PHP array function

```php
$array_of_strings = array("Ab", "C#", "D", "Fm");
$mixed_content = array("Shelby", 5, "GTO", 1964.5);
```

In PHP, an ordered array can be constructed with the built-in PHP function: `array()`. The `array()` function returns an array. Each of the arguments with which the function was invoked becomes an element in the array (in the order they were passed in).

Pass in arguments by including them, comma-separated, between the parentheses.

PHP array_pop function

```php
$some_numbers = ["1", 2, "three"];
echo array_pop($some_numbers); // "three"
// $some_numbers is now: ["1", 2]
```

The PHP built-in `array_pop()` function takes an array as its argument. It permanently removes the last element of an array and returns the removed element.

To use the function place an array or a variable with an array as its value in between the parentheses.

PHP unshift function

```php
$joke = [7, "ate", "nine!"];
$scared = array_unshift($joke, "Why is 6", "afraid of 7?",
"Because");
echo $scared; // 6
// $joke is now: ["Why is 6", "afraid of 7?", "Because", 7,
"ate", "nine!"]
```

The PHP built-in `array_unshift()` function takes an array as its first argument. The arguments that follow are elements to be added to the array.

The function adds each of the elements to the beginning of the array and returns the new number of elements in the array.

To use the function, place an array or a variable with an array as its value in between the parentheses.

Reassign Array Element

```php
$num_array = [1, 2, 3];

$num_array[1] = 4; // Reassign value of second element

echo $num_array[1]; // Prints: 4
```

In PHP, individual elements in an array can easily be reassigned. To do this, we access the location of an element in the array using square brackets (`[]`), and then assign a new value using the assignment operator (`=`).

Numerical Keys in Associative Arrays

```php
$num_array = [1000 => "one thousand", 100 => "one hundred",
600 => "six hundred"];
echo $num_array[1000]; // Prints: one thousand

$ordered = [99, 1, 7, 8];
$ordered["special"] = "Cool!";
echo $ordered[3]; // Prints: 8
echo $ordered["special"]; // Prints: Cool!

$num_array = [1000 => "one thousand", 100 => "one hundred",
600 => "six hundred"];
$num_array[] = "New Element in \$num_array";
echo $num_array[1001]; // Prints: New Element in $num_array

$animals_array = ["panda"=>"very cute", "lizard"=>"cute",
"cockroach"=>"not very cute"];
array_push($animals_array, "New Element in \$animals_array");
echo $animals_array[0]; // Prints: New Element in
$animals_array
```

Associative arrays can use integers or strings as keys.

In PHP, ordered arrays are just arrays in which integer keys have been assigned to the values automatically. For example, the first element is associated with key `0`, the second with `1`, etc. Even though ordered arrays are the same structure as associative arrays, it's recommended that you treat the two separately.

When adding an element to an array without specifying a key, PHP associates it with the "next" key. If no integer keys have been used, it will associate it with the key `0`, otherwise it will associate it one more than the largest integer used so far.

PHP Ordered Arrays

```php
$array_of_integers = array(3, 2, 1);
$array_of_strings = ["one", "2", "twenty"];
$array_of_mixed_content = ["two", 0, 1.0];
echo $array_of_integers[0]; // 3
echo $array_of_strings[2]; // "twenty"
echo $array_of_mixed_content[1] // 0
```

In PHP, an ordered array is a data structure representing a list of ordered, stored data. The location of an element in the array is known as its index. The elements in an ordered array are arranged in ascending numerical order starting with zero.

PHP Apending Arrays

```php
$string_array = ["first element", "second element"];
$string_array[] = "third element";
// $string_array is now:  ["first element", "second element",
"third element"]
```

In PHP, elements can be added to the end of an array by attaching square brackets ([]) at the end of the array's name, followed by typing the assignment operator (=), and then finally by typing the element to be added to the array.

PHP Count Function

```php
$devices = array("watch", "phone", "tablet", "laptop");
$number_of_devices = count($devices);
echo $number_of_devices; //Returns: 4
```

The built-in PHP `count()` function takes an array as its argument and returns the number of elements in that array.

To use the function place an array or a variable with an array as its value in between the parentheses.

