Web Technology

Lecture 17: Introduction to PHP
Common Functions

UNIVERSITY OF ABERDEEN

computing@aberdeen

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];
}
```

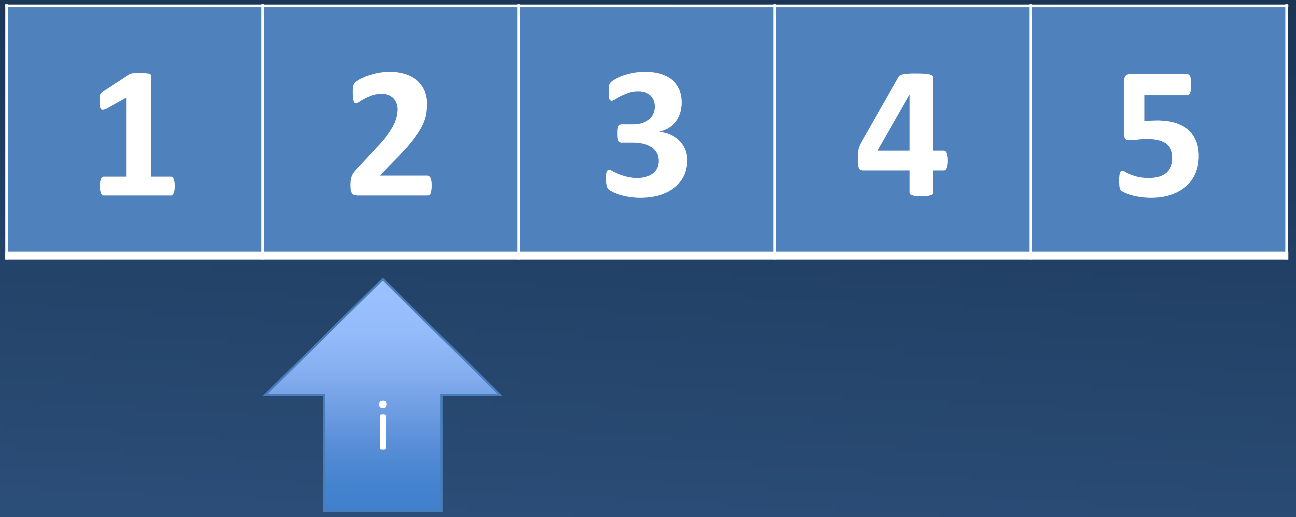
a) 3

b) 4

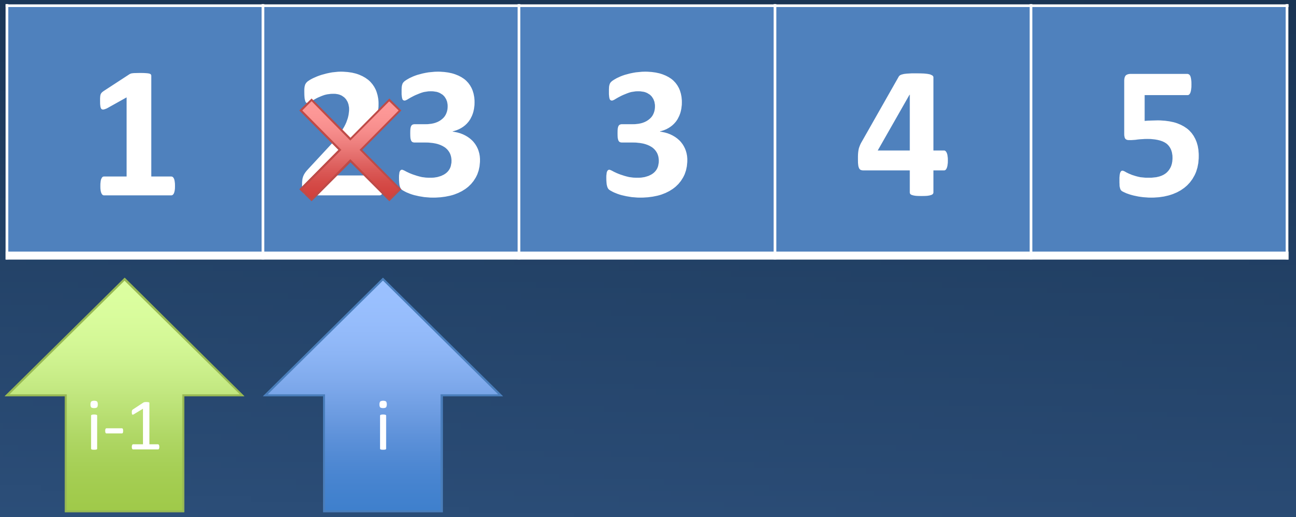
c) 8

d) 10

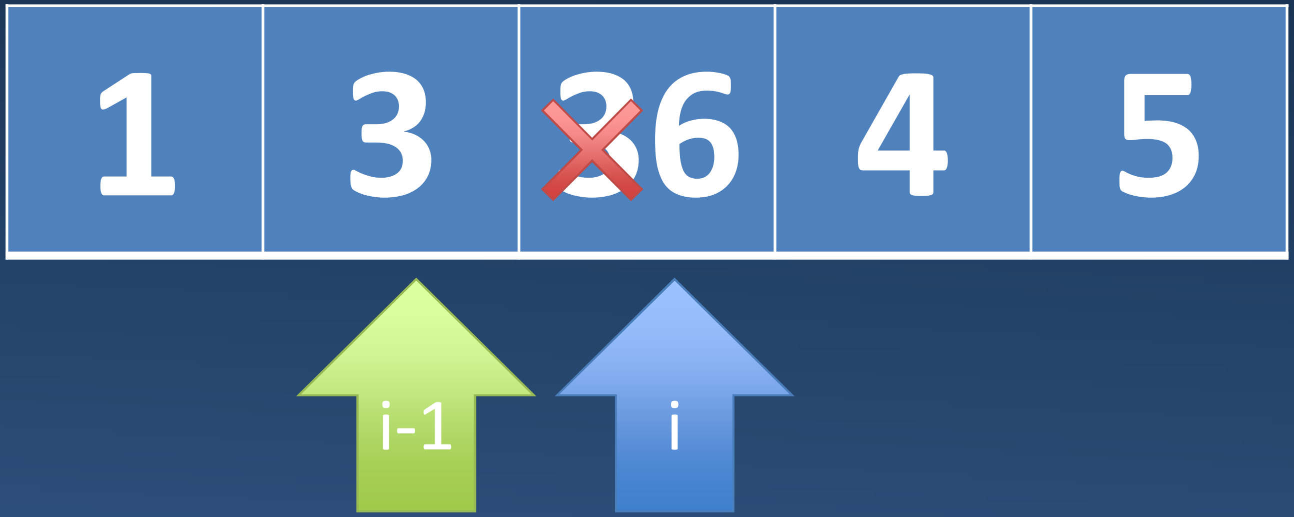
```
$numbers = array(1,2,3,4,5);
for ($i = 1; $i < 5; $i++){
    $numbers[$i] = $numbers[$i-1] +
                                   $numbers[$i];
}
```



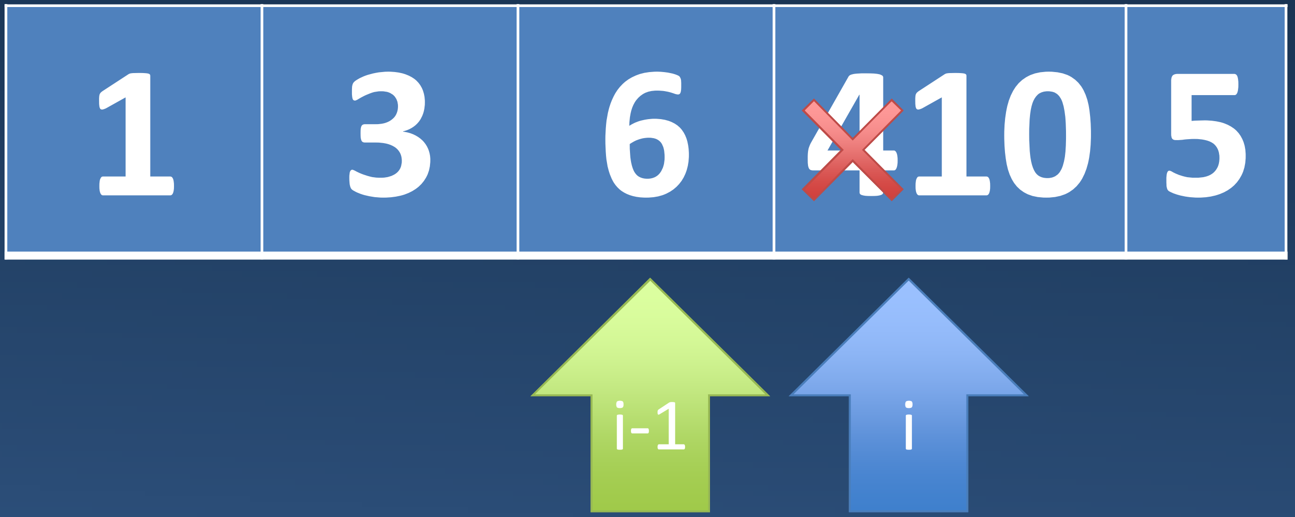
