Chapter 13

How to create and use functions

Objectives

Applied

1. Create any of the functions that your applications require. These functions may need to pass arguments by value or reference, provide default values for arguments, or provide for a variable number of arguments.
2. Call any of the functions that your applications require.
3. Create and use function libraries and namespaces.

Objectives (continued)

Knowledge

1. Describe the creation and use of functions.
2. Distinguish between passing an argument by value and passing an argument by reference.
3. Describe local scope and global scope as it applies to the variables within functions, and describe the scope of functions themselves.
4. Describe the use of function libraries and namespaces.

Terms

function

parameter

parameter list

return statement

argument

argument list

function call

calling a function

The syntax for a function

```
function function_name([$param_1,
                        $param_2, ... ,
                        $param_n]) {
    // Code for function
    [return [value];]
}
```

A function with no parameters that returns a value

```
function coin_toss() {
    $coin = (mt_rand(0, 1) == 0) ? 'heads' : 'tails';
    return $coin;
}
```

Calling this function:

```
echo coin_toss();          // Displays heads or tails
```

A function with one parameter

```
function display_error($error) {
    echo '<p class="error">' . $error . '</p>';
}
```

A function with three parameters

```
function avg_of_3($x, $y, $z) {
    $avg = ($x + $y + $z) / 3;
    return $avg;
}
```

Calling these functions

```
$average = avg_of_3(5, 2, 8);  // $average is 5

display_error('Value out of range.');
```

An argument passed by value

```
function add_3_by_val($value) {
    $value += 3;
    echo '<p>Number: ' . $value . '</p>';
}

$number = 5;
add_3_by_val($number);                          // Displays 8
echo '<p>Number: ' . $number . '</p>';          // Displays 5
```

An argument passed by reference

```
function add_3_by_ref(&$value) {
    $value += 3;
    echo '<p>Number: ' . $value . '</p>';
}

$number = 5;
add_3_by_ref($number);                          // Displays 8
echo '<p>Number: ' . $number . '</p>';          // Displays 8
```

How to modify a string that's passed by reference:

```
function wrap_in_tag(&$text, $tag) {
    $before = '<'  . $tag . '>';
    $after  = '</' . $tag . '>';
    $text = $before . $text . $after;
}


$message = 'Value out of range.';
wrap_in_tag($message, 'p');
echo $message;              // <p>Value out of range.</p>
```

How to return multiple values

```
function array_analyze($array, &$sum, &$prod, &$avg) {
    $sum = array_sum($array);
    $prod = array_product($array);
    $avg = $sum / count($array);
}

$list = array(1, 4, 9, 16);
array_analyze($list, $s, $p, $a);
echo '<p>Sum: ' . $s . '<br />
         Product: ' . $p . '<br />
         Average ' . $a .
     '</p>';
```

By default, functions do not have access to global variables

A variable with global scope

```
$a = 10;                // $a has global scope
function show_a() {
    echo $a;            // Inside function, $a is NULL
}
show_a();               // Displays nothing
```

How to access a global variable from a function

```
$b = 10;                    // $b has global scope
function show_b() {
    global $b;              // $b refers global variable $b
    echo $b;
}
show_b();                   // Displays 10
```

To access several global variables, use the predefined $GLOBALS array

Like $_GET and $_POST, it is an autoglobal variable available anywhere in the code.

```
$c = 10;                        // $c has global scope
function show_c() {
    $c = $GLOBALS['c'];         // $c refers to global $c
    echo $c;
}
show_c();                       // Displays 10
```

A variable with local scope

```
function show_d() {
    $d = 10;          // $d has local scope within function
    echo $d;
}
echo $d;              // Outside function, $d is NULL
```

When a default value is provided for a parameter, that parameter becomes optional.

A function with one default parameter

```
function get_rand_bool_text($type = 'coin') {
    $rand = mt_rand(0, 1);
    switch ($type) {
        case 'coin':
            $result = ($rand == 1) ? 'heads' : 'tails';
            break;
        case 'switch':
            $result = ($rand == 1) ? 'on' : 'off';
            break;
    }
    return $result;
}
```

Calling a function with a default parameter value

```
echo get_rand_bool_text();

echo get_rand_bool_text('switch');
```

Default parameter values must be scalar values, arrays of scalar values, or NULL

A function with an optional parameter

```
function is_leap_year($date = NULL) {
    if (!isset($date)) {
        $date = new DateTime();
    }
    if ($date->format('L') == '1') return true;
    else return false;
}
```

