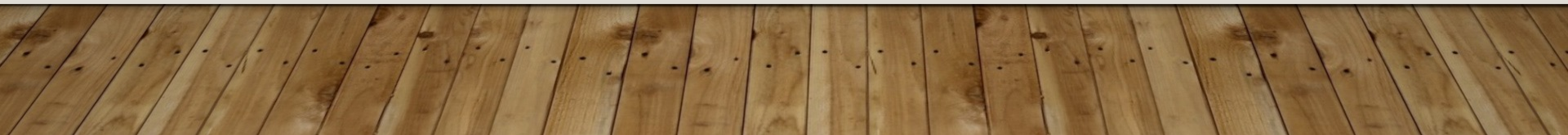
ISIT307 - WEB SERVER PROGRAMMING

LECTURE 4 – MANIPULATING ARRAYS



LECTURE PLAN

- Manipulate array elements
- Declare and initialize associative arrays
- Iterate through an array
- Find and extract elements and values
- Sort, combine, and compare arrays
- Use arrays in Web forms
- Multidimensional arrays

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

```
$TopSellers = array(
  "Chevrolet Impala",
  "Chevrolet Malibu",
  "Chevrolet Silverado",
  "Ford F-Series",
  "Toyota Camry",
  "Toyota Corolla",
  "Nissan Altima",
  "Honda Accord",
  "Honda Civic",
  "Dodge Ram");
```

```
echo "<h2>Original Array</h2>\n";
echo "<pre>\n";
print_r($TopSellers);
echo "</pre>\n";

array_shift($TopSellers);
echo "<h2>Array after Shifting</h2>\n";
echo "<pre>\n";
print_r($TopSellers);
echo "</pre>\n";

array_unshift($TopSellers, "Honda CR-V", "Honda");
echo "<h2>Array after Unshifting</h2>\n";
echo "<pre>\n";
print_r($TopSellers);
echo "</pre>\n";
```

ADDING AND REMOVING ELEMENTS FROM THE BEGINNING OF AN ARRAY

Original Array

```
Array
(
    [0] => Chevrolet Impala
    [1] => Chevrolet Malibu
    [2] => Chevrolet Silverado
    [3] => Ford F-Series
    [4] => Toyota Camry
    [5] => Toyota Corolla
    [6] => Nissan Altima
    [7] => Honda Accord
    [8] => Honda Civic
    [9] => Dodge Ram
)
```

Array after Shifting

```
Array
(
    [0] => Chevrolet Malibu
    [1] => Chevrolet Silverado
    [2] => Ford F-Series
    [3] => Toyota Camry
    [4] => Toyota Corolla
    [5] => Nissan Altima
    [6] => Honda Accord
    [7] => Honda Civic
    [8] => Dodge Ram
)
```

Array after Unshifting

```
Array
(
    [0] => Honda CR-V
    [1] => Honda
    [2] => Chevrolet Malibu
    [3] => Chevrolet Silverado
    [4] => Ford F-Series
    [5] => Toyota Camry
    [6] => Toyota Corolla
    [7] => Nissan Altima
    [8] => Honda Accord
    [9] => Honda Civic
    [10] => Dodge Ram
)
```

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

ADDING AND REMOVING ELEMENTS FROM THE END OF AN ARRAY - EXAMPLE

```
$HospitalDepts = array(
     "Anesthesia",
     "Molecular Biology",
     "Neurology",
     "Pediatrics");
array_pop($HospitalDepts); // Removes "Pediatrics"
array_push($HospitalDepts, "Psychiatry", "Pulmonary
  Diseases");
// adds "Psychiatry", "Pulmonary Diseases" at the end
```

ADDING AND REMOVING ELEMENTS WITHIN AN ARRAY

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

```
array_splice(array_name, start,
        number_to_delete, values_to_insert);
```

ADDING AND REMOVING ELEMENTS WITHIN AN ARRAY - EXAMPLE

- To add an element within an array, include a value of 0 as the third argument of 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, 2, 0, "Ophthalmology");
```

Array after adding at 2

