



Chapter 6

Arrays

Arrays in PHP

- To create a new array variable in PHP uses `array()`.
- PHP support two types of arrays:
 1. **Index arrays**- each element is referenced by a **numeric index**, usually starting from zero.
e.g `$authors = array("Steinbeck", "Kafka", "Tolkien", "Dickens");`
 2. **Associative arrays**- is also referred to as a hash or map. With associative arrays, each element is referenced by a **string index**.
e.g `$myBook = array("title" => "The Grapes of Wrath",
"author" => "John Steinbeck", "pubYear" => 1939);`

Accessing Array Elements

Example

- `$authors = array("Steinbeck", "Kafka", "Tolkien", "Dickens");`
`$pos = 2;`
`echo $authors[$pos + 1]; // Displays "Dickens"`
- `$authors = array("Steinbeck", "Kafka", "Tolkien", "Dickens");`
`$authors[2] = "Melville";`
- `$authors = array("Steinbeck", "Kafka", "Tolkien", "Dickens");`
`$authors[] = "Orwell";` //add a new element to the end of the array

Outputting an Entire Array with print_r()



```
< html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" >
< head >
< title > Outputting Arrays with print_r() < /title >
< link rel="stylesheet" type="text/css" href="common.css" / >
< /head >
< body >
< h1 > Outputting Arrays with print_r() < /h1 >
< ?php
$authors = array( "Steinbeck", "Kafka", "Tolkien", "Dickens" );
$myBook = array( "title" => "The Grapes of Wrath",
"author" => "John Steinbeck",
"pubYear" => 1939 );
echo ' < h2 > $authors: < /h2 > < pre > ';
print_r ( $authors );
echo ' < /pre > < h2 > $myBook: < /h2 > < pre > ';
print_r ( $myBook );
echo " < /pre > ";
? >
< /body >
< /html >
```

Outputting Arrays with print_r()

\$authors:




```
Array
(
    [0] => Steinbeck
    [1] => Kafka
    [2] => Tolkien
    [3] => Dickens
)
```

\$myBook:

```
Array
(
    [title] => The Grapes of Wrath
    [author] => John Steinbeck
    [pubYear] => 1939
)
```

Try It Out

- Create a PHP page (exercise6-1.php) that creates an array (\$a) to store the following information of 3 persons by static array() statement, then display the content of \$a on the page by using print_r function. Note that James Bond does not have any family member, but “Family member” must appear in the array as a “key”

 <ul style="list-style-type: none"> ● Name: Mickey Mouse ● Age: 80 ● Nationality: USA ● Family member: <ul style="list-style-type: none"> ➤ Girl friend 	 <ul style="list-style-type: none"> ● Name: Yoichi Kogure ● Age: 49 ● Nationality: Japan ● Family member: <ul style="list-style-type: none"> ➤ Wife ➤ Daughter ➤ Son 	 <ul style="list-style-type: none"> ● Name: James Bond ● Age: 100 ● Nationality: England ● Family member:
--	--	--

Exercise (6-1.php)



```
<?php
$a= array(array('Name'=>"Mickey
    Mouse",'Age'=>80,'Nationality'=>"USA",'Family
    member'=>array('Girl friend')),
    array('Name'=>"Yoichi Kogure",'Age'=>
    49,'Nationality'=>"Japan",'Family
    member'=>array('Wife','Daughter','Son')),
    array('Name'=>"James Bond",'Age'=>
    100,'Nationality'=>"England",'Family member'=>array())) ;
echo "<pre>";
print_r ($a);
echo "</pre>";
?>
```

Extracting a Range of Elements with `array_slice()` and count with `count()`

- `array_slice()` that you can use to extract a range of elements from an array.

```
$authors = array( "Steinbeck", "Kafka", "Tolkien", "Dickens" );  
$authorsSlice = array_slice( $authors, 1, 2 );  
print_r( $authorsSlice );// Displays "Array ( [0] => Kafka [1] => Tolkien )"
```
- `count()` returns the number of elements as an integer:

```
$authors = array( "Steinbeck", "Kafka", "Tolkien", "Dickens" );  
echo count( $authors );// Displays "4"
```

