jinyPHP 3권 - 03. 배열

**02. 배열**

====================

배열은 여러 개의 값을 가지는 유형의 데이터 타입입니다. 여러 개의 값을 가지고 있는 배열의 특성상 값의 정렬, 분리, 결합등의 추가적인 작업들이 가능합니다.

PHP는 배열 데이터에 대해서 추가 작업들을 할 수 있도록 내부함수를 제공합니다.

**02.1 배열생성**

====================

배열 변수를 생성할 수 있습니다. 또한 배열을 여러 개의 변수로 분할 하여 처리를 할 수도 있습니다. 배열을 변수화 하여 데이터를 처리하는 것은 배열을 활용하는데 매우 유용합니다.

**02.1.1 변수할당**

====================

배열 변수를 생성하기 위해서는 데이터의 값을 배열타입으로 저장을 하면 됩니다. 배열타입의 값을 초기에 성성을 하기 위해서는 array() 함수를 이용합니다.

**|내부함수| 배열생성**

|  |
| --- |
| array **array** ([ mixed $... ] ) |

내부함수 array()는 입력된 값에 대한 배열데이터를 반한합니다. 각각의 데이터는 콤마(,)로 구분하면 키를 삽입을 할때는 => 기호를 같이 사용을 합니다.

**예제파일) array-01.php**

|  |
| --- |
| <?php  // 배열 변수를 생성합니다.  // 입력한 값에 대한 배열 값을 반환 합니다.  $arr1 = array("hojin","jiny","james","eric");  // 배열값을 foreach 를 통하여 하나씩 출력을 합니다.  foreach ($arr1 as $value) {  echo "$value <br>";  }  echo "===== <br>";    // 키형태의 배열을 지정합니다.  $arr2 = array('firstname' => "hojin", 'lastname' => "lee" );  foreach ($arr2 as $key => $value) {  echo "{$key} => {$value} ";  print\_r($arr);  }  ?> |

**화면출력)**

hojin

jiny

james

eric

=====

firstname => hojin lastname => lee

배열은 값의 묽음처리 입니다. 묽음 형태의 배열을 처리하는 것은 유사한 데이터를 처리하는데 있어서 많은 변수의 명을 통하여 자원을 낭비하는 것을 줄이기 위해서 입니다.

**|내부함수| 배열의 변수화**

|  |
| --- |
| array **list** ( mixed $var1 [, mixed $... ] ) |

하지만 반대로 배열을 여러 개의 변수로 분할 할 수 있습니다. 내부함수 list()는 배열의 값을 지정한 각각의 변수에 값을 할당할 수 있습니다. 배열의 데이터를 여러 개의 변수에 분배하여 저장을 할 때 편리합니다.

**예제파일) 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>";  }  ?> |

**화면출력)**

변수 orage = orage

변수 apple = apple

변수 grape = grape

=====

0==>orage

1==>apple

2==>grape

**|내부함수|**

|  |
| --- |
| array **array\_fill** ( int $start\_index , int $num , mixed $value ) |

내부함수 array\_fill()는 지정한 값으로 배열 채우기 합니다. 지정한 값을 배열의 시작위치(start\_index) 부터 반복횟수(num) 만큼 반복하여 배열을 생성합니다.

**예제파일) array-03.php**

|  |
| --- |
| <?php  $a = array\_fill(5, 6, 'aaa');  $b = array\_fill(-2, 4, 'bbb');  print\_r($a);  print\_r($b);  ?> |

**화면출력)**

Array ( [5] => aaa [6] => aaa [7] => aaa [8] => aaa [9] => aaa [10] => aaa )

Array ( [-2] => bbb [0] => bbb [1] => bbb [2] => bbb )

**|내부함수|**

|  |
| --- |
| array **array\_fill\_keys** ( array $keys , mixed $value ) |

내부함수 array\_fill\_keys()는 입력받은 배열의 값을 키 값으로 사용하여 배열의 데이터를 채우기 합니다.

**예제파일) array-04.php**

|  |
| --- |
| <?php  $keys = array('foo', 5, 10, 'bar');  $a = array\_fill\_keys($keys, 'banana');  print\_r($a);  ?> |

**화면출력)**

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

**02.1.2 값과 키**

====================

PHP는 다양한 형태의 최신 배열표기 및 처리를 지원합니다. 일반배열과 키값을 가지고 있는 연상배열 모두를 사용할 수 있습니다. 내부함수 each() 함수는 배열의 키와 값을 한쌍으로 반한홥니다. list() 함수와 결합하여 배열의 키와 값을 추출할 수 있습니다.

