S1 Einführung in Web-bezogene Sprachen

Wofür benötigt man Web-bezogene Sprachen?

Gestaltung von Web-Dokumenten:

Beschreibung der Struktur, der Hypertext-Verweise, der Präsentation durch Annotationen (mark-up)

Sprachen: HTML, XML

Programmierung von Web-Diensten:

Erzeugung individueller Web-Seiten, Zugriff auf Dateien und Datenbanken, interaktive Elemente, wie Formulare, auf Web-Seiten

Script-Sprachen: spezielle Programmiersprachen für solche Zwecke
PHP, JavaScript, Perl, VBScript, ASP, SQL

Mit passenden Hilfsmitteln werden auch allgemeine Programmiersprachen eingesetzt, wie Java, C++

beide Zwecke hängen zusammen - Sprachen werden integriert:

Web-Seiten (in HTML) enthalten Programme (in PHP),
Programme (in PHP) erzeugen Web-Seiten (in HTML)

In dieser Vorlesung wird in die Benutzung von **HTML** und **PHP** eingeführt.
JavaScript und XML werden kurz gezeigt.

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 201

Ziele:

Die Zwecke Web-bezogener Sprachen kennenlernen

in der Vorlesung:

- 2 sehr verschiedene Zwecke - 2 verschiedene Gruppen von Sprachen,
- Strukturbeschreibung mit HTML,
- Programmierung mit Script-Sprachen,
- Beispiele dafür auf den nächsten Folien.

Beispiel für eine dynamische, interaktive Web-Seite Telefonverzeichnis

Anklicken der URL
fordert ein Formular an

Eintragen einer Anfrage
fordert eine Suche im Telefonbuch an

Ergebnis der Suche
wird angezeigt

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 202

Ziele:

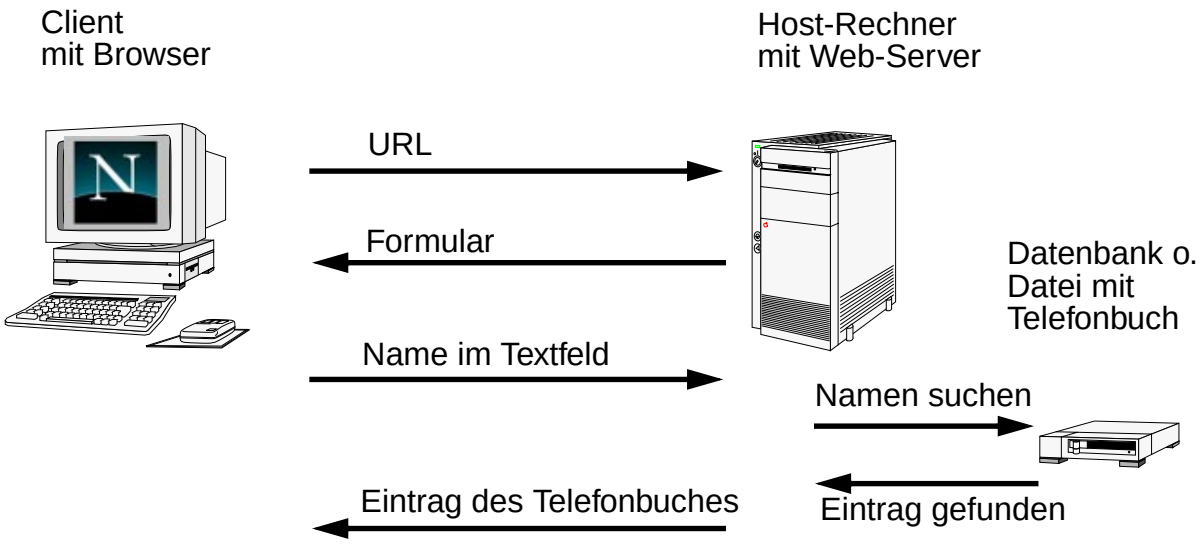
Interaktion des Beispiels verstehen

in der Vorlesung:

Das Beispiel wird erklärt:

- Die 3 Schritte werden erklärt.
- Mit den folgenden Folien wird die Realisierung erklärt.
- Hier kann man das Beispiel benutzen: Telefonbuch

Interaktion zwischen Client und Server



© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 203

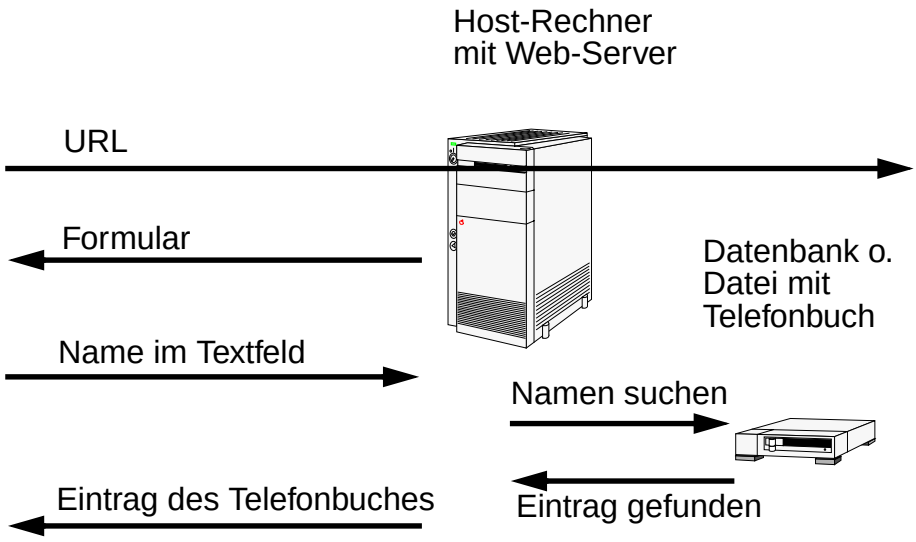
Ziele:

Interaktion zwischen Browser und Web-Server verstehen

in der Vorlesung:

Die Schritte werden erklärt.

Interaktion mit PHP-Programm im HTML-Text



HTML-Text mit PHP-Programm:

```
Überschrift
falls neue Anfrage
    sende HTML-Formular
sonst
    öffne Telefonbuch
    wiederhole:
        suche Namen
        schicke Eintrag
Fusstext
```

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 204

Ziele:

Interaktion in der Programmstruktur wiedererkennen

in der Vorlesung:

Die Struktur des HTML-PHP-Textes wird erklärt:

Struktur des PHP-Programms im HTML-Text

```
<html>
<head>
<title>Uni Telefonbuch Paderborn </title>
</head>
<body>
<h3>Das Uni-Telefonbuch</h3>
<?php // Typischer Aufbau einer PHP-Seite:
if (!isset($_REQUEST['wunsch'])) {
    echo <<<FORMULARANZEIGE
        <h4>Wen suchen Sie?</h4>
        <form action="http://ag-kastens/../unitel.php"
              method="POST">
        <p><input type=text name=wunsch>
        <p><input type=submit value=Abschick>
        </form>
    FORMULARANZEIGE;
}
else {
    $wunsch = $_REQUEST['wunsch'];
    echo "<h4>Ihre Suchergebnisse</h4><p>\n<pre>\n";
    $fp = fopen("telefonbuch.txt" ,"r");
    while (!feof($fp)) {
        $line = fgets($fp, 64);
        if (preg_match("/$wunsch/i", $line)) {
            echo "\t$line";
    }   }
    fclose($fp);
}
?>
</pre>
</body></html>
```

HTML-Text mit PHP-Programm:

- Überschrift
- falls neue Anfrage
  - sende HTML-Formular
- sonst
  - öffne Telefonbuch
  - wiederhole:
    - suche Namen
    - schicke Eintrag
- Fusstext

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 205

Ziele:

Die Struktur im echten Programm wiedererkennen

in der Vorlesung:

Die 4 Textstücke werden erklärt.