Stepping Through an Array

Function	Description
<code>current()</code>	Returns the value of the current element pointed to by the pointer, without changing the pointer position.
<code>key()</code>	Returns the index of the current element pointed to by the pointer, without changing the pointer position.
<code>next()</code>	Moves the pointer forward to the next element, and returns that element's value.
<code>prev()</code>	Moves the pointer backward to the previous element, and returns that element's value.
<code>end()</code>	Moves the pointer to the last element in the array, and returns that element's value.
<code>reset()</code>	Moves the pointer to the first element in the array, and returns that element's value.

Stepping Through an Array(Cont'd)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en" >
<head >
<title > Stepping Through an Array </title >
<link rel="stylesheet" type="text/css" href="common.css" />
</head >
<body >
<h1 > Stepping Through an Array </h1 >
<?php
$authors = array( "Steinbeck", "Kafka", "Tolkien", "Dickens" );
echo "<p> The array: " . print_r( $authors, true ) . "</p>";
echo "<p> The current element is: " . current( $authors ) . "</p>";
echo "<p> The next element is: " . next( $authors ) . "</p>";
echo "<p> ...and its index is: " . key( $authors ) . "</p>";
echo "<p> The next element is: " . next( $authors ) . "</p>";
echo "<p> The previous element is: " . prev( $authors ) . "</p>";
echo "<p> The first element is: " . reset( $authors ) . "</p>";
echo "<p> The last element is: " . end( $authors ) . "</p>";
echo "<p> The previous element is: " . prev( $authors ) . "</p>";
?>
</body >
</html >
```

Stepping Through an Array

The array: Array ([0] => Steinbeck [1] => Kafka [2] => Tolkien [3] => Dickens)

The current element is: Steinbeck.

The next element is: Kafka.

...and its index is: 1.

The next element is: Tolkien.

The previous element is: Kafka.

The first element is: Steinbeck.

The last element is: Dickens.

The previous element is: Tolkien.

PHP function: each ()

```
$myBook = array( "title" => "The Grapes of
Wrath",
"author" => "John Steinbeck",
"pubYear" => 1939 );
$element = each( $myBook );
echo "Key: " . $element[0] . " < br/ > ";
echo "Value: " . $element[1] . " < br/ > ";
echo "Key: " . $element["key"] . " < br/ > ";
echo "Value: " . $element["value"] . " < br/ > ";
```

Element Index	Element Value
0	The current element's key
"key"	The current element's key
1	The current element's value
"value"	The current element's value

This code displays:

Key: title

Value: The Grapes of Wrath

Key: title

Value: The Grapes of Wrath

PHP function: each () (Cont'd)



```
< !DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >
< html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" >
< head >
< title > Using each() with a while loop < /title >
< link rel="stylesheet" type="text/css" href="common.css" / >
< /head >
< body >
< h1 > Using each() with a while loop < /h1 >
< dl >
< ?php
$myBook = array( "title" => "The Grapes of Wrath",
"author" => "John Steinbeck", "pubYear" => 1939 );
while ( $element = each( $myBook ) ) {
echo "< dt > $element[0] < /dt > ";
echo "< dd > $element[1] < /dd > ";
}
? >
< /dl >
< /body >
< /html >
```

Using each() with a while loop

title	The Grapes of Wrath
author	John Steinbeck
pubYear	1939

Using for each to Loop Through Values

```
$authors = array( "Steinbeck", "Kafka", "Tolkien", "Dickens" );  
foreach ( $authors as $val ) {  
    echo $val . " < br/ > ";  
}
```

Output:

Steinbeck

Kafka

Tolkien

Dickens

Using foreach to Loop Through Keys and Values

