PHP Arrays

PHP equality operators

- The equality operators
  - == Equal
  - != Not equal
  - <> Not equal
- The equality operators perform type coercion. Type coercion converts data from one type to another. PHP follows these rules that are listed in the table A.
- When converting to the Boolean type, these are false: null, 0, 0.0, “0”, an empty string, and an empty array. All others are true.
- The identity operators don not perform type coercion. If two operands are of different types, the results are false.
  - === Equal
  - !== Not equal

Table A

Operand1	Operand2	Action
Null	***String***	***Convert null to an empty string and then compare two strings***
Boolean or null	***Not a string***	***Convert both to Boolean and compare***
String	***Number***	***Convert string to number and compare two numbers***
Numeric String	***Numeric String***	***Convert strings to numbers and compare two numbers***
Text string	***Text string***	***Compare strings as if using the strcmp function***

Examples of comparison

```
echo 'null == \'\' is '.(null == '').'<br>';    // true as null is coverted to the empty string
echo 'null == false is '.(null == false).'<br>'; // true as null is coverted to false
echo 'null == 0 is '.(null == 0).'<br>';     // true as null is coverted to zero
echo 'false == \'0\' is '.(false == '0').'<br>';  // true as empty strings are converted to false
echo 'true == \'false\' is '.(true == 'false').'<br>';  // true as non-empty strings are converted to
true
echo '3.5 == "\n3.5 inches" is '.(3.5 == "\n3.5 inches").'<br>';  // true as the string is converted to
a number and then compare
echo 'INF == \'INF\' is '.(INF == 'INF').'<br>';  // false as the 'INF' is converted to zero
echo '0 == \'\' is '.(0 == '').'<br>';       // true as the empty string is converted to zero
echo '0 == \'harry\' is '.(0 == 'harry').'<br>';  // true as any string that is not numeric is converted
to zero
echo '<br>'.'<br>';

echo 'null === false is '.(null === false).'<br>';
echo '3.5 !== "\n3.5 inches" is '.(3.5 !== "\n3.5 inches").'<br>';
```

Arrays

- An array can store one or more elements. Each element consists of an index and a value. An index can be either an integer or a string. A value can be any PHP data type.
- By default, PHP uses integer indexes that starts from zero.
- unset() function
  - Deletes the value in the specified array elements. Or set the array to NULL to delete the empty array
- array_values()
  - Returns all values from the specified array after any NULL values have been removed and the array has been reindexed.
- count($array)
  - Returns the number of elements in an array. This function doesn’t count gaps in the array.
- end($array)
  - Moves the cursor to the last element in the array.
- key($array)
  - Returns the index of the array elements that the cursor is on.
- Isset($var)
  - Returns a true if the specified array element contains a value.

Examples of arrays

```
$names = array('Nancy', 'Peter', 'David');

$names2 = array();
$names2[0] = 'Nancy';
$names2[1] = 'Peter';
$names2[2] = 'David‘

unset($names2[1]);            // delete Peter

$names3 = array_values($names2);        // get a new array {Nancy, David)

end($names2);
$last = key($names2);
$names_string = "";
for ($i = 0; $i < count($names3); $i++) {
   if (isset($names3[$i]))
      $names_string .= $names3[$i].' ';
}
echo $names_string.'<br>';
```

Associative arrays

- An associative array uses a string as the index for the value that’s stored in the array. And the index is called a key.
- An array with integer keys that have gaps between them can also be called an associative array.