```
Array
(
    [0] => Anesthesia
    [1] => Molecular Biology
    [2] => Ophtamology
    [3] => Neurology
    [4] => Pediatrics
)
```

ADDING AND REMOVING ELEMENTS WITHIN AN ARRAY (CONTINUED)

- To add more than one element within an array, pass the `array()` construct as the fourth argument of the `array_splice()` function
- Separate the new 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("Opthalmology",
                 "Otolaryngology"));
```

Array after adding at 3

```
Array
(
    [0] => Anesthesia
    [1] => Molecular Biology
    [2] => Neurology
    [3] => Opthalmology
    [4] => Otolaryngology
    [5] => Pediatrics
)
```

ADDING AND REMOVING ELEMENTS WITHIN AN ARRAY (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);
```

Array after delete at 1 delete 2

```
Array
(
    [0] => Anesthesia
    [1] => Pediatrics
)
```

ADDING AND REMOVING ELEMENTS WITHIN AN ARRAY (CONTINUED)

- The `unset()` function removes array elements and other variables
  - Pass the array name and index number of the element you want to remove
- To remove multiple elements, separate each index name and element number with commas, for example:

```
unset($HospitalDepts[1], $HospitalDepts[2]);
```

- The `unset()` function do not renumber the next elements in the array
- The `array_values()` function can be used to renumber the indexed array elements
  - do not operate directly on an array

REMOVING DUPLICATE ELEMENTS

- The `array_unique()` function removes duplicate elements from an array
  - Pass the name of the array from which you want to remove duplicate elements
- The `array_unique()` function returns a new array (with removed duplicate values), but does not renumber the indexes

REMOVING DUPLICATE ELEMENTS - EXAMPLE

```
$TopSellers = array(
     "Ford F-Series", "Chevrolet  Silverado", "Toyota Camry",
     "Honda Accord", "Toyota Corolla", "Ford F-Series",
      "Honda Civic", "Honda CR-V", "Honda Accord",
      "Nissan Altima", "Toyota Camry",
     "Chevrolet Impala", "Dodge Ram", "Honda CR-V");
echo "<p>The top selling vehicles are:</p><p>";

$TopSellers = array_unique($TopSellers);
$TopSellers = array_values($TopSellers);

for ($i=0; $i<count($ TopSellers); ++$i) {
     echo "{$TopSellers[$i]}<br />";
}
echo "</p>";
```

DECLARING AND INITIALIZING ASSOCIATIVE ARRAYS

- With associative arrays, you specify an element's key by using the array operator (=>)
  - The syntax for declaring and initializing an associative array is:

```
$array_name = array(key=>value, ...);
```

- Example

```
$ProvincialCapitals = array(
"Newfoundland and Labrador" => "St. John's",
"Prince Edward Island" => "Charlottetown",
"Nova Scotia" => "Halifax",
"New Brunswick" => "Fredericton");
```

DECLARING AND INITIALIZING ASSOCIATIVE ARRAYS

- An associative array can be also created by assigning values to elements

```
$ProvincialCapitals["Newfoundland and Labrador"] =
"St. John's";
$ProvincialCapitals["Prince Edward Island"] =
"Charlottetown";
$ProvincialCapitals["Nova Scotia"] = "Halifax";
```

- Creates an array if it doesn't exist, or If the array does exist each assignment statement overwrites any existing elements that already use the same key or appends any new keys and values to the end of the array

DECLARING AND INITIALIZING ASSOCIATIVE ARRAYS

- If a new element is added to an associative array without specifying a key, the new element is assigned an index of 0 or the next available integer

```
$Territories["North"] = "Nunavut";
$Territories["NothWest"] = "Northwest Territories";
$Territories[] = "Yukon Territory";
```

- In PHP, you can declare an array and use a starting index other than 0 without creating empty elements, for example:

```
$Territories[100] = "Nunavut";
$Territories[] = "Northwest Territories";
$Territories[] = "Yukon Territory";
```

ITERATING THROUGH AN ARRAY

- The **internal array pointer** refers to the currently selected element in an array
- A **foreach** statement allows looping through the elements of an array, but it does not change the position of the internal array pointer - for that purpose advanced foreach should be used
- Some useful functions for array pointer iteration:

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 array pointer to the next element
end(*array*)	Moves the internal array pointer to the last element
key(*array*)	Returns the key of 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

ITERATING THROUGH AN ARRAY - EXAMPLES

