Presidency University, Bengaluru

CSE 256
INTERNET TECHNOLOGIES

TUTORIAL SHEET-11

PHP: Hypertext Preprocessor

PHP, or PHP: Hypertext Preprocessor, has become the most popular server-side scripting language for creating dynamic web pages

PHP is an open-source technology that's supported by a large community of users and developers.

PHP is *platform independent*—implementations exist for all major UNIX, Linux, Mac and Windows operating systems.

PHP also supports many databases, including MySQL.

Before Proceeding

To run a PHP script, PHP must first be installed on your system.

We assume that you've completed XAMPP installation .

This ensures that the Apache web server, MySQL DBMS and PHP are configured properly

Also we assume that you are at a comfortable position to appreciate HTML,CSS ,Javascript and HTML-form controls

PHP code is embedded directly into text-based documents, such as HTML, though these script segments are interpreted by the server before being delivered to the client.

PHP script file names end with .php

PHP, like JavaScript, is a dynamically typed language.

Unlike JavaScript it uses classes and functions in a way consistent with other object-oriented languages such as C++, C#, and Java, though with some minor exceptions.

The syntax for loops, conditionals, and assignment is identical to JavaScript, only differing when you get to functions, classes, and in how you define variables.

PHP code is inserted between the delimiters <?php and ?> and can be placed anywhere in HTML markup.

```
<html>
<?php
   $name = "Paul"; // declaration and initialization
?><!-- end PHP script -->
   <head>
      <meta charset = "utf-8">
      <title>Simple PHP document</title>
   </head>
   <body>
      <!-- print variable name's value -->
      <h1><?php print( "Welcome to PHP, $name!" ); ?></h1>

    </body>
 </html>
```

All variables are preceded by a $ and are created the first time they're encountered by the PHP interpreter.

PHP statements terminate with a semicolon (;).

a **single-line comment**, which begins with two slashes (//)

Multiline comments begin with delimiter **/*** on the first line of the comment and end with delimiter ***/** at the end of the last line of the comment.

function **print**

All operations are executed on the server before the HTML5 document is sent to the client.

You can see by viewing the source of a PHP document
that the code sent to the client does not contain any PHP code.

Another example

```
<?php
$user = "Randy";
?>
<!DOCTYPE html>
<html>
<body>
<h1>Welcome <?php echo $user; ?></h1>
<p>
The server time is
<?php
echo "<strong>";
echo date("H:i:s");
echo "</strong>";
?>
</p>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>
<h1>Welcome Randy</h1>
<p>
The server time is <strong>02:59:09</strong>
</p>
</body>
</html>
```

Variables, Data Types, and Constants

Variables in PHP are dynamically typed, which means that you as a programmer do not have to declare the data type of a variable.

Variables are also **loosely typed** in that a variable can be assigned different data types over time.

To declare a variable you must preface the variable name with the dollar ($) symbol.

name of a variable is case-sensitive

variable names can also contain the underscore character

Type	Description
`int, integer`	Whole numbers (i.e., numbers without a decimal point).
`float, double, real`	Real numbers (i.e., numbers containing a decimal point).
`string`	Text enclosed in either single (`''`) or double (`""`) quotes. [*Note:* Using double quotes allows PHP to recognize more escape sequences.]
`bool, boolean`	`true` or `false`.
`array`	Group of elements.
`object`	Group of associated data and methods.
`resource`	An external source—usually information from a database.
`NULL`	No value.

Writing to Output

To output something that will be seen by the browser, you can use the echo() function.

```
echo ("hello");
```

There is also an equivalent shortcut version that does not require the parentheses.

