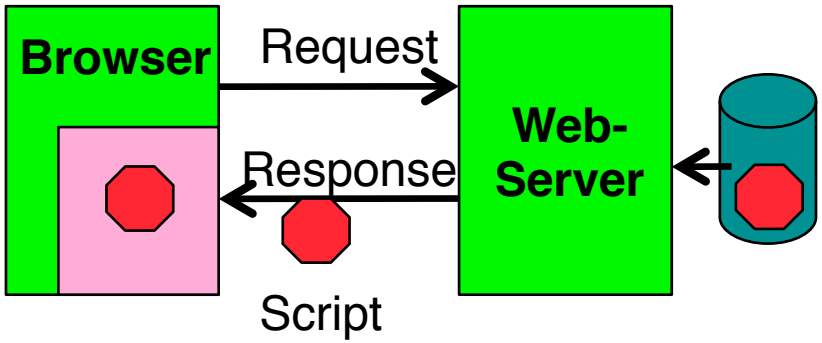
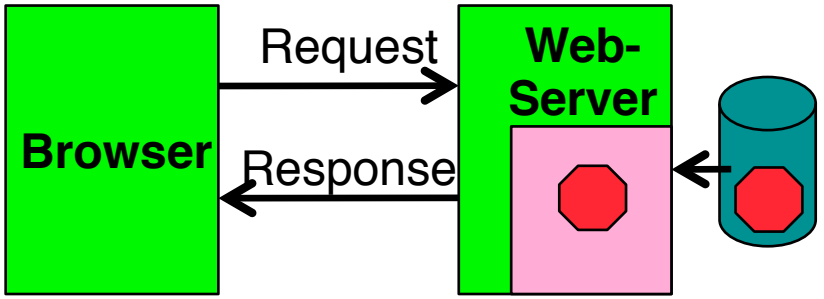
Chapter 3: Interactive Web Applications

Dynamic Web Contents

- Contents shown to user in browser is dependent on some external variables
- Examples of external variables:
  - Date and time
  - Contents of an information archive (e.g. recent news)
  - Actions of the user
    - » Pointing to elements
    - » Clicking at a certain position
    - » Filling out forms
- Wide-spread applications:
  - E-Commerce
  - Interpersonal communication media (forums, discussion boards)
  - Mass media (news and other information services)

