

php Arrays (Part 2)

MIS 4530

Dr. Garrett

Iterating Through an Array

- The **internal array pointer** refers to the currently selected element in an array

Table 7-1 Array pointer iteration functions

Function	Description
<code>current(array)</code>	Returns the current array element
<code>each(array)</code>	Returns the key and value of the current array element and moves the internal array pointer to the next element
<code>end(array)</code>	Moves the internal array pointer to the last element
<code>key(array)</code>	Returns the key of the current array element
<code>next(array)</code>	Moves the internal array pointer to the next element
<code>prev(array)</code>	Moves the internal array pointer to the previous element
<code>reset(array)</code>	Resets the internal array pointer to the first element

Iterating Through an Array (continued)

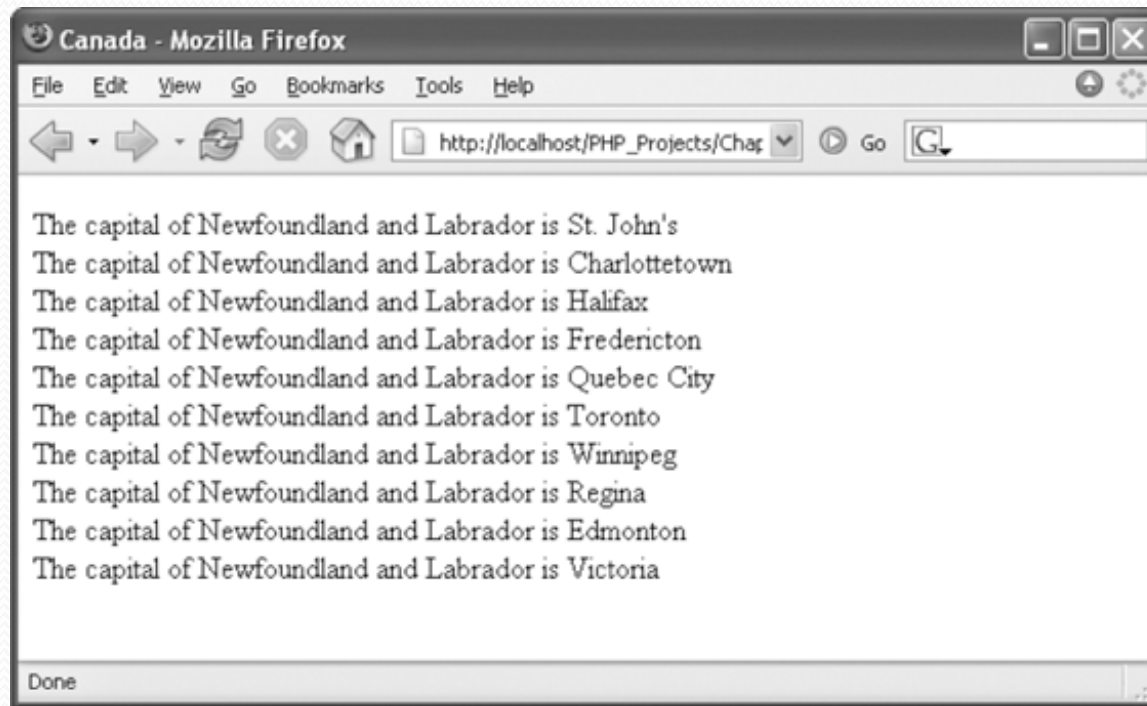


Figure 7-7 Output of an array without advancing the internal array pointer

Determining if a Value Exists

- The `in_array()` function returns a Boolean value of true if a given value exists in an array
- The `array_search()` function determines whether a given value exists in an array and
 - Returns the index or key of the first matching element if the value exists, or
 - Returns false if the value does not exist

```
if (in_array("Neurology", $HospitalDepts))  
    echo "<p>The hospital has a Neurology department.</p>";
```

Determining if a Key Exists

- The `array_key_exists()` function determines whether a given index or key exists
- You pass two arguments to the `array_key_exists()` function:
 - The first argument represents the key to search for
 - The second argument represents the name of the array in which to search

