

Hochschule Darmstadt

Fachbereich Informatik

Entwicklung webbasierter Anwendungen



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

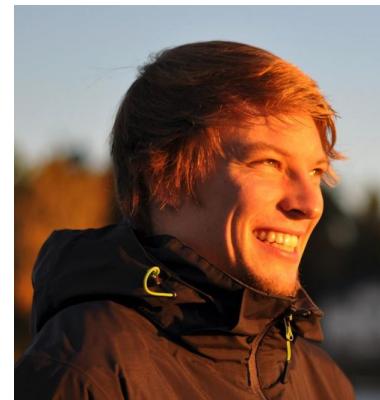
Organisatorisches Vorstellung

Jakob Oesterling, B.Sc.



Masterstudent Informatik (h_da)

Thomas Sauer, B.Sc.



Masterstudent Informatik (h_da)

Patrick Rath, B.Sc..



Masterstudent Informatik (h_da)

Fraunhofer SIT

Freier Softwareentwickler
Onmatic GmbH

Software-Trainee bei Accso
GmbH





Organisatorisches Kommunikation

- Kommunikation an einem Ort:

ewahda.slack.com

- Anmeldung mit @stud.h-da.de Adresse (wichtig!)
 - Bitte Klarnamen eintragen (in den Einstellungen)
- Absprachen, Ankündigungen, Fragen, Diskussion, Dateien



Kommunikation

- Dieses Skript weicht (leicht) vom Skript der letzten Semester ab
 - Neuere Beispiele
 - Veränderte Diagramme und Grafiken
 - Inhaltlich größtenteils gleich
 - Neue und abgeänderte Folien werden mit einer Markierung versehen (pentagon)
- Ablauf der ersten Vorlesungen:
 - Einstimmung von Dozenten und Studierenden
 - Direktes Feedback (von beiden Seiten) am Ende der Stunde

Welche Vorkenntnisse habt ihr bereits?
Besondere Wünsche für den Gastvortrag?



Hochschule Darmstadt

Fachbereich Informatik

1. Einleitung



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Zielsetzung

■ aus der Modulbeschreibung:

- ⇒ Die Studierenden sollen
 - Aktuelle Auszeichnungssprachen kennen und anwenden
 - Skriptsprachen für client- und serverseitige Webprogrammierung anwenden
 - ein **Dokument Objekt Modell** verstehen
 - die Architektur webbasierter Client/Server-Anwendungen mit Datenbankanbindung verstehen
 - Methoden und Techniken zur Entwicklung webbasierter Anwendung
 - Sicherheitsaspekte im Kontext von Webanwendungen verstehen
- ⇒ Konkret: Nach der Veranstaltung...
 - kennt Ihr den Sinn, Zweck und die Grenzen der verschiedenen Techniken
 - versteht Ihr das Zusammenspiel der verschiedenen Techniken
 - kennt Ihr die wesentlichen Standards
 - seid Ihr in der Lage, komplexe und standardkonforme Webseiten zu erstellen
 - habt Ihr die Grundlagen, um sich in diverse andere Web-Techniken einzuarbeiten

1. Einleitung

Konkrete Inhalte des Veranstaltung

- Entwurf
- HTML Grundlagen
- Formulare und Layout
- CSS und dynamisches Layout
- ECMAScript, DOM, AJAX
- Webserver Konfiguration (Apache), CGI
- Objektorientiertes PHP, MVC Framework
- PHP mit Datenbankanbindung (MySQLi)
- HTTP
- Sicherheit
- Professionelle Webentwicklung (Entwicklung, Test, Web-Projektverwaltung uvm.)

Die verschiedenen Themen werden nicht vollständig behandelt – es geht in EWA "nur" um die Grundlagen !

1. Einleitung

Aufgabe im Praktikum: Pizzaservice



Bestellung

Margherita	4,00 €
Salami	4,50 €
Hawaii	5,50 €
14.00 €	

Margherita
Salami
Hawaii

Meine Lieferadresse

Kunde

bestellt im Ofen fertig unterwegs

Margherita	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Salami	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Tonno	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Hawaii	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

[Neue Bestellung](#)

Bäcker

bestellt im Ofen fertig

Margherita	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Margherita	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Hawai	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>

Fahrer

Müller, Freßgasse 11, 65000 Frankfurt

Tonno, Calzone, Margherita, Hawaii, Tonno

Preis: 13,00 €

gebacken unterwegs ausgeliefert

Meier, Hauptstr. 5

Tonno, Tonno, Margherita

Preis: 10,50 €

gebacken unterwegs ausgeliefert

1. Einleitung

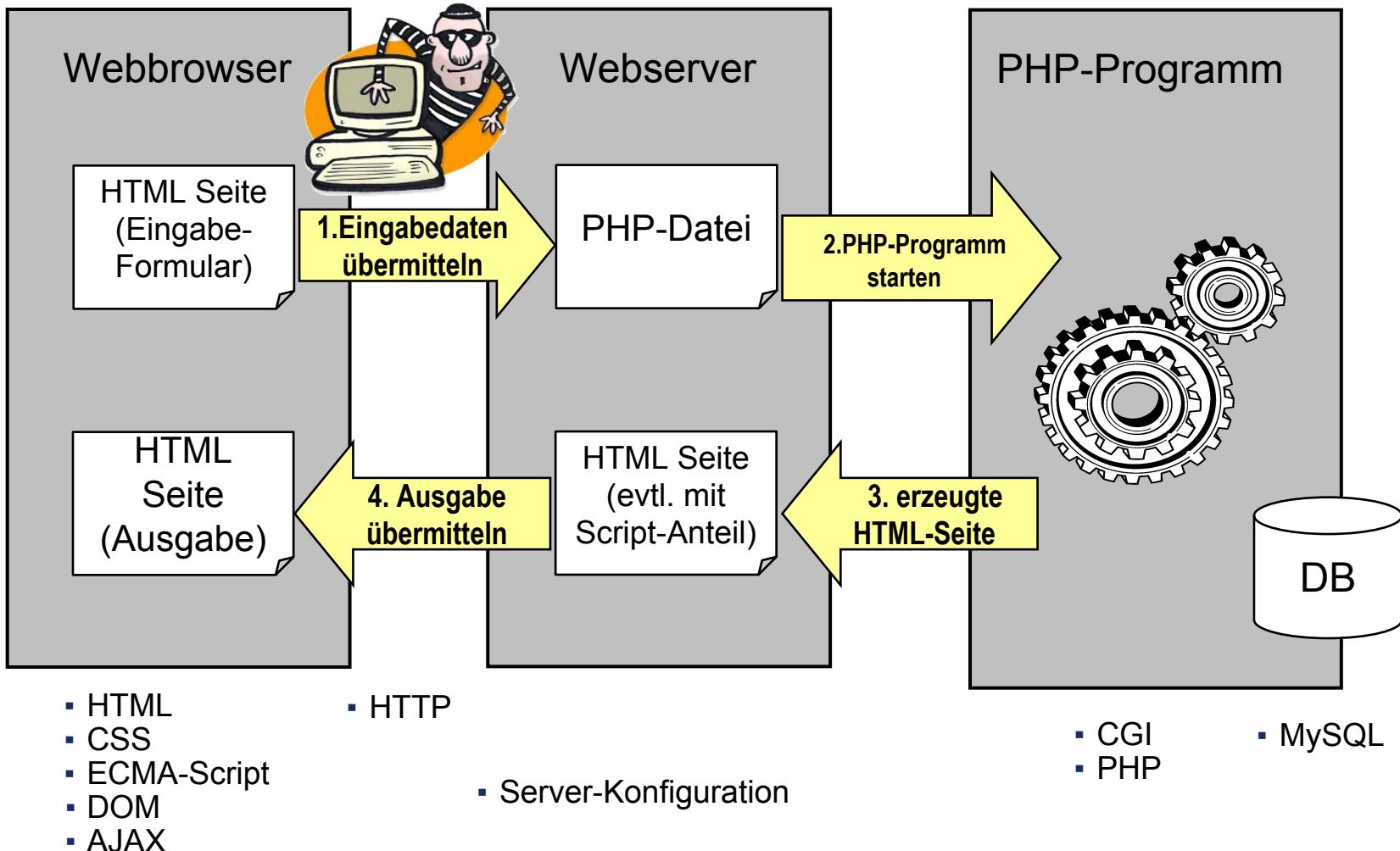
Praktikumsregeln

- Anwesenheitspflicht (falls noch nicht komplett testiert)
- Abnahme max. 10 Minuten pro Gruppe
- 2er Gruppen (max. eine 3er Gruppe bei ungerader Anzahl)
- Beide Praktikumspartner müssen den Code kennen
- Falls ein Praktikumstermin schief läuft: Vorzeigen beim nächsten Termin
- Abnahme von mehreren Praktika mit vorheriger Absprache

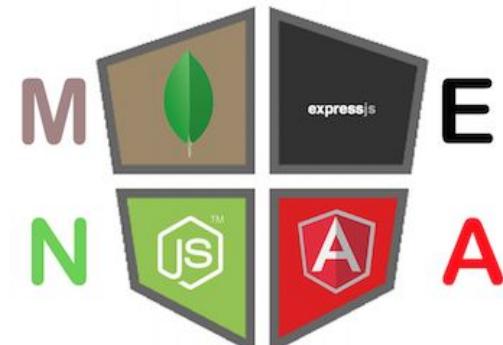
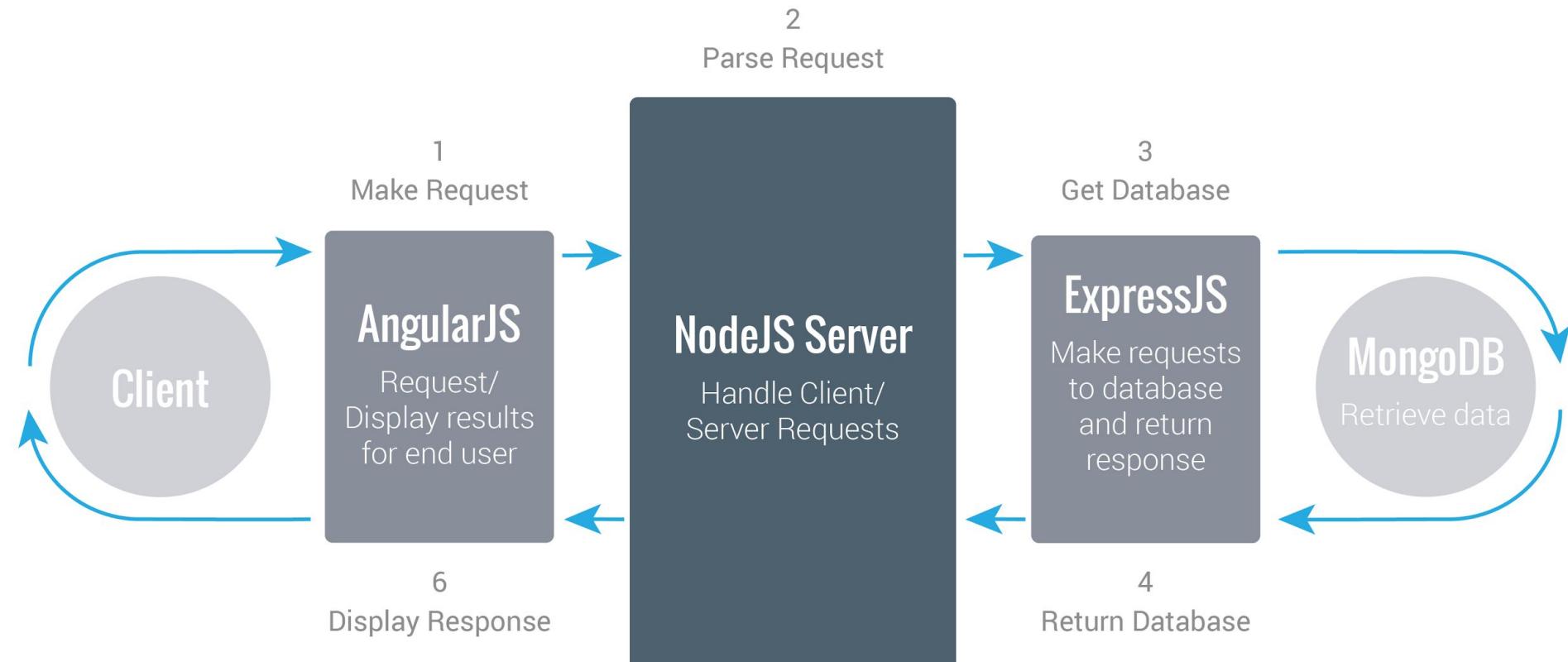


1. Einleitung

Einsatz der Technologien im Zusammenhang



Alternativer Einsatz mit dem MEAN Stack



1. Einleitung

Webquellen und Software

Webquellen

- ⇒ Stefan Münz:
<http://webkompetenz.wikidot.com/docs:html-handbuch>
- ⇒ Damir Enseleit: SELFPHP
<http://www.selfphp.info>
- ⇒ MDN: JavaScript Guide
<https://developer.mozilla.org/en/JavaScript/Guide>
- ⇒ Online-Bücher als PDF
<http://www.oreilly.de/online-books>
 - K. Janowicz: "Sicherheit im Internet"
 - S. Kersken: "Praktischer Einstieg in MySQL mit PHP"
 - uvm.

Standards

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

Freie Software, Dokus, Tutorials

- ⇒ Tutorials und Referenzen
w3schools.com
- ⇒ HTML Editor + Validator
<http://www.phase5.info/>
<http://notepad-plus-plus.org/>
<http://brackets.io/>
<https://atom.io/>
<https://www.jetbrains.com/phpstorm/>
- ⇒ XAMPP (Webserver, MySQL, PHP)
<http://www.apachefriends.org/de/xampp.html>

1. Einleitung

Webquellen und Software

Webquellen

- ⇒ Allgemeine Einführung in **Web-Entwicklung**
 - <https://developer.mozilla.org/en-US/Learn>
- ⇒ **HTML** Referenz von Mozilla
 - <https://developer.mozilla.org/de/docs/Web/HTML/Reference>
- ⇒ sehr gute **CSS** Reference von Sara Soueidan
 - http://tympanus.net/codrops/css_reference/
- ⇒ **JavaScript** Referenz und Einführung
 - <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- ⇒ **PHP** Offizielle Doku, Beispiele, Referenzen
 - <http://php.net>

Tools

- ⇒ **Chrome/Firefox Inspector**
 - Shift+Cmd+i bzw alt+cmd+i: Werkzeug zum Live “Debugging” von Webseiten
- ⇒ <http://codepen.io/> und <https://jsfiddle.net/>
 - Code-Snippets austauschen, anzeigen, debuggen



Webquellen

Weitere Tools

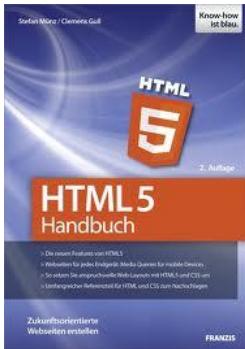
- ⇒ Sublime Text - guter **Editor** zum Erstellen von HTML/CSS/JS/PHP
- ⇒ Atom - weiterer sehr moderner Editor

Interessante Links/Blogs zum Thema Web-Entwicklung

- ⇒ scotch.io
- ⇒ codrops.com
- ⇒ wdrl.info



1. Einleitung Literatur



Eric Freeman und Elisabeth Robson,
"HTML5-Programmierung von Kopf bis Fuß",
O'Reilly; 2012



Stefan Münz, Clemens Gull,
"HTML 5 Handbuch", 2. Auflage
Franzis Verlag GmbH, 2012



Carsten Möhrke,
"Besser PHP programmieren",
Galileo Computing, 2009

Mark Lubkowitz,
"Webseiten programmieren und gestalten",
Galileo Computing, 2007



Sverre H. Huseby,
"Sicherheitsrisiko Web-Anwendung",
dpunkt.verlag, 2004



Hochschule Darmstadt

Fachbereich Informatik

1.1 Softwaretechnik für webbasierte Anwendungen



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Motivation

Das Thema kommt im Kapitel "Professionelle Webentwicklung" später noch mal ausführlicher!

- auch webbasierte Anwendungen sind Softwaresysteme !
 - ⇒ es gilt weiterhin alles, was man über Softwaretechnik, Software Ergonomie und GUIs gelernt hat
 - ⇒ nicht vor lauter
 - Design* ⇒ Grafik, Animation, Gimmicks
 - den
 - Entwurf ⇒ Analyse, Architektur, Datenorganisation
 - vergessen !
- Die Programmiersprachen, -umgebungen und die Aufgaben verleiten oft zum Hacken !
- Ein Konzept hat aber noch nie geschadet..... und ein bewusstes Vorgehen auch nicht !

Anforderungsanalyse: Funktionale Anforderungen

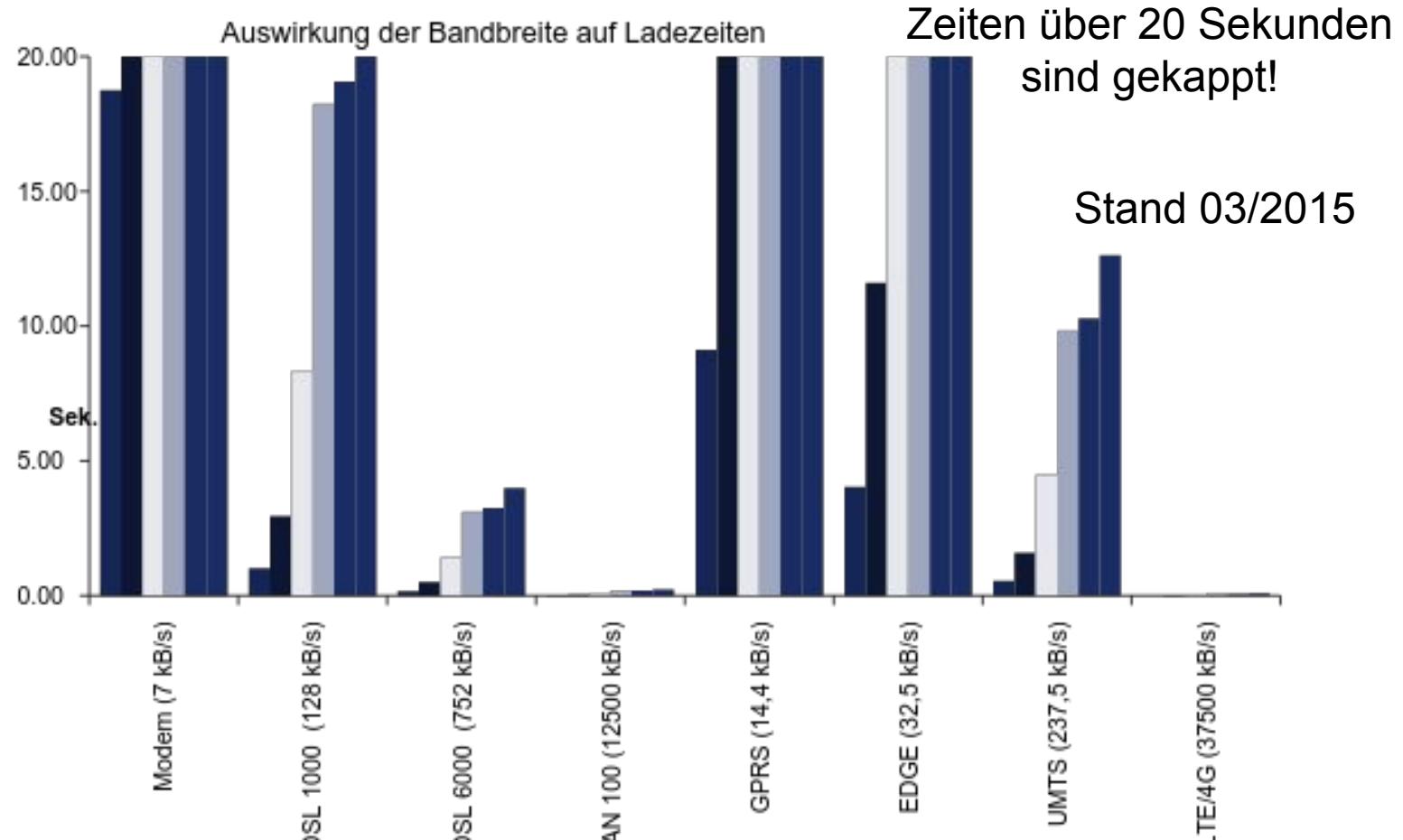
genau wie in
Entwicklung
nutzerorientierter
Anwendungen

- 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, Sprache, Ausstattung, Ausbildung,
PC-Erfahrung, Internet-Zugang, Benutzungsfrequenz
- Verkaufsargument
 - ⇒ Nutzen, Mehrwert ggü. Konkurrenz (z.B. Printmedien ↑, lokaler Software)
- Erscheinungsbild
 - ⇒ Reine Information oder auch Darstellung?
- Zielmedium
 - ⇒ Internet, Intranet, Laptop, Smartphone, Smartwatch, Tablet, Fernseher

Anforderungsanalyse: 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

Nicht-funktionale Anforderungen: Problem 1: Transferraten



⇒ Man muss (besonders für mobile Anwendungen) immer noch mit Datenvolumen geizen!

Einschub zeigen der Datenverbindungen

Nicht-funktionale Anforderungen: Problem 2: Plattformabhängigkeit

"Plattform" traditionell: Betriebssystem

- ⇒ Unix-Varianten (Workstation), Windows (PC),
OS X (Mac), MVS (Mainframe) etc.
- ⇒ oft nur für bestimmte Hardware verfügbar

lediglich eine
Verlagerung der
Abhängigkeit

"Plattform" im Web: Browser + Version

- ⇒ Internet Explorer, Firefox, Safari, Chrome etc.
- ⇒ Erhebliche Unterschiede durch
 - verzögerte oder spezielle Umsetzung der Standards
 - Plug-Ins (z.B. Flash, nicht für alle Systeme verfügbar)
 - Auflösung und spezielle Bedienelemente (z.B. Smartphone mit Touchscreen)

⇒ Es ist schwierig, eine Webseite für die verschiedenen Zielsysteme zu entwickeln!

Nicht-funktionale Anf.: Problem 3: what you see is NOT what you get !

■ Darstellung erfolgt über HTML

- ⇒ HTML ist eine Auszeichnungssprache (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

⇒ Das Aussehen einer Webseite muss auf vielen Zielsystemen geprüft werden!

Nicht-funktionale Anforderungen: Problem 4: Auslieferbarkeit

- Im Internet ist es üblich, dass neue Anwendungen einfach als neue Webseite "ausgeliefert" werden
 - ⇒ es gibt selten Ankündigungen und keine Installationsprozeduren
 - ⇒ die Anwendung muss mit diversen Browsern, Versionen und Endgeräten laufen
 - ⇒ es darf zu keinen Inkompatibilitäten mit Daten von vorhergehenden Versionen kommen
 - in den Browsern gespeicherte Daten (Cookies)
 - auf dem Webserver gespeicherte Daten (z.B. Einträge in einer Datenbank)
 - ⇒ Daten aus alten Versionen dürfen nicht verloren gehen

⇒ Das Ausliefern einer Webseite ist ein kontinuierlicher Prozess, der viel Disziplin erfordert!

Design

■ Festlegungen

- ⇒ Laufzeitumgebung
(z.B. Datenbank, Webserver, Browser, optimale Auflösung, ...)
- ⇒ Entwicklungsumgebung
(z.B. Programmiersprache mit Version, Frameworks, Testtools,...)
- ⇒ Verwendung von Standards
(z.B. HTML5, XHTML1.1, strict)
- ⇒ Referenztests
(z.B. HTML-Validator bei W3C)
- ⇒ Konventionen für die Fehlersuche (Logging, Debugging)
- ⇒ usw.

■ Klassisches Design

- ⇒ Definition von Klassen mit Verantwortlichkeiten
- ⇒ Interaktion der Klassen
- ⇒ Verteilung der Anwendung auf Webbrowser, Webserver

Test

- Eigentlich ganz normale Tests !
 - ⇒ Funktionalität
 - ⇒ nicht-funktionale Anforderungen (Unbedingt die 4 Probleme testen!)
 - ⇒ Ausnahmefälle
 - ⇒ möglichst viel automatisiert
- Besonders wichtig
 - ⇒ Verständlichkeit und Bedienbarkeit
 - mit dem Auftraggeber bzw. künftigen Benutzern
 - ⇒ Portabilität
 - Browser, Plugins, CPU, Bildschirme, Datenverbindung
 - ⇒ Performanz
 - verschiedene Browser, Ein- und Ausgabegeräte
 - geringe Transferrate
 - hohe Benutzerlast

Sinnvolle Arbeitsergebnisse (Meilensteine)

- Visionsdokument mit Sinn und Zweck der Website
- Übersicht der Funktionalität (z.B. Use Case Diagramm)
- Beschreibung der nicht-funktionalen Anforderungen
- Navigationsübersicht (Zustandsdiagramm)
- Screen-Diagramme (Skizzen der Seiten)
- Modellierung der SW-Struktur
 - ⇒ Aktivitätsdiagramm
 - ⇒ Klassendiagramm
 - ⇒ Sequenzdiagramm
 - Zusammenspiel von Client (HTML-Seiten) und Server (z.B. PHP)
 - ⇒ Komponentendiagramm
 - Verteilung auf Webserver, Client etc.
- Testplan

vgl. "Entwicklung nutzerorientierter Anwendungen" im 3. Semester

Hochschule Darmstadt

Fachbereich Informatik

1.2 Ergonomie für webbasierte Anwendungen



h_da

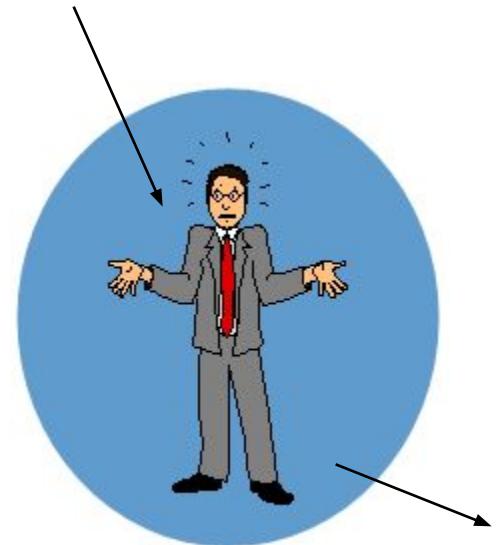
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Mensch-Maschine-Schnittstelle

- Der Systemzustand muss auf einen Blick erkennbar sein
(ohne ihn dabei zu verändern!)
 - ⇒ Grund-Forderung aus der Software-Ergonomie
- 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

- **Orientierungskomponente** 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



1.2 Ergonomie für webbasierte Anwendungen

Layoutbeispiel: Bundesrat

Bundesrat

Deutsch | English | Français |
Seitenübersicht | Hilfe | Impressum | Datenschutz |

Sie sind hier: Startseite www.bundesrat.de > Service

Struktur und Aufgaben
Organe und Mitglieder
Bundesrat / Bundestag
Gremien und Konferenzen
Parlamentsmaterialien
Termine und Veranstaltungen
Presse
Service
Bundesrat aktuell - Archiv
Besuch beim Bundesrat
Informationsmaterial
Bundesrat und Schule
Stellenangebote
Öffentliche Ausschreibungen
E-Mail Abonnement
RSS-Nachrichtendienst
Abkürzungen
Anschrift und Anfahrt
Kontaktformular
Erweiterte Suche
Suche
Alle Bereiche
Suche starten

Service

In dieser Rubrik finden Sie unter anderem das Archiv aktueller Meldungen sowie Informationen zu Stellenausschreibungen, zu Ausschreibungen des Sekretariats des Bundesrates und zu folgenden Themen:

Besuch beim Bundesrat
Der Besucherdienst bietet verschiedene Veranstaltungen an, bei denen Sie die Arbeit des Bundesrates kennen lernen und das historische Gebäude besichtigen können.

Informationsmaterial / Literaturtipps
In verschiedenen Publikationen informiert der Bundesrat über seine Aufgaben und Funktionen. Für Jugendliche und Multiplikatoren werden spezielle Broschüren und andere Medien angeboten. Das Informationsmaterial kann kostenlos heruntergeladen oder beim Bestellservice angefordert werden.

In den Literaturtipps veröffentlicht die Bibliothek Leseempfehlungen zu aktuellen Themen mit Bezug zum Bundesrat.

Seite drucken | Seite empfehlen

Top-Ergebnis
beim
BITV Test

Hintergrund

Orientierung

Präsentation

Navigation

Barrierefreie Webseiten

In Deutschland
gilt die BITV

(Barrierefreie Informationstechnik-Verordnung)

- Webseiten sollen auch für Menschen mit einer Behinderung zugänglich sein
 - ⇒ Sehschwächen (z.B. Rot-Grün-Blindheit) aber auch Hör-, Lern-, Lese-, motorische Schwächen uvm.
 - ⇒ insbesondere soll eine Webseite vorgelesen werden können
 - Screenreader lesen die Seite mit Text2Speech vor
 - Braille-Zeilen geben Blindenschrift zeilenweise aus
 - Tabellen werden von links nach rechts und von oben nach unten gelesen
 - Bilder und Videos sind als solche nicht darstellbar
 - Der Inhalt muss logisch gruppiert und angeordnet sein – und nicht nach der Anordnung auf dem Bildschirm
 - ⇒ Die Bedienung muss mit vereinfachten Tastaturen möglich sein



Textausgabe mit Braille-Zeile
Bild: SBV



www.computer-fuer-behinderte.de

Responsiveness

■ Ausgabe auf vielen Unterschiedlichen Displaygrößen

- ⇒ Smartwatch
- ⇒ Smartphone
- ⇒ Tablet
- ⇒ Desktop
- ⇒ Smart TV

Beispiele

+++ http://www.ard.de/home/ard/ARD_Startseite/21920/index.html

+++ <https://www.h-da.de/?1>

--- <http://www.heise.de/>

Hochschule Darmstadt

Fachbereich Informatik

2. Webclient



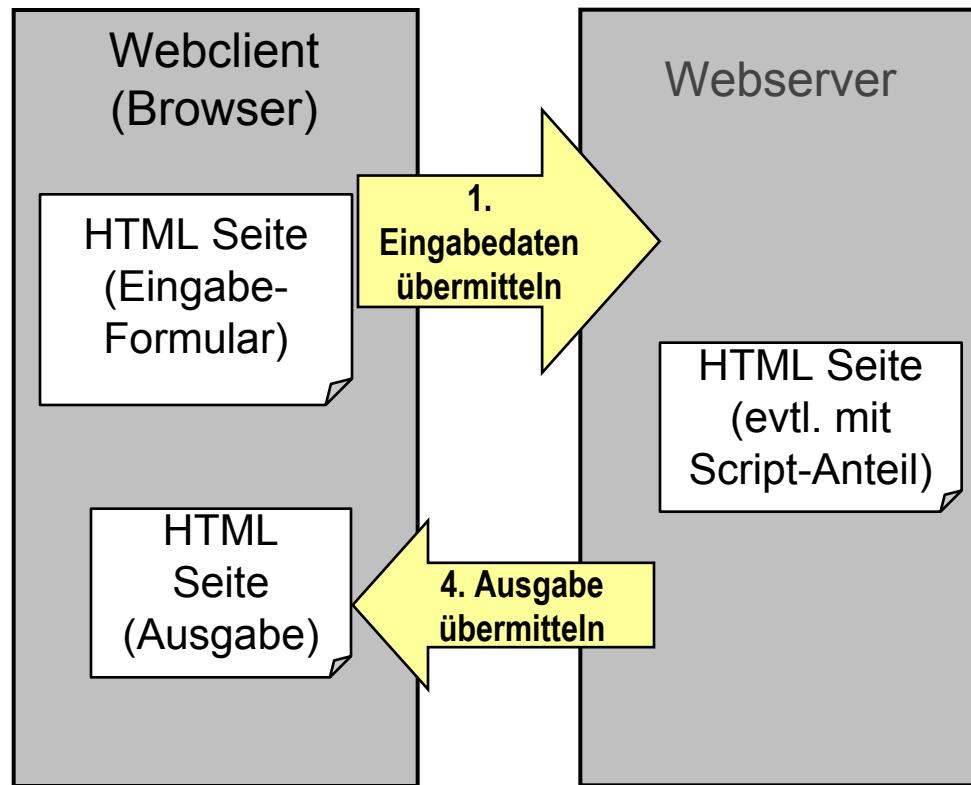
h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

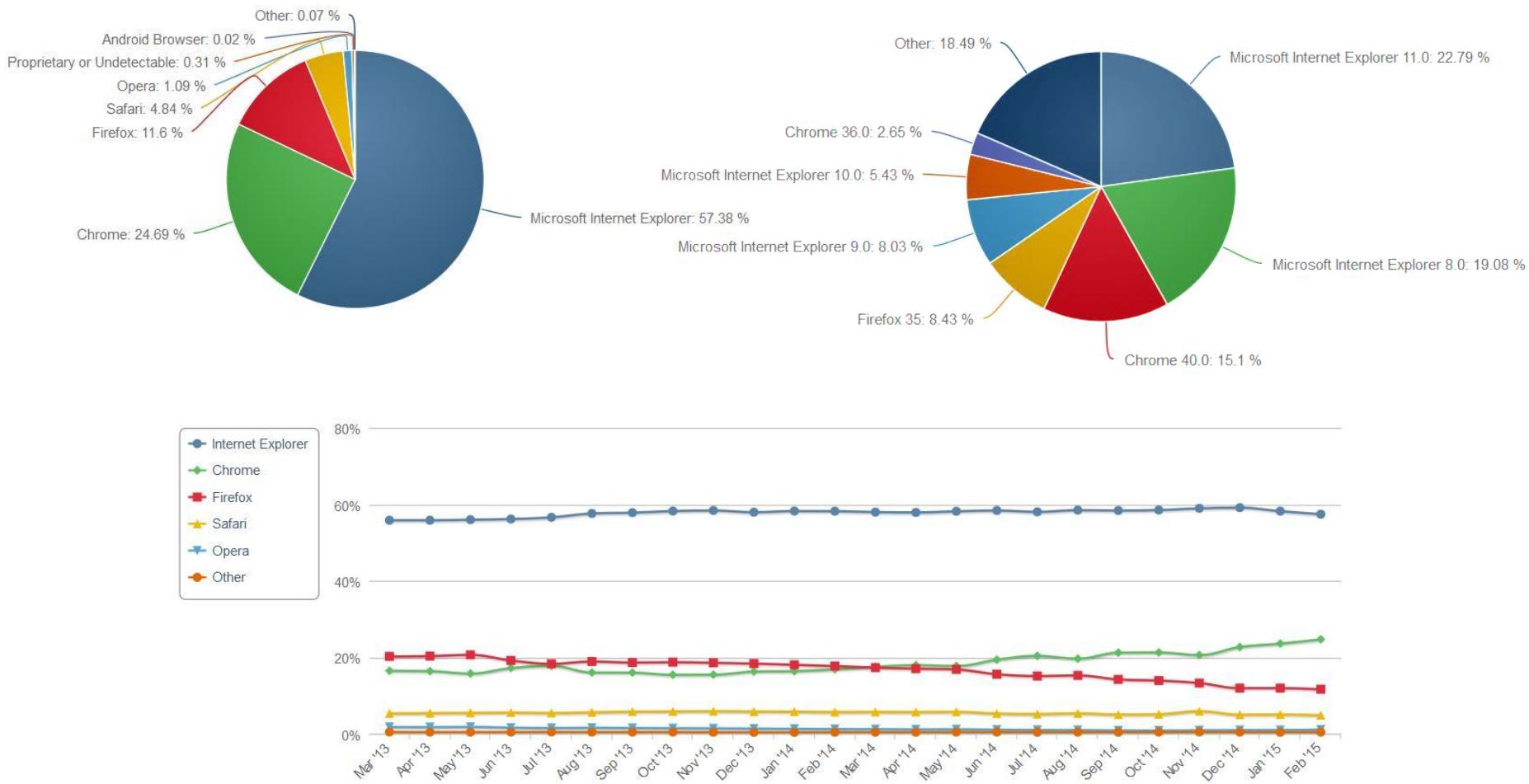
Der Webclient



- HTML
- CSS
- ECMA-Script
- DOM
- AJAX

2. Webclient

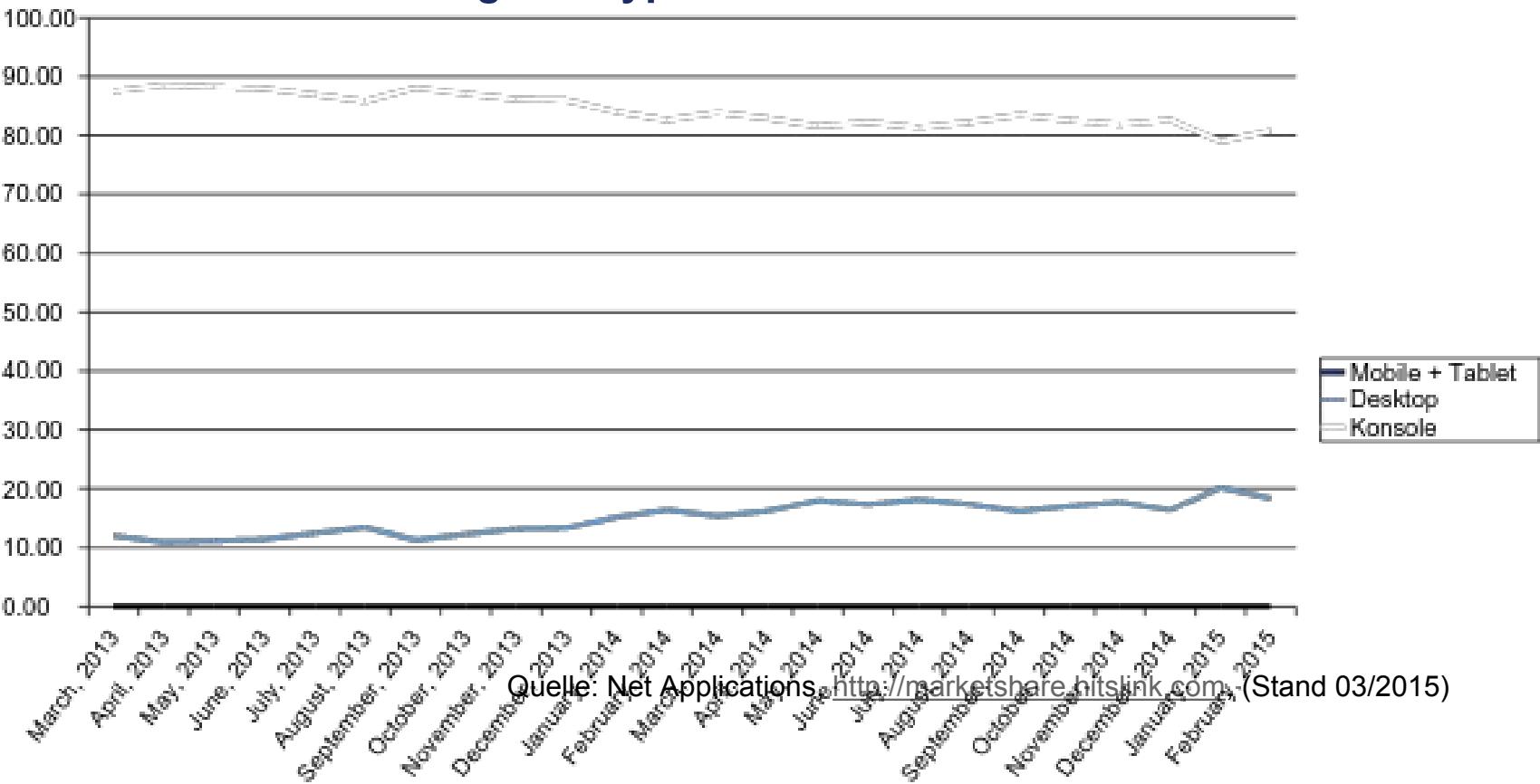
Marktanteile und Trend der Browser für Desktops



Quelle: Net Applications, <http://marketshare.hitslink.com>, (Stand 03/2015)

2. Webclient

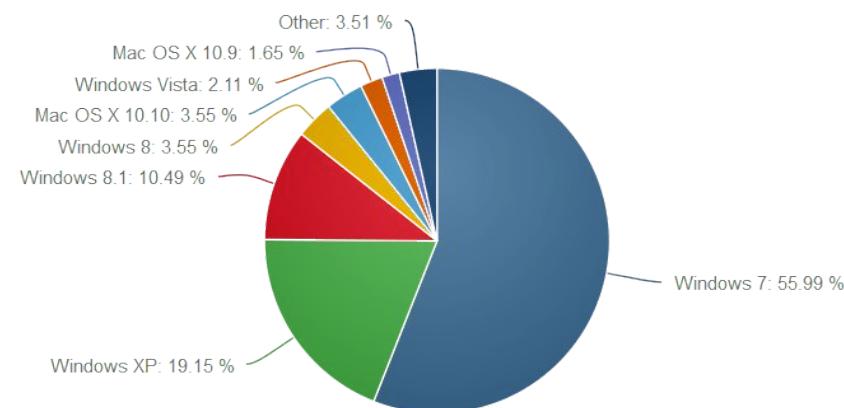
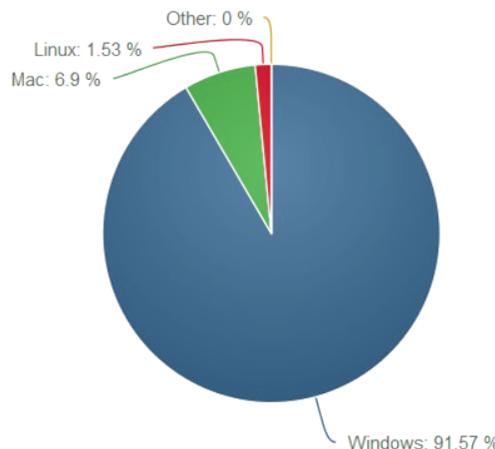
Browseranteil nach Endgerätetyp



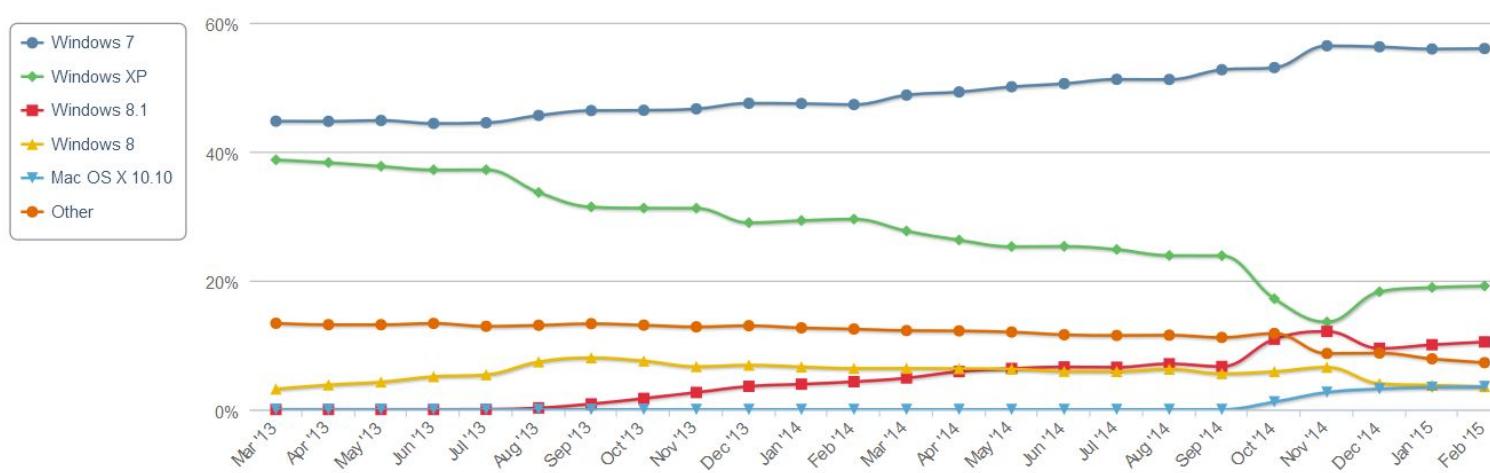
Ca. 20% der Internetzugriffe kommt 2015 von mobilen Geräten.
Die Tendenz ist stark steigend. 2011 waren es nur 4%.

2. Webclient

Marktanteile und Trend der Betriebssysteme von Desktop - Webclients

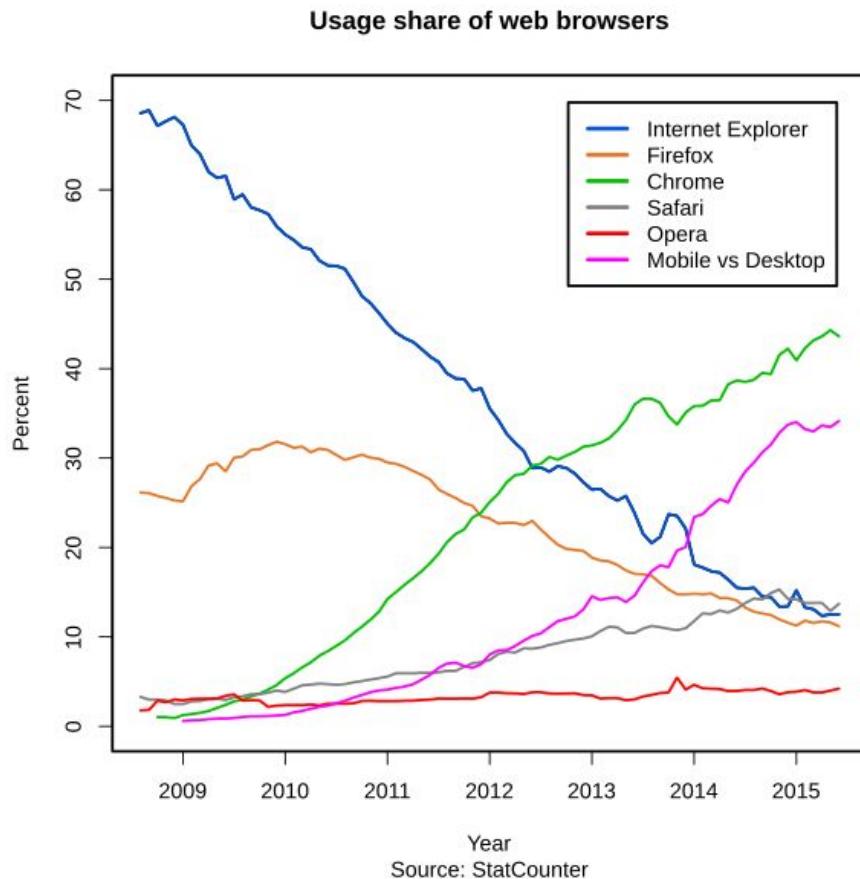


Quelle: Net Applications, <http://marketshare.hitslink.com>, (Stand 03/2015)



Fazit: Windows und der Internet Explorer sind für Webanwendungen Pflicht!

Marktanteile und Trend der Web-browser



Source: https://upload.wikimedia.org/wikipedia/commons/thumb/8/86/Usage_share_of_web_browsers_%28Source_StatCounter%29.svg/600px-Usage_share_of_web_browsers_%28Source_StatCounter%29.svg.png

Hochschule Darmstadt

Fachbereich Informatik

2.1 HTML



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Was ist HTML ?

- Markup Language
 - ⇒ Markup Language: Auszeichnungssprache
 - ⇒ markiert und attributiert Bestandteile eines Fließtexts
 - ⇒ Layout bewusst nicht definiert
 - ⇒ Browser setzen Auszeichnung in visuelle Darstellung um
- HyperText: Verweise auf andere Dokumente
 - ⇒ komfortable Querverweise zu anderen Stellen im eigenen Projekt oder zu beliebigen anderen Dokumenten im Web
 - ⇒ URL (Uniform Resource Locator)
- Text-Dateien (kein Binärformat)
 - ⇒ mit jedem Texteditor zu bearbeiten
 - ⇒ leicht zu generieren und zu debuggen

HTML heißt HyperText Markup Language

2.1 HTML

SGML – XML – HTML - XHTML

■ SGML (Standard Generalized Markup Language)

- definiert seit 1986 eine Sprachfamilie zur Auszeichnung von Text (ISO-Norm 8879)
- verwendet die Metasprache DTD (Document Type Definition) zur Definition konkreter Sprachen

■ XML (Extensible Markup Language) ist eine Untermenge von SGML

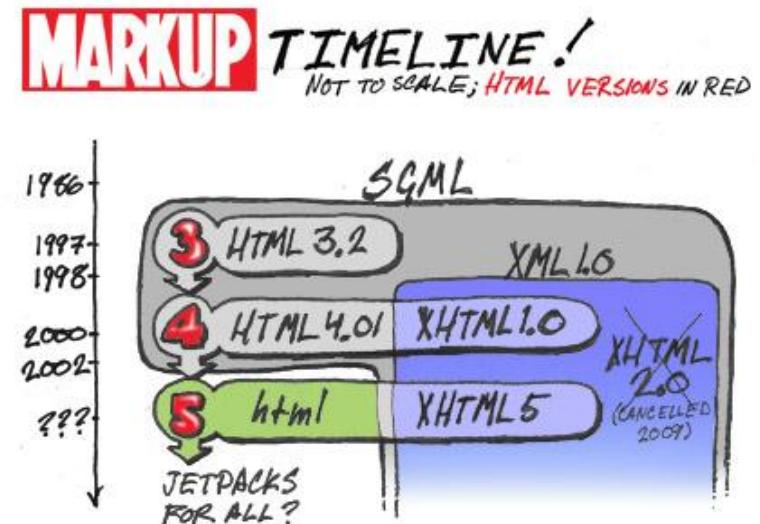
- z.B. mit spezieller Klammerstruktur

■ HTML und XHTML sind konkrete Sprachen

- HTML und XHTML definieren ausgehend von SGML bzw. XML eine Sprache (Dokumenttyp) mit Tags, Attributen und einer Grammatik
- XHTML beinhaltet strenge Regeln und behandelt Verstöße als Fehler
- HTML lässt viele Freiheiten (ist weniger eindeutig) und interpretiert tolerant

■ HTML5 wurde von SGML losgelöst und als eigenständige Sprache definiert

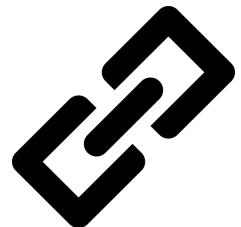
- für HTML5 gibt es eine HTML-und eine XHTML-Syntax ("XHTML5")



<http://jasonpriem.org/2010/04/markup-languages-whos-who>

Bemerkungen zum Werdegang

- Die Entwicklung von HTML war ein Trauerspiel
 - ⇒ Anstatt ein ausgereiftes Dokument-Format um URLs zu erweitern wurde HTML seit 1992 jahrelang "neu" erfunden
 - ⇒ Der Kampf um Browser-Markanteile führte zu proprietären HTML-Varianten
 - ⇒ der Standardisierungsprozess hinkte immer hinter dem Entwicklungsstand der Browser her
 - ⇒ Erst mit HTML4 gab es 1998 einen angemessenen Standard
 - ⇒ HTML5 wurde 10 (!) Jahre später (2009) als Working Draft veröffentlicht
 - ⇒ Die Browser halten bis heute keinen Standard vollständig ein und beinhalten Bugs – oder interpretieren den Standard speziell
- Leidtragende sind die Nutzer
 - ⇒ Webentwickler können ihre Ziele nur schwer realisieren
 - ⇒ Surfer leiden unbewusst (wissen nicht, wie es gemeint war)





2.1 HTML

Was ist eigentlich HTML5?

- Alle reden von HTML5 – und alle meinen unterschiedliche Dinge
 - ⇒ Das WWW-Konsortium "W3C" (<http://www.w3.org>) definiert traditionell den Standard und achtet auf eine saubere Darstellung
 - Üblicherweise wird der Standard in einem langen Prozess versioniert
 - Der Standard liegt weit hinter der Realität im Web zurück
 - ⇒ "WHATWG" (Web Hypertext Application Technology Working Group) beschreibt als "Living Standard" die neuesten Entwicklungen
 - Änderungen sind in der Webwelt normal und sollen in einem schnellen Prozess in den Living Standard von HTML5 aufgenommen werden
 - Die Änderungen sind fester Bestandteil von HTML5
 - HTML5 ist keine Version, sondern ein eigenständiger Name (deshalb fehlt auch das Leerzeichen vor der 5)
 - ⇒ In der Umgangssprache steht HTML5 für das interaktive Internet mit ansprechendem Design (Web 2.0)
 - Das beinhaltet nicht nur HTML5, sondern eine Mischung diverser Webtechniken wie CSS, JavaScript, Frameworks, APIs



2.1 HTML

Was bringen HTML5 & Co.?

- HTML5 & Co. bringen viele Neuerungen. Einige davon sind
 - ⇒ Ausnutzung von leistungsstarken Clients
 - Aufwändige Grafiken (Canvas)
 - Unterstützung von Threads (Workers in Javascript)
 - Datenspeicherung (MicroData)
 - ⇒ Unterstützung von unterschiedlichen Clients
 - Erkennung der Auflösung (CSS3 Media Queries)
 - Offlinezugriffe und Lokalisierung
 - ⇒ Modernisierung des alten Standards
 - Eingabeelemente mit Überprüfungen (z.B. Email-Felder)
 - Audio und Video jetzt integriert
 - Neue Tags (z.B. Header, Footer, Navigationsbereich)
 - Schickeres Design (z.B. Schatten und runde Ecken in CSS3)

HTML5 ist das, was man im Web heute braucht!



HTML5: Pro und Contra

■ Contra

- ⇒ W3C und WHATWG propagieren unterschiedliche Standards
- ⇒ Neue Tags und Attribute werden von vielen Browsern nicht erkannt
(das ist nicht wirklich neu)

■ Pro

- ⇒ HTML5 wurde (bzw. wird) abwärtskompatibel entwickelt und selbst alte Browser reagieren gnädig auf unbekannte Inhalte
- ⇒ Die neue Funktionalität mit HTML4 zu implementieren ist kaum machbar
- ⇒ Es gibt diverse Webseiten, die zeigen wie weit die neuen Features in den verschiedenen Browsern umgesetzt sind (z.B. <http://caniuse.com>)
- ⇒ Es gibt diverse JavaScript-Frameworks, die dabei helfen alte Browser zu unterstützen (z.B. <http://modernizr.com>)

Wir verwenden HTML5 !

HTML5 und ordentlicher Code

- HTML5 wurde abwärtskompatibel und browserfreundlich definiert
 - ⇒ für viele Sprachkonstrukte, die unter HTML4 einen Fehler erzeugten, ist jetzt das Browserverhalten definiert – und es sind keine Fehler mehr
 - viele fehlende Tags werden automatisch erkannt
 - schließende Tags können oft weggelassen werden
 - Groß/Kleinschreibung der Tags ist nicht festgelegt usw.
 - ⇒ Praktische Konsequenzen
 - Code der gegen viele Regeln eines sauberen Entwurfs verstößt, kann trotzdem HTML5-konform sein
 - um HTML5-Code auf einen ordentlichen Stil zu überprüfen, reicht der normale Konformitätscheck (HTML-Validator) nicht mehr aus
 - ⇒ Lösung
 - HTML5 sollte in professionellen Projekten zusätzlich mit einem statischen Code-Analyse-Tool auf die Einhaltung von Implementierungsrichtlinien geprüft werden (vgl. Lint für C++)

2.1.1 HTML Grundlagen

Implementierungsrichtlinien in dieser Veranstaltung

■ Implementierungsrichtlinien für EWA

- ⇒ Kein Layout / Design in HTML sondern konsequenter Einsatz von CSS
- ⇒ Einrückungen wie in normalen Programmiersprachen
- ⇒ Weitere Regeln, die HTML5 einschränken, werden auf den folgenden Folien mit diesem Symbol markiert



■ Die Einschränkungen sind gering und fordern keine große Umstellung

- ⇒ Die Syntax für ein paar HTML-Konstrukte ist etwas strenger
- ⇒ Der HTML-Code wird aber definitiv besser lesbar

Ordentlicher HTML5-Code ist die Zielsetzung in dieser Veranstaltung!

Hochschule Darmstadt

Fachbereich Informatik

2.1.1 HTML Grundlagen



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

2.1.1 HTML Grundlagen

Grundgerüst einer (ordentlichen) HTML5-Datei

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Text des Titels</title>
  </head>
  <body>
    <p>Eigentlicher Inhalt</p>
  </body>
</html>
```



Dokumenttyp

Zeichensatz

Titel für
Browserfenster



Schreibregeln

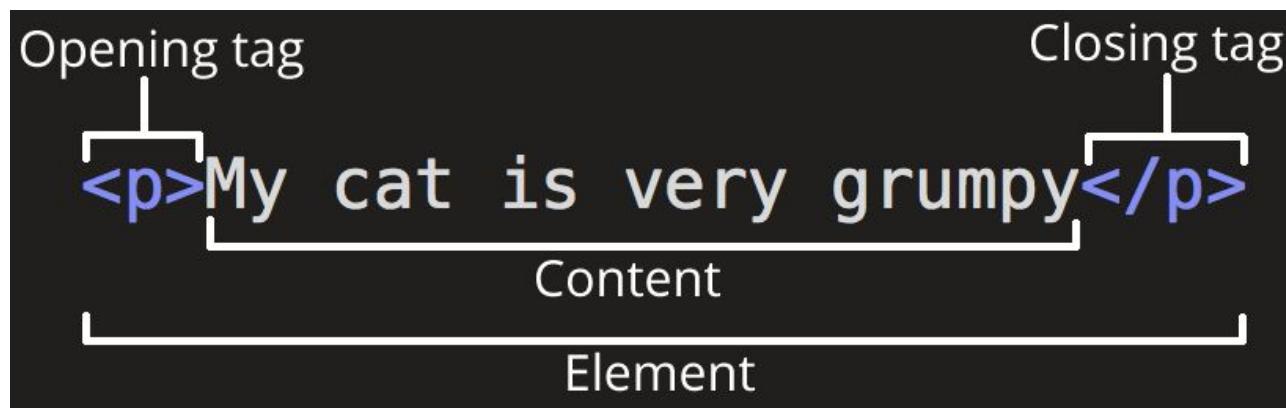
- Leerzeichen, Tabulator und Zeilenvorschub sind Trenner
 - ⇒ Anzahl spielt keine Rolle, außer in Attributwerten
 - ⇒ Ausnahme: in `<pre>` Abschnitten (=preformatted)
- Einrückung dient nur der Lesbarkeit
 - ⇒ wird vom Browser ignoriert
 - ⇒ wird von manchen WYSIWYG Tools ruiniert
- Kommentare
 - `<!--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



2.1.1 HTML Grundlagen

Tags & Attribute



The diagram shows the structure of an HTML element with an attribute. It consists of an opening tag with an attribute, the content, and a closing tag. The attribute is labeled as such.

Attribute
<p class="editor-note">
Content
Element
</p>

My cat is very grumpy

Source: https://developer.mozilla.org/en-US/Learn/Getting_started_with_the_web/HTML_basics

2.1.1 HTML Grundlagen

Tags (Marken) und Attribute

- Tags (Marken) markieren Abschnitte im Text
 - ⇒ Tag-Name steht in spitzen Klammern
 - ⇒ gleicher Name für öffnendes und schließendes Tag
 - ⇒ schließendes Tag kenntlich an zusätzlichem /
 - ⇒ Der Name des Tags wird kleingeschrieben
 - ⇒ <h2>Willkommen in unserem Hotel</h2>
- öffnende Tags können zusätzliche Attribute enthalten
 - ⇒ Attribute haben einen Namen und in der Regel einen Wert
 - ⇒ Attributwerte werden in Anführungszeichen geschrieben

<h2 id="hallo">Willkommen in unserem Hotel</h2>
- mit Tags markierte Abschnitte können verschachtelt sein

<h2>Willkommen in unserem Hotel</h2>



2.1.1 HTML Grundlagen

Sonderfall Standalone-Tags

- es gibt einige wenige "Standalone-Tags"
 - ⇒ leere Elemente = Abschnitte ohne sichtbaren Inhalt
 - ⇒ werden durch einen / vor der schließenden Klammer des öffnenden Tags hervorgehoben

Willkommen

auf unserer Homepage

oder

<meta charset="UTF-8" />

- alternative Schreibweisen (die wir aber nicht verwenden)

,
 oder
</br>



Strukturierung von Text

Überschriften

`<h1>` Überschrift der höchsten Gliederungsebene

`<h6>` Überschrift der niedrigsten Gliederungsebene

heading1 ... heading6

Abschnitte

`<p>` Textabsatz

`<div>` allgemeiner Block

div = division = Bereich

`` Inline-Element kein Block

Aufzählungen (nummeriert oder auch nicht)

``, ``, ``

ordered list, unordered list, list item

Zeilenumbruch erzwingen und verhindern

`
` expliziter Zeilenumbruch (standalone tag)

kein Block

` ` geschütztes Leerzeichen – verhindert Zeilenumbruch

`­` soft hyphen – Bindestrich bei Bedarf

z.B. 3.
Kapitel

alle außer
`` und `
`
erzeugen einen Block



2.1.1 HTML Grundlagen

Strukturierung von Webseiten

- Der Text innerhalb des `<body>`-Tags kann gegliedert werden
 - ⇒ `<section>` Abschnitt - ein logischer Bereich einer Webseite (z.B. der News Bereich)
 - ⇒ `<article>` Artikel
ein Textabschnitt, der eigenständig einen Inhalt abdeckt (z.B. eine einzelne News)
 - ⇒ `<body>` kann mehrere `<section>`s und `<article>`s enthalten – auch verschachtelt
 - ⇒ `<nav>` Navigationsbereich – enthält Verknüpfungen zur Navigation
 - ⇒ `<header>` und `<footer>` Kopf- / Fußzeilenbereich
ein Bereich mit Überschriften bzw. Logos, Datum usw. für das Gesamtdokument (im `<body>`) oder `<section>`s und `<article>`s
 - ⇒ `<aside>` Blöcke an der Seite
z.B. für Seitenleisten (außer Navigation), Zitate, Anmerkungen

`<section>`s und
`<article>`s haben eine
Überschrift (`<h2>...<h6>`)!





2.1.1 HTML Grundlagen

Strukturierung von Webseiten am Beispiel

h_da FB INFORMATIK

body

heading von body

heading der section (<h2>)

section

// h_da Fachbereich Informatik / Forschung & Partner

section

nav

- Organisation
- Studieninteressierte
- Studium
- Labore
- Verschiedenes
- International
- Forschung & Partner
- Online Belegsystem
- Intern

Direktlinks

- Stundenpläne
- Personen
- Sitemap
- Suche
- RSS-Feed
- Fachschaft

Login

article

Benutzername:

Password:

Angemeldet bleiben:

Anmelden

heading des article (<h3>)

article

Die Forschung an der Hochschule Darmstadt zielt darauf ab eine wissenschaftlich fundierte Hochschulausbildung auf hohem Niveau zu garantieren. Aufgrund des besonderen Praxisbezugs der Fachhochschulen wird keine Grundlagenforschung betrieben, sondern vielmehr angewandte Forschung und Entwicklung, häufig in Form von Projekten in Kooperation mit externen Unternehmen.

heading des article (<h3>)

article

Mit dem Institut für Angewandte Informatik (aida) und dem Zentrum für Angewandte Informatik (Z.A.I.) gehören zwei Institute dem Fachbereich Informatik an.
aida entwickelt in Zusammenarbeit mit Partnerunternehmen Forschungsprototypen und evaluiert neue Informatik-Technologien im Hinblick auf ihre praktische Einsetzbarkeit.
Das Zentrum für Angewandte Informatik e. V. ist ein gemeinnütziger Verein, der zur Förderung der Forschung und Lehre auf dem Gebiet der Angewandten Informatik an der Hochschule Darmstadt gegründet wurde.

Partnerhochschulen

Der Fachbereich Informatik der Hochschule Darmstadt kooperiert zur Zeit mit mehr als 15 ausländischen Hochschulen und hat damit die notwendige Infrastruktur für die Mobilität von Studenten und Dozenten sowie weitergehende transnationale Zusammenarbeit geschaffen.

Copyright Fachbereich Informatik - Hochschule Darmstadt / Forschung & Partner - Darmstadt / Rhein-Main / Herborn
Letzte Änderung: 12.06.2015 | Verantwortung: Dekanat des FB Informatik

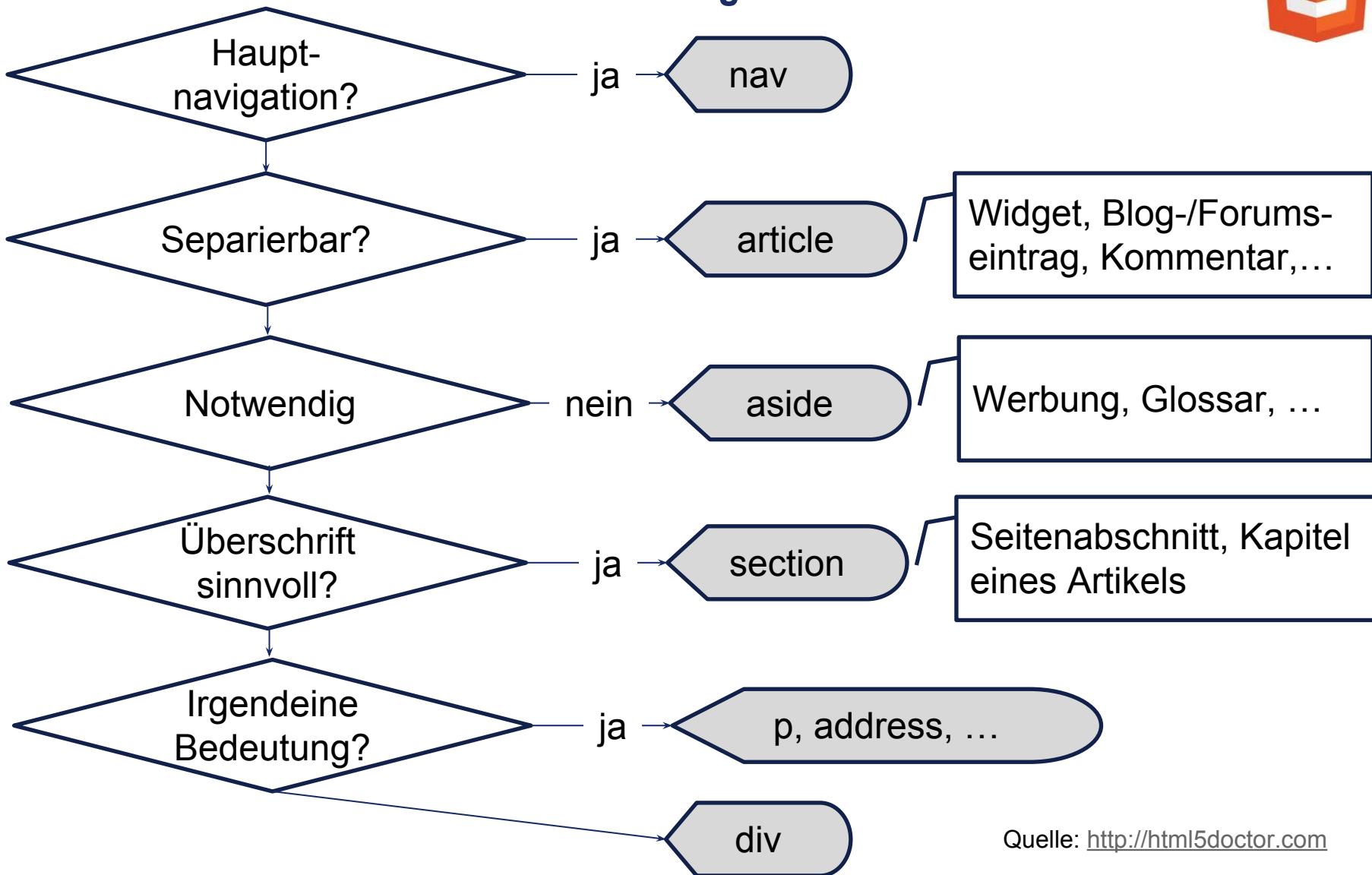
footer von section

[Impressum] [Fachbereich Informatik] [HOCHSCHULE DARMSTADT]

footer (von body)

2.1.1 HTML Grundlagen

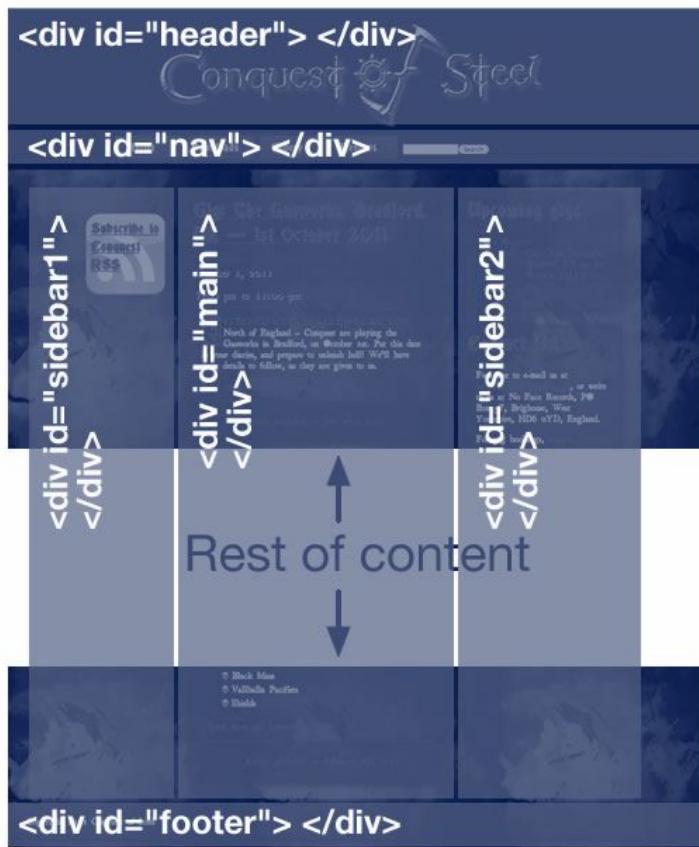
Welches Strukturelement ist das Richtige?

Quelle: <http://html5doctor.com>

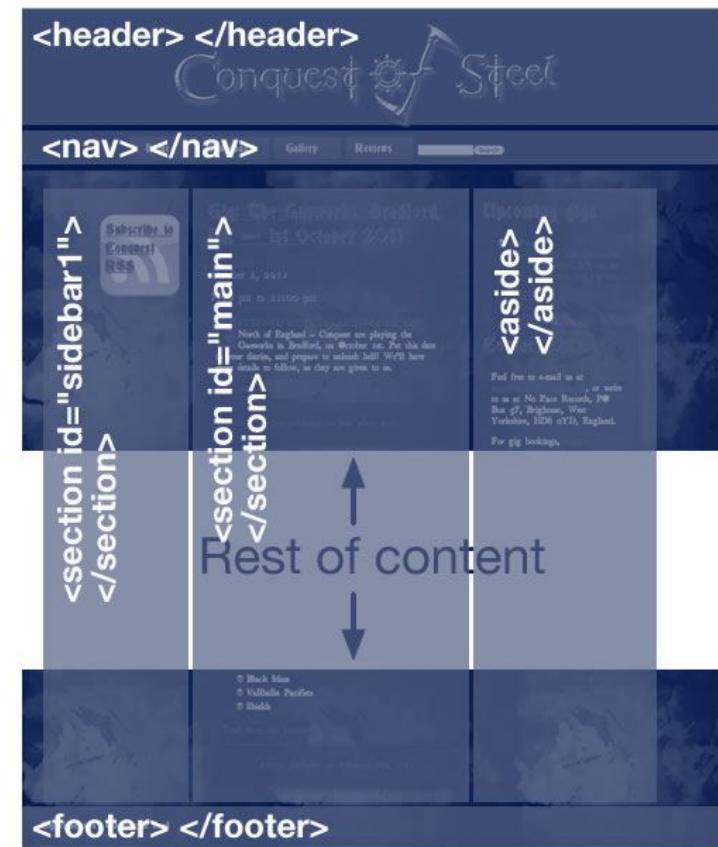
2.1.1 HTML Grundlagen

Vergleich HTML4 und HTML5

■ HTML4



■ HTML5



Quelle: http://www.w3.org/wiki/HTML_structural_elements

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 (siehe CSS)
- ⇒ `lang, dir` Landessprache und Textlaufrichtung

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





2.1.1 HTML Grundlagen

Weitere HTML5-Attribute

- http://www.w3schools.com/tags/ref_standardattributes.asp
- Beispiel: Normal sind nur Links und Bilder draggable
 - mit dem "draggable" attribut wird alles draggable
- Oft genutzt: das data-* attribut, um eigene Datentypen zu generieren
 - mit einem Namen ohne Großbuchstaben
 - Bsp.: data-preis, data-key, data-xxx
 - In HTML oder auch über das DOM
 - Wird wichtig, wenn man z.B. HTML Daten per JavaScript auslesen möchte



```
<ul>
  <li data-student-type="erstie">Erstsemesterstudent</li>
  <li data-student-type="slacker">Student im 18. Semester</li>
  <li data-student-type="master">Masterstudent</li>
</ul>
```

Pizza-Service Beispiel:

```
<ol>
  <li data-preis="4.99">Pizza Salami</li>
  ...
</ol>
```



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 kursiv)
 - <**q cite="URL"**> Zitat mit Quellenangabe
 - <**blockquote**> Zitat als Block gesetzt

2.1.1 HTML Grundlagen

Grenzfall: physische Formatierung

- definiert das visuelle Erscheinungsbild
- nicht verpönt, aber besser zu vermeiden – es gibt oft andere Tags mit einer echten Bedeutung!

- ein paar Beispiele:

** fette Schrift (bold)**

<i> kursive Schrift (italic)

<sup> Schrift hochgestellt

<sub> Schrift tiefgestellt

<pre> vorformatierter Text (nur für Quellcode)

Block

Struktur einer Tabelle (1)

Grundidee:

<table>				
<tr>	<th> </th>	<th> </th>	<th> </th>	</tr>
<tr>	<td> </td>	<td> </td>	<td> </td>	</tr>
<tr>	<td> </td>	<td> </td>	<td> </td>	</tr>
</table>				

nach S. Münz: SELFHTML

2.1.1 HTML Grundlagen

Struktur einer Tabelle (2)

```
<table>
  <caption>Meine Tabelle</caption>
  <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>
```

nur zur Darstellung
tabellarischer Daten

Tabellen-Überschrift

zeilenweise (tr = table row)



beliebig viele Zeilen und Spalten

Struktur einer Tabelle (3)

■ Spaltenbreite definieren

- ⇒ falls der Browser die Spaltenbreite nicht automatisch bestimmen soll, kann über CSS die Breite festgelegt werden
- ⇒ die Spalten werden vor dem ersten Tabelleneintrag "benannt"

```
<colgroup>
  <col id="col1" />
  <col id="col2" />
</colgroup>
```

Für einen schnellen Aufbau großer Tabellen:
Die Breite aller Spalte festlegen!
Eine id macht die Spalte für CSS adressierbar:
#col1 {width:80%;}

■ Zellen verbinden

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

auch
kombinierbar

■ Tabellenüberschrift

- ⇒ unmittelbar nach dem `<table>-Tag` `<caption>...</caption>`
- ⇒ Unterschrift mit `<caption style="caption-side:bottom">`

Aufgabe

Erstelle eine Tabelle in HTML

Pizza		Preis
Icon1	Tonno	6,99
Icon2	Hawaii	5,99
Icon3	Salame	5,99

Pizzatabelle

Grafiken einbinden

■ Das ``-Tag erlaubt das Einbinden von Bildern

standalone tag

- ⇒ für Bilddateien der Formate GIF, PNG, JPG und seit HTML5 SVG
- ⇒ notwendige Attribute:

`src` Quelldatei

`alt` Ersatztext, der ausgegeben wird, wenn die Grafik nicht angezeigt werden kann (Screenreader, Ladeprobleme)
Dient die Grafik nur zur Veranschaulichung

(z.B. Bild einer Pizza neben dem Namen der Pizza)

wird `alt=""` gesetzt

- ⇒ optionale Attribute

`width`, `height` Breite und Höhe in Pixeln

`title` Text, der als Tooltip angezeigt wird

- ⇒ Beispiel

```

```

Bilddateien

Per JavaScript
gezeichnete Grafiken
mit <canvas>

- GIF oder PNG für Grafiken (z.B. Screenshots, ClipArts)
 - ⇒ GIF: 256 Farben Palette, komprimiert
 - ⇒ Nachfolger: PNG mit Palette und True Color, komprimiert, verlustfrei
 - ⇒ Freistellen (transparenter Hintergrund) möglich
- JPEG für Fotos
 - ⇒ Echtfarben, komprimiert, verlustbehaftet
 - ⇒ Freistellen nur bei PNG
- SVG (Scalable Vector Graphics) für technische Zeichnungen
 - ⇒ XML-basiertes Format für 2D-Vektorgrafiken
 - ⇒ ohne Qualitätsverlust skalierbar
- Möglichkeiten zur Speicherplatzersparnis
 - ⇒ Hintergrund mit kleiner Grafik "kacheln"
 - ⇒ Größe und Farbtiefe reduzieren
 - ⇒ Beschränkung auf wenige Grafiken



Hauptproblem
Übertragungszeit!



Audio und Video

- Audio und Video gehören erst seit HTML5 zum Standard
 - ⇒ Vor HTML5 waren Multimediadateien nur über Plugins (z.B. FlashPlayer) abspielbar (und die Plugins waren nicht betriebssystemübergreifend verfügbar)
 - ⇒ Seit HTML5 gibt es "nur noch" Uneinigkeit über die vom Browser zu unterstützenden Formate (wegen Lizenzrechten)
 - ⇒ Damit die AV-Datei von allen Browsern abgespielt wird, müssen derzeit 2 Formate bereitgestellt werden
 - mp3 und ogg für Audio bzw. ogv und mp4 für Video

```
<audio autoplay controls>
    <source src="myAudio.mp3" />
    <source src="myAudio.ogg" />
</audio>
<video width="320" height="240" poster="bild.jpg" controls autoplay>
    <source src="myVideo.mp4" />
    <source src="myVideo.ogv" />
    Ihr Browser unterstützt leider kein Video-Tag
</video>
```

Der Browser wählt
ein Format, das er
unterstützt!

- ⇒ Abspielen erfolgt im "streaming mode" wegen der 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="P. Rath" />  
<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="refresh" content="5">
```

lädt die aktuelle
Seite nach 5
Sekunden erneut

- Setzen des verwendeten Zeichensatzes

```
<meta charset="UTF-8">
```

Markdown

- Alternative Methoden HTML-Seiten zu schreiben mit Markdown
- Erfunden von John Gruber und Aaron Schwartz
- Vereinfachte Syntax die in valides HTML übersetzt wird

<http://markdown.de/>

Hochschule Darmstadt

Fachbereich Informatik

2.1.2 Hyperlinks



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

2.1.2 Hyperlinks

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
- "Hyperlink" ist lediglich eine technische Realisierung !
- Einsatzgebiet klären und gestalterisch unterscheiden

2.1.2 Hyperlinks

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
 - ⇒ nicht zu viele Verweise auf dieselbe Stelle innerhalb einer Seite
- Verweis sollte unmittelbar erkennbar sein
 - ⇒ nicht erst nach "Abtasten" mit der Maus
- alle Seiten vollständig verlinken
 - ⇒ "Zurück"-Button des Browsers sollte innerhalb einer Website überflüssig sein - aber er sollte möglichst auch funktionieren

2.1.2 Hyperlinks

Ziele von Verweisen

- eine Datei, die der Browser als Seite darstellen kann
 - ⇒ meistens HTML, aber auch anderes möglich
 - ⇒ 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:j.bond@fbi.h-da.de">J. Bond</a>
<a href="ftp://www.xyz.de/setup.zip">Download</a>
<a href="file:///c:/lokal.htm">lokale Datei</a>
```

2.1.2 Hyperlinks

Verweise

■ 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

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

auch: localhost

absolut

■ Groß-/Kleinschreibung beachten

- ⇒ Server laufen meist unter Unix und Unix ist case sensitive bezüglich Datei- und Verzeichnisnamen

beliebter Fehler unter Windows

2.1.2 Hyperlinks

Absolute und relative Verweise

ohne Angabe von Server und Verzeichnispfad

- 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)

2.1.2 Hyperlinks

Verweise innerhalb einer Datei ("Anker")

- wird häufig eingesetzt für "Inhaltsverzeichnis" am Anfang einer Datei
 - ⇒ z.B. bei Wikipedia Quellen
- Verweisziel definieren per id in beliebigem Tag
`<h2 id="Erl">Erläuterung</h2>`
- Verweis definieren
siehe die `Erläuterung` unten
- der Verweis kann auch zu einer bestimmten Position in einer anderen Datei zeigen
`Erläuterung`
`...`
- der Browser scrollt die Seite so, dass der Anker an der Oberkante des Fensters erscheint

veraltet:

`...`

Zusammenfassung

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

Jetzt können Sie eine einfache HTML-Seite schreiben!

Hochschule Darmstadt

Fachbereich Informatik

2.1.3 HTML Formulare



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Formular - Beispiel



Steuerelemente für Formulare

Bitte machen Sie Ihre Eingaben

Einzeliges Textfeld

Ihre Eingabe

Textfeld mit Scrollbalken

Viiiiiel Text

Bitte wählen Sie aus

List-Box

- 1. Möglichkeit
- 2. Möglichkeit
- 3. Möglichkeit

Combo-Box

- 1. Möglichkeit

Radiobuttons

Ja Naja Nein

Checkboxen

Flag1 Flag2

Schaltflächen

Abschicken Zurücksetzen

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)
- Bereich mit Eingabeelementen im HTML-Body markieren

```
<form action="/cgi-bin/Echo.pl" id="form1"
      accept-charset="UTF-8" method="get">
```

Steuerelemente (Eingabefelder, Auswahllisten, Buttons...) und sonstige HTML-Tags und CSS-Formatierung

```
</form>
```

 - ⇒ accept-charset zur Sicherheit gegen willkürliche Benutzereinstellung
 - ⇒ falls das Steuerelement außerhalb des Formulars liegt, kann der Bezug über `form="form1"` hergestellt werden
- Alternative Aktion: Formulardaten per eMail verschicken
 - ⇒ `action="mailto:Meier@xyz.de"`
 - ⇒ unsicher, weil von der Installation beim Surfer abhängig

hier: Übergabe der Daten an Perl-Skript

Eingabefelder

placeholder erscheint
nur ohne value

Einzeiliges Textfeld	<input type="text" value="Ihre Eingabe"/>
Textfeld mit Scrollbalken	<input type="text" value="Viiiiiel Text"/>

■ einzeilige Textbox

```
<input type="text" name="zuname" value="Muster" size="30"
       maxlength="40" placeholder="Ihre Eingabe"
       readonly />
```

standalone tag

- ⇒ name und value wird an Server übermittelt
- ⇒ value kann vorbelegt sein
- ⇒ size und maxlength für Anzeigelänge und Maximalgröße
- ⇒ placeholder wird angezeigt, bevor man eine Eingabe macht
- ⇒ mit readonly reine Anzeige (ausgegraut)



■ Variante: Passwortfeld mit *-Anzeige

- ⇒ wie oben, jedoch type="password"
- ⇒ keine verschlüsselte Übertragung!

■ mehrzeiliges Textfeld (bei Bedarf mit Scrollbalken)

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



2.1.3 HTML Formulare

Auswahllisten

Listbox

```
<select name="top4[]" size="3" multiple>
    <option selected> 1. Möglichkeit</option>
    <option> 2. Möglichkeit</option>
    <option value="3"> 3. Möglichkeit</option>
    <option> 4. Möglichkeit</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

Combobox

- ⇒ Eine Listbox mit `size="1"` liefert eine aufklappende Liste mit Optionen

Mehrfachauswahl mit zusätzlichem Attribut `multiple`

- ⇒ Bei erlaubter Mehrfachauswahl eckige Klammern an den Namen hängen (z.B. `name="top4[]"`)! PHP macht dann daraus ein Array!



