SWINBURNE UNIVERSITY OF TECHNOLOGY

Advanced Web Development: Arrays

Week 6

Outline

- Manipulating array elements
- Declaring and initialising associative arrays
- Using iteration functions
- Finding and extracting elements and values
- Operating Arrays: Sort, combine, and compare arrays
- Working with multidimensional arrays

Reading: Textbook Chapter 6
PHP Arrays
http://au.php.net/manual/en/book.array.php

MANIPULATING ARRAY ELEMENTS

Using Array

Post New Message page of the Discussion Forum script

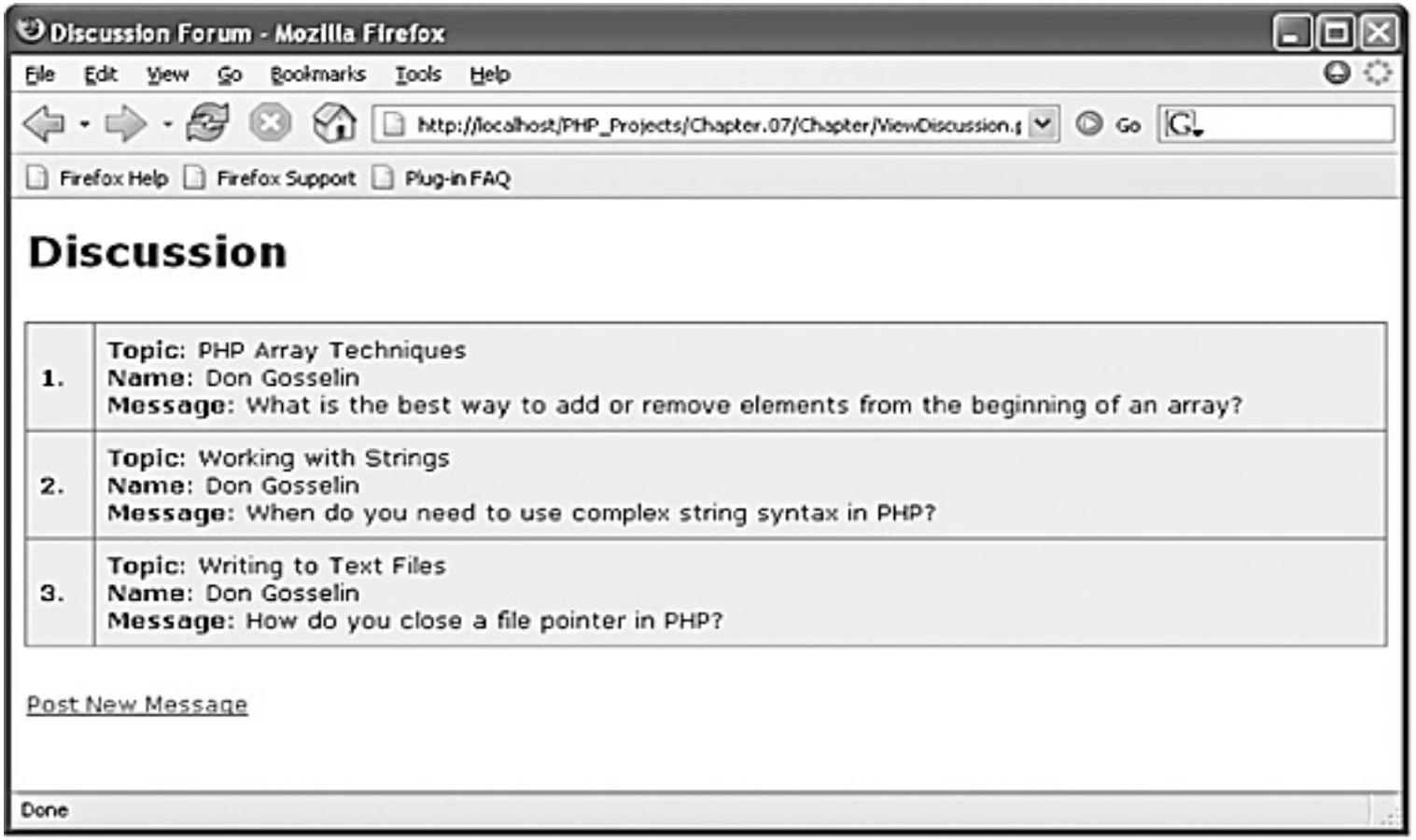
Using Array (continued)

```
...
$topic = $_POST["topic"];
$name = $_POST["name"];
$message = $_POST["message"];
$postMessage = $topic . "~" .  $name . "~" .
                                $message .  "\n";
$messageStore = fopen("messages.txt", "a");
fwrite($messageStore, "$postMessage");
fclose($messageStore);
echo "<p><strong>Topic</strong>: $topic<br />";
echo "<strong>Name</strong>: $name<br />";
echo "<strong>Message</strong>: $message</p>";
...
```

