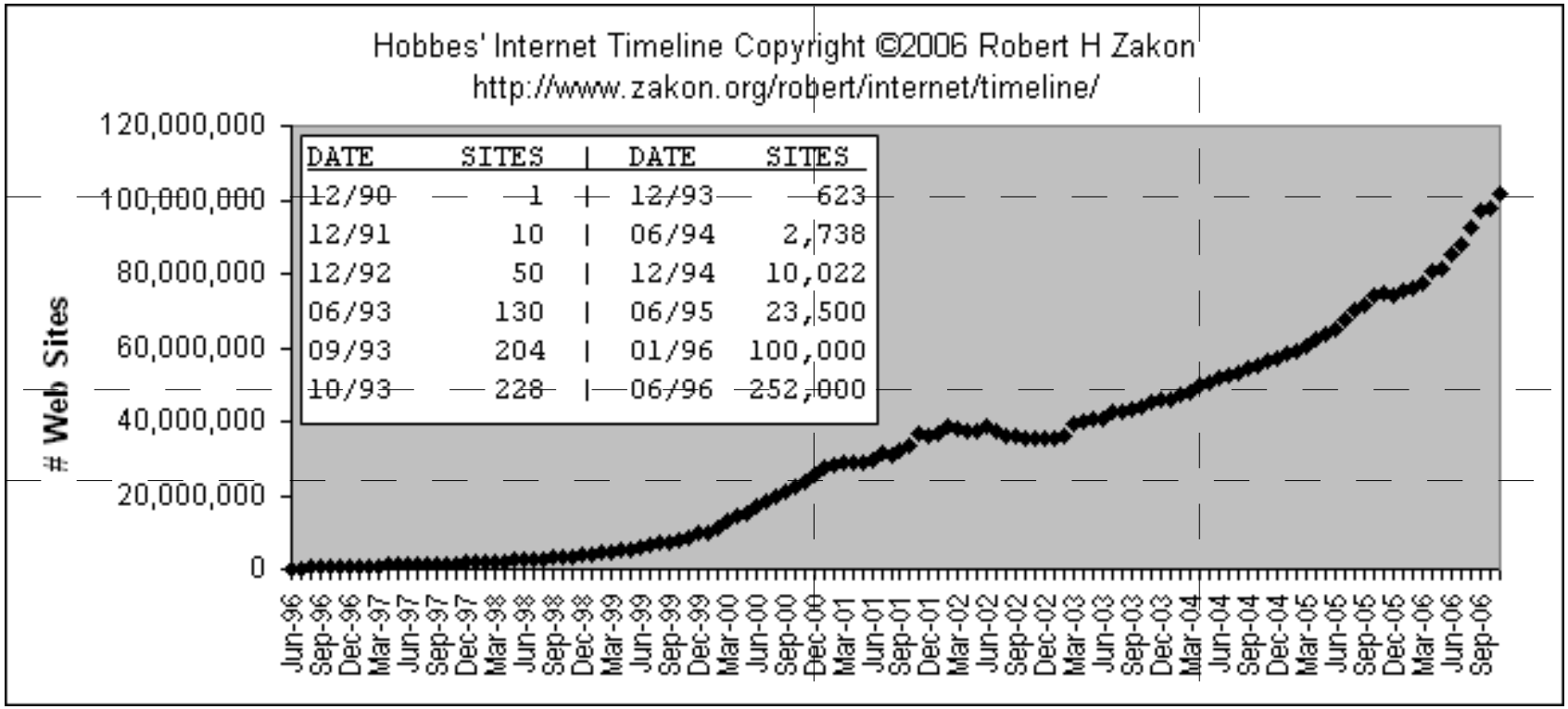
World Wide Web

(„Web 1.0“)

Geschichte

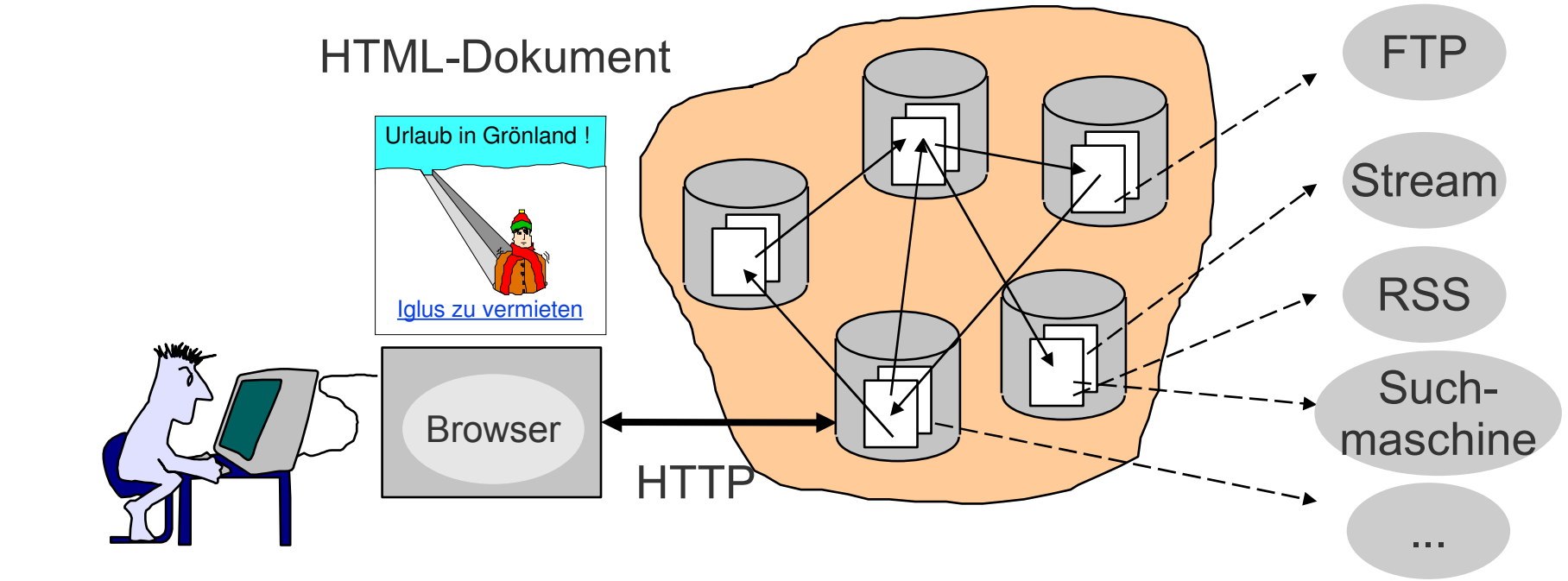
03/89	Erster Vorschlag von Tim Berners-Lee am CERN in Genf für ein „Netz“ von Dokumenten à la Hypertext
12/91	Erster rein textbasierter Prototyp gezeigt auf der Konferenz „Hypertext '91“ in San Antonio, TX
12/92	weltweit ca. 50 WWW-Server
02/93	erstes grafisches Browser-Interface (Mosaic) von Marc Andreessen vorgestellt
1994	Gründung von Netscape durch M. Andreessen
	Gründung des WWW-Konsortiums (www.w3.org)
	Erste internationale WWW-Konferenz in Genf
1995	Börsengang von Netscape für 1.5 Milliarden US $
heute	„WWW = Internet“

Anzahl der WWW-Server



DATE	SITES	DATE	SITES
12/90	1	12/93	623
12/91	10	06/94	2,738
12/92	50	12/94	10,022
06/93	130	06/95	23,500
09/93	204	01/96	100,000
10/93	228	06/96	252,000

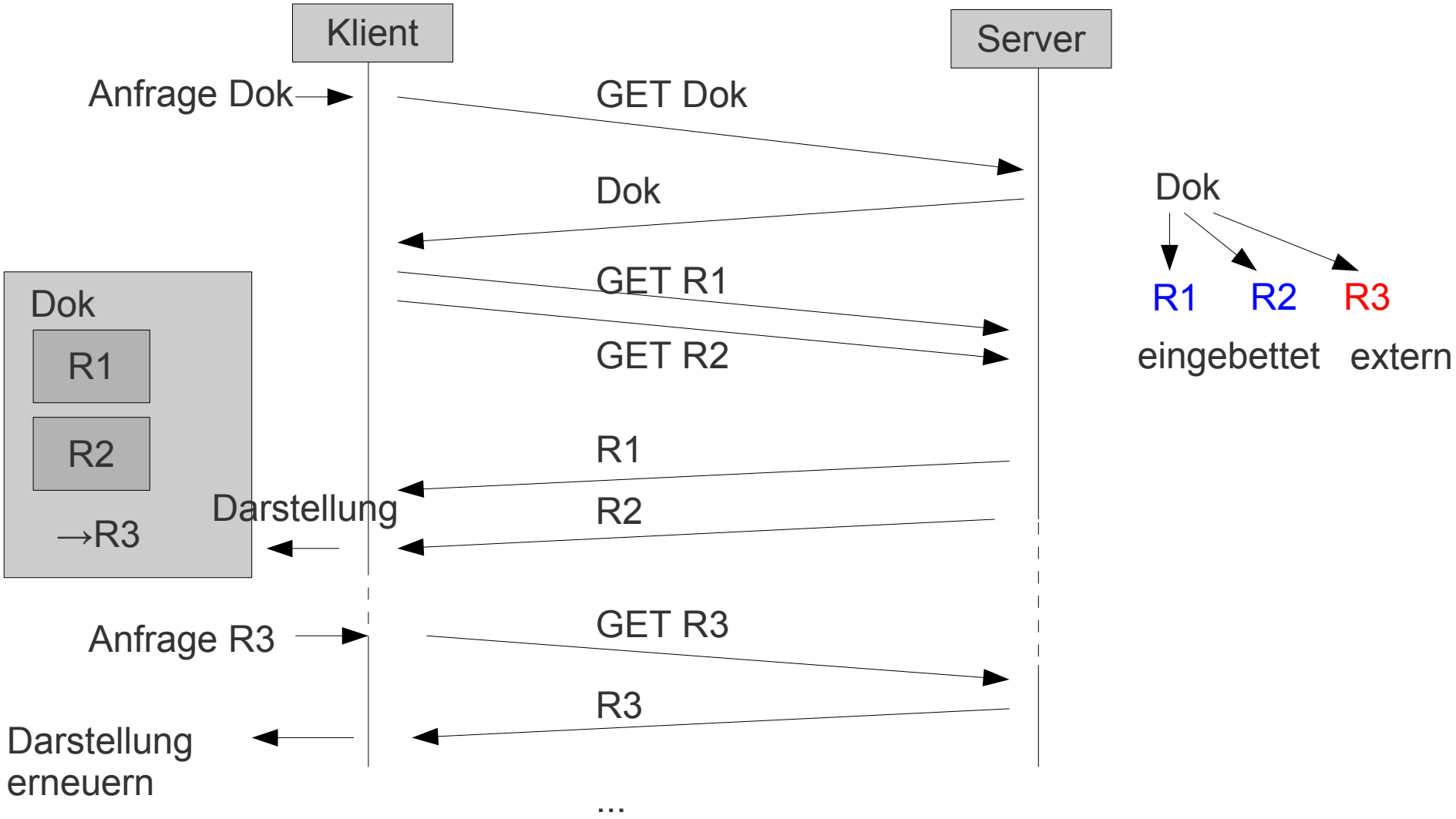
Architektur



Grundlegende Technik des WWW

- Ressourcen
  - allgemein für „nutzbare Datenquellen“: Text, Bilder, Ton, Film, Dateien...
- Jede Ressource beinhaltet Daten, die von einem Klientenprogramm („Browser“) geeignet darzustellen sind.
  - dazu müssen die Browser ggf. erweitert werden („Plugin“)
- Ressourcen können auf andere Ressourcen verweisen
  - mit dem „Uniform Resource Locator“ (URL)
- Referenzierte Ressourcen
  - können eingebunden sein (Bilder)
  - bei Benutzeraktion („Anklicken“) geladen werden