**|내부함수| 배열의 쌍값**

|  |
| --- |
| array **each** ( array &$array ) |

배열이 담긴 each() 함수를 호출할때마다 다음 배열 위치로 전달됩니다.

**예제파일) 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>";  }  ?> |

**화면출력)**

원소 0= 0

원소 1= orage

원소 key= 0

원소 value= orage

1==>apple

2==>grape

위의 실습에서 배령의 키와 값을 추울하여 봅니다. 첫번째 명령에서 1번, 반복문에서 2번 실행되어 총 3개의 배열 정보가 출력됩니다. each() 함수는 4가지의 키값으로 데이터를 반환하는데 0, key 와 1, value 입니다.

연상배열에서 값만 추출한 다른 배열을 생성할 수있습니다. 내부함수 array\_values() 는 배열의 값을 반환합니다.

**|내부함수| 배열 값**

|  |
| --- |
| array **array\_values** ( array $array ) |

**예제파일) array-06.php**

|  |
| --- |
| <?php  // 배열을 생성합니다.  $arr = array('firstname' => "hojin", 'lastname' => "lee" );  while(list($key,$val) = each($arr)){  echo $key. "==>" .$val. ", ";  }  echo "<br> ==== <br>";  **$arr2 = array\_values($arr);**  while(list($key,$val) = each($arr2)){  echo $key. "==>" .$val. ", ";  }  ?> |

**화면출력)**

firstname==>hojin, lastname==>lee,

====

0==>hojin, 1==>lee,

**|내부함수| 배열 인덱스키**

|  |
| --- |
| array **array\_keys** ( array $array [, mixed $search\_value = null [, bool $strict = false ]] ) |

내부함수 array\_keys()는 배열의 인덱스 키 이름만을 반환 받을 수 있습니다.

**예제파일) array-07.php**

|  |
| --- |
| <?php  $arr = array('firstname' => "hojin", 'lastname' => "lee" );  $index = array\_keys($arr);  while(list($key,$val) = each($index)){  echo $key. "==>" .$val. "<br>";  }  ?> |

**화면출력)**

0==>firstname

1==>lastname

**|내부함수|**

|  |
| --- |
| array **array\_column** ( array $input , mixed $column\_key [, mixed $index\_key = null ] ) |

내부함수 array\_column()는 입력된 다중 배열에서 단일 열에서 값 만을 추출하여 배열로 반환합니다.

**예제파일) array-08.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);  ?> |

**화면출력)**

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

**|내부함수|**

|  |
| --- |
| array **array\_flip** ( array $array ) |

내부함수 array\_flip()는 배열의 키와 값을 서로 교환합니다. array\_flip ()은 배열의 키가 값이 됩니다. 값은 다시 배열의 키로 변경을 하여 배열로 처리합니다.

**예제파일) array-09.php**

|  |
| --- |
| <?php  $input = array("a"=>"oranges", "b"=>"apples", "c"=>"pears");  $flipped = array\_flip($input);  print\_r($flipped);  ?> |

**화면출력)**

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

**02.1.3 변수 결합과 분리**

====================

내부함수 compact()는 외부의 변수명을 응용하여 배열을 생성할 수 있습니다. 변수명은 키값으로 사용됩니다.

**|내부함수| 변수 배열생성**

|  |
| --- |
| array **compact** ( mixed $varname1 [, mixed $... ] ) |

**예제파일) array-10.php**

|  |
| --- |
| <?php    $one = "하나";  $two = "두개";  $three = "세개";  $four = "네게";  // 배열에 변수의 값을 넣기 위해서는 "" 로 변수명을 입력 하면 됩니다.  **$num** = array("three", "four");  // 배열을 넣을때는 배열명만 인수로 전달 합니다.  $arr = **compact**("one", "two", **$num**);  while(list($key,$val) = each($arr)){  echo $key. "==>" .$val. "<br>";  }  ?> |

**화면출력)**

one==>하나

two==>두개

three==>세개

four==>네게

**|내부함수| 배열 변수화**

|  |
| --- |
| int **extract** ( array &$array [, int $flags = EXTR\_OVERWRITE [, string $prefix = NULL ]] ) |

내부함수 extract()는 배열의 키이름으로 각각 변수화 합니다. compact() 함수의 반대기능입니다.

