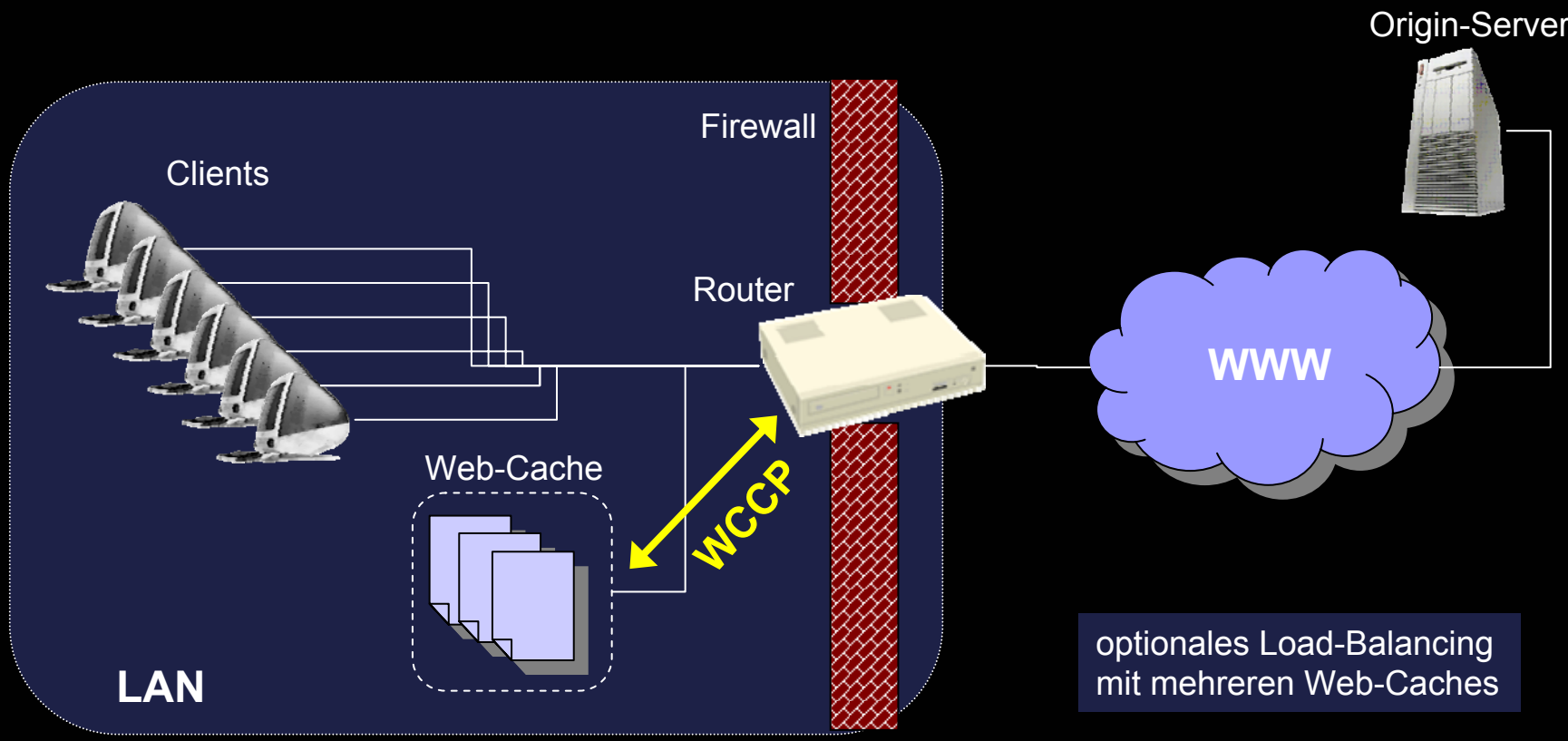
1. Client fordert Ressource an, die nicht im Cache liegt
2. Ressource wird in den Cache und zurück zum Client übertragen
3. Cache fordert zusätzlich alle Ressourcen an, die in der ursprünglich übertragenen Ressource verlinkt sind



2. World Wide Web

Web-Caching im LAN

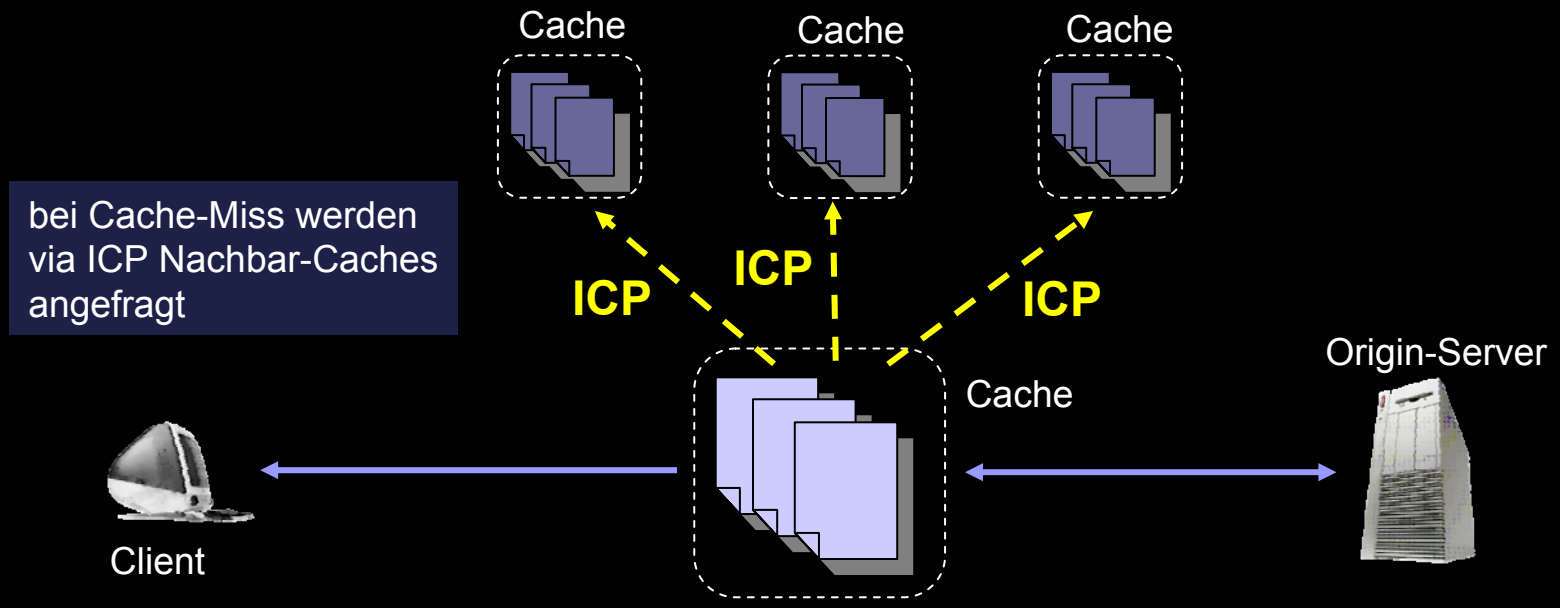
- Cisco Web Cache Communication Protocol (**WCCP**)



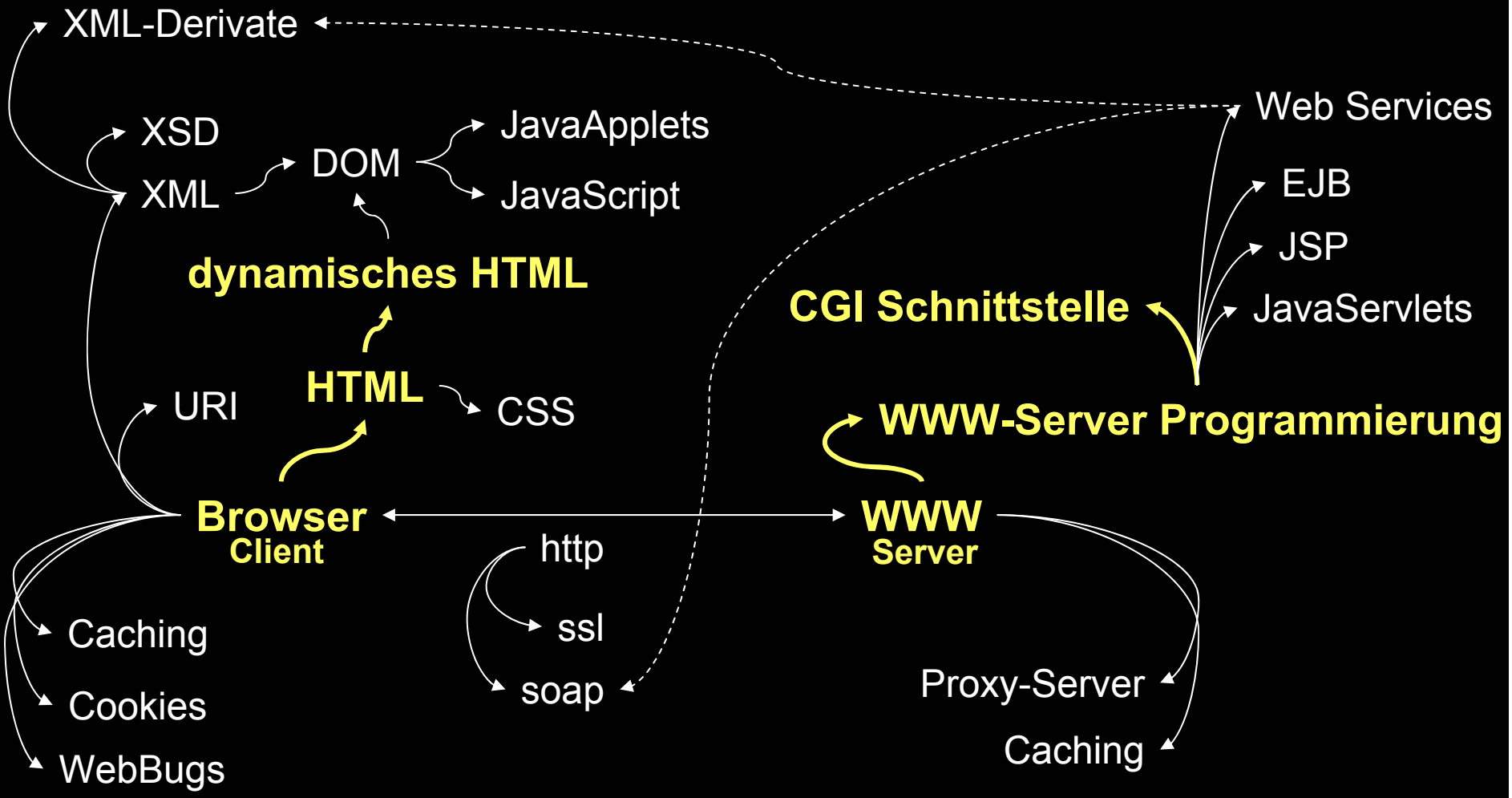
2. World Wide Web

Kooperative Web-Caches

- in großen Netzen bilden meist mehrere Web-Caches einen kooperativen Web-Cache-Verbund
- Kooperation der Caches erfolgt via **Internet Cache Protocol** (**ICP**)



2. World Wide Web



2.7 CGI-Schnittstelle

Dynamisches HTML

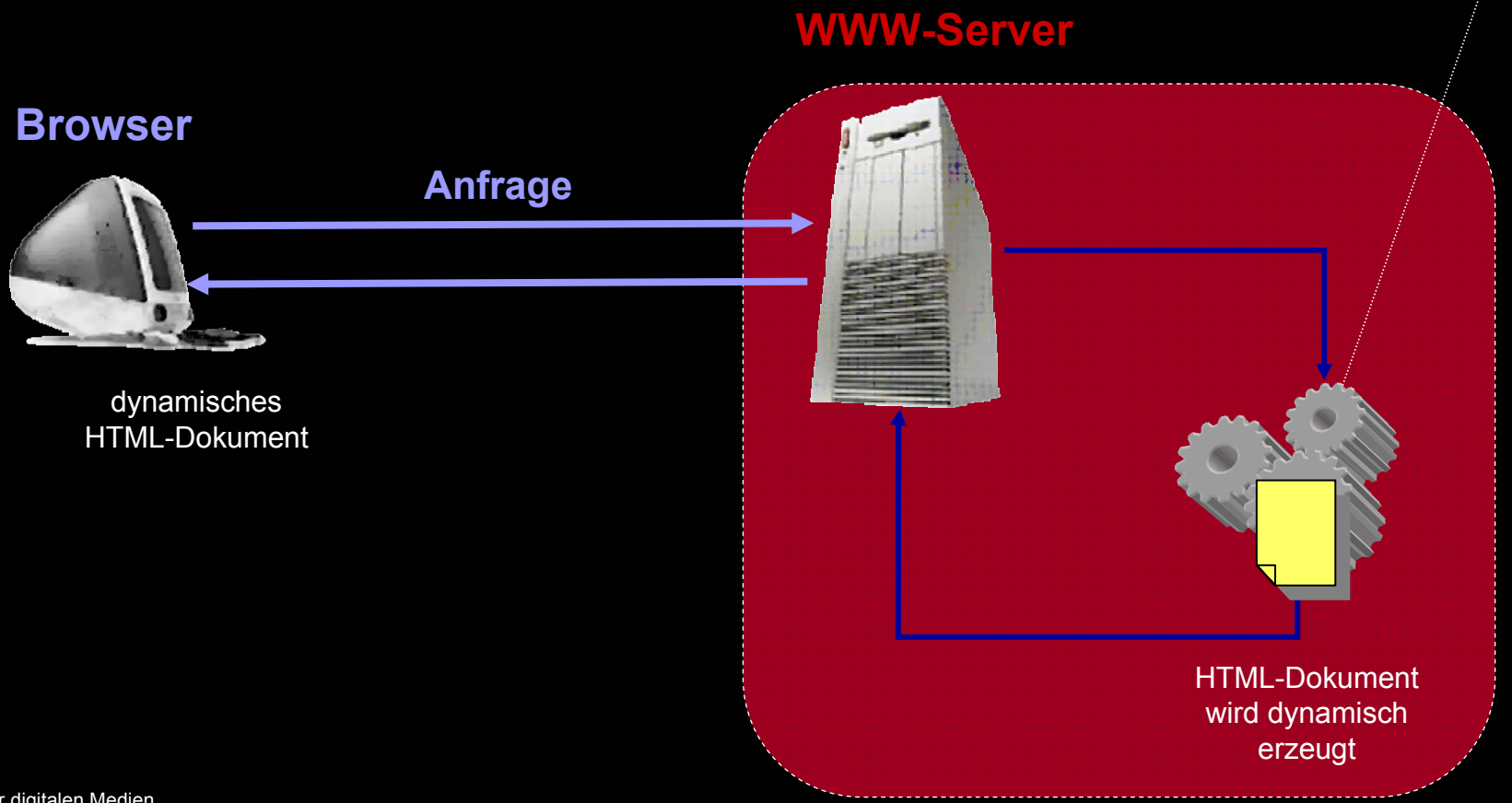
- HTML-Dokumente können **statisch**,
  - d.h. bereits vorgefertigt und auf Abruf am WWW-Server abgespeichert vorliegen oder
- **dynamisch** erzeugt werden,
  - d.h. das vom Browser angefragte HTML-Dokument **wird erst auf die Anfrage hin erzeugt.**
  - ist sinnvoll, wenn z.B. Anfragen aus einer Datenbank beantwortet werden müssen
    - z.B. Warenkataloge (vgl. amazon.de)
    - z.B. Suchmaschinen (vgl. google.de)
    - z.B. Zeitungen (vgl. heise.de)