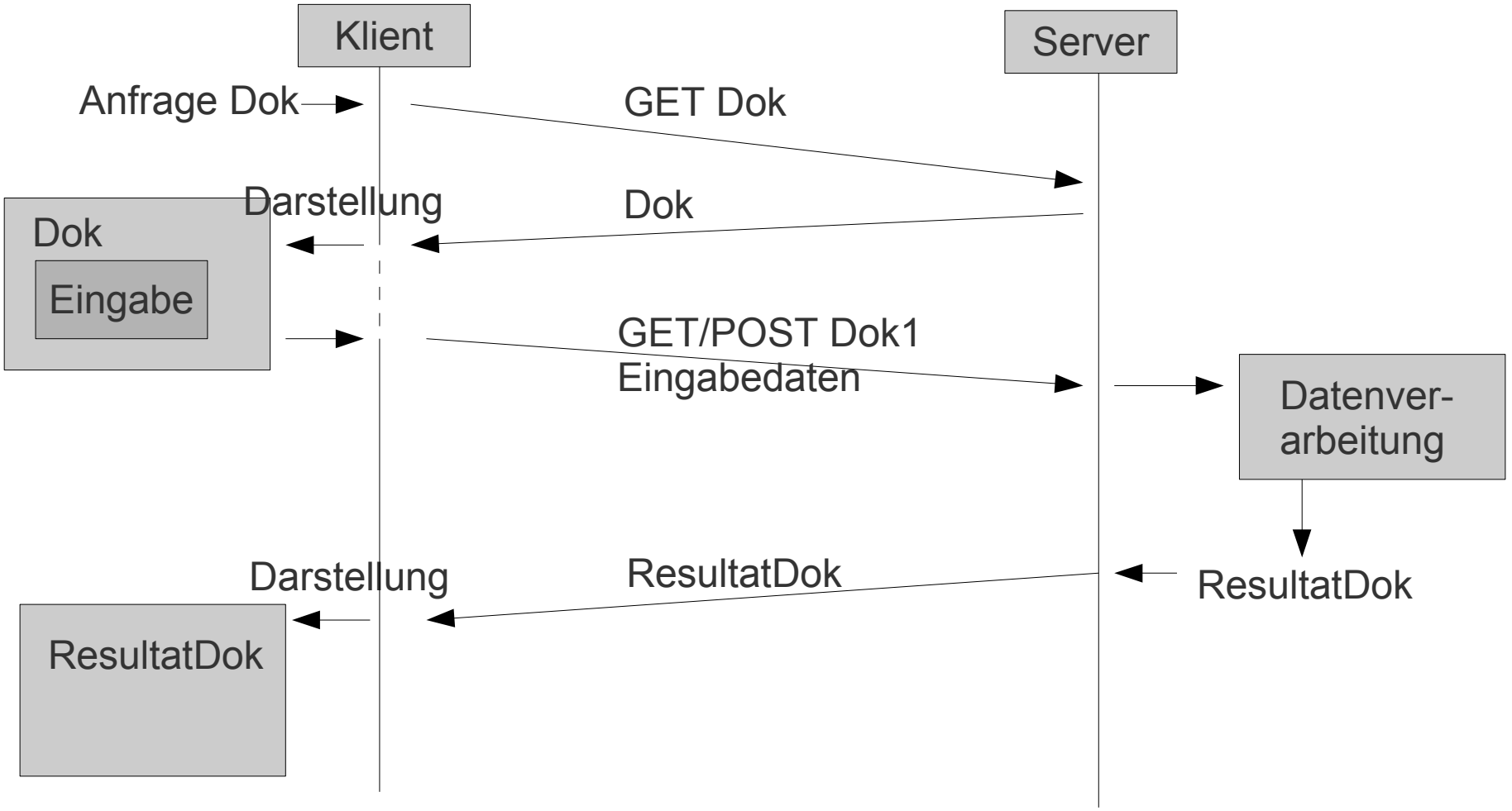
Einfache Ressourcenabfrage



Ressourcen

- Ressourcen können zeitlich veränderlich sein
  - nicht nur Dateien zum Herunterladen
  - Server liefert sein aktuelles Ergebnis zur Anfrage des Klienten; kann bei der nächsten, identischen Anfrage anders aussehen
  - Anwendung: z.B. Nachrichtenticker
- Klient kann Seite wiederholt laden
  - Aktualisierung ohne Benutzerinteraktion
- Klient kann geladenen Code ausführen
  - also mehr als nur statisches Darstellen
  - „Anwendung aus dem Web“

Dateneingabe durch Klient



→ Benutzerinteraktion, Formulare

Architektur: Komponenten

- Wie werden Dokumente übertragen?
  → *Hypertext Transfer Protocol (HTTP)*
- Wie werden Dokumente adressiert?
  → *Uniform Resource Locators (URL)*
- Wie werden Dokumentinhalte dem Benutzer zugänglich?
  → *Web-Browser und Einbindung externer Viewer*
- Wie werden Inhalte beschrieben?
  → *Hypertext Markup Language (HTML)*

Hypertext Markup Language

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
         "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
        <meta HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html; charset=utf-8">
        <title>Startseite von Michael Zapf</title>
        <meta NAME="GENERATOR" CONTENT="Selfmade by Michael Zapf">
        <meta NAME="CREATED" CONTENT="20020209;22562800">
        <link REL="stylesheet" TYPE="text/css" HREF="std.css">
</head>
<body>
<table class="noborder">
<tr>
<td class="upperleft"> </td>
<td class="upperright"> </td>
</tr>
<tr>
<td class="lowerleft">
<img src="hintenlinks1.jpg" alt="linker Balken">
</td>
<td class="lowerright"><p class="space"> </p>
<h1>Zu Hause bei Michael Zapf</h1>
<table class="noborder">
<thead> ...
```

Hypertext Markup Language

- Beschreibung der logischen Struktur **und** des Layouts von Dokumenten
  - Vordefinierte, nicht erweiterbare Menge von Marken („tags“) zur Bezeichnung von Dokumentteilen
  - logische Struktur des Dokuments nur unzureichend abbildbar
  - keine präzisen Formatierungsbefehle; konkretes Layout bestimmt Klient
  - spezielle Marken in HTML verweisen auf andere Seiten („Verweis“, „link“)

HTML

- Basiert auf SGML (Standard Generalized Markup Language)
  - ISO-Standard 8879
  - Markenmenge flexibel erweiterbar für benutzerdefinierte Markierungen
- Versionen

1.0	1993	Textauszeichnungen: fett, kursiv
2.0	1995	Formulare
3.0	nicht std.	
3.2	1997	Tabellen, Textfluss, Applets
4.0	1997	Stylesheets, Skripte, Rahmen; Aufteilung in strict, frameset, transitional
4.01	1999	Aktuelle HTML-Version
5	2007	Arbeitspapier

HTML-Erweiterungen

- Spracherweiterungen, u.a.
  - Rahmen (Netscape: Frames)
  - Browsersteuerung (Netscape: Javascript)
  - Aktive Elemente (Microsoft: ActiveX, Sun: Java)

  → Unstandardisierter Wildwuchs:
  nur bestimmte Browser verstehen die Erweiterung

  → Webseiten werden für Auflösungen und Webbrowser „optimiert“!
- Nicht standardkonforme Dokumente verursachen Probleme
- Dokumente können auf Konformität mit Standard geprüft werden:
  http://validator.w3.org/

XHTML

- HTML 4.01 als letzte HTML-Version
  - Eindämmung des Wildwuchses
  - Missbilligung von rein präsentationsbezogenen Elementen
  - aufgeteilt in Varianten Strict/Transitional/Frameset
  - http://www.w3.org/TR/html401/
- Aufräumen der Definition durch Neuformulierung in XML
  - XHTML (aktuell 1.1): http://www.w3.org/TR/xhtml11/
    - 2.0 als Entwurf (2007)
  - Dokumentaufbau muss sich an striktere Regeln halten
    - Marken immer paarig, Attribute immer Name=Wert usw.
  - enthält alle Elemente von HTML 4.01 – und ist im Prinzip erweiterbar

HTML-Struktur

- Textdatei, in der Teile mit Marken („Tags“) ausgezeichnet sind
  - Marken sind paarig (<Marke>...</Marke>) und können beliebig geschachtelt sein
    - <marke1>...<marke2>...</marke2></marke1>
  - Manche Marken können einzeln auftauchen*
    - <p>, <img>,
 ...
  - Manche Marken erfordern Attribute
    - Name=Wert oder einfach Name
    - <a href=“datei.html“>

*nicht in XHTML

Grundstruktur

Kopf

Inhalt

```
<HTML>
 <HEAD>
  <META NAME="author" CONTENT="fritz@vs">
  <TITLE>Titel der Seite</TITLE>
 </HEAD>

 <BODY>
  <H1>Überschrift</H1>
  Eigentlicher Text
 </BODY>
</HTML>
```

Auszeichnung mit Marken

- Beispiele für HTML-Tags

`<HTML>`	Start einer HTML-Seite
`<HEAD>`	Kopf der Seite
`<TITLE>`	Titel der Seite
`<BODY>`	Rumpf der Seite
`<H1>, <H2>, ...`	Überschrift der Stufe 1,2,... („heading“)
`<B>`	fett drucken („bold“)
`<UL>`	ungeordnete Liste („unordered list“)
`<OL>`	geordnete Liste
`<LI>`	Listenelement („list item“)
` `	neue Zeile („break“)
`<A>`	Hyperlink auf andere Seite („anchor“)
`<IMG>`	Laden eines Bildes
`<PRE>`	vorformatierter Text („preformatted“)

Auszeichnung mit Marken

```
Dies ist ein <b>fetter</b> Text.
```

Dies ist ein **fetter** Text.

```
Dies ist ein <a href="neueSeite.html">Verweis</a>.
```

Dies ist ein Verweis.

```
<UL>
  <LI>Element 1
  <LI>Element 2
  <LI>Element 3
    <OL>
      <LI>Element 3.1
      <LI>Element 3.2
    </OL>
</UL>
```

- Element 1
- Element 2
- Element 3
  1. Element 3.1
  2. Element 3.2

Auszeichnung mit Marken

- Tags für Tabellen (jeweils einige Attribute möglich):

`<TABLE> ... </TABLE>`	Tabellendefinition
`<TR> ... </TR>`	Tabellenzeile
`<TH> ... </TH>`	Zeilen- oder Spaltenüberschrift
`<TD> ... </TD>`	Datenfelder

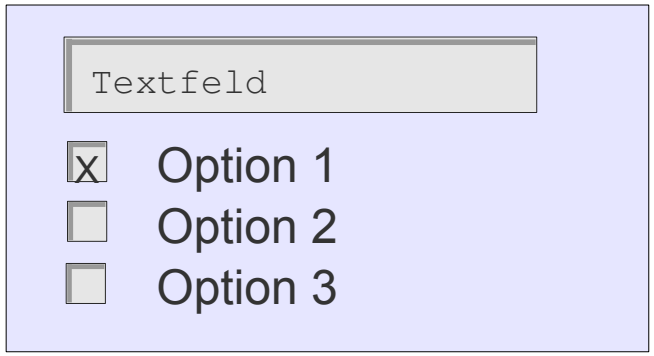
```
<TABLE>
   <CAPTION>Beispiel einer HTML-Tabelle</CAPTION>
   <TR><TH></TH><TH>Spalte 1</TH><TH>Spalte 2</TH></TR>
   <TR><TH>Zeile 1</TH><TD>Feld 1-1</TD><TD>Feld 1-2</TD></TR>
   <TR><TH>Zeile 2</TH><TD>Feld 2-1</TD><TD>Feld 2-2</TD></TR>
</TABLE>
```

Beispiel einer HTML-Tabelle

	Spalte 1	**Spalte 2**
Zeile 1	Feld 1-1	Feld 1-2
Zeile 2	Feld 2-1	Feld 2-2

Benutzerinteraktion: Formulare

- Nutzen die Möglichkeit des Sendens von Daten über HTTP
  - Webserver leitet die Daten an ein externes Programm weiter, z.B. → CGI-Skript
- HTML-Tag <form>
- Aufbau
  - Textfelder („text“), Textzonen („textarea“)
  - Knöpfe („submit“), Radioknöpfe („radio“), Markierboxen („checkbox“)
  - Auswahllisten („select“)
  - Klickknöpfe („button“)
    nur Knopf ohne Datenübertragung
  - Begleitdaten, nicht
    dargestellt („hidden“)



Benutzerinteraktion

- Versteckte Felder (hidden)
  - z.B. Sitzungsinformationen, die nicht dargestellt werden, aber dem Empfängerskript zugänglich gemacht werden sollen
- Beispiel eines Formulars

