Chapter

02

배열

배열은 여러 개의 값을 가지는 유형의 데이터 타입입니다. 여러 개의 값을 가지고 있는 배열의 특성상 값의 정렬, 분리, 결합 등의 추가적인 작업이 가능합니다.

PHP는 배열 데이터에 대해서 추가 작업을 할 수 있도록 내장 함수를 제공합니다.

02.1 배열 생성

배열 변수를 생성할 수 있습니다. 또한 배열을 여러 개의 변수로 분할하여 처리할 수도 있습니다. 배열을 변수화하여 데이터를 처리하는 것은 배열을 활용하는 데 매우 유용합니다.

02.1.1 변수 할당

배열 변수를 생성하기 위해서는 데이터의 값을 배열 타입으로 저장하면 됩니다. 배열 타입의 값을 초기에 생성하기 위해서는 array() 함수를 이용합니다.

| 내장 함수 | 배열 생성

```
array array ([ mixed $... ] )
```

내장 함수 array()는 입력된 값에 대한 배열 데이터를 반환합니다. 각각의 데이터는 콤마(,)로 구분하며 키를 삽입할 때는 => 기호를 사용합니다.

예제 파일 | **array-01.php**

```php
<?php
    // 배열 변수를 생성합니다.
    // 입력한 값에 대한 배열값을 반환합니다.
    $arr1 = array("hojin","jiny","james","eric");

    // 배열값을 foreach를 통하여 하나씩 출력합니다.
    foreach ($arr1 as $value) {
        echo "$value <br>";
    }

    echo "===== <br>";

    // 키 형태의 배열을 지정합니다.
    $arr2 = array('firstname' => "hojin", 'lastname' => "lee" );
    foreach ($arr2 as $key => $value) {
        echo "{$key} => {$value} ";
        print_r($arr);
    }

?>
```

화면 출력

```
hojin
jiny
james
eric
=====
firstname => hojin lastname => lee
```

배열은 값의 묶음 처리입니다. 묶음 형태의 배열을 처리하는 것은 유사한 데이터를 처리하는 데 있어서 많은 변수의 이름을 통하여 자원을 낭비하는 것을 줄이기 위해서입니다.

| 내장 함수 | 배열의 변수화

```
array list ( mixed $var1 [, mixed $... ] )
```

하지만 반대로 배열을 여러 개의 변수로 분할할 수도 있습니다. 내장 함수 list()는 배열의 값을 지정한 각각의 변수에 값을 할당할 수 있습니다. 배열의 데이터를 여러 개의 변수에 분배하여 저장할 때 편리합니다.

예제 파일 | **array-02.php**

```
<?php
    // 배열을 생성합니다.
    $arr = array('orage', 'apple', 'grape');

    // 배열의 값을 각각의 변수에 값을 매칭하여 대입합니다.
    list($orage, $apple, $grape) = $arr;

    echo "변수 orage = $orage <br>";
    echo "변수 apple = $apple <br>";
    echo "변수 grape = $grape <br>";

    echo "<br> ===== <br><br>";

    while (list($key,$val) = each($arr)) {
        echo $key. "==>" .$val. "<br>";
    }

?>
```

화면 출력

```
변수 orage = orage
변수 apple = apple
변수 grape = grape
=====
```

```
0==>orage
1==>apple
2==>grape
```

| 내장 함수 |

array **array_fill** (int $start_index , int $num , mixed $value)

내장 함수 array_fill()은 지정한 값으로 배열을 채웁니다. 지정한 값을 배열의 시작 위치(start_index)부터 반복 횟수(num)만큼 반복하여 배열을 생성합니다.

예제 파일 | **array-03.php**

```php
<?php
   $a = array_fill(5, 6, 'aaa');
   $b = array_fill(-2, 4, 'bbb');
   print_r($a);
   print_r($b);

?>
```

화면 출력

```
Array ( [5] => aaa [6] => aaa [7] => aaa [8] => aaa [9] => aaa [10] => aaa )
Array ( [-2] => bbb [0] => bbb [1] => bbb [2] => bbb )
```

| 내장 함수 |

array **array_fill_keys** (array $keys , mixed $value)

내장 함수 array_fill_keys()는 입력받은 배열의 값을 키 값으로 사용하여 배열의 데이터를 채웁니다.

예제 파일 | **array-04.php**

```php
<?php
   $keys = array('foo', 5, 10, 'bar');
```

```
    $a = array_fill_keys($keys, 'banana');
    print_r($a);

?>
```

화면 출력

```
Array ( [foo] => banana [5] => banana [10] => banana [bar] => banana )
```

02.1.2 값과 키

PHP는 다양한 형태의 최신 배열 표기 및 처리를 지원합니다. 일반 배열과 키 값을 가지고 있는 연상 배열 모두 사용할 수 있습니다. 내장 함수 each() 함수는 배열의 키와 값을 한 쌍으로 반환합니다. list() 함수와 결합하여 배열의 키와 값을 추출할 수 있습니다.

| 내장 함수 | 배열의 쌍 값

```
array each ( array &$array )
```

배열이 담긴 each() 함수를 호출할 때마다 다음 배열 위치로 전달됩니다.

예제 파일 | array-05.php

```
<?php
    // 배열을 생성합니다.
    $arr = array('orage', 'apple', 'grape');

    $test = each($arr);
    echo "원소 0= ".$test[0]."<br>";
    echo "원소 1= ".$test[1]."<br>";
    echo "원소 key= ".$test[key]."<br>";
    echo "원소 value= ".$test[value]."<br>";

    echo "<br><br>";

    while (list($key,$val) = each($arr)) {
        echo $key. "==>" .$val. "<br>";
```

```
15      }
16
17  ?>
```

화면 출력

```
원소 0= 0
원소 1= orage
원소 key= 0
원소 value= orage

1==>apple
2==>grape
```

위의 실습에서 배열의 키와 값을 추출해 봅니다. 첫 번째 명령에서 1번, 반복문에서 2번 실행되어 총 3개의 배열 정보가 출력됩니다. each() 함수는 네 가지의 키 값으로 데이터를 반환하는데, 0, key, 1, value입니다.

연상 배열에서 값만 추출한 다른 배열을 생성할 수있습니다. 내장 함수 array_values()는 배열의 값을 반환합니다.

| 내장 함수 | 배열값

```
array array_values ( array $array )
```

예제 파일 | **array-06.php**

```
1   <?php
2       // 배열을 생성합니다.
3       $arr = array('firstname' => "hojin", 'lastname' => "lee" );
4       while (list($key,$val) = each($arr)) {
5           echo $key. "==>" .$val. ", ";
6       }
7       echo "<br> ==== <br>";
8
9       $arr2 = array_values($arr);
10      while (list($key,$val) = each($arr2)) {
11          echo $key. "==>" .$val. ", ";
```

```
12      }
13
14  ?>
```

화면 출력

```
firstname==>hojin, lastname==>lee,
====
0==>hojin, 1==>lee,
```

| 내장 함수 | 배열 인덱스 키

array **array_keys** (array $array [, mixed $search_value = null [, bool $strict = false]])

내장 함수 array_keys()는 배열의 인덱스 키 이름만을 반환받을 수 있습니다.

예제 파일 | **array-07.php**

```
1   <?php
2      $arr = array('firstname' => "hojin", 'lastname' => "lee" );
3      $index = array_keys($arr);
4
5      while (list($key,$val) = each($index)) {
6         echo $key. "==>" .$val. "<br>";
7      }
8
9   ?>
```

화면 출력

```
0==>firstname
1==>lastname
```

| 내장 함수 |

array **array_column** (array $input , mixed $column_key [, mixed $index_key = null])

내장 함수 array_column()은 입력된 다중 배열에서 단일 열에서 값만을 추출하여 배열로 반환합니다.

예제 파일 | **array-08.php**

```php
<?php
   $records = array(
      array(
         'id' => 2135,
         'first_name' => '홍',
         'last_name' => '길동',
      ),
      array(
         'id' => 3245,
         'first_name' => '이',
         'last_name' => '호진',
      ),
      array(
         'id' => 5342,
         'first_name' => '장',
         'last_name' => '승빈',
      )
   );

   $first_names = array_column($records, 'last_name');
   print_r($first_names);

?>
```

화면 출력

```
Array ( [0] => 길동 [1] => 호진 [2] => 승빈 )
```

| 내장 함수 |

array **array_flip** (array $array)

내장 함수 array_flip()은 배열의 키와 값을 서로 교환합니다. array_flip()은 배열의 키가 값이 됩니다. 값은 다시 배열의 키로 변경하여 배열로 처리합니다.