```
< body >
< h1 > Using foreach < /h1 >
< dl >
< ?php
$myBook = array( "title" => "The Grapes of Wrath", "author" => "John
Steinbeck", "pubYear" => 1939 );
foreach ( $myBook as $key => $value ) {
echo " < dt > $key < /dt > ";
echo " < dd > $value < /dd > ";
}
? >
< /dl >
< /body >
< /html >
```

Using foreach

```
title
    The Grapes of Wrath
author
    John Steinbeck
pubYear
    1939
```

Working with Multidimensional Arrays



```
<?php
$myBooks = array(
array(
"title" => "The Grapes of Wrath",
"author" => "John Steinbeck",
"pubYear" => 1939
),
array(
"title" => "The Trial",
"author" => "Franz Kafka",
"pubYear" => 1925
),
array(
"title" => "The Hobbit",
"author" => "J. R. R. Tolkien",
"ubYear" => 1937
),
);
echo "<pre>";
print_r ( $myBooks );
echo "</pre>";
?>
```

```
Array
(
    [0] => Array
        (
            [title] => The Grapes of Wrath
            [author] => John Steinbeck
            [pubYear] => 1939
        )
    [1] => Array
        (
            [title] => The Trial
            [author] => Franz Kafka
            ["pubYear"] => 1925
        )
    [2] => Array
        (
            [title] => The Hobbit
            [author] => J. R. R. Tolkien
            [ubYear] => 1937
        )
)
```

Try It Out

- Suppose we receive the array (\$a) in question 1 as a result from external program, so that the content (such as number of persons, name, number of family members, etc.) will change from time to time (i.e. variable). Now, create a PHP page (exercise6-2.php) and copy all the PHP code from exercise6-1.php. Then, add PHP code that programmatically creates another array (\$b) (not by static array() statement) that has the following format of information from the give array (\$a).



● Name: Mickey Mouse
● Number of family members: 1



● Name: Yoichi Kogure
● Number of family members: 3



● Name: James Bond
● Number of family members: 0

Exercise (6-2.php)

```
<?php
$a= array(array('Name'=>"Mickey Mouse", 'Age'=>80, 'Nationality'=>"USA",
    'Family member'=>array(0=>'Girl friend')),
    array('Name'=>"Yoichi Kogure",'Age'=> 49,'Nationality'=>"Japan",'Family
    member'=>array(0=>'Wife',1=>'Daughter',2=>'Son')),
    array('Name'=>"James Bond",'Age'=> 100, 'Nationality'=>"England",
    'Family member'=>array()));
foreach($a as $key => $value) {
    $b[$key]["Name"]=$a[$key]["Name"];
    $b[$key]['Number of family members']=count($a[$key]['Family member']);
}
echo "<pre>";
print_r ($b);
echo "</pre>";
?>
```


Sorting Arrays

- `sort()` and `rsort()` – For sorting indexed arrays
- `asort()` and `arsort()` – For sorting associative arrays
- `ksort()` and `krsort()` – For sorting associative arrays **by key** rather than by value
- `array_multisort()` – A powerful function that can sort multiple arrays at once, or multidimensional arrays

Sorting Arrays (Examples)



```
$authors = array( "Steinbeck", "Kafka", "Tolkien", "Dickens" );
```

```
sort( $authors );
```

```
print_r( $authors );
```

```
// Displays "Array ( [0] => Dickens [1] => Kafka [2] =>  
Steinbeck [3] => Tolkien )"
```

```
rsort( $authors );
```

```
print_r( $authors );
```

```
// Displays "Array ( [0] => Tolkien [1] => Steinbeck [2] =>  
Kafka [3] => Dickens )"
```

Sorting Arrays(Example)



```
$myBook = array("title"=> "Bleak House","author"=> "Dickens","year" => 1853 );
```

```
asort( $myBook );
```

```
print_r( $myBook );
```

```
// Displays "Array ( [title] => Bleak House [author] => Dickens [year] => 1853 )"
```

