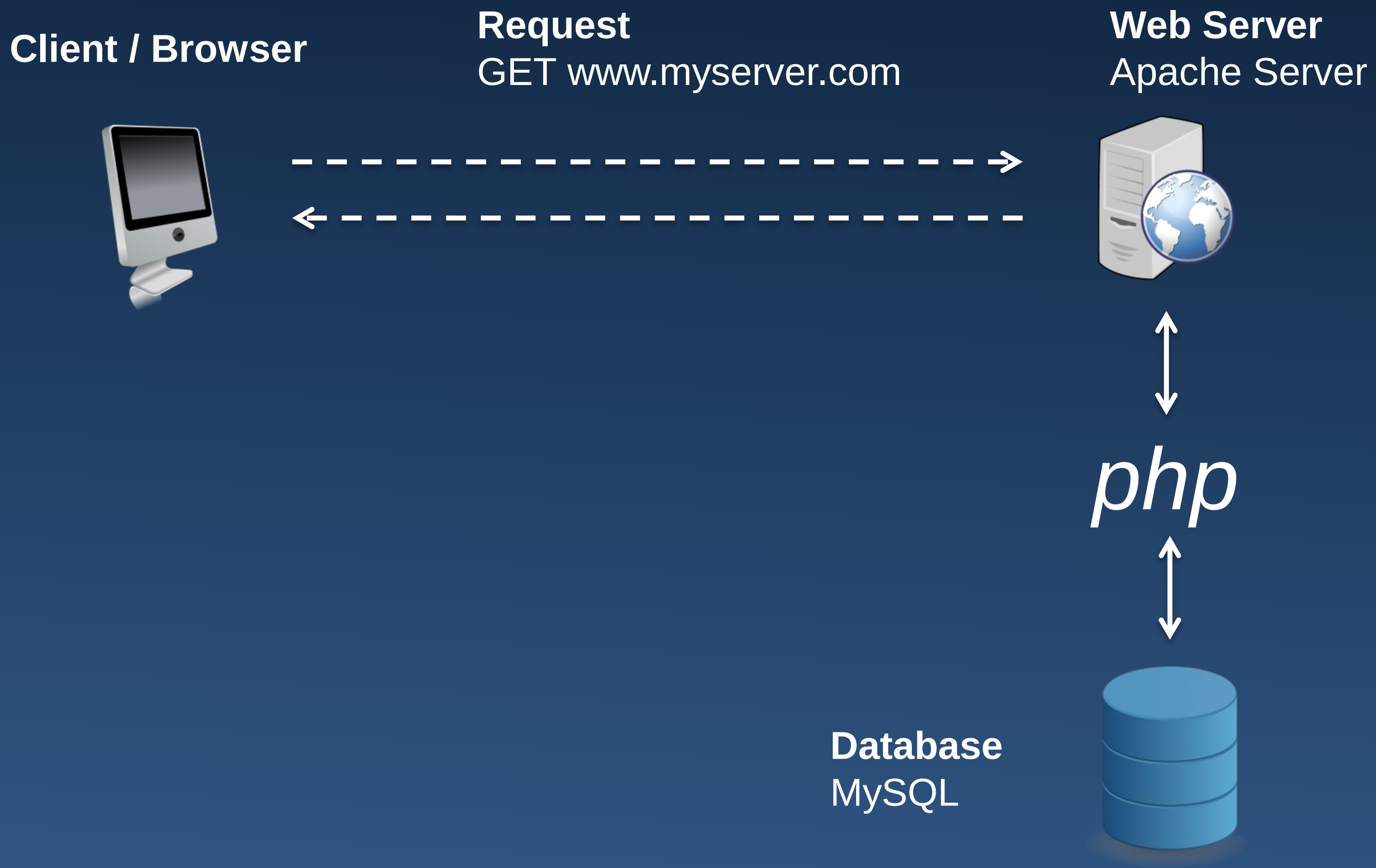
# Web Technology

Lecture 14: Introduction to PHP

computing@aberdeen

# Three-tiered Web Site



## What is PHP?

"a widely-used Open Source, general-purpose, server-side scripting language that is especially **suited for Web development** and can be embedded into HTML"