예제 파일 | **array-09.php**

```php
<?php
    $input = array("a"=>"oranges", "b"=>"apples", "c"=>"pears");
    $flipped = array_flip($input);

    print_r($flipped);

?>
```

화면 출력

```
Array ( [oranges] => a [apples] => b [pears] => c )
```

02.1.3 변수 결합과 분리

내장 함수 compact()는 외부의 변수명을 응용하여 배열을 생성할 수 있습니다. 변수명은 키 값으로 사용됩니다.

| 내장 함수 | 변수 배열 생성

```
array compact ( mixed $varname1 [, mixed $... ] )
```

예제 파일 | **array-10.php**

```php
<?php

    $one = "1개";
    $two = "2개";
    $three = "3개";
    $four = "4개";

    // 배열에 변수의 값을 넣기 위해서는 ""로 변수명을 입력하면 됩니다.
    $num = array("three", "four");

    // 배열을 넣을 때는 배열명만 인수로 전달합니다.
    $arr = compact("one", "two", $num);

```

```
14      while (list($key,$val) = each($arr)) {
15          echo $key. "==>" .$val. "<br>";
16      }
17
18  ?>
```

화면 출력

```
one==>1개
two==>2개
three==>3개
four==>4개
```

| 내장 함수 | 배열 변수화

int **extract** (array &$array [, int $flags = EXTR_OVERWRITE [, string $prefix = NULL]])

내장 함수 extract()는 배열의 키 이름으로 각각 변수화합니다. compact() 함수의 반대 기능입니다.

예제 파일 | **array-11.php**

```
1   <?php
2       $arr = array("blue"=>"파랑색","red"=>"빨강색","black"=>"검정색");
3
4       // 배열을 변수로 생성합니다.
5       extract($arr);
6
7       echo "blue = ".$blue ."<br>";
8       echo "red = ".$red ."<br>";
9       echo "black = ".$black ."<br>";
10
11  ?>
```

화면 출력

```
blue = 파랑색
red = 빨강색
black = 검정색
```

02.1.4 결합 및 분리

내장 함수 array_merge()는 입력되는 배열을 하나의 배열 형태로 결합하여 반환합니다.

| 내장 함수 | 배열 결합

```
array array_merge ( array $array1 [, array $... ] )
```

예제 파일 | **array-12.php**

```
<?php
   $arr1 = array('지니', '호진');
   $arr2 = array('철수', '영수');
   $arr3 = array('제니퍼', '홍길동');

   $arr = array_merge($arr1, $arr2, $arr3);

   while (list($key,$val) = each($arr)) {
      echo $key. "==>" .$val. "<br>";
   }

?>
```

화면 출력

```
0==>지니
1==>호진
2==>철수
3==>영수
4==>제니퍼
5==>홍길동
```

| 내장 함수 | 재귀적 배열 결합

```
array array_merge_recursive ( array $array1 [, array $... ] )
```

2개 이상의 배열을 재귀적으로 병합합니다. 내장 함수 array_merge_recursive()는 다수

의 배열 요소를 병합하여 배열의 끝에 추가합니다.

병합될 배열에 동일한 문자열 키가 있는 경우 이러한 키의 값은 배열로 병합됩니다. 이는 재귀적으로 수행되므로 값 중 하나가 배열 자체이면 함수는 해당 항목과 병합합니다. 그리고 배열에 동일한 숫자 키가 있으면 나중에 값이 원래 값을 덮어쓰지 않고 뒤에 추가됩니다.

예제 파일 | **array-13.php**

```
<?php
    $ar1 = array("color" => array("favorite" => "red"), 5);
    $ar2 = array(10, "color" => array("favorite" => "green", "blue"));
    $result = array_merge_recursive($ar1, $ar2);

    print_r($result);

?>
```

화면 출력

```
Array ( [color] => Array ( [favorite] => Array ( [0] => red [1] => green ) [0]
=> blue ) [0] => 5 [1] => 10 )
```

| 내장 함수 | 배열 결합 (키/값)

```
array array_combine ( array $keys , array $values )
```

내장 함수 array_combine()은 첫 번째 배열을 키 값으로, 두 번째 배열을 값 형태로 배열을 결합합니다.

예제 파일 | **array-14.php**

```
<?php
    $a = array('green', 'red', 'yellow');
    $b = array('avocado', 'apple', 'banana');

    // $a 배열은 키명으로 사용하고, $b는 데이터로 사용합니다.
    // 2개의 배열을 연상 배열 형태로 결합합니다.
```

```
7       $arr = array_combine($a, $b);
8       print_r($arr);
9
10  ?>
```

화면 출력

```
Array ( [green] => avocado [red] => apple [yellow] => banana )
```

| 내장 함수 | 일부 배열 분리

array **array_slice** (array $array , int $offset [, int $length = NULL [, bool $preserve_keys = false]])

내장 함수 array_slice()는 배열의 일부분의 요소만을 추출하여 반환합니다.

예제 파일 | **array-15.php**

```
1   <?php
2       // 배열을 생성합니다.
3       $arr = array('하나', '둘', '셋', '넷', '다섯', '여섯', '일곱', '여덟', '아홉',
        '열');
4
5       // 첫 번째 인자: 추출 시작 값
6       // 처음 2부터 끝까지.
7       $slice = array_slice($arr, 2);
8       while (list($key,$val) = each($slice)) {
9           echo $key. "==>" .$val. ", ";
10      }
11
12      echo "<br>";
13
14      // 두 번째 인자: 추출 데이터 개수
15      // 처음 1부터, 3개의 데이터를 추출
16      $slice = array_slice($arr, 1,3);
17      while (list($key,$val) = each($slice)) {
18          echo $key. "==>" .$val. ",";
19      }
20
```

```
21      echo "<br>";
22
23      // 처음 인자가 음수이면 끝에서 -인자부터 시작하여, 두 번째 인자 개수를 추출
24      // 마지막 -5 위치부터(여섯), 3개의 데이터를 추출
25      $slice = array_slice($arr, -5,3);
26      while (list($key,$val) = each($slice)) {
27          echo $key. "==>" .$val. ",";
28      }
29
30  ?>
```

화면 출력

```
0==>셋, 1==>넷, 2==>다섯, 3==>여섯, 4==>일곱, 5==>여덟, 6==>아홉, 7==>열,
0==>둘, 1==>셋, 2==>넷,
0==>여섯, 1==>일곱, 2==>여덟,
```

| 내장 함수 |

array **array_chunk** (array $array , int $size [, bool $preserve_keys = false])

내장 함수 array_chunk()는 배열을 크기 요소가 있는 배열로 분할합니다. 마지막 chunk는 크기 요소보다 작은 요소를 포함할 수 있습니다.

예제 파일 | **array-16.php**

```
1   <?php
2      $array = array('a', 'b', 'c', 'd', 'e');
3      print_r(array_chunk($array, 2));
4      echo "<br>";
5
6      print_r(array_chunk($array, 3));
7      echo "<br>";
8
9   ?>
```

화면 출력

```
Array ( [0] => Array ( [0] => a [1] => b ) [1] => Array ( [0] => c [1] => d ) [2]
=> Array ( [0] => e ) )
```

```
Array ( [0] => Array ( [0] => a [1] => b [2] => c ) [1] => Array ( [0] => d [1]
=> e ) )
```

02.1.5 출력

내장 함수 array_reverse()는 배열의 내용을 역순으로 출력합니다.

| 내장 함수 | 역순 배열

array **array_reverse** (array $array [, bool $preserve_keys = false])

예제 파일 | **array-17.php**

```php
<?php
   $arr = array("hojin","jiny","james","eric");
   $reverse = array_reverse($arr);

   while (list($key,$val) = each($reverse)) {
      echo $key. "==>" .$val. "<br>";
   }

?>
```

화면 출력

```
0==>eric
1==>james
2==>jiny
3==>hojin
```

02.1.6 추가 및 삭제

내장 함수 array_push()는 배열의 마지막 위치에 새로운 원소 데이터를 추가합니다.

| 내장 함수 | 배열 원소 추가(마지막)

int **array_push** (array &$array , mixed $value1 [, mixed $...])

예제 파일 | **array-18.php**

```
1   <?php
2      // 배열을 생성합니다.
3      $arr = array('orage', 'apple', 'grape');
4
5      // $arr 배열에 2개의 원소를 추가합니다.
6      array_push($arr, 'banana', 'tomato');
7
8      while (list($key,$val) = each($arr)) {
9         echo $key. "==>" .$val. "<br>";
10     }
11
12  ?>
```

화면 출력

```
0==>orage
1==>apple
2==>grape
3==>banana
4==>tomato
```

| 내장 함수 | 배열 원소 제거

mixed **array_pop** (array &$array)

내장 함수 array_pop()은 배열의 마지막 요소 하나를 제거합니다. 이는 데이터 처리 알고리즘 pop과 유사합니다.