2.7 CGI-Schnittstelle

Dynamisches HTML

- Dynamisches HTML

Steuerung notwendig
z.B. Eingabeparameter

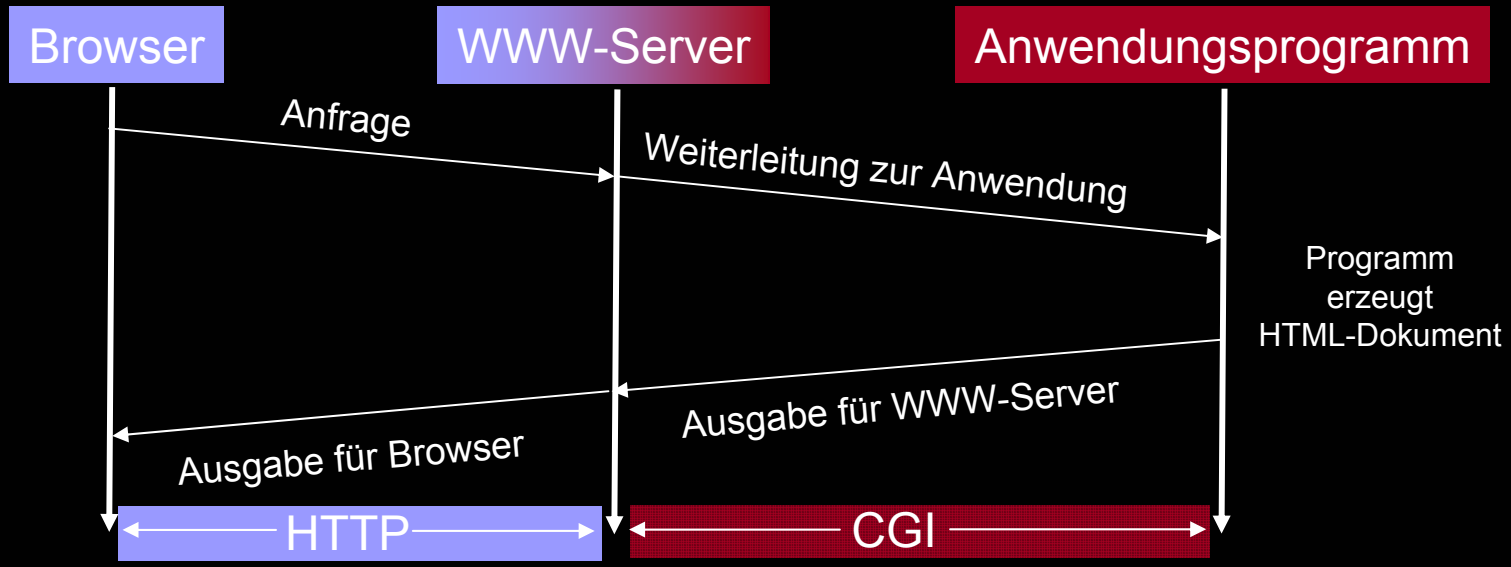


2.7 CGI-Schnittstelle

Dynamisches HTML

- **Dynamisches HTML**
  - WWW-Server bietet **Standard-Schnittstelle** für serverseitige Programme, mit deren Hilfe das dynamische HTML-Dokument erzeugt wird

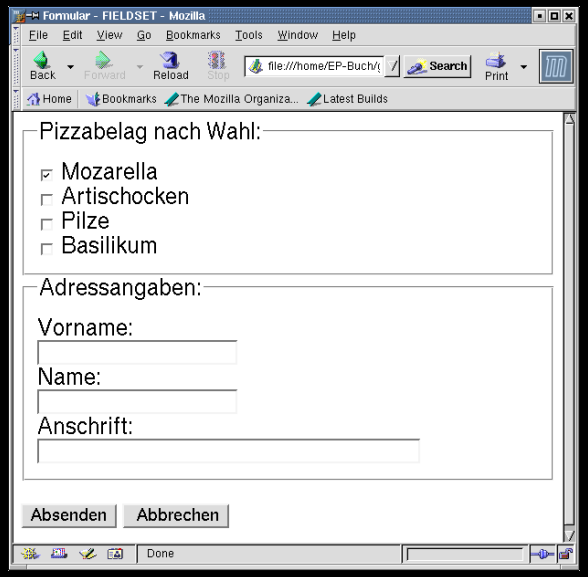
Common Gateway Interface (CGI)



2.7 CGI-Schnittstelle

Dynamisches HTML

- **Dynamisches HTML**
  - zur Steuerung der Anwendungsprogramme auf dem WWW-Server müssen **Parameter übergeben** werden können
    - Bsp.: Ausfüllen und Abschicken eines HTML-Formulars




```
<form name="PizzaPizza" action="input.htm"
 method = "GET">

<fieldset>
 <legend>Pizzabelag nach Wahl:</legend>
 <p>
 <input type="checkbox"
  name="zutat" value="mozarella" checked>
  Mozarella<br>
 <input type="ch…
  …
```

2.7 CGI-Schnittstelle

Methoden GET und POST im HTML-Formular

- **GET**
  - Formulareingaben werden an den im `action`-Attribut angegebenen **URL** hinter einem „**?**“ **angehängt**.
  - HTTP-Server übergibt dem CGI-Programm, das über den angegebenen URL aufgerufen wird die übergebenen Formulardaten in der **Umgebungsvariablen `QUERY_STRING`**.
- **POST**
  - Es wird eine **direkter HTTP-Request** an den im `action`-Attribut angegebenen URL initiiert.
  - **Formulardaten** werden in den **Body** des HTTP-Requests übergeben.
  - Body des HTTP-Requests wird dem über den URL aufgerufenen CGI-Programm über die **Standard-Eingabe** zur Verfügung gestellt.

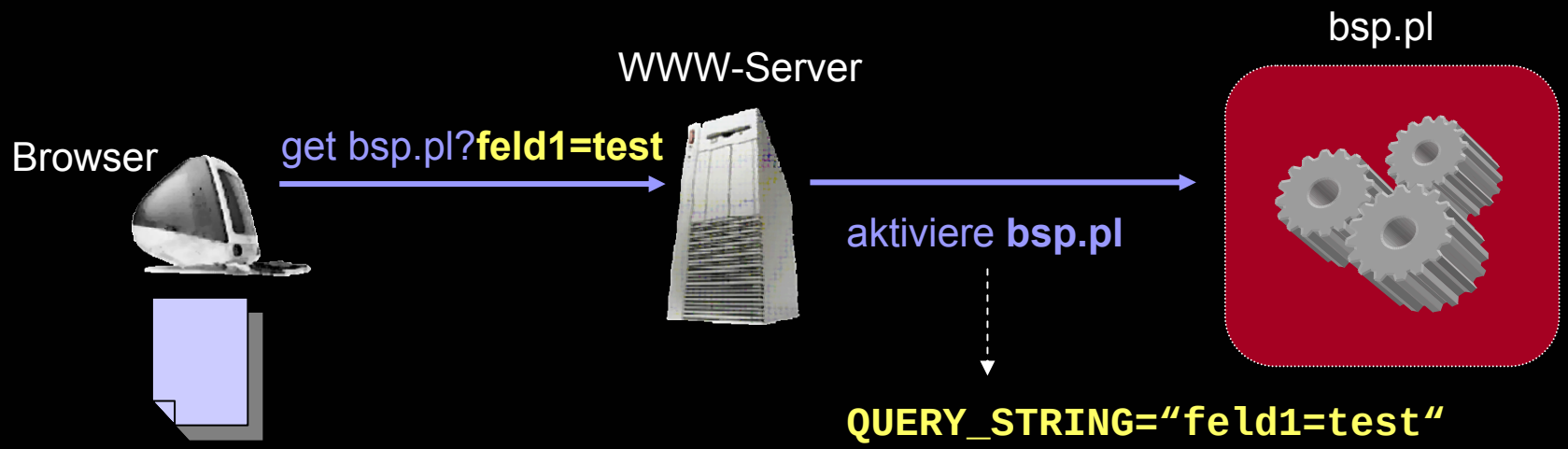
2.7 CGI-Schnittstelle

Umgebungsvariablen des HTTP-Servers

- **Server** belegt vor Aufruf des durch den **Client** initiierten CGI-Programms bestimmte Umgebungsvariablen, z.B.
  - `CONTENT_LENGTH`
  - `SERVER_NAME`
  - `GATEWAY_INTERFACE`
  - `SERVER_PROTOCOL`
  - `SERVER_PORT`
  - `REQUEST_METHOD`
  - `PATH_INFO`
  - `SCRIPT_NAME`
  - `QUERY_STRING`
  - `REMOTE_HOST`
  - …

2.7 CGI-Schnittstelle

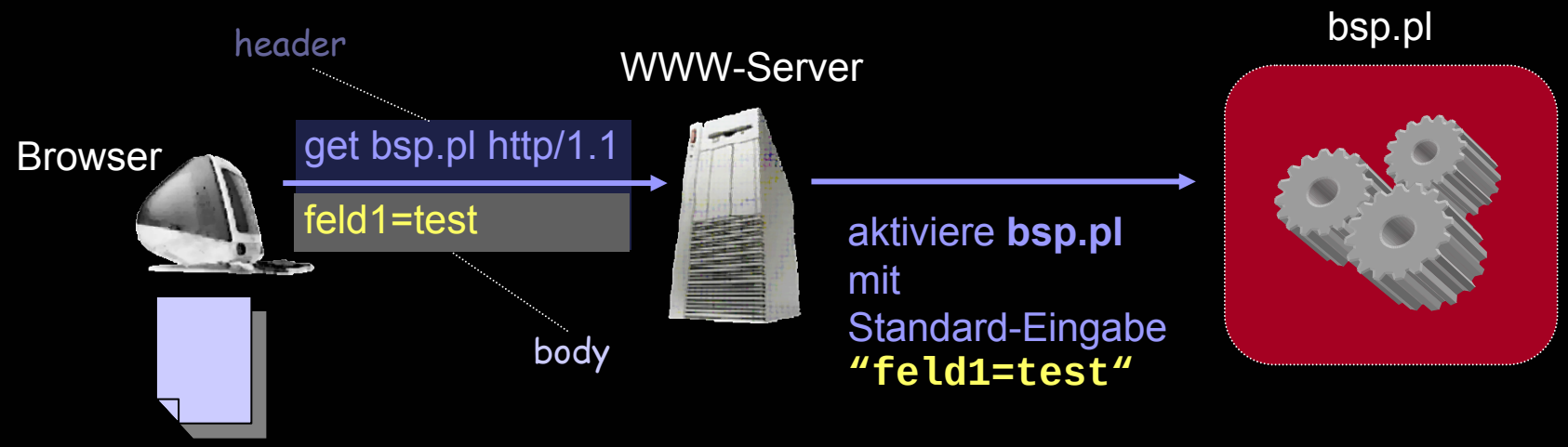
CGI-Programme – Parameterübergabe (GET)




```
feld1="test"
action="bsp.pl"
method="GET"
```

2.7 CGI-Schnittstelle

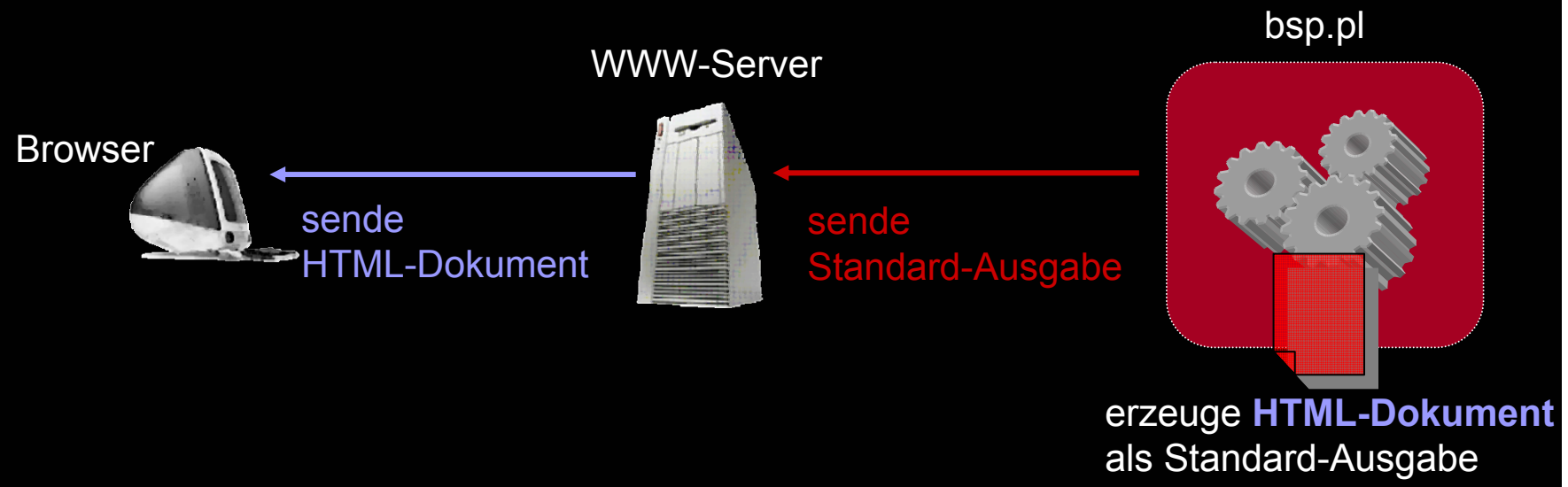
CGI-Programme – Parameterübergabe (POST)



