Chapter 2

How to code a PHP application

Objectives

Applied

1. Given the specifications for a PHP application that requires only the skills and language elements presented in this chapter, code, test, and debug the application. That includes these skills:
    - Creating variables with valid names and assigning values to them
    - Using literals and concatenating strings
    - Using the built-in $_GET and $_POST arrays
    - Using echo statements to display data on a page
    - Coding string and numeric expressions
    - Using compound assignment operators
    - Using the built-in number_format, date, isset, empty, and is_numeric functions

Objectives (continued)

Applied (continued)

- Coding conditional expressions
- Coding if, while, and for statements
- Using built-in functions like include and require to pass control to another page

2. Access and use the online PHP documentation.

Objectives (continued)

Knowledge

1. Explain how PHP is embedded within an HTML document.
2. Distinguish between PHP statements and comments.
3. Describe these PHP data types: integer, double, Boolean, and string.
4. List the rules for creating a PHP variable name.
5. Describe the code for declaring a variable and assigning a value to it.
6. Describe the use of the built-in $_GET and $_POST arrays.
7. Describe the use of the echo statement.
8. Describe the rules for evaluating an arithmetic expression, including order of precedence and the use of parentheses.

Objectives (continued)

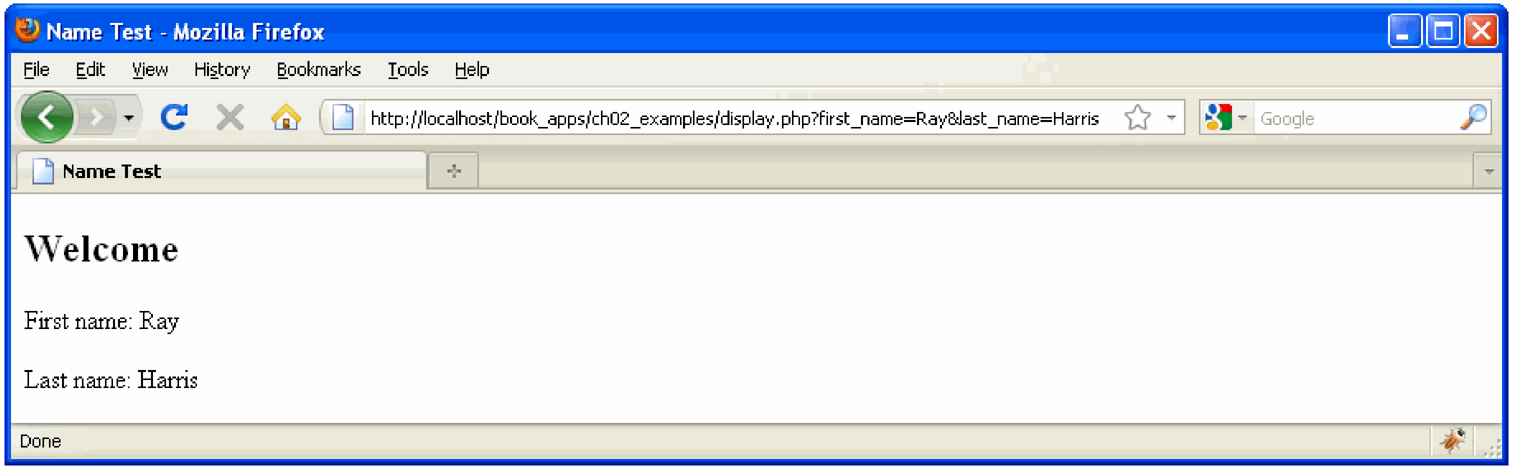
Knowledge (continued)

9. Describe the use of these built-in functions: number_format, date, isset, is_numeric, include, and require.
10. Describe the rules for evaluating a conditional expression, including order of precedence and the use of parentheses.
11. Describe the flow of control of an if, while, or for statement.

A PHP file that includes HTML and embedded PHP

```
<?php
    // get the data from the request
    $first_name = $_GET['first_name'];
    $last_name = $_GET['last_name'];
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional
    ...>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Name Test</title>
        <link rel="stylesheet" type="text/css"
              href="main.css"/>
    </head>
    <body>
        <h2>Welcome</h2>
        <p>First name: <?php echo $first_name; ?></p>
        <p>Last name: <?php echo $last_name; ?></p>
    </body>
</html>
```

The PHP file displayed in a browser


Name Test - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Google
Name Test
Done

http://localhost/book_apps/ch02_examples/display.php?first_name=Ray&last_name=Harris

Welcome

First name: Ray

Last name: Harris

PHP code: comments and statements

```
<?php
    /*********************************************
     * This program calculates the discount for a
     * price that's entered by the user
     ********************************************/

    // get the data from the form
    $list_price = $_GET['list_price'];

    // calculate the discount
    $discount_percent = .20;  // 20% discount
    $discount_amount =
        $subtotal * $discount_percent;
    $discount_price =
        $subtotal - $discount_amount;
?>
```

Another way to code single-line comments

```
# calculate the discount
$discount_percent = .20;      # 20% discount
```

Syntax rules

- PHP statements end with a semicolon.
- PHP ignores extra whitespace in statements.

The six PHP data types

integer

double

boolean

string

array

object

Integer values (whole numbers)

```
15                  // an integer
-21                 // a negative integer
```

Double values (numbers with decimal positions)

```
21.5                // a floating-point value
-124.82             // a negative floating-point value
```

The two Boolean values

```
true                // equivalent to true, yes, or on
false               // equivalent to false, no, or off
```

String values

```
'Ray Harris'        // a string with single quotes
"Ray Harris"        // a string with double quotes
''                  // an empty string
null                // a NULL value
```

Double values that use scientific notation

```
3.7e9               // equivalent to 3700000000
4.5e-9              // equivalent to 0.0000000037
-3.7e9              // equivalent to -3700000000
```

How assign string expressions

Use single quotes to improve PHP efficiency

```
$first_name = 'Bob';
$last_name = 'Roberts';
```

Assign NULL values and empty strings

```
$address2 = '';                 // an empty string
$address2 = null;               // a NULL value
```

Use double quotes for variable substitution

```
$name = "Name: $first_name";         // Name: Bob
$name = "$first_name $last_name";    // Bob Roberts
```

Mix single and double quotes for special purposes

```
$last_name = "O'Brien";              // O'Brien
$line = 'She said, "Hi."';           // She said, "Hi."
```

Using the assignment operator (=) as you declare a variable and give it a value

```
$count = 10;                // an integer literal
$list_price = 9.50;         // a double literal
$first_name = 'Bob';        // a string literal
$first_name = "Bob";        // a string literal
$is_valid = false;          // a Boolean literal

$product_count = $count;    // $product_count is 10
$price = $list_price;       // $price is 9.50
$name = $first_name;        // $name is "Bob"
$is_new = $is_valid;        // $is_new is FALSE
```

Rules for creating variable names

- Variable names are case-sensitive.
- Variable names can contain letters, numbers, and underscores.
- Variable names can't contain special characters.
- Variable names can't begin with a digit or two underscores.
- Variable names can't use names that are reserved by PHP such as the variable named $this that's reserved for use with objects.

