

## 8. Skriptsprachen

- 8.1 Clientseitige Web-Skripte: JavaScript
- 8.2 Document Object Model (DOM)
- 8.3 Serverseitige Web-Skripte: PHP



Weiterführende Literatur:

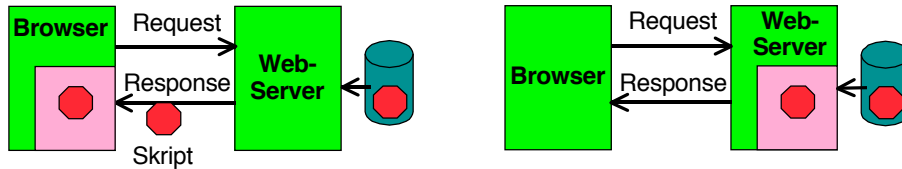
Wolfgang Dehnhardt: JavaScript, VBScript, ASP, Perl, PHP, XML:  
Skriptsprachen für dynamische Webauftritte, Carl Hanser 2001

<http://www.selfphp.info>

## HTTP-Client-Server-Kommunikation

- Grundprinzip von HTTP:
  - Client (*Browser*) schickt Anfrage (*request*) über IP-Verbindung an Server
  - Mögliche "Methoden":
    - » GET: Liefere Inhalt zu URL
    - » HEAD: Wie GET, aber ohne echte Lieferung der Daten (nur „Header“)
    - » POST: Akzeptiere im Rumpf mitgelieferte Daten
  - Diverse "Header Codes" in der Anfrage,  
z.B. Browsertyp, Host, Zeichensatz-Encoding, gewünschte Sprachen, ...
  - Server schickt Antwort (*response*)
    - » Hauptinhalt: HTML-Code
    - » Header-Codes auch in der Antwort
- Funktion des Servers:
  - Dateien bereitstellen
- Funktion des Clients:
  - HTML-Inhalte darstellen

## Serverseitige vs. clientseitige Dynamik



- Clientseitige Dynamik:
  - Browser enthält Ausführungsmaschine für Skripte
  - Skript ist Teil der Antwort vom Server
  - Web-Server muss Skriptsprache nicht kennen
  - Beispiel: JavaScript

- Serverseitige Dynamik:
  - Web-Server enthält Ausführungsmaschine für Skripte
  - Skript wird vor Beantwortung der Anfrage ausgeführt und liefert HTML-Text
  - Browser muss Skriptsprache nicht kennen
  - Beispiel: PHP

## Technologien für serverseitige Dynamik

- Common Gateway Interface (CGI):
  - Ermöglicht Aufruf beliebiger Programme beim Server (z.B. in C)
  - Programm erzeugt (schreibt) HTML-Textdatei (Response)
  - Häufig Skriptsprachen benutzt (z.B. Perl, Python)
  - Manchmal spezielle Bibliotheken für Webseiten verfügbar
- Spezielle Server-Skriptsprachen:
  - Entworfen für Einbettung in HTML
  - Plug-Ins für gängige Web-Server-Software
  - geeignet für "mittelgrosse" dynamische Anwendungen
- Programmiersprachen-Einbettung in Web-Server:
  - z.B. Java *Servlets*, geschrieben in Java
  - Aufgerufen vom Server über standardisiertes API
  - Generierung von Servlets aus Skript-ähnlichen Sprachen z.B. Java Server Pages (JSP)
  - geeignet für "grosse" dynamische Web-Anwendungen (Servlets/JSP: siehe Vorlesung Medientechnik!)