```
$ProvincialCapitals = array(
   "Newfoundland and Labrador" => "St. John's",
   "Prince Edward Island" => "Charlottetown",
   "Nova Scotia" => "Halifax",
   "New Brunswick" => "Fredericton",
   "Quebec" => "Quebec City");
-------
foreach ($ProvincialCapitals as $Capital) {
   echo "The capital of " . key($ProvincialCapitals)
                            ." is $Capital<br />\n"; }
------
foreach ($ProvincialCapitals as $Capital) {
   echo "The capital of " . key($ProvincialCapitals) .
                                " is $Capital<br />\n";
   next($ProvincialCapitals); }
-----
foreach ($ProvincialCapitals as $Province => $Capital)
{
    echo "The capital of $Province is $Capital<br />\n";
}
```

The capital of Newfoundland and Labrador is St. John's
The capital of Newfoundland and Labrador is Charlottetown
The capital of Newfoundland and Labrador is Halifax
The capital of Newfoundland and Labrador is Fredericton
The capital of Newfoundland and Labrador is Quebec City

The capital of Newfoundland and Labrador is St. John's
The capital of Prince Edward Island is Charlottetown
The capital of Nova Scotia is Halifax
The capital of New Brunswick is Fredericton
The capital of Quebec is Quebec City

The capital of Newfoundland and Labrador is St. John's
The capital of Prince Edward Island is Charlottetown
The capital of Nova Scotia is Halifax
The capital of New Brunswick is Fredericton
The capital of Quebec is Quebec City

FINDING AND EXTRACTING ELEMENTS AND VALUES

- One of the most basic methods for finding a value in an array is to use a looping statement to iterate through the array until you find the value

DETERMINING IF A VALUE EXISTS

- Rather than iterating thru the elements of an array, can be used `in_array()` and `array_search()` functions to determine whether a value exists in an array
- 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

DETERMINING IF A VALUE EXISTS

- Examples:

```
if (in_array("Charlottetown", $ProvincialCapitals))
     echo "<p>The capital 'Charlottetown' is in the list.</p>";
----

$keyCap = array_search ("Charlottetown", $ProvincialCapitals);
if ($keyCap !== FALSE)
       echo "<p>The 'Charlottetown' is capital of $keyCap.</p>";
```

DETERMINING IF A KEY EXISTS

- The `array_key_exists()` function determines whether a given index or key exists and returns TRUE or FALSE
- 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

```
if (array_key_exists("Quebec", $ProvincialCapitals))
     echo "<p>The key 'Quebec' is in the list.</p>";
```

DETERMINING IF A KEY EXISTS

- The `array_keys()` function returns an indexed array that contains all the keys in an associative array

```
$provinces = array_keys($ProvincialCapitals);
```

- As a second argument to the `array_keys()` function can be passed a value that specifies an element value for which to search
  - The keys are returned only for elements that match the specified value

RETURNING A PORTION OF AN ARRAY

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

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

RETURNING A PORTION OF AN ARRAY - EXAMPLE