```
arsort( $myBook );
```

```
print_r( $myBook );
```

```
// Displays "Array ( [year] => 1853 [author] => Dickens [title] => Bleak House )"
```

```
$myBook = array( "title" => "Bleak House","author" => "Dickens","year" => 1853 );
```

```
ksort( $myBook );
```

```
print_r( $myBook );
```

```
// Displays "Array ( [author] => Dickens [title] => Bleak House [year] => 1853 )"
```

Sorting Arrays (Example)



```
krsort( $myBook );
```

```
print_r( $myBook );
```

```
// Displays “Array ( [year] => 1853 [title] => Bleak House  
[author] => Dickens )”
```

- `array_multisort()` lets you sort multiple related arrays at the same time
- `array_multisort()` sorts a two-dimensional array.
 - `array_multisort($array1, $array2, ...);`

Multi- sort (Example)



```
<?php
$myBooks = array(
    array(
        "title" => "The Grapes of Wrath",
        "author" => "John Steinbeck",
        "pubYear" => 1939
    ),
    array(
        "title" => "Travels With Charley",
        "author" => "John Steinbeck",
        "pubYear" => 1962
    ),
    array(
        "title" => "The Trial",
        "author" => "Franz Kafka",
        "pubYear" => 1925
    ),
    array(
        "title" => "The Hobbit",
        "author" => "J. R. R. Tolkien",
        "pubYear" => 1937
    ),
    array(
```

```
        "title" => "A Tale of Two Cities",
        "author" => "Charles Dickens",
        "pubYear" => 1859
    ),
);
array_multisort( $myBooks );
echo "<pre>";
print_r( $myBooks );
echo "</pre>";

?>
```

Adding and Removing Array Elements



- `array_unshift()` – Adds one or more new elements to the start of an array
- `array_shift()`– Removes the first element from the start of an array
- `array_push()`— Adds one or more new elements to the end of an array
- `array_pop()`— Removes the last element from the end of an array
- `array_splice()`— Removes element(s) from and/or adds element(s) to any point in an array

Adding and Removing Array Elements

```
$authors = array( "Steinbeck", "Kafka", "Tolkien", "Dickens" );
echo array_unshift( $authors, "Hardy", "Melville" ) . " < br/ > "; // Displays "6"
print_r( $authors ); // Displays "Array ( [0] => Hardy [1] => Melville [2] => Steinbeck [3] => Kafka [4] => Tolkien [5] => Dickens )"
```

```
$myBook = array( "title" => "The Grapes of Wrath", "author" => "John Steinbeck", "pubYear" => 1939 );
echo array_shift( $myBook ) . " < br/ > "; // Displays "The Grapes of Wrath"
print_r( $myBook ); // Displays "Array ( [author] => John Steinbeck [pubYear] => 1939 )"
```

```
$authors = array( "Steinbeck", "Kafka", "Tolkien", "Dickens" );
echo array_push( $authors, "Hardy", "Melville" ) . " < br/ > "; // Displays "6"
print_r( $authors ); // Displays "Array ( [0] => Steinbeck [1] => Kafka [2] => Tolkien [3] => Dickens [4] => Hardy [5] => Melville )"
```

Adding and Removing Array Elements



```
$myBook = array( "title" => "The Grapes of Wrath", "author" => "John  
Steinbeck", "pubYear" => 1939 );  
  
echo array_pop( $myBook ) . " < br/ > "; // Displays "1939"  
  
print_r( $myBook ); // Displays "Array ( [title] => The Grapes of Wrath  
[author] => John Steinbeck )"  
  
$authors = array( "Steinbeck", "Kafka", "Tolkien" );  
  
array_splice( $authors, 1, 0, array( "authorName" => "Milton" ) );  
  
print_r( $authors ); // Display Array ( [0] => Steinbeck [1] => Milton [2] =>  
Kafka [3] => Tolkien )
```


Merging Arrays Together