Schaltflächen und verborgene Felder



- allgemeine Schaltflächen für JavaScript-Ereignisse

```
<input type="button" name="Start" value="Startseite"
      onclick="self.location.href='http://www.xyz.de/' />
```

- Schaltfläche zum Absenden der Formulardaten

- ⇒ wie oben, jedoch **type="submit"** ; **onclick** nicht nötig
- ⇒ **type="button"** notwendig, wenn submit nicht getriggert werden soll!

- Schaltfläche zum Löschen der Formulardaten

- ⇒ wie oben, jedoch **type="reset"** ; **onclick** nicht nötig

- verborgenes Datenfeld (z.B. für Sessionverwaltung)

- ⇒ **<input type="hidden" name="sessionID" value="4711" />**

- in HTML5 gibt es noch diverse andere Typen für das Input-Tag

- ⇒ **<input type="email" ... />** oder auch **date**, **number**, **color** uvm.
- ⇒ diese Typen überprüfen automatisch die Eingabe



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



2.1.3 HTML Formulare

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="OK" value="1" />
<input type="radio" name="OK" value="2" />
<input type="radio" name="OK" value="3" checked />
```

- ⇒ Vorauswahl durch Attribut **checked**
 - ⇒ Zur Beschriftung ist **<label>** geeignet (nächste Folie)

- Checkboxen für Boole'sche Eingabe

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

- ⇒ übermittelt wird der **value** nur für angekreuzte Checkboxen
 - ⇒ Vorauswahl durch Attribut **checked**
 - ⇒ Beschriftung erfolgt mit Labels

The screenshot shows a user interface for selecting preferences. It includes three radio buttons labeled "Ja", "Naja", and "Nein", with "Ja" being the selected option. Below this is a section for checkboxes with two items, "Flag1" and "Flag2", where "Flag1" has a checked mark.

Die Beschriftungen sind **<label>**

"on" wenn value fehlt

Beschriftung von Formularelementen

Vorname:	<input type="text" value="Ihr Vorname"/>
Zuname:	<input type="text" value="Ihr Nachname"/>
Auswahl:	<input checked="" type="checkbox"/>

- Formularelemente haben kein Attribut für Text
 - ⇒ Der Text "Vorname" und das Eingabefeld "Ihr Vorname" im Beispiel haben (für den Browser) keinen Zusammenhang
- Mit Hilfe von "Labels" wird ein logischer Bezug zwischen Formularelement und Beschriftungstext hergestellt
 - ⇒ `<label>Zuname:</label>`
 - ⇒ Das `<label>` umschließt "sein" Formularelement
 - ⇒ Alternativ wird der Bezug über eine `id` und das Attribut `for` hergestellt:
`<label for="nachname"></label>`
 - ⇒ anwendbar für `<input>`, `<select>` und `<textarea>`
- Vorteile bei der Verwendung
 - ⇒ Beim Klicken auf den (zugeordneten) Text wird das Eingabefeld selektiert bzw. die Checkbox selektiert
 - ⇒ Unterstützung von Screenreadern

2.1.3 HTML Formulare

Gruppierung von Formularelementen

- Größere Formulare bestehen häufig aus *Gruppen* von Elementen. Ein typisches Bestellformular besteht beispielsweise aus Elementgruppen wie "Absender", "bestellte Produkte" und "Formular absenden"
 - ⇒ Eine zusammengehörige Gruppe von Formularelementen wird durch `<fieldset>...</fieldset>` umrahmt
 - ⇒ Dazwischen können Sie beliebige Teile Ihres Formulars definieren.
- Unterhalb des einleitenden `<fieldset>`-Tags und vor den ersten Formularinhalten der Gruppe sollte eine Gruppenüberschrift (z.B. Formular) für die Elementgruppe vergeben werden.
 - ⇒ Schließen Sie den Gruppenüberschriftentext in die Tags `<legend>...</legend>` ein
- Vorteil bei der Verwendung
 - ⇒ Formatierung nach Wunsch über HTML/CSS
 - ⇒ Web-Browser kann Elementgruppen durch Linien oder ähnliche Effekte optisch sichtbar machen

Bitte machen Sie Ihre Eingaben

Vorname:

Ihr Vorname

Zuname:

Ihr Nachname

Auswahl:

Attribute für Eingabefelder

■ Hinweistexte

- ⇒ werden bei Eingabefeldern angezeigt, bevor man eine Eingabe macht
- ⇒ `placeholder="Ihr Nachname"`



■ Tabulatorreihenfolge

- ⇒ normalerweise entsprechend der Reihenfolge in der HTML-Datei
- ⇒ oder explizit setzen mit Attribut `tabindex="1"` usw.

■ Tastaturkürzel definieren

- ⇒ `accesskey="x"` springt mit alt+x sofort in das entsprechende Eingabefeld
- ⇒ Dies ist im Browser nicht erkennbar und muss beschriftet werden !



2.1.3 HTML Formulare

Attribute zur Validierung von Eingabefelder

■ required

```
<input type="email"  
       required />
```

E-Mail

Bitte füllen Sie dieses Feld aus.

■ pattern

```
<input pattern="[0-9]{5}" name="plz"  
       title="Fünfstellige Postleitzahl in Deutschland." />
```

Ohne required, darf das Feld trotz
pattern auch leer bleiben!

PLZ

Bitte halten Sie sich an das vorgegebene Format:
Fünfstellige Postleitzahl in Deutschland.

■ min..max

```
<input name="bday" type="date" max="1994-12-31" />
```

2.1.3 HTML Formulare

Zusammenfassung

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

- Grundidee Formulare (Übertragung von Daten an den Web Server)
- Aufbau von Formularen
 - ⇒ 1- und mehrzeiliges Textfeld (<input type="text" ... /> bzw. <textarea> ... </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" ... />
 - ⇒ Abschicken von Formularen
<input type="submit" ... />
 - ⇒ Beschriftung von Formularelementen
<label>, <fieldset> und <legend>

Daten werden nur für Formularelemente übertragen, die innerhalb eines <form>s liegen und ein name-Attribut haben!

Wohin die Übertragung geht, legt das action-Attribut des <form>s fest.

Hochschule Darmstadt

Fachbereich Informatik

2.1.4 HTML Werkzeuge



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Zeichencodierung

ASCII-Tabelle

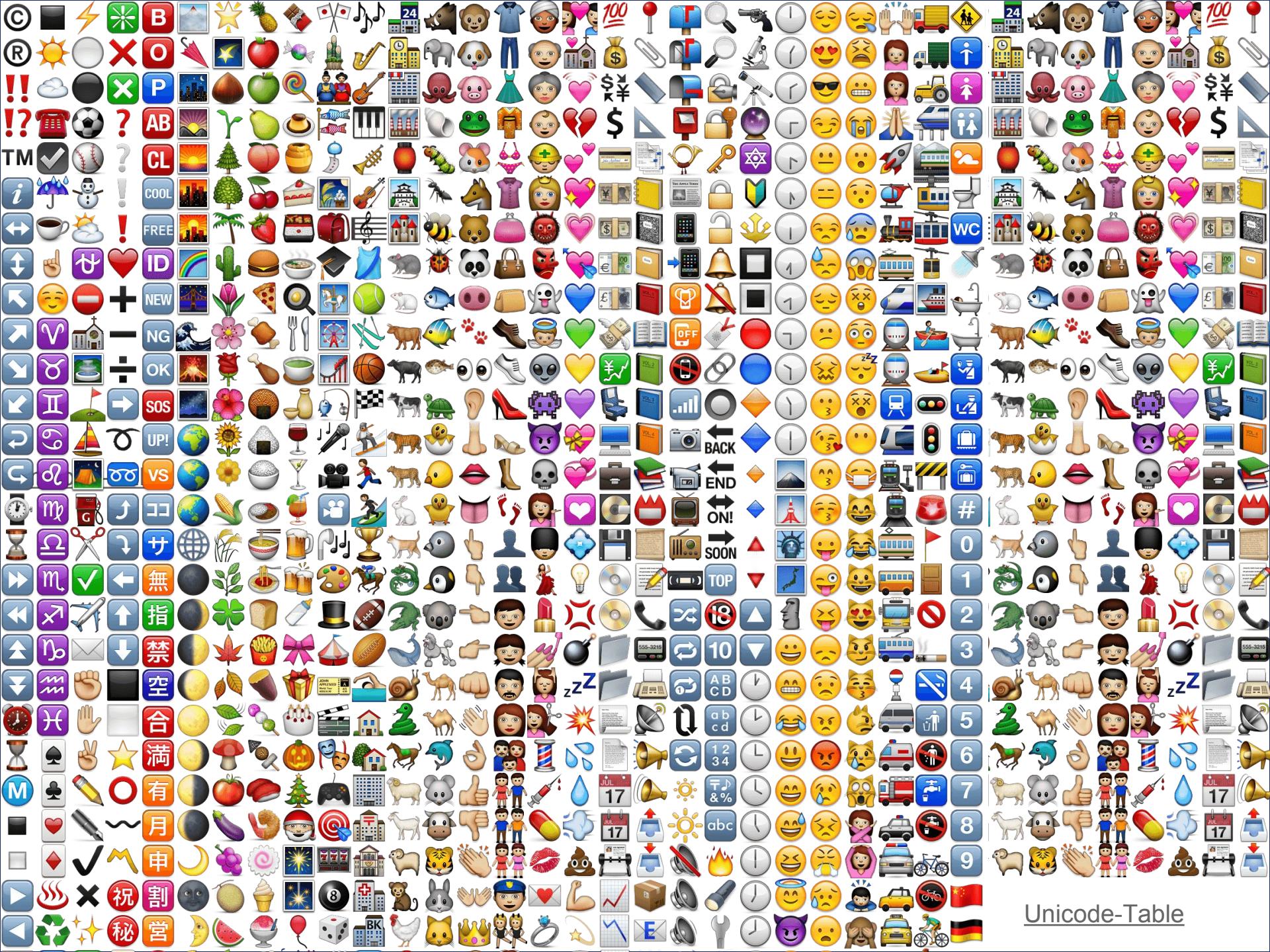
Dezimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
	Hex.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
32	20		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
48	30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	50	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	-
96	60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	70	p	q	r	s	t	u	v	w	x	y	z	{		}	~	

Windows-1252 (CP1252)																
!	"	#	\$	%	&	'	()	*	+	,	-	.	/		
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
P	Q	R	S	T	U	V	W	X	Y	Z	[\	^			
~	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	
p	q	r	s	t	u	v	w	x	y	z	{		}	~		
,	f	"	..	†	‡	‰	š	č	€	ł						
,	„	“	•	—	—	—	—	—	—	—	—	—	—	—	—	—
í	đ	£	¤	¥	₩	₪	₪	₪	₪	₪	₪	₪	₪	₪	₪	₪
ó	±	²	³	⁴	⁵	⁶	⁷	⁸	⁹	⁹	⁹	⁹	⁹	⁹	⁹	⁹
À	Á	Â	Ã	Å	Æ	Ç	É	Ê	Ë	Ì	Í	Î	Ï	Ӯ	ӻ	ӻ
Đ	Ñ	Ò	Ó	Ô	Ӯ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ
à	á	â	ã	å	æ	ç	é	ê	ë	ì	í	î	ï	Ӯ	ӻ	ӻ
ő	ñ	ò	ó	ô	Ӯ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ
ö	÷	ø	ú	û	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ	Ӱ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+13Ax	D	R	T	Ճ	Ծ	Ի	Ծ	Ծ	Ր	Յ	Ա	Ջ	Ե	ԾՎ	Պ	Ժ
U+13Bx	Ֆ	Ր	Ճ	Ո	Ծ	Ծ	Գ	Մ	՚	Ծ	ՕՒ	Հ	՚	Յ	Թ	Ւ
U+13Cx	Գ	Ճ	հ	Զ	՚	Ծ	Լ	Ծ	Ծ	Վ	Ծ	Ծ	Ծ	Ծ	Ծ	Ծ
U+13Dx	Փ	Ց	Ր	Լ	Ո	Ծ	Ծ	Ճ	Ճ	Վ	Ծ	Ծ	Ծ	Ծ	Ծ	Ծ
U+13Ex	Ւ	Ց	Պ	Գ	Վ	Խ	Կ	Ճ	Ծ	Գ	Ծ	Ծ	Ծ	Ծ	Ծ	Ծ
U+13Fx	Բ	Ճ	Ւ	Գ	Վ											

Unicode Consortium
Code Chart for the
Cherokee Language

Source: <http://www.unicode.org/charts/PDF/U13A0.pdf>



Zeichenkodierung und Fonts

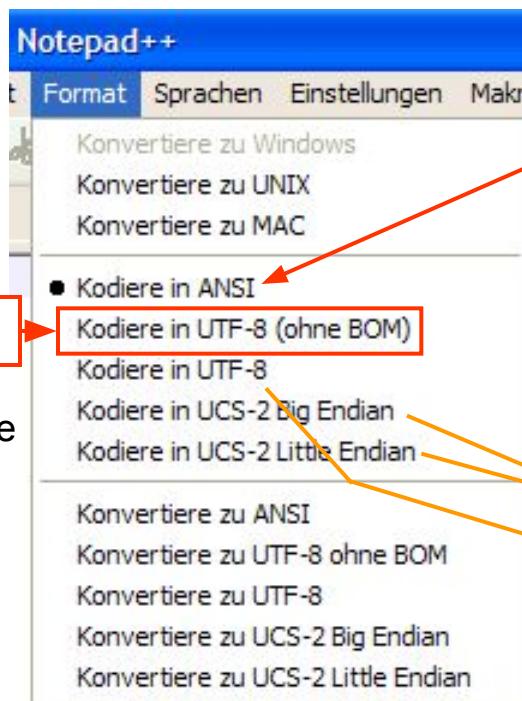
<http://de.selfhtml.org/html/allgemein/zeichen.htm>
<http://de.selfhtml.org/inter/zeichenkodierungen.htm>
<http://www.unicode.org/charts/>

- Zeichenkodierungen definieren die Abbildung Code → Zeichen
 - ⇒ Codepage 850 8E → Ä MS DOS
 - ⇒ Codepage 1252 (Latin I) C4 → Ä MS Windows (*alt*)
 - ⇒ ISO-8859-1 C4 → Ä ähnlich •
 - ⇒ Unicode 00C4 → Ä MS Windows (*neu*)
 UTF-8 ist eine kompakte Unicode-Kodierung mit 1..4 Byte pro Zeichen
 - ⇒ benannte Zeichen in HTML Ä → Ä *reines ASCII*
 - ⇒ die Zeichenkodierung ist eine Eigenschaft der Datei
und wird in XML und HTML im Header dokumentiert
- Fonts definieren die Abbildung Code → Aussehen
 - ⇒ OpenType und TrueType basieren auf Unicode
 - ⇒ Impact 00C4 → Ä
 - ⇒ Times New Roman 00C4 → Ä
 - ⇒ verschiedene Textabschnitte können mit verschiedenen Fonts dargestellt werden;
dies wird per CSS festgelegt

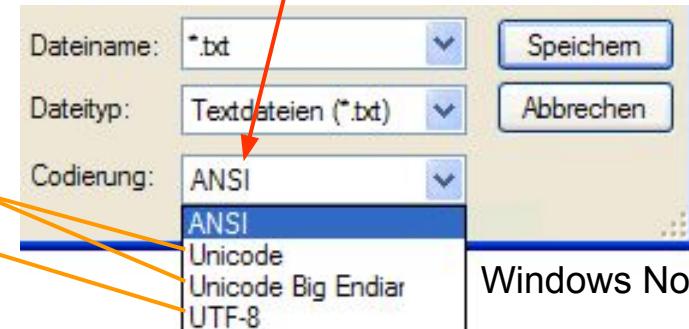
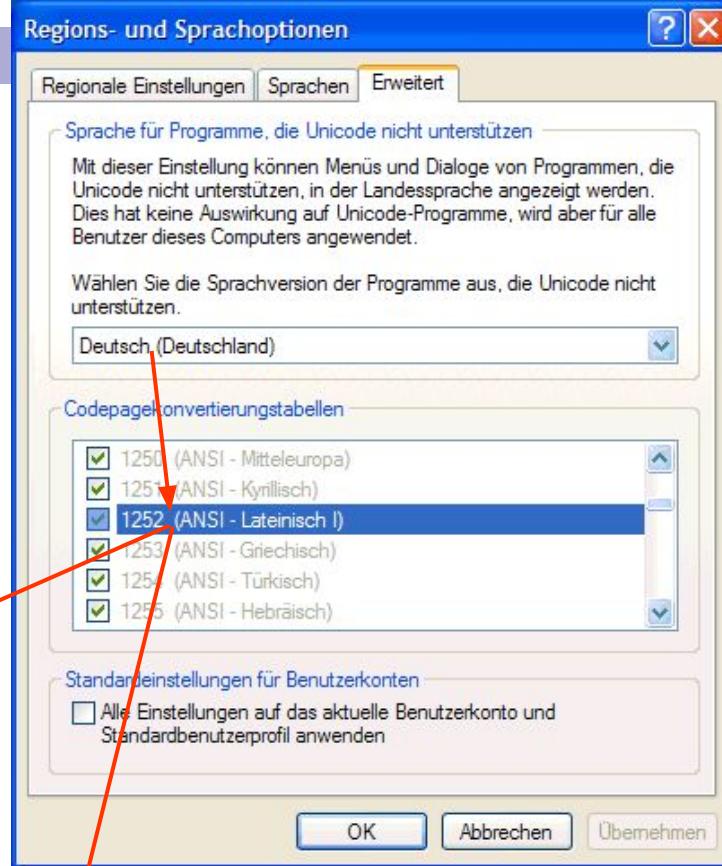
2.1.4 HTML Werkzeuge

Zeichenkodierung im Editor

- die Datei muss auch wirklich mit der angegebenen Zeichencodierung erstellt sein
 - ⇒ Einstellung des Editors
 - ⇒ Default des Betriebssystems



UCS-2 = Unicode



mit BOM (= Byte Order Mark) am Dateianfang
(optional bei UTF-8; zur Unterscheidung von ISO)

2.1.4 HTML Werkzeuge

Zeichenkodierung systemweit einheitlich

- vorzugsweise UTF-8 systemweit als Zeichenkodierung einsetzen
 - ⇒ Projekt von vorneherein mit UTF-8 aufsetzen
 - ⇒ nachträgliche Umstellung ist mühsam
 - ⇒ uneinheitliche Kodierung würde explizite Konvertierungen erfordern
- PHP-Dateien in UTF-8 ohne BOM (Byte Order Mark) kodieren
 - ⇒ BOM besteht aus Bytesequenz EF BB BF am Dateianfang
 - ⇒ BOM würde von PHP sofort ausgegeben, vor evtl. HTTP Header
 - ⇒ **Achtung:** eine UTF-8 Datei ohne BOM kann der Editor von einer ISO Datei nur unterscheiden, wenn sie auch tatsächlich Umlaute enthält (ggf. hilfsweise als Kommentar einfügen)
- Zeichenkodierung und Sortierreihenfolge gleich beim Anlegen der Datenbank festlegen
 - ⇒

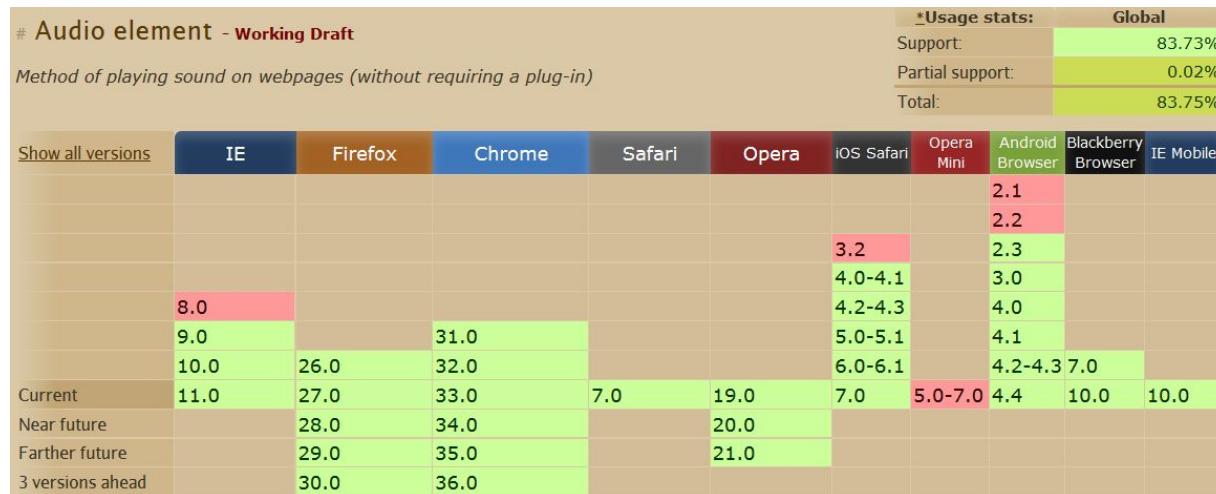
```
CREATE DATABASE `db`  
DEFAULT CHARACTER SET utf8  
COLLATE utf8_unicode_ci;
```
 - ⇒ vorzugsweise einheitlich für alle Tabellen und Felder
- Zeichensatz für die Kommunikation zwischen PHP und Datenbank definieren
 - ⇒ `$mysqli->set_charset("utf8");`

2.1.4 HTML Werkzeuge

HTML Browser

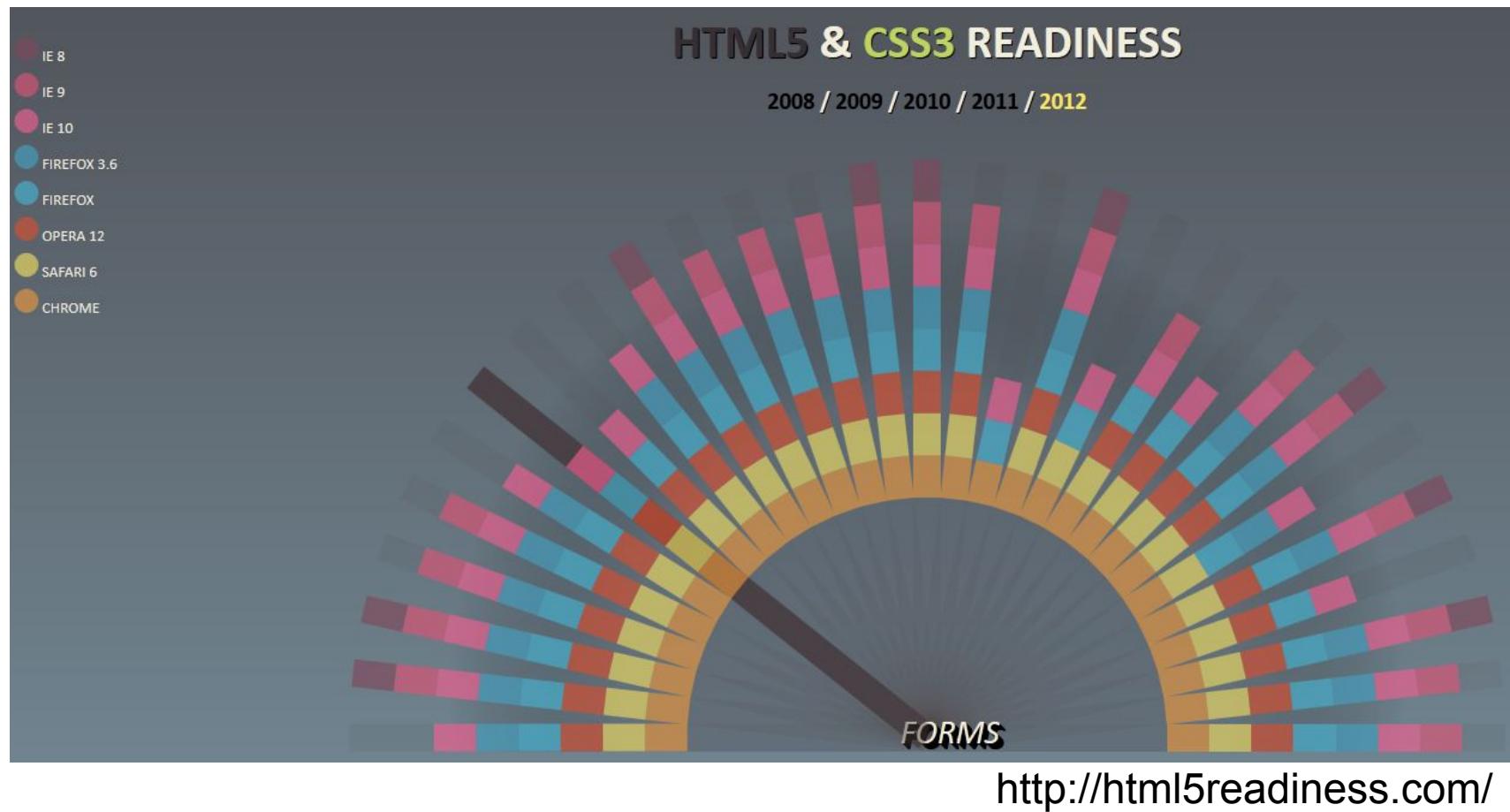


- Es gibt eine Vielzahl verschiedener Browser in verschiedenen Versionen
 - ⇒ Die Unterstützung von "neueren" Features ist nicht sicher
 - Im Web gibt es diverse Seiten, welche die Umsetzung verfolgen z.B. <http://caniuse.com> oder <http://html5readiness.com/>
 - Für ältere Browser muss oft eine Rückfalllösung entwickelt werden
 - ⇒ Webseiten unbedingt für verschiedene Browser testen
 - z.B. bei <http://browsershots.org>



2.1.4 HTML Werkzeuge

Stand der Umsetzung



Oder auch: When can I use? <http://caniuse.com>

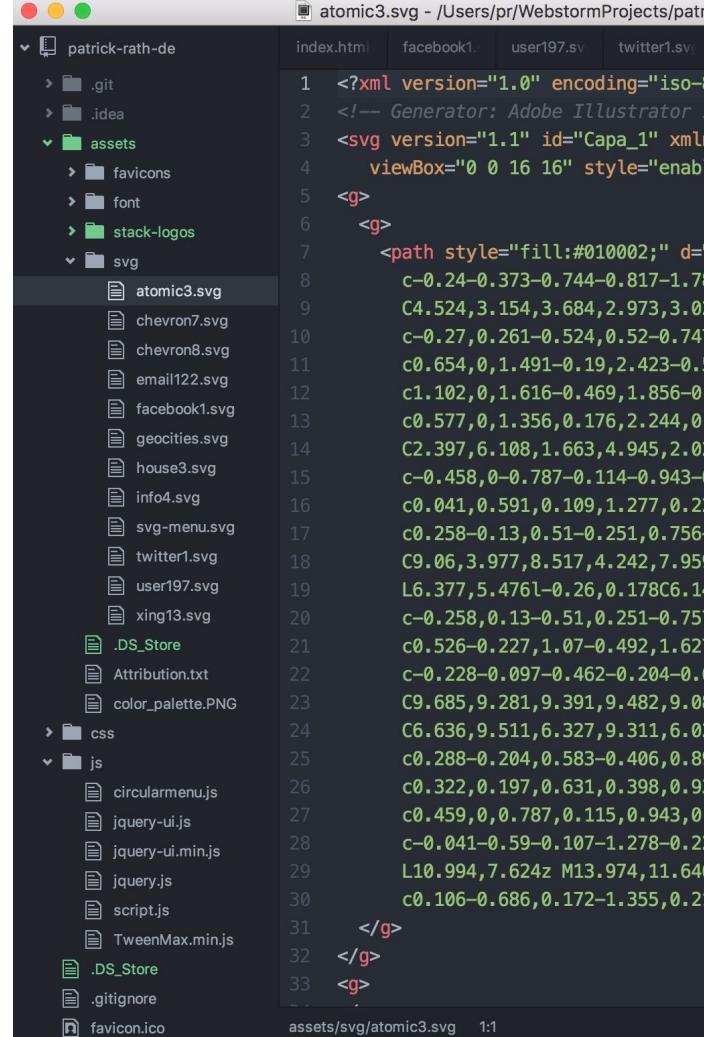
2.1.4 HTML Werkzeuge

HTML Editor

- 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

2.1.4 HTML Werkzeuge

Moderne HTML Editoren - Beispiel



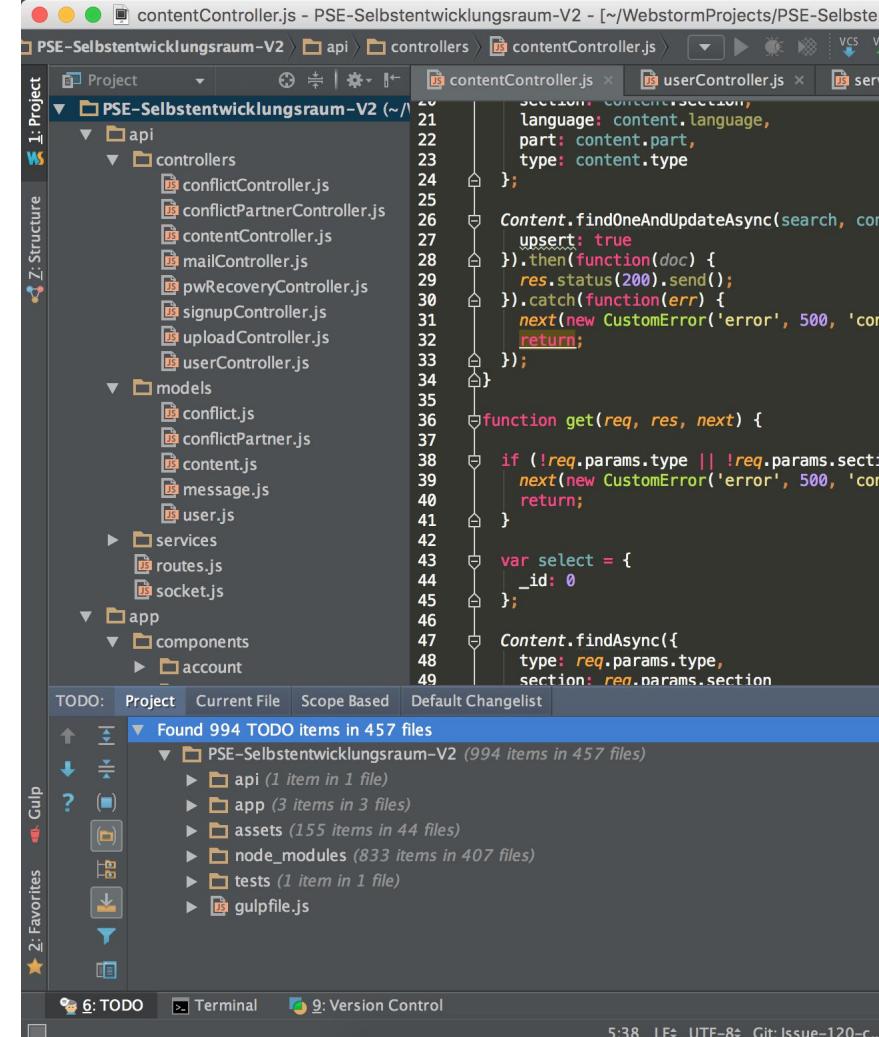
The screenshot shows the Atom IDE interface with the file 'atomic3.svg' open. The code is an SVG XML document. The Atom interface includes a file tree on the left, a code editor with syntax highlighting, and various status icons at the top.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<!— Generator: Adobe Illustrator 10.0 —>
<svg version="1.1" id="Capa_1" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 16 16" style="enable-background: none; height: 100%; width: 100%;">
<g>
<g>
<path style="fill:#010002;" d="M 10.24-0.373-0.744-0.817-1.785C4.524,3.154,3.684,2.973,3.027c-0.27,-0.261-0.524,0.52-0.747c0.654,0,1.491-0.19,2.423-0.52c1.102,0,1.616-0.469,1.856-0.8c0.577,0,1.356,0.176,2.244,0.4C2.397,6.108,1.663,4.945,2.024c-0.458,0-0.787-0.114-0.943-0.1c0.041,0.591,0.109,1.277,0.222c0.258-0.13,0.51-0.251,0.756-0.2C9.06,3.977,8.517,4.242,7.959,L6.377,5.476l-0.26,0.178C6.145c-0.258,0.13-0.51,0.251-0.757c0.526-0.227,1.07-0.492,1.627-0.228c-0.228-0.097-0.462-0.204-0.695C9.685,9.281,9.391,9.482,9.083C6.636,9.511,6.327,9.311,6.027c0.288-0.204,0.583-0.406,0.892c0.322,0.197,0.631,0.398,0.932c0.459,0,0.787,0.115,0.943,0.3c-0.041-0.59-0.107-1.278-0.222L10.994,7.624z M13.974,11.646c0.106-0.686,0.172-1.355,0.213
</g>
</g>
<g>

```

Atom IDE



The screenshot shows the WebStorm IDE interface with the file 'contentController.js' open. The code is a Node.js module. The WebStorm interface includes a file tree on the left, a code editor with syntax highlighting, and a 'TODO' tool window at the bottom. The 'TODO' window lists 994 items across 457 files.

```

Content.findOneAndUpdateAsync(search, { upsert: true })
  .then(function(doc) {
    res.status(200).send();
  })
  .catch(function(err) {
    next(new CustomError('error', 500, 'contentController'));
  });
}

function get(req, res, next) {
  if (!req.params.type || !req.params.section)
    next(new CustomError('error', 500, 'contentController'));
  var select = {
    _id: 0
  };
  Content.findAsync({
    type: req.params.type,
    section: req.params.section
  })
}

var select = {
  _id: 0
};

```

TODO: Project Current File Scope Based Default Changelist

Found 994 TODO items in 457 files

- PSE-Selbstentwicklungsraum-V2 (994 items in 457 files)
 - api (1 item in 1 file)
 - app (3 items in 3 files)
 - assets (155 items in 44 files)
 - node_modules (833 items in 407 files)
 - tests (1 item in 1 file)
 - gulpfile.js

WebStorm IDE

HTML Prüfung

- Browser ignorieren normalerweise unbekannte oder falsche Tags und Attribute
 - ⇒ es gibt keine Fehlermeldung, allenfalls Fehlverhalten
 - ⇒ das Verhalten im Fehlerfall hängt stark vom Browser ab
- seit HTML5 sind ganz offiziell viele Konstrukte erlaubt, die in HTML 4 noch Fehler gewesen wären
 - ⇒ diverse (schließende) Tags sind optional
 - ⇒ unbekannte Attribute werden ignoriert
- deshalb: HTML Code vor der Veröffentlichung prüfen
 - ⇒ anhand der Spezifikation: Syntax, Tag- und Attributnamen, Schachtelungsregeln, etc. mit einem Validator (z.B. validator.w3.org)
 - ⇒ für HTML5 eventuell zusätzlich auf die Einhaltung von "Programmierrichtlinien" prüfen
 - ⇒ auch generiertes HTML (z.B. aus PHP) prüfen!

2.1.4 HTML Werkzeuge

HTML Validator – W3C

<http://validator.w3.org>

The screenshot shows the W3C Markup Validation Service interface. At the top, it says "Markup Validation Service" and "Check the markup (HTML, XHTML, ...) of Web documents". Below that, there are navigation links: "Jump To:", "Notes and Potential Issues", and "Congratulations · Icons". A green banner at the top of the main content area says "This document was successfully checked as HTML5!". The "Result:" section shows "Passed, 2 warning(s)". The "Source :" section displays the following HTML code:

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="UTF-8" >
    <title>Text des Titels</title>
    <link rel="stylesheet" href="style.css"> <!-- </link> -->
  </head>
  <body>
    <p>Inhalt ohne Abschlusstag
    <p>Eigentlicher Inhalt</P>
    Inhalt ohne Format <br>
  <!-- </body> -->
<!-- </html> -->
```

- ⇒ HTML5 wird validiert, obwohl es einige bedenkliche Konstrukte enthält

WYSIWYG Tools

GoLive, Dreamweaver, FrontPage, Publisher

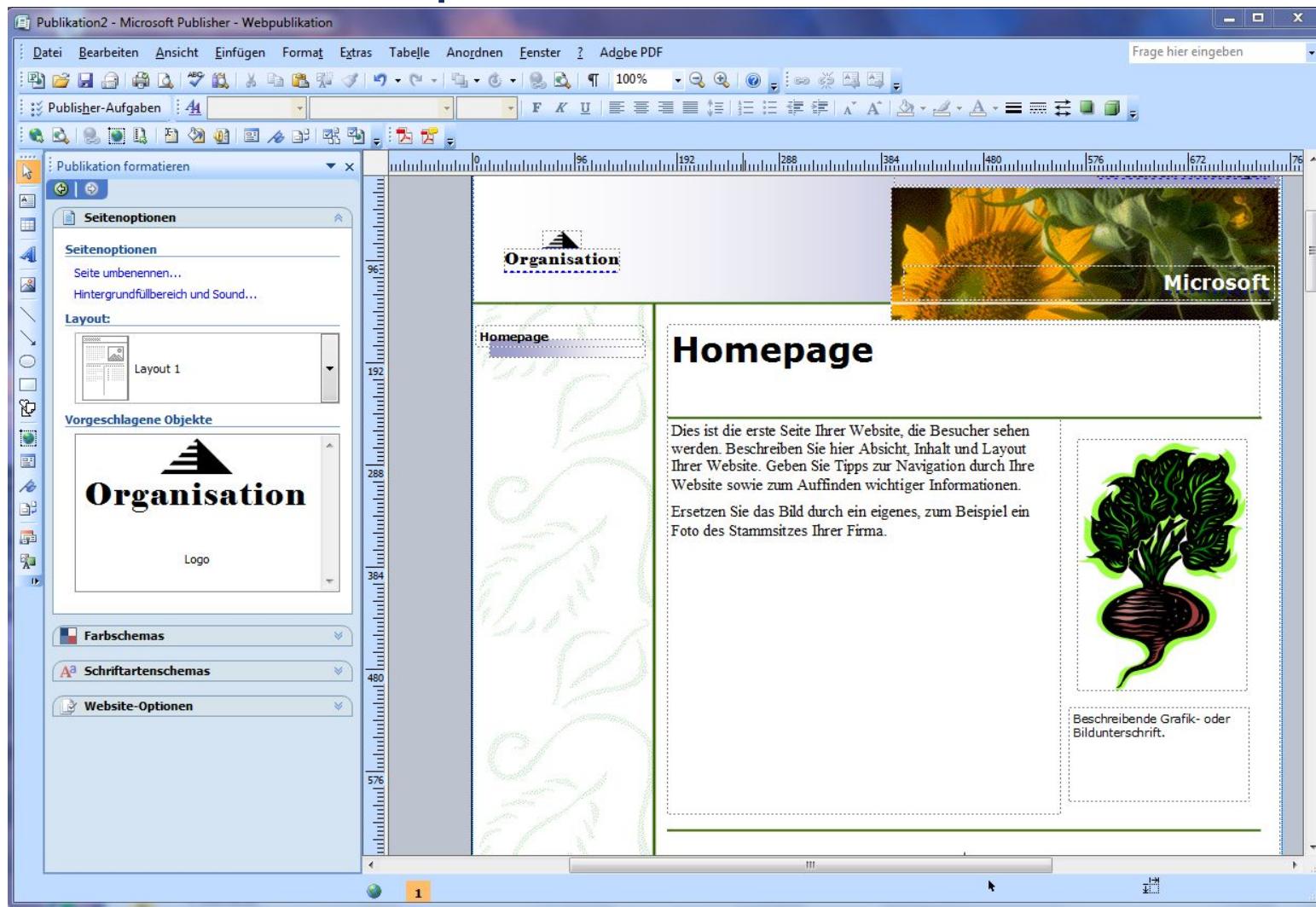
- Hoher Komfort (ähnlich Word)
 - ⇒ HTML wird nicht mehr "programmiert"; Anweisungen und Attribute werden problemorientiert über Dialoge definiert
 - ⇒ HTML ist nur "internes Datenformat"
 - ⇒ HTML kann vom Autor betrachtet werden; muss aber nicht
- nur die Formatiermöglichkeiten von HTML sind erlaubt
 - ⇒ 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

2.1.4 HTML Werkzeuge

WYSIWYG Tool – Beispiel: Microsoft Publisher 2007



Export aus anderen Tools

- früher unbrauchbar, mittlerweile erstaunlich gut
 - ⇒ z.B. PowerPoint für MSIE mit XML und Style Sheets
 - ⇒ z.B. MS Excel Leistungsübersicht im Online Belegsystem
 - ⇒ sogar in der Größe skalierbar
- generierter HTML-Code sehr komplex, viele Dateien
 - ⇒ praktisch schon fast wieder proprietäres Dateiformat
- nicht unterstützt: HTML-Rohform als Zwischenstufe bei Konvertierung existierender Dokumente
 - ⇒ Export für Nachbearbeitung in HTML Tool

Zusammenfassung HTML

- HTML-Grundlagen
 - ⇒ Grundgerüst: DOCTYPE, <html>, <head>, <body>, <title>, charset...
 - ⇒ Schreibregeln und Syntax
 - ⇒ Tags und Attribute
 - ⇒ Hyperlinks
- Formulare
 - ⇒ Buttons, Listen, Datenübermittlung
- Werkzeuge

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

Hochschule Darmstadt

Fachbereich Informatik

2.1.5 HTML Layout



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

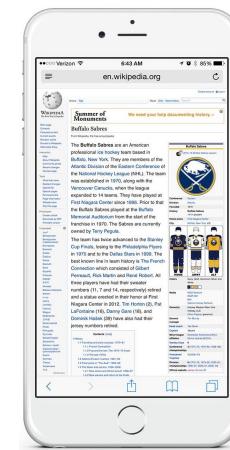
fbi

FACHBEREICH INFORMATIK

2.1.5 HTML Layout

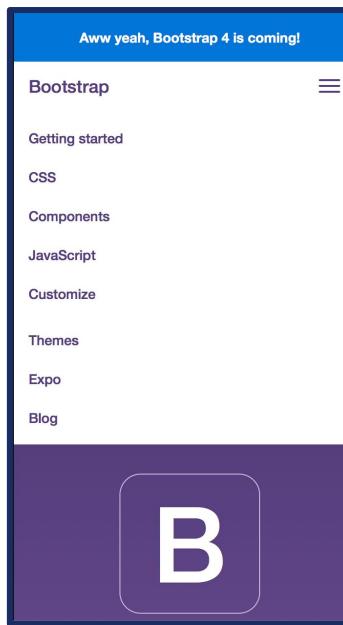
Problematik des Layouts

- HTML beschreibt Fließtext
 - ⇒ absichtlich keine feste Platzierung
 - ⇒ Anordnung der Tags erfolgt nach der Reihenfolge in der HTML-Datei
 - ⇒ keine Überdeckung von Objekten
- Darstellung hängt vom System des Betrachters ab
 - ⇒ Egal ob 4K-Monitor, Smart TV, Handy, Tablet oder Netbook
 - ⇒ Informationsdarstellung mit sehr verschiedenen Auflösungen und Schriftgrößen (ggfs. mit automatischem Zoom)

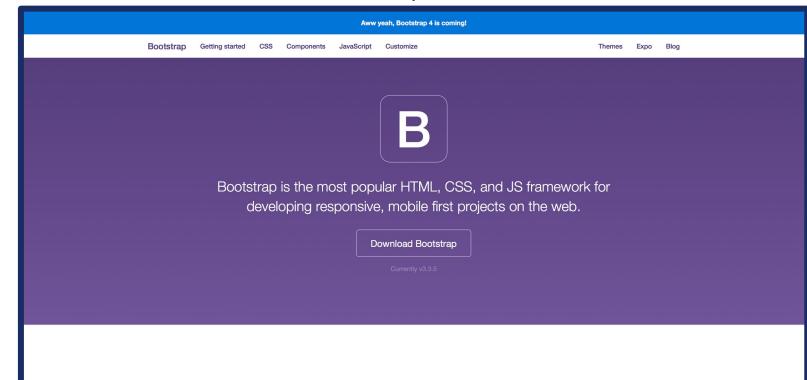


2.1.5 HTML Layout

Problematik des Layouts (I)

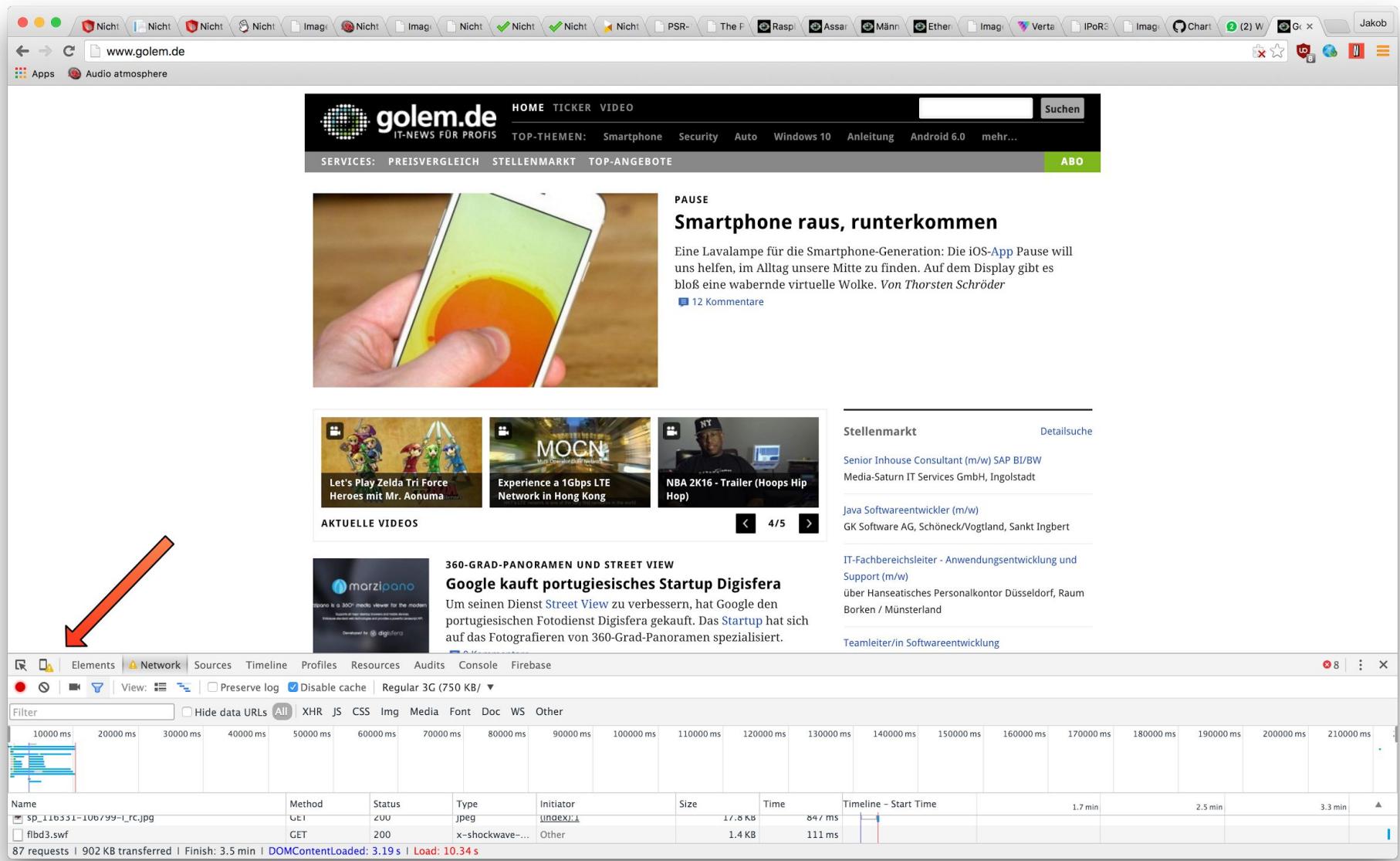


Gemacht für Maus als eingabe



Gemacht für Finger als eingabe

Zum testen der Darstellung auf verschiedenen Displaygrößen



The screenshot shows a web browser window displaying a news article from [golem.de](http://www.golem.de). The article is titled "Smartphone raus, runterkommen" and discusses a Lavalamp app for iOS. Below the main content, there are three video thumbnails under the heading "AKTUELLE VIDEOS". The developer tools are open at the bottom, specifically the Network tab, which shows a timeline of network requests. A red arrow points to the Network tab.

golem.de IT-NEWS FÜR PROFIS

TOP-THEMEN: Smartphone Security Auto Windows 10 Anleitung Android 6.0 mehr...
SERVICES: PREISVERGLEICH STELLENMARKT TOP-ANGEBOTE

PAUSE

Smartphone raus, runterkommen

Eine Lavalampe für die Smartphone-Generation: Die iOS-App Pause will uns helfen, im Alltag unsere Mitte zu finden. Auf dem Display gibt es bloß eine wabernde virtuelle Wolke. Von Thorsten Schröder

12 Kommentare

AKTUELLE VIDEOS

360-GRAD-PANORAMEN UND STREET VIEW

Google kauft portugiesisches Startup Digisfera

Um seinen Dienst Street View zu verbessern, hat Google den portugiesischen Fotodienst Digisfera gekauft. Das Startup hat sich auf das Fotografieren von 360-Grad-Panoramen spezialisiert.

Stellenmarkt Detailsuche

Senior Inhouse Consultant (m/w) SAP BI/BW
Media-Saturn IT Services GmbH, Ingolstadt

Java Softwareentwickler (m/w)
GK Software AG, Schöneck/Vogtland, Sankt Ingbert

IT-Fachbereichsleiter - Anwendungsentwicklung und Support (m/w)
Über Hanseatisches Personalkontor Düsseldorf, Raum Borken / Münsterland

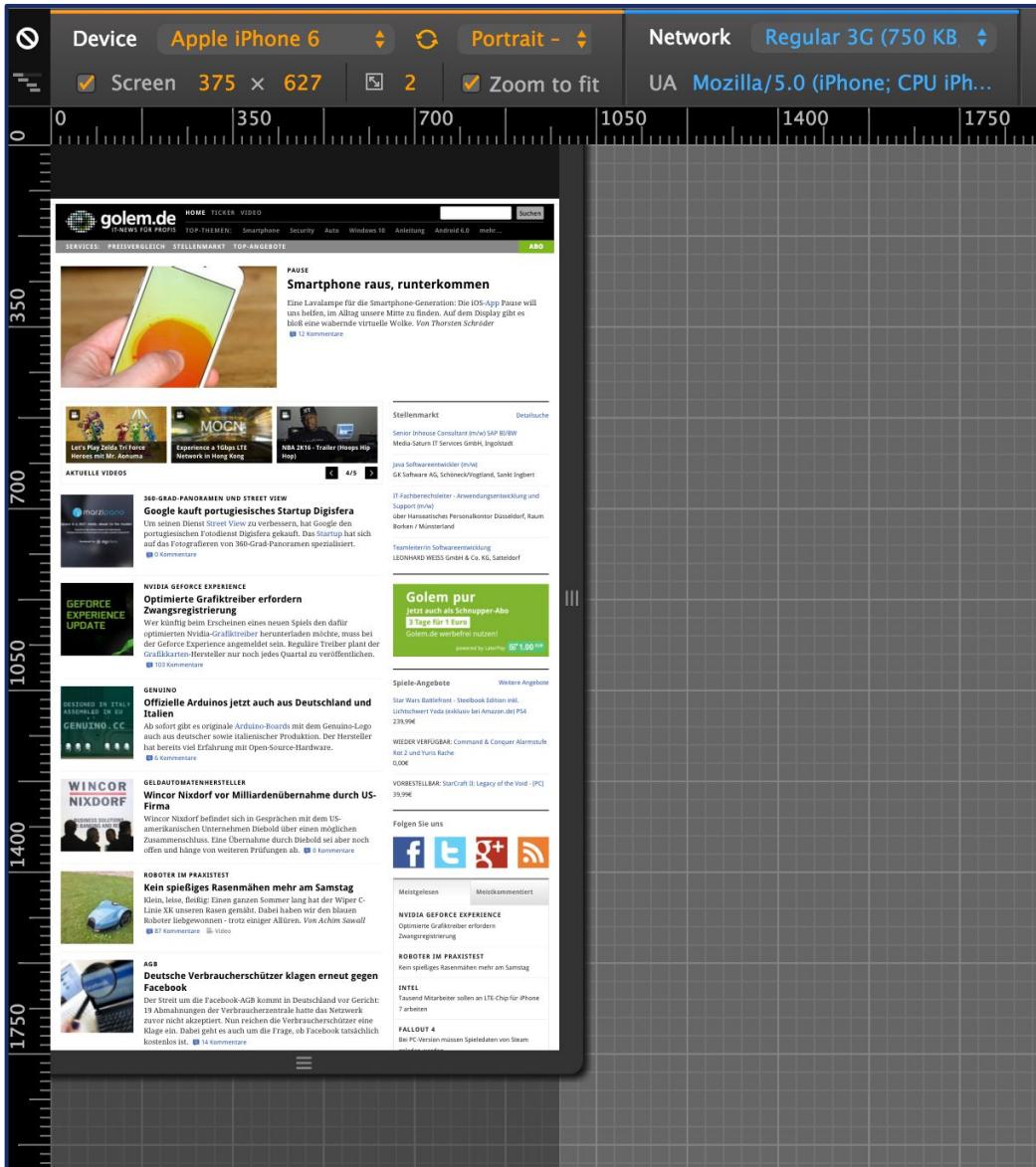
Teamleiter/in Softwareentwicklung

Elements Network Sources Timeline Profiles Resources Audits Console Firebase

Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Other

Name	Method	Status	Type	Initiator	Size	Time	Timeline – Start Time	1.7 min	2.5 min	3.3 min
sp_116331-106799-l_rc.jpg	GET	200	jpeg	index:1	17.8 kB	847 ms				
fb3.swf	GET	200	x-shockwave-...	Other	1.4 kB	111 ms				

87 requests | 902 KB transferred | Finish: 3.5 min | DOMContentLoaded: 3.19 s | Load: 10.34 s



Problematik des Layouts (II)

- Dynamische (sich anpassende) Layouts sind schwierig zu entwerfen
 - ⇒ die meisten Seitengestalter denken in statischen Layouts
 - ⇒ traditionell sind in HTML zwei "Layoutmanager" verfügbar:
 - `<table>` für Layoutzwecke verpönt
 - `<frameset>` seit HTML5 verboten
 - ⇒ stattdessen wird heute CSS verwendet
- Entwurfsmethodik sinngemäß übertragen
 - ⇒ siehe "Dynamisches Layout" in "Entwicklung nutzerorientierter Anwendungen"
- Eine Tabelle ist (immer noch) häufig Basis des Seitenlayouts
 - ⇒ statisches Layoutraster durch Bemaßung in Pixel
 - ⇒ dynamisches Layoutraster durch Bemaßung in Prozent

Tabelle als Layoutmanager

verpönt im Hinblick
auf Barrierefreiheit !

- Eine Tabelle ist (immer noch) häufig Basis des Seitenlayouts
 - ⇒ normalerweise „blinde“ Tabelle, d.h. ohne Rand
 - ⇒ Freiformen, Grafiken, Buttons etc. als eingebettete Elemente
 - ⇒ verhindert mögliche Layoutanpassungen wegen tabellarische Anordnung von nicht-tabellarischem Inhalt

	Margherita	4,00 €	Warenkorb	
	Salami	4,50 €	Roma Margherita Napoli	14,50 €
	Hawaii	5,50 €		
	Tonno	5,00 €		

- ⇒ ein Screenreader liest die Tabelle von links nach rechts und von oben nach unten



Layoutvorbereitung für die CSS-Formatierung

- In HTML wird eine Seite als inhaltlich logische Sequenz von Blöcken aufgebaut
 - ⇒ jeder Block wird mit einem Tag gekennzeichnet
 - mit einem passenden Standard-Tag z.B. `<h1>...</h1>`
 - oder sonst mit `<div>...</div>`
 - ⇒ Elemente, die speziell formatiert werden sollen, aber keinen Block erzeugen sollen, werden durch `...` gekennzeichnet
 - ⇒ die Reihenfolge innerhalb der HTML-Seite ist so gewählt, dass sie der logischen Reihenfolge entsprechen
 - ⇒ diese Sequenz definiert auch die Vorlesereihenfolge des Screenreaders
- Die einzelnen Blöcke werden dann später mit CSS formatiert, positioniert und ausgerichtet
 - ⇒ z.B. CSS-Attribute für den Textfluss: `float`, `margin`, `clear`
 - ⇒ www.fbi.h-da.de ist so aufgebaut

2.1.5 HTML Layout

Layoutvorbereitung für die CSS-Formatierung - Beispiel

```
<header><h1>Kopfzeile</h1></header>
<nav><ul>
  <li>Menu1</li> <li>Menu2</li>
</ul></nav>
<section>
  <article>Inhalt1</article>
  <article>Inhalt2</article>
</section>
<footer>Fußzeile</footer>
```

Nicht gerade schön – aber logisch
genau das, was wir mit HTML wollen!



Kopfzeile

- Menu1
- Menu2

Inhalt1
Inhalt2
Fußzeile

Zusammenfassung

- Problematik des Layouts
 - ⇒ WYSI(not)WYG
 - ⇒ Barrierefreies Layout
- Layout in HTML unerwünscht
 - ⇒ Tabelle für Layout-Zwecke
 - ⇒ Nachteile mit Screenreader
- Vorbereitung für die CSS-Formatierung

Jetzt beherrschen Sie die Grundlagen von HTML und können die Elemente auf einer HTML-Seite anordnen!

Hochschule Darmstadt

Fachbereich Informatik

2.2 CSS



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

HTML ohne CSS (Beispiel Zen Garden ohne CSS)

css Zen Garden

The Beauty of CSS Design

A demonstration of what can be accomplished visually through CSS-based design.

The Road to Enlightenment

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, and broken CSS support.

So What is This About?

There is clearly a need for CSS to be taken seriously by graphic artists. The Zen Garden aims to excite, inspire, and encourage participation. To begin, view some of the existing designs in the list. Clicking on any one will load the style sheet into this very page. The code remains the same, the only thing that has changed is the external .css file. Yes, really.

Requirements

We would like to see as much CSS1 as possible. CSS2 should be limited to widely-supported elements only. The css Zen Garden is about functional, practical CSS and not the latest bleeding-edge tricks viewable by 2% of the browsing public.

Unfortunately, designing this way highlights the flaws in the various implementations of CSS.

[xhtml](#) [css](#)

Select a Design:

- [Under the Seal](#) by [Eric Stoltz](#)
- [Make 'em Proud](#) by [Michael McAughon and Scotty Reifsnyder](#)

Archives:

- [next designs »](#)
- [View All Designs](#)

Resources:

- [View This Design's CSS](#)
- [CSS Resources](#)



Beispiel (Zen Garden mit CSS)



Quelle: <http://www.csszengarden.com>

siehe auch <http://www.oswd.org>

Hochschule Darmstadt

Fachbereich Informatik

2.2.1 CSS Grundlagen



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Cascading Style Sheets

- Definition physischer Attributsätze für
 - ⇒ vordefinierte logische Formate (Überschrift 2, Aufzählung, ...)
 - ⇒ selbstdefinierte Format-Klassen
 - ⇒ einzelne (Text-)Blöcke
- vielfältige gestalterische Attribute
 - ⇒ Schriftart, -größe, -stil, Zeichen- und Zeilenabstand, Einrückung
 - ⇒ Text- und Hintergrundfarbe, Rahmen
- Seitengestaltung für Druck
- freie Platzierung und Überlappung von Objekten
 - ⇒ vgl. Autorensysteme
 - ⇒ Basis für Animation von Objekten

CSS Version	
1.0	1996
2.0	März 1998
2.1	Juni 2011
3.0	Draft 2012
“There will never be a CSS4”	

vergleichbar mit
Formatvorlagen in Word

Potential der CSS

- exakte Bestimmung des Erscheinungsbilds
 - ⇒ wichtiger Schritt in Richtung WYSIWYG
 - ⇒ die variable Bildschirmauflösung bleibt ein Problem
- verbesserte Exportierbarkeit aus anderen Tools
 - ⇒ Erstellung von Webseiten mit gewohnten Tools
 - ⇒ kein Fachwissen und keine Tricks mehr erforderlich
- Optimierung für verschiedene Ausgabemedien
 - ⇒ Bildschirm, Drucker, TV, Tablet, Screenreader...
- 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



2.2.1 CSS Grundlagen

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

- "eingebettet" im HTML-Code

- ⇒ gilt nur für diese eine HTML-Datei

```
<style>
```

```
/* ... Style-Sheet-Definitionen ... */
```

```
</style>
```

gehört in den
HTML-<head>

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>
```

- ⇒ HTML Inline-Tag `` zur Markierung eines Teilbereich eines Objekts

primärer Zweck

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

Unterstützung verschiedener Ausgabemedien

- verschiedene Bereiche innerhalb eines CSS

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

- verschiedene CSS-Dateien in HTML einbinden

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

Hochschule Darmstadt

Fachbereich Informatik

2.2.2 CSS - Formate definieren



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

2.2.2 CSS - Formate definieren

Browser-Default-Formatierung im Vergleich - ohne CSS

- Browser haben unterschiedliche Default-Stile
 - ⇒ hier Internet Explorer 6 und Firefox 2



Um ein definiertes Layout zu erhalten,
muss man die Standard-Formate selbst definieren!

siehe [Normalize.css](#)

2.2.2 CSS - Formate definieren

Standard-Formate modifizieren

■ Definition in CSS

vorzugsweise für Ausgestaltung
logischer Formate

```
h3 { text-align: center; color:#33FF00; }  
p { border: 1px solid black;  
    font-family:Arial, Helvetica; }  
* { color:green; } /* Universalselektor gilt für alle Tags*/  
ul { list-style:none; } /* verbirgt die Aufzählungspunkte */
```

■ Anwendung in HTML

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

kein Attribut
in HTML

⇒ ohne CSS zeigt der Browser die "schlichte" Version

2.2.2 CSS - Formate definieren

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

Eine Überschrift mit Style-Sheets

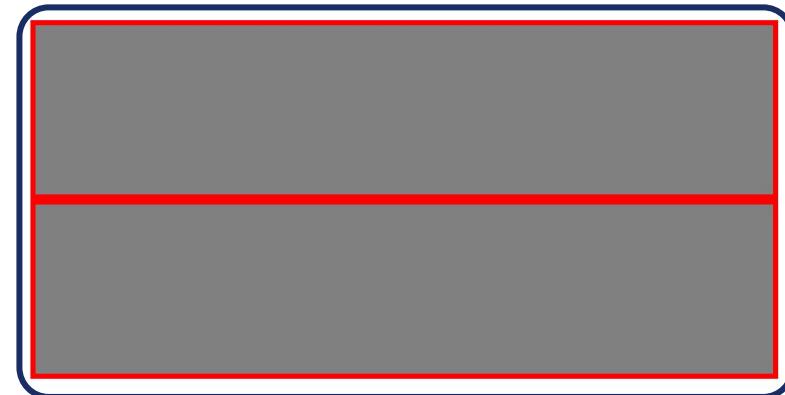
Ein Fließtext mit *Style-Sheets*

HTML

```
1 <div id="foo">
2   </div>
3 <div id="bar">
4   </div>
```

CSS

```
1 #foo, #bar {
2   border: 3px solid red;
3   height: 100px;
4   background-color: grey;
5 }
```



Mit Komma zwischen Selectoren gilt für beide IDs.

HTML

```
1 <div id="foo">
2   <div id="bar">
3     </div>
4   </div>
5
```

CSS

```
1 #foo #bar {
2   border: 3px solid red;
3   height: 100px;
4   background-color: grey;
5 }
```



Ohne Komma zwischen Selectoren gilt nur wenn auf ID 1, ID 2 folgt.

2.2.2 CSS - Formate definieren

Eigene Format-Klassen

■ Definition in CSS

- ⇒ Unterklassen für Standard Formate

```
p.Hinweis { color:red; }  
p.Fussnote { color:black; }
```

- ⇒ allgemein verwendbar

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

aber keine Klasse mit
Ableitung etc. wie in OO

■ 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

2.2.2 CSS - Formate definieren

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

2.2.2 CSS - Formate definieren

Individuelle Objekt Formate

```
<h1 id="title">Websitenname</h1>

<input type="text" autocomplete="on" class="inputfield"/>
```

Einmalig auf Website
also **ID**

Mehrfach vorhanden auf
Website also **class**

2.2.2 CSS - Formate definieren

Pseudo-Formate

- Sonderfall:
 - ⇒ definieren Eigenschaften, die keine Attribute von HTML-Blöcken sind
- Darstellung von Hyperlinks - Festlegung in CSS
- ```
a:link { color:blue;} /* normaler Link */
a:visited { color:green;} /* bereits besucht */
a:active { color:red;} /* gerade angeklickt */
a:hover { color:yellow;} /* unter dem Mauszeiger */
a:focus { color:black;} /* mit Tastatur angewählt */
```
- ```
p:first-line { font-weight:bold; }  
p:first-letter { font-size:36pt; color:red; }
```

M

an 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.

2.2.2 CSS - Formate definieren

Weitere Selektoren (eine kleine Auswahl)

- Flexible Möglichkeit, um Tags auszuwählen:
 - ⇒ **[target]** Wählt alle Tags mit einem Attribut target
 - ⇒ **[target=xyz]** Wählt alle Tags mit einem Attribut target, das den Wert xyz hat
- Auswahl mit Parametern
 - ⇒ **p:nth-child(2)** Wählt alle p-Tags, die das 2-te Kind von irgendeinem Tag sind
 - ⇒ **p:nth-child(3n+1)** Wählt alle p-Tags, die das 1-te, 4-te, 7-te,... Kind von irgendeinem Tag sind
 - ⇒ Anwendung für "gestreifte" Tabellen (**even** und **odd** sind vordefiniert):
 - **tr:nth-child(even)** { background-color: LightGrey; }
 - **tr:nth-child(odd)** { background-color: white; }

Es gibt noch viele weitere Selektoren!

Hochschule Darmstadt

Fachbereich Informatik

2.2.3 CSS - Attribute



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Farben und Hintergrundbilder

Farbschema entwerfen mit
<http://kuler.adobe.com>

■ Farbattribute

background-color Hintergrundfarbe

color Textfarbe

border-color Rahmenfarbe

text-shadow schattierter Text

`background-color: white;`

■ 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

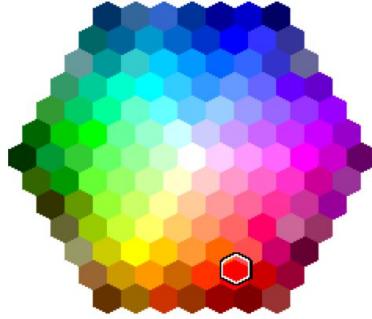
Darkorange diverse Farben mit Namen

■ Hintergrundbild

⇒ nicht nur für gesamte Seite, sondern auch für einzelne Blöcke

background-image:url(bild.gif)

Pick a Color:



Selected Color:



#FF0000

rgb(255, 0, 0)

Red

Or Enter a Color:

Or Use HTML5:



Shades:



Hex:

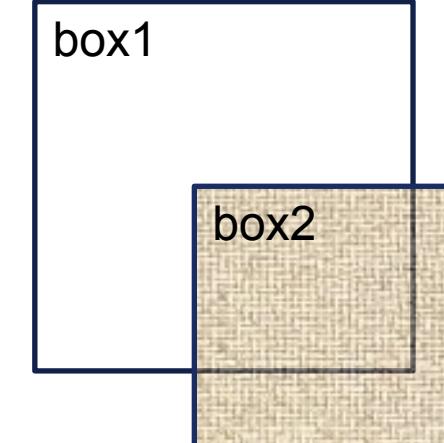
#000000
#1A0000
#330000
#4C0000
#660000
#800000
#990000
#B20000
#CC0000
#E60000
#FF0000
#FF1919
#FF3333
#FF4D4D
#FF6666
#FF8080
#FF9999
#FFB2B2
#FFCCCC
#FFE6E6
#FFFFFF

Transparenz

Durchsichtige Elemente

- ⇒ `opacity: 0.5;`
- ⇒ Einstellung über eine Zahl zwischen 0 (voll transparent) und 1 (volle Deckkraft)
- ⇒ Macht das gesamte Element mit Rahmen, Hintergrund etc. transparent
- ⇒ Ein- und Ausblendeffekte (mit Javascript)

Vorsicht alle davon betroffenen Elemente können nicht wieder weniger transparenz bekommen!



Transparent als „Farbe“

- ⇒ `background-color: transparent;`
- ⇒ `background-color: rgba(220,160,140,0.5);`
- ⇒ Macht nur das jeweilige Element transparent

2.2.3 CSS - Attribute

Schrift

- **font-family:**
 - ⇒ Arial, Helvetica, "Times New Roman"
 - ⇒ serif, sans-serif, cursive, fantasy, monospace
- **font-style:**
 - ⇒ italic, normal
- **font-size:**
 - ⇒ 12pt, 35px, 3em, 1.5cm, large
- **font-weight:**
 - ⇒ bold, bolder, lighter, 100 .. 900
- **font:**
 - ⇒ kompakte Kombination o.g. Attributwerte

Vorsicht was passiert
wenn Schriftart auf
Gerät nicht
verfügbar?!
Deshalb [Google Fonts](#)

Die Bedeutung der
verschiedenen Maßeinheiten
kommt im nächsten Abschnitt!

2.2.3 CSS - Attribute

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
- Achtung: width und height beziehen sich auf den Inhalt !



Screenshot of the golem.de website showing a news article about a smartphone pause app.