Calling a function with an optional parameter

```
$is_leap_year = is_leap_year();

$is_leap_year =
    is_leap_year(new DateTime('March 15, 2015'));
```

A function with one required and two default parameters

```
function display_error($error,
                       $tag = 'p',
                       $class = 'error') {
    $opentag  = '<'  . $tag . ' class="' . $class . '">';
    $closetag = '</' . $tag . '>';
    echo $opentag . $error . $closetag;
}
```

Calling the function

```
echo display_error('Out of range');

echo display_error('Out of range', 'li');
```

Functions for working with variable-length parameter lists

```
func_get_args() /*returns an array containing the
arguments passed to the function.*/

func_num_args()

func_get_arg($i)
```

A function that adds a list of numbers

```
function add() {
    $numbers = func_get_args();
    $total = 0;
    foreach($numbers as $number) {
        $total += $number;
    }
    return $total;
}

$sum = add(5, 10, 15);          // $sum is 30
```

A function that averages one or more numbers

```
function average($x) {   // $x forces one argument
    $count = func_num_args();
    $total = 0;
    for($i = 0; $i < $count; $i++) {
        $total += func_get_arg($i);
    }
    return $total / $count;
}

$avg = average(75, 95, 100);   // $avg is 90
```

A library of functions (the cart.php file)

```
<?php
    // Add an item to the cart
    function cart_add_item(&$cart, $name, $cost,
                           $quantity) {
        $total = $cost * $quantity;
        $item = array(
            'name' => $name,
            'cost' => $cost,
            'qty'  => $quantity,
            'total' => $total
        );
        $cart[] = $item;
    }
```

A library of functions (the cart.php file) (cont.)

```
// Update an item in the cart
function cart_update_item(&$cart, $key, $quantity) {
    if (isset($cart[$key])) {
        if ($quantity <= 0) {
            unset($cart[$key]);
        } else {
            $cart[$key]['qty'] = $quantity;
            $total = $cart[$key]['cost'] *
                          $cart[$key]['qty'];
            $cart[$key]['total'] = $total;
        }
    }
}
```

A library of functions (the cart.php file) (cont.)

```
    // Get cart subtotal
    function cart_get_subtotal($cart) {
        $subtotal = 0;
        foreach ($cart as $item) {
            $subtotal += $item['total'];
        }
        $subtotal = round($subtotal, 2);
        $subtotal = number_format($subtotal, 2);
        return $subtotal;
    }
?>
```

Code that uses the library

```
// load the library
require_once('cart.php');

// create an array to store the cart
$cart = array();

// call methods from the library
cart_add_item($cart, 'Flute', 149.95, 1);
cart_update_item($cart, 0, 2);   // update first item
$subtotal = cart_get_subtotal($cart);

// display the result
echo 'This subtotal is $' . $subtotal;
```

Functions for working with the include path

```
get_include_path()
set_include_path($path)
```

The default include path

Windows

```
.;C:\xampp\php\PEAR
```

Mac or Linux

```
.:/Applications/XAMPP/xamppfiles/lib/php/pear
```

How to get the include path

```
$include_path = get_include_path();
```

How to set the include path

Windows

```
set_include_path($include_path .
    ';C:\xampp\htdocs\book_apps\lib');
```

Mac or Linux

```
set_include_path($include_path .
    ':/Applications/XAMPP/htdocs/book_apps/lib');
```

How to include a file after the path has been set

```
require_once cart.php;
```

Function scope in PHP

All functions are global and can be used anywhere

Function names must be unique

A unique prefix will often be used for a set of library functions to avoid naming conflicts:

```
cart_add_item(…)
cart_update_item(…)
cart_get_subtotal(…)
```

But, this can be hard to maintain…

Namespaces: version 5.3

- A group of names not in the global scope
- Allows you to use functions which already have names in the global scope

How to create a namespace in a file

Using the statement syntax

```
<?php
namespace cart;
    // Functions in cart namespace
?>
```

Using the brace syntax

```
<?php
namespace cart {
    // Functions in cart namespace
}
?>
```

How to create nested namespaces

```
<?php
namespace murach\cart {
    // Functions in murach\cart namespace
}
?>
```

How to use the functions in a namespace

Create a file that contains a namespace with one function

```
<?php
namespace murach\errors {
    function log($error) {
        echo '<p class="error">' . $error . '</p>';
    }
}
?>
```