```
<FORM ACTION="http://www.a.com/cgi-bin/submit.pl" METHOD="GET">
  <INPUT TYPE="text" NAME="name" SIZE="30">
  <INPUT TYPE="submit" VALUE="Senden">
  <INPUT TYPE="reset" VALUE="Neu">
  <INPUT TYPE="hidden" name="session_id" value="423914556">
...
</FORM>
```

Klicken auf den Knopf erzeugt eine GET-Abfrage auf Rechner www.a.com:

GET /cgi-bin/submit.pl?name=Michael&session_id=423914556 HTTP/1.1

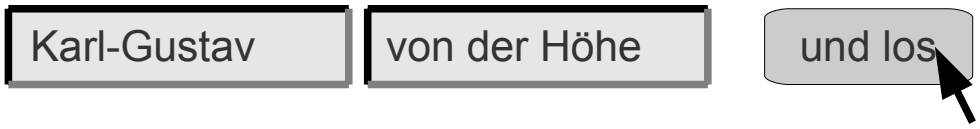
Benutzerinteraktion

- GET-Methode
  - Alle Felddaten werden an die URL gehängt
    http://server/empfaenger.cgi?Vorname=michael&Auswahl=pizza_1&Anmerkung=Bitte+noch+warm+ausliefern
  - Empfängerskript muss den Parameterstring analysieren
  - begrenzte Länge (nicht spezifiziert, abhängig vom Browser)
- POST-Methode
  - Felddaten befinden sich in der Nutzlast der HTTP-Anfrage
  - Empfängerskript bekommt den Datenstrom über die „Standardeingabe“ (d.h. als wären die Daten lokal eingetippt worden)

Beispielformular

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html><head><title>Formulartest</title>
</head><body>

<h1>URL-Encoded</h1>
<form action="http://ruegen.vs.eecs.uni-kassel.de/cgi-bin/meincgi"
          enctype="application/x-www-form-urlencoded" method="post">
   <input type="text" name="Vorname"/> <br>
   <input type="text" name="Nachname"/> <br>
   <input type="submit" name="und los"/>
</form>

<h1>Form data</h1>
<form action="http://ruegen.vs.eecs.uni-kassel.de/cgi-bin/meincgi"
           enctype="multipart/form-data" method="post">
   <input type="text" name="Vorname"/> <br>
   <input type="text" name="Nachname"/> <br>
   <input type="submit" name="und los"/>
</form>

</body></html>
```

Vorgabe, wenn weggelassen



Formulardatenübergabe bei POST

- Was über die Standardeingabe zu lesen ist

x-www-form-urlencoded:

```
Vorname=Karl-Gustav&Nachname=von+der+H%C3%B6he&und+los=Absenden
```

form-data:

```
-----------M3qbeDDGYKyth6n45LxxcBleLJfbJSnfUdF6oSMNmIGRP1VUMRPwcBZ
Content-Disposition: form-data; name="Vorname"

Karl-Gustav
-----------M3qbeDDGYKyth6n45LxxcBleLJfbJSnfUdF6oSMNmIGRP1VUMRPwcBZ
Content-Disposition: form-data; name="Nachname"

von der Höhe
-----------M3qbeDDGYKyth6n45LxxcBleLJfbJSnfUdF6oSMNmIGRP1VUMRPwcBZ
Content-Disposition: form-data; name="und los"

Absenden
-----------M3qbeDDGYKyth6n45LxxcBleLJfbJSnfUdF6oSMNmIGRP1VUMRPwcBZ--
```

Weiteres zur Verarbeitung → Abschnitte CGI, Servlets

Erstellung von HTML-Dokumenten

- Einfacher Text wird mit Marken angereichert
- per Hand eingefügt
  - Vorteil: Übersichtliche Gestaltung, sparsamer Einsatz, Flexibilität
  - Nachteil: Fehler durch falsche oder vergessene Marken
- HTML-Editoren und -Exportfilter
  - Vorteil: Gestaltung der Webseite bequem
  - Nachteil: Meist völlig mit Marken überladene Erzegnisse, kaum verständlich – und nur schwer zu editieren.
- Trend: CMS
  - Content-Management-Systeme (Typo3, Joomla, ...)
  - Nur noch Inhalte einfügen
  - Layout wird vollständig von CMS organisiert

Darstellung und Inhalt

- Elemente beziehen sich auf Format oder Inhalt

```
<b>fett</b>
<i>kursiv</i>
<br>Zeilenumbruch
<font size="7">Zeichengröße</font>
```

formatbezogen

```
<strong>betont</strong>
<em>hervorgehoben</em>
<p>Absatz
<h1>,<h2>,... Überschriften
```

inhaltsbezogen

Es ist möglich, dieselbe Darstellung mit unterschiedlichen Mitteln zu erreichen, obwohl diese Mittel im Prinzip nicht gleichwertig sind.

Darstellung und Inhalt

- Entkopplung von Darstellung und logischem Aufbau
  - Vorteil: Darstellung kann für verschiedene Endgeräte leicht angepasst werden, ohne das Dokument strukturell zu ändern
- HTML dient **nicht** zum Realisieren eines bestimmten, festen Layouts
  - Informationen werden entsprechend der logischen Dokumentstruktur geeignet durch einen Browser dargestellt
  - Benutzer kann Einfluss auf Darstellung nehmen (z.B. kleinere Zeichengrößen verbieten)

Darstellung und Inhalt

- Verwendung von HTML zunehmend im gewerblichen / privaten Bereich
  - aber damit andere Erwartungshaltung!
    - Dokument muss so und nicht anders aussehen
    - Viele Zusatzelemente, Interaktivität
- Aufwändige Layouts überfordern Autor
  - Hilfsprogramme, Webdesign-Programme
  - aber: Logische Struktur kann nur vom Autor bestimmt werden
  - Physische Darstellung unmittelbar zu erfassen

Da die Neigung besteht, eine Seite in einer bestimmten Weise *aussehen* zu lassen, wird mehr Gewicht auf das Layout als auf die Struktur gelegt.

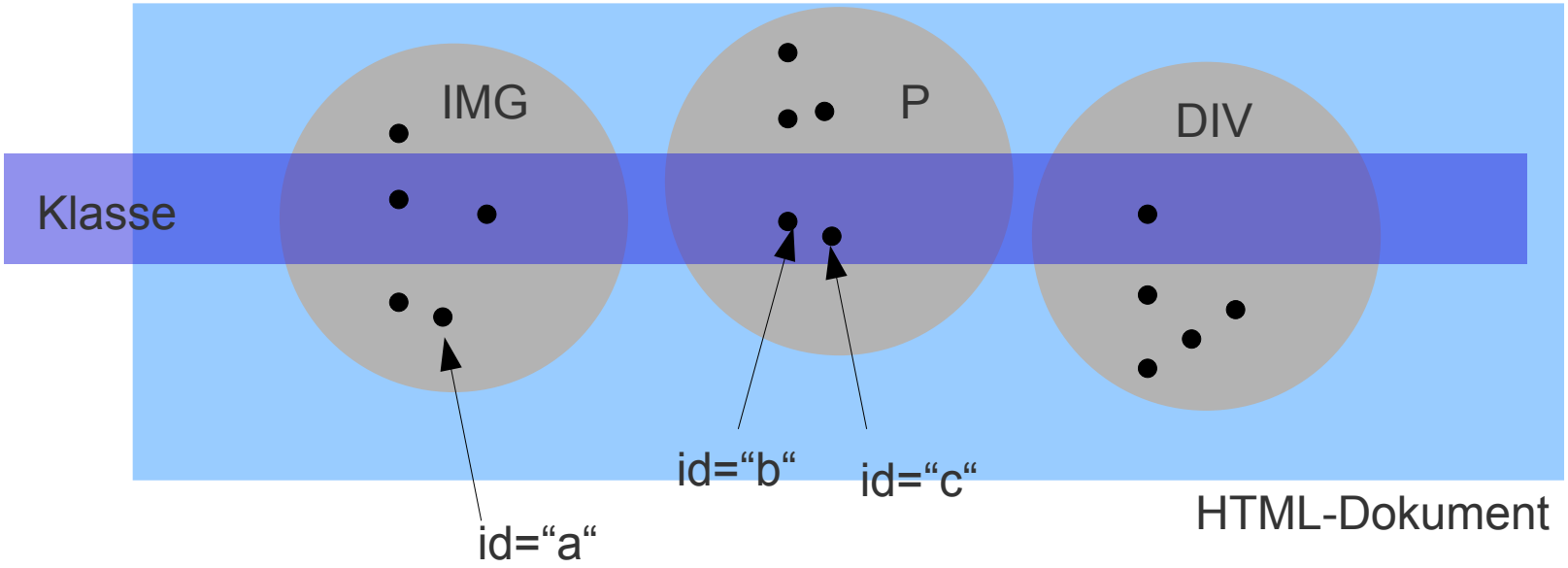
- HTML sieht anpassendes Layout vor
  - Hilfsprogramme versuchen, ein bestimmtes Layout zu erzwingen
  - Komplizierte Konstruktionen mit geschachtelten Tabellen

Cascading Stylesheets

- Ergänzender Standard zu HTML
- Präsentation der HTML-Elemente definieren
  - Abstände, Schriftart, Größe, Farbe, Fettdruck etc.
- CSS häufig noch in Version 1.0 anzutreffen
  - Version 2.0 berüchtigt für fehlerhafte Implementierungen
  - Neuer Kandidat 2.1: http://www.w3.org/TR/CSS21
- Ziel
  - Präsentation völlig von Inhalt abkoppeln
  - Verschiedene Ausgabemedien: visuell, Sprache/Ton, Braille, ...

Cascading Stylesheets: Idee

- Stilparameter werden für HTML-Elemente definiert
- Allgemeines Format
  - CSS-Definition besteht aus einer Liste von Einträgen der Form
    Selektor { (Name: Wert;)* }
  - Selektor referenziert einen oder mehrere Marken (Tags) im HTML-Dokument
  - Nutzung des „class“- und „id“-Attributs



Selektoren

*	jedes Element
E	jedes Element des Typs E (z.B. IMG, DIV, ...)
E F	jedes Element des Typs F, das in einem E eingeschachtelt ist
E>F	jedes Element des Typs F, das Kind von E ist
E:first-child	das erste Kind von E
E[attr]	jedes Element des Typs E, das ein Attribut attr hat
E.C	jedes Element des Typs E aus der Klasse C
E#ID	das Element mit der gegebenen ID*

Eigenschaften der umschließenden Klasse
werden „geerbt“:

```
p { color:red; }
p.c1 { background-color:yellow; }
```

→ Text wird dargestellt als: Text

*ID ist eindeutig *je Element*

CSS-Beispiel

```
<html>
<head>
   <style type="text/css">
      h1 {font-size:48pt; color:#FF0000; font-style:italic;}
      h1.hinterlegt {background-color:#FFFF00;}
      *.hinterlegt {background-color:#FF0000;}
   </style>
</head>
<body>
      <h1 class="hinterlegt">Hinterlegte Überschrift</h1>
      <p class="hinterlegt">Hinterlegter Abschnitt</p>
</body>
</html>
```

Hinterlegte Überschrift

Hinterlegter Abschnitt

Einbindung von CSS

```
<html>
<head>
    <title>Titel der Datei</title>
    <link rel="stylesheet" type="text/css" href="style.css"/>
    <style type="text/css">
        h1 { font-size:48pt; font-weight:bold }
        *.normal { color:black; font-family:serif }
    </style>
</head>
<body>
    <h1>Eine Überschrift</h1>
    <p class="normal" style="color:#FF0000;">
    Ein gewöhnlicher Abschnitt</p>
</body>
</html>
```