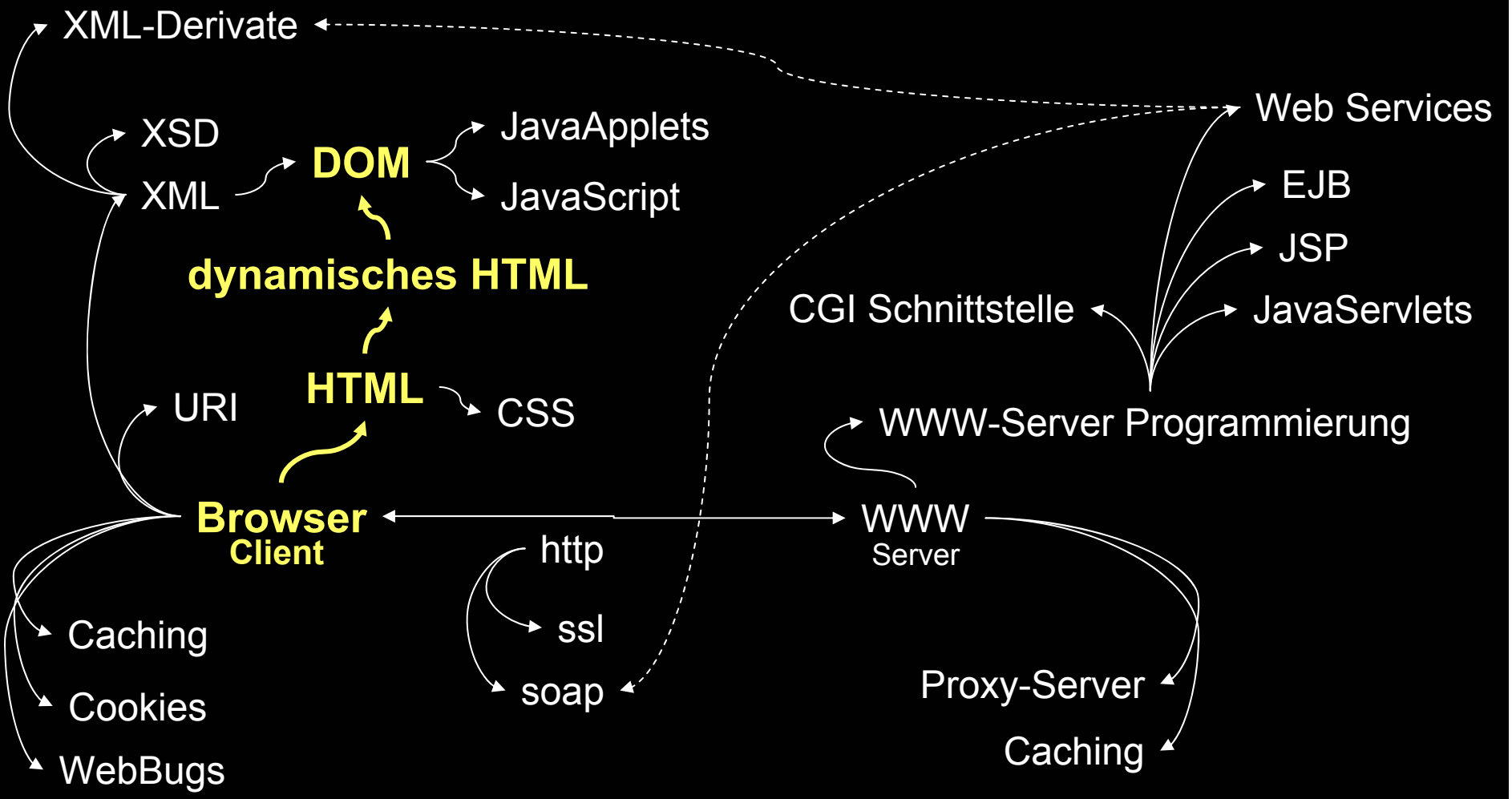

```
feld1=“test“
action=“bsp.pl“
method=“POST“
```

2.7 CGI-Schnittstelle

CGI-Programme - Standardausgabe



2. World Wide Web



2.8 Webprogrammierung – DOM

Webprogrammierung

- unterscheide
  - **Serverseitige Webprogrammierung**
    - Anwendungsprogramm wird vom WWW-Browser (Client) über den WWW-Server **auf dem Server-Rechner aktiviert und dort ausgeführt**
    - Verarbeitungsergebnisse werden als dynamisch generiertes HTML-Dokument an den Client zurück gegeben
    - standardisierte Schnittstelle: **CGI-Interface**
    - Anwendungsprogramme auf Serverseite können in **beliebiger Programmiersprache** erstellt werden +
    - → **Java Servlets**
    - → **Java Server Pages**
    - → **Enterprise Java Beans**

2.8 Webprogrammierung – DOM

Webprogrammierung

- unterscheide
  - **Clientseitige Webprogrammierung**
    - Ausführbarer/interpretierbarer Programmcode wird vom WWW-Server zusammen mit dem angeforderten HTML-Dokument an den WWW-Browser (Client) übergeben und **am Client-Rechner ausgeführt**
    - WWW-Client benötigt entsprechendes **Plugin**, das zusammen mit dem Client den übergebenen Programmcode interpretieren und ausführen kann
    - → **Java-Applets**
    - → **Javascript-Programme**

2.8 Webprogrammierung – DOM

Webprogrammierung

- Wie greife ich mit einer Programmiersprache auf Hypermedia-Dokumente und deren Inhalte zu?

Document Object Model (DOM)

- **Plattform- und Sprachunabhängige (objektorientierte) Schnittstelle (API)** für den Zugriff (lesen/erzeugen/modifizieren) auf beliebige Dokumente
- vom W3C standardisiert
  - **DOM Core-Model**
    kompakte Low Level Beschreibung für beliebig strukturierte Dokumente
  - **DOM HTML-Model**
    Objekte und Methoden zur einfachen Manipulation von HTML-Dokumenten
  - **DOM XML-Model**
    Objekte und Methoden zur einfachen Manipulation von XML-Dokumenten

2.8 Webprogrammierung – DOM

Document Object Model

- **DOM Design**
  - DOM stellt standardisierte Menge von Objekten und Methoden zur Verfügung
  - in **IDL** (Interface Definition Language) gemäß **CORBA** 2.3.1 spezifiziert
  - **DOM Level 1**
    - Core-/HTML-/XML-Modelle
    - Manipulation von Dokumenten
    - Navigation in Dokumenten
  - **DOM Level 2**
    - Style Sheet Objektmodell / Ereignismodell und Unterstützung von XML-Namespaces
  - **DOM Level 3**
    - Lade-/Speichervorgänge / Validierung von XML-Dokumenten /…

2.8 Webprogrammierung – DOM

Document Object Model

- **DOM Objektmodell**
  - DOM modelliert Dokument entsprechend seiner **Struktur**
  - Darstellung als **Strukturbaum (Graph)**
  - DOM definiert **Objekte** mit zugehörigen **Attributen** und **Methoden** zur Manipulation der Objekte
    - schirmt Objekte vor unerwünschtem Zugriff ab
  - DOM legt fest
    - **Schnittstellen** und **Objekte** zur Darstellung/Manipulation von Dokumenten
    - **Semantik** der definierten Objekte/Schnittstellen zur Beschreibung von Attributen und Verhalten
    - **Beziehungen** und **Kompatibilität** der einzelnen Objekte/Schnittstellen

2.8 Webprogrammierung – DOM

Document Object Model

- **DOM Objektmodell**
  - zu jedem Dokument gehören:
    - Wurzelknoten
    - Dokument-Typ-Knoten
    - Kommentarknoten
    - Anweisungsknoten
  - Bindung an eine bestimmte Programmiersprache erfolgt über Language Specific Binding
    - z.B. an Java oder JavaScript
  - Anwendung muss stets kompletten DOM-Strukturbaum einlesen
    - Gut geeignet, wenn Zugriff nicht vorhersagbar bzw. repetitiv
    - Ineffizient, wenn sequentieller Zugriff bzw. einmaliger selektiver Zugriff

2.8 Webprogrammierung – DOM

Document Object Model

- Strukturbaum

```
<table>
 <tbody>
  <tr>
   <td> 123 </td>
   <td> M. Schulze </td>
  </tr>
  <tr>
   <td> 234 </td>
   <td> S. Meier </td>
  </tr>
 </tbody>
</table>
```

Tabelle → HTML Quellcode

PNR	Name
123	M. Schulze
234	S. Meier

Tabelle → Browseransicht

2.8 Webprogrammierung – DOM

Document Object Model

- Strukturbaum

PNR	Name
123	M. Schulze
234	S. Meier

Tabelle → Browseransicht

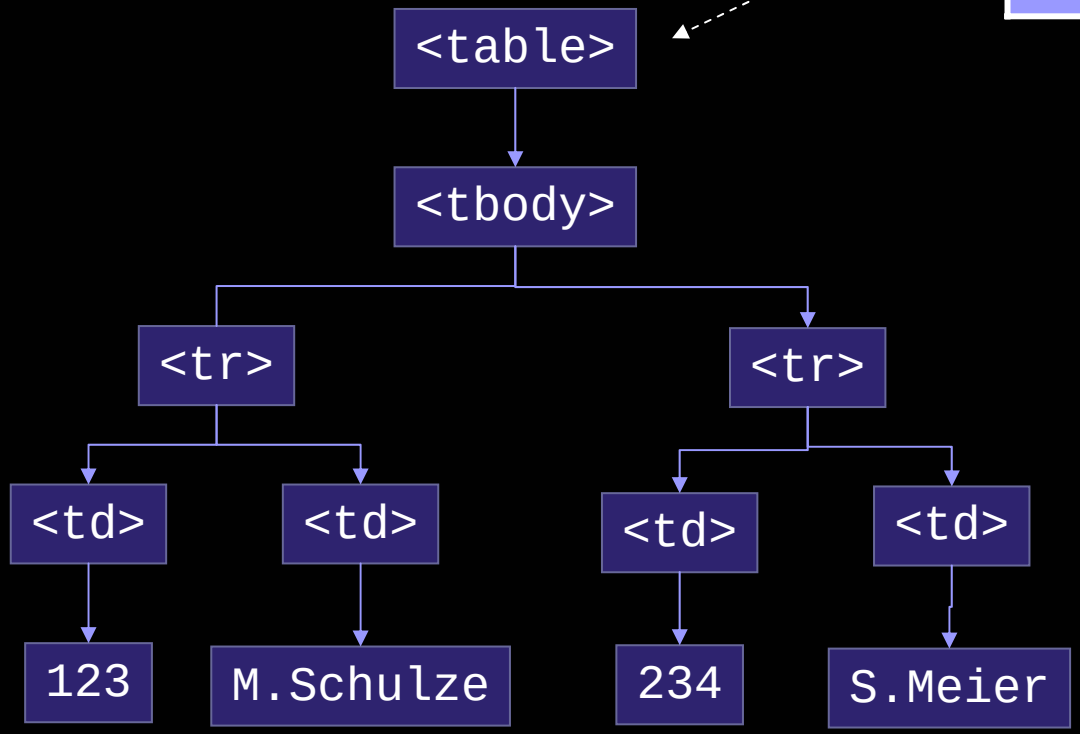
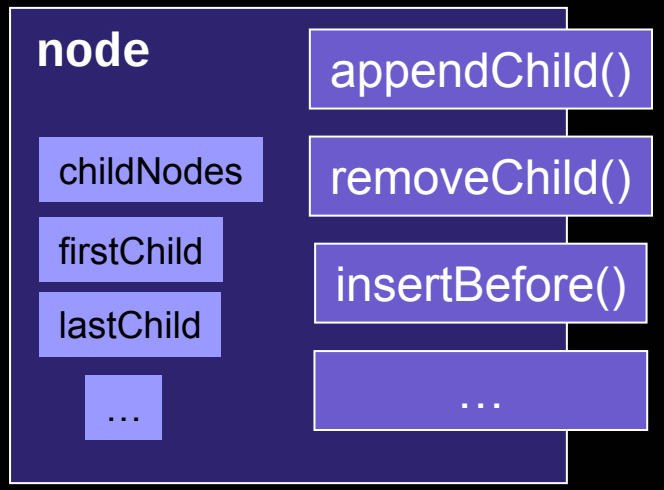


