CS2003
Advanced Internet Programming

Building Internet Applications
09 – Web Development with PHP (I)

Slides by Jason T. Jacques (http://www.cs.st-and.ac.uk/~jtj2)

Objectives

At the end of these lectures you should be able to:

- Recognise a basic PHP script
- Execute PHP scripts on your school host server
- Write a simple PHP program
- Identify some common caveats of PHP

Web technologies you are familiar with:

- Markup
  - HTML
- Presentation
  - CSS (Cascading Style Sheets)
- Programming
  - Client-side
    - JavaScript
  - Server-side

PHP

PHP: Overview

- popular web programming language
  - Facebook, Wikipedia, Yahoo!
- scripting language
  - interpreted at runtime
- c-type syntax
  - like JavaScript
- weak, dynamic type system
  - automatic type coercion
- large standard library
  - and extension system

- used inline with HTML
  - pre-processed by server

```
<html>
    <body>

        My first program, says:

        <?php

            /*
             * Example 1: Hello
             * file: 01-hello.php
             */

            echo "Hello, World!";

        ?>

    </body>
</html>
```

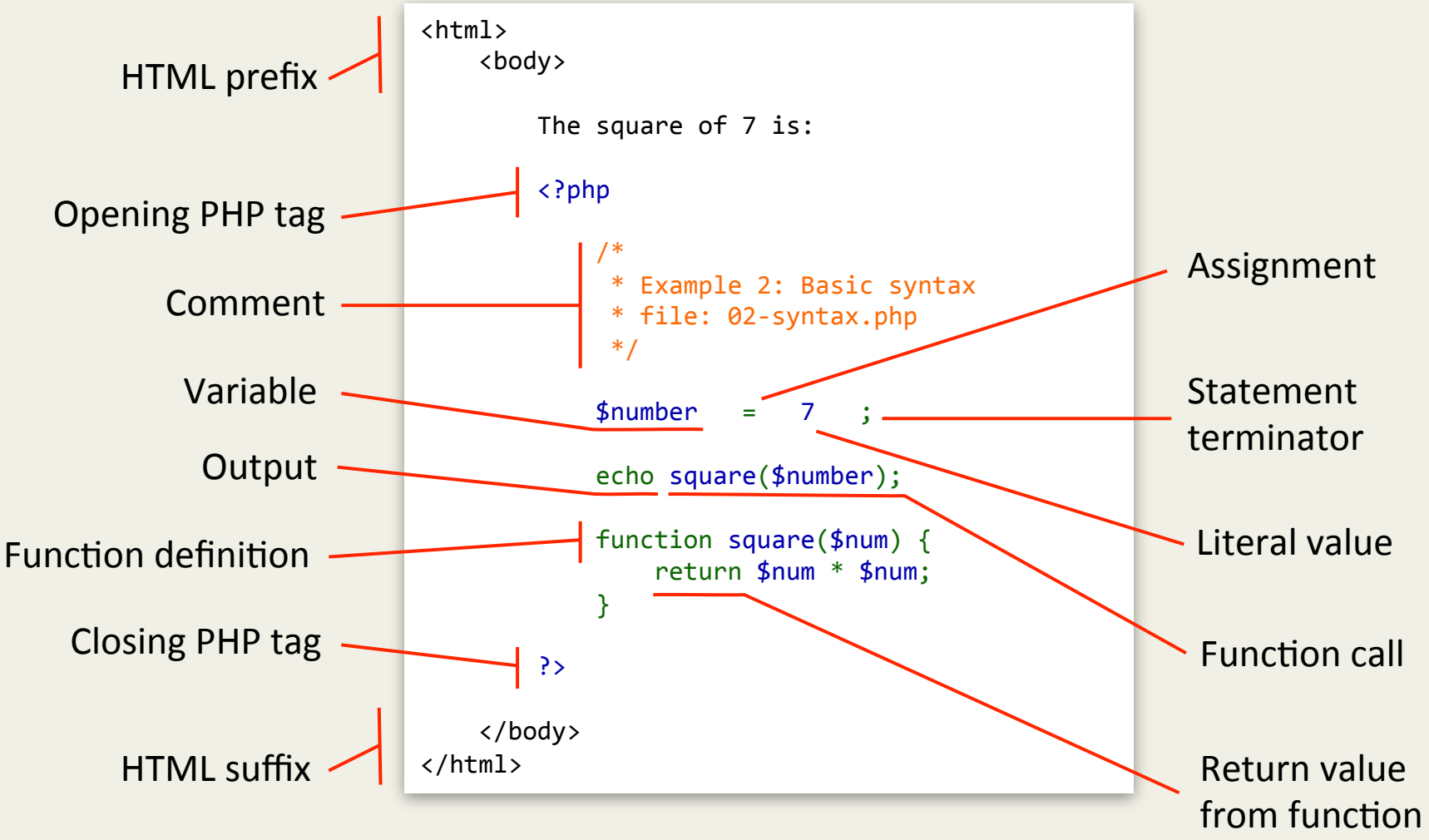
Executing PHP scripts

- PHP files (typically)
  - use the '.php' extension
  - are executed on a webserver
- Files should be uploaded to your cs home directory
  - \\<username>.home.cs.st-andrews.ac.uk\<username>
- You can access your files at
  - http://<username>.host.cs.st-andrews.ac.uk/
- All examples will be provided