Determining if a Key Exists (continued)

```
$GamePieces["Dancer"] = "Daryl";  
$GamePieces["Fat Man"] = "Dennis";  
$GamePieces["Assassin"] = "Jennifer";  
if (array_key_exists("Fat Man", $GamePieces))  
    echo "<p>{$GamePieces["Fat Man"]} is already  
    'Fat Man'.</p>";  
else {  
    $GamePieces["Fat Man"] = "Don";  
    echo "<p>{$GamePieces["Fat Man"]} is now  
    'Fat Man'.</p>";  
}
```

Returning a Portion of an Array

- The `array_slice()` function returns a portion of an array and assigns it to another array
- The syntax for the `array_slice()` function is:

```
array_slice(array_name, start, characters_to_return);
```

Returning a Portion of an Array (continued)

```
$TopGolfers = array("Tiger Woods", "Vijay Singh", "Ernie Els",  
"Phil Mickelson", "Retief Goosen", "Padraig Harrington", "David  
Toms", "Sergio Garcia", "Adam Scott", "Stewart Cink");  
$TopFiveGolfers = array_slice($TopGolfers, 0, 5);  
echo "<p>The top five golfers in the world are:</p><p>";  
for ($i=0; $i<count($TopFiveGolfers); ++$i) {  
    echo "{$TopFiveGolfers[$i]}<br />";  
}  
echo "</p>";
```

Returning a Portion of an Array (continued)

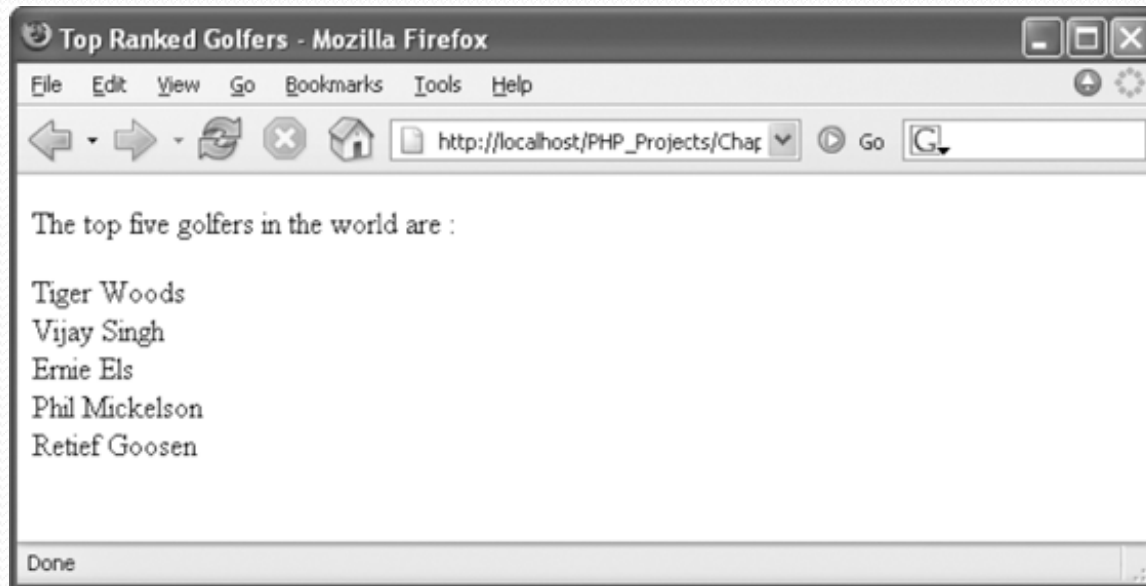


Figure 7-8 Output of an array returned with the `array_slice()` function

Sorting Arrays

- The most commonly used array sorting functions are:
 - `sort()` and `rsort()` for indexed arrays
 - `ksort()` and `krsort()` for associative arrays

Sorting Arrays (continued)

Function	Description
<code>array_multisort(array[, array, ...])</code>	Sorts multiple arrays or multidimensional arrays
<code>arsort(array[, SORT_REGULAR SORT_NUMERIC SORT_STRING])</code>	Performs a reverse sort of values in an associative array and maintains the existing keys
<code>asort(array[, SORT_REGULAR SORT_NUMERIC SORT_STRING])</code>	Sorts an associative array by value and maintains the existing keys
<code>krsort(array[, SORT_REGULAR SORT_NUMERIC SORT_STRING])</code>	Performs a reverse sort of an associative array by key

Table 7-2 Array sorting functions

Sorting Arrays (continued)