Using Array (continued)

```
...
if (!file_exists("messages.txt") || filesize("messages.txt") == 0) {
   echo "<p>There are no messages posted.</p>";
} else {
   $messageArray = file("messages.txt");
   for ($i = 0; $i < count($messageArray); $i++) {
      $curMessage = explode("~", $messageArray[$i]);
      echo "<tr>";
      echo "<td><strong>" . ($i + 1) . "</strong>.</td>";
      echo "<td><strong>Topic</strong>: "
           . stripslashes($curMessage[0]) . "<br />";
      echo "<strong>Name</strong>: "
           . stripslashes($curMessage[1]) . "<br />";
      echo "<strong>Message</strong>: "
           . stripslashes($curMessage[2]);
      echo "</td></tr>";
   }
}
...
```

Using Array (continued)



Message Posted page of the Discussion Forum script

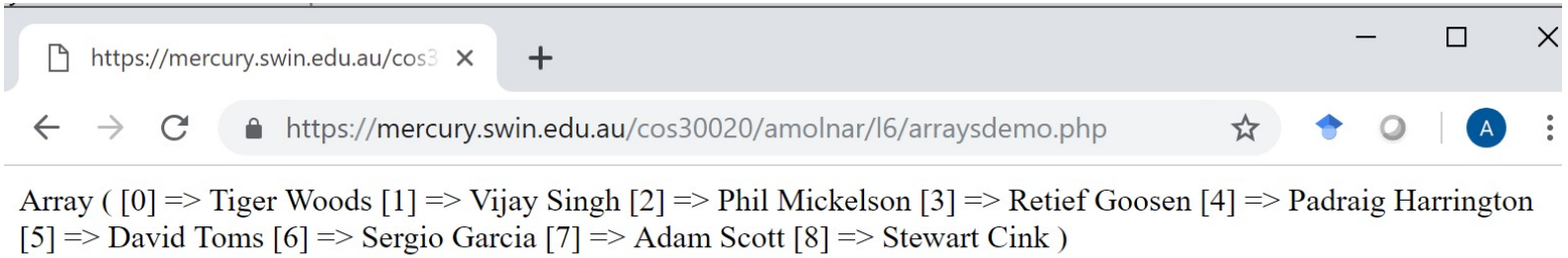
Adding and Removing Elements from the Beginning of an Array

- The `array_shift()` function removes the first element from the beginning of an array
  - Pass the name of the array whose first element you want to remove
- The `array_unshift()` function adds one or more elements to the beginning of an array
  - Pass the name of an array followed by comma-separated values for each element you want to add

Adding and Removing Elements from the Beginning of an Array (continued)

```
$topGolfers = array(
        "Ernie Els",
        "Phil Mickelson",
        "Retief Goosen",
        "Padraig Harrington",
        "David Toms",
        "Sergio Garcia",
        "Adam Scott",
        "Stewart Cink");
array_shift($topGolfers);
array_unshift($topGolfers, "Tiger Woods", "Vijay Singh");
print_r($topGolfers);
```

Adding and Removing Elements from the Beginning of an Array (continued)



Output of an array modified with the `array_shift()` and `array_unshift()` functions