예제 파일 | **array-19.php**

```
1   <?php
```

```php
    // 배열을 생성합니다.
    $arr = array('orage', 'apple', 'grape');

    // 마지막 배열 원소를 삭제합니다.
    $temp = array_pop($arr);
    echo "배열에서 마지막 $temp 요소 하나를 제거합니다.<br> ";

    while (list($key,$val) = each($arr)) {
        echo $key. "==>" .$val. "<br>";
    }

?>
```

화면 출력

```
배열에서 마지막 grape 요소 하나를 제거합니다.
0==>orage
1==>apple
```

| 내장 함수 | 배열 원소 추가(처음)

```
int array_unshift ( array &$array , mixed $value1 [, mixed $... ] )
```

내장 함수 array_unshift()는 배열 앞쪽에 새로운 원소를 추가합니다.

예제 파일 | array-20.php

```php
<?php
    // 배열을 생성합니다.
    $arr = array('orage', 'apple', 'grape');

    // $arr 배열에 2개의 원소를 추가합니다.
    array_unshift($arr, 'banana', 'tomato');

    while (list($key,$val) = each($arr)) {
        echo $key. "==>" .$val. "<br>";
    }

?>
```

화면 출력

```
0==>banana
1==>tomato
2==>orage
3==>apple
4==>grape
```

| 내장 함수 | 배열 원소 제거(처음)

mixed **array_shift** (array &$array)

내장 함수 array_shift()는 배열의 맨 처음 요소 하나를 제거합니다. 배열을 좌측으로 하나 밀어서 제거하는 형태입니다.

예제 파일 | **array-21.php**

```
1  <?php
2     // 배열을 생성합니다.
3     $arr = array('orage', 'apple', 'grape');
4
5     // 배열 원소를 삭제합니다.
6     $temp = array_shift($arr);
7     echo "배열에서 처음 $temp 요소 하나를 제거합니다.<br> ";
8
9     while (list($key,$val) = each($arr)) {
10       echo $key. "==>" .$val. "<br>";
11    }
12
13 ?>
```

화면 출력

```
배열에서 처음 orage 요소 하나를 제거합니다.
0==>apple
1==>grape
```

array **array_splice** (array &$input , int $offset [, int $length = count($input) [, mixed $replacement = array()]])

내장 함수 array_splice()는 일부분의 데이터를 삭제한 후에 값을 반환합니다.

예제 파일 | **array-22.php**

```php
<?php
    // 배열을 생성합니다.
    $arr = array('하나', '둘', '셋', '넷', '다섯', '여섯', '일곱', '여덟', '아홉',
    '열');
    echo "원본<br>";
    while (list($key,$val) = each($arr)) {
        echo $key. "==>" .$val. ", ";
    }
    echo "<br><br>";

    // 첫 번째 인자: 삭제 시작 값
    // 2까지의 원소를 삭제하고 반환
    $slice = array_splice($arr, 7);
    while (list($key,$val) = each($slice)) {
        echo $key. "==>" .$val. ", ";
    }
    echo "를 삭제하였습니다.<br>";
    echo "<br>";

    $slice = array_splice($arr, 3,3);
    while (list($key,$val) = each($slice)) {
        echo $key. "==>" .$val. ", ";
    }
    echo "를 삭제하였습니다.<br>";
    echo "<br>";

?>
```

화면 출력

원본

```
0==>하나, 1==>둘, 2==>셋, 3==>넷, 4==>다섯, 5==>여섯, 6==>일곱, 7==>여덟, 8==>아홉,
9==>열,

0==>여덟, 1==>아홉, 2==>열, 를 삭제하였습니다.

0==>넷, 1==>다섯, 2==>여섯, 를 삭제하였습니다.
```

02.2 배열 검사

배열은 여러 개의 데이터를 포함하고 있습니다. 같은 유형의 데이터의 묶음을 처리하는 데 있어서 값을 검사하고 처리하는 데 매우 유용합니다.

PHP는 배열의 데이터를 검색하고 처리하는 몇 가지 내장 함수를 지원합니다.

02.2.1 검사

내장 함수 in_array()는 배열에 값이 존재하는지 검사할 수 있습니다. 값이 있는 경우에는 true 값을 반환합니다.

| 내장 함수 | 배열 검색

```
bool in_array ( mixed $needle , array $haystack [, bool $strict = FALSE ] )
```

예제 파일 | array-23.php

```php
<?php
   $arr = array("blue"=>"파랑색","red"=>"빨강색","black"=>"검정색");

   if (in_array("파랑색",$arr)) {
      echo "값이 존재합니다.";
   } else {
      echo "배열값이 없습니다.";
   }

?>
```

화면 출력

값이 존재합니다.

| 내장 함수 | 배열 인덱스 키 확인

```
bool array_key_exists ( mixed $key , array $array )
```

내장 함수 array_key_exists()는 연상 배열 안에 인덱스 키 값이 존재하는지를 확인합니다.

예제 파일 | array-24.php

```
<?php
   $array = array('first' => null, 'second' => 4);

   // returns true
   if (array_key_exists('first', $array)) {
      echo "배열에 키 값이 존재합니다.";
   } else {
      echo "키 값이 없습니다.";
   }

?>
```

화면 출력

배열에 키 값이 존재합니다.

| 내장 함수 |

```
mixed array_search ( mixed $needle , array $haystack [, bool $strict = false ] )
```

내장 함수 array_search()는 지정한 배열에서 값을 검색하여 일치하는 키 값을 반환합니다.

예제 파일 | **array-25.php**

```php
<?php
   $array = array(0 => 'blue', 1 => 'red', 2 => 'green', 3 => 'red');

   $key = array_search('green', $array); // $key = 2;
   print_r($key);
   echo "<br>";

   $key = array_search('red', $array);   // $key = 1;
   print_r($key);

?>
```

화면 출력

```
2
1
```

02.2.2 중복

내장 함수 array_count_values()는 배열 안에 들어 있는 동일한 값의 개수를 반환합니다.

| 내장 함수 | **중복값 체크**

```
array array_count_values ( array $array )
```

예제 파일 | **array-26.php**

```php
<?php
   $arr = array('orage', 'apple', 'grape', 'orage', 'apple', 'grape',
   'orage');

   $arr_count = array_count_values($arr);
   while (list($key,$val) = each($arr_count)) {
      echo $key. "==>" .$val. "<br>";
   }
```

```
8
9   ?>
```

화면 출력

```
orage==>3
apple==>2
grape==>2
```

| 내장 함수 | 중복값 제거

```
array array_unique ( array $array [, int $sort_flags = SORT_STRING ] )
```

내장 함수 array_unique()는 입력된 배열값 중에서 중복된 내용은 제거합니다.

예제 파일 | array-27.php

```
1   <?php
2      $arr = array("a" => "green", "red", "b" => "green", "blue", "red");
3      $result = array_unique($arr);
4      print_r($result);
5
6   ?>
```

화면 출력

```
Array ( [a] => green [0] => red [1] => blue )
```

02.3 배열 위치

배열은 여러 개의 데이터의 값을 가지고 있습니다. 인덱스 또는 키를 통하여 데이터에 접근할 수 있습니다.

그 외에 배열의 위치 값을 이용하여 데이터에 접근 및 값을 읽어올 수 있습니다. PHP는 배열의 위치 값을 제어할 수 있는 몇 가지 내장 함수를 지원하고 있습니다.

02.3.1 위치 이동

| 내장 함수 | 배열 포인트 초기화

```
mixed reset ( array &$array )
```

내장 함수 reset()은 배열의 포인트를 초기화합니다. 배열의 처음 위치를 가리키게 됩니다.

| 내장 함수 | 배열 포인트 다음

```
mixed next ( array &$array )
```

내장 함수 next()는 배열의 다음 포인트로 이동합니다. 함수를 한 번 호출할 때마다 배열의 포인트를 다음으로 이동합니다.

| 내장 함수 | 배열 포인트 마지막

```
mixed end ( array &$array )
```

내장 함수 end()는 배열의 마지막 포인트로 이동합니다.

| 내장 함수 | 배열 포인트 이전

```
mixed prev ( array &$array )
```

내장 함수 prev()는 배열의 이전 포인트로 이동합니다. 함수를 한 번 호출할 때마다 배열의 포인트를 이전으로 이동합니다.