Stylesheet-Datei

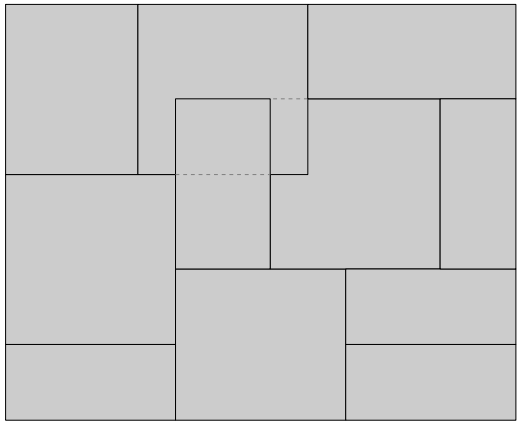
Zentrale Definition

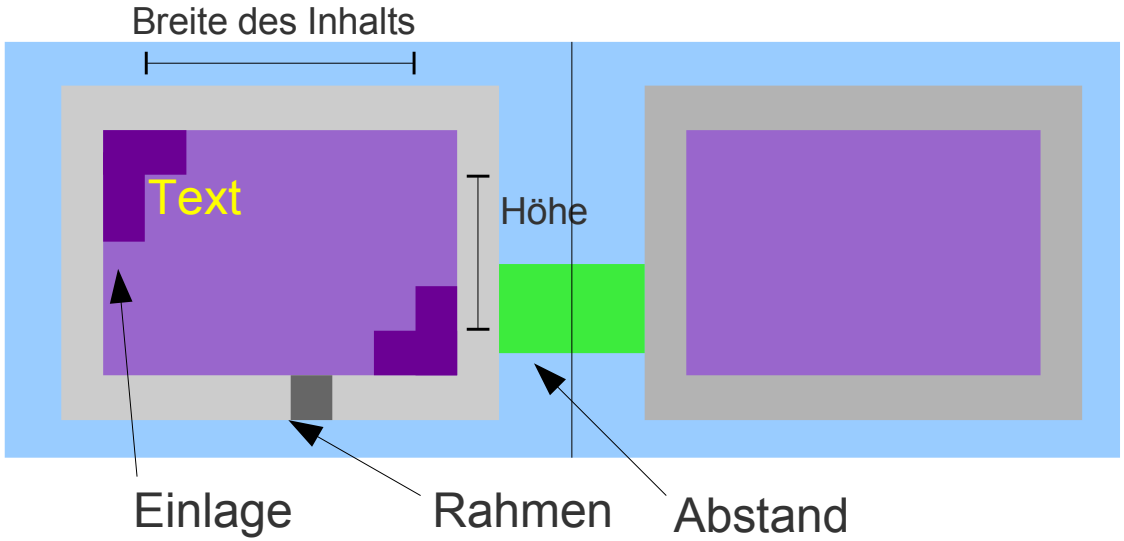
„inline“

Vorrang: Datei < zentral < inline

CSS-Layout

- Box-Modell
  - Seite stellt sich als Anordnung von Boxen dar
  - Boxen können sich überlappen
  - Boxen können Text beinhalten
  - Boxen können positioniert werden (absolut oder relativ)





Gesamtbreite=
Breite
+ Rahmen (2x)
+ Einlage (2x)

(entsprechend Höhe)

Elemente des CSS (Auswahl)

Zeichensatz	font-weight font-size color ...	Bestimmt das Aussehen des Textes font-weight: bold
Ausrichtung	vertical-align text-align ...	Bestimmt die Ausrichtung von Text innerhalb eines Blockelements oder die gegenseitige Ausrichtung text-align: center
Abstand	margin ~ -top ~ -bottom ~ -left ~ -right	Bestimmt den Abstand zum nächsten Element (Box) margin-bottom: 2ex

Maßeinheiten: ohne=px=Pixel, em=Schriftgröße, ex=Größe des Buchstabens „x“, %, pt, cm,...
em/ex ermöglichen Angaben abhängig von der Schriftgröße, also sehr praktisch, wenn der Betrachter eine andere Schriftgröße wählt.

Elemente des CSS (Auswahl)

Einlage	padding ~ -top ~ -bottom ~ -left ~ -right	Bestimmt die Ausmaße der Einlage padding-top: 10em
Rahmen	border(-width/-color/-style) ~ -top(-width/-color/-style) ~ -bottom~ ~ -left~ ~ -right~	Rahmenstil, Farbe, Dicke border: 10px solid red
Hintergrund	background-color ~ -image ...	Hintergrundbild wird per URL eingebunden background-image:url(http://server/image.jpg) background-color:#aa5034

Farbe als #RRGGBB

Elemente des CSS (Auswahl)

Listen	list-style ...	Kästchen als Listenpunkte list-style: square
Tabellen	table-layout border-collapse empty-cells ...	Gemeinsame Linien zwischen Zellen nur einmal zeichnen border-collapse:collapse
Pseudoelemente und -klassen	:link :visited :focus :hover :active :first-line ...	Beim Überfahren des Links mit dem Mauszeiger das Aussehen ändern: a:hover { background-color: yellow } Auch für andere Elemente geeignet h1:hover { font-weight: bold }

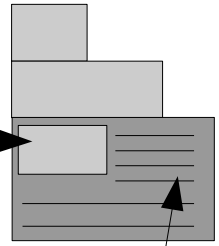
Elemente des CSS (Auswahl)

Positionierung

position
(min-/max-)width
(min-/max-)height
float
clear

Boxen positionieren (absolut oder relativ)
div { position:absolute; top:10px ... }

Boxen umfließen lassen
float:left



Inhalt (Text, Bilder usw.)
vermeidet schwebende Box

ferner
- Audioausgabesteuerung
- Drucksteuerung
- Mauszeiger
- Bildbearbeitungsfilter (nur Internet Explorer)

CSS-Probleme

- Nicht alle Browser unterstützen alle Eigenschaften oder interpretieren sie richtig
  - IE-Fehler (bis 5.5) im Box-Modell: Gesamtbreite und -höhe falsch berechnet (es werden Rahmen und Einlage nicht addiert) → Layout auf verschiedenen Browsern unterschiedlich!
- Manche Elemente sind nicht eindeutig als Präsentation oder Inhalt zu identifizieren
  - insbesondere Tabellen
- erfordert semantischen Entwurf und widerspricht dem intuitiven Vorgehen
  - anders denken: nicht das Aussehen als Ausgangspunkt nehmen, sondern die Bedeutung (Semantik)
  - das klappt schon bei der einfachen Textverarbeitung nicht (Stichwort: Formatvorlagen)

Weitere Informationen über CSS z.B. bei **de.selfhtml.org**

JavaScript

```
<html>
<head>
   <title>Test</title>
</head>
<body>
<script type="text/javascript">
        var Jetzt = new Date();
        var Tag = Jetzt.getDate();
        var Monat = Jetzt.getMonth() + 1;
        var Jahr = Jetzt.getYear();
        var Stunden = Jetzt.getHours();
        var Minuten = Jetzt.getMinutes();
        var NachVoll = ((Minuten < 10) ? ":0" : ":");
        if (Jahr < 2000)
          Jahr = Jahr + 1900;
          document.write("<h2>Guten Tag!<\/h2><b>Heute ist der " +
            Tag + "." + Monat + "." + Jahr + ". Es ist jetzt " +
            Stunden + NachVoll + Minuten + " Uhr<\/b>");
</script>
</body>
</html>
```

de.selhhtml.org

JavaScript

- Von Netscape lizenzierte objektorientierte Skriptsprache;
  - aktuelle Version 1.5
  - Grundfunktionalität als „ECMAScript“ standardisiert
  - Microsoft-Pendant ist **JScript**, konform zu ECMAScript.
- Javascript ermöglicht Aktivität auf Browserseite
  - Keine zusätzliche Kommunikation zwischen Webserver und Webbrowser
  - Weniger bekannt: Server-seitiges Javascript (insbesondere bei Netscape-Servern)
- Nur geringe Ähnlichkeit mit Java
  - Namenswahl aus Marketinggründen

JavaScript

- „Dynamisches HTML“ (DHTML): Manipulation der Darstellung auf Klientenseite
  - HTML + CSS + JavaScript (neuerdings auch HTML + DOM* + JavaScript)
- Ziel von JavaScript
  - Verhalten des Browsers steuern
  - Eigenschaft von Elementen der dargestellten Seite ändern
    - Stilparameter wie Farbe, Schriftart
    - Position
    - Ereignisverarbeitung definieren

*DOM: Document Object Model, siehe Folge „XML“

JavaScript

- Anwendungsbeispiele
  - Testen von Formulardaten auf Korrektheit vor deren Absendung
  - Browsereigenschaften erfragen (Hersteller, Version, Multimediaunterstützung etc.)
  - Auf Nutzer-Events reagieren (z.B. Mausklick)
  - Sich verändernde Web-Seiten (aktuelle Uhrzeit einblenden, Aufklappen/Einklappen von Paragraphen, Bewegen von Objekten usw.)
  - Abschicken von E-Mails

JavaScript

- JavaScript wird direkt in der HTML-Datei oder in separaten Dateien zur Verfügung gestellt
- Browser besitzen eingebauten JavaScript-Interpreter
  - aber wieder viele Inkompatibilitäten / Fehlimplementierungen!
- Ausführung vor Darstellung der Seite, während, oder bei Benutzeraktionen
- Aus Sicherheitsgründen eingeschränkte Rechte (Sandbox); z.B. kein Zugriff auf das Dateisystem

JavaScript

- Beispiel

```
<script type="text/javascript">
  var d = new Date();
  document.write(d.getHours()+":"+d.getMinutes());
</script>
```

- Externe JavaScript-Datei
  - Ermöglicht die Wiederverwendung von Code

produkt.js:

```
function produkt(x,y) {
  var z = x*y;
  return z;
}
```

JavaScript

- Einbindung der externen .js-Datei

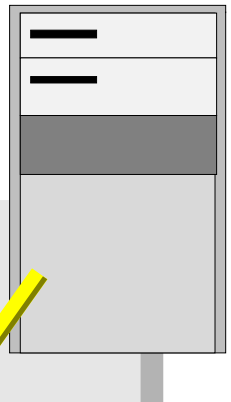
```
<html>
<head>
	<title>JavaScript-Test</title>
	<script src="produkt.js" type="text/javascript">
	</script>
</head>
<body>
	<form name="Form" action="">
		<input type="text" name="x" size="3">
		<input type="text" name="y" size="3">
		<input type="button" value="Ergebnis" onClick='document.Form.p.value=
			produkt(document.Form.x.value,document.Form.y.value);'>
		<input type="text" name="p" size="3">
	</form>
</body>
</html>
```

JavaScript-Problem

- Entreißt dem Anwender die Kontrolle über sein Programm
  - Browsersteuerung ermöglicht das ungewollte Öffnen von Fenstern
  - Modifikation der Maustasten
  - Verhindern des Verlassens der Seite
  - Benutzeraktionen simulierbar
- vorsätzliche Täuschung des Anwenders bezüglich der Webinhalte, insbesondere Ziel von Verweisen
- Lücken hatten schon schwerwiegende Folgen
  - Auslesen der Seitenchronik
  - Ungewolltes Versenden von Mails

Weitere Informationen über JavaScript z.B. bei **de.selfhtml.org**

Serverseitiges Skripting



```
<table class="noborder">
<tr>
<td class="upperleft"> </td>
<td class="upperright"> </td>
</tr>
<tr>
<td class="lowerleft">
<img src="hintenlinks1.jpg" alt="linker Balken">
</td>
<td class="lowerright"><p class="space"> </p>
<h1>Zu Hause bei Michael Zapf</h1>
<table class="noborder">
<thead> ...
```

Server-Side Includes (SSI)