Adding and Removing Elements from the End of an Array

- The **`array_pop()`** function removes the last element from the end of an array
  - Pass the name of the array whose last element you want to remove
- The **`array_push()`** function adds one or more elements to the end of an array
  - Pass the name of an array followed by comma-separated values for each element you want to add

```
$hospitalDepts = array("Anesthesia", "Molecular Biology",
                       "Neurology", "Pediatrics");
array_pop($hospitalDepts);  //removes Pediatrics from end
array_push($hospitalDepts,"Psychiatry","Pulmonary Diseases");
   // adds these to the end of array
```

Adding and Removing Elements Within an Array

- The **`array_splice()`** function
  - adds or removes array elements
  - renumbers the indexes in the array
- The syntax for the `array_splice()` function is:

```
array_splice(array_name, starting element,
   elements_to_delete, values_to_insert);
```

array_splice() Function

- To add an element within an array, include a value of 0 as the third argument

```
$hospitalDepts = array(
        "Anesthesia",                 // first element (0)
        "Molecular Biology",          // second element (1)
        "Neurology",                  // third element (2)
        "Pediatrics");                // fourth element (3)
array_splice($hospitalDepts, 3, 0, "Ophthalmology");
```

Result: "Ophthalmology" is added between "Neurology" and "Pediatrics“

`array_splice()` Function (continued)

- To add more than one element within an array, pass the `array()` construct as the fourth argument, separate the new `array()` element values by commas

```
$hospitalDepts = array(
        "Anesthesia",                 // first element (0)
        "Molecular Biology",          // second element (1)
        "Neurology",                  // third element (2)
        "Pediatrics");                // fourth element (3)
array_splice($hospitalDepts, 3, 0,
        array("Ophthalmology","Otolaryngology"));
```

Result: "Ophthalmology" and "Otolaryngology" are added between "Neurology" and "Pediatrics"

`array_splice()` Function (continued)

- Delete array elements by omitting the fourth argument from the `array_splice()` function

```
$hospitalDepts = array(
        "Anesthesia",               // first element (0)
        "Molecular Biology",        // second element (1)
        "Neurology",                // third element (2)
        "Pediatrics");              // fourth element (3)
array_splice($hospitalDepts, 1, 2);
```

Result: deletes 2nd & 3rd elements ("Molecular Biology" and "Neurology")

Note: If no 3rd argument, all elements starting from the specified position are deleted

DECLARING AND INITIALISING ASSOCIATIVE ARRAYS

SWINBURNE UNIVERSITY OF TECHNOLOGY

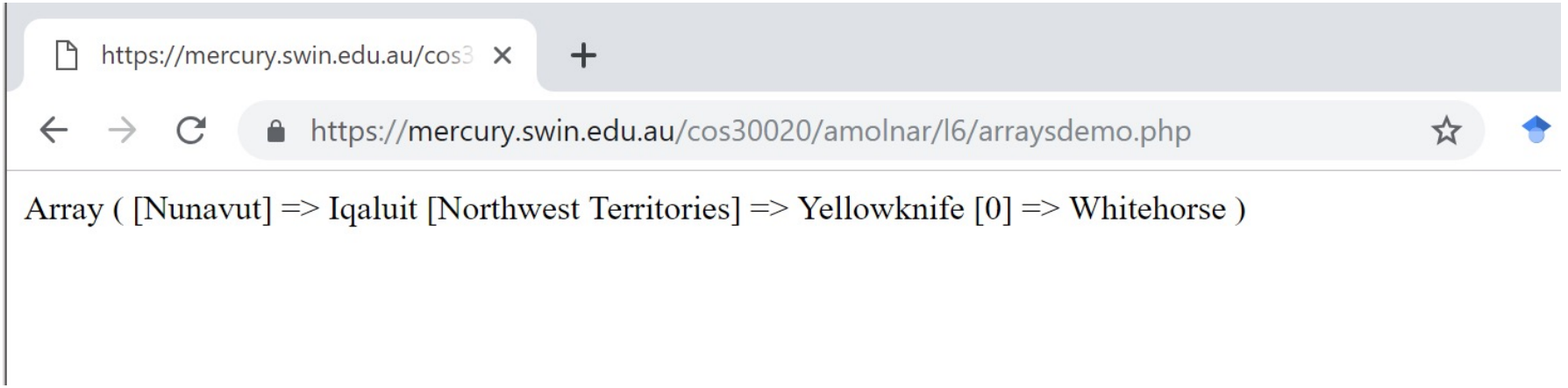
Declaring and Initialising Associative Arrays