Call a function that is stored in the namespace

```
// load the file that stores the namespace
require_once 'errors.php';

// call the log function
murach\errors\log('Invalid value');

// create an alias and use it to call the log function
use murach\errors as e;
e\log('Invalid value');
```

Variable Functions

Functions whose name is stored in a variable

A variable function

```
$function = (mt_rand(0,1) == 1) ?
    'array_sum' : 'array_product';
$values = array(4, 9, 16);
 $result = $function($values);    /* 29 for array_sum,
 576 for array_product */
```

How variable functions and callbacks work

- A *variable function* is a function name stored in a variable as a string. When PHP encounters a variable function, it evaluates the variable and attempts to call the function.
- To call a variable function, code the variable name followed by a set of parentheses. Within the parentheses, code the argument list for the function.
- You can use a variable function when the function isn't known until runtime.
- You can use a variable function in a function that uses a callback. A *callback* is a function that's passed as an argument to another function.

A function that uses a callback

```
function validate($data, $functions) {
    $valid = true;
    foreach ($functions as $function) {
        $valid = $valid && $function($data);
    }
    return $valid;
}

function is_at_least_18($number) {
    return $number >= 18;
}

function is_less_than_62($number) {
    return $number < 62;
}

$age = 25;
$functions = array(
    'is_numeric', 'is_at_least_18', 'is_less_than_62');
$is_valid_age = validate($age, $functions);    // TRUE
```

Language constructs that can't be used in variable functions

```
die
eval
list
print
echo
include
require
unset
empty
include_once
require_once
exit
isset
return
```

A function for sorting an array with a custom comparison function

```
usort($array, $function)
```

- Built in PHP function
- Sorts an array according to a user-defined comparison function
- The comparison function you create must accept two arguments
- It must return:

  -1 if the arguments are in the correct order

  1 if they are not in the correct order

  0 of they are equal

How to create and use an anonymous function

A custom sorting function

```
$compare_function = function ($left, $right) {
    $l = (float) $left;
    $r = (float) $right;
    if ($l < $r) return -1;
    if ($l > $r) return 1;
    return 0;
};
```

Code that tests the custom sorting function

```
$a = 3;
$b = 5;
$result = $compare_function($a, $b);  // -1
```

Code that uses the custom sorting function

```
$values = array(5, 2, 4, 1, 3);
usort($values, $compare_function);    // 1, 2, 3, 4, 5
```

An anonymous function is similar to a variable function, but it is never given a name in the global namespace.

An array of arrays

```
$employees = array (
    array('name' => 'Ray',  'id' => 5685),
    array('name' => 'Mike', 'id' => 4302),
    array('name' => 'Anne', 'id' => 3674),
    array('name' => 'Pren', 'id' => 1527),
    array('name' => 'Joel', 'id' => 6256)
);
```

A function to sort the array by any column

```
function array_compare_factory($sort_key) {
    return function ($left, $right) use ($sort_key) {
        if ($left[$sort_key] < $right[$sort_key]) {
            return -1;
        } else if ($left[$sort_key] >
                        $right[$sort_key]) {
            return 1;
        } else {
            return 0;
        }
    };
}
```

Code that sorts the array by the name column

```
$sort_by_name = array_compare_factory('name');
usort($employees, $sort_by_name);
```