- „Statische“ HTML-Seiten oft nicht ausreichend
- „Dynamische Seiten“ = dynamische Verarbeitung und Generierung von Inhalten auf Serverseite
- Anweisungen in HTML-Seiten
  - Diese werden **vor** Auslieferung vom Webserver interpretiert
  - Ergebnis ist eine gewöhnliche HTML-Seite
- Server muss diese Vorrichtung aktiviert haben
  - sonst werden diese Anweisungen uninterpretiert an den Klienten geschickt

In httpd.conf oder .htaccess (sofern das Überschreiben von Vorgaben erlaubt ist):

```
Options +Includes
AddType text/html .shtml
AddHandler server-parsed .shtml
```

Server-Side Includes

- Allgemeines Format
  <!-- #element attribut1=wert1 attribut2=wert2 ... -->
- Beispiele:
  - Datum/Zeit einblenden
    <!--#echo var="DATE_LOCAL" -->
  - CGI-Skript ausführen
    <!--#include virtual="/cgi-bin/counter.pl" -->
  - Änderungsdatum
    <!--#echo var="LAST_MODIFIED" -->
  - Einbinden einer Datei auf dem gleichen Server
    <!--#include virtual="/footer.html" -->

Common Gateway Interface (CGI)

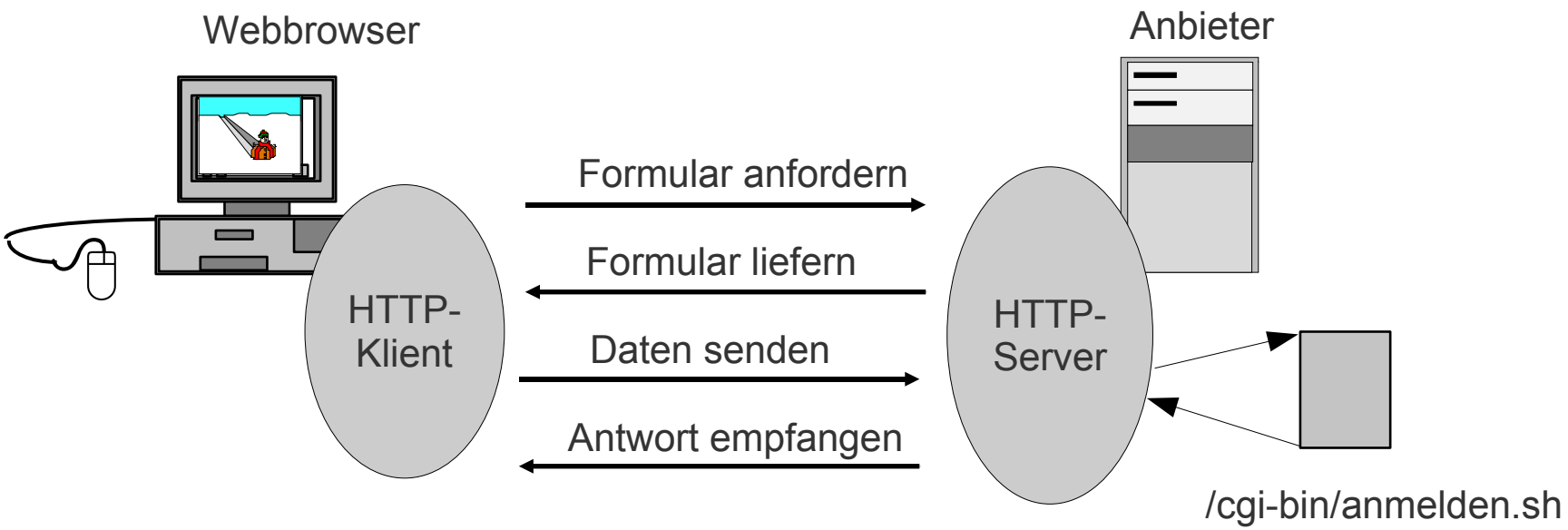
- Flexiblere Form der dynamischen Erzeugung als SSI
- CGI definiert Schnittstelle für den Aufruf von Programmen, die Übergabe von Argumenten und Ergebnissen
  - unterstützt verschiedene Programmiersprachen, z.B. C, C++, TCL, Shells, Perl, Python, ...
  - Einsatzbereiche - Beispiele:
    - Aufbereitung und Bereitstellung von Daten aus Datenbanken
    - Erfassung von Benutzereingaben
    - Gateways zu anderen Diensten

CGI

- Ein CGI-Skript kann auf verschiedene Arten aufgerufen werden:
  - Formular
    - <form action="/cgi-bin/guestbook.pl" method="get">
    - <form action="/cgi-bin/stats.pl" method="post">
  - Verweis
    - Tagesstatistik
  - Grafikreferenz
    - <img src="/cgi-bin/counter.pl">
  - Server-Side Include
    - <!-- #exec cgi="/cgi-bin/counter.pl" -->
  - Automatisches Laden
    - <meta http-equiv="refresh" content="0; URL=/cgi-bin/welcome.pl">

CGI

- Konfiguration und Ablauf



- Ausführbares Programm zur Behandlung der Eingaben liegt per Konvention im Verzeichnis /cgi-bin/
- Aufruf mit Parametern vom Client, Antwort vom Server meist mit dynamisch generierter Seite

CGI-Parameterübergabe

- Übergabe durch GET oder POST
- GET
  - nur MIME-Typ application/x-www-form-urlencoded
  - $QUERY_STRING (Umgebungsvariable, <= 1024 Zeichen) wird an das CGI-Skript übergeben und beinhaltet den URL-Teil hinter dem „?“
  - Wird vom Skript analysiert und verarbeitet
- POST
  - application/x-www-form-urlencoded oder multipart/form-data
  - Einlesen der Parameter aus der Standardeingabe

CGI-Parameterübergabe

- Skript antwortet mit neuer Seite
  - Standardausgabe wird an Anfrager verschickt
- Für HTML muss auch ein Kopf geschrieben werden!
- Beispiel

```
#!/bin/sh
cat > /tmp/test.out
echo "Content-Type: text/html; charset=utf8"
echo ""
echo "<html><body>Ergebnis<pre>"
cat /tmp/test.out
echo "</pre></body></html>"
```

Kopiert Standardeingabe in eine Datei (POST vorausgesetzt)

Kopf

Beispiel: Wir kopieren einfach den Inhalt der Anfrage direkt in die Antwort*

*so wurden die Ausgaben für x-www-form-urlencoded und multipart/form-data gewonnen

CGI: Sicherheitsrisiken

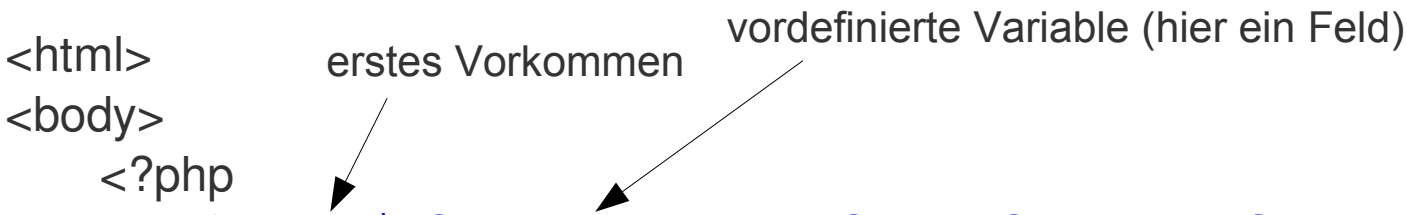
- CGI sind „von jedem“ aufrufbar – keine systematische Sicherheit
- Ausführung eines Programms ohne Benutzerkonto auf der Maschine
- Berechtigung des CGI-Programms ist die Berechtigung des WWW-Servers
  - Server nicht als root starten, sondern als wwwuser
- Problem der Ausführung dynamisch erzeugter System-Kommandos

PHP: Hypertext Preprocessor

- Häufig verwendete Server-seitige Skriptsprache, Open-Source-Software (OSS); http://www.php.net
  - wird oft zusammen mit MySQL (relationale Datenbank) verwendet
  - verfügbar für viele Plattformen (Windows, Linux, Unix, etc.)

```
<html>
    <head>
        <title>PHP-Test</title>
    </head>
    <body>
        <?php echo "<p>Hallo Welt</p>"; ?>
    </body>
</html>
```

PHP und HTML




```
<html>
<body>
    <?php
        if (strstr($_SERVER["HTTP_USER_AGENT"], "MSIE")) {
    ?>
    <h3>strstr hat true zurückgegeben</h3>
    <center>Ihr Browser meldet sich als Internet Explorer</center>
    <?php
        } else {
    ?>
    <h3>strstr hat false zurückgegeben</h3>
    <center>Ihr Browser meldet sich nicht als Internet Explorer</center>
    <?php
        }
    ?>
</body>
</html>
```

PHP und Formulare

- Formular

```
<form action="welcome.php" method="POST">
        Enter your name: <input type="text“ name="name" />
        Enter your age: <input type="text" name="age" />
        <input type="submit" />
</form>
```

- Verarbeiten der Formulardaten

```
<html>
  <body>
    Welcome <?php echo $_POST["name"]; ?>.
    You are <?php echo $_POST["age"]; ?> years old!
  </body>
</html>
```

PHP und Cookies

- Cookie setzen

  <?php setcookie("uname", $name, time()+36000); ?>

- Cookie abfragen

```
<?php
  if (isset($uname))
      echo "Welcome " . $uname . "!<br />";
  else
      echo "You are not logged in!<br />";
?>
```

PHP und Datenbanken

```
<?php
    $conn=odbc_connect('northwind','','');
    if (!$conn) {
        exit("Keine Verbindung: " . $conn);
    }
    $sql="SELECT * FROM customers";
    $rs=odbc_exec($conn,$sql);
    if (!$rs) {
        exit("SQL-Fehler");
    }
    echo "<table><tr><th>Firma</th><th>Kontakt</th></tr>";
    while (odbc_fetch_row($rs)) {
        $compname=odbc_result($rs,"CompanyName");
        $conname=odbc_result($rs,"ContactName");
        echo "<tr><td>$compname</td>";
        echo "<td>$conname</td></tr>";
    }
    odbc_close($conn);
    echo "</table>";
?>
```

Wichtige Rolle in aktuellen Web-Anwendungen!

Active Server Pages (ASP)

- ASP ist Microsofts Technologie für *Server-Side Scripting*
- keine eigene Sprache, sondern Umgebung zum Einsatz von Skriptsprachen
- Baut auf Windows Scripting Host auf
  - Visual Basic Script (VBScript), JScript oder Perl möglich
  - VBScript ist üblich

ASP.NET

- ASP.NET ist die nächste Generation von ASP
- Teil von Microsofts .NET-Framework
- ASP.NET stellt Dienste für die Erzeugung, Bereitstellung und Ausführung von Web-Anwendungen und Web Services zur Verfügung
- Verwendung von allen durch .NET unterstützten Sprachen (Visual Basic, C#, C++, JScript, …)
- Entwicklung von Web-Anwendungen mit „Web-Forms“- Programmiermodell

ASP.NET-Beispiel

```
<%@Page language="c#" %>
<html>
<head>
    <script runat="server">
        public void B_Click (object sender, System.EventArgs e) {
            Label1.Text = "Hello, the time is " + DateTime.Now;
        }
    </script>
</head>
<body>
    <form method="post" runat="server">
        <asp:Button onclick="B_Click" Text="Drück mich" runat="server“ /> <p>
        <asp:Label id=Label1 runat="server" />
    </form>
</body>
</html>
```

Java

Grundlagen

- Java ist eine Objekt-orientierte Sprache
- Einfachvererbung von Klassen, Mehrfachimplementierung von Schnittstellen
- Java-Programme werden meist in Bytecode übersetzt und von einer virtuellen Maschine (VM) durch einen Bytecode-Interpreter ausgeführt
- hierdurch wird (zu einem gewissen Grad) Plattform-Unabhängigkeit sichergestellt
- lediglich die VM muss auf dem Zielsystem implementiert werden

Grundlagen

- keine neue Idee, siehe z.B. p-code bei UCSD Pascal
- Kompilierung in Maschinensprache mittels Just-in-Time Compiler (JIT)
  - früher optional, jetzt Standard
- gut geeignet für mobilen Code (mobile Agenten etc.)