Examples of associative arrays

```
$keys = array('C#', 'Java', 'PHP');
$books = array('C#' => 59.99, 'Java' => 49.99,
'PHP' => 52.99);

for($i = 0; $i < count($keys); $i++) {
   echo $books[$keys[$i]];
}

echo '<br>integer keys...<br>';
$keys = array(0, 5, 10, 15, 20);
$prices = array();
$prices[0] = 56.34;
$prices[5] = 46.34;
$prices[10] = 76.34;
$prices[15] = 86.34;
$prices[20] = 96.34;

echo '<br>Adding...<br>';
$keys[5] = 6;
$prices[6] = 77.77;

echo '<br>Deleting...<br>';
unset($prices[5]);
```

Loop using foreach

```
$books = array('C#' => 59.99, 'Java' => 49.99, 'PHP'
=> 52.99);
echo '<ul>';
foreach ($books as $price) {
   echo "<li>$price</li>";
}
echo '</ul>';

echo '<br><ul>';
foreach ($books as $title => $price) {
   echo "<li>$title = $price</li>";
}
echo '</ul>';

$prices = array();
$prices[0] = 56.34;
$prices[5] = 46.34;
$prices[10] = 76.34;
$prices[15] = 86.34;
$prices[20] = 96.34;

unset($prices[5], $prices[15]);
echo '<br><ul>';
foreach ($prices as $key => $price) {
   echo "<li>$key = $price</li>";
}
echo '</ul>';
```

Array functions

- Please refer to the following URL for array functions:
  - http://www.php.net/manual/en/function.array-diff-key.php
- The range() function creates an array containing a range of elements. This function returns an array of elements from low to high. Note: If the low parameter is higher than the high parameter, the range array will be from high to low.
- array_fill — Fill an array with values
- array_pad — Pad array to the specified length with a value
- array_merge — Merge one or more arrays
- array_slice — Extract a slice of the array
- array_splice — Remove a portion of the array and replace it with something else

Examples

```
$numbers = range(1, 5);
foreach ($numbers as $num) {
    echo "$num";
}

$numbers = range(1, 5, 2);
foreach ($numbers as $num) {
    echo "$num";
}

$numbers = array_fill(2, 5, 8);
foreach ($numbers as $key => $num) {
    echo "$key -> $num";
}

$numbers = array_pad($numbers, 10, 1);
foreach ($numbers as $key => $num) {
    echo "$key -> $num";
}

$numbers1 = array(10, 20);
$numbers2 = array(30, 40, 50);
$numbers = array_merge($numbers1, $numbers2);
echo implode(', ', $numbers).'<br>';

$numbers = array_slice($numbers2, 1);
echo implode(', ', $numbers).'<br>';

$numbers1 = array(10, 20, 90, 100);
array_splice($numbers1, 1, 2, $numbers2);
echo implode(', ', $numbers1).'<br>';
```

More arrays functions

- The array_sum() function returns the sum of all the values in the array.
- The in_array() function searches an array for a specific value.
- The array_key_exists() function checks an array for a specified key, and returns true if the key exists and false if the key does not exist.
  - Tip: Remember that if you skip the key when you specify an array, an integer key is generated, starting at 0 and increases by 1 for each value.
- The array_search() function search an array for a value and returns the key.
- The array_count_values() function counts all the values of an array.

Examples

```
$numbers = array(10, 20, 30, 40, 50);
$sum = array_sum($numbers);
echo "sum = $sum".'<br>';

$books = array('C#' => 59.99, 'Java' => 49.99, 'PHP' => 52.99);
$is_found = in_array(49.99, $books); // true
echo "is_found = $is_found".'<br>';
$is_found = in_array('49.99', $books); // true
echo "is_found = $is_found".'<br>';
$is_found = in_array('49.99', $books, true); // false
echo "is_found = $is_found".'<br>';

$key_exists = array_key_exists('Java', $books); // true
echo "key_exists = $key_exists".'<br>';
$key = array_search(49.99, $books);            // Java
echo "key = $key".'<br>';

$books = array('C#','C', 'C','Java', 'C++', 'C#', 'C', 'Java', 'Java', 'PHP', 'C');
$frequency = array_count_values($books);
echo '<br><ul>';
foreach ($frequency as $key => $occurrences) {
    echo "<li>$key -> $occurrences</li>";
}
echo '</ul>';
```

Sorting arrays

- The sort() function sorts an indexed array in ascending order.
- The rsort() function sorts an indexed array in descending order.
- The asort() function sorts an associative array in ascending order, according to the value.
- The arsort() function sorts an associative array in descending order, according to the value.
- The ksort() function sorts an associative array in ascending order, according to the key.
- The krsort() function to sort an associative array in descending order, according to the key.