SYNTAX

& CORE FUNCTIONS

Basic syntax



Variables

- Variables are prefixed with '$'
  - names are case sensitive
- Numbers 'types' use standard operators
  - +, -, *, /, %
- Arrays
  - defined with array();
  - accessed with []
  - zero-indexed

```php
<?php

    /*
     * Example 3: Variables
     * file: 03-variables.php
     * note: this script has no output
     */

    $i = 5;
    $t = "some text";

    $number = 6 * 2;              // 12
    $number = $number - $i;       // 7

    $array = array(1, 2, "three");
    $map = array("apples" => 2, "oranges" => 6);

    $a = $array[2];               // "three"
    $m = $map["oranges"];         // 6

?>
```

String (text) manipulation

- String concatenation with the '.' operator
- Search in a string
  - strpos($haystack, $needle) ;
- Replace data in string
  - str_replace($search , $replace , $in) ;
- Slice strings
  - substr($string , $start , [$length]) ;

```php
<?php

    /*
     * Example 4: Strings
     * file: 04-strings.php
     * note: this script has no output
     */

    $text = "this text " . "is concatenated";
    $number = "this results " + "in zero";

    $number = strpos($text, "is");  // 2

    $text = str_replace("dogs", "cats", "I like dogs");
                                        // I like cats

    $cats = substr($text, 2);       // like cats

    $like = substr($text, 2, 4);    // like

?>
```

Dealing with arrays

- Length
  - count($array) ;
- Keys
  - array_keys($array);
- Searching
  - in_array($needle , $haystack) ;
  - case sensitive
- Sorting
  - asort($array) ;
  - array_reverse ($array) ;
  - maintains index

```php
<?php

    /*
     * Example 5: Arrays
     * file: 05-arrays.php
     * note: this script has no output
     */

    $array = array("bob"   => 12,
                   "julie" => 15,
                   "sam"   => 9);

    $count = count($array);        // 3

    $keys = array_keys($array);
        // array("bob", "julie", "sam")

    $in_array = in_array(9, $array);      // true

    $in_keys   = in_array("sam", $keys);  // true
    $case_test = in_array("SAM", $keys);  // false

    asort($array);      // sam, bob, julie
    $reverse = array_reverse($array);
                        // julie, bob, sam

?>
```

Output

- Standard output
  - echo
  - print
- Debugging
  - print_r ($var) ;
    - recursive
    - useful with arrays
  - var_dump ($var);
    - shows “native” type

```
<pre>
<?php

    /*
     * Example 6: Output
     * file: 06-output.php
     */

    print "Hello,";     // no new line
    echo " World!";

    echo "\r\n";        // carriage return
    print "this is on a new line\r\n";

    $array = array("bob"   => 12,
                   "julie" => 15,
                   "sam"   => 9);

    print_r($array);
    asort($array);      // sam, bob, julie
    print_r($array);

    $number = 5.2;
    var_dump($number);

?>
```

Logic && flow control (I)

- Flow control
  - if ( ) {
    } else if ( ) {
    } else {
    }
  - switch ( ) {
      case : ;
                break;
      default: ;
    }

```
<pre>
<?php

    /*
     * Example 7: if
     * file: 07-if.php
     */

    if ( 0 ) {
        echo "0:  true" . "\r\n";
    } else {
        echo "0:  false" . "\r\n";
    }

    if ( 0.1 ) {
        echo "0.1:  true" . "\r\n";
    } else {
        echo "0.1:  false" . "\r\n";
    }

    if ( "text" ) {
        echo "text:  true" . "\r\n";
    } else {
        echo "text:  false" . "\r\n";
    }

    if ( "hello" + "world") {
        echo "hello + world:  true" . "\r\n";
    } else {
        echo "hello + world:  false" . "\r\n";
    }

?>
```

Logic && flow control (II)

