

PHP – Dynamic Web Pages, Without Knowing Beans

Kimberly Bobrow Jennery

Intelliclass

kimberly@bobrow.net

kimberly@jennery.com





Agenda

- Introduction
- Basic Syntax
- General Language Constructs
- File Handling
- Sending Email
- Accessing Databases
- Error Handling
- Demos throughout
- Questions, discussion



Assumed Knowledge

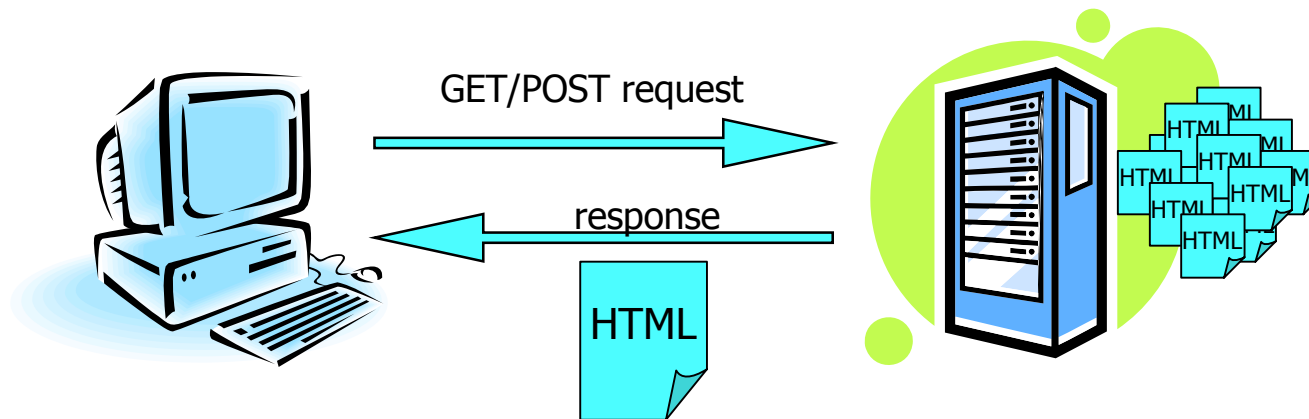
- Basic HTML concepts and tags
- Basic programming concepts



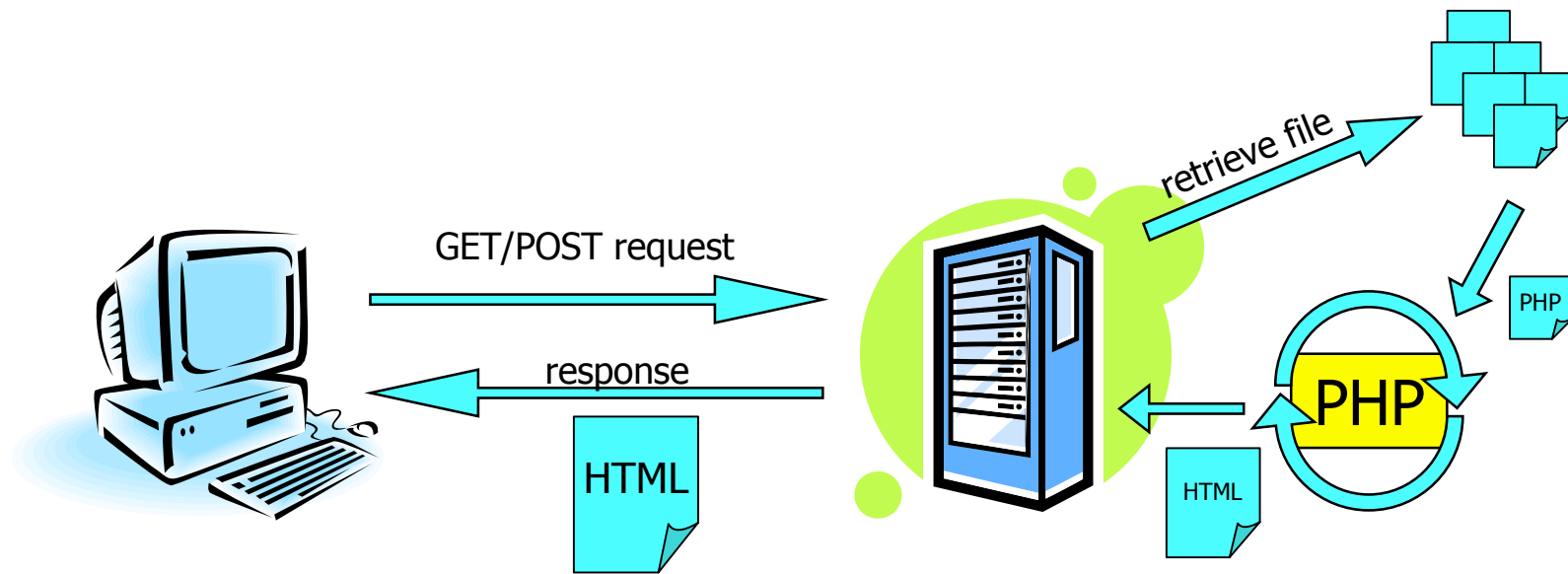
PHP – What Is It?