**예제파일) array-11.php**

|  |
| --- |
| <?php  $arr = array("blue"=>"파랑색","red"=>"발강색","black"=>"검정색");  // 배열을 변수로 생성합니다.  extract($arr);  echo "blue = ".$blue ."<br>";  echo "red = ".$red ."<br>";  echo "black = ".$black ."<br>";  ?> |

**화면출력)**

blue = 파랑색

red = 발강색

black = 검정색

**02.1.4 결합 및 분리**

====================

내부함수 array\_merge()는 입력되는 배열을 하나의 배열 형태로 결합하여 반환을 합니다.

**|내부함수| 배열결합**

|  |
| --- |
| array **array\_merge** ( array $array1 [, array $... ] ) |

**예제파일) 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>";  }  ?> |

**화면출력)**

0==>지니

1==>호진

2==>철수

3==>영수

4==>제니퍼

5==>홍길동

**|내부함수| 재귀적 배열결합**

|  |
| --- |
| array **array\_merge\_recursive** ( array $array1 [, array $... ] ) |

두 개 이상의 배열을 재귀적으로 병합합니다. 내부함수 array\_merge\_recursive ()는 다수의 배열 요소를 병합하여 배열의 끝에 추가합니다.

병합될 배열에 동일한 문자열 키가있는 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 덮어 쓰지 않고 뒤에 추가됩니다.

**예제파일) 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);  ?> |

**화면출력)**

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

**|내부함수| 배열결합 (키/값)**

|  |
| --- |
| array **array\_combine** ( array $keys , array $values ) |

내부함수 array\_combine()는 첫번재 배열을 키값으로, 두번째 배열을 값 형태로 배열을 결합합니다.

**예제파일) array-14.php**

|  |
| --- |
| <?php  $a = array('green', 'red', 'yellow');  $b = array('avocado', 'apple', 'banana');  // $a 배열은 키명으로 사용고, $b는 데이터로 사용합니다.  // 두개의 배열을 연상배열 형태로 결합니다.  $arr = array\_combine($a, $b);  print\_r($arr);  ?> |

**화면출력)**

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

**|내부함수| 일부배열 분리**

|  |
| --- |
| array **array\_slice** ( array $array , int $offset [, int $length = NULL [, bool $preserve\_keys = false ]] ) |

내부함수 array\_slice()는 배열의 일부분의 요소만을 추출하여 반환합니다.

**예제파일) array-15.php**

|  |
| --- |
| <?php  // 배열을 생성합니다.  $arr = array('하나', '둘', '셋', '넷', '다섯', '여섯', '일곱', '여덟', '아홉', '열');    // 첫번째 인자: 추출 시작값  // 처음 2부터 끝까지.  $slice = array\_slice($arr, 2);  while(list($key,$val) = each($slice)){  echo $key. "==>" .$val. ", ";  }  echo "<br>";    // 두번째 인자: 추출 데이터 갯수  // 처음 1부터, 3개의 데이터를 추출  $slice = array\_slice($arr, 1,3);  while(list($key,$val) = each($slice)){  echo $key. "==>" .$val. ",";  }  echo "<br>";    // 처음 인자가 음수이면 끝에서 -인자 수터 시작하여 , 두번째 인자 갯수를 추출  // 마지막 -5 위치부터(여섯), 3개의 데이터를 추출  $slice = array\_slice($arr, -5,3);  while(list($key,$val) = each($slice)){  echo $key. "==>" .$val. ",";  }  ?> |

**화면출력)**

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

0==>둘,1==>셋,2==>넷,

0==>여섯,1==>일곱,2==>여덜,

**|내부함수|**

|  |
| --- |
| array **array\_chunk** ( array $array , int $size [, bool $preserve\_keys = false ] ) |

내부함수 array\_chunk()는 배열을 크기 요소가있는 배열로 분할합니다. 마지막 chunk는 크기 요소보다 작은 요소를 포함 할 수 있습니다.

**예제파일) array-16.php**

|  |
| --- |
| <?php  $array = array('a', 'b', 'c', 'd', 'e');  print\_r(array\_chunk($array, 2));  echo "<br>";  print\_r(array\_chunk($array, 3));  echo "<br>";  ?> |

**화면출력)**

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**

====================

내부함수 array\_reverse()는 배열의 내용을 역순으로 출력합니다.