예제 파일 | **array-28.php**

```
1   <?php
```

```php
    // 배열을 생성합니다.
    $arr = array('하나', '둘', '셋', '넷', '다섯', '여섯', '일곱', '여덟', '아홉',
    '열');
    echo "원본<br>";
    while (list($key,$val) = each($arr)) {
       echo $key. "==>" .$val. ", ";
    }
    echo "<br> ======================== <br>";

    // 배열의 포인터를 처음으로 이동합니다.
    reset($arr);

    // 디음 배열의 포인터를 이동합니다.
    next($arr);
    echo "현재 포인트 = " . current($arr) . "<br>";

    // 디음 배열의 포인터를 이동합니다.
    next($arr);
    echo "현재 포인트 = " . pos($arr) . "<br>";

    // 마지막 배열의 포인터를 이동합니다.
    end($arr);
    echo "현재 포인트 = " . current($arr) . "<br>";

    // 이전 배열의 포인터를 이동합니다.
    prev($arr);
    echo "현재 포인트 = " . pos($arr) . "<br>";

?>
```

화면 출력

원본
0==>하나, 1==>둘, 2==>셋, 3==>넷, 4==>다섯, 5==>여섯, 6==>일곱, 7==>여덟, 8==>아홉,
9==>열,
========================
현재 포인트 = 둘
현재 포인트 = 셋
현재 포인트 = 열
현재 포인트 = 아홉

위의 예제는 배열의 위치를 지정하는 함수들을 이용하여 배열의 데이터를 출력하는 예제입니다.

02.3.2 위치 확인

배열의 위치 함수를 이용하면 배열의 데이터를 가리키는 포인터를 변경하게 됩니다. 현재 가리키고 있는 값과 키를 알 수 있는 함수들도 함께 제공합니다.

| 내장 함수 | 배열 현재 위치

```
mixed current ( array &$array )
```

내장 함수 current()는 배열의 현재 위치의 값을 반환합니다. pos() 함수는 current() 함수의 또 다른 이름의 별칭입니다.

| 내장 함수 | 배열 포인트 키 값

```
mixed key ( array &$array )
```

내장 함수 key()는 현재의 배열 포인트의 키 값을 출력합니다.

예제 파일 | **array-29.php**

```
1  <?php
2     $arr = array( 'fruit1' => 'apple', 'fruit2' => 'orange', 'fruit3' =>
      'grape', 'fruit4' => 'apple', 'fruit5' => 'apple');
3
4     while ($name = current($arr)) {
5        if ($name == 'apple') {
6           echo key($arr).'<br>';
7        }
8
9        next($arr);
10    }
11 ?>
```

화면 출력

```
fruit1
fruit4
fruit5
```

02.4 배열 정렬

배열을 이용하면 배열의 값을 알파벳 또는 숫자를 기준으로 오름차순, 내림차순으로 정렬할 수 있습니다.

02.4.1 오름차순

| 내장 함수 | 오름차순 정렬

```
bool sort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

내장 함수 sort()를 이용하면 배열의 값을 오름차순으로 정렬합니다.

| 내장 함수 | 이중 오름차순 정렬

```
bool asort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

내장 함수 asort()는 연상 배열의 값일 경우 네임 키를 우선적으로 오름차순으로 정렬된 상태에서 서브 데이터 값을 오름차순으로 정렬합니다.

| 내장 함수 | 이름 키 오름차순 정렬

```
bool ksort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

내장 함수 ksort()는 연상 배열일 경우 이름 키에 대해서 오름차순으로 정렬합니다.

02.4.2 내림차순

| 내장 함수 | 이름 키 오름차순 정렬

```
bool rsort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

내장 함수 rsort()는 오름차순의 반대인 내림차순으로 배열의 값을 정렬합니다.

| 내장 함수 | 이름 키 내림차순 정렬

```
bool krsort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

내장 함수 krsort()는 연상 배열일 경우 이름 키에 대해서 내림차순으로 정렬합니다.

| 내장 함수 | 내림차순 배열

```
bool arsort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

내장 함수 arsort()는 인덱스 값을 유지한 상태에서 배열 요소의 데이터 값만 내림차순으로 재정렬합니다.

02.4.3 정렬 예제

내림차순 및 오름차순 정렬을 예제를 통하여 확인하도록 하겠습니다.

예제 파일 | array-30.php

```
1  <?php
2
3      $arr = array('one'=>'하나', 'two'=>'둘', 'three'=>'셋', 'four'=>'넷',
       'five'=>'다섯', 'six'=>'여섯', 'seven'=>'일곱');
4
5      echo "오름차순 정렬<br>";
```

```
6       sort($arr);
7       while (list($key,$val) = each($arr)) {
8          echo $key. "==>" .$val. ", ";
9       }
10
11      echo "<br>";
12      echo "내림차순 정렬<br>";
13      rsort($arr);
14      while (list($key,$val) = each($arr)) {
15         echo $key. "==>" .$val. ", ";
16      }
17
18      echo "<br>";
19      echo "이중 오름차순 정렬<br>";
20      asort($arr);
21      while (list($key,$val) = each($arr)) {
22         echo $key. "==>" .$val. ", ";
23      }
24
25      echo "<br>";
26      echo "이름 키 오름차순 정렬<br>";
27      ksort($arr);
28      while (list($key,$val) = each($arr)) {
29         echo $key. "==>" .$val. ", ";
30      }
31
32      echo "<br>";
33      echo "이름 키 내림차순 정렬<br>";
34      krsort($arr);
35      while (list($key,$val) = each($arr)) {
36         echo $key. "==>" .$val. ", ";
37      }
38
39  ?>
```

화면 출력

오름차순 정렬
0==>넷, 1==>다섯, 2==>둘, 3==>셋, 4==>여섯, 5==>일곱, 6==>하나,
내림차순 정렬
0==>하나, 1==>일곱, 2==>여섯, 3==>셋, 4==>둘, 5==>다섯, 6==>넷,

```
이중 오름차순 정렬
6==>넷, 5==>다섯, 4==>둘, 3==>셋, 2==>여섯, 1==>일곱, 0==>하나,
이름 키 오름차순 정렬
0==>하나, 1==>일곱, 2==>여섯, 3==>셋, 4==>둘, 5==>다섯, 6==>넷,
이름 키 내림차순 정렬
6==>넷, 5==>다섯, 4==>둘, 3==>셋, 2==>여섯, 1==>일곱, 0==>하나,
```

02.4.4 자연 순서 정렬

내장 함수 natsort()는 '자연 순서' 알고리즘을 사용해 배열을 정렬합니다

| 내장 함수 |

bool **natsort** (array &$array)

키와 값을 유지하면서 영숫자 문자열을 사람이 직접 분류하는 것과 같은 정렬 알고리즘을 구현합니다.

예제 파일 | **array-31.php**

```php
<?php
    $array1 = $array2 = array("img12.png", "img10.png", "img2.png",
    "img1.png");

    echo "기본 정렬<br>";
    asort($array1);
    print_r($array1);

    echo "<br>";

    echo "자연 순서<br>";
    natsort($array2);
    print_r($array2);


?>
```

화면 출력

```
기본 정렬
Array ( [3] => img1.png [1] => img10.png [0] => img12.png [2] => img2.png )
자연 순서
Array ( [3] => img1.png [2] => img2.png [1] => img10.png [0] => img12.png )
```

| 내장 함수 |

bool **natcasesort** (array &$array)

내장 함수 natcasesort()는 natsort()의 대소문자를 구별하지 않는 버전입니다.

netcasesort()는 대문자와 소문자를 구별하지 않는 '자연 순서' 알고리즘을 사용해 배열을 정렬합니다.

예제 파일 | **array-32.php**

```
1  <?php
2     $array1 = $array2 = array('IMG0.png', 'img12.png', 'img10.png',
      'img2.png', 'img1.png', 'IMG3.png');
3
4     echo "기본 정렬<br>";
5     sort($array1);
6     print_r($array1);
7
8     echo "<br>";
9
10    echo "자연 순서<br>";
11    natcasesort($array2);
12    print_r($array2);
13
14 ?>
```

화면 출력

```
기본 정렬
Array ( [0] => IMG0.png [1] => IMG3.png [2] => img1.png [3] => img10.png [4]
=> img12.png [5] => img2.png )
자연 순서
```

```
Array ( [0] => IMG0.png [4] => img1.png [3] => img2.png [5] => IMG3.png [2] =>
img10.png [1] => img12.png )
```

02.4.5 다차원 정렬

내장 함수 array_multisort()는 다중 또는 다차원 배열을 정렬합니다. array_multisort()는 한 번에 여러 배열을 정렬하거나, 하나 이상의 차원(다차원) 배열을 정렬하는 데 사용할 수 있습니다.

| 내장 함수 |

```
bool array_multisort ( array &$array1 [, mixed $array1_sort_order = SORT_ASC [,
mixed $array1_sort_flags = SORT_REGULAR [, mixed $... ]]] )
```

예제 파일 | **array-33.php**

```
1  <?php
2     $ar1 = array(10, 100, 100, 0);
3     $ar2 = array(1, 3, 2, 4);
4     array_multisort($ar1, $ar2);
5
6     var_dump($ar1);
7     var_dump($ar2);
8
9  ?>
```

화면 출력

```
array(4) { [0]=> int(0) [1]=> int(10) [2]=> int(100) [3]=> int(100) }
array(4) { [0]=> int(4) [1]=> int(1) [2]=> int(2) [3]=> int(3) }
```

위 예제에서는 다중 정렬합니다. 첫 번째 배열은 0, 10, 100, 100 형태로 정렬됩니다. 첫 번째 정렬 순서를 따라서 두 번째 배열은 4, 1, 2, 3 형태로 정렬됩니다.