How to declare a constant

```
define('MAX_QTY', 100);       // an integer constant
define('PI', 3.14159265);     // a double constant
define('MALE', 'm');          // a string constant
```

Using a constant

- Since the value of a constant can’t be changed, don’t code the $ when you declare it or use it.
- Most programmers use all caps for constants.

How to use the concatenation operator (.)

How to use the concatenation operator for simple joins

```
$first_name = 'Bob';
$last_name = 'Roberts';
$name = 'Name: ' . $first_name;          // Name: Bob
$name = $first_name . ' ' . $last_name;  // Bob Roberts
```

How to join a number to a string

```
$price = 19.99;
$price_string = 'Price: ' . $price;    // Price: 19.99
```

The syntax for the echo statement

```
echo string_expression;
```

How to use an echo statement within HTML

```
<p>Name: <?php echo $name; ?></p>
```

How to use an echo statement to output HTML tags and data

```
<?php
    echo '<p>Name: ' . $name . '</p>';
?>
```

Common arithmetic operators

Operator	Example	Result
`+`	`5 + 7`	`12`
`-`	`5 - 12`	`-7`
`*`	`6 * 7`	`42`
`/`	`13 / 4`	`3.25`
`%`	`13 % 4`	`1`
`++`	`$counter++`	adds 1 to counter
`--`	`$counter--`	subtracts 1 from counter

Some simple numeric expressions

```
$x = 14;
$y = 8;
$result = $x + $y;          // 22
$result = $x - $y;          // 6
$result = $x * $y;          // 112
$result = $x / $y;          // 1.75
$result = $x % $y;          // 6
$x++;                       // 15
$y--;                       // 7
```

The order of precedence

Order	Operators	Direction
1	`++`	Left to right
2	`--`	Left to right
3	`*   /   %`	Left to right
4	`+   -`	Left to right

Order of precedence and the use of parentheses

```
3 + 4 * 5                   // 23
(3 + 4) * 5                 // 35
```

The compound assignment operators

`.=`	Append a string expression to the variable
`+=`	
`-=`	
`*=`	
`/=`	
`%=`	

Two ways to append string data to a variable

The standard assignment operator

```
$name = 'Ray ';
$name = $name . 'Harris';       // 'Ray Harris'
```

A compound assignment operator

```
$name = 'Ray ';
$name .= 'Harris';              // 'Ray Harris'
```

Three ways to increment a counter variable

The standard assignment operator

```
$count = 1;
$count = $count + 1;
```

The compound assignment operator

```
$count = 1;
$count += 1;
```

The increment operator

```
$count = 1;
$count++;
```

More examples

How to append numeric data to a string variable

```
$message = 'Months: ';
$months = 120;
$message .= $months;          // 'Months: 120'
```

How to work with numeric data

```
$subtotal = 24.50;
$subtotal += 75.50;           // 100
$subtotal *= .9;              // 90 (100 * .9)
```

A function for formatting numbers

```
number_format($number[, $decimals])
```

Statements that format numbers

```
$nf = number_format(12345);                 // 12,345
$nf = number_format(12345, 2);              // 12,345.00
$nf = number_format(12345.674, 2);          // 12,345.67
$nf = number_format(12345.675, 2);          // 12,345.68
```

To remove or change the decimal or comma

number_format ($number , $decimals, $dec_point=‘ ‘, $thousands_sep=‘ ‘)

A function for getting the current date

```
date($format)
```

Commonly used characters for date formatting

Character	Description
`Y`	A four-digit year such as 2010.
`y`	A two-digit year such as 10.
`m`	Numeric representation of the month with leading zeroes (01-12).
`d`	Numeric representation of the day of the month with leading zeroes (01-31).

Statements that format a date

```
$date = date('Y-m-d');    // 2010-06-12
$date = date('m/d/y');    // 06/12/10
$date = date('m.d.Y');    // 06.12.2010
$date = date('Y');        // 2010
```

An HTML form that does an HTTP GET request

```
<form action="display.php" method="get">
    <label>First name: </label>
    <input type="text" name="first_name"/><br />
    <label>Last name: </label>
    <input type="text" name="last_name"/><br />
    <label>&nbsp;</label>
    <input type="submit" value="Submit"/>
</form>
```