Tabelle → Strukturbaum

2.8 Webprogrammierung – DOM

Document Object Model

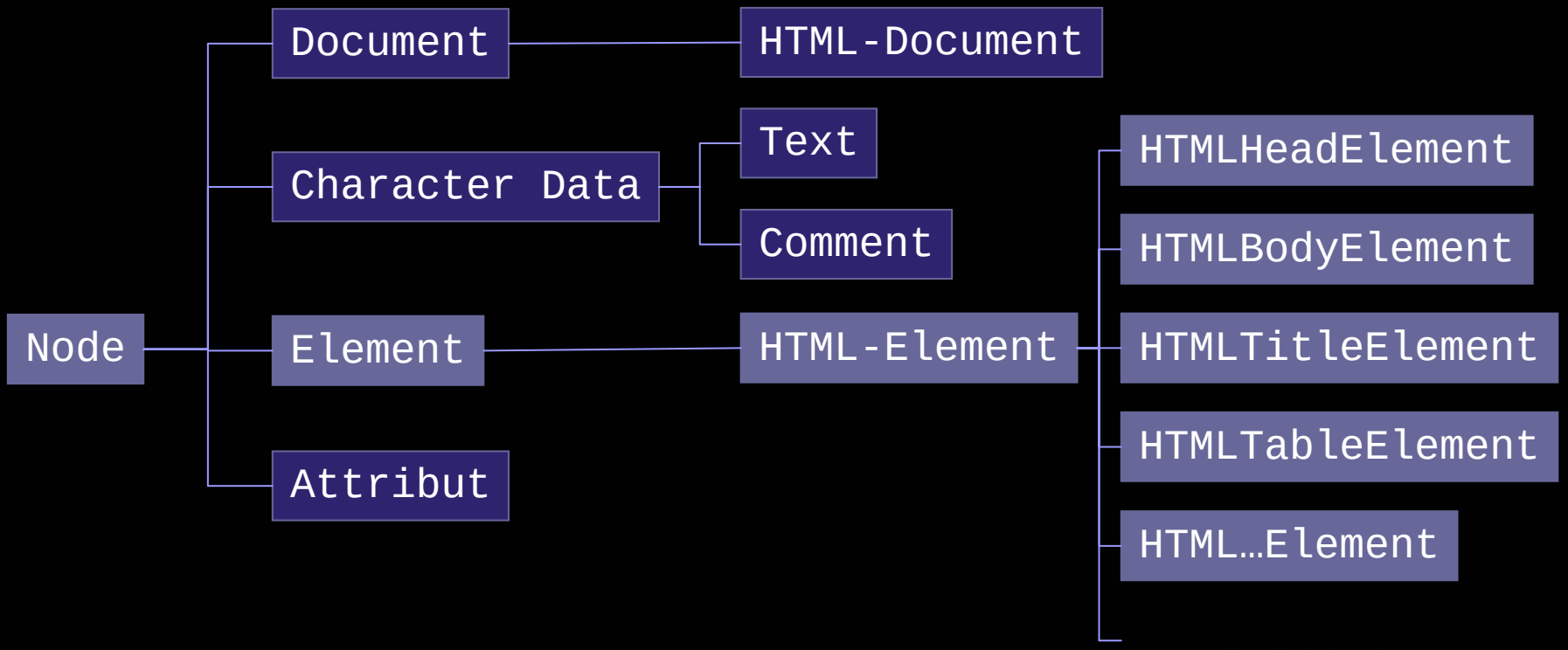
- **Referenz**
  - Zentrale Klasse: **node**
    - **Attribute:**
      - node.childNodes
      - node.firstChild
      - node.lastChild
      - node.parentNode
      - …
    - **Methoden:**
      - node.appendChild()
      - node.removeChild()
      - node.insertBefore()
      - node.cloneNode()
      - …



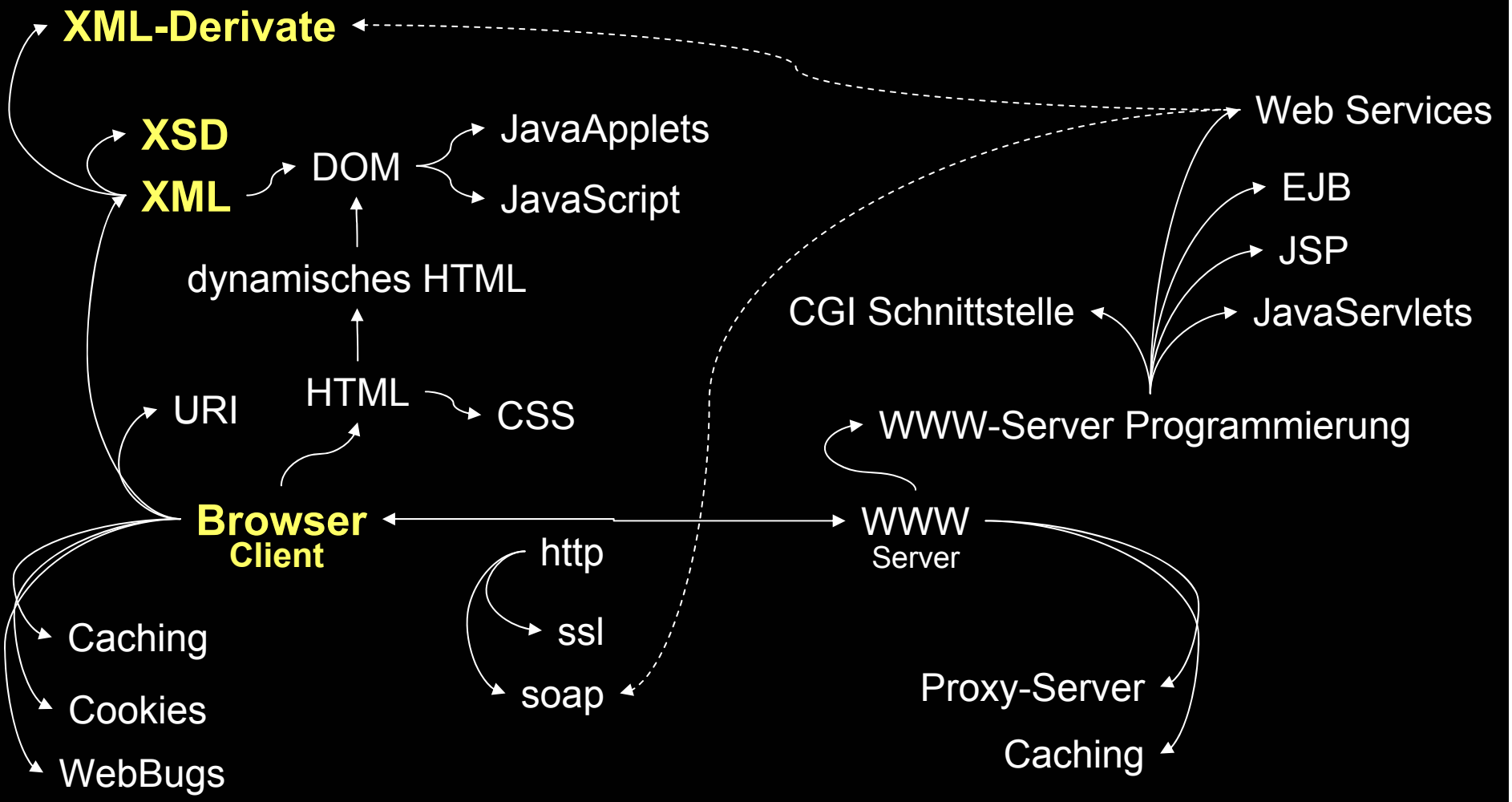
2.8 Webprogrammierung – DOM

Document Object Model

- Klassenhierarchie



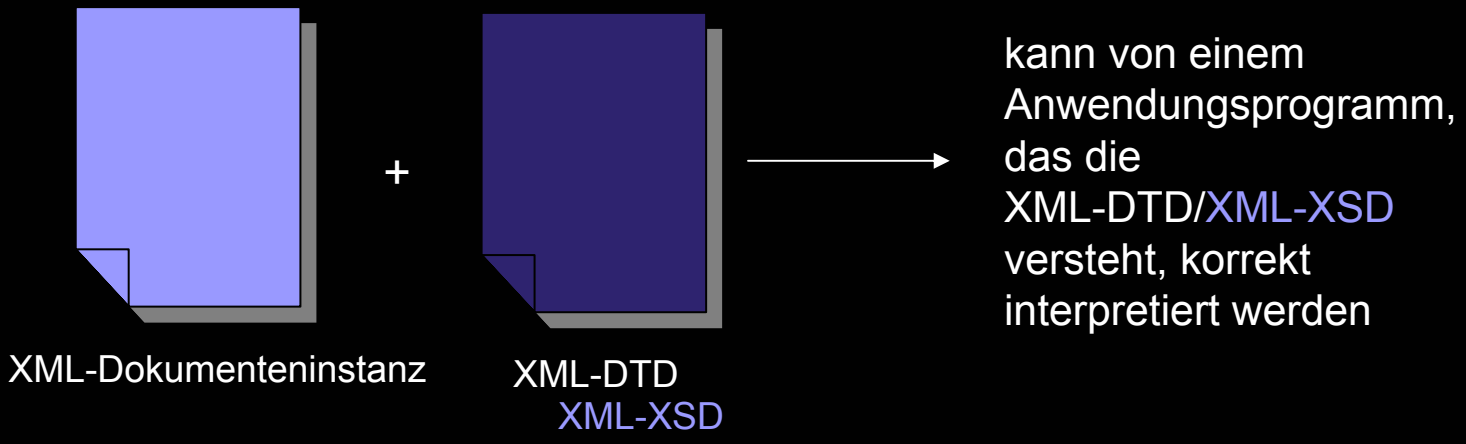
2. World Wide Web



2.9 XML und XSL

XML – Extensible Markup Language

- **Was XML alles kann**
  - XML gestattet die Definition **beliebiger neuer Tags (Metasprache zur Definition neuer Markupsprachen)**
  - die Definition neuer Tags erfolgt in einem speziellen Dokument, der **Document Type Definition** (**DTD**) / **XML Schema Definition**



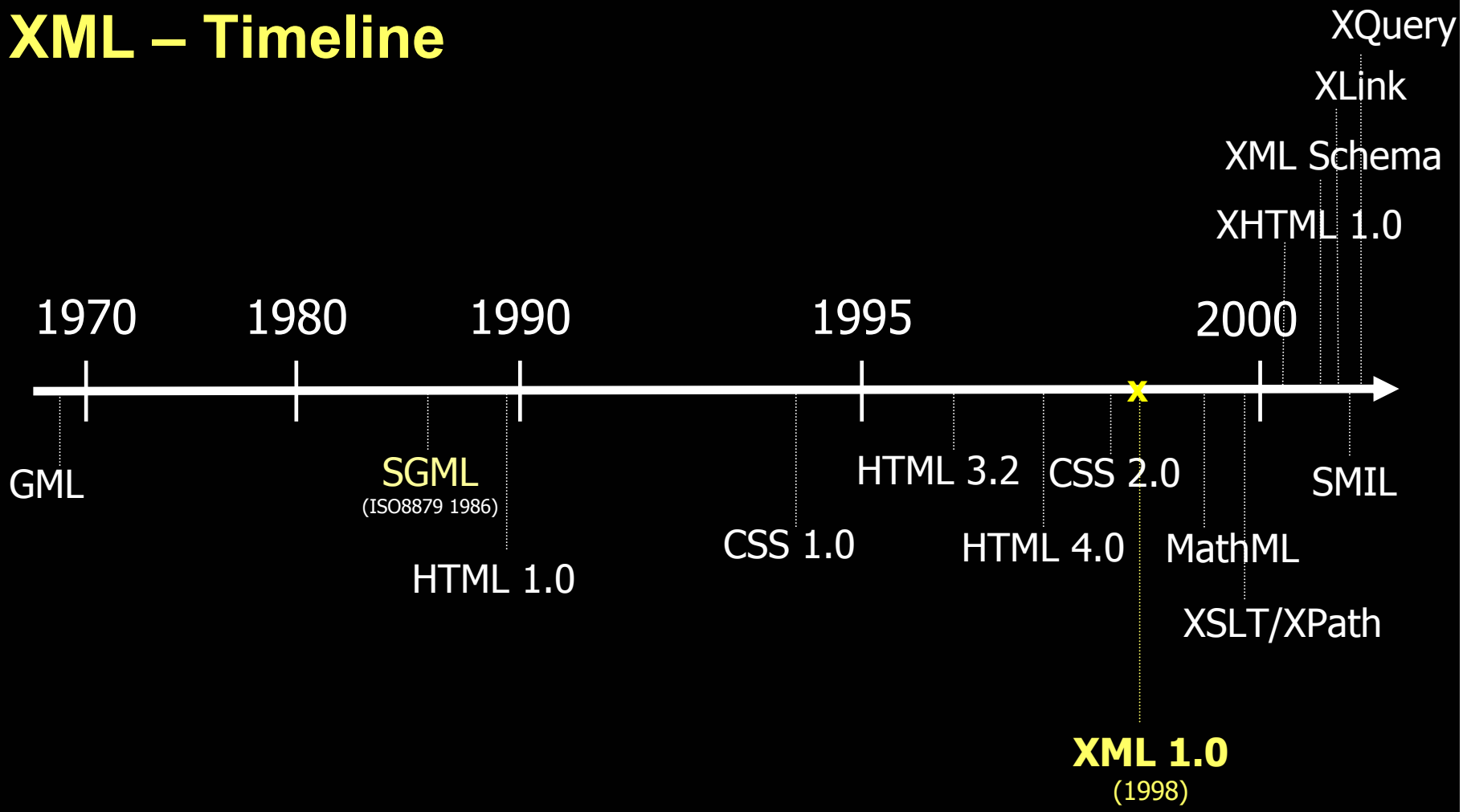
2.9 XML und XSL

XML – Sprachfamilie

- Strukturbeschreibung: **XML** **(XHTML)**
  DTD / XML Schema Definition (XSD)
- Layout/Darstellung: **XSL** (Extensible Style Language)
  XSL-FO
- Objekt-Identifikation: **XPath, XPointer, XQuery, XForms**
- Transformationen: **XSLT** (XSL Transformations)
- Hyperlinks: **XLink, XPointer, XPath**
- Abfragen: **XQuery**

2.9 XML und XSL

XML – Timeline



2.9 XML und XSL

XML – Extensible Markup Language

- Was XML alles kann…

```
<h2>
  Max Mustermann
</h2>
<p>
  Sesamstr. 49a <br>
  <b>93123  Bad Sulzdetfurth </b>
</p>
```

HTML

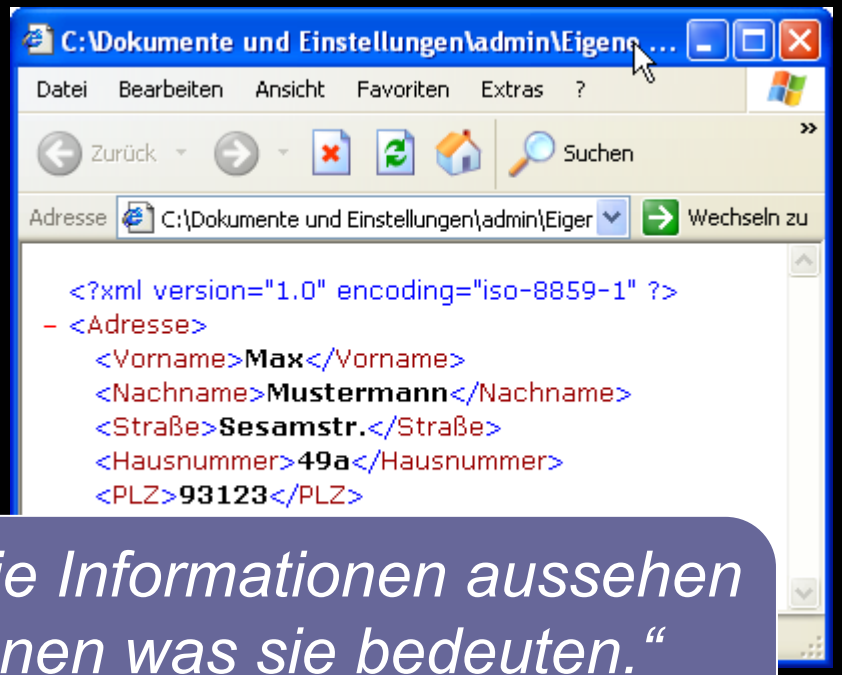
```
<Adresse>
  <Vorname> Max </Vorname>
  <Nachname> Mustermann </Nachname>
  <Straße> Sesamstr. </Straße>
  <Hausnummer> 49a </Hausnummer>
  <PLZ> 93123 </PLZ>
  <Ort> Bad Sulzdetfurth </Ort>
</Adresse>
```