02.5 배열 외부 함수

배열의 데이터를 처리할 때 처리하고자 하는 방식으로 별도의 외부 함수를 생성하여 작업을 위임할 수 있습니다. 생성한 외부 함수에 배열의 데이터를 하나씩 전달하여 함수를 처리하게 됩니다.

02.5.1 외부 함수 호출

내장 함수 array_walk()는 배열의 인자값을 응용한 사용자 정의 함수를 호출할 수 있습니다.

| 내장 함수 | 배열 사용자 함수 호출

bool **array_walk** (array &$array , callable $callback [, mixed $userdata = NULL])

예제 파일 | **array-34.php**

```
<?php
   $arr = array('하나'=>'orage', '둘'=>'apple', '셋'=>'grape');

   function arr_print($arr1, $key) {
      // 호출된 배열의 값을 출력합니다.
      echo $key ." = ". $arr1."<br>";
   }

   // 배열값과 사용자 정의 함수 arr_print를 호출합니다.
   array_walk($arr,'arr_print');

   // 배열의 포인터를 처음 위치로 초기화합니다.
   reset($arr);

   echo "<br>";

   function arrKey_print(&$arr1, $key, $value) {
      echo $value . $key ." = ". $arr1 . "<br>";

```

```
20      }
21      // 배열값과 사용자 정의 함수 arrKey_print를 호출합니다.
22      array_walk($arr,'arrKey_print',"테스트2: ");
23
24      // 배열의 포인터를 처음 위치로 초기화합니다.
25      reset($arr);
26
27  ?>
```

화면 출력

```
하나 = orage
둘 = apple
셋 = grape

테스트2: 하나 = orage
테스트2: 둘 = apple
테스트2: 셋 = grape
```

| 내장 함수 |

bool **array_walk_recursive** (array &$array , callable $callback [, mixed $userdata = NULL])

내장 함수 array_walk_recursive()는 지정한 콜백 함수를 배열 각각의 모든 요소에게 반복적으로 함수를 적용합니다.

예제 파일 | **array-35.php**

```
1   <?php
2      $sweet = array('a' => 'apple', 'b' => 'banana');
3      $fruits = array('sweet' => $sweet, 'sour' => 'lemon');
4
5      print_r($fruits);
6      echo "<br>";
7
8      function test_print($item, $key)
9      {
10           echo "$key ==> $item <br>";
```

```
11      }
12
13      array_walk_recursive($fruits, 'test_print');
14
15  ?>
```

화면 출력

```
Array ( [sweet] => Array ( [a] => apple [b] => banana ) [sour] => lemon )
a ==> apple
b ==> banana
sour ==> lemon
```

위의 예제는 다중 배열로 설정되어 있습니다. 다중 배열의 각각의 요소에 사용자 지정 test_print() 함수를 적용합니다.

02.5.2 외부 함수 정렬

내부 정렬 알고리즘 이외에 별도의 함수를 생성하여 정렬 처리를 위임할 수 있습니다.

| 내장 함수 | 외부 함수 데이터 정렬

```
bool uasort ( array &$array , callable $value_compare_func )
```

내장 함수 uasort()는 배열 정보와 외부 함수를 통하여 정렬합니다. 이때 인덱스 키 값은 유지합니다.

예제 파일 | **array-36.php**

```
1   <?php
2       // 비교 함수
3       function cmp($a, $b) {
4           if ($a == $b) return 0;
5           else return ($a < $b) ? -1 : 1;
6       }
7
8       // 배열
```

```
    $array = array('a' => 5, 'b' => 7, 'c' => -1, 'd' => -6, 'e' => 3, 'f'
    => 2, 'g' => 8, 'h' => -3);
    print_r($array);

    echo "<br>";

    // 외부 비교 함수를 통하여 정렬
    uasort($array, 'cmp');
    print_r($array);

?>
```

화면 출력

```
Array ( [a] => 5 [b] => 7 [c] => -1 [d] => -6 [e] => 3 [f] => 2 [g] => 8 [h] =>
-3 )
Array ( [d] => -6 [h] => -3 [c] => -1 [f] => 2 [e] => 3 [a] => 5 [b] => 7 [g]
=> 8 )
```

| 내장 함수 | 외부 함수 키 정렬

bool **uksort** (array &$array , callable $key_compare_func)

내장 함수 uksort()는 배열 정보와 외부 함수를 통하여 정렬합니다.

예제 파일 | **array-37.php**

```
<?php
    // 비교 함수
    function cmp($a, $b) {
        if ($a == $b) return 0;
        else return ($a < $b) ? -1 : 1;
    }

    // 배열
    $array = array('a' => 5, 'b' => 7, 'c' => -1, 'd' => -6, 'e' => 3, 'f'
    => 2, 'g' => 8, 'h' => -3);
    print_r($array);

    echo "<br>";
```

```
13
14      // 외부 비교 함수를 통하여 정렬
15      uksort($array, 'cmp');
16      print_r($array);
17
18  ?>
```

화면 출력

```
Array ( [a] => 5 [b] => 7 [c] => -1 [d] => -6 [e] => 3 [f] => 2 [g] => 8 [h] =>
-3 )
Array ( [a] => 5 [b] => 7 [c] => -1 [d] => -6 [e] => 3 [f] => 2 [g] => 8 [h] =>
-3 )
```

| 내장 함수 | 외부 함수 값 정렬

bool **usort** (array &$array , callable $value_compare_func)

내장 함수 usort()는 배열 정보와 외부 함수를 통하여 정렬합니다.

예제 파일 | **array-38.php**

```
1   <?php
2       // 비교 함수
3       function cmp($a, $b) {
4           if ($a == $b) return 0;
5           else return ($a < $b) ? -1 : 1;
6       }
7
8       // 배열
9       $array = array('a' => 5, 'b' => 7, 'c' => -1, 'd' => -6, 'e' => 3, 'f'
        => 2, 'g' => 8, 'h' => -3);
10      print_r($array);
11
12      echo "<br>";
13
14      // 외부 비교 함수를 통하여 정렬
15      usort($array, 'cmp');
16      print_r($array);
```

```
17
18  ?>
```

화면 출력

```
Array ( [a] => 5 [b] => 7 [c] => -1 [d] => -6 [e] => 3 [f] => 2 [g] => 8 [h] =>
-3 )
Array ( [0] => -6 [1] => -3 [2] => -1 [3] => 2 [4] => 3 [5] => 5 [6] => 7 [7]
=> 8 )
```

02.5.3 콜백

내장 함수 array_filter()는 콜백 함수를 이용하여 배열을 필터링합니다.

| 내장 함수 |

```
array array_filter ( array $array [, callable $callback [, int $flag = 0 ]] )
```

배열의 각 값을 반복하여 콜백 함수에 전달합니다. 콜백 함수가 true를 반환하면 array의 현재 값이 결과 배열로 반환됩니다. 배열 키는 보존됩니다.

예제 파일 | array-39.php

```
1   <?php
2      function odd($var)
3      {
4         // returns whether the input integer is odd
5         return($var & 1);
6      }
7
8      function even($var)
9      {
10        // returns whether the input integer is even
11     return(!($var & 1));
12     }
13
14     $array1 = array("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);
15     $array2 = array(6, 7, 8, 9, 10, 11, 12);
```

```
16
17      echo "Odd :\n";
18      print_r(array_filter($array1, "odd"));
19
20      echo "Even:\n";
21      print_r(array_filter($array2, "even"));
22
23  ?>
```

화면 출력

```
Odd : Array ( [a] => 1 [c] => 3 [e] => 5 )
Even: Array ( [0] => 6 [2] => 8 [4] => 10 [6] => 12 )
```

| 내장 함수 |

array **array_map** (callable $callback , array $array1 [, array $...])