Sort arrays

```
$fruit = array('orange', 'apple', 'lemon', 'peaches', 'cherries', 'strawberries', 'grapes');
sort($fruit);
echo implode(',', $fruit); //apple,cherries,grapes,lemon,orange,peaches,strawberries

echo '<br>';
$numbers = array(90, '20', 30, 40, '10');
sort($numbers, SORT_NUMERIC);
echo implode(',', $numbers); //10,20,30,40,90
echo '<br>';
rsort($numbers, SORT_NUMERIC);
echo implode(',', $numbers); //90,40,30,20,10

$books = array('C#' => 51.99, 'PHP' => 49.99, 'Java' => 52.99);
asort($books);
print_array($books);
ksort($books);
print_array($books);
arsort($books);
print_array($books);
krsort($books);
print_array($books);

function print_array($books) {
    echo '<br><ul>';
    foreach ($books as $key => $price) {
        echo "<li>$key -> $price</li>";
    }
    echo '</ul>';
}
```

Manipulating Arrays

- The array_unique($array) function removes duplicate values from an array. If two or more array values are the same, the first appearance will be kept and the other will be removed.
- The array_reverse($array) function returns an array in the reverse order.
- The shuffle($array) function randomizes the order of the elements in the array. This function assigns new keys for the elements in the array. Existing keys will be removed.
- The array_rand($array,$num) function returns a random key from an array, or it returns an array of random keys if you specify that the function should return more than one key.

Examples

```
$books = array('C#','C', 'C','Java', 'C++', 'C#', 'C', 'Java', 'Java', 'PHP', 'C');
$books_unique = array_unique($books); //C#,C,Java,C++,PHP
echo implode(',', $books_unique);

echo '<br>';
$books_reverse = array_reverse($books_unique); //PHP,C++,Java,C,C#
echo implode(',', $books_reverse);

echo '<br>';
$books_random = $books_reverse;
shuffle($books_random); //PHP,C++,Java,C,C#
echo implode(',', $books_random);


echo '<br>';
$books = array('C#' => 51.99, 'PHP' => 49.99, 'JAVA' => 52.99);
$book_reverse = array_reverse($books);
print_array($book_reverse);

function print_array($books) {
    echo '<br><ul>';
    foreach ($books as $key => $price) {
        echo "<li>$key -> $price</li>";
    }
    echo '</ul>';
}
```