- **PHP: Hypertext Preprocessor**
 - Geekily recursive name
- Server side scripting language for producing dynamic web pages
- Runs on a web server
 - Takes PHP code as input
 - Interprets/Processes PHP code
 - Creates web pages as output
- Supported on all popular web servers, and most operating systems
- Open Source

Static Web Pages



Dynamic Web Pages





Identifying PHP Code

Tags used to escape from HTML:

- Always works: (Most Portable)

```
<?php
```

```
?>
```

- Shorthand: (Requires server support, not as portable)

```
<?
```

```
?>
```

Note: The file itself requires .php extension (not .htm or .html)





Simple Escaping Example

```
<p>This text will not be parsed</p>
```

```
<?php
```

```
echo '<p>This text will be parsed.</p>';
```

```
?>
```

```
<p> This text won't.</p>
```

Advanced Escaping Example

```
<?php
  if ($expression) {
    ?>
```

HTML
Not interpreted

→ `This is true.`

```
<?php
} else {
  ?>
```

→ `This is false.`

```
<?php
```

```
}
```

```
?>
```



Let's Look at Some Sample Code

- `escapeExample.php`



Basic Syntax

- Similar to Java and C based languages
 - Uses semicolon for statement end
 - White space is ignored
 - Uses curly braces for blocks of code { }
 - Case-sensitive... sometimes!



Comments

// Java style – one line, end of line comment

- Comments to the end of the line, or the end of the current block of php code, whichever comes first

shell style – one line, end of line comment

- Comments to the end of the line, or the end of the current block of php code, whichever comes first

/* multi-line
comment */



Types and Casting

- Boolean: (bool) (boolean)
- Integer: (int) (integer)
- Floating point number: (float) (double) (real)
- String: (string)
- Array: (array)
- Object: (object)
- Resource
- NULL



Variables

- \$ followed by name of variable, which must start with letter or underscore
- Not type specific
- Case sensitive
- Not required to be declared before initialized
- Assignments done by value
- Assign by reference in PHP 4 with & operator



Variables – Examples

```
<?php  
$name = "Joe Smith";  
$age = 25;  
$salary = 50000.00;  
?>
```



-
- Used to output a string
 - Can output variables
 - Easy concatenation of strings and other data types



echo Examples

```
<?php
$hello = "Hello";
echo "Hello World";
echo $hello." World";
echo "<br>Remember \"quotes\"";
?>
```



Strings

- Can be constants or held in variables
- Can be defined in single or double quotes
- There are a large number of core functions for string handling, including C-like printf



String Examples

```
<?php
$double_greeting = "Hello World";
$single_greeting = 'Hello World';
echo $single_greeting;
echo "<br>";
echo $double_greeting;
echo "I don't need to escape";
echo 'I said "I do not!"';
```





Operators

- Assignment operator
 - =
- Arithmetic operators
 - + - * / %
- Comparison operators
 - == != < > <= >=
- String operator
 - .
- Combination operators
 - += -= *= /= .=
- Pre/post decrement/increment operators





include and require

- Takes a file name and includes the file's contents into the current script
- Good for common code, headers, *etc.*
- After a failed include command, the rest of the script is still executed
- After a failed require command, the rest of the script is abandoned



include and require Examples

```
<?php
include("stdHeaders.php");
echo 'This is after the included file';
?>
```

```
<?php
require("stdVariables.php");
echo $br."This won't work if the require
  fails.";
?>
```