## Beispiel: Server-Skriptsprache PHP

- PHP:
  - Personal Home Page Toolkit
  - PHP Hypertext Preprocessor
- OpenSource Entwicklung:
  - siehe [www.php.net](http://www.php.net)
  - lizenzfrei benutzbar
- Syntax an C angelehnt, aber mehrere Varianten unterstützt
- Einfache Kernsprache, umfangreiche Funktionsbibliothek
  - über 500 Funktionen!
  - etwas unübersichtlich
  - spezialisiert auf Aufgaben der Webseiten-Programmierung

## Voraussetzungen für praktische Experimente

- Auch bei lokalen (Ein-Rechner-)Experimenten
  - Installation eines Web-Servers
    - » OpenSource: *Apache*
    - » Microsoft *Internet Information Server*
  - Aufruf der HTML-Dateien über Web-Server (<http://...>)
- Bereitstellung und Installation der PHP-Software als Plug-In für den verwendeten Web-Server
- In den meisten praktischen Fällen: Installation eines relationalen Datenbanksystems (z.B. MySQL)
- Insider-Kürzel für bestimmte Konfigurationen (Beispiele):
  - LAMP: Linux, Apache, MySQL, PHP
  - WIMP: Windows, Internet Information Server, MySQL, PHP
  - MOXAMP: MacOS X, Apache, MySQL, PHP (hier verwendet)

## Beispiel: "Hello World" in PHP und JavaScript

```
<html>
<head><TITLE>Hello World mit JavaScript</TITLE></head>
<body>
  <h1>
    <script type="text/javascript">
      document.write("Hello World!");
    </script>
  </h1>
</body>
</html>
```

JavaScript

```
<html>
<head><title>Hello World mit PHP</title></head>
<body>
  <h1>
    <?php
      echo "Hello World!";
    ?>
  </h1>
</body>
</html>
```

PHP

## Einbettung von PHP in HTML

- XML-Stil (hier verwendet):
  - Analog zu *Processing Instructions* von XML
  - `<?php PHP-Text ?>`
- SGML-Stil:
  - Kurze und weit verbreitete "Urform"
  - `<? PHP-Text ?>`
- HTML-Stil:
  - Analog zur JavaScript-Einbettung
  - `<script language="php"> PHP-Text </script>`

## (Lästige) Details: Syntaktische Unterschiede

- Generell stärkere Anlehnung an Shell-Skriptsprachen
  - Variablen beginnen immer mit "\$"
  - Viele UNIX-Kommandos direkt verfügbar, z.B.

```
echo "Beispiel";
```

(statt in JavaScript: `document.write("Beispiel");`)
- Verschiedene Varianten für Steueranweisungen, z.B.:

```
if (bedingung1) anw1 elseif (bedingung2) anw2 else anw3;
if (bedingung1): anwfolge1 elseif (bedingung2): anwfolge2
else: anwfolge3 endif;
```
- Schwach typisiert, aber geringfügig anderes Typsystem zu JavaScript
- Arrays einschließlich assoziativer Arrays, aber etwas andere Syntax und Bibliothek als in JavaScript
- PHP ist weitgehend objektorientiert, kennt Klassen und Vererbung in Java-Syntax.

## Beispiel: Fibonacci-Funktion mit PHP (Version 1)

```
<body> ...
  <h2>
    <?php
      function fib($n){
        if ($n==0)
          return 0;
        else
          if ($n==1)
            return 1;
          else
            return fib($n-1)+fib($n-2);
      };
      echo "fib(3) = ", fib(3), "<br>";
      echo "fib(8) = ", echo fib(8), "<br>";
    ?>
  </h2>
</body>
</html>
```

## Fibonacci-Funktion mit PHP (Version 2): Eingabeseite mit Aufruf von PHP-Skript

```
<body>
  <h1>
    Fibonacci-Funktion (Eingabe)
  </h1>
  <h2>
    Bitte Zahlwert f&uuml;r Berechnung eingeben:
    <form name="formular" action="fibonacci2b.php">
      <input type="text" name="eingabe"
        value="0"><br>
      <input type="submit" value="Berechnen">
    </form>
  </h2>
</body>
</html>
```

## Fibonacci-Funktion mit PHP (Version 2): Ergebnisseite

```
<body>
  <h1>
    Fibonacci-Funktion (Ergebnis)
  </h1>
  <h2>
    <?php
      function fib($n){ wie in Version 1 };
      echo "fib($eingabe) = ";
      echo fib($eingabe);
      echo "<br>";
    ?>
    <br>
    <a href="fibonacci2a.html">Neue Berechnung</a>
  </h2>
</body>
```