# A Short History of PHP

- Created 1994 by Rasmus Lerdorf to enhance his own personal Web page – hence its original name, **Personal Home Page Tools**
- Lerdorf freely distributed the program source and it became an Apache Software Foundation (Open source software collaborative) project
- Eventual name change to **PHP Hypertext Preprocessor**
- Now used on some ~ 80 million Web sites

# PHP Advantages

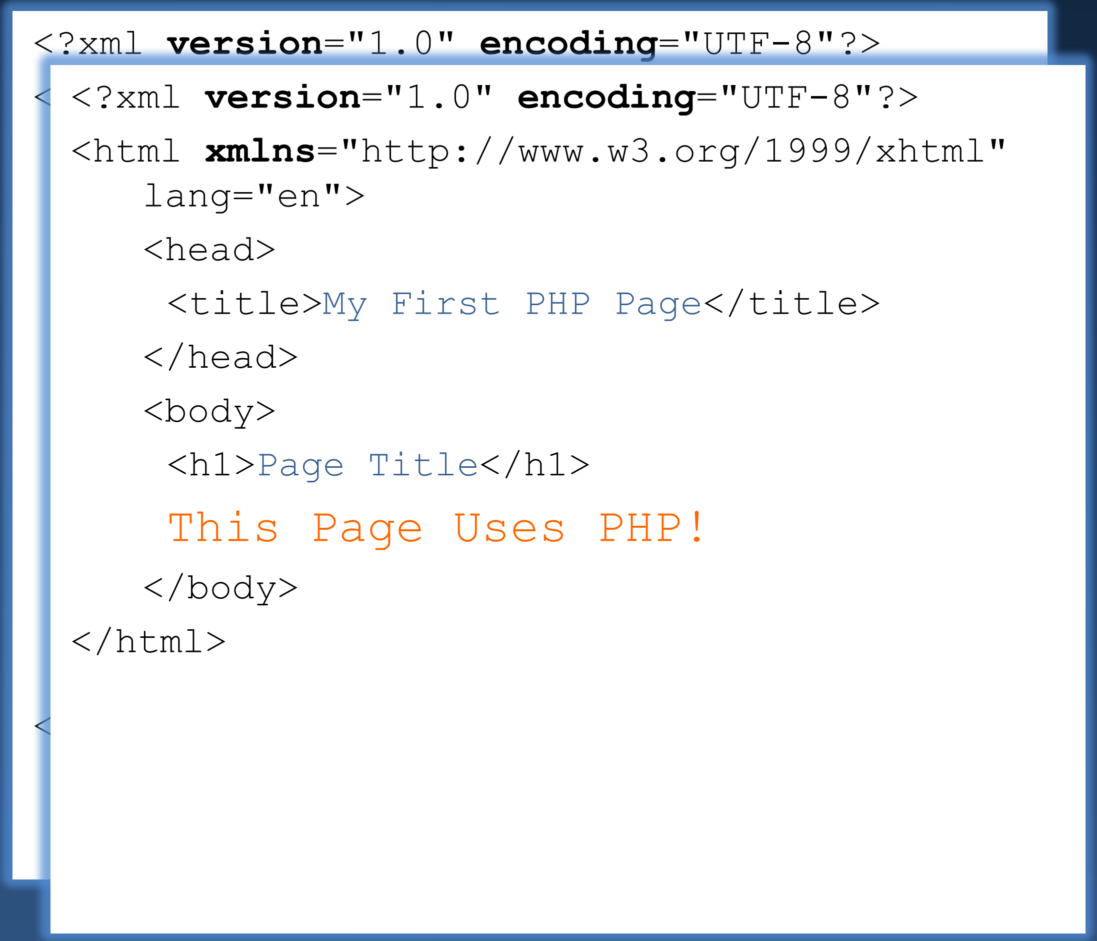
- **Dynamic** – can create HTML based on external changing information
- **Easy to use** – Standard syntax based on C
- **Open Source** – available for free download and is well supported by the open source software community
- **Multiple Platform** – PHP can be installed on UNIX and Windows based machines, and, therefore, projects can be ported from one environment to another with ease
- **Language Support For Databases** – Support for a wide variety of database systems e.g. MySQL/MariaDB