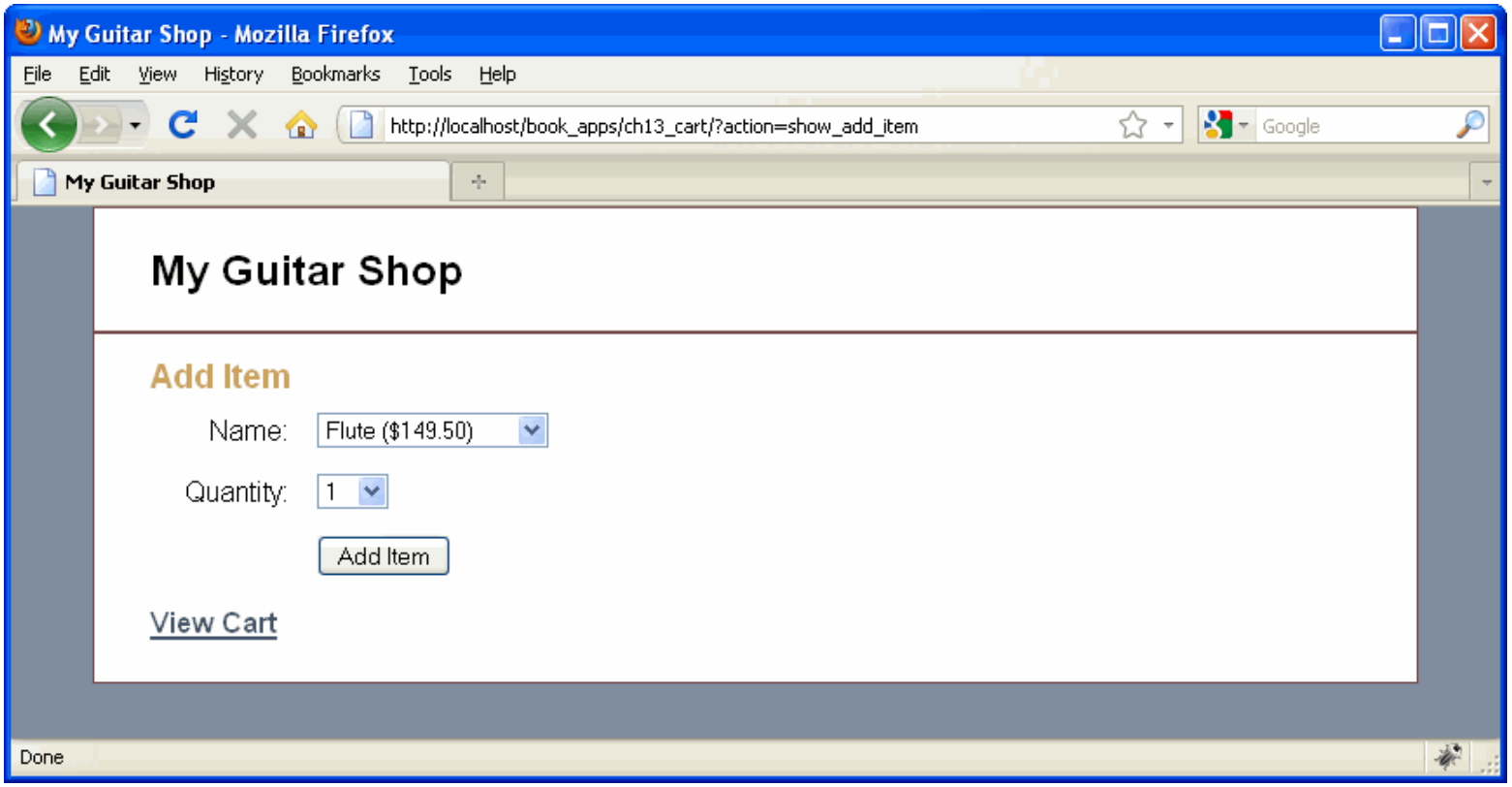
Code that sorts the array by the id column

```
$sort_by_id = array_compare_factory('id');
usort($employees, $sort_by_id);
```