Server-Side vs. Client-Side Realisation

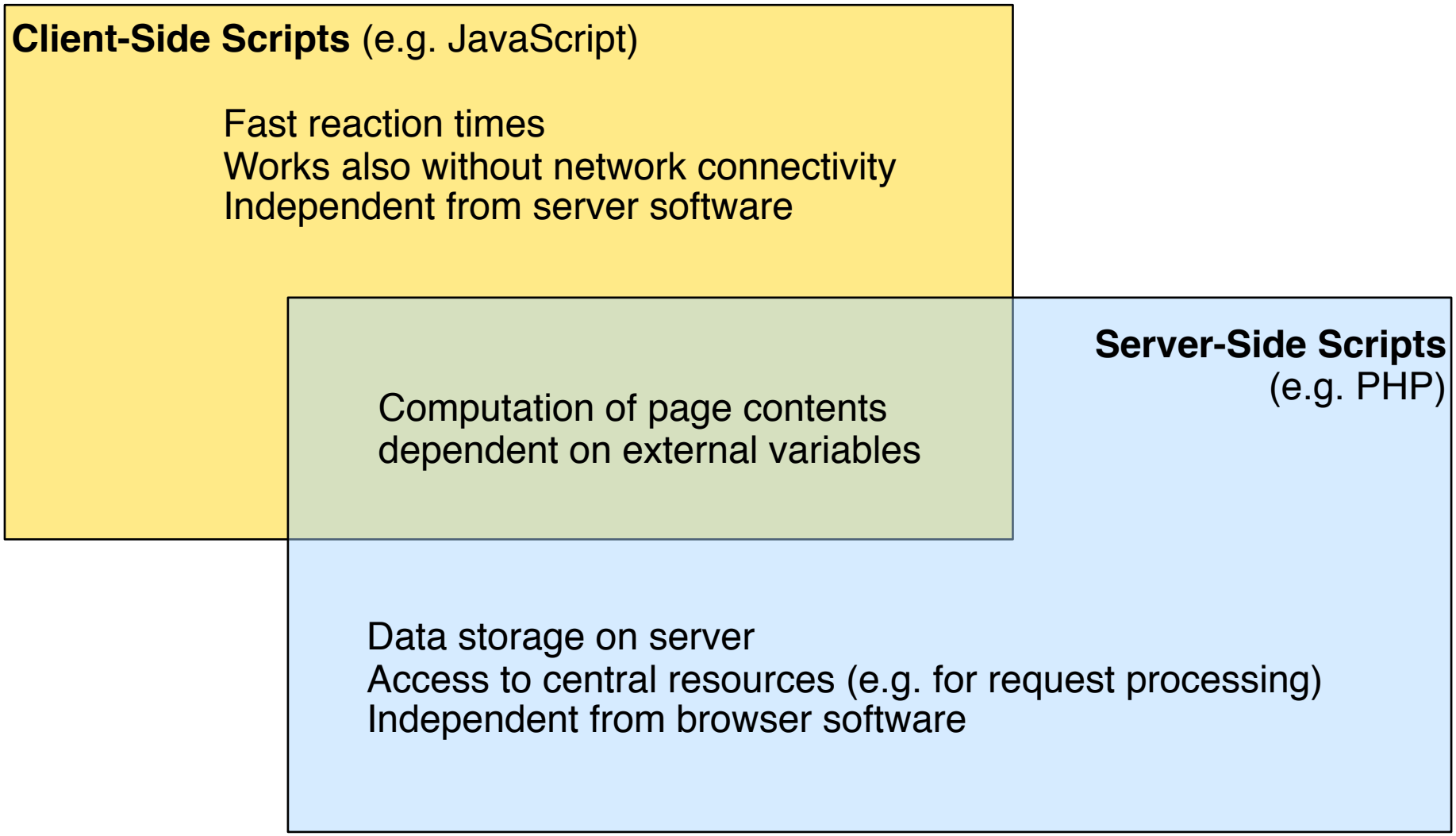




- Client-side realisation:
  - Browser contains execution engine for scripts
  - Web server does not need to execute scripts
  - Script ist sent to client as part of server response
  - Example: JavaScript

- Server-side realisation:
  - Web server contains execution engine for scripts
  - Browser does not need to execute scripts
  - Script is executed on server and computes response to client
  - Example: PHP

Server Scripts vs. Client Scripts



Common Gateway Interface (CGI)

- A request can identify an executable command on the server
  - Command is executed
  - Parameters are passed to it via environment variables (e.g. QUERY_STRING)
- Informal standard, by a developer community in 1993
  - Current standard (1.1) is documented at NCSA (http://hoohoo.ncsa.illinois.edu/cgi/)
  - IETF RFC 3875
- CGI programs can be written in any executable language:
  - Programming languages (e.g. C/C++, Java)
  - Scripting languages (e.g. Unix shells, Perl, TCL)
- Typical locations on server file system:
  - **`/cgi-bin`**
  - **`/cgi-src`**

Principles of Writing CGI Code

- Passing parameters to the CGI program:

  http://www.example.com/cgi-bin/example.cgi?paraminfo

  – Program example.cgi is executed

  – String “paraminfo” is made accessible for the program in the environment variable QUERYSTRING

- Passing information to the browser:

  – The CGI program has to write the data in a form displayable by the browser

  – Always the first line is a MIME type specification, e.g.:

```
Content-type: text/html
```

- Example for a very simple CGI program:

```
#!/bin/sh
echo "Content-Type: text/plain"
echo ""
echo "Hello, world."
```

Drawbacks of CGI

- High danger of security problems:
  - Injection of malicious script code (through program errors)
- Calling a CGI command is expensive:
  - Creating a new process (in Unix)
  - Sometimes on demand compilation
  - Generally not suitable to high load situations

- Alternatives to CGI:
  - SCGI (Simple CGI)
  - FastCGI (single persistent process to handle queries)
  - WSGI (Web Server Gateway Interface) for Python
  - Microsoft Internet Server Application Programming Interface (IISAPI)
  - Server modules
    - » E.g. script language modules for Apache

Chapter 3: Interactive Web Applications

R. Lerdorf, K. Tatroe, P. MacIntyre, T. Apandi (Hg.), A. Randal (Hg.): Programming PHP, 2nd. ed., O'Reilly 2006

Server-Side Script Language PHP

(Only an example for a server-side script language!)

- PHP:
  - **P**ersonal **H**ome **P**age Toolkit
    - » 1995, Rasmus Lerdorf
    - » 2003, new by Zeev Suraski, Andi Gutmans
  - **P**HP **H**ypertext **P**reprocessor (recursive acronym, backronym)
- Current version: 5.3 (June 2009), 6 in preparation
- OpenSource project:
  - see www.php.net
  - Can be used and modified freely (PHP license)
- Syntax loosely oriented towards C
  - Variations of possible syntax
- Extensive function library
  - being extended by community

Prerequisites for Using PHP in Practice

- Always (even if using just one computer)
  - Installation of a Web server
    - » OpenSource: *Apache*
    - » Microsoft *Internet Information Server*
  - Invocation of PHP always indirectly by loading pages from server (http://...)
    - » Loading from local computer: http://localhost/...
- Installation of PHP software as plug-in for used Web server
- Very often also installation of a data base system (e.g. MySQL)
- Frequently used acronyms for specific configurations:
  - LAMP: Linux, Apache, MySQL, PHP
  - WIMP: Windows, Internet Information Server, MySQL, PHP
  - MOXAMP: MacOS X, Apache, MySQL, PHP