Ein erster Eindruck von HTML

```
<html>
<head>
<title>Uni Telefonbuch Paderborn </title>
</head>
<body>
<h3>Das Uni-Telefonbuch</h3>
```

Das Dokument und seine **Textelemente werden annotiert** mit sogenannten **Tags**, z. B.

`<title> ... </title>`
`<h3> ... </h3>`

Die Tags haben Namen und **kennzeichnen Anfang und Ende** einer Struktur.

Strukturen können **geschachtelt** sein.

So wird die **Struktur** des Dokumentes und seine **Darstellung** beschrieben.

```
</pre>
</body></html>
```

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 206

Ziele:

Die Schreibweise von HTML kennenlernen

in der Vorlesung:

Am Beispiel werden erklärt:

- Tags in der Rolle von Klammern,
- Schachtelung.

Ein erster Eindruck von PHP

```
if (!isset($_REQUEST['wunsch'])) {
    echo <<<FORMULARANZEIGE
        <h4>Wen suchen Sie?</h4>
        <form action="http://ag-kastens/../unitel.php"
              method="POST">
        <p><input type=text name=wunsch>
        <p><input type=submit value=Abschick>
        </form>
    FORMULARANZEIGE;
}
else {
    $wunsch = $_REQUEST['wunsch'];
    echo "<h4>Ihre Suchergebnisse</h4><p>\n<pre>\n";
    $fp = fopen("telefonbuch.txt" ,"r");
    while (!feof($fp)) {
        $line = fgets($fp, 64);
        if (preg_match("/$wunsch/i", $line)) {
            echo "\t$line";
    }   }
    fclose($fp);
}
```

PHP ist eine Programmiersprache mit

- **Ablaufstrukturen** wie bedingten Anweisungen und Schleifen:

  ```
  if (..){..} else {..}
  ```

  ```
  while (...) {...}
  ```

- **Variablen, Zuweisungen** und **Funktionsaufrufen**:

  ```
  $line = fgets ($fp, 64);
  ```

Das Programm wird vom Web-Server ausgeführt. Die Ausgabe wird auf dem Browser des Client angezeigt.

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 207

Ziele:

Programmkonstrukte im Beispiel wiedererkennen

in der Vorlesung:

Die Programmkonstrukte werden am Beispiel informell erklärt.

Integration von PHP-Programm und HTML-Text

```
<html>
<head>
<title>Uni Telefonbuch Paderborn </title>
</head>
<body>
<h3>Das Uni-Telefonbuch</h3>
<?php // Typischer Aufbau einer PHP-Seite:
if (!isset($_REQUEST['wunsch'])) {
    echo <<<FORMULARANZEIGE
        <h4>Wen suchen Sie?</h4>
        <form action="http://ag-kastens/../unitel.php"
              method="POST">
        <p><input type=text name=wunsch>
        <p><input type=submit value=Abschick>
        </form>
    FORMULARANZEIGE;
}
else {
    $wunsch = $_REQUEST['wunsch'];
    echo "<h4>Ihre Suchergebnisse</h4><p>\n<pre>\n";
    $fp = fopen("telefonbuch.txt" ,"r");
    while (!feof($fp)) {
        $line = fgets($fp, 64);
        if (preg_match("/$wunsch/i", $line)) {
            echo "\t$line";
    }   }
    fclose($fp);
}
?>
</pre>
</body></html>
```

PHP-Programm in HTML-Dokument eingebettet, geklammert durch

```
<?php
  ...
?>
```

HTML-Text wird vom PHP-Programm ausgegeben, durch echo-Anweisung:

```
echo "<h4>...</h4>";
```

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 208

Ziele:

Integration verstehen

in der Vorlesung:

Das Zusammenwirken wird an diesem Beispiel erklärt.

Verständnisfragen:

Beschreiben Sie das Zusammenwirken!

E1. Einführung zu Eigenschaften von Sprachen

Sprachen in der Informatik werden

- für bestimmte **Zwecke** geschaffen
  hier: **Auszeichnungssprachen** (HTML) und
  Skriptsprachen (PHP, Javascript)
  weitere Aufgabengebiete für Sprachen in dieser Einführung;
- je nach Zweck und **Niveau** mit einfachen oder komplexen, wenigen oder zahlreichen **Sprachkonstrukten** ausgestattet;
- durch **Regeln formal oder informell definiert**; sie legen fest:
  Notation der Symbole (Lexeme),
  Struktur der Sätze (Syntax),
  Bedeutung der Konstrukte (Semantik);
- durch Software-Werkzeuge **übersetzt** oder **interpretiert**

Alle diese Aspekte beeinflussen die **Eigenschaften der Sprachen**.

Die **Verbreitung der Sprachen** wird auch beeinflusst durch

- Erlernbarkeit und Handhabbarkeit,
- Verfügbarkeit von Werkzeugen,
- Marktmechanismen

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 209

Ziele:

Allgemeine Aspekte zu Sprachen

in der Vorlesung:

Die einzelnen Aspekte werden in dieser Einleitung weiter ausgeführt und im Vorlesungsstrang E vertieft.

4 Ebenen der Spracheigenschaften

Ein **Satz einer textuellen* Sprache** ist eine **Folge von Zeichen** eines zu Grunde liegenden **Alphabetes**

Beispiel: ein PHP-Programm ist ein Satz der Sprache PHP;
hier ein Ausschnitt daraus:

```
$line = fgets ($fp, 64);
```

Die **Struktur eines Satzes** wird in 2 Ebenen definiert:

1. **Notation von Grundsymbolen (Lexemen, token)**
2. **Syntaktische Struktur**

Die **Bedeutung eines Satzes** wird in 2 weiteren Ebenen an Hand der Struktur für jedes Sprachkonstrukt definiert:

3. **statische Semantik**
   Eigenschaften, die vor der Ausführung bestimmbar sind.
4. **dynamische Semantik**
   Eigenschaften, die erst während der Ausführung bestimmbar sind.

Auf jeder der 4 Ebenen gibt es auch Regeln, die korrekte Sätze erfüllen müssen.

*) Es gibt auch **visuelle Sprachen**. Ihre Sätze werden aus graphischen Symbolen zusammengesetzt.

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 210

Ziele:

Übersicht über die 4 Ebenen

in der Vorlesung:

Die 4 Ebenen werden an dem Beispiel kurz erklärt.
Visuelle Sprachen werden hier nicht weiter vertieft.

Ebene 1: Notation von Grundsymbolen

Ein **Grundsymbol** wird aus einer **Folge von Zeichen des Alphabetes** gebildet. Die Regeln zur Notation von Grundsymbolen werden z. B. durch **reguläre Ausdrücke formal definiert** (siehe E2).

```
$line = fgets ($fp, 64);
```

Typische Grundsymbole in Programmiersprachen

4 Symbolklassen:	Beispiele aus PHP
Bezeichner (identifier) Namen für Variable, Funktionen, ...	`$line` `fgets`
Literale (literals) Zahlwerte, Zeichenreihen	`64` `"telefonbuch.txt"`
Wortsymbole (keywords) kennzeichnen Sprachkonstrukte	`while` `if`
Spezialzeichen Operatoren, Separatoren	`<=` `=` `;` `{` `}`

Zwischenräume, **Tabulatoren**, **Zeilenwechsel** und **Kommentare** zwischen den Grundsymbolen dienen der Lesbarkeit und sind sonst bedeutungslos

Kommentar	`/* Kommentar in C-Notation */`

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 211

Ziele:

Grundsymbole verstehen

in der Vorlesung:

Es wird an Beispielen erklärt:

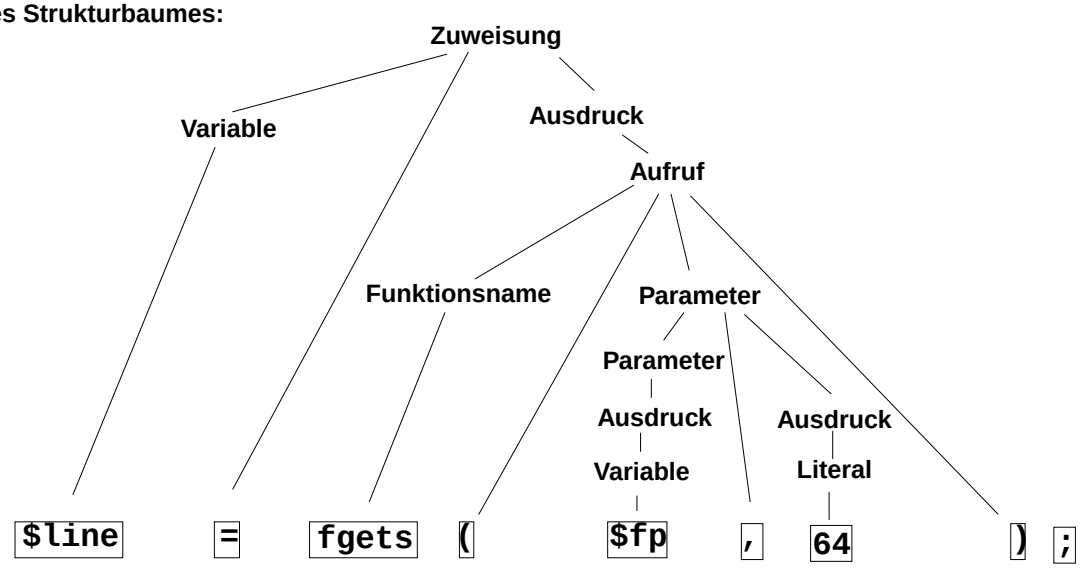
- Eingabe als Folge von Grundsymbolen,
- Klassen von Grundsymbolen,
- informelle Regeln zur Notation,
- Kommentare,
- Trennung aufeinanderfolgender Grundsymbole.

Ebene 2: Syntaktische Struktur

Ein Satz einer Sprache wird **in seine Sprachkonstrukte** gegliedert. Sie sind meist ineinander **geschachtelt**. Diese **syntaktische Struktur** wird durch einen **Strukturbaum** dargestellt. Die **Grundsymbole sind Blätter** in diesem Baum.

Die Syntax einer Sprache wird durch eine **kontextfreie Grammatik präzise definiert**. Die Grundsymbole sind die Terminalsymbole der Grammatik (siehe E2).

Teil des Strukturbaumes:



© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 212

Ziele:

Syntaktische Struktur am Beispiel vestehen

in der Vorlesung:

Am Beispiel wird erklärt:

- Baum repräsentiert die Struktur,
- innere Knoten repräsentieren Sprachkonstrukte,
- Grundsymbole sind Blattknoten,
- Sprachkonstrukte sind nach Strukturregeln aufgebaut,
- sie sind in der kontextfreien Grammatik definiert.

Ebene 3: Statische Semantik

Eigenschaften von Sprachkonstrukten, die ihre Bedeutung (Semantik) beschreiben, soweit sie **an der Programmstruktur festgestellt** werden können (**statisch**), ohne das Programm auszuführen.

Typische Eigenschaften der statischen Semantik (für **übersetzte** Sprachen):

- **Bindung von Namen**:
  Regeln, die einer **Anwendung** eines Namens seine **Definition** zuordnen.
  z. B. „Zu dem Funktionsnamen in einem Aufruf muss es eine Funktionsdefinition mit gleichem Namen geben."
- **Typregeln**:
  Sprachkonstrukte wie **Ausdrücke** und **Variable** liefern bei ihrer Auswertung einen **Wert eines bestimmten Typs**. Er muss im Kontext zulässig sein und kann die Bedeutung von Operationen näher bestimmen.
  z. B. „Die Operanden des + Operators müssen Zahlwerte sein."
  `5 + "Text"` ist in vielen Sprachen ein Typfehler

In der Sprache **PHP** gehören
die **Bindungsregeln** zur **statischen Semantik**,
die **Typregeln** aber zur **dynamischen Semantik**, da sie erst bei der Ausführung des Programms angewandt werden können.

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 213

Ziele:

Statische Semantik verstehen

in der Vorlesung:

An Beispielen wird erklärt:

- Begriff: statisch,
- Bindung von Namen,
- Typregeln,
- nur für übersetzte Sprachen

Ebene 4: Dynamische Semantik

Eigenschaften von Sprachkonstrukten, die ihre Wirkung beschreiben und **erst bei der Ausführung (dynamisch) bestimmt oder geprüft** werden können.

Typische Regeln der **dynamischen Semantik** beschreiben

- welche **Voraussetzungen für eine korrekte Ausführung** eines Sprachkonstruktes erfüllt sein müssen,
  z. B.
  „Ein numerischer Index einer Array-Indizierung, wie in **`$var[$i]`**,
  darf nicht kleiner als 0 sein.“
- welchen **Effekt die Ausführung** eines Sprachkonstruktes verursacht,
  z. B.
  „Eine Zuweisung der Form *Variable = Ausdruck* wird wie folgt ausgewertet:
  Die Speicherstelle der Variablen auf der linken Seite wird bestimmt.
  Der Ausdruck auf der rechten Seite wird ausgewertet.
  Das Ergebnis ersetzt dann den Wert an der Stelle der Variablen.“

In der Sprache PHP gehören auch die Typregeln zur dynamischen Semantik.

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 214

Ziele:

Wirkung und Prüng bei der Ausführung verstehen

in der Vorlesung:

Die typischen Regeln werden an Beispielen erklärt.

Übersetzung von Sprachen

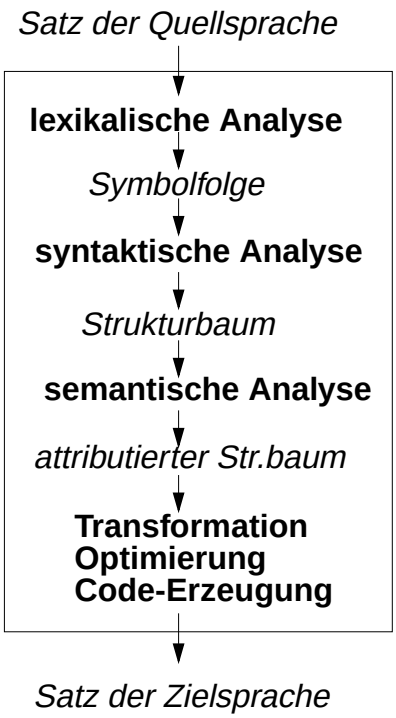
Ein **Übersetzer** transformiert jeden korrekten Satz (Programm) der **Quellsprache** in einen **gleichbedeutenden** Satz der **Zielsprache**.

- Die meisten **Programmiersprachen zur Software-Entwicklung** werden übersetzt, z. B. C, C++, Java, Ada, Modula
- Zielsprache ist dabei meist eine **Maschinensprache** eines realen Prozessors oder einer abstrakten Maschine.
- Übersetzte Sprachen haben eine **stark ausgeprägte statische Semantik**.
- Der Übersetzer prüft die Regeln der statischen Semantik, wie Bindungs- und Typregeln; er findet viele Arten von **Fehlern vor der Ausführung**.

