Working with Ajax and PHP

Starting with PHP

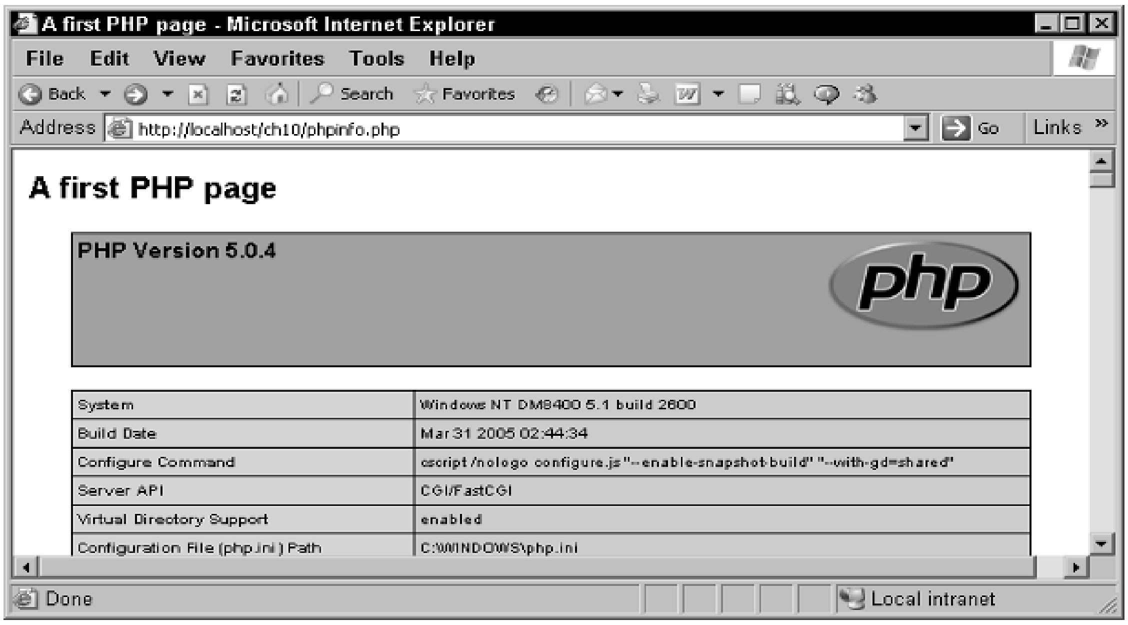
- PHP scripts are stored in files with the extension .php (like checkprice.php) inside <? and ?> as follows:

```
<?php
.
. Your PHP goes here....
.
?>
```

- PHP is that you can intersperse HTML and PHP at will.
- A PHP-enabled server will execute the PHP code inside the <?...?> sections, and just send the HTML on as usual.
- Here's an example that runs the built-in PHP function phpinfo, which creates an HTML table that tells you about your PHP installation.
- Each PHP statement is ended with a semicolon (;).

```
<html>
<head>
<title>
A first PHP page
</title>
</head>
<body>
<h1>
A first PHP page
</h1>
<?php
phpinfo();
?>
</body>
</html>
```

The phpinfo function displays the table you see in the figure, and the header, A first PHP page, comes from the HTML you've placed in phpinfo.php.


A first PHP page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/ch10/phpinfo.php
Go Links
Done
Local intranet

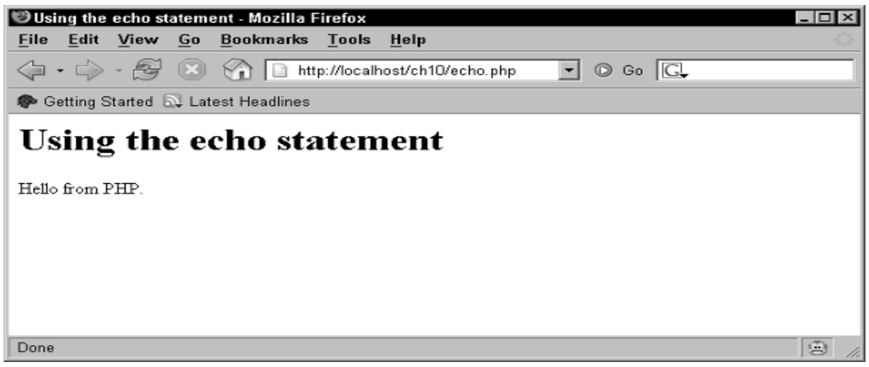
A first PHP page

PHP Version 5.0.4

System	Windows NT DM8400 5.1 build 2600
Build Date	Mar 31 2005 02:44:34
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--with-gd=shared"
Server API	CGI/FastCGI
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS\php.ini

- **Using the echo statement:-**
  The echo statement is used to display a message in php.

```
<html>
<head>
<title>
Using the echo statement
</title>
</head>
<body>
<h1>
Using the echo statement
</h1>
<?php
echo “Hello from PHP.”;
?>
</body>
</html>
```


Using the echo statement - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/ch10/echo.php
Go
Getting Started
Latest Headlines
Done

Using the echo statement

Hello from PHP.

- **Comments in php:**

  There are two types of comments in php:
  1] Multiline comment
  2] Single line comment

Multiline comment

```
<?php
/* Start by displaying a
message to the user */
        echo "Hello from PHP.";
?>
```

Singleline comment

The single line comments can be given by using // or #.

```
<?
// Start by displaying a
# message to the user
        echo "Hello from PHP.";
?>
```

- **Getting a Handle on Variables**
  - The variables in php can be created by using $ sign.
  - The syntax to create a variable is:
    $varname;
  - The variable name starts with $ sign.
    For eg: $peaches=1;