Activation of PHP Module in Apache

- Example (MacOS 10.5):
  - Apache + PHP module are pre-installed
  - Configuration needs to be updated (remove a comment sign)
- /etc/apache2/httpd.conf:

```
# This is the main Apache HTTP server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See <URL:http://httpd.apache.org/docs/2.2> for detailed information.
...
LoadModule bonjour_module     libexec/apache2/mod_bonjour.so
LoadModule php5_module        libexec/apache2/libphp5.so
#LoadModule fastcgi_module     libexec/apache2/mod_fastcgi.so
```

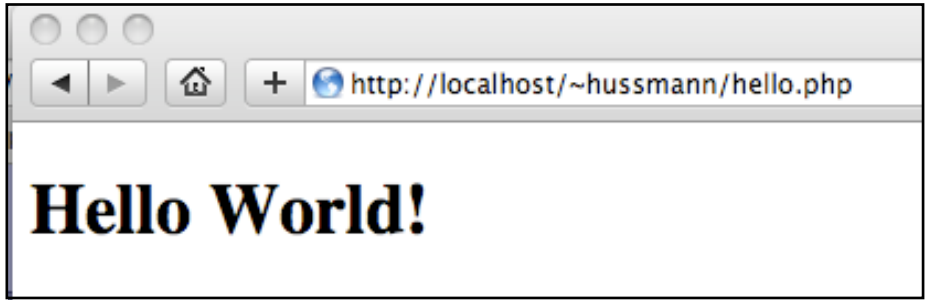
Hello World in PHP

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//
  EN" "http://www.w3.org/TR/html4/loose.dtd">


<html>
<head>
  <title>Hello World with PHP</title>
</head>


<body>
  <h1>
     <?php echo "Hello World!"; ?>
  </h1>
</body>
</html>
```

File hello.php
in Web server directory



Embedding of PHP into HTML

- XML style (used here):
  - Like *Processing Instructions* in XML

```
<?php    PHP Text   ?>
```

- SGML style:
  - Widely used in older scripts
  - Not really recommendable: PHP language not specified

```
<?    PHP Text   ?>
```

- HTML style:
  - Using HTML tag:

```
<script language="php">    PHP Text   </script>
```

A More Useful Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
   <title>User Agent Test with PHP</title>
</head>

<body>
   <h1>User agent used:</h1>
   <p>
       <?php echo $_SERVER['HTTP_USER_AGENT']; ?>
   </p>
   <p>
       <?php
              if (strpos($_SERVER['HTTP_USER_AGENT'], 'MSIE') == FALSE) {
                     echo "You are not using Internet Explorer.";
              }
       ?>
   </p>
</body>
</html>
```

PHP Syntax (1)

- Inheritance from shell scripts
  - Variables start with "$"
  - Some UNIX commands part of the language, e.g.:

```
echo "Hello";
```

- Control statements exist in different versions, e.g.:

```
if (bedingung1)
  anw1
elseif (bedingung2)
  anw2
else anw3;

if (bedingung1):          anwfolge1
elseif (bedingung2):      anwfolge2
else:                     anwfolge3
endif;
```

PHP Syntax (2)

- Various comment styles:
  - One-line comment, C style:

```
echo "Hello"; // Hello World
```

  - One-line comment, Perl style / Unix shell style:

```
echo "Hello"; # Hello World
```

  - "One line" ends also at end of PHP block
  - Multi-line comment, C-style:

```
echo "Hello"; /* Comment
  spreads over multiple lines */
```

  - Do not create nested C-style comments!

- Instruction must always be terminated with ";"
  - Exception: end of PHP block contains implicit ";"

PHP Type System

- Scalar types:
  - boolean, integer, float (aka double), string
- Compound types:
  - array, object
- Special types:
  - resource, NULL
  - Resource type: refers to external resource, like a file

- "The type of a variable is not usually set by the programmer; rather, it is decided at runtime by PHP depending on the context in which that variable is used."

  (PHP Reference Manual)

Arrays in PHP (1)

- An array in PHP is actually an ordered map
  - Associates values to keys
  - Keys can be integer or string (even mixed in same array)
  - Multi-dimensional arrays (arrays of arrays) are supported
- Multiple use of the array data structure for array, list, hash table, dictionary, stack, queue, …
- Creating arrays (examples):

```
<?php
  $arr = array("foo" => "bar", 12 => true);
  echo $arr["foo"]; // bar
  echo $arr[12];    // 1
?>
<?php
  $arr = array("somearray" => array(6 => 5, 13 => 9, "a" => 42));
  echo $arr["somearray"][6];    // 5
  echo $arr["somearray"][13];   // 9
  echo $arr["somearray"]["a"];  // 42
?>
```

Arrays in PHP (2)

- Arrays with strictly numerical keys
  - Implicit position numbers as keys

```
$array = array( 7, 8, 0, 156, -10);
  // this is the same as array(0 => 7, 1 => 8, ...)
```

- Arrays as collections

```
$colors = array('red', 'blue', 'green', 'yellow');
foreach ($colors as $color) {
    echo "Do you like $color?\n";
}
```

- Assignment operations on arrays always mean copying of values!