내장 함수 array_map()은 지정된 배열 요소 각각에 콜백 함수를 적용합니다. array_map()은 배열 각각에 콜백 함수를 적용한 후 array1의 모든 요소를 포함하는 배열을 반환합니다. 콜백 함수로 전달되는 매개변수의 개수는 array_map()에 전달된 배열의 개수와 일치해야 합니다.

예제 파일 | **array-40.php**

```
1   <?php
2       function cube($n)
3       {
4           // 3제곱 값을 반환
5           return($n * $n * $n);
6       }
7
8       $a = array(1, 2, 3, 4, 5);
9       $b = array_map("cube", $a);
10      print_r($b);
11
12  ?>
```

화면 출력

```
Array ( [0] => 1 [1] => 8 [2] => 27 [3] => 64 [4] => 125 )
```

| 내장 함수 |

mixed **array_reduce** (array $array , callable $callback [, mixed $initial = NULL])

내장 함수 array_reduce()는 콜백 함수를 반복적으로 배열 요소에 적용하여 배열을 단일 값으로 줄입니다.

예제 파일 | **array-41.php**

```php
<?php
    function sum($carry, $item)
    {
        $carry += $item;
        return $carry;
    }

    function product($carry, $item)
    {
        $carry *= $item;
        return $carry;
    }

    $a = array(1, 2, 3, 4, 5);
    $x = array();

    var_dump(array_reduce($a, "sum"));
    // int(15)

    var_dump(array_reduce($a, "product", 10));
    // int(1200), because: 10*1*2*3*4*5
    // 입력한 10도 같이 포함하여 동작함

    var_dump(array_reduce($x, "sum", "No data to reduce"));
    // string(17) "No data to reduce"
    // 문자열의 개수만큼 반복된 17번의 $carry 값이 출력됨
?>
```

화면 출력

```
int(15)
int(1200)
string(17)
"No data to reduce"
```

02.6 배열 연산

배열은 여러 개의 유사한 값들을 가지고 있습니다. 이러한 특성을 위해서 배열의 상태나 연산 등의 특수 처리를 할 수도 있습니다.

02.6.1 개수

배열을 처리할 때는 반복문을 사용하여 처리하는 경우가 대부분 입니다. 이런 경우 배열을 처리하기 위해서 반복 횟수를 알아야 합니다.

PHP는 배열의 크기를 통하여 요소의 개수를 확인할 수 있는 특별한 함수를 제공합니다.

| 내장 함수 |

```
int count ( mixed $array_or_countable [, int $mode = COUNT_NORMAL ] )
```

내장 함수 count()는 배열의 데이터의 개수가 몇 개가 있는지 확인할 수 있는 간단한 함수입니다.

count() 함수를 이용하면 배열에 들어 있는 데이터 요소의 개수를 확인할 수 있습니다. sizeof()는 count()의 또 다른 별칭입니다.

예제 파일 | array-42.php

```
1  <?php
2     $arr = array("hojin","jiny","james","eric");
3     echo "배열의 개수는 = ". count($arr) . " 입니다.<br>";
```

```
4
5       echo "배열의 개수는 = ". sizeof($arr) . " 입니다.<br>";
6
7   ?>
```

화면 출력

```
배열의 개수는 = 4 입니다.
배열의 개수는 = 4 입니다.
```

| 내장 함수 |

array **array_pad** (array $array , int $size , mixed $value)

내장 함수 array_pad()는 입력된 배열과 함께 지정된 개수(size)의 배열을 반환합니다.

지정한 배열의 개수의 값이 양수면, 입력 배열 뒤쪽에 $value 값으로 채워진 배열을 생성하여 반환합니다. 또는 지정한 배열의 개수의 값이 음수면, 입력 배열 앞쪽에 $value 값으로 채워진 배열을 생성합니다.

만일 지정한 배열의 개수가 입력된 배열의 개수보다 작은 경우에는 패딩 처리를 하지 않습니다. 한 번에 최대 생성할 수 있는 배열의 개수는 1048576개입니다.

예제 파일 | **array-43.php**

```
1   <?php
2       $input = array(12, 10, 9);
3
4       $result = array_pad($input, 5, 0);
5       // result is array(12, 10, 9, 0, 0)
6       print_r($result);
7       echo "<br>";
8
9       $result = array_pad($input, -7, -1);
10      // result is array(-1, -1, -1, -1, 12, 10, 9)
11      print_r($result);
12      echo "<br>";
13
```

```
14    $result = array_pad($input, 2, "noop");
15    // not padded
16    print_r($result); echo "<br>";
17
18 ?>
```

화면 출력

```
Array ( [0] => 12 [1] => 10 [2] => 9 [3] => 0 [4] => 0 )
Array ( [0] => -1 [1] => -1 [2] => -1 [3] => -1 [4] => 12 [5] => 10 [6] => 9 )
Array ( [0] => 12 [1] => 10 [2] => 9 )
```

02.6.2 범위

내장 함수 range()는 정수값을 갖는 배열이 있을 때 값의 범위를 구하여 데이터를 추출할 수 있습니다.

| 내장 함수 | 정수값의 배열

```
array range ( mixed $start , mixed $end [, number $step = 1 ] )
```

예제 파일 | **array-44.php**

```
1  <?php
2     // 배열
3     $array = array('a' => 5, 'b' => 7, 'c' => -1, 'd' => -6, 'e' => 3, 'f'
      => 2, 'g' => 8, 'h' => -3);
4
5     $a = range(3,10);
6     print_r($a);
7
8  ?>
```

화면 출력

```
Array ( [0] => 3 [1] => 4 [2] => 5 [3] => 6 [4] => 7 [5] => 8 [6] => 9 [7] =>
10 )
```

02.6.3 연산 처리

내장 함수 array_sum()은 숫자값의 배열이 있을 때 배열값의 총합을 출력합니다.

| 내장 함수 | 배열의 합

```
number array_sum ( array $array )
```

예제 파일 | array-45.php

```
<?php
    // 배열
    $array = array('a' => 5, 'b' => 7, 'c' => -1, 'd' => -6, 'e' => 3, 'f'
    => 2, 'g' => 8, 'h' => -3);

    $a = array_sum($array);
    echo "배열의 합 = ".$a;

?>
```

화면 출력

```
배열의 합 = 15
```

| 내장 함수 | 배열의 곱

```
number array_product ( array $array )
```

내장 함수 array_product()은 입력된 배열의 값을 모두 곱하여 출력합니다.

예제 파일 | array-46.php

```
<?php

    $a = array(1, 2, 3, 4, 5);
    echo "배열의 곱 = " . array_product($a);


?>
```

화면 출력

```
배열의 곱 = 120
```

02.6.4 비정렬

배열을 처리하는 데 있어서 정렬 처리하여 사용하는 경우가 많이 있습니다. 이와 반대로 배열의 데이터를 비정렬화하여 처리할 수 있습니다.

PHP는 데이터의 비정렬화를 할 수 있는 몇 가지 함수를 제공합니다.

| 내장 함수 | 배열을 뒤섞음

```
bool shuffle ( array &$array )
```

내장 함수 shuffle()은 입력된 배열의 순서를 뒤섞어서 출력합니다.

예제 파일 | **array-47.php**

```
1  <?php
2     // 배열
3     $array = array('a' => 5, 'b' => 7, 'c' => -1, 'd' => -6, 'e' => 3, 'f'
      => 2, 'g' => 8, 'h' => -3);
4     print_r($array);
5
6     echo "<br> === 배열 뒤섞음 === <br>";
7
8     shuffle($array);
9     print_r($array);
10
11 ?>
```

화면 출력

```
Array ( [a] => 5 [b] => 7 [c] => -1 [d] => -6 [e] => 3 [f] => 2 [g] => 8 [h] =>
-3 )
=== 배열 뒤섞음 ===
Array ( [0] => 7 [1] => -6 [2] => 8 [3] => 5 [4] => 3 [5] => 2 [6] => -3 [7] =>
-1 )
```

| 내장 함수 | 배열 난수

```
mixed array_rand ( array $array [, int $num = 1 ] )
```

내장 함수 array_rand()는 입력된 배열에서 지정한 수만큼의 임의의 키를 반환합니다.

예제 파일 | array-48.php

```
<?php
    $country = array("korea", "usa", "japan", "china", "frach", "canada");

    // 배열에서 임의 값 2개를 추출합니다.
    $keys = array_rand($country, 2);

    echo $country[$keys[0]] . "\n";
    echo $country[$keys[1]] . "\n";

?>
```

화면 출력

```
japan china
```

| 내장 함수 |

```
array array_replace ( array $array1 , array $array2 [, array $... ] )
```

내장 함수 array_replace()는 배열 array1의 값을 다음 입력되는 배열의 동일한 키를 기준으로 값을 바꿉니다. 또한 세 번째 배열에서 동일한 키 값이 있을 경우 계속 대체됩니다. 배열은 순서대로 이전 값을 덮어씁니다.