# PHP and its uses

- PHP can generate HTML programmatically
- PHP is useful when:
  - You want a webpage to change depending on input
  - You want to give different outputs to different users
- Using PHP means you don't need to write multiple webpages for different inputs
- Using PHP means you can use a general webpage with common information, but containing PHP that outputs HTML dependent on the given input

# Embedding PHP in HTML

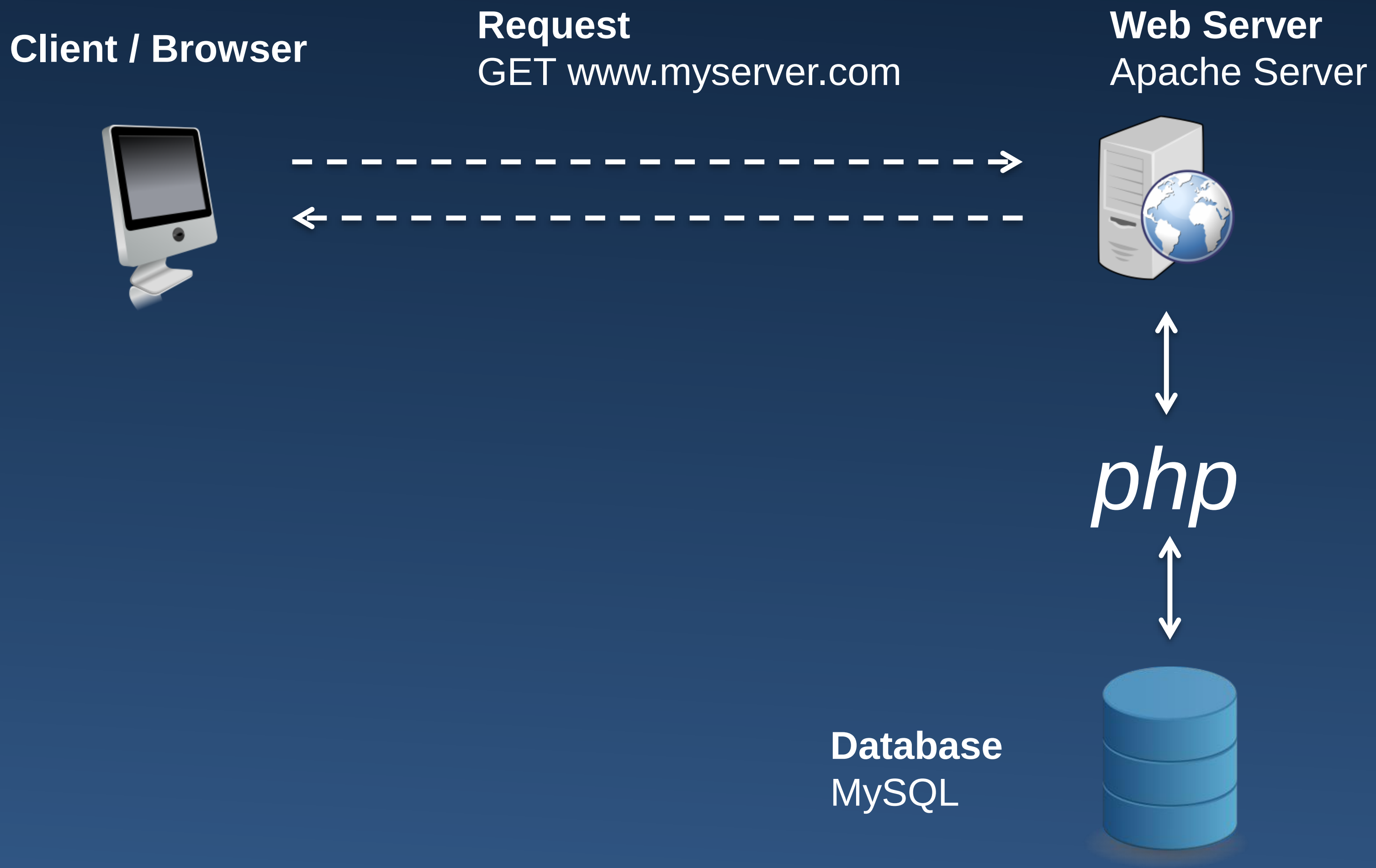
- PHP is in "blocks" mixed in with regular HTML
- A PHP block is marked by `<?php` to begin it, and `?>` to end it.
- The document must be saved with extension .php (not .html)



```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml"
    lang="en">
    <head>
     <title>My First PHP Page</title>
    </head>
    <body>
     <h1>Page Title</h1>
     This Page Uses PHP!
    </body>
</html>
```