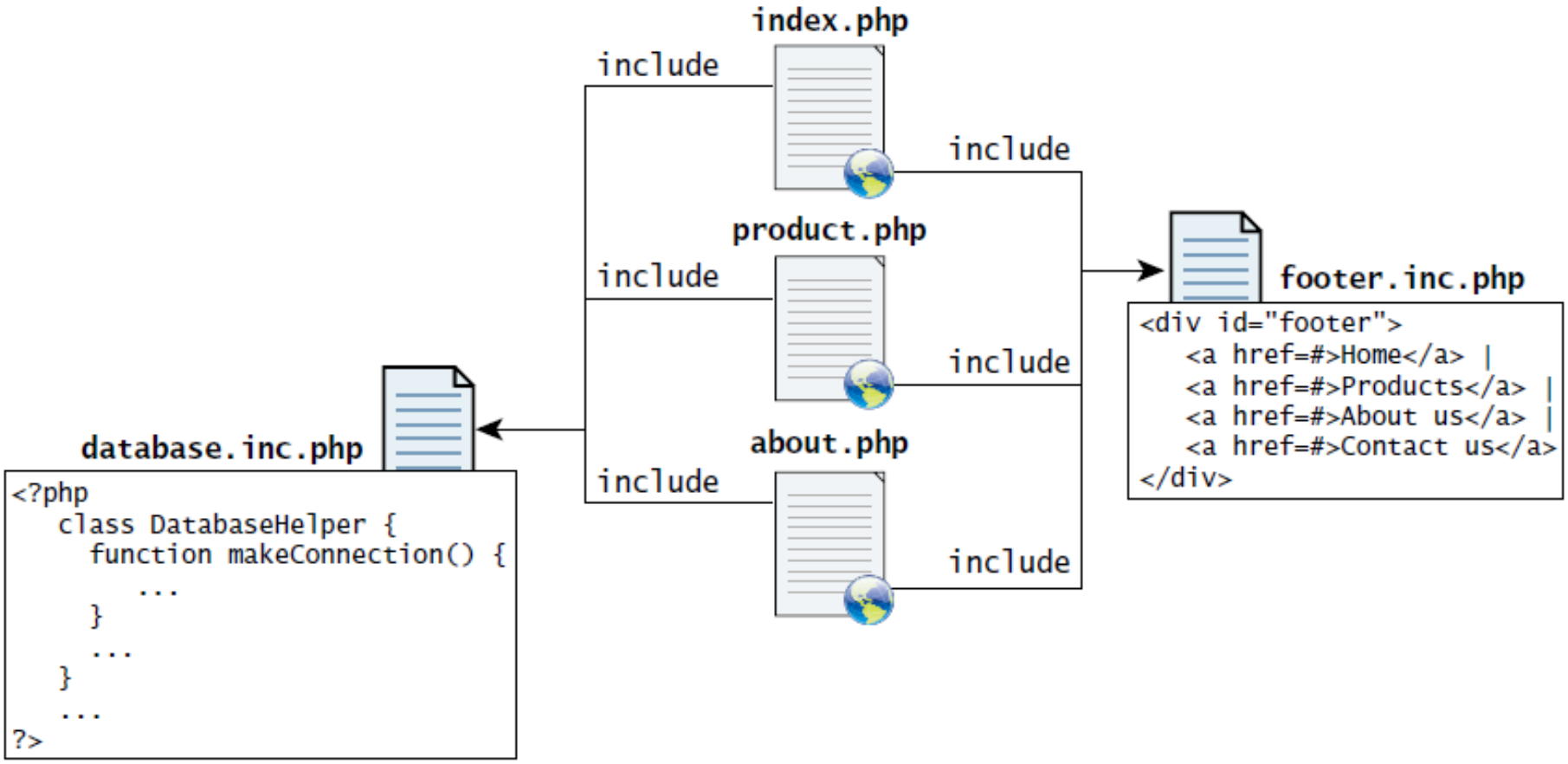
```
echo "hello";
```

Strings can easily be appended together using the concatenate operator, which is the period (.) symbol. Consider the following code:

```
$username = "Ricardo";
echo "Hello". $username;
```

Include Files

PHP does have one important facility to include or insert content from one file into another.



PHP provides four different statements for including files, as shown below.

```
include "somefile.php";
include_once "somefile.php";

require "somefile.php";
require_once "somefile.php";
```

The difference between include and require lies in what happens when the specified file cannot be included, With include, a warning is displayed and then execution continues. With require, an error is displayed and execution stops.

Functions

Having all your code in the main body of a script makes it hard to reuse, maintain, and understand.

As an alternative, PHP allows you to define functions.

Just like with JavaScript, a function in PHP contains a small bit of code that accomplishes one thing.

Functions can exist all on their own, and can then be called from anywhere that needs to make use of them, so long as they are in scope.

Later you will write functions inside of classes, which we will call methods.