**|내부함수| 역순배열**

|  |
| --- |
| array **array\_reverse** ( array $array [, bool $preserve\_keys = false ] ) |

**예제파일) array-17.php**

|  |
| --- |
| <?php  $arr = array("hojin","jiny","james","eric");  $reverse = array\_reverse($arr);  while(list($key,$val) = each($reverse)){  echo $key. "==>" .$val. "<br>";  }  ?> |

**화면출력)**

0==>eric

1==>james

2==>jiny

3==>hojin

**02.1.6 추가 및 삭제**

====================

내부함수 array\_push()는 배열의 마지막 위치에 새로운 원소 데이터를 추가합니다.

**|내부함수| 배열 원소추가(마지막)**

|  |
| --- |
| int **array\_push** ( array &$array , mixed $value1 [, mixed $... ] ) |

**예제파일) array-18.php**

|  |
| --- |
| <?php  // 배열을 생성합니다.  $arr = array('orage', 'apple', 'grape');  // $arr 배열에 두개의 원소를 추가합니다.  array\_push($arr, 'banana', 'tomato');  while(list($key,$val) = each($arr)){  echo $key. "==>" .$val. "<br>";  }  ?> |

**화면출력)**

0==>orage

1==>apple

2==>grape

3==>banana

4==>tomato

**|내부함수| 배열 원소제거**

|  |
| --- |
| mixed **array\_pop** ( array &$array ) |

내부함수 array\_pop()은 배열의 마지막 요소 하나를 제거합니다. 이는 데이터처리 알고리즘 pop 과 유사합니다.

**예제파일) array-19.php**

|  |
| --- |
| <?php  // 배열을 생성합니다.  $arr = array('orage', 'apple', 'grape');  // 마지막 배열 원소를 삭제 합니다.  **$temp = array\_pop($arr);**  echo "배열에서 마지막 $temp 요소 하나를 제거합니다.<br> ";  while(list($key,$val) = each($arr)){  echo $key. "==>" .$val. "<br>";  }  ?> |

**화면출력)**

배열에서 마지막 grape 요소 하나를 제거합니다.

0==>orage

1==>apple

**|내부함수| 배열 원소추가(처음)**

|  |
| --- |
| int **array\_unshift** ( array &$array , mixed $value1 [, mixed $... ] ) |

내부함수 array\_unshift()는 배열 앞쪽에 새로운 원소를 추가합니다.

**예제파일) array-20.php**

|  |
| --- |
| <?php  // 배열을 생성합니다.  $arr = array('orage', 'apple', 'grape');    // $arr 배열에 두개의 원소를 추가합니다.  array\_unshift($arr, 'banana', 'tomato');    while(list($key,$val) = each($arr)){  echo $key. "==>" .$val. "<br>";  }  ?> |

**화면출력)**

0==>banana

1==>tomato

2==>orage

3==>apple

4==>grape

**|내부함수| 배열 원소제거(처음)**

|  |
| --- |
| mixed array\_shift ( array &$array ) |

내부함수 array\_shift()는 배열의 맨 처음 요소 하나를 제거합니다. 배열을 좌측으로 하나 밀어서 제거하는 형태 입니다.

**예제파일) array-21.php**

|  |
| --- |
| <?php  // 배열을 생성합니다.  $arr = array('orage', 'apple', 'grape');    // 배열 원소를 삭제 합니다.  $temp = array\_shift($arr);  echo "배열에서 처음 $temp 요소 하나를 제거합니다.<br> ";    while(list($key,$val) = each($arr)){  echo $key. "==>" .$val. "<br>";  }  ?> |

**화면출력)**

배열에서 처음 orage 요소 하나를 제거합니다.

0==>apple

1==>grape

**|내부함수| 일부배열 삭제**

|  |
| --- |
| array **array\_splice** ( array &$input , int $offset [, int $length = count($input) [, mixed $replacement = array() ]] ) |

내부함수 array\_splice()는 일부분의 데이터를 삭제한 후에 값을 반환합니다.

**예제파일) array-22.php**

|  |
| --- |
| <?php  // 배열을 생성합니다.  $arr = array('하나', '둘', '셋', '넷', '다섯', '여섯', '일곱', '여덜', '아옵', '열');  echo "원본<br>";  while(list($key,$val) = each($arr)){  echo $key. "==>" .$val. ", ";  }  echo "<br><br>";  // 첫번째 인자: 삭제 시작값  // 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>";  ?> |

**화면출력)**

원본

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

0==>여덜, 1==>아옵, 2==>열, 를 삭제하였습니다.

0==>넷, 1==>다섯, 2==>여섯, 를 삭제하였습니다.

**03.2 배열검사**

====================

배열은 여러 개의 데이터를 포함하고 있습니다. 같은 유형의 데이터의 묽음을 처리하는데 있어서 값을 검사하고 처리하데 매우 유용합니다.