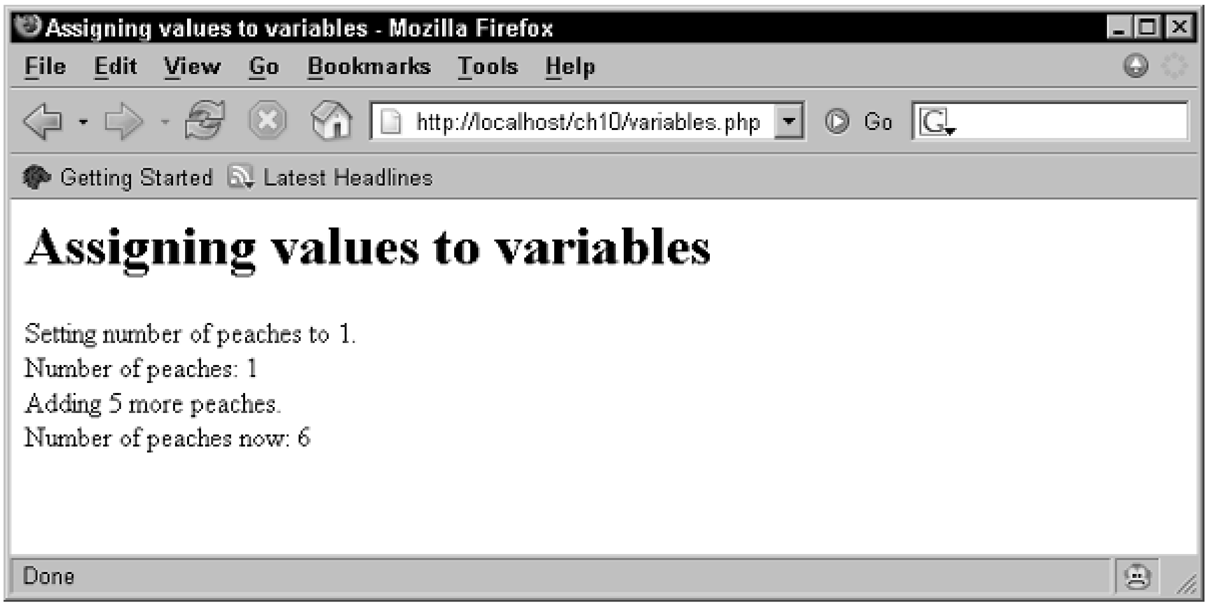
```
echo "Number of peaches: ", $peaches, "<br>";
```

There are two things to note here:

- You can pass multiple items to the echo statement if you separate the items with commas.
- You're sending HTML back to the browser, so to skip to the next line, you use HTML like <br>.

- PHP example, variables.php, that assigns a value to $peaches and then changes the value in that variable by adding 5 to it:

```
<html>
<head>
<title>
Assigning values to variables
</title>
</head>
<body>
<h1>
Assigning values to variables
</h1>
<?php
echo “Setting number of peaches to 1.<br>”;
$peaches = 1;
echo “Number of peaches: “, $peaches, “<br>”;
echo “Adding 5 more peaches.<br>”;
$peaches = $peaches + 5;
echo “Number of peaches now: “, $peaches, “<br>”;
?>
</body>
</html>
```


Assigning values to variables - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/ch10/variables.php
Go
Getting Started
Latest Headlines
Done

Assigning values to variables

```
Setting number of peaches to 1.
Number of peaches: 1
Adding 5 more peaches.
Number of peaches now: 6
```

- **String Variable**

  $string = “Hello from PHP.”;

  In JavaScript, you join strings with the + operator, but in PHP, you use the **dot (.)** operator instead:

  $string = “Hello “ . “from “ . “PHP.”;

PHP string built in string functions:

1. trim: Trims spaces from the beginning and end of a string
2. substr: Extracts substrings from a string
3. strpos: Finds the location of a substring in a string
4. ucfirst: Capitalizes the first character of a string
5. substr_replace: Replaces text in a string
6. strtoupper: Converts a whole string to uppercase

Example:

```
<?php
echo trim(“ No problem.”), “<br>”;
echo substr(“No problem.”, 3, 7), “<br>”;
echo “‘problem’ starts at position “, strpos(“No problem.”, “problem”),
“<br>”;
echo ucfirst(“no problem.”), “<br>”;
echo “‘No problem.’ is “, strlen(“No problem.”), “ characters long.<br>”;
echo substr_replace(“No problem.”, “problems.”, 3, 8), “<br>”;
echo strtoupper(“No problem.”), “<br>”;
?>
```

Output:

```
No problem.
problem
‘problem’ starts at position 3
No problem.
```

```
'No problem.' is 11 characters long.
No problems.
ABC
NO PROBLEM.
```

Arrays in PHP

- An array is a special variable, which can hold more than one value at a time.
- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

  ```
  $cars1="Volvo";
  $cars2="BMW";
  $cars3="Toyota";
  ```
- An array can hold many values under a single name, and you can access the values by referring to an index number.
- In PHP, the array() function is used to create an array:

  ```
  array();
  ```
- The index can be assigned automatically (index always starts at 0):

  ```
  $cars=array("Volvo","BMW","Toyota");
  ```

  or

  The index can be assigned manually:

  ```
  $cars[0]="Volvo";
  $cars[1]="BMW";
  $cars[2]="Toyota";
  ```
- For eg:

  ```
  <?php
  $cars=array("Volvo","BMW","Toyota");
  echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
  ?>
  ```
- **Get The Length of an Array - The count() Function**

  ```
  <?php
  $cars=array("Volvo","BMW","Toyota");
  ```

```
echo count($cars);
?>
```

- In PHP, you can also refer to items in an array with a text index if you prefer, like this:

```
$data["temperature"] = 81;
echo $data["temperature"]; //displays 81
```