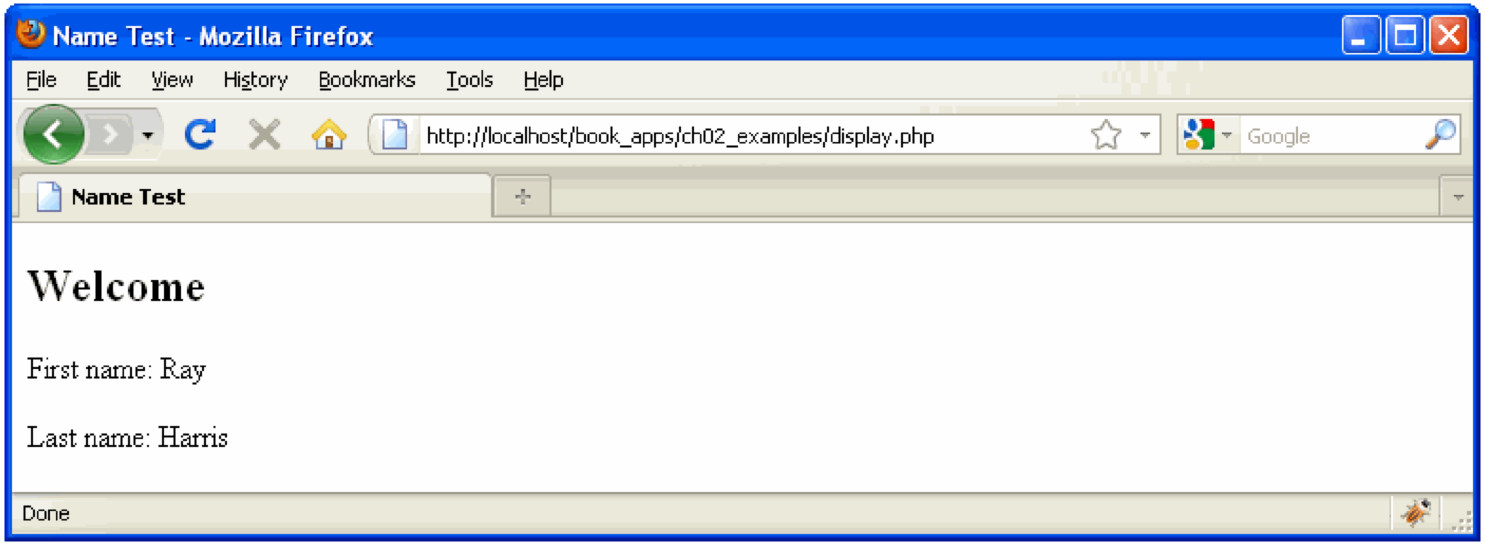
The URL for the HTTP GET request

```
//localhost/.../display.php?first_name=Ray&last_name=Harris
```

Getting the data and storing it in variables

```
$first_name = $_GET['first_name'];
$last_name = $_GET['last_name'];
```

A PHP page for an HTTP POST request


Name Test - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/book_apps/ch02_examples/display.php
Google
Name Test
Done

Welcome

First name: Ray

Last name: Harris

An HTML form that specifies the POST method

```
<form action="display.php" method="post">
```

Code that gets the data from the $_POST array

```
$first_name = $_POST['first_name'];
$last_name = $_POST['last_name'];
```

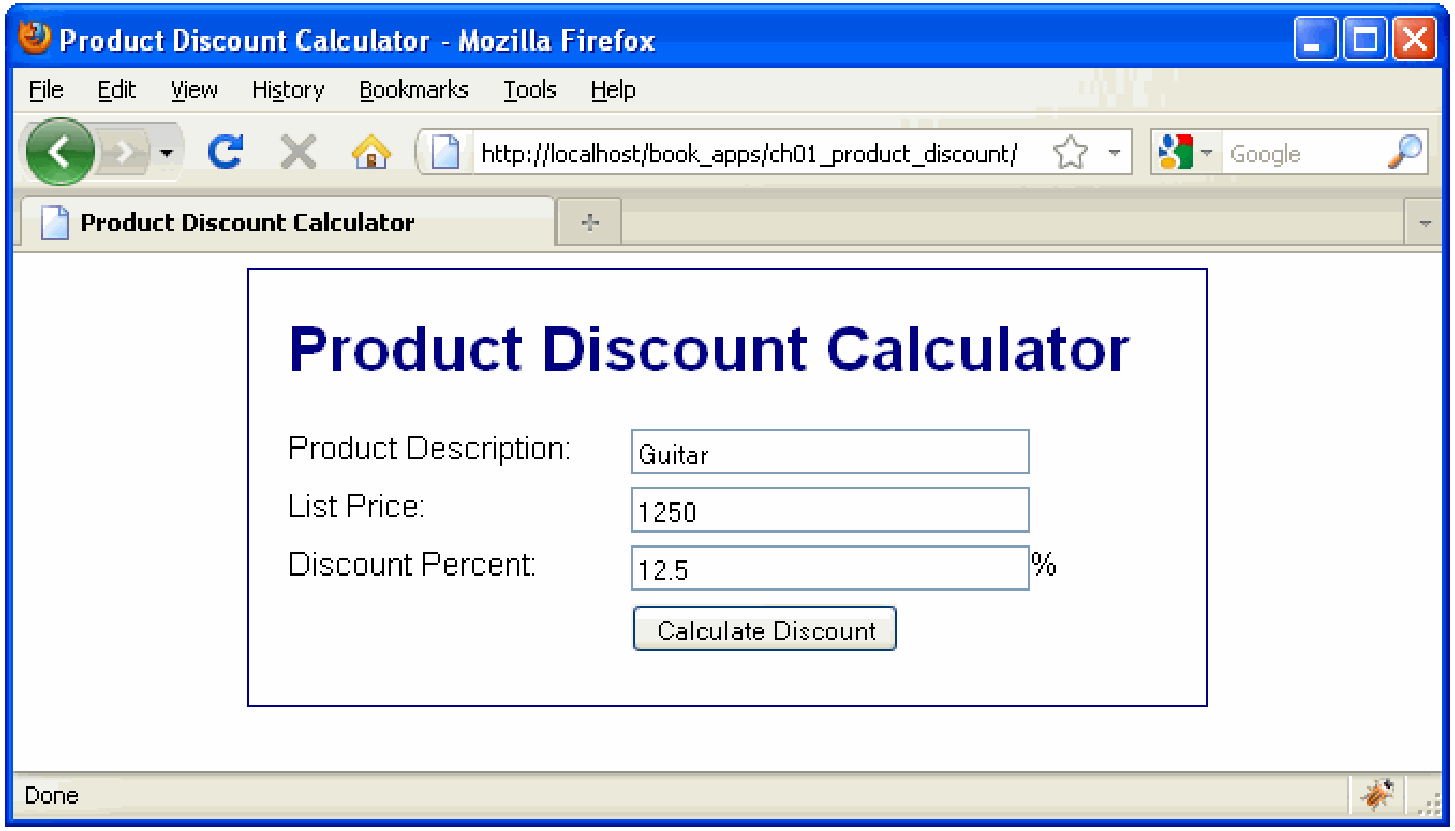
When to use the HTTP GET method

- When the request is for a page that gets data from a database server.
- When the request can be executed multiple times without causing any problems.