Header:

- golem.de - IT-NEWS FÜR PROFIS
- HOME TICKER VIDEO
- TOP-THEMEN: Smartphone, Security, Auto, Windows 10, Anleitung, Android 6.0, mehr...
- SERVICES: PREISVERGLEICH, STELLENMARKT, TOP-ANGEBOTE
- Suchen
- section#index-promo 926 x 240 ABO

Article Preview:

Smartphone raus, runterkommen
 Eine Lavalampe für die Smartphone-Generation: Die iOS-App Pause will uns helfen, im Alltag unsere Mitte zu finden. Auf dem Display gibt es bloß eine wabernde virtuelle Wolke. Von Thorsten Schröder
 18 Kommentare

Job Listings:

- Stellenmarkt: Support Center Manager (m/w) über JobLeads GmbH, Düsseldorf
- Detailsuche

Bottom Navigation:

- Inspektor, Konsole, Debugger, Stilbearbeitu..., Laufzeitanaly..., Netzwerkanal..., Speicher

Developer Tools:

html.js > body.index.golem-flip-std-body > div.golem-flip-std > div#golem-flip-std > div#grandwrapper.golem-zo-grandwrapper > div#screen > section#index-promo > header.cluster-header

926x240 relative

Außenabstand 0

Rand 0

Innenabstand 0

20 0 436 490x240 0 0 0 0

```
.bar {  
    margin-top: 5px;  
    margin-right: 10px;  
    margin-bottom: 15px;  
    margin-left: 20px;  
}
```



```
.foo {  
    margin: 5px 10px 15px 20px;  
}
```



2.2.3 CSS - Attribute

Ausrichtung und Rand

Ausrichtung

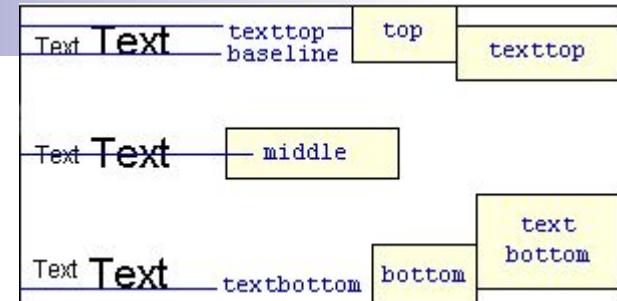
- ⇒ `line-height` Zeilenhöhe in Längenmaß
- ⇒ `text-indent` Texteinrückung in Längenmaß
- ⇒ `text-align: left, center, right, justify` (Blocksatz)
- ⇒ `vertical-align: top, middle, bottom, text-top, text-bottom`

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`

Abgerundete Ecken

- ⇒ `border[-top, -bottom][-left, -right]-radius: x radius y radius`
- ⇒ z.B. `border-top-left-radius: 3em 2em` oder `border-radius: 5%`

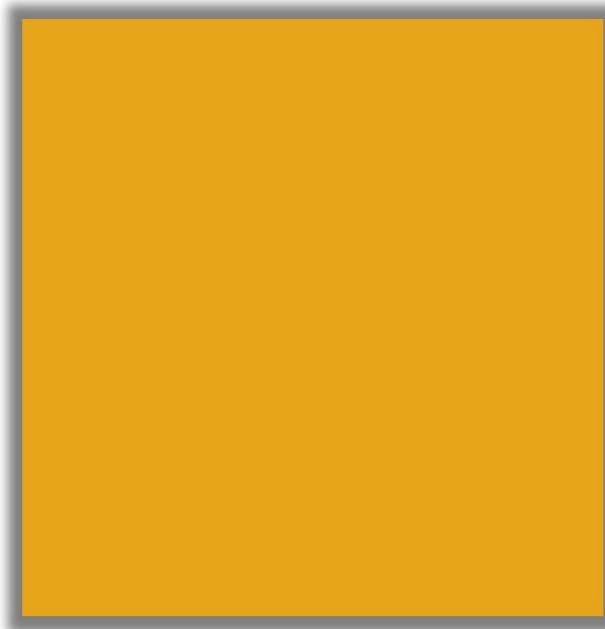


Quelle: SelfHTML

Ausrichtung und Rand

Schatten

- ⇒ `box-shadow: inset x-Achse y-Achse Blur-radius Spread-radius
rgba(r,g,b, Opacity)`
- ⇒ z.B. `box-shadow:1px 1px 10px 10px rgba(0,0,0, 0.5);`



2.2.3 CSS - Attribute

Erscheinungsbild einer Tabelle

```
<table>  
  <tr> <th></th> <th></th> <th></th> </tr>  
  <tr> <td></td> <td></td> <td></td> </tr>  
  <tr> <td></td> <td></td> <td></td> </tr>  
</table>
```

Erscheinungsbild einer Tabelle

<th></th>	<th></th>	<th></th>
<td></td>	<td></td>	<td></td>
<td></td>	<td></td>	<td></td>
</table>		

- Tabellen werden mit den üblichen Elementen formatiert:
 - ⇒ `width`, `height`, `padding`, `border`, `margin`,
 - ⇒ `text-align`, `vertical-align`
- Üblicherweise werden die Linien angeschaltet
 - ⇒ `table, th, td { border: 1px solid black; }`
 - ⇒ dann hat aber jede Zelle einen Rahmen, d.h. die Linien sind doppelt
Lösung: `border-collapse: collapse;`
- Beispiel: Gestreifte Tabelle

```
tr:nth-child(even) { background-color: LightGrey; }
tr:nth-child(odd) { background-color: white; }
th {background-color:black; color:white; }
table, th, td { border: 1px solid black;
border-collapse:collapse;}
```

HTML

```
1 <table>
2   <tr>
3     <th>THead1</th>
4     <th>THead2</th>
5     <th>THead3</th>
6   </tr>
7   <tr>
8     <td>TData1</td>
9     <td>TData2</td>
10    <td>TData3</td>
11  </tr>
12  <tr>
13    <td>TData1</td>
14    <td>TData2</td>
15    <td>TData3</td>
16  </tr>
17 </table>
```

CSS

```
1 td {
2   border: 1px solid red;
3 }
4
5 th {
6   border: 1px solid green;
7 }
8
9 table {
10   border: 1px solid blue;
11   //border-collapse: collapse;
12 }
```

THead1	THead2	THead3
TData1	TData2	TData3
TData1	TData2	TData3

HTML

```
1 <table>
2   <tr>
3     <th>THead1</th>
4     <th>THead2</th>
5     <th>THead3</th>
6   </tr>
7   <tr>
8     <td>TData1</td>
9     <td>TData2</td>
10    <td>TData3</td>
11  </tr>
12  <tr>
13    <td>TData1</td>
14    <td>TData2</td>
15    <td>TData3</td>
16  </tr>
17 </table>
```

CSS

```
1 td {
2   border:1px solid red;
3 }
4
5 th {
6   border: 1px solid green;
7 }
8
9 table {
10   border: 1px solid blue;
11   border-collapse: collapse;
12 }
```

THead1	THead2	THead3
TData1	TData2	TData3
TData1	TData2	TData3

HTML

Tidy

CSS

Tidy

```
1 ▼ td {  
2     border:1px solid red;  
3 }  
4  
5 ▼ th {  
6     border: 1px solid green;  
7 }  
8 ▼ tr:nth-child(even) { background-color:  
    LightGrey; }  
9 ▼ tr:nth-child(odd) { background-color:  
    white; }  
10 ▼ table {  
    border: 1px solid blue;  
    border-collapse: collapse;  
11 }  
12 }  
13 }
```

THead1	THead2	THead3
TData1	TData2	TData3

Zeigen und Verbergen

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

display:

inline Element wird im laufenden Textfluss angezeigt. Der Text "fließt" in Lücken, welche die anderen Elemente bieten

block Rechteckig begrenztes Element steht alleine in einer Zeile

inline-block Rechteckig begrenztes Element, das als Block im Textfluss bleibt (siehe folgendes Beispiel)

none Element wird nicht angezeigt, folgende Blöcke verschieben sich

- Anzeige bzw. Nichtanzeige mit Platzhalter

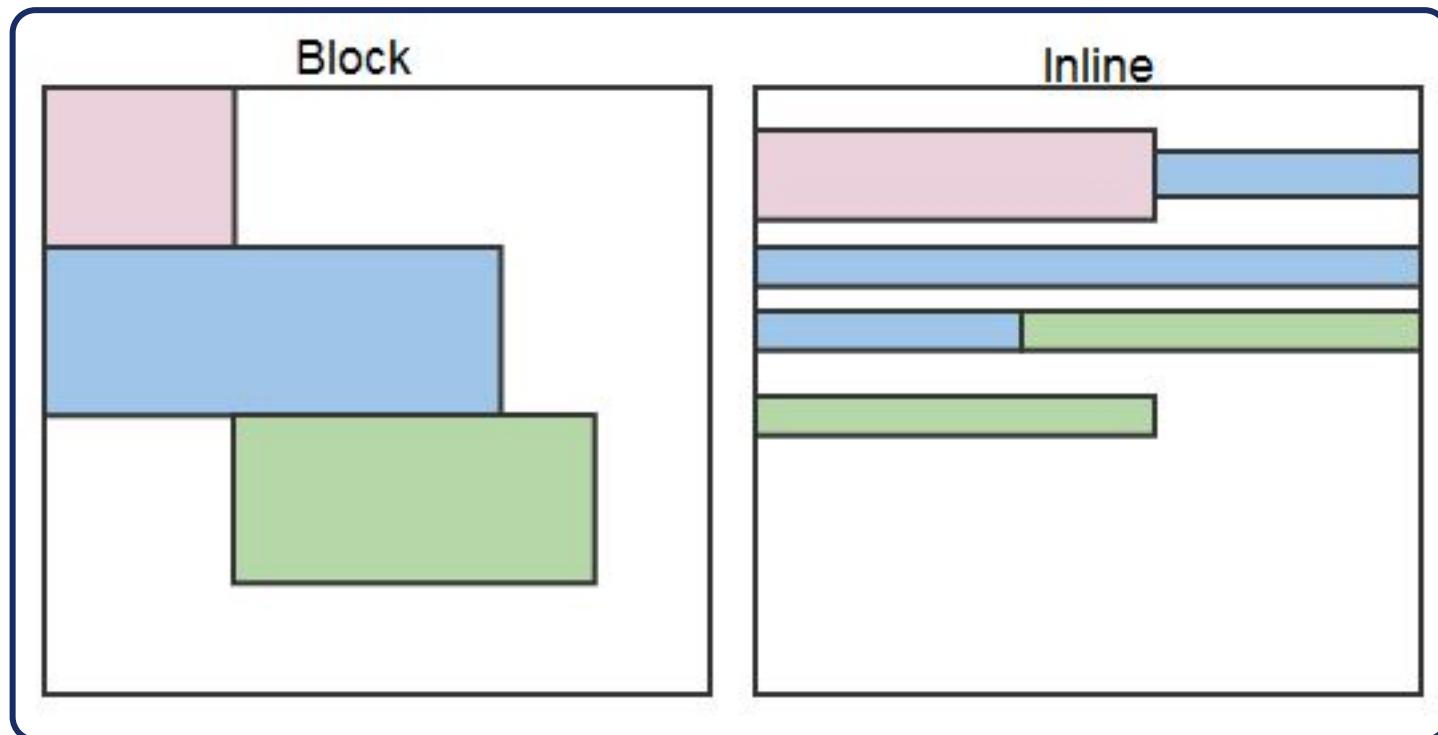
visibility:

visible Element wird angezeigt

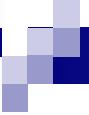
hidden Element wird versteckt (folgende Blöcke bleiben stehen)

Auf- und Zuklappen von Unterpunkten im Inhaltsverzeichnis mit JavaScript

Ähnlich aber nicht gleich!



<http://blog.4psa.com/css-display-and-the-basic-box-model/>

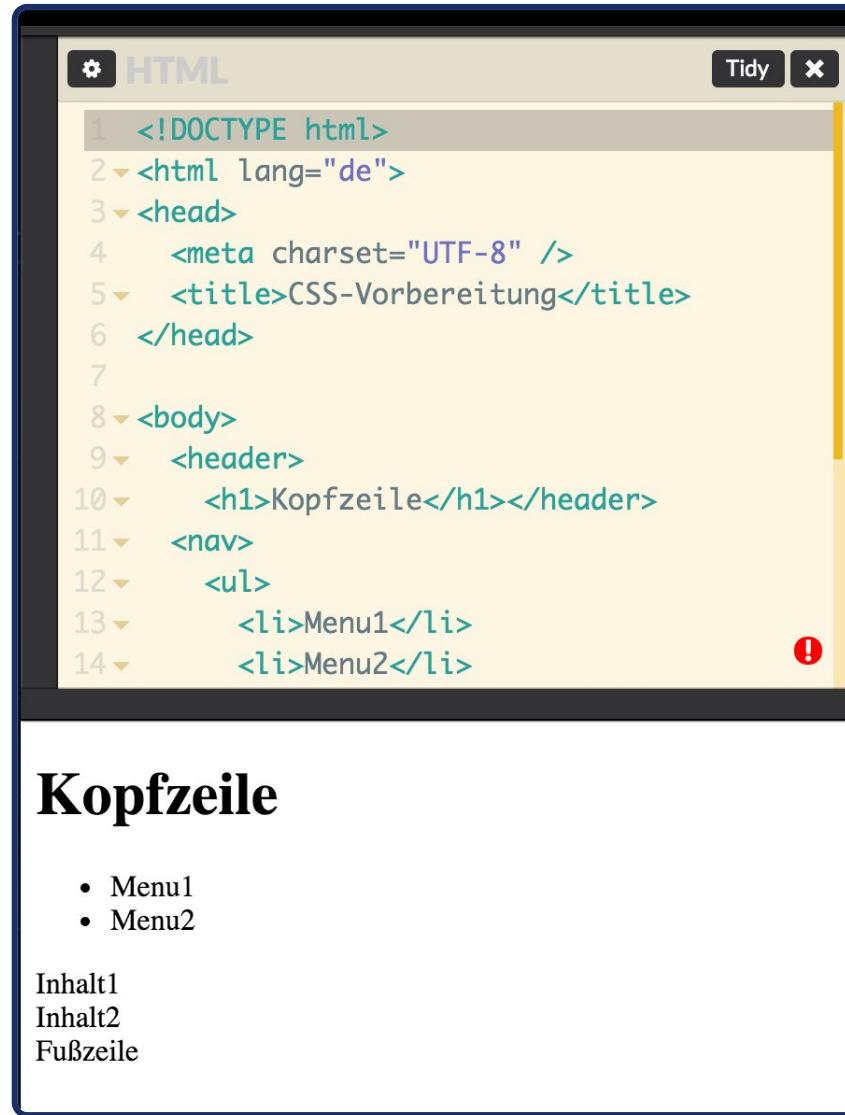


Übersicht zu *display*:

Value	Description
inline	Default value. Displays an element as an inline element (like)
block	Displays an element as a block element (like <p>)
flex	Displays an element as an block-level flex container. New in CSS3
inline-block	Displays an element as an inline-level block container. The inside of this block is formatted as block-level box, and the element itself is formatted as an inline-level box
inline-flex	Displays an element as an inline-level flex container. New in CSS3
inline-table	The element is displayed as an inline-level table
list-item	Let the element behave like a element
run-in	Displays an element as either block or inline, depending on context
table	Let the element behave like a <table> element
table-caption	Let the element behave like a <caption> element
table-column-group	Let the element behave like a <colgroup> element
table-header-group	Let the element behave like a <thead> element
table-footer-group	Let the element behave like a <tfoot> element
table-row-group	Let the element behave like a <tbody> element
table-cell	Let the element behave like a <td> element
table-column	Let the element behave like a <col> element
table-row	Let the element behave like a <tr> element
none	The element will not be displayed at all (has no effect on layout)
initial	Sets this property to its default value. Read about <i>initial</i>
inherit	Inherits this property from its parent element. Read about <i>inherit</i>



Ohne CSS



The screenshot shows a browser developer tools window with the 'HTML' tab selected. The code editor displays the following HTML structure:

```
1 <!DOCTYPE html>
2 <html lang="de">
3 <head>
4   <meta charset="UTF-8" />
5   <title>CSS-Vorbereitung</title>
6 </head>
7
8 <body>
9   <header>
10    <h1>Kopfzeile</h1></header>
11   <nav>
12    <ul>
13     <li>Menu1</li>
14     <li>Menu2</li>
```

A red exclamation mark icon is located in the bottom right corner of the code editor area, indicating a warning or error.

The rendered content below the code editor shows the following structure:

Kopfzeile

- Menu1
- Menu2

Inhalt1
Inhalt2
Fußzeile

HTML + CSS

The screenshot shows a web-based code editor with two panes. The left pane is labeled "HTML" and contains the following code:

```
1 <!DOCTYPE html>
2 <html lang="de">
3 <head>
4   <meta charset="UTF-8" />
5   <title>CSS-Vorbereitung</title>
6 </head>
7
8 <body>
9   <header>
10  <h1>Kopfzeile</h1></header>
11  <nav>
12    <ul>
13      <li>Menu1</li>
14      <li>Menu2</li>
15    </ul>
16  </nav>
17  <section>
18    <article>Inhalt1</article>
19    <article>Inhalt2</article>
20  </section>
21  <footer>Fußzeile</footer>
22 </body>
```

The right pane is labeled "CSS" and contains the following style sheet:

```
1 * { padding:0pt; margin:0pt; } /* keine Default Abstände */
2 body {color:black; background-color: WhiteSmoke;
3   font:1em Verdana;}
4 footer, header {clear: both; text-align:center;
5   color: white; background-color: grey;}
6 article {display: inline-block; width: 15em;
7   border: 1px solid black; margin:5px;}
8 nav {display: block; margin: 5px; border: 2px solid grey;
9   text-align:center; float:left;}
10 nav li {font-size: 1.5ex; margin: 1px;
11   background-color: Lavender;}
12 ul {list-style:none;
13   border: 1pt solid white;
14   text-align: left;}
```

Below the editor, a preview window shows the resulting layout. It features a header with the title "Kopfzeile", a navigation menu with items "Menu1" and "Menu2", two articles with the titles "Inhalt1" and "Inhalt2", and a footer with the text "Fußzeile". A red exclamation mark icon is positioned between the editor panes.

<http://codepen.io/ewahda/pen/PPOeWo?editors=110>

Hochschule Darmstadt

Fachbereich Informatik

2.2.4 Maßeinheiten



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

2.2.4 Maßeinheiten

Absolute Maßeinheiten für statisches Layout

■ px (Pixel)

- ⇒ für Darstellung am Bildschirm
- ⇒ verbreitet für Bilder und eingebettete Objekte
 - vermeidet Skalierung (hässliche Artefakte bei älteren Browsern)

“...pixels don't equal physical pixels. Today, px in a stylesheet means 1/96th of an inch....”

2.2.4 Maßeinheiten

Absolute Maßeinheiten für statisches Layout

■ pt (point), cm, mm, in (inch)

- ⇒ für Druckausgabe mit fester Papiergröße
- ⇒ pt üblich für Schriften
- ⇒ cm, mm, in für Positionen und Abstände

■ Umrechnung

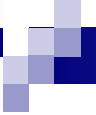
- ⇒ 1 inch = 1 Zoll = 2.54 cm
- ⇒ 1 pt = 1/72 inch = 0.0352778 cm; die echte Größe auf dem Bildschirm ist aber abhängig von der Bildschirmauflösung:
 - Windows 96 dpi oder 120 dpi, Mac und Java 72 dpi
 - reine Rechengröße, berücksichtigt nicht die realen Maße des Bildschirms

dots per inch = Pixel pro Zoll

Relative Maßeinheiten für dynamisches Layout

eigentlich zu bevorzugen

- % prozentualer Anteil bezogen auf
 - ⇒ *für font-size*: die font-size des umschließenden Blocks
 - Sonderfall für <body>: die im Browser wählbare Schriftgröße
 - ⇒ *für width und height*: die Breite bzw. Höhe des umschließenden Blocks
 - Sonderfall für <body>: des Innenbereichs des Browserfensters
 - ⇒ *für margin und padding*: die Breite bzw. Höhe des eigenen Blocks
- em, ex Höhe von M bzw. x der elementeigenen Schrifthöhe
 - ⇒ *für font-size*: die font-size des umschließenden Blocks
 - Sonderfall für <body>: die im Browser wählbare Schriftgröße
 - ⇒ *für andere Attribute*: Wert von font-size des selben Elements
 - ermöglicht z.B. padding / margin abhängig von der Schriftgröße
- rem Höhe von M im root-Element
 - ⇒ in der Regel das Body-Tag mit Schriftgröße von 16pt
 - ⇒ ermöglicht relative Größenfestlegungen unabhängig von der Verschachtelungstiefe und vereinfacht die Berechnung



Absolute Lengths

The absolute length units are fixed and a length expressed in any of these will appear as exactly that size.

Absolute length units are not recommended for use on screen, because screen sizes vary so much. However, they can be used if the output medium is known, such as for print layout.

Unit	Description	
cm	centimeters	<button>Try it</button>
mm	millimeters	<button>Try it</button>
in	inches (1in = 96px = 2.54cm)	<button>Try it</button>
px *	pixels (1px = 1/96th of 1in)	<button>Try it</button>
pt	points (1pt = 1/72 of 1in)	<button>Try it</button>
pc	picas (1pc = 12 pt)	<button>Try it</button>

* Pixels (px) are relative to the viewing device. For low-dpi devices, 1px is one device pixel (dot) of the display. For printers and high resolution screens 1px implies multiple device pixels.

http://www.w3schools.com/cssref/css_units.asp



Relative Lengths

Relative length units specify a length relative to another length property. Relative length units scale better between different rendering mediums.

Unit	Description	
em	Relative to the font-size of the element (2em means 2 times the size of the current font)	Try it
ex	Relative to the x-height of the current font (rarely used)	Try it
ch	Relative to width of the "0" (zero)	
rem	Relative to font-size of the root element	
vw	Relative to 1% of the width of the viewport*	Try it
vh	Relative to 1% of the height of the viewport*	Try it
vmin	Relative to 1% of viewport's* smaller dimension	Try it
vmax	Relative to 1% of viewport's* larger dimension	Try it
%		



Tip: The em and rem units are practical in creating perfectly scalable layout!

* Viewport = the browser window size. If the viewport is 50cm wide, 1vw = 0.5cm.

2.2.4 Maßeinheiten

Zoom statt Wahl der Schriftgröße

vgl. mit Firefox und Internet Explorer:

<http://www.csszengarden.com> Oceanscape

<http://www.tagesschau.de/>



■ historische Situation (Hintergrund des Standards)

- ⇒ Skalierung von Bildern wurde in Browsern schlecht unterstützt (Nearest Neighbour Resampling mit hässlichen Artefakten)
- ⇒ Skalierung der Seite wurde realisiert durch Skalierung des Textes mit neuem Umbruch; Bilder blieben unverändert

■ heutige Situation

- ⇒ gute Skalierung von Bildern kein Problem dank hoher Rechenleistung
- ⇒ Benutzer erwarten Skalierung von Bildern **und** Texten

■ Lösung: Einstellbarkeit der Schriftgröße wird ersetzt durch Zoom

- ⇒ Firefox und Internet Explorer zoomen alle Maßeinheiten gleichartig
 - die Schriftgröße wird geändert und das Layout ggf. neu umgebrochen
 - geht auch für Seiten mit statischem Layout (erfordert i.a. horizontales Scrollen)
 - ist perfekt für Seiten mit dynamischem Layout (kein horizontales Scrollen)
- ⇒ Internet Explorer bietet zusätzlich eine Einstellung für die Textgröße; bei Firefox heißt das Äquivalent "Nur Text zoomen"
 - betrifft standardkonform nur %, em, ex
 - Bilder gehen nicht mit in der Größe
 - nur akzeptabel für Seiten mit dynamischem Layout

leidiges Problem:
viele Webentwickler haben
absolute und relative Maße
nicht sinnvoll eingesetzt;
viele Seiten skalierten
schlecht

entschärft o.g. Problem

Empfehlungen für dynamische Layouts

dagegen für Druckausgabe alles in pt festlegen

- vgl. "Entwicklung nutzerorientierter Anwendungen"
Parameter dynamischer Layouts:
Fenstergröße, Schriftgröße, Textlänge
- Schriftgröße in %, em oder rem (benutzerdefinierbar via Browser)
- Ränder, Außen- und Innenabstände in em (passend zur Schrift)
- Bilder in em (passend zur Schrift)
- Blockgröße
 - ⇒ fensterabhängig in % (falls Umbruch möglich) oder
 - ⇒ schriftabhängig in em (falls zu klein für Umbruch)
- Blockposition
 - ⇒ vorzugsweise fließend (siehe nächster Abschnitt)
 - ⇒ abwägen: fensterabhängig in % oder schriftabhängig in em
 - ⇒ möglichst nicht absolut positionieren

suboptimal:
alles in px festlegen
und sich auf die Zoom-
Funktion moderner
Browser verlassen

generell absolute und relative
Maßeinheiten nicht mischen !

Hochschule Darmstadt

Fachbereich Informatik

2.2.5 CSS - Layout



h_da

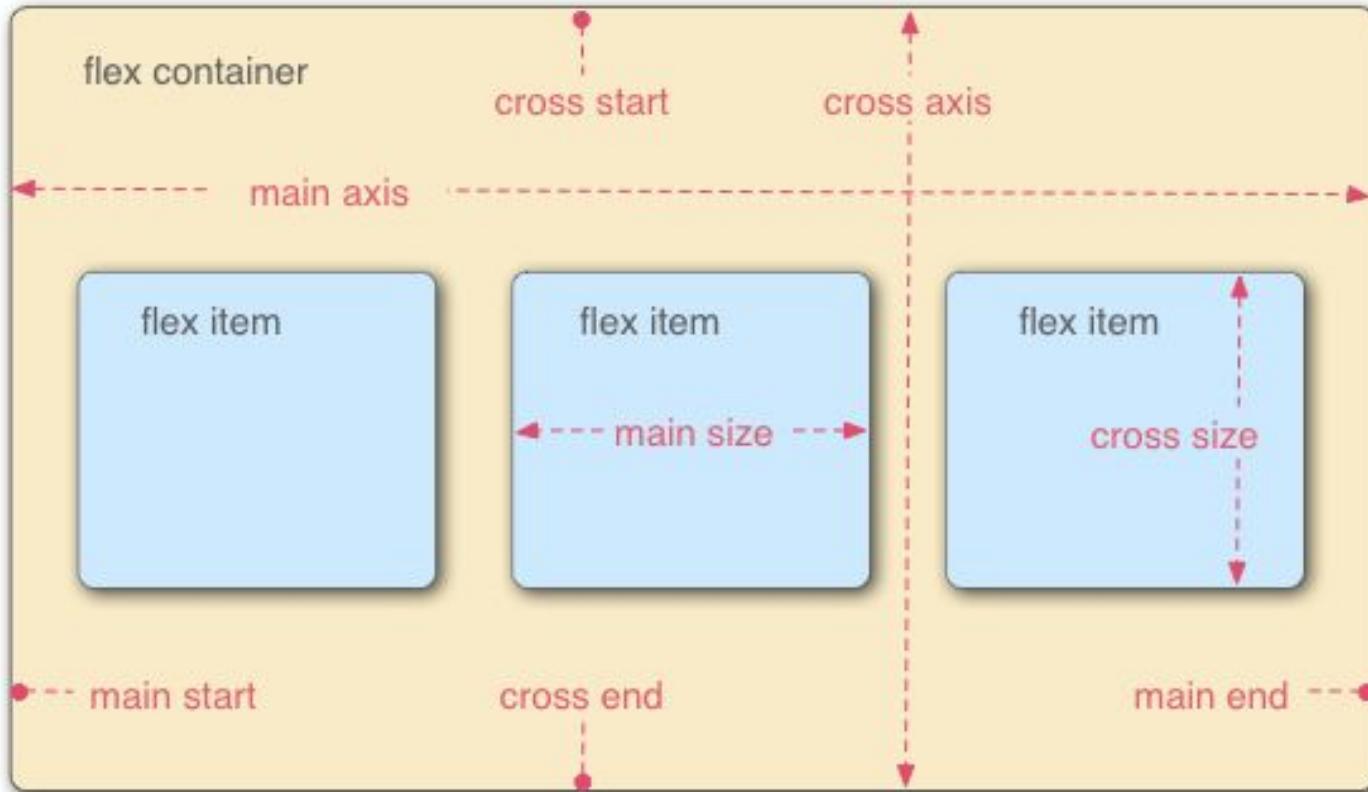
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

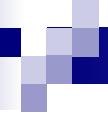
fbi

FACHBEREICH INFORMATIK

Attribute für das Layout

Flexbox





The screenshot shows a web development interface with two code editors and a preview area.

HTML Editor:

```
1 <body>
2   <div class="container">
3     <div class="item">
4       0
5     </div>
6     <div class="item">
7       1
8     </div>
9     <div class="item">
10      2
11    </div>
12    <div class="item">
13      3
14    </div>
15    <div class="item">
16      4
17    </div>
```

CSS Editor:

```
1 .container{
2   border: 1px solid green;
3   width: 50rem;
4   height: 15em;
5   display:flex;
6   //flex-direction: column;
7   flex-direction: row;
8   flex-wrap: wrap;
9   justify-content: flex-start;
10 }
11 .item{
12   margin: 0.5em;
13   border: 1px dotted red;
14   width: 5em;
15   height: 5em;
16 }
```

Preview Area:

The preview shows a horizontal grid of 10 boxes, indexed from 0 to 9. Boxes 0 through 7 are in the first row, and boxes 8 and 9 are in the second row. Each box has a red dotted border and contains its index number.

<http://codepen.io/ewahda/pen/OyOZzM?editors=110>



Container Attribute



flex-direction



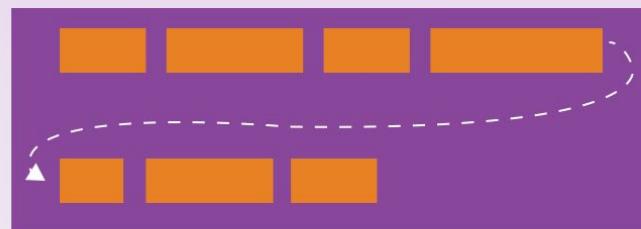
This establishes the main-axis, thus defining the direction flex items are placed in the flex container. Flexbox is (aside from optional wrapping) a single-direction layout concept. Think of flex items as primarily laying out either in horizontal rows or vertical columns.

CSS

```
row | row-reverse | column | column-reverse;
```

- `row` (default): left to right in `ltr`; right to left in `rtl`
- `row-reverse`: right to left in `ltr`; left to right in `rtl`
- `column`: same as `row` but top to bottom
- `column-reverse`: same as `row-reverse` but bottom to top

flex-wrap



By default, flex items will all try to fit onto one line. You can change that and allow the items to wrap as needed with this property. Direction also plays a role here, determining the direction new lines are stacked in.

CSS

```
.container{  
  flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

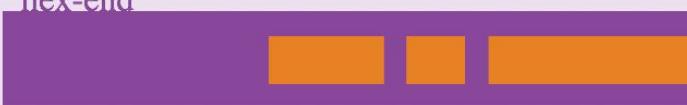
- `nowrap` (default): single-line / left to right in `ltr`; right to left in `rtl`
- `wrap`: multi-line / left to right in `ltr`; right to left in `rtl`
- `wrap-reverse`: multi-line / right to left in `ltr`; left to right in `rtl`

justify-content

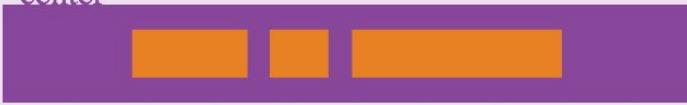
flex-start



flex-end



center



space-between

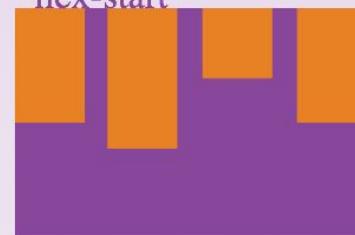


space-around



align-items

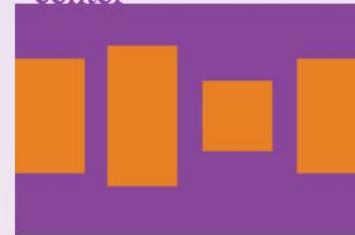
flex-start



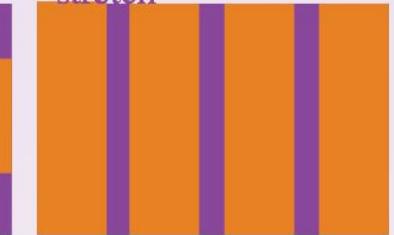
flex-end



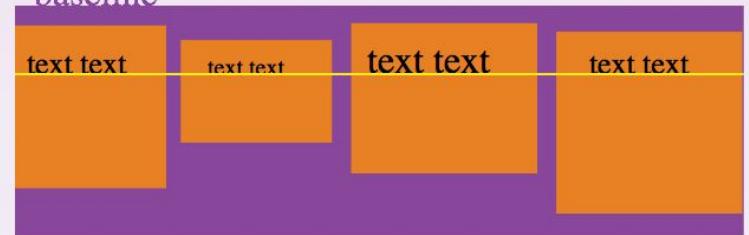
center



stretch



baseline



align-content

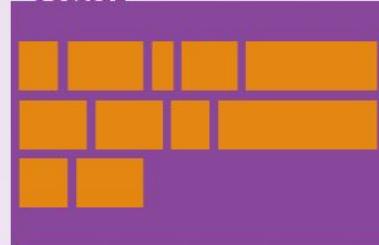
flex-start



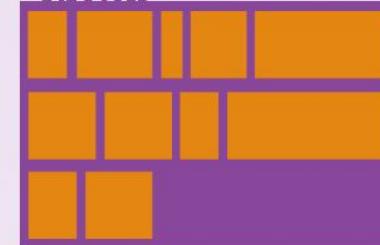
flex-end



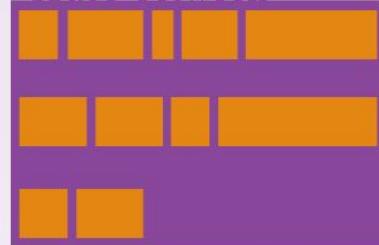
center



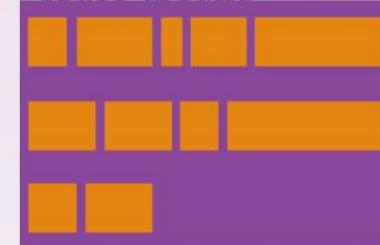
stretch



space-between



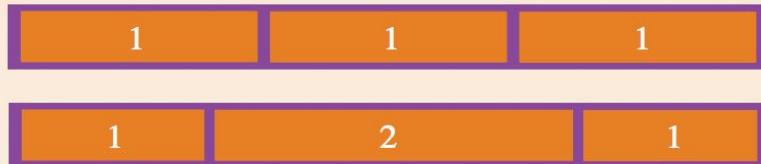
space-around



Item Attribute



flex-grow



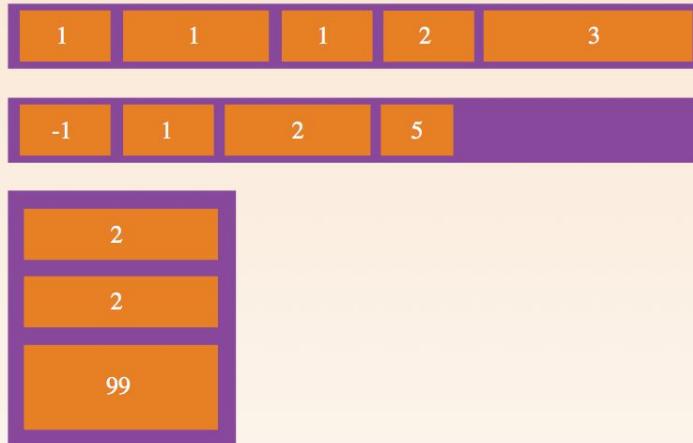
flex-shrink

This defines the ability for a flex item to shrink if necessary.

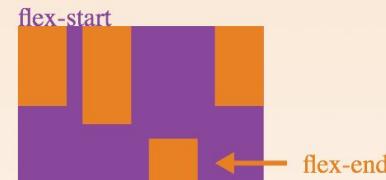
CSS

```
.item {  
  flex-shrink: <number>; /* default 1 */  
}
```

order



align-self



This allows the default alignment (or the one specified by `align-items`) to be overridden for individual flex items.

Please see the `align-items` explanation to understand the available values.

CSS

```
.item {  
  align-self: auto | flex-start | flex-end  
}
```

Note that `float`, `clear` and `vertical-align` have no effect on a flex item.

2.2.5 CSS - Layout

Attribute für das Layout

■ 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` Blockgröße passt sich an
`hidden` Inhalt beschneiden
`scroll` Inhalt verschiebbar mit Scroll-Balken (immer sichtbar)
`auto` Scroll-Balken nur bei Bedarf



```
<span style="display:block;  
float:right; width:40%;">
```

Breiten- und Höhenangaben

mit **overflow** festlegen,
was mit überstehenden
Elementen passieren soll

Breitenangaben

- ⇒ `width: 15%; /* relativ zur Umgebungsgröße */`
- ⇒ `min-width: 30em;`
z.B. falls das <body>-Tag eine bestimmte Mindestgröße haben soll;
unterhalb `min-width` wird die gesamte Seite gescrollt
- ⇒ `max-width: 50em;`
z.B. falls eine Randspalte eine bestimmte Maximalgröße haben soll

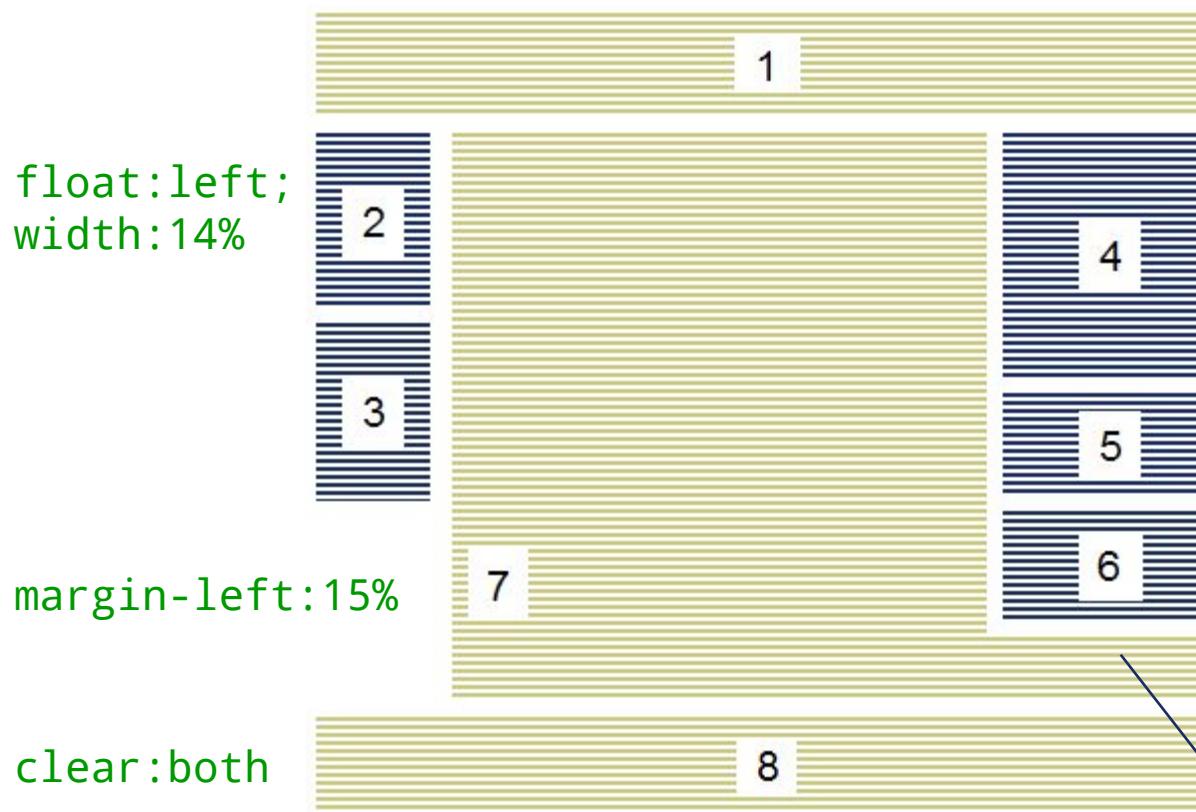
Höhenangaben

- ⇒ `height: 40em;`
- ⇒ `min-height: 30em;`
- ⇒ `max-height: 50em;`

min-... und **max-...**
in % wäre sinnlos

2.2.5 CSS - Layout

Layout-Prinzip: 3-spaltig mit Kopf- und Fußzeile



Klammer um Randblöcke:

```
<div style="float:right">
  <p id="Block4"></p>
  <p id="Block5"></p>
  <p id="Block6"></p>
</div>
```

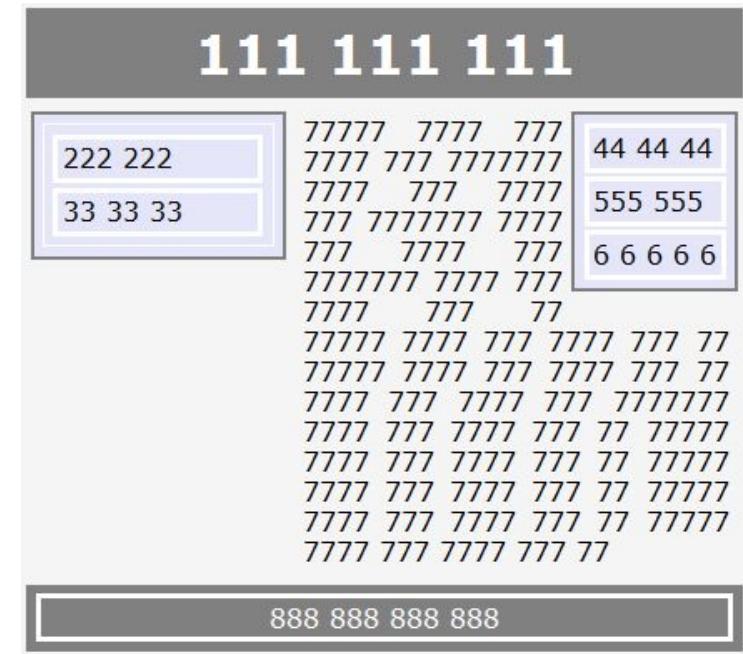
Das "Umfließen" klappt nur, wenn der Abschnitt zeilenweise fließt

2.2.5 CSS - Layout

Beispiel: HTML 3-"spaltig" mit Kopf-, Menü und Fußzeile

```
...
<body>
  <header><h1>111 111 111</h1></header>
  <nav><ul>
    <li>222 222</li>
    <li>33 33 33</li>
  </ul></nav>

  <div class="Rechts">
    <p>44 44 44</p>
    <p>555 555</p>
    <p>6 6 6 6 6</p>
  </div>
  <section>
    <article>77777 7777 777 ... 777 7777777 77</article >
  </section>
  <footer ><p>888 888 888 888</p></footer>
</body>
```



2.2.5 CSS - Layout

Beispiel: CSS 3-spaltig mit Kopf-, Menü und Fußzeile

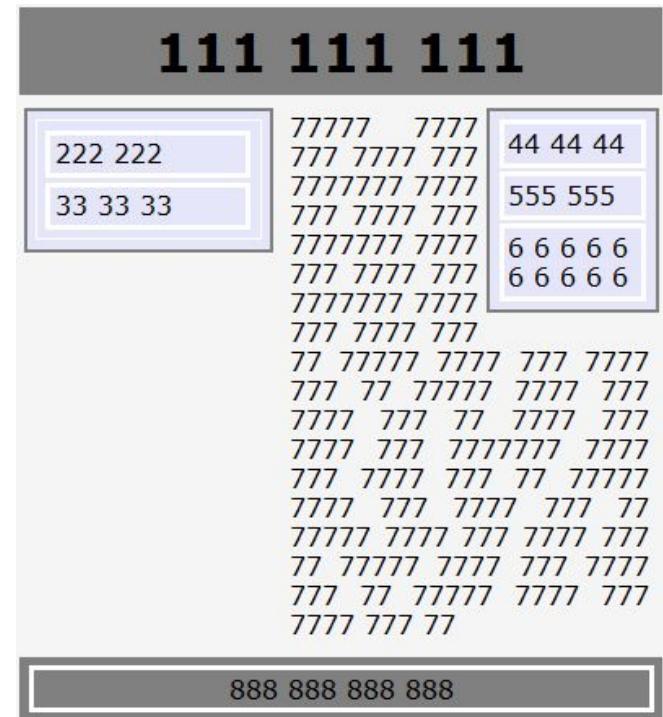
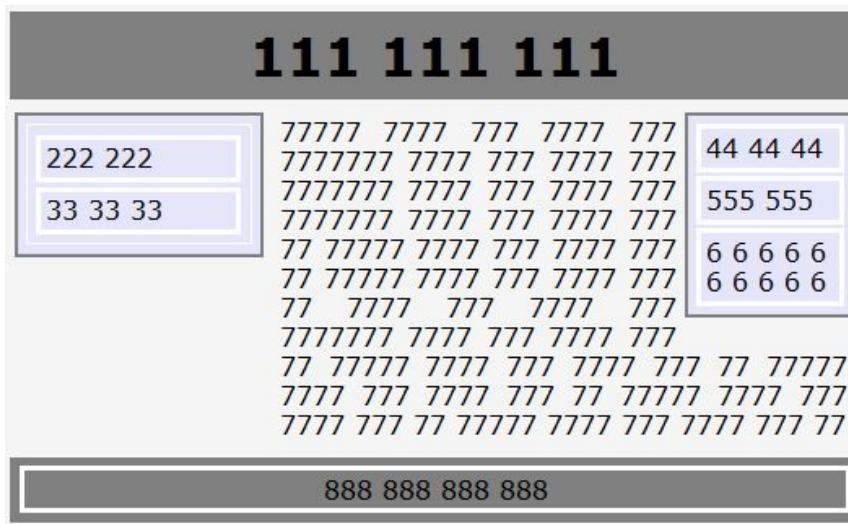
```
/* Design*/
* {margin: 2pt; padding: 2pt; color:black;}
body { color:black; background-color: WhiteSmoke; font:1em
      Verdana;}
footer, header, {color: white; background-color: grey;
                 text-align:center;}
nav, .Rechts {margin: 5px; border: 2px solid grey;
              background-color: Lavender;}
li, p {border: 3px solid white;}
ul {list-style:none; border: 1pt solid white; text-align: left; }

/* Layout */
body {min-width:25em;}
footer, header {clear: both;}
nav {display: block; float:left; width: 9em; min-width:5em;}
article {text-align: justify; margin-left:10em; }
.Rechts {float: right; display: block; max-width: 10em;}
```



Beispiel: Das Ergebnis in verschiedenen Fenstergrößen

- Kopf- und Fußzeile
- Menüliste als "Buttons" in einer Zeile
- Der mittlere Inhaltsbereich (7er) umfließt den rechten Block



Flexbox für Layout etc. verwenden.
Float nur wenn Text wirklich etwas
umfließen soll



2.2.5 CSS - Layout

Platzierung und Tiefenstaffelung

■ beliebige Platzierung mit Verdeckung

⇒ **position:**

static (normaler Elementfluss; Normaleinstellung)

relative (zu ursprünglicher Position im Elementfluss)

absolute (bezogen auf Elternelement)

fixed (bezogen auf Browserfenster; scrollt nicht mit)

⇒ **top**, (bzw. **bottom**), **left**, (bzw. **right**), **width**, **height**

- mit JavaScript dynamisch änderbar ⇒ "Animation"

■ Tiefenstaffelung explizit mit **z-index**

⇒ z.B. **z-index:3**

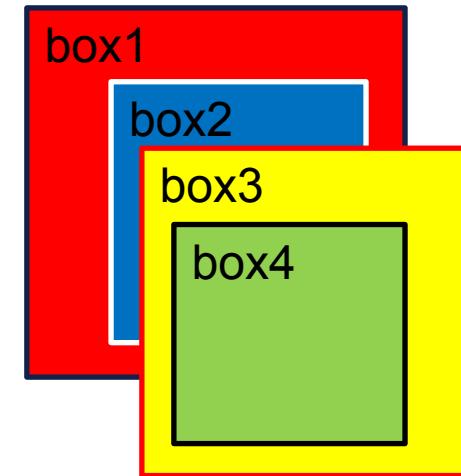
⇒ je größer die Zahl, desto weiter vorne

⇒ positionierte Elemente ohne z-index (d.h. z-index=0) entsprechend der Reihenfolge in der HTML-Datei

- vor allen Elementen, die nicht positioniert sind

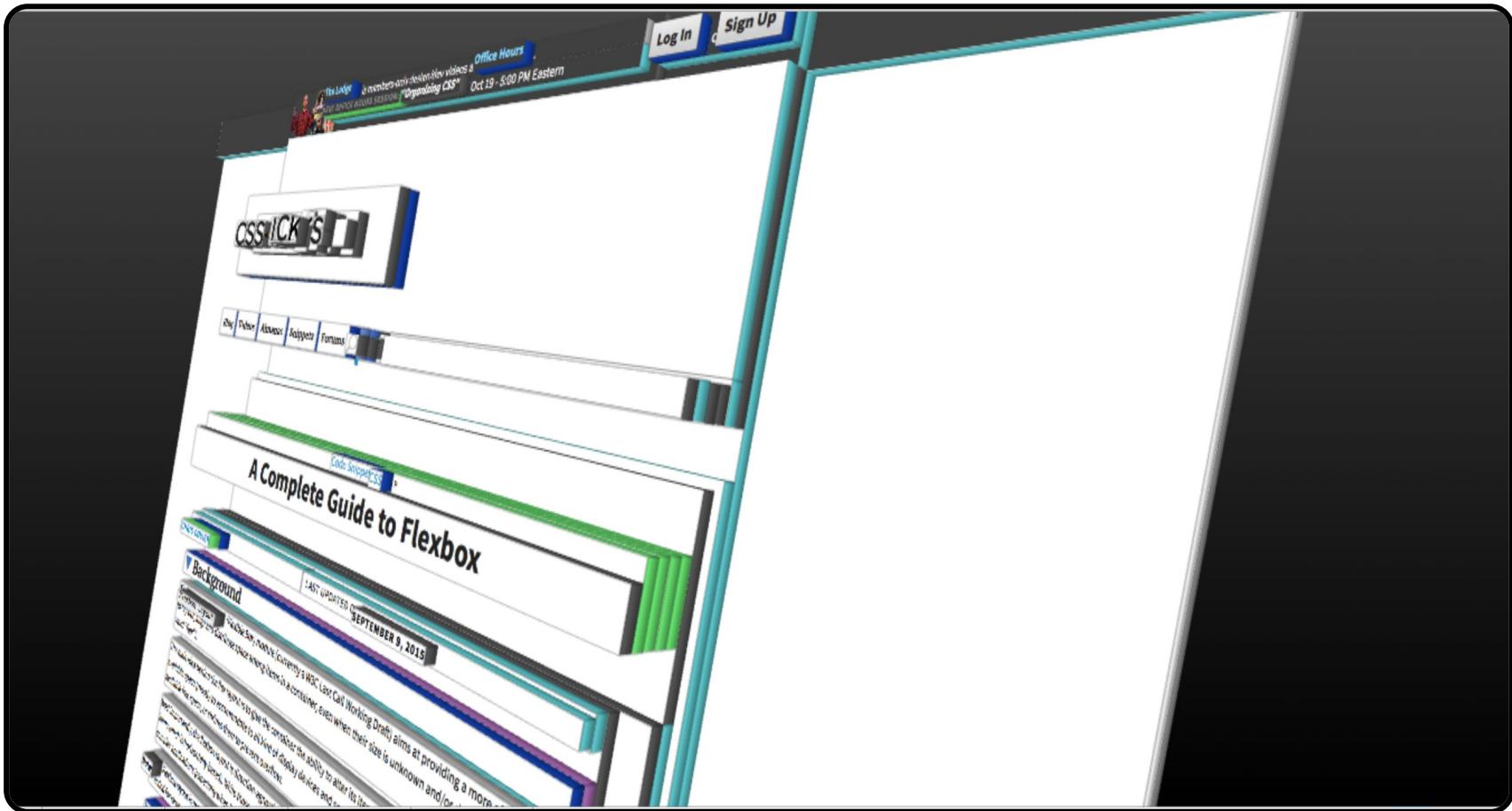
- oben in der Datei ⇒ im Hintergrund

- unten in der Datei ⇒ im Vordergrund



Beispiel: **position: absolute; top: 35px; left: 240px; width: 150px; height: 150px; z-index: 1**

Z-Index visualisiert



Hintergrundgrafik

- Auswahl des Bildes durch
 - ⇒ `background-image:url(mein_bild.jpg)`
- Wiederholung ("Kacheln der Grafik")
 - ⇒ `background-repeat:`
 - `repeat` (Wiederholung horizontal und vertikal),
 - `repeat-x` (Wiederholung nur horizontal),
 - `repeat-y` (Wiederholung nur vertikal),
 - `no-repeat`
- Positionierung (sofern nicht gekachelt)
 - ⇒ `background-position: x y;`
 - mit x aus `left`, `center` oder `right` und
 - mit y aus `top`, `center` oder `bottom`
- Beispiel:

```
.content2 {background-image:url(mein_bild.jpg);  
background-position: center center;}
```

Media Queries

- Mit Media Queries können Eigenschaften des Ausgabegerätes abgefragt werden
 - ⇒ Spezielle CSS-Abschnitte oder Dateien je nach Art des Ausgabegerätes, Fenstergröße, Displaygröße, Farbtiefe, Orientierung
- Beispiele
 - ⇒ Smartphones

```
<link rel="stylesheet" href="smartphone.css"  
      media="only screen and (min-device-width : 320px)  
            and (max-device-width : 480px)" />
```
 - ⇒ Layout für kleine Bildschirmgrößen linearisieren

```
@media screen and (max-width: 600px) {  
    header, footer, .main { float: none; width: auto; }  
}
```

<http://maddesigns.de/css3-responsive/template/#60>

Ergänzungen für die Barrierefreiheit

■ Der Aufbau eines barrierefreien Layouts

- ⇒ ist mit CSS möglich
- ⇒ wäre erheblicher Aufwand, wenn man es als getrenntes zweites Layout umsetzen würde – und würde sich finanziell nur rechnen, wenn die Website eine entsprechende Zielgruppe ansprechen soll
- ⇒ ist aber gar nicht so aufwändig, wenn man sich an die Standards und Empfehlungen hält
- ⇒ Zu HTML5 gehört die "WAI-ARIA-Spezifikation" (*Web Accessibility Initiative - Accessibility for Rich Intranet Applications*)
 - Definiert u.a. spezielle Attribute, welche unsichtbar die Funktion eines Tag beschreiben: z.B. `role="banner"` oder `role="search"`
- ⇒ kann auf diversen Seiten und mit Tools geprüft werden
 - z.B. <http://wave.webaim.org>, WebAccessibilityToolbar, LynxView, TAW3
- ⇒ gehört zum "guten Stil" eines professionellen Webauftritts



W3C Web Content Accessibility Guidelines 2.0

WCAG 2.0
Theme Song
bei YouTube



Wahrnehmbar

- Biete Alternativ-Texte für Inhalte, die nicht textuell sind
- Biete Untertitel und Alternativtexte für Audio-Inhalte und Videos
- Mache den Inhalt anpassbar; und verfügbar für Hilfsgeräte
- Verwende genügend Kontrast damit Dinge leicht zu sehen und zu hören sind

```

```

A screenshot of the official website of the German Bundestag. It features the German eagle logo, links for Gebärdensprache, Leichte Sprache, English, Français, and a search bar with a magnifying glass icon. The main title "Deutscher Bundestag" is displayed above a large image of the Reichstag dome's glass and steel structure.

A screenshot of the DB Bahn website. It includes the DB Bahn logo, a search bar with a magnifying glass icon, and links for Startseite, Kontakt, Häufige Fragen, Sitemap, and Deutschland. There are also icons for language selection and font size adjustment.

Quelle: <http://www.w3.org/TR/WCAG20>, Übersetzung: Ralf Hahn



2.2.5 CSS - Layout

W3C Web Content Accessibility Guidelines 2.0

WCAG 2.0
Theme Song
bei YouTube



Bedienbar

- Mache alle Funktionen über die Tastatur zugreifbar
- Lass Anwender genug Zeit den Inhalt zu lesen und zu benutzen
- Verwende keine Inhalte, die Krämpfe verursachen
- Hilf Anwendern bei der Navigation und beim Finden von Inhalten

A screenshot of the DB Bahn website. At the top, there is a navigation bar with links for "Startseite", "Kontakt", "Häufige Fragen", "Sitemap", "Deutschland" (with a dropdown arrow), and "A A A" (font size controls). There is also a search bar with a magnifying glass icon. Below this is a red header bar with links for "Angebotsberatung", "Fahrplan & Buchung", "Services", "BahnCard", "Geschäftsreisen", "Urlaub", "Meine Bahn", and "Login". A "Logout" button is visible on the right. At the bottom of the header, there is a "Auskunft & Tickets" link. The main content area shows a blurred image of a person's face.

Zu schnelle Bildfolgen

Von Peter Riesbeck



Es war ein verwirrendes Blitzlicht-Gewitter: 54-mal in einer Sekunde wechselten die Farben zwischen blau, rot und grün. Zu viel für einige Kinder, die vor dem Fernseher saßen. Mit Krämpfen, Sehstörungen und epileptischen Anfällen wurden vor drei Jahren in Japan mehrere Hundert Kinder in die Klinik eingeliefert. Sie hatten nichts anderes getan, als vorm Fernseher zu sitzen und sich die Zeichentrickserie "Pokemon" anzuschauen. Konkret: eine flackernde Kampfszene, in der ein Computer-Virus vernichtet wurde. Wissenschaftler von der Universität Pisa glauben nun entschlüsselt zu haben, warum die Kinder einen TV-Flush erlitten. Die Kleinen leiden an fotosensitiver Epilepsie, berichten die Forscher in der März-Ausgabe des Wissenschaftsmagazins "Neuroscience". Unter 1.000 Kindern leiden etwa fünf an dieser

Quelle: <http://www.w3.org/TR/WCAG20>, Übersetzung: Ralf Hahn

Informatik, Entwicklung webbasierter Anwendungen, SS2015
WS2015/2016

W3C Web Content Accessibility Guidelines 2.0

WCAG 2.0
Theme Song
bei YouTube



Verständlich

- Mache Text lesbar und verständlich
- Lasse Inhalte so erscheinen und sich so verhalten wie man es erwartet
- Hilf Anwendern Fehler zu vermeiden und zu korrigieren

Quelle: <http://www.w3.org/TR/WCAG20>, Übersetzung: Ralf Hahn



W3C Web Content Accessibility Guidelines 2.0

WCAG 2.0
Theme Song
bei YouTube



Robust

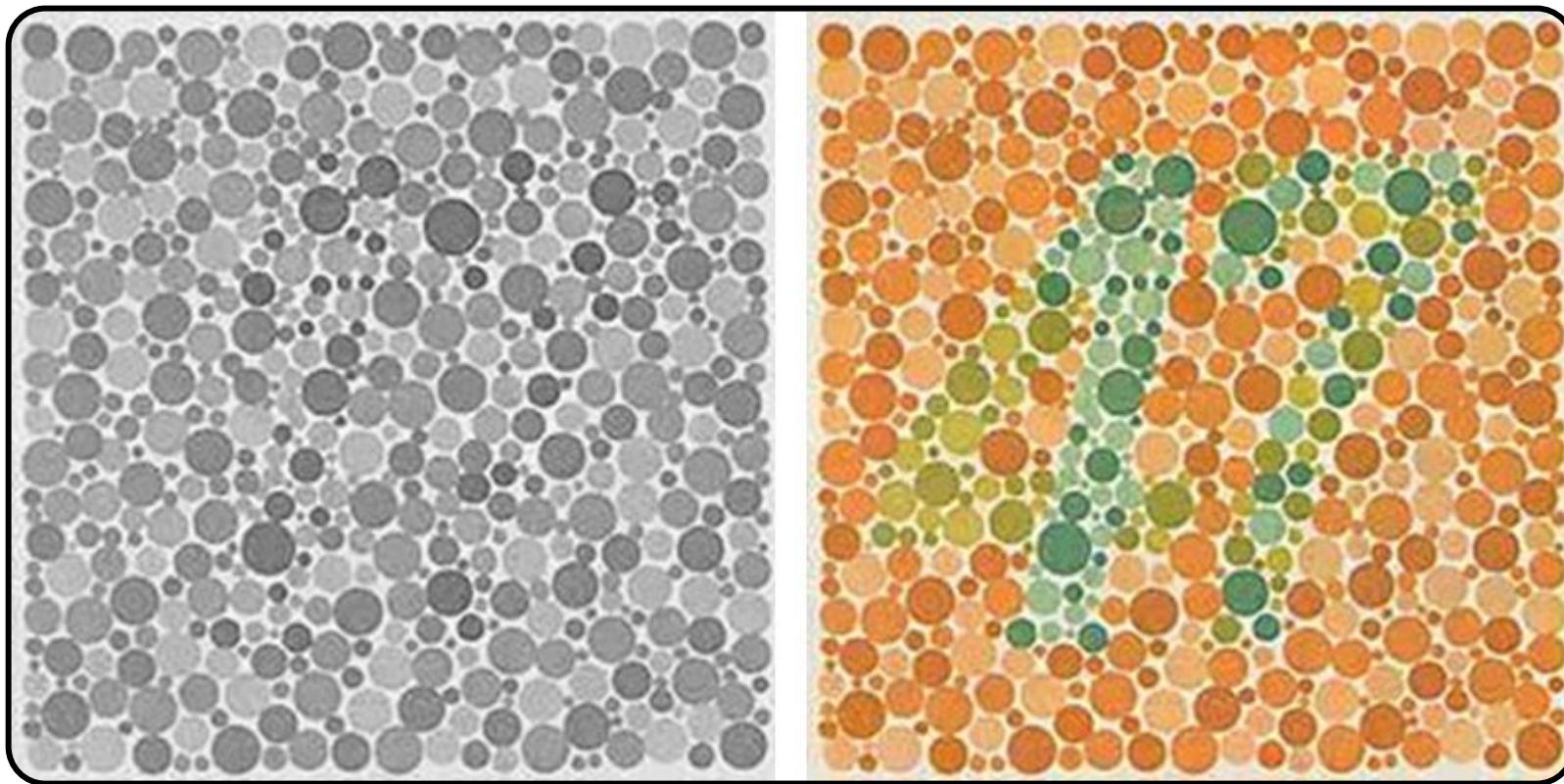
Maximiere die Kompatibilität mit aktuellen und zukünftigen Technologien

Quelle: <http://www.w3.org/TR/WCAG20>, Übersetzung: Ralf Hahn



Farbensehen

www.ichbinfarbenblind.de



- Sehen Sie ein 17? Eine 47? Gar keine Zahl?
 - ⇒ Ca. 10% der Menschen haben eine Farbenfehlsichtigkeit

Farbensehen



Quelle: Assessment of inherited colour vision defects in clinical practice, *Clin Exp Optom* 2007; 90: 3: 157–175

- Können Sie erkennen, welches Fleisch roh ist?
 - ⇒ 22% der Menschen mit Farbfehlsehigkeit können keinen Unterschied sehen

2.2.5 CSS - Layout

Konkrete Tipps zur Umsetzung der Barrierefreiheit

- ⇒ Layout ausschließlich über CSS definieren
- ⇒ Verwendung relativer Maßeinheiten damit die Browsereinstellungen berücksichtigt werden
- ⇒ keine zappelnden Animationen oder Popups mit schwer treffbaren Elementen (z.B. "Schließen-Kreuz" in vielen Foren)
- ⇒ Zusätzliche Navigierbarkeit über "versteckte" Menüs
 - erlaubt direktes Anspringen und Selektieren von Hauptinhalt, Navigationsleiste, Header etc.
 - für den "normalen" Leser nicht sichtbar, aber vom Screenreader vorlesbar
 - Über separate CSS Abschnitte für den Screenreader (media "speech")

2.2.5 CSS - Layout

Technische Umsetzung für ein zusätzliches Navigationsmenu

```
<nav class="barrierefrei"><ul>
  <li><a href="#nav" title="Link zur Navigationsleiste">
    Zur Navigationsleiste</a></li> ...
</ul></nav>
```

```
.barrierefrei {display: none;} /*normalerweise versteckt */
@media braille, speech { /* für speech, braille */
  .barrierefrei { display: block; } /* sichtbar */
}
```

@media erst ab IE9!
Alte verbreitete Screenreader können das nicht!

Hochschule Darmstadt

Fachbereich Informatik

2.2.6 CSS - Kaskadierung



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Sinn einer Format-Hierarchie

- Corporate Identity ⇒ Corporate Design

- ⇒ Firmenlogo, Designrahmen

- 1. einheitl. Erscheinungsbild des Dokuments

- ⇒ vgl. PowerPoint: Design übernehmen

- ⇒ Farben, Schriften, Hintergrund, Ausrichtung

- 2. eine Auswahl von Layout-Typen für Seiten

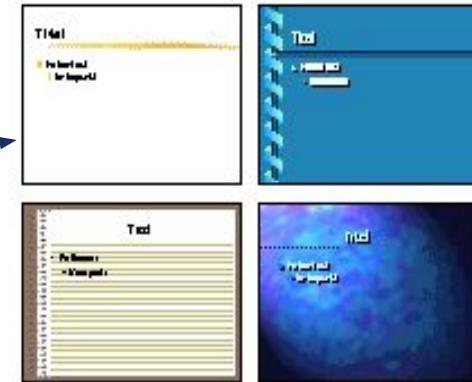
- ⇒ vgl. PowerPoint: Folienlayout

- ⇒ 0/1/2 Textblöcke, mit/ohne Bild, hor./vert. geteilt

- 3. Besonderheiten der einzelnen Seite

- ⇒ Abweichung vom Layout-Typ

- 4. 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"> <!-- embedded für  
--> </style> 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 wider- sprüchlicher 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

*!important > inline > id > class > elementname > **

Zusammenfassung

- CSS-Grundlagen
 - ⇒ Grundidee, Grundgerüst, HTML-Einbindung
 - ⇒ Schreibregeln und Syntax
 - ⇒ Formate modifizieren und definieren
- CSS-Attribute
 - ⇒ Farben, Hintergrund, Schriften, Ausrichtung, Ränder, Platzierung,...
- CSS-Maßeinheiten
 - ⇒ relative, absolute Maße, Zoom vs. Schriftgröße
- CSS-Layout
 - ⇒ Anordnung von Blöcken mit float, Überdeckung, Hintergrundbilder
- CSS-Kaskadierung

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

Hochschule Darmstadt

Fachbereich Informatik

2.2.7 CSS - Animationen



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Nicht Klausur relevant



The screenshot shows a developer tools interface with two panes: HTML and CSS.

HTML pane:

```
1<div>
2</div>
```

CSS pane:

```
1/* The animation code */
2@keyframes example {
3    from {background-color: red;}
4    to {background-color: yellow;}
5}
6
7/* The element to apply the animation
8to */
9div {
10    width: 400px;
11    height: 400px;
12    background-color: red;
13    animation-name: example;
14    animation-duration: 10s;
```

The right side of the interface features a large yellow rectangular area, which is the visual representation of the CSS styles applied to the element in the HTML pane.

HTML

```
1 <div>
2 </div>
```

CSS

```
1 /* The animation code */
2 @keyframes example {
3     0% {background-color: red;}
4     25% {background-color: yellow;}
5     50% {background-color: blue;}
6     100% {background-color: green;}
7 }
8
9 /* The element to apply the animation
to */
10 div {
11     width: 400px;
12     height: 400px;
13     background-color: red;
14     animation-name: example;
15     animation-duration: 40s;
16 }
```

Tidy

X

Tidy

X

The screenshot shows a code editor interface with two tabs: 'HTML' and 'CSS'. The 'HTML' tab contains the following code:`1 <div>
2 </div>`

The 'CSS' tab contains the following code:`1 /* The animation code */
2 @keyframes example {
3 0% {background-color: red;}
4 25% {background-color: yellow;}
5 50% {background-color: blue;}
6 100% {background-color: green;}
7 }
8
9 /* The element to apply the animation
to */
10 div {
11 width: 400px;
12 height: 400px;
13 background-color: red;
14 animation-name: example;
15 animation-duration: 40s;
16 }`

On the right side of the interface, there is a preview window showing a solid blue rectangle, which is the result of applying the CSS animation to the 'div' element.

204

Verändert von Thomas Sauer, Patrick Rath und Jakob Oesterling, WS2015/2016



```
HTML
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<div></div>
</body>
</html>

CSS
/* The animation code */
@keyframes example {
  0% {background-color: red; left:0px; top:0px;}
  25% {background-color: yellow; left:200px; top:0px;}
  50% {background-color: blue; left:200px; top:200px;}
  75% {background-color: green; left:0px; top:200px;}
  100% {background-color: red; left:0px; top:0px;}
}

/* The element to apply the animation to */
div {
  width: 400px;
  height: 400px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```



The **animation-delay** property specifies a delay for the start of an animation.

The **animation-iteration-count** property specifies the number of times an animation should run.

```
animation-iteration-count: infinite;
```

The **animation-direction** property is used to let an animation run in reverse direction or alternate cycles.

```
animation-direction: reverse;
```

The **animation-timing-function** property specifies the speed curve of the animation.

The **animation-timing-function** property can have the following values:

ease – specifies an animation with a slow start, then fast, then end slowly (this is default)

linear – specifies an animation with the same speed from start to end

ease-in – specifies an animation with a slow start

ease-out – specifies an animation with a slow end

ease-in-out – specifies an animation with a slow start and end

cubic-bezier(n,n,n,n) – lets you define your own values in a **cubic-bezier** function

http://www.w3schools.com/css/css3_animations.asp

für eine Liste fertiger Animationen die sinnvoll sind siehe <https://daneden.github.io/animate.css/>



Hochschule Darmstadt

Fachbereich Informatik

2.2.8



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK



Nicht Klausur relevant



Variablen

```
$font-stack: Helvetica, sans-serif  
$primary-color: #333  
  
body  
  font: 100% $font-stack  
  color: $primary-color
```

wird zu

```
body {  
  font: 100% Helvetica, sans-serif;  
  color: #333;  
}
```

Nesting

```
nav
  ul
    margin: 0
    padding: 0
    list-style: none

  li
    display: inline-block

    a
      display: block
      padding: 6px 12px
      text-decoration: none
```

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}

nav li {
  display: inline-block;
}

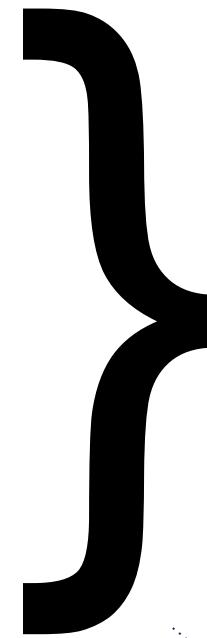
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}

body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}
```

Imports

```
// _reset.sass  
  
html,  
body,  
ul,  
ol  
  margin: 0  
  padding: 0
```

```
//base.sass  
  
@import reset  
  
body  
  font: 100% Helvetica, sans-serif  
  background-color: #efefef
```



```
html, body, ul, ol {  
  margin: 0;  
  padding: 0;  
}  
  
body {  
  font: 100% Helvetica, sans-serif;  
  background-color: #efefef;
```

Spart HTTP-Requests
im vergleich zum
normalen CSS import

Mixins

```
=border-radius($radius)
  -webkit-border-radius: $radius
  -moz-border-radius:     $radius
  -ms-border-radius:    $radius
  border-radius:        $radius

.box
  +border-radius(10px)
```

```
.box {
  -webkit-border-radius: 10px;
  -moz-border-radius:   10px;
  -ms-border-radius:   10px;
  border-radius:        10px;
```

Vererbung

```
.message
  border: 1px solid #ccc
  padding: 10px
  color: #333

.success
  @extend .message
  border-color: green

.error
  @extend .message
  border-color: red

.warning
  @extend .message
  border-color: yellow
```

```
.message, .success, .error, .warning {
  border: 1px solid #cccccc;
  padding: 10px;
  color: #333;
}

.success {
  border-color: green;
}

.error {
  border-color: red;
}

.warning {
  border-color: yellow;
}
```

Berechnungen

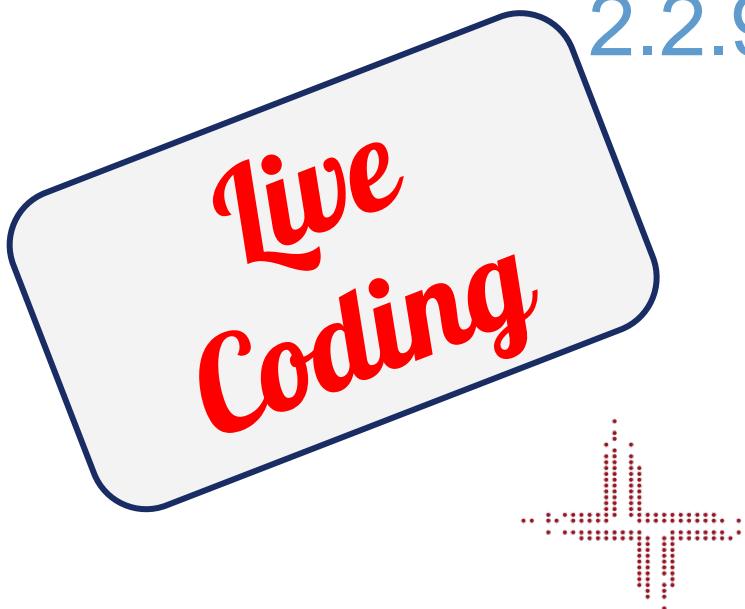
```
.container {  
    width: 100%;  
  
    article[role="main"] {  
        float: left;  
        width: 600px / 960px * 100%;  
  
        aside[role="complimentary"] {  
            float: right;  
            width: 300px / 960px * 100%;  
        }  
    }  
}
```

```
.container {  
    width: 100%;  
}  
  
article[role="main"] {  
    float: left;  
    width: 62.5%;  
}  
  
aside[role="complimentary"] {  
    float: right;  
    width: 31.25%;  
}
```

Hochschule Darmstadt

Fachbereich Informatik

2.2.9 Bootstrap



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Nicht Klausur relevant



Hochschule Darmstadt

Fachbereich Informatik

2.3 Clientseitige Programmierung



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Interaktion mit Webseiten auf dem Client

- Seitenumschaltung
 - ⇒ Linking, (transparente) Schaltflächen, sensitive Grafik HTML
- Eingabeformulare
 - ⇒ Listbox, Checkbox, Radiobutton HTML
 - ⇒ Konsistenzprüfung der Eingaben
(auf dem Client !) ECMAScript
- objektbezogene Ereignisbehandlung
 - ⇒ Veränderung der HTML-Seite
(auf dem Client !) ECMAScript,
DOM, Ajax
- allgemeine Programmierung
 - ⇒ aufwändige Visualisierung (Konfigurator beim Autokauf),
spezielle Widgets (TreeControl)

Java Applet
Flash
ECMAScript!



2.3 Clientseitige Programmierung

Zur Abgrenzung: Server-seitig

- Content Management System
- Durchsuchen großer Datenmengen
 - ⇒ Datenbankabfrage (z.B. Fahrplanauskunft)
 - ⇒ Volltextsuchmaschine
- Speicherung von Daten
 - ⇒ Gästebuch, Schwarzes Brett, Bestellungen
- Realisierungsmöglichkeiten
 - ⇒ CGI, proprietäre Server-APIs (NSAPI, ISAPI)
 - ⇒ PHP, ASP, JSP
 - ⇒ Java Servlet
 - ⇒ Java Applet mit RMI (remote method invocation)
 - ⇒ NodeJS
 - ⇒ ...



JavaScript

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



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
 - ⇒ alle Eigenschaften per Skript änderbar
 - ⇒ Ereignisbehandlung mit zugeordneten Skripten
- Seiten können vom Surfer modifiziert werden (z.B. Warenkorb)
 - ⇒ ermöglicht Anzeige von Berechnungsergebnissen (z.B. Gesamtkosten)
 - ⇒ ermöglicht Auf- und Zuklapp-Effekte
 - ⇒ ermöglicht lokale Animationen
 - ⇒ ermöglicht 3D-Rendering, komplette Applikationen und andere aufwändige Berechnungen direkt im Browser (Clientseitig)





Beispiel (ECMAScript eingebettet in HTML)

```

1  <!DOCTYPE html>
2  <html lang="de">
3  <head> <meta charset="UTF-8"/>
4      <title>Sinn des Lebens</title>
5      <script>
6          var Hinweis = "Hurra! Quadratzahlen!";
7          alert(Hinweis);
8          function schreibeQuadrate() {
9              "use strict";
10             var i, x, SinnDesLebens, Satzteil;
11             SinnDesLebens = 42;
12             Satzteil = "Das Quadrat von ";
13             for (i = SinnDesLebens; i > 0; i = i - 1) {
14                 x = i * i;
15                 document.getElementById('body').innerHTML +=
16                     "<p>" + Satzteil + i + " ist " + x + "</p>";
17             }
18         }
19     </script>
20 </head>
21 <body id="body" onload="schreibeQuadrate();">
22 </body>
23 </html>

```

aktiviert "modernes
ECMAScript"

fügt HTML in
das aktuelle
Dokument ein

Ausführung sobald
die Seite geladen ist



Historie

"JavaScript is to Java as Ham is to Hamster"

- Ursprung: JavaScript (Netscape) in Navigator 2.0
 - ⇒ von Netscape an Microsoft lizenziert; 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: 5.1 Edition, Juni 2011
 - neu: ES2015 (ES6), Mitte 2015
 - ⇒ autorisiert von W3C, übernommen ~~an~~ optional, wenn Interesse besteht am Ende des Semesters
 - Aktuell von TC-39 (TC = Technical Committee)
 - ⇒ <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>

darauf konzentrieren wir uns

Hochschule Darmstadt

Fachbereich Informatik

2.3.1 ECMAScript: Definition



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Begriffsdefinition

- **ECMAScript != JavaScript**
 - ⇒ ECMAScript beschreibt den Standard
 - ⇒ JavaScript ist ein Dialekt dieses Standards mit zusätzlichen Features
 - Beispielsweise die DOM API (kommt später)
- **ES5**
 - ECMAScript 5 ist der Standard, der die letzten Jahre vorherrschend war
 - Diesen Standard verwenden wir in der Vorlesung!
- **ES6 (== ES2015)**
 - Neuer ECMAScript Standard, mit neuen Features
 - Klassen, immutable Variablen, destructuring etc.
- Ab **ES2015** (Mitte 2015) soll jedes Jahr ein aktualisierter ES-Standard erscheinen (**ES2016** existiert bereits als draft)



2.3.1 ECMAScript: Definition

Vergleich mit C

- Syntax sehr ähnlich
 - ⇒ Zuweisung, **if**, **switch**, **for**, **while**, **do**, **try catch**, **//**, **/* ... */**
 - ⇒ Konstanten, Operatoren (Stringverkettung mit **+**)
- Variablen sind dynamisch typisiert (keine Typdeklaration)
 - ⇒ Zahlen sind immer Gleitkommazahlen
 - ⇒ Schlüsselworte **var** bzw. **function** statt Typ in der Deklaration
- Objekterzeugung mit **new**
 - ⇒ wie in Java; kein **delete**
- nicht zeilenorientiert
 - ⇒ **;** wird am Zeilenende automatisch eingefügt, falls es fehlt
(kein Zeilenende hinter **return** oder vor **++** und **--** lassen !)

man kann einfach mal drauflos schreiben...

Einfache Literale (Notation für Werte)

- Notation wie in C
 - ⇒ Integer-Literal `var antwort = 42;`
 - ⇒ Floating-Point Literal `var pi = 3.14;`
 - ⇒ String-Literal `var Gruss = "Hello World";`
 - ⇒ Boolean-Literal `var isPrime = true;`
- Schlüsselworte `null, true, false`
- Besonderheit für Strings
 - ⇒ wahlweise mit `"..."` oder `'...'`
ermöglicht String im String (z.B. String mit HTML-Attributwert)

2.3.1 ECMAScript: Definition

Array

das kann C nicht...

- dynamisch erzeugtes und erweiterbares Objekt
 - ⇒ ganzzahliger Index im Bereich 0..length-1
 - ⇒ Elementtyp beliebig und nicht notwendigerweise einheitlich
- Erzeugung
 - ⇒ ohne Längenangabe für dynamische Erweiterung

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

```
var Vektor1 = [];
```
 - ⇒ 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"

das kann C nicht...

- dynamisch erzeugtes und erweiterbares Objekt
  - ⇒ String als Index
  - ⇒ Elementtyp beliebig und nicht notwendigerweise einheitlich
  - ⇒ vgl. Datenstruktur / struct / Hashtabelle / map / dictionary
- Erzeugung, Erweiterung und Zugriff
  - ⇒ `var Vektor = [];`
  - `Vektor["posLeft"] = 45;`
  - `var Element = Vektor["posLeft"];`
- Verarbeitung
  - ⇒ häufig mit Hilfe von `for (... in ...)`  
`for (var Element in Vektor) {...Vektor[Element]...}`
  - ⇒ Vorsicht! `for...in` zählt alle Attribute eines Objekts (hier: das "Array") auf
    - dazu gehören auch geerbte Methoden (ggf. ausfiltern mit `hasOwnProperty`)
  - ⇒ echte Array-Funktionen wie `length` gibt es hier nicht
  - ⇒ der Zugriff über den Index `Vektor[i]` ist für ein assoziatives Array undefiniert

### 2.3.1 ECMAScript: Definition

## Literale für Arrays

- konstruieren Objekte
- dürfen auch Ausdrücke als Elemente enthalten, d.h. die Elemente müssen nicht ihrerseits Literale sein
- werden im Rahmen der JSON genutzt (siehe Abschnitt "Ajax")
  - ⇒ Literal für numerisch indiziertes Array  
`var Tiere = ["Hund", "Hamster", "Affe"];`  
ist Abkürzung für  
`var Tiere = new Array ("Hund", "Hamster", "Affe");`
  - ⇒ Literal für assoziatives Array (erzeugt eigentlich ein Objekt)  
`var Preise =`  
`{ Margherita: 4.0, Salami: 4.5, Hawaii: 5.5 }`

## 2.3.1 ECMAScript: Definition

### Funktionen

- Deklaration mit Schlüsselwort **function**
- Rückgabe von Werten aus Funktionen durch **return**
  - ⇒ ein Rückgabeparameter wird nicht deklariert
  - ⇒ Klammern nicht vergessen !
- Parameterübergabe wie in Java
  - ⇒ formale Parameter sind lokale Variable
  - ⇒ aktuelle Parameter werden bei Aufruf in formale Parameter kopiert
  - ⇒ Objekte werden per Referenz übergeben, nicht geklont
  - ⇒ d.h. call by value für einfache Datentypen, call by reference für Objekte
- Beispiel

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

Oder

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

### 2.3.1 ECMAScript: Definition

## Vordefinierte Funktionen

- Vordefinierte Funktionen können einfach aufgerufen werden

⇒ `eval(Zkette)`

never ever use this!

interpretiert übergebenes Argument als Code; gibt Ergebnis zurück (z.B. Objekt)

⇒ `isFinite(Wert)`

auf numerischen Wertebereich prüfen

⇒ `isNaN(Wert)`

auf nicht-numerischen Wert prüfen

⇒ `parseFloat(Zkette)` in Kommazahl umwandeln

⇒ `parseInt(Zkette)` String(anfang) in Ganzzahl umwandeln

⇒ `Number(Wert)` Datum in Zahl umwandeln

⇒ `String(Wert)` In Zeichenkette umwandeln

⇒ `encodeURI(Zkette)` CGI-Parameter für URL (de)kodieren

⇒ `decodeURI(Zkette)` (vgl Abschnitt CGI)

⇒ `Zahl.toFixed(n)` Erzwingt n Nachkommastellen

⇒ ...

⇒ <http://www.ecma-international.org/ecma-262/5.1/#sec-15>

Übersicht und Details  
zu allen ECMA  
Functionen

## Ausnahmebehandlung

- wie in Java / C++, Ausnahmeobjekte jedoch dynamisch typisiert

```
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...