```
$authors = array( "Steinbeck", "Kafka" );  
$moreAuthors = array( "Tolkien", "Milton" );  
  
print_r( array_merge( $authors, $moreAuthors ) ); // Displays "Array ( [0] => Steinbeck [1] => Kafka [2] => Tolkien [3] => Milton )"  
  
$authors = array( "Steinbeck", "Kafka" );  
$moreAuthors = array( "Tolkien", "Milton" );  
array_push( $authors, $moreAuthors );  
  
print_r( $authors ); // Displays "Array ( [0] => Steinbeck [1] => Kafka [2] => Array ( [0] => Tolkien [1] => Milton ) )"  
  
$myBook = array( "title" => "The Grapes of Wrath", "author" => "John Steinbeck", "pubYear" => 1939 );  
$myBook = array_merge( $myBook, array( "numPages" => 464 ) );  
print_r ( $myBook ); // Displays "Array ( [title] => The Grapes of Wrath [author] => John Steinbeck [pubYear] => 1939 [numPages] => 464"
```

Converting Between Arrays and Strings

- To convert a string to an array, PHP uses **explode()** string function.

```
<?php
```

```
$fruitString = "apple,pear,banana,strawberry,peach";
```

```
$fruitArray = explode( ",", $fruitString );
```

```
echo $fruitArray[0]."<br>";// apple
```

```
echo $fruitArray[4]; //peach
```

```
?>
```

- To convert an array to a string, PHP uses **implode()** string function

```
$fruitArray = array( "apple", "pear", "banana", "strawberry", "peach" );
```

```
$fruitString = implode( ",", $fruitArray );
```

```
echo $fruitString; // Displays "apple,pear,banana,strawberry,peach"
```

Converting an Array to a List of Variables

```
$myBook = array( "The Grapes of Wrath", "John Steinbeck", 1939 );
```

```
list( $title, $author, $pubYear ) = $myBook;
```

```
echo $title . " < br/ > "; // Displays "The Grapes of Wrath"
```

```
echo $author . " < br/ > "; // Displays "John Steinbeck"
```

```
echo $pubYear . " < br/ > "; // Displays "1939"
```

-A classic use of `list()` is with functions such as `each()` that return an indexed array of values.

```
$myBook = array( "title" => "The Grapes of Wrath", "author" => "John  
Steinbeck", "pubYear" => 1939 );
```

```
while ( list( $key, $value ) = each( $myBook ) ) {
```

```
echo " < dt > $key < /dt > ";
```

```
echo " < dd > $value < /dd > ";
```

```
}
```

```
title
    The Grapes of Wrath
author
    John Steinbeck
pubYear
    1939
```

Exercises

1. Imagine that two arrays containing book and author information have been pulled from a database:

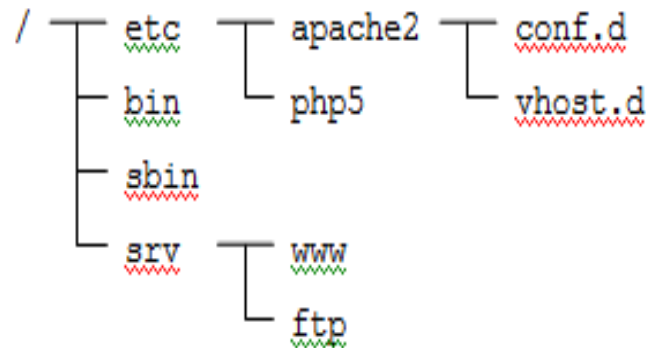
```
$authors = array( "Steinbeck", "Kafka", "Tolkien", "Dickens", "Milton", "Orwell" );  
  