When to use the HTTP POST method

- When the request is for a page that writes data to a database server.
- When executing the request multiple times may cause problems.
- When you don’t want to include the parameters in the URL for security reasons.
- When you don’t want users to be able to include parameters when they bookmark a page.
- When you need to transfer more than 4 KB of data.

The first page (index.html)

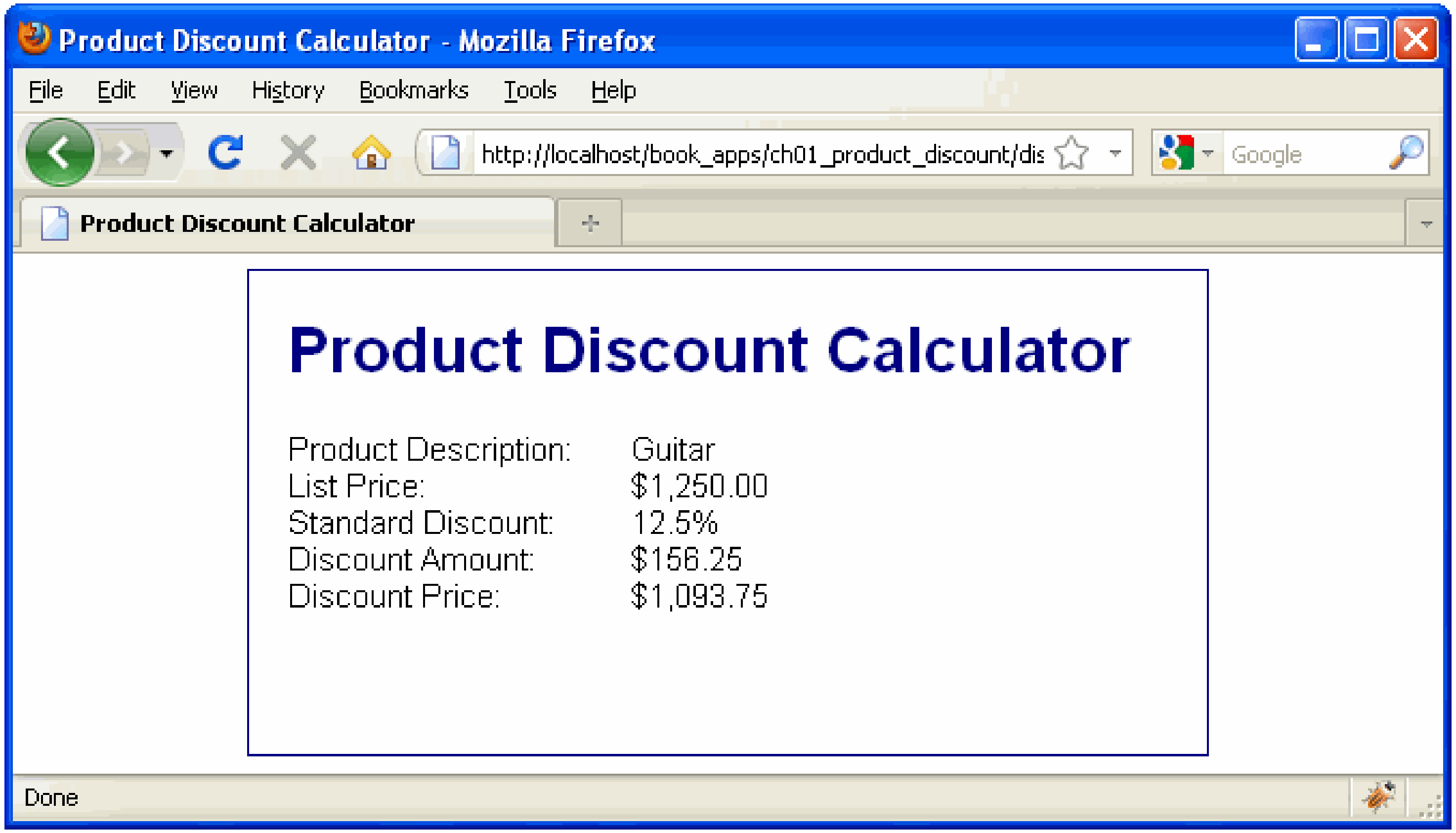

Product Discount Calculator - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/book_apps/ch01_product_discount/
Google
Product Discount Calculator
Done

Product Discount Calculator

Product Description:	Guitar
List Price:	1250
Discount Percent:	12.5 %

Calculate Discount

The second page (product_discount.php)


Product Discount Calculator - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/book_apps/ch01_product_discount/dis
Google
Product Discount Calculator
Done

Product Discount Calculator

Product Description:	Guitar
List Price:	$1,250.00
Standard Discount:	12.5%
Discount Amount:	$156.25
Discount Price:	$1,093.75

The code for the form on the first page

```
<form action="display_discount.php" method="post">

    <div id="data">
        <label>Product Description:</label>
        <input type="text"
               name="product_description"/><br />
        <label>List Price:</label>
        <input type="text" name="list_price"/><br />
        <label>Discount Percent:</label>
        <input type="text" name="discount_percent"/>%<br />
    </div>

    <div id="buttons">
        <label>&nbsp;</label>
        <input type="submit" value="Calculate Discount" />
        <br />
    </div>

</form>
```

The PHP file (display_discount.php)

```
<?php
    // get the data from the form
    $product_description = $_POST['product_description'];
    $list_price = $_POST['list_price'];
    $discount_percent = $_POST['discount_percent'];

    // calculate the discount and discounted price
    $discount = $list_price * $discount_percent * .01;
    $discount_price = $list_price - $discount;

    // apply formatting to the dollar and percent amounts
    $list_price_formatted =
        "$".number_format($list_price, 2);
    $discount_percent_formatted = $discount_percent."%";
    $discount_formatted = "$".number_format($discount, 2);
    $discount_price_formatted =
        "$".number_format($discount_price, 2);
?>
```

The PHP file (display_discount.php) (continued)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional
    ...>
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
    <title>Product Discount Calculator</title>
    <link rel="stylesheet" type="text/css"
          href="main.css"/>
</head>

<body>
    <div id="content">
        <h1>Product Discount Calculator</h1>

        <label>Product Description:</label>
        <span><?php echo $product_description; ?>
        </span><br />
```

The PHP file (display_discount.php) (continued)