```
$ThreeProvinces = array_slice($ProvincialCapitals, 2, 3);
echo "Random three provinces are <br />";
foreach ($ThreeProvinces as $Province => $Capital) {
    echo "$Province with capital $Capital, <br />";
}
```

Nova Scotia with capital Halifax,
New Brunswick with capital Fredericton,
Quebec with capital Quebec City,

SORTING ARRAYS

- The most commonly used array sorting functions that operate directly on the array are:
  - `sort()` and `rsort()` for indexed arrays (if are used with associative arrays the keys are replaced with sequential indexes)
  - `asort()`, `arsort()`, `ksort()` and `krsort()` for associative arrays

Function	Description
`array_multisort(array[, array, ...])`	Sorts multiple arrays or multidimensional arrays
`arsort(array[, SORT_REGULAR | SORT_NUMERIC | SORT_STRING])`	Sorts an array in descending order (largest to smallest) by value and maintains the existing keys for an associative array
`asort(array[, SORT_REGULAR | SORT_NUMERIC | SORT_STRING])`	Sorts an array in ascending order (smallest to largest) by value and maintains the existing keys for an associative array
`krsort(array[, SORT_REGULAR | SORT_NUMERIC | SORT_STRING])`	Sorts an array in descending order by key and maintains the existing keys for an associative array
`ksort(array[, SORT_REGULAR | SORT_NUMERIC | SORT_STRING])`	Sorts an array in ascending order by key and maintains the existing keys for an associative array
`natcasesort(array)`	Performs a case-sensitive natural order sort by value and maintains the existing keys for an associative array
`natsort(array)`	Performs a case-insensitive natural order sort by value and maintains the existing keys for an associative array

(continued)

Function	Description
`rsort(array[, SORT_REGULAR | SORT_NUMERIC | SORT_STRING])`	Sorts an array in descending order by value, removes any existing keys for an associative array, and renumbers the indexes starting with 0
`sort(array[, SORT_REGULAR | SORT_NUMERIC | SORT_STRING])`	Sorts an array in ascending order by value, removes any existing keys for an associative array, and renumbers the indexes starting with 0
`uaksort(array[, comparison_function])`	Sorts an array in ascending order by value using a comparison function and maintains the existing keys for an associative array
`uksort(array[, comparison_function])`	Sorts an array in ascending order by key using a comparison function and maintains the existing keys for an associative array
`usort(array[, comparison_function])`	Sorts an array in ascending order by value using a comparison function, removes any existing keys for an associative array, and renumbers the indexes starting with 0

COMBINING ARRAYS

- To append one array to another, use the addition (+) or the additive compound assignment operator (+=)
  - ignores any array elements in the secondary array where the indexes or keys already exist in the primary array, works well for associative arrays, if the array keys are not overlapping
- 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, ...);
```

- The `array_combine()` function create a new associative array that uses the values from one array as keys and element values from another array.

COMBINING ARRAYS - EXAMPLE

```
$Provinces = array("Newfoundland and Labrador", "Prince Edward
                                                Island", "Nova Scotia");
$Territories = array("Nunavut", "Northwest Territories");
$Canada = array_merge($Provinces, $Territories);
print_r($Canada);

$Provinces1 = array("Newfoundland and Labrador" => "St. John's",
   "Prince Edward Island" => "Charlottetown",
   "Nova Scotia" => "Halifax");
$Provinces2 = array( "New Brunswick" => "Fredericton",
                                          "Quebec" => "Quebec City");
$Canada1 = array_merge($Provinces1, $Provinces2);
print_r($Canada1);

$Territories[] = "NortEast";
$Canada2 = array_combine( $Territories, $Provinces);
print_r($Canada2);
```

COMBINING ARRAYS – EXAMPLE OUTPUT

```
Array
(
    [0] => Newfoundland and Labrador
    [1] => Prince Edward Island
    [2] => Nova Scotia
    [3] => Nunavut
    [4] => Northwest Territories
)
```

```
Array
(
    [Newfoundland and Labrador] => St. John's
    [Prince Edward Island] => Charlottetown
    [Nova Scotia] => Halifax
    [New Brunswick] => Fredericton
    [Quebec] => Quebec City
)
```

```
Array
(
    [Nunavut] => Newfoundland and Labrador
    [Northwest Territories] => Prince Edward Island
    [NortEast] => Nova Scotia
)
```