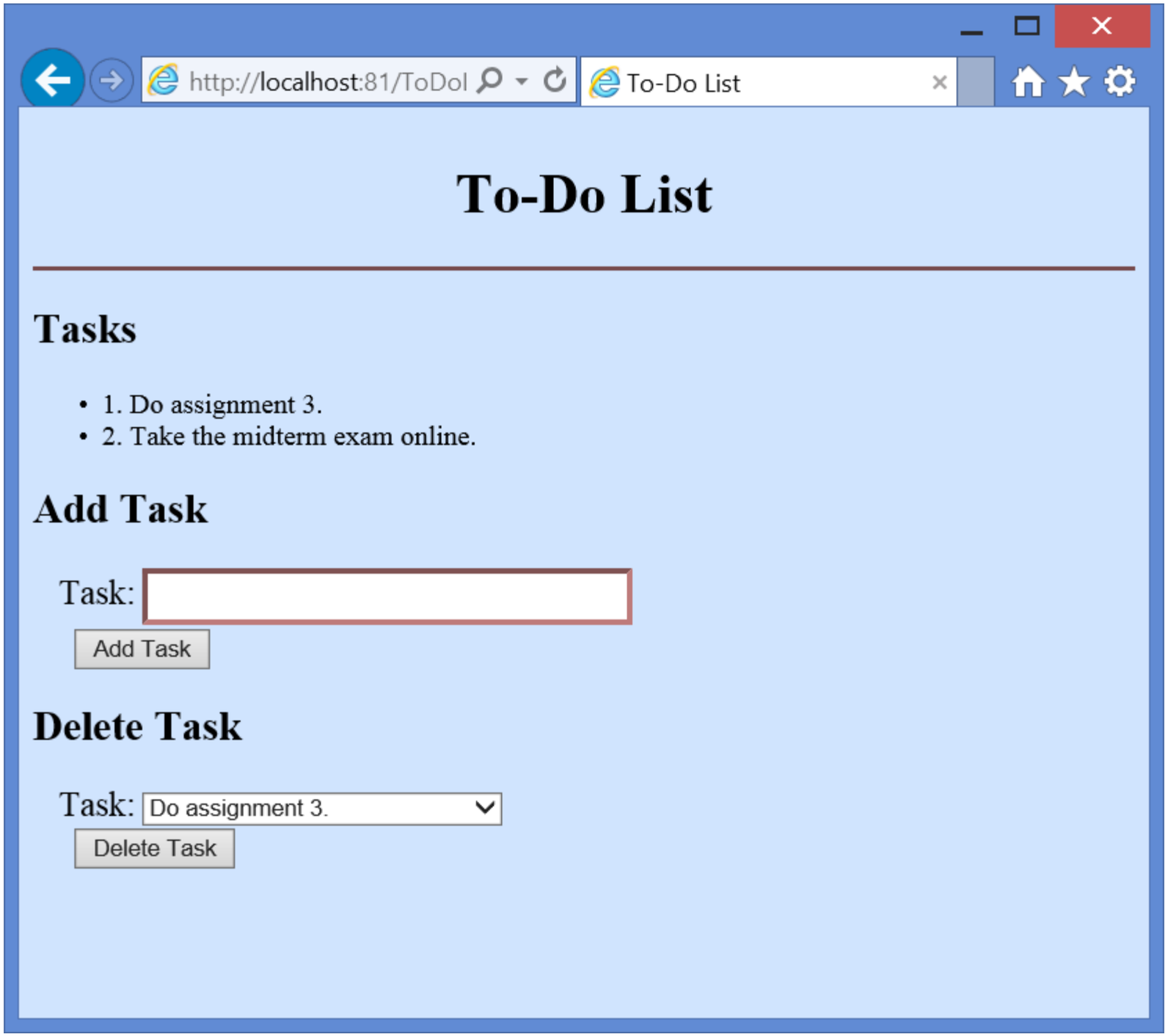
Array of arrays

```
$matrix = array();
for ($i = 0; $i < 5; $i++) {
    $matrix[$i] = array();
}
// add values to each element
for ($i = 0; $i < 5; $i++) {
    for ($j = 0; $j < 5; $j++) {
        $matrix[$i][$j] = $i * 5 +$j;
    }
}
// print 5 arrays of arrays
for ($i = 0; $i < 5; $i++) {
    echo implode(',  ', $matrix[$i]);
    echo '<br>';
}

// Create a shopping cart
$items = array();
$items['id'] = 1234;
$items['name'] = 'computer';
$items['price'] = 1200;
$cart[] = $items;

$items = array();
$items['id'] = 2234;
$items['name'] = 'software';
$items['price'] = 100;
$cart[] = $items;

// print 5 arrays of arrays
for ($i = 0; $i < 2; $i++) {
    echo implode(',  ', $cart[$i]);
    echo '<br>';
}
```

Case Study: To-do List

- The to-do list application demonstrates the use of arrays.
- The application shows a list of tasks, and lets the user add a new task to the list, and it includes a delete function that lets the user delete a task from the list.

The user-interface of To-do list



The MVC pattern of To-do list