```
        <label>List Price:</label>
        <span><?php echo $list_price_formatted; ?>
        </span><br />

        <label>Standard Discount:</label>
        <span><?php echo $discount_percent_formatted; ?>
        </span><br />

        <label>Discount Amount:</label>
        <span><?php echo $discount_formatted; ?>
        </span><br />

        <label>Discount Price:</label>
        <span><?php echo $discount_price_formatted; ?>
        </span><br />
    </div>
</body>
</html>
```

The relational operators

Operator	Example

`==`	`$last_name == "Harris"`
	`$test_score == 10`

`!=`	`$first_name != "Ray"`
	`$months != 0`

`<`	`$age < 18`
`<=`	`$investment <= 0`
`>`	`$test_score > 100`
`>=`	`$rate / 100 >= 0.1`

The logical operators in order of precedence

Operator	Example
`!`	`!is_numeric($age)`
`&&`	`$age > 17 && $score < 70`
`||`	`!is_numeric($rate) || $rate < 0`

Three functions for checking variable values

```
isset($var)
empty($var)
is_numeric($var)
```

Function calls that check variable values

```
isset($name)            // TRUE if $name has been set
                        // and is not NULL
empty($name)            // TRUE if $name is empty
is_numeric($price)      // TRUE if $price is a number
```

An if statement with no other clauses

```
if ( $price <= 0 ) {
    $message = 'Price must be greater than zero.';
}
```

An if statement with an else clause

```
if ( empty($first_name) ) {
    $message = 'You must enter your first name.';
} else {
    $message = 'Hello ' . $first_name.'!';
}
```

An if statement with else if and else clauses

```
if ( empty($investment) ) {
    $message = 'Investment is a required field.';
} else if ( !is_numeric($investment) )  {
    $message = 'Investment must be a valid number.';
} else if ( $investment <= 0 )  {
    $message = 'Investment must be greater than zero.';
} else {
    $message = 'Investment is valid!';
}
```

A compound conditional expression

```
if ( empty($investment) || !is_numeric($investment)
                        || $investment <= 0 ) {
    $message =
        'Investment must be a valid number > zero.';
}
```

A nested if statement

```
if ( empty($months) || !is_numeric($months)
                    || $months <= 0 ) {
    $message = 'Please enter a number of months > zero.';
} else {
    $years = $months / 12;
    if ( $years > 1 ) {
        $message = 'A long-term investment.';
    } else {
        $message = 'A short-term investment.';
    }
}
```

A while loop that stores the numbers 1 through 5

```
$counter = 1;
while ($counter <= 5) {
    $message = $message . $counter . '|';
    $counter++;
}
// $message = 1|2|3|4|5|
```

A for loop that stores the numbers 1 through 5

```
for ($counter = 1; $counter <= 5; $counter++) {
    $message = $message . $counter . '|';
}
// $message = 1|2|3|4|5|
```

A while loop that calculates the future value of a one-time investment

```
$investment = 1000;
$interest_rate = .01;
$years = 25;
$future_value = $investment;

$i = 1;
while ($i <= $years) {
  $future_value =
      ($future_value + ($future_value * $interest_rate);
  $i++;
}
```

A for loop that calculates the future value of a one-time investment

```
$investment = 1000;
$interest_rate = .01;
$years = 25;
$future_value = $investment;

for ($i = 1; $i <= $years; $i++) {
  $future_value =
        ($future_value + ($future_value * $interest_rate));
}
```

Built-in functions that pass control

include(*$path*)

- Inserts and runs the specified file. If this functions fails, it causes a warning that can allow the script to continue.

include_once(*$path*)

- Same as include, but it makes sure the file is included only once.

require(*$path*)

- Same as include, but if it fails it causes a fatal error that stops the script.

require_once(*$path*)

exit([*$status*])

- Exits the current php script

die([*$status*])

- Same as exit

The include function

```
include 'index.php';    // parentheses are optional
include('index.php');   // index.php in the current
                        // directory
```

The require function

```
require('index.php');   // index.php in the current
                        // directory
```

The exit function

```
exit;                   // parentheses are optional
exit();
exit('Unable to connect to DB.');
                        // passes a message to the browser
```

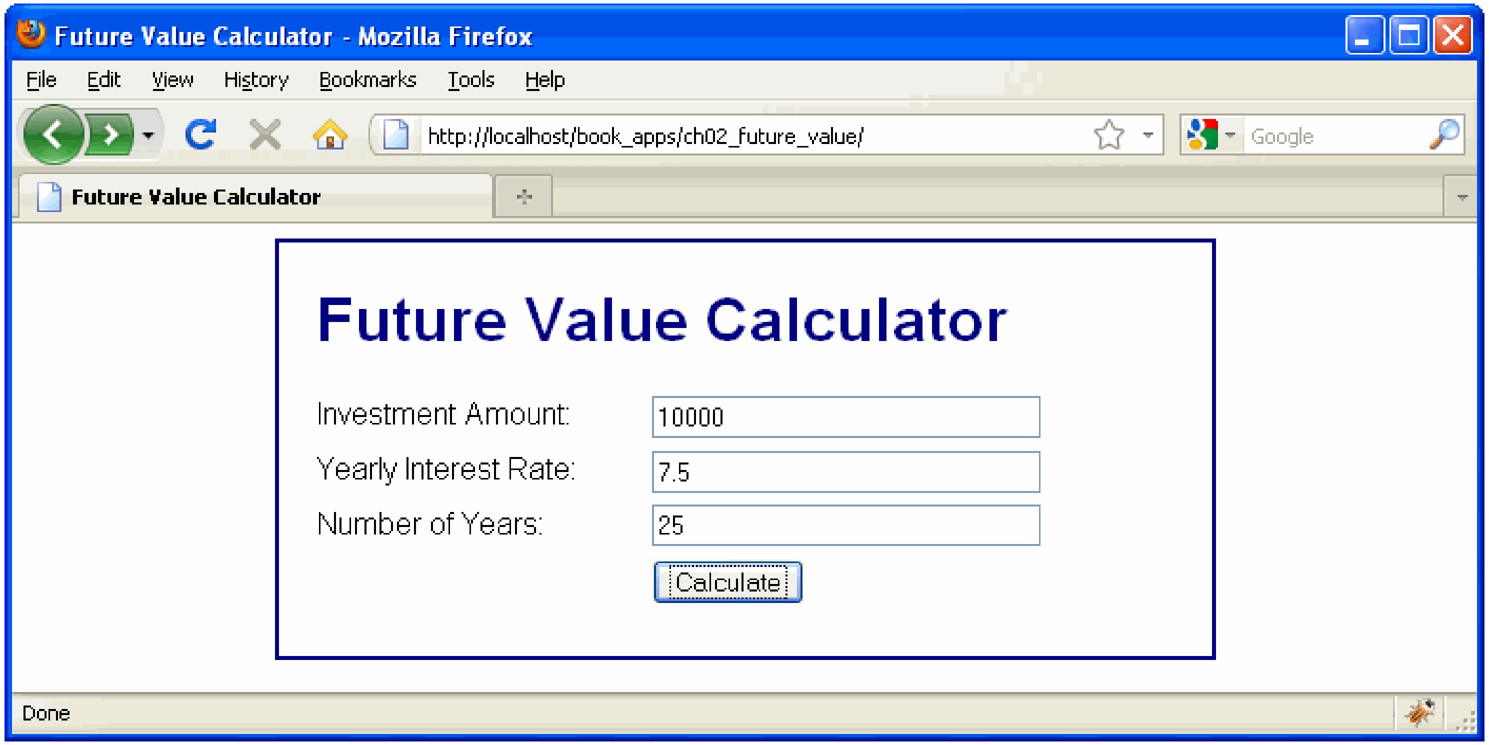
How to pass control to another PHP file in the current directory