Handling Your Data with Operators

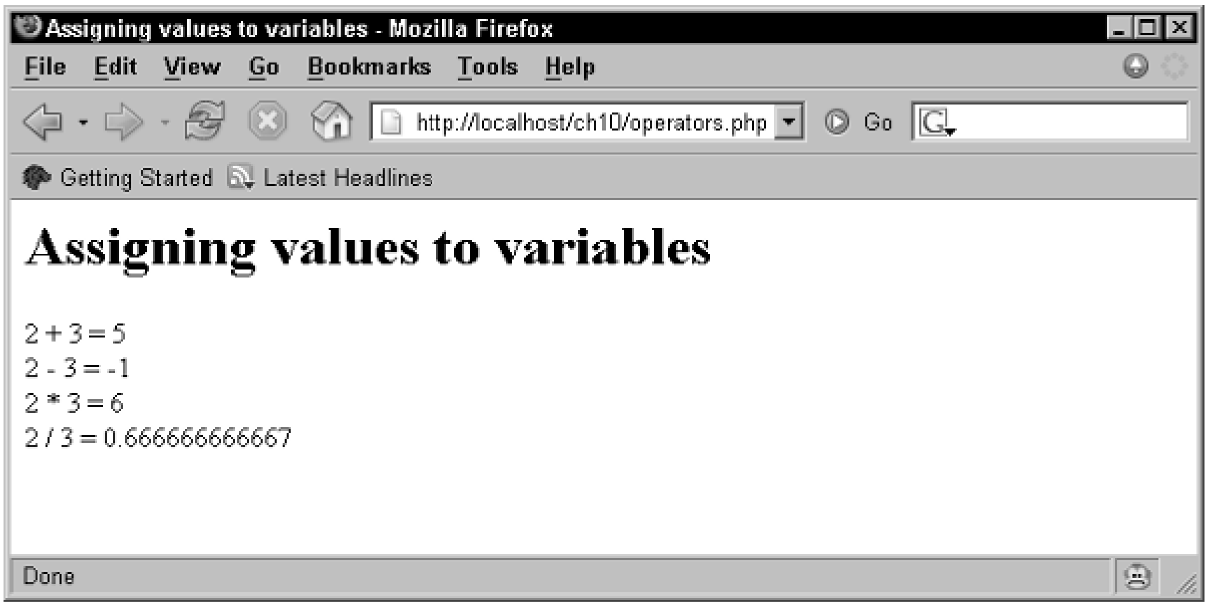
PHP has plenty of operators to handle your data, and most of them are the same as the operators in JavaScript. Here's a sampling of PHP operators:

- new
- [
- ! ~ ++ --
- / %
- + - .
- == !=
- &
- |
- &&
- ||
- ? :
- = += -= *= /= .= %= &= |= ^= <<= >>=

These operators work as you'd expect. Here's an example, operators.html, which puts a few of these operators to work:

```
<html>
<head>
<title>
Assigning values to variables
</title>
</head>
<body>
<h1>
Assigning values to variables
</h1>
<?
echo "2 + 3 = ", 2 + 3, "<br>";
echo "2 - 3 = ", 2 - 3, "<br>";
echo "2 * 3 = ", 2 * 3, "<br>";
echo "2 / 3 = ", 2 / 3, "<br>";
```

```
?>
</body>
</html>
```


Assigning values to variables - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/ch10/operators.php
Go
Getting Started
Latest Headlines
Done

Assigning values to variables

```
2 + 3 = 5
2 - 3 = -1
2 * 3 = 6
2 / 3 = 0.666666666667
```

- The list of PHP operators earlier in this section is given in terms of *operator precedence* in PHP, with higher-precedence operators first. Operator precedence indicates which operator will be executed first if there's a conflict.

- For example, what will the following statement display?

  ```
  echo 2 + 3 * 4;
  ```

  Will the 2 be added to the 3 and then multiplied by 4 to give you 20? Or will the 3 be multiplied by 4 and then added to 2 to give you 14? In PHP, the multiplication operator, *, has higher precedence than the addition operator, +, so the * is executed first. So, 2 + 3 * 4 becomes 2 + 12, which gives you 14.

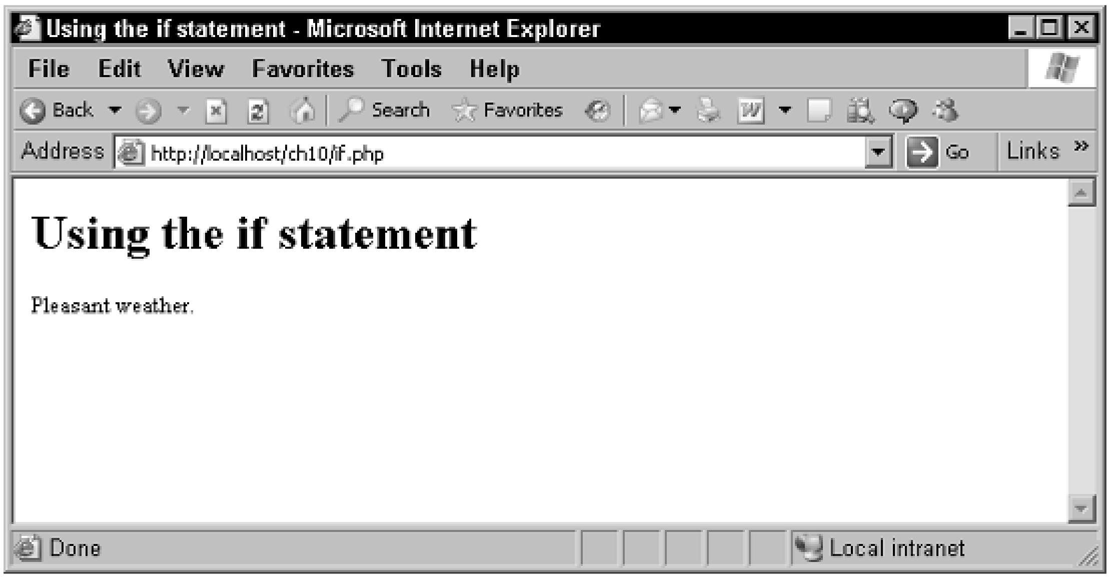
Making Choices with the if Statement

- PHP use if statements to make choices at runtime.
- Here's an example, if.php, which tests whether the value in a variable named $temperature is less than 80 degrees:

```
<html>
<head>
<title>
Using the if statement
</title>
```

```
</head>
<body>
<h1>
Using the if statement
</h1>
<?
$temperature = 75;
if ($temperature < 80) {
echo "Pleasant weather.";
}
?>
</body>
</html>
```

In this case, $temperature holds a value of 75, so the statement echo "Pleasant weather."; is executed.


