INTRODUCTION TO WEB PROGRAMMING II

A

LECTURE NOTE

PREPARED

BY

MUHAMMAD SALISU ALI

FEDERAL UNIVERSITY DUTSE

16 APRIL, 2021

How to work with an array

- An array is a data type that contains one or more items called elements. Each element stores a value that you can refer to with an index. The length of an array indicates the number of elements that it contains.
- By default, PHP uses integer indexes where 0 is the first element, 1 is the second element, and so on.
- The syntax for creating an array

  $array_name = array([value1[, value2, ...]])
- The syntax for referring to an element of an array

  $array_name[index];
- How to create an array of names
  - With one statement

    $names = array('Muhammad Ali', 'Hassan Garba', 'Fatima Ahmad', 'Haruna Abubakar');
  - With multiple statements

    $names = array();

    $names[0] = 'Muhammad Ali';

    $names[1] = 'Hassan Garba';

    $names[2] = 'Fatima Ahmad';

- How to create an array of discounts With one statement

```
$discounts = array(0, 5, 10, 15);
```

  - With multiple statements

```
$discounts = array();
$discounts[0] = 0;
$discounts[1] = 5;
$discounts[2] = 10;
$discounts[3] = 15;
```

- How to use the print_r function to view an array’s indexes and values

```
print_r($names); // Output: Array ( [0] => Muhammad Ali [1] => Hassan Garba [2] =>
Fatima Ahmad)
```

- **How to add and delete elements**
  - Arrrays in PHP are dynamic, you can change the length of an array by adding or removing elements from the array.
  - To add an element to the end of an array, you can leave the index out of the brackets when assigning a value to the array. Then, PHP takes the highest index, adds one to it, and uses that value as the index for the new element.
  - You can set and get elements from an array by coding the index for the element between brackets.
  - When you add or delete elements from an array, you may leave gaps in the array that contain NULL values.
  - You can access array elements using variable substitution in a double-quoted string. If necessary, you can place braces around the array name and index to separate the array element from the rest of the text in the string.
  - The syntax for adding an element to the end of an array
    - ✓ $array_name[] = $value;
  - Functions for removing the values from elements in an array
    - ✓ **unset(**$var1[, $var2 ...]**)** - Deletes the value in the specified array element or deletes the entire array by setting it to a NULL value.
    - ✓ **array_values(**$array**)** - Returns all values from the specified array after any NULL values have been removed and the array has been reindexed.

- How to add a value to the end of an array
  - ✓ $letters = array('a', 'b', 'c', 'd');
  - ✓ $letters[] = 'e';
- How to set a value at a specific index
  - ✓ $letters = array('a', 'b', 'c', 'd');
  - ✓ $letters[0] = 'e';
  - ✓ $letters[3] = 'f';
  - ✓ $letters[5] = 'g';
- How to remove elements that contain NULL values and reindex an array
  - ✓ $letters = array('a', 'b', 'c', 'd');
  - ✓ unset($letters[2]);
  - ✓ $letters = array_values($letters);
- How to use array elements with variable substitution

```
$name = array (‘Zainab', ‘Salisu');
echo "First Name: $name[0]";
echo "First Name: {$name[0]}";
```

- How to get values from an array
  - ✓ $letters = array('a', 'b', 'c', 'd');
  - ✓ $letter1 = $letters[0];
  - ✓ $letter2 = $letters[1];
  - ✓ $letter4 = $letters[4];
- How to delete values from an array
  - ✓ $letters = array('a', 'b', 'c', 'd');
  - ✓ unset($letters[2]);
  - ✓ unset($letters);
- How to use for loops to work with arrays
  - ✓ For loops are commonly used to process the data in arrays. In this case, the counter for the loop is used as the index for each element in the array.
- Functions for loops that work with arrays
  - ✓ **count(**$array**)** - Returns the number of elements in an array. This function doesn't count gaps in the array.
  - ✓ **end(**$array**)** - Moves the cursor for the array to the last element in the array.
  - ✓ key($array) - Returns the index of the array element that the cursor is on.
  - ✓ **isset(**$var**)** - Returns a TRUE value if the specified variable or array element contains a value. Otherwise, it returns a FALSE value.