PHP는 배열의 데이터를 검색하고 처리하는 몇 개의 내부함수들을 지원합니다.

**03.2.1 검사**

====================

내부함수 in\_array()는 배열에 값이 존재하는지 검사를 할 수 있습니다. 값이 있는 경우에는 true 값을 반환합니다.

**|내부함수| 배열 검색**

|  |
| --- |
| bool **in\_array** ( mixed $needle , array $haystack [, bool $strict = FALSE ] ) |

**예제파일) array-23.php**

|  |
| --- |
| <?php  $arr = array("blue"=>"파랑색","red"=>"발강색","black"=>"검정색");  if(in\_array("파랑색",$arr)){  echo "값이 존재합니다.";  } else {  echo "배열 값이 없습니다.";  }  ?> |

**화면출력)**

값이 존재합니다.

**|내부함수| 배열 인텍스키 확인**

|  |
| --- |
| bool **array\_key\_exists** ( mixed $key , array $array ) |

내부함수 array\_key\_exists()는 연상배열안에 인텍스 키값이 존재하는지를 확인합니다.

**예제파일) 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 "키값이 없습니다.";  }  ?> |

**화면출력)**

배열에 키 값이 존재합니다.

**|내부함수|**

|  |
| --- |
| mixed **array\_search** ( mixed $needle , array $haystack [, bool $strict = false ] ) |

내부함수 array\_search()는 지정한 배열에서 값을 검색하여 일치하는 키값을 반환합니다.

**예제파일) array-25.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);  ?> |

**화면출력)**

2

1

**03.2.2 중복**

====================

내부함수 array\_count\_values()는 배열안에 들어 있는 동일한 값의 개수를 반환합니다.

**|내부함수| 중복값 체크**

|  |
| --- |
| array **array\_count\_values** ( array $array ) |

**예제파일) array-26.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>";  }  ?> |

**화면출력)**

orage==>3

apple==>2

grape==>2

**|내부함수| 중복값 제거**

|  |
| --- |
| array **array\_unique** ( array $array [, int $sort\_flags = SORT\_STRING ] ) |

내부함수 array\_unique()는 입력된 배열값 중에서 중복된 내용은 제거합니다.

**예제파일) array-27.php**

|  |
| --- |
| <?php  $arr = array("a" => "green", "red", "b" => "green", "blue", "red");  $result = array\_unique($arr);  print\_r($result);  ?> |

**화면출력)**

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

**03.4 배열위치**

======================

배열은 여러 개의 데이터의 값을 가지고 있습니다. 인덱스 또는 키를 통하여 데이터를 접근을 할 수 있습니다.

그외에 배열의 위치값을 이용하여 데이터에 접근 및 값을 읽어 올 수 있습니다. PHP는 배열의 위치값을 제어할 수 있는 몇가지 내부함수들을 지원하고 있습니다.

**03.4.1 위치이동**

======================

**|내부함수| 배열포인트 초기화**

|  |
| --- |
| mixed **reset** ( array &$array ) |

내부함수 reset()는 배열의 포인트를 초기화 합니다. 배열의 처음 위치를 가르키게 됩니다.

**|내부함수| 배열포인트 다음**

|  |
| --- |
| mixed **next** ( array &$array ) |

내부함수 next()는 배열의 다음포인트로 이동을 합니다. 함수를 한번 호출할때마다 배열의 포인트를 다음으로 이동합니다.

**|내부함수| 배열포인트 마지막**

|  |
| --- |
| mixed **end** ( array &$array ) |

내부함수 end()는 배열의 마지막 포인트로 이동합니다.

**|내부함수| 배열포인트 이전**

|  |
| --- |
| mixed **prev** ( array &$array ) |

내부함수 prev()는 배열의 이전포인트로 이동합니다. 함수를 한번 호출할때마다 배열의 포인트를 이전으로 이동합니다.

**예제파일) array-28.php**

|  |
| --- |
| <?php  // 배열을 생성합니다.  $arr = array('하나', '둘', '셋', '넷', '다섯', '여섯', '일곱', '여덜', '아옵', '열');  echo "원본<br>";  while(list($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>";  ?> |

**화면출력)**

원본

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

=======================

현재 포인트 = 둘

현재 포인트 = 셋

현재 포인트 = 열

현재 포인트 = 아옵

위의 실험은 배열의 위치를 지정하는 함수들을 이용하여 배열의 데이터를 출력해 보는 예제입니다.