## Direkter Aufruf der Ergebnisseite

- In Formular eingegebene Werte werden an (über "action") aufgerufenes Skript übergeben
  - In PHP als Variablen verfügbar
- GET-Methode:
  - HTML: method="get" in "form"-Tag (default!)
  - Variablenwerte werden als Bestandteil der URL codiert und übergeben:  
`http://host.dom/pfad/fibonacci2.php?eingabe=12`
  - Damit kann die Ergebnisseite im Beispiel auch direkt aufgerufen werden
- POST-Methode:
  - HTML: method="post" in "form"-Tag
  - Variablenwerte werden im Rumpf der Nachricht codiert und übergeben
  - Schwerer von aussen zu "manipulieren"

## Kombination von Eingabe- und Ergebnisseite

```
<body>
  <h1>
    Fibonacci-Funktion
  </h1>
  <h2>
    <?php
      function fib($n){      wie oben  };
      echo "fib($eingabe) = ";
      echo fib($eingabe);
      echo "<br>";
    ?>
    <br>
    Bitte Zahlwert f&uuml;r neue Berechnung eingeben:
    <form name="formular" action="fibonacci2.php">
      <input type="text" name="eingabe" value="0"><br>
      <input type="submit" value="Berechnen">
    </form>
  </h2>
</body>
```

## Beispiel für Browseranzeige



## Dauerhafte Speicherung von Information

- Webseiten-Inhalt soll oft von gespeicherter Information abhängen
  - E-Commerce, E-Government, ...
  - Personalisierung von Seiten
  - Diskussionsforen
  - ...
- Serverseitige Speicherung:
  - Grosse Datenmengen (Datenbank)
    - » aber auch einfache Dateien möglich
  - Aktualisierung durch externe Programme
  - Anbindung an komplexe Programmsysteme
- Clientseitige Speicherung:
  - Geringe Datenmengen
  - Starke Einschränkungen aus Sicherheitsgründen
  - Für Identifikation etc.: "Cookies"

## Cookies

- Kleine im Browser (bzw. einer vom Browser kontrollierten Datei) gespeicherte Dateneinheiten
- Cookie enthält:
  - Name (String), auch Schlüssel genannt
  - Wert (String)
  - Verfallsdatum
  - optional: Domäne, Pfadname, Sicherheitsinformation
- Übertragung zwischen Client und Server im HTTP-Protokoll
  - Zu jeder Anfrage nach einem Dokument werden alle *zugehörigen* Cookies an den Server gesandt.
- Auf ein Cookie kann nur das Programm/der Server zugreifen, der das Cookie erzeugt hat
- Clientseitige Erzeugung/Zugriff: z.B. mit JavaScript
- Serverseitige Erzeugung/Zugriff: z.B. mit PHP

## Cookies in PHP (1)

```
<?php
    $zeit = time() + $tim * 60;
    setcookie($key, $val, $zeit);
    echo "<tt>Cookie: ", $key, "=", $val, "<br>";
    echo "gültig bis: ",
        date("d. F Y G:i:s", $zeit);
    echo "</tt><br>";
?>
```

- Benutzergegebene Parameter (rot dargestellt, gesetzt in separater HTML-Datei mit <form>):
  - Verfallszeit (hier in Minuten ab "jetzt")
  - Name
  - Wert
- Achtung: Apache-spezifische Funktion "setcookie" ...