Es gibt auch **Übersetzer für andere Sprachen**:

Textformatierung:	LaTeX	-> PostScript
Spezifikationssprachen:	UML	-> Java

Übersetzer und seine Phasen



© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 215

Ziele:

Die Aufgaben von Übersetzern verstehen

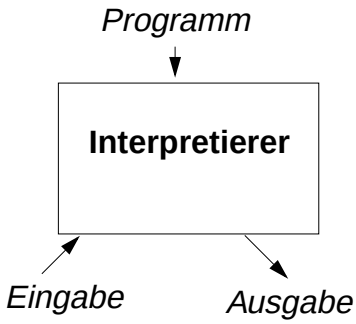
in der Vorlesung:

Es werden erklärt:

- die Begriffe,
- die Übersetzungsschritte,
- der Zusammenhang zu den Ebenen der Spracheigenschaften.

Interpretation von Sprachen

Ein **Interpretierer** liest einen Satz (Programm) einer Sprache und führt ihn aus.



Sprachen, die so **strikt interpretiert** werden:

- haben **einfache Struktur** und **keine statische Semantik**,
- Bindungs- und Typregeln werden erst bei der Ausführung geprüft,
- nicht ausgeführte Programmteile bleiben ungeprüft,

z. B. Lisp, Prolog

Manche Interpretierer erzeugen vor der Ausführung eine **interne Repräsentation des Satzes**;
dann können auch Struktur und Regeln der statischen Semantik vor der Ausführung geprüft werden,
z. B. **Skriptsprachen** PHP, JavaScript, Perl

Interpretierer können **auf jedem Rechner verfügbar** gemacht werden und in andere Software (Browser) **integriert** werden.

Interpretation kann 10-100 mal **zeitaufwändiger** sein als die Ausführung von übersetztem Maschinencode.

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 216

Ziele:

Das Prinzip Interpretation verstehen

in der Vorlesung:

Es wird erklärt

- wie strikt interpretiert wird,
- wie Interpretation mit Übersetzungsphasen kombiniert wird,
- welche Konsequenzen sich für Spracheigenschaften ergeben.

Zwecke von Sprachen: allgemeine Software-Entwicklung

Anforderungen:

- Algorithmen klar und effizient formulieren
- komplexe Datenstrukturen klar und effizient formulieren
- prüfbare Regeln (statische Semantik) einhalten, dadurch Fehler reduzieren
- modulare Gliederung großer Programme mit expliziten Schnittstellen
- Schnittstellenprüfung beim Zusammensetzen von Bibliothekskomponenten

Konsequenzen:

- umfangreiche, komplexe Sprachen: viele Konstrukte, viele Regeln
- relativ hoher Schreibaufwand, durch explizite, redundante Angaben

Nutzen

hoch bei großen Software-Systemen
recht umständlich für kleine, einfache Aufgaben

Sprachstile: imperativ, objektorientiert, (funktional)

Sprachen: Modula-2, Ada, C++, Eiffel, Java, (SML)

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 217

Ziele:

Sprachklasse charakterisieren

in der Vorlesung:

Anforderungen und Konsequenzen begründen

Zwecke von Sprachen: Skriptsprachen

Scripting:
Zusammensetzen von Kommandos zu einem wiederverwendbaren „Drehbuch“

Anforderungen:

- kleine, einfache Aufgaben lösen ohne komplexe Algorithmen und Datenstrukturen
- existierende Funktionen nutzen und verknüpfen
- Verzicht auf Sicherheit durch prüfbare Regeln zugunsten kürzerer Programme
- Textverarbeitung und Ein- und Ausgabe sind wichtig
- gute Verfügbarkeit und Handhabbarkeit

Konsequenzen:

- einfache Sprachen: wenige Konstrukte, wenige Regeln, kurze Programme
- dynamische Typprüfung
- interpretiert, d. h. ohne Übersetzung ausführbar

Herkunft: Kommandosprachen von Betriebssystemen, JCL, Unix Shell

Sprachstile: imperativ, objektorientiert

Sprachen: PHP, Perl, JavaScript, Python

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 218

Ziele:
Sprachklasse charakterisieren

in der Vorlesung:
Anforderungen und Konsequenzen begründen

Zwecke von Sprachen: Auszeichnungssprachen

Auszeichnungssprache: Markup language

Anforderungen:

- Struktur von Dokumenten beschreiben: Überschriften, Absätze, Listen, Tabellen
- hierarchische Gliederung, auch Hypertext-Verweise
- Darstellung der Strukturen beschreiben - aber abtrennbar
- von Menschen schreib- und lesbar
- keine Programmierung - aber integrierbar

Konsequenzen:

- Strukturelemente werden mit lesbaren Markierungen gekennzeichnet (markup)
- geklammerte, geschachtelte Strukturen
- Menge und Bedeutung der Markierungen festlegbar (SGML, XML)
- Darstellung getrennt beschreibbar (HTML + CSS)

Sprachen: SGML, HTML, XML

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 219

Ziele:

Sprachklasse charakterisieren

in der Vorlesung:

Anforderungen und Konsequenzen begründen.

- Das Darstellen einer HTML-Datei in der spezifizierten Form kann man auch als Interpretation auffassen.
- In HTML eingebettete Programme werden vom Interpretierer deren Sprache ausgeführt.

Sprachen für weitere Zwecke

Zugriff auf Datenbanken Selektionen aus Relationen, Verknüpfungen übersetzt in Datenbankzugriffe	SQL
Spezifikation von Hardware-Bausteinen	VHDL
Spezifikation von Software-Komponenten	UML
allgemeine Spezifikation	SETL, Z
Spezifikation von Grammatiken	EBNF
Textsatz	TeX, LaTeX, PostScript

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 220

Ziele:

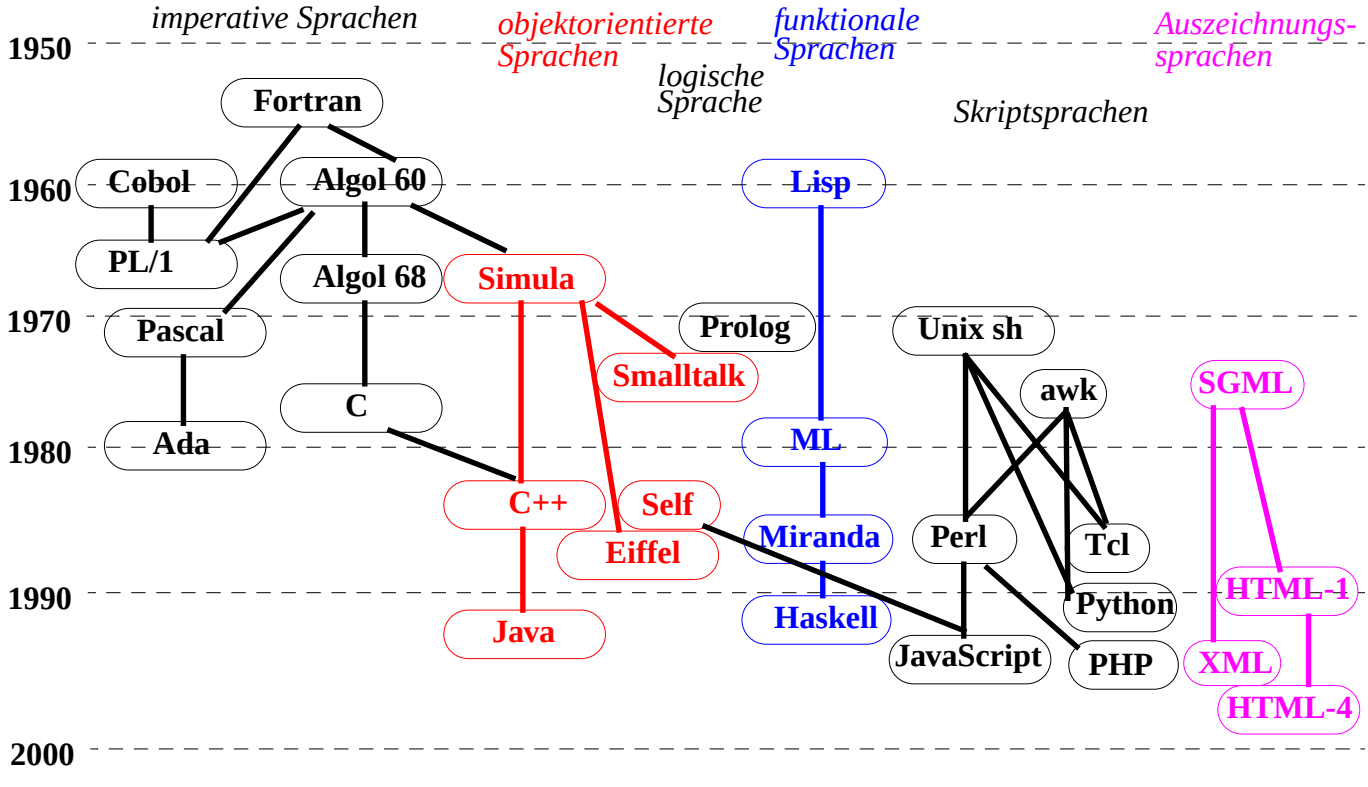
Eindruck von Sprachen für andere Zwecke

in der Vorlesung:

Bedeutung allgemeiner und spezieller Spezifikationssprachen hervorheben.

E EWS-2.21

Entstehungszeit und Verwandtschaft wichtiger Sprachen



nach [D. A. Watt: Programmiersprachen, Hanser, 1996; Seite 5] und [Computer Language History http://www.levenez.com/lang]

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 221

Ziele:

Sprachen zeitlich einordnen und klassifizieren

in der Vorlesung:

Kommentare zur zeitlichen Entwicklung.

Verwandschaft zwischen Sprachen:

- Notation: C, C++, Java;
- gleiche zentrale Konzepte, wie Datentypen, Objektorientierung;
- Teilsprache: Algol 60 ist Teilsprache von Simula, C von C++;
- gleiches Anwendungsgebiet: z. B. Fortran und Algol 60 für numerische Berechnungen in wissenschaftlich-technischen Anwendungen

nachlesen:

Text dazu im Buch von D. A. Watt

Übungsaufgaben:

Verständnisfragen:

In welcher Weise können Programmiersprachen miteinander verwandt sein?

S2 HTML

HTML steht für **Hypertext Markup Language**: eine **Auszeichnungssprache**, in der man **Dokumente** durch **Hyperlinks (Verweise)** miteinander **verknüpfen** kann.

Auszeichnungen (Markups) sind typografische Anweisungen für

- die Strukturierung von Texten in Abschnitte, Absätze, Listen, Tabellen, usw.
- die Gestaltung von Zeichen: Schrifttyp, Schriftgrad, Schriftstil, Schriftfarbe
- die Bildmontage: Platzierung und Größe
- das Layout des Dokumentes: Tabellen und Frames
- die Verknüpfung von Dokumenten: Anker, Links, sensitive Bildbereiche
- die Dialoggestaltung: Formulare, Schaltelemente

HTML-Markups oder **Tags** haben die Form **`<table> ... </table>`**
Sie treten meist als **Klammern von Anfangs- und Ende-Tag** auf.

Tags können zusätzlich **Attribute** haben,

```
<table border="3" bgcolor="#2332FF">
```

HTML wurde ursprünglich aus der **Meta-Sprache SGML** abgeleitet.

HTML wird nun auch aus der **Meta-Sprache XML** abgeleitet: XHTML.

© 2006 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 222

Ziele:

Charakterisierung von HTML

in der Vorlesung:

Die genannten Eigenschaften werden kurz erklärt.

S2.1 Auszeichnung von Strukturen und Texten

```
<!DOCTYPE HTML PUBLIC
 "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
   <head>
      <title>Hallo Welt</title>
   </head>
   <body>
      <h1>Hello World!</h1>
   </body>
</html>
```

Grundstruktur jeder HTML-Datei:

`<!DOCTYPE ...>`:
Kommentar mit optionaler Angabe der HTML Definition

`<html>`:
äußere Klammer

`<head>`:
Angaben über das Dokument

`<body>`:
Inhalt des Dokumentes

© 2006 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 223

Ziele:

Grundstruktur kennenlernen

in der Vorlesung:

- Die Bedeutung der Tags,
- die Schreibweise von Kommentaren,
- der Verweis auf die HTML-Definition

werden erklärt.

Schreibweise von Tags und Attributen

Tags haben die Form `<TagName>` (Anfangs-Tag) `</TagName>` (Ende-Tag)

Zwischen **<** und **`TagName`** dürfen keine Zwischenräume stehen.

Tags, die **Strukturen** kennzeichen, sollen **immer klammernd** geschrieben werden, auch wenn das Ende-Tag optional ist.

Tags können **Attribute** haben.
Sie ordnen dem gekennzeichneten Bereich bestimmte Eigenschaften zu.

Jedes Attribut hat die Form `Name="Wert"` z. B. `border="4"`

Das Anfangs-Tag kann zwischen dem Tag-Namen und dem **>** eine Folge von Attributen enthalten `<table border="4" frame="box">`

Verwenden Sie **nur notwendige Attribute**. Viele Attribute sind veraltet oder werden von verschiedenen Browsern unterschiedlich behandelt.

Detaillierte Angaben zur Darstellung werden besser mit anderen Mitteln (z. B. Style Sheets, siehe Folie 2.32) gemacht als mit Attributen.

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 224

Ziele:

Attribute kennenlernen

in der Vorlesung:

Die Schreibweisen werden erklärt.

Überschriften und Absätze

```
<html><head>
   <title>&Uuml;berschriften</title>
</head>
<body>
   <h1>1. Kapitel</h1>
   <p> In diesem Kapitel beschreiben wir
       die Tags von HTML.
       <br>Jeder Abschnitt ist einer Klasse

       von Tags gewidmet.
   </p>
   <h2>1.1 Struktur-Tags</h2>
   <p>Struktur-Tags gliedern ein
      Dokument hierarchisch in
      Abschnitte und Unterabschnitte.
   </p>
   <h3>Absatz-Tag</h3>
   <h4>Attribute des Absatz-Tag
   </h4>
```

<h1>, **<h2>**, ..., **<h6>**:
Überschriften fallenden Ranges

<p>:
Jeder Absatz wird explizit ausgezeichnet.

**
**:
erzwingt Zeilenwechsel.

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 225

Ziele:

HTML-Tags an Beispielen

in der Vorlesung:

Am Beispiel wird erklärt:

- die Rolle der Überschrift-Tags,
- Zeilenumbruch und Absätze.