- Code that stores 10 random numbers in an array

```
$numbers = array();
for ($i = 0; $i < 10; $i++) {
   $numbers[] = mt_rand(1, 100);
}
```

- Code that displays the elements of an array

```
$numbers_string = ‘ ';
for ($i = 0; $i < count($numbers); $i++) {
       $numbers_string .= $numbers[$i] . ' ';
}
echo $numbers_string;
```

- Code that computes the sum and average of an array of prices

```
$prices = array(141.95, 212.95, 411, 10.95);
$sum = 0;
for ($i = 0; $i < count($prices); $i++) {
    $sum += $prices[$i];
}
$average = $sum / count($prices);
```

- How to skip gaps in an array

```
$numbers = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
unset($numbers[2], $numbers[6]);
end($numbers); $last = key($numbers);
$numbers_string = '';
for($i = 0; $i <= $last; $i++) {
    if (isset($numbers[$i])) {
        $numbers_string .= $numbers[$i] . ' ';
    }
}
echo $numbers_string;       // Displays: 1 2 4 5 6 8 9 10
```

How to create and use an associative array

- In PHP, you can also use strings as index values. When an array uses strings for its indexes, the array is known as an **associative array**. The indexes in an associative array are often called *keys*.
- An associative array uses a string as the index for the value that's stored in the array. When using an associative array, the index is commonly called a key.
- An array with integer keys that have gaps between them can also be thought of as an associative array.
- Although not a good practice, an array can have both integer and string indexes.
- The syntax for creating an associative array

  array([key1 => value1, key2 => value2, ...])
- How to create an associative array of state tax rates
  - With one statement
    - ✓ $tax_rates = array('NC' => 7.75, 'CA' => 8.25, 'NY' => 8.875);
  - With multiple statements
    - ✓ $tax_rates = array();
    - ✓ $tax_rates['NC'] = 7.75;
    - ✓ $tax_rates['CA'] = 8.25;
    - ✓ $tax_rates['NY'] = 8.875;

- How to create an associative array of country codes
  - With one statement

```
$country_codes = array('DEU' => 'Germany', 'JPN' => 'Japan', 'ARG' => 'Argentina', 'USA' =>
'United States');
```

  - With multiple statements

```
$country_codes = array();
$country_codes['DEU'] = 'Germany';
$country_codes['JPN'] = 'Japan';
$country_codes['ARG'] = 'Argentina';
$country_codes['USA'] = 'United States';
```

- How to create an associative array of telephone extensions
  - ✓ $ext = array();
  - ✓ $ext[10] = 'Sales';
  - ✓ $ext[13] = 'Customer Service';
  - ✓ $ext[16] = 'Returns';
  - ✓ $ext[18] = 'Warehouse';
- How to create an array that contains integer and string indexes
  - ✓ $employees = array();
  - ✓ $employees[0] = Mukhtar';
  - ✓ $employees[1] = ‘Rabiu';
  - ✓ $employees[2] = 'John';
  - ✓ $employees['senior'] = ‘Faisal';
  - ✓ $employees['newest'] = 'Prince';
- How to use the print_r function to view an array
  - ✓ print_r($tax_rates);

  // Output: Array ([NC] => 7.75 [CA] => 8.25 [NY] => 8.875)

- How to add and delete elements
- To set or get a value in an associative array, specify the key for the value within the brackets.
- If you use an empty pair of brackets to add an element to an array, PHP uses an integer key for the element, which usually isn't what you want for an associative array.
- You can access array elements using variable substitution in a double-quoted string. To do that for a string key, you don't need to code quotes around the key. However, if you place braces around the element to separate it from other text in the string, you must include quotes around the key.
- How to set a value with a specific key
  - ✓ $name = array('first' => 'Ray', 'last' => 'Harris');
  - ✓ $name['middle'] = 'Thomas';
- What happens when you omit the key when adding a value
  - ✓ $name = array('first' => 'Ray', 'last' => 'Harris');
  - ✓ $name[] = 'Thomas‘;
- How to get a value at a specified key
  - ✓ $name = array('first' => 'Ray', 'last' => 'Harris');
  - ✓ $first_name = $name['first'];
  - ✓ $last_name = $name['last'];