## Cookies in PHP (2)

```
<?php
  echo "<b>Gesetzte Cookies:</b><br>\n";
  echo $HTTP_COOKIE, "<br><br>\n";
  while (list($key, $wert) = each($HTTP_COOKIE_VARS))
    echo $key, "=", $wert, "<br>\n";
?>
```

- Aktuelle Cookies verfügbar über eingebaute Variable
  - codiert in \$HTTP\_COOKIE
  - in assoziativem Array \$HTTP\_COOKIE\_VARS
- Verwendete Array-Funktionen:
  - `each()`: Durchläuft alle Array-Elemente
  - `list()`: Weist Array-Werte an Variablen-Tupel zu

## Ein einfaches Diskussions- forum (1)

- Interaktive Eingabe von Beiträgen
- Aktuelle Anzeige aller Beiträge
- Speicherung des aktuellen Diskussions-Standes in Datei auf Server
- Nur 53 Zeilen HTML+PHP !

### Diskussionsforum

#### Neuer Beitrag:

Name:   
Beitrag (1 Zeile):

#### Bisherige Beiträge:

3 Beiträge

##### Beitrag Nr. 1:

Name: Max  
Beitrag: Das ist interessant.

## Ein einfaches Diskussionsforum (2)

- Beispieldinhalt der Datei "forum.txt":
  - Je zwei Zeilen gehören zusammen.
  - Erste Zeile: Name
  - Zweite Zeile: Inhalt

**Max**

Das ist interessant.

**Moritz**

Das ist eher langweilig.

## Ein einfaches Diskussionsforum (3)

- Ausgabe der aktuell bekannten Beiträge aus Datei
- Verwendete Dateifunktion:
  - `file()`: Wandelt Dateiinhalt in String-Array
- Verwendete Arrayfunktion:
  - `count()`: Länge des Arrays

```
<h2>Bisherige Beitr&auml;ge:</h2>
<?php
    $inhalt = file("forum.txt");
    echo "<h3>", count($inhalt)/2, " Beitr&auml;ge</h3>";
    $i = 0;
    while ($i < count($inhalt)) {
        echo "<h3>Beitrag Nr. ", ($i+2)/2, " :</h3>";
        echo "<b>Name: &nbsp;</b>", $inhalt[$i++], "<br>";
        echo "<b>Beitrag: &nbsp;</b>", $inhalt[$i++], "<br>";
    }
?>
```

## Ein einfaches Diskussionsforum (4)

- Code zum Erweitern der Datei entweder in separatem Skript oder in gleicher Datei wie Anzeige-Skript (hier gezeigt)
  - Abfrage, ob Eintragen nötig (Variable \$eintragen zeigt, ob Einfügen-Schaltfläche gedrückt wurde)
- Verwendete Dateifunktionen:
  - `fopen()`, `fclose()`: Datei öffnen ("a"=append), schliessen
  - `fputs()`: Zeichenreihe schreiben

```
<?php
    if ($eintragen != "" &&
        $name != "" && $beitrag != "") {
        $datei = fopen("forum.txt", "a");
        fputs($datei,$name . "\n");
        fputs($datei,$beitrag . "\n");
        fclose($datei);
    }
?>
```

## Weitere Eigenschaften von Server-Skripten

- Datenbankanschluss
  - z.B. über Open Database Connectivity (ODBC)
  - Funktionen zum Öffnen, Abfragen, Manipulieren von Datenbanken und Tabellen in Datenbanken
- Reguläre Ausdrücke
  - zur flexiblen Zeichenreihen-Verarbeitung
- Erzeugung von Print-Dokumenten
  - spezielle Bibliothek zur PDF-Erzeugung in PHP
- Weitere Protokolle
  - z.B. FTP, Email-Protokolle (SMTP, POP, IMAP)
- Diverses:
  - XML-Verarbeitung
  - Bildbearbeitung
  - Rechtschreibprüfung
  - Passwortsicherheit
  - ...

## Server-Skripte vs. Client-Skripte

### Client-Skripte

Schnelle Reaktion  
Funktion auch ohne Netzanbindung  
Unabhängigkeit von Server-Software

Berechnung von Seiteninhalt aus  
Benutzereingaben und anderen  
äusseren Umständen

### Server-Skripte

Datenhaltung auf Server  
Zugriff auf zentrale Ressourcen (z.B. zur Weiterverarbeitung)  
Unabhängigkeit von Browser-Software