To create a new function you must think of a name for it, and consider what it will do.

Functions can return values to the caller, or not return a value.

They can be set up to take or not take parameters.

```
function getNiceTime() {
    return date("H:i:s");
}
```

To call a function you must use its name with the () brackets

```
$output = getNiceTime();/ echo getNiceTime();
```

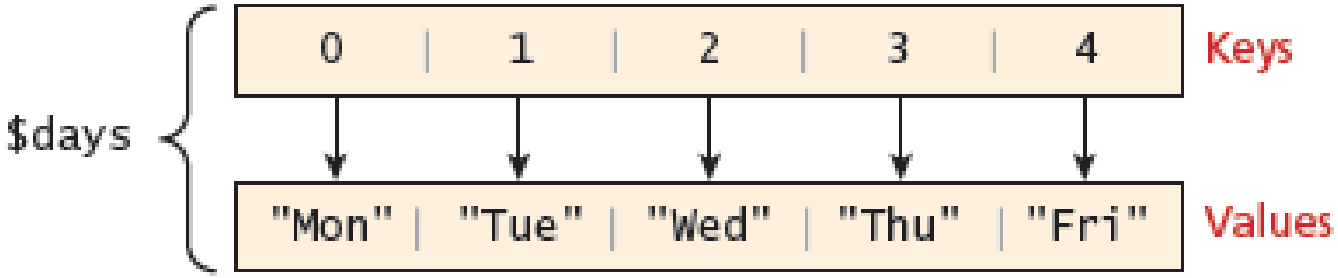
Initializing and Manipulating Arrays

PHP provides the capability to store data in arrays.

Arrays are divided into elements that behave as individual variables.

Array names, like other variables, begin with the $ symbol.

Individual array elements are accessed by following the array's variable name with an index enclosed in square brackets ([]).



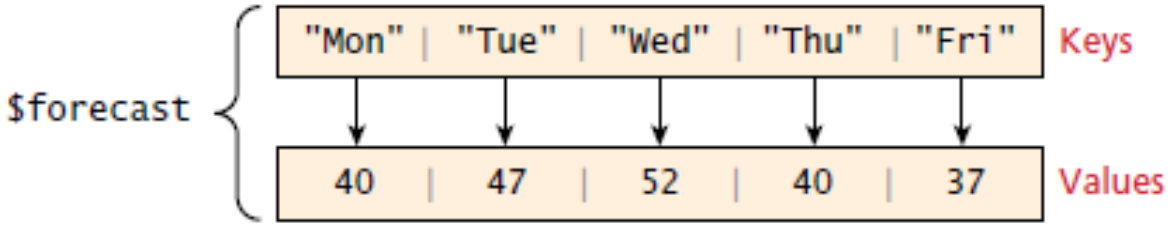
```
$days = array();
$days[0] = "Mon";
$days[1] = "Tue";
$days[2] = "Wed";

$days = array("Mon","Tue","Wed","Thu","Fri");

$days = ["Mon","Tue","Wed","Thu","Fri"]; // alternate syntax
```

In PHP, you are also able to explicitly define the keys in addition to the values.

```
$forecast = array("Mon" => 40, "Tue" => 47, "Wed" => 52, "Thu" => 40, "Fri" => 37);
```