Object-Orientation in PHP (1)

```php
<?php
  class SimpleClass {

     // property declaration


  public $var = 'a default value';

     // method declaration
     public function displayVar() {
        echo $this->var;
     }
  }
```

Property access with "->" operator

Visibilities:
public, private, protected

```php
$instance = new SimpleClass();
$instance->var = 'property value';
$instance->displayVar();
```

Object-Oriented Concepts in PHP

- Static class properties and methods
  - "static" keyword
- Class Inheritance:
  - "extends" keyword in class definition
- Class Abstraction:
  - "abstract" keyword in class definition
- Scope Resolution operator ("::"):
  - Access to static, constant or overridden properties or methods of a class

```php
<?php
        class MyClass {
                const CONST_VALUE = 'A constant value';
        }
        $classname = 'MyClass';
        echo $classname::CONST_VALUE; // As of PHP 5.3.0
?>
```

  - In combination with "self" and "parent" keywords (denoting classes): Possibility to access overridden version of a method (cf. "super" in Java)

Example: Fibonacci Function in 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) = ", fib(8), "<br>";
     ?>
  </h2>
</body>
</html>
```

fibonacci1.php

HTML Reminder: Forms

- User input in HTML:
  `<form>` Element
- Sub-element:
  - **`<input type=`*`ty`* **`name=`*`name`*`>`**
    Allowed types (*ty)* (selection):

`checkbox`	Check box (Attribute **`checked`**)
`radio`	Radio button (Attribute **`checked`**)
`text`	Textzeile
`textarea`	Multi-line text input area
`password`	Text input area not displaying the input
`file`	File selection
`button`	General button
`submit`	Button to send form contents
`reset`	Button to reset form contents

  - **`<select name=`*`name`*`>`** Pop-up menu for selection from options
    List of options: Sub-elements **`<option>`**
    `<option selected>` defines "pre-selected" values

Forms and Server-Side Scripts

- User input into forms
  - Has to be transferred to server
  - Is evaluated in the server script
  - Can be displayed afterwards in a way determined by the script
- HTML: **`action`** attribute for tag **`<form>`**
  - Specifies the server page to process the input
  - Can contain embedded script
- PHP:
  - Well suited for processing input from forms
  - Special syntactic support for form values
    - » (Old versions of PHP: Simply made available as variables)
- Example:
  - **`<form name="formular" action="script.php">`**

GET and POST Methods in HTTP

Hypertext Transfer Protocol (HTTP) supports two methods for passing parameter values to called documents/scripts:

- GET Method:
  - Values of variables coded and transmitted within URL:

    `http://host.dom/pfad/fibonacci2.php?eingabe=12`
  - Parameters can be passed just by creating a certain URL (without forms)
  - Suitable for simple requests
- POST Method:
  - Values of variables not visible in URL
  - Web server reads parameter values from standard input (like the HTML text)
  - (Slightly) more difficult to "manipulate“
- HTML: Attribut **`method`** for form tag **`<form>`**
  - **`method="get"`** (default!) or **`method="post"`**

Fibonacci Function in PHP (Version 2): Input Form Calling PHP Script

```
<body>
  <h1>
     Fibonacci Function (Input)
  </h1>
  <h2>
     Please enter number:
     <form name="fibform" action="fibonacci2b.php">
          <input type="text" name="fibinput"
            value="0"><br>
          <input type="submit" value="Compute">
     </form>
  </h2>
</body>
</html>
```

fibonacci2a.html

Fibonacci-Funktion in PHP (Version 2): Result Page

```
<body>
   <h1>
          Fibonacci Function (Result)
   </h1>
   <h2>
          <?php
               $fibinput = $_REQUEST['fibinput'];
               function fib($n){ as in version 1 };
               echo "fib($fibinput) = ";
               echo fib($fibinput);
               echo "<br>";
          ?>
          <br>
          <a href="fibonacci2a.html">New Computation</
   a>
   </h2>
</body>
```

fibonacci2b.php

Variables, Parameter Passing and Security

- Global arrays $_REQUEST, $_GET, $_POST
  - for accessing external values determined at call time (like form input)
  - $_REQUEST contains all parameters given in request,
    $_GET and $_POST contains all parameters passed by the resp. method
  - Obtaining individual variable values by array lookup:

    ```
    $_REQUEST['var'];
    ```

- Older PHP versions (up to 4.2.0):
  - Huge security hole by not distinguishing between external values (like form input) and local variables
    - » External values were directly accessible through variables
      (like "$fibinput")
  - Weak type system of PHP makes situation worse
  - Manipulations of URL (GET parameter values) may enable setting of internal variables (e.g. “authorization_successful”...!)
  - Old behaviour can still be enabled by server configuration

Combination of Input and Result Pages

