Introduction to PHP arrays

By definition, an array is a list of elements. So, for example, you may have an array that contains a list of products.

PHP provides you with two types of arrays: **indexed** and **associative**.

The keys of the indexed array are integers that start at 0. Typically, you use indexed arrays when you want to access the elements by their positions.

The keys of an associative array are strings. And you use associative arrays when you want to access elements by string keys.

Indexed arrays

Creating arrays

In PHP, you can use the array() construct or [] syntax to define an array. The [] syntax is shorter and more convenient.

1) **Creating an array using array() construct**

To define an array, you use the array() construct. The following example creates an empty array:

```
<?php

$empty_array = array();
```

To create an array with some initial elements, you place a comma-separated list of elements within parentheses of the array() construct.

For example, the following defines an array that has three numbers:

```php
<?php

$scores = array(1, 2, 3);
```
Code language: HTML, XML (xml)

2) **Creating an array using the [] syntax**

PHP provides a more convenient way to define arrays with the shorter syntax [], known as JSON notation. The following example uses [] syntax to create a new empty array:

```php
<?php

$empty_array = [];
```

The following example uses the [] syntax to create a new array that consists of three numbers:

```php
<?php

$scores = [1, 2, 3];
```

Displaying arrays

To show the contents of an array, you use the var_dump() function. For example:

```php
<?php

$scores = [1, 2, 3];
var_dump($scores);
```

Output:

```
array(3) {
  [0]=> int(1)
  [1]=> int(2)
  [2]=> int(3)
}
```

Or you can use the print_r() function:

```php
<?php
```

```
$scores = array(1, 2, 3);
print_r($scores);
```

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
)
```

To make the output more readable, you can wrap the output of the print_r() function inside a <pre> tag. For example:

```
<?php

$scores = [1, 2, 3];

echo '<pre>';
print_r($scores);
echo '</pre>';
```

Output:

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
)
```

It’s more convenient to define a function that prints out an array like this:

```
<?php

function print_array($data)
{
        echo '<pre>';
        print_r($data);
        echo '</pre>';
}
```

```
$scores = [1, 2, 3];

print_array($scores);
```

Output:

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
)
```

And then you can reuse the function whenever you want to display an array.

Accessing array elements

To access an element in an array, you specify the index of the element within the square brackets:

```
$array_name[index]
```

Note that the index of the first element of an array begins with zero, not one.

The following example shows how to access the first element of the array:

```
<?php

$scores = [1, 2, 3];
echo $scores[0];
```

Output:

```
1
```

Adding an element to the array

To add an element to an array, you use the following syntax:

```
$array_name[] = new_element;
```

PHP will calculate the highest numerical index plus one each time you assign an element to the array.

The following example shows how to add the number 4 to the $scores array:

```
<?php

$scores = [1, 2, 3];
$scores[] = 4;
```

In this example, we defined an array that consists of three numbers initially. Then, we added the number 4 to the array.

It's possible to use an index when you add a new element to the array. For example:

```
$scores = [1, 2, 3];
$scores[3] = 4;
```

But doing this, you have to calculate the new index manually. It is not practical. Also, if the index is already is used, the value will be overwritten.

Changing array elements

The following statement changes the element located at the index to the $new_element:

```
$array_name[index] = $new_element;
```

For example, to change the first element of the $scores array from 1 to zero, you do it as follows:

```
<?php

$scores = [1, 2, 3];
$scores[0] = 0;
```

Removing array elements

To remove an element from an array, you use the unset() function. The following removes the second element of the $scores array:

```
<?php

$scores = [1, 2, 3];
unset($scores[1]);
```

Getting the size of an array

To get the number of elements in an array, you use the count() function. For example:

```
<?php

$scores = [1, 2, 3, 4, 5];

echo count($scores);
```

Output:

```
5
```

Summary

- Use the array() construct or [] syntax to create a new array.
- For the indexed array, the first index begins with zero.
- To access an array element, use an index in the square bracket $array_name[index].
- Use the count() function to get the number of elements in an array.

Associative arrays

Associative arrays are arrays that allow you to keep track of elements by names rather than by numbers.

Creating associative arrays

To create an associative array, you use the array() construct:

```
<?php

$html = array();
```

or the JSON notation syntax:

```
<?php

$html = [];
```

Adding elements to an associative array

To add an element to an associative array, you need to specify a key. For example, the following adds the title to the $html array:

```
<?php

$html['title'] = 'PHP Associative Arrays';
$html['description'] = 'Learn how to use associative arrays in PHP';

print_r($html);
```

Output:

```
Array
(
    [title] => PHP Associative Arrays
    [description] => Learn how to use associative arrays in PHP
)
```

Accessing elements in an associative array

To access an element in an associative array, you use the key. For example, the following shows how to access the element whose key is title in the $html array:

```
<?php

$html['title'] = 'PHP Associative Arrays';
$html['description'] = 'Learn how to use associative arrays in PHP';

echo $html['title'];
```

Output:

```
PHP Associative Arrays
```

Summary

- Use an associative array when you want to reference elements by names rather than numbers.