### 2.3.1 ECMAScript: Definition

## Ordentliches ECMAScript!? Use Strict

- ECMAScript erlaubt viele Konstrukte, die fehleranfällig sind und es fehlen bewährte Programmierkonstrukte
- Der Befehl "**use strict**"; aktiviert eine strengere Interpretation
  - ⇒ der Browser bricht dann bei einem Fehler die Ausführung ab (!!)
    - und wirft Exceptions die vorher nicht geworfen worden wären
  - ⇒ Variablen müssen deklariert werden
  - ⇒ die Verwendung des **with**-Befehls ist verboten
  - ⇒ diverse zweifelhafte Konstrukte sind verboten
  - ⇒ je nach Platzierung gültig für eine Funktion oder das ganze Skript
  - ⇒ alte Browser, die "**use strict**"; nicht kennen, ignorieren den Befehl
  - ⇒ Vorsicht bei Altlasten und fremden Bibliotheken! Das nachträgliche Aktivieren von "**use strict**"; kann zu unerwarteten Ergebnissen führen
    - Code bricht plötzlich ab
    - Variablen liefern unerwartete Werte



## Ordentliches ECMAScript (II)

- Empfohlener Umgang mit "use strict":
  - ⇒ bei Neuentwicklungen unbedingt einsetzen
  - ⇒ In Funktionen verwenden (und nicht für das ganze Skript)

So verwenden  
wir es in EWA !

```
function schreibeQuadrat() {
 "use strict";
 ...
}
```

- Zusätzlich kann auch ein Analysetool wie
    - ⇒ JSLint (<http://www.jslint.com>)
      - Linter von Douglas Crockford - etwas veraltet
    - ⇒ JSHint (<http://jshint.com/>) - modernerer Fork von JSLint
    - ⇒ ESLint (<http://eslint.org/>) - Unterstützung von neusten ES-Standards
- verwendet werden.

# Hochschule Darmstadt

## Fachbereich Informatik

### 2.3.2 ECMAScript: Objektbasiert



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

## 2.3.2 ECMAScript: Objektbasierend

# Objektbasierend statt objektorientiert

- "...Object-based, not object-oriented. Complex object features are left out..."
  - ⇒ ECMA Script kennt keine Klassen (vor ES2015), verwirrt den Begriff "Objekt"
    - soll einfacher sein für Novizen ???
    - für OO-Programmierer sehr gewöhnungsbedürftig
  - ⇒ (fast!) alles und jedes ist ein Objekt, selbst eine Funktion
    - Zitat aus einer früheren Version des Standards:  
"All 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, werden instanziert)
  - ⇒ **Array, Boolean, Date, Number, String**
- "Objekte" (eigentlich Funktionsbibliotheken, global verfügbares Objekt bzw. nur Konstanten und "statische" Methoden)
  - ⇒ **Math, RegExp** (reguläre Ausdrücke)
- (echte) Objekte der Laufzeitumgebung (global verfügbar)
  - ⇒ **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 einfacher Zugriffsschutz (private / protected / public)
- `Objekt = new KonstruktorFunktion (Parameter);`
  - ⇒ `new` deutet dynamische Allokation an
    - `this.Attributname = Wert` definiert ein Attribut
    - `this.Methodename = Funktion` definiert eine Methode
  - ⇒ eine solche Funktion darf wiederum intern `this` verwenden
- Oder auch als Objekt-Literal anlegen:
  - ⇒ `var adresse = { name: "Hahn", stadt: "Darmstadt" }`
  - ⇒ erzeugt ein Objekt `adresse` mit Attributen `name` und `stadt`

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

## 2.3.2 ECMAScript: Objektbasierend

### Beispiel: Klasse Bruch im "C++ Stil"

#### 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);
}
```

alles  
public!  
private  
gibt es  
nicht

#### 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

## Exkurs: Closures

- Eine Closure ist eine lokale Variable einer Funktion, die selbst dann noch besteht, wenn die Funktion bereits returned wurde.

```
function sayHello2(name) {
 var text = 'Hello ' + name; // Local variable
 var sayAlert = function() { alert(text); }
 return sayAlert;
}

var say2 = sayHello2('Bob');
say2(); // alerts "Hello Bob"
```

- Längere Erklärung: <http://stackoverflow.com/a/111111>
- Beispiele:
  - <https://jsfiddle.net/haaner87/3ppf9nvc/3/>
  - <https://jsfiddle.net/haaner87/phzzx357/1/>

### 2.3.2 ECMAScript: Objektbasiert

## Beispiel: Klasse Bruch mit Closure

### Anwendung der Klasse

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

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

Closure (Abschluss) kapselt Zugriff  
auf globale Variable im Kontext der  
Definition, nicht des Aufrufs

private  
public

### Klassendefinition

```
function CBruch (Zaehler, Nenner)
{
 var m_Zaehler = Zaehler;
 this.m_Nenner = Nenner;

 this.getZaehler = function () {
 return m_Zaehler;
 }

 this.Mal = function (b) {
 return new CBruch
 (m_Zaehler * b,
 this.m_Nenner * b);
 }
}
```

namenlose  
Funktionen  
werden für  
jedes Objekt  
angelegt

## 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
    - Deswegen bei JS wichtig: Testing!
- Nachteil bei Verzicht: geringere Geschwindigkeit
  - ⇒ Laufzeittypprüfung kostet Rechenzeit
  - ⇒ es kann kein typspezifischer Code generiert werden
    - Leistungsstarke Clients erlauben heutzutage dennoch rechenintensive Operationen im Browser

## 2.3.2 ECMAScript: Objektbasierend

# Vererbung – es geht doch!

## ■ spezielle Eigenschaft jedes Objekts: **prototype**

- ⇒ enthält Zeiger auf Objekt der Basisklasse; kann **null** sein
- ⇒ keine Mehrfachvererbung
- ⇒ nicht für Objekte des DOM (für die ist prototype eine Eigenschaft wie jede andere)

## ■ impliziter Zugriff auf Attribute und Methoden des prototype

- ⇒ werden durch gleichnamige Attribute und Methoden des neuen Objekts verdeckt

```
1 function Basis(arg) { // Konstruktor = Klassendefinition
2 this.att = arg; // public Attribut
3 this.meth1 = function(arg) { // public final
4 return "Basis.meth1(" + this.att + arg + ")";
5 }
6 }
// public overridable:
7 Basis.prototype.meth2 = function(arg) {
8 return "Basis.meth2(" + this.att + arg + ")";
9 }
10
11 Subklasse.prototype = new Basis(); // Ableitung
12 Subklasse.prototype.constructor = Subklasse;
13
14 function Subklasse(arg2) { // Konstruktor
15 Basis.call(this, arg2); // Konstruktoraufruf
16 this.att2 = arg2; // public Attribut
17 this.meth1 = function(arg) { // public final
18 return "Subklasse.meth1(" + arg + ")";
19 }
20 }
21
22 // public overridable:
23 Subklasse.prototype.meth2 = function(arg)
24 var t = Basis.prototype.meth2.call(this, arg);
25 return "Subklasse=>" + t;
26
27
28
29 var b = new Basis("b");
30 var s = new Subklasse("s");
```

zur Erklärung siehe

<http://www.coolpage.com/developer/javascript/Correct OOP for Javascript.html>

# Hochschule Darmstadt

## Fachbereich Informatik

### 2.3.3 ECMAScript: Skript und HTML



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

### 2.3.3 ECMAScript: Skript und HTML

## Platzierung von Skripten

### ■ "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

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

### ■ seit HTML5 ist `type="text/javascript"` optional

### ■ falls Skriptausführung im Browser abgeschaltet ist

```
<noscript>
 <p>Bitte aktivieren Sie JavaScript !</p>
</noscript>
```

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

in `<head>`  
oder `<body>`

DOM ist aber erst  
komplett für  
Ereignis-Handler !

### Ereignisse und Handler

#### ■ vordefinierte Ereignisse

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

nicht ECMA Script,  
sondern HTML/JS

#### ■ Zuordnung "inline" im HTML-Tag

- ⇒ normalerweise: Funktionsaufruf als Event-Handler  
(beliebige ECMAScript-Anweisungen sind aber möglich)

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

oder

```
<input type="radio" onclick="document.forms['f1'].submit();"
... />
```

optimal für Initialisierung

## Reihenfolge von Maus-Ereignissen

Achtung !  
onmouse... funktioniert nicht (richtig)  
auf Touchscreens

### ■ 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
- ⇒ `onclick` down und up innerhalb des selben Elements
- ⇒ `ondblclick`

### ■ beim Verlassen

- ⇒ `onmouseout` Cursor verlässt den Objektbereich



## 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>
```

Achtung!: Nur clientseitige Formularvalidierung, die  
umgangen werden kann!

## 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 in `<body onload="Init()">`
- 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>

- ⇒ <body onload="Initialisieren();"  
onunload="Aufraeumen();">
- ⇒ oder alternativ über DOM zuweisen
- ⇒ document.getElementsByTagName  
("body")[0].onload = Initialisieren
- ⇒ document.getElementsByTagName  
("body")[0].onunload = Aufraeumen

in HTML

im Skript

### ■ body.onload ruft "Konstruktor" der Seite

- ⇒ tritt ein, nachdem die HTML-Datei komplett geladen wurde
- ⇒ Zugriff auf DOM jetzt möglich (ist nun komplett aufgebaut)
- ⇒ window.onload nach Laden von Bildern, Skripten, CSS-Dateien

Funktionszeiger  
(ohne Klammern!)

### ■ 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 Funktionsparameter (ohne Klammern hinterm Namen )
- ⇒ Alternativ: als anonyme Funktion:

```
1 window.setTimeout(function(){
2 alert('verzögertes hallo!')
3 }, 2000);
```
- ⇒ 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)



# Hochschule Darmstadt

## Fachbereich Informatik

### 2.3.4 ECMAScript: Dokument Objekt Modell – DOM



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

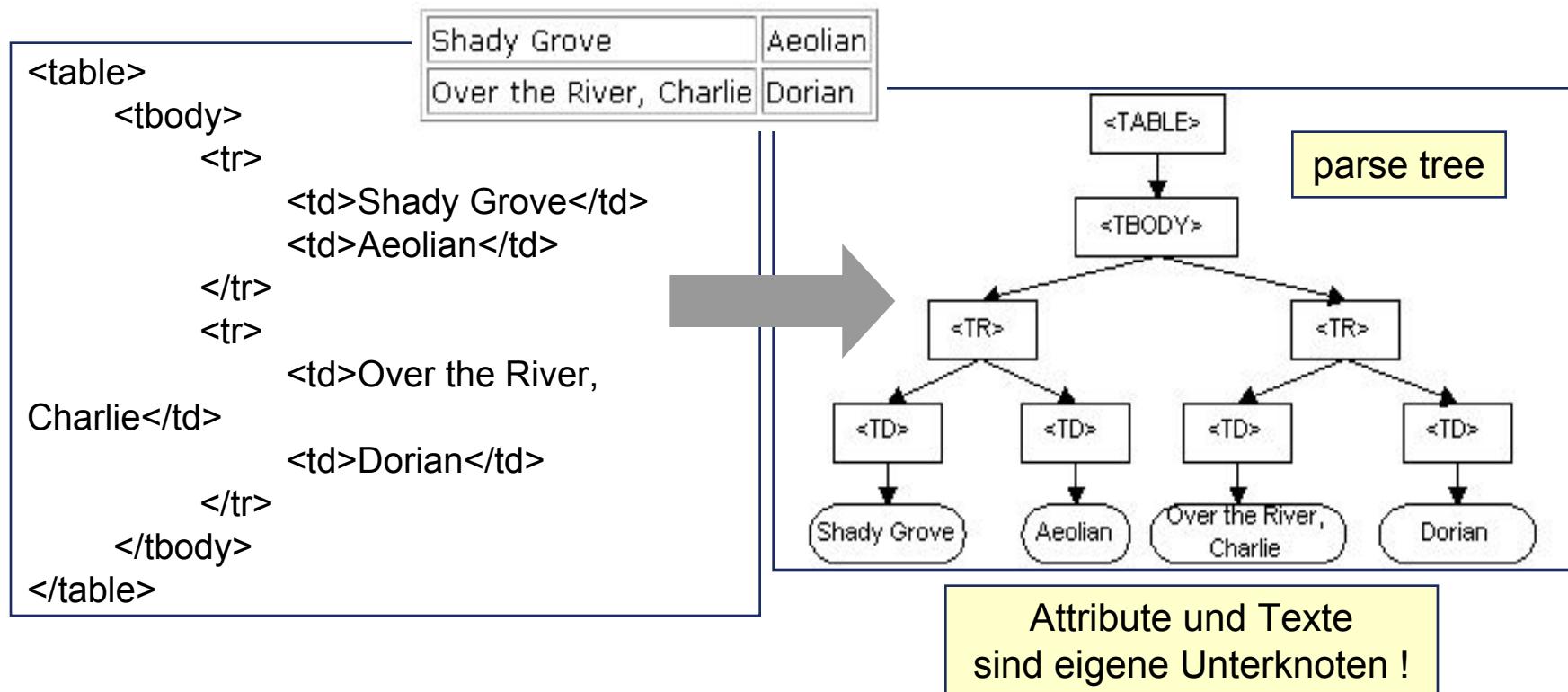
**fbi**

FACHBEREICH INFORMATIK

## 2.3.4 ECMAScript: Dokument Objekt Modell – DOM

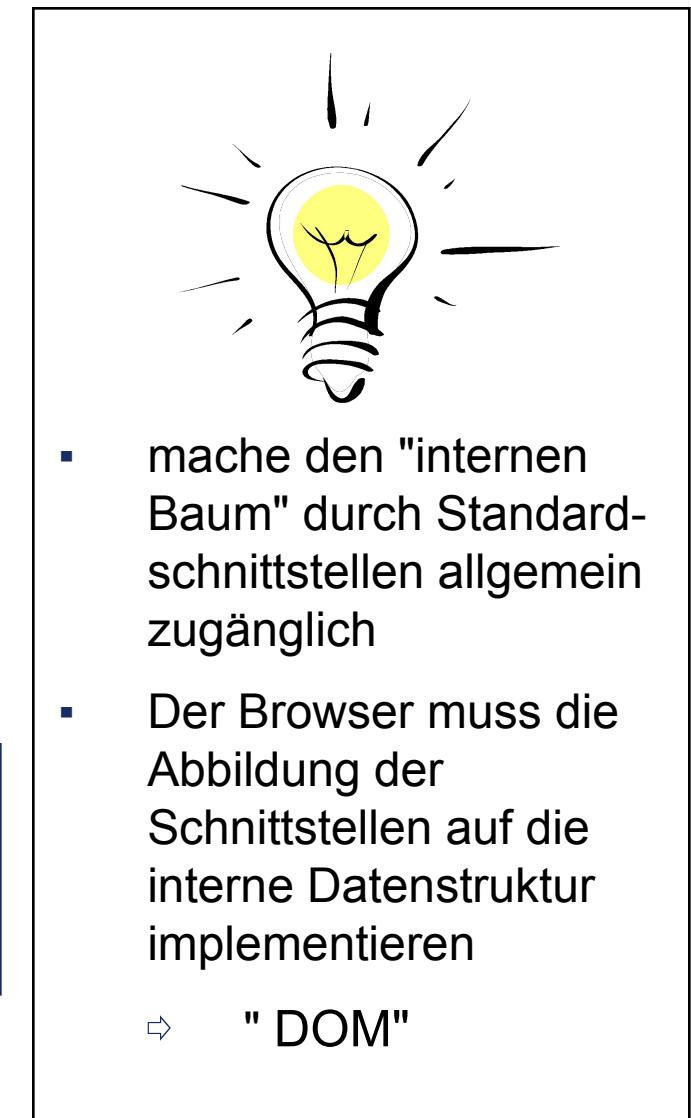
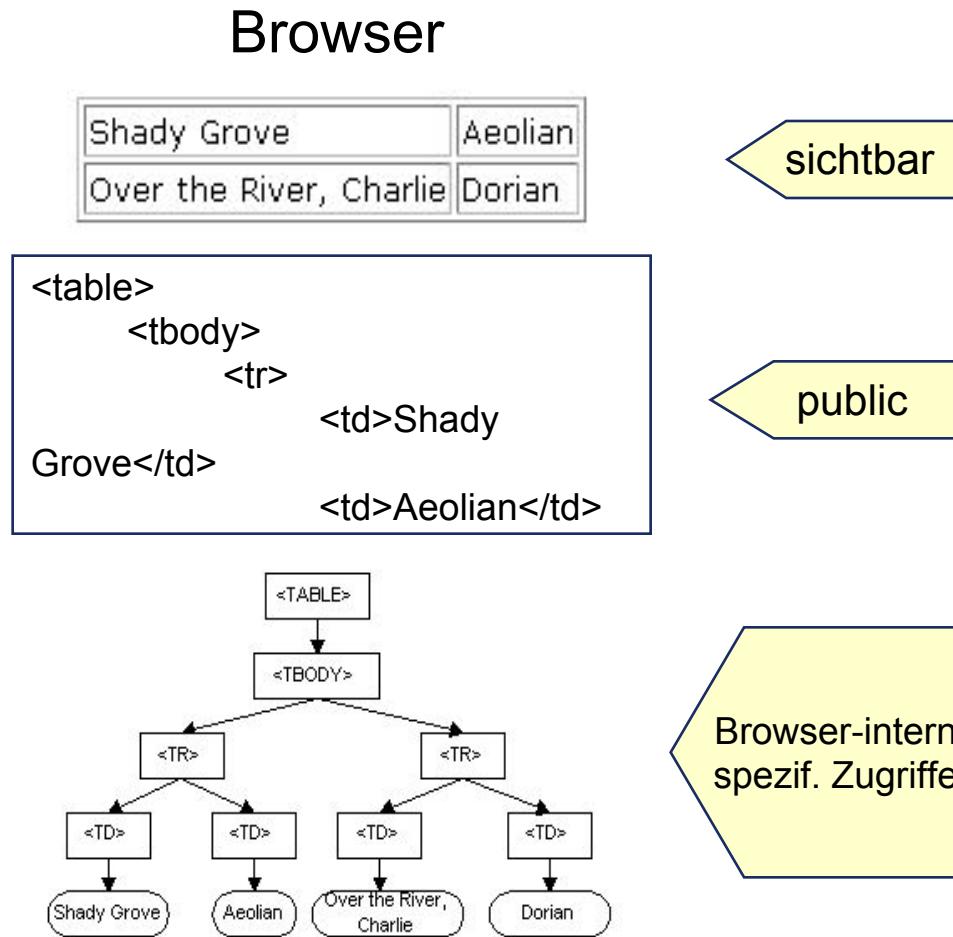
### HTML als Baumstruktur

- ergibt sich zwanglos aus der Schachtelung von Tags



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

## Grundproblem / Grundidee DOM



## Sinn und Zweck des DOM

<http://www.w3.org/TR/DOM-Level-3-Core>

- Status: DOM Level 3 ist W3C Empfehlung
  - ⇒ Vorläufer: browserspezifisches Dynamic HTML
- API für XML Dokumente, spezialisiert für HTML
  - ⇒ Programmierschnittstelle, die Zugriffsmöglichkeiten auf HTML Seiten definiert
  - ⇒ Anwendungen: Animationen mit Dynamic HTML, Rich Internet Applications, Autorenwerkzeuge, Archivierung usw.
- für Skriptsprachen browserunabhängig
  - ⇒ soweit diese auf das HTML Dokument zugreifen (vgl. VBA)
- AJAX (Asynchronous JavaScript And XML)
  - ⇒ Webseiten komfortabel wie klassische GUI-Clients

## DOMs in anderen Systemen

- häufig liegt ein DOM zu Grunde, wenn dokument-basierende Systeme per Skript automatisierbar sind
  - ⇒ ein Großteil des Lernaufwands für den Entwickler steckt nicht in der Skriptsprache, sondern im jeweiligen DOM
- DOMs in Autorensystemen
  - ⇒ Director, Flash, ToolBook
  - ⇒ Attribute und Methoden sind feste Bestandteile von Lingo / ActionScript / OpenScript
- DOMs in Office Programmen
  - ⇒ MS Word, Excel, PowerPoint, Access für Visual Basic
  - ⇒ DOM und Skriptsprache von OpenOffice leider grundverschieden

## 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

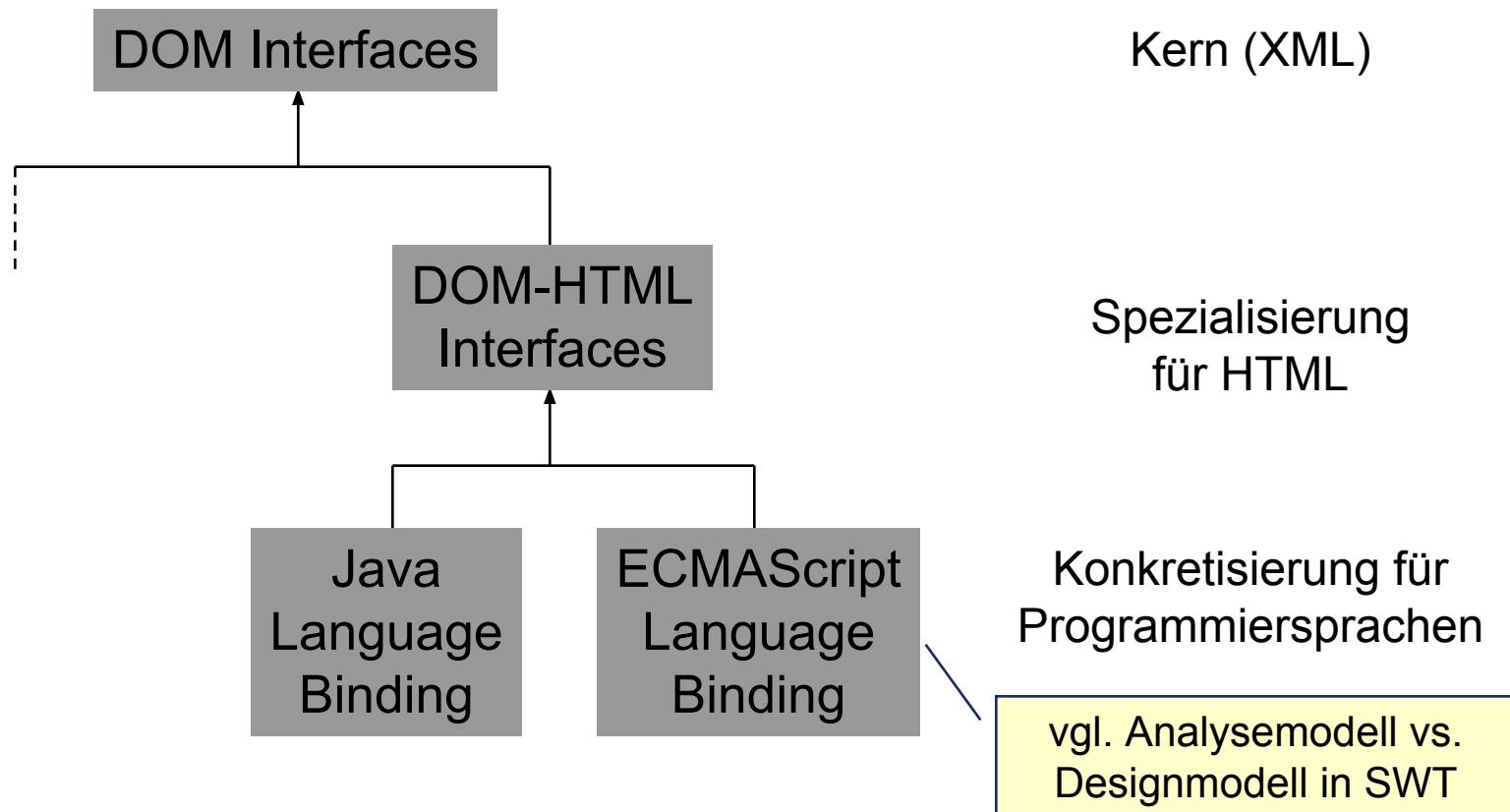
### ■ 2. Schritt: Abbildung auf Programmiersprachen

- ⇒ Implementierung der Interfaces in echter Programmiersprache
- ⇒ bisher Java und JavaScript (ECMAScript)

"language binding"

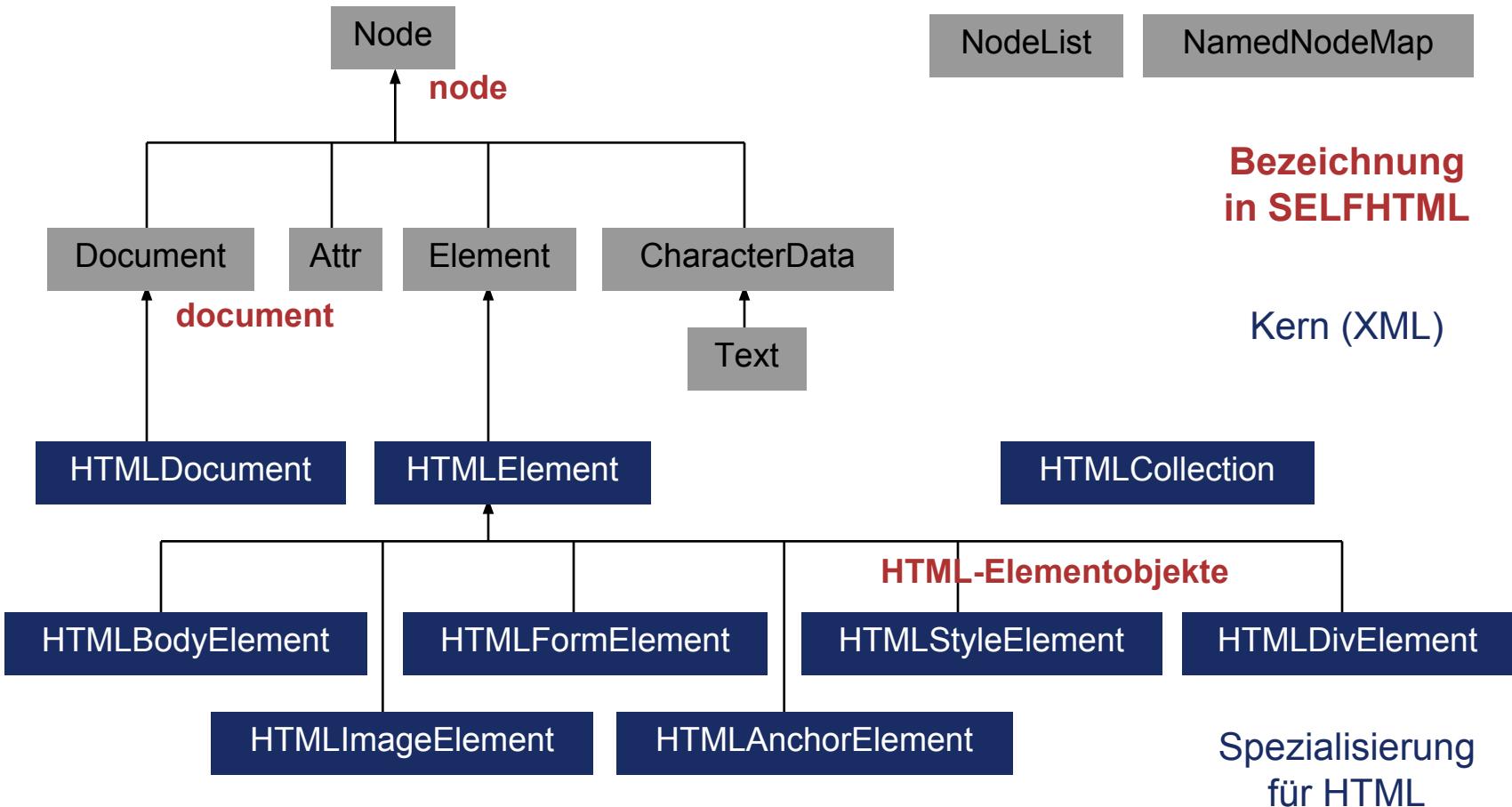
## 2.3.4 ECMAScript: Dokument Objekt Modell – DOM

### Hierarchie der Standardisierungsdokumente



## 2.3.4 ECMAScript: Dokument Objekt Modell – DOM

### Hierarchie der Interfaces und Klassen



# Hochschule Darmstadt

## Fachbereich Informatik

### 2.3.5 ECMAScript: Zugriff auf DOM



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

## Zugriff auf Knoten über Kern-Klassen

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

nur für `document`

## Zugriff auf Knoten über HTML-Collections

- die Klassen HTMLxxxx 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

- **name** ist nur bei manchen Tags zulässig
  - ⇒ muss nicht eindeutig sein
    - bei Radiobuttons: Gruppenbildung über denselben Namen
  - ⇒ wird für verschiedene Zwecke eingesetzt
    - `<a>` Sprungmarke (veraltet; in HTML5 id in beliebigem Tag)
    - `<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.

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

# 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:
  - ⇒ wahlfreier Zugriff ohne Nennung der Collection unter Verwendung des name-Attributs
  - ⇒ `document.MeinFormular.Eingabe.value = "alt";`  
`document.MeinBild.src = "bild.gif";`

findet man oft – ist aber nicht standard-konform!  
Diese Methode verwenden wir NICHT!

## Baumoperationen über Kern-Klassen

sehr hilfreich: Mozilla DOM Inspector

- 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.parentNode`

`Node.firstChild`, `Node.lastChild`

`Node.nextSibling`, `Node.previousSibling`

⇒ Achtung: Blanks zwischen Tags werden in leere Textknoten abgebildet!

Dagegen hilft `Node.firstElementChild`, `Node.lastElementChild`, `Node.nextElementSibling`, `Node.previousElementSibling`

- Methoden zur Strukturänderung

`Node.appendChild(...)`, `Node.removeChild(...)`

`Element.setAttributeNode(...)`

`Element.removeAttribute(...)`

## 2.3.5 ECMAScript: Zugriff auf DOM

### Zugriff auf Attribute

- alle Attributwerte in DOM2 HTML sind Strings
- Zugriff über Kern-Klassen (allgemein)
  - ⇒ 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");
var bilddatei = meinBild.getAttribute("src");
```
  - ⇒ (Attributknoten allokieren und in Baum einbauen)
- Zugriff über HTML-Klassen (kompakt)
  - ⇒ alle Klassen HTMLxxxElement haben ihre jeweiligen Attribute
  - ⇒ 

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

DOM2 Core

DOM2 HTML

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



## 2.3.5 ECMAScript: Zugriff auf DOM

## Zugriff auf vom Programmierer definierte Datenattribute

- Zu jedem Tag können Datenattribute hinzugefügt werden
  - ⇒ mit dem Prefix "data-" zur Kennzeichnung
  - ⇒ mit einem Namen ohne Großbuchstaben
  - ⇒ Bsp.: data-preis, data-key, data-xxx
  - ⇒ In HTML oder auch über das DOM
  - ⇒ Der Zugriff erfolgt im DOM über ...dataset.preis
- Beispiel Pizza-Service:
  - ⇒ Definition: `<li id="S17" data-preis="4.99">Salami</li>`
  - ⇒ Zugriff im DOM:

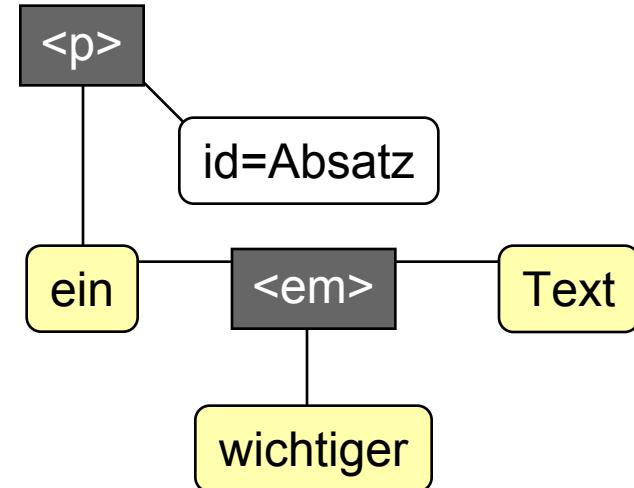
```
var Pizza=document.getElementById("S17");
var Preis=parseFloat(Pizza.dataset.preis);
//konventionelle Methode klappt immer:
var Preis=parseFloat(Pizza.getAttribute("data-preis"));
```

Siehe auch: <http://www.w3.org/html/wg/drafts/html/master/dom.html#custom-data-attribute>

## 2.3.5 ECMAScript: Zugriff auf DOM

### Zugriff auf Text

```
<p id="Absatz">ein
 wichtiger 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)

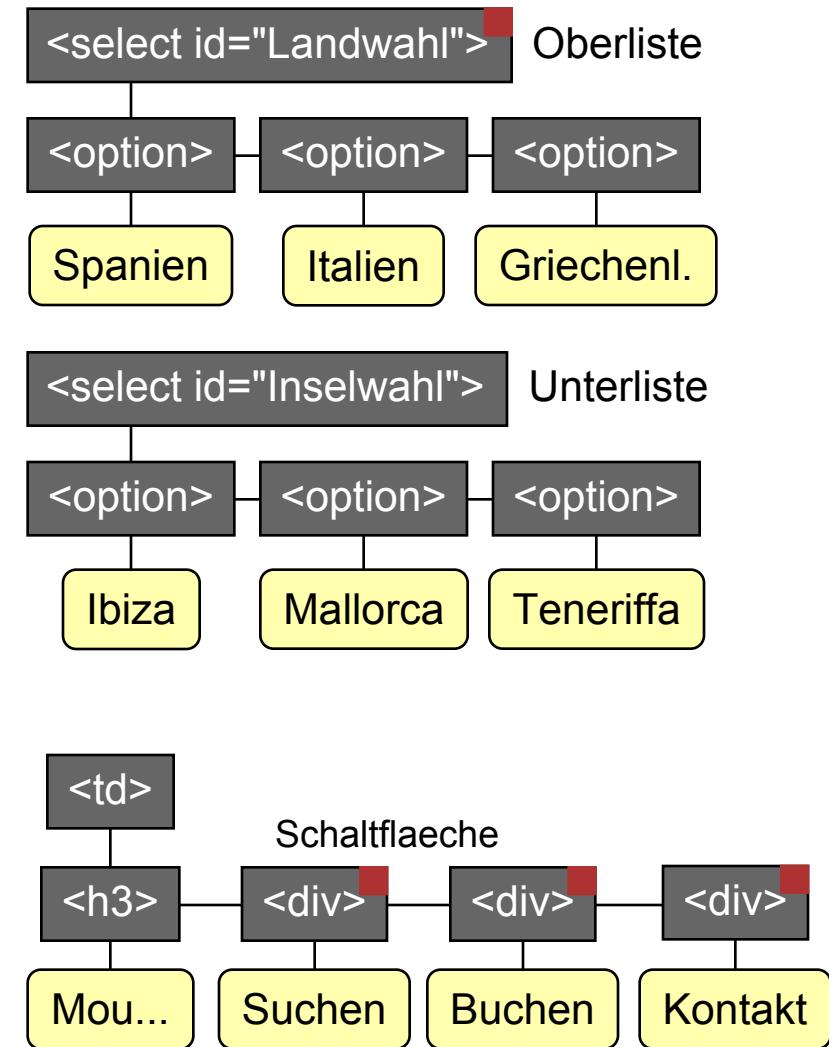
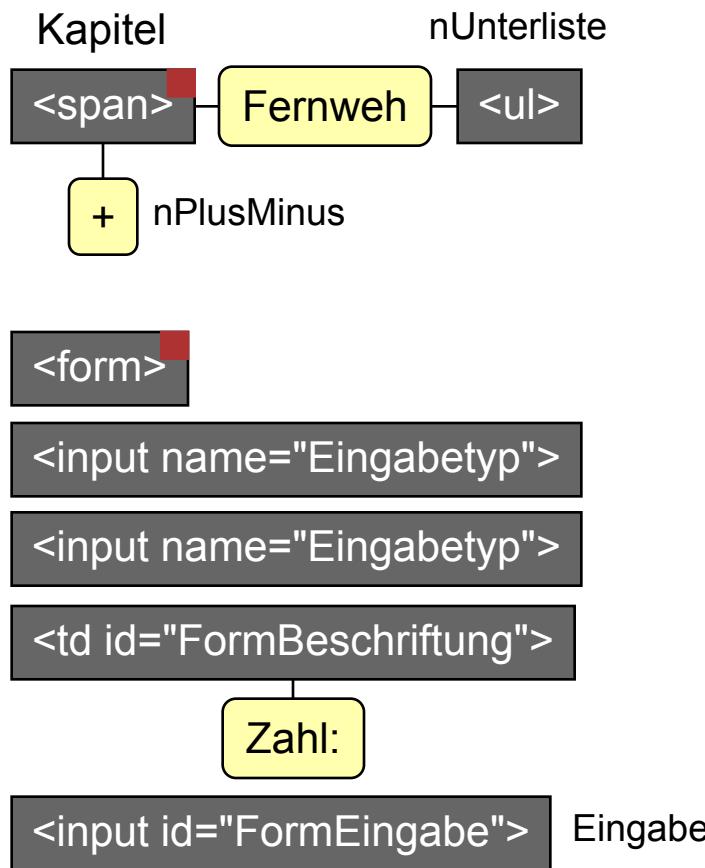
```
var Absatz = document.getElementById("Absatz");
var ein = Absatz.firstChild.nodeValue;
var em = Absatz.firstChild.nextSibling;
var wichtiger = em.firstChild.nodeValue;
var Text= Absatz.lastChild.nodeValue;
```

## Zugriff auf Styles

- auf inline-Styles des HTML-Elements, nicht auf CSS-Datei
  - ⇒ `<p id="Hugo" style="font-size:1.2em; 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 = "1.2em";`

## 2.3.5 ECMAScript: Zugriff auf DOM

# Beispiele für ECMAScript und DOM



## 2.3.5 ECMAScript: Zugriff auf DOM

# Beispiel ECMA-Script und DOM (I)



### Beispiele für ECMAScript und DOM

**Auf- und Zuklappen von Unterpunkten**

- + Fernweh
- + Kontakt

---

Kapitel      nUnterliste

```

 Fernweh
 + nPlusMinus

```

**Auswahlhierarchie**

Wählen Sie erst das Land...

- Spanien
- Italien
- Griechenland

...und dann die Insel:

- Ibiza
- Mallorca
- Teneriffa

---

<select id="Landwahl"> Oberliste

```

<option> <option> <option>
 Spanien Italien Griechenl.

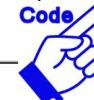
```

<select id="Inselwahl"> Unterliste

```

<option> <option> <option>
 Ibiza Mallorca Teneriffa

```



```

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);
 }
}

```

## 2.3.5 ECMAScript: Zugriff auf DOM

# Beispiel ECMA-Script und DOM (II)



**Mouseover-Effekte**

Suchen  
Buchen  
Kontakt

<td>  
<h3> Schaltflaeche <div> <div> <div>  
Mou... Suchen Buchen Kontakt

**Überprüfung von Eingaben**

nur Buchstaben   
nur Zahlen   
Buchstaben:   
Abschicken

<form>  
<input name="Eingabetyp">  
<input name="Eingabetyp">  
<td id="FormBeschriftung">  
    Zahl:  
<input id="FormEingabe"> Eingabe



```
// 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");
}
```



### 2.3.5 ECMAScript: Zugriff auf DOM

## Beispiel: DOM-Zugriffe aus ECMA-Script (DOM Core)

- // UnterlisteID ist die ID meiner Select-Liste  
`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 einhängen  
`Unterliste.appendChild(neuesElement);`

Low-Level Verfahren  
DOM Core



## Beispiel: DOM-Zugriffe aus ECMA-Script (DOM HTML)

- // UnterlisteID ist die ID meiner Select-Liste

```
var Unterliste = document.getElementById
("UnterlisteID");
```

- // offline!!

// den neuen Teilbaum anlegen und initialisieren

```
var neuesElement =
document.createElement("option");
neuesElement.text = "Meine Option";
neuesElement.value = "1"; // optional
neuesElement.selected = true; // optional
```

⇒ new Option(...) ist veraltet und kein Standard!

erzeugt bereits ein  
HTMLOptionElement

High-Level Verfahren  
DOM HTML

- // und die neue Option hinten anhängen

```
Unterliste.options[Unterliste.length]=neuesElement;
```

# Hochschule Darmstadt

## Fachbereich Informatik

### 2.3.6 Eventhandling



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

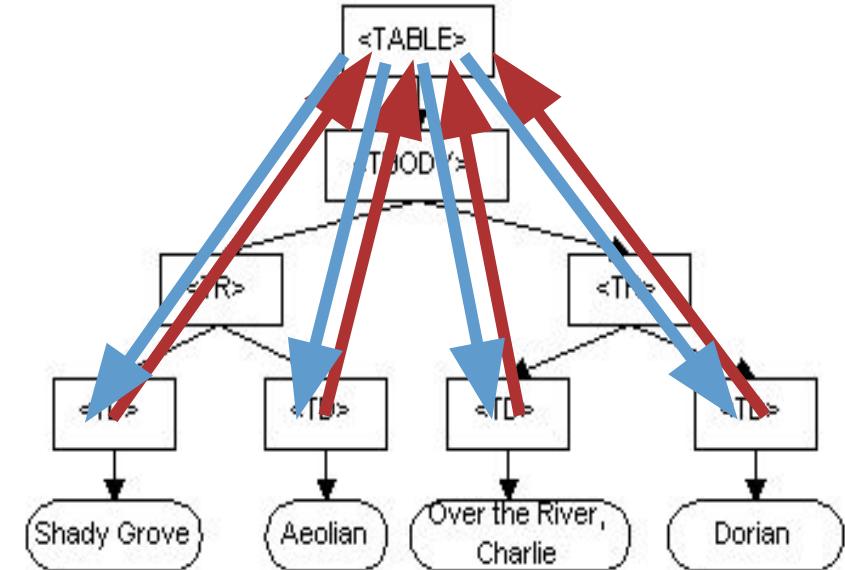
**fbi**

FACHBEREICH INFORMATIK

## 2.3.5 ECMAScript: Zugriff auf DOM

### Ereignisweiterleitung

- typisch für Ereignisorientierung
  - ⇒ vgl. ToolBook, Director, Windows
- Ereignisse werden in der DOM-Hierarchie weitergeleitet
  - ⇒ **Netscape (event capturing):** in der HTML-Schachtelungsstruktur von außen nach innen
  - ⇒ **Microsoft (event bubbling):** in der HTML-Schachtelungsstruktur von innen nach außen
  - ⇒ W3C: erst **von außen nach innen**, dann **von innen nach außen**
- Zuordnung (Registrierung) von Handlern
  - ⇒ an allen besuchten Elementen können Handler aufgerufen werden
    - default ist **bubbling**; **capturing** nur via addEventListener-Parameter
  - ⇒ meistens wird aber pro Ereignis nur ein Handler registriert



## 2.3.5 ECMAScript: Zugriff auf DOM

### Ereignisweiterleitung: Beispiele

- Event Bubbling



- Event Bubbling Beispiel 2



- StopPropagation() Beispiel



**Wichtig** für die Beispiele: in JSFiddle unter dem Menupunkt “Frameworks & Extensions nicht “onLoad” sondern “no wrap - in <head>” auswählen! (JSFiddle-spezifisches Problem, dass uns nicht interessiert)



# Hochschule Darmstadt

## Fachbereich Informatik

### 2.3.6 Ajax



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

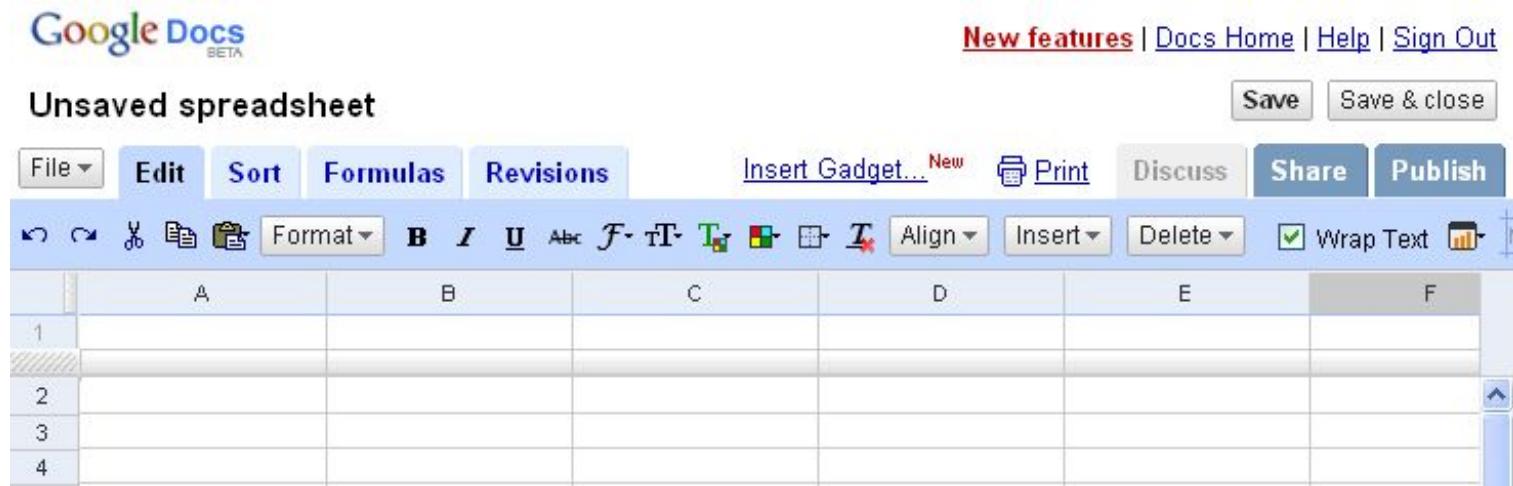
### 2.3.6 Ajax

## Ajax – Asynchronous JavaScript and XML

- Ziel: webbasierte Anwendungen sollen sich anfühlen wie klassische GUI-basierende
  - ⇒ ohne Neuladen der Seite für jede Kleinigkeit
  - ⇒ Rich Internet Applications
  - ⇒ z.B. Google Docs, Office 265 Online

Begriff geprägt  
von J.J.Garrett  
2005

vorher schon  
intensiv genutzt  
durch Google, z.  
B.  
Google Maps



The screenshot shows the Google Docs interface for an unsaved spreadsheet. At the top, there's a navigation bar with 'File', 'Edit', 'Sort', 'Formulas', 'Revisions' (selected), 'Insert Gadget...', 'Print', 'Discuss', 'Share', and 'Publish'. Below the navigation bar is a toolbar with various icons for text styling (bold, italic, underline) and alignment. The main area is a grid with columns labeled A through F and rows labeled 1 through 4. A blue hand icon is positioned in the bottom right corner of the grid.

## Ajax – Grundidee



- **Funktionsprinzip**
  - ⇒ HTML-Seite wird nicht neu geladen
  - ⇒ Daten werden bei Bedarf im Hintergrund vom Server nachgeladen
  - ⇒ JavaScript behandelt verschiedene Ereignisse und manipuliert bei Bedarf das DOM
- **Komponenten sind bereits weitgehend bekannt**
  - ⇒ HTML, CSS, JavaScript, DOM, (XML)
- **Neuerungen**
  - ⇒ XMLHttpRequest zum Laden im Hintergrund
  - ⇒ JavaScript wird essentiell – ohne JavaScript funktioniert AJAX nicht
  - ⇒ es gibt diverse Frameworks zur Erleichterung des Umgangs mit JavaScript und DOM (z.B. jQuery, prototype.js, Dojo)

"asynchron"



## Hinführendes Beispiel – Teil1

- Wir entwickeln eine HTML-Seite, welche auf Knopfdruck einen String auf einer Webseite einfügt
  - ⇒ einfaches HTML mit einem Button, etwas Javascript und DOM

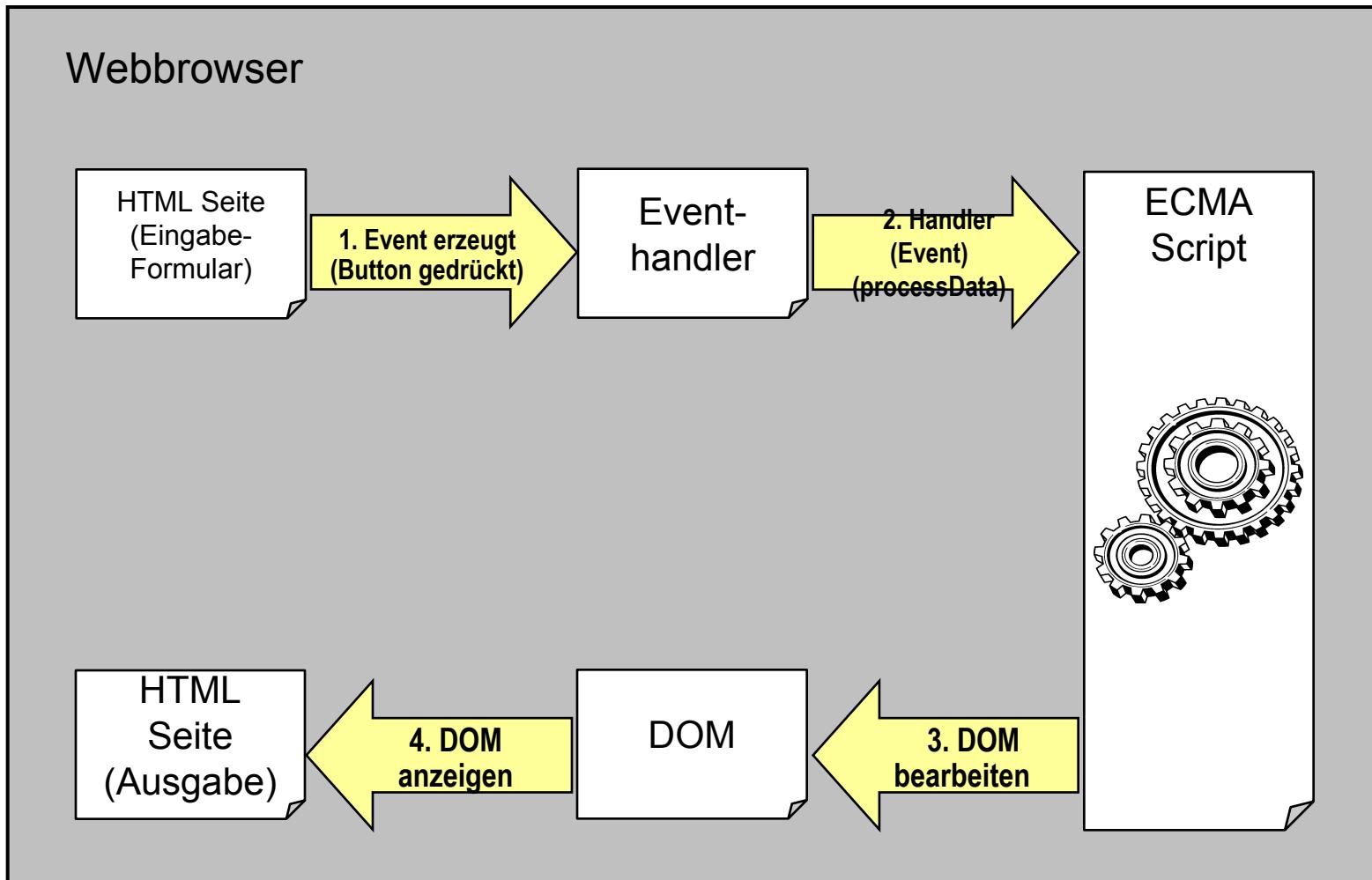
```
<!DOCTYPE html>
<html lang="de">
<head>
 <meta charset="UTF-8" />
 <title>Test</title>
 <script type="text/javascript">
function processData () {
 var myText = document.getElementById("meinText");
 myText.firstChild.nodeValue = '26.01.2009 11:51:02';
}
</script>
</head><body><h1>Hier steht der Text</h1>
<p id="meinText">mein Text</p>
<form action="index.html"> <input type="button" value="Los!">
 onclick="processData()"/>
</form> </body> </html>
```



Der Text wird bei  
Knopfdruck eingefügt

## 2.3.6 Ajax

### Schematischer Ablauf zu Beispiel 1



## Hinführendes Beispiel – Teil 2

- Es sollen nun Daten eingefügt werden, die der Webserver liefert
  - ⇒ Ein Webserver bietet die Möglichkeit Programme aufzurufen z.B. PHP-Code um die aktuelle Uhrzeit als String abzufragen
  - ⇒ aber dann wartet ("hängt") der Browser bis der Webserver antwortet
  - ⇒ und wie kann man neue Daten an eine bereits geladenen HTML-Seite übertragen, ohne die ganze Seite neu zu laden?
- **XMLHttpRequest** ermöglicht genau diese Funktion
  - ⇒ Festlegung einer Funktion zur Verarbeitung von (zurückkommenden) Daten ("Callback-Handler")
  - ⇒ Festlegen der Abfrage (URL etc.)
    - in Unserem Fall: Abfrage der public API von Github
  - ⇒ Absenden der Abfrage mit asynchroner Antwort

```
var request = new XMLHttpRequest(); // RequestObject anlegen
request.open("GET", url); // URL für HTTP-GET festlegen
request.onreadystatechange = processData; // Callback-Handler zuordnen
request.send(); // Request abschicken
```



## 2.3.6 Ajax

### Callbacks

#### ■ Callback-Beispiel:

```
1 function mySuccessFunction(message){
2 doSomeWork();
3 console.log('Success:' +message);
4 }
5
6 function myFailureFunction(message){
7 preventShitFromHittingTheFan();
8 console.log('Error:' +message);
9 }
10 }
11
12 function mySuperAwesomeFunction(successCallback, failureCallback){
13 var retMsg = anyOldFunctionWithAReturnValue();
14
15 if(retMsg.indexOf('error') == -1){
16 successCallback(retMsg);
17 }else{
18 failureCallback(retMsg);
19 }
20 }
21
22 mySuperAwesomeFunction(mySuccessFunction, myFailureFunction);
23 //oder
24 mySuperAwesomeFunction(myOtherSuccessFunction, myOtherFailureFunction);
```

Callback-Funktionen



### 2.3.6 Ajax

## Codeauszüge zu Beispiel 2 (Teil 1)



```
<!-- HTML wie in Beispiel 1 -->
<input type="button" id="myb" value="Los!" onclick="requestData()"/>
...
// request als globale Variable anlegen (haesslich, aber bequem)
var request = new XMLHttpRequest();

function requestData() { // Daten asynchron anfordern
 request.open("GET", "php/0_zeit.php"); // URL für HTTP-GET
 request.onreadystatechange = processData; //Callback-Handler zuordnen
 request.send(null); // Request abschicken
}
```

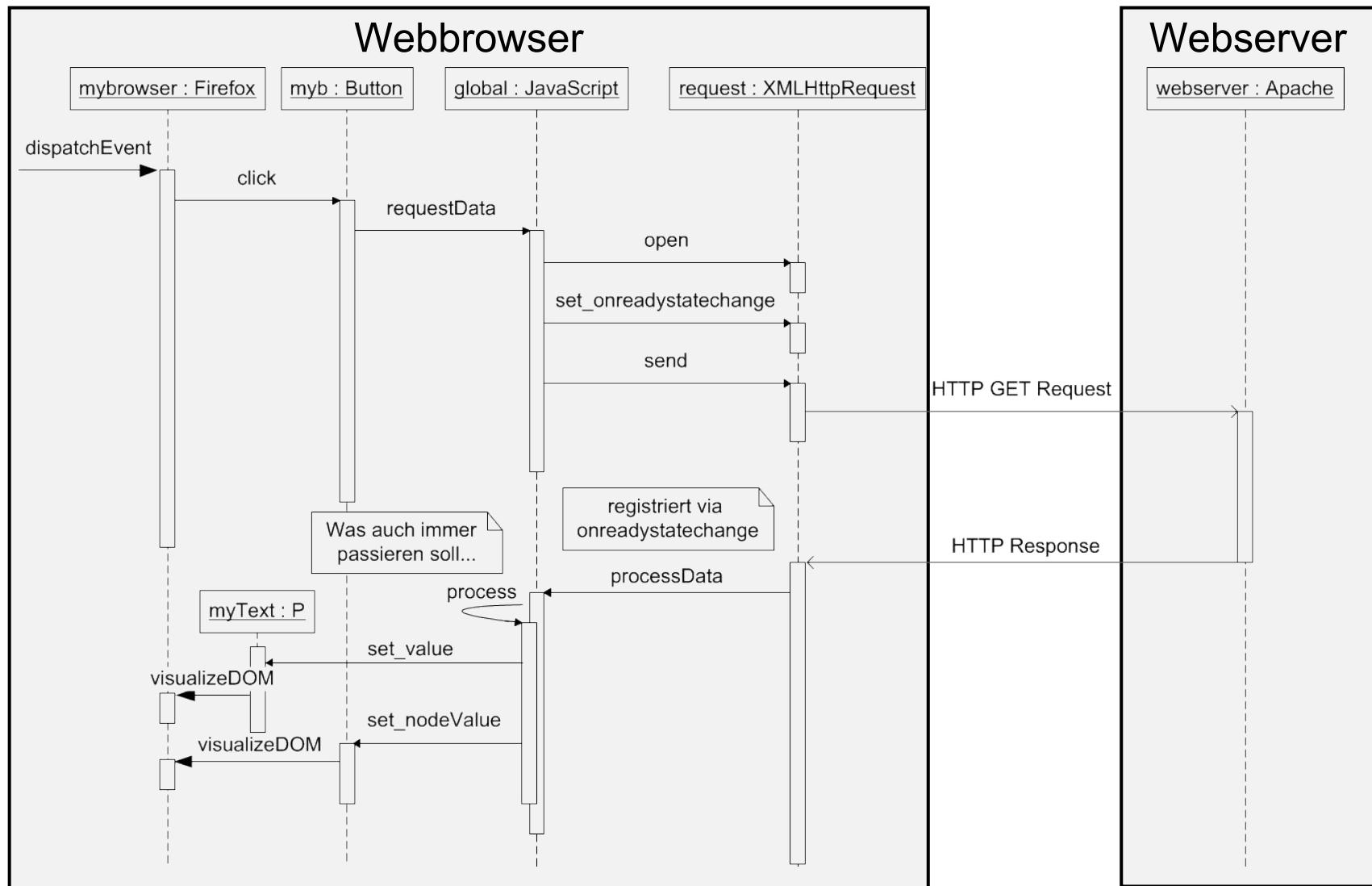
## Codeauszüge zu Beispiel 2 (Teil 2)

```
function processData() {
 if(request.readyState == 4) { // Uebertragung = DONE
 if (request.status == 200) { // HTTP-Status = OK
 if(request.responseText != null)
 process(request.responseText); // Daten verarbeiten
 else error ("Dokument ist leer");
 } else error ("Uebertragung fehlgeschlagen");
 } else ; // Uebertragung laeuft noch
}

function process (intext) { // Text ins DOM einfuegen
 var myText = document.getElementById("myText");
 myText.firstChild.nodeValue = intext;
 document.getElementById("myb").value="Fertig!";
}
```

## 2.3.6 Ajax

# Sequenzdiagramm zu Beispiel 2



### 2.3.6 Ajax

## XMLHttpRequest im Detail (1)

<http://www.w3.org/TR/XMLHttpRequest>

aus W3C Working Draft:



```
interface XMLHttpRequest
{
 // event handler
 attribute EventListener onreadystatechange; // Zuordnung der Callback-Funktion

 // state
 const unsigned short UNSENT = 0;
 const unsigned short OPEN = 1;
 const unsigned short SENT = 2;
 const unsigned short LOADING = 3;
 const unsigned short DONE = 4;
 readonly attribute unsigned short readyState; // aktueller Zustand der Übertragung
 ↓
```



## 2.3.6 Ajax

### XMLHttpRequest im Detail (2)

```
↓
// request

void open(in DOMString method, in DOMString url); // Anforderung vorbereiten
void open(in DOMString method, in DOMString url, in boolean async);
void open(in DOMString method, in DOMString url, in boolean async,
 in DOMString user);
void open(in DOMString method, in DOMString url, in boolean async,
 in DOMString user, in DOMString password);

void setRequestHeader(in DOMString header, in DOMString value); // für HTTP

void send(); // Anforderung abschicken
void send(in DOMString data);
void send(in Document data);

void abort(); // Übermittlung abbrechen
↓
```

ggf. mit GET-Daten

POST-Daten

### 2.3.6 Ajax

## XMLHttpRequest im Detail (3)

↓

```
// response
DOMString getAllResponseHeaders(); // HTTP Header der Antwort
DOMString getResponseHeader(in DOMString header);
```

```
readonly attribute DOMString responseText; // Ergebnis als Text ...
readonly attribute Document responseXML; // ... oder XML DOM
```

```
readonly attribute unsigned short status; // HTTP Status als Nummer ...
readonly attribute DOMString statusText; // ... oder Klartext
};
```

### ■ XMLHttpRequest sendet und erwartet standardmäßig UTF-8

- ⇒ wenn das Gesamtsystem mit UTF-8 arbeitet, sind keine besonderen Maßnahmen erforderlich
- ⇒ sonst muss in PHP mit utf8\_decode ausgewertet und mit utf8\_encode geantwortet werden

### 2.3.6 Ajax

## XML in AJAX

- Bisher haben wir nur einfache Strings vom Webserver übertragen und überhaupt kein XML verwendet
  - ⇒ damit lassen sich aber nur schwer komplexere Datensätze übertragen
  - ⇒ **XMLHttpRequest** erlaubt auch die Übertragung von XML
  - ⇒ der Zugriff erfolgt dann für einen **request** über **request.responseXML** (statt **request.responseText**)
  - ⇒ der XML-Ausdruck wird automatisch eingelesen und als ein separates DOM zur Verfügung gestellt
  - ⇒ der Zugriff erfolgt mit normalen DOM-Methoden – also z.B. **request.responseXML.getElementsByTagName('zeit');**
- Das Umwandeln der Daten von bzw. nach XML ist umständlich
  - ⇒ "Serialisierung" der Objekte mit Attributen
  - ⇒ XML ist zwar sehr flexibel – aber es geht einfacher, wenn man diese Flexibilität nicht braucht

## Alternative Datenübertragung mit JSON

### ■ JSON (JavaScript Object Notation)

- ⇒ Einfache und schlanke Notation zur Darstellung von Listen, Strings, Zahlen und assoziativen Arrays
- ⇒ die gängigen Programmiersprachen bieten eine JSON Bibliothek zum Serialisieren von Objekten nach JSON
- ⇒ der ECMA Script Befehl `JSON.parse()` deserialisiert ein übergebenes Argument und liefert das Ergebnis als Object (z.B. assoziatives Array)
- ⇒ Umwandlung von JSON -> String mit `JSON.stringify()`

Objekt "Kunde" in XML:

```
<kunde>
 <id>481048</id>
 <name>James Bond</name>
 <email>hurz@glgl.de</email>
 <artikel>17</artikel>
 <artikel>22</artikel>
</kunde >
```

Objekt "Kunde" in JSON:

```
{
 "id" : 481048,
 "name" : "James Bond",
 "email" : "hurz@glgl.de",
 "artikel" : [
 17,
 22
]
}
```

### 2.3.6 Ajax

## AJAX – was kann bzw. muss man noch machen?

- Mit einem **XMLHttpRequest** können auch Daten an den Server geschickt werden
  - ⇒ per GET: `request.open("GET", "script.php?name="+encodeURI(value))`
  - ⇒ per POST: `request.send("name="+value)`
  - ⇒ das auf dem Webserver aufgerufene Programm (url aus `request.open`) muss die Daten dann entsprechend verarbeiten
- Bei der Datenübertragung und bei der Darstellung kann sehr viel schief gehen
  - ⇒ die verschiedenen Browser reagieren oft unterschiedlich
  - ⇒ ordentliche Fehlerbehandlung und gründliche Tests sind sehr wichtig



### 2.3.6 Ajax

## Event-Loop und Worker Threads

- Ein Browserfenster muss viele Events verarbeiten
  - ⇒ Seitenaufbau, Handler, Timer Callbacks, XMLHttpRequest Callbacks etc.
- Es gibt einen GUI Thread pro Browser-Tab bzw. -Fenster
  - ⇒ in diesem Thread läuft eine Message Loop
  - ⇒ Alle Aktivitäten werden in diesem Thread sequentialisiert
  - ⇒ Aktivitäten unterbrechen sich nicht gegenseitig
  - ⇒ gleiches Verhalten wie bei Java Swing
- Die Laufzeit von Handlern muss "kurz" sein, sonst friert die GUI ein!
- Für aufwändige Berechnungen gibt es separate Threads:
  - ⇒ "Web Worker"
  - ⇒ haben keinen direkten Zugriff auf DOM



```
1 | while (queue.waitForMessage()) {
2 | queue.processNextMessage();
3 | }
```

### 2.3.6 Ajax

## AJAX - / Javascript-Frameworks

- Es gibt diverse Frameworks, welche Bedienelemente, Animationen, Drag&Drop, Hilfsfunktionen für DOM uvm. zur Verfügung stellen

- ⇒ JavaScript-Bibliotheken mit Funktionen, die eingebunden werden

- Dojo unter <http://dojotoolkit.org>
    - prototype.js unter <http://www.prototypejs.org>
    - Script.aculo.us unter <http://script.aculo.us> (basiert auf prototype.js)
    - jQuery unter <http://jquery.com/>

Die Frameworks verwenden oft unterschiedliche Paradigmen!

- ⇒ Mit solchen Frameworks ist der Einsatz von Effekten sehr einfach.  
Zum Beispiel mit Script.aculo.us :

```
var myText = document.getElementById("myText");
myText.appear();
myText.pulsate({pulses: 5, duration: 1.5 });
<h1 onclick="this.fade()">Gleich kommt's!</h1>
```

- Ein gutes Design will aber trotzdem gelernt sein...

- ⇒ allein durch Effekte werden weder Inhalt noch Layout besser

# Vor- und Nachteile

## ■ Technisch:

- ⇒ kein Plug-In erforderlich
  - aber JavaScript muss aktiv sein
- ⇒ aufwendig zu testen
  - wie jede richtige Applikation
- ⇒ leidet unter Browser Bugs
  - wie alles im Web
- ⇒ MVC-Architektur vermisst Push vom Server
  - Polling als Ersatz
- ⇒ Ladezeit für Frameworks vs. schnelleres Nachladen
- ⇒ Barrierefreiheit leidet

## ■ sieht aus wie ein Browser, verhält sich aber nicht so:

- ⇒ kein Zurück-Button
  - Benutzer erwarten ein "Undo"
- ⇒ kein Lesezeichen setzbar
- ⇒ von Suchmaschinen nicht auffindbar
- ⇒ visuelles Feedback sollte Benutzer vertrösten
  - Sanduhr während Ladezeit

## Zusammenfassung I

### ■ Skripte: Große „Freiheit“ in der Entwicklung

- ⇒ Mache das Browser-Ergebnis möglichst eindeutig
  - seit HTML5 ist das DOM zu jedem HTML-Konstrukt genau definiert
  - vor HTML5 erzeugten verschiedene Browser verschiedene DOMs zu einer Webseite – und verhielten sich entsprechend unterschiedlich
  - 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 und gründlich Testen
  - Validatoren für HTML, CSS
  - Codeanalyse mit JSLint
  - Skript-Debugger

z.B. Browser-Developer Tools für Firefox verwenden: Firebug, WebDeveloper, HTML-Validator, DOM Inspector...

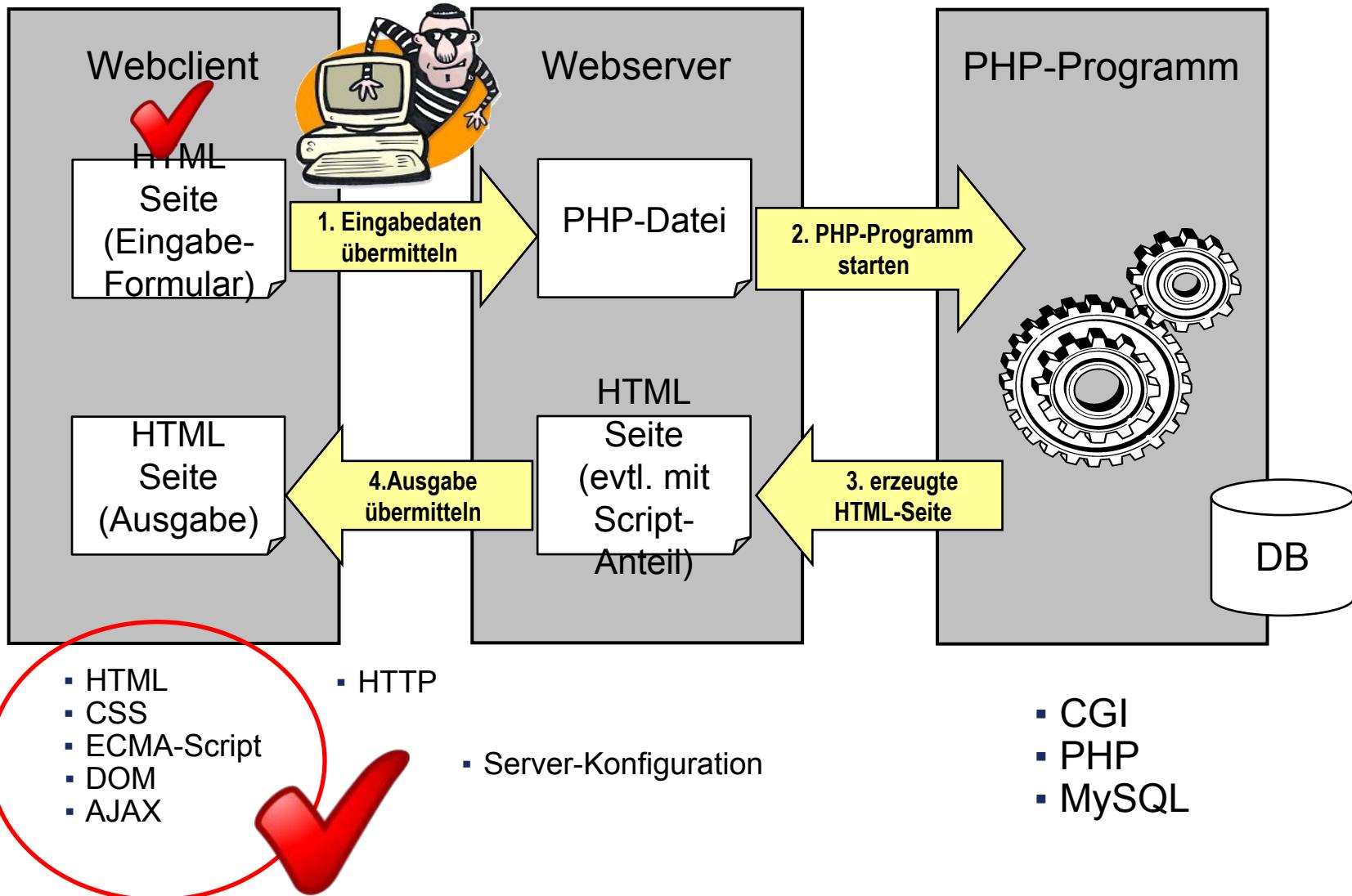
## Zusammenfassung II

- 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
- Ajax

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

## 2. Webclient

# Übersicht



# Hochschule Darmstadt

## Fachbereich Informatik

### 3. Webserver



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

## 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



**HTTP = Hypertext Transfer Protocol**

**HTTP V0.9 1991 Hatte nur GET**

**HTTP V1.0 1996 standardisiert als RFC1945**

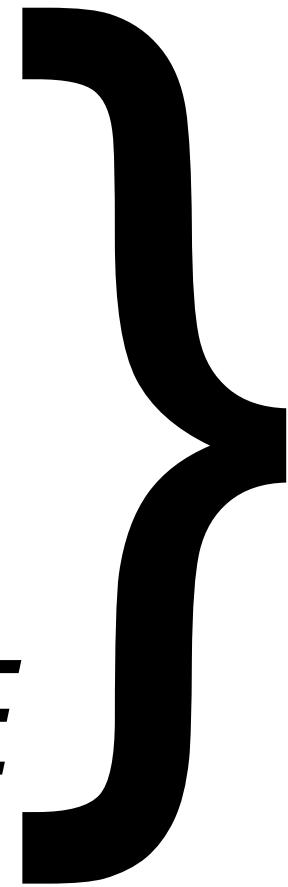
**HTTP V1.1 1996 ist bereits auf dem Vormarsch und findet weite Verbreitung**

**HTTP V1.1 1999 standardisiert als RFC 2616**

*SPDY 2012 von Google*

**HTTP V2.0 2015 standardisiert als RFC 7540**

*GET*  
*POST*  
*PUT*  
*DELETE*



**CRUD**



Create = POST

Read = GET

Update = PUT

Delete = DELETE

## Anfrage Methoden

GET  
POST  
PUT  
DELETE



CRUD

HEAD ähnlich GET aber nur Header ohne Content

TRACE liefert die Anfrage so zurück, wie der Server sie empfangen hat.

OPTIONS liefert eine Liste der vom Server unterstützten Methoden und Merkmale.

CONNECT wird von Proxyservern implementiert, die in der Lage sind, SSL-Tunnel zur Verfügung zu stellen.