PHP Nested Arrays

```php
$array_of_arrays = ["a", "Not that lucky", 2, ["smoke detectors", "d", ["boring street in", "U.S."], "nothing happens"], "."];
echo $array_of_arrays[3][2][1] //Returns "U.S."
$array_of_arrays[3][2][1] = "America"; // "U.S." changed to "America"
echo $array_of_arrays[3][2][1] //Returns "America"
```

In PHP, nested arrays are arrays that contain other arrays as elements. Chained operations can be used to access and/or change elements within a nested array.

PHP array_push function

```php
$some_numbers = ["1", 2, 3.0];
echo array_push($some_numbers, "four"); //Returns: 4
// $some_numbers is now: ["1", 2, 3.0, "four"]
```

The PHP built-in array_push() function takes an array as its first argument. The arguments that follow are elements to be added to the array.

The function adds each of the elements to the end of the array and returns the new number of elements in the array.

To use the function place an array or a variable with an array as its value in between the parentheses.

Accessing and Adding Elements

```php
$nums = ["2" => 2,"4" => 3, "6" => 6];

echo $nums["2"]; // Accesses value and Prints: 2

$nums["8"] = 8;  // Adds new value 8
echo $nums["8"]; // Prints: 8

$nums["4"] = 4;  // Changes value of key "4"
echo $nums["4"]; // Prints: 4
```

In PHP, we can access the value that a given key points to using square brackets (`[]`) and the key. To add new elements to an *associative array*, we can use the assignment operator (`=`). We can also change existing elements using the same syntax that adds new array elements.

PHP array_shift function

```php
$some_numbers = ["1", 2, "three"];
echo array_shift($some_numbers); // Returns "1"
// $some_numbers is now: [2, "three"]
```

The PHP built-in `array_shift()` function takes an array as its argument, permanently removes the first element from the array, and returns the removed element. All the elements in the array will be shifted to an index one smaller than their previous index.

To use the function, place an array or a variable with an array as its value in between the parentheses.

PHP Accessing Array Values

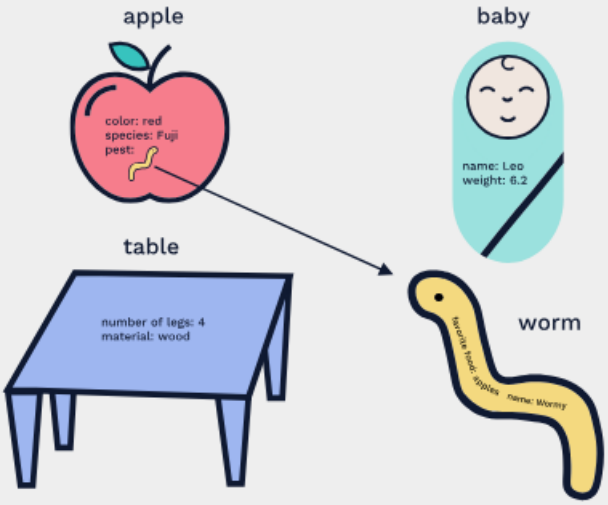
```php
$suit_up = ["top hat", "&", "tails"];
echo $suit_up[0]; // top hat

$index_value = 2;
$odd_choices = [99, "red", "balloons"];
echo $odd_choices[$index_value]; // balloons
```

In PHP, the individual elements in an array can be accessed using the array variable's name, and the location index surrounded by square brackets `[]`.

Remember that in PHP the location index starts at zero.

PHP Associative Array



In PHP, *associative arrays* are map-like structures, where keys are associated with values. When we need to access a specific value, we can use the *associated* key to find it.

In a PHP ordered array, the index locations are the keys. However, the PHP array type also allows us to assign meaningful keys to values. These data structures are called *associative arrays*.

For example, in the following associative array `table`, the key `"num_legs"` points to the value `4`, and the key `"material"` points to the value `"wood"`:

```
$table = ["num_legs" => 4, "material" => "wood"];
```

PHP implode function

```php
$pieces = [1, "2", "three", 4.0];
$glue = " and a ";
echo implode($glue, $pieces); // 1 and a 2 and a three and a 4

//Note - the arrays do not have to always be labelled
"$pieces" & "$glue"
```

The built-in `implode()` function makes a string from an array. The function takes two arguments: a string to place between each element of an array and the array to be joined together.

The function returns a single string with the string argument, known as the `$glue`, inserted between each element in the array, known as the `$pieces`.