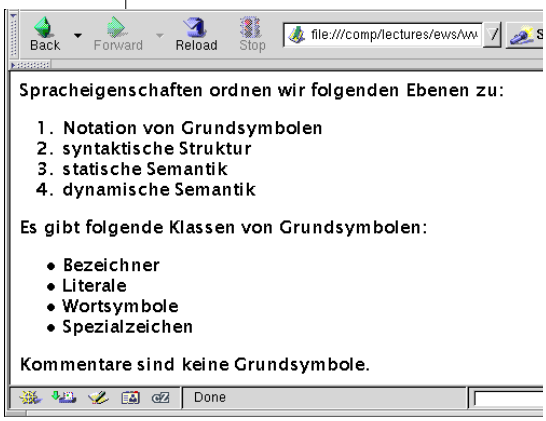
Aufzählungen

```
<html><head>
   <title>Aufz&auml;hlungen</title>
</head><body>
   <p> Spracheigenschaften ordnen wir
       folgenden Ebenen zu:
      <ol>
         <li>Notation von Grundsymbolen</li>
         <li>syntaktische Struktur</li>
         <li>statische Semantik</li>
         <li>dynamische Semantik</li>
      </ol>
      Grundsymbolklassen sind:
      <ul>
         <li>Bezeichner</li>
         <li>Literale</li>
         <li>Wortsymbole</li>
         <li>Spezialzeichen</li>
      </ul>
      Kommentare sind keine
      Grundsymbole.
   </p>
</body></html>
```

<ol>:
nummerierte Aufzählung von Textelementen

<ul>:
Aufzählung von Textelementen mit Markierung („Bullet“)

<li>:
Kennzeichnung der Textelemente



© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 226

Ziele:

HTML-Tags an Beispielen

in der Vorlesung:

Am Beispiel wird erklärt:

- Listen und Listenelemente sauber klammern.
- nummerieren, wenn es auf die Reihenfolge ankommt, oder auf Elemente bezuggenommen werden soll.
- Man kann die Anordnung auch mit Buchstaben kennzeichnen lassen oder die Form der Bullets bestimmen. Wir tun das nicht im HTML-Text (sondern in einem Style Sheet, siehe Abschnitt S2.2).

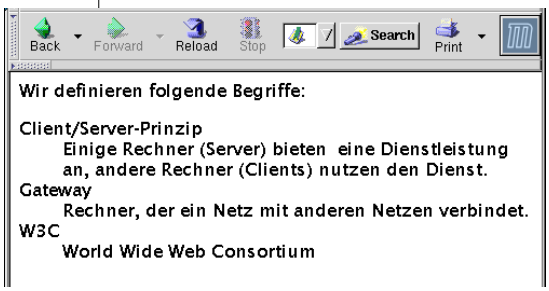
Definitionslisten

```
<html><head>
   <title>Definitionslisten</title>
</head><body>
   <p> Wir definieren folgende Begriffe:
      <dl>
         <dt>Client/Server-Prinzip</dt>
         <dd>Einige Rechner (Server) bieten
             eine Dienstleistung an, andere
             Rechner (Clients) nutzen den
             Dienst.</dd>
         <dt>Gateway</dt>
         <dd>Rechner, der ein Netz mit
             anderen Netzen verbindet.</dd>
         <dt>W3C</dt>
         <dd>World Wide Web Consortium</dd>
      </dl>
   </p>
</body></html>
```

<dl>:
eine Liste mit Paaren von Begriffen und Erklärungen dazu

<dt>:
markiert den Begriff

<dd>:
markiert die Erklärung



© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 227

Ziele:

HTML-Tags an Beispielen

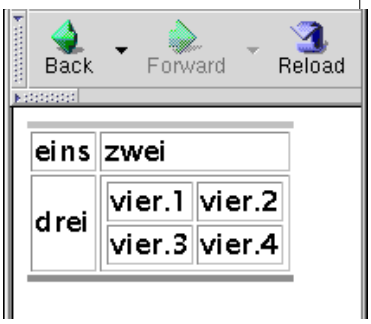
in der Vorlesung:

Am Beispiel wird erklärt:

- Liste von Paaren und
- ihre Formatierung.

Tabellen

```
<html><head>
   <title>Tabellen</title>
</head><body>
   <table border="4" frame="hsides">
      <tr><td>eins</td> <td>zwei</td>
      </tr>
      <tr><td>drei</td>
          <td><table border="2" frame="void">
                 <tr><td>vier.1</td>
                     <td>vier.2</td>
                 </tr>
                 <tr><td>vier.3</td>
                     <td>vier.4</td>
                 </tr>
              </table>
          </td>
      </tr>
   </table>
</body></html>
```



`<table>`:
eine Tabelle

Attribut **`border`**:
Breites des Randes um die Zellen

Attribut **`frame`**:
äußerer Rahmen;
Werte:
`void`, **`box`**,
`above`, **`below`**,
`hsides`, **`vsides`**,
`lhs`, **`rhs`**

`<tr>`:
Tabellenzeile

`<td>`:
Tabellenzelle,
kann selbst eine Tabelle enthalten

© 2006 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 228

Ziele:

HTML-Tags an Beispielen

in der Vorlesung:

Am Beispiel wird erklärt:

- Zeilen- und Spaltenstruktur,
- geschachtelte Tabellen,
- Rand- und Rahmenattribute.

Anker

```
<html><head>
   <title>Anker</title>
</head><body>
   <a name="SeitenAnfang">
   <p> Es folgt eine Liste wichtiger Links:
   <ul>
      <li> <a href="http://ag-kastens.upb.de/lehre/material/ews">
          Skript zu EWS</a>
      </li>
      <li> <a href="http://ag-kastens.upb.de/lehre/material/gdp">
          Skript zu GdP</a>
      </li>
   </ul>
      Hier kommen Sie wieder zum
      <a href="#SeitenAnfang">
      Anfang der Seite</a>
   </p>
</body></html>
```

<a name="XYZ">:
platziert den Anker **XYZ**; wird adressiert mit **#XYZ**

Text :
setzt einen Link zu **Adr** mit dem angegebenen **Text**

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 229

Ziele:

Anker einsetzen können

in der Vorlesung:

Am Beispiel wird erklärt:

- 2 Arten von Ankern:
- Link, der in dieses oder ein anderes Dokument verweist (href-Attribut);
- Position einer Stelle, auf die man verweisen kann (name-Attribut).

Repräsentation spezieller Zeichen

Zeichen, die in der Notation von **HTML eine spezielle Bedeutung** haben, müssen **umschrieben** werden, wenn sie in normalem Text nicht als HTML-Zeichen gemeint sind

<	<	`<p>Das Tag <p> kennzeichnet einen Absatz</p>`
>	>	
&	&	für jedes Zeichen gibt es auch einen numerischen Code: &
"	"	
		non-breaking space, Leerzeichen, das nicht durch Zeilenumbruch ersetzt werden darf

Allgemeine Form: `&Zeichenname;` oder `&#Zahl;` wobei die Zahl das Zeichen codiert

HTML-Beschreibungen enthalten Listen mit solchen Zeichendefinitionen (*entities*), z. B.

ä	ä	
Ä	Ä	
ß	ß	`<p>Ma&szlig;kr&uuml;ge</p>`
ñ	ñ	`<p>El ni&ntilde;o</p>`
©	©	`<p>&copy;2006 bei Dr. M. Thies</p>`

© 2006 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 230

Ziele:

Schreibweise von Sonderzeichen lernen

in der Vorlesung:

An den Beispielen wird erklärt:

- Notwendigkeit der Notation,
- allgemeine Schreibweise,
- Hinweis auf Referenzlisten

Zweck-bezogene Hervorhebungen von Text

Textstücke können gekennzeichnet werden, damit sie durch **besonderen Schriftsatz hervorgehoben** werden. Dafür sollte man den **Zweck der Hervorhebung** kennzeichnen und **nicht die Darstellung** festlegen, z. B.

```
<p>Vor Verlassen des Raumes das <em>Licht ausschalten</em>,
aber <strong>nicht den Notausschalter</strong> benutzen!
```

Hier ist der Zweck, Betonung und starke Betonung, angegeben. Die Darstellung im Schriftsatz kann man unabhängig davon festlegen.
Das ist in folgendem Beispiel nicht mehr möglich und daher **nicht empfohlen:**

```
<p>Vor Verlassen des Raumes das <b>Licht ausschalten</b>,
aber <font color="red">nicht den Notausschalter</font> benutzen!
```

Weitere Beispiele für **Zweck-bezogene Hervorhebungen**:

<cite>	Zitat
<samp>	Beispiel
<dfn>	Definition
<code>	Quell-Code
<kbd>	Tastatureingabe
<var>	Variablenname

Zweck wird durch das class-Attribut individuell bestimmt

****
für elementaren Fließtext

** **
für Textblöcke mit Unterstrukturen

© 2006 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 231

Ziele:

Unterscheidung: Formatierung und ihr Zweck

in der Vorlesung:

An den Beispielen wird erklärt:

- mehr Möglichkeiten, wenn *nicht* explizit formatiert wird.
- Bedeutung der Tags,
- Hinweis auf Style Sheets.