PATCH ähnlich PUT

# Unterschied von GET

```
GET /annas/banane HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: obj=true; admin=true; JSESSIONID=q31taeoild4215x0l3690a3sw; userid=rh5v92ierojhgi8ilfm1bkka7q; actions=%5Bfalse%2Ctrue%2Ctrue%2Ctrue%2Cfalse%5D
```

# und POST

```
POST /index.php/category HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 46
Cache-Control: no-cache
Origin: chrome-extension://mkhojklkhkdaghjjfdnphfphiaiohkef
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: /*
Accept-Encoding: gzip, deflate
Accept-Language: de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: obj=true; admin=true; JSESSIONID=q31taeoild4215x0l3690a3sw; userid=rh5v92ierojhgi8ilfm1bkka7q; actions=%5Bfalse%2Ctrue%2Ctrue%2Ctrue%2Cfalse%5D

name=Mustermann&testkey=testvale&banane=ananas|
```

## form-data

```
POST /index.php/category HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 445
Cache-Control: no-cache
Origin: chrome-extension://mkhojklkhkdaghjjfdnphfphiaiohkef
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryzKp4A6hr8T3v7lcL
Accept: /*
Accept-Encoding: gzip, deflate
Accept-Language: de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: obj=true; admin=true; JSESSIONID=q31taeoild4215x0l3690a3sw; userid=rh5v92ierojhgi8ilfm1bkka7q; actions=%5Bfalse%2Ctrue%2Ctrue%2Ctrue%2Cfalse%5D

-----WebKitFormBoundaryzKp4A6hr8T3v7lcL
Content-Disposition: form-data; name="name"

Musterman
-----WebKitFormBoundaryzKp4A6hr8T3v7lcL
Content-Disposition: form-data; name="testkey"

testvalue
-----WebKitFormBoundaryzKp4A6hr8T3v7lcL
Content-Disposition: form-data; name="hurz"

der hase
-----WebKitFormBoundaryzKp4A6hr8T3v7lcL
Content-Disposition: form-data; name="der Igel"

hurz
-----WebKitFormBoundaryzKp4A6hr8T3v7lcL--
```

## x-www-form-urlencoded

```
POST /index.php/category HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 46
Cache-Control: no-cache
Origin: chrome-extension://mkhojklkhkdaghjjfdnphfphiaiohkef
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */
Accept-Encoding: gzip, deflate
Accept-Language: de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: obj=true; admin=true; JSESSIONID=q31taeoild4215x0l3690a3sw; userid=rh5v92ierojhgi8ilfm1bkka7q; actions=%5Bfalse%2Ctrue%2Ctrue%2Ctrue%2Cfalse%5D
name=Mustermann&testkey=testvale&banane=ananas|
```

# JSON

```
POST /index.php/category HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 65
Cache-Control: no-cache
Origin: chrome-extension://mkhojklkhkdaghjjfdnphfphiaiohkef
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86 Safari/537.36
Content-Type: text/plain; charset=UTF-8
Accept: /*
Accept-Encoding: gzip, deflate
Accept-Language: de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: obj=true; admin=true; JSESSIONID=q31taeoild4215x0l3690a3sw; userid=rh5v92ierojhgi8ilfm1bkka7q; actions=%5Bfalse%2Ctrue%2Ctrue%2Ctrue%2Cfalse%5D
{
 "hase": "Igel",
 "hurz": "Elefant",
 "name": "Musterman"
}
```

# Die Wichtigsten:

- 200 = Alles ok
- 403 = Forbidden
- 404 = Not found
- 500 = Internal Server Error

weitere unter <https://de.wikipedia.org/wiki/HTTP-Statuscode>

# Wie sieht HTTP aus?

Das sieht der HTTP-Server

Anfrage an Wikipedia



## HTTP 2.0



<http://www.http2demo.io/>

- Dauerhafte Verbindung zum Server
- Löst *Long Polling* ab
- Probleme mit Proxys

### Anfrage Webbrowser

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

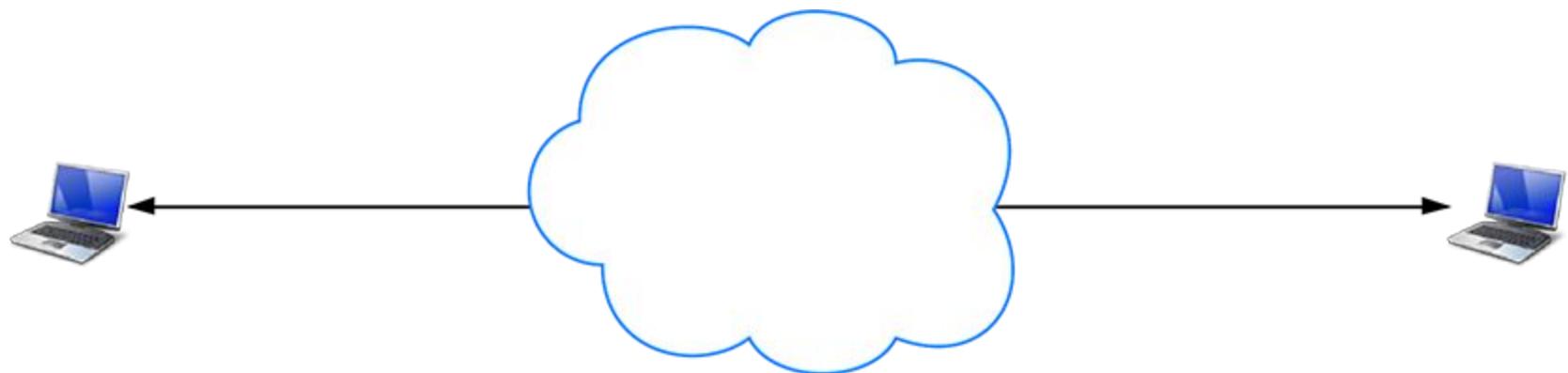
### Anfrage Webbrowser

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept:
s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

- P2P für Video
- Ermöglicht VoIP
- Chat
- File Transfer
- Das sind allein IP-Adressen

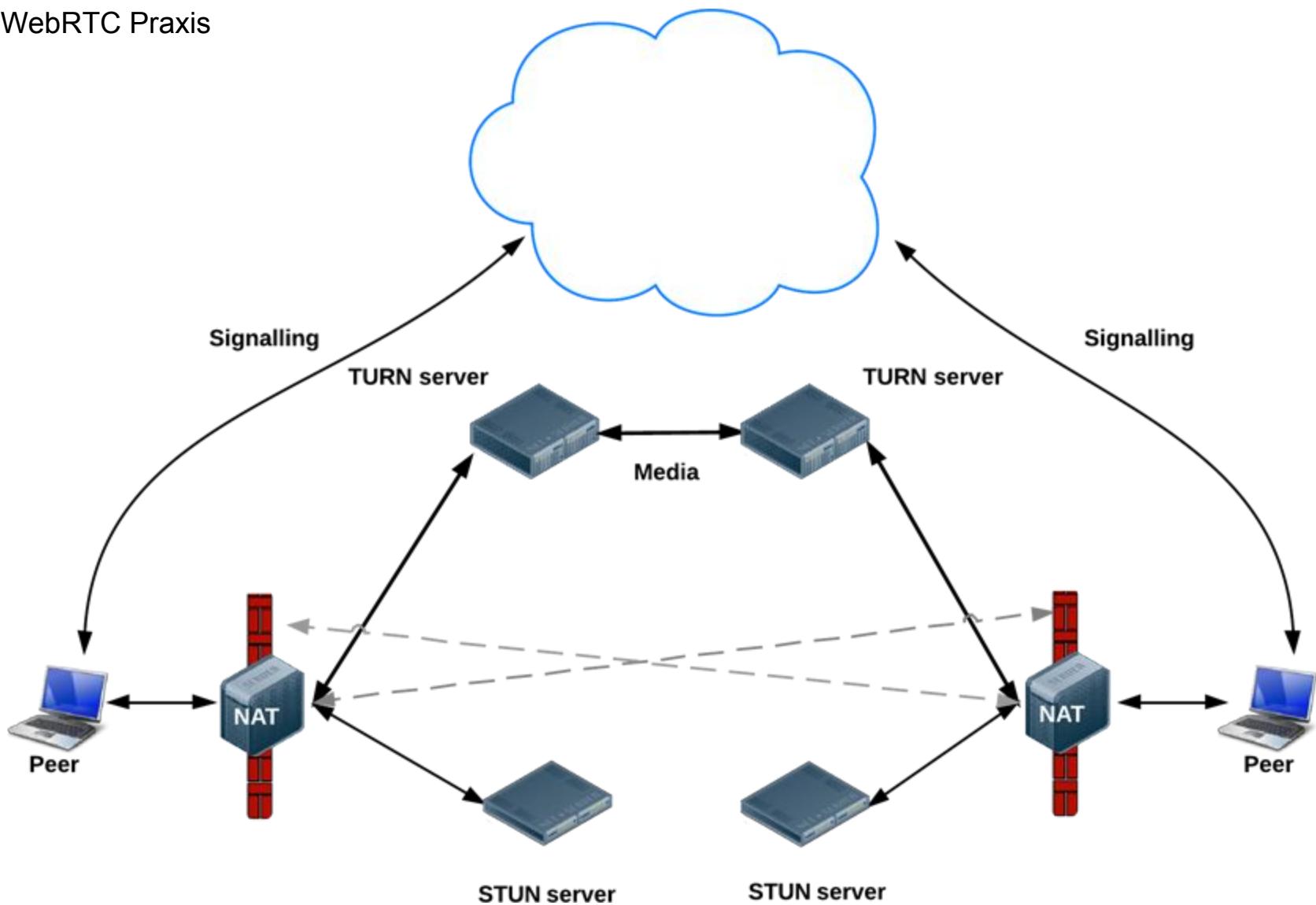


## WebRTC Theorie



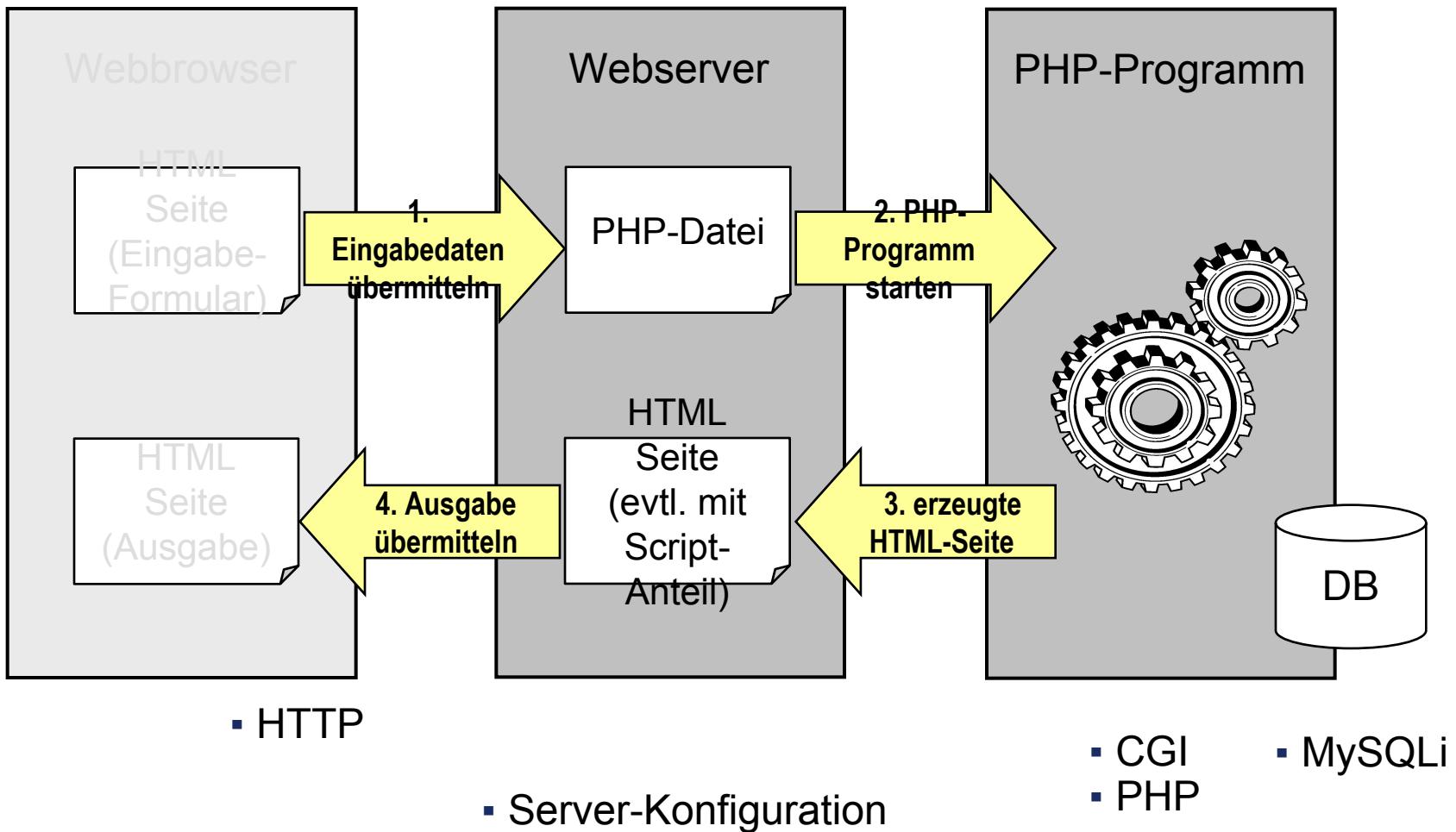
<http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/#after-signaling-using-ice-to-cope-with-nats-and-firewalls>

## WebRTC Praxis



### 3. Webserver

## Der Webserver



# Hochschule Darmstadt

## Fachbereich Informatik

### 3.1 Webserver Software



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

### 3.1 Webserver Software

## Marktanteil von Webservern



Developer	January 2015	Percent	February 2015	Percent	Change	Quelle:
Apache	348,460,753	39.74%	342,480,920	38.77%	-0.97	<a href="http://news.netcraft.com/">http://news.netcraft.com/</a>
Microsoft	241,276,347	27.52%	253,484,221	28.69%	1.18	
nginx	128,083,920	14.61%	130,093,899	14.73%	0.12	Web Server Survey: „Market Share for Active Sites“
Google	20,209,649	2.30%	20,238,057	2.29%	-0.01	

## Verfügbare Webserver (Auswahl)

- Apache2
  - ⇒ defacto-Standard im Web. OpenSource-Produkt und Freeware. für UNIX-Plattformen und für MS Windows/DOS verfügbar.

- Microsoft's Internet Information Server (IIS)
  - ⇒ Kommerzieller Webserver für Windows Server



- Google Web Server (GWS)
  - ⇒ Google betreibt damit ca. 10 Millionen eigene Websites, Blogs etc. GWS steht der Allgemeinheit nicht zur Verfügung

- nginx
  - ⇒ freier Webserver unter BSD-Lizenz
  - ⇒ kleiner und schlanker Webserver

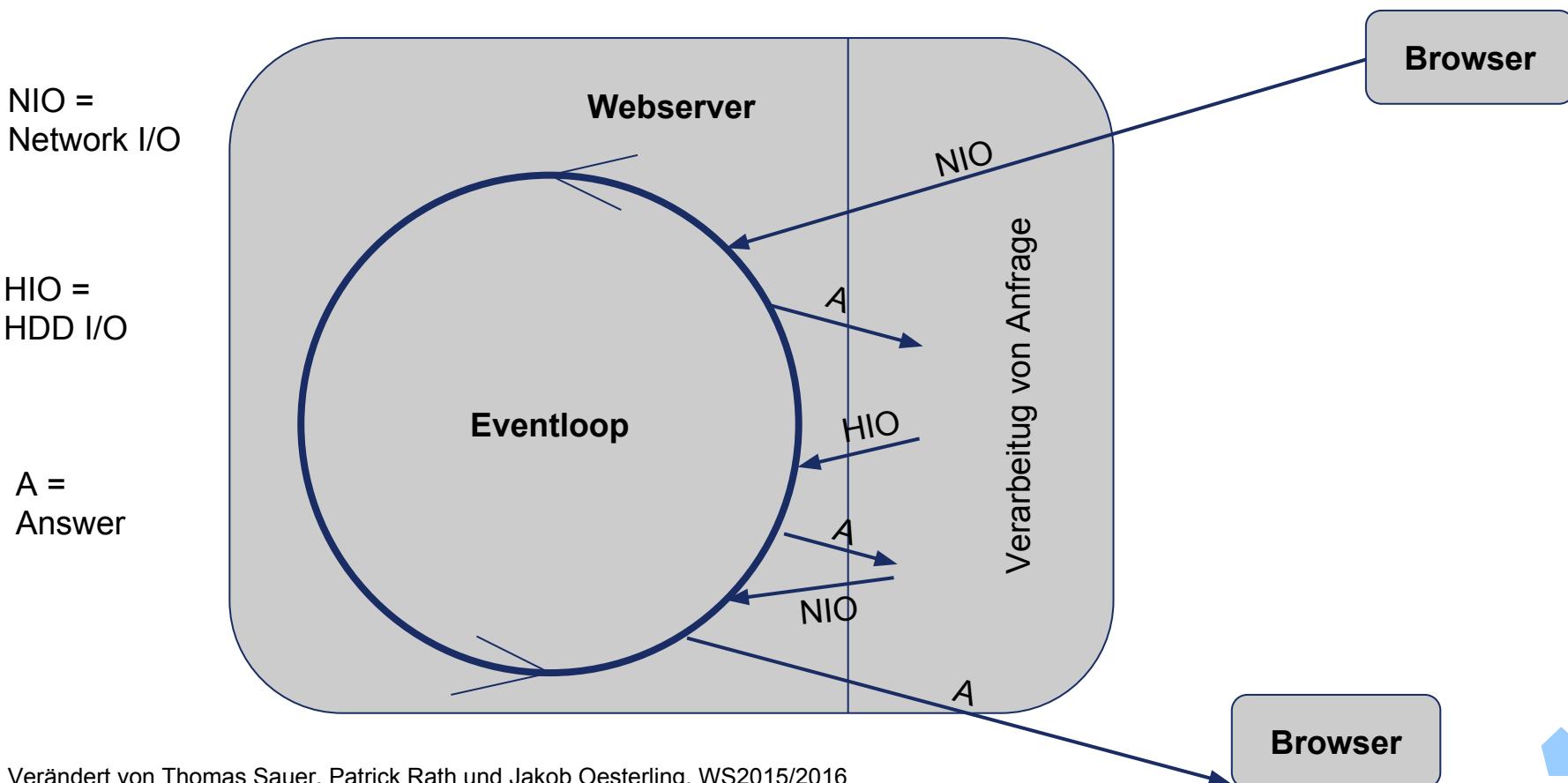


- lighttpd
  - ⇒ Webserver unter BSD-Lizenz mit Optimierung auf Massendaten
  - ⇒ eingesetzt z.B. bei YouTube oder SourceForge



## Skalierung/Performance von Webservern

- I/O von Festplatte/Netzwerk ist ein langsam bei Computern
- libevent, epoll, etc.** stellen eine Lösung parat indem sie *asynchrone event orientierte programmierung* ermöglichen (programmieren mit Callbacks & Eventloop)
- Dadurch kein warten auf I/O somit viele Verbindungen gleichzeitig möglich
- Zusätzlich Multithreading
- Geschrieben in low-level Sprachen wie C also ohne Garbage Collection



# Hochschule Darmstadt

## Fachbereich Informatik

### 3.1.1 Webserver: Installation und Konfiguration



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

### 3.1.1 Webserver: Installation und Konfiguration

## Allgemeines

- Konfiguration erfolgt je nach Webserver entweder GUI, 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.
- Typischerweise drei Benutzerrollen (relevant für Zugriffsrechte):
  - ⇒ Administrator
  - ⇒ Anbieter von Inhalten/Webseiten
  - ⇒ Besucher der Webseiten      *im Skript meist "User" bzw. "Benutzer" genannt*

## 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
  - ⇒ z.B. /usr/bin/perl unter Unix



## 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)

### 3.1.1 Webserver: Installation und Konfiguration

## 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
- weltweit am häufigsten eingesetzte Webserver
  - ⇒ <http://httpd.apache.org/>
  - ⇒ aktuelle Version (03.2015): Apache 2.4.12
- 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>



### 3.1.1 Webserver: Installation und Konfiguration

## Apache

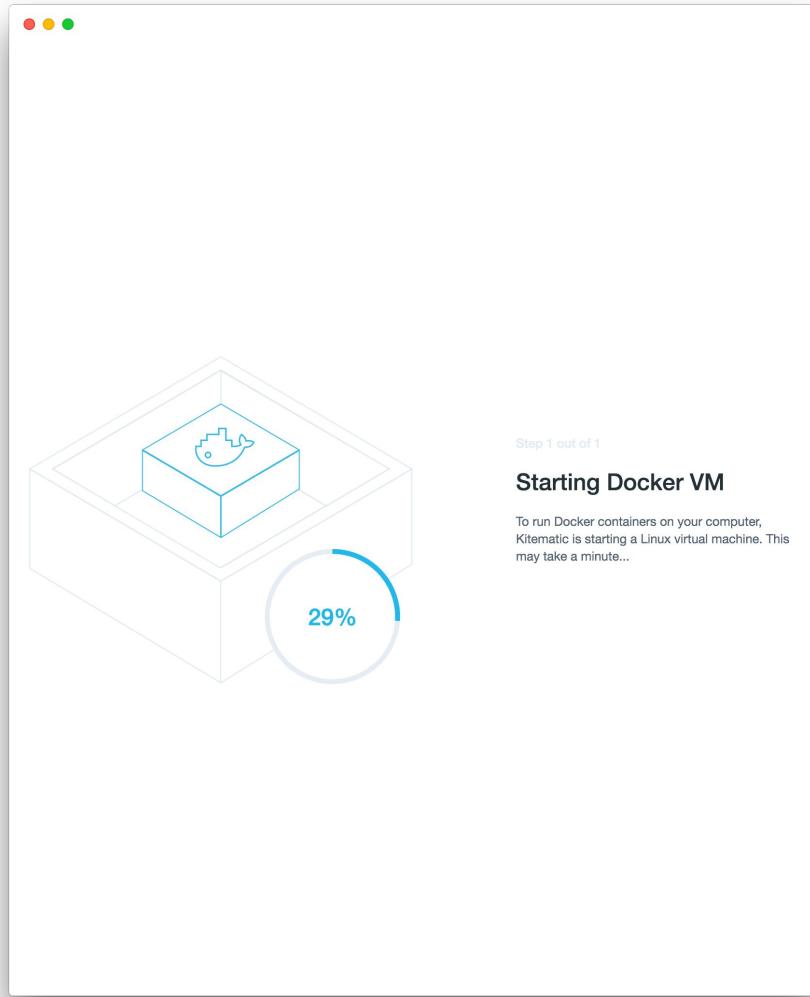


**ubuntu**   
Delivered by Canonical



### 3.1.1 Webserver: Installation und Konfiguration

## Apache



### 3.1.1 Webserver: Installation und Konfiguration

## Apache

A screenshot of the Docker Hub website. At the top, there's a search bar with the placeholder "Search image on Docker Hub" and a "CREATE" button. Below the search bar, there are three tabs: "All", "Recommended" (which is selected), and "My Repos".

The main area is titled "Containers" and shows a list of running containers with their names and images:

- mongodb (mongodb:latest)
- postgis (postgis:latest)
- rabbitmq (rabbitmq:latest)
- scipyserver (scipyserver:latest)

Below this, the "Recommended" section displays a grid of 15 popular Docker images, each with a thumbnail, name, description, star count, and a "CREATE" button:

- hello-world-nginx** by kitematic: A light-weight nginx container that demonstrates the features of Kitematic. (17 stars)
- ghost** by official: Ghost is a free and open source blogging platform written in JavaScript. (171 stars)
- jenkins** by official: Official Jenkins Docker image. (806 stars)
- redis** by official: Redis is an open source key-value store that functions as a data structure server. (1307 stars)
- rethinkdb** by RethinkDB: RethinkDB is an open-source, document database that makes it easy to build and scale real-time... (125 stars)
- minecraft** by kitematic: The Minecraft multiplayer server allows two or more players to play Minecraft together. (27 stars)
- solr** by official: Solr is the popular, blazing-fast, open source enterprise search platform built on Apache... (33 stars)
- elasticsearch** by official: Elasticsearch is a powerful open source search and analytics engine that makes data easy to... (540 stars)
- postgres** by official: The PostgreSQL object-relational database system provides reliability and data integrity. (1254 stars)
- ubuntu-upstart** by official: Upstart is an event-based replacement for the /sbin/init daemon which starts processes... (42 stars)
- memcached** by official: Free & open source, high-performance, distributed memory object caching system. (157 stars)
- rabbitmq** by official: RabbitMQ is a highly reliable enterprise messaging system based on the emerging AMQP... (290 stars)
- celery** by official: (2 stars)
- mysql** by official: (0 stars)

At the bottom of the page are links for "DOCKER CLI", "MESSAGING", and "SETTINGS".

## 3.1.1 Webserver: Installation und Konfiguration

# Apache

The screenshot shows two side-by-side Docker application windows. Both windows have a header with a red circle, yellow triangle, green square, a user icon, and the text 'theanonymous'.

**Left Window (Logs):**

- Containers:** mongoDB (mongod:latest) is selected, shown in blue. Other containers listed are postgres (postgres:latest), rabbitmq (rabbitmq:latest), and scipyserver (scipyserver:latest).
- Buttons:** START, RESTART, EXEC.
- Logs:** The container logs for mongoDB show extensive log output from July 2015, detailing memory usage, connection counts, and various system messages. Key lines include:
 

```
2015-07-21T15:53:29.334+0000 [clientcurlsrmon] mem (MB) res:37 virt:390
 ...
 2015-07-22T12:02:33.702+0000 [signalProcessingThread] shutdown: going to flush diaglog...
 2015-07-22T12:02:33.702+0000 [signalProcessingThread] shutdown: going to close listening sockets...
 2015-07-22T12:02:33.702+0000 [signalProcessingThread] shutdown: waiting for f_reallocator...
 2015-07-22T12:02:33.702+0000 [signalProcessingThread] shutdown: closing all files...
 2015-07-22T12:02:33.702+0000 [signalProcessingThread] closeAllFiles() finished
 2015-07-22T12:02:33.702+0000 [signalProcessingThread] shutdown: removing fs lock...
 2015-07-22T12:02:33.702+0000 [signalProcessingThread] dbexit: really exiting now
```
- Bottom Buttons:** DOCKER CLI, Chat, Settings.

**Right Window (Settings):**

- Containers:** mongoDB (mongod:latest) is selected.
- Buttons:** START, RESTART, EXEC.
- Tab Bar:** Home, Settings (selected).
- General Tab:**
  - Container Name:** mongoDB
  - Save Button:**
- Environment Variables Tab:**

KEY	VALUE
PATH	/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
AUTH	yes
- Delete Container:** DELETE CONTAINER button.

### 3.1.1 Webserver: Installation und Konfiguration

## Apache



VAGRANT



Puppet

Protobox





## Apache Grundeinstellungen Windows

- Konfig-Datei `httpd.conf` im Verzeichnis ...\\xampp\\apache\\conf
  - ⇒ Definition und Verwendung von Variablen  
`Define DOCROOT "C:/web"`  
Wurzelverzeichnis der Apache-Installation  
`ServerRoot "C:/xampp/apache"`
  - ⇒ Port, über den der Server kommuniziert (Standard)  
`Listen 80`
  - ⇒ eMail-Adresse des Administrators für Probleme  
`ServerAdmin na.me@h-da.de`
  - ⇒ Mapping URL ⇒ Verzeichnis für die Startseite (wenn der Besucher nur den Servernamen als URL eingibt, landet er in diesem Verzeichnis)  
`DocumentRoot "C:/xampp/apache/htdocs"` oder besser  
`DocumentRoot ${DOCROOT}` nach vorherigem `Define`
  - ⇒ Suchreihenfolge nach Dateinamen, falls URL ohne Dateinamen gegeben  
`DirectoryIndex index.html index.php`

Um die Wartung zu vereinfachen,  
sollten mehrfach auftretende Pfade  
als Variablen definiert werden!  
(DRY: Don't repeat yourself)

### 3.1.1 Webserver: Installation und Konfiguration

## Apache Grundeinstellungen Linux



### Konfig-Dateien unter /etc/apache2/

```
anwender@ubuntu:/var/log/apache2$ less /etc/apache2/
apache2.conf conf-enabled/ magic mods-enabled/
conf-available/ envvars mods-available/ ports.conf sites-available/
 ports.conf sites-enabled/
```

# Summary of how the Apache 2 configuration works in Debian:  
# The Apache 2 web server configuration in Debian is quite different to  
# upstream's suggested way to configure the web server. This is because Debian's  
# default Apache2 installation attempts to make adding and removing modules,  
# virtual hosts, and extra configuration directives as flexible as possible, in  
# order to make automating the changes and administering the server as easy as  
# possible.

# It is split into several files forming the configuration hierarchy outlined  
# below, all located in the /etc/apache2/ directory:

#  
# /etc/apache2/  
# |-- apache2.conf  
# | '-- ports.conf  
# '-- mods-enabled  
# |-- \*.load  
# '-- \*.conf  
# '-- conf-enabled  
# '-- \*.conf  
# '-- sites-enabled  
# '-- \*.conf  
#  
# \* apache2.conf is the main configuration file (this file). It puts the pieces  
# together by including all remaining configuration files when starting up the  
# web server.

Listen 80

```
<IfModule ssl_module>
 Listen 443
</IfModule>

<IfModule mod_gnutls.c>
 Listen 443
</IfModule>
```

## Apache Grundeinstellungen Linux



### ■ Konfig-Datei `/etc/apache2/apache2.conf`

- ⇒ Definition und Verwendung von Variablen

`Define DOCROOT "/web"`

Wurzelverzeichnis der Apache-Installation

`ServerRoot "/usr/sbin/apache"`

Um die Wartung zu vereinfachen,  
sollten mehrfach auftretende Pfade  
als Variablen definiert werden!  
(DRY: Don't repeat yourself)

- ⇒ Port, über den der Server kommuniziert (Standard)

`Include ports.conf`

- ⇒ eMail-Adresse des Administrators für Probleme

`ServerAdmin na.me@h-da.de`

- ⇒ Mapping URL ⇒ Verzeichnis für die Startseite (wenn der Besucher nur den Servernamen als URL eingibt, landet er in diesem Verzeichnis)

`DocumentRoot "/var/www/htdocs"` oder besser

`DocumentRoot ${DOCROOT}` nach vorherigem `Define`

- ⇒ Suchreihenfolge nach Dateinamen, falls URL ohne Dateinamen gegeben

`DirectoryIndex index.html index.php`

## Mapping für weitere Verzeichnisse



- Alias definiert die Abbildung URL  $\Rightarrow$  Verzeichnis
  - $\Rightarrow$  Dokumente können in anderen Verzeichnissen abgelegt werden als mit DocumentRoot festgelegt wurde
  - $\Rightarrow$  Beim Zugriff auf ein Alias über eine URL wird Groß-Kleinschreibung unterschieden - auch unter Windows
  - $\Rightarrow$  `Define MANUAL "C:/Dokumentation"`  
`Alias /manual ${MANUAL}`
- ScriptAlias definiert für Server-Skripte die Abbildung URL  $\Rightarrow$  Verzeichnis
  - $\Rightarrow$  d.h. für Dateien, die nicht zum Client gesendet sondern im Server ausgeführt werden
  - $\Rightarrow$  `Define PHP "C:/php"`  
`ScriptAlias /php/ ${PHP}/`



Der / am Ende bewirkt, dass auch beim Zugriff in der URL ein / angegeben werden muss!

## Mapping für Anbieter-Verzeichnisse



- Gliederung der Dokumente nach Anbietern
    - ⇒ weitere Variante zur Definition der Abbildung URL ⇒ Verzeichnis
    - ⇒ z.B. für persönliche Homepages von Dozenten auf www.fbi.h-da.de
  - Aufruf der Startseite mit ~Anbietername
    - ⇒ z.B. http://www.fbi.h-da.de/~r.hahn
- UserDir bezieht sich  
nicht auf die Besucher  
der Webseite!
- ```
<IfModule mod_userdir.c>
    UserDir "C:/Anbieterverzeichnisse/"
</IfModule>
```
- Anbieter-Verzeichnisse werden i.a. von den Anbietern selbst gepflegt
 - ⇒ eventuell mit eigenen Berechtigungs-Dateien (.htaccess)
 - ⇒ Zugriffsrechte gut überlegen und steuern mit **AllowOverride**

Optionen für Verzeichnisse



- Einstellungen für Verzeichnisse werden innerhalb einer `<Directory>`-Anweisung gesetzt
 - ⇒ CGI-Skripte ausführen
 - `ExecCGI`
 - ⇒ Verknüpfungen zu anderen Verzeichnissen folgen
 - `FollowSymLinks` `SymLinksIfOwnerMatch`
 - ⇒ Inhaltsverzeichnis zeigen, wenn Indexdatei (z.B. `index.html`) fehlt
 - `Indexes`
 - ⇒ Standardeinstellung setzen
 - `All`
 - ⇒ Beispiel:

```
Define EWA "C:/Links/ewa"
```

```
Alias /ewa ${EWA}
```

```
<Directory ${EWA}>
```

```
    Options ExecCGI Indexes FollowSymLinks
```

```
</Directory>
```

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

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
 - ⇒ Mozilla verwendet den übermittelten MIME-Type,
Internet Explorer verwendet die Datei-Endung

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`



```
User-agent: googlebot          # all Google services
Disallow: /private/             # disallow this directory

User-agent: googlebot-news      # only the news service
Disallow: /                      # disallow everything

User-agent: *                    # any robot
Disallow: /something/           # disallow this directory
```



Hochschule Darmstadt

Fachbereich Informatik

3.1.2 Webserver: Zugriffsschutz und Sicherheit



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Zugriffsschutz allgemein



- Ein Webserver ermöglicht den Zugriff auf Dateien im Filesystem
 - ⇒ es muss festgelegt werden, wer auf welche Verzeichnisse zugreifen darf
 - z.B. nur bestimmte IPs, bestimmte Domains usw.
- Fehler bei der Konfiguration sind kritisch.
Deshalb werden die Rechte in 2 Stufen definiert:
 1. Zentraler Zugriffsschutz
 - Grundlegende Rechte werden innerhalb der Apache-Konfiguration (httpd.conf) festgelegt.
 - Diese Datei kann nur von einem Administrator bearbeitet werden
 - Änderungen an dieser Datei erfordern einen Neustart des Apache
 2. Lokaler Zugriffsschutz (z.B. für Anbieterverzeichnisse)
 - Diese Rechte können von den Anbietern der Inhalte vergeben werden
 - Änderungen erfordern keinen Neustart des Webservers
 - In der httpd.conf muss aktiviert werden, dass der Webserver in Verzeichnissen nach einer bestimmten Datei (.htaccess) sucht und deren Inhalt als Zugriffsregeln interpretiert



Zentraler Zugriffsschutz in httpd.conf

- ## ■ Standardeinstellung sehr restriktiv

⇒ <Directory />

Unix root

AllowOverride None .htaccess wirkungslos

Require all denied niemand hat Zugriff

</Directory>

alles andere
muss dann
explizit
freigegeben
werden

- (Dokumenten-)Verzeichnisse gezielt öffnen

```
⇒ <Directory "C:/Programme/Apache/htdocs">
```

`AllowOverride All` .htaccess wird akzeptiert

Require all granted alle haben Zugriff

</Directory>

Verzeichnis-
bezogener
Schutz wird an
Unter-
verzeichnisse
rekursiv weiter
gegeben

- Dateiname für verzeichnisspezifischen Schutz festlegen

⇒ AccessFileName .htaccess Name festlegen

```
<Files ".ht*">
```

Require all denied und Zugriff über Browser sperren

</Files>

Lokaler Zugriffsschutz mit Berechtigungsdateien



- Der Webserver soll im Verzeichnis C:\myDir eine Datei .htaccess mit lokalen Zugriffsberechtigungen akzeptieren

- ⇒ Voraussetzung (in der httpd.conf)

- Der Name der Berechtigungsdatei entspricht dem festgelegten Namen (üblicherweise .htaccess):

`AccessFileName .htaccess`

- Die Verwendung von lokalen Berechtigungsdateien ist (für das betroffenen Verzeichnis) erlaubt

```
<Directory "C:\myDir">
    AllowOverride All
    Require all granted
</Directory>
```

Großzügige Freigaben
sollten lokal
eingeschränkt werden!

- ⇒ Die Berechtigungsdatei bezieht sich auf das Verzeichnis, in dem sie steht
 - darin werden Bedingungen formuliert, die für den Zugriff erfüllt sein müssen
 - z.B. nur lokaler Zugriff durch `Require local`

Zugriffsschutz mit Require

Require ersetzt deny/allow!



- Innerhalb der .htaccess-Dateien oder auch in Directory-Containern der httpd.conf können Zugriffsregeln definiert werden:
 - ⇒ Einzelne Regeln mit **Require**, z.B.
 - **Require all granted** allen den Zugriff erlauben
 - **Require all denied** allen den Zugriff verweigern
 - **Require local** alle lokalen Zugriffe zulassen (IPv4, IPv6,...)
 - **Require not host .de** Zugriffe erlauben, die nicht von *.de kommen
 - **Require ip 192.168.2** Zugriffe von IP 192.168.2.* erlauben
 - ⇒ Gruppen von Regeln mit
 - **<RequireAny>** – eine der Bedingungen muss erfüllt sein
 - **<RequireAll>** – alle aufgeführte Bedingungen müssen erfüllt sein
 - **<RequireNone>** – keine der Regeln darf erfüllt sein
 - diese Anweisungsblöcke können auch verschachtelt werden

Es gibt noch viele weitere Konfigurationsmöglichkeiten

http://httpd.apache.org/docs/current/mod/mod_authz_core.html

Zugriffsschutz mit lokaler Passworddatei



- Authentisierung
 - ⇒ Neben der Anbindung an Verzeichnisdienste (z.B. LDAP) unterstützt Require auch lokale Passworddateien
 - ⇒ ein Anbieter kann so seine eigene Zugangsliste mit Accounts verwenden
- Verwendung einer lokalen Passworddatei für ein Verzeichnis
 - ⇒ .htaccess (in dem Verzeichnis)

```
AuthType Basic  
AuthName "Text für Authentisierungs-Popup"  
AuthUserFile c:\pw.sec  
Require valid-user
```

- ⇒ Passwort-Datei pw.sec an sicherem Ort ablegen!
Erzeugen mit: htpasswd.exe -c pw.sec Besucher

```
hahn:$apr1$dFJ6Tos3$3A3C11djFtCH3A7.cVDVDO  
studi:$apr1$sWyY6auL$N.1h9cvH/fo7YdUMHcxX0/
```



Zugriffsschutz - Beispiele



- a) "alle" - offen für die ganze Welt

Require all granted

- b) "niemand" – im Web nicht
verfügbar

Require all denied

- c) "alle außer..." Rechner von
bestimmten Domains

```
<RequireAll>
  Require not host .to
  Require not host .tv
</RequireAll>
```

- d) "niemand, außer..."

bestimmten IP-Adressen

< RequireAny>

Require ip 141.100

Require ip 192.168

</RequireAny>

- e) "niemand, außer mit Passwort"

AuthType Basic

AuthName "Anzeigetext"

AuthUserFile c:\pw.sec

Require valid-user

Sicherheit von Apache-Passwörtern



- Basic Authentication wird quasi im Klartext (base64-kodiert) übertragen und die HTML-Antwort ebenfalls
 - ⇒ etwas besser: MD5 Digest; noch besser: verschlüsselte Verbindung
- 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
 - ⇒ Problem unter Unix: verschiedene User (Anbieter) gegeneinander abschotten



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 (oft inkonsistenten) Syntax
 - bei Fehlern in der Konfiguration lässt sich Apache nicht mehr starten!
 - Pfade brauchen teilweise am Ende einen "/"
 - Manchmal wird "/" verwendet, manchmal "\\"
 - unbedingt die Beispiele in den Kommentaren als Vorlage beachten !
 - ⇒ beim Zugriff auf ein Verzeichnis über ein Alias wird die richtige Schreibweise (für den Alias) sogar unter Windows erzwungen
- Windows ignoriert Groß-Kleinschreibung in Pfaden und Dateinamen
 - ⇒ Skripte und Links werden unter Windows auch gefunden, wenn die Groß-Kleinschreibung der Dateinamen falsch ist
 - ⇒ beim Verschieben eines Projekts auf einen Web Server unter Unix werden diese Dateien nicht mehr gefunden

Test auf Fehler in httpd.conf:
apache\bin\httpd.exe -t

Zusammenfassung

■ Grundlagen

- ⇒ Was ist ein Webserver?
- ⇒ Webserver am Markt
- ⇒ Wie sieht HTTP aus

■ Grundeinstellungen

- ⇒ IP-Adresse & Ports, Log-Dateien
- ⇒ Verzeichnisse für Dokumente, Skripte, Anbieter
- ⇒ Besondere Dateinamen (index.html, .htaccess,...)
- ⇒ Pfade zu Skripten und ausführbaren Programmen (z.B. Perl)
- ⇒ MIME-Typen und Dateiendungen

■ Apache

- ⇒ Grundeinstellungen
- ⇒ Zugriffsberechtigungen und Zugriffsschutz



Hochschule Darmstadt

Fachbereich Informatik

3.2 CGI



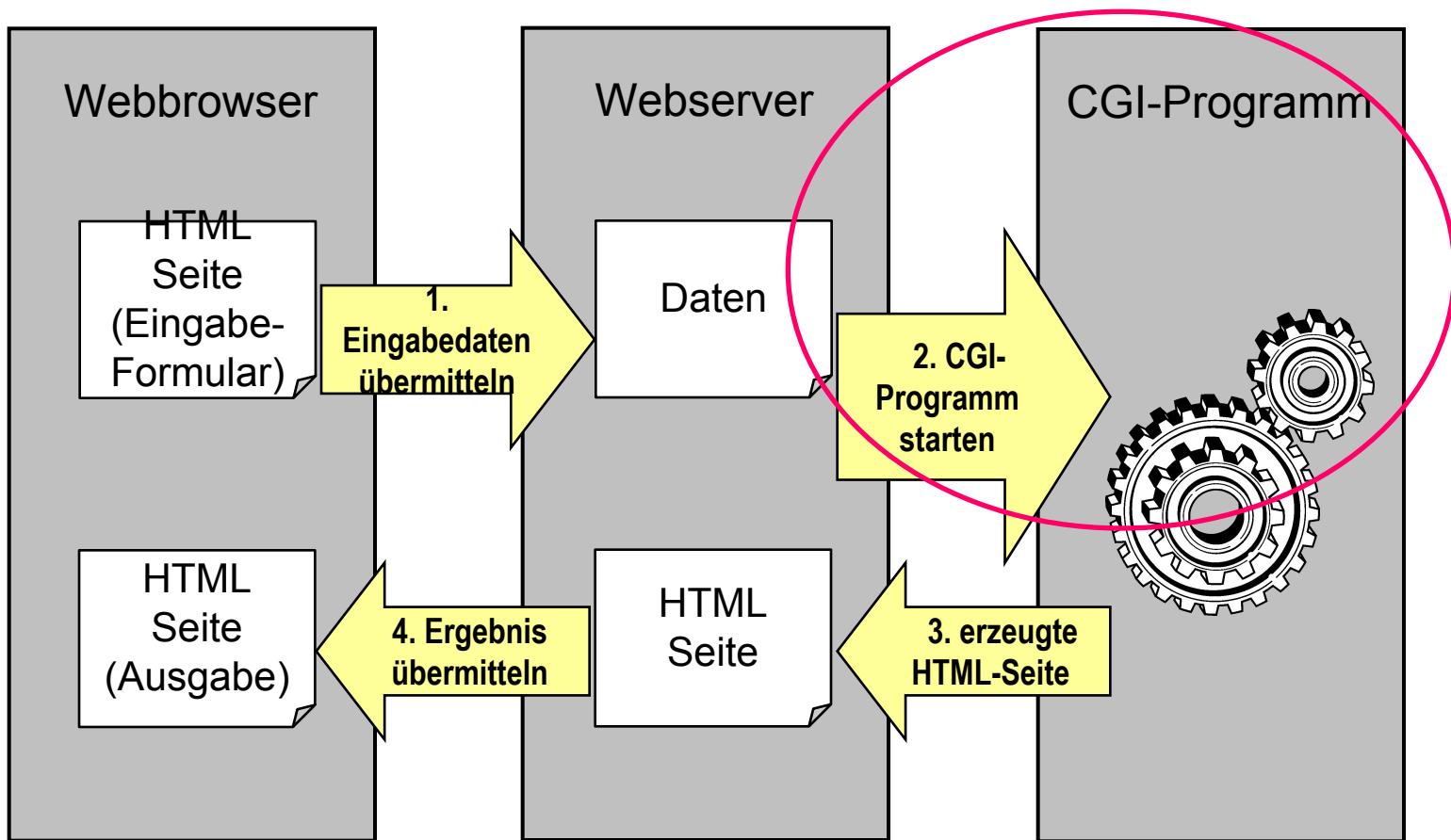
h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

CGI - Common Gateway Interface



CGI - Common Gateway Interface

Es gibt noch alternative Kommunikationswege!
Wir beschränken uns auf CGI !

- 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 / unterstützt
- 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 .../cgi-bin/printenv.pl
(In Standard-Installation enthalten)



- Werte werden bei einem CGI-Aufruf vom Webserver gesetzt

- ⇒ Informationen zum Request

REQUEST_URI=/cgi-bin/CgiTest.exe?Name=x&Kommentar=y

SCRIPT_NAME=/cgi-bin/CgiTest.exe

HTTP_REFERER=http://localhost/CgiTestFormular.htm

HTTP_COOKIE=Keksname=Kruemel; nochEinKeks=hart

- ⇒ und je nach Methode

REQUEST_METHOD=GET

sofern gesetzt

QUERY_STRING=Name=x&Kommentar=y

- ⇒ oder

REQUEST_METHOD=POST

Auf das Ende achten!

CONTENT_LENGTH=28

CONTENT_TYPE=application/x-www-form-urlencoded

QUERY_STRING= (ist leer)

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

Umgebungsvariablen (2)

- ⇒ 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
```

Umgebungsvariablen (3)

- ⇒ Informationen über den Client (Browser)

HTTP_ACCEPT=image/gif, 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

Es können auch noch die installierten Fonts und Browser Plugins abgefragt werden!
Diese Daten machen Ihren Rechner in der Regel eindeutig erkennbar
vgl. <https://panopticlick.eff.org/>

Codierung von Parametern in einer URL (I)

der Browser macht das automatisch
in PHP: urlencode()
in ECMAScript: encodeURI()

- 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
 - ⇒ Achtung: ``
- 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 Parametern in einer URL (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**
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
 - als Bilddaten im GIF oder JPEG-Format

Es gibt kein
Daten-Ende-
Zeichen



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>" << endl;
    cout << "<html lang=\"de\">" << 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";
}
```

Regex-Syntax
gewöhnungs-
bedürftig...

...aber sehr
mächtig



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-cgi.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: `Los`

Anwendung: Statistik-Abfragen

CGI-Scripts, die keinen Input vom Anwender benötigen

- ⇒ **Grafikreferenz:**

Beispiel: ``

Anwendung: grafische Zugriffszähler

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:
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

Hochschule Darmstadt

Fachbereich Informatik

3.3 PHP



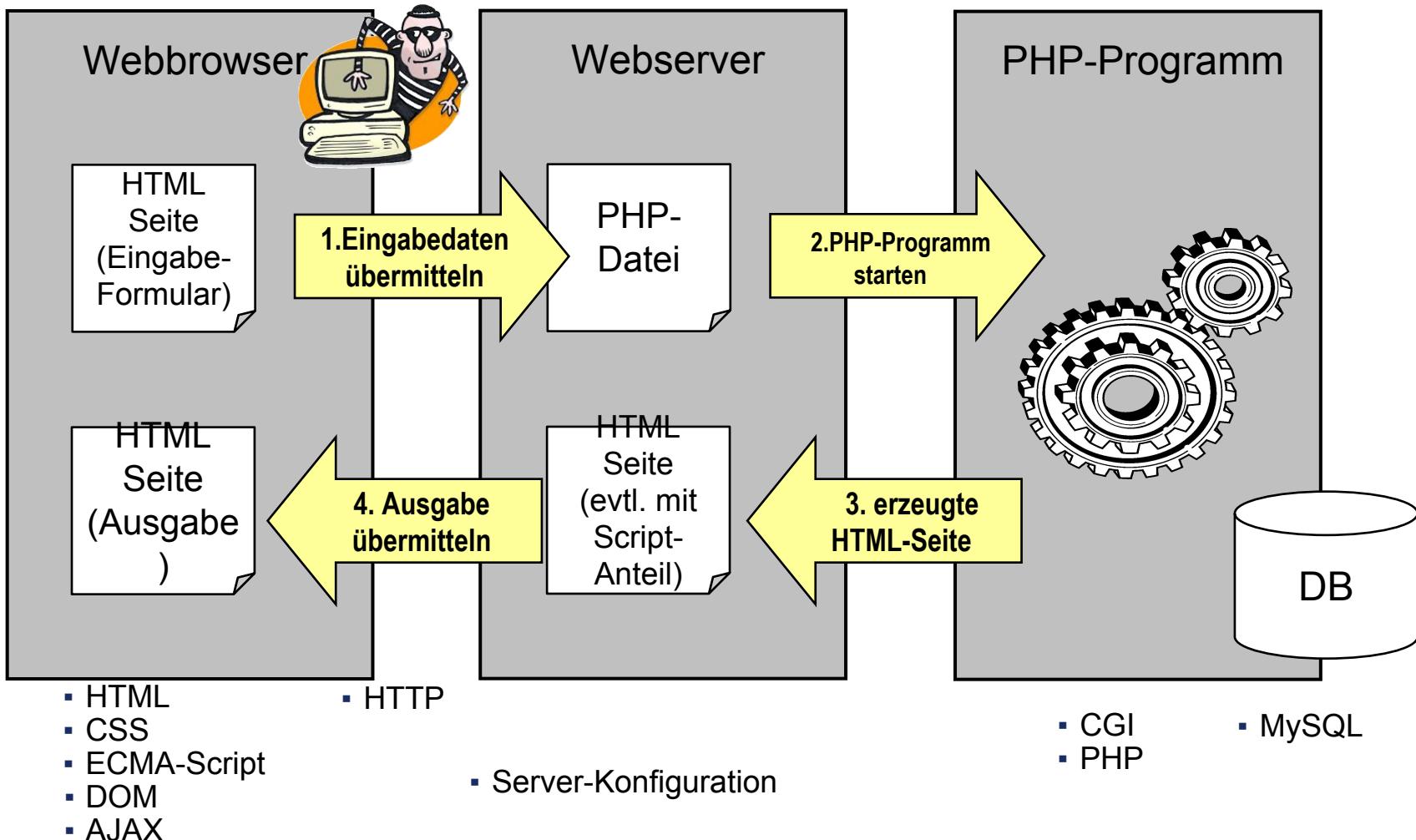
h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Einsatz der Technologien im Zusammenhang



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 scheinen 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 ECMAScript)
- ⇒ 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

diese ursprüngliche Idee ist längst veraltet

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
 - ⇒ Interpilier wie Perl 5
 - ⇒ Performancegewinn von Faktor 2-5x im Einzelfall 100x.
 - ⇒ neuer, schnellerer Sprachkern "Zend"
 - ⇒ viele, neue Funktionen (mehrere Tausend)
 - ⇒ echte Komposition/Assoziation von Objekten immer noch nicht unterstützt
- 2003: PHP 5.0
 - ⇒ diverse Optimierungen
 - ⇒ endlich richtige **Objektorientierung**
- 2009: PHP 5.2.8 in der XAMPP-Suite enthalten (kein PHP 4 mehr!)
- 2015 PHP 7 (Voraussichtlich November)



Open Source

Hochschule Darmstadt

Fachbereich Informatik

3.3.1 PHP Grundlagen



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Primitives Beispiel

```
<?php
    header ("Content-type: text/html");
?>
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8" />
    <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
 - ⇒ "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
 - `<? ... ?>`
 - `<?= $Variable ?>`
 - `<% ... %>`
 - `<script language="php"> ... </script>`
- 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++
- dynamisch typisiert ähnlich JavaScript
 - ⇒ Funktionsdeklaration mit **function**
 - ⇒ keine Variablen-deklaration
 - ⇒ 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,...

3.3.1 PHP Grundlagen

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 und Sonderzeichen 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];`
- \0 ist (im Gegensatz zu C++) keine Endemarke, ausgenommen jedoch Funktionen, die als "nicht binary safe" markiert sind

Der Zugriff über
`$MeinString{$Pos};`
ist deprecated!

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`
 - ⇒ `$map["abc"] = 0.05; // Zugriff per Schlüssel`
 - ⇒ `$oFarbe = array('Erdbeere'=>'rot', 'Banane'=>'gelb');`
// Zuweisung als Tupel (Item, Value): \$oFarbe['Erdbeere'] = 'rot'
- Abarbeitung indiziert:
 - ⇒ `for ($i=0; $i<count($arr); $i++)`
`echo ($arr[$i]);`
- Abarbeitung als Kollektion:
 - ⇒ `foreach($oFarbe as $obst => $farbe)`
`echo (" $obst hat $farbe \n");`



Beispiel: Umgebungsvariablen mit PHP

```
<?php header ("Content-type: text/html"); ?>
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8" />
        <title>Umgebungsvariablen</title>
    </head>
    <body>
        <h2>Umgebungsvariablen</h2>
        <pre>
<?php
foreach($_ENV as $key => $value) {
    echo "$key=$value\n";
}
?>
        </pre>
    </body>
</html>
```

\$_ENV erfordert in PHP.ini:
variables_order = "EGPCS"

3.3.1 PHP Grundlagen

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)

"Super-Globals" erfordern
keine global-Deklaration

Zugriff auf Formulardaten

■ PHP stellt globale assoziative Arrays bereit

- ⇒ Name des Formularelements dient als Index

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

PHP hat den
QUERY_STRING
bereits dekodiert



■ PHP bildet bis Version 5.4 (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

- ⇒ vor PHP 5.4 in php.ini: `register_globals = Off`

■ vor PHP 5.4: `magic_quotes_gpc = Off` (sonst wird " zu \'")

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

3.3.1 PHP Grundlagen

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");  
?>
```

oder gleich \$_REQUEST verwenden

Strukturierung und Wiederverwendung (veraltet)

kleiner Nachteil:
PHP muss immer
mehrere Dateien öffnen

- 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 einbinden
`require_once "Pfadname.php";`
 - ⇒ vergleichbar mit #include in C++
 - ⇒ **`require_once`** und **`include_once`** vermeiden Endlos-Einbindungen
 - ⇒ 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 !
- Variablen sind "require-übergreifend" sichtbar

3.3.1 PHP Grundlagen

Konfiguration von PHP

- **php.ini** ist zentrale Konfigurationsdatei
- seit PHP 5.4 ohnehin abgeschafft:
 - ⇒ **register_globals = Off**
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!



PHP Version 5.5.6	
System	Windows NT RAHADA 6.1 build 7601 (Windows 7 Business Edition Service Pack 1) i586
Build Date	Nov 12 2013 11:29:52
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x86
Configure Command	cscript /nologo configure.js --enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\ sdk\shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\ sdk\shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\ sdk\shared" "--enable-object-out-dir=../obj" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"

Apache für PHP konfigurieren



■ PHP unter Verwendung des CGI

- ⇒ PHP-Installationsverzeichnis zum Skript-Verzeichnis erklären:
ScriptAlias /php/ "/apache/php/"
- ⇒ PHP-Interpreter (CGI-Version) für Dateien mit MIME-Type PHP aufrufen:
Action application/x-httpd-php /php/php-cgi.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-Interpreter wird für jeden Request erneut geladen

PHP-Dateien können in jedem Dokument-Verzeichnis liegen

■ effizient: PHP als Apache-Module

- ⇒ wird eingebunden, wenn Apache compiliert wird oder als Dynamic Shared Object (vgl. DLL unter Windows) beim Start
- ⇒ In der Installation von XAMPP ist PHP5 integriert

PHP-Interpreter wird einmalig bei Apache-Start geladen

Parallele Ausführung von Skripten

Im Internet ist das ganz normal ...

- mehrere Aufrufe desselben Skripts können gleichzeitig ausgeführt werden (d.h. quasi-parallel, nebenläufig, in verschiedenen Threads oder Prozessen)
 - ⇒ Sequenz (im Prinzip): Client1: \$no = get_no_of_entries()
Client2: delete entry[0]
Client1: for (i=0 to \$no-1) entry[i]=...
 - ⇒ Threads müssen sorgfältig programmiert sein, weil sie sich Speicher teilen; Prozesse kosten mehr Speicher und Rechenzeit, sind dafür sicherer
- Skript muss reentrant sein (kein Schreib-/Lesezugriff auf globalen Speicher) – was bedeutet das in PHP ?
 - ⇒ Apache erzeugte ursprünglich für jeden Request einen eigenen Prozess
 - kann seit 2.0 auch Multi-Threading; das wird für PHP nicht empfohlen!
 - ⇒ die PHP Laufzeitumgebung stellt jedem Request einen unabhängigen Satz von statischen und globalen Variablen zur Verfügung
 - damit wird auch bei threadbasierter Ausführung ein eigener Prozess simuliert
 - ⇒ aber: manche PHP-Bibliotheken nutzen eigenen globalen Speicher und sind daher nicht für threadbasierter Ausführung geeignet (meist nicht dokumentiert)
- Verwendung einer DB Storage Engine, die Transaktionen und Locking unterstützt (z.B. InnoDB oder BerkeleyDB, aber nicht MyISAM)
 - ⇒ Vorsicht: DB-Optimierungen schalten oft die Serialisierung ab

Hochschule Darmstadt

Fachbereich Informatik

3.3.2 PHP Entwicklung



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

PHP Entwicklung

- Eigene Funktionen definieren mit *function*

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

⇒ Optionale Funktions-Parameter mit Default-Wert:

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

⇒ Call by Reference

```
function change(&$myReference)
```

- Während der Entwicklung: alle Fehlercodes ausgeben mit *error_reporting(E_ALL)*;

SELFPHP

<http://www.selfphp.info>

Doku

<http://php.net>

Im Produktivbetrieb die Fehlermeldungen in Log-Datei schreiben und nicht ausliefern!

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)

```
$myvar = 'Variable';           " als &quot; oder \"
echo "{$myvar}n werden ausgewertet\n";
```

```
//Ausgabe: Variablen werden ausgewertet [newline]
```

komplexe
Berechnungen
und
Abgrenzung
von Variablen-
namen in {}

- Heredoc Notation

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

```
echo <<<EOT      kein Semikolon!
```

```
{$myvar}n werden ausgewertet
```

```
EOT;
```

```
//Ausgabe: Variablen werden ausgewertet [newline]
```

" als "

Schluss"tag" muss in der
ersten Spalte stehen
- und ganz alleine!

Strings und Zeichenkodierung in PHP

- PHP 5 Strings sind single byte strings (1 Byte pro Zeichen)
 - ⇒ passt perfekt zu ISO-8859-x Zeichensätzen
 - ⇒ aber UTF-8 stellt manche Zeichen mit 2 bis 4 Bytes dar
 - ⇒ für PHP 6 wird Unicode-Unterstützung diskutiert
- manche Stringfunktionen arbeiten daher nicht richtig für Umlaute
 - ⇒ `strlen` und `strpos` zählen falsch
 - ⇒ aber `str_replace ($search, $replace, $subject)` funktioniert
- Ausweg: PHP 5 Extension `mbstring`
 - ⇒ kein default, muss explizit installiert werden (in XAMPP enthalten)
 - ⇒ bietet Ersatz für die gängigen Stringfunktionen
 - ⇒ Funktionen erkennbar am Präfix `mb_`
 - ⇒ Initialisierung: `mb_internal_encoding("UTF-8");`
- andere PHP Funktionen, die mit Strings arbeiten, sorgfältig testen !

Metazeichen in Strings

hier sind nicht Umlaute gemeint,
sondern Zeichen, die in der Syntax
von HTML / ECMAScript / SQL eine
besondere Bedeutung haben

- im erzeugten HTML bzw. den ECMAScript 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": `"`
- 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

3.3.2 PHP Entwicklung

Metazeichen in Strings - sprachübergreifend

- übermittelte Formulardaten werden Strings für

- ⇒ HTML zum Anzeigen (ersetzt & < > '')

```
$Wert = htmlspecialchars($Wert);
```

- ⇒ SQL für Anfragen etc. (ersetzt '')

```
$Wert = $mysqli->real_escape_string($Wert);
```

- Erzeugen von JavaScript in HTML aus PHP-echo:

PHP ↓

Browser ↓

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

- Einfacher mit heredoc-Notation:

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

\$myvar = <<<LABEL ... LABEL;
liest den umschlossenen Text in die
Variable \$myvar !

3.3.2 PHP Entwicklung

Übergabe mit Mehrfachauswahl – Das Problem



```
<form action="21_FormularEcho.php" method="get">
  <p>Multi-Select:</p>
  <p><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>
```

Multi-Select:

Nr.1
Nr.2
Nr.3
Nr.4

absenden mit GET

ergibt für GET: <http://localhost/..../Echo.php?myselect=2&myselect=4>

überträgt (URL): myselect=2&myselect=4

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

PHP

mit POST sieht man nichts,
aber das Problem bleibt!

⇒ Es wird nur der letzte Wert als Umgebungsvariable übernommen!

3.3.2 PHP Entwicklung

Übergabe mit Mehrfachauswahl – Die Lösung



```
<form action="21_FormularEcho.php" method="get">


Multi-Select:


<!-- ohne [] steht nur das letzte Select in PHP zur Verfügung --&gt;
<p><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>


<input type="submit" value="absenden" /> mit GET


</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"]) &&
 is_numeric ($_POST["ElemA"]))
 $Datensatz["ElemA"] = $_POST["ElemA"];
 Speichern ($Datensatz);`

Syntax und
Wertebereich
überprüfen

3.3.2 PHP Entwicklung

Überprüfung auf unerwünschte Zeichen

- am einfachsten mit regulären Ausdrücken
 - ⇒ `bool preg_match (string pattern, string subject);`

Beispiele

ereg vermeiden; ist nicht binary safe und deprecated

- 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 _

Objektorientierung (I)

brauchbar seit PHP 5

- Klassen und Objekte ähnlich Java
 - ⇒ Objekte werden ausschließlich mit `new` erzeugt und über Referenzen angesprochen
 - ⇒ Attribute deklarieren mit: `var $Attribut;`
 - besser noch: `public / protected / private $Attribut;`
 - ⇒ Basisklassenkonstruktor explizit aufrufen
 - `parent::` bezeichnet Basisklasse
 - `$this` verweist auf Objekt
 - > für Zugriff auf Attribute (ohne `$`) und Methoden
- Vererbung
 - ⇒ `class a extends b { ... }` // aber keine Mehrfachvererbung
 - ⇒ dafür `interface i { ... }` und `class c implements i { ... }`
- wenig genutzt in älteren Bibliotheken
 - ⇒ Bibliotheken bestehen oft noch aus Gruppen von Funktionen
 - ⇒ OO-Beispiel: herstellerunabhängige Kapselung des Datenbankzugriffs in PHP Data Objects PDO

Objektorientierung (II)

■ Neue PHP-Sprachelemente im OO-Umfeld:

- ⇒ `public, protected, private` statt `var` //keine privaten Klassen
- ⇒ `static $x = 0; const y = 0;`
- ⇒ `$instance = new SimpleClass();` // weist Referenz zu
- ⇒ `$Kopie = clone $instance;`
- ⇒ `class a extends b { ... }`
- ⇒ `function f1 (a $param) { ... }`
- ⇒ `abstract class ...`, `abstract function ...`
- ⇒ `interface i { ... }` `class c implements i { ... }`
- ⇒ `final function f2 { ... }` // wie Java; d.h. nicht virtual
- ⇒ magic methods mit dem Namen `__xxx` werden implizit aufgerufen
`__construct, __destruct, __clone`
- ⇒ Vergleich: `==` gleiche Attributwerte `==` selbes Objekt

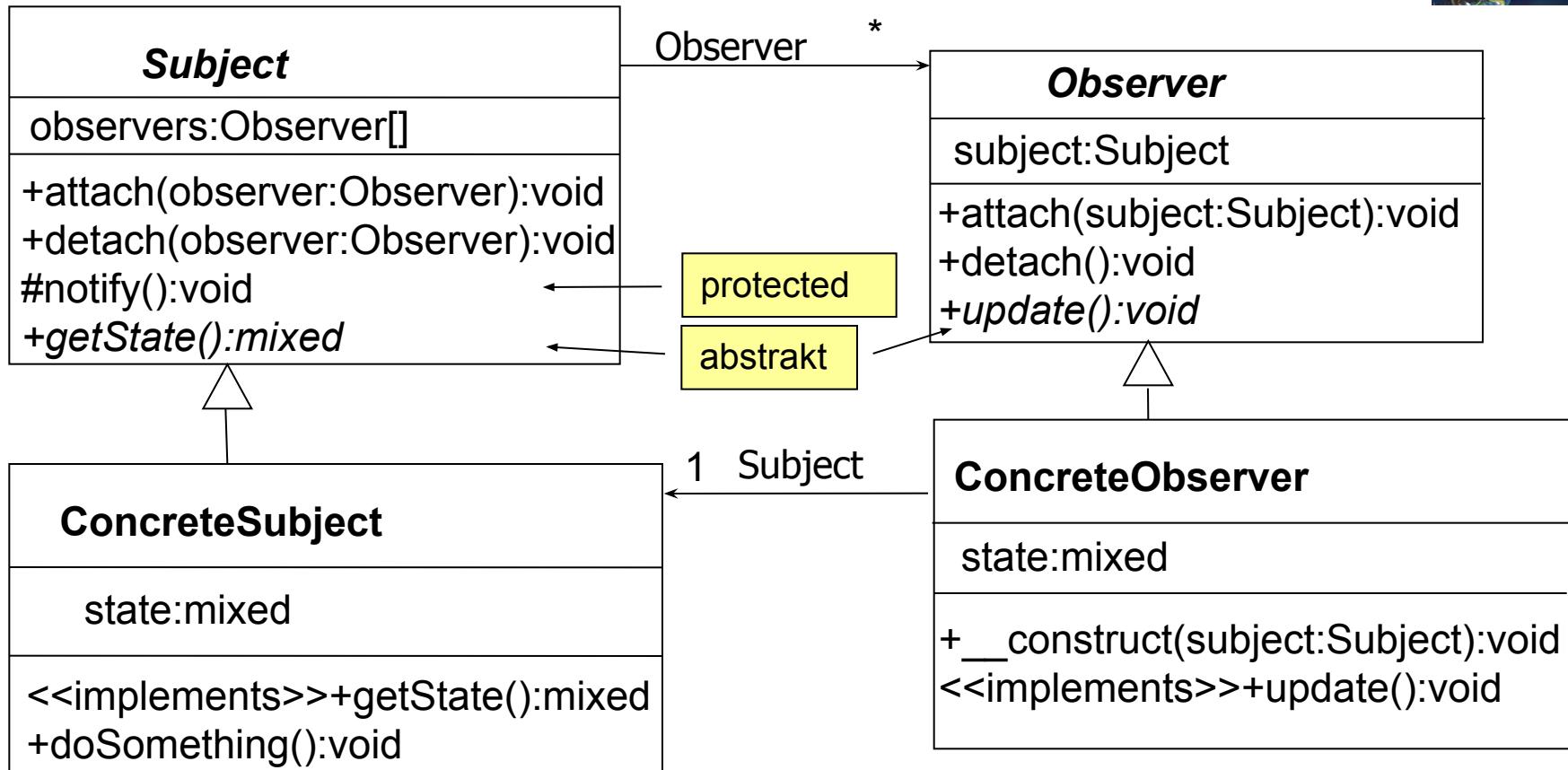
neu: Type Hinting für Klassen

Mit PHP5 sind die üblichen OO-Konzepte umsetzbar !



3.3.2 PHP Entwicklung

Einsatz von Klassen am Beispiel Observermuster (I)



Quelle: S. Bergmann <http://www.professionelle-softwareentwicklung-mit-php5.de>

3.3.2 PHP Entwicklung

Einsatz von Klassen am Beispiel Observermuster (II)

```

require_once 'Observer.php';
abstract class Subject {
    protected $observers = array();
    public function attach(Observer $observer) {
        $this->observers[] = $observer;
    }
    public function detach(Observer $observer) {
        for ($i = 0; $i < sizeof($this->observers); $i++) {
            if ($this->observers[$i]===$observer) {
                unset($this->observers[$i]);
            }
        }
    }
    protected function notify() {
        for ($i=0; $i<sizeof($this->observers); $i++)
            if (isset($this->observers[$i]))
                $this->observers[$i]->update();
    }
}

```

Datentyp Observer
bekannt machen

erzeugt Array mit
Lücke(n)

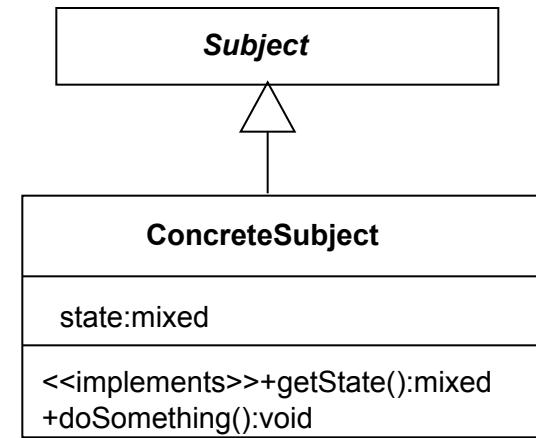
<i>Subject</i>
observers:Observer[]
+attach(observer:Observer):void
+detach(observer:Observer):void
#notify():void
+getState():mixed

Quelle: S. Bergmann <http://www.professionelle-softwareentwicklung-mit-php5.de>

3.3.2 PHP Entwicklung

Einsatz von Klassen am Beispiel Observermuster (III)

```
class ConcreteSubject extends Subject {  
    protected $state = NULL;  
    public function getState() {  
        return $this->state;  
    }  
    public function onEvent() {  
        $this->notify();  
    }  
    //weitere Methoden  
}
```



Observer und ConcreteObserver werden analog umgesetzt !

Quelle: S. Bergmann <http://www.professionelle-softwareentwicklung-mit-php5.de>

3.3.2 PHP Entwicklung

Fehlerbehandlung – die traurige Art

- Häufige Art der Fehlerbehandlung
 - ⇒ sofortiger Abbruch

```
die ("ungültiger Parameter"); // Meldung an Browser
```

oder

```
exit (5); // Fehlercode an OS
```
 - ⇒ Unterdrückung der Fehlermeldungen mit dem Fehlerkontrolloperator @
(darf vor jeder Expression stehen)

```
@$f = fopen("myfile.txt", "r");
```
- Normalerweise endet ein PHP-Programm am Dateiende und erzeugt dann (hoffentlich) eine sinnvolle HTML-Ausgabe
 - ⇒ Bei Fehlern entsteht eine unvollständige HTML-Datei und es hängt vom Browser ab, was angezeigt wird
 - ⇒ es entstehen häufig leere Seiten oder Layout-Trümmer
 - ⇒ Der Anwender erhält oft keine Rückmeldung wo das Problem liegt – geschweige denn, ob er etwas dagegen tun kann

3.3.2 PHP Entwicklung

Ausnahmebehandlung – wie es sein sollte

- Seit PHP 5 gibt es echte Ausnahmebehandlung

```
try { // hier kommt der problematische Aufruf
    if (...) { // was auch immer schief gehen kann
        throw new Exception("Meine Fehlermeldung");
    }
}
catch (Exception $e) { // typisiert !
    echo $e->getMessage(); // besser: komplettes HTML
}
```

- ⇒ HTML-Schachtelungsstruktur kann ordentlich abgeschlossen werden
- ⇒ sinnvolle Meldungen können gezielt angezeigt werden
(z.B. Alternative Kontaktmöglichkeit oder
Info, dass der Admin benachrichtigt wurde) etc.
- ⇒ `finally` (vgl. Java und ECMAScript) gibt es nicht in PHP

3.3.2 PHP Entwicklung

Ausnahmebehandlung: Beispiel

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

function dividiere ($dividend, $divisor) {
    if ($divisor == 0){ // Es soll durch null geteilt werden -> Exception
        throw new Exception("Division durch null ist nicht definiert");
    }
    $erg = $dividend/$divisor;
    return $erg;
}
// hier kommt der HTML-Header und der Beginn des <body>
try {                      // leitet den Block ein, in dem Exceptions abgefangen werden
    dividiere(2,0);    // generiert eine Exception
}
catch (Exception $fehler){ // fängt die Exception ab
    echo "<p>Ausnahmefehler: ".$fehler->getMessage()."</p>";
}
// hier kommen das Ende des <body> und die üblichen schließenden Tags
```

Abgewandelt von: C. Möhrke "Besser PHP programmieren"

3.3.2 PHP Entwicklung

Ausnahmebehandlung für verschiedene Fälle

- Häufig sollen Fehler durch die Anwendung unterschieden werden
 - ⇒ das Öffnen einer nicht vorhandenen Datei erzeugt immer einen Fehler
 - ⇒ dieser Fehler ist manchmal kritisch (Datenbank nicht vorhanden) und manchmal eher nicht (css-Datei fehlt)
 - ⇒ Durch Ableiten von der Klasse **Exception** können verschiedene Fehlertypen definiert und unterschiedlich abgefangen werden
 - ⇒ Der Konstruktor muss definiert werden!
- ```
class criticalError extends Exception {
 public function __construct ($Message) {
 parent::__construct($Message);
 }
}
class noncriticalError extends Exception {...}
...
try{...}
catch (criticalError $fehler){...}
catch (noncriticalError $fehler){...}
```

Abgewandelt von: C. Möhrke "Besser PHP programmieren"

### 3.3.2 PHP Entwicklung

## PHP5 vs. PHP4

- PHP5 ist mit Blick auf Abwärtskompatibilität zu PHP4 entwickelt – aber es gibt dennoch einige wichtige Unterschiede:
  - ⇒ PHP4 übergibt stets eine Kopie von Objekten; PHP 5 übergibt Referenzen. Kopien müssen bei Bedarf explizit erzeugt werden (`$Kopie = clone $instance;`)
  - ⇒ In PHP4 hieß der Konstruktor genauso wie die Klasse (und wurde bei Namensänderungen oft vergessen). Der Konstruktor hat ab PHP5 einen klassenunabhängigen Namen (`__construct`) und muss innerhalb der Klasse definiert werden.
  - ⇒ Objekte können in PHP5 erst erzeugt werden, nachdem die Klasse deklariert ist
  - ⇒ Es wurden die bereits genannten neuen Schlüsselwörter eingeführt (z.B. `abstract`, `clone`, `final` usw.)

### 3.3.2 PHP Entwicklung

## PHP bietet sehr viele Funktionen

- Array-Funktionen
- Crack-Funktionen
- Dateisystem-Funktionen
- Datums- und Zeit-Funktionen
- Functions-Funktionen
- Image-Funktionen
- Klassen und Objekt-Funktionen
- Kontroll-Mechanismen
- Mail-Funktionen
- Mathematische-Funktionen
- MySQL-Funktionen

Nachbau der aus C++ bekannten STL:  
Standard PHP Library (SPL)

- PDF-Funktionen
- PHP-Deklaration
- PHP-Informationen
- Reguläre Ausdrücke
- Session-Funktionen
- String-Funktionen
- URL-Funktionen
- Variablen-Funktionen
- Verzeichnis-Funktionen
- Sonstige-Funktionen

<http://www.selfphp.de> oder <http://www.php.net/>  
bieten eine Übersicht der PHP-Funktionen



### 3.3.2 PHP Entwicklung

## Anwendung von PHP: PDF-Erzeugung

- Es gibt mehrere Bibliotheken zum Erzeugen von PDFs
  - ⇒ **pdflib** (<http://www.pdflib.com>)
    - ist bereits bei XAMPP dabei und kann als PHP-Modul aktiviert werden
    - für kommerzielle Anwendungen fallen Lizenzkosten an
  - ⇒ **fpdf** (<http://fpdf.org/>)
    - eine PHP-Klasse, die Funktionen zum Erstellen von PDFs bietet
    - funktioniert ohne nennenswerte Installation
    - kostenlos und auch kommerziell frei einsetzbar

- PDF-Bibliotheken definieren ein spezielles DOM

- ⇒ die Einarbeitung ist die größte Arbeit

```
<?php /* fpdf Tutorial 1*/
require('fpdf.php');
$pdf=new FPDF();
$pdf->AddPage();
$pdf->SetFont('Arial','B',16);
$pdf->Cell(40,10,'Hello World!');
$pdf->Output();
?>
```



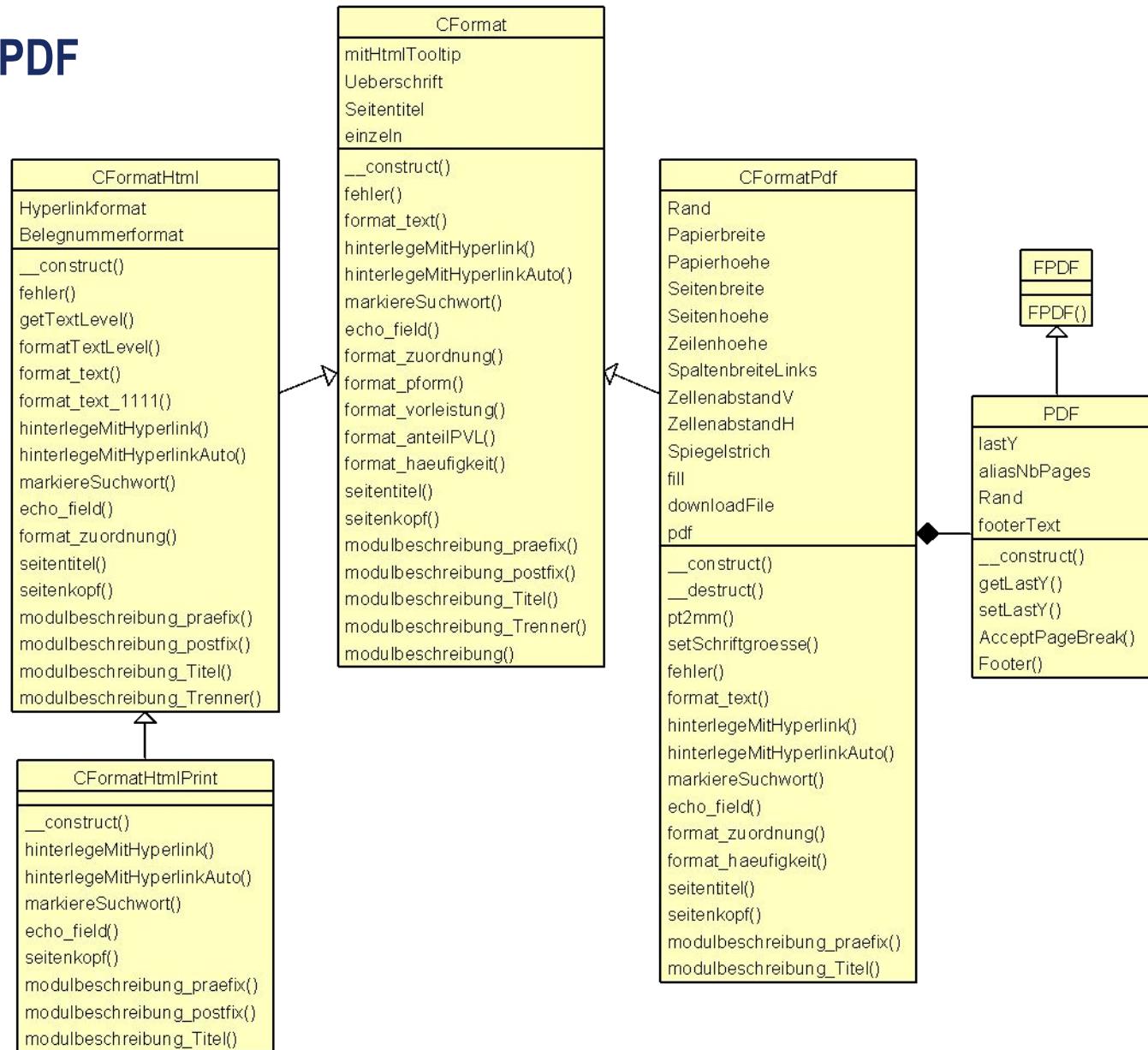
### 3.3.2 PHP Entwicklung

## Multiview HTML / PDF

### Beispiel aus Modulhand- buch:

- ⇒ HTML
- ⇒ HTML  
für Druck
- ⇒ PDF

### wartungs- freundlich durch Klassen- hierarchie



### 3.3.2 PHP Entwicklung

## 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"
- 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 !

besser wäre: in der Anwendung



### 3.3.2 PHP Entwicklung

## Zusammenfassung

### ■ Grundlagen

- ⇒ Idee
- ⇒ Historie

### ■ Notation

- ⇒ Grundstil C++
- ⇒ Integration von HTML <?php ... ?>
- ⇒ Variablennamen beginnen mit \$
- ⇒ Strings ., "...", '...', echo <<<EOT
- ⇒ Arrays und deren Übergabe
- ⇒ Umwandlung von Sonderzeichen
- ⇒ Objektorientierung
- ⇒ Ausnahmebehandlung

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

# Hochschule Darmstadt

## Fachbereich Informatik

### 3.3.3 Datenbankanbindung mit PHP



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

## 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
  - ⇒ PostgreSQL kostenlos unter PostgreSQL License
  - ⇒ MySQL kostenlos unter GPL; kostenpflichtige kommerzielle Lizenz
- Häufig im Bundle:  
Linux bzw. Windows + Apache + MySQL + PHP
  - ⇒ z.B. auch bei XAMPP

LAMP / WAMP

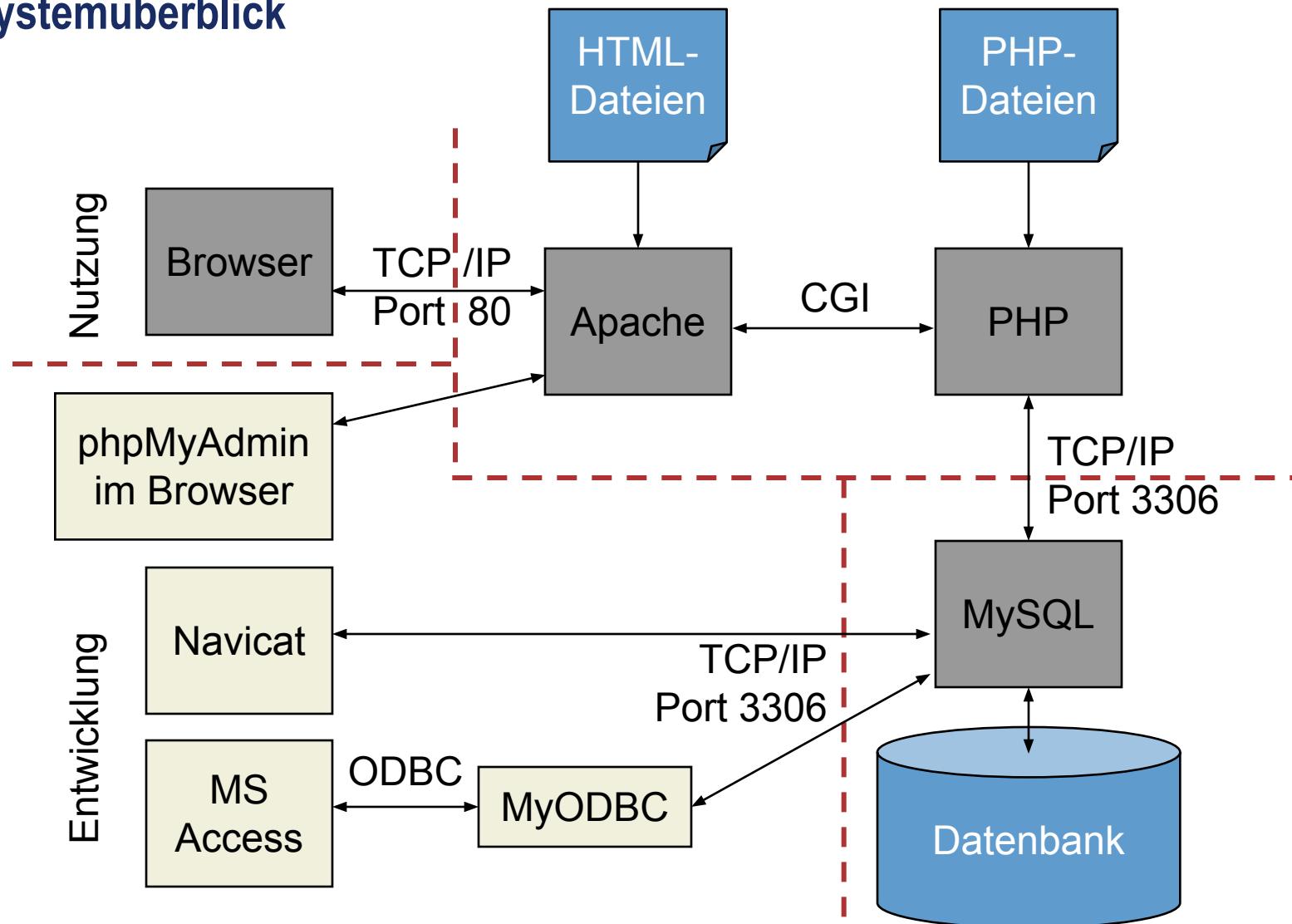
### 3.3.3 Datenbankanbindung mit PHP

## Datenbank MySQL

- MySQL arbeitet wahlweise mit verschiedenen Storage Engines
  - ⇒ mit Transaktionen (Pro: Storage Engine InnoDB)
  - ⇒ ohne Transaktionen (Classic: Storage Engine MyISAM)
- die (PHP-)Anwendung selbst benötigt nur den Datenbank-Server mysqld.exe
- ein Front-End ist erforderlich für DB-Entwicklung und -Administration
  - ⇒ z.B. Navicat für Windows, Mac, Linux (<http://www.navicat.com/>)
  - ⇒ als Webanwendung **phpMyAdmin** (<http://localhost/phpmyadmin>)
    - auch Remote verwendbar
    - wird mit XAMPP installiert
  - ⇒ sehr komfortabel durch QBE: Microsoft Access über **MyODBC**

### 3.3.3 Datenbankanbindung mit PHP

## Systemüberblick



### 3.3.3 Datenbankanbindung mit PHP

## Beispiel: Administration mit phpMyAdmin



- phpMyAdmin zur Bearbeitung einer MySQL-Datenbank

The screenshot shows the phpMyAdmin interface for the 'reisebuero' database. The left sidebar lists the database structure with tables: flug, hotel, kunde, pauschalangebot, and zielflughafen. The 'kunde' table is currently selected and highlighted in green. The main area displays a grid of tables with their respective details: flug (96 rows, MyISAM), hotel (45 rows, MyISAM), kunde (2 rows, MyISAM), pauschalangebot (48 rows, MyISAM), and zielflughafen (20 rows, MyISAM). Below the grid, there are summary statistics: 5 Tabellen (Tables) and Gesamt (Total) 211 rows, all of which are MyISAM type.