**03.4.2 위치확인**

======================

배열의 위치함수를 이용하게 되면 배열의 데이터를 가르키는 포인터에 변경이 되게 됩니다. 현재 가르키고 있는 값과 키을 알수 있는 함수들도 같이 제공을 합니다.

**|내부함수| 배열 현재위치**

|  |
| --- |
| mixed **current** ( array &$array ) |

내부함수 current()는 배열의 현재 위치의 값을 반환합니다. pos() 함수는 current() 함수의 또다른 이름의 별칭입니다.

**|내부함수| 배열포인트 키값**

|  |
| --- |
| mixed **key** ( array &$array ) |

내부함수 key()는 현재의 배열포인트의 키값을 출력합니다.

**예제파일) array-29.php**

|  |
| --- |
| <?php  $arr = array( 'fruit1' => 'apple', 'fruit2' => 'orange', 'fruit3' => 'grape', 'fruit4' => 'apple', 'fruit5' => 'apple');  while ($name = current($arr)) {  if ($name == 'apple') {  echo key($arr).'<br>';  }    next($arr);  }  ?> |

**화면출력)**

fruit1

fruit4

fruit5

**03.5 배열정렬**

======================

배열을 이용하면 배열의 값을 알파벳 또는 숫자를 기준으로 오름차순 내림차순으로 정렬을 할 수 있습니다.

**03.5.1 오름차순**

======================

**|내부함수| 오름차순 정렬**

|  |
| --- |
| bool **sort** ( array &$array [, int $sort\_flags = SORT\_REGULAR ] ) |

내부함수 sort()를 이용하면 배열의 값을 오름차순으로 정렬 합니다.

**|내부함수| 이중 오름차순 정렬**

|  |
| --- |
| bool **asort** ( array &$array [, int $sort\_flags = SORT\_REGULAR ] ) |

내부함수 asort()는 연상배열의 값일 경우 네임키를 우선적으로 오름차순으로 정렬된 상태에서 서브 데이터값을 오름차순으로 정렬을 합니다.

**|내부함수| 이름키 오름차순 정렬**

|  |
| --- |
| bool **ksort** ( array &$array [, int $sort\_flags = SORT\_REGULAR ] ) |

내부함수 ksort()는 연상배열일 경우 이름키에 대해서 오름차순으로 정렬을 합니다.

**03.5.2 내림차순**

======================

**|내부함수| 내림차순 정렬**

|  |
| --- |
| bool **rsort** ( array &$array [, int $sort\_flags = SORT\_REGULAR ] ) |

내부함수 rsort()는 오름차순의 반내인 내림차순으로 배열의 값을 정렬 합니다.

**|내부함수| 이름키 내림차순 정렬**

|  |
| --- |
| bool **krsort** ( array &$array [, int $sort\_flags = SORT\_REGULAR ] ) |

내부함수 krsort()는 연상배열일 경우 이름키에 대해서 내림차순으로 정렬을 합니다.

**|내부함수| 내림차순 배열**

|  |
| --- |
| bool **arsort** ( array &$array [, int $sort\_flags = SORT\_REGULAR ] ) |

내부함수 arsort()는 인텍스 값을 유지한 상태에서 배열 요소의 데이터 값만 내림차순으로 재정렬 합니다.

**03.5.3 정렬 예제**

======================

내림차순 및 오름차순 정렬을 코드 실험을 통하여 확인해 보도록 하겠습니다.

**예제파일) array-30.php**

|  |
| --- |
| <?php  $arr = array('one'=>'하나', 'two'=>'둘', 'three'=>'셋', 'four'=>'넷', 'five'=>'다섯', 'six'=>'여섯', 'seven'=>'일곱');    echo "오름차순 정렬<br>";  sort($arr);  while(list($key,$val) = each($arr)){  echo $key. "==>" .$val. ", ";  }  echo "<br>";  echo "내림차순 정렬<br>";  rsort($arr);  while(list($key,$val) = each($arr)){  echo $key. "==>" .$val. ", ";  }  echo "<br>";  echo "이중 오름차순 정렬<br>";  asort($arr);  while(list($key,$val) = each($arr)){  echo $key. "==>" .$val. ", ";  }  echo "<br>";  echo "이름키 오름차순 정렬정렬<br>";  ksort($arr);  while(list($key,$val) = each($arr)){  echo $key. "==>" .$val. ", ";  }  echo "<br>";  echo "이름키 내림차순 정렬정렬<br>";  krsort($arr);  while(list($key,$val) = each($arr)){  echo $key. "==>" .$val. ", ";  }  ?> |

**화면출력)**

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,

이름키 오름차순 정렬정렬

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

이름키 내림차순 정렬정렬

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

**03.5.4 자연순서 정렬**

======================

내부함수 natsort()은 「자연 순서」알고리즘을 사용해 배열을 정렬합니다

**|내부함수|**

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

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

**예제파일) array-31.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);  ?> |

**화면출력)**

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 )

**|내부함수|**

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

내부함수 natcasesort()는 natsort()의 대소 문자를 구별하지 않는 버전입니다.