$books = array(  
    array("title" => "The Hobbit", "authorId" => 2, "pubYear" => 1937),  
    array("title" => "The Grapes of Wrath", "authorId" => 0, "pubYear" => 1939),  
    array("title" => "A Tale of Two Cities", "authorId" => 3, "pubYear" => 1859),  
    array("title" => "Paradise Lost", "authorId" => 4, "pubYear" => 1667),  
    array("title" => "Animal Farm", "authorId" => 5, "pubYear" => 1945),  
    array("title" => "The Trial", "authorId" => 1, "pubYear" => 1925));
```

Exercises

1. Instead of containing author names as strings, the \$booksarray contains numeric indices (keyed on “ authorId “) pointing to the respective elements of the \$authors array. Write a script to add an “ authorName ” element to each associative array within the \$books array that contains the author name string pulled from the \$authors array. Display the resulting \$books array in a Web page.
2. Imagine you are writing a version of the computer game Minesweeper. Use arrays to create and store a minefield on a 20 x 20 grid. Place ten mines randomly on the grid, then display the grid, using asterisks (*) for the mines and periods (.) for the empty squares. (Hint: To return a random number between 0 and 19 inclusive, use rand(0, 19).)

Assignment

- (1) Write PHP code that receives the values of rows and cols and outputs a HTML table depending upon the received values (rows and cols).
- (2) Below is a part of directory structure in Linux server. Create a PHP page that creates an array (\$c) to store this structure. You should also show the content of the array by using print_r function.





Chapter 7

Functions

What Is a Function?

- A **function** also called a **subroutine** in some other languages and it contains block of code that performs a specific task.
- A **function** often accepts one or more arguments, which are values passed to the function by the code that calls it.
- A **function** may also optionally return a value that can then be read by the calling code.

Example

- `functionName(argument1, argument2);`
- `$returnVal = functionName(argument);`
- `print(functionName(argument));`

Calling Functions

```
< !DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >
< html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" >
< head >
< title > Square roots < /title >
< link rel="stylesheet" type="text/css" href="common.css" / >
< /head >
< body >
< h1 > Square roots < /h1 >
< ?php
echo "The square root of 9 is: " . sqrt( 9 ) . ". < br/ > ";
echo "All done! < br/ > ";
? >
< /body >
< /html >
```

Square roots
The square root of
9 is: 3.
All done!

Working with Variable Functions

Example:

```
$squareRoot = "sqrt";  
echo "The square root of 9 is: " . $squareRoot( 9 ) . " . <br/ > ";  
echo "All done! <br/ > ";
```

Example:

```
$trigFunctions = array( "sin", "cos", "tan" );  
$degrees = 30;  
foreach ( $trigFunctions as $trigFunction ) {  
echo "$trigFunction($degrees) = " . $trigFunction( deg2rad( $degrees ) ) .  
" <br/ > ";  
}
```

Output:

```
sin(30) = 0.5  
cos(30) = 0.866025403784  
tan(30) = 0.57735026919
```

Writing Your Own Functions

```
< !DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >
< html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" >
< head >
< title > Saying hello with style < /title >
< link rel="stylesheet" type="text/css" href="common.css" / >
< /head >
< body >
< h1 > Saying hello with style < /h1 >
< ?php
function helloWithStyle( $font, $size ) {
echo " < p style=\"font-family: $font; font-size: {$size}em;\" > Hello, world! < /p > ";
}
helloWithStyle( "Helvetica", 2 );
helloWithStyle( "Times", 3 );
helloWithStyle( "Courier", 1.5 );
? >
< /body >
< /html >
```

Saying hello with style

Hello, world!

Hello, world!

Hello, world!

Optional Parameters and Default Values



```
< !DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >
< html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" >
< head >
< title > Saying hello with style < /title >
< link rel="stylesheet" type="text/css" href="common.css" / >
< /head >
< body >
< h1 > Saying hello with style < /h1 >
< ?php
function helloWithStyle( $font, $size=1.5 ) {
echo " < p style=\"font-family: $font; font-size: {$size}em;\" > Hello, world! < /p > ";
}
helloWithStyle( "Helvetica", 2 );
helloWithStyle( "Times", 3 );
helloWithStyle( "Courier" );
? >
< /body >
< /html >
```

Returning Values from Your Functions