```
<body>
  <h1>
      Fibonacci Function
  </h1>
  <h2>
      <?php
            function fib($n){  as above  };
            $eingabe = $_REQUEST['fibinput'];
            echo "fib($fibinput) = ";
            echo fib($fibinput);
            echo "<br>";
      ?>
      <br>
      Please enter number:
      <form name="fibform" action="fibonacci2.php">
            <input type="text" name="fibinput" value="0"><br>
            <input type="submit" value="Compute">
      </form>
  </h2>
</body>
```

fibonacci2.php

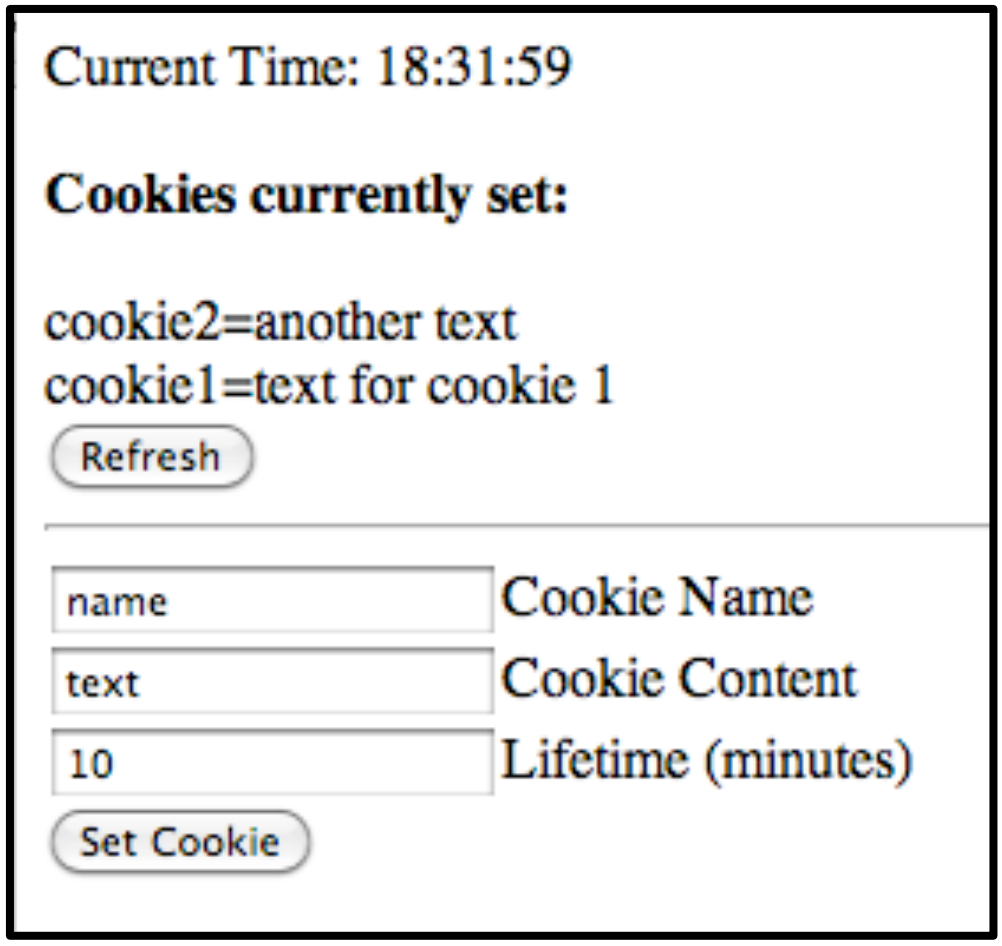
Permanent Storage of Information

- Displayed content very often comes from server or client side storage
  - E-Commerce, E-Government, ...
  - Personalized pages
  - Discussion fora
  - ...
- Server-side storage:
  - Huge amounts of data (database)
    - » or simple files!
  - Data update by external software
  - Integration with arbitrary software systems
- Client-side storage:
  - Small amounts of data
  - Security-based restrictions (information accessible for user)
  - Frequently used for status storage, identification etc.: "Cookies"

Cookies

- Small data units stored in the browser storage area, controlled by browser
- Cookie contains:
  - *Name* (String), also called *key*
  - *Value* (String)
  - *Expiration date*
  - optional: domain, path, security information
- HTTP transfers cookies between client and server
  - In each request all *related* cookies are sent from client to server
- Cookie is accessible only for the program/server which has created it
- Client-side creation/access: e.g. with JavaScript
- Server-side creation/access: e.g. with PHP
  - Cookies available in global array $_COOKIE

Cookies in PHP: Screenshot



Accessing Cookies

Displaying a list of all cookies currently set (for this application):

```
<html>
  <?php
    date_default_timezone_set('Europe/Berlin');
    echo "Current Time: ", date("G:i:s"), "<br><br>\n";
    echo "<b>Cookies currently set:</b><br><br>\n";
    while (list($k, $v) = each($_COOKIE))
          echo $k, "=", $v, "<br>\n";
  ?>
...
</html>
```

HTML Form for Setting a Cookie