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

**예제파일) array-32.php**

|  |
| --- |
| <?php  $array1 = $array2 = array('IMG0.png', 'img12.png', 'img10.png', 'img2.png', 'img1.png', 'IMG3.png');  echo "기본 정렬<br>";  sort($array1);  print\_r($array1);  echo "<br>";  echo "자연 순서<br>";  natcasesort($array2);  print\_r($array2);  ?> |

**화면출력)**

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 )

**03.5.5 다차원 정렬**

======================

내부함수 array\_multisort()는 다중 또는 다차원 배열 정렬 합니다. array\_multisort ()는 한 번에 여러 배열을 정렬하거나 하나 이상의 차원, 다차원 배열을 정렬하는 데 사용할 수 있습니다.

**|내부함수|**

|  |
| --- |
| bool **array\_multisort** ( array &$array1 [, mixed $array1\_sort\_order = SORT\_ASC [, mixed $array1\_sort\_flags = SORT\_REGULAR [, mixed $... ]]] ) |

**예제파일) array-33.php**

|  |
| --- |
| <?php  $ar1 = array(10, 100, 100, 0);  $ar2 = array(1, 3, 2, 4);  array\_multisort($ar1, $ar2);  var\_dump($ar1);  var\_dump($ar2);  ?> |

**화면출력)**

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 다중정렬 합니다. 첫 번째 배열은 0, 10, 100, 100 형태로 정렬됩니다. 첫번째 정렬 순서를 따라서 두 번째 배열은 4, 1, 2, 3 형태로 정렬이 됩니다.

**03.6 배열 외부함수**

====================

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

**03.6.1 외부함수 호출**

====================

내부함수 array\_walk()는 배열의 인자값을 응용한 사용자 정의 함수를 호출 할 수 있습니다.

**|내부함수| 배열 사용자 함수호출**

|  |
| --- |
| bool **array\_walk** ( array &$array , callable $callback [, mixed $userdata = NULL ] ) |

**예제파일) array-34.php**

|  |
| --- |
| <?php  $arr = array('하나'=>'orage', '둘'=>'apple', '셋'=>'grape');  function arr\_print($arr1, $key){  // 호출된 배열의 값을 출력합니다.  echo $key ." = ". $arr1."<br>";  }  // 배열값과 사용정의 함수 arr\_print 를 호출합니다.  array\_walk($arr,'arr\_print');  // 배열의 포인터를 처음위치로 초기화 합니다.  reset($arr);  echo "<br>";  function arrKey\_print(&$arr1, $key, $value){  echo $value . $key ." = ". $arr1 . "<br>";  }  // 배열값과 사용정의 함수 arrKey\_print 를 호출합니다.  array\_walk($arr,'arrKey\_print',"테스트2: ");  // 배열의 포인터를 처음위치로 초기화 합니다.  reset($arr);  ?> |

**화면출력)**

하나 = orage

둘 = apple

셋 = grape

테스트2: 하나 = orage

테스트2: 둘 = apple

테스트2: 셋 = grape

**|내부함수|**

|  |
| --- |
| bool **array\_walk\_recursive** ( array &$array , callable $callback [, mixed $userdata = NULL ] ) |

내부함수 array\_walk\_recursive()은 지정한 콜백함수를 배열 각각의 모든 요소에게 반복적으로 함수를 적용합니다.

**예제파일) array-35.php**

|  |
| --- |
| <?php  $sweet = array('a' => 'apple', 'b' => 'banana');  $fruits = array('sweet' => $sweet, 'sour' => 'lemon');  print\_r($fruits);  echo "<br>";  function test\_print($item, $key)  {  echo "$key ==> $item <br>";  }  array\_walk\_recursive($fruits, 'test\_print');  ?> |

**화면출력)**

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

a ==> apple

b ==> banana

sour ==> lemon

위의 예제는 다중배열로 설정되어 있습니다. 다중 배열의 각각의 요소에 사용자 지정 test\_print()함수를 적용합니다.