예제 파일 | array-49.php

```
<?php
    $base = array("orange", "banana", "apple", "raspberry");
    $replacements = array(0 => "pineapple", 4 => "cherry");

```

```
    $basket = array_replace($base, $replacements);
    print_r($basket);

?>
```

화면 출력

```
Array ( [0] => pineapple [1] => banana [2] => apple [3] => raspberry [4] =>
cherry )
```

| 내장 함수 |

array **array_replace_recursive** (array $array1 , array $array2 [, array $...])

내장 함수 array_replace_recursive()는 배열의 요소를 재귀적으로 대체합니다. array_replace_recursive()는 array1의 값을 다음 배열의 값과 동일하게 대체합니다.

첫 번째 배열의 키가 두 번째 배열에 존재하면 기존 값은 두 번째 배열의 값으로 대체됩니다. 키가 첫 번째 배열이 아닌 두 번째 배열에 존재하면 첫 번째 배열에 키가 만들어집니다. 키 값이 첫 번째 배열에만 있는 경우 그대로 유지합니다.

예제 파일 | **array-50.php**

```
<?php
    $base = array('citrus' => array( "orange") ,
                  'berries' =>array("blackberry", "raspberry"), );
    $replacements = array('citrus' => array('pineapple'),
                  'berries' => array('blueberry'));

    $basket = array_replace_recursive($base, $replacements);
    print_r($basket);
    echo "<br>";

    $basket = array_replace($base, $replacements);
    print_r($basket);

?>
```

화면 출력

```
Array ( [citrus] => Array ( [0] => pineapple ) [berries] => Array ( [0] =>
blueberry [1] => raspberry ) )
Array ( [citrus] => Array ( [0] => pineapple ) [berries] => Array ( [0] =>
blueberry ) )
```

02.6.5 변환

내장 함수 array_change_key_case()는 인덱스 키의 대문자 또는 소문자로 변환합니다.

| 내장 함수 | 인덱스 키 대소문자 변환

array **array_change_key_case** (array $array [, int $case = CASE_LOWER])

예제 파일 | **array-51.php**

```
1  <?php
2     $arr = array("FirSt" => 1, "SecOnd" => 4);
3
4     // 대문자
5     $UPPER = array_change_key_case($arr, CASE_UPPER);
6     print_r($UPPER);
7
8     echo "<br> ===== <br>";
9     // 소문자
10    $LOWER = array_change_key_case($arr, CASE_LOWER);
11    print_r($LOWER);
12
13 ?>
```

화면 출력

```
Array ( [FIRST] => 1 [SECOND] => 4 )
=====
Array ( [first] => 1 [second] => 4 )
```

02.6.6 집합

내장 함수 array_intersect()는 키 값은 유지하면서 2개의 배열의 교차 부분(교집합)을 계산하여 배열로 반환합니다. 값을 기준으로 비교합니다.

| 내장 함수 |

```
array array_intersect ( array $array1 , array $array2 [, array $... ] )
```

예제 파일 | **array-52.php**

```
<?php
    $array1 = array("a" => "green", "red", "blue");
    $array2 = array("b" => "green", "yellow", "red");
    $result = array_intersect($array1, $array2);
    print_r($result);

?>
```

화면 출력

```
Array ( [a] => green [0] => red )
```

| 내장 함수 |

```
array array_intersect_assoc ( array $array1 , array $array2 [, array $... ] )
```

내장 함수 array_intersect_assoc()는 array_intersect()와 달리 키 값도 비교하는 데 사용합니다. 2개의 배열을 비교하여 교차 부분(교집합)만 배열로 반환합니다.

예제 파일 | **array-53.php**

```
<?php
    $array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
    $array2 = array("a" => "green", "b" => "yellow", "blue", "red");
    $result_array = array_intersect_assoc($array1, $array2);
```

```
    print_r($result_array);

?>
```

화면 출력

```
Array ( [a] => green )
```

| 내장 함수 |

array **array_intersect_key** (array $array1 , array $array2 [, array $...])

내장 함수 array_intersect_key()는 키 값을 사용하여 배열의 교차 부분을 계산합니다. array_intersect_key()는 array1의 키를 기준으로 교차되는 모든 항목의 배열을 반환합니다.

예제 파일 | **array-54.php**

```
<?php
    $array1 = array('blue'  => 1, 'red'  => 2, 'green'  => 3,
                    'purple' => 4);
    $array2 = array('green' => 5, 'blue' => 6, 'yellow' => 7,
                    'cyan'   => 8);

    var_dump(array_intersect_key($array1, $array2));

?>
```

화면 출력

```
array(2) { ["blue"]=> int(1) ["green"]=> int(3) }
```

| 내장 함수 |

array **array_intersect_uassoc** (array $array1 , array $array2 [, array $...], callable $key_compare_func)

내장 함수 array_intersect_uassoc()는 콜백 함수를 사용하여 인덱스를 비교합니다. array_intersect_uassoc()는 array1을 기준으로 배열의 교차 부분을 비교합니다.

예제 파일 | **array-55.php**

```php
<?php
   $array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
   $array2 = array("a" => "GREEN", "B" => "brown", "yellow", "red");

   print_r(array_intersect_uassoc($array1, $array2, "strcasecmp"));

?>
```

화면 출력

```
Array ( [b] => brown )
```

| 내장 함수 |

array **array_intersect_ukey** (array $array1 , array $array2 [, array $...], callable $key_compare_func)

내장 함수 array_intersect_ukey()는 외부 콜백 함수를 통하여 두 배열의 교차 부분(교집합)을 계산합니다. array_intersect_ukey()는 일치하는 키를 가진 array1을 기준으로 배열을 반환합니다.

예제 파일 | **array-56.php**

```php
<?php
   function key_compare_func($key1, $key2)
   {
      if ($key1 == $key2)
         return 0;
      else if ($key1 > $key2)
         return 1;
      else
         return -1;
   }
```

```
11
12      $array1 = array('blue'  => 1, 'red'  => 2, 'green'  => 3,
13                      'purple' => 4);
14      $array2 = array('green' => 5, 'blue' => 6, 'yellow' => 7,
15                      'cyan'   => 8);
16
17      var_dump(array_intersect_ukey($array1, $array2, 'key_compare_func'));
18
19  ?>
```

화면 출력

```
array(2) { ["blue"]=> int(1) ["green"]=> int(3) }
```

| 내장 함수 |

array **array_uintersect** (array $array1 , array $array2 [, array $...], callable $value_compare_func)

내장 함수 array_uintersect()는 콜백 함수를 통하여 배열의 교차점을 비교합니다.

예제 파일 | **array-57.php**

```
1   <?php
2      $array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
3      $array2 = array("a" => "GREEN", "B" => "brown", "yellow", "red");
4
5      print_r(array_uintersect($array1, $array2, "strcasecmp"));
6
7   ?>
```

화면 출력

```
Array ( [a] => green [b] => brown [0] => red )
```

| 내장 함수 |

array **array_uintersect_assoc** (array $array1 , array $array2 [, array $...], callable $value_compare_func)

내장 함수 array_uintersect_assoc()는 인덱스 검사를 통해 배열의 교차점을 계산 후에 콜백 함수를 통하여 데이터를 비교합니다.

예제 파일 | **array-58.php**

```
<?php
   $array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
   $array2 = array("a" => "GREEN", "B" => "brown", "yellow", "red");

   print_r(array_uintersect_assoc($array1, $array2, "strcasecmp"));

?>
```

화면 출력

```
Array ( [a] => green )
```

| 내장 함수 |

array **array_uintersect_uassoc** (array $array1 , array $array2 [, array $...], callable **$value**_compare_func , callable **$key**_compare_func)

내장 함수 array_uintersect_uassoc()는 인덱스 검사를 통해 배열의 교차점을 찾은 후에 별도의 콜백 함수를 통하여 데이터와 인덱스를 비교합니다.

예제 파일 | **array-59.php**

```
<?php
   $array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
   $array2 = array("a" => "GREEN", "B" => "brown", "yellow", "red");

   print_r(array_uintersect_uassoc($array1, $array2, "strcasecmp",
```

```
    "strcasecmp"));

?>
```

화면 출력

```
Array ( [a] => green [b] => brown )
```

02.7 비교

배열의 데이터 값, 키를 비교하여 처리할 수 있습니다.