```
< !DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >
< html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" >
< head >
< title > Normal and bold text < /title >
< link rel="stylesheet" type="text/css" href="common.css" / >
< /head >
< body >
< h1 > Normal and bold text < /h1 >
< ?php
function makeBold( $text ) {
return " < b > $text < /b > ";
}
$normalText = "This is normal text.";
$boldText = makeBold( "This is bold text." );
echo " < p > $normalText < /p > ";
echo " < p > $boldText < /p > ";
? >
< /body >
< /html >
```

Normal and bold text

This is normal text.

This is bold text.

Understanding Variable Scope

```
< !DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >
< html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" >
< head >
< title > Understanding variable scope < /title >
< link rel="stylesheet" type="text/css" href="common.css" / >
< /head >
< body >
< h1 > Understanding variable scope < /h1 >
< ?php
function helloWithVariables() {
$hello = "Hello, ";
$world = "world!";
return $hello . $world;
}
echo helloWithVariables() . " < br/ > ";
echo "The value of \$hello is: '$hello' < br/ > ";
echo "The value of \$world is: '$world' < br/ > ";
? >
< /body >
< /html >
```

Understanding variable scope

```
Hello, world!
The value of $hello is: "
The value of $world is: "
```

Working with Global Variables

```
function setup() {  
  
  global $myGlobal;  
  
  $myGlobal = “Hello there!”;  
  
}  
  
function hello() {  
  
  global $myGlobal;  
  
  echo “$myGlobal < br/ > ”;  
  
}  
  
setup();  
  
hello(); // Displays “Hello there!”
```

Using Static Variables to Preserve Values

```
function nextNumber() {  
    static $counter = 0;  
    return ++$counter;  
}  
  
echo "I've counted to: " . nextNumber() . " < br/ > ";  
echo "I've counted to: " . nextNumber() . " < br/ > ";  
echo "I've counted to: " . nextNumber() . " < br/ > ";
```

the code displays:

I've counted to: 1

I've counted to: 2

I've counted to: 3

Creating Anonymous Functions

- PHP lets you create **anonymous functions** — that is, functions that have no name.

Example:

```
$mode = “+”;
```

```
$processNumbers = create_function( ‘$a, $b’, “return \ $a  
$mode \ $b;” );
```

```
echo $processNumbers( 2, 3 ); // Displays “5”
```

Try It Out

- This example script takes a block of text and sorts all the words within the text by the number of letters in each word, shortest word first. To do this, it uses anonymous functions, along with PHP's built-in `usort()`, `preg_replace()`, `array_unique()`, and `preg_split()` functions.
- Save the script as `sort_words_by_length.php` in your document root folder, then run it in your browser.

Here document (HereDoc)

- Can use here document or HereDoc to escape PHP code directly.
- Start your here document with “<<<” followed by a word (such as HERE).
- You must end your here document by a line that immediately begins with the word that you started the here document.

```
<html><body>
<h1>Sorting words in a block of text by length</h1>
<?php
$myText = <<<END_TEXT
But think not that this famous town has only harpooners,
cannibals, and bumpkins to show her visitors. Not at all.
Still New Bedford is a queer place. Had it not been for us
whalemen, that tract of land would this day perhaps have
been in as howling condition as thecoast of Labrador.
END_TEXT;
echo "<h2>The text:</h2>";
echo "<div style='width: 30em;'>$myText</div>";
$myText = preg_replace( "/[\.,.]/", "", $myText );
$words = array_unique( preg_split( "/[ \n\r\t]+/", $myText )
);
usort( $words, create_function( '$a, $b', 'return strlen($a) -
strlen($b);
' ) );
echo "<h2>The sorted words:</h2>";
echo "<div style='width: 30em;'>";
foreach ( $words as $word ) {
echo "$word ";
}
echo "</div>";
?>
</body> </html>
```

Sorting words in a block of text by length

The text:

But think not that this famous town has only harpooners, cannibals, and bumpkins to show her visitors. Not at all. Still New Bedford is a queer place. Had it not been for us whalemen, that tract of land would this day perhaps have been in as howling condition as the coast of Labrador.

The sorted words:

a il al us of as in to is But New all for day the not has Had Not her and this that land only have been show town coast would Still tract queer think place famous perhaps howling Bedford Labrador whalemen visitors bumpkins cannibals condition harpooners

Passing References to Your Own Functions