# Three-tiered Web Site



# Basic PHP

- Each statement ends with a ;
- echo and print – create text (HTML) that is sent back to the client

```
echo ('hello');
print ('<h2> Contact us </h2>');
```

- include – inserts the contents of another file or web page. If including a PHP page that code is executed.

```
include ('header.php');
```

# Using variables in PHP

```
<?php
  $hello = 'Hello world!';
  echo($hello);
?>
```

Variables denoted by use of **$**

The variable $**hello** is assigned the String **"Hello world!"**

The $**hello** variable is then passed as a parameter to the PHP **echo** directive, which echoes the value held by the variable to the screen

## Quotes in PHP

- In PHP you can enclose strings in single or double quotes:

```
<?php $mystring = "cheese" ?>
<?php $mystring = 'cheese' ?>
```

- Single quotes are for plain strings, means no parsing for escape characters or variable names
- Single quotes are faster to execute, so avoid double quotes unless you have reasons to use them

## Using Variables

- Store a value in a variable, then use it later:

```
$name = 'David';
echo $name;
```

- Use them in a string:

```
print ('My name is ' . $name);
```

Or

```
print ("My name is $name");
```

- Use them to do maths:

```
$length = 6;
$width = 8;
$area = $length * $width;
```

# Arrays

- Arrays are created with the `array()` command

```
$people = array('Tom','Paul','John');
```

- Stored values are accessed by an index number, which is in [] after the variable name and starts from 0

`$people[0]` is Tom

`$people[2]` is John

- An array variable win an index can be used as a normal variable

```
$results[3] = 5+6;
```

## Arrays

- Arrays are not fixed size in PHP
  - Unlike C, C++, Java

```
$people = array('Tom','Paul','John');
```

- Want to add another person? It's easy

```
$people[] = 'Matt';
```

- Question: What's the index of Matt?

## Associative Arrays

- PHP allows arrays to be indexed by strings instead of numbers:

```
$carprices = array('Volvo' => 40000,
   'BMW' => 100000, 'Ford' => 15000);

echo $carprices['Volvo']; //outputs 40000
$carprices['Lada'] = 100; //adds a new car to the array
echo $carprices['Lada']; //outputs 100
```

# Operators

- Math operators

  `+ - * /`

- String operators

  `.` To join string together

- Comparison operators

  `> < <= >= == !=`

- Logical Operators

  `&&` (AND), `||` (OR)

## Control Strictures - Basics

- Control structures are used to determine which pieces of PHP actually get executed
- Conditional – code is executes only when a condition is true:

```
if (condition is true) { do this }
```

- Loop – execute the same section of the code repeatedly

```
while (condition is true) { do this }
for (specific number of times) { do this }
foreach (element in array) { do this }
```

## Control Structures – Conditions

- A condition is anything that is evaluated to true or false:

```
$name == 'John';
$age < 30;
(($name != 'Tom') && ($age > 30));
(($name == 'Paul') || ($age < 30));
```

## Control Structures – if statement

- Used when you want an action to depend on something:

```
if ($temp > 21) {
   print ('warm');
} elseif (($temp > 15) && ($temp < 21)) {
   print ('moderate');
} else {
   print('cold');
}
```