- With associative arrays, you specify an element's key by using the array operator (=>)
- The syntax for declaring and initialising an associative array:
  - `$array_name = array(key=>value, ...);`
    e.g.
    ```
    $capitals = array("Ontario"=>"Toronto",
    "Alberta"=>"Edmonton", ...);
    ```
  - `$array_name[key] = value;`
    e.g.
    ```
    $capitals["Ontario"] = "Toronto";
    $capitals["Alberta"] = "Edmonton";
    ...
    ```
- The sytax to refer to an element in an associate array

  e.g. `echo $capitals["Ontario"];`

Declaring and Initialising Associative Arrays

```
$territorialCapitals["Nunavut"] = "Iqaluit";
$territorialCapitals["Northwest Territories"] = "Yellowknife";
$territorialCapitals[] = "Whitehorse"; // next indexed element
print_r($territorialCapitals);
```



Output of array with associative and indexed elements

USING ITERATION FUNCTIONS

SWINBURNE UNIVERSITY OF TECHNOLOGY

Iterating Through an Array

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

From PHP 7.2 each is deprecated

Iterating Through an Array (continued)

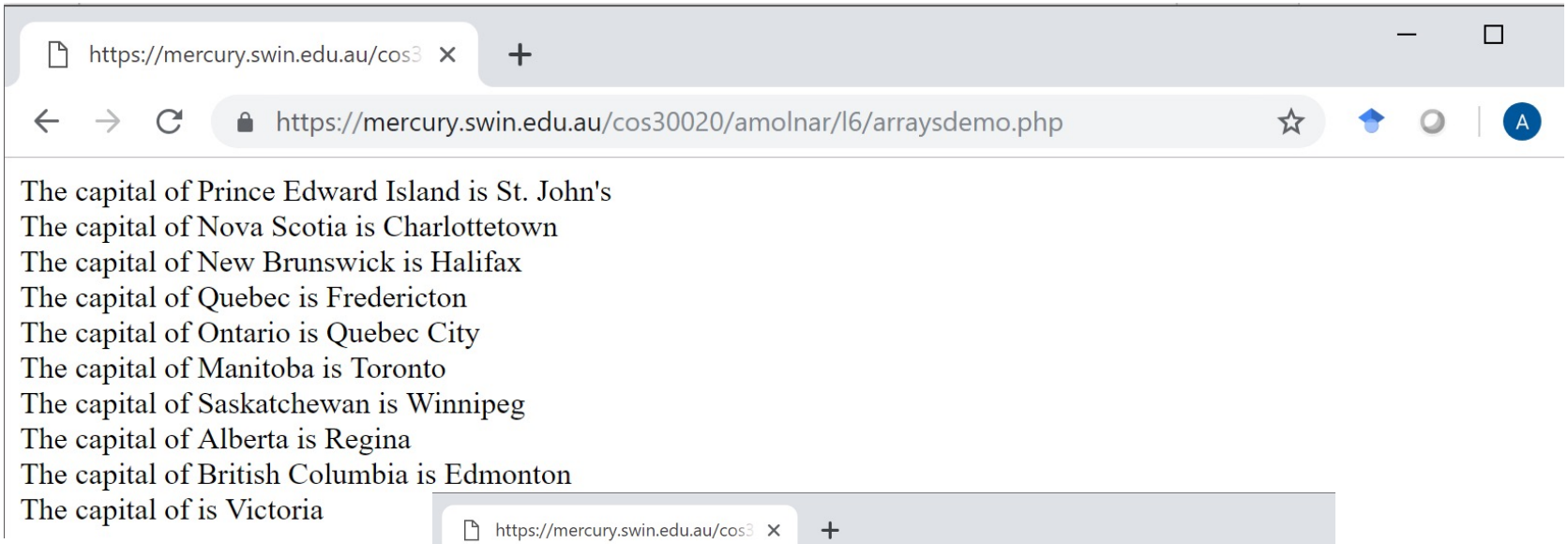
```
$capitals = array(
"NewFounderland and Labrador"=>"St. John's",
"Prince Edward Island"=>"Charlottetown",
"Nova Scotia"=>"Halifax",
"New Brunswick"=>"Fredericton",
"Quebec"=>"Quebec City",
"Ontario"=>"Toronto",
"Manitoba"=>"Winnipeg",
"Saskatchewan"=>"Regina",
"Alberta"="Edmonton",
"British Columbia"=>"Victoria");
foreach ($capitals as $capital) {
   echo "The capital of ",
    key($capitals), " is $capital<br>";
}
```