```
if ($is_valid) {
    include('process_data.php');
    exit();
}
```

How to navigate up and down directories

```
include('view/header.php');   // down one directory
include('./error.php');       // in the current directory
include('../error.php');      // up one directory
include('../../error.php');   // up two directories
```

The first page

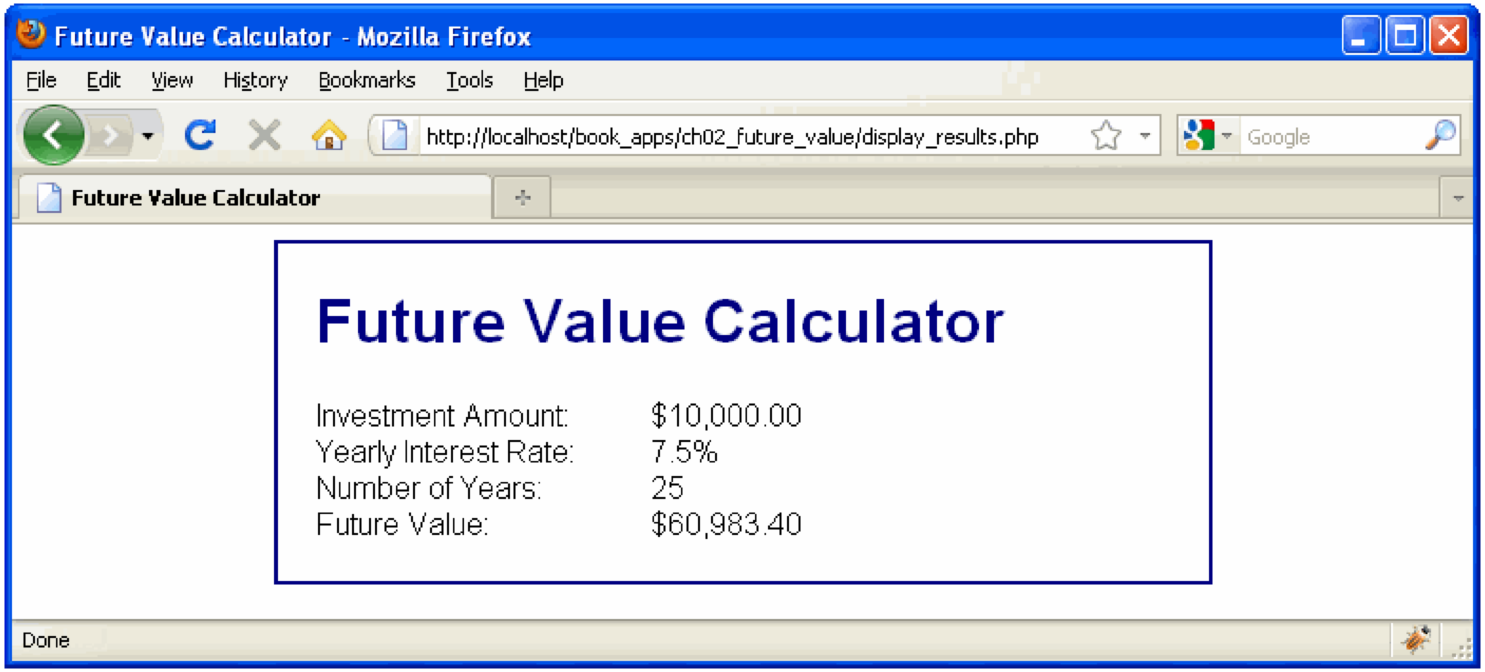

Future Value Calculator - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/book_apps/ch02_future_value/
Google
Future Value Calculator
Done

Future Value Calculator

Investment Amount:	10000
Yearly Interest Rate:	7.5
Number of Years:	25

Calculate

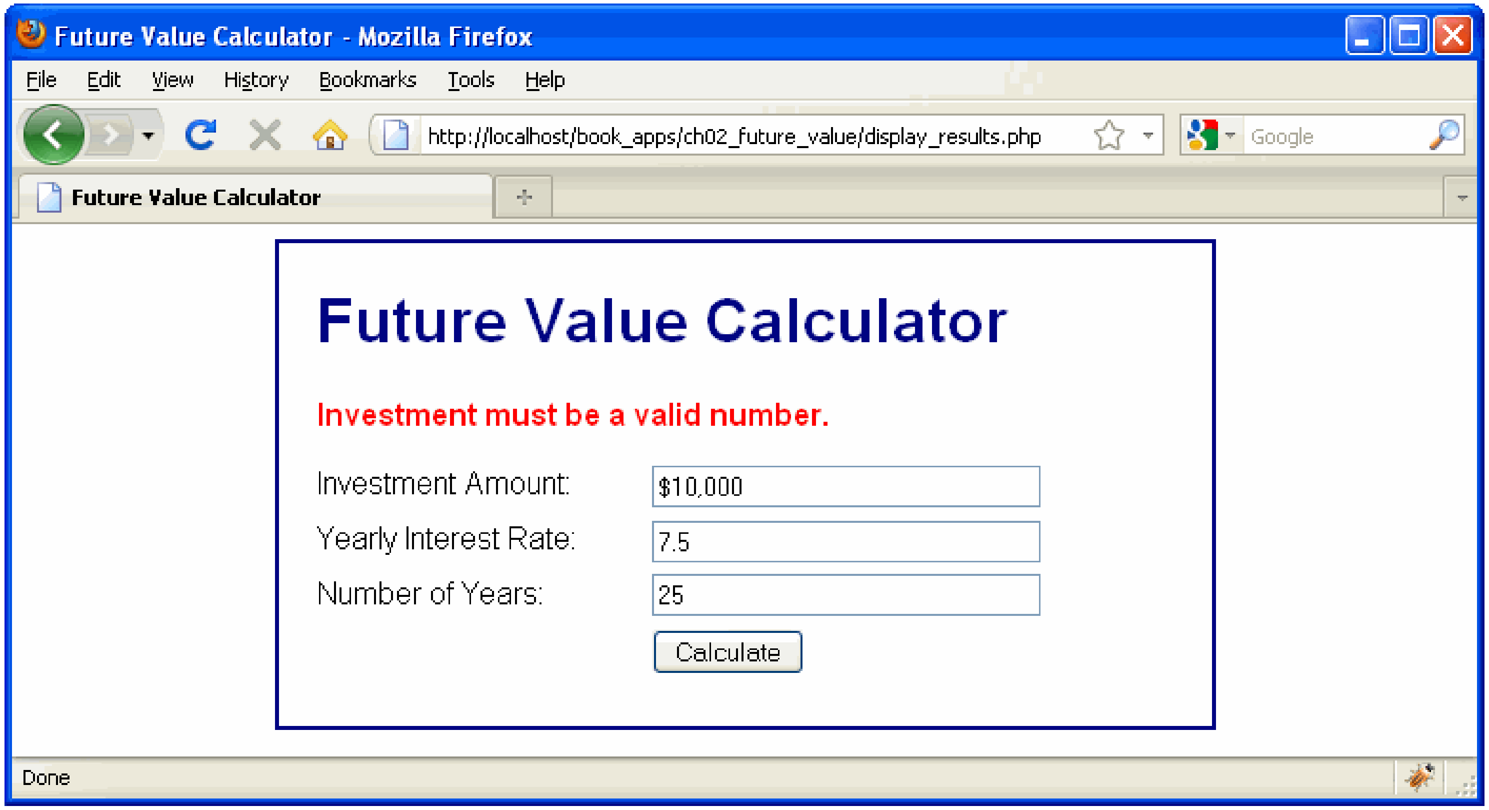
The second page


Future Value Calculator - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/book_apps/ch02_future_value/display_results.php
Google
Future Value Calculator
Done

Future Value Calculator

Investment Amount:	$10,000.00
Yearly Interest Rate:	7.5%
Number of Years:	25
Future Value:	$60,983.40

The first page with an error message


Future Value Calculator - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Google
Future Value Calculator
Done

http://localhost/book_apps/ch02_future_value/display_results.php

Future Value Calculator

Investment must be a valid number.

Investment Amount:	$10,000
Yearly Interest Rate:	7.5
Number of Years:	25

Calculate

The index.php file

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional
  ...>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Future Value Calculator</title>
  <link rel="stylesheet" type="text/css" href="main.css"/>
</head>

<body>
  <div id="content">
  <h1>Future Value Calculator</h1>
  <?php if (!empty($error_message)) { ?>
      <p class="error"><?php echo $error_message; ?></p>
  <?php } ?>
  <form action="display_results.php" method="post">

    <div id="data">
      <label>Investment Amount:</label>
      <input type="text" name="investment"
             value="<?php echo $investment; ?>"/><br />
```