XML

2.9 XML und XSL

XML – Extensible Markup Language

- **Was XML alles kann…**



„HTML sagt Ihnen wie die Informationen aussehen sollen, aber XML sagt Ihnen was sie bedeuten.“

Quelle: Charles F. Goldfarb, „The XML Handbook“

2.9 XML und XSL

XML – Extensible Markup Language

SGML (Standard Generalized Markup Language)

Ψ $\cup$

HTML | XML $\ni$ XHTML

- Instanz von SGML
- Vermischung von Inhalt und Layout
- Darstellung von Information

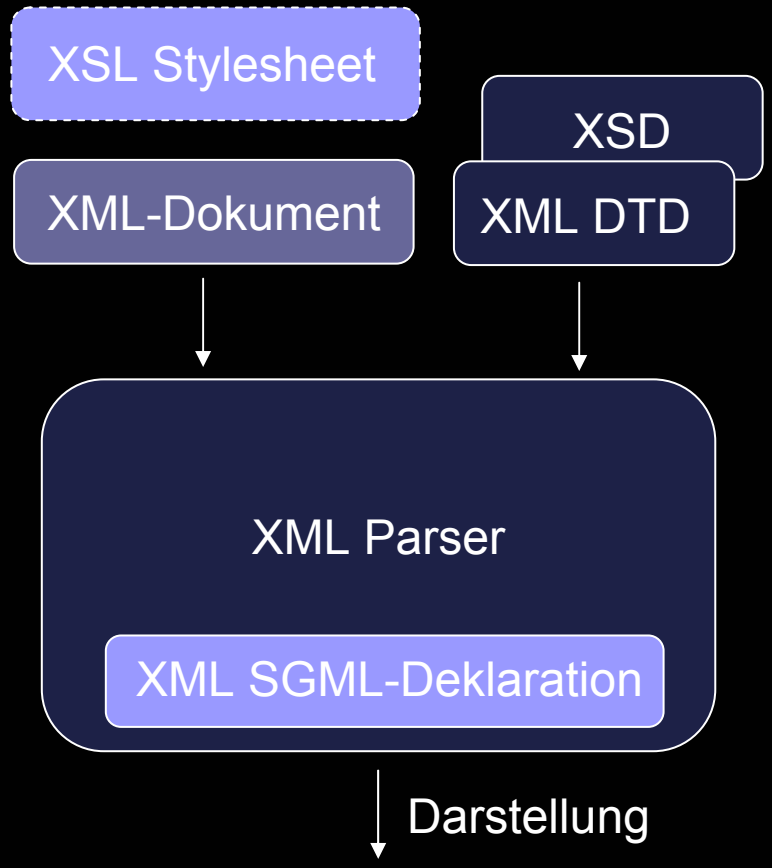
- Teilmenge von SGML
- Strikte Trennung von Inhalt und Layout
- Beschreibung von Information

XHTML ist Instanz von XML

2.9 XML und XSL

XML – Extensible Markup Language

- **HTML** unterstützt stets nur einen **vordefinierten Dokumententyp**
- **XML** erlaubt die Definition **eigener Dokumenten-Typen** über die Definition einer zugehörigen **DTD** (**Document Type Definition**) bzw. **XML Schema Definition** (**XSD**)



2.9 XML und XSL

XML–Dokument

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE address SYSTEM "Beispiel.dtd">
<Adresse>
  <Vorname> Max </Vorname>
  <Nachname> Mustermann </Nachname>
  <Straße> Sesamstr. </Straße>
  <Hausnummer> 49a </Hausnummer>
  <PLZ  prefix="D"> 93123 </PLZ>
  <Ort> Bad Sulzdetfurth </Ort>
</Adresse>
```

XML Deklaration

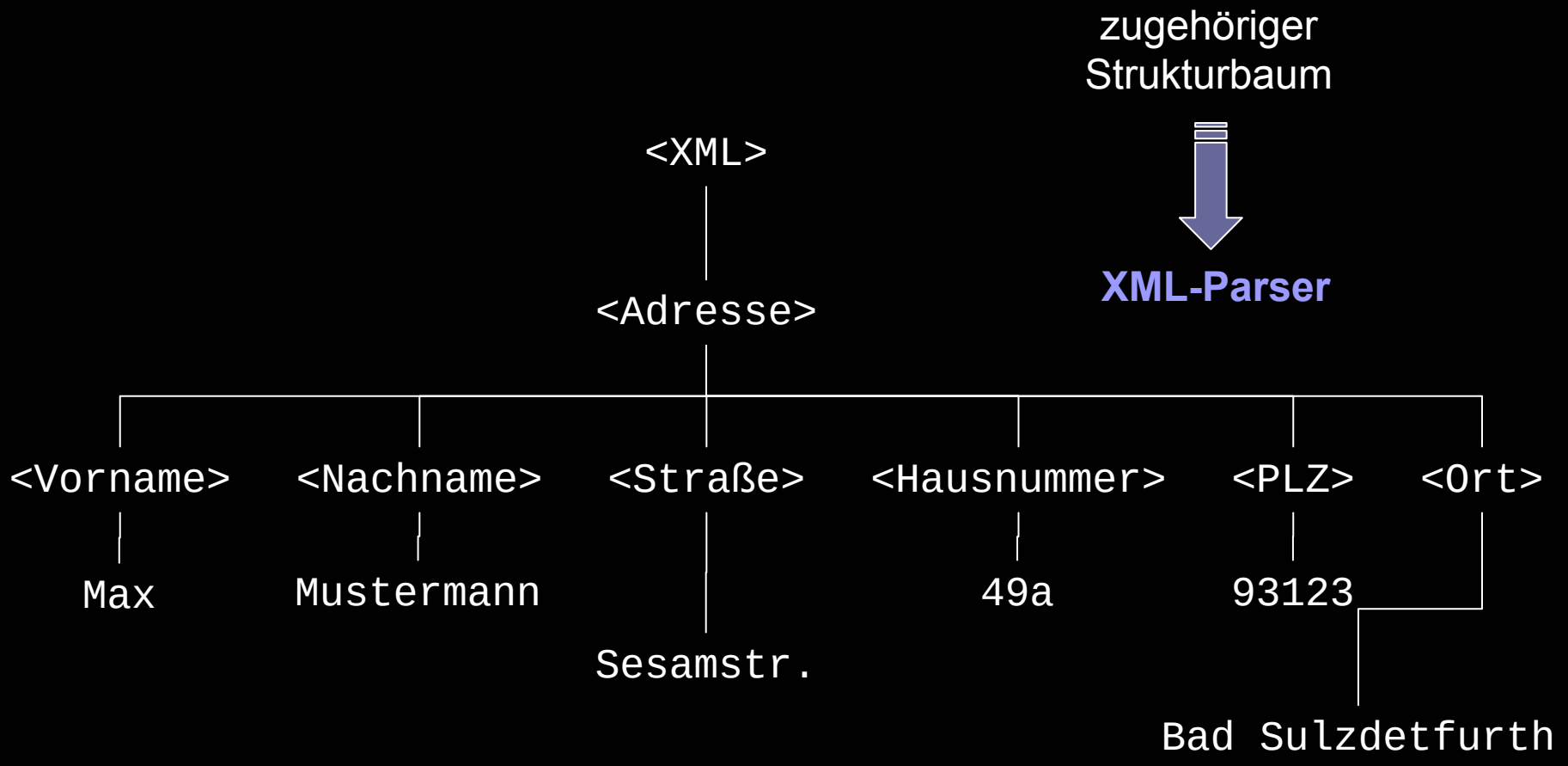
XML DTD

XML Tags

XML Attribute

2.9 XML und XSL

XML–Dokument



2.9 XML und XSL

XML–Parser

- **DOM – Domain Object Model**
  - **objektmodellorientiert**
  - XML-Dokument als Baumstruktur:
    - Jeder Knoten entspricht einem Objekt
    - Jedes XML-Element kann beliebig viele Unter-Elemente besitzen
  - Sehr **flexibel**, aber
    - **hoher Speicher-/CPU-Verbrauch**

2.9 XML und XSL

XML–Parser

- **SAX – Simple API for XML**
  - **ereignisorientiert**
  - liest XML-Dokument **sequentiell** ein
  - behandelt XML-Elemente wie eintretende **Ereignisse**
  - „Event-Handler“ muß vom Entwickler selbst programmiert werden
  - sehr **schnell, geringer Speicheraufwand**, aber
    - kann XML-Dokument nur sequentiell verarbeiten

2.9 XML und XSL

XML – Extensible Markup Language

- **Es geht auch ohne DTD**

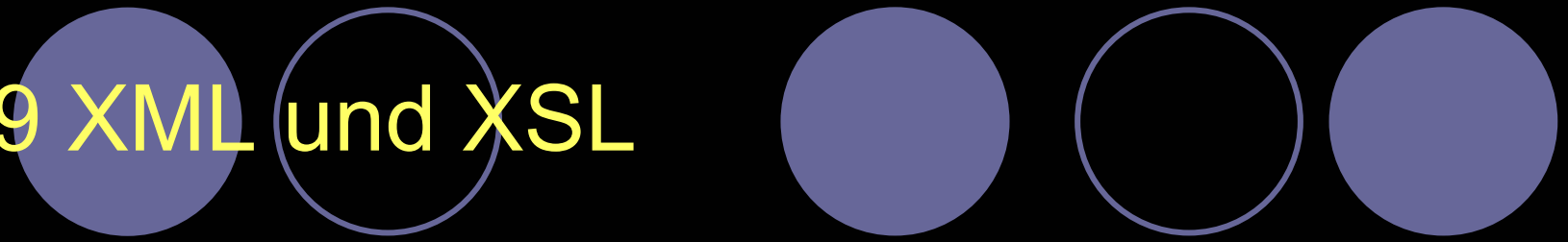
Max Mustermann
Sesamstr. 49a

```
<Adresse>
 <Vorname> Max </Vorname>
 <PLZ> 95123 </PLZ>
 <Ort> Bad Sulzdetfurth </Ort>
</Adresse>
```

syntaktisch korrekt ??

Wohlgeformte XML-Dokumente:
Syntaktisch korrekt, d.h. alle angefangenen Tags werden wieder geschlossen und sind stets korrekt geschachtelt,
keine Mehrfachattribute.

2.9 XML und XSL



XML – Extensible Markup Language

- DTD liefert Semantik

```
<Adresse>
 <Vorname> Max </Vorname>
 <Nachname> Mustermann </Nachname>
 <Straße> Sesamstr. </Straße>
 <Hausnummer> 49a </Hausnummer>
 <PLZ> 93123 </PLZ>
 <Ort> Bad Sulzdetfurth </Ort>
</Adresse>
```

XML

```
<!ELEMENT Adresse (Vorname,
  Nachname, Straße, Hausnummer,
  PLZ, Ort)>

<!ELEMENT Vorname CDATA>
<!ELEMENT Nachname CDATA>
<!ELEMENT Straße CDATA>
….
```

DTD

validierte XML-Dokumente:
wohlgeformt und den Definitionen der zugeordneten DTD entsprechend
(== formale Grammatik)

2.9 XML und XSL

XML – Namensräume

- XML-Autoren definieren eigene XML Element-Typen (Tags)
  - XML-Tags mit demselben Namen, aber unterschiedlicher Bedeutung und unterschiedlichen Attributen

⇒ **Ambiguität** und **Namenskonflikte**

```
< CompactDisk Autor="HS">
  <Titel>Remixes</Titel>
  <Track Nummer="1">
    <Titel>Night over Manaus</Titel>
    <Autor>Boozoo Bajou</Autor>
  </Track>
</CompactDisk>
```

2.9 XML und XSL

XML – Namensräume

- **XML Namensraum** legt eindeutigen Kontext für XML Element-Typ fest
  - definiert Namen, die jeweils bestimmter DTD zugeordnet werden
  - benötigt **keine formale Struktur**
  - **Zerlegung** komplexer Strukturen in handhabbare Teilbereiche
  - **Mischung** verschiedener Namensräume

- jeweils eigene Namensräume für Elemente und XML-Entitäten pro Dokument / Dokumentenklasse (Gültigkeitsbereich)
- jeweils eigener Namensraum für Attribute pro Element (Gültigkeitsbereich)
- jeweils eigene universelle Namensräume für Fremddaten-Entitäten und Prozessanweisungen

2.9 XML und XSL

XML – Namensräume

- Beispiel:

(1) Deklaration:

```
xmlns:compactDisk = “http://www.uni-jena.de/cd.dtd“
xmlns:track = “http://www.beispiel.de/track.dtd“
```

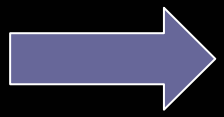
(2) Verwendung:

```
<compactDisk:titel> … </buch:titel>
<track:titel> … </artikel:titel>
```

2.9 XML und XSL

XML Schema Definition

- XML-DTDs werden in eigener (nicht XML-konformer) Syntax realisiert
  - Extended Backus-Naur Form (**EBNF**)
  - Spezieller Interpreter notwendig
- XML-DTDs unterstützen lediglich den Datentyp “text“
- XML-DTDs unterstützen nicht das Vererbungskonzept



XML Schema Definition:
- XML-spezifische Sprachdefinition die auf XML-Syntax beruht

noch einfacher → RelaxNG

2.9 XML und XSL

XML – Syntax

- XML-Elemente bestehen aus **Start-Tag** und **Ende-Tag**