Zeichensatz: utf8

Kollation: utf8\_unicode\_ci

(vgl. <http://dev.mysql.com/doc/refman/5.5/en/charset-unicode-sets.html>)

|                          | Zielflughafen | Land      |
|--------------------------|---------------|-----------|
| <input type="checkbox"/> |               | München   |
| <input type="checkbox"/> |               | Frankfurt |
| <input type="checkbox"/> |               | Barcelona |

### 3.3.3 Datenbankanbindung mit PHP

## SQL - zur Erinnerung

- ganz primitiv: MySQL über die Kommandozeile
  - ⇒ mysql.exe ist das mitgelieferte Kommandozeilen-Front-End
- \mysql\bin\mysql.exe bzw. C:\xampp\mysql\bin\mysql.exe
  - ⇒ use Reisebuero

```
SELECT zimmerart, kategorie FROM hotel ORDER BY kategorie;
UPDATE hotel SET preis='150' WHERE preis='116';
DELETE FROM hotel WHERE ort='Alanya';
INSERT INTO zielflughafen (zielflughafen, land)
VALUES ('Rom', 'Italien');
```

- ⇒ quit

- meist besser als PHP Algorithmen:
  - ⇒ Abfragen über mehrere Tabellen: INNER / LEFT / RIGHT JOIN
  - ⇒ Aggregatfunktionen: COUNT, MIN, MAX etc.
  - ⇒ Verschachtelte Abfragen durch Unterabfragen

### 3.3.3 Datenbankanbindung mit PHP

## PHP-Schnittstellen zu MySQL

### ■ MySQL - Extension

- ⇒ ursprüngliche und immer noch häufig eingesetzte Anbindung
- ⇒ rein prozedurale Schnittstelle
- ⇒ Empfohlen für ältere MySQL-Versionen (< V 4.1.3)

Wir verwenden  
MySQLi!

### ■ MySQLi - Extension (oft auch MySQL *improved*)

- ⇒ Objektorientierte Schnittstelle (zusätzlich zur prozeduralen Schnittstelle)
- ⇒ ab PHP5, für neuere MySQL-Versionen ( $\geq$  V 4.1.3) dringend empfohlen
- ⇒ Diverse Verbesserungen: Prepared Statements, Multiple Statements, Transaktionen, verbessertes Debugging uvm.

### ■ PHP Data Objects (PDO)

- ⇒ Datenbank-Abstraktions-Schicht für PHP
- ⇒ Einheitliche Schnittstelle unabhängig von der verwendeten Datenbank
- ⇒ Austausch der Datenbank mit sehr geringem Aufwand, aber evtl. Einschränkung der verfügbaren Funktionalität durch Vereinheitlichung

### 3.3.3 Datenbankanbindung mit PHP

## MySQLi Klassen

### ■ Die MySQLi-Erweiterung enthält folgende Klassen

#### ⇒ MySQLi

- Aufbau und Verwaltung einer Verbindung zwischen PHP und dem MySQL-Server
- Einstellung der Verbindungskonfiguration (z.B. SSL)

#### ⇒ MySQLi\_STMT

- Verwaltung und Vorbereitung einer Datenbankabfrage ("prepared statement")
- ermöglicht die Verknüpfung von PHP-Variablen mit Datenbankabfragen

#### ⇒ MySQLi\_Result

- Verwaltung einer Menge von Ergebnissen, die als Antwort auf eine Query zurückgeliefert werden
- Sollte nach der Verwendung wieder freigegeben werden

### 3.3.3 Datenbankanbindung mit PHP

## MySQLi im Standardeinsatz

- TCP/IP-Verbindung aufbauen

```
$Connection = new MySQLi($host, $user, $pwd, $DB_name);
$Connection->set_charset("utf8");
```

⇒ viele (optionale) Parameter; liefert ein Objekt der Klasse **MySQLi**

- Ergebnistabelle abfragen oder SQL-Aktion ausführen

```
$Recordset = $Connection->query ($SQLabfrage);
```

⇒ liefert ein Objekt der Klasse **MySQLi\_Result**

- nächste Zeile aus Ergebnistabelle in Array übertragen und auf einzelne Felder des Datensatzes zugreifen

```
while ($Record = $Recordset->fetch_assoc())
 $Wert = $Record['Feldname'];
```

- Speicher der Ergebnistabelle freigeben

```
$Recordset->free();
```

- TCP/IP-Verbindung schliessen

```
$Connection->close();
```

Fehlerbehandlung  
nicht vergessen !

Deutschland  
Frankreich  
Italien  
Spanien  
Türkei



**PHP**

### 3.3.3 Datenbankanbindung mit PHP

## Beispiel: Verbindungsauflaufbau und Datenbankabfrage

```

<?php
try {
 // MIME-Type der Antwort definieren (*vor* allem HTML):
 header("Content-type: text/html; charset=UTF-8");
 // alle möglichen Fehlermeldungen aktivieren:
 error_reporting (E_ALL);

 // Datenbank öffnen und abfragen:
 require_once 'pwd.php'; // Passwort & Co. einlesen
 $Connection = new MySQLi($host, $user, $pwd, "Reisebuero");

 // Verbindung prüfen:
 if (mysqli_connect_errno())
 throw new Exception("Connect failed: ".mysqli_connect_error());
 if (!$Connection->set_charset("utf8"))
 throw new Exception("Charset failed: ".$Connection->error);

 // SQL-Abfrage festlegen:
 $SQLabfrage = "SELECT Land FROM zielflughafen GROUP BY Land";
 $Recordset = $Connection->query ($SQLabfrage);
 if (!$Recordset)
 throw new Exception("Query failed: ".$Connection->error);
}
?>

```

**SQL**

Deutschland  
Frankreich  
Italien  
Spanien  
Türkei

PHP

### 3.3.3 Datenbankanbindung mit PHP

## Beispiel: Generierung einer Auswahlliste

```
<!DOCTYPE html><html lang="de"><head> ... </head>
<body>
 <p>Bitte wählen Sie ein Land:</p>
 <form id="Auswahl" action="Result.php" method="post">
<?php
 $AnzahlRecords = $Recordset->num_rows;
 echo ("\\t\\t<p><select name=\"AuswahlLand\" size=\"$AnzahlRecords\">\n");
 while ($Record = $Recordset->fetch_assoc()) {
 echo ("\\t\\t\\t<option>".htmlspecialchars($Record["Land"])."</option>\n");
 }
 $Recordset->free();
 $Connection->close();
 echo ("\\t\\t</select></p>\n");
?>
 <p><input type="submit" value="Flughäfen anzeigen"/></p>
</form>
 <p><input type="button" value="Flughafen einfügen" onclick="window.location.
 href='Add.php'"/></p>
</body></html><?php
} catch (Exception $e) { echo $e->getMessage(); }
```

HTML, SQL und PHP sind stark vermischt ! Das erschwert die Wartung, aber solche Lösungen findet man häufig. Im nächsten Kapitel behandeln wir einen objektorientierten Ansatz.

HTML

## Spezialfall: Mehrere Datensätze im selben Formular

ähnlich zur "Übergabe mit Mehrfachauswahl" in Formularen !

- Üblicherweise verwendet man die Feldnamen aus der DB als Elementnamen im Formular
  - ⇒ Primärschlüssel bei Bedarf in <input type="hidden">
- bei mehreren Datensätzen Unterscheidung durch indizierte Namen
  - ⇒ HTML erlaubt [ ] im Attribut name; PHP macht daraus ein Array

// generieren:

```
$i = 0;
while ($Record = $Recordset->fetch_assoc()) {
 echo "<input type=\"hidden\" name=\"id[$i]\" value=\"$Record['id']?>");
 echo "<input type=\"text\" name=\"attr[$i]\" value=\"$Record['attr']?>");
 $i++;
}
```

// auswerten:

```
for ($i=0; $i<count($_POST['id']); $i++) {
 $id = $Connection->real_escape_string($_POST['id'][$i]);
 $attr = $Connection->real_escape_string($_POST['attr'][$i]);
 $SQL = "UPDATE table SET attr='$attr' WHERE id='$id'";
}
```

## Kritische Abschnitte sichern

- ⇒ Konsequenz der Parallelverarbeitung unabhängiger HTTP-Requests
  - Zugriff auf gemeinsame Ressourcen (Dateien, Datensätze) muss gesichert sein
  - vgl. Mutex, Semaphore, Monitor
  - bei Datenbanken voneinander abhängige Mehrfachzugriffe mit Transaction und Lock einklammern
- ⇒ eventuelle Fehler treten erst unter Last auf
  - schwer zu testen
  - Entwurf sorgfältig durchdenken

Die Umsetzung unterscheidet sich stark für andere Datenbanken (Oracle, PostgreSQL) !  
 vgl. Isolation Level Serializable;  
 Transaction Rollback ist unerwünscht

*z.B. Belegung beschränkter Plätze*

```
query("Begin Transaction;");
query("Lock Table belegt Write;");
$result = query(
 "Select count(*) From belegt;");
$anzahl = fetch_row($result)[0];
```

*hier darf kein anderer Prozess unterbrechen und ein Insert Into ausführen; anderer Prozess soll warten*

```
if ($anzahl < 16)
 query("Insert Into belegt ...");
else
 echo("ist schon voll");
 query("Unlock Tables;");
 query("Commit");
```

# Hochschule Darmstadt

## Fachbereich Informatik

### 3.3.4 Systemarchitektur mit Seiten-Klassen



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Programmieren mit PHP

- Seit PHP 5 bietet PHP alles was man von einer modernen Programmiersprache erwartet:
  - ⇒ leider wird dies bei der Entwicklung häufig vergessen:
    - Funktions-Orientierung statt Objekt-Orientierung
    - undokumentierter Spaghetti-Code
    - Mischung von Algorithmen, Ausgabe, Datenbankzugriffen etc.
    - keine nennenswerte Ausnahmebehandlung
    - Verwendung von globalen Variablen
  - ⇒ dabei sind PHP-Programme häufig der Kern von großen Webauftritten, die in vielen Personenjahren entwickelt wurden
    - die Software wird weiterentwickelt
    - Wartung wird zunehmend wichtiger

Vergessen Sie nicht die ganz normalen (?!)  
Regeln eines guten Software Designs  
z.B. Geheimnisprinzip, starker Zusammenhalt, lose Kopplung...

## Programmierstil (I)

- Was halten Sie von folgendem Beispiel?

```
<?php
function Header($myTitle){
 echo "<html><head><title>";
 echo "$myTitle";
 echo "</title><body>";
}
function Footer(){
 echo "</body></html>";
}
***** main *****/
Header ("Hallo");
echo "PHP ist toll!";
Footer ();
?>
```

Stil:  
"PHP mit HTML"

## Programmierstil (II)

- Was halten Sie von folgendem Beispiel?

```
<html>
 <head>
 <title>Hallo</title>
 </head>
 <body>
 <?php echo "PHP ist toll!";?>
 </body>
</html>
```

Stil:  
"HTML mit PHP"

## Programmierstil (III)

- Es gibt verschiedene Möglichkeiten PHP und HTML zu kombinieren
  - ⇒ HTML mit eingebettetem PHP
    - ist übersichtlicher, wenn es vorwiegend um die HTML-Ausgabe geht
    - erlaubt die Erzeugung der HTML-Seite mit entsprechenden Tools.  
Die variablen Teile werden später als PHP-Abschnitte eingebaut.
  - ⇒ PHP mit eingebettetem HTML *geht auch objektorientiert*
    - ist übersichtlicher, wenn es vorwiegend um Berechnungen geht
    - also eher angebracht beim Auswerten von Daten, Datenbankoperationen, Algorithmen etc.
  - ⇒ die Heredoc-Notation kombiniert die Vorteile o.g. Ansätze
    - mehrzeiliger String
    - kein \ als Escape-Zeichen erforderlich vor "
    - Variablen werden durch ihren Wert substituiert
  - ⇒ Alternativ kann man mit "Templates" arbeiten
    - Die Webseite wird mit Platzhaltern erstellt und eingelesen  
`$inhalt = file_get_contents("template.html");`
    - anschließend werden die Platzhalter durch Inhalte ersetzt (`str_replace()`)

```
echo <<<EOT
<html lang="de">
<head> ...
 <title>$headline</title>
</head><body>
EOT;
```

### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Ein Negativbeispiel: Typischer (!?) prozeduraler PHP Code

```
<?php
echo <<<EOT
 <div class="bestellung"> <h1> Bestellung</h1> <table class="left">
EOT;
$SQLabfrage = "SELECT * FROM pizza ORDER BY pizzanummer";
// Datenbank öffnen und abfragen:
$Connection1 = new MySQLi($host, $user, $pwd, "pizzaservice");
// check connection
if (mysqli_connect_errno()) { printf ("Connect failed: %s\n",
 mysqli_connect_error()); exit(); }
if (!$Connection1->set_charset("utf8"))
 printf ("Charset failed: %s\n", $Connection1->error);
$Recordset = $Connection1->query ($SQLabfrage);
if (!$Recordset){ printf("Query failed: %s\n", $Connection1->error);
 exit();}

$Record = $Recordset->fetch_assoc();
while ($Record) {
 $Bez = $Record["Bezeichnung"];
 echo ("\t<tr>\n");
 echo ("\t\t<td><img src=\"$Datei\" class=\"pizza\" title=\"Pizza\"
 onclick=\"InsertPizza("$Bez");\">\n");
 $Record = $Recordset->fetch_assoc();
}
$Recordset->free(); $Connection1->close();
echo "</table>";
```

**PHP**

**HTML**

**DB**

**Java Script**

**Geht das auch ordentlich???**

### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Bewertung des "Prozeduralen PHP Codes"

Anforderung	Bewertung	Erfüllt?
Funktionserfüllung?	so weit ausprobiert...	●
Wiederverwendbarkeit?	Copy & Paste	:(
Testbarkeit?	keine Klasse, kein Unit-Test	:(
Wartbarkeit?	unübersichtlich, unverständlich	:(
Sicherheit?	viele Stellen mit kritischen Zugriffen	:(

Entwurfsprinzipien	Erfüllt?
Schwache Kopplung?	:(
Starker Zusammenhalt?	:(
Geheimnisprinzip? Kapselung?	:(

Es werden so ziemlich alle Prinzipien verletzt, die man in Softwaretechnik in vielen Jahren entdeckt hat!

Das ist für einzelne Webseiten akzeptabel, aber für einen größeren oder professionellen Webauftritt undenkbar!

## Webseiten im Vergleich zu klassischen GUIs



- Wie finden wir geeignete Klassen in einer webbasierten Anwendung ?
  - ⇒ erinnern wir uns an ENA oder an Java Swing ...
- Eine HTML-Seite ist vergleichbar mit einem Fenster in Java Swing
  - ⇒ PHP Seiten-Klassen analog zu Java Fenster-Klassen
- Ein HTML-Block in einer Seite ist vergleichbar mit einem Java Panel
  - ⇒ PHP Block-Klassen analog zu Java Panel-Klassen
- Eine Seiten-Klasse hat im Rahmen einer webbasierten Anwendung typischerweise 3 Teilaufgaben (= Methoden) zu erfüllen:
  1. Auswertung der (z.B. von einem Formular) übermittelten Daten und Schreiben dieser Daten in die Datenbank      `processReceivedData()`
  2. Abfrage von Daten aus der Datenbank      `getViewData()`
  3. Generierung und Ausgabe einer HTML-Seite mit Hilfe der Daten aus 2.      `generateView()`

Diese Namen verwenden  
wir auch in unseren  
„Seiten-Klassen“

### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Designklassen, Analyseklassen, GUI, Webseiten, Datenmodell

- Wie passt das alles zusammen? Ein Versuch
- ENA User Research wer sind die Benutzer?
  - ⇒ Stereotypen, Anwendungsszenarien was brauchen sie?
- OOAD Analyse(klassen) Student
  - ⇒ GUI Analyse: was soll er tun können?  
Navigationsübersicht, Screendiagramme
  - ⇒ DB Analyse: was muss man über ihn  
Entity Relationship Diagram speichern?
- OOAD Design(klassen)
  - ⇒ ENA Java Swing:  
Screen => Fensterklassen, Panelklassen
  - ⇒ EWA Webtechnologien:  
Screen = Seite => Seitenklassen, Blockklassen
  - ⇒ DB Relationenmodell:  
SQL Data Definition, Objektrelationales Mapping

am Beispiel OBS

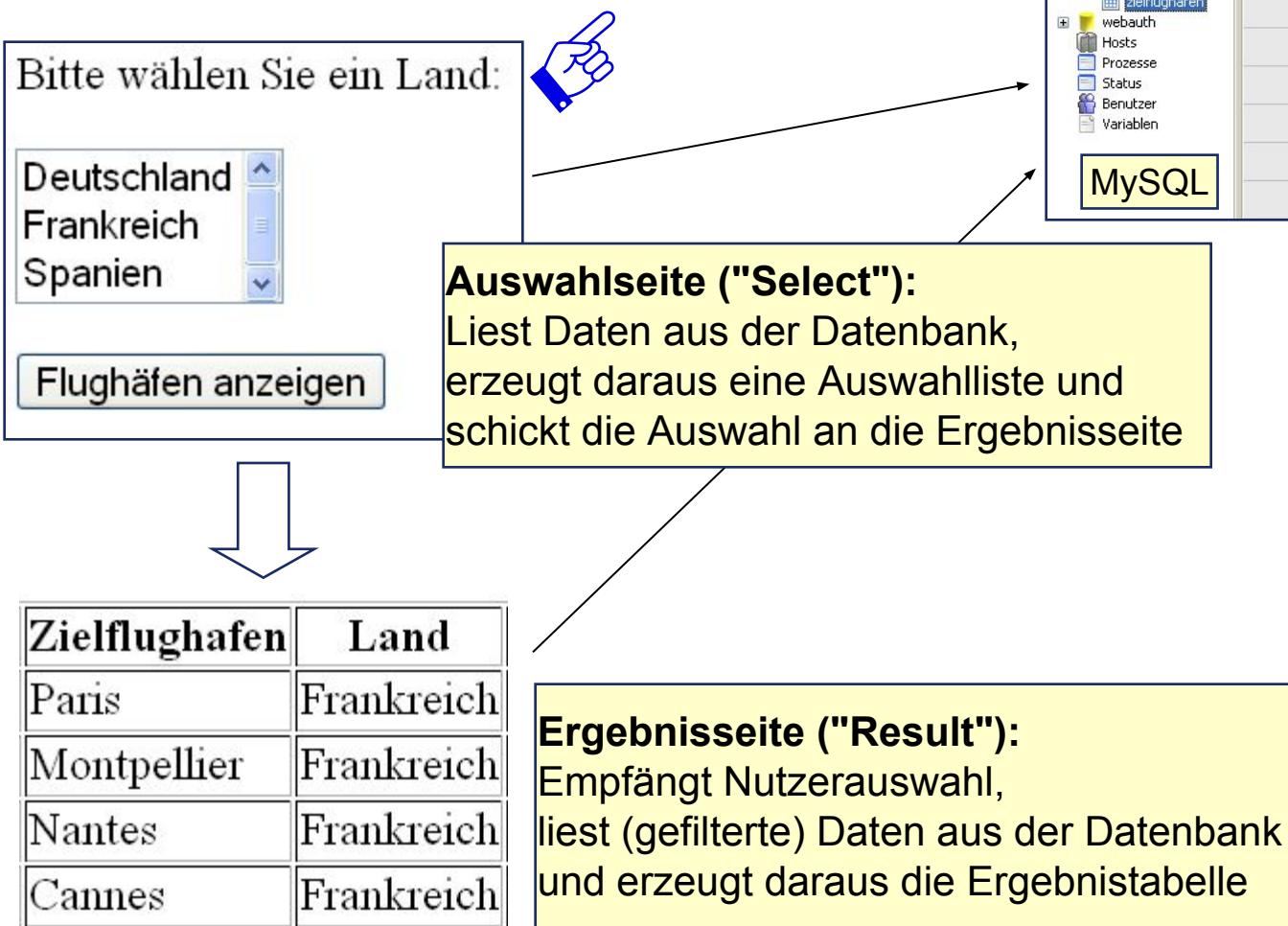
Was brauchen sie?

Belegen, Termine, Noten

Student - belegt - Vorlesung

### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Anwendungsbeispiel



## PHP-Klassen für HTML-Seiten



### ⇒ Page (abstrakte Klasse)

- verwaltet die Datenbankanbindung
- Beinhaltet Hilfsfunktionen zur Erzeugung einer HTML-Seite
- Vorverarbeitung in Methode `processReceivedData()`

### ⇒ Select

- Erzeugt eine HTML-Seite mit einem Auswahlfeld und sendet die Auswahl an Result
- Beinhaltet die Hilfsfunktion `insert_option()` zum Einfügen von Elementen in eine Select-Box
- Implementiert die Methoden `getViewData()`, `generateView()` und `main()`
- `processReceivedData()` bleibt leer, da keine Daten empfangen werden

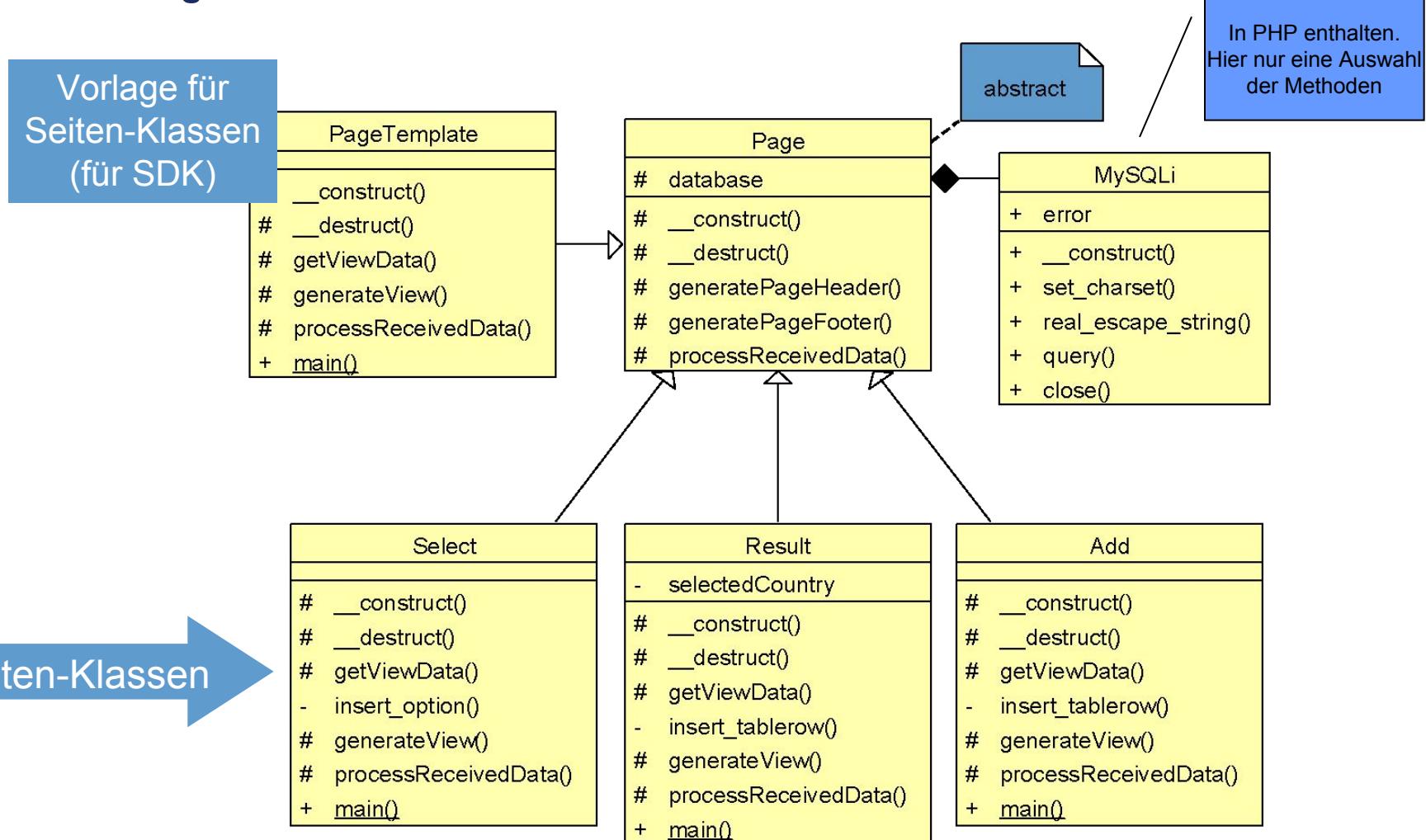
`main()` ist in php kein Schlüsselwort!  
Die Methode muss explizit beim Laden aufgerufen werden.

### ⇒ Result (Add analog)

- Erhält beim Aufruf ausgewählte Daten und erzeugt eine entsprechende HTML-Seite
- Beinhaltet die Methoden `getViewData()`, `generateView()`, `main()` und `processReceivedData()`
- Beinhaltet die Hilfsfunktion `insert_tablerow()` zum Einfügen von Zeilen in eine Tabelle,
- Beinhaltet ein Attribut zur Übergabe der Daten zwischen `getViewData()` und `generateView()`

### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Klassendiagramm



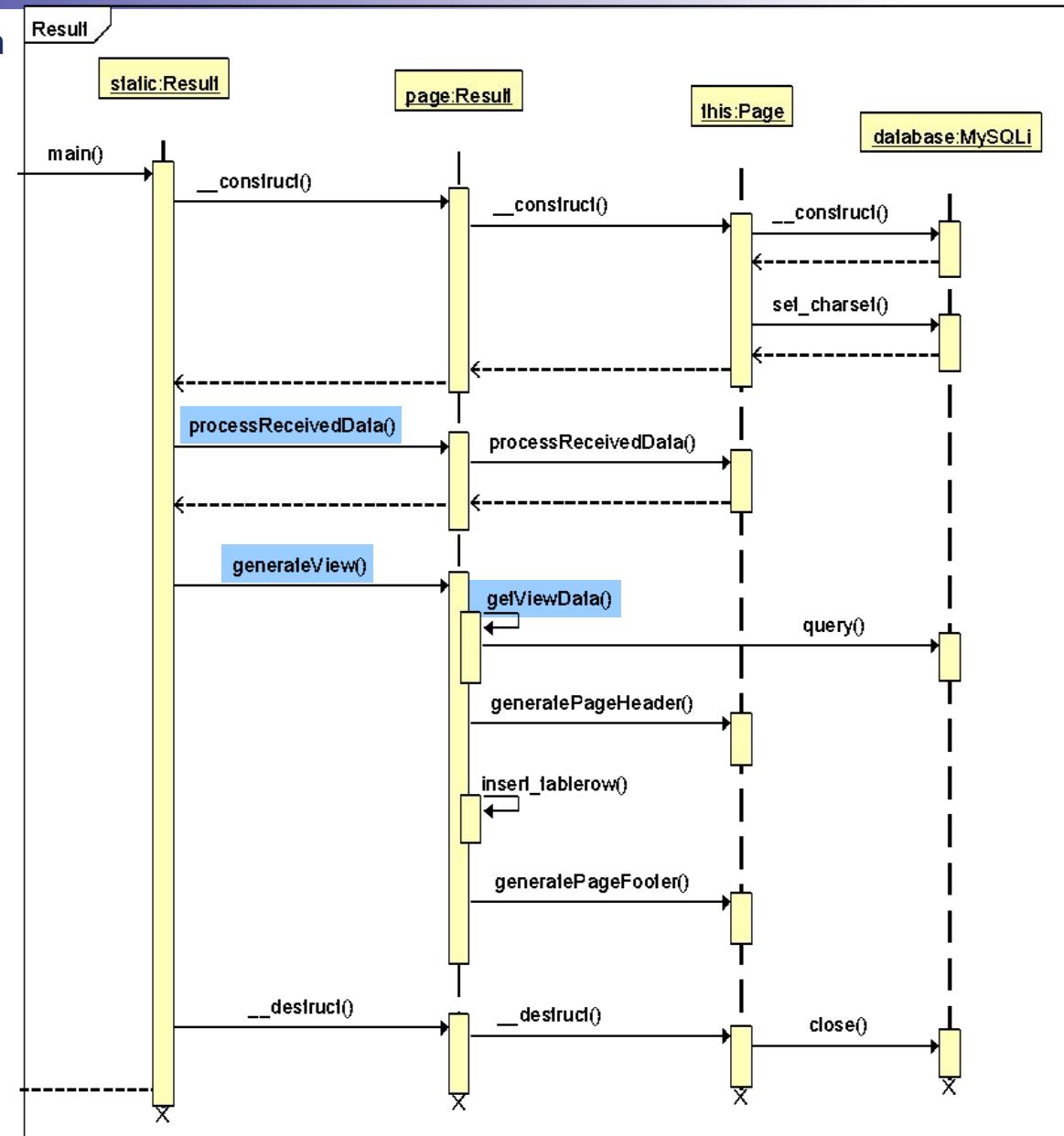
UML-Diagramme erstellt mit BOUML <http://bouml.free.fr/>

### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Seqenzdiagramm

- Beim Laden der Seitenklasse **Result** wird die (statische) Methode main() ausgeführt

- das "Objekt" **static** repräsentiert hier statische Attribute und Methoden der Klasse **Result**
- diverse Aufrufe gehen an die von der Klasse **Page** geerbten Methoden



### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Gemeinsame Basisklasse 'Page'

```
<?php
abstract class Page
{
 protected $database = null; // Referenz auf Datenbankobjekt

 protected function __construct() // öffnet die Datenbank
 { ... }

 protected function __destruct() // schließt die Datenbank
 { ... }

 protected function generatePageHeader($title = '')
 { ... } // generiert Anfang der HTML-Seite

 protected function generatePageFooter()
 { ... } // generiert Ende der HTML-Seite

 protected function processReceivedData()
 { ... }
}
```



### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Seiten-Klasse 'Result': Klassenrahmen und Aufruf

```
<?php
require_once './Page.php';
class Result extends Page
{
 private $selectedCountry;

 protected function __construct() {
 parent::__construct();
 $this->selectedCountry = 'xxx'; // selects nothing
 }

 protected function __destruct() {
 parent::__destruct();
 }
}

Result::main(); // nur main() wird direkt ausgeführt
```



hier werden die Methoden von den folgenden Folien eingefügt

### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Seiten-Klasse 'Result': main() und processReceivedData()

```
public static function main() {
 try {
 $page = new Result();
 $page->processReceivedData();
 $page->generateView();
 }
 catch (Exception $e) {
 header("Content-type: text/plain; charset=UTF-8");
 echo $e->getMessage();
 }
}

protected function processReceivedData() {
 parent::processReceivedData();
 if (isset($_POST["AuswahlLand"])) {
 $this->selectedCountry = $_POST["AuswahlLand"];
 }
}
```

Hauptprogramm  
mit Fehlerbehandlung

Verarbeitung der  
Formulardaten

Die Klasse 'Select' hat das Formular generiert,  
das hier von 'Result' ausgewertet wird! ☺

### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Seiten-Klasse 'Result': getViewData()

```
protected function getViewData() {
 // build SQL query from form parameters

 $countries = array();
 $selCountry = $this->database->real_escape_string($this->selectedCountry);
 $sql = "SELECT * FROM zielflughafen WHERE Land=\"$selCountry\"";
 $recordset = $this->database->query ($sql);
 if (!$recordset)
 throw new Exception("Fehler in Abfrage: ".$this->database->error);
 // read selected records into result

 array
 while ($record = $recordset->fetch_assoc()) {
 $airport = $record["Zielflughafen"];
 $country = $record["Land"];
 $countries[$airport] = $country;
 }
 $recordset->free();
 return $countries;
}
```

liest per SQL aus der Datenbank und liefert ein assoziatives Array

### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Seiten-Klasse 'Result': generateView()

```
protected function generateView() {
 $countries = $this->getViewData(); // before first HTML output
 $this->generatePageHeader('Ergebnis');
 echo <<<EOT
 <h1>Ausgewählte Flughäfen:</h1>
 <table>
 <tr><th>Zielflughafen</th><th>Land</th></tr>
EOT;
 foreach($countries as $airport => $country) {
 $airport = htmlspecialchars($airport);
 $country = htmlspecialchars($country);
 $this->insert_tablerow("\t\t\t", $airport, $country);
 }
 echo <<<EOT
 </table>
 <p><input type="button" value="Neue Auswahl"
 onclick="window.location.href='Select.php'" /></p>
EOT;
 $this->generatePageFooter();
}
```

generiert HTML aus  
dem assoziativen  
Array

### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Seiten-Klasse 'Result': insert\_tablerow()

```
private function insert_tablerow($indent, $entry1="", $entry2="") {
 echo $indent."<tr>\n";
 echo $indent."\t<td>$entry1</td>\n";
 echo $indent."\t<td>$entry2</td>\n";
 echo $indent."</tr>\n";
}
```

Hilfsfunktion für  
generateView()

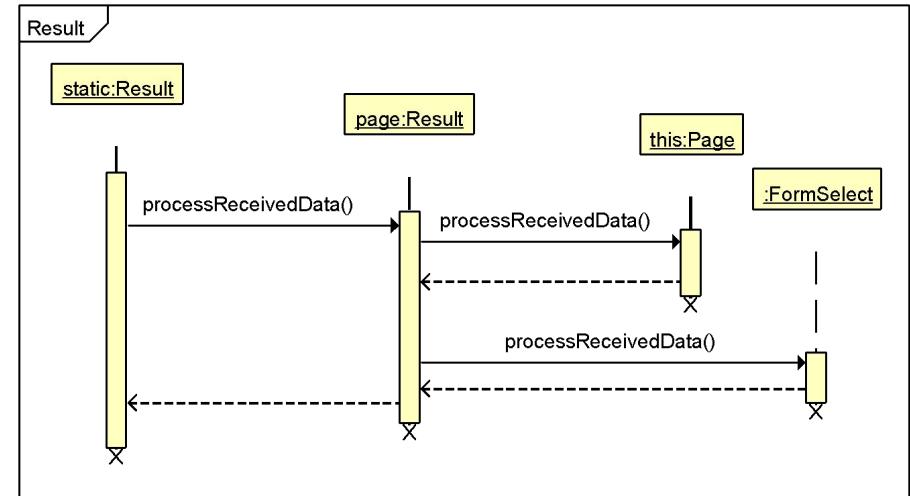
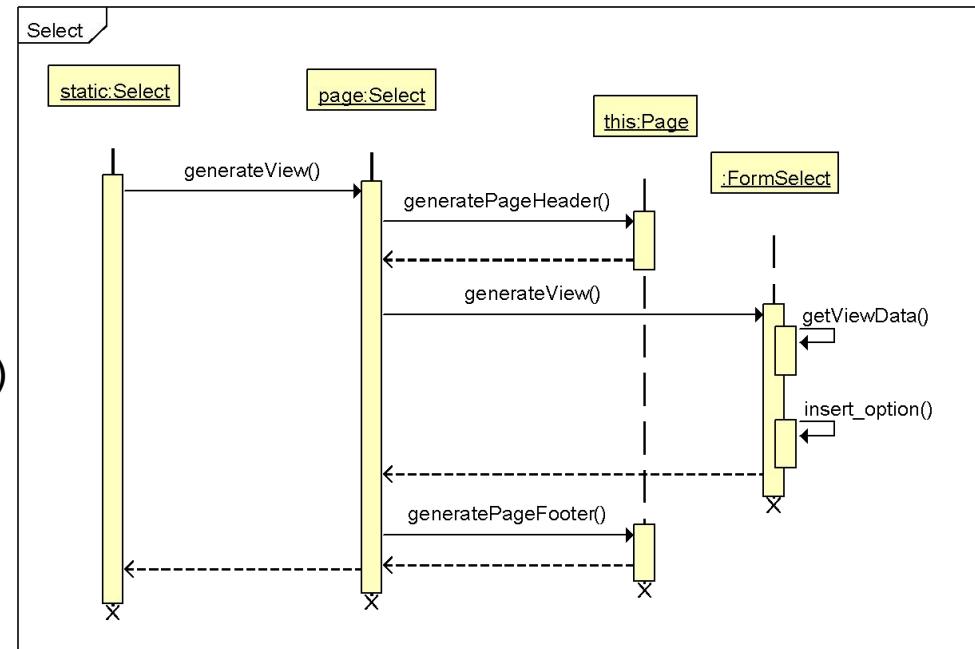
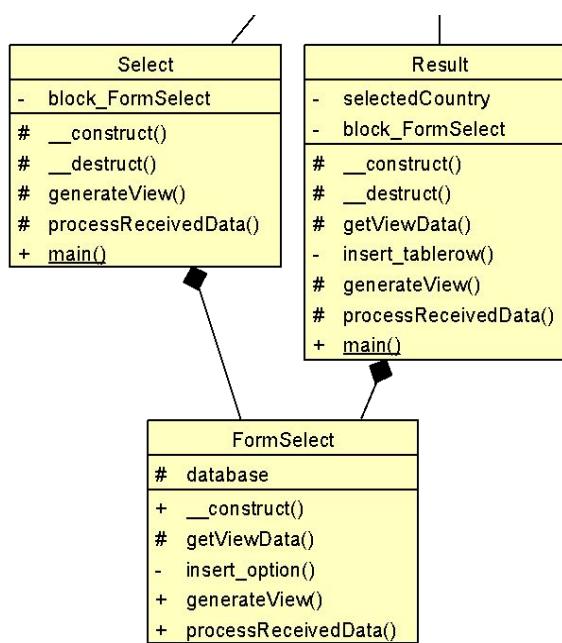
## Block-Klassen

- einige Defizite der bisherigen Seiten-Klassen:
  - ⇒ gleiche bzw. ähnliche Strukturen auf verschiedenen Seiten werden mehrfach implementiert (im Beispiel die Tabelle der Flughäfen)
  - ⇒ bei komplexeren Seiten werden die Methoden `getViewData()` und `generateView()` unübersichtlich groß (tritt im trivialen Beispiel nicht auf)
  - ⇒ Generierung und Auswertung von Formularen finden oft auf verschiedenen Seiten statt (im Beispiel `Select` und `Result`)
- üblicher Lösungsansatz: Funktionalität auf weitere Methoden und/oder Klassen aufteilen
  - ⇒ spezialisiert für GUI-Objekte: Block-Klassen einführen
  - ⇒ Block-Klassen repräsentieren beliebige HTML Block Tags
  - ⇒ besonders nützlich für `<form>` Tags
    - Generierung und Auswertung des Formulars in derselben Klasse
    - eventuelle Änderungen oder Erweiterungen betreffen nur noch eine Klasse

### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Block-Klasse FormSelect

- FormSelect wird Komponente von Select und Result
  - ⇒ Select ruft generateView()
  - ⇒ Result ruft processReceivedData()
- alle Teilaufgaben des Formulars sind nun gekapselt in FormSelect



### 3.3.4 Systemarchitektur mit Seiten-Klassen

## FormSelect->generateView() in class Select

```
class Select extends Page
{
 private $block_FormSelect;
 protected function __construct() {
 parent::__construct();
 $this->block_FormSelect = new FormSelect($this->database);
 }
 protected function generateView() {
 $this->generatePageHeader('Auswahl');
 echo <<<HERE
<p>Bitte wählen Sie ein Land:</p>
HERE;
 $this->block_FormSelect->generateView();
 echo <<<HERE
<p><input type="button" value="Flughafen einfügen"
 onclick="window.location.href='Add.php'" /></p>
HERE;
 $this->generatePageFooter();
 }
}
```



### 3.3.4 Systemarchitektur mit Seiten-Klassen

## FormSelect->processReceivedData() in class Result

```
class Result extends Page
{
 private $selectedCountry;
 private $block_FormSelect;

 protected function __construct() {
 parent::__construct();
 $this->selectedCountry = 'xxx';
 $this->block_FormSelect = new FormSelect($this->database);
 }

 protected function processReceivedData() {
 parent::processReceivedData();
 $this->block_FormSelect->processReceivedData(
 $this->selectedCountry); // Parameter nach Bedarf
 }
}
```

### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Bewertung des Ansatzes "Seiten- und Block-Klassen"

Anforderung	Bewertung	Erfüllt?
Funktionserfüllung?	mit Unit-Tests automatisiert testbar	😊
Wiederverwendbarkeit?	Wiederholung von main()	●
Testbarkeit?	Klassen erlauben Unit-Tests	😊
Wartbarkeit?	nach Einarbeitung ok, aber fehleranfällig, weil Aufrufe in bestimmter Reihenfolge erfolgen müssen	●
Sicherheit?	wenige Stellen mit kritischen Zugriffen	😊

Entwurfsprinzipien	Erfüllt?
Schwache Kopplung?	●
Starker Zusammenhalt?	😊
Geheimnisprinzip? Kapselung?	😊

Es werden schon viele Prinzipien erfüllt,  
aber es geht noch besser !

### 3.3.4 Systemarchitektur mit Seiten-Klassen

## Verbleibende Defizite

- manche Dinge werden in jeder Seiten-Klasse wiederholt
  - ⇒ Methode main()
  - ⇒ Ausnahmebehandlung try ... catch
  - ⇒ Aufruf von generatePageHeader() und generatePageFooter()

das könnte ein **Framework** lösen

- jede Seiten- bzw. Block-Klasse greift auf die Datenbank zu
  - ⇒ und wenn man die Datenbank austauschen will oder muss ?
- jede Seiten- bzw. Block-Klasse generiert HTML
  - ⇒ und wenn man andere Clients bedienen will, z.B. mobile Geräte ?

man könnte Daten (**model**) und deren Darstellung (**view**) trennen

↳ **Model-View-Controller Framework**

# Hochschule Darmstadt

## Fachbereich Informatik

### 3.3.5 Model-View-Controller Framework



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

## Funktionsprinzip eines Frameworks

- Ziel: Vorgabe einer klaren Struktur für die Anwendung
  - ⇒ und Lösungen für Standardaufgaben:  
Datenhaltung, Sessionverwaltung, sichere Formulare
- bei Verwendung eines Frameworks schreibt der Entwickler kein "Hauptprogramm" mehr
  - ⇒ das Hauptprogramm steckt im Framework
  - ⇒ zentrale Steuerungslogik
- die Anwendung besteht nur noch aus einzelnen Teilen (i.a. Klassen)
  - ⇒ diese müssen bestimmte Schnittstellenkonventionen erfüllen
  - ⇒ und werden vom Framework zu gegebener Zeit aufgerufen
- Umkehrung der Steuerung (Inversion of control)
  - ⇒ traditionell: die Anwendung ruft Methoden aus Bibliotheken
  - ⇒ mit Framework: das Framework ruft Methoden der Anwendung
  - ⇒ Hollywood-Prinzip: "don't call us, we'll call you"

## Drei-Schichten-Architektur

- Die "Standardtechniken" des Software-Engineering sind auch auf die Architektur einer Webanwendung anwendbar
  - ⇒ z.B. starker Zusammenhalt, schwache Kopplung, Kapselung, Auslagern von variablen Teilen usw.
  - ⇒ eine Aufteilung der Anwendung nach Schichten (z.B. Three-Tier Architecture) ist naheliegend

Für Webanwendungen:

**Präsentationsschicht**  
User interface

HTML generieren

**Geschäftslogik**  
Anwendungslogik

(Formular-)Daten  
verarbeiten

**Persistenzschicht**  
Datenspeicherung

DB abfragen,  
DB aktualisieren

## Model-View-Controller – die Grundidee



### ■ Trenne eine Anwendung in Model, View und Controller

#### ⇒ View(s)

- kümmern sich "nur" um die Darstellung der Daten
- reichen Eingaben weiter an den Controller
- sind leicht austauschbar

#### ⇒ Controller

- nimmt Eingaben an und stellt fest, was das für das Model bedeutet

#### ⇒ Model

- enthält Anwendungslogik
- kapselt Datenbankzugriffe

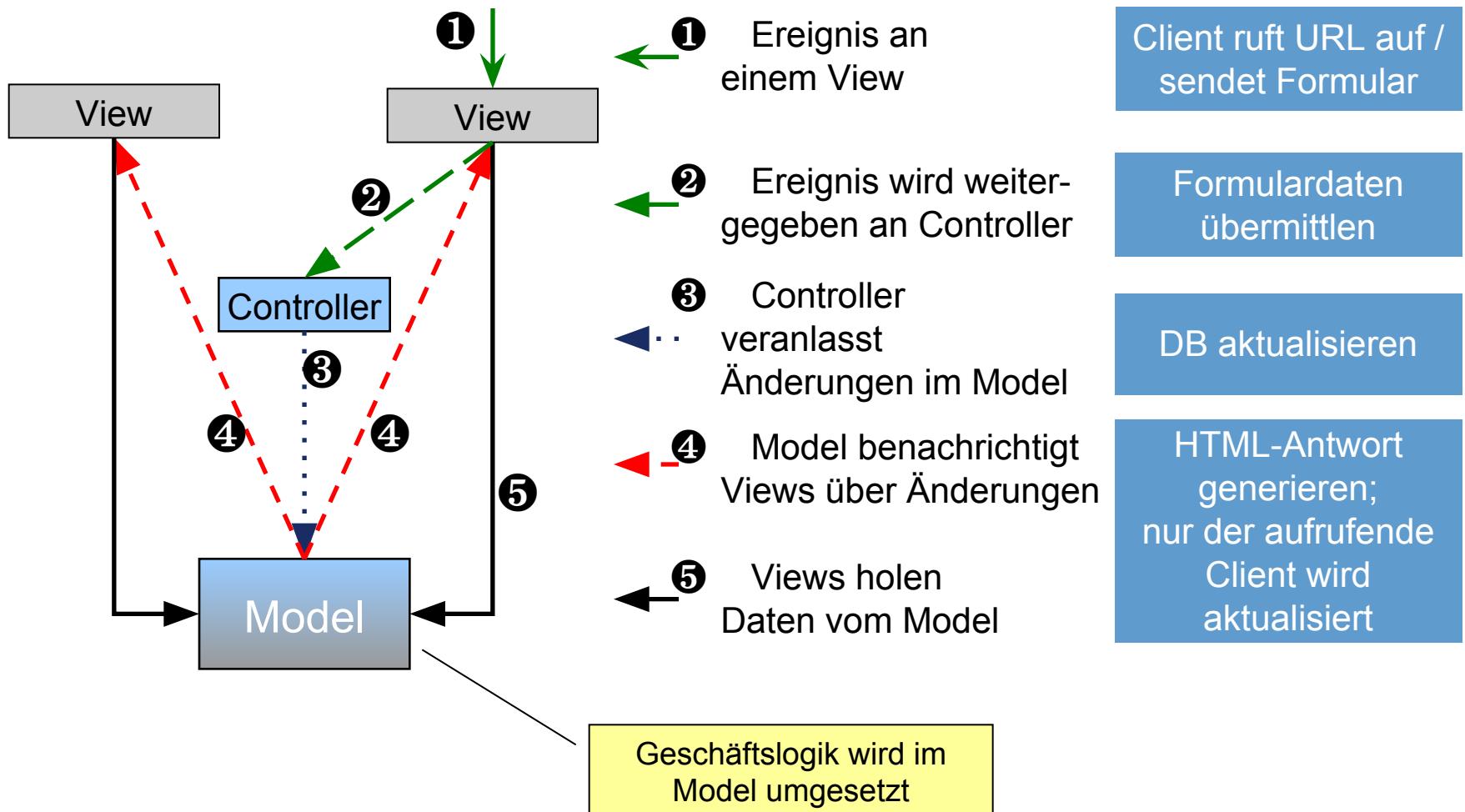
### ■ Die Teile sind entkoppelt und leicht austauschbar

#### ⇒ durch eine Observer-Schnittstelle wird das Model vom Rest entkoppelt

Diese Aufteilung hat sich schon in vielen  
Anwendungen bewährt

## Model-View-Controller in lokaler Anwendung

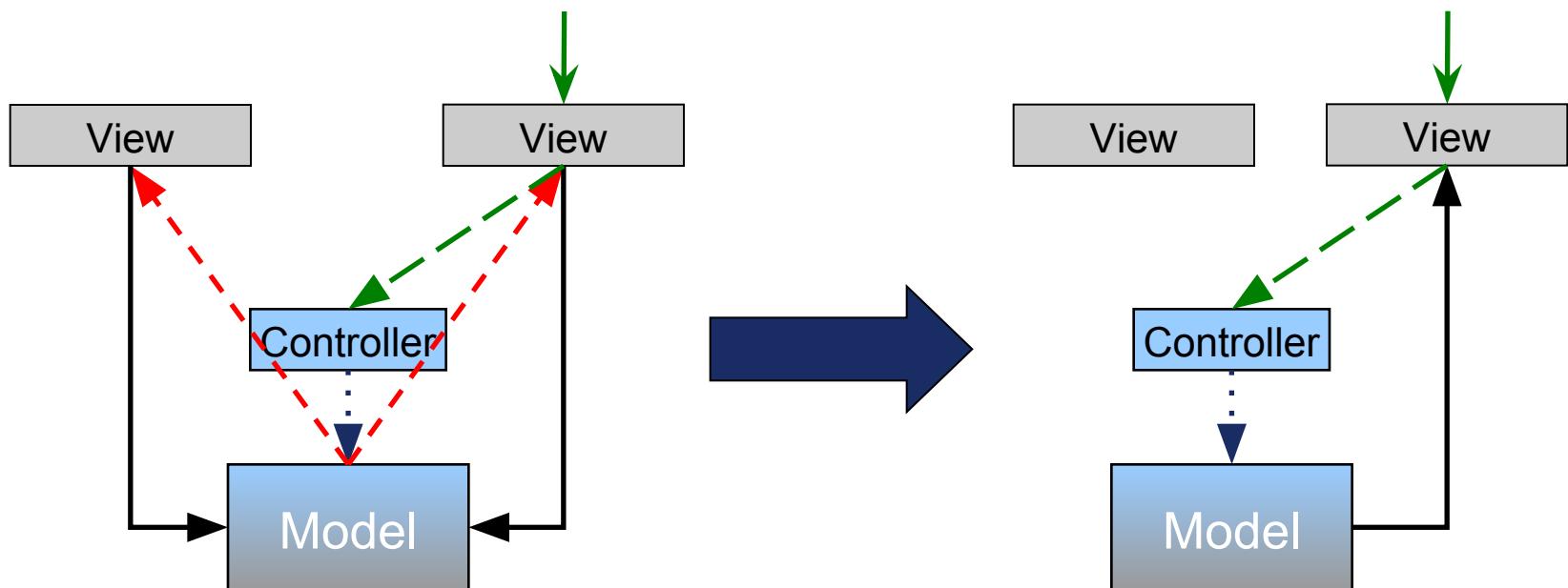
Für Webanwendungen:



### 3.3.5 Model-View-Controller Framework

## Model-View-Controller im Web

- Für Webanwendungen hat MVC einige Besonderheiten
  - ⇒ der View zeigt nicht (wie bei Desktop-Anwendungen) selbst die Seite an, sondern erzeugt lediglich den (HTML-)Code, den ein Browser anzeigt
  - ⇒ weitere Views werden in anderen Browserfenstern gezeigt
  - ⇒ die Benachrichtigung der anderen Views ist über HTTP nicht möglich
    - man hat sich daran gewöhnt, dass eine (bereits geladene) Webseite sich nicht automatisch aktualisiert; dazu muss man die Seite neu anfordern



### 3.3.5 Model-View-Controller Framework

## Model-View-Controller am Beispiel Pizzaservice

### Pizzaservice – Ablauf einer Bestellung

#### ⇒ View(s)

- es gibt einen View (zur Erzeugung der HTML-Seite) für die Bestellung und je einen View für die Statusanzeige für Fahrer, Kunde und Bäcker
- das abgeschickte Formular wird an den Controller geschickt

#### ⇒ Controller

- nimmt Daten an und stellt fest, was diese Daten (derzeit) für das Model bedeuten
- aktualisiert das Model gemäß der übergebenen Daten
- die neue Bestellung wird zur Abspeicherung an das Model übergeben

#### ⇒ Model

- bietet Methoden zum Erledigen der üblichen Aufgaben des Pizzaservices (z. B. Einfügen und Löschen einer Bestellung samt Daten, Ändern des Status, Abfragen von Informationen)
- speichert übergebene Bestelldaten und liefert Daten an den View

The screenshot shows a user interface for a pizza service. At the top, a blue header bar contains the word "Bestellung". Below it is a table listing three pizza types: Margherita (4,00 €), Salami (4,50 €), and Hawaii (5,50 €). To the right of the table is a large, empty white area for displaying the order. At the bottom right of the screen are three buttons: "Alle Löschen" (Delete All), "Auswahl Löschen" (Delete Selection), and "Bestellen" (Place Order).

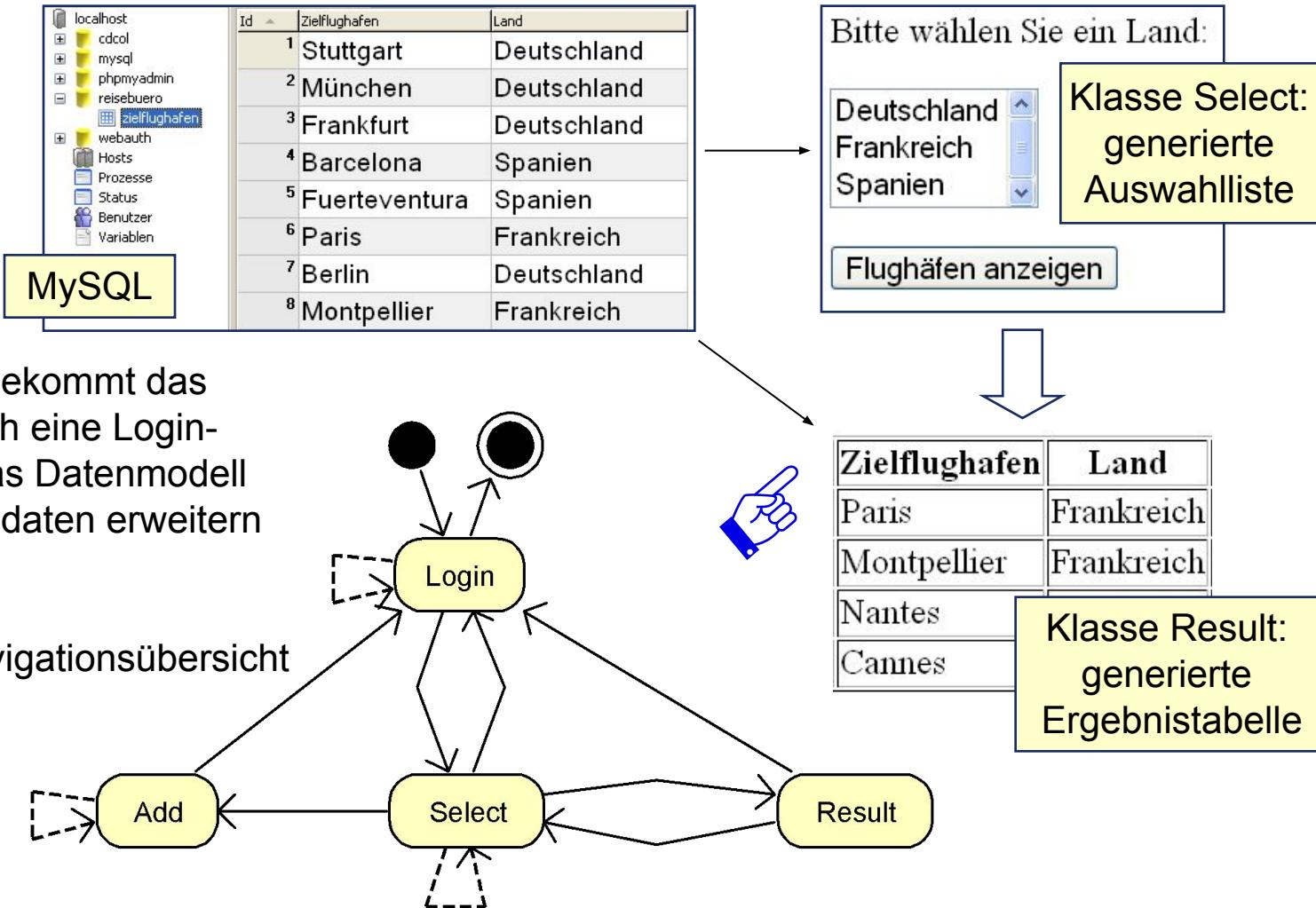
	Margherita	4,00 €
	Salami	4,50 €
	Hawaii	5,50 €

0 €

Alle Löschen  
Auswahl Löschen  
Bestellen

### 3.3.5 Model-View-Controller Framework

## Das bekannte Anwendungsbeispiel nun realisiert als MVC Framework

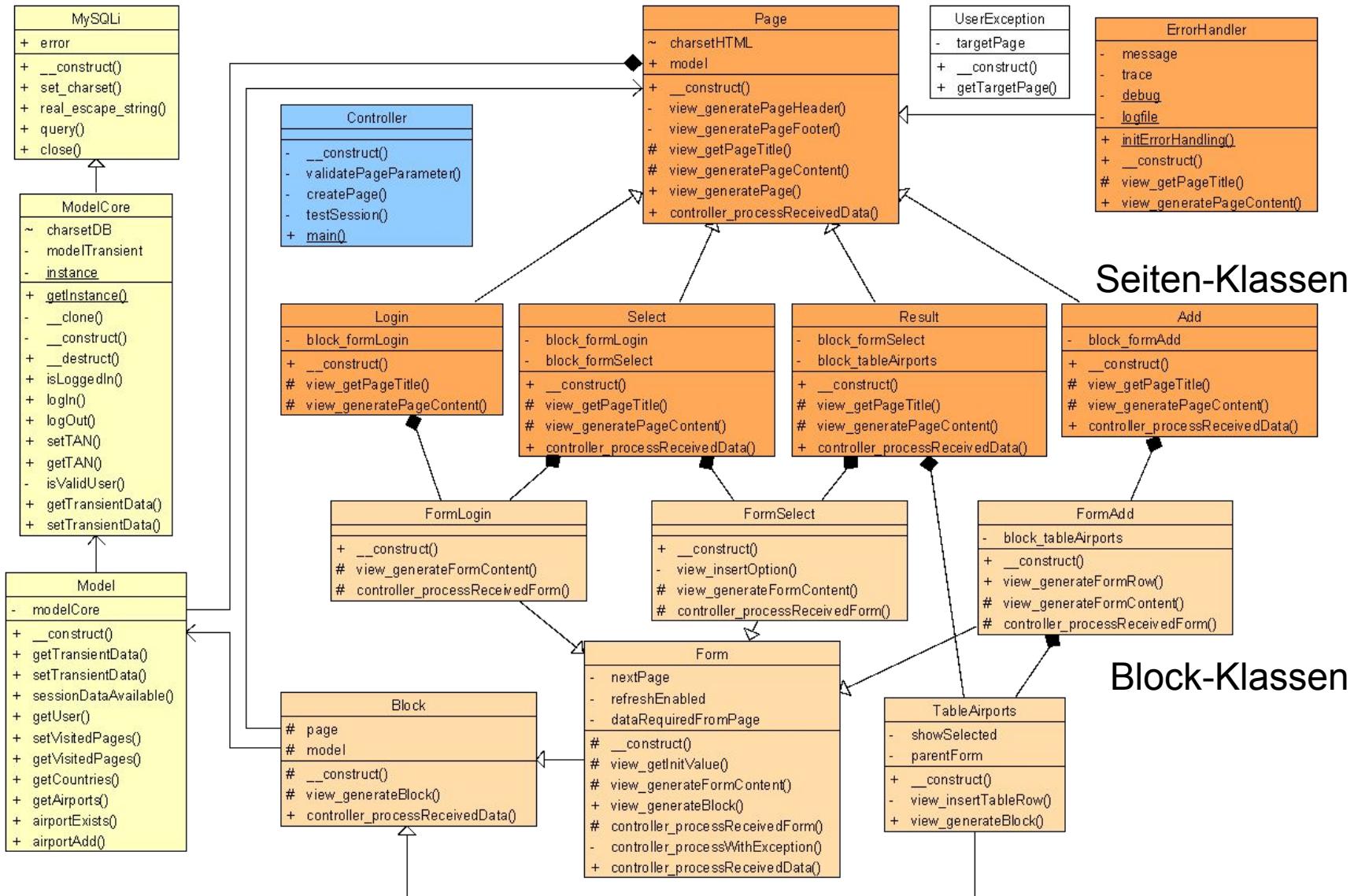


zusätzlich bekommt das Beispiel noch eine Login-Seite um das Datenmodell um Sessiondaten erweitern zu können

Navigationsübersicht

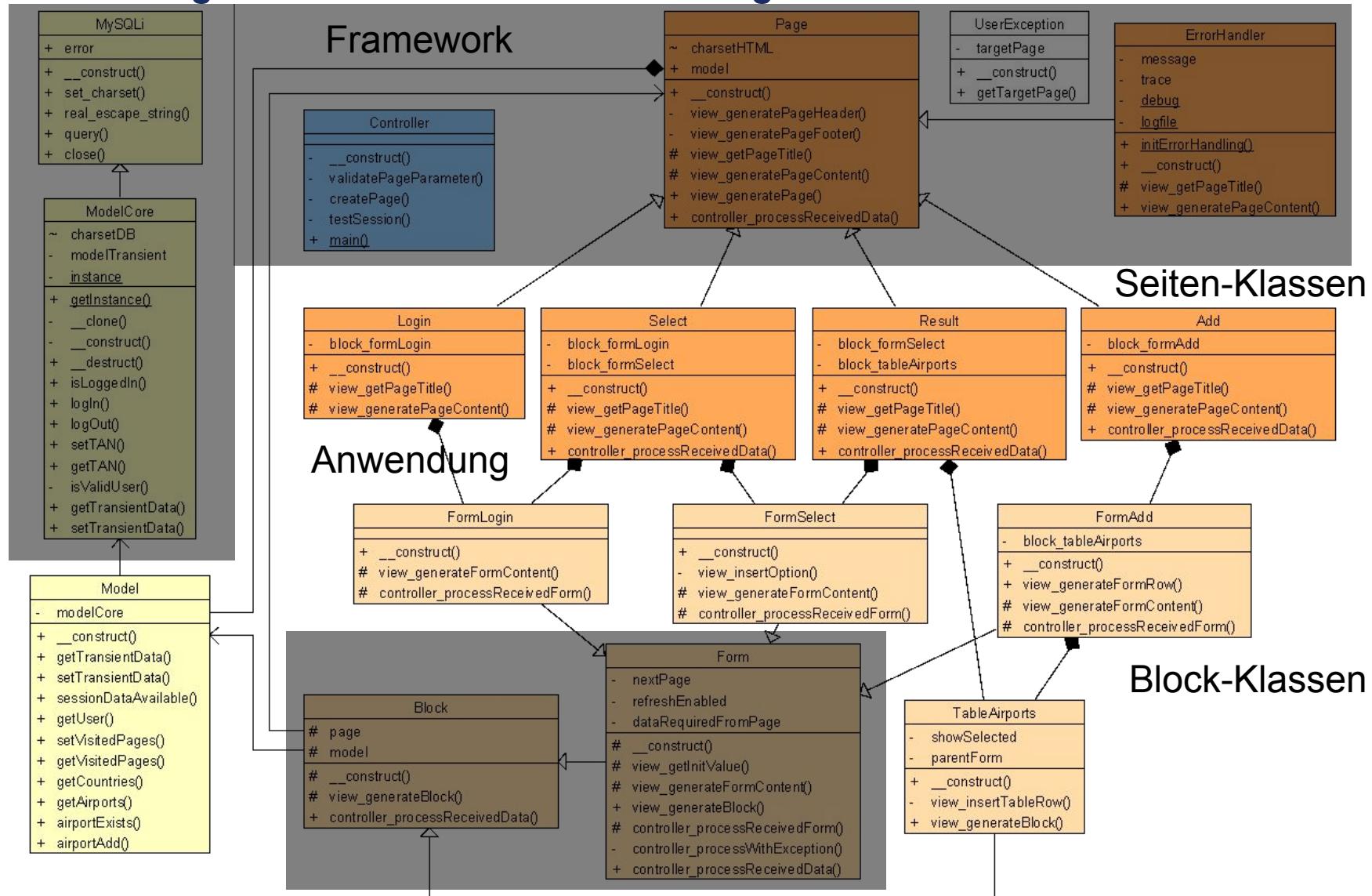
### 3.3.5 Model-View-Controller Framework

## Klassendiagramm des Anwendungsbeispiels



### 3.3.5 Model-View-Controller Framework

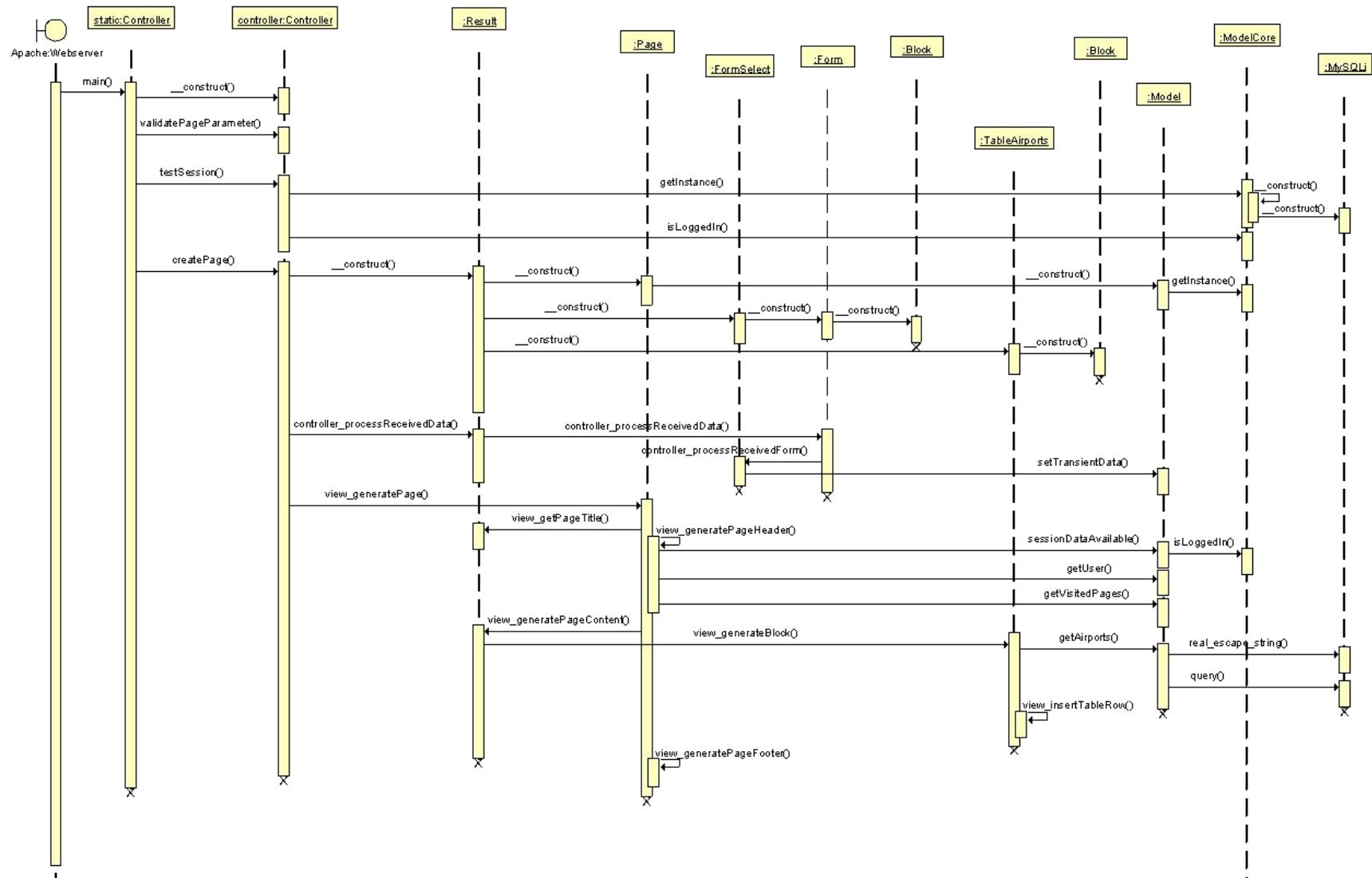
## Aufteilung in Framework und Anwendung



### 3.3.5 Model-View-Controller Framework

## Sequenzdiagramm (Überblick)

selbst das ist nur ein Ausschnitt ...