```
<form>
    <input type="text" name="key" value="name">
       Cookie Name<br>
    <input type="text" name="val" value="text">
       Cookie Content<br>
    <input type="text" name="tim" value="10">
       Lifetime (minutes)<br>
    <input type="submit" name="set"
       value="Set Cookie"><br>
</form>
```

- Page loaded via `action` is identical to page containing the form ("cookietest.php") – omitting the `action` attribute is sufficient.
- Due to server-side execution, the actual setting action can only be carried out when the next pages is loaded!
- "`name`" attribute of `submit` button required for distinction to other buttons ("refresh" in the example).

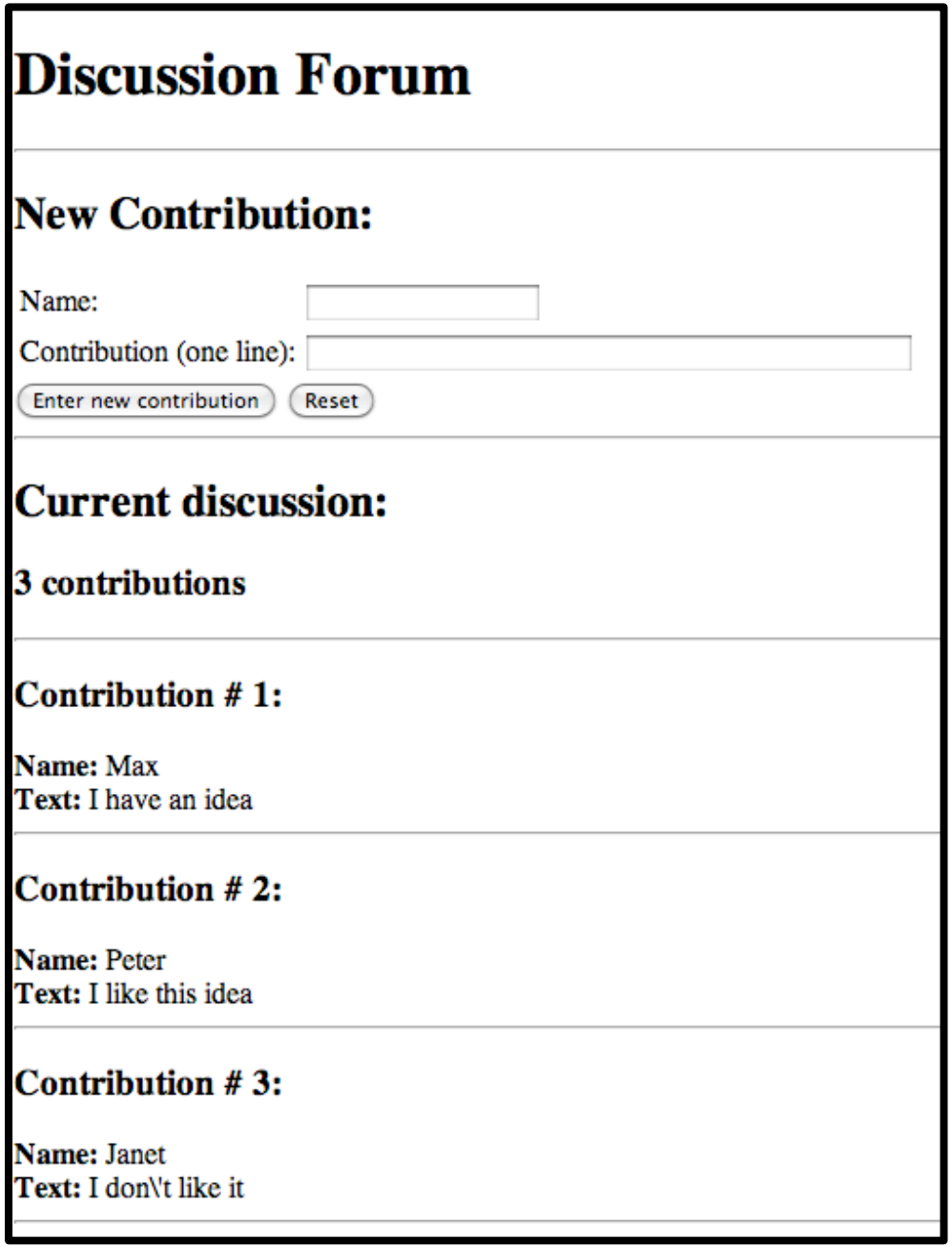
Setting the Cookie