```
<start-tag> … </ende-tag>
```

- Ausnahme: **leeres Element**

```
<leeres-tag/>
```

2.9 XML und XSL

XML – Syntax

- Bestandteile der XML 1.0 Syntax:
  - **Text-Literale**: `“Textliteral“`
  - **Character Data**: Jeglicher Text, der weder Teil des Markup, eines Elements oder Attributs ist
  - **Attribute**: `<autor name=“F. Schiller“>`
  - **Entity-Referenzen**: Definition eines Platzhalters
    `<!ENTITY copyright "&#xA9;">`
  - **Anweisungen**: `<?target …anweisung… ?>`
  - **Kommentare**: `<!-- ..kommentar.. -->`
  - **CDATA-Blöcke**: Einbettung von Text, der sonst als Markup interpretiert wird.
    `<![CDATA[…textblock…]]>`

2.9 XML und XSL

XML – Syntax

- **XML-Prolog**
  - **XML-Deklaration:**

```
<?xml version="1.0" encoding="UTF-8"
      standalone="yes">
```

- version: obligatorische Versionsnummer
- encoding: **UTF-8/16/32 …**
- standalone: **yes/no** (wird DTD benötigt?)

2.9 XML und XSL

XML – Syntax

- **XML-Prolog**
  - **XML-Referenzen:**
    - Angabe des verwendeten XML-DTDs
    - Referenz auf externe Teilmenge (globale URI)

```
<!DOCTYPE name SYSTEM “URI“>
```

    - Referenz auf externe Teilmenge (lokales Dateisystem)

```
<!DOCTYPE name PUBLIC “lok. URI“ “glob. URI“>
```

    - Auf interne Teilmenge:

```
<!DOCTYPE name [DTD-Definitionen]>
```

2.9 XML und XSL

XML – Syntax

- **Document Type Definition – DTD**
  - DTD legt fest:
    - welche **Elementtypen** gibt es
    - welchen **Inhalt** dürfen sie haben
    - welche **Attribute** sind erlaubt
    - welche **Werte** sie dürfen annehmen
  - **XML-Elementtyp:**

```
<!ELEMENT name (inhalt)>
```

2.9 XML und XSL

XML – Syntax

- **XML-Elementtyp:**
  - Elementinhalt

```
<!ELEMENT Buch (Titel, Autor+, Kapitel+)>
```

  - Sequenzen

```
<!ELEMENT Kapitel (Überschrift, Einleitung?,
                   Inhalt) >
```

  - Alternativen

```
<!ELEMENT Pause (Tee|Kaffee, Zucker?, Kuchen*) >
```

2.9 XML und XSL

XML – Syntax

- **XML-Elementtyp**- Schlüsselwörter:
  - #PCDATA-Element: „Parsed Character Data“ (erlaubt Schachtelung)

```
<!ELEMENT Text (#PCDATA)>
```

  - Leeres Element

```
<!ELEMENT leeresElement EMPTY>
```

  - Beliebiger Inhalt

```
<!ELEMENT Kommentar ANY>
```

2.9 XML und XSL

XML – Syntax

- XML-Attribute

```
<!ATTLIST elementname attributname
          attributtyp vorgabewert >
```

- Attributtypen:
  - CDATA

```
<!ATTLIST Pause Dauer CDATA #FIXED “15min“>
```

  - ID

```
<!ATTLIST Buch ISBN ID #REQUIRED>
```

2.9 XML und XSL

XML – Syntax

- **Attributtypen:**
  - **NMTOKEN**:
    Attributwert wird nicht als Text, sondern als **Symbol** behandelt

```
<!ATTLIST Buch Zugriff NMTOKEN #REQUIRED>

<Buch Zugriff="geheim"> …
```

z.B: Koppelung an XML-Elementtyp Person

```
<!ATTLIST Person Zugriffsrecht NMTOKEN #REQUIRED>

<Person Zugriffsrecht="geheim">…
```

2.9 XML und XSL

XML – Syntax

- **DTD Beispiel**:

```
<!ELEMENT Buch (Titel, Subtitel?, Autor+, Preis)>
<!ATTLIST Buch ISBN ID #REQUIRED)>
<!ELEMENT Titel (#PCDATA)>
<!ELEMENT Subtitel (#PCDATA)>
<!ELEMENT Autor (#PCDATA)>
<!ELEMENT Preis (#PCDATA)>
```

2.9 XML und XSL

XML – Syntax

- zugehöriges XML-Dokument:

```
<!DOCTYPE Buch SYSTEM “buch.dtd“>
<Buch ISBN=“3-411-20900-3“>
   <Titel>Duden</Titel>
   <Subtitel>
      Rechtschreibung der deutschen Sprache
   </Subtitel>
   <Autor>Dudenredaktion, Hrsg.</Autor>
   <Preis>10,95</Preis>
</Buch>
```

2.9 XML und XSL

XML – Syntax

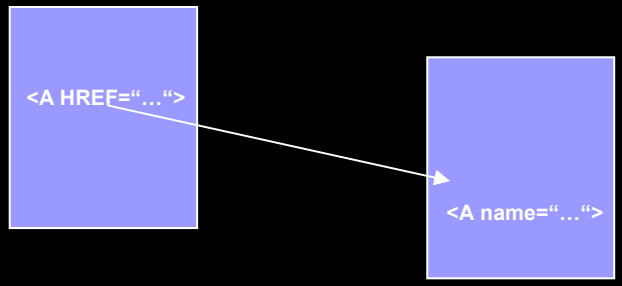
- im Vergleich dazu → XML-Schema:

```
<?xml version="1.0" encoding="utf-8"?>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:complexType name="BuchTyp">
        <xsd:element name="Buch">
          <xsd:attribute name="ISBN" type="xsd:string"
            use="required"/>
          <xsd:all>
            <xsd:element name="Titel" type="xsd:string"/>
            <xsd:element name="Subtitel" type="xsd:string"
                         minOccurs="0" maxOccurs="1"/>
            <xsd:element name="Autor" type="xsd:string"
                         minOccurs="1"/>
            <xsd:element name="Preis" type="xsd:decimal"/>
          </xsd:all>
        </xsd:element>
      </xsd:complexType>
...
```

2.9 XML und XSL

XLink, XPath und XPointer

- XML dient der Strukturierung von Dokumenten im WWW
- Notwendigkeit der Definition von Hyperlinks zur **Dokumentenverknüpfung** im WWW

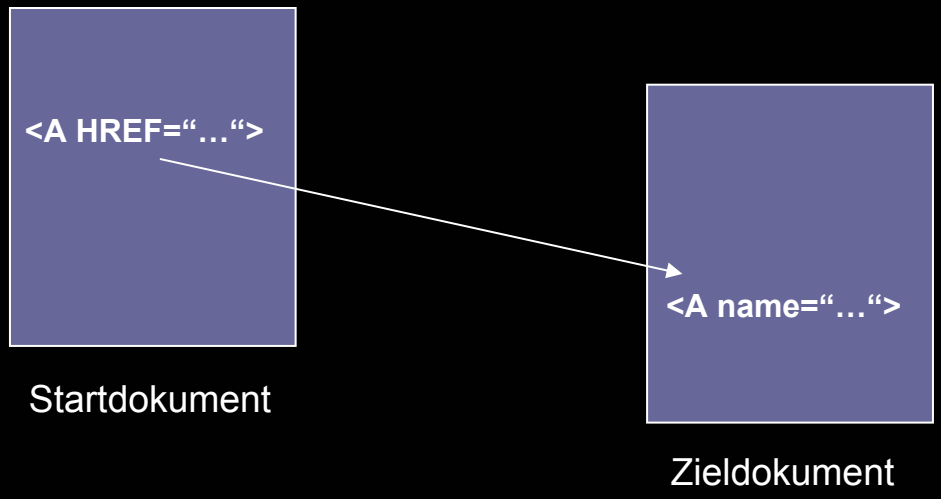


- XML Linking Language – **XLink**
  - Definition von XML-Hyperlinks
- XML Pointer Language – **XPointer** und **XPath**
  - Definition von Ankerpunkten in XML Dokumenten
  - Definition von Dokumentenfragmenten

2.9 XML und XSL

XLink

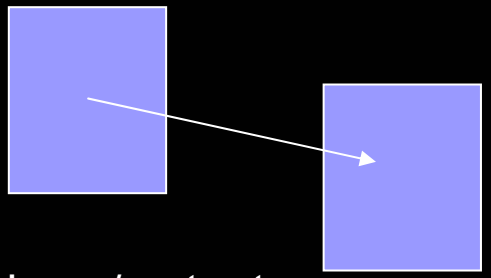
- **HTML <A> -Element**
  - **Unidirektionale** Verknüpfung zwischen **genau zwei** Informationsressourcen
  - Stets **Bestandteil** des Start-Dokuments
  - **Darstellung** hängt von Browser ab



2.9 XML und XSL

XLink

- **XLink – Simple Links**
  - entspricht HTML-Hyperlink
  - zusätzliche Attribute: typ / role / title / show / actuate

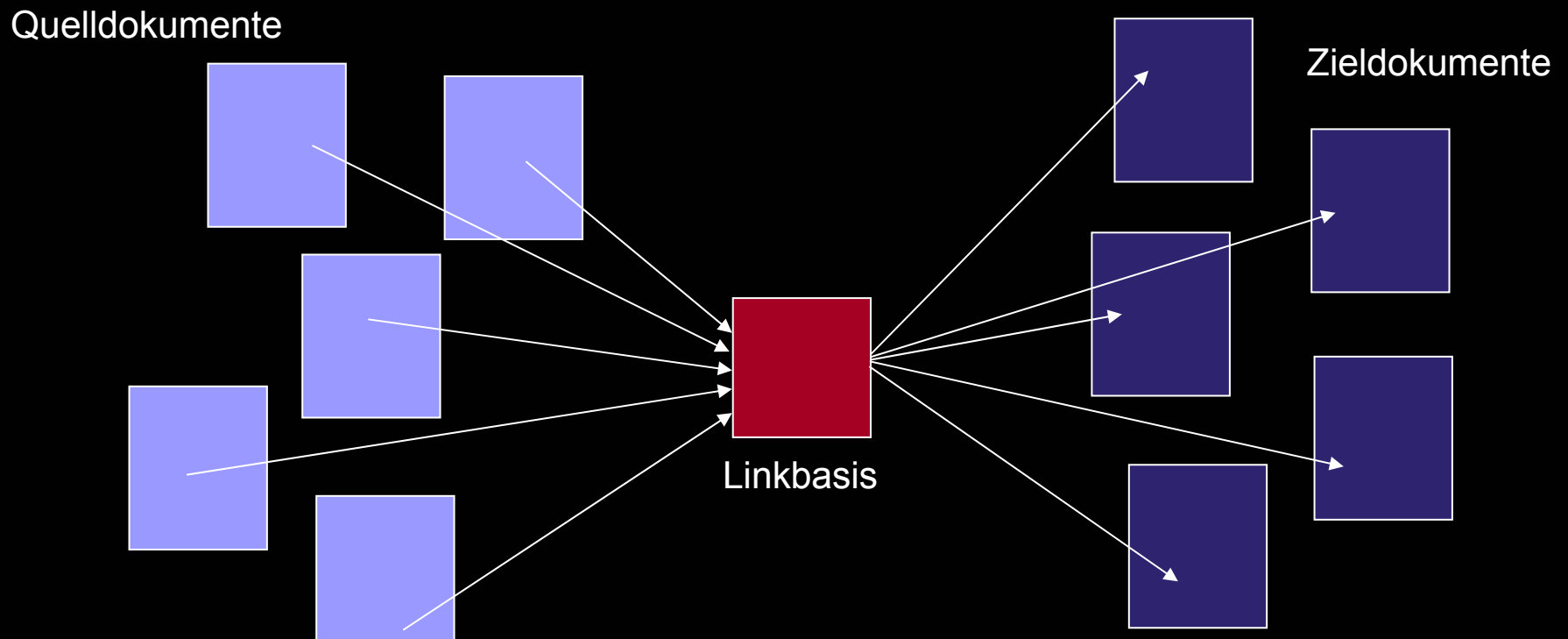


```
<publications
  xmlns:xlink="http//www.w3.org/1999/xlink/"
  xlink:href="publications.xml"
  xlink:role="http://www.bsp.de/bibliography"
  xlink:title="Veröffentlichungen"
  xlink:show="new"
  xlink:actuate="onRequest"
/>
```

2.9 XML und XSL

XLink

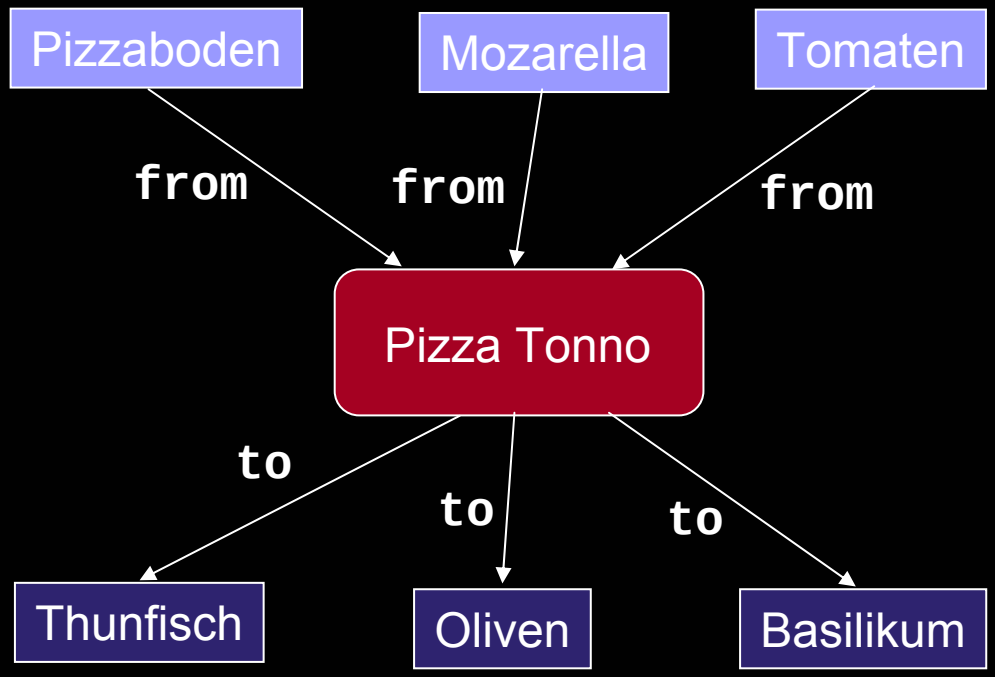
- **XLink – Out-of-Line Links**
  - Realisierung von bidirektionalen **n:m** Linkbeziehungen
  - **Linkbasis** ist nicht Bestandteil von Quell-/Zieldokument



2.9 XML und XSL

XLink

- **XLink – Out-of-Line Links**
  - Beispiel



2.9 XML und XSL

XLink

- XLink – Out-of-Line Links