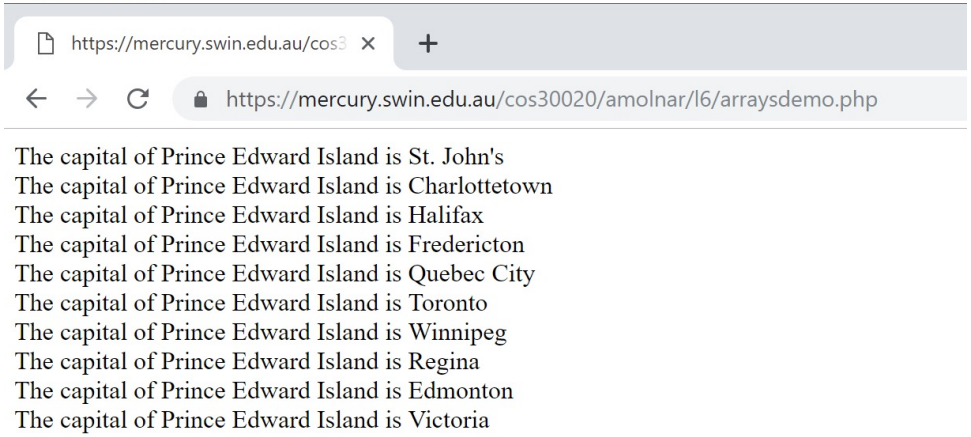
See next slide for output ☹ – correct as follows ☺

```
foreach ($capitals as $capital) {
   echo "The capital of ",
    key($capitals), " is $capital<br>";
    next($capitals);

}
```

Iterating Through an Array (continued)