Decision Making

- if
- if/else
- elseif
- switch



if / elseif / else Example

```
<?php
if ($salary > 50000) {
    echo 'No free handouts.';
} elseif ($salary > 30000) {
    echo 'See if you can get a raise.';
} else {
    echo 'Please apply for assistance.';
}
?>
```





switch Example

```
<?php
switch($pet) {
  case 'dog':
    echo 'woof!';
    break;
  case 'cat':
    echo 'meow!';
    break;
  default:
    echo 'blurp!';
    break;
}
```





Let's Look at Some Sample Code

- `decisionExample.php`



Arrays

- Arrays in PHP are ordered maps – they map values to keys
- Keys may be integers or strings
- Undefined keys are zero based
- There are a large number of core functions for array handling



Array Examples

```
$days = array('Sunday', 'Monday', 'Tuesday',  
    'Wednesday', 'Thursday', 'Friday',  
    'Saturday');  
echo $days[3];
```

```
$dayName = array('Sun' => 'Sunday', 'Mon' =>  
    'Monday', 'Tue' => 'Tuesday', 'Wed' =>  
    'Wednesday', 'Thu' => 'Thursday', 'Fri' =>  
    'Friday', 'Sat' => 'Saturday');  
echo $dayName['Wed'];
```



Array Examples

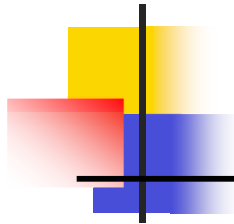
```
$days;
```

```
days[0] = 'Sunday';
```

```
days[1] = 'Monday';
```

```
days[2] = 'Tuesday';
```

```
$months = array(1=> 'January', 'February',  
                'March');
```



Looping

- while
- do/while
- for
- foreach



while Example

```
<?php
$counter = 10;
while ($counter > 0) {
    echo $counter.'<br>';
    $counter--;
}
echo 'blast off!';
?>
```



do-while Example

```
<?php
$counter = 10;
do {
    echo $counter.'<br>';
    $counter--;
} while ($counter > 0);
echo 'blast off!';
?>
```



for Example

```
<?php
for ($counter = 10; $counter > 0; $counter--) {
    echo $counter.'<br>';
}
echo 'blast off!';
?>
```



foreach

- Used with arrays
- Iterates through an array
- Syntax:

```
foreach (array_expression as  
    $value) statement
```

```
foreach (array_expression as $key  
    => $value) statement
```



foreach Example

```
$days = array('Sunday', 'Monday',  
    'Tuesday', 'Wednesday',  
    'Thursday', 'Friday',  
    'Saturday');  
  
foreach ($days as $d) {  
    echo '<br>'.$d;  
}
```



foreach Example

```
$days = array('Sun' => 'Sunday', 'Mon'  
=> 'Monday', 'Tue' => 'Tuesday', 'Wed'  
=> 'Wednesday', 'Thu' => 'Thursday',  
'Fri' => 'Friday', 'Sat' =>  
'Saturday');  
  
foreach ($days as $abbrev => $day) {  
    echo '<br>'.$abbrev." is ".$day;  
}
```



Let's Look at Some Sample Code

- `ArrayLoopExample.php`
- `showThumbs1.php`



Functions

- Prior to PHP 4, all functions were required to be defined before being referenced
- All functions, regardless of where defined, have global scope
- Conditional functions don't exist until they are reached in the code, and executed



Functions

- **Basic syntax:**

```
function foo($arg_1, $arg_2)
{
    echo "Example function.\n";
    ...
    return $retval;
}
```

- **Calling syntax:**

```
$result = foo('bar', 'baz');
```



Classes and Objects

- PHP Class: A collection of variables and functions working with these variables
- In general, must be declared in a single PHP block (except within a method declaration, a new PHP block may be started)
- Must be within a single source file
- Instances of classes are called objects
- `$this` pseudo-variable available inside class methods

Class Example

```
class Animal {  
    var $sound;  
    function Animal() {  
        $this->sound = "mrrrrrrrrrrrrrrrp!";  
    }  
    function makeSound() {  
        echo '<br>'.$this->sound;  
    }  
}
```

```
$anim = new Animal;  
$anim->makeSound();
```