Java-basierte Internettechnologien

- Java-Applets
  - Java-Programme, die in Webseiten referenziert werden
    - früher mittels dediziertem Tag (APPLET), zwischenzeitlich EMBED, heute allgemeiner über OBJECT
  - werden vom Browser geladen und einem „Plug-in“ zur Ausführung zugeführt; Darstellung meist innerhalb der Seite
- Java-Servlets
  - CGI auf „Java-Art“
  - erweitern Web-Server um dynamische, Java-basierte Seiten
  - werden über URLs angesprochen und liefern Seiteninhalt
  - werden vom Web Server instantiiert und ausgeführt

Java-basierte Internet-Technologien

- Java Server Pages (JSP)
  - Erweiterung der Servlet-Technologie zur einfachen Anfertigung von Web-Seiten, die statischen und dynamischen Inhalt mischen
- XML Server Pages (XSP)
  - Erweiterung der Servlet-Technologie durch Produktion eines XML-Dokuments durch ein Java-Programm, vermischt mit direktem XML-Code
  - wird anschließend mittels „XSL-Transformation“ zu HTML konvertiert

Applets

- Kurzform des Kunstworts „applicationlet“ - „Prográmmchen“
- Ermöglichen Aktivität auf Client-Seite
- Haben meist eine GUI (AWT, Swing, etc.)
- Können (fast) die gesamte Java-API nutzen

Applets

- Einbettung über (spezielle) APPLET-Marke

```
<applet code="Album.class"
        archive="Album.jar"
        align=left width=820 height=800 mayscript>
   <param name="PICWIDTH" value="320"/>
   <param name="PICHEIGHT" value="240"/>
   <param name="PICDPI" value="62.5"/>
Achtung: Dieser Text wird angezeigt, wenn der Browser kein Java
unterstützt, oder wenn die Unterstützung abgeschatet ist.
</applet>
```

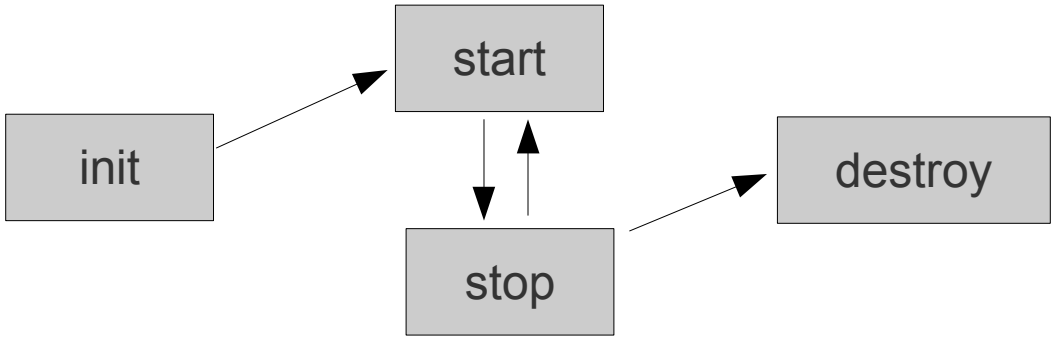
Applets

- Einbettung über OBJECT-Marke (ab HTML 4.0, soll APPLET ablösen)

```
<object classid="java:meinPackage.ViewerApplet.class"
        archive="ViewerApplet.jar"
        codetype="application/java-vm" standby="Lade Applet"
        width="800"
        height="600">
  <param name="baseurl" value="http://server/sma/"/>
  <param name="savedir" value="saved"/>
  <param name="port" value="10000"/>
</object>
```

Technik der Applets

- *java.applet.Applet* definiert so genannte Lebenszyklus-Methoden, die überschrieben werden müssen



public void init(): Seite wird erstmalig aufgerufen
public void start(): Seite mit Applet wird angezeigt
public void stop(): Seite mit Applet wird ersetzt/geschlossen
public void destroy(): Applet wird entsorgt

Applet-Beispiel

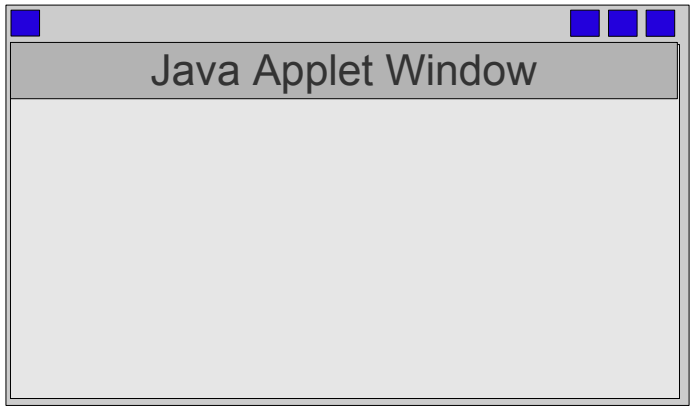
```
public class Simple extends Applet {
  StringBuffer m_sbBuffer = new StringBuffer();

  public void init()    { addItem("init... ");   }
  public void start()   { addItem("start... ");  }
  public void stop()    { addItem("stop... ");   }
  public void destroy() { addItem("destroy..."); }

  void addItem(String sNewWord) {
    m_sbBuffer.append(sNewWord);
    repaint();
  }

  public void paint(Graphics g) {
    g.drawString(m_sbBuffer.toString(), 5, 15);
  }
}
```

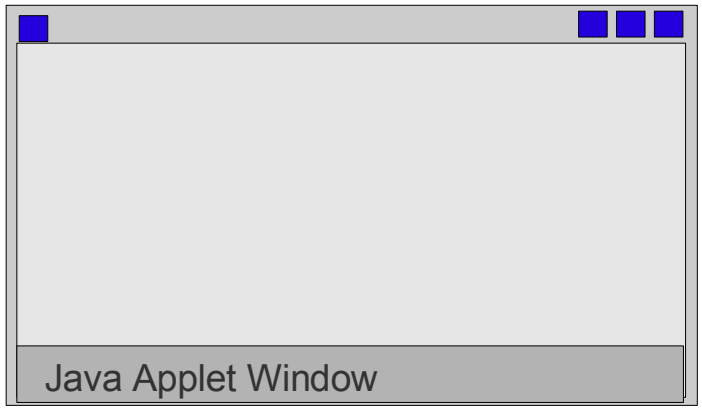
Applets und Sicherheit

- Ausführung von geladenen Applets birgt offensichtliche Risiken
- Die Java-Sprache enthält einige Sicherheitsvorkehrungen
  - keine Zeiger
  - Typsicherheit zur Laufzeit (nicht nur zum Zeitpunkt der Übersetzung)
  - echte Feldgrenzen mit Überprüfung
  - Byte Code Verifier kann geladenen Bytecode verifizieren

Applets und Sicherheit

- Applets unterliegen standardmäßig weiteren Sicherheitsvorkehrungen
  - kein Zugriff auf Dateien
  - Kommunikation nur mit dem Ursprungsrechner des Applets
  - bestimmte Systemparameter können nicht ausgelesen werden
  - Eigenständige Applet-Fenster sind speziell gekennzeichnet





LiveConnect

- Ermöglicht die Kommunikation zwischen Java, Javascript und Plugins (ab Netscape 3.0)
- Java → JavaScript

```
<APPLET CODE="MyApplet1.class"
NAME="MyApplet" WIDTH=150 HEIGHT=25 MAYSCRIPT></APPLET>
```

```
import netscape.javascript.*;
...
public writePage(String page) {
  JSObject win = JSObject.getWindow(applet);
  win.eval("document.open();");
  win.eval("document.write('"+page+"');");
  win.eval("document.close();");
}
```

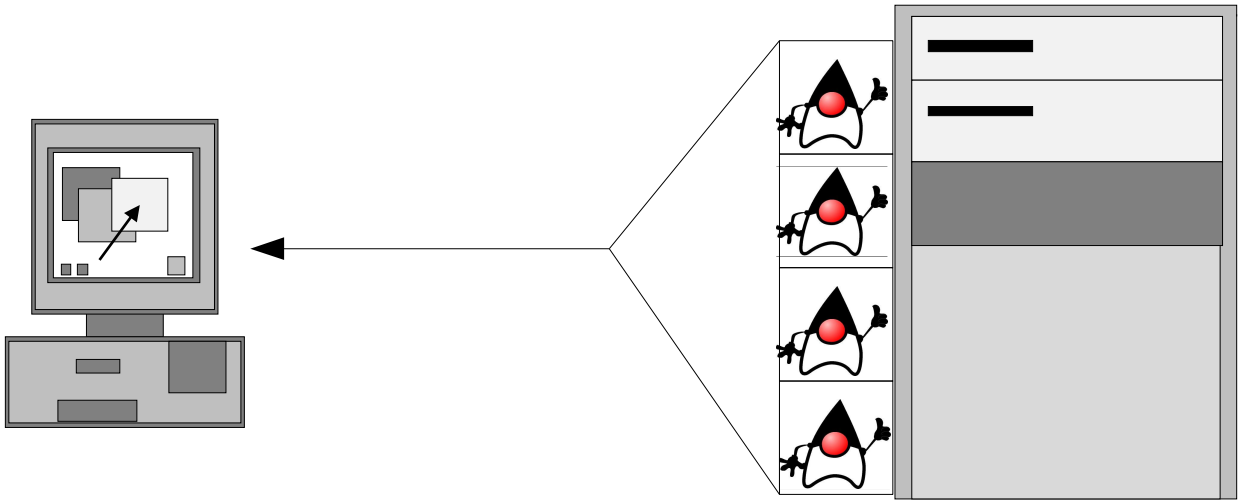
LiveConnect

- Javascript → Java

```
public MyApplet1 extends Applet {
        public void setString(String s) {...}
        ...
}
```

```
<APPLET CODE="MyApplet1.class" NAME="MyApplet"
  WIDTH=150 HEIGHT=25></APPLET>

<FORM NAME="form">
  <INPUT TYPE="button" VALUE="Set"
      onClick="document.MyApplet.setString(document.form.str.value)">
  <INPUT TYPE="text" SIZE="20" NAME="str">
</FORM>
```