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 (keys and indexes are not renumbered)

```
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 (keys and indexes are not renumbered)

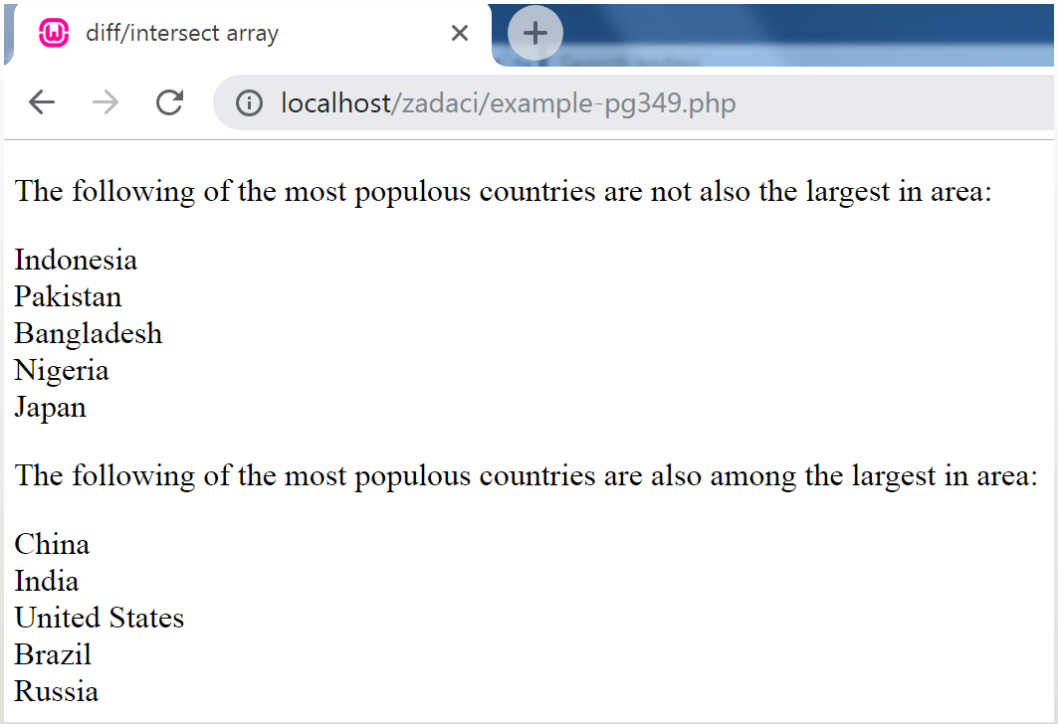
```
new_array = array_intersect($array1,  $array2,
                                    $array3, ...);
```

COMPARING ARRAYS - EXAMPLE

```
$Top10inArea = array("Russia", "China", "Canada", "United States", "Brazil",
                     "Australia", "India", "Argentina", "Kazakhstan", "Algeria");
$Top10inPopulation = array("China", "India", "United States", "Indonesia",
             "Brazil", "Pakistan", "Bangladesh", "Russia", "Nigeria", "Japan");