Function	Description
<code>ksort(array[, SORT_REGULAR SORT_NUMERIC SORT_STRING])</code>	Sorts an associative array by key
<code>natcasesort(array)</code>	Performs a case-sensitive natural order sort by value and maintains the existing indexes or keys
<code>natsort(array)</code>	Performs a natural order sort by value and maintains the existing indexes or keys
<code>rsort(array[, SORT_REGULAR SORT_NUMERIC SORT_STRING])</code>	Performs a reverse sort of values in an indexed array and renumbers the indexes
<code>sort(array[, SORT_REGULAR SORT_NUMERIC SORT_STRING])</code>	Sorts an indexed array by value and renumbers the indexes

Table 7-2 Array sorting functions (continued)

Sorting Arrays (continued)

Table 7-2 Array sorting functions (continued)

Function	Description
<code>uk_sort(array[, comparison_function])</code>	Uses a comparison expression to sort an associative array by keys, maintaining the existing keys
<code>usort(array[, comparison_function])</code>	Uses a comparison expression to sort an indexed array by values, renumbering the indexes

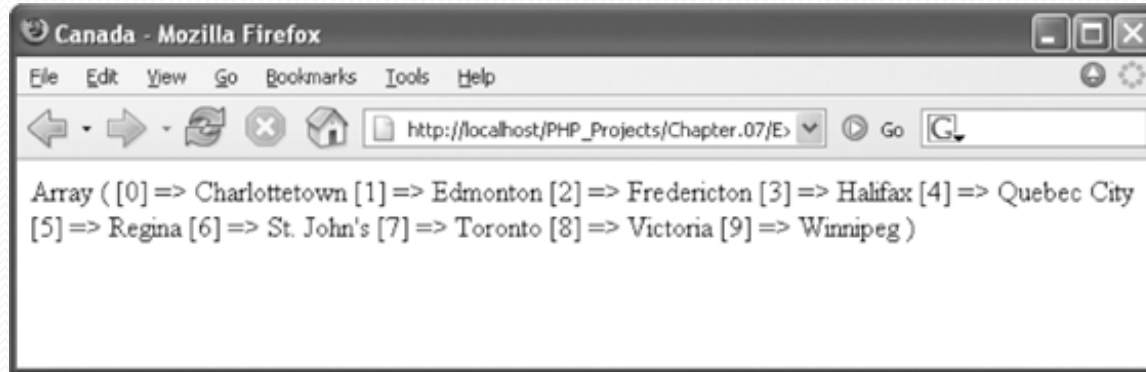
- If the `sort()` and `rsort()` functions are used on an associative array, the keys are replaced with indexes

Sorting Arrays (continued)



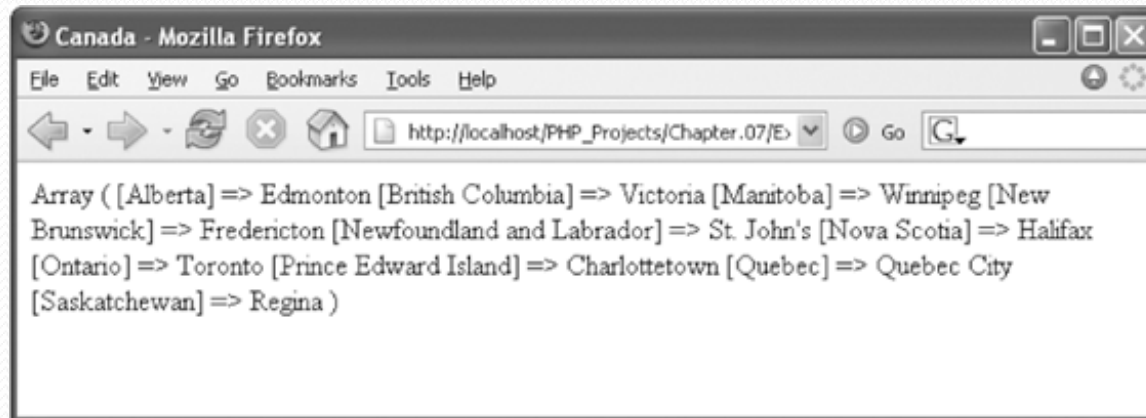
Figure 7-9 Output of an array after applying the `sort()` and `rsort()` functions