Using the if statement - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/ch10/if.php
Go Links
Done
Local intranet

```
Using the if statement

Pleasant weather.
```

PHP also has an else statement, which works just as it does in JavaScript:

```
<body>
<h1>
Using the if statement
</h1>
<?
$temperature = 95;
if ($temperature < 80) {
```

echo “Pleasant weather.”;

}

else {

echo “Too hot.”;

}

?>

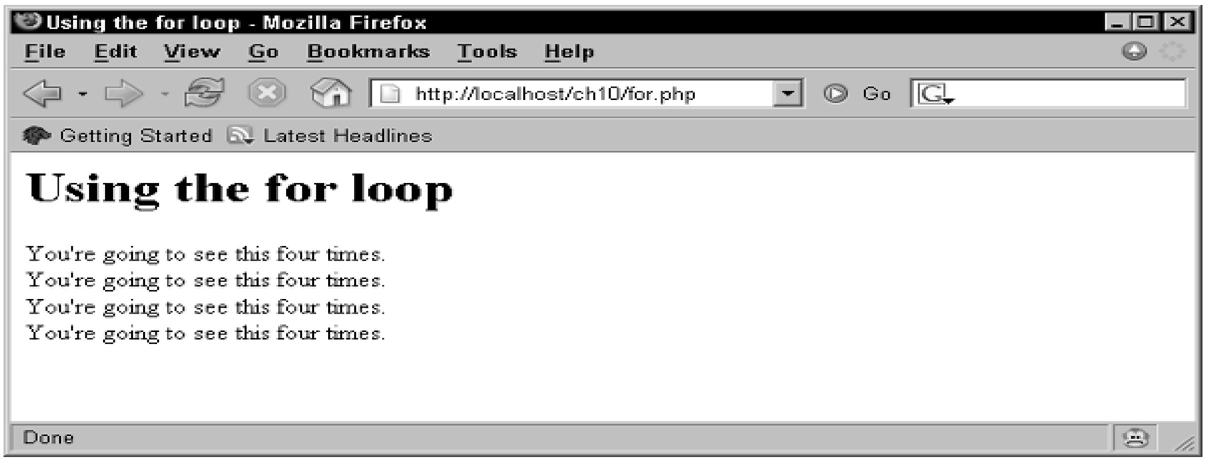
</body>

Round and Round with Loops

PHP also supports several loop statements. The for loop works just as it does in JavaScript; in fact, the only real difference is that you have to give the loop index variable an initial $, following the PHP way of naming variables.

1] for loop

```
<html>
<head>
<title>
Using the for loop
</title>
</head>
<body>
<h1>
Using the for loop
</h1>
<?
for ($loopCounter = 0; $loopCounter < 4; $loopCounter++){
echo “You’re going to see this four times.<br>”;
}
?>
</body>
</html>
```


Using the for loop - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/ch10/for.php
Go
Getting Started
Latest Headlines
Done

Using the for loop

```
You're going to see this four times.
You're going to see this four times.
You're going to see this four times.
You're going to see this four times.
```

2] while loop

PHP also has a while loop that keeps looping while its condition is true.

```
<html>
<head>
<title>
Using the while loop
</title>
</head>
<body>
<h1>
Using
the for
statement
in PHP.
Using the while loop
</h1>
<?php
$loopIndex = 1;
while ($loopIndex <= 4)
{
echo “You’re going to see this four times.<br>”;
$loopIndex++;
}
?>
</body>
</html>
```

3] do-while loop

PHP also has a do...while loop that checks its condition at the end of the loop, not the beginning, which is useful if the condition you want to test isn’t even set until the body of the loop is executed.

```
<?
$loopIndex = 1;
do {
echo “You’re going to see this four times.<br>”;
$loopIndex++;
} while ($loopIndex <= 4)
?>
```

3] foreach loop

PHP also has a foreach loop, which lets you automatically loop over arrays and other multiple-item objects.

This loop is handy because you don’t have to explicitly know how many items there are in an array to loop over it — all you have to do is give a name of a variable that will be filled with the current array item each time through the loop.

This example, xml.php, sends XML back to the server, using a foreach loop to create the XML document:

```
<?
header('Content-Type: text/xml');
$data = array('This', 'is', 'XML.');
echo '<?xml version="1.0" ?>';
echo '<document>';
foreach ($data as $value)
{
echo '<data>';
echo $value;
echo '</data>';
}
echo '</document>';
?>
```

Handling HTML Controls:

- When a Web page is sent to the server, you can extract the data from HTML controls yourself in a PHP script.
- To send data to the server when a Submit button is clicked, you'll need to set the following attributes of the HTML form containing the text field:

1] action: This attribute is assigned the URL to which the form data will be sent. You can omit this attribute, in which case its default is the URL of the current PHP document.

2] method: Specifies the method for sending data to the server. If you set it to GET (the default) this method sends all form name/value pair information in a URL that looks like:

URL?name=value&name=value&name=value

If you use the POST method, the contents of the form are encoded as with the GET method, but they are sent in hidden environment variable.

For example, this Web page, text.html, asks the user to enter his nickname in a text field named “nickname”, and then it posts that data to a PHP script named **phptext.php**

```
<html>
<head>
<title>
Sending data in text fields
</title>
</head>
<body>
<center>
<h1>
Sending data in text fields
</h1>
<form method=”post” action=”phptext.php”>
Enter your nickname:
<input name=”nickname” type=”text”>
<br>
<br>
<input type=”submit” value=”Submit”>
</form>
</center>
</body>
</html>
```

<u>To read a data from text field:</u>

1] If you sent data to the server by using the GET method, you can recover that data from the PHP $_GET array like this:

$_GET[“*nickname*”]

Where *nickname* is the name you gave to the text field (with the HTML name attribute).