```
<?php
   if ($_GET['set']) {
      $key = $_GET['key'];
      $val = $_GET['val'];
      $tim = $_GET['tim'];
      $exp = time() + $tim * 60;
      setcookie($key, $val, $exp);
   }
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
 Transitional//EN" "http://www.w3.org/TR/html4/loose">
<html>
...
```

- "`name`" attribute of `submit` button used to decide whether `"set"` button was pressed
- setcookie() call has to be very first output of page, to be transmitted together with the headers (HTTP requirement).

A Simple Discussion Forum (1)

- Interactive submission of text contributions
- Display of all submissions available on server
- Server uses simple text file for storage
- Altogether approx. 50 lines of HTML+PHP !



A Simple Discussion Forum (2)

Contents of file "forum.txt":

- Each two consecutive lines represent one contribution.
- First line: Name
- Second line: Text

```
Max
I have an idea
Peter
I like this idea
```

A Simple Discussion Forum (3)

Display of the full content of the file 'forum.txt'

- Used file function:
  - **`file()`:** Converts file content to string array
- Used array function:
  - **`count()`:** Length of array

```
<h2>Current discussion:</h2>
<?php
   $content = file("forum.txt");
   echo "<h3>", count($content)/2, " contributions</h3>";
   echo "<hr>";
   $i = 0;
   while ($i < count($content)) {
       echo "<h3>Contribution # ", ($i+2)/2, ":</h3>";
           echo "<b>Name: </b>", $content[$i++], "<br>";
           echo "<b>Text: </b>", $content[$i++], "<br>";
           echo "<hr>";
           }
?>
```

forum.php

A Simple Discussion Forum (4)

Extending the file 'forum.txt' with a new contribution

- Parameter $newcontrib indicates whether the "enter contribution" button was pressed

- Used file functions:
  - **`fopen(), fclose():`** Open file ("a"=append), close file
  - **`fputs():`** Write string to file

```
<?php
  $newcontrib = $_REQUEST['newcontrib'];
  $name = $_REQUEST['name'];
  $contrib = $_REQUEST['contrib'];
  if ($newcontrib != "" && $name != "" && $contrib != "") {
     $file = fopen("forum.txt", "a");
     if ($file) {
           fputs($file,$name . "\n");
           fputs($file,$contrib . "\n");
           fclose($file);
     }
  }
?>
```

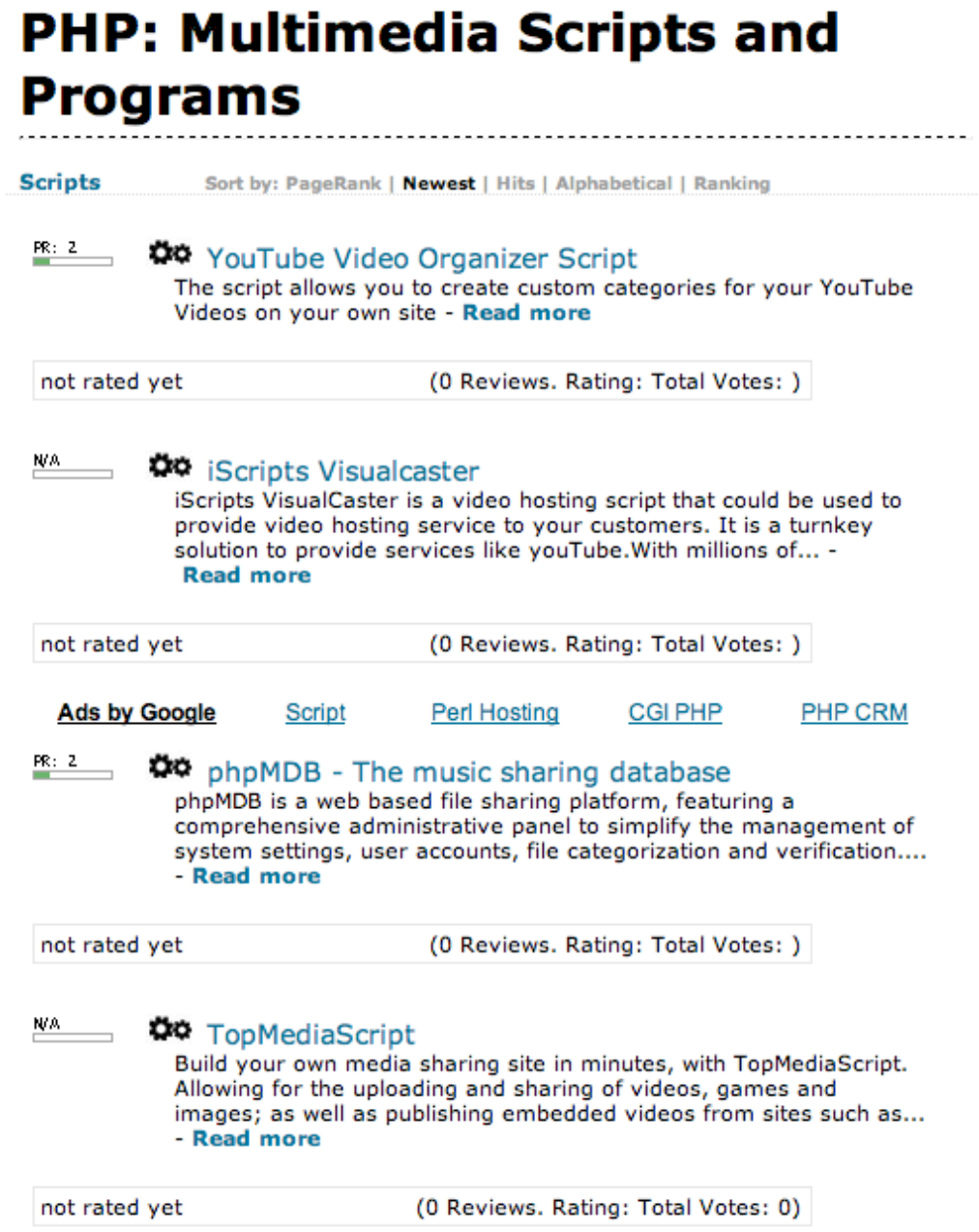
Potential Enabled by Server-Side Scripts

- Receive and store user input
  - In various forms of persistent storage
    - » Plain text files, XML files, data base
- Process input and compute results
  - Depending on various information available on server side
- Create output suitable for being displayed in Web browsers
  - HTML, may include JavaScript
- Make use of advanced features offered by Web browsers
  - Examples: Cookies, user agent identification

Applications to Multimedia

- PHP is not directly multimedia-related, but HTML-oriented
- HTML allows media embedding
- The combination of HTML + PHP + media embedding enables the creation of new digital media
- Examples for interactivity added to media playback, realizable by PHP scripts
  - Selection of media, e.g. search functions
    - » Using forms and backend data base
  - User-specific recommendations
    - » Using cookies
  - Aggregating (explicit and implicit) user input
    - » Frequency of use for individual media (charts)
    - » Correlation of use across media (collective recommendation)
    - » Tagging

Examples for PHP Multimedia Scripts



www.webscriptsdirectory.com

Multimedia Functions in PHP Library (1)

- See e.g. Multimedia chapter of tutorial "Practical PHP Programming" http://www.tuxradar.com/practicalphp/11/0/0
- Example: Creating an image