Sorting Arrays (continued)



```
Array ( [0] => Charlottetown [1] => Edmonton [2] => Fredericton [3] => Halifax [4] => Quebec City [5] => Regina [6] => St. John's [7] => Toronto [8] => Victoria [9] => Winnipeg )
```

Figure 7-10 Output of an associative array after executing the `sort()` function



```
Array ( [Alberta] => Edmonton [British Columbia] => Victoria [Manitoba] => Winnipeg [New Brunswick] => Fredericton [Newfoundland and Labrador] => St. John's [Nova Scotia] => Halifax [Ontario] => Toronto [Prince Edward Island] => Charlottetown [Quebec] => Quebec City [Saskatchewan] => Regina )
```

Figure 7-11 Output of an associative array after executing the `ksort()` function

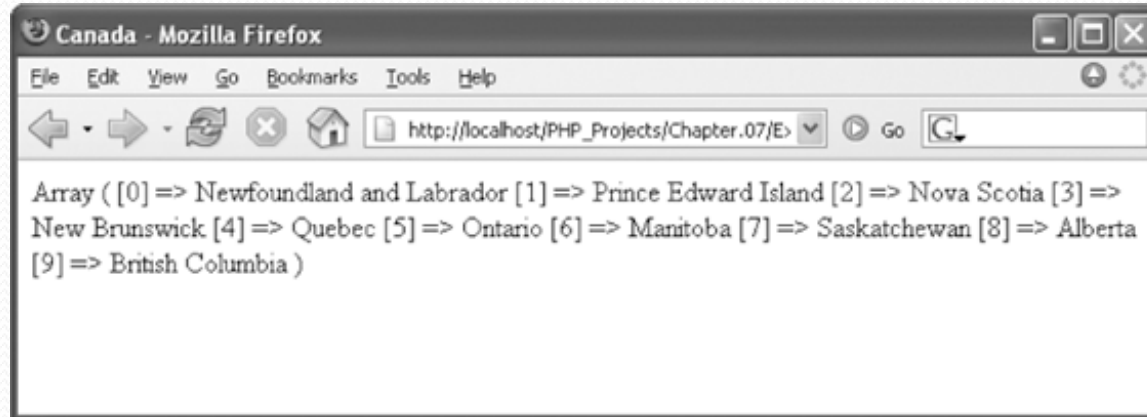
Combining Arrays

- To append one array to another, use the addition (+) or the compound assignment operator (+=)
- To merge two or more arrays use the `array_merge()` function
- The syntax for the `array_merge()` function is:

```
new_array = array_merge($array1, $array2,  
$array3, ...);
```

Combining Arrays (continued)

```
$Provinces = array("Newfoundland and Labrador", "Prince Edward  
Island", "Nova Scotia", "New Brunswick", "Quebec", "Ontario",  
"Manitoba", "Saskatchewan", "Alberta", "British Columbia");  
$Territories = array("Nunavut", "Northwest Territories", "Yukon  
Territory");  
$Canada = $Provinces + $Territories;  
print_r($Canada);
```



```
Canada - Mozilla Firefox  
File Edit View Go Bookmarks Tools Help  
http://localhost/PHP_Projects/Chapter.07/E>  
Array ( [0] => Newfoundland and Labrador [1] => Prince Edward Island [2] => Nova Scotia [3] =>  
New Brunswick [4] => Quebec [5] => Ontario [6] => Manitoba [7] => Saskatchewan [8] => Alberta  
[9] => British Columbia )
```

Figure 7-12 Output of two combined indexed arrays

Comparing Arrays

- The `array_diff()` function returns an array of elements that exist in one array but not in any other arrays to which it is compared
- The syntax for the `array_diff()` function is:

```
new_array = array_diff($array1, $array2,  
$array3, ...);
```
- The `array_intersect()` function returns an array of elements that exist in all of the arrays that are compared

Comparing Arrays (continued)

- The syntax for the `array_intersect()` function is:

```
new_array = array_intersect($array1,  
$array2, $array3, ...);
```

Comparing Arrays (continued)



Figure 7-13 Output of an array created with the `array_intersect()` function

Creating Two-Dimensional Indexed Arrays

- A **multidimensional array** consists of multiple indexes or keys
- A **two-dimensional array** has two sets of indexes or keys

Creating Two-Dimensional Indexed Arrays (continued)

```
$USDollars = array(  
    104.6100, // Yen  
    0.7476, // Euro  
    0.5198, // UK Pound  
    1.2013, // Canadian Dollar  
    1.1573 // Swiss Francs  
);
```