Output of an array using `foreach`

without advancing the internal array pointer

FINDING AND EXTRACTING ELEMENTS AND VALUES

SWINBURNE UNIVERSITY OF TECHNOLOGY

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

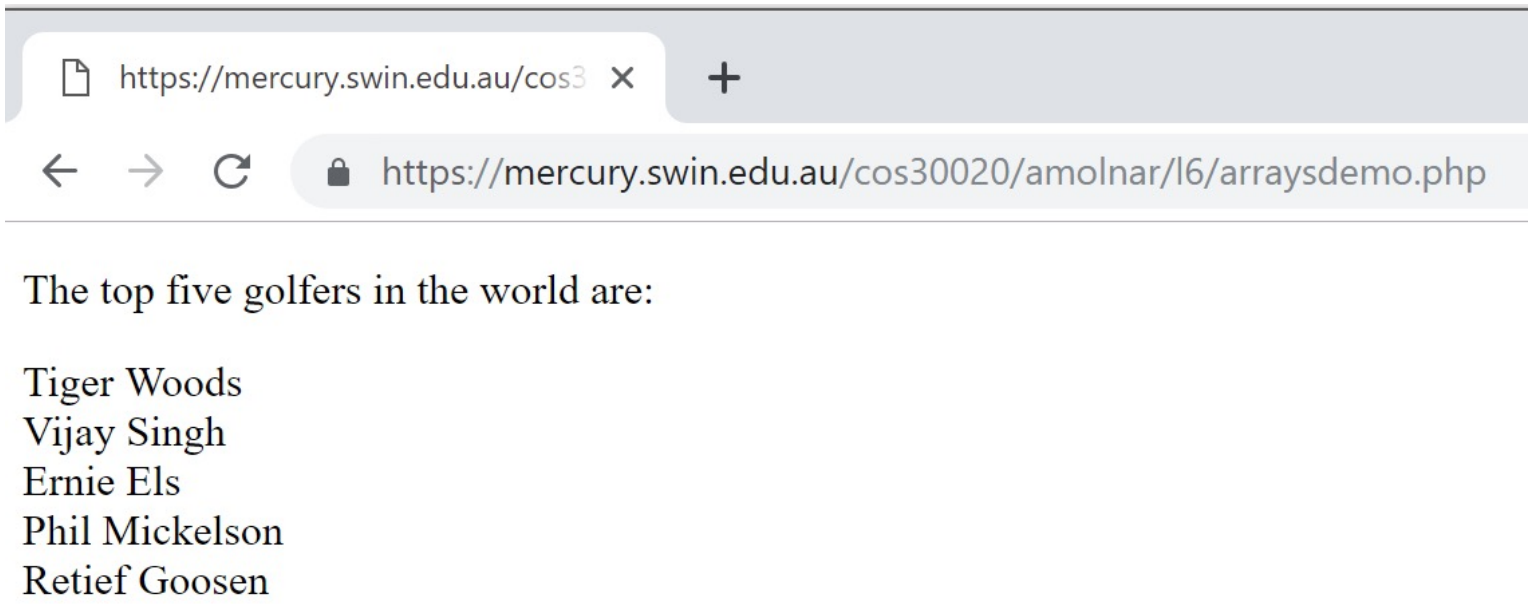
- The **`array_slice()`** function returns a portion of an array and assigns it to another array
- The syntax for the `array_slice()` function is:

```
new_array = array_slice(array_name, starting element,
elements_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)



Output of an array returned with the `array_slice()` function

OPERATING ON ARRAYS: SORT, COMBINE AND COMPARE

SWINBURNE UNIVERSITY OF TECHNOLOGY

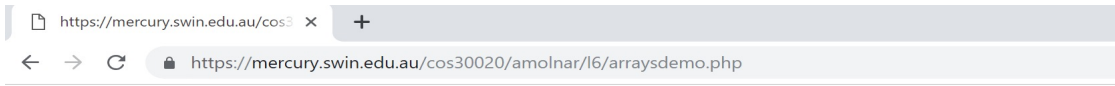
Sorting Arrays

The most commonly used array sorting functions are:

- **`sort()`** and **`rsort()`** for *indexed arrays*
  - `sort()` sorts an indexed array by value and renumbers the indexes
  - `rsort()` performs a reverse sort
- **`ksort()`** and **`krsort()`** for *associative arrays by key*

Also see: http://www.php.net/manual/en/array.sorting.php

Sorting Arrays (continued)



The top five golfers in the world are:

Tiger Woods
Vijay Singh
Ernie Els
Phil Mickelson
Retief Goosen

The top five golfers in the world in alphabetical order are:

Ernie Els
Phil Mickelson
Retief Goosen
Tiger Woods
Vijay Singh

The top five golfers in the world in reverse alphabetical order are:

Vijay Singh
Tiger Woods
Retief Goosen
Phil Mickelson
Ernie Els

Output of an array after applying the `sort()` and `rsort()` functions

Combining Arrays

- To append one array to another, use the addition (+) or the compound assignment operator (+=)
  Note: only array elements with unique keys are appended. duplicated indexes/keys are ignored
- For example

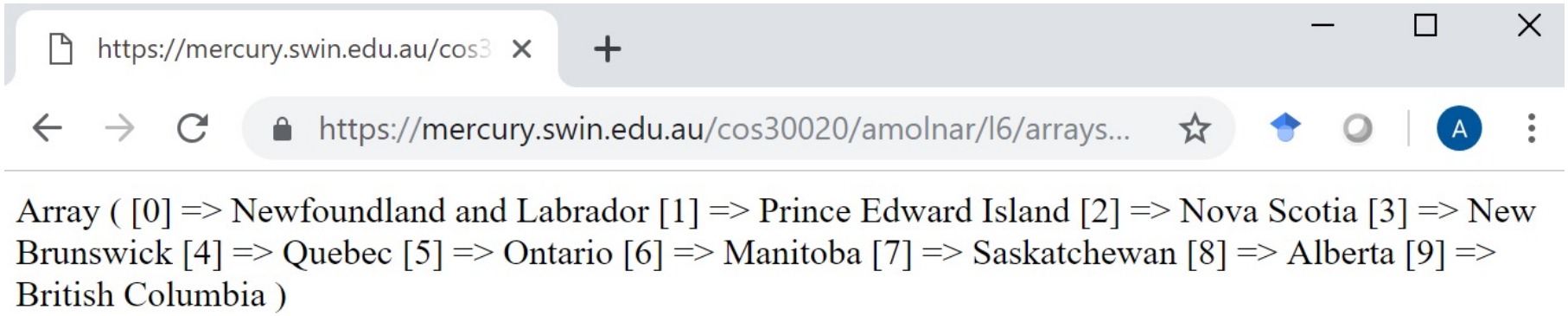
```
$provinces = array("Newfoundland and Labrador",
   "Prince Edward Island", "Nova Scotia", "New Brunswick", "Quebec",
   "Ontario", "Manitoba", "Saskatchewan", "Alberta", "British
   Columbia");