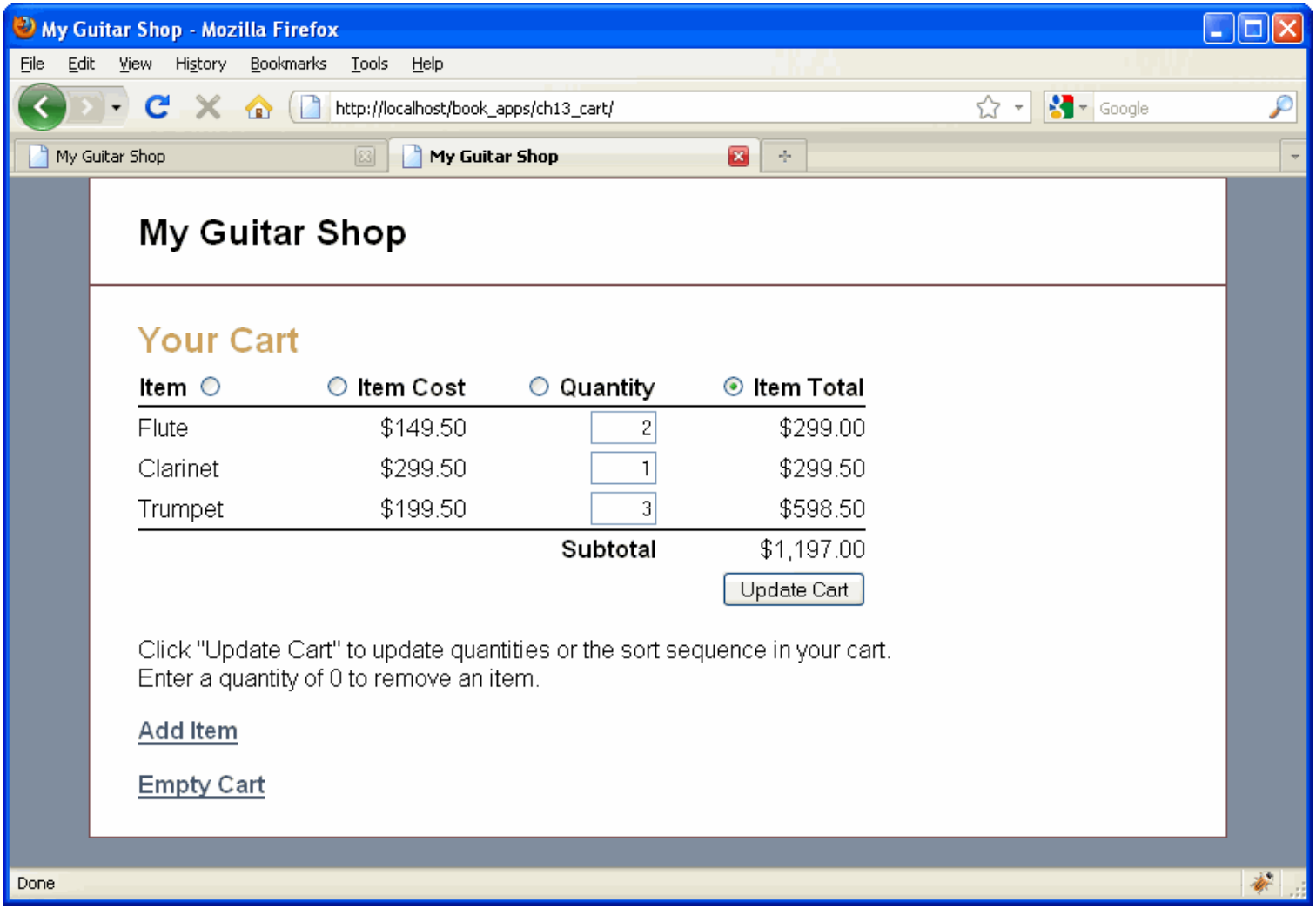
How closures work

- A *closure* is an inner function that has access to the outer function's variables. To create a closure, code a use clause in the inner function.
- To allow the inner function to change the outer function's variable, use the reference operator (&) in the use clause.
- The outer function's variables are available after it has finished executing as long as there is a reference to the inner function.
- The inner function is an anonymous function that is returned by the outer function or stored in a parameter that was passed by reference. You can store it in a variable and call it as a variable function like you would an anonymous function.

The Add Item page



The Cart page



The cart.php file

```
<?php
namespace cart {

    // Add an item to the cart
    function add_item($key, $quantity) {
        // Same code as cart application for chapter 12
    }

    // Update an item in the cart
    function update_item($key, $quantity) {
        // Same code as cart application for chapter 12
    }

    // Get cart subtotal
    function get_subtotal () {
        // Same code as cart application for chapter 12
    }
```

The cart.php file (continued)

```
// Get a function for sorting the cart on the specified key
    function compare_factory($sort_key) {
        return function($left, $right) use ($sort_key) {
            if ($left[$sort_key] == $right[$sort_key]) {
                return 0;
            } else if ($left[$sort_key] <
                       $right[$sort_key]) {
                return -1;
            } else {
                return 1;
            }
        };
    }

// Sort the cart on the specified key
    function sort($sort_key) {
        $compare_function = compare_factory($sort_key);
        uasort($_SESSION['cart13'], $compare_function);
    }
}
?>
```

The index.php file

```
<?php
// Start session management
session_start();

// Create a cart array if needed
if (empty($_SESSION['cart13'])) $_SESSION['cart13'] =
    array();

// Create a table of products
$products = array();
$products['MMS-1754'] = array('name' => 'Flute',
                              'cost' => '149.50');
$products['MMS-6289'] = array('name' => 'Trumpet',
                              'cost' => '199.50');
$products['MMS-3408'] = array('name' => 'Clarinet',
                              'cost' => '299.50');

// Include cart functions
require_once('cart.php');
```