### 3.3.5 Model-View-Controller Framework

## Controller

- zentraler Einstieg in das Skript
  - ⇒ Aufruf im Browser:  
index.php?page=Add
- kapselt `main()` und die Fehlerbehandlung
- instanziert eine Seiten-Klasse abhängig vom Parameter `page`
- weitere Anteile des Controllers sind verteilt in den Methoden `processReceivedData()` der Seiten- und Block-Klassen
  - ⇒ ggf. Vorverarbeitung von übermittelten Daten



```
final class Controller {
 private function createPage($pageClassName) {
 require_once "./pages/$pageClassName.
php";
 $page = new $pageClassName();
 $page->controller_proce
$page->view_generatePage(),
 }
 public static function main($debug = false) {
 try {
 mb_internal_encoding(Page::
charsetHTML);
 $controller = new Controller($debug);
 $page = $controller->
validatePageParameter();
 if ($controller->sessionIsActive
($page))
 $controller->createPage($page);
 }
 catch (Exception $e) {
 // output error message as a HTML

 $errorHandler = new ErrorHandler($e);
 $errorHandler->view_generatePage();
 }
 }
}
```

page

}

}

Controller.php

hier noch ohne UserException

index.php  
indexTest.php

Controller.main()

Controller.main(true);

## Model: Persistenz und Verfügbarkeit

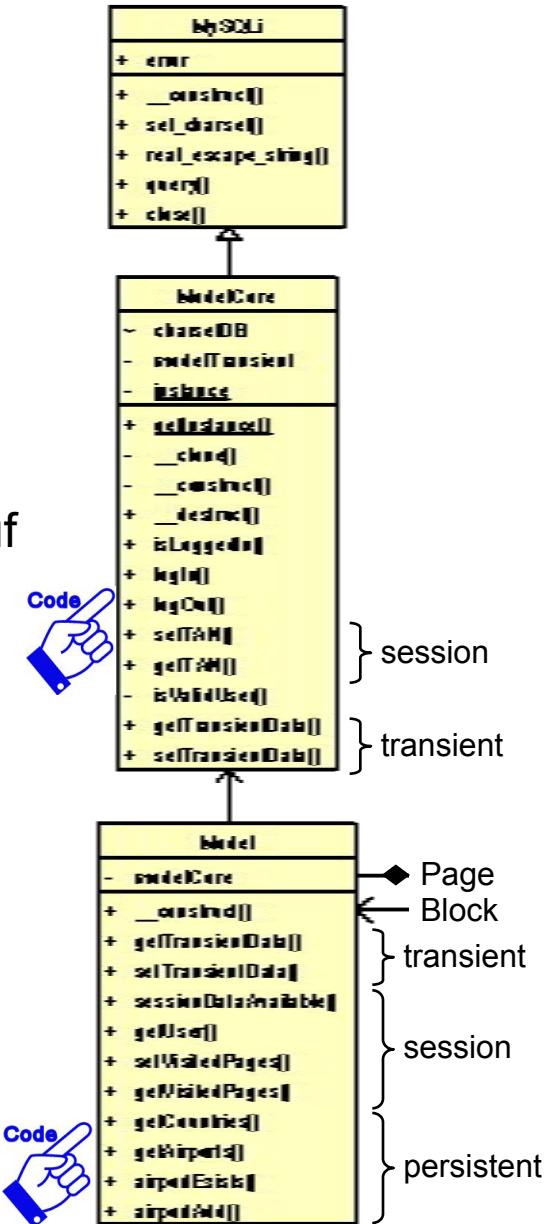
Speicherort

- Man kann das Datenmodell hinsichtlich Persistenz und Verfügbarkeit in drei Bereiche unterteilen:
- Flüchtige lokale Daten (**transient data**)      **Objektattribute**
  - ⇒ gehen verloren nach Ausführung des aktuellen Skripts
  - ⇒ privat für den aktuellen Benutzer
  - ⇒ z.B. Suchparameter einer Abfrage
- Sitzungsdaten (**session data**) **Session Variable**
  - ⇒ begrenzte Haltbarkeit für die aktuelle Sitzung
  - ⇒ privat für den aktuellen Benutzer
  - ⇒ z.B. Kundennummer, Inhalt des Warenkorbs
- Dauerhafte globale Daten (**persistent data**)      **Datenbank**
  - ⇒ unbegrenzte Haltbarkeit
  - ⇒ öffentlich für alle Sitzungen aller Benutzer
    - Beschränkung ggf. in der Anwendung oder der Datenbank
  - ⇒ z.B. Liste der Kunden, Liste aller Bestellungen

### 3.3.5 Model-View-Controller Framework

## Model: das Datenmodell im Detail

- ModelCore ist realisiert als Singleton
  - ⇒ stellt sicher, dass es systemweit nur eine Instanz gibt
  - ⇒ erbt die Datenbankschnittstelle via MySQLi
  - ⇒ verwaltet Login/Logout und die Session Variablen
  - ⇒ kapselt die transienten Daten
- Model ist eine "normale" Klasse mit einer Referenz auf das ModelCore Objekt
  - ⇒ von einem Singleton kann man schlecht erben
  - ⇒ nur Model soll Methoden von MySQLi aufrufen
  - ⇒ wird instanziert vom Konstruktor der Basisklasse Page
    - alternativ können seitenspezifische Unterklassen von Model implementiert und in Konstruktoren von Seiten-Klassen instanziert werden
  - ⇒ eine Referenz auf das aktuelle Model wird über die Konstruktoren an die Klasse Block durchgereicht
    - damit ist das Model in allen Seiten- und Block-Klassen verfügbar



## Model: Weiterverarbeitung von Abfrageergebnissen

- Select-Abfragen über MySQLi liefern ein MySQLi\_Result Objekt
  - ⇒ Variante 1
    - MySQLi\_Result wird in Model::getCountries mittels while-Schleife in ein Array übertragen
    - in FormSelect::view\_generateFormContent wird das Array mittels foreach-Schleife in HTML übertragen
    - Vorteil: MySQLi\_Result ist in Model gekapselt
    - Nachteil: zwei Schleifen (Mehraufwand für Entwickler und Prozessor)
  - ⇒ Variante 2
    - Model::getAirports liefert ein MySQLi\_Result Objekt ab
    - in TableAirports::view\_generateBlock wird das MySQLi\_Result mittels while-Schleife in HTML übertragen
    - Vorteil: nur eine Schleife
    - Nachteil: View wird abhängig von der verwendeten Datenbank (hier MySQLi)
- Kompromiss
  - ⇒ Kapselung von MySQLi\_Result in einer neuen Klasse mit Iterator-Methode und automatischem Aufruf von MySQLi\_Result::free() beim letzten Datensatz

## Model: Objektrelationales Mapping

das verfolgen wir nicht weiter

- bisher werden alle Daten einheitlich über "generische" Klassen verwaltet
  - ⇒ MySQLi\_Result, Array, Assoziatives Array, mysqli\_result::fetch\_object
  - ⇒ wenn man relational denken kann, genügt das meistens auch
- die unterschiedliche Semantik der Daten kann man aber auch besser in die objektorientierte Welt abbilden
  - ⇒ pro Datenbank-Tabelle (bzw. Select-Abfrage / Database View)
    - eine Klasse für die Datensätze (ein Datensatz wird ein Objekt dieser Klasse)
    - eine Containerklasse für die Tabelle (speichert mehrere Datensatzobjekte)
  - ⇒ die Attribute der Datensatzklasse entsprechen den Spalten der Tabelle
    - Fremdschlüssel werden zu Objektreferenzen
  - ⇒ die Methoden sind immer die gleichen
    - für die Datensatzklasse: new, get, set, ...
    - für die Containerklasse: insert, delete, iterate, ...
- "Objektrelationales Mapping" oder ORM
  - ⇒ diese Klassen werden automatisch aus einer Beschreibung des Datenmodells generiert (z.B. XML)
  - ⇒ dann muss "nur noch" die Umsetzung der Geschäftslogik im Model manuell implementiert werden

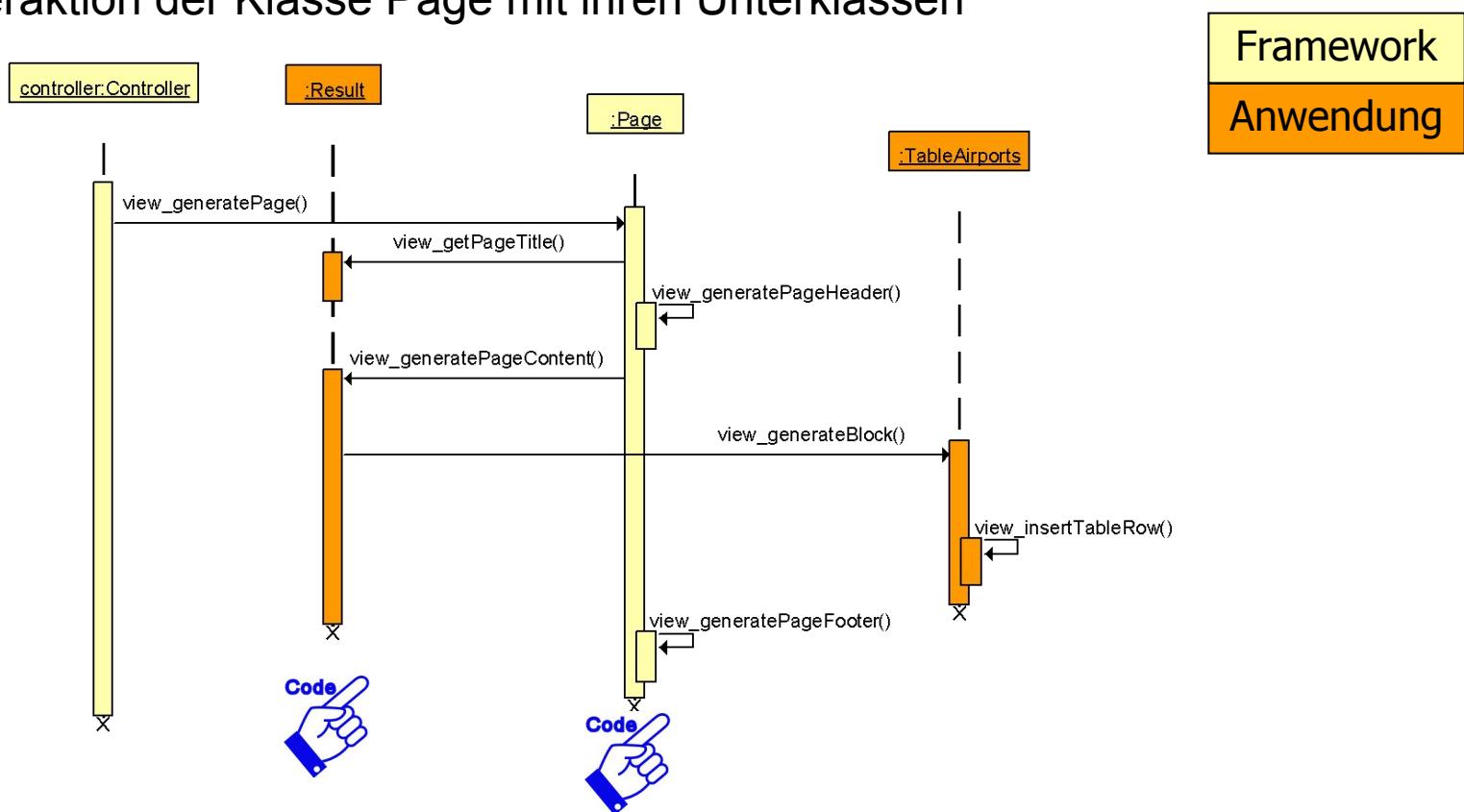


dieses Thema wird  
vertieft im  
Wahlpflichtmodul  
30.2520 Java EE  
Datenbankanwendungs-  
entwicklung

### 3.3.5 Model-View-Controller Framework

## View: aufgeteilt auf Seiten-Klassen und Block-Klassen

- Der View wird implementiert mittels Seiten- und Block-Klassen
  - wie im bisherigen Ansatz
- Das Framework-Prinzip "Inversion of Control" findet sich in der Interaktion der Klasse Page mit ihren Unterklassen



# Hochschule Darmstadt

## Fachbereich Informatik

### 3.3.6 Standardprobleme und Lösungen mit einem Framework



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

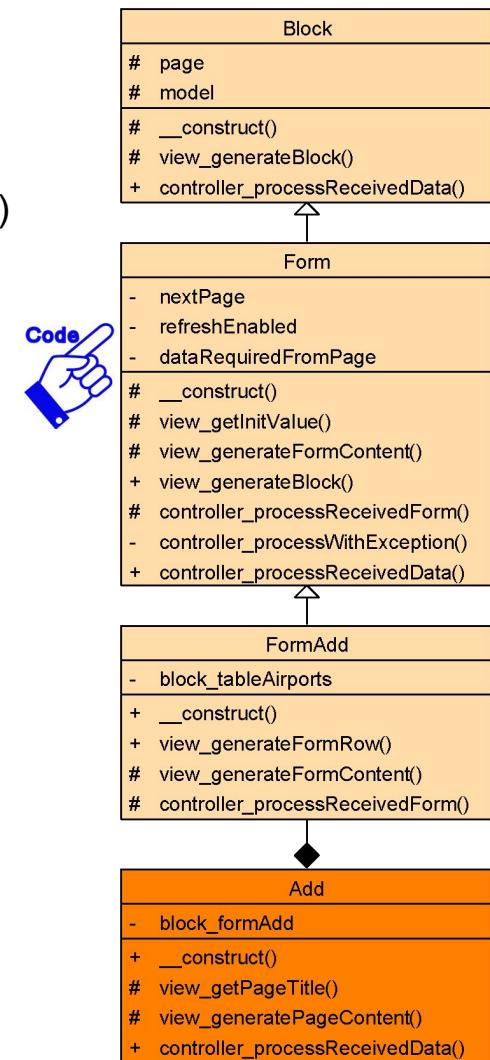
**fbi**

FACHBEREICH INFORMATIK

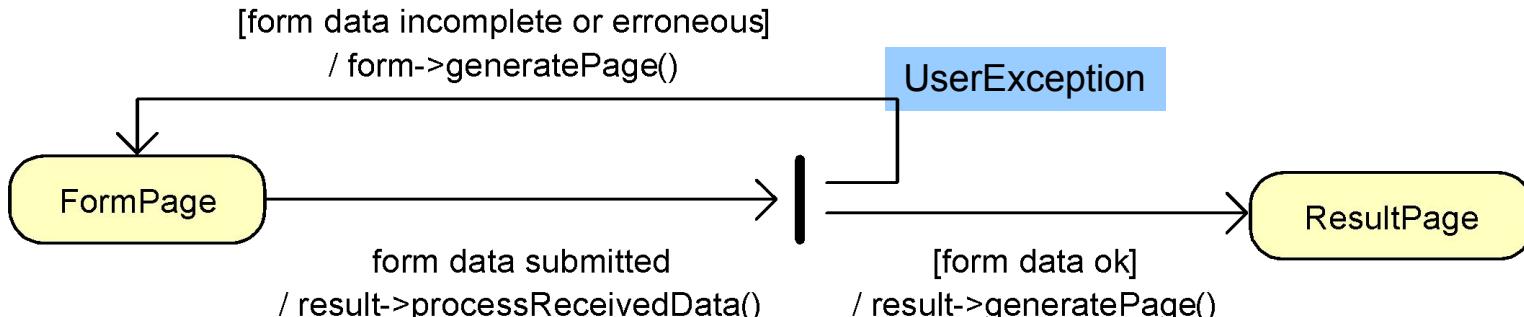
### 3.3.6 Standardprobleme und Lösungen mit einem Framework

## View-Controller: Formular sichern gegen wiederholte Verarbeitung

- Basisklasse Form generiert das <form>-Tag
  - ⇒ mit <input type="hidden" name="TAN">
  - ⇒ mit <input type="hidden" name="formPage"> (⇒ nächste Folie)
- refreshEnabled wählt GET bzw. POST
  - ⇒ false / POST: abgesichert mittels TAN gegen Reload der Seite und Zurück-Button des Browsers
  - ⇒ true / GET: Benutzer kann Lesezeichen auf Suchergebnis setzen; keine TAN
- wiederholte Auswertung wird verhindert durch TAN
  - ⇒ Form::view\_generateBlock erzeugt eine TAN als Zufallszahl
    - speichert die TAN als Session Variable im Server
    - schreibt die TAN verborgen ins Formular (⇒ Browser)
  - ⇒ Form::controller\_processReceivedData vergleicht die vom Browser übermittelte und die gespeicherte TAN
  - ⇒ FormSelect::controller\_processReceivedForm wird nur aufgerufen, wenn beide TANs übereinstimmen
  - ⇒ alle abgeleiteten Form-Klassen erben dieses Verhalten



## View-Controller: Formular sichern gegen Eingabefehler (1)

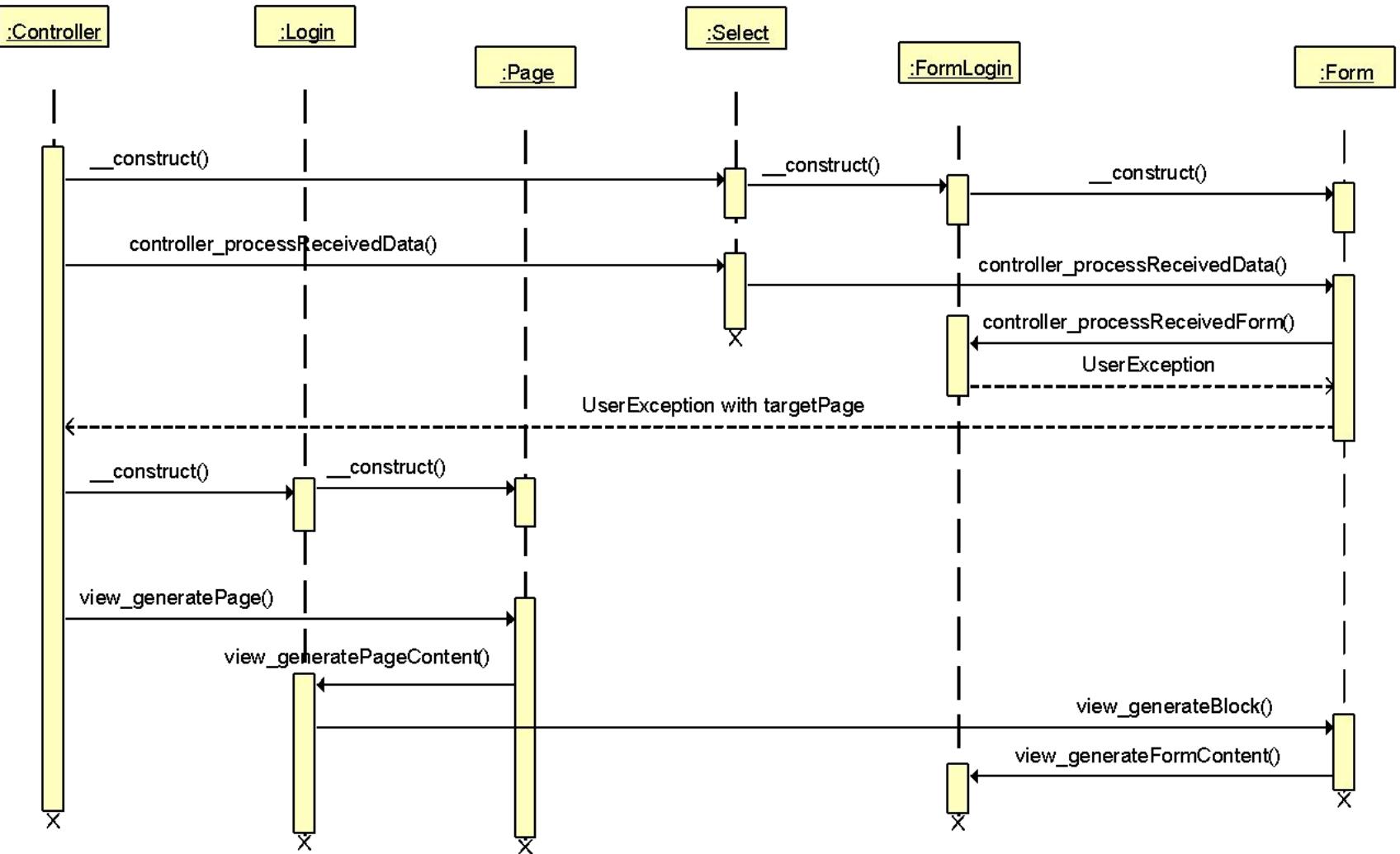
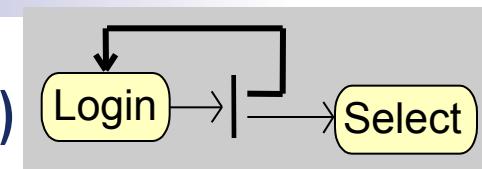


- Ziel: bei unvollständiger oder fehlerhafter Eingabe soll das Formular erneut gezeigt werden, inkl. bereits eingegebener Daten
  - `Form::view_generateBlock` verbirgt den Namen der FormPage im Formular
- der Browser sendet das ausgefüllte Formular immer an die ResultPage
  - `Form::controller_processReceivedData` sichert die übermittelten Daten transient im Model
  - `FormXxx::controller_processReceivedForm` prüft und verarbeitet die übermittelten Daten und wirft im Fehlerfall eine UserException

- ohne UserException:
  - `Controller::createPage` ruft auf `ResultPage::view_generatePage`
- mit UserException:
  - `Form::controller_processWithException` fügt zur UserException den Namen der FormPage hinzu
  - `Controller::createPage` fängt die UserException ein, erzeugt erneut eine FormPage und ruft auf `FormPage::view_generatePage (errmsg)`
  - `FormXxx::view_generateFormContent` initialisiert seine Formularelemente aus dem Model per `Form::view_getInitValue`

*nur dies hat die Anwendung zu tun; den Rest macht das Framework*

## View-Controller: Formular sichern gegen Eingabefehler (2)



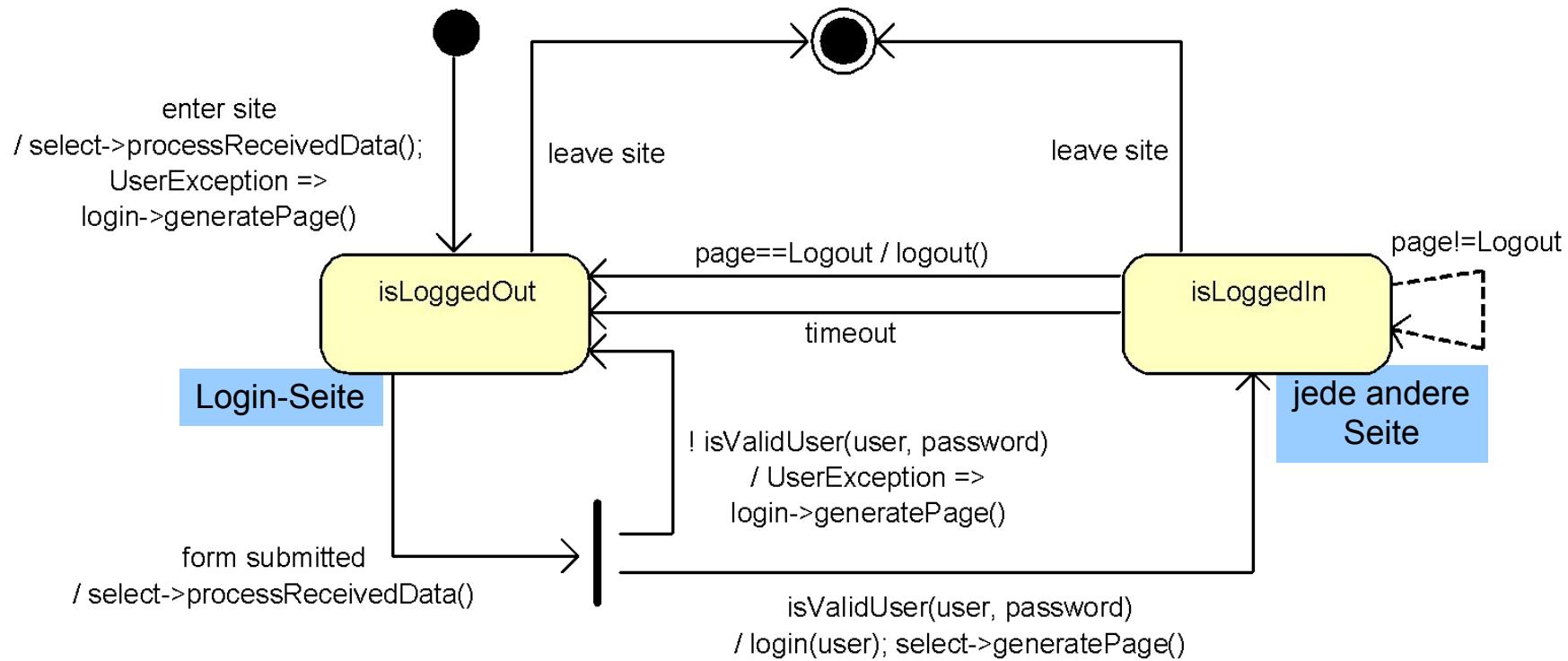
## View-Controller: Login und Sessionverwaltung

siehe Abschnitt  
„Sessionverwaltung“

- Man muss unterscheiden zwischen Session und Login/Logout
  - ⇒ Session: eine Folge von Skript-Aufrufen mit eigenen persistenten Daten
  - ⇒ Login ... Logout: ein Benutzer ist authentifiziert; das ist typischerweise ein Teilabschnitt einer Session
- ModelCore::\_\_construct startet bzw. restauriert die Session mit Hilfe der PHP-Funktion session\_start()
- innerhalb der Session kann man sich einloggen
  - ⇒ unbekannter Benutzername oder falsches Passwort führen zu einer UserException im Login-Formular
  - ⇒ wenn sich ein Benutzer per Login authentifiziert hat, wird sein Benutzername in \$\_SESSION['user'] gespeichert
    - die SessionID wird zur Sicherheit ausgetauscht
  - ⇒ die Existenz von \$\_SESSION['user'] zeigt an, dass ein Benutzer eingeloggt ist
  - ⇒ beim Logout wird \$\_SESSION['user'] gelöscht
    - die Session selbst bleibt aktiv für das nächste Login
    - das TAN-Verfahren in Klasse Form sichert gegen Reload von Accountdaten aus History
- solange kein Benutzer eingeloggt ist, wird nur die Login-Seite gezeigt

### 3.3.6 Standardprobleme und Lösungen mit einem Framework

## View-Controller: Login ... Logout als Zustandsdiagramm



### 3.3.6 Standardprobleme und Lösungen mit einem Framework

## Fehlerbehandlung: Initialisierung



```
class UserException extends Exception { // user vs. system
 public function __construct($message, $targetPage = "")
 // $targetPage may point to prior
 }

final class ErrorHandler extends Page {
 private static $debug = true;
 private static $logfile = "./framework/log.txt";
 public static function initErrorHandler($debug) {
 self::$debug = $debug;

 // activate all available error messages and redirect them to log file:
 ini_set("error_reporting", E_ALL); // report anything
 set_error_handler("error_handler"); // register to convert errors to Exceptions
 if (!self::$debug) { // but in production environment
 ini_set("display_errors", 0); // do not show messages directly
 ini_set("log_errors", 1); // but log them
 ini_set("error_log", self::$logfile); // into this file
 }
 }
}
function error_handler($errno, $errstr, $errfile, $errline) { // global function
 // convert old PHP errors to modern ErrorExceptions
 // this PHP callback function is registered by ErrorHandler::initErrorHandler
 throw newErrorException($errstr, 0, $errno, $errfile, $errline);
}
ErrorHandler::initErrorHandler($debug); // called in Controller::__construct
```



Statische Elemente der Klasse:

- Auswahl Debug / Release
- Fehlerprüfungen aktivieren
- Umlenken in Log-Datei
- PHP Fehler in Exceptions umwandeln

### 3.3.6 Standardprobleme und Lösungen mit einem Framework

## Fehlerbehandlung: Meldung anzeigen

```
final class ErrorHandler extends Page {
 private $message = "";
 private $trace = "";

 public function __construct(Exception $e) {
 parent::__construct(false);
 $this->message = $e->getMessage();
 if (!$e instanceof UserException) { // this is a system error: show trace
 $this->trace = str_replace("\n", "\r\n", $e->getTraceAsString());
 if (!self::$debug) { // write message to log file only; hide it from
 user
 file_put_contents(self::$logfile,
 "\r\n$this->message\r\n$this->trace\r\n\r\n", FILE_APPEND);
 $this->message = "Es ist ein Fehler aufgetreten.";
 $this->trace = "";
 }
 }
 }

 public function view_generatePageContent() {
 echo "<h1>Fehler</h1>\n";
 echo "<p>$this->message</p>\n";
 if ($this->trace)
 echo "<pre>$this->trace</pre>\n";
 }
}
```

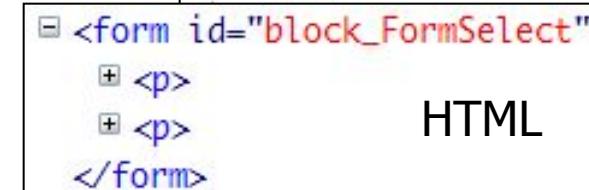
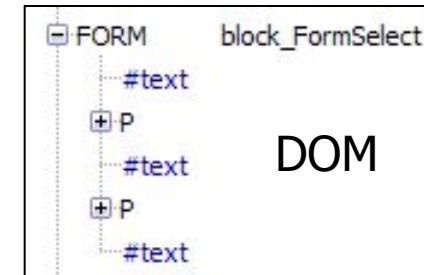
Klasse erbt von Page

- zeigt Meldung und verfolgt die Aufrufe zurück
- schreibt in Log-Datei
- volle Anzeige im Debug; sonst Details nur in Log
- Ausgabe von validem HTML

Eine anwendungsorientierte Log-Datei ist extrem wichtig, damit der Betreiber der Seite eventuell auftretende Probleme überhaupt mitbekommt

## Wartung: Finde die richtige Klasse

- Beginne am Graphical User Interface – an der Webseite
  - ⇒ identifiziere den Ort des Problems bzw. die zu erweiternde Stelle
- Rufe die Seite im Browser auf
  - ⇒ finde die Seiten-Klasse über die URL  
`http://www.xyz.de/index.php?page=Result`
- inspiziere den Block mittels rechtem Mausklick
  - ⇒ verwende Mozilla Add-ons DOM Inspector oder Firebug
  - ⇒ finde die Block-Klasse über die HTML id:  
`<form id="block_FormSelect" ...`



- das erfordert strenge Namenskonventionen:
  - ⇒ page-Parameter = Klassenname = Dateiname
  - ⇒ block id = Klassenname = Dateiname

## Typische Funktionen von Web-Frameworks

- Vorgabe der Systemarchitektur
  - ⇒ meist Model-View-Controller
  - ⇒ ereignisorientierte Programmierung
- Anbindung von Datenbanken
  - ⇒ ORM Object-Relational Mapping
  - ⇒ Datenbankabstraktionsschicht
- Entwicklerkomfort
  - ⇒ Authentifizierung der Benutzer
  - ⇒ Validierung von Eingaben
  - ⇒ Generierung von HTML-Seiten mittels Schablonen ("Templates")
  - ⇒ Vermeidung wiederholter Neugenerierung von Seiten ("Caching")
  - ⇒ Unterstützung für Ajax
  - ⇒ Integrierte Entwicklungsumgebung

Das haben praktisch alle Web-Frameworks gemeinsam

Hier unterscheiden sich die Frameworks wirklich

## Freie und kommerzielle Frameworks

### ■ Kleine Auswahl - ohne Anspruch auf Vollständigkeit:

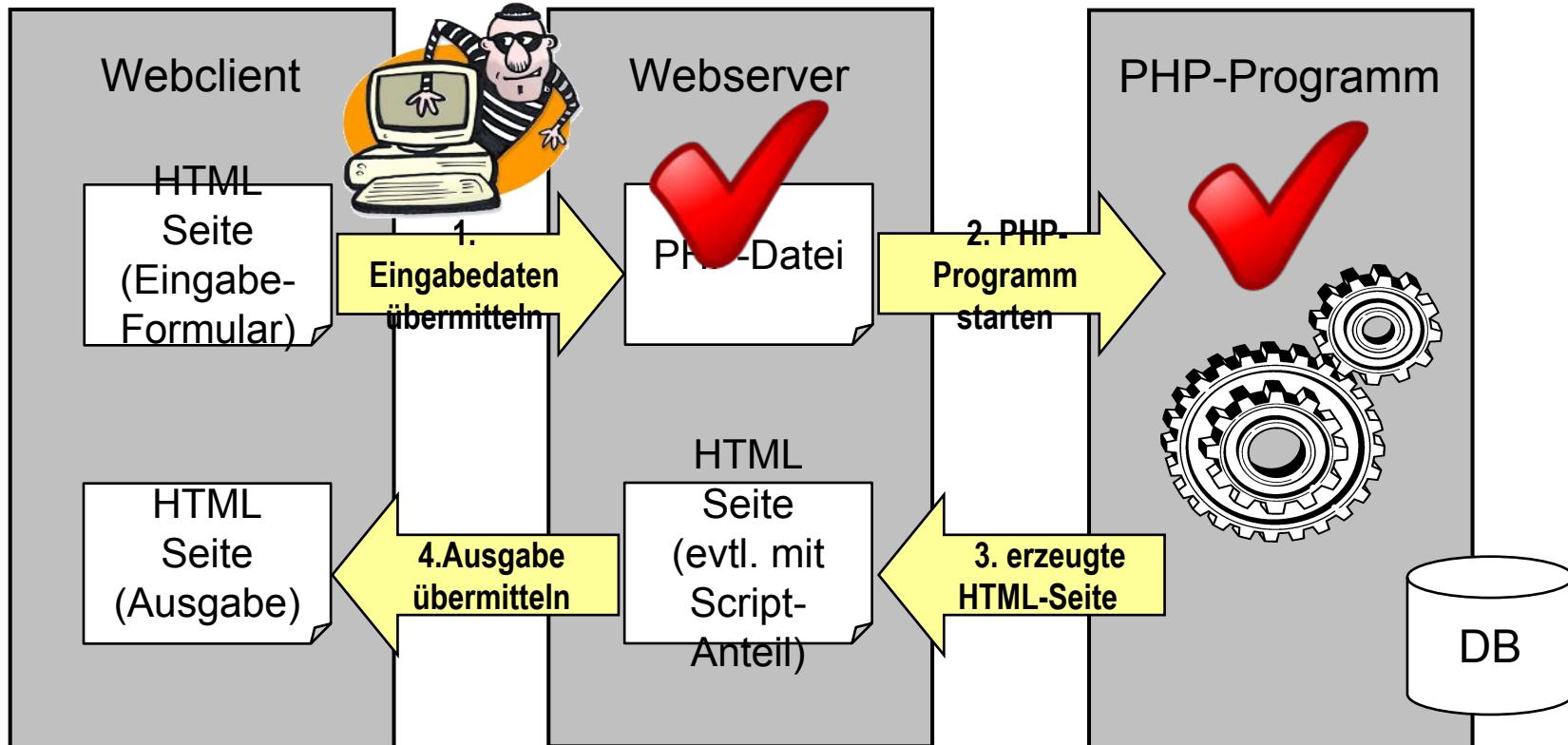
- ⇒ Symfony
  - orientiert an Rails, reichhaltige Features
- ⇒ Zend Framework
  - Komponentenorientiert, objektorientiert, mächtig, mit vielen Freiheitsgraden
- ⇒ CakePHP
  - Rails Adaption
- ⇒ Prado
  - orientiert an ASP.NET, ereignisorientiert

Referenzmodell: "Konvention vor Konfiguration" vgl. Ruby on Rails

Welches Framework Sie auch verwenden – wenn Sie MVC und ORM verstanden haben, fällt Ihnen die Einarbeitung leichter

### 3. Webserver

## Übersicht



- HTML
- CSS
- ECMA-Script
- DOM
- AJAX

- HTTP

- Server-Konfiguration

- CGI
- PHP
- MySQLi
- Seitenklassen
- Frameworks

## Zusammenfassung: Zeichenkodierung – wo überall festlegen?

- Dateikodierung im Editor einstellen auf UTF-8 ohne BOM
- HTTP Header per PHP für die Übermittlung Server → Client
  - ⇒ `header("Content-type: text/html; charset=UTF-8");`
- HTML (und ggf. XML) Header für die Interpretation durch den Browser
  - ⇒ `<meta charset="UTF-8" />`
  - ⇒ nur falls XML: `<?xml version="1.0" encoding="UTF-8" ?>`
- im HTML Formular für die Übermittlung Client → Server
  - ⇒ `<form action="..." method="post" accept-charset="UTF-8">`
- ECMAScript XMLHttpRequest sendet und erwartet standardmäßig UTF-8
  - ⇒ *keine besonderen Maßnahmen erforderlich*
- PHP Strings sind standardmäßig Singlebyte Strings
  - ⇒ Kodierung für Multibyte Strings setzen: `mb_internal_encoding("UTF-8");`
  - ⇒ spezielle Funktionen für multi byte strings verwenden: `mb_...()`
- beim Anlegen der Datenbank per SQL
  - ⇒ `CREATE DATABASE `db` DEFAULT CHARACTER SET utf8  
COLLATE utf8_unicode_ci;`
  - ⇒ `CREATE TABLE `table` (...) DEFAULT CHARSET=utf8;`
- nach Verbinden mit der DB für die Verbindung PHP ←→ MySQL
  - ⇒ `$mysqli->set_charset("utf8");`

# Hochschule Darmstadt

## Fachbereich Informatik

### 4. Zwischen Webclient und Webserver



**h\_da**

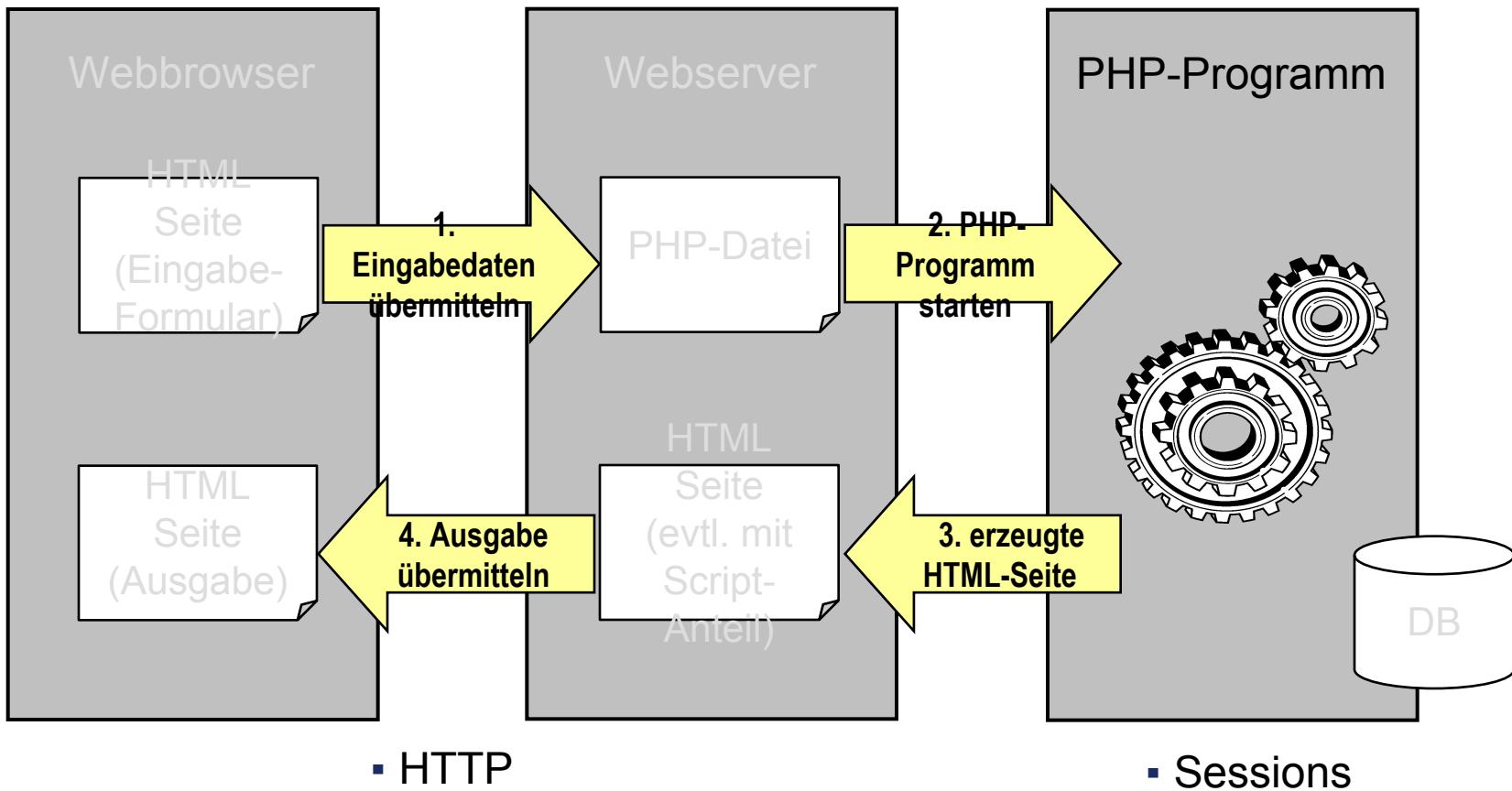
HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

#### 4. Zwischen Webclient und Webserver

## Zwischen Webclient und Webserver



# Hochschule Darmstadt

## Fachbereich Informatik

### 4.1 HTTP



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

# 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:  
**GET SP http://www.xyz.de/datei.html SP HTTP/1.1 CRLF**
  - ⇒ Server ⇒ Browser:  
**HTTP/1.1 SP 200 SP OK CRLF**  
**Content-Type: text/html CRLF CRLF**  
**<!DOCTYPE html><html lang="de">...</html>**

# HTTP-Nachrichten

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

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



## Request-Line (Methode, URI, Version)

- 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)

```
GETSP/index.htmlSPHTTP/1.1CRLF
```



## 4.1 HTTP

## Response-Line (Version, Status, Reason)

### ■ HTTP Statusmeldungen

Antwort vom Server in Response-Line:

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

Status für  
Menschen

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

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

## 4.1 HTTP

# 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/html**CRLF

## ■ 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-cache**CRLF

## 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



## 4.1 HTTP

## HTTP Header (3)

### ■ Response Header

- ⇒ Server

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

- ⇒ WWW-Authenticate

- verlangt vom Client eine Authentifizierung

## 4.1 HTTP

# 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  $\Rightarrow$  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  $\Rightarrow$  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 kommt's</p>

## Fazit

- HTTP ist ein einfaches Frage-Antwort-Protokoll
  - ⇒ zur Übertragung von Daten in einem Netzwerk
  - ⇒ Es wird hauptsächlich im WWW zur Kommunikation zwischen Webserver und Webclient eingesetzt
  - ⇒ HTTP setzt auf einem Transportprotokoll auf (im Normalfall TCP)
- 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
  - ⇒ einen Bezug zwischen Aktionen muss die Anwendung bei Bedarf herstellen (z.B. SessionIDs für einen Warenkorb)
  - ⇒ Angriffe (z.B. automatische Login-Versuche) sind schwer zu erkennen

Wenn Sie HTTP mal "live" sehen wollen:  
Firefox Addon "Live HTTP headers" installieren

# Hochschule Darmstadt

## Fachbereich Informatik

### 4.2 Sessionverwaltung



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

## 4.2 Sessionverwaltung

### Beispiel (Prinzip)

Ob BOL tatsächlich diese Seite so erzeugt, ist nicht bekannt

### Beispiel: Warenkorb (bei BOL)

Mit PHP erzeugte HTML-Seite

Der Warenkorb enthält diese Positionen:

software	 <b>WISO Sparbuch 2005</b> Versandfertig innerhalb von 24 Std.	1	Einzelpreis EUR 28,88	EUR	<b>28,88</b>	<input type="button" value="auf Merkliste"/>
			Lieferkosten	EUR	0,00	<input type="button" value="löschen"/>

Bitte beachten Sie, dass **für Geschenksendungen und Sendungen außerhalb Deutschlands** weitere Kosten entstehen können.

Falls Sie die Bestellmenge ändern, klicken Sie anschließend auf "aktualisieren", um die Zwischensumme neu berechnen zu lassen.



Eintrag aus einer Datenbank

erste Überprüfung der Eingabe mit ECMA\_Script

Layout mit CSS

Aber woher weiß der Webserver welcher Warenkorb zu welchem Kunden gehört?

## 4.2 Sessionverwaltung

# Zusammenhängende Webseiten - Beispiel

- Formular-Folge mit mehreren Seiten  
(z.B. Shop, Warenkorb, Bestellung,...)
- Was passiert, wenn die URL parallel in mehreren Browsern geöffnet wird?
- Woher weiß der Webserver / ein CGI-Skript, dass die Aufrufe zusammen gehören?

The screenshot illustrates a session involving three pages of a shopping website:

- Product Details Page:** Shows the book "Per Anhalter durch die Galaxis" by Douglas Adams. It includes a thumbnail, a short description, and a "Jetzt bestellen" button.
- Shopping Cart Page:** Displays the item from the cart with a summary table. The table shows:

Position	Produkt	Einzelpreis EUR	Gesamtpreis EUR	Aktionen
1	Per Anhalter durch die Galaxis	7,95	7,95	auf Merkliste lösen
- User Login Page:** A form for new users to register, asking if they have an account and providing fields for username and password.

## Die Problematik

- 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?
- jeder Aufruf eines CGI-Skripts ist ein neuer Aufruf eines selbständigen Programms
  - ⇒ idealerweise in einem eigenen Prozess
  - ⇒ CGI-Skripten können keine Daten über globale Variable austauschen oder in solchen retten
  - ⇒ nur persistente Speicher sind brauchbar: Dateien / Datenbanken
  - ⇒ Wie kann man Daten zwischen Aufrufen austauschen?
  - ⇒ ...der Bezug kann nur über den Aufruf hergestellt werden!

### Grundidee

- Verwende einen Identifier, der zwischen Client, Webserver und CGI-Anwendung ausgetauscht wird und simuliere damit eine feste Verbindung !
  - ⇒ Erzeugung durch den Webserver / die CGI-Anwendung
  - ⇒ Speicherung beim Client oder (temporär) als Parameter in der URL
  - ⇒ Zusätzliche Übertragung der ID bei jedem Aufruf dieser URL
- Sinnvolle Unterstützung
  - ⇒ Webserver:
    - Methoden zum Erstellen, Löschen und Verwalten solcher IDs
    - Methoden zum Verwalten der Daten einer Verbindung
  - ⇒ Webclient:
    - Abspeicherung und Übertragung der IDs



### 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

persistent bis zum Ende der Session; Zugriff beschränkt auf diese Session

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

in jedem Fall  
leicht manipulierbar

## HTTP Cookie

Server speichert Daten clientseitig  
(nicht nur für Sessionverwaltung)



- HTTP Cookies sind (evtl. dauerhaft) vom Webclient gespeicherte Daten
  - ⇒ sind einer bestimmten Website zugeordnet
  - ⇒ können von Client und Server gelesen und geschrieben werden
  - ⇒ Der Webclient sendet die Daten bei Zugriffen auf eine Webseite an den zugehörigen Webserver
- Cookies werden gesetzt
  - ⇒ per Meta-Tag `<meta http-equiv="set-cookie" content="Keks=Wert; expires=friday, 31-dec-09 23:59:59 gmt;" />`
  - ⇒ per HTTP-Header (gemäß RFC 2109 / RFC 2965)  
`Set-cookie: Keks=Wert; domain=h-da.de;`  
`expires=friday, 31-dec-09 23:59:59 gmt;`
  - ⇒ oder auch mit JavaScript über HTMLDocument:  
`document.cookie = "nochnKeks=xy";`

## Cookies unter PHP

natürlich nicht nur für  
Sessionverwaltung

### Funktionsdeklaration

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

### Cookie setzen

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

Serverzeit

⇒ verfällt nach 1 Stunde

Per HTTP-Header!  
Vor der ersten Ausgabe

### Cookie löschen

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

⇒ mit denselben Parametern wie beim Setzen !

⇒ Verfallszeit in der Vergangenheit bewirkt sofortiges Löschen

### Cookie-Wert auslesen

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

# Cookies beschränken und teilen

URL, die das Cookie setzt, z.B.: <https://obs.fbi.h-da.de/obs/index.php>

- beschränken auf Domäne
  - ⇒ default: Server Name der URL obs.fbi.h-da.de
  - ⇒ erweitern für alle Subdomänen .fbi.h-da.de ⇒ www.fbi.h-da.de etc.
- beschränken auf Pfad (unterhalb der Domäne)
  - ⇒ default: Pfad der setzenden URL/obs/
  - ⇒ alle Pfade der Domäne /
  - ⇒ anderer Pfad /mhb/
- Cookies von Drittanbietern (third-party cookies)
  - ⇒ Beispiel: eine Webseite holt ein Werbebanner von einer anderen Domäne; das Werbebanner kann ein Cookie setzen
  - ⇒ solche Cookies können i.a. im Browser separat gesperrt werden
- Cookie "Sharing" über kooperierende Top Level Domains
  - ⇒ jeder Server setzt sein eigenes Cookie, jedoch mit demselben Wert
  - ⇒ die Server übergeben sich den Wert als Parameter von Redirects

## SessionID mit Cookies übertragen

- Der Webserver übergibt dem Webclient eine SessionID zur Abspeicherung als Cookie
  - ⇒ normalerweise per HTTP-Header

```
Set-cookie: Session=081512345678; domain=h-da.de;
expires=friday, 31-dec-09 23:59:59 gmt;
```
- Der Webclient liefert das Cookie ab sofort bei allen Aufrufen dieser Webseite mit
  - ⇒ CGI kann Umgebungsvariable HTTP\_COOKIE abfragen  
`HTTP_COOKIE=Session=081512345678; nochnKeks=xy`
  - ⇒ der Webserver / CGI "weiß" welcher Client zugreift
- Problem: Benutzer kann Cookies abschalten
  - ⇒ Anwendung kann sich also nicht darauf verlassen und sollte – wenn möglich – auch ohne Cookies funktionieren

## SessionID ohne Cookies

nur verfügbar mit session.  
use\_only\_cookies=Off

- optimal bei HTML-Formularen
  - ⇒ Verstecktes Formularelement mit generierter ID
  - ⇒ `<input type="hidden" name="SessionID" value="081512345678">`
  - ⇒ 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=081512345678">`
  - ⇒ Client schickt dieses per GET sichtbar zurück

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

## SessionID ohne Cookies

- optimal bei HTML-Formularen
  - ⇒ Verstecktes Formularelement mit generierter ID
  - ⇒ `<input type="hidden" name="SessionID" value="081512345678">`
  - ⇒ 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=081512345678">`
  - ⇒ Client schickt dieses per GET sichtbar zurück
  - ⇒ gefährlich: Besucher der Seite könnte die SessionID versehentlich per Link an andere weitergeben. Der Empfänger kann damit die Session öffnen!

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

## PHP Sessionverwaltung

- generiert und übermittelt SessionID
  - per Cookie oder URL
- sichert und restauriert persistente Variable
  - in Datei oder DB
- Session eröffnen (Login) bzw. restaurieren (Folgeseiten)
  - ⇒ am Anfang des PHP-Programms: `session_start();`
- persistente Variable registrieren
  - ⇒ `$_SESSION["Zustand"] = 5;`
  - ⇒ am Programmende werden persistente Variable autom. gesichert
- danach Zugriff auf persistente Variable
  - ⇒ `$myID = $_SESSION["Zustand"];`
- Session beenden (Logout)
  - ⇒ `session_destroy();`
  - außerdem bei Bedarf:  
`$_SESSION = array(); // löscht Daten`  
`setcookie(...); // Cookie löschen`

# PHP Sessionverwaltung mit Cookies – ein Problem (und eine Lösung)

- Beim Öffnen einer Session wird festgelegt, wie lange die Session gültig bleibt
  - ⇒ default: so lange der Browser geöffnet bleibt
  - ⇒ Änderung auf eine Lebensdauer: vor `session_start()` - über `session_set_cookie_params(int $lifetime_in_sec)`
- Aber Vorsicht
  - ⇒ auch wenn die Lebenszeit relativ angegeben ist - der Wert für das Cookie wird aus der Uhrzeit des Webservers und der Lebenszeit berechnet und auf dem Client absolut gesetzt!
  - ⇒ Wenn die Uhrzeit auf dem Client falsch gesetzt ist, kann es sein, dass das Cookie sofort ungültig ist bzw. länger gilt als gewünscht
- Lösung
  - ⇒ Das Cookie wird ohne Einschränkung ausgeliefert und übertragen
  - ⇒ Der Server entscheidet selbst, ob eine Session noch gültig ist
  - ⇒ Konfigurierbar in der PHP.ini über `session.gc_maxlifetime`

# Sessions und Login



- beim (erfolgreichen) Login

```
session_start();
if ... isset($_POST['Username']) && preg_match(...) {
 $_SESSION["User"] = $_POST['Username'];
 $_SESSION['LastAccess'] = time();
}
```

- beim Zugriff

```
session_start();
if (isset($_SESSION['LastAccess']) &&
 time()-$_SESSION['LastAccess']<SessionTimeoutSec)) {
 $User = $_SESSION["User"];
 $_SESSION['LastAccess'] = time(); // Inaktivitätsdauer = 0
 $SQLabfrage = ... WHERE User ="$User";
}
```

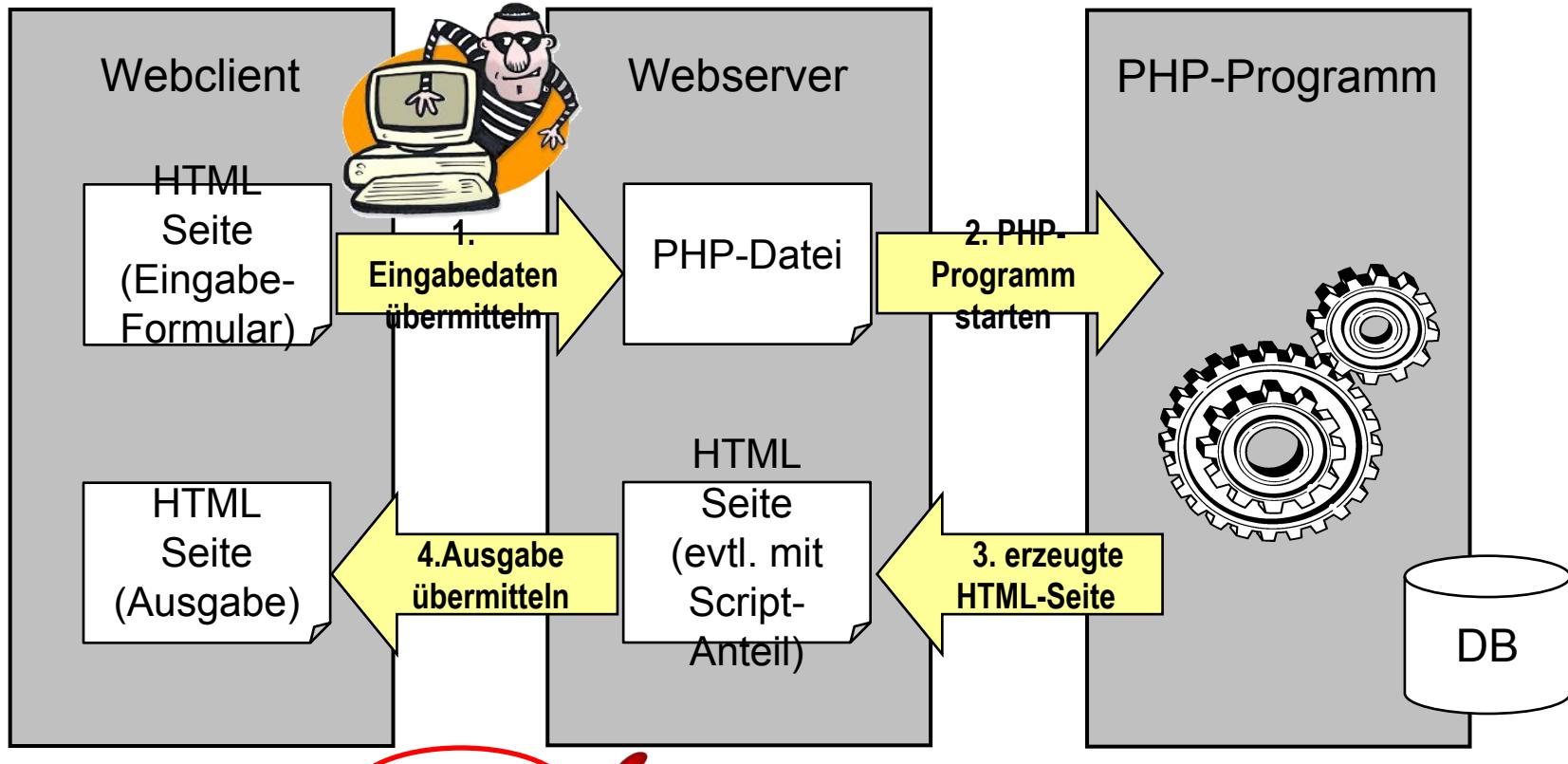
# Sessions und Cookies im Browser

- Browser speichern eine Vielzahl von Cookies
  - ⇒ darunter auch session-id's uvm.
  - ⇒ das ermöglicht kundenspezifische Empfehlungen „Diese DVD könnte Sie interessieren...“
  - ⇒ aber auch eine erschreckende Transparenz z. B. Werbung auf anderen Websites, die genau das bietet, was Sie gestern gesucht haben...



## 4. Zwischen WebClient und Webserver

### Übersicht



- HTML
- CSS
- ECMA-Script
- DOM
- AJAX

- HTTP

- Server-Konfiguration

- CGI
- PHP
- MySQLi
- Sessions

# Hochschule Darmstadt

## Fachbereich Informatik

### 5. Sicherheit



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

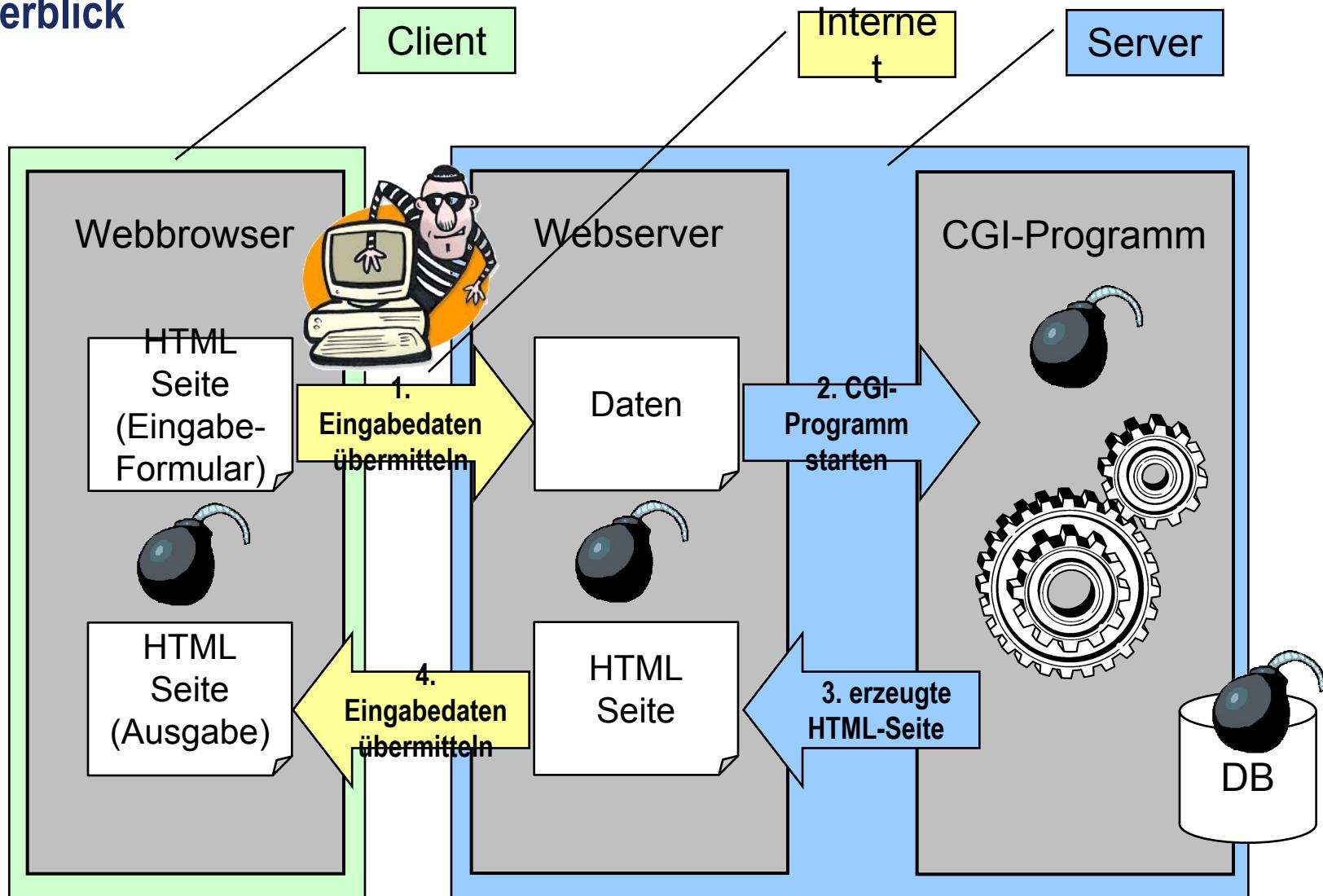
## Vorbemerkung

- Dieser Abschnitt bietet lediglich einen Einstieg in das Thema
  - ⇒ es werden ständig neue Sicherheitslücken und Angriffsformen entdeckt
  - ⇒ es gibt große Unterschiede zwischen den Betriebssystemen, Browsern, Webservern etc.
  - ⇒ der Sicherheitsbedarf hängt von der Anwendung ab
- Für den Betrieb einer professionellen Webanwendung ist Sicherheit ein Dauerthema
  - ⇒ wenn Sie das nicht leisten können, mieten Sie sich einen Server mit einem entsprechenden Update-Service für Betriebssystem, Apache, PHP, DB etc.
  - ⇒ dann müssen Sie "nur" noch kontinuierlich Sicherheitslücken in Ihrer Anwendung beseitigen

**Sie sollen hier für das Thema  
Sicherheit sensibilisiert werden !**

## 5. Sicherheit

# Überblick



## Datenübertragung mit GET

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

Name:	Hahn
Text:	blabla
<input type="button" value="absenden"/> mit GET	

- 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 (fehlende oder zusätzliche)
  - ⇒ unerwartete Werte

# Datenübertragung mit POST

Name:  
Hahn

Text:  
Blabla

mit POST

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



- Parameter werden im HTTP Body übertragen
- Formularfelder sind über den HTML-Code zugänglich
- Mit etwas HTML-Kenntnissen können Werte und Parameter manipuliert werden
  - ⇒ unerwartete Parameter (fehlende oder zusätzliche)
  - ⇒ 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 nicht erwarteter wird übermittelt
  - ⇒ 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

## Formularparameter grundsätzlich validieren

ereg vermeiden; ist nicht binary safe und deprecated

### ■ Wo ?

- ⇒ serverseitig zur Sicherheit
- ⇒ clientseitig (JavaScript) nur als Benutzerkomfort
  - der Benutzer könnte JavaScript abgeschaltet haben;  
ein Hacker könnte dies gezielt tun

### ■ Problem der Vollständigkeit

- ⇒ die gültigen Zeichenfolgen spezifizieren (z.B. über regulären Ausdruck)
- ⇒ nicht die ungültigen, sonst würden vergessene Fälle akzeptiert

### ■ Achtung

- ⇒ Der PHP-Befehl **ereg()** (zur Ersetzung mit regulären Ausdrücken)  
ist nicht binärsicher, d.h. er sucht nur bis zur ersten '\0' nach  
Übereinstimmungen mit einem regulären Ausdruck



## Angriffe allgemein

- Es gibt viele Installationen mit Standardkonfiguration (Webserver, Datenbank, CGI, PHP,... vgl. XAMPP)
  - ⇒ Webserver geben großzügig Auskunft über die Versionen
  - ⇒ Default-Passwörter sind bekannt
  - ⇒ installierte Skripte sind teilweise bekannt
  - ⇒ Quellcode ist verfügbar
  - ⇒ Sicherheitslücken können gesucht und erprobt werden
- Angriffsziele (Auswahl):
  - ⇒ 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 oder aus Datenbank)
  - ⇒ ...
- Lahmlegen des Servers
  - ⇒ z.B. durch Überlastung mit Requests ("Denial of Service Attack")
  - ⇒ Löschen von Dateien
  - ⇒ Passwörter nicht im Klartext speichern: Hash verwenden

**gewissenhaft und restriktiv  
konfigurieren!**

## Code-Injektion

- Große Gefahr durch Ausführung von Systembefehlen (oder anderen Programmen) mit Parametern aus einem Formular: (durch unerwünschte Befehle innerhalb des Parameters)

- ⇒ 2. Unix-Befehl mit ; angehängt

Beispiel: Suche nach Unix-Benutzer löscht alle Dateien

Eingabe per Formular: gesuchter Name, z.B. james

Implementierung: `passthru('finger '.$_POST['name']);`

Angriff: `james; rm -rf /`

- ⇒ String als PHP-Code ausführen mit `eval("PHP-Code")`

Beispiel: Berechnung von arithmetischen Ausdrücken zeigt Passworddatei an

Eingabe per Formular: ein Ausdruck, z.B. `(3*4)+5`

Implementierung: `eval("echo ${_POST['name']}");`

Angriff: `file_get_contents('/etc/passwd')`

- Systemaufrufe in CGI Skripten sollten möglichst vermieden werden

- ⇒ Inhalt eventuell mit `escapeshellarg()` etc. sichern

passthru gibt einen String zur Ausführung an das System und gibt das Ergebnis zurück

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

## Zugriff auf Dateien im Server

- Anwendung: Dateiverwaltung (z.B. Fotoalbum) via Web
  - Link auf Bild: <[vergrößern](fetch.php?img=237.jpg)>
  - Angriff: `../../../../etc/passwd`
- Problem: Manipulierte Dateinamen können andere Dateien liefern bzw. abrufen als beabsichtigt
- 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
  - ⇒ zumindest den Pfad fest kodieren
- in Dateinamen keine .. und keine shell-Steuerzeichen zulassen
  - ⇒ besser: in Dateinamen nur a..z, A..Z, 0..9, -, \_ zulassen

## HTTP und HTTPS

- HTTP ist unverschlüsselt
  - ⇒ kann mit Sniffer (z.B. Wireshark) abgehört werden
  - ⇒ Passwort und persönliche Daten im Klartext
- HTTPS ist verschlüsselt mit SSL
  - ⇒ erfordert (gekauftes) Zertifikat und Zertifizierungshierarchie
  - ⇒ selbstgemachte Zertifikate sind nach Installation genauso sicher
  - ⇒ Angriffsmöglichkeit für "man in the middle" während Installation
- Konsequenz für Entwickler:
  - ⇒ jede Website mit Benutzerverwaltung sollte HTTPS verwenden
- Konsequenz für Anwender:
  - ⇒ nicht dasselbe Passwort für verschlüsselte und unverschlüsselte Verbindungen verwenden

## Zusätzliche Formularparameter

- PHP kann Formularparameter als einfache globale Variable bereitstellen (ab PHP 5.4 nicht mehr verfügbar)
- PHP interpretiert nicht initialisierte Variablen als false
- Angriff und Abwehr:
  - ⇒ Entwickler wertet nicht initialisierte Variable aus
  - ⇒ Hacker fügt weiteres Formularelement ein und initialisiert so die Variable
  - ⇒ Entwickler sollte jede Variable initialisieren
    - Warnung aktivieren: error\_reporting(E\_ALL);
  - ⇒ Entwickler sollte besser über \$\_GET oder \$\_POST auf Formularparameter zugreifen
    - zur Unterscheidung von Hilfsvariablen

```
if (...)
 $ok = true;

if ($ok)
 ...

<input type="hidden"
 name="ok" value="1">
```

## Angriff auf die Datenbank: SQL-Injektion

- Formularparameter werden oft in SQL-Anweisungen integriert
- Angriff: Hacker sendet einen Parameterstring, der die korrekte SQL-Anweisung verändert oder eine weitere anfügt
- Abwehr
  - ⇒ Entwickler muss jeden Parameter auf Gültigkeit prüfen (z.B. regul. Ausdr.)
  - ⇒ Entwickler muss SQL-Sonderzeichen ersetzen mit `mysqli::real_escape_string()`

```
$sql="SELECT * FROM student WHERE
"
 . "matn 1' OR '1=1' AND
pwd= '$pwd',
 Password:
```

## Zugriff auf Datensätze fremder Benutzer

- Jeder Seitenabruf innerhalb einer Session muss einem bestimmten Benutzer zugeordnet werden
  - ⇒ z.B. über Matrikelnummer oder Kundennummer
- Angriff
  - ⇒ Hacker könnte Benutzeridentifikation fälschen, falls im Klartext transferiert
- Abwehr:
  - ⇒ Entwickler: Benutzeridentifikation nicht aus (versteckten) Formular- oder URL-Parametern entnehmen
  - ⇒ Entwickler: Login identifiziert den Benutzer erstmalig; auf Folgeseiten erfolgt dies über SessionID und Session-Variable
    - Benutzerdaten werden bei Bedarf über Session-Variable weitergegeben

## Session-Hijacking: Einbruch in fremde Session

- Basis: Ausspähen der SessionID
  - ⇒ Benutzer ist eingeloggt, SessionID wird mit jeder Seite und Seitenanforderung übertragen
  - ⇒ Hacker ersetzt eigene SessionID durch fremde
- Angriff:
  - ⇒ Abhören des Netzverkehrs mit Sniffer
  - ⇒ Blick oder Foto über die Schulter des Benutzers
- Abwehr
  - ⇒ Verschlüsselte Verbindung (HTTPS) verwenden
  - ⇒ Benutzer sollte sich nicht beobachten lassen
  - ⇒ Benutzer sollte Cookies nicht abschalten (damit PHP die SessionID nicht an die URL anhängt)
  - ⇒ Benutzer sollte nicht eingeloggt bleiben und weggehen
  - ⇒ Entwickler sollte lange und kryptische SessionID generieren
  - ⇒ Entwickler kann zusätzlich HTTP\_REFERER und REMOTE\_ADDR prüfen

## Einbruch in fremde Anwendungsfälle

### ■ Basis

- ⇒ Es gibt Benutzer mit verschiedenen Rollen  
z.B. OBS mit Sekretariat, Dozenten, Admin

### ■ Angriff

- ⇒ Benutzer einer eingeschränkten Rolle ruft Anwendungsfall einer mächtigeren Rolle auf
  - z.B. durch Manipulation der GET-Parameter

### ■ Abwehr

- ⇒ rollenspezifisch eingeschränktes Menü ist komfortabel für den Benutzer, genügt aber nicht für die Sicherheit
- ⇒ Entwickler muss jeden Seitenaufruf rollenspezifisch kontrollieren (zwecks Übersichtlichkeit tabellarisch implementieren)

## Browser History

- Der Browser speichert die zuletzt aufgerufenen URLs einschließlich Parametern, egal ob GET oder POST verwendet wurde
  - ⇒ also auch ein Login mit Benutzername und Passwort
- Angriff
  - ⇒ nach dem Ausloggen eines Benutzers kann der nächste Benutzer aus der Browser History das Login seines Vorgängers erneut aufrufen
  - ⇒ insbesondere an öffentlichen Surfstationen (Internet Café, Web Kiosk)
  - ⇒ auf diese Weise hat einmal ein Student im OBS seinem Vorgänger am Web Kiosk im Foyer des Fbl dessen Plätze gelöscht und sie dann schnell selbst belegt ...
- Abwehr
  - ⇒ Benutzer: Browser nach Gebrauch schließen und History löschen
    - Problem: Web Kioske lassen das meistens nicht zu
  - ⇒ Entwickler: POST-Daten bei erneuter Übertragung nicht verarbeiten
    - z.B. mit Hilfe des TAN-Verfahrens, siehe Folie "View-Controller: sichere Formulare" im Abschnitt "Model-View-Controller Framework"

## Unerwünschte Skripte auf einem Client ausführen

- Basis:
  - ⇒ Eine Website (z.B. ein Forum oder Auktionshaus) prüft eingestellte Beiträge nicht gründlich und erzeugt aus den fremd-eingestellten Texten HTML-Seiten, die von Benutzern abgerufen werden
  - ⇒ "Sorglose Surfer" (z.B. auf der Suche nach PHP-Tipps in diversen Foren und Blogs) sind gefährdet
- Angriff: Cross-Site-Scripting (XSS)
  - ⇒ Ein Hacker stellt "Texte" mit Javascript-Anteilen ein (z.B. eine Login-Maske, welche die Eingaben weiterleitet)
  - ⇒ Die Skripte werden auf dem Client des Benutzers ausgeführt
- Abwehr:
  - ⇒ Benutzer: Javascript abschalten (dann funktionieren aber viele Webseiten nicht mehr vernünftig)
  - ⇒ Entwickler: Unbedingt alle Ausgaben so kodieren, dass HTML-Code als normaler Text erscheint – und keinesfalls interpretiert wird
    - `htmlspecialchars()` aufrufen

## Unerwünschte Skripte auf einem Server ausführen

### Basis

- ⇒ Ein Webauftritt integriert einen Teil eines anderen Webauftritts  
Beispiel: Auf www.fbi.h-da.de könnte eine Karte mit Wegbeschreibung von einem Kartenanbieter integriert sein

### Angriff (Variante des Cross-Site-Scripting):

- ⇒ Jemand hackt sich in das Netzwerk der Hochschule, fängt die Anfragen an den Kartenanbieter ab und ersetzt sie durch "erweiterte" Karten ("man in the middle")
- ⇒ Gefahr für Server, wenn der gelieferte Code ausgeführt wird
- ⇒ Gefahr für Client, falls der gelieferte Code JavaScript enthält

### Abwehr:

- ⇒ Den Code nicht ausführen, sondern nur ausliefern:  
Entwickler kann **readfile ("URL")** (liefert String) statt **require** bzw. **include ("URL")** verwenden
- ⇒ Admin kann Öffnen von URLs in PHP.INI generell sperren:  
**allow\_url\_fopen = 0**

dann geht  
readfile("URL")  
auch nicht mehr!

## Ausspähen des Datenbank-Passworts

- Basis
  - ⇒ Das Passwort zum Zugriff von PHP auf die Datenbank steht im PHP-Quellcode im Klartext
- Angriff:
  - ⇒ direkter Abruf der Passwort-Datei wenn URL bekannt
  - ⇒ durch Fehler im Webserver oder im PHP-Interpreter könnte eine PHP-Datei gezeigt statt ausgeführt werden
- Abwehr:
  - ⇒ Entwickler: Passwort in eigene Datei in geschütztem Verzeichnis
  - ⇒ Entwickler: Passwort in .php-Datei, nicht etwa .txt oder .html
  - ⇒ Entwickler: Passwort-Datei bei Open Source-Veröffentlichung rausnehmen (oder bei Installation abfragen)
  - ⇒ Admin: Datenbank nur vom localhost ansprechbar
    - nur möglich, wenn keine weiteren Datenbank-Clients existieren

## Infos für Hacker aus Fehlermeldungen

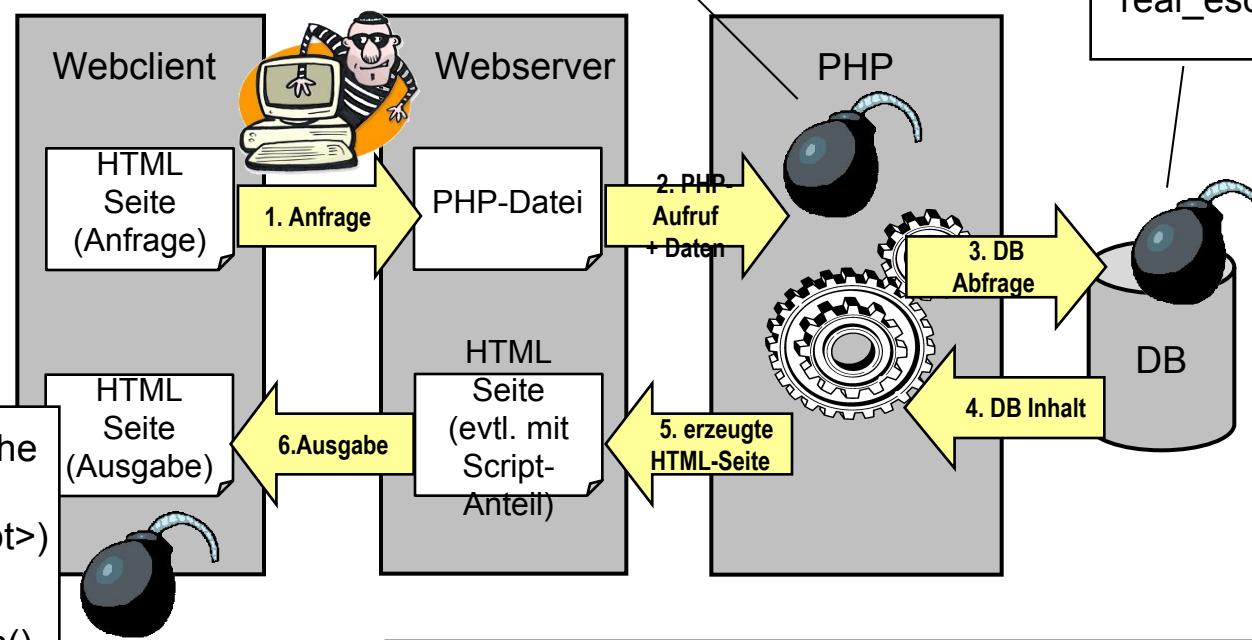
- Fehlermeldungen sind wichtig, auch im Produktivbetrieb
  - ⇒ aber dann bitte in einer Log-Datei und nicht auf dem Bildschirm
- Fehlermeldungen aus Bibliotheken nicht dem Benutzer zeigen
  - ⇒ versteht er meist ohnehin nicht
  - ⇒ enthalten evtl. wertvolle Hinweise für Hacker
  - ⇒ Entwickler muss dem Benutzer anwendungsbezogene Meldungen zeigen
  - ⇒ Entwickler muss systembezogene Meldungen in Log-Datei schreiben
    - `ini_set("error_reporting", E_ALL); // alles melden`
    - `ini_set("display_errors", 0); // aber nicht ausgeben`
    - `ini_set("log_errors", 1); // sondern loggen`
    - `ini_set("error_log", "./mylog.txt"); // in diese Datei`
    - Vorsicht: `mysqli::error()` zeigt u.U. DB-Passwort in Fehlermeldung

## 5. Sicherheit

# Pflichtmaßnahmen in PHP zur Sicherheit

Ausschluss von ausführbaren Befehlen  
Übergebene Daten genau überprüfen;  
Vergleich mit Listen / reg. Ausdrücken

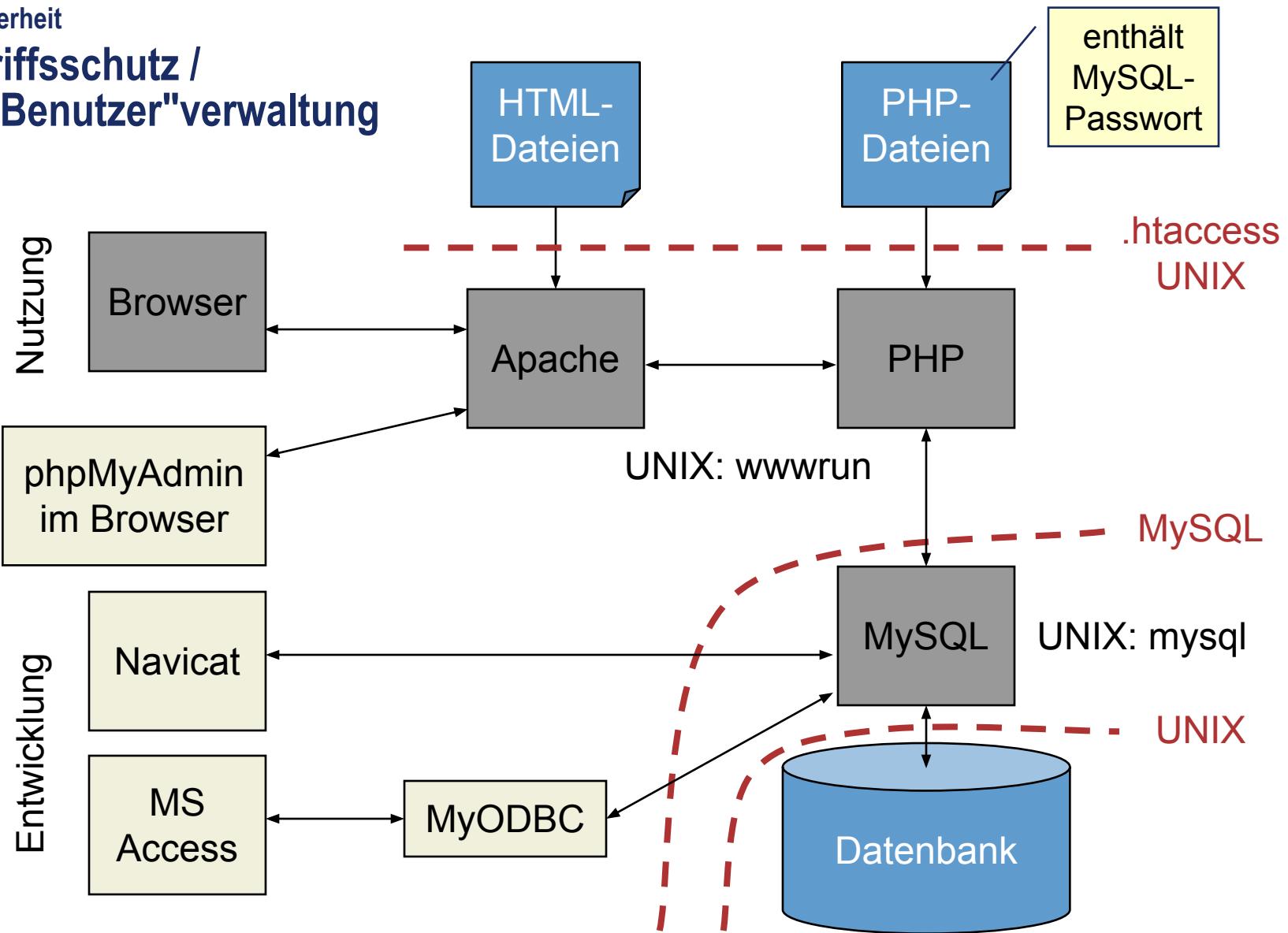
Für MySQL  
"gefährliche"  
Zeichenfolgen  
umwandeln:  
mysql::  
real\_escape\_string(...)



Für Client gefährliche  
Zeichenfolgen  
(z.B. XSS mit <script>)  
umwandeln:  
`htmlspecialchars()`  
außerdem HTML-  
konform und  
international!

Tipp: Das "HTML\_Quickform2"-Paket von PEAR wird benutzt, um HTML-Formulare komfortabel aus PHP zu erzeugen. Optional erzeugt es auch eine Überprüfungsroutine (validation), die serverseitig und auch clientseitig (Javascript) funktioniert.