S2.2 Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) ist eine Sprache zur **Definition von Formateigenschaften für HTML-Auszeichnungen**.

CSS wird verwendet, um **Formatierungsangaben** von den Auszeichnungen der Struktur und der Zweck-bezogenen Hervorhebungen zu **trennen**:

HTML-Datei:
Struktur und Rolle der Elemente im Dokument

```
<h1>Unsere Uni</h1>
<p>Eine der 5 Fakult&auml;ten ist
die <span class="eim">
Fakult&auml;t EIM</span>.</p>
...
```

benutzt zur Formatierung →

CSS-Datei:
Formatierung der markierten Elemente

```
h1 {font-size:24;}

.eim {color:blue}
...
```

Ergebnisse:

- **konsistente Formatierung im ganzen Dokument**
- konsistente Formatierung **in vielen Dokumenten** (Corporate Identity)
- **einfache Änderung** der Formatierung in der CSS-Datei - HTML-Dateien bleiben unverändert

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 232

Ziele:

Motivation zur Verwendung von CSS

in der Vorlesung:

- Trennung zwischen HTML- und CSS-Datei,
- Wiederverwendung von Formatierungen,
- explizit formatierte HTML-Dateien sind schlecht wartbar.

Notationen von Formatangaben in CSS

Im <head>-Abschnitt der HTML-Datei wird ein **Link** auf die CSS-Datei eingetragen mit dem **relativen Dateinamen** oder der vollen URL:

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

CSS-Angaben können auch in die HTML-Datei eingebettet werden - sind dann aber **nicht wiederverwendbar**: `<style type="text/css"> CSS-Angaben </style>`

Formatangaben in CSS für bestimmte Arten von HTML-Tags:

```
h1, h2  {text-align:center;color:red;}
em      {font-weight:bold;}
strong  {font-weight:bold;color:red;}
```

allgemeine Form: `HTML-Tags{ Folge von Formatangaben }`

Formatangabe: `Name:Wert;` keine Leerzeichen; nur mehrteilige Werte in "

Beispiele:

```
font-family:Times;
font-style:italic;
text-decoration:underline;
list-style-type:disc;
list-style-type:lower-alpha;
```

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 233

Ziele:

Zuordnung von Formatangaben verstehen

in der Vorlesung:

An den Beispielen wird erklärt:

- Zuordnung der CSS-Datei,
- Schreibweise von Formatangaben und
- Zuordnung zu HTML-Tags,
- Hinweis auf Listen von Namen und Werten für Formateigenschaften, z. B. CSS2 Reference in einer Sammlung von Tutorials.

Freie Klassifikation von HTML-Tags

Bestimmte **Zwecke der Hervorhebung** kann man durch
frei erfundene Namen klassifizieren, z. B.

```
Die Idee, <span class="meinAnteil">der Entwurf</span> und die ..
```

```
<div class="meinAnteil">
<h3>Entwurf</h3>
<p>Das Software-System besteht aus ...</p>
...
</div>
```

In einem CSS Style Sheet kann man solchen Klassen Formatierangaben zuordnen:

```
.meinAnteil {font-weight:bold;color:blue;}
```

oder in einer anderen CSS-Datei bescheidener auf die Hervorhebung verzichten:

```
.meinAnteil {}
```

Auch den Struktur-Tags mit festgelegter Bedeutung können über das Klassenattribut Formateigenschaften zugeordnet werden:

```
<p class="meinAnteil">Der Entwurf des Systems ist ...</p>
```

Allerdings wäre eine Klammerung mit **** konsequenter.

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 234

Ziele:

Umgang mit freien Klassifikationen lernen

in der Vorlesung:

An Beispielen wird erklärt:

- Zweck benennen,
- Klassenattribut benutzen,
- span- und div-Tags einsetzen,
- Formateigenschaften den Klassen zuordnen.

Beispiel: 2 verschiedene Formatierungen

```
<html><head>
    <title>CSS Beispiel</title>
    <link rel="stylesheet" type="text/css"
       href="cssbsp.css">
</head><body>
    <h1>Das Fantasie-System</h1>
    <p> In diesem Kapitel beschreiben wir
    die Aufgabe, den
    <span class="ukas">Entwurf</span>
    und die Implementierung des
    Fantasie-Systems.</p>

    <h2>Aufgabe</h2>
    <p>Zu den <dfn>Aufgaben</dfn>
    geh&ouml;ren blah und blah</p>

    <div class="ukas">
    <h2>Entwurf</h2>
    <p>Der <dfn>Entwurf</dfn> des Systems ist
    gut gelungen blah blah</p>

    <h4>Detail-Entwurf</h4>
    <p>Der Detail-Entwurf ist auch ganz
    toll.</p>
    </div>

    <h2>Implementierung</h2>
</body></html>
```


```
h1, h2, h3 {text-align:left;}
dfn    {font-style:italic;font-weight:bold;}
.ukas {font-weight:bold;color:blue;}
```


```
h1, h2, h3 {text-align:center;}
dfn {font-style:italic;}
.ukas {}
```

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 235

Ziele:

Zuordnung von CSS-Definitionen im Zusammenhang sehen

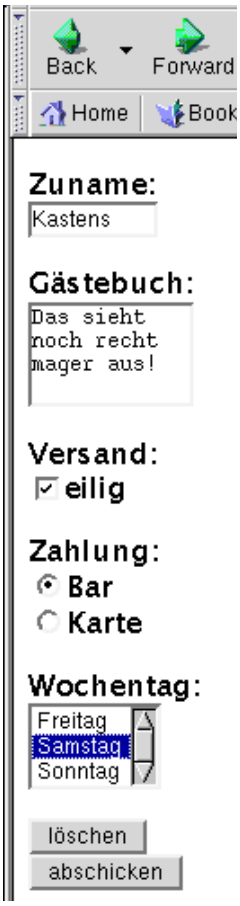
in der Vorlesung:

Der Inhalt der Folie wird an Beispielen erklärt.

S2.3 Formulare mit Eingabeelementen

Graphische Elemente für **interaktive Eingaben**:
z. B. Textfelder, Auswahllisten, Schaltelemente, usw.

- strukturiert und gestaltet mit allgemeinen Dokument-Tags
- zu Formularen zusammengefasst



Eingabedaten werden

- zum **Web-Server gesandt** und dort verarbeitet,
  siehe Beispiel Telefonbuch;
  siehe Abschnitt *W4 Interaktive, dynamische Web-Seiten*
- **im Browser** geprüft und/oder verarbeitet
  siehe Abschnitt *S4 JavaScript*

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 236

Ziele:

Einführung zu Eingabeelementen

in der Vorlesung:

- Interaktion mit Web-Seiten,
- Beispiele werden gezeigt,
- Notwendigkeit der Elemente.

Ein einfaches Formular

```
<form action="http://www.upb.de" method="get">
   <p>Ihr Zuname:
      <input type="text" name="Zuname" size="10">
   </p>
   <input type="submit" value="abschicken">
</form>
```

`<form>`
fasst Eingabeelemente (**`<input>`**) und weitere Elemente zu einem Formular **zusammen**. Alle Eingabedaten eines Formulars werden gemeinsam verarbeitet.