Servlets

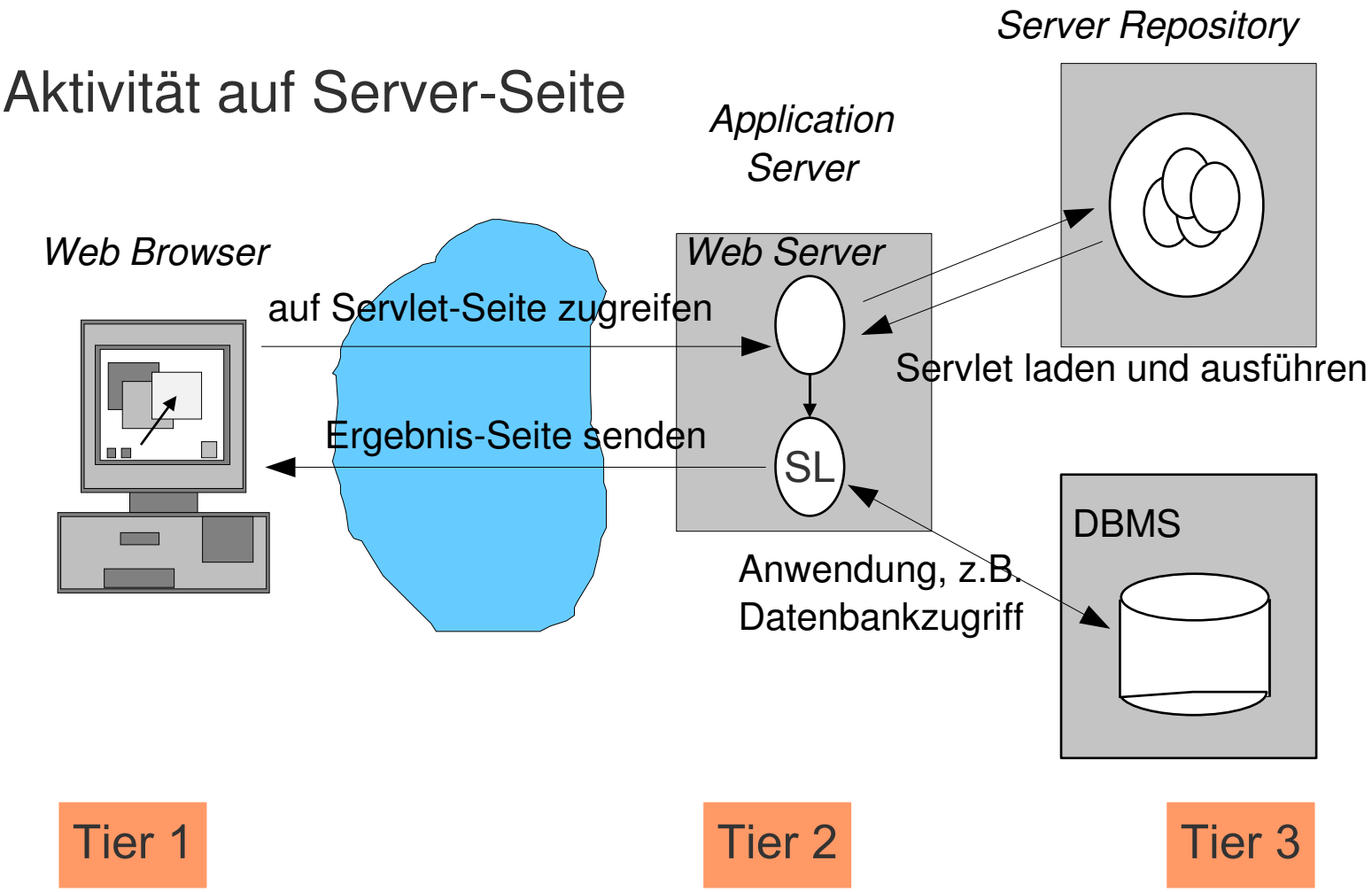


Servlets

- Servlets = CGI auf „Java-Art“
  - http://java.sun.com/products/servlet/
  - zustandslose Java-Klassen, die über einen Web Server mittels einer URL angesprochen werden
- Ausführungsschritte
  - Analyse der Request-Parameter
  - Bearbeiten des Request
  - Erzeugen der Ausgabe (Response)
- benutzen Container-Dienste (z.B. Session-Management, Authentifizierung, Autorisierung)

Servlet-API ist Bestandteil der Java 2 Platform Enterprise Edition

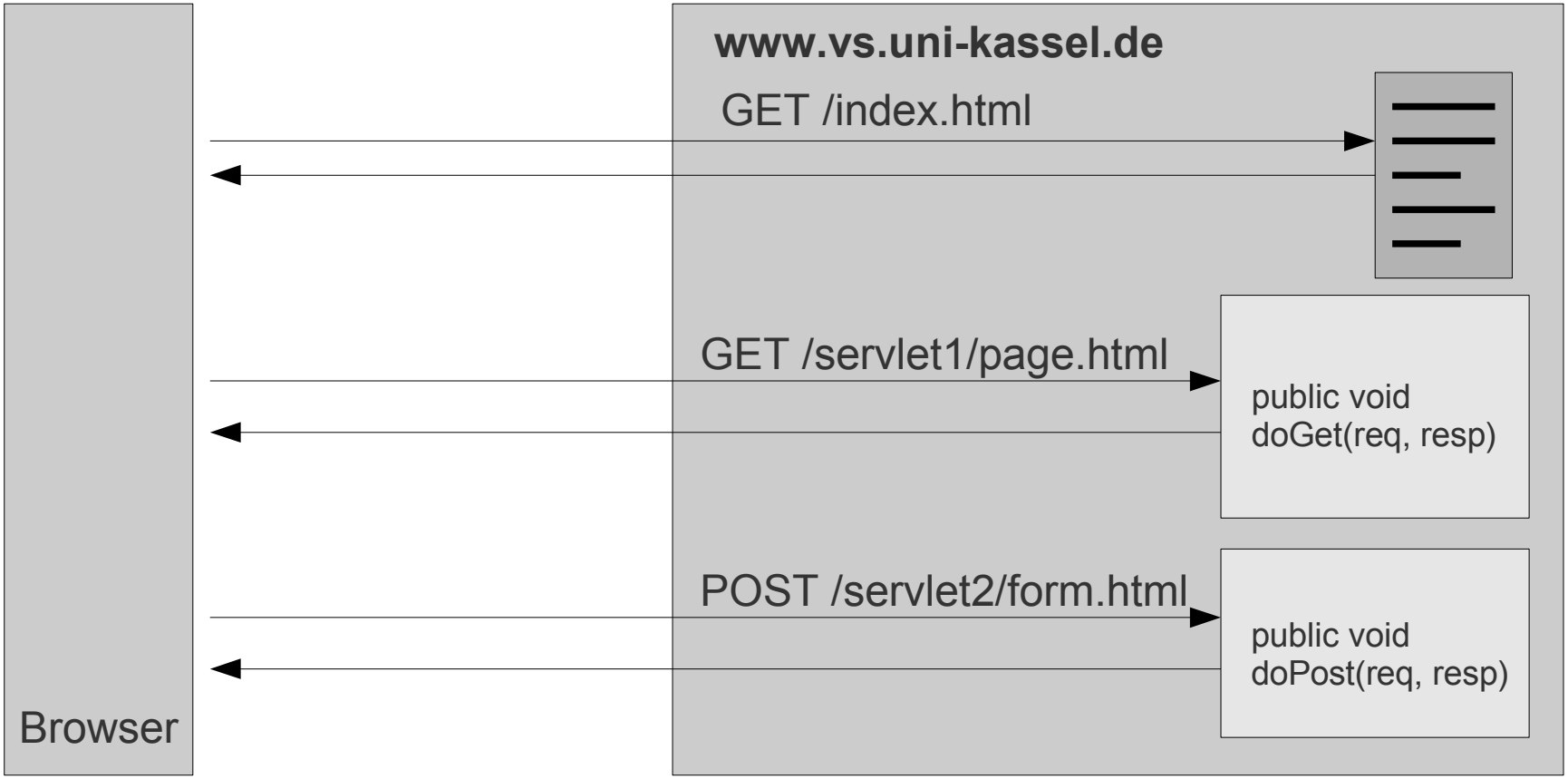
Servlets


Server Repository
Aktivität auf Server-Seite
Application Server
Web Browser
Web Server
auf Servlet-Seite zugreifen
Servlet laden und ausführen
Ergebnis-Seite senden
SL
DBMS
Anwendung, z.B. Datenbankzugriff
Tier 1
Tier 2
Tier 3

HTTP-Servlets

- Implementieren **javax.servlet.http.HttpServlet**
  - Abgeleitet von ***GenericServlet***, welches ***javax.servlet.Servlet*** implementiert
- Überschreiben von Methoden für die HTTP-Operationen
  - **GET → public void doGet(HTTPServletRequest, HTTPServletResponse)**
  - **POST → public void doPost(HTTPServletRequest, HTTPServletResponse)**
  - **...**
- HTTP-Request und -Response als Parameter

HTTP-Servlets



HTTP-Servlets

- Lebenszyklus-Methoden (geerbt von **GenericServlet**)
  - init() Initialisierung des Servlets
  - destroy() Freigeben benutzter Ressourcen
- ***javax.servlet.http.HttpServletRequest***
  - Repräsentiert die HTTP-Anfrage vom Klient
  - Properties (Getter/Setter-Methoden)
    - Enumeration getAttributeNames()
    - String getAttribute(String name)
    - void setAttribute(String name, Object o)...

HTTP-Servlets

- ***javax.servlet.http.HttpServletResponse***
  - Repräsentiert die HTTP-Antwort an den Klient
  - Inhalte
    - getWriter()
    - setContentType(String type)
    - setContentLength(int len)
  - Statuscode, Fehler, Umlenkung
    - setStatus(SC_NO_CONTENT)
    - sendRedirect(“http://www.vs.uni-kassel.de/new”)

Beispiele

```
public class ServletA extends HttpServlet {
  public void doGet(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {
    PrintWriter out = res.getWriter();
    res.setContentType("text/html");
    out.println("<html><head>");
    out.println("<title>Servlet-Beispiel</title>");
    out.println("</head><body>");
    out.println("Datum und Zeit:");
    out.println(new java.Util.Date());
    out.println("</body></html>");
    out.flush();
  }
}
```

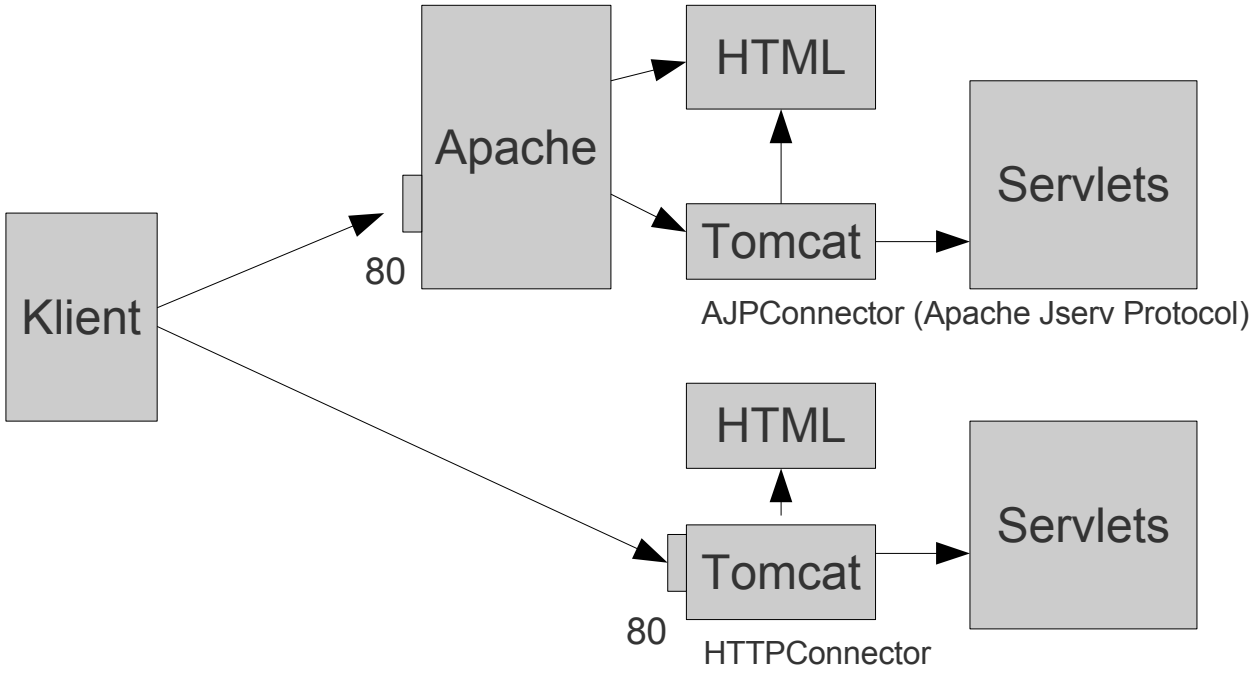
Beispiele

```
<FORM METHOD="POST"
  ACTION="/servlet/FormServlet">
  <INPUT NAME="firstname" TYPE="text" SIZE="30">
  <INPUT NAME="lastname"  TYPE="text" SIZE="30">
  <INPUT NAME="age"       TYPE="text" SIZE="3">
  <INPUT TYPE="SUBMIT">
</FORM>
```

