Continue to Advance PHP

6.4 PHP Check End-Of-File - feof()

The feof() function checks if the "end-of-file" (EOF) has been reached.

The feof() function is useful for looping through data of unknown length.

The example below reads the "webdictionary.txt" file line by line, until end-of-file is reached:

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
</body>
</html>
```

Output:

AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets

HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language

6.5 PHP Read Single Character - fgetc()

The fgetc() function is used to read a single character from a file. The example below reads the "webdictionary.txt" file character by character, until end-of-file is reached: Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);   }
fclose($myfile);
?>
</body>
</html>
```

Output:

AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets HTML = Hyper Text Markup Language PHP = PHP Hypertext Preprocessor SQL = Structured Query Language SVG = Scalable Vector Graphics XML = EXtensible Markup Language

6.6 PHP Create File - fopen()

The fopen() function is also used to create a file. Maybe a little confusing, but in PHP, a file is created using the same function used to open files. If you use fopen() on a file that does not exist, it will create it, given that the file is opened for writing (w) or appending (a). The example below creates a new file called "testfile.txt". The file will be created in the same directory where the PHP code resides Example

```
$myfile = fopen("testfile.txt", "w")
```

6.7 PHP Write to File - fwrite()

The fwrite() function is used to write to a file. The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.

The example below writes a couple of names into a new file called "newfile.txt": Example

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "John Doe\n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

Notice that we wrote to the file "newfile.txt" twice. Each time we wrote to the file we sent the string $txt that first contained "John Doe" and second contained "Jane Doe". After we finished writing, we closed the file using the fclose() function. If we open the "newfile.txt" file it would look like this:

John Doe
Jane Doe

6.8 PHP Overwriting

Now that "newfile.txt" contains some data we can show what happens when we open an existing file for writing. All the existing data will be ERASED and we start with an empty file.

In the example below we open our existing file "newfile.txt", and write some new data into it:

Example

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = "Mickey Mouse\n";
fwrite($myfile, $txt);
$txt = "Minnie Mouse\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

Output:If we now open the "newfile.txt" file, both John and Jane have vanished, and only the data we just wrote is present:

Mickey Mouse
Minnie Mouse

7.1 What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

7.1 Create Cookies With PHP

A cookie is created with the setcookie() function. Syntax

setcookie(*name, value, expire, path, domain, secure, httponly*);

Only the *name* parameter is required. All other parameters are optional.

7.3 PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer). We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset() function to find out if the cookie is set:

Example:

```
<!DOCTYPE html>
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
<p><strong>Note:</strong> You might have to reload the page to see the value of the cookie.</p>
</body>
</html>
```

Output: **Cookie 'user' is set!**
Value is: John Doe

Note: You might have to reload the page to see the value of the cookie.

Note: The setcookie() function must appear BEFORE the <html> tag.

Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

7.4 Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the setcookie() function:

```
<!DOCTYPE html>
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name]; }
?>
<p><strong>Note:</strong> You might have to reload the page to see the new value of the cookie.</p>
</body>
</html>
```

Output: **Cookie 'user' is set!**
Value is: Alex Porter

Note: You might have to reload the page to see the new value of the cookie.

7.5 Delete a Cookie

To delete a cookie, use the setcookie() function with an expiration date in the past Example:

```
<!DOCTYPE html>
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>
<?php
echo "Cookie 'user' is deleted.";
?>
</body>
</html>
```

Output: Cookie 'user' is deleted.

7.6 Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the setcookie() function, then count the $_COOKIE array variable Example:

```
<!DOCTYPE html>
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>
<?php
if(count($_COOKIE) > 0) {
   echo "Cookies are enabled.";
} else {
   echo "Cookies are disabled.";
}
?>
</body>
</html>
```

Output: Cookies are enabled.

8.1 PHP 5 Sessions

A session is a way to store information (in variables) to be used across multiple pages.

Unlike a cookie, the information is not stored on the users computer.

8.2 What is a PHP Session?

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser. So; Session variables hold information about one single user, and are available to all pages in one application.

Tip: If you need a permanent storage, you may want to store the data in a database.

8.3 Start a PHP Session

A session is started with the session_start() function. Session variables are set with the PHP global variable: $_SESSION. Now, let's create a new page called "demo_session1.php". In this page, we start a new PcfHP session and set some session variables Example:

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>
</body>
</html>
```

Output: Session variables are set.

Note: The session_start() function must be the very first thing in your document. Before any HTML tags.

8.4 Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php"). Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

Also notice that all session variable values are stored in the global $_SESSION variable:

Example:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
</body>
</html>
```

Output:

Favorite color is green.
Favorite animal is cat.

Another way to show all the session variable values for a user session is to run the following code:

Example:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
print_r($_SESSION);
?>
</body>
</html>
```

Output: **Array ([favcolor] => green [favanimal] => cat)**