```
$numbers = array(1,2,3,4,5);
for ($i = 1; $i < 5; $i++){
    $numbers[$i] = $numbers[$i-1] +
                                    $numbers[$i];
}
```



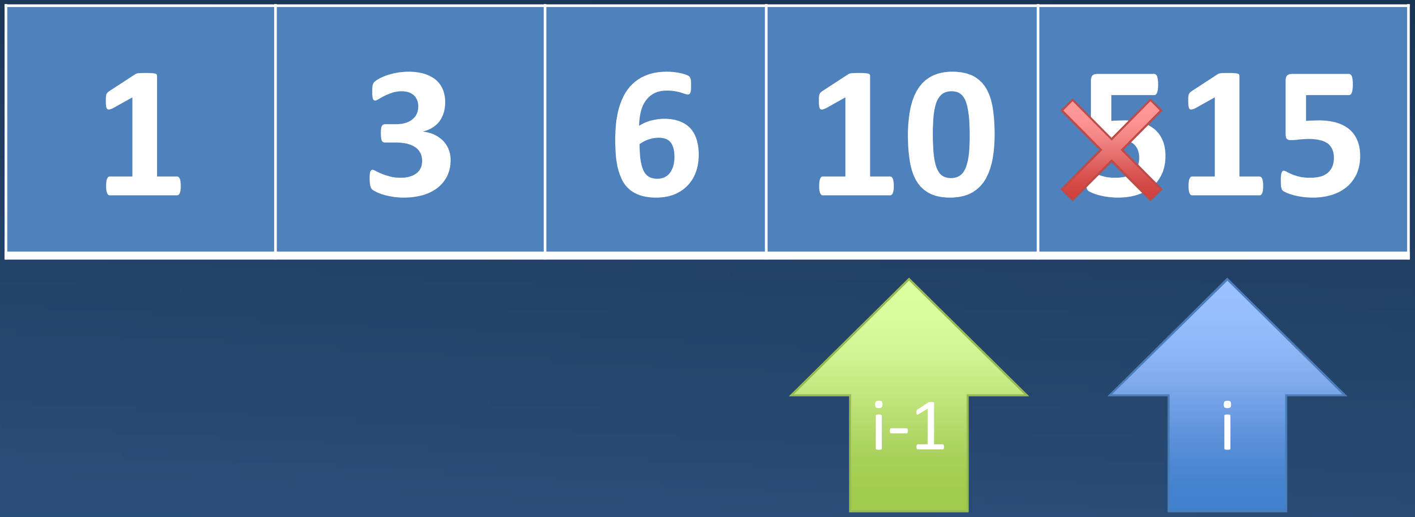
```
$numbers = array(1,2,3,4,5);
for ($i = 1; $i < 5; $i++){
    $numbers[$i] = $numbers[$i-1] +
                                    $numbers[$i];
}
```



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



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



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

1	3	6	10	15

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

This lecture

- Common PHP Tasks
  - Handling dates and times
  - Service side includes
  - PHP sessions
  - Email
  - Form validation

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/

The PHP `date()` Function

- The PHP `date()` function formats a timestamp to a more readable date and time.
- A timestamp is a sequence of characters, denoting the date and/or time at which a certain event occurred.