```
<?php
    $image = imagecreate(400,300);
   // do stuff to the image
   imagejpeg($image, '', 75);
   imagedestroy($image);
?>
```

File: picture1.php

```
<HTML>
   <TITLE>PHP Art</TITLE>
<BODY>
   <IMG SRC="picture1.php" />
</BODY>
</HTML>
```

- Computer graphics functions, like:

```
$white = imagecolorallocate($image, 255, 255, 255);
imagefilledrectangle($image, 10, 10, 390, 290, $white);
```

Multimedia Functions in PHP Library (2)

```
<?php
    $image = imagecreate(400,300);
    $gold = imagecolorallocate($image, 255, 240, 00);
    $white = imagecolorallocate($image, 255, 255, 255);

    imagefilledrectangle($image, 0, 0, 400, 300, $gold);

    for ($i = 4, $j = 3; $i < 400; $i += 8, $j += 6) {
      imagefilledrectangle($image, $i, $j, 400 - $i, $j+3, $white);
    }

    imagepng($image);
    imagedestroy($image);
?>
```

Creating Flash Movies from PHP (1)

- ***Ming*** is an open-source library for creating SWF (Shockwave for Flash) movies from PHP scripts, using an object-oriented style.

```php
<?php
    $mov = new SWFMovie();
    $mov->setDimension(200,20);

    $shape = new SWFShape();
    $shape->setLeftFill($shape->addFill(0xff, 0, 0));
    $shape->movePenTo(0,0);
    $shape->drawLineTo(199,0);
    $shape->drawLineTo(199,19);
    $shape->drawLineTo(0,19);
    $shape->drawLineTo(0,0);

    $mov->add($shape);
    header('Content-type: application/x-shockwave-flash');
    $mov->output();
?>
```

```
<EMBED src="ming1.php" menu="false" quality="best" bgcolor="#FFFFFF" swLiveConnect="FALSE" WIDTH="200" HEIGHT="200"
TYPE="application/x-shockwave-flash" PLUGINSPAGE="http://www.macromedia.com/shockwave/download/index.cgi?
P1_Prod_Version=ShockwaveFlash">
```

Creating Flash Movies from PHP (2)

- Creating an animation (here animated text):

```php
<?php
    $font = new SWFFont("Impact.fdb");
    $text = new SWFText();
    $text->setFont($font);
    $text->moveTo(300, 500);
    $text->setColor(0, 0xff, 0);
    $text->setHeight(200);
    $text->addString("Text is surprisingly easy");

    $movie = new SWFMovie();
    $movie->setDimension(6400, 4800);

    $displayitem = $movie->add($text);

    for($i = 0; $i < 100; ++$i) {
        $displayitem->rotate(-1);
        $displayitem->scale(1.01, 1.01);
        $movie->nextFrame();
    }

    header('Content-type: application/x-shockwave-flash');
    $movie->output();
?>
```