The index.php file (continued)

```
// Get the sort key
if (isset($_POST['sortkey'])) {
    $sort_key = $_POST['sortkey'];
} else {
    $sort_key = 'name';
}

// Get the action to perform
if (isset($_POST['action'])) {
    $action = $_POST['action'];
} else if (isset($_GET['action'])) {
    $action = $_GET['action'];
} else {
    $action = 'show_add_item';
}
```

The index.php file (continued)

```
// Add or update cart as needed
switch($action) {
    case 'add':
        cart\add_item($_POST['productkey'],
            $_POST['itemqty']);
        include('cart_view.php');
        break;
    case 'update':
        $new_qty_list = $_POST['newqty'];
        foreach($new_qty_list as $key => $qty) {
            if ($_SESSION['cart13'][$key]['qty'] != $qty) {
                cart\update_item($key, $qty);
            }
        }
        cart\sort($sort_key);
        include('cart_view.php');
        break;
    case 'show_cart':
        cart\sort($sort_key);
        include('cart_view.php');
        break;
```

The index.php file (continued)

```
    case 'show_add_item':
        include('add_item_view.php');
        break;
    case 'empty_cart':
        unset($_SESSION['cart13']);
        include('cart_view.php');
        break;
}
?>
```

The cart_view.php file

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 ...>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>My Guitar Shop</title>
    <link rel="stylesheet" type="text/css" href="main.css"/>
</head>
<body>
<div id="page">
    <div id="header">
        <h1>My Guitar Shop</h1>
    </div>
    <div id="main">
    <h1>Your Cart</h1>
    <?php if (count($_SESSION['cart13']) == 0) : ?>
        <p>There are no items in your cart.</p>
    <?php else: ?>
```

The cart_view.php file (continued)

```
<form action="." method="post">
    <input type="hidden" name="action"
           value="update"/>
    <table>
        <tr id="cart_header">
            <th class="left">
                Item <input type="radio"
                <?php if ($sort_key == 'name') : ?>
                    checked="checked"
                <?php endif; ?>
                name="sortkey" value="name"/></th>
            <th class="right">
                <input type="radio"
                <?php if ($sort_key == 'cost') : ?>
                    checked="checked"
                <?php endif; ?>
                    name="sortkey" value="cost"/>
                Item Cost</th>
```

The cart_view.php file (continued)

```
            <th class="right" >
                <input type="radio"
                <?php if ($sort_key == 'qty') : ?>
                    checked="checked"
                <?php endif; ?>
                    name="sortkey" value="qty"/>
                Quantity</th>
            <th class="right">
                <input type="radio"
                <?php if ($sort_key == 'total') : ?>
                   checked="checked"
                <?php endif; ?>
                    name="sortkey" value="total"/>
            Item Total</th>
        </tr>
```

The cart_view.php file (continued)

```
<?php foreach( $_SESSION['cart13']
          as $key => $item ) :
    $cost =
        number_format($item['cost'],  2);
    $total =
        number_format($item['total'], 2);
?>
<tr>
    <td>
        <?php echo $item['name']; ?>
    </td>
    <td class="right">
        $<?php echo $cost; ?>
    </td>
    <td class="right">
        <input type="text" class="cart_qty"
         name=
           "newqty[<?php echo $key; ?>]"
         value=
           "<?php echo $item['qty']; ?>"/>
    </td>
```

The cart_view.php file (continued)

```
                <td class="right">
                    $<?php echo $total; ?>
                </td>
            </tr>
            <?php endforeach; ?>
            <tr id="cart_footer">
                <td colspan="3"><b>Subtotal</b></td>
                <td>$<?php echo cart\get_subtotal(); ?>
                </td>
            </tr>
            <tr>
                <td colspan="4" class="right">
                    <input type="submit"
                           value="Update Cart"
                           id="update_button" />
                </td>
            </tr>
        </table>
```

The cart_view.php file (continued)

```
            <p>Click "Update Cart" to update quantities
               or the sort sequence
               in your cart.<br />Enter a quantity of 0 to
               remove an item.
            </p>
        </form>
    <?php endif; ?>
    <p><a href=".?action=show_add_item">Add Item</a></p>
    <p><a href=".?action=empty_cart">Empty Cart</a></p>

    </div><!-- end main -->
</div><!-- end page -->
</body>
</html>
```