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException {
   firstname = request.getParameter("firstname");
   lastname  = request.getParameter("lastname");
   ageString = request.getParameter("age");
   ...
}
```

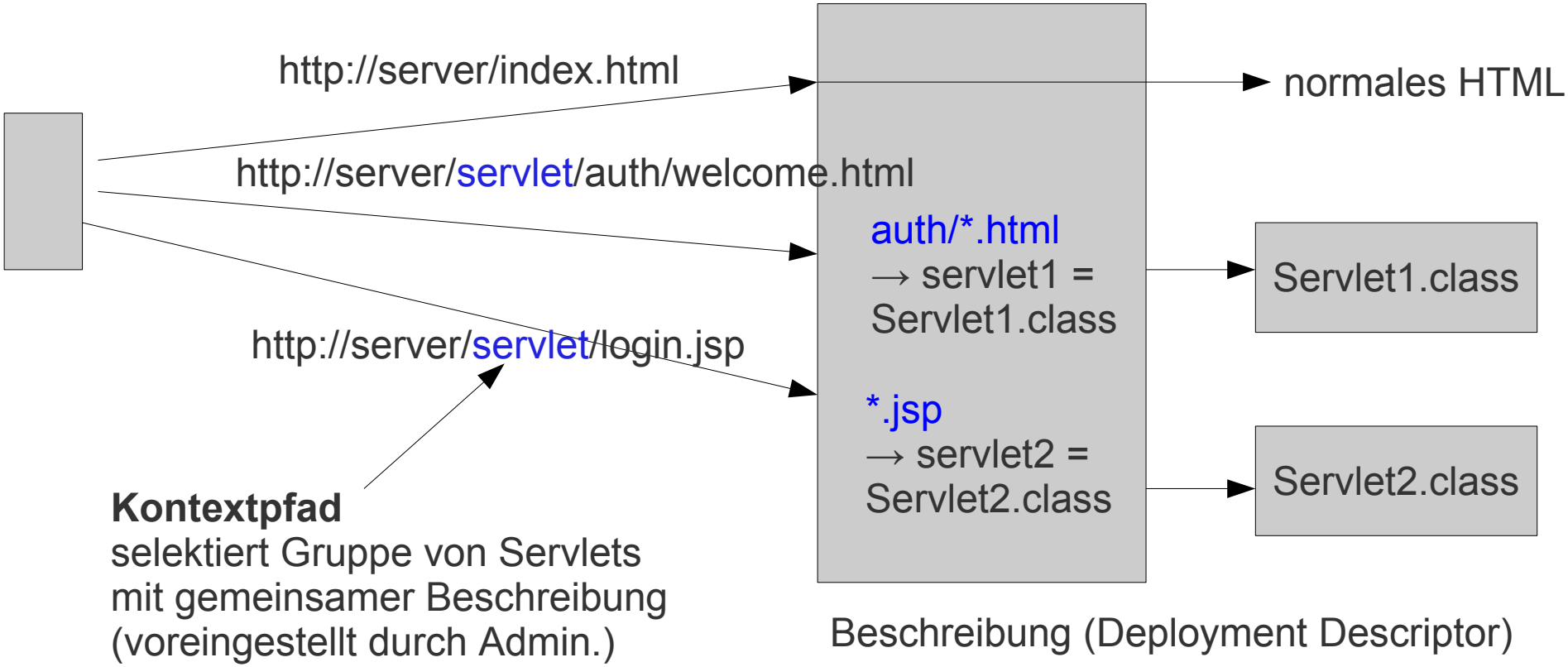
Servlet-Betrieb

- Servlets werden in einem „Servlet-Container“ ausgeführt
- Verbreitete Implementierung: **Tomcat**
  - http://tomcat.apache.org/
  - kann eigenständig als Webserver laufen oder mit Apache-Webserver gekoppelt werden



Servlet-Container

- Selektion des Servlets anhand der Anfrage-URL



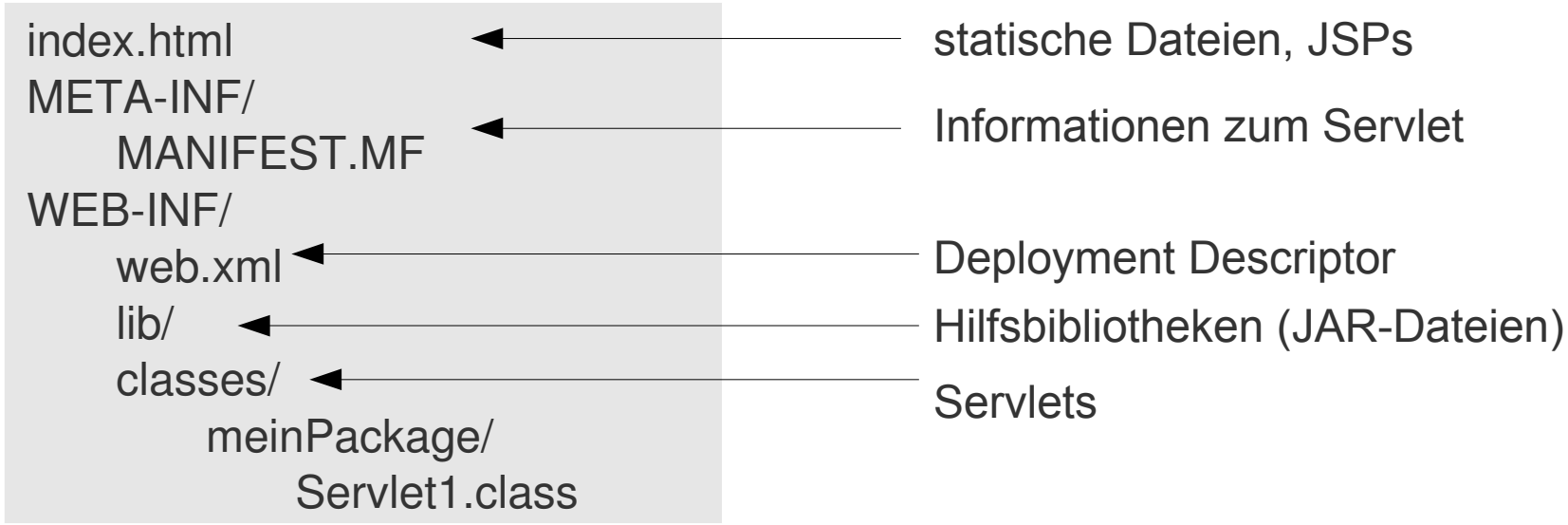
Bereitstellung (Deployment)

- Gesteuert über Datei web.xml

```
<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC
   "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
   "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>
  <servlet>
    <!-- Servlet-Konfiguration … -->
    <servlet-name>servlet1</servlet-name>
    <servlet-class>meinPackage.Servlet1</servlet-class>
  </servlet>
  <servlet-mapping>
     <servlet-name>servlet1</servlet-name>
     <url-pattern>auth/*.html</url-pattern>
  </servlet-mapping>
  <!-- weitere Servlets/JSP -->
  ...
</web-app>
```

Servlet-Paket

- WAR-Datei: Web Archive (im Prinzip JAR-Datei)
- Struktur

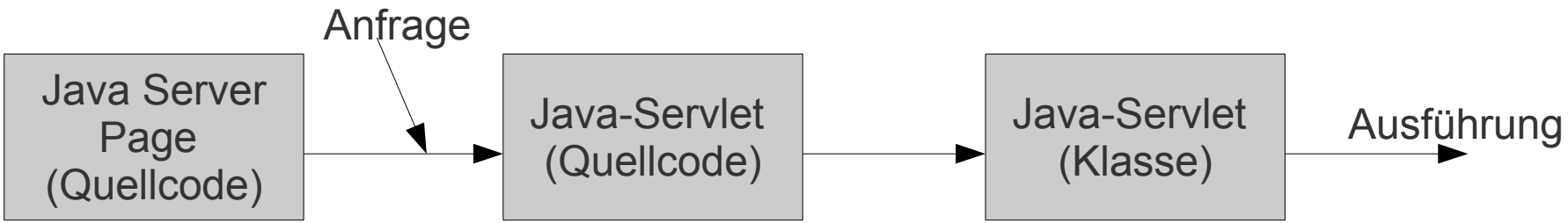


Zustände im Servlet

- Problem
  - HTTP und Servlets sind zustandslos
  - Viele Anwendungen benötigen aber Zustand (z.B. abhängig vom Kunden)
  - Personalisierte Aktienkursliste, Einkaufswagen, Authentifizierung etc.
- Lösung
  - Servlet-Container bietet Sitzungsmanagement als Service an
  - Abbildung auf Cookies oder URL
  - Schnittstelle **javax.servlet.http.HttpSession**

Java Server Pages

- Servlet-Programmierung häufig zu aufwändig
  - z.B. muss das Servlet die ganze HTML-Seite selbst konstruieren, inklusive aller Tags
- Java Server Pages (aktuell 2.0)
  - im Prinzip HTML-Seiten, die statischen (static template code) und dynamischen Inhalt (JSP Elements) mischen (Ähnlichkeit mit PHP, hier eben unter Verwendung von Java)
  - **JSP** muss vor dem Einsatz in eine **Servlet**-Klasse kompiliert werden
    - On-Demand-Behandlung: wird erst bei Anfrage kompiliert (falls nicht schon geschehen)



JSP: HTML und Java-Code

- Wie man HTML und Java in einer Datei mischt

```
<html>
  <head>
    <title>JSP-Beispiel</title>
  </head>
  <body>
    Datum und Zeit:<%= new java.util.Date()%><br/>
    <% for (int i=0; i<5; i++) { %>
      Das Quadrat von <%= i %> ist <%= i*i %><br/>
    <% } %>
  </body>
</html>
```

JSP: Direktiven

- Kommunikation mit der JSP Engine
  <%@ directive {attribute="value"}* %>
- JSP 1.0

  <%@ page import="java.util.*" %>
  Importieren von Bibliotheken

  <%@ page language="java" %>
  Einstellungen, die für die Seite gelten.

  <%@ include file="companyBanner.html" %>
  Einbindung statischer Dokumente

  ...

JSP: Scripting

- Deklarationen: **<%! Deklaration %>**
  - Definieren und überschreiben von Methoden, z.B. *jspInit* und *jspDestroy*
    <%! public void jspInit() { ... } %>
    <%! public void jspDestroy() { ... } %>
- Scriptlets: **<% Java-Codefragment %>**
  - Wird in den Rumpf der service-Methode eingefügt
    <% java.util.Date date = new java.util.Date();
    out.println(date); %>
- Ausdrücke: **<%= Ausdruck %>**
  - Ausdruck auswerten und Ergebnis in String umwandeln
  - String in Seite einfügen: **<%= sDate %>**

JSP: Implizite Objekte

- Innerhalb von Scriptlets und Ausdrücken kann auf diese Objekte unmittelbar zugegriffen werden (sind also vordefiniert)
  - Beispiel: <% out.println(request.getRemoteHost()); %>
-

Name	**Typ**
request	HttpServletRequest
response	HttpServletResponse
session	HttpSession
out	JspWriter
application	ServletContext
config	ServletConfig
page	equivalent zu this
exception	java.lang.Throwable

JSP: Sitzungsverwaltung

- Sitzungsverwaltung über *session*-Objekt
  - Zugang zu Formulardaten über *getParameter*
  - Sitzungsdaten als Name-Wert-Paare (*setAttribute*)

```
<HTML><BODY>
  <FORM METHOD=POST ACTION="Save.jsp">
        Ihr Name?
    <INPUT TYPE="TEXT" NAME="username" SIZE="20">
    <INPUT TYPE="TEXT" NAME="email" SIZE="20">
    <INPUT TYPE="SUBMIT">
  </FORM>
</BODY></HTML>
```

Form.html

```
<%
  String name = request.getParameter("username");
  session.setAttribute("username", name);
  String email = request.getParameter("email");
  session.setAttribute("email", email);
%>
```

Save.jsp

Zusammenfassung

- Begriff „WWW“ wird heutzutage synonym für „Internet“ gebraucht
  - ist aber nur ein Dienst, der auf der durch das Internet gebotenen Grundlage erbracht wird
- WWW: eine Fülle von Informationsangeboten
  - Übliche Benutzeraktionen
    - Seiten konsumieren (lesen, drucken, ...)
    - Verweise anklicken
    - Formulare abschicken
  - Steigende Interaktivität: →Web 2.0
- Dynamische Seiten
  - serverseitig: PHP, Servlets
  - klientenseitig: Java, JavaScript, Flash