- Comparison operators
  - <, >, <=, >=, ==, !=
  - uses type coercion
  - for type checking use
    - ===
    - !==
- Boolean operators
  - &&, ||
  - short-circuit logic

```
<pre>
<?php

    /*
     * Example 8: Logic
     * file: 08-logic.php
     */

    if ( "text" == "text" ) {
        echo "text == text:  true" . "\r\n";
    } else {
        echo "text == text:  false" . "\r\n";
    }

    if ( "text" == 0 ) {
        echo "text == 0:  true" . "\r\n";
    } else {
        echo "text == 0:  false" . "\r\n";
    }

    if ( "text" === 0 ) {
        echo "text === 0:  true" . "\r\n";
    } else {
        echo "text === 0:  false" . "\r\n";
    }

    $array = array("jenny" => 3, "cliff" => 1, "peter" => 2);
    print_r($array);

    if( false && asort($array) ) { }

    print_r($array);
?>
```

Loops (I)

- while (true) {
  // code
  }
- do {
  // code
  } while (true);
- for ($i = 0; $i < 5; $i++) {
  // code
  }

```
<pre>
<?php

    /*
     * Example 9: Loops
     * file: 09-loops.php
     */

    $sleepy = false;

    while($sleepy == false) {
        echo "Writing code." . "\r\n";
        $sleepy = true;
    }

    do {
        echo "Still coding." . "\r\n";
        $hours_required = 8;
    } while($sleepy == false);

    for($i = 1; $i <= $hours_required; $i++) {
        echo "Sleep (hour " . $i . ")" . "\r\n";
    }

?>
```

Loops (II)

- PHP has a built-in construct for array handling
- foreach ($array as $key => $value) { }
  - can be used with or without '$key'

```
<pre>
<?php

    /*
     * Example 10: Loops
     * file: 10-foreach.php
     */

    $stuarts = array("Chickens", "Crocodiles",
                "Fish", "Platypus", "Ants");

    foreach($stuarts as $name) {
        echo $name . " lay eggs" . "\r\n";
    }

    $fruits = array("oranges" => 3,
                    "apples"  => 6,
                    "bananas" => 4,
                    "mango"   => 2);

    echo "I have:" . "\r\n";
    foreach($fruits as $fruit => $num) {
        echo "   " . $num . " " . $fruit . "\r\n";
    }

?>
```

Functions and scope (I)

- Functions are declared using the 'function' keyword
- Have their own scope
- Values provided by parameters
- Values returned using the 'return' keyword
- Can be declared anywhere in file

```
<pre>
<?php

    /*
     * Example 11: Functions
     * file: 11-functions.php
     */

    $change = 0.07;
    $input = 2;

    echo next_number($input) . "\r\n";

    function next_number($number) {
        $change = 4;    // different '$change'
        $number = add_numbers($number, $change);
        return $number;
    }

    function add_numbers($num1, $num2) {
        return $num1 + $num2;
    }

    echo $change;

?>
```

Functions and scope (II)

- Can provide default parameter values
  - Alternative to Java's method "overloading"
- By default parameters are pass-by-value
  - can be pass-by-reference prefixing with '&'
  - could update the input and pass a different value back
    - e.g. success/failure

```
<pre>
<?php

    /*
     * Example 12: Advanced functions
     * file: 12-adv-func.php
     */

    $input = 2;

    echo next_number($input) . "\r\n";
    echo next_number($input) . "\r\n";

    function next_number($number) {
        $number = add_numbers($number);
        return $number;
    }

    function add_numbers($num1, $num2 = 3) {
        return $num1 + $num2;
    }


    echo next_number_ref($input) . "\r\n";
    echo next_number_ref($input) . "\r\n";

    function next_number_ref(&$number) {
        $number = add_numbers($number);
        return $number;
    }

?>
```

FILES

Files I/O (Bulk)

- file_get_contents ($filename);
  - reads whole file
  - returns a text string
  - works on ‘streams’
- file_put_contents ($filename, $content) ;
  - outputs whole file
  - overwrites existing file
  - creates file if required

```php
<?php

    /*
     * Example 13: Bulk File I/O
     * file: 13-file-bulk.php
     */

    // if we don't have a copy of the page, save one

    if(is_file("twitter.html") == false) {

        $data = file_get_contents("http://m.twitter.com/");
            // can read from remote sources

        file_put_contents("twitter.html", $data);

    } else {

        $data = file_get_contents("twitter.html");

    }

    $data = str_replace("Twitter", ”St Andrews", $data);
            // make some changes

    echo $data;

?>
```

Files I/O (line by line)

- For more granular control, three stages
  - open
    - fopen ($filename , $mode) ;
  - read and write
    - fgets ($file) ;
    - fputs ($file , $data) ;
  - close
    - fclose ($file) ;
- Manage position
  - end of file?
    - feof ($file) ;
  - get position, move
    - ftell ($file) ;
    - fseek ($file , $offset) ;

```php
<?php

    /*
     * Example 14: File I/O (Lines)
     * file: 14-file-line.php
     */

    $file = fopen("last-run.txt", "c+");

    if($file) {
        while(feof($file) == false) {    // check for eof
            $buffer = fgets($file);
            if($buffer) {            // ignore blank lines
                $last = $buffer;
            }
        }

        echo "Last run at " . $last;

        fputs($file, time() . "\r\n");
        fclose($file);
    }

?>
```