The index.php file (continued)

```
        <label>Yearly Interest Rate:</label>
        <input type="text" name="interest_rate"
               value="<?php echo $interest_rate; ?>"/><br />

        <label>Number of Years:</label>
        <input type="text" name="years"
               value="<?php echo $years; ?>"/><br />
      </div>

      <div id="buttons">
        <label>&nbsp;</label>
        <input type="submit" value="Calculate"/><br />
      </div>

    </form>
    </div>
</body>
</html>
```

The display_results.php file

```
<?php
    // get the data from the form
    $investment = $_POST['investment'];
    $interest_rate = $_POST['interest_rate'];
    $years = $_POST['years'];

    // validate investment entry
    if ( empty($investment) ) {
        $error_message = 'Investment is a required field.';
    } else if ( !is_numeric($investment) )  {
        $error_message =
            'Investment must be a valid number.';
    } else if ( $investment <= 0 ) {
        $error_message =
            'Investment must be greater than zero.';
```

The display_results.php file (continued)

```
// validate interest rate entry
} else if ( empty($interest_rate) ) {
    $error_message =
        'Interest rate is a required field.';
} else if ( !is_numeric($interest_rate) )  {
    $error_message =
        'Interest rate must be a valid number.';
} else if ( $interest_rate <= 0 ) {
    $error_message =
        'Interest rate must be greater than zero.';
```

The display_results.php file (continued)

```
// if no invalid entries,
// set error message to empty string
} else {
    $error_message = '';
}

// if an error message exists, go to the index page
if ($error_message != '') {
    include('index.php');
    exit(); }

// calculate the future value
$future_value = $investment;
for ($i = 1; $i <= $years; $i++) {
    $future_value =
        ($future_value +
            ($future_value * $interest_rate * .01));
}
```

The display_results.php file (continued)

```
    // apply currency and percent formatting
    $investment_f = '$'.number_format($investment, 2);
    $yearly_rate_f = $interest_rate.'%';
    $future_value_f = '$'.number_format($future_value, 2);
?>
```