```
<xlink:extended
      xmlns:xlink="http//www.w3.org/1999/xlink/"
      xlink:role="http://www.pizza.de/pizzaworld"
      xlink:title="Pizza Tonno">
  <xlink:locator        href="Pizzaboden.xml"
                        role="http://www.pizza.de/basis"
                        title="Pizzaboden"/>
        ……..
  <xlink:locator        href="Basilikum.xml"
                        role="http://www.pizza.de/spezial"
                        title="Basilikum"/>
        ……..
  <xlink:arc            from= ="http://www.pizza.de/basis"
                        to= ="http://www.pizza.de/spezial"
                        show="new"
                        actuate="onRequest"/>
<xlink:extended\>
```

2.9 XML und XSL

XML Pointer Language – XPointer

- Festlegung eines **Sprungziels** innerhalb eines Dokuments
  - **HTML**: **<A NAME="…">**
- **Probleme von HTML-Ankern:**
  - Einfügen von Sprungzielen in **schreibgeschützte** und **fremde Dokumente**
  - es muss stets das **gesamte Zieldokument** übertragen werden

2.9 XML und XSL

XML Pointer Language – XPointer

URI#xpointer(ankerbeschreibung)

- Bsp.: `http://www.bsp.de/gedicht.xml#xpointer(zeile2)`
  `http://www.bsp.de/gedicht.xml#xpointer(1/1/2)`

```
<gedicht>
  <strophe ID="strophe1">
    <zeile ID="zeile1">Dreifach ist des Raumes Maß:</zeile>
    <zeile ID="zeile2">Rastlos fort ohn' Unterlaß</zeile"
    <zeile ID="zeile3">Strebt die Länge fort ins Weite,</zeile>
    <zeile ID="zeile4">Endlos gießet sich die Breite,</zeile>
    <zeile ID="zeile5">Grundlos senkt die Tiefe sich</zeile>
  </strophe>
</gedicht>
```

2.9 XML und XSL

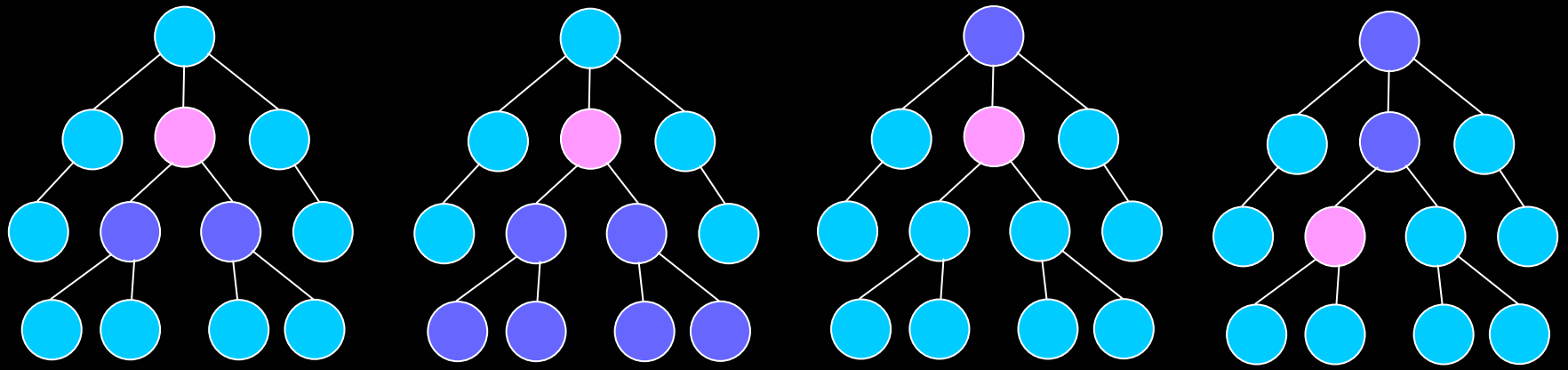
XPath

- Navigation innerhalb des abstrakten Dokumenten-Strukturbaumes

Kontextknoten

adressierte Knoten

`z.B. gedicht.xml#child::gedicht[position()<3]`



2.9 XML und XSL

XSL und XSLT

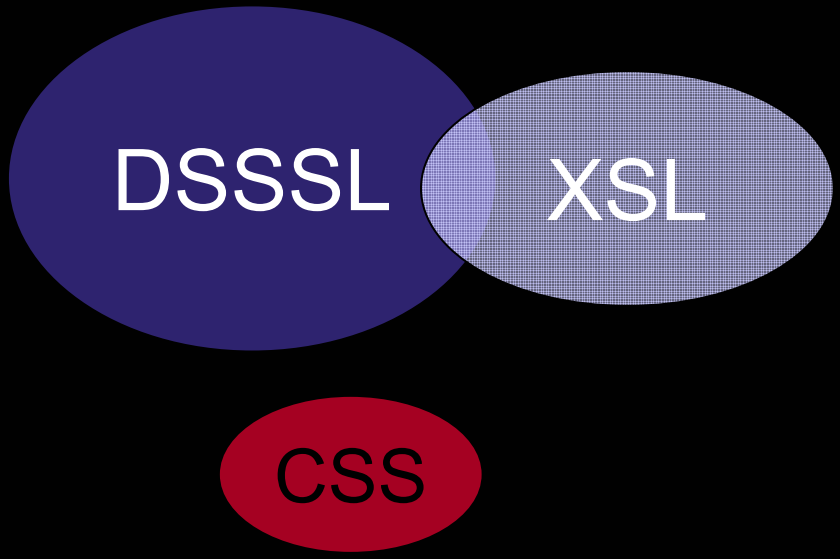
- XML-Dokument ist hierarchisch strukturierte **Informationssammlung**
- vergleichbar mit hierarchischer oder relationaler Datenbank
- über **Transformationen** können XML-Dokumente für eine Vielzahl von Anwendungen nutzbar gemacht werden

  - **Strukturelle Transformationen**
    - Filterung/Sortierung/Umformatierung für andere Anwendungen (XSLT)
  - **Dynamische Dokumente**
    - Oft interaktive Filterung /Reorganisation eines XML-Dokuments (XSLT)
  - **Darstellungstransformationen**
    - Layoutspezifische Transformation über Stylesheets (CSS) oder XSL-FO

2.9 XML und XSL

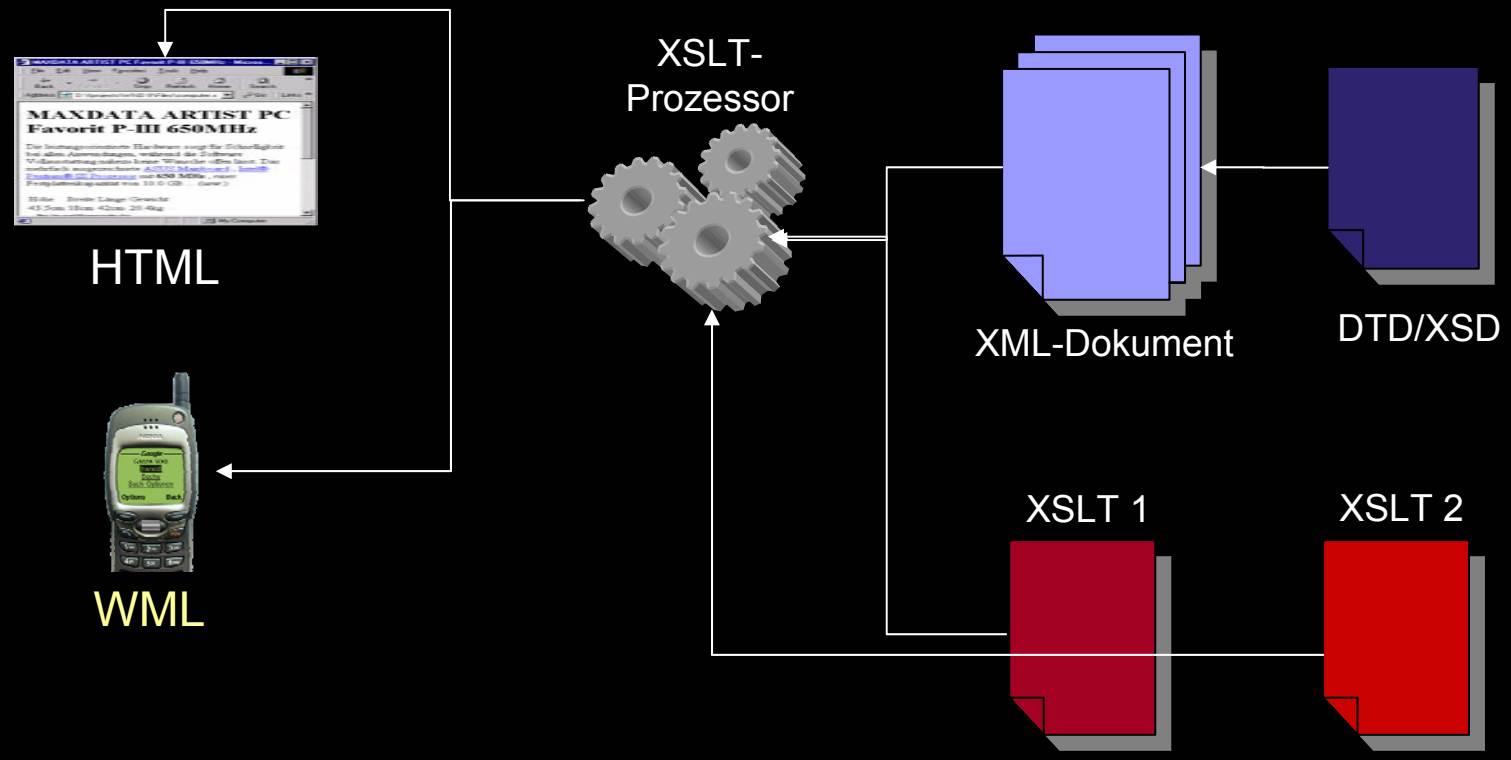
XSL und XSLT

- **Extensible Style Language (XSL)** zur Transformation von XML-Dokumenten ist in drei separate Sprachen aufgeteilt:
  - **XSLT** – Spezifikation der Transformationen
  - **XSL-FO** – Layout-Transformationen (Formatierung) mit XSL
  - **XPath** – Zugriff auf XML-Strukturen



2.9 XML und XSL

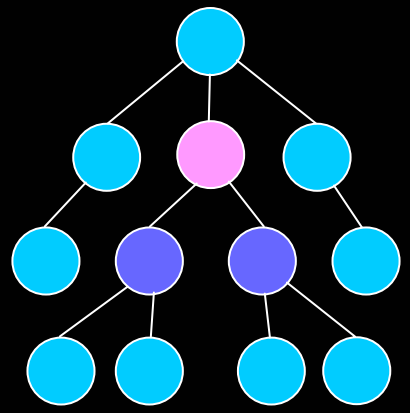
XSL und XSLT



2.9 XML und XSL

XSL und XSLT

- XSL arbeitet auf dem **abstrakten Strukturbaum** des XML-Dokuments (XPath)
- Ein XSL-Stylesheet besteht aus einer Sammlung von Transformationsregeln (**Templates**)
- Strukturbaum wird durchlaufen, für jeden Knoten wird passendes Template aus XSL-Stylesheet gesucht und angewendet



Strukturbaum

+

XSL-Stylesheet

neues Dokument

2.9 XML und XSL

XSL und XSLT

```
<xsl:template      match = Suchabfrage
                   name = Name
                   priority = Prioritität
                   … >
   … Template Instruktionen …
</xsl:template>
```

2.9 XML und XSL

XSL und XSLT

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE movie SYSTEM "Beispiel.dtd">
<movie>
 <title>North by Northwest</title>
 <cast>
 <director>Alfred Hitchcock</director>
  <actor type=“protagonist”>Cary Grant</actor>
  <actor>Eve Marie Saint</actor>
  <actor type=“antagonist”> James Mason </actor>
 </cast>
 <year> 1959</year>
</movie>
```

XML-Dokument

2.9 XML und XSL

XSL und XSLT

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<HTML>
<HEAD>
  <TITLE> <xsl:value-of select="movie/titel" /> </TITLE>
</HEAD>
<BODY>
<H1> <xsl:value-of select="movie/titel" /> </H1>
<P> Darsteller: </P>
<UL>
  <xsl:for-each select="movie/cast/actor">
  <LI> <xsl:value-of /> </LI>
  </xsl:for-each>
</UL>
</HTML>
</xsl:template>
```

XSL-Stylesheet

2.9 XML und XSL

XSL und XSLT

```
<HTML>
<HEAD>
  <TITLE> North by Northwest </TITLE>
</HEAD>
<BODY>
<H1> North by Northwest </H1>
<P> Darsteller: </P>
<UL>
  <LI> Cary Grant</LI>
  <LI> Eve Marie Saint</LI>
  <LI> James Mason</LI>
</UL>
</HTML>
```

Ergebnis der XSL-Transformation

2.9 XML und XSL

XML Derivate

- **Wireless Markup Language - WML**
  - Teil des **WAP**-Standards (Wireless Access Protocol)
  - Teilmenge von **XHTML**
  - Webseite wird als **Karteikarten-stapel** dargestellt