Saving structured data

- Data can be read from and written to .csv files
  - Excel compatible
- Get line from file
  - fgetcsv ($file) ;
    - returns array of fields
- Write line to CSV file
  - fputcsv ($file , $array) ;
    - writes array as a line
    - does **NOT** save keys

```
<pre>
<?php

    /*
     * Example 15: File I/O (CSV)
     * file: 15-file-csv.php
     */

    $file = fopen("data-file.csv", "w+");
    if($file) {

        fputcsv($file, array("Fruit",  "Number") );
        fputcsv($file, array("oranges", 3      ) );
        fputcsv($file, array("apples",  6      ) );
        fseek($file, 0);        // rewind

        $data = array();

        while(feof($file) == false) {
            $fields = fgetcsv($file);
            if($fields) {
                array_push($data, $fields);
            }                   // add fields to data
        }
        fclose($file);
    }

    print_r($data);

?>
```

Reusing code

- Code reuse is essential for good maintenance
- Including library code
  - require_once ($file) ;
  - include_once ($file) ;
- Including code in many places
  - require ($file) ;
  - include ($file) ;
- If an include is missing PHP will continue to run
  - (with a minor error)
  - not so for a missing require

```php
<?php

    /*
     * Example 16: Code reuse
     * file: 16-include.php
     */

    $numbers = array();

    for($i = 0; $i < 7; $i++) {
        include("16-include-show-winner.php");
    }

?>
```

```php
<?php

    require_once("16-include-get-winner.php");
    $ball = get_winner();

    $colors = array("white", "royalblue", "pink",
                    "lightgreen", "yellow");
    $color = $colors[floor($ball/10)];

?>
<div style="float: left; padding: 1em; margin: 1em; width: 1em;
            background-color: <?php echo $color; ?>;
            -moz-border-radius: 2em; border: 0.1em solid black;
            border-radius: 2em;"><?php echo $ball; ?></div>
```

```php
<?php

    function get_winner() {
        return rand(1, 49);
    }

?>
```

OBJECTS

Objects and Classes

- PHP was originally just an imperative language
- Also has a complete object model

- Define new classes with the ‘class’ keyword
- Access members with the ‘->’ token
- Static access with the ‘::’ token

```
<pre>
<?php

    /*
     * Example 17: Objects and Classes
     * file: 17-objects.php
     */

    class Dog {                                 // define class

        var $type;                              // add field

        function Dog($type = "Terrier") {       // constructor
            $this->type = $type;
        }

        function bark() {                       // method
            echo "Woof!" . "\r\n";
        }

    }

    Dog::bark();            // static access

    $bailey = new Dog();    // create new object

    $bailey->bark();        // call method
    echo $bailey->type;     // access field

?>
```

QUIZ

Quiz

What is the value of $var in the following code?

```
<?php

    $var = "to be," . "or not to be";

?>
```

A: “to be, or not to be”

B: 0

C: “to be,or not to be”

D: false

Quiz

What is the value of $var in the following code?

```php
<?php

    $var = "to be," . "or not to be";

?>
```

No spaces

A: “to be, or not to be”

B: 0

C: “to be,or not to be”

D: false

Quiz

Does the if statement execute?

```
<?php

    if ( "0" == 0 ) {
        echo "im in ur if, executing ur codes?";
    }
?>
```

Yes

No

Quiz

Does the if statement execute?

```php
<?php

    if ( "0" == 0 ) {
        echo "im in ur if, executing ur codes?";
    }
?>
```

Need to use ===
to check ‘types’

Yes

No

Quiz

What is the value of $var in the following code?

```php
<?php

    $var = "7" * 4 + 1;

?>
```

A: 1

B: 29

C: 0

D: 221

Quiz

What is the value of $var in the following code?

```
<?php

    $var = "7" * 4 + 1;

?>
```

String is coerced into a number

A: 1

B: 29

C: 0

D: 221

Quiz

What value is displayed by the following code?

```php
<?php

    echo substr("ABCD", 3, 1);

?>
```

A

B

C

D

Quiz

What value is displayed by the following code?

```php
<?php

    echo substr("ABCD", 3, 1);

?>
```

Strings, like arrays, are zero-indexed

A

B

C

D

Objectives

At the end of these lectures you should be able to:

- Recognise a basic PHP script
- Execute PHP scripts on your school host server
- Write a simple PHP program
- Identify some common caveats of PHP