```
function resetCounter( &$c ) {  
  
    $c = 0;  
  
}  
  
$counter = 0;  
  
$counter++;  
  
$counter++;  
  
$counter++;  
  
echo "$counter < br/ > "; // Displays “3”  
  
resetCounter( $counter );  
  
echo "$counter < br/ > "; // Displays “0”
```

Returning References from Your Own Functions

```
$myNumber = 5;  
  
function &getMyNumber() {  
  
    global $myNumber;  
  
    return $myNumber;  
  
}  
  
$numberRef = &getMyNumber();  
  
$numberRef++;  
  
echo "\$myNumber = $myNumber < br/ > "; // Displays “6”  
  
echo "\$numberRef = $numberRef < br/ > "; // Displays “6”
```

Creating the Fibonacci Sequence with Recursion

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Fibonacci sequence using recursion</title>
<link rel="stylesheet" type="text/css" href="common.css" />
<style type="text/css">
th { text-align: left; background-color: #999; }
th, td { padding: 0.4em; }
tr.alt td { background: #ddd; }
</style>
</head>
<body>
<h2>Fibonacci sequence using recursion</h2>
<table cellpadding="0" border="0" style="width: 20em; border: 1px solid #666;">
<tr>
<th>Sequence #</th>
<th>Value</th>
</tr>
```

Creating the Fibonacci Sequence with Recursion (Cont'd)

```
<?php
$iterations = 10;
function fibonacci( $n ) {
if ( ( $n == 0 ) || ( $n == 1 ) ) return $n;
return fibonacci( $n-2 ) + fibonacci( $n-1 );
}
for ( $i=0; $i <= $iterations; $i++ )
{
?>
<tr?>?php if ( $i % 2 != 0 ) echo ' class="alt"' ?>>
<td>F<sub><?php echo $i?></sub></td>
<td><?php echo fibonacci( $i )?></td>
</tr>
<?php
}
?>
</table>
</body>
</html>
```

Fibonacci sequence using recursion

Sequence #	Value
F ₀	0
F ₁	1
F ₂	1
F ₃	2
F ₄	3
F ₅	5
F ₆	8
F ₇	13
F ₈	21
F ₉	34
F ₁₀	55

Exercises

1. Write a function that takes an array argument, and returns a string containing XHTML markup for a definition list where each key of the array is a term, and each corresponding value is a definition.

(Hint: An XHTML definition list element consists of `< dl > ... </dl >` tags. Inside these tags, terms are marked up using `< dt > ... </dt >` tags, and definitions using `< dd > ... </dd >` tags.)

2. A factorial of any given integer, n , is the product of all positive integers between 1 and n inclusive. So the factorial of 4 is $1 \times 2 \times 3 \times 4 = 24$, and the factorial of 5 is $1 \times 2 \times 3 \times 4 \times 5 = 120$. This can be expressed recursively as follows:

- ❑ If $n == 0$, return 1. (This is the base case)
- ❑ If $n > 0$, compute the factorial of $n - 1$, multiply it by n , and return the result

Write a PHP script that uses a recursive function to display the factorials of the integers 0 to 10.

Assignment

Create a PHP page (Chapter 6 Assignment2.php) that has a function to display the content of given array that contains Linux tree directory structure (variable) like you have created in Exercise5-3. The function should display the content by using HTML `<div>` tag with CSS `margin-left` like shown below.