$Result = array_diff($Top10inPopulation, $Top10inArea);
$Result = array_values($Result);
echo "<p>The following of the most populous countries
                                    are not also the largest in area:</p>\n";
for ($i = 0; $i < count($Result); ++$i) {
        echo "{$Result[$i]}<br />\n";
}
$Result = array_intersect($Top10inPopulation, $Top10inArea);
$Result = array_values($Result);
echo "<p>The following of the most populous countries are also among
                                                the largest in area:</p>\n";
for ($i = 0; $i < count($Result); ++$i) {
        echo "{$Result[$i]}<br />\n";
}
```

COMPARING ARRAYS – EXAMPLE OUTPUT



EXAMPLE – MESSAGE BOARD

USING ARRAYS IN WEB FORMS

- PHP arrays can be used with HTML form input elements - enabling to store the form data in an array
- For that purpose the name attribute of the input element needs to use array notation
  - the data from any element with the same value for the name attribute will be appended to an array with that name
  - the indexes of the array elements can be used if there is a need, otherwise the only need is to append an opening and closing square bracket ([]) to the value of the name attribute

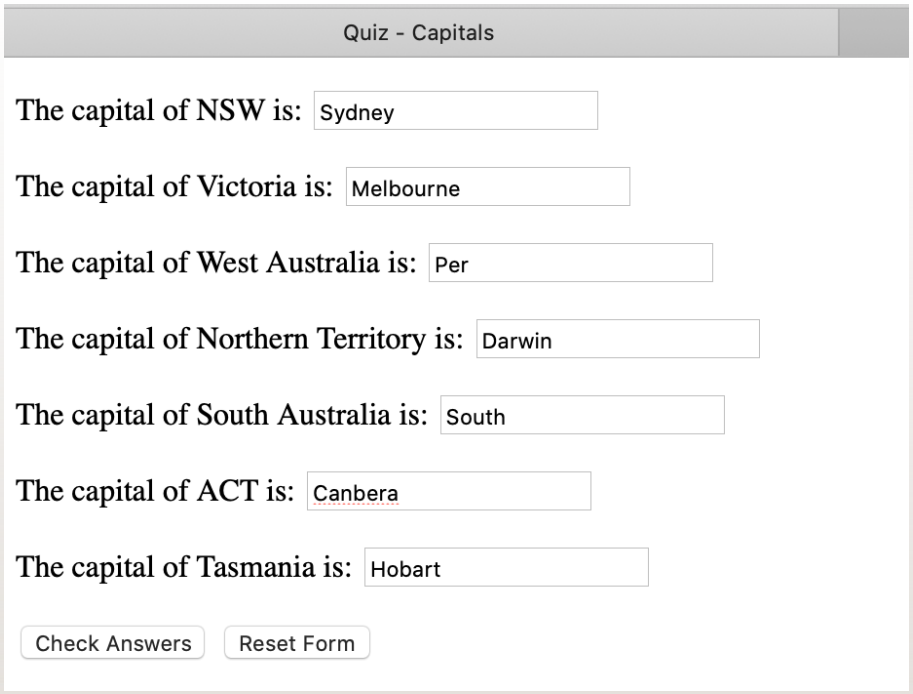
```
name='req[]' or name='req[0]' or name='req[Q1]'
```

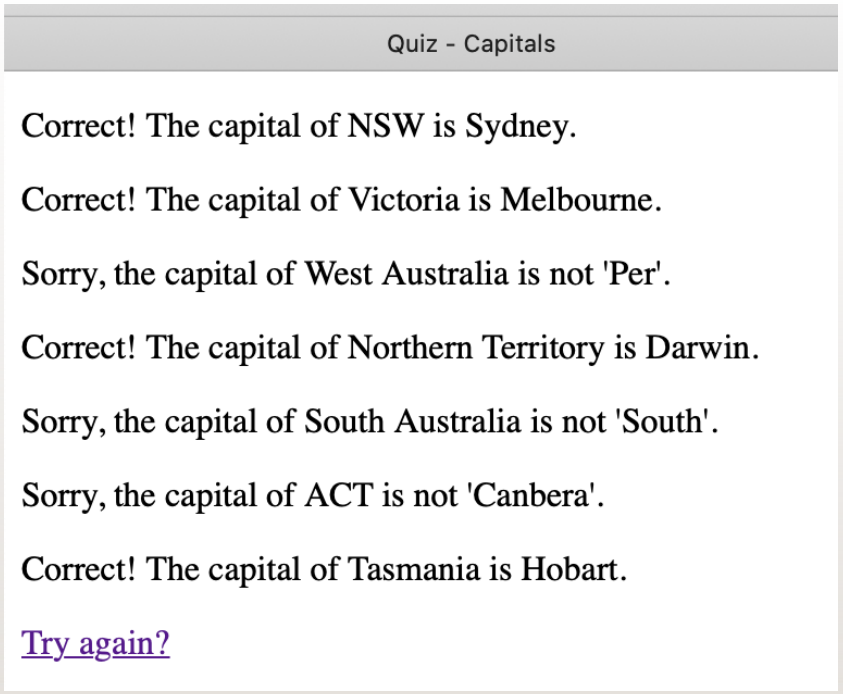
USING ARRAYS IN WEB FORMS EXAMPLE

```
<form action='ProcessForm.php' method='post' >
<p>Enter the first answer: <input type='text' name='answers[]' /></p>
<p>Enter the second answer:<input type='text' name='answers[]' /></p>
<p>Enter the third answer:<input type='text' name='answers[]' /></p>
<input type='submit' name='submit' value='submit' />
</form>