2] If you sent the data by using the POST method, you can access the data in the text field as

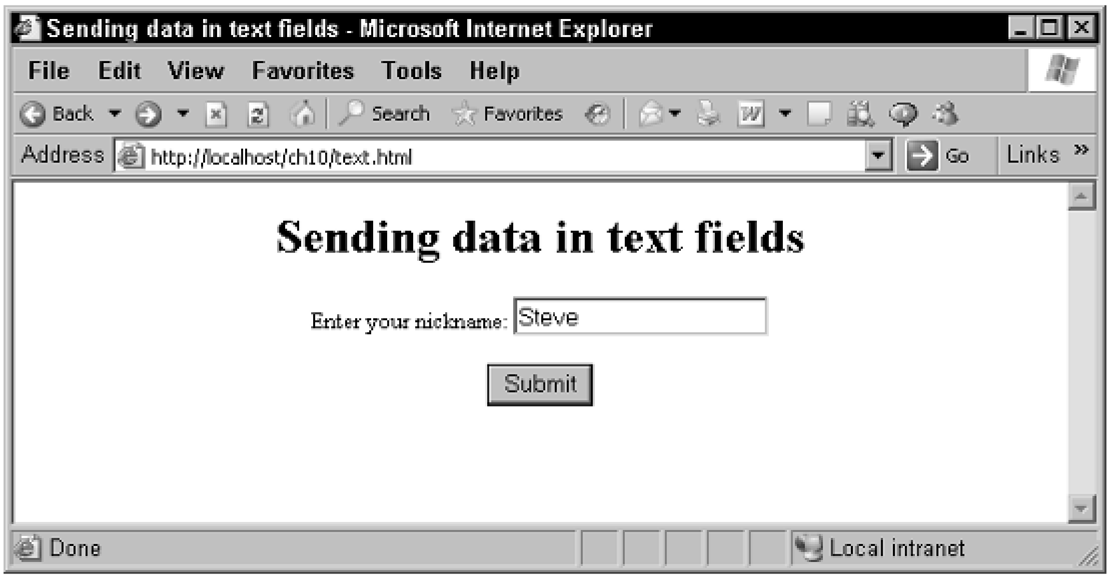
$_POST[“*nickname*”]

3] There’s another PHP array named $_REQUEST that lets you get that data regardless of whether you used the GET method or the POST method.

Continuing with the example, here’s how to use $_REQUEST to recover the text the user entered into the nickname text field:

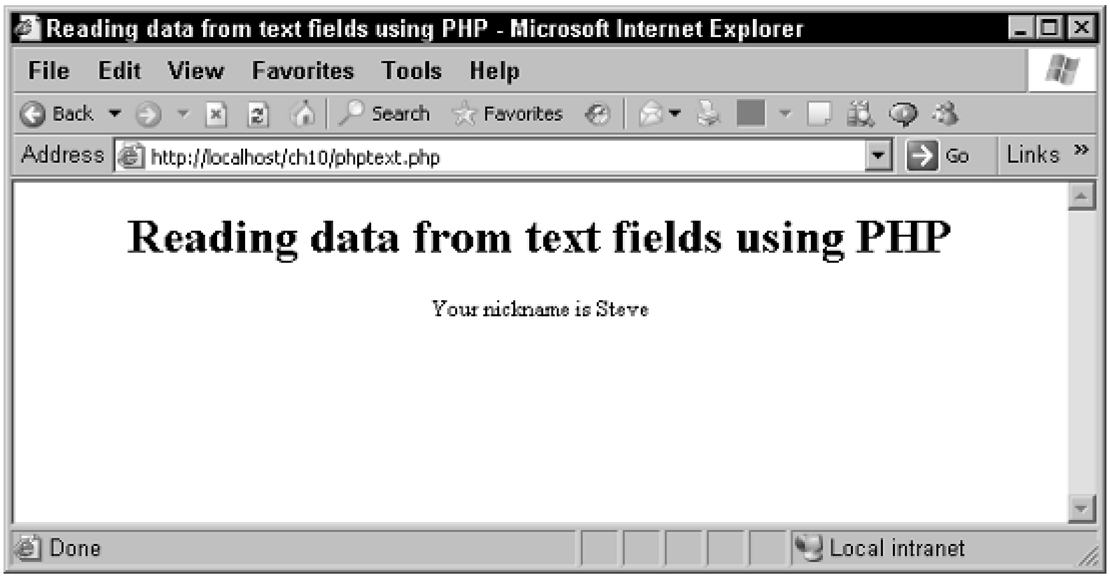
phptext.php

```
<?php
echo “Your nickname is”;
echo $_REQUEST[“nickname”];
?>
```


Sending data in text fields - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/ch10/text.html
Go Links
Submit
Done
Local intranet

Sending data in text fields

Enter your nickname: Steve


Reading data from text fields using PHP - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/ch10/phptext.php
Go Links
Done
Local intranet