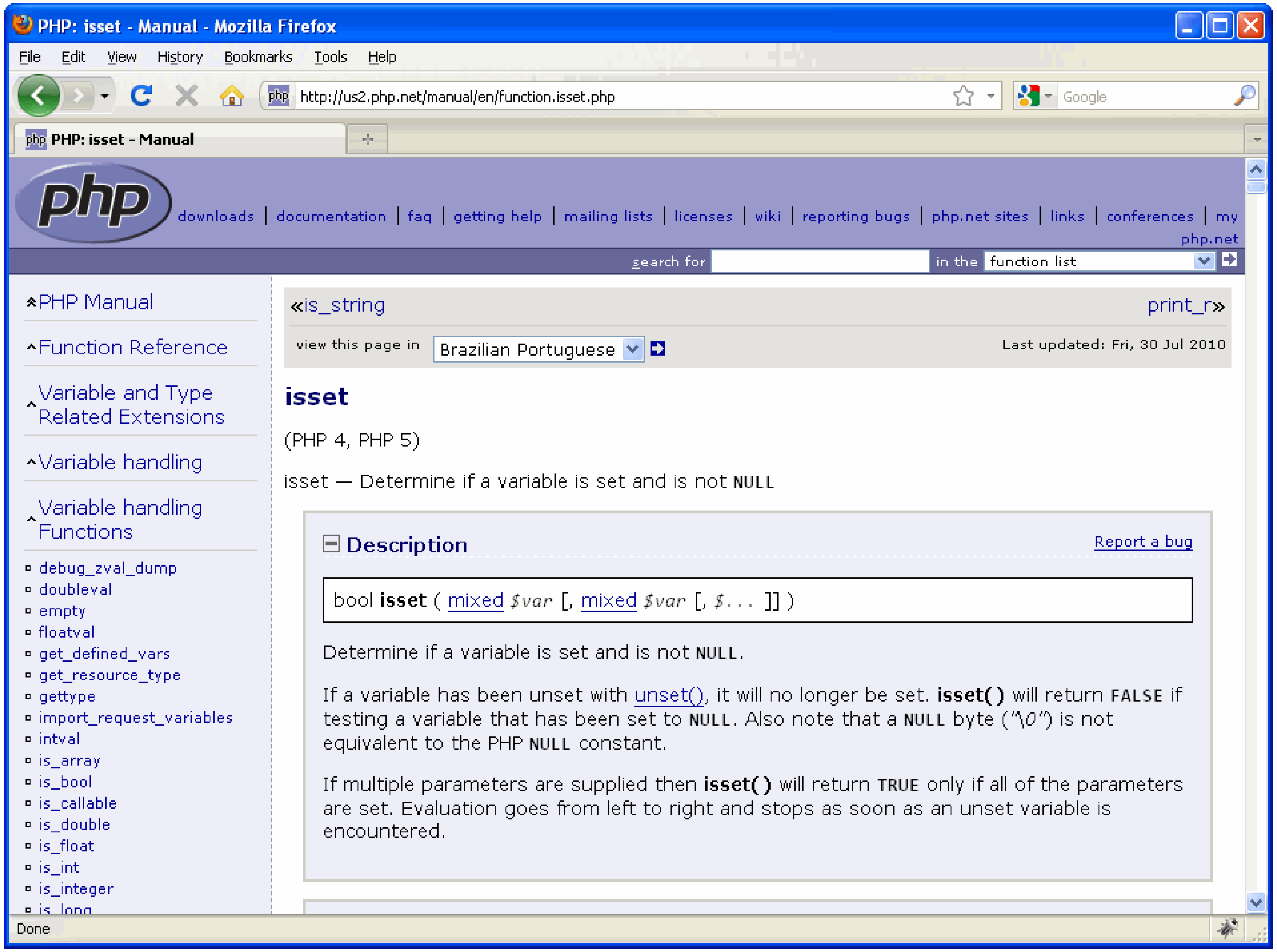
The display_results.php file (continued)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional
    ...>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Future Value Calculator</title>
    <link rel="stylesheet" type="text/css"
href="main.css"/>
</head>
<body>
    <div id="content">
        <h1>Future Value Calculator</h1>
        <label>Investment Amount:</label>
        <span><?php echo $investment_f; ?></span><br />
        <label>Yearly Interest Rate:</label>
        <span><?php echo $yearly_rate_f; ?></span><br />
        <label>Number of Years:</label>
        <span><?php echo $years; ?></span><br />
        <label>Future Value:</label>
        <span><?php echo $future_value_f; ?></span><br />
    </div>
</body>
```

The URL for the PHP documentation

`http://php.net/docs.php`

Documentation for the if statement


PHP: isset - Manual - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://us2.php.net/manual/en/function.isset.php
PHP: isset - Manual
downloads | documentation | faq | getting help | mailing lists | licenses | wiki | reporting bugs | php.net sites | links | conferences | my php.net
search for in the function list
PHP Manual
Function Reference
Variable and Type Related Extensions
Variable handling
Variable handling Functions
debug_zval_dump
doubleval
empty
floatval
get_defined_vars
get_resource_type
gettype
import_request_variables
intval
is_array
is_bool
is_callable
is_double
is_float
is_int
is_integer
is_long
«is_string
print_r»
view this page in Brazilian Portuguese
Last updated: Fri, 30 Jul 2010
Report a bug
Done

isset

(PHP 4, PHP 5)

isset — Determine if a variable is set and is not NULL

Description

```
bool isset ( mixed $var [, mixed $var [, $... ]] )
```

Determine if a variable is set and is not NULL.

If a variable has been unset with unset(), it will no longer be set. **isset()** will return FALSE if testing a variable that has been set to NULL. Also note that a NULL byte ("\0") is not equivalent to the PHP NULL constant.

If multiple parameters are supplied then **isset()** will return TRUE only if all of the parameters are set. Evaluation goes from left to right and stops as soon as an unset variable is encountered.

How to access the PHP manual

- On the first page of the web site, click on the name of the language that you want to use. That will access the first page of the PHP manual.

How to use the PHP manual

- Click on PHP Manual in the left pane of the window to display the contents for the manual in the main pane.
- Scroll down the contents until you find the link you're looking for, click on it, and continue this process until the right information is displayed.

How to find the documentation for a function when you know its name

- Type the function name in the Search For text box and press the Enter key.