PHP - Compound Types

Data types in PHP can be of "scalar type" or "compound type". Integer, float, Boolean and string types are scalar types, whereas array and object types are classified as compound types. Values of more than one types can be stored together in a single variable of a compound type.

In PHP, objects and arrays are the two compound data types.

- An array is an ordered collection of elements of other data types, not necessarily of the same type.
- An object is an instance of either a built-in or a user defined class, consisting of properties and methods.

Arrays in PHP

An array is a data structure that stores one or more data values in a single variable. An array in PHP is an ordered map that associates the values to their keys.

- There are two ways to declare an array in PHP. One is to use the built-in array() function, and the other is to put the array elements inside square brackets.
- An array which is a collection of only values is called an **indexed array**. Each value is identified by a positional index staring from 0.
- If the array is a collection of key-value pairs, it is called as an **associative array**. The key component of the pair can be a number or a string, whereas the value part can be of any type.

The array() Function in PHP

The built-in array() function uses the parameters given to it and returns an object of array type. One or more comma-separated parameters are the elements in the array.

```
array(mixed ...$values): array
```

Each value in the parenthesis may be either a singular value (it may be a number, string, any object or even another array), or a key-value pair. The association between the and its value is denoted by the "=>" symbol.

Example

Take a look at this following example −

```
$arr1 = array(10, "asd", 1.55, true);

$arr2 = array("one"=>1, "two"=>2, "three"=>3);

$arr3 = array(
   array(10, 20, 30),
   array("Ten", "Twenty", "Thirty"),
   array("physics"=>70, "chemistry"=>80, "maths"=>90)
);
```

Using Square Brackets [ ]

Instead of the array() function, the comma-separated array elements may also be put inside the square brackets to declare an array object. In this case too, the elements may be singular values or a string or another array.

```
$arr1 = [10, "asd", 1.55, true];

$arr2 = ["one"=>1, "two"=>2, "three"=>3];

$arr3 = [
   [10, 20, 30],
   ["Ten", "Twenty", "Thirty"],
   ["physics"=>70, "chemistry"=>80, "maths"=>90]
];
```

Accessing Array Elements

To access any element from a given array, you can use the array[key] syntax. For an indexed array, put the index inside the square bracket, as the index itself is anyway the key.

</> Open Compiler

```
<?php
   $arr1 = [10, 20, 30];
   $arr2 = array("one"=>1, "two"=>2, "three"=>3);

   var_dump($arr1[1]);
   var_dump($arr2["two"]);
?>
```

It will produce the following **output** −

```
int(20)
int(2)
```

Array Traversal in PHP

You can also use the foreach loop to iterate through an indexed array.

Open Compiler

```
<?php
   $arr1 = [10, 20, 30, 40, 50];
   foreach ($arr1 as $val){
      echo "$val\n";
   }
?>
```

It will produce the following **output** −

```
10
20
30
40
50
```

Note that PHP internally treats the indexed array as an associative array, with the index being treated as the key. This fact can be verified by the var_dump() output of the array.

We can unpack each element of the indexed array in the key and value variables with the **foreach** syntax −

</>   Open Compiler

```
<?php
   $arr1 = [10, 20, 30, 40, 50];
   foreach ($arr1 as $key => $val){
      echo "arr1[$key] = $val" . "\n";
   }
?>
```

It will produce the following **output** −

```
arr1[0] = 10
arr1[1] = 20
arr1[2] = 30
arr1[3] = 40
arr1[4] = 50
```

The foreach loop is also used for iterating through an associative array, although any other type of loop can also be used with some maneuver.

Let us look at the **foreach** loop implementation, with each k-v pair unpacked in two variables.

</>   Open Compiler

```
<?php
   $capitals = array(
      "Maharashtra"=>"Mumbai",
      "Telangana"=>"Hyderabad",
      "UP"=>"Lucknow",
      "Tamilnadu"=>"Chennai"
   );

   foreach ($capitals as $k=>$v) {
      echo "Capital of $k is $v" . "\n";
   }
?>
```

It will produce the following **output** −

```
Capital of Maharashtra is Mumbai
Capital of Telangana is Hyderabad
Capital of UP is Lucknow
Capital of Tamilnadu is Chennai
```

Objects in PHP

In PHP, an object is a compound data type. It is an instance of either a built in or user defined class. Given below is a simple PHP class –

```
class SayHello {
   function hello() {
      echo "Hello World";
   }
}
```

To declare an object of a class, we need to use the **new** operator.

```
$obj=new SayHello;
```

We can now call its method –

Open Compiler

```
<?php
   class SayHello {
      function hello() {
         echo "Hello World". PHP_EOL;
      }
   }

   $obj=new SayHello;
   var_dump(gettype($obj));
   $obj->hello();
?>
```

It will produce the following **output** –

```
string(6) "object"
Hello World
```

stdClass

PHP provides stdClass as a generic empty class which is useful for adding properties dynamically and casting. An object of stdClass is null to begin with. We can add properties to it dynamically.

Open Compiler

```php
<?php
   $obj=new stdClass;
   $obj->name="Deepak";
   $obj->age=21;
   $obj->marks=75;

   print_r($obj);
?>
```

It will produce the following **output** −

```
stdClass Object (
   [name] => Deepak
   [age] => 21
   [marks] => 75
)
```

Array to Object Conversion in PHP

An array in PHP can be typecast to an object as follows −

Open Compiler

```php
<?php
   $arr=array("name"=>"Deepak", "age"=>21, "marks"=>75);
   $obj=(object)$arr;

   print_r($obj);
?>
```

It will produce the following **output** −

```
stdClass Object (
   [name] => Deepak
   [age] => 21
   [marks] => 75
)
```

Object to Array Conversion in PHP

Conversely, an object can be cast to an array. Take a look at the following example −

</> Open Compiler

```php
<?php
   $obj=new stdClass;
   $obj->name="Deepak";
   $obj->age=21;
   $obj->marks=75;

   $arr=(array)$obj;
   print_r($arr);
?>
```

It will produce the following **output** −

```
Array
(
   [name] => Deepak
   [age] => 21
   [marks] => 75
)
```

Scalar Type to Object Type Conversion in PHP

A variable of any scalar type can also be converted to an object by type casting. The value of the scalar variable becomes the value of the object's scalar property.

</> Open Compiler

```php
<?php
   $name="Deepak";
   $age=21;
   $percent=75.50;

   $obj1=(object)$name;
   print_r($obj1);

   $obj2=(object)$age;
   print_r($obj2);

   $obj3=(object)$percent;
   print_r($obj3);
?>
```

It will produce the following **output** −

```
stdClass Object
(
   [scalar] => Deepak
)
stdClass Object
(
   [scalar] => 21
)
stdClass Object
(
   [scalar] => 75.5
)
```