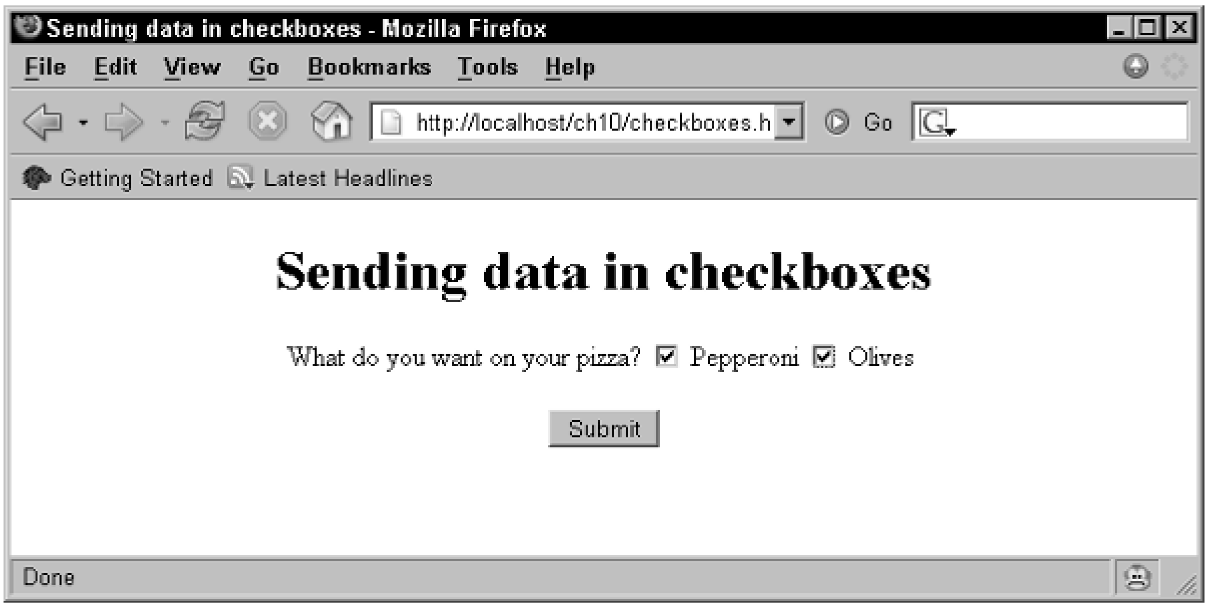
Reading data from text fields using PHP

Your nickname is Steve

<u>Checking out data from check boxes:</u>

The following example checkboxes.html, which asks the user what toppings he/she wants on pizza:

```
<html>
<head>
<title>Sending data in checkboxes</title>
</head>
<body>
<center>
<h1>Sending data in checkboxes</h1>
<form method="POST" action="checkboxes.php">
What do you want on your pizza?
<input name="pepperoni" type="checkbox" value="Pepperoni">
Pepperoni
<input name="olives" type="checkbox" value="Olives">
Olives
<br>
<br>
<input type="submit" value="Submit">
</form>
</center>
</body>
</html>
```


Sending data in checkboxes - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/ch10/checkboxes.h
Go
Getting Started
Latest Headlines
Done

Sending data in checkboxes

What do you want on your pizza? ☑ Pepperoni ☑ Olives

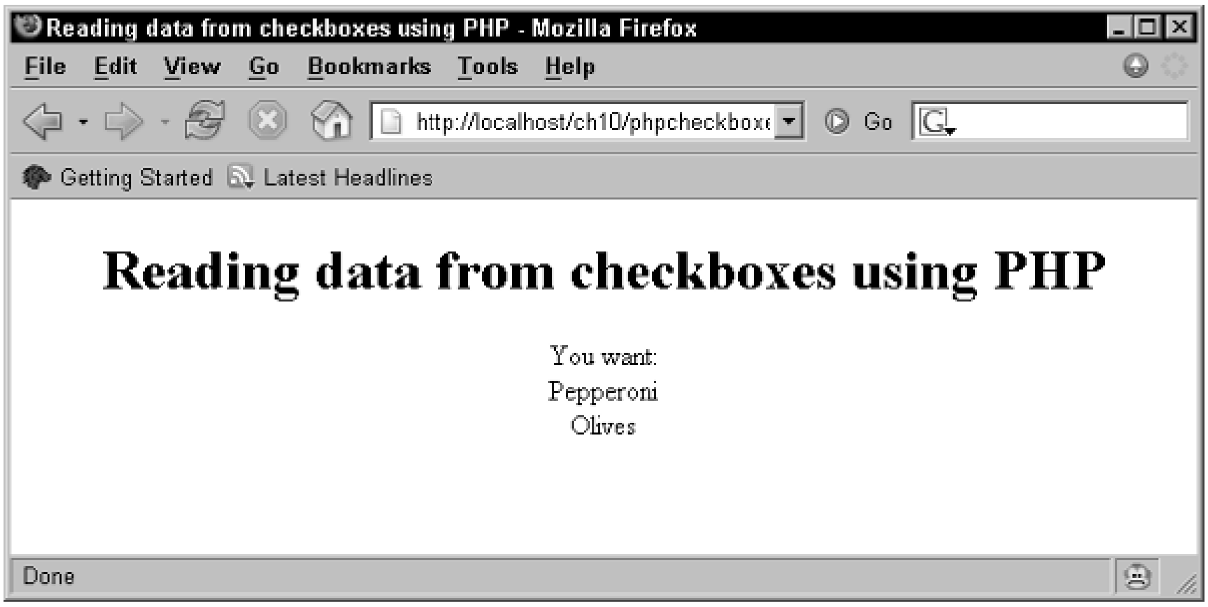
Submit

- You can determine whether a check box has been checked with the PHP isset function, which returns true if the parameter corresponding to an HTML control has been set, and false otherwise.
- If a check box has been checked, you can get the text that has been assigned to the check box's value attribute (that's "pepperoni" or "olives" in this example) using the $_GET, $_POST, or $_REQUEST arrays.

- In the following PHP code, phpcheckboxes.php, where you can recover the names of the toppings the user requested:

phpcheckboxes.php

```
<?php
        echo "Reading data from checkboxes using PHP";
        echo "You Want :" ;
        if (isset($_REQUEST["pepperoni"]))
                echo $_REQUEST["pepperoni"], "<br>";
        if (isset($_REQUEST["olives"]))
                echo $_REQUEST["olives"], "<br>";
?>
```


Reading data from checkboxes using PHP - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/ch10/phpcheckboxe
Go
Getting Started
Latest Headlines
Done

Reading data from checkboxes using PHP

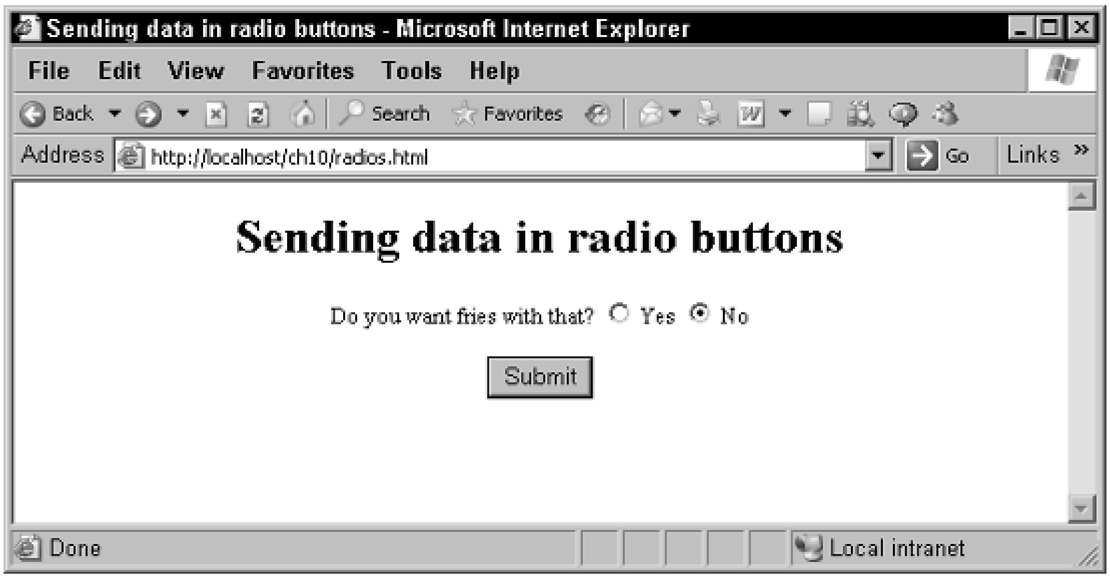
You want:
Pepperoni
Olives

Tuning in data from radio buttons:

radios.html

```
<html>
<head>
<title>Sending data in radio buttons</title>
</head>
<body>
<center>
<h1>Sending data in radio buttons</h1>
<form method="POST" action="phpradios.php">
Do you want fries with that?
<input name="radios" type="RADIO" value="Yes">
Yes
<input name="radios" type="RADIO" value="No">
No
<br>
```

```
<br>
<input type="SUBMIT" value="Submit">
</form>
</center>
</body>
</html>
```



Sending data in radio buttons - Microsoft Internet Explorer

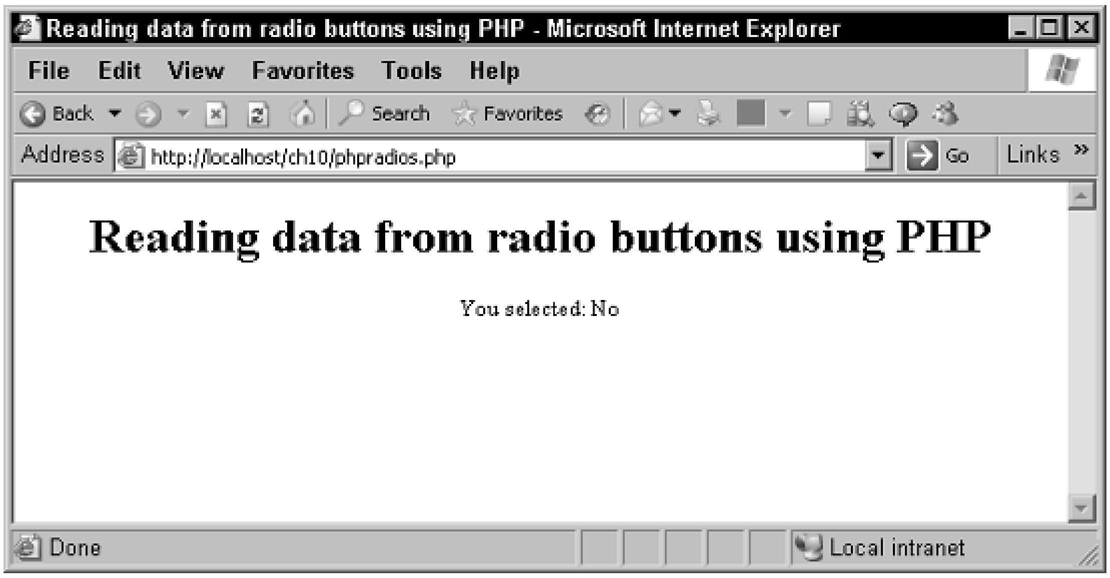
Address http://localhost/ch10/radios.html

Sending data in radio buttons

Do you want fries with that? Yes [✓ No]

- To recover the radio button that was selected in the radio button group, you use the name of the *group* with $_REQUEST, instead of having to work with each individual control as with check boxes.
- You can see how this works in **phpradios.php:**

```
<?php
    echo "Reading data from radio buttons using PHP";
    echo "You selected: ", $_REQUEST["radios"];
?>
```



Reading data from radio buttons using PHP - Microsoft Internet Explorer

Address http://localhost/ch10/phpradios.php

Reading data from radio buttons using PHP

You selected: No

Sending Data to the Server

- In Ajax, you don't usually rely on form submission to send data to the server.
- The usual way is to add your data to the end of the URL and use the GET method.
- Consider the following example, in which the code encodes the data to send to the server using a parameter named scheme:

```
function getOptions(scheme)
{
var url = "options2.php?scheme=" + scheme;
if(XMLHttpRequestObject)
{
        XMLHttpRequestObject.open("GET", url, true);
        XMLHttpRequestObject.onreadystatechange = function()
        {
        if (XMLHttpRequestObject.readyState == 4 &&
                XMLHttpRequestObject.status == 200)
                {
        var xmlDocument = XMLHttpRequestObject.responseXML;
        options = xmlDocument.getElementsByTagName("option");
        listOptions();
                }
        }
XMLHttpRequestObject.send(null);
}
}
```

In PHP on the server, you can recover the data in the scheme parameter as $_GET["scheme"].

Reading Files

- PHP lets you work with files on the server.
- To read from a file, you can use the PHP **fopen function** to open that file on the server.
- Here's how you typically use this function:

fopen (*filename*, *mode*)

Here, filename is the name of the file you're opening, and mode indicates how you want to open the file:

Sr.No	Mode	Description
1	**'r'**	Open the file for reading only.
2	**'r+'**	Open the file for reading and writing.
3	**'w'**	Open the file for writing only and truncate the file to zero length. If the file does not exist, PHP will attempt to create it.
4	**'w+'**	Open the file for reading and writing and truncate the file to zero length. If the file does not exist, PHP attempts to create it.

5	**‘a’**	Open the file for appending only. If the file does not exist, PHP will attempt to create it.
6	**‘a+’**	Open the file for reading and writing, starting at the end of the file. If the file does not exist, PHP will attempt to create it.
7	**‘x’**	Create and open the file for writing only. If the file already exists, the fopen call will not create the file and will return FALSE.
8	**‘x+’**	Create and open the file for reading and writing. If the file already exists, the fopen call will not create the file and will return FALSE.