```
<wml>
 <head> … </head>
 <card>
  …
  </card>
  <card>
  …
  </card>
</wml>
```

2.9 XML und XSL

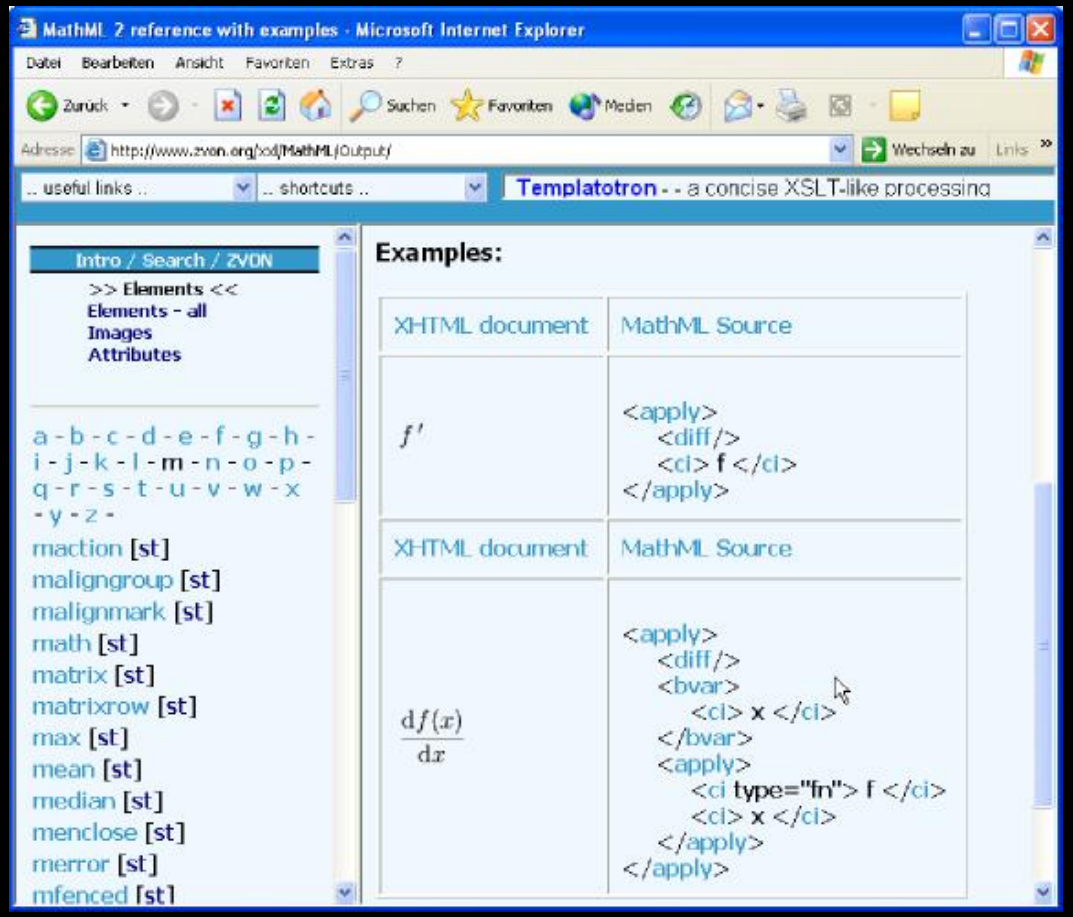
XML Derivate

- **Voice XML**
  - Zur telefonischen Ausgabe von XML-Dokumenten
- **Scalable Vector Graphics (SVG)**
  - Kodierung von 2-dim Vektorgrafiken
- **Vector Markup Language (VML)**
- **Hyper Graphics Markup Language (HGML)**
  - Grafikaufbereitung für mobile Endgeräte
- **Synchronized Multimedia Integration Language (SMIL)**
  - Synchronisation von Multimedia-Datenströmen

2.9 XML und XSL

XML Derivate

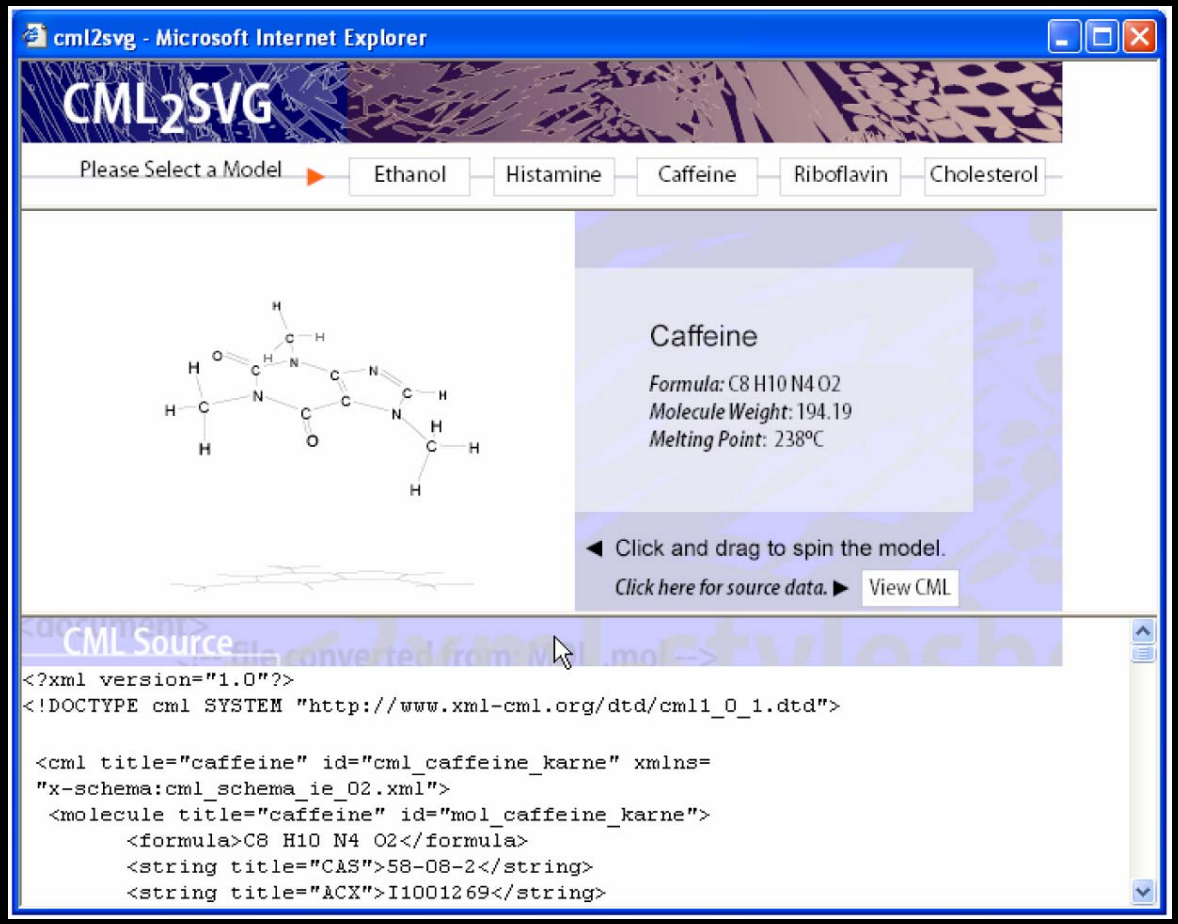
- MathML



2.9 XML und XSL

XML Derivate

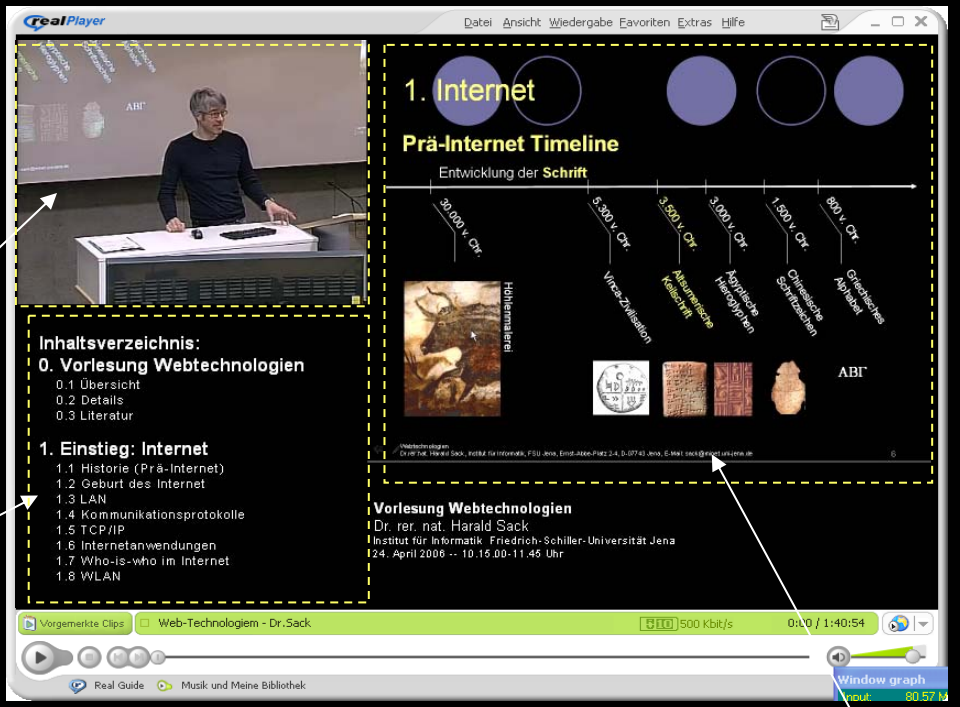
- Chemical Markup Language (CML)



2.9 XML und XSL

XML Derivate

- **Synchronized Multimedia Integration Language (SMIL)**



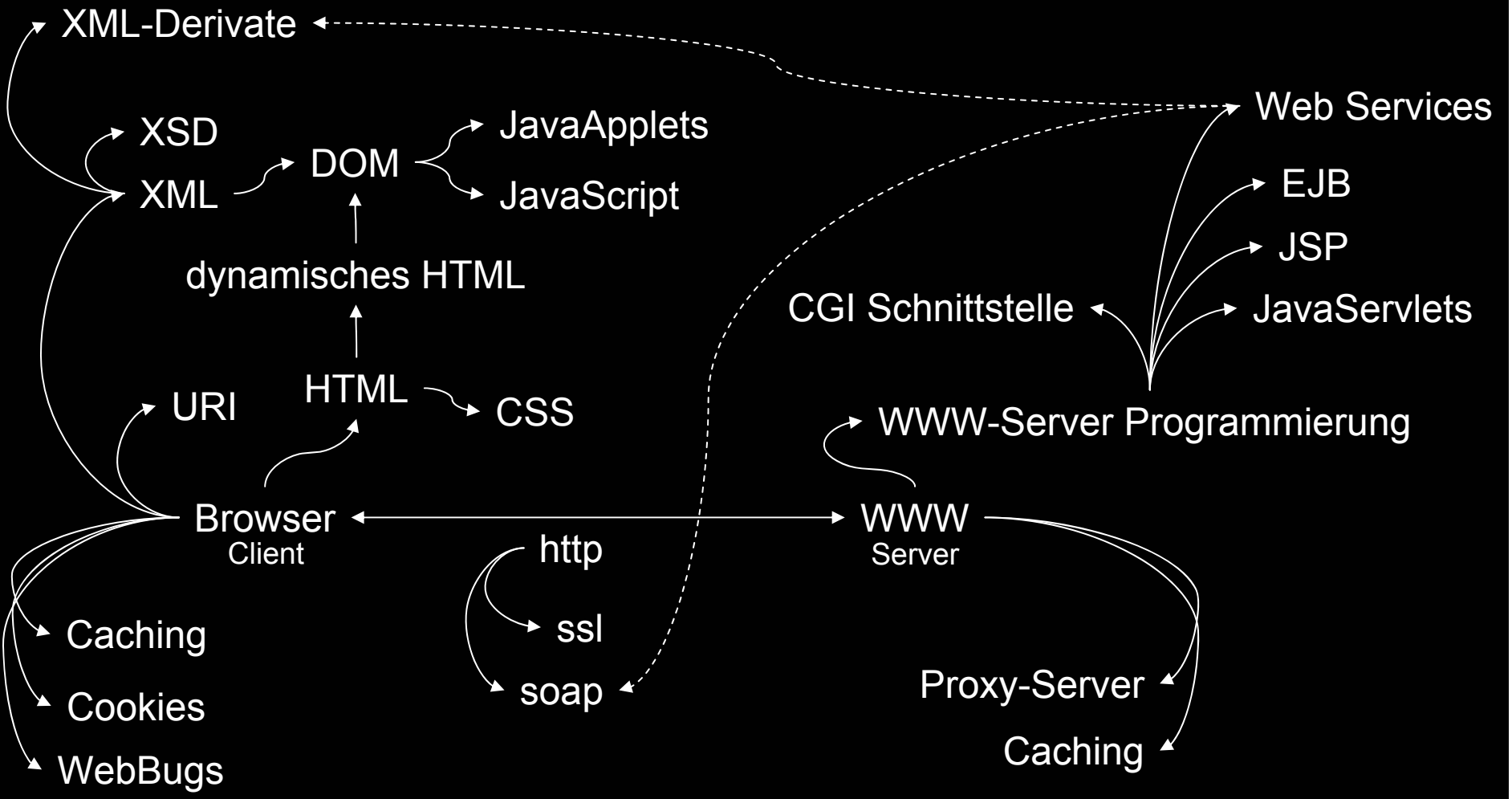
2.9 XML und XSL

XML Derivate

- **Synchronized Multimedia Integration Language (SMIL)**

```
<smil>
 <head>
  <layout>
   <root-layout width="832" height="516" background-color="black" />
   <region id="desktop" left="320" top="0" width="512" height="384“ … />
   <region id="text" left="20" top="246" width="300" height="250" … />
   <region id="video" left="0" top="0" width="320" height="240" />
  </layout>
 </head>
 <body>
  <par>
   <video id="video1" src="video_20060424.rm"  region="video" />
   <video id="desktop1" src="desktop_20060424.rm" begin="id(video1)(2s)" region="desktop" />
   <text src="text1_20060424.rt" region="text" />
   …
  </par>
 </body>
</smil>
```

2. World Wide Web



2. World Wide Web

Literatur

- Ch. Meinel, H. Sack:
  WWW – Kommunikation, Internetworking, Web Technologien, Springer, 2004.

- D. Wessels:
  Web Caching, O'Reilly, 2001.

2. World Wide Web

Literatur

- R. Eckstein, S. Eckstein:
  XML und Datenmodellierung, dPunkt Verlag, 2004.