—————→ **mrrrrrrrrrrrrrrrp!**

```
class Cat extends Animal {  
    function Cat() {  
        $this->sound = "meow!";  
    }  
}
```

```
$kitty = new Cat;  
$kitty->makeSound();
```

—————→ **meow!**



Let's Look at Some Sample Code

- `classExample.php`



File Handling

- Must declare read/write/*etc.* intentions when opening a file
- Basic functionality with standard methods:
 - fcreate
 - fopen
 - fread
 - fwrite
 - unlink (for file deletion)
 - fclose
- Care should be taken not to edit/delete the wrong files, corrupt files, fill a hard drive with garbage, *etc.*



File Handling Examples

```
$fileName = "test.txt";  
$fHandle = fopen($fileName, 'w');  
fwrite($fHandle, "First line\n");  
fwrite($fHandle, "Second line\n");  
fclose($fHandle);
```

```
$fh = fopen("test.txt", 'r');  
$allText = fread($fh, filesize($fileName));  
echo $allText;
```



Let's Look at Some Sample Code

- `fileAccessExample.php`



HTML Form Handling

- User data will be in the associative array variables `$_POST` or `$_GET`, depending on the method used by the form
- All user data and cookies will be in `$_REQUEST` regardless of the method used by the form
- Only 'name' attributes from the form will be available in the `$_POST` and `$_GET` arrays. 'id' attributes are not transmitted.



HTML Form Handling Example

Form:

```
<form action="formDemo.php" method="post">
  <p>Your name: <input type="text" name="name" /></p>
  <p>Your age: <input type="text" name="age" /></p>
  <p><input type="submit" /></p>
</form>
```

PHP Script:

```
Hi <?php echo $_POST['name']; ?>.
You are <?php echo $_POST['age']; ?> years old.
```





Let's Look at Some Sample Code

- formDemo.htm
- formDemo.php



Handling File Uploads

- Two part process:
 - HTML form to handle the user interface
 - php script to handle the saving of the file on the server
- File upload form must have attribute *enctype="multipart/form-data"*
- Use global array `$_FILES` for file information, as of PHP 4.1.0 (use `$HTTP_POST_FILES` in earlier versions)
- Files will, by default, be stored in the server's default temporary directory



File Upload Example

Form:

```
<!-- The data encoding type, enctype, MUST be specified as below -->
<form enctype="multipart/form-data" action="uploadFile.php"
  method="POST">
  <!-- MAX_FILE_SIZE must precede the file input field -->
  <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
  <!-- Name of input element determines name in $_FILES array -->
  Send this file: <input name="userfile" type="file" />
  <input type="submit" value="Send File" />
</form>
```



File Upload Example

php script:

```
<?php
$uploaddir = './uploads/';
$uploadfile = $uploaddir.basename($_FILES['userfile']['name']);

if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile))
{
    echo "File is valid, and was successfully uploaded.\n";
} else {
    echo "File upload failed!\n";
}
?>
```





Let's Look at Some Sample Code

- `uploadExample.htm`
- `uploadExample.php`



Sessions

- Used for passing data from one page to another during a user's visit to your site
- Start a session with the `start_session()` method
- A session must be started before any other HTML or text is sent
- End a session with the `end_session()` method
- Update the associative array `$_SESSION` with session data



Session Sample

```
session_start();

if(isset($_SESSION['views']))
    $_SESSION['views']++;
else
    $_SESSION['views'] = 1;

echo "Pageviews = ".$_SESSION['views'];
```



Let's Look at Some Sample Code

- `sessionExample.php`



Cookies

- Create a cookie with the `setcookie()` method
- A cookie must be set before any other HTML or text is sent
- Incoming cookies are stored in the global `$_COOKIE` associative array



Cookie Example

```
//Calculate 30 days in the future
//seconds * minutes * hours * days + current time
$inOneMonth = 60 * 60 * 24 * 30 + time();
setcookie(lastVisit, date("G:i:s - m/d/y"), $inOneMonth);

if(isset($_COOKIE['lastVisit'])) {
    $visit = $_COOKIE['lastVisit'];
    echo "Your last visit was - ". $visit;
}
else {
    echo "First time here, or first time for a while!";
}
```





Let's Look at Some Sample Code

- `cookieExample.php`



Security

- A full discourse on PHP security would be a 90+ minute talk on its own... or more!
- As always, you must balance between risk and usability
- Considerations:
 - Installation choices
 - Filesystem security
 - Database security
 - Error reporting