`action`-Attribut
Ziel, an das die Eingabe geschickt wird;
Web-Adresse;
Web-Seite enthält ggf. ein Programm, das die Eingabe verarbeitet

`method`-Attribut
charakterisiert die **Art der Übertragung** zum Ziel;
hier **`get`**: als Parameter der URL

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 237

Ziele:

form-Klammer verstehen

in der Vorlesung:

Am Beispiel wird erklärt:

- Zusammenfassung,
- Attribute,
- Name-Wert-Paar als Parameter der Ziel-URL.
- Hier kann man das Beispiel benutzen: Formular-Beispiel

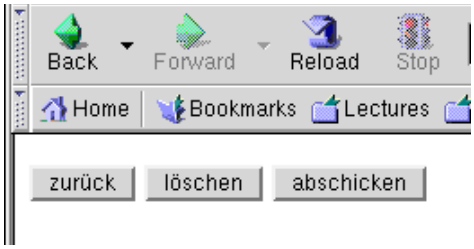
Schaltflächen (Knöpfe)

```
<input type="button" value="zur&uuml;ck"
   onclick="javascript:history.back()">

<input type="reset" value="l&ouml;schen">

<input type="submit" value="abschicken">
```

<input type="button"
charakterisiert eine **allgemeine Schaltfläche**

value-Attribut
Beschriftung der Schaltfläche

onclick-Attribut
Aktion, die beim Betätigen ausgeführt wird;
hier Aufruf einer Funktion in JavaScript



<input type="reset"
spezielle Schaltfläche: alle **Eingabeelemente** in den **Initialzustand zurücksetzen**

<input type="submit"
spezielle Schaltfläche: **Eingabe abschicken**;
eine davon in jedem Formular!

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 238

Ziele:

Knöpfe einsetzen können

in der Vorlesung:

Am Beispiel wird erklärt:

- spezielle Bedeutung von reset und submit,
- allgemeiner Knopf löst programmierte Aktion aus.

Einzeiliges Textfeld

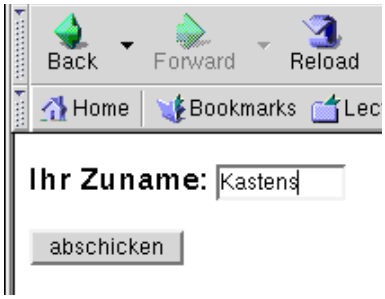
```
<input type="text" name="Zuname" size="10" maxlength="60">
```

<input type="text"
charakterisiert **einzeiliges Textfeld**

name-Attribut
identifiziert das Eingabeelement
im verarbeitenden Programm

size-Attribut
Größe des angezeigten Textfeldes in Zeichen

maxlength-Attribut
Anzahl der Zeichen, die **maximal eingegeben** werden können
Scrolling, wenn **`maxlength > size`**



© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 239

Ziele:

Textzeile einsetzen können

in der Vorlesung:

Am Beispiel wird erklärt:

- Attribute,
- Ergebnis als Parameterpaar

Mehrzeiliges Textfeld

```
<p>Eintrag ins G&auml;stebuch:<br>
   <textarea name="Eintrag" rows="5" cols="20">Hier schreiben!
   </textarea>
</p>
```

<textarea>
charakterisiert **mehrzeiliges Textfeld;**
zwischen den Tags kann
Initialisierung stehen

name-Attribut
identifiziert das Eingabeelement
im verarbeitenden Programm

rows-Attribut
Anzahl der angezeigten **Zeilen**

cols-Attribut
Anzahl der angezeigten **Spalten**

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 240

Ziele:

Textblöcke einsetzen können

in der Vorlesung:

Am Beispiel wird erklärt:

- Größe und
- Initialisierung.

Auswahlliste

```
<p>Wochentag:<br>
   <select name="tag" size="3">
      <option value="fr">Freitag
      <option value="sa">Samstag
      <option value="so">Sonntag
      <option value="mo">Montag
   </select>
</p>
```

`<select>`
Klammert eine Liste von Einträgen, aus der einer ausgewählt werden kann

`name`-Attribut
identifiziert das Eingabeelement im verarbeitenden Programm

`size`-Attribut
Anzahl der angezeigten **Einträge**

`value`-Attribut
für das gewählte Element wird das Paar (**name, value**) **übertragen**

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 241

Ziele:

Auswahlliste einsetzen können

in der Vorlesung:

Am Beispiel wird erklärt:

- Struktur,
- Attribute,
- weiteres Attribut `multiple`: Selektion mehrerer Einträge,
- Ergebnis als Parameterpaar

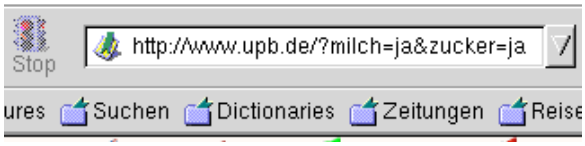
Checkbox

```
<p>Nehmen Sie zum Kaffee<br>
   <input type="checkbox" name="milch" value="ja">Milch<br>
   <input type="checkbox" name="zucker" value="ja">Zucker<br>
</p>
```

<input type="checkbox"
ein Element, dass man **auswählen** kann

name-Attribut
identifiziert das Eingabeelement
im verarbeitenden Programm

value-Attribut
wenn das Element gewählt ist, wird das Paar (**name, value**) **übertragen**



© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 242

Ziele:

Checkbox einsetzen können

in der Vorlesung:

Am Beispiel wird erklärt:

- Attribute,
- weiteres Attribut `checked`: initiale Markierung,
- Ergebnis als Parameterpaar

Auswahlknöpfe

```
<p>Zahlungsart:<br>
   <input type="radio" name="pay" value="cash">Bar<br>
   <input type="radio" name="pay" value="card">Kreditkarte<br>
   <input type="radio" name="pay" value="order">&Uuml;berweisung
</p>
```

`<input type="radio"`
eine Gruppe von Knöpfen, von denen man genau einen **auswählen** kann (radio buttons)

`name`-Attribut
identifiziert das Eingabeelement im verarbeitenden Programm; alle Elemente der Gruppe haben den gleichen Namen

`value`-Attribut
wenn der Knopf gewählt ist, wird das Paar (**name, value**) **übertragen**

© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 243

Ziele:

Auswahlknöpfe einsetzen können

in der Vorlesung:

Am Beispiel wird erklärt:

- Attribute,
- Ergebnis als Parameterpaar

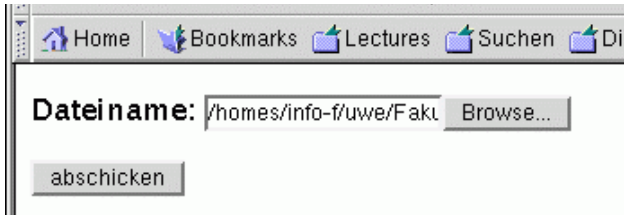
Datei zum Web-Server senden

```
<form action="http://www.upb.de" method="get"
      enctype="multipart/form-data">
  <p>Dateiname:
    <input type="file" name="Datei">
  </p>
```

<input type="file"
Datei versenden;
Name direkt angeben oder im Dateiselektor auswählen

name-Attribut
identifiziert das Eingabeelement

enctype="multipart/form-data"
weiteres Attribut im **form**-Tag nötig



© 2005 bei Prof. Dr. Uwe Kastens

Vorlesung Einführung in Web-bezogene Sprachen WS 2006 / Folie 244

Ziele:

Dateiauswahl einsetzen können

in der Vorlesung:

Am Beispiel wird erklärt:

- Attribute,
- weiteres Attribut `accept="MIME-Typ"`: schränkt Dateityp ein, z. B. `accept="image/jpeg"`
- Ergebnis als Parameterpaar