- The fopen function returns a file handle, which stands for the file from then on in your code.
- You pass this file handle to various functions to read from the file, or write to it, and so on.
- For example, there are a variety of ways to read data from a file using PHP functions such as fgets, which reads a line of text from a file.
- To read a line of text, you pass it a file’s handle.
- Say you have a file, file.txt, on the server that has these contents:

file.txt

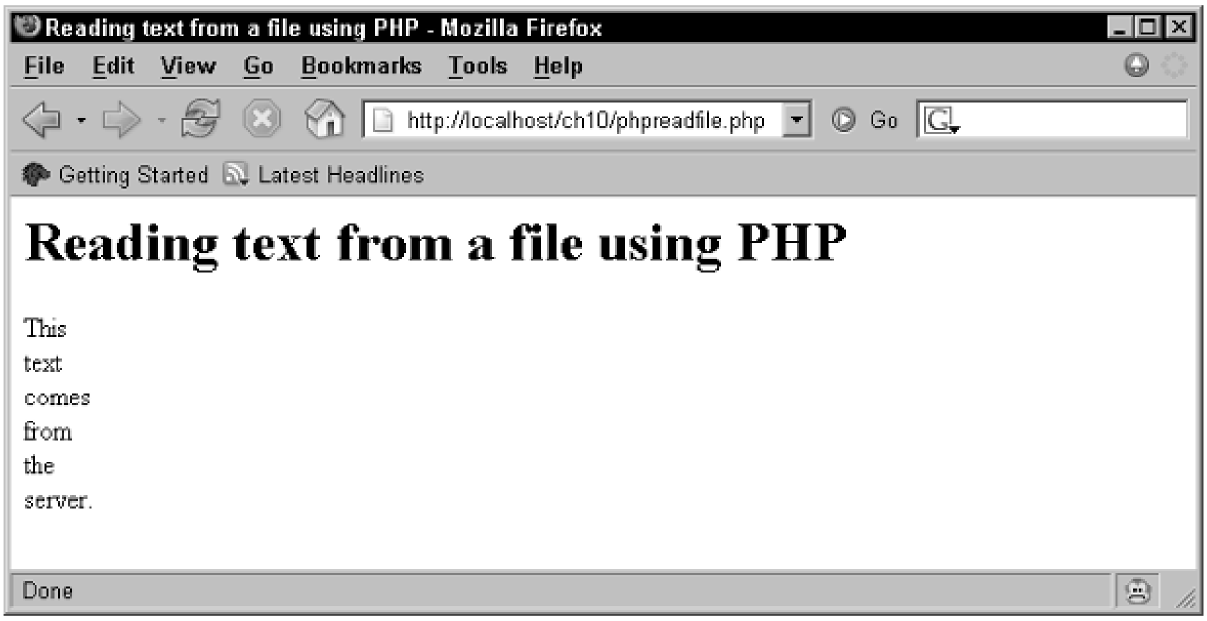
```
This
text
comes
from
the
server.
```

- How would you read this text?
- You can open file.txt with fopen and read successive lines with fgets in a loop. You can determine when you’ve reached the end of the file with the feof function, which returns true when you’re at the end of the file.
- Here’s how the text in file.txt can be read and displayed by PHP in **phpreadfile.php.**

```
<?php
$handle = fopen(“file.txt”, “r”);
while (!feof($handle))
{
      $text = fgets($handle);
      echo $text, “<br>”;
}
fclose($handle);
?>
```

- Note the expression !feof($handle). This expression uses the PHP “not” operator, !, which reverses true to false and false to true.
- So !feof($handle) is true while you haven’t reached the end of the file yet.
- Note also the use of fclose at the end of the code to *close* the file.

- When you're done with a file, you should close it with fclose. (Closing a file is the complementary operation to opening it.)


Reading text from a file using PHP - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/ch10/phpreadfile.php
Go
Getting Started
Latest Headlines
Done

Reading text from a file using PHP

```
This
text
comes
from
the
server.
```

- This example uses fgets to read strings of text from a file on the server.
- PHP offers other ways to do that as well, such as fgetc (which reads individual characters) and fread (which reads data byte by byte).

Writing Files

- You can also write to a file on the server using PHP and the **fwrite function.**
- For example, say that you wanted to create the file **file.txt** on the server, with the same content I use in the previous sections.
- You can start by putting the text that you want in this file in a variable named $text.

  $text = "This\ntext\ncomes\nfrom\nthe\server.";
- Note the \n codes here: Each such code stands for a newline character that breaks the text up into separate lines, just as the original text for this file.
- To be able to write files on the server, you first have to make sure you have permission to do so.
- If you aren't authorized to write to files, you can't use examples like this one.
- To write to file.txt, you just have to open that file for writing (passing a mode of "w" to fopen), and then use fwrite to write to the file the text you want.
- The fwrite function returns true if it is successful and FALSE otherwise.

Here's what creating the file looks like in phpfilewrite.php:

```
<?php
$handle = fopen("file.txt", "w");
```

```
$text = "This\ntext\ncomes\nfrom\nthe\server.";
if (fwrite($handle, $text) == FALSE) {
echo "Could not create file.txt.";
}
else {
echo "Created file.txt OK.";
}
fclose($handle);
?>
```

Opening a file with "w" truncates it to zero length first (before you start writing) so the current contents are lost. In addition to creating files this way, you can also open them for appending, using fopen mode "a", which means anything you add to the file will be added to the end of the file.

- **Working with Databases**
  - PHP excels at connections to various database systems, which can be good for Ajax programmers who want to retrieve data from the server.
  - PHP has many built-in functions to work with various database systems; one popular choice is MySQL (www.mysql.com). PHP comes with built-in functions like
    - **mysql_connect (to connect to a MySQL database server)**
    - **mysql_select_db (to select a database to work with)**
    - **mysql_query (to send an SQL query to the database)**
    - **mysql_fetch_array (to convert the results of a query to an array)**

Here's how you can fetch the products database and the pencils table inside it, displaying the values in the type and number fields in the table's rows in an HTML table:

```
<?
$connection = mysql_connect("localhost","root","");
$db = mysql_select_db("products", $connection);
$query = "SELECT * FROM pencils";
$result = mysql_query($query);
echo "<table border='1'>";
echo "<tr>";
echo "<th>Type</th><th>Number</th>";
echo "</tr>";
while ($row = mysql_fetch_array($result))
{
echo "<tr>";
echo "<td>", $row['name'], "</td><td>", $row['number'],
"</td>";
echo "</tr>";
}
echo "</table>";
mysql_close($connection);
?>
```