# Zugriffsschutz / "Benutzer"verwaltung



## Interessante Links

- Bundesamt für Sicherheit in der Informationstechnik (BSI)

<https://www.bsi.bund.de>

- ⇒ Lagebericht zur IT-Sicherheit in Deutschland
  - ⇒ IT Grundschutz
  - ⇒ uvm.

- OWASP : Open Web Application Security Project

<https://www.owasp.org>

- ⇒ Eine Projekt zur Vermittlung von Know-How im Bereich IT-Sicherheit
  - ⇒ Eine Anwendung, die (absichtlich) Sicherheitslücken aufweist, welche in verschiedenen Lektionen gefunden werden sollen

# Hochschule Darmstadt

## Fachbereich Informatik

## 6. Professionelle Webentwicklung



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

## Motivation

### ■ Professionelle Webentwicklung bedeutet

- ⇒ dass Sie eine Webanwendung "professionell" entwickeln  
d.h. beruflich als Fachmann bzw. Fachfrau
- ⇒ dass in der Regel jemand von Ihrem Arbeitsergebnis "lebt" und für eine adäquate Qualität bezahlt
- ⇒ dass je nach Qualitätsanforderungen (z.B. Wartbarkeit, Testbarkeit, Robustheit, Verteilbarkeit, Sicherheit etc.)
  - ein entsprechendes Vorgehen gewählt und
  - entsprechende Ergebnisse erarbeitet werden
- ⇒ dass Sie ein bewusstes Vorgehen haben

Auch für webbasierte Anwendungen gilt alles, was Sie über Softwaretechnik, Software Ergonomie und GUIs gelernt haben!

## Web Engineering

### ■ Web Engineering

- ⇒ ist ein Zweig des Software Engineering, der sich mit der Entwicklung von Webanwendungen beschäftigt
- ⇒ ist wichtig für komplexe Websites wie Portalsysteme, Shops etc.
- ⇒ passt die klassischen Methoden der Softwaretechnik für den gesamten Lebenszyklus einer Webanwendung an – und berücksichtigt dabei die Besonderheiten von Webanwendungen

### ■ Besonderheiten bei Webapplikationen

- ⇒ hoher Gestaltungsanteil
- ⇒ Benutzbarkeit ist extrem wichtig
- ⇒ Kunde/Auftraggeber weiß oft nicht (genau), was er will

Es entstehen spezielle Vorgehensmodelle und spezielle Arbeitstechniken für die einzelnen Phasen der Entwicklung!

## "Do It Yourself" oder "Off the Shelf"-Entwicklung?

- Es gibt für typische Websites fertige Lösungen (Off-the-Shelf)
  - ⇒ Homepage-Pakete bei vielen Internet-Providern
  - ⇒ mit branchenspezifischen Vorlagen und konfigurierbarem Design
  - ⇒ mit konfigurierbaren Elementen wie Feedback-Formular, Foto-Galerie, Kontaktformular, Gästebuch, Shops, Maps usw.
  - ⇒ mit eigener Domain und Emailadresse
  - ⇒ gegen geringe monatliche Kosten
- Den Betrieb übernimmt der Provider
  - ⇒ Hardware, Aktualisierung und Wartung von Webserver & Co.
  - ⇒ Unterstützung für neue Clients (z.B. Smartphones)
- Derartige Homepage-Pakete sind oft sehr gute Lösungen
  - ⇒ für Webauftritte ohne exotische Anforderungen
  - ⇒ die Pflege der Inhalte ist einfach (nach dem Aufsetzen des Auftritts) und erfordert keine Spezialisten wie bei einer "Do It Yourself"-Entwicklung

# Hochschule Darmstadt

## Fachbereich Informatik

### 6.1 Vorgehensmodelle in der Webentwicklung



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

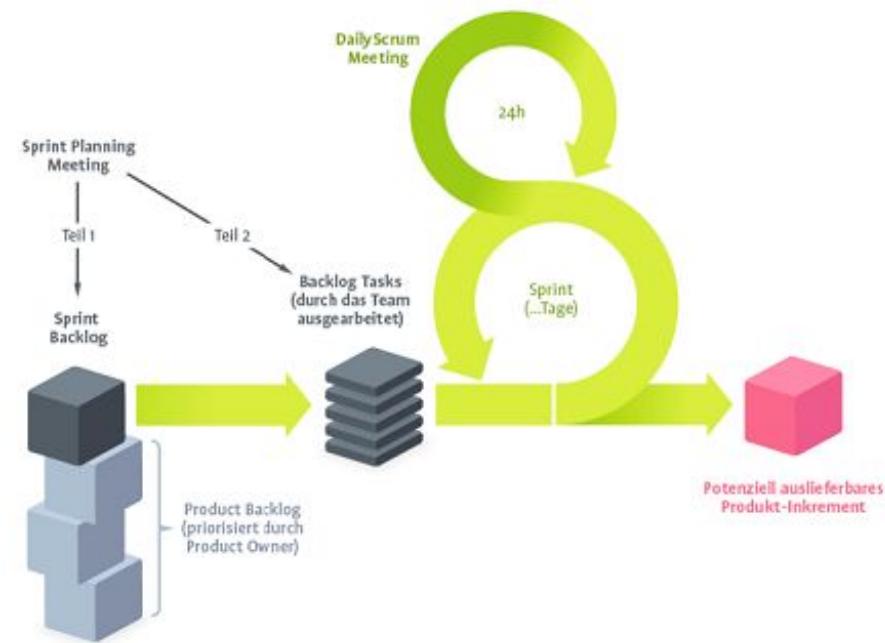
**fbi**

FACHBEREICH INFORMATIK

## 6.1 Vorgehensmodelle in der Webentwicklung

# Web Engineering: Vorgehensmodell Scrum

- Iterative und inkrementelle Entwicklung mit hoher Kundenbeteiligung
- Prozess zur Durchführung der Entwicklung in kurzen iterativen Zyklen ("Sprints").
- Fokussiert auf die Entwicklung und sollte in anderes Vorgehensmodell eingebettet werden
- Einsatzfähige Teilergebnisse für den Auftraggeber schon nach relativ kurzer Zeit



Adaption von "Agile Software Development with Scrum" von K. Schwaber, M. Beedle

[http://slides.liip.ch/about-liip/slide\\_18.html](http://slides.liip.ch/about-liip/slide_18.html)

# Hochschule Darmstadt

## Fachbereich Informatik

## 6.2 Arbeitstechniken in der Webentwicklung



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

## Anpassung von Arbeitstechniken (Beispiele)

Phase	Arbeitstechnik (Beispiel)
■ Anforderungsanalyse	■ Story Cards statt Use Cases
■ Architektur / Design	■ Spezielle Architekturen ■ Zusätzliche Modelle für Navigation, Präsentation, Hypertext etc.
■ Implementierung	■ Spezielle Programmiersprachen und Implementierungsrichtlinien
■ Test	■ Automatisierte Tests, automatisierte GUI-Tests
■ Wartung	■ Erzeugung von Dokumentationen aus Code und Testfällen

Im folgenden werden einige Beispiele für die Phasen präsentiert!

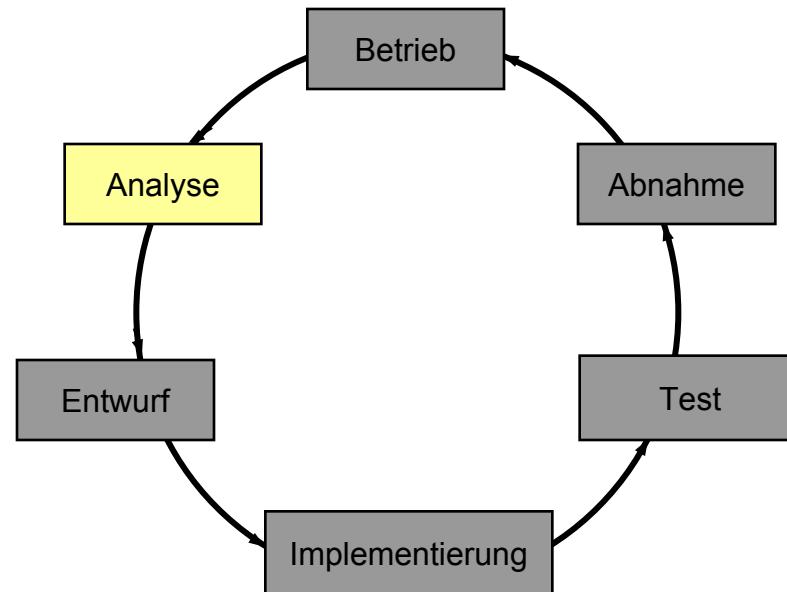
Einige davon verwenden PHP.

In anderen Sprachen gibt es aber ähnliche Lösungen.

# Hochschule Darmstadt

## Fachbereich Informatik

### 6.2.1 Arbeitstechniken in der Analyse



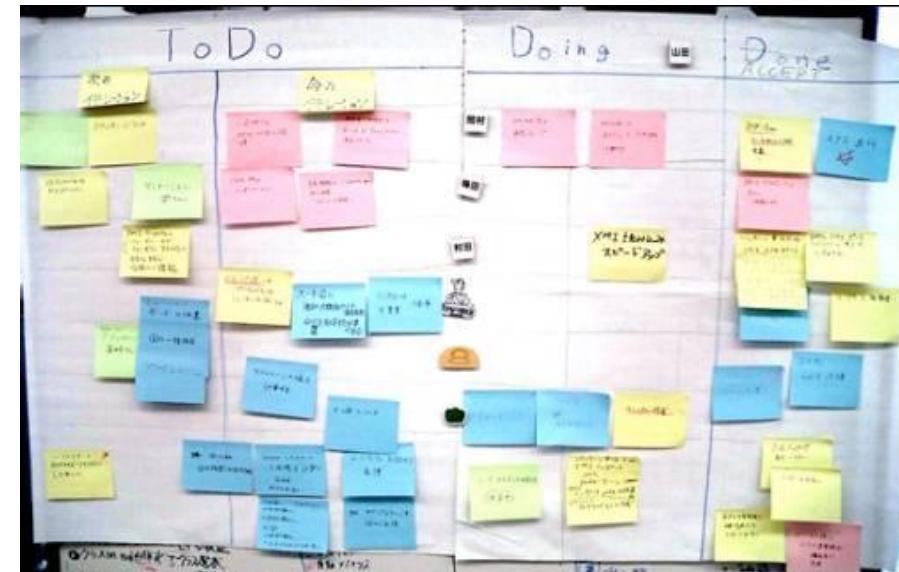
## 6.2.1 Arbeitstechniken in der Analyse

### Erfassung und Verwaltung von Anforderungen

Das kennen Sie schon !  
vgl. Praktikum Aufgabe 1

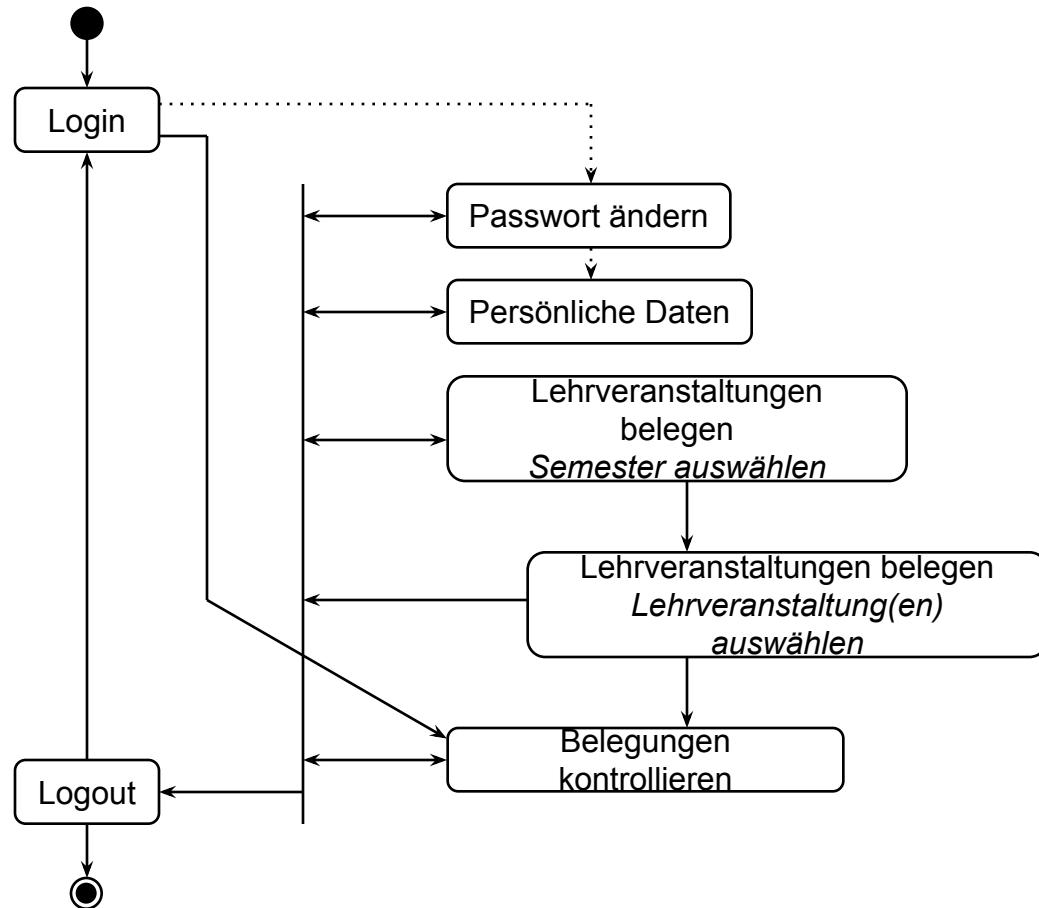
#### ■ Story Cards

- ⇒ enthalten je eine User Story – d.h. eine Funktionalität, die sich der Auftraggeber wünscht
- ⇒ z.B. "Warenkorb füllen":  
Der Kunde klickt auf die gewünschten Pizzasymbole und die Pizzen werden in den Warenkorb übernommen. Der aktuelle Preis wird sofort angezeigt.
- ⇒ erlauben schnelle und unkomplizierte Erfassung von Anforderungen
- ⇒ Die Karten werden zur Projektplanung und Verfolgung eingesetzt  
(Priorisierung und Aufwandsschätzung)



## 6.2.1 Arbeitstechniken in der Analyse

# Navigationsübersicht als Zustandsdiagramm



Zustände

generierte  
HTML-Seiten

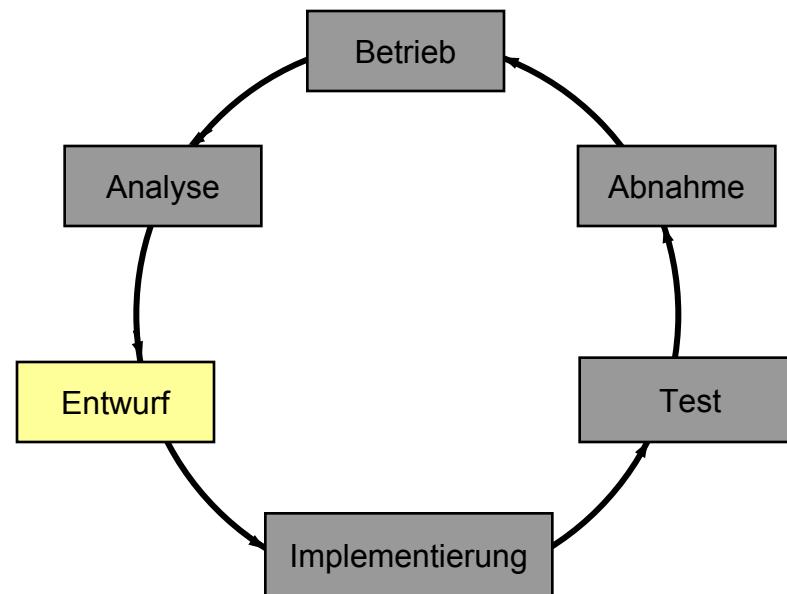
Zustands-  
übergänge

URL-Aufruf /  
Formular senden  
↓  
PHP-Programm  
ausführen

# Hochschule Darmstadt

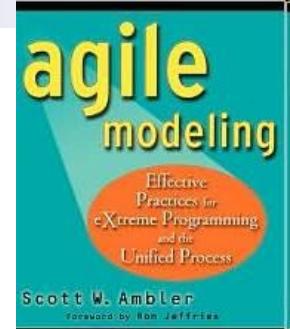
## Fachbereich Informatik

### 6.2.2 Arbeitstechniken im Entwurf

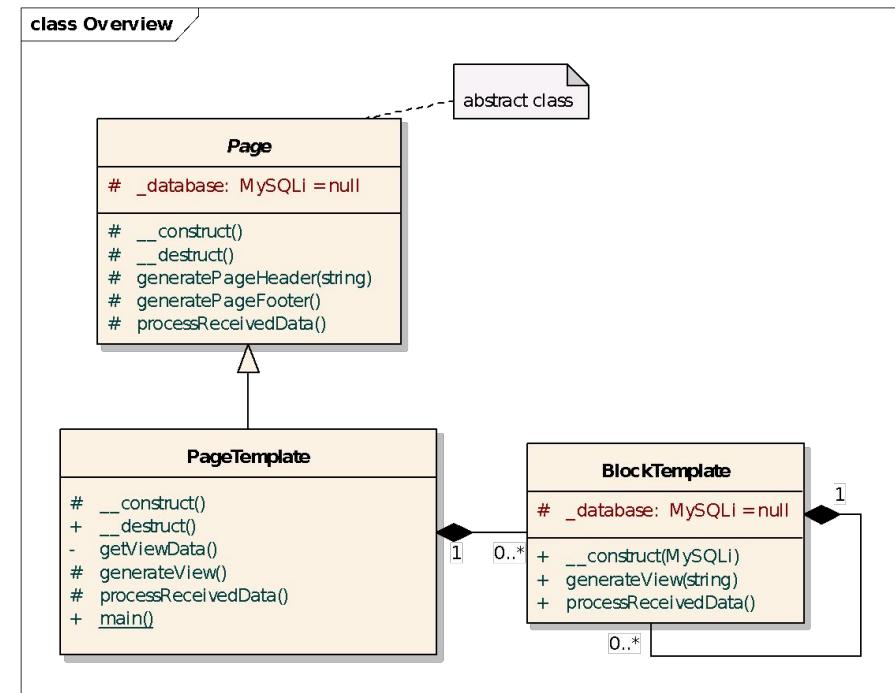


## 6.2.2 Arbeitstechniken im Entwurf

### Modellierung in Webprojekten



- Auch Webprojekte haben ein Konzept – und manchmal ist es sogar dokumentiert !
- Moderne Case-Tools können auch mit PHP-Code umgehen
  - ⇒ Entwurf und Dokumentation von Klassen
  - ⇒ Unterstützung der UML
  - ⇒ Import aus vorhandenen PHP-Dateien
  - ⇒ Export von Coderahmen

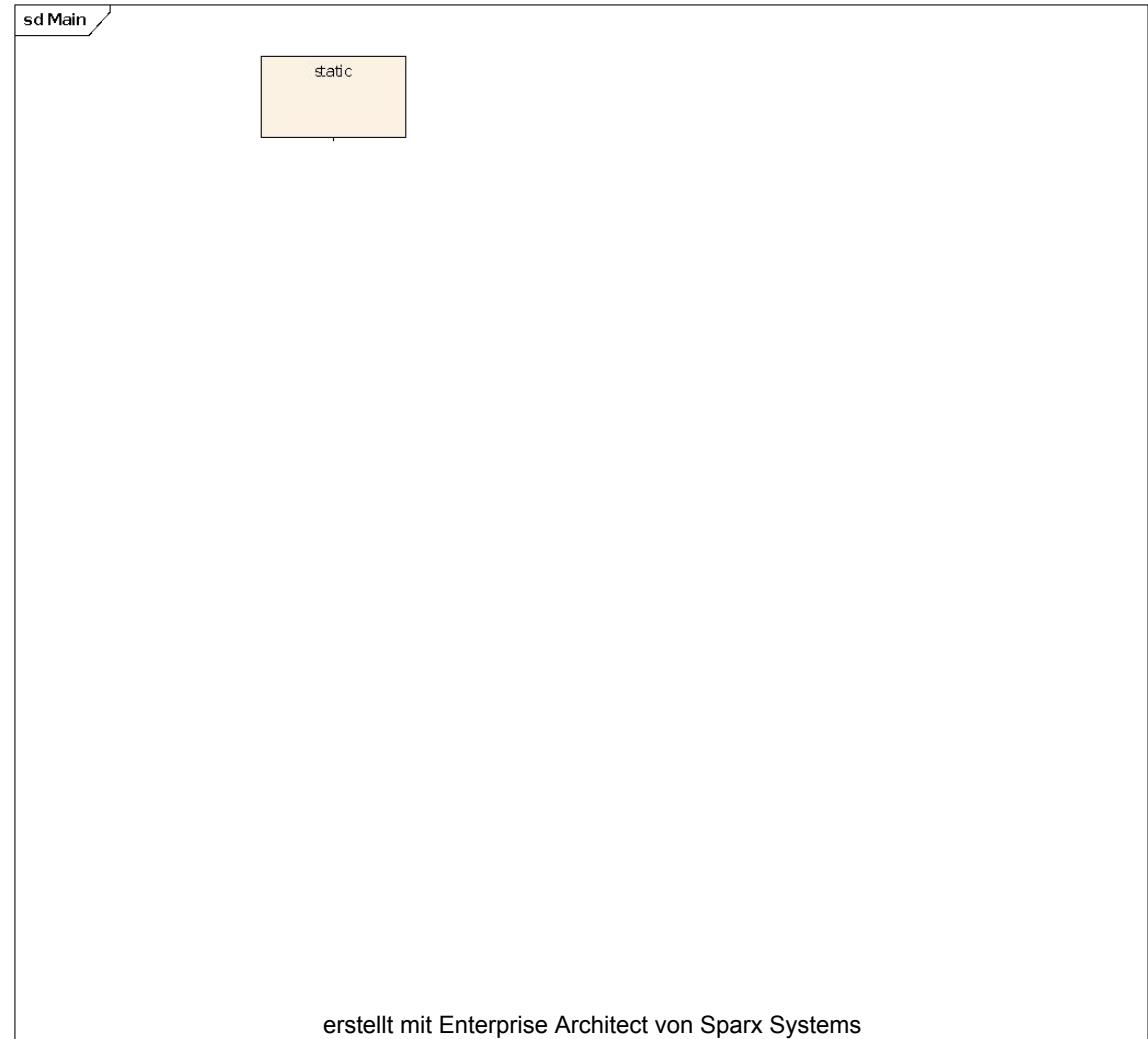


erstellt mit Enterprise Architect von Sparx Systems

## 6.2.2 Arbeitstechniken im Entwurf

### Modellierung in Webprojekten (II)

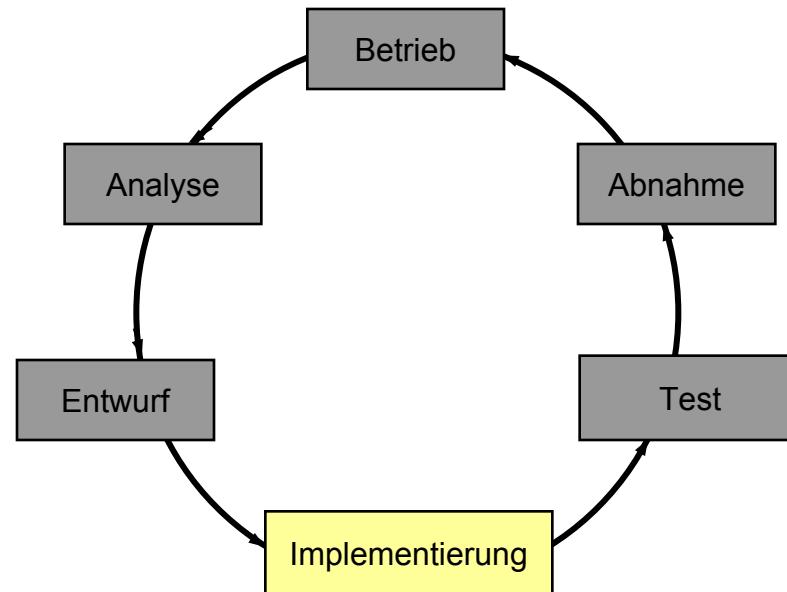
- Die Interaktion der Klassen kann modelliert und dokumentiert werden
- "Test" der Klassen als Trocken-Übung vor der Implementierung
- Grundlage für Kommunikation im Projekt



# Hochschule Darmstadt

## Fachbereich Informatik

### 6.2.3 Arbeitstechniken in der Implementierung



### Programmierrichtlinien

<http://framework.zend.com/manual/de/coding-standard.html>

#### ■ Eine Auswahl von Regeln im Zend-Standard:

##### ⇒ Fehlervermeidung

- In jeder **switch** Anweisungen muss ein **default:** enthalten sein
- **if** für Einzelanweisungen (ohne **{}**) ist unerwünscht
- In PHP-Dateien wird ein schließendes PHP-Tag **?>** am Dateiende weggelassen (Vermeidet unabsichtlichen "Beginn" der Ausgabe)

##### ⇒ Verbesserung der Lesbarkeit

- Strenge Regeln für Einrückungstiefe, Leerzeichen, Klammerungen etc.
- Leerzeichen statt Tabs, Zeilenlänge von 80 Zeichen
- Vorgaben für Kommentare (Ort, Häufigkeit, Schlüsselworte)
- Namenskonventionen für Klassennamen, Methoden, Attribute

"Coding Standards sind in jedem Entwicklungsprojekt wichtig, aber sie sind speziell dann wichtig wenn viele Entwickler an dem gleichen Projekt arbeiten. Coding Standards helfen sicherzustellen, dass der Code von hoher Qualität ist, weniger Fehler hat, und einfach zu warten ist."

(Zend Coding Standard)

## 6.2.3 Arbeitstechniken in der Implementierung

# Überprüfung von Programmierrichtlinien

## ■ Programmierrichtlinien werden kontrolliert

- ⇒ z.B. mit PHP\_CodeSniffer
  - zeigt Verstöße gegen Richtlinien an!
  - Aufruf: `phpcs my_source.php` (prüft PEAR-Standard)
  - oder z.B. `phpcs --standard=zend mysource.php` (Zend-Standard)
- ⇒ Anfangs ist die Erfüllung der Richtlinien eher nervig, aber nach einer Gewöhnungsphase geht es schnell und die Wartbarkeit wird besser

```
FOUND 24 ERROR(S) AND 4 WARNING(S) AFFECTING 22 LINE(S)

1 | ERROR | End of line character is invalid; expected "\n" but found "\r\n"
2 | ERROR | You must use "/**/" style comments for a file comment
3 | ERROR | Space before opening parenthesis of function call prohibited
5 | ERROR | Space before opening parenthesis of function call prohibited
30 | WARNING | Inline control structures are discouraged
31 | ERROR | Spaces must be used to indent lines; tabs are not allowed
32 | WARNING | Inline control structures are discouraged
33 | ERROR | Spaces must be used to indent lines; tabs are not allowed
49 | ERROR | Expected "foreach (...) <\n"; found "foreach(...)<\n"
50 | ERROR | Line indented incorrectly; expected at least 4 spaces, found 1
50 | ERROR | Spaces must be used to indent lines; tabs are not allowed
```

## Dokumentation im Code



- Erzeugung einer Dokumentation aus Kommentaren im Quellcode
  - ⇒ es müssen bestimmte Schlüsselworte und Regeln beachtet werden
    - Kommentare stehen über Klassen, Methoden und Attributen
    - am Anfang einer Datei beschreibt ein Kommentar den Inhalt der Datei usw.
  - ⇒ z.B. mit PHP Documentor
    - `phpdoc -f myfile -t ./doc` bzw. `phpdoc -d mydir -t ./doc`
  - ⇒ Doxygen bietet ähnliche Funktionalität, aber mit einem GUI

```
/*
*
* Initialisierung der bestellten Pizza mit Status & ID der Speisekarte
* Die ID wird erst beim Speichern gesetzt, liefert vorher UNDEF
*
* @param $pizzaID PizzaID der Pizza in der Speisekarte
* @param $status Bestell_Status der Pizza
*
* @return Boolean true, wenn alles geklappt hat
public function initialize($pizzaNr, $kunde
...
}
```

```
initialize($pizzaNr,
 $kundenNr,
 $status
)
```

Initialisierung der bestellten Pizza mit Status & ID der Speisekarte Die ID wird erst beim Speichern gesetzt, liefert vorher UNDEF

**Parameter:**

\$pizzaID PizzaID der Pizza in der Speisekarte  
\$status Bestell\_Status der Pizza

**Rückgabe:**

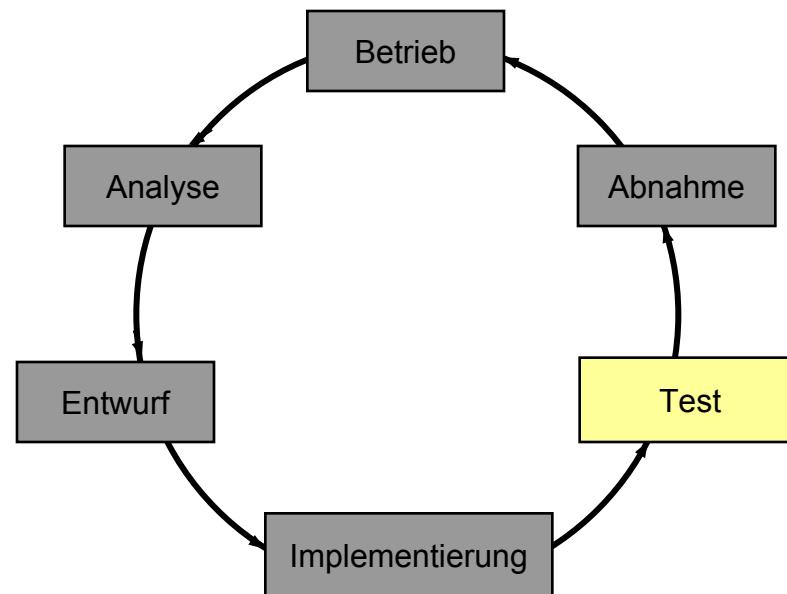
Boolean true, wenn alles geklappt hat

Ausgabe als HTML,  
XML, RTF uvm.

# Hochschule Darmstadt

## Fachbereich Informatik

### 6.2.4 Arbeitstechniken im Test



#### 6.2.4 Arbeitstechniken im Test

## Testen in Webprojekten

- Üblicherweise findet Webentwicklung in vielen Iterationen statt
  - ⇒ Nach jeder Iteration muss getestet werden
  - ⇒ Automatisierte Tests sind die einzige Möglichkeit für einen systematischen Test
  - ⇒ Es gibt Tools
    - zur automatisierten Durchführung von Unit-Tests
    - zur Erzeugung von Dokumentationen der Testfälle
    - zur Berechnung der Fehlerüberdeckung
    - für Performance-Analysen und Profiling
    - für automatisierte GUI-Tests

# Unit-Tests mit PHP – Das Prinzip

Whenever you are tempted to type something into a print statement or a debugger expression, write it as a test instead.

--Martin Fowler

- Für PHP gibt es über PEAR ein Paket für PHP-Unit-Tests

<http://www.phpunit.de>

- Für PHP gibt es über PEAR ein Paket für PHP-Unit-Tests
  - ⇒ ähnlich zu CPP-Unit oder JUnit
  - ⇒ Festlegen der Tests
    - zu jeder Klasse wird eine Testklasse angelegt  
(die Testklasse implementiert Schnittstellen vom Testframework)
    - die Methoden der zu testenden Klasse werden in den Tests aufgerufen
    - es wird das erwartete "Soll-Ergebnis" festgelegt
    - bei Bedarf können vor und nach jedem Test Befehle ausgeführt werden
  - ⇒ Durchführung der Tests
    - ein einfacher Aufruf startet alle geschriebenen Tests
    - Tritt ein Unterschied zwischen Ist-Ergebnis und Soll-Ergebnis auf, erfolgt eine entsprechende Meldung
    - `phpunit tests_for_myfile.php` führt die Unit-Tests in der Datei tests\_for\_myfile.php aus

## 6.2.4 Arbeitstechniken im Test

# PHPUnit: Eine Beispielklasse und eine Testklasse

```
class calc{
 public $l; // enthaelt den linken Operanden
 public $r; // enthaelt den rechten Operanden
 public function __construct($l,$r) // Konstruktor {
 $this->l = $l;
 $this->r = $r;
 }
 ...
 public function sub() {
 return $this->l + $this->r;
 }
}
```



```
class calcTest extends PHPUnit_Framework_TestCase {
 public function testSub() {
 $l = 5; $r = 3;
 $erg = $l - $r; // Ergebnis auf alternativem Weg berechnen
 $obj = new calc ($l, $r);
 $this->assertEquals($erg, $obj->sub());
 }
 ...
}
There was 1 failure:
1) testSub(calcTest)
Failed asserting that <integer:8> matches expected value <integer:2>.C:\Links\ewa\Vorlesung\Links\PHP\90_calctest.php:16
```



## 6.2.4 Arbeitstechniken im Test

### PHPUnit: Hilfsfunktionen

- Leere Testklasse automatisch anlegen:
  - ⇒ `phpunit --skeleton MeineKlasse`  
liest MeineKlasse.php ein und erzeugt MeineKlasseTest.php –  
für jede Methode wird ein (leerer) Testfall erzeugt
  - ⇒ neue PHPUnit-Version: `phpunit-skelgen --test MeineKlasse`
- Zugriff auf private (!) Attribute
  - ⇒ `$mytestclass->readAttribute  
($object, 'NameOfPrivateAttribute');`  
liefert den Wert des Attributs *NameOfPrivateAttribute*
  - ⇒ dadurch können Methoden getestet werden, ohne sich auf die Funktionsfähigkeit von anderen Methoden zu verlassen
- Zugriff auf private Methoden
  - ⇒ möglich durch Einführung einer neuen Klasse, die alle **private**-Methoden erbt und **public** macht. Für diese Klasse werden dann die Tests erstellt.

## 6.2.4 Arbeitstechniken im Test

### PHPUnit: Dokumentation der Tests

- Die Namen der Testmethoden werden automatisch zu Text umformatiert
  - ⇒ Aus `testInitializePassesParametersToObject()` wird "Initialize passes parameters to object"
  - ⇒ HTML-Ausgabe  
`phpunit --testdox-html out.html tests_for_myfile.php`
  - ⇒ Bildschirm-Ausgabe  
`phpunit --testdox tests_for_myfile.php`

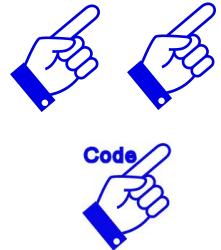


AngebotenePizza

```
[x] Initialize passes parameters to object
[x] Safe stores object in database
[x] Delete removes entry from database
[] Delete from database with invalid identifier
[x] Get pizza id returns identifier
[x] Get preis return preis
[x] Get name returns name
[x] Get bild returns name of bild
```

## 6.2.4 Arbeitstechniken im Test

### PHPUnit: Anweisungsüberdeckung



- Während eines Testdurchlaufs werden die ausgeführten Programmteile markiert

- ⇒ `phpunit --coverage-html ./coverage tests_for_myfile.php`
- ⇒ führt die Unit-Tests in `tests_for_myfile.php` aus und erzeugt im Verzeichnis `coverage` einen HTML-Report der Anweisungsüberdeckung

D:\users\Ralf\Documents\Hahn\14SS\EWA\Vorlesung\Links\PHP / 90\_calc.php

	Code Coverage											
	Classes and Traits			Functions and Methods				Lines				
Total	<div style="width: 25%; background-color: orange;"></div>	<div style="width: 75%; background-color: lightgray;"></div>	50.00%	1 / 2	<div style="width: 75%; background-color: green;"></div>	<div style="width: 25%; background-color: lightgray;"></div>	75.00%	3 / 4	CRAP	<div style="width: 91.67%; background-color: green;"></div>	91.67%	11 / 12
DivException	<div style="width: 100%; background-color: green;"></div>		100.00%	1 / 1				0				
calc	<div style="width: 0%; background-color: lightgray;"></div>		0.00%	0 / 1	<div style="width: 75%; background-color: green;"></div>	<div style="width: 25%; background-color: lightgray;"></div>	75.00%	3 / 4	6.02	<div style="width: 91.67%; background-color: green;"></div>	91.67%	11 / 12
<code>__construct(\$l,\$r) // Konstruktor</code>					<div style="width: 100%; background-color: green;"></div>		100.00%	1 / 1	1	<div style="width: 100%; background-color: green;"></div>	100.00%	3 / 3
<code>sub()</code>					<div style="width: 100%; background-color: green;"></div>		100.00%	1 / 1	1	<div style="width: 100%; background-color: green;"></div>	100.00%	1 / 1
<code>mul()</code>					<div style="width: 0%; background-color: lightgray;"></div>		0.00%	0 / 1	2	<div style="width: 0%; background-color: lightgray;"></div>	0.00%	0 / 1
<code>div(\$throwException = false)</code>					<div style="width: 100%; background-color: green;"></div>		100.00%	1 / 1	3	<div style="width: 100%; background-color: green;"></div>	100.00%	7 / 7

## PHPUnit: Anweisungsüberdeckung (II)



- Die Anweisungsüberdeckung zeigt an, welche Anweisung wie oft ausgeführt wurden
  - ⇒ nicht ausgeführte Teile werden rot markiert
  - ⇒ jede Anweisung sollte mindestens (!) einmal ausgeführt werden
  - ⇒ fehlende Tests sollten ergänzt werden
  - ⇒ 100% Anweisungsüberdeckung sind keine Garantie für Fehlerfreiheit

```

4 class calc
5 {
6 public $l; // enthaelt den linken Operanden
7 public $r; // enthaelt den rechten Operanden
8 public function __construct($l,$r) // Konstruktor
9 {
10 $this->l = $l;
11 $this->r = $r;
12 }
13 // gibt das Ergebnis der Subtraktion zurück
14 public function sub()
15 {
16 // Hier rauskommentieren
17 return $l - $r;
18 }
19 // Gibt das Ergebnis der Multiplikation zurück
20 public function mul()
21 {
22 return $this->l * $this->r;
23 }
24 }
```

5 tests cover line 10

- calcTest::testSub
- calcTest::testDiv
- calcTest::testDivByZero
- calcTest::testDivByZeroThrowsException
- calcTest::testClassDivException

Der Konstruktor wurde 5 mal ausgeführt!

Die methode mul() wurde nicht ausgeführt! Es gibt keinen Test dafür.

## 6.2.4 Arbeitstechniken im Test

### Profiling mit PHP

- Während eines PHP-Aufrufs kann die Ausführungszeit und -häufigkeit der Programmteile protokolliert werden
  - ⇒ wenn PHP entsprechend konfiguriert ist (xdebug.profiler) wird eine "cachegrind"-Datei erzeugt
  - ⇒ `phpunit tests_for_myfile.php`
- Spezielle Viewer (z.B. wincachegrind) bereiten die Daten auf
  - ⇒ Profiling erlaubt gezielte Performance-Optimierung und
  - ⇒ erleichtert die Suche nach Speicherfehlern

## 6.2.4 Arbeitstechniken im Test

# Profiling mit PHP



WinCacheGrind - [C:\Links\ewa\Worlesung\Links\PHP\Pizzaservice\index.php (trace.out)]

File View Profiler Tools Window Help

index.php

index.php (main)

php::header

php::error\_reporting

require\_once::pwd.php

require\_once::bestellung\_head.php

require\_once::bestellung\_body.php

php::mysqli->mysql

php::mysqli\_connect\_errno

php::mysqli->query

php::mysqli\_result->fetch\_assoc

php::mysqli\_result->fetch\_assoc

php::mysqli\_result->fetch\_assoc

php::mysqli\_result->fetch\_assoc

php::mysqli\_result->fetch\_assoc

php::mysqli\_result->fetch\_assoc

php::mysqli\_result->free

php::mysqli->close

require\_once::kunde\_body.php

require\_once::baecker\_body.php

require\_once::fahrer\_body.php

require\_once::C:\Links\ewa\Worlesung\Links\PHP\Pizzaservice\bestellung\_body.php

File: C:\Links\ewa\Worlesung\Links\PHP\Pizzaservice\bestellung\_body.php

Self time: - (0,51%) Cumulative time: 0,6ms (4,98%)

Line by Line Overall

Find: Regular expression

Function Avg. Self Avg. Cum. Total Self Total Cum. Calls

php::mysqli->query	0,8ms	0,8ms	6,5ms	6,5ms	8
php::mysqli->mysql	0,5ms	0,5ms	3,4ms	3,4ms	7
php::mysqli->close	-	-	0,4ms	0,4ms	7
php::mysqli_result->fetch_assoc	-	-	-	-	37
php::mysqli_result->free	-	-	-	-	7
php::mysqli_connect_errno	-	-	-	-	7

Sum of total self time: 10ms (80,91%) Sum of calls: 73

Num.	Self	Cum.	Called by	Called from	Stack trace
1	0,2ms	0,2ms	require_once::bestellung_head.php	bestellung_head.php (13)	require_once::bestellung_
2	-	-	require_once::bestellung_body.php	bestellung_body.php (19)	require_once::bestellung_
3	-	-	require_once::baecker_body.php	baecker_body.php (12)	require_once::baecker_
4	-	-	require_once::baecker_body.php	baecker_body.php (34)	require_once::baecker_
5	-	-	require_once::fahrer_body.php	fahrer_body.php (12)	require_once::fahrer_
6	-	-	require_once::fahrer_body.php	fahrer_body.php (36)	require_once::fahrer_

Allocated memory: 122.152 bytes

# GUI Tests mit Selenium

<http://seleniumhq.org/projects/ide>

- Bisher haben wir "nur" Klassen mit UnitTests getestet
  - ⇒ aber wie testet man die gesamte Webanwendung incl. GUI?
- Automatisch ausführbare GUI-Tests wären praktisch
  - ⇒ Selenium IDE (Firefox Addon)
    - kann Mausklicks und Eingaben im Browser (intelligent) aufzeichnen und später wieder abspielen
    - dabei können Zusicherungen ("asserts") definiert und beim Abspielen überprüft werden: Prüfung von Texten, Inhalten, Formaten, Popups uvm.
    - Die Zuordnung zu Seitenelementen kann über die HTML-Id's erfolgen und ist dann unabhängig von der Positionierung
    - rechte Maustaste bietet in der Webseite sinnvolle Überprüfungen an
    - Vorsicht: Manchmal schlagen Tests fehl, wenn die Abfragen schneller ablaufen als die Kommunikation zwischen Webserver und Client. Die Geschwindigkeit muss dann korrigiert werden.

Selenium bietet eine geniale und einfache Möglichkeit  
automatisierbare GUI-Tests zu erstellen!

## 6.2.4 Arbeitstechniken im Test

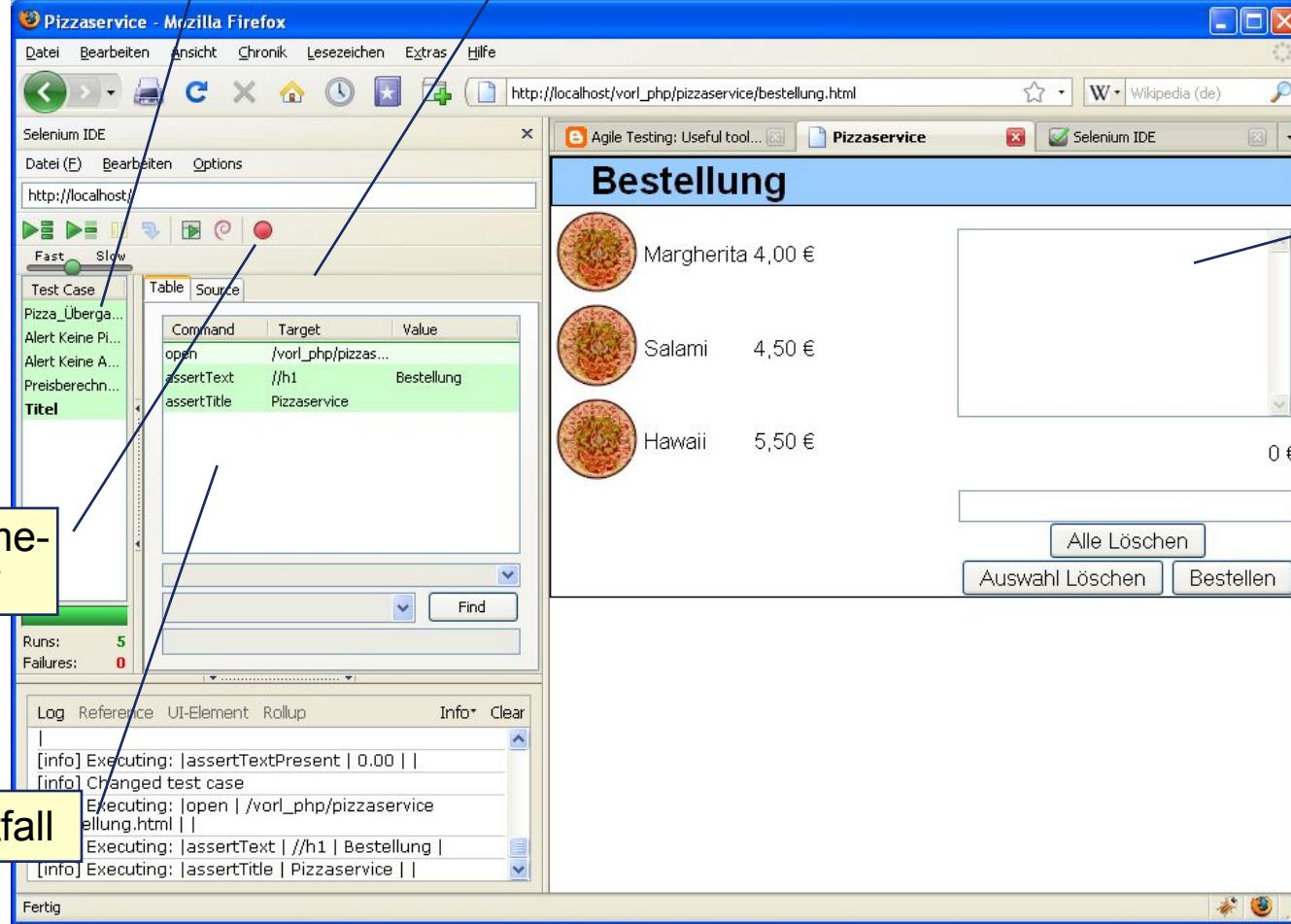
### Selenium IDE



The screenshot shows the Selenium IDE interface running in Mozilla Firefox. The sidebar on the left displays a test case titled "Pizza\_Überga...". The main window shows a "Bestellung" page for a pizza service, listing three pizzas: Margherita (4,00 €), Salami (4,50 €), and Hawaii (5,50 €). The total price is 0 €. Buttons for "Alle Löschen", "Auswahl Löschen", and "Bestellen" are visible at the bottom.

Annotations:

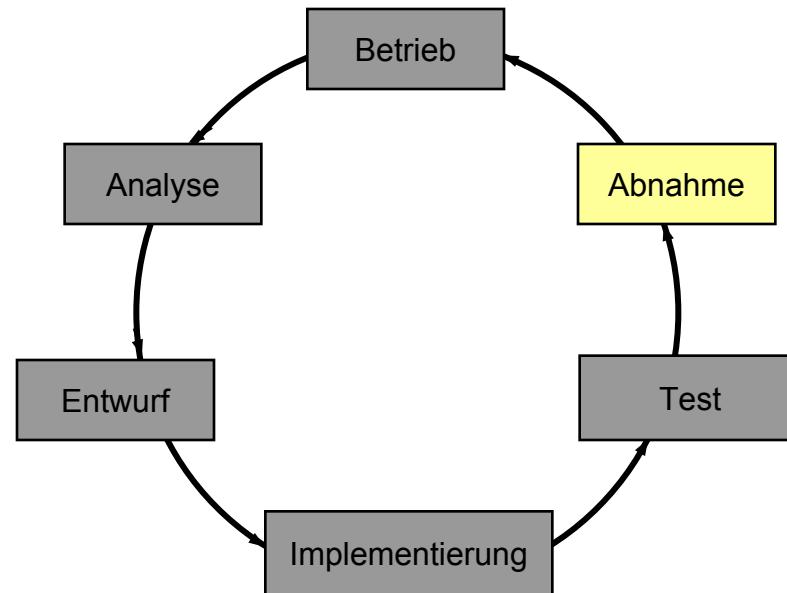
- Aufnahmeknopf** (Screenshot button) points to the green button in the toolbar of the Selenium IDE.
- Testfall** (Test case) points to the title of the test case in the sidebar.
- Testfälle (Testsuite)** points to the sidebar header.
- Selenium als Sidebar (Ansichtsoption)** points to the title of the main window.
- Webseite im Test** (Website under test) points to the "Bestellung" page displayed in the main window.



# Hochschule Darmstadt

## Fachbereich Informatik

### 6.2.5 Arbeitstechniken in der Abnahme



## 6.2.5 Arbeitstechniken in der Abnahme

# Checkliste für die Abnahme (I)

## Checkliste zur Gestaltung barrierefreier Webanwendungen und Webauftritte

Web-Baukasten der Friedrich-Alexander-Universität Erlangen  
Stand: Mai 2007 (Version 1.6)

Vorgaben		erfüllt	k.o.-Krit.
1. Aufbau	1. Die vollständige Bedienbarkeit für das Ausgabemedium Screen ist bei einer Bildschirmauflösung von mindestens 800 x 600 Pixel gewährleistet.	ja nein	
	2. Falls Frames verwendet werden, sind diese mit sinnvollen Titeln und Namen versehen?	ja nein trifft nicht zu	
	3. Für das Ausgabemedium „handheld“ ist ein eigenes Stylesheet vorhanden.	ja nein	
	4. Für das Ausgabemedium „print“ ist ein eigenes Stylesheet vorhanden.	ja nein	
2. Inhalte	1. Für die Gestaltung und Positionierung von Elementen werden Stylesheets verwendet.	ja nein	
	2. Die Webanwendung ist auch ohne Stylesheets (CSS) nutzbar.	ja nein	<b>k.o.</b>
	3. Die Markup-Sprache HTML Version 4.x oder XHTML Version 1.x wird gemäß den Regeln des W3C korrekt („valide“) eingesetzt.	ja nein	<b>k.o.</b>
	4. CSS Version 2 wird entsprechend den W3C-Standards korrekt eingesetzt.	ja nein	
	5. Zur reinen Texteinrückung werden keine Listen und Listenelemente verwendet.	ja nein	
	6. Wird eine dem Inhalt angemessene, einfache und klare Sprache benutzt?	ja nein	
3. Schrift	1. Die Schriftgrößen sind variabel.	ja nein	<b>k.o.</b>
	2. Für Überschriften werden die Überschriftentags von <h1> bis <h6> benutzt.	ja nein	
	3. Unterstreichungen werden nur für Links verwendet.	ja nein	
	4. Hintergrund und Schrift heben sich kontrastreich voneinander ab.	ja nein	

## 6.2.5 Arbeitstechniken in der Abnahme

# Checkliste für die Abnahme (II)

Vorgaben		erfüllt	k.o.Krit.
<b>4. Grafiken und Bilder</b>	<p>1. Es sind keine animierten Grafiken vorhanden.</p> <p>2. Grafiken sind mit aussagekräftigen Alternativtexten beschrieben. <b>Ausnahme:</b> Grafiken ohne informative Funktionen, z.B. Farbflächen, dekorative Farbübergänge und Schmuckgrafiken</p>	ja nein	
<b>5. Farben und Kontraste</b>	<p>1. Mit Farbe dargestellte Informationen sind auch ohne Farbe nutzbar bzw. stützen sich nicht ausschließlich auf Farbangaben.</p> <p>2. Problematische Farbkombinationen werden vermieden.</p>	ja nein	<b>k.o.</b>
<b>6. Navigationsmechanismen</b>	<p>1. Grafische Bedienelemente sind mit aussagekräftigen Alternativtexten versehen, die auch das Ziel des Links bezeichnen.</p> <p>2. Eine Inhaltsübersicht (Sitemap) oder eine ähnliche Orientierungshilfe ist vorhanden.</p> <p>3. Das Ziel von Hyperlinks ist eindeutig identifizierbar (keine Formulierungen wie „Klicken Sie hier“, sondern nur Links mit beschreibendem Text).</p> <p>4. Bei Links auf nicht-HTML-Seiten (z.B. PDF, Word, usw.) ist das Datenformat der verlinkten Datei beschrieben.</p> <p>5. Linkziele werden durch informative „title“-Attribute klargestellt (z.B. Link öffnet PDF-Datei in neuem Fenster).</p> <p>6. Das Angebot ist auch ohne Maus, also nur mit der Tastatur bedienbar.</p> <p>7. Auf automatische Popup-Fenster wird verzichtet, das Öffnen neuer Fenster erfolgt mit Ankündigung.</p> <p>8. Bei komplexen Webseiten sind Sprungmarken zum Inhalt, zur Navigation und weiteren herausragenden und wichtigen Inhaltsbereichen bereitzustellen.</p>	ja nein trifft nicht zu	<b>k.o.</b>

## 6.2.5 Arbeitstechniken in der Abnahme

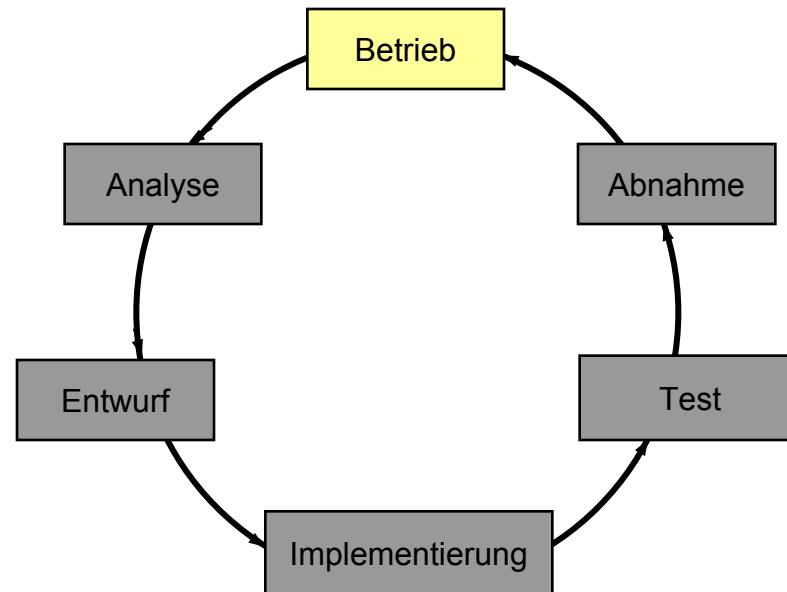
### Checkliste für die Abnahme (III)

Vorgaben		erfüllt	k.o.Krit.
7. Tabellen	1. Es werden keine verschachtelten Tabellen verwendet.	ja nein	
	2. Bei <b>Datentabellen</b> sind Zeilen- und Spaltenüberschriften gekennzeichnet; zur Kennzeichnung werden <code>&lt;td&gt;</code> für Datenzellen und <code>&lt;th&gt;</code> für Überschriften verwendet.	ja nein trifft nicht zu	
	3. Es werden keine <b>Layouttabellen</b> verwendet.	ja nein	<b>k.o.</b>
8. Formulare	1. Formularfelder sind durch das Element <code>&lt;fieldset&gt;</code> gruppiert und damit in logische Blöcke eingeteilt.	ja nein trifft nicht zu	
	2. Die jeweilige Gruppe wird benannt durch den Einsatz des <code>&lt;legend&gt;</code> - Elements.	ja nein trifft nicht zu	
	3. Zwischen den einzelnen Formularfragen und den Eingabefeldern wird mit Hilfe von <code>&lt;label&gt;</code> eine logische Verknüpfung hergestellt.	ja nein trifft nicht zu	
9. Scripte und Applets	1. Werden clientseitige Scripte verwendet?	ja nein	
	2. Webseiten sind auch verwendbar, wenn clientseitige Scripte, Applets oder programmierte Objekte im Browser abgeschaltet sind oder der Browser sie nicht unterstützt.	ja nein trifft nicht zu	
	3. Es sind keine flackernden, blinkenden oder sich bewegende Elemente (z.B. Laufschriften) vorhanden.	ja nein	<b>k.o.</b>
	4. ActiveX und VBScript dürfen nicht verwendet werden. Ist dies erfüllt?	ja nein	<b>k.o.</b>
10. Webauftritt insgesamt	1. Der Webauftritt insgesamt erreicht bei der Durchführung des BITV-Tests mindestens 80 Punkte („eingeschränkt zugänglich“). (Der BITV-Test wird nur ausgefüllt, wenn bereits eine Seite zur Abnahme vorliegt. )	ja nein nicht geprüft	<b>k.o.</b>

# Hochschule Darmstadt

## Fachbereich Informatik

### 6.2.6 Arbeitstechniken im Betrieb



## 6.2.6 Arbeitstechniken im Betrieb

# Fehlermeldungen beim Surfen

- Die Suche nach "Warning: Access denied for user" liefert über 1 Million Treffer bei Google

- die gefundenen Webseiten enthalten oft fehlerhaften PHP-Code
- die Fehlermeldungen werden nicht unterdrückt
- manche Meldungen enthalten User oder Passwort



**Warning:** mysql\_connect() [[function.mysql-connect](#)]: Access denied for user 'thue4b1'@'lw-s09.sserv.de' (using password: YES) in [/www/thue/thue4b1/html/mysql.php](#) on line 3

**Warning:** mysql\_select\_db() [[function.mysql-select-db](#)]: Access denied for user 'thue4b1'@'localhost' (using password: NO) in [/www/thue/thue4b1/html/mysql.php](#) on line 4

**Warning:** mysql\_select\_db() [[function.mysql-select-db](#)]: A link to the server could not be established in [/www/thue/thue4b1/html/mysql.php](#) on line 4

**verbraucherzentrale  
Thüringen**

**Warning:** mysql\_connect() [[function.mysql-connect](#)]:  
Access denied for user 'thue4b1'@'lw-s09.sserv.de'  
(using password: YES) in [/www/thue/thue4b1  
/html/DBStuff.inc](#) on line 4

**Warning:** mysql\_select\_db():  
supplied argument is not a valid  
MySQL-Link resource in  
[/www/thue/thue4b1  
/html/DBStuff.inc](#) on line 5

**Verbraucherzentrale Thüringen e.V.**

**Warning:** mysql\_connect() [[function.mysql-connect](#)]: Access denied for user 'thue4b1'@'lw-s09.sserv.de' (using password: YES) in [/www/thue  
/thue4b1/html/mysql.php](#) on line 3

**Warning:** mysql\_select\_db() [[function.mysql-select-db](#)]: Access denied for user 'thue4b1'@'localhost' (using password: NO) in [/www/thue  
/thue4b1/html/mysql.php](#) on line 4

**Warning:** mysql\_select\_db() [[function.mysql-select-db](#)]: A link to the server could not be established in [/www/thue/thue4b1/html/mysql.php](#) on line 4

**Warning:** mysql\_query() [[function.mysql-query](#)]: Access denied for user 'thue4b1'@'localhost' (using password: NO) in [/www/thue/thue4b1](#)

**Warning:** mysql\_query() [[function.mysql-query](#)]:  
Access denied for user 'thue4b1'@'localhost'  
(using password: NO) in [/www/thue/thue4b1  
/html/footer.php](#) on line 45

**Warning:** mysql\_query() [[function.mysql-query](#)]: A link to the server could not be established in [/www/thue/thue4b1  
/html/footer.php](#) on line 45

www....

# Entwicklungs- vs. Produktivumgebung

- Entwickler und Anwender haben eine unterschiedliche Einstellung zu Fehlermeldungen
  - ⇒ Für einen Entwickler sind Fehlermeldungen wichtige Informationen
  - ⇒ Für einen Anwender sind Fehlermeldungen bestenfalls ärgerliche oder unverständliche Texte
  - ⇒ In einer Entwicklungsumgebung sollten Fehlermeldungen sichtbar sein – in einer Produktivumgebung sollten sie "unsichtbar" protokolliert werden
- Alle Web-Systeme bieten die Möglichkeit Fehler wahlweise in eine Log-Datei zu schreiben oder auf den Bildschirm zu bringen
  - ⇒ z.B. im PHP-Code :

```
ini_set("display_errors", 0); // 0=aus
ini_set("error_reporting", E_ALL); // alle Meldungen
ini_set("log_errors", 1); // Logging an
ini_set("error_log", "./mylog.txt"); // Logfile
```
  - ⇒ Immer gewissenhaft konfigurieren

## 6.2.6 Arbeitstechniken im Betrieb

### Web Page Analyzer



- Es gibt diverse Tools und Services im Internet um Webseiten zu analysieren
  - ⇒ z.B. <http://websiteoptimization.com> oder auch das Firefox Addon YSlow
  - ⇒ Liefert Analyse der abgerufenen Dateien (HTML, CSS, Grafiken etc.)
    - mit Anzahl und Größe der Dateien
    - mit geschätzten Downloadzeiten für verschiedene Geschwindigkeiten
    - Gibt Hinweise auf Performance Probleme und Optimierungsmöglichkeiten

#### External Objects

External Object	QTY
Total HTML:	1
Total HTML Images:	0
Total CSS Images:	9
Total Images:	9
Total Scripts:	2
Total CSS imports:	4
Total Frames:	0
Total Iframes:	0

Ergebnis von  
websiteoptimization.com  
für [www.fbi.h-da.de](http://www.fbi.h-da.de)  
07/2009 (Auszug)

#### Download Times\*

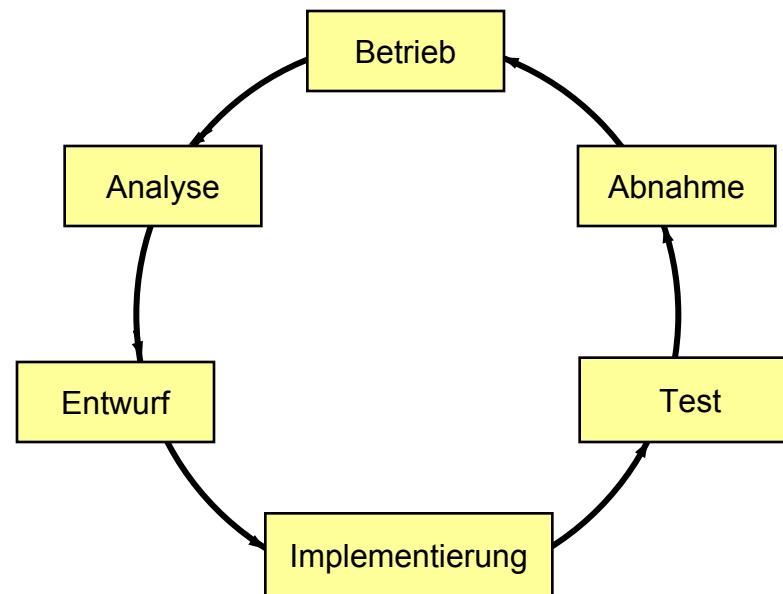
Connection Rate	Download Time
14.4K	42.99 seconds
28.8K	23.09 seconds
33.6K	20.25 seconds
56K	13.43 seconds
ISDN 128K	6.33 seconds
T1 1.44Mbps	3.47 seconds

- **TOTAL\_IMAGES** - Caution. You have a moderate amount of images on this page (9). Consider using fewer images on the site or try reusing the same image in multiple pages to take advantage of caching. Using CSS techniques such as colored backgrounds, borders, or spacing instead of graphic techniques can help reduce HTTP requests.

# Hochschule Darmstadt

## Fachbereich Informatik

### 6.2.7 Zusammenfassung



## Arbeitstechniken in der Webentwicklung

- Es wurden Vorgehensmodelle und Arbeitstechniken für die Webentwicklung vorgestellt
  - ⇒ Die Vorgehensmodelle erleichtern eine iterative und inkrementelle Entwicklung von Webanwendungen
  - ⇒ Auch wenn die meisten Techniken mit PHP demonstriert wurden – es gibt diese oder ähnliche Techniken auch für Ruby, Java etc.
  - ⇒ Die beschriebene Ansätze werden oft für große Systeme eingesetzt – sie sind aber durchaus auch in kleinen Projekten sinnvoll und einsetzbar (z.B. Unitests, Selenium Tests)

Auch für webbasierte Anwendungen können (und sollten) Sie alles anwenden, was Sie über Softwaretechnik, Software Ergonomie und GUIs gelernt haben!

# Hochschule Darmstadt

## Fachbereich Informatik

### 6.3 Content Management Systeme



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

# Webauftritte mit vielen Bearbeitern

- Viele Webauftritte werden von vielen Anwendern bearbeitet
  - ⇒ z.B. der Auftritt einer Hochschule <http://www.fbi.h-da.de>
  - ⇒ viele Anwender kennen weder HTML noch CSS & Co.  
(und wollen es auch nicht kennen lernen)
  - ⇒ dennoch soll
    - der Inhalt von Laien bearbeitet werden können
    - der Auftritt einheitlich aussehen
    - keine Gefahr bestehen, versehentlich das System zu beschädigen
- Ein (Web) Content Management System ("CMS") bietet diese (und weitere) Funktionalität
  - ⇒ Designer gestalten graphische Elemente und entwerfen Styles
  - ⇒ Programmierer implementieren Stylesheets und Sonderfunktionen
  - ⇒ Autoren schreiben Inhalte in vorgegebene Templates
  - ⇒ Bekannte Vertreter sind z.B. Joomla oder Typo3

## 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

## Informationen zur Person



**Aufgabe** Lehre

**Fachgebiete** Grundlagen der Informatik, Objektorientierte Analyse und Design, Software Engineering, Software Produktlinien

**Telefon** +49 (6151) 16-8424

**Fax** +49 (6151) 16-8935

**E-Mail** [Ralf Hahn](mailto:Ralf.Hahn@h-da.de)

**Büro** Gebäude D 14, Raum 1.08

**Sprechzeiten** bitte per Email anmelden 



Prof. Dr. Ralf Hahn

ruft Editor für dieses  
Element auf

## 6.3 Content Management Systeme

## TYPO3: Frontend II: Editor

**Inhalt:**

Aufgabe  
Lehre

Fachgebiete  
Grundlagen der Informatik, Objektorientierte Analyse und Design, Software Eng

Telefon  
+49 (6151) 16-8424

Fax  
+49 (6151) 16-8935

E-Mail-Link Text  
Ralf Hahn

E-Mail-Link URL  
ralf.hahn@h-da.de 

Büro  
Gebäude D 14, Raum 1.08

Sprechzeiten  
bitte per Email anmelden

Foto  
Ralf\_2012\_150x200.JPG



GIF PNG JPEG  
 Keine Datei ausgewählt.

Eingabemaske gibt Standard-elemente vor

Bild zuerst als Upload zur Verfügung stellen

## 6.3 Content Management Systeme

# TYPO3: Backend



The screenshot shows the TYPO3 Backend interface. On the left, the page tree (content structure) is displayed under the 'WEB' category, specifically under the 'Seite' section. The tree shows a hierarchy of pages, including 'Hahn, Ralf' (containing 'WS 2013/14' and 'Frühere Lehrveranstaltungen'), 'Frühere Lehrveranstaltungen' (containing 'SS 2013', 'SS 2012', 'SS 2011', 'SS 2010', 'SS 2009/10', 'SS 2009', 'SS 2008/09', and 'SS 2008'), and 'Software Engineering' and 'Praxisprojekt II' pages for each semester. On the right, a detailed view of the 'Hahn, Ralf' page is shown. The page contains information about Prof. Dr. Ralf Hahn, including his title ('Lehre'), subject ('Grundlagen der Informatik, Objektorientierte Analyse und Design, Software Engineering, Software Produktlinien'), contact details ('Telefon +49 (6151) 16-8424', 'Fax +49 (6151) 16-8935', 'E-Mail-Link URL ralf.hahn@h-da.de'), office ('Büro Gebäude D 14, Raum 1.08'), and speaking hours ('Sprechzeiten bitte per Email anmelden'). It also features a photo of Ralf Hahn. Below this, there are sections for 'Lehrveranstaltungen im WS 2013/2014' and 'Lehrveranstaltungen im SS 2013', each listing their respective start dates and times.

Vernetzungsstruktur  
der Seiten

Inhalte für verschiedene  
Abschnitte

# TYPO3: Ergebnis

h\_da  
HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES  
fbi  
FACHBEREICH INFORMATIK

>Login ▾ | Backend | Suche | Sitemap | RSS

FACHBEREICH INFORMATIK

Hahn, Ralf

WS 2013/14 ▾ Frühere Lehrveranstaltungen ▾ zurück zu meiner Startseite

h\_da Fachbereich Informatik

**HAHN, Ralf**

Informationsarchitektur

Person

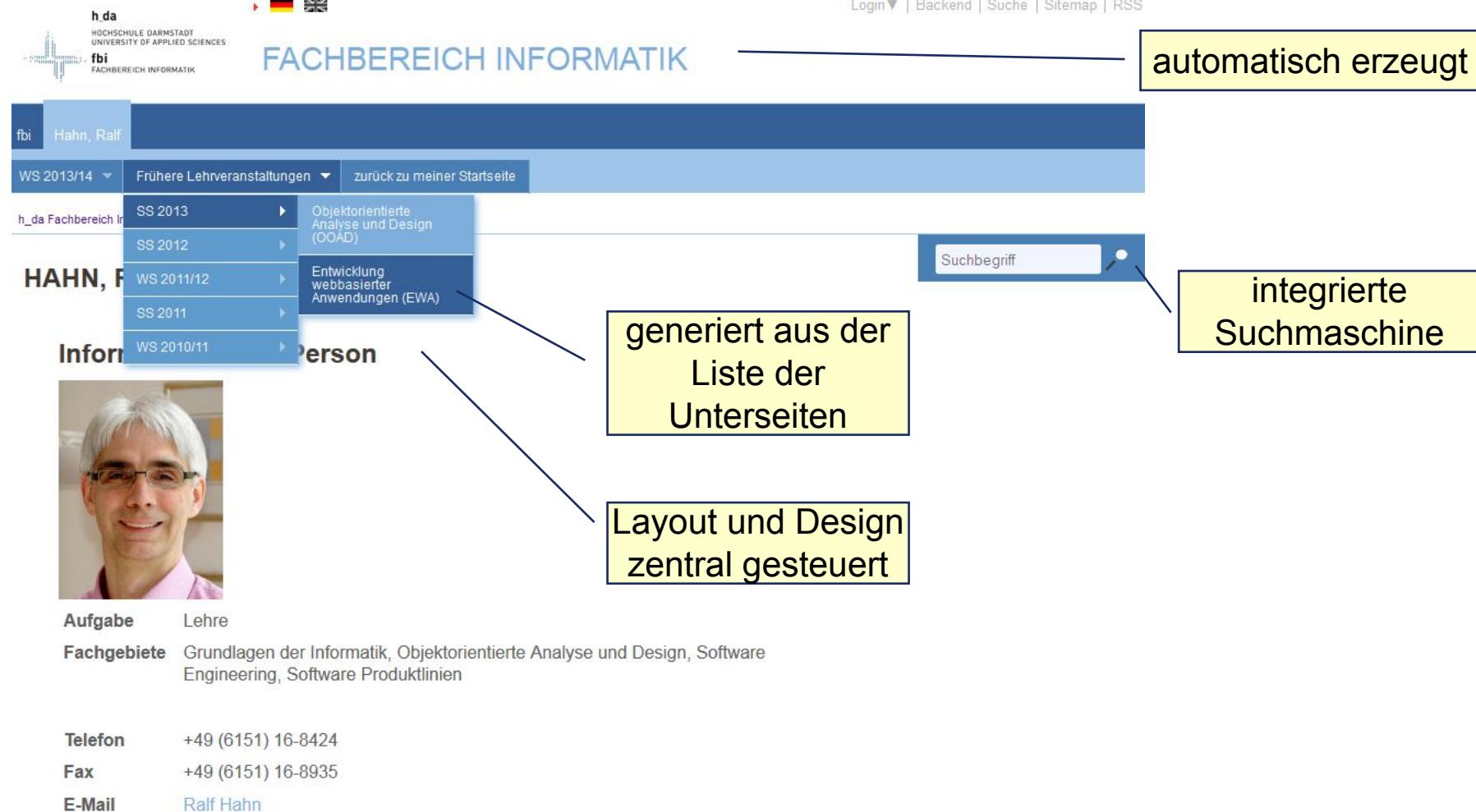
Suchbegriff

automatisch erzeugt

integrierte Suchmaschine

generiert aus der Liste der Unterseiten

Layout und Design zentral gesteuert



**Aufgabe** Lehre

**Fachgebiete** Grundlagen der Informatik, Objektorientierte Analyse und Design, Software Engineering, Software Produktlinien

**Telefon** +49 (6151) 16-8424

**Fax** +49 (6151) 16-8935

**E-Mail** [Ralf.Hahn@h-da.de](mailto:Ralf.Hahn@h-da.de)

# Hochschule Darmstadt

## Fachbereich Informatik

### 7. Zusammenfassung



**h\_da**

HOCHSCHULE DARMSTADT  
UNIVERSITY OF APPLIED SCIENCES

**fbi**

FACHBEREICH INFORMATIK

## 7. Zusammenfassung

# Behandelte Themen

### 1. Einleitung

- 1.1 Softwaretechnik für webbasierte Anw.
- 1.2 Ergonomie für webbasierte Anw.

### 2. Webclient

- 2.1 HTML
- 2.2 CSS
- 2.3 ECMA Script
- 2.4 AJAX

### 3. Webserver

- 3.1 Webserver Software
- 3.2 CGI
- 3.3 PHP

### 4. Zwischen Webclient und Webserver

- 4.1 HTTP
- 4.2 Sessionverwaltung

### 5. Sicherheit

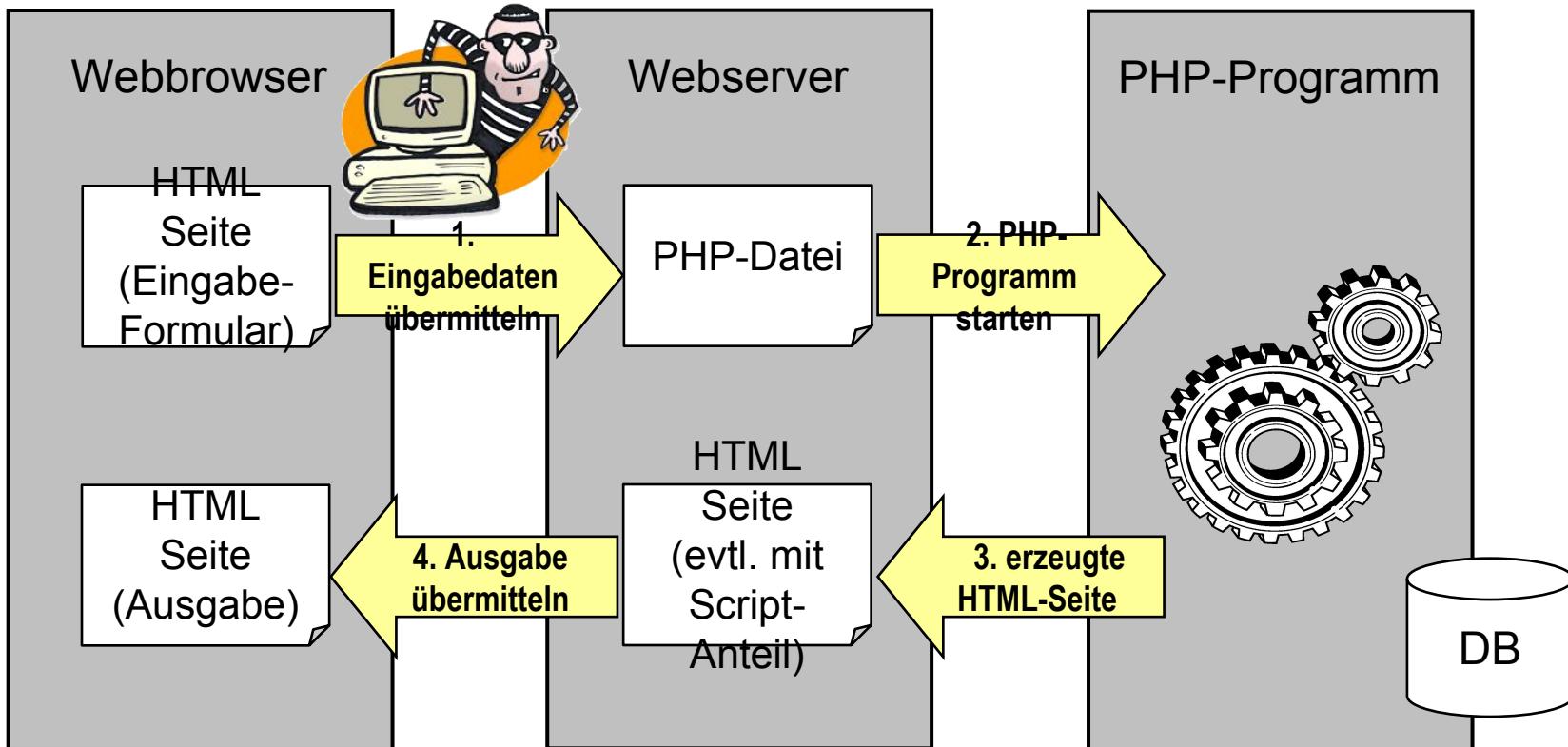
### 6. Professionelle Webentwicklung

- 6.1 Vorgehensmodelle in der Webentw.
- 6.2 Arbeitstechniken in der Webentw.
- 6.3 Content Management Systeme

### 7. Zusammenfassung

## 7. Zusammenfassung

# Einsatz der Technologien im Zusammenhang



- HTML
  - CSS
  - ECMA-Script
  - DOM
  - AJAX
- HTTP
  - Sessions
  - Server-Konfiguration

- CGI
- PHP
- MySQL

## Zielsetzung erreicht?

### ■ aus der Modulbeschreibung:

- ⇒ Die Studierenden sollen
  - Aktuelle Auszeichnungssprachen kennen und anwenden
  - Skriptsprachen für client- und serverseitige Webprogrammierung anwenden
  - ein Dokument Objekt Modell verstehen
  - die Architektur webbasierter Client/Server-Anwendungen mit Datenbankanbindung verstehen
  - Methoden und Techniken zur Entwicklung webbasierter Anwendung
  - Sicherheitsaspekte im Kontext von Webanwendungen verstehen
- ⇒ 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
  - haben Sie die Grundlagen, um sich in diverse andere Web-Techniken einzuarbeiten