Sending email

- Send email using the `mail()` function
- PHP must have access to the *sendmail* binary on your system during compile time
- Syntax for `mail()`:

```
bool mail ( string to, string subject, string  
           message [, string additional_headers [, string  
           additional_parameters]] )
```



mail() Notes

```
bool mail ( string to, string subject, string message [, string  
    additional_headers [, string additional_parameters]] )
```

to:

The formatting of this string must comply with RFC 2822. Some examples are:

user@example.com

user@example.com, anotheruser@example.com

User <user@example.com>

User <user@example.com>, Another User anotheruser@example.com

from:

This must not contain any newline characters, or the mail may not be sent properly.

message:

Each line should be separated with a LF (\n). Lines should not be larger than 70 characters.



mail() Notes

```
bool mail ( string to, string subject, string message [, string  
    additional_headers [, string additional_parameters]] )
```

additional_headers (optional)

String to be inserted at the end of the email header.

This is typically used to add extra headers (From, Cc, and Bcc). Multiple extra headers should be separated with a CRLF (`\r\n`).

When sending mail, the mail *must* contain a From header. This can be set with the *additional_headers* parameter, or a default can be set in `php.ini`.

Failing to do this will result in an error message similar to Warning: mail(): "sendmail_from" not set in `php.ini` or custom "From:" header missing. The From header also sets Return-Path under Windows.

Return Values

Returns TRUE if the mail was successfully accepted for delivery, FALSE otherwise.

It is important to note that just because the mail was accepted for delivery, it does NOT mean the mail will actually reach the intended destination.





mail() Example

```
mail ("kimberly@bobrow.net",  
      "Simple Email message",  
      $long_text_msg);
```



Accessing Databases

- PHP supports many databases
- PHP also has ODBC support
- MySQL is commonly used with PHP



Talking to MySQL from PHP

```
<?php
include( 'dbinfo.php' );

$conn = mysql_connect( $dbhost, $dbuser, $dbpass );
mysql_select_db($dbname);

$sql = 'SELECT * FROM employee';
$tmpResult = mysql_query( $sql );
while ( $tmpRow = mysql_fetch_row( $tmpResult ) ) {
    // do stuff with data in $tmpRow;
}

mysql_close ( $conn );
?>
```





Let's Look at Some Sample Code

- `uploadImage.htm`
- `uploadImage.php`
- `showThumbs.php`



Error Handling

- Script logic
 - Checking return codes
 - Data validation
 - *etc.*
- Exceptions
- User defined error handler
- Error logging



Exceptions

- PHP 5 only
- Standard try/catch format:

```
try {  
    $error = 'Always throw this error';  
    throw new Exception($error);  
  
    // Code following an exception is not executed.  
    echo 'Never executed';  
  
} catch (Exception $e) {  
    echo 'Caught exception: ', $e->getMessage(), "\n";  
}
```



Error Handling Functions

- You can define your own error handling function with the `set_error_handler()` method
- Good for situations where you need to do final cleanup of data or files in the event of a critical error

- **Syntax:**

```
mixed set_error_handler ( callback error_handler [, int error_types] )
```



Error Logging

- You can send a message to the web server's error log, a TCP port, or to a file with the `error_log()` method

- **Syntax:**

```
bool error_log ( string message [, int message_type  
                [, string destination [, string extra_headers]] ] )
```



Changes for PHP 5

- PHP 5 has improved performance and some new capabilities, but is intended to be mostly backwards compatible.
- Migrating from PHP 4 to PHP 5 should be mostly seamless.
- There are some new reserved words
- The object model was overhauled for performance and additional features, including addressing by reference
- Some system defaults have changed



Useful Functions to Know About

- Array functions
- Calendar functions
- Date and Time functions
- Class and Object functions
- Directory functions
- Image functions
- Math functions
- String functions
- HTTP functions
- ZIP file functions



Useful Resources

- <http://www.apachefriends.org/en/xampp.html>
 - Quick installation of Apache, php, mySQL, *etc.*
- <http://us3.php.net/docs.php>
 - PHP documentation



Demos, Discussion, Questions?





Contact Info

Kimberly Bobrow Jennery

kimberly@jennery.com

kimberly@bobrow.net