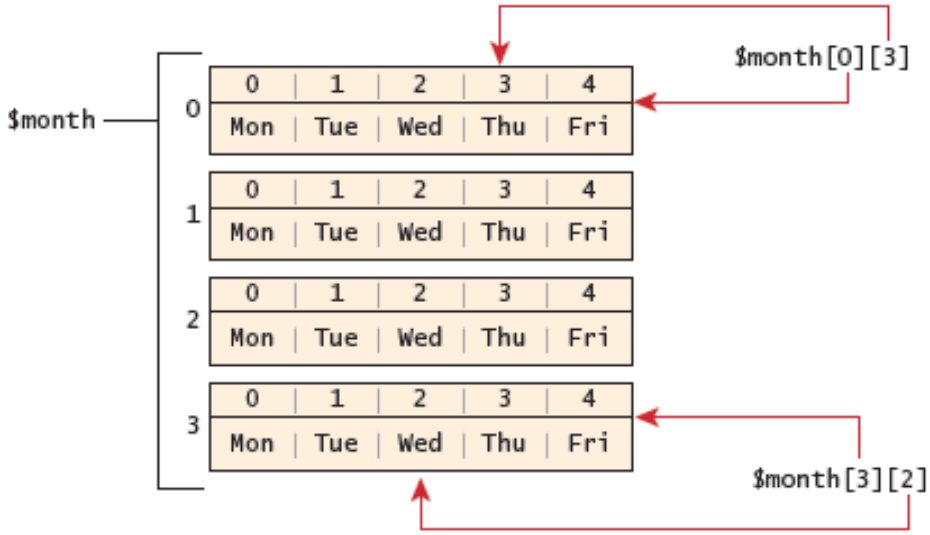
key

value

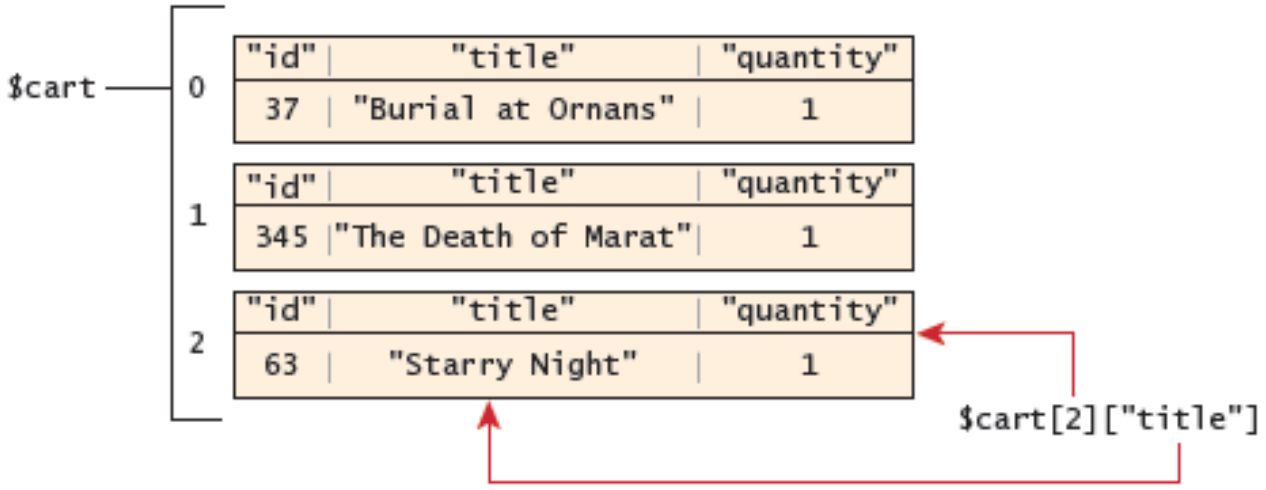



```
echo $forecast["Tue"];  // outputs 47
echo $forecast["Thu"];  // outputs 40
```

```
$month = array
  (
  array("Mon","Tue","Wed","Thu","Fri"),
  array("Mon","Tue","Wed","Thu","Fri"),
  array("Mon","Tue","Wed","Thu","Fri"),
  array("Mon","Tue","Wed","Thu","Fri")
  );

echo $month[0][3];   // outputs Thu
```



```php
$cart = array();
$cart[] = array("id" => 37, "title" => "Burial at Ornans",
                "quantity" => 1);
$cart[] = array("id" => 345, "title" => "The Death of Marat",
                "quantity" => 1);
$cart[] = array("id" => 63, "title" => "Starry Night", "quantity" => 1);

echo $cart[2]["title"];  // outputs Starry Night
```



Iterating through an Array

```php
// while loop
$i=0;
while ($i < count($days)) {
    echo $days[$i] . "<br>";
    $i++;
}


// do while loop
$i=0;
do {
    echo $days[$i] . "<br>";
    $i++;
} while ($i < count($days));


// for loop
for ($i=0; $i<count($days); $i++) {
    echo $days[$i] . "<br>";
}
```

```php
// foreach: iterating through the values
foreach ($forecast as $value) {
    echo $value . "<br>";
}

// foreach: iterating through the values AND the keys
foreach ($forecast as $key => $value) {
    echo "day" . $key . "=" . $value;
}
```