## Example if Statement

```
<html>
<body>
<?php
  $d=date('D');
  if ($d=='Fri') echo 'Have a nice weekend!';
?>

</body>
</html>
```

```
OUTPUT of Fri 18 March 2011:
Have a nice weekend!
```

## Control Structures - loops

- Loops are used to repeat actions (e.g. creating a table row by row)
  - while – repeat until a certain condition is met

```
$numLines = 0;
while (thereAreModeLinesToProcess()) {
  print ("a line!<br />\n");
  $numLines++;
}
```

  - For – repeat a certain number of times

```
for {$i =0; $i < 10; $i++) {
  print("line $1<br />\n");
}
```

  - Foreach – interates over an array

```
foreach ($array as $element) {
  print ("Element $element <br />\n");
}
```

## Example Foreach

```
<html>
<body>

<?php
  $x=array('one','two','three');
  foreach ($x as $value)
    {
    echo $value . '<br />';
    }
?>

</body>
</html>
```

```
OUTPUT:
one
two
three
```

## Foreach with Associative Arrays

```
foreach ($array as $key => $value) {
    echo "$key: $value";
 }

$carprices = array ('Volvo' => 40000, 'BMW' => 100000);
```

- Keys are the names of the indexes (e.g. Volvo)
- Values are the values associated with that key (e.g. 40,000)

## Foreach with Associative Arrays

```
<html>
<body>
<p>
<?php
  $carprices = array('Volvo' => 40000, 'BMW' => 100000, 'Ford' => 15000);

foreach ($carprices as $car => $price) {
         echo ("Car type: $car, Price: &pound;$price<br>\n");
}
?>
</p>
</body>
</html>
```

```
OUTPUT:
Car type: Volvo, Price: £40000
Car type: BMW, Price: £100000
Car type: Ford, Price: £15000
```

# Inline php

- Can also use alternate syntax, if your code is mainly HTML and only uses PHP for variables

- <?=  ?> is shorthand for echo

```
<?php $foods=array('ham','cheese','eggs'); ?>
<html>
<body>
<h1>Food list</h1>
<ul>
 <?php foreach ($foods as $f): ?>
   <li>Delicious <?=$f?></li>
 <?php endforeach; ?>
</ul>
</body>
</html>
```

## Functions

- A section of code can be used repeatedly by creating a function
- A function has a name
- A function is called by using its name
- The effect of a function may be controlled through parameters

```
displayDetails('Tom');
```

- Functions may return values

```
$area = pow($base,2);
```

## Creating a Function

- A Function is a combination of: the name, the parameters, the code and the returned value.

```
function someName($param1, $param2)
{
   code;
   code;
   return value;
}
```

# Example Function

```php
<?php
function writeName($fname)
{
    echo $fname . ' Refsnes.<br />';
}
  echo 'My name is ';
  writeName('Kai Jim');
  echo "My sister's name is ";
  writeName('Hege');
  echo "My
  writeNam
?>
```

```
OUTPUT:
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes.
My brother's name is Stale Refsnes.
```

# Built-in Functions

- Built-in functions are pre-made pieces of code that are executed by a call to the function.
- PHP has a **LARGE** set of functions
  - File System
  - Mail
  - Audio, video and image Manipulation
  - Date and Time
  - Compression
  - Credit card processing
  - Cryptography
  - Database

http://www.php.net/manual/en/

# Question?

After the following PHP code has executed, what is the value of numbers[3]?

```
$numbers = array(1,2,3,4,5);
for ($i = 1; $i < 5; $i++){
    $numbers[$i] = $numbers[$i-1] +
                   $numbers[$i];
}
```

a) 3

b) 4

c) 8

d) 10

# PHP: Here be dragons

- PHP is less strict than other languages
- If you have written nonsense, PHP will ignore the nonsense and carry on, it rarely crashes and stops the script

- Used improperly, opens your site up to a world of security problems
- Just because PHP allows you to do it, it doesn't make it good code

# Summary

**In this lecture**

- PHP Basics
- PHP Variables
- PHP Conditionals
- PHP Loops
- PHP if statements
- PHP Functions

**What next?**

- Advanced PHP functions
- Sessions
- PHP form handling