```
date(format,timestamp)
```

PHP `date()` - Format the Date

- The required *format* parameter in the date() function specifies how to format the date/time. Here are some characters that can be used:
  - d - Represents the day of the month (01 to 31)
  - m - Represents a month (01 to 12)
  - Y - Represents a year (in four digits)
- Other characters, like"/", ".", or "-" can also be inserted between the letters to add additional formatting:

```
<?php
   echo date("Y/m/d") . "<br />";
   echo date("Y.m.d") . "<br />";
   echo date("Y-m-d")
?>
```

- The output of the code above could be something like this:

```
2011/03/11
2011.03.11
2011-03-11
```

PHP `date()` - Adding a Timestamp

- The optional *timestamp* parameter in the date() function specifies a timestamp. If you do not specify a timestamp, the current date and time will be used.
- The `mktime()` function returns the Unix timestamp for a date.
- The Unix timestamp contains the number of seconds between the Unix Epoch (January 1 1970 00:00:00 GMT) and the time specified.

```
mktime(hour, minute, second, month, day, year, is_dst)
```

- To go one day in the future we simply add one to the day argument of `mktime()`:

```
$tomorrow = mktime(0, 0, 0, date("m"), date("d") +1,
                                                 date("Y"));

echo "Tomorrow is ".date("Y/m/d", $tomorrow);
```

Server Side Includes (SSI)

- You can insert the content of one PHP file into another PHP file before the server executes it, with the `include()` or `require()` function.
- The two functions are identical in every way, except how they handle errors:
  - `include()` generates a warning, but the script will continue execution
  - `require()` generates a fatal error, and the script will stop
- These two functions are used to create functions, headers, footers, or elements that will be reused on multiple pages.

PHP `include()` Function

- The `include()` function takes all the content in a specified file and includes it in the current file.
- If an error occurs, the `include()` function generates a warning, but the script will continue execution.
- Assume that you have a standard header file, called "header.php". To include the header file in a page, use the `include()` function:

```
<html>
<body>

 <?php include("header.php"); ?>
 <h1>Welcome to my home page!</h1>
 <p>Some text.</p>

</body>
</html>
```

PHP `require()` Function

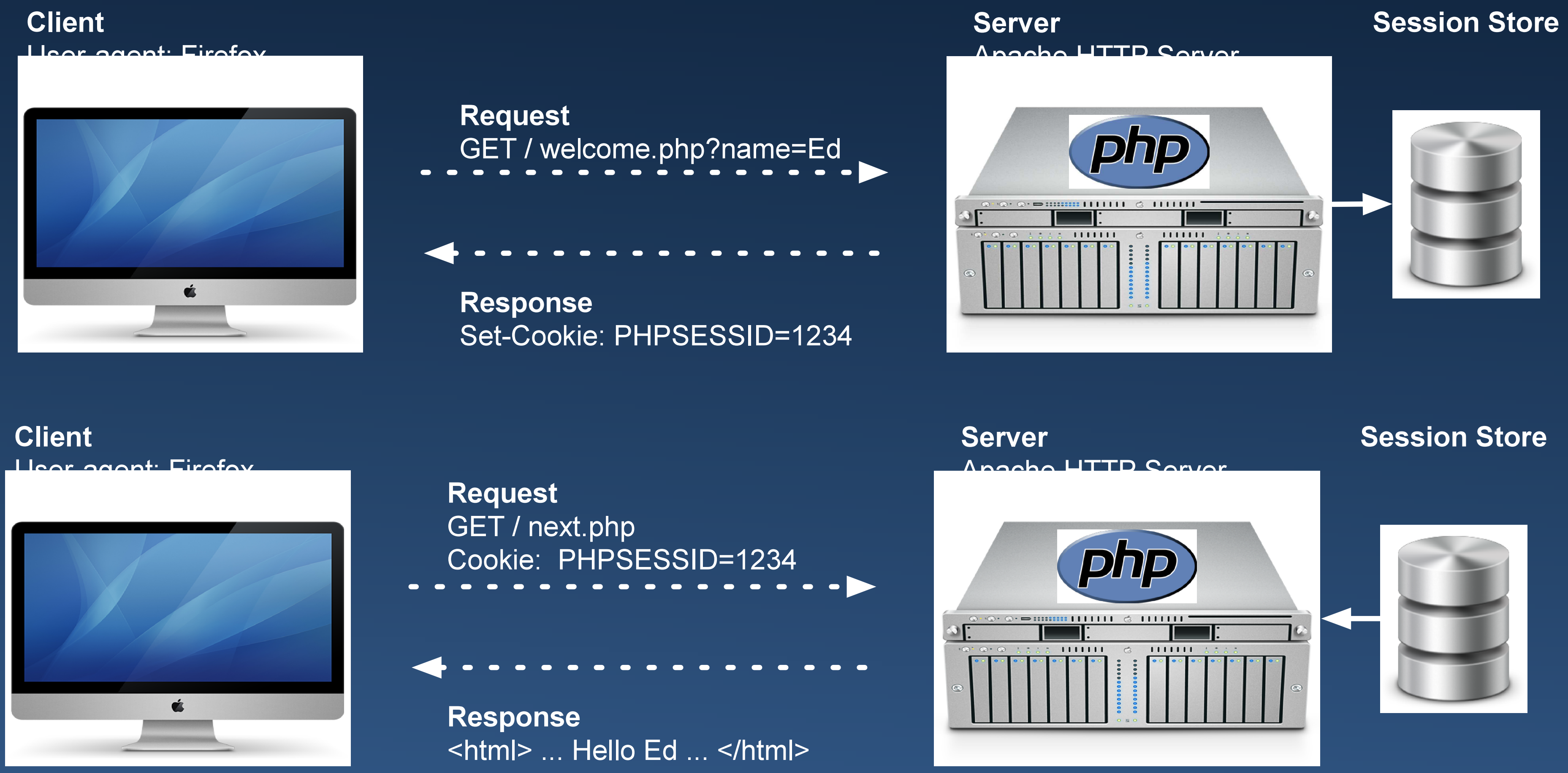
- The `require()` function is identical to `include()`, except that it handles errors differently.
- If an error occurs, the `include()` function generates a warning, but the script will continue execution. The `require()` generates a fatal error, and the script will stop.