Table 7-3 Currency conversion table

	U.S. \$	Yen	Euro	U.K. Pound	Canadian \$	Swiss Franc
U.S. \$	1	104.61	0.7476	0.5198	1.2013	1.1573
Yen	0.009559	1	0.007146	0.004969	0.011484	0.011063
Euro	1.3377	139.9368	1	0.6953	1.6070	1.5481
U.K. Pound	1.9239	201.2592	1.4382	1	2.3112	2.2265
Canadian \$	0.8324	87.0807	0.6223	0.4327	1	0.9634
Swiss Franc	0.8641	90.3914	0.6459	0.4491	1.0380	1

Creating Two-Dimensional Indexed Arrays (continued)

```
$USDollars = array(1, 104.61, 0.7476, 0.5198, 1.2013, 1.1573);  
$Yen = array(0.009559, 1, 0.007146, 0.004969, 0.011484, 0.011063);  
$Euro = array(1.3377, 139.9368, 1, 0.6953, 1.6070, 1.5481);  
$UKPound = array(1.9239, 201.2592, 1.4382, 1, 2.3112, 2.2265);  
$CanadianDollar = array(0.8324, 87.0807, 0.6223, 0.4327, 1, 0.9634);  
$SwissFranc = array(0.8641, 90.3914, 0.6459, 0.4491, 1.0380, 1);
```

Creating Two-Dimensional Indexed Arrays (continued)

```
$ExchangeRates = array($USDollars, $Yen, $Euro, $UKPound,  
$CanadianDollar, $SwissFranc);
```

Table 7-4 Elements and indexes in the `$ExchangeRates []` array

	0 (U.S. \$)	1 (Yen)	2 (Euro)	3 (U.K. Pound)	4 (Canadian \$)	5 (Swiss Franc)
0 (U.S. \$)	1	104.61	0.7476	0.5198	1.2013	1.1573
1 (Yen)	0.009559	1	0.007146	0.004969	0.011484	0.011063
2 (Euro)	1.3377	139.9368	1	0.6953	1.6070	1.5481
3 (U.K. Pound)	1.9239	201.2592	1.4382	1	2.3112	2.2265
4 (Canadian \$)	0.8324	87.0807	0.6223	0.4327	1	0.9634
5 (Swiss Franc)	0.8641	90.3914	0.6459	0.4491	1.0380	1

Creating Two-Dimensional Associative Arrays

Keys

	"U.S. \$"	"Yen"	"Euro"	"U.K. Pound"	"Canadian \$"	"Swiss Franc"
"U.S. \$"	1	104.61	0.7476	0.5198	1.2013	1.1573
"Yen"	0.009559	1	0.007146	0.004969	0.0114484	0.011063
"Euro"	1.3377	139.9368	1	0.6953	1.6070	1.5481
"U.K. Pound"	1.9239	201.2592	1.4382	1	2.3112	2.2265
"Canadian \$"	0.8324	87.0807	0.6223	0.4327	1	0.9634
"Swiss Franc"	0.8641	90.3914	0.6459	0.4491	1.0380	1

Keys

Elements

Elements

Figure 7-14 Elements and keys in the `$ExchangeRates []` array

Creating Multidimensional Arrays with a Single Statement

```
$ExchangeRates = array(  
    array(1, 104.61, 0.7476, 0.5198, 1.2013, 1.1573), // U.S. $  
    array(0.009559, 1, 0.007146, 0.004969, 0.011484, 0.011063), // Yen  
    array(1.3377, 139.9368, 1, 0.6953, 1.6070, 1.5481), // Euro  
    array(1.9239, 201.2592, 1.4382, 1, 2.3112, 2.2265), // U.K. Pound  
    array(0.8324, 87.0807, 0.6223, 0.4327, 1, 0.9634), // Canadian $  
    array(0.8641, 90.3914, 0.6459, 0.4491, 1.0380, 1) // Swiss Franc  
);
```

Working with Additional Dimensions

Table 7-5 The Alaska table of a three-dimensional array

		Quarters of the year			
		Q1	Q2	Q3	Q4
Salesperson	Sam	874	76	98	890
	Jane	656	133	64	354
	Lisa	465	668	897	64
	Hiroshi	31	132	651	46
	Jose	654	124	126	456