02.7.1 값의 비교

| 내장 함수 |

```
array array_diff ( array $array1 , array $array2 [, array $... ] )
```

내장 함수 array_diff()는 배열의 차이를 계산합니다. array1을 다른 배열과 비교합니다. 비교 후에 array1에는 있지만 다른 배열에는 없는 값들을 반환합니다.

예제 파일 | array-60.php

```
<?php
    $array1 = array("a" => "green", "red", "blue", "red");
    $array2 = array("b" => "green", "yellow", "red");
    $result = array_diff($array1, $array2);

    print_r($result);

?>
```

화면 출력

```
Array ( [1] => blue )
```

array **array_udiff** (array $array1 , array $array2 [, array $...], callable $value_compare_func)

내장 함수 array_udiff()는 콜백 함수를 사용하여 배열의 차이를 계산합니다. 데이터 비교를 위한 내부 함수를 사용하는 array_diff()와는 차이가 있습니다.

예제 파일 | **array-61.php**

```php
<?php
   // Arrays to compare
   $array1 = array(new stdclass, new stdclass,
              new stdclass, new stdclass,
             );

   $array2 = array(
              new stdclass, new stdclass,
             );

   // Set some properties for each object
   $array1[0]->width = 11; $array1[0]->height = 3;
   $array1[1]->width = 7;  $array1[1]->height = 1;
   $array1[2]->width = 2;  $array1[2]->height = 9;
   $array1[3]->width = 5;  $array1[3]->height = 7;

   $array2[0]->width = 7;  $array2[0]->height = 5;
   $array2[1]->width = 9;  $array2[1]->height = 2;

   function compare_by_area($a, $b) {
      $areaA = $a->width * $a->height;
      $areaB = $b->width * $b->height;

      if ($areaA < $areaB) {
         return -1;
      } elseif ($areaA > $areaB) {
         return 1;
      } else {
```

```
            return 0;
        }
    }

    print_r(array_udiff($array1, $array2, 'compare_by_area'));

?>
```

화면 출력

```
Array ( [0] => stdClass Object ( [width] => 11 [height] => 3 ) [1] => stdClass
Object ( [width] => 7 [height] => 1 ) )
```

| 내장 함수 |

array **array_diff_assoc** (array $array1 , array $array2 [, array $...])

내장 함수 array_diff_assoc()는 인덱스를 기준으로 배열의 차이를 계산합니다.

array1과 array2를 비교하여 차이를 반환합니다. 비교 후에 array1에 있고 다른 배열에는 없는 값들을 반환합니다. array_diff()와 달리 배열 키도 비교에 사용합니다.

예제 파일 | **array-62.php**

```
<?php
    $array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
    $array2 = array("a" => "green", "yellow", "red");
    $result = array_diff_assoc($array1, $array2);
    print_r($result);

?>
```

화면 출력

```
Array ( [b] => brown [c] => blue [0] => red )
```

array **array_udiff_assoc** (array $array1 , array $array2 [, array $...], callable $value_compare_func)

내장 함수 array_udiff_assoc()는 콜백 함수를 통하여 인덱스 검사하고 배열의 데이터를 비교합니다.

예제 파일 | **array-63.php**

```
<?php
   class cr {
      private $priv_member;
      function cr($val)
      {
         $this->priv_member = $val;
      }

      static function comp_func_cr($a, $b)
      {
         if ($a->priv_member === $b->priv_member) return 0;
         return ($a->priv_member > $b->priv_member)? 1:-1;
      }
   }

   $a = array("0.1" => new cr(9), "0.5" => new cr(12),
               0 => new cr(23), 1=> new cr(4), 2 => new cr(-15),);
   $b = array("0.2" => new cr(9), "0.5" => new cr(22),
               0 => new cr(3), 1=> new cr(4), 2 => new cr(-15),);

   $result = array_udiff_assoc($a, $b, array("cr", "comp_func_cr"));
   print_r($result);
?>
```

화면 출력

```
Array ( [0.1] => cr Object ( [priv_member:cr:private] => 9 ) [0.5] => cr Object ( [priv_member:cr:private] => 12 ) [0] => cr Object ( [priv_member:cr:private] => 23 ) )
```

| 내장 함수 |

array **array_udiff_uassoc** (array $array1 , array $array2 [, array $...], callable $value_compare_func , callable $key_compare_func)

내장 함수 array_udiff_uassoc()는 콜백 함수로 데이터와 인덱스를 비교합니다.

먼저 인덱스 검사를 통해 배열의 차이점을 계산합니다. 계산된 데이터 및 인덱스를 콜백 함수를 통하여 비교합니다.

예제 파일 | **array-64.php**

```
<?php

   class cr {
      private $priv_member;
      function cr($val)
      {
         $this->priv_member = $val;
      }

      static function comp_func_cr($a, $b)
      {
         if ($a->priv_member === $b->priv_member) return 0;
         return ($a->priv_member > $b->priv_member)? 1:-1;
      }

      static function comp_func_key($a, $b)
      {
         if ($a === $b) return 0;
         return ($a > $b)? 1:-1;
      }
   }

   $a = array("0.1" => new cr(9), "0.5" => new cr(12),
               0 => new cr(23), 1=> new cr(4), 2 => new cr(-15),);
   $b = array("0.2" => new cr(9), "0.5" => new cr(22),
               0 => new cr(3), 1=> new cr(4), 2 => new cr(-15),);

```

```
28    $result = array_udiff_uassoc($a, $b, array("cr", "comp_func_cr"),
    array("cr", "comp_func_key"));
29    print_r($result);
30
31 ?>
```

화면 출력

```
Array ( [0.1] => cr Object ( [priv_member:cr:private] => 9 ) [0.5] => cr Object (
[priv_member:cr:private] => 12 ) [0] => cr Object ( [priv_member:cr:private]
=> 23 ) )
```

| 내장 함수 |

array **array_diff_uassoc** (array $array1 , array $array2 [, array $...], callable $key_compare_func)

내장 함수 array_diff_uassoc()는 사용자 지정 콜백 함수를 통하여 배열을 검사합니다. array1과 array2를 비교하여 차이를 반환합니다.

사용자가 제공하는 콜백 함수는 인덱스 비교에 사용됩니다.

예제 파일 | **array-65.php**

```
1  <?php
2     function key_compare_func($a, $b)
3     {
4        if ($a === $b) {
5           return 0;
6        }
7        return ($a > $b)? 1:-1;
8     }
9
10    $array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");
11    $array2 = array("a" => "green", "yellow", "red");
12    $result = array_diff_uassoc($array1, $array2, "key_compare_func");
13    print_r($result);
14 ?>
```

화면 출력

```
Array ( [b] => brown [c] => blue [0] => red )
```

02.7.2 키의 비교

| 내장 함수 |

array **array_diff_key** (array $array1 , array $array2 [, array $...])

내장 함수 array_diff_key()는 키를 사용하여 배열의 차이점을 계산합니다.

array1의 키와 array2의 키를 비교하여 차이를 배열로 반환합니다. 이 함수는 array_diff()와 비슷하지만 비교 값이 키에서 수행된다는 점이 다릅니다.

예제 파일 | **array-66.php**

```
1  <?php
2     $array1 = array('blue' => 1, 'red' => 2, 'green' => 3, 'purple' => 4);
3     $array2 = array('green' => 5, 'yellow' => 7, 'cyan' => 8);
4
5     var_dump(array_diff_key($array1, $array2));
6
7  ?>
```

화면 출력

```
array(3) { ["blue"]=> int(1) ["red"]=> int(2) ["purple"]=> int(4) }
```

| 내장 함수 |

array **array_diff_ukey** (array $array1 , array $array2 [, array $...], callable $key_compare_func)

내장 함수 array_diff_ukey()는 사용자 콜백 함수를 통하여 배열의 키를 비교합니다.

array1의 키와 array2의 키를 비교하여 차이를 반환합니다. 이 함수는 array_diff()와 비슷합니다만 비교 값 대신 키에서 수행된다는 점에서 차이가 있습니다.

예제 파일 | **array-67.php**

```php
<?php
   function key_compare_func($key1, $key2)
   {
      if ($key1 == $key2)
         return 0;
      else if ($key1 > $key2)
         return 1;
      else
         return -1;
   }

   $array1 = array('blue'  => 1, 'red'  => 2,
                   'green'  => 3, 'purple' => 4);
   $array2 = array('green' => 5, 'blue' => 6,
                   'yellow' => 7, 'cyan'   => 8);

   var_dump(array_diff_ukey($array1, $array2, 'key_compare_func'));

?>
```

화면 출력

```
array(2) { ["red"]=> int(2) ["purple"]=> int(4) }
```