```
<?php
  require("wrongFile.php");
  echo "Hello World!"; //not executed
?>
```

PHP Sessions

- When you are working with an application, you open it, do some changes and then you close it (Session).
- An application need to remember certain information between a request and another.
- A PHP session allows to store persistent data between different requests.
- Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID.

PHP Sessions



Starting a PHP Session

- Before you can store user information in your PHP session, you must first start up the session.
- The `session_start()` function must appear BEFORE the `<html>` tag:

```
<?php session_start(); ?>

<html>
<body>

</body>
</html>
```

Storing a Session Variable

- The correct way to store session variables is to use the PHP `$_SESSION` variable:

```
<?php
 session_start();
 $_SESSION['name']=$_GET["name"];
?>
<html>
<body>
</body>
</html>
```

Retrieving a Session Variable

- The session data is retrieved by using the PHP `$_SESSION` variable:

```
<html>
<body>
<?php
  echo "Hello ". $_SESSION['name'];
?>
</body>
</html>
```

Destroying a Session

- If you wish to delete some session data, you can use the `unset()` or the `session_destroy()` function.
- The `unset()` function is used to free the specified session variable:

```
<?php
  unset($_SESSION['name']);
?>
```

- You can also completely destroy the session by calling the `session_destroy()` function:

```
<?php
  session_destroy();
?>
```

The PHP mail() Function

- This function used to send emails from inside a script.

```
mail(to, subject, message, headers, parameters);
```

- `to` (Required) specifies the receiver.
- `subject` (Required) specifies the subject of the email.
- `message` (Required) defines the message to be sent.
- `headers` (Optional) specifies additional headers, like From, Cc, and Bcc.
- `parameters` (Optional) Specifies an additional parameters.

PHP Simple E-Mail

```
<?php
 $to = "someone@example.com";
 $subject = "Test mail";
 $message = "Hello! This is a simple email message.";
 $from = "someonelse@example.com";
 $headers = "From: $from";
 if (mail($to, $subject, $message, $headers)){
    echo "Mail Sent.";
 } else {
   echo "Failed to send mail.";
 }
?>
```

Forms and PHP

- (X)HTML form generates a request to the server.
- Form information is attached either to the requests URI (GET) or body (POST)
- Simple form:

```
<form action="contactus.php" method="GET">
 <p>
 Email Address <input type="text" name="email" /> <br/>
 Message <input type="text" name="message" />
 <input type="submit" value="Send" />
</p>
</form>
```

- When clicking the submit-button, the user's name and message attached to the URI:

```
/contactus.php?email=john@email.com&message=...
```

Handling Forms in PHP

- It is very easy to handle the sent (**GET** or **POST**) information in PHP.
- Information is stored in associative array called **GET**, **POST** and **REQUEST**
- Array keys are the form elements names and corresponding values are the users input
- Getting input in PHP:

```
$email = $_GET['email'];
```
(If sent via GET)

```
$email = $_POST['email'];
```
(If sent via POST)

```
$email = $_REQUEST['email'];
```

Example – Guess a Number

```
<h1>Guess Number Game 2</h1>
<form action="guess.php" method="GET">
<p>Give a number between 1-10:
 <input type="text" name="number" >
 <input type="submit" value="Guess" />
</p>
</form>

<?php
define("SECRETNUMBER", 7);
$number = $_GET['number'];
if($number < SECRETNUMBER)
{
  print "The secret number is higher.";
}
else if ($number > SECRETNUMBER)
{
  print "The secret number is lower.";
}
else if ($number == SECRETNUMBER)
{
  print "That's right!";
}
?>
```

Question

Which of the following checks the user has entered their name is a form submitted via POST or GET? (`trim()` removes whitespace from beginning and end of a string)

```
1. if (trim($_GET['name']) == ''){ … }

2. if (trim($_POST['name']) == ''){ … }

3. if (trim($_SESSION['name']) == ''){ … }

4. if (trim($_REQUEST['name']) == ''){ … }
```

References

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