$territories = array("Nunavut", "Northwest Territories",
   "Yukon Territory");

$canada = $provinces + $territories; //territories ignored

print_r($canada);
```

Combining Arrays (continued)



Output of two combined indexed arrays

- The keys 0, 1 and 2 already exists in $provinces, thus the 3 elements from $territories are not appended

Combining Arrays (continued)

- + and += works best on associative arrays, especially if the arrays involved do not have any common keys.
- For example

```
$arr1 = array ("one"=>"apple", "two"=>"banana");
$arr2 = array ("three"=>"cherry", "four"=>"grapes");
$arr3 = $arr1 + $arr2;
print_r($arr3);
```

Output

```
Array ( [one] => apple [two] => banana [three] => cherry [four] =>
   grapes )
```

Combining Arrays (continued)

- 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, ...);
```

Note: duplicated associative keys overwrite, elements of numeric keys are appended

Combining Arrays (continued)

- For example, given

```
$arr1 = array ("one"=>"apple", "two"=>"banana");
$arr2 = array ("three"=>"cherry", "two"=>"grapes");
```

- Duplicate keys ignored

```
$arr3 = $arr1 + $arr2;
print_r($arr3);
```

Output: `Array ( [one] => apple [two] => banana [three] => cherry )`

- Duplicate keys overwritten

```
$arr4 = array_merge ($arr1, $arr2);
print_r($arr3);
```

Output: `Array ( [one] => apple [two] => grapes [three] => cherry )`

Combining Arrays (continued)

- array_merge works best with arrays having numeric keys
- For example

```
$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");

$canada = array_,merge ($provinces, $territories);

print_r($canada);                                    //territories appended
```

Output:

```
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 [10] => Nunavut [11] => Northwest Territories [12] => Yukon
Territory )
```

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

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

WORKING WITH TWO DIMENSIONAL ARRAYS

SWINBURNE UNIVERSITY OF TECHNOLOGY

Creating Two-Dimensional Indexed Arrays

- A **multidimensional array** consists of multiple indexes or keys

  Note: Novice programmers rarely need to use arrays larger than two dimensions

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

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);

$exchangeRates = array($usDollars, $yen, $euro, $ukPound,
  $canadianDollar, $swissFranc);
```

Elements and indexes in the `$exchangeRates[row][col]` 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

Elements and keys in the `$exchangeRates[row][col]` 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
);
```

Summary

- The `array_shift()` function removes the first element from the beginning of an array
- The `array_unshift()` function adds one or more elements to the beginning of an array
- The `array_pop()` function removes the last element from the end of an array
- The `array_push()` function adds one or more elements to the end of an array
- The `array_splice()` function adds or removes array elements

Summary (continued)

- 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
- The `array_key_exists()` function determines whether a given index or key exists
- The `array_slice()` function returns a portion of an array and assigns it to another array
- 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 `array_intersect()` function returns an array of elements that exist in all of the arrays that are compared