---
if (is_array($_POST['answers']) {
     $Index = 0;
     foreach ($_POST['answers'] as $Answer) {
       ++$Index;
       echo "The answer for question $Index is '$Answer' <br />\n";
     }
}
```

USING ARRAYS IN WEB FORMS EXAMPLE – QUIZ CAPITALS





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)

```
$Ounces = array(1, 0.125, 0.0625, 0.03125,0.0078125);
$Cups = array(8, 1, 0.5, 0.25, 0.0625);
$Pints = array(16, 2, 1, 0.5, 0.125);
$Quarts = array(32, 4, 2, 1, 0.25);
$Gallons = array(128, 16, 8, 4, 1);
$VolumeConversions = array($Ounces, $Cups, $Pints,
                                    $Quarts, $Gallons);
```

	Ounces	Cups	Pints	Quarts	Gallons
Ounces	1	0.125	0.0625	0.03125	0.0078125
Cups	8	1	0.5	0.25	0.0625
Pints	16	2	1	0.5	0.125
Quarts	32	4	2	1	0.25
Gallons	128	16	8	4	1

CREATING TWO-DIMENSIONAL ASSOCIATIVE ARRAYS

```
$Ounces = array("ounces" => 1, "cups" => 0.125,
          "pints" => 0.0625, "quarts" => 0.03125,
          "gallons" => 0.0078125);
$Cups = array("ounces" => 8, "cups" => 1,
          "pints" =>0.5, "quarts" => 0.25,
          "gallons" => 0.0625);
$Pints = array("ounces" => 16, "cups" => 2,
          "pints" =>1, "quarts" => 0.5,
          "gallons" => 0.125);
$Quarts = array("ounces" => 32, "cups" => 4,
          "pints" =>2, "quarts" => 1,
          "gallons" => 0.25);
$Gallons = array("ounces" => 128, "cups" => 16,
     "pints" =>8, "quarts" => 4, "gallons" => 1);
```

CREATING TWO-DIMENSIONAL ASSOCIATIVE ARRAYS (CONTINUED)

```
$VolumeConversions = array("Ounces" => $Ounces,
     "Cups" => $Cups, "Pints" => $Pints,
     "Quarts" => $Quarts, "Gallons" => $Gallons);
```

Keys ↓

	"Ounces"	"Cups"	"Pints"	"Quarts"	"Gallons"
"Ounces"	1	0.125	0.0625	0.03125	0.0078125
"Cups"	8	1	0.5	0.25	0.0625
"Pints"	16	2	1	0.5	0.125
"Quarts"	32	4	2	1	0.25
"Gallons"	128	16	8	4	1

← Keys

} Elements

} Elements

CREATING MULTIDIMENSIONAL ARRAYS WITH A SINGLE STATEMENT

```
$VolumeConversions = array(
      array(1, 0.125, 0.0625, 0.03125, 0.0078125), // Ounces
      array(8, 1, 0.5, 0.25, 0.0625), // Cups
      array(16, 2, 1, 0.5, 0.125), // Pints
      array(32, 4, 2, 1, 0.25), // Quarts
      array(128, 16, 8, 4, 1) // Gallons
);
```

OR

CREATING MULTIDIMENSIONAL ARRAYS WITH A SINGLE STATEMENT

```
$VolumeConversions = array(
"ounces" => array("ounces" => 1,"cups" => 0.125,
                  "pints" => 0.0625, "quarts" => 0.03125,
                   "gallons" => 0.0078125),
"cups" => array("ounces" => 8, "cups" => 1, "pints" =>0.5,
           "quarts" => 0.25, "gallons" => 0.0625),
"pints" => array("ounces" => 16, "cups" => 2, "pints" =>1,
           "quarts" => 0.5, "gallons" => 0.125),
"quarts" => array("ounces" => 32, "cups" => 4, "pints" =>2,
           "quarts" => 1, "gallons" => 0.25),
"gallons" => array("ounces" => 128, "cups" => 16,
           "pints" =>8, "quarts" => 4, "gallons" => 1));
```

ARRAYS – MORE EXAMPLES

- Example 1
- Example 2
- Example 3