- How to delete values from an array
  - ✓ $name = array('first' => 'Ray', 'last' => 'Harris');
  - ✓ unset($name['first']);
  - ✓ unset($name);
- How to use variable substitution with array elements
  - ✓ $name = array('first' => 'Ray', 'last' => 'Harris');
- How to set a value with a specific key
  - ✓ echo "First Name: $name['first']";
  - ✓ echo "First Name: $name[first]";
  - ✓ echo "First Name: {$name['first']}";
- You can use a foreach statement to create a foreach loop that accesses only those elements in an array that are defined.
- The syntax of a foreach loop

```
foreach ($array_name as [ $key => ] $value) {
   // Statements that use $key and $value
}
```

- A foreach loop that displays the values in an associative array

```
$tax_rates = array('NC' => 7.75, 'CA' => 8.25, 'NY' => 8.875);
echo '<ul>';
foreach ($tax_rates as $rate) {
   echo '<li>$rate</li>';
}
echo '</ul>';
```

- A foreach loop that displays the keys and values

```
$tax_rates = array('NC' => 7.75, 'CA' => 8.25, 'NY' => 8.875);
echo '<ul>';
foreach ($tax_rates as $state => $rate) {
    echo '<li>$state ($rate)</li>';
}
echo '</ul>';
```

- A foreach loop that displays the values in a regular array

```
$numbers = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
unset($numbers[2], $numbers[6]);
$numbers_string = ‘’;
foreach($numbers as $number) {
  $numbers_string .= $number . ' ';
}
echo $numbers_string;
```

- More examples
  - ✓ $array = array(1, 1, 1, 1, 1, 8 => 1, 4 => 1, 19, 3 => 13);
  - ✓ $firstquarter = array(1 => 'January', 'February', 'March');
  - ✓ $sweeties = array("a" => "pear", "b" => "strawberry", "c" => "cherry");

Multi-dimensional Arrays

- Multi-dimensional array is an array or arrays within another array.
- Has more than one index or key. The following example shows how to create and use multi-dimensional array.

```
$arrays = array(
        array(
            'name' => 'Muhammad Salis Ali',
            'email' => 'msa@gmail.com',
            'phone' => '07036322054'
        ),
        array(
            'name' => 'Radiya Ahmad',
            'email' => 'hky@gmail.com',
            'phone' => '08132833361'
        ),
        array(
            'name' => 'Hassan Garba',
            'email' => 'kanti@yahoo.com',
            'phone' => '08142568925'
        )
 );
```

- How to access the elements of a multi-dimensional array
  - ✓ echo 'Name: ' . $arrays[0]['name'].'<br>';
  - ✓ echo 'Email: ' . $arrays[0]['email'].'<br>';
  - ✓ echo 'Phone: ' . $arrays[0]['phone'].'<br>';
  - ✓ echo 'Name: ' . $arrays[1]['name'] . '<br>';
  - ✓ echo 'Email: ' . $arrays[1]['email'].'<br>';
  - ✓ echo 'Phone: ' . $arrays[1]['phone'].'<br>';
  - ✓ echo 'Name: ' . $arrays[2]['name'] . '<br>';
  - ✓ echo 'Email: ' . $arrays[2]['email'].'<br>';
  - ✓ echo 'Phone: ' . $arrays[2]['phone'].'<br>';
- You can use **foreach** loop to display all the elements of multidimensional array
- **print_r** can be used to debug or display all the elements of a multidimensional array.

- Other examples

```
 $fruits = array (
     "fruits"  => array("a" => "orange", "b" => "banana", "c" => "apple"),
     "numbers" => array(1, 2, 3, 4, 5, 6),
     "holes"   => array("first", 5 => "second", "third")
 );
```

- Some other functions for working with arrays
  - ✓ array_fill — Fill an array with values
  - ✓ array_product — Calculate the product of values in an array
  - ✓ array_push — Push one or more elements onto the end of array
  - ✓ array_rand — Pick one or more random keys out of an array
  - ✓ array_reduce — Iteratively reduce the array to a single value using a callback function
  - ✓ array_replace_recursive — Replaces elements from passed arrays into the first array recursively
  - ✓ array_replace — Replaces elements from passed arrays into the first array
  - ✓ array_reverse — Return an array with elements in reverse order
  - ✓ array_search — Searches the array for a given value and returns the first corresponding key if successful

- array_fill (int $start_index , int $num , mixed $value) : array - fills an array with num entries of the value of the value parameter, keys starting at the start_index parameter.
  - ✓ $a = array_fill(5, 6, 'banana');
  - ✓ $b = array_fill(-2, 4, 'pear');
- array_product (array $array) : number - returns the product of values in an array.
  - ✓ $c = array(2, 4, 6, 8);
  - ✓ echo "product(c) = " . array_product($c) . "\n";
- array_sum(array $array) : number - returns the sum of values in an array.
  - ✓ $a = array(2, 4, 6, 8);
  - ✓ echo "sum(a) = " . array_sum($a) . "\n";
- array_push (array &$array [, mixed $...]) : int - treats array as a stack, and pushes the passed variables onto the end of array. The length of array increases by the number of variables pushed.
  - ✓ $stack = array("orange", "banana");
  - ✓ array_push($stack, "apple", "raspberry");
- array_pop (array &$array) : mixed - pops and returns the value of the last element of array, shortening the array by one element.
  - ✓ $stack = array("orange", "banana", "apple", "raspberry");
  - ✓ $fruit = array_pop($stack);

- array_shift (array &$array) : mixed - shifts the first value of the array off and returns it, shortening the array by one element and moving everything down.
  - ✓ $stack = array("orange", "banana", "apple", "raspberry");
  - ✓ $fruit = array_shift($stack);
- array_unshift (array &$array [, mixed $...]) : int - prepends passed elements to the front of the array.
  - ✓ $queue = array("orange", "banana");
  - ✓ array_unshift($queue, "apple", "raspberry");
- array_slice (array $array , int $offset [, int $length = NULL [, bool $preserve_keys = FALSE]]) : array - returns the sequence of elements from the array array as specified by the offset and length parameters.
  - ✓ $input = array("a", "b", "c", "d", "e");
  - ✓ $output = array_slice($input, 2);
  - ✓ $output = array_slice($input, -2, 1);
  - ✓ $output = array_slice($input, 0, 3);
- array_search(mixed $needle , array $haystack [, bool $strict = FALSE]) : mixed - Searches haystack for needle. If the third parameter strict is set to TRUE then the function will search for identical elements in the haystack.
  - ✓ $array = array(0 => 'blue', 1 => 'red', 2 => 'green', 3 => 'red');
  - ✓ $key = array_search('green', $array); // $key = 2;
  - ✓ $key = array_search('red', $array);   // $key = 1;

- array_splice(array &$input , int $offset [, int $length = count($input) [, mixed $replacement = array()]]) : array - Removes the elements designated by offset and length from the input array, and replaces them with the elements of the replacement array, if supplied.
  - ✓ $input = array("red", "green", "blue", "yellow");
  - ✓ array_splice($input, 2);

    // $input is now array("red", "green")
  - ✓ $input = array("red", "green", "blue", "yellow");
  - ✓ array_splice($input, 1, -1);

    // $input is now array("red", "yellow")
  - ✓ $input = array("red", "green", "blue", "yellow");
  - ✓ array_splice($input, 1, count($input), "orange");

    // $input is now array("red", "orange")
  - ✓ $input = array("red", "green", "blue", "yellow");
  - ✓ array_splice($input, -1, 1, array("black", "maroon"));

    // $input is now array("red", "green", "blue", "black", "maroon")
  - ✓ $input = array("red", "green", "blue", "yellow");
  - ✓ array_splice($input, 3, 0, "purple");