**03.6.2 외부함수 정렬**

====================

내부 정렬알고리즘 이외에 별도의 함수를 생성하여 정렬처리를 위임할 수 있습니다.

**|내부함수| 외부함수 데이터 정렬**

|  |
| --- |
| bool **uasort** ( array &$array , callable $value\_compare\_func ) |

내부함수 uasort()는 배열정보와 외부함수를 통하여 정렬을 합니다. 이때 인덱스 키 값은 유지합니다

**예제파일) array-36.php**

|  |
| --- |
| <?php  // 비교함수  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>";  // 외부 비교함수를 통하여 정렬  uasort($array, 'cmp');  print\_r($array);  ?> |

**화면출력)**

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 )

**|내부함수| 외부함수 키 정렬**

|  |
| --- |
| bool **uksort** ( array &$array , callable $key\_compare\_func ) |

내부함수 uksort()는 배열정보와 외부함수를 통하여 정렬을 합니다

**예제파일) array-37.php**

|  |
| --- |
| <?php  // 비교함수  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>";  // 외부 비교함수를 통하여 정렬  uksort($array, 'cmp');  print\_r($array);  ?> |

**화면출력)**

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 )

**|내부함수| 외부함수 값 정렬**

|  |
| --- |
| bool **usort** ( array &$array , callable $value\_compare\_func ) |

내부함수 usort()는 배열정보와 외부함수를 통하여 정렬을 합니다

**예제파일) array-38.php**

|  |
| --- |
| <?php  // 비교함수  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>";  // 외부 비교함수를 통하여 정렬  usort($array, 'cmp');  print\_r($array);  ?> |

**화면출력)**

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 )

**03.6.3 콜백**

====================

내부함수 array\_filter()는 콜백 함수를 이용하여 배열을 필터링합니다.

**|내부함수|**

|  |
| --- |
| 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.

**예제파일) array-39.php**

|  |
| --- |
| <?php  function odd($var) {  // returns whether the input integer is odd  return($var & 1);  }  function even($var) {  // returns whether the input integer is even  return(!($var & 1));  }  $array1 = array("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);  $array2 = array(6, 7, 8, 9, 10, 11, 12);  echo "Odd :\n";  print\_r(array\_filter($array1, "odd"));  echo "Even:\n";  print\_r(array\_filter($array2, "even"));  ?> |

**화면출력)**

Odd : Array ( [a] => 1 [c] => 3 [e] => 5 )

Even: Array ( [0] => 6 [2] => 8 [4] => 10 [6] => 12 )

**|내부함수|**

|  |
| --- |
| array **array\_map** ( callable $callback , array $array1 [, array $... ] ) |

내부함수 array\_map()는 지정된 배열요소 각각에 콜백함수를 적용합니다. array\_map ()은 배열 각각에 콜백 함수를 적용한 후 array1의 모든 요소를 ​​포함하는 배열을 반환합니다. 콜백 함수로 전달되는 매개 변수의 개수는 array\_map ()에 전달 된 배열의 개수와 일치해야합니다.

**예제파일) array-40.php**

|  |
| --- |
| <?php  function cube($n) {  // 3승값을 반환  return($n \* $n \* $n);  }  $a = array(1, 2, 3, 4, 5);  $b = array\_map("cube", $a);  print\_r($b);  ?> |

**화면출력)**

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

**|내부함수|**

|  |
| --- |
| mixed **array\_reduce** ( array $array , callable $callback [, mixed $initial = NULL ] ) |

내부함수 array\_reduce()는 콜백 함수를 반복적으로 배열 요소에 적용하여 배열을 단일 값으로 줄입니다.

**예제파일) array-41.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 갯수 만큼 반복된 17번의 $carry 값이 출력됨  ?> |

**화면출력)**

int(15)

int(1200)

string(17)

"No data to reduce"

**03.7 배열연산**

======================

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

**03.7.1 갯수**

======================

배열을 처리할 때는 반복문을 사용하여 처리하는 경우가 대부분 입니다. 이런경우 배열을 처리하기 위해서 반복횟수를 알아야 합니다.

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

**|내부함수| 배열갯수**

|  |
| --- |
| int **count** ( mixed $array\_or\_countable [, int $mode = COUNT\_NORMAL ] ) |

내부함수 count()는 배열의 데이터의 개수가 몇개가 있지는 확인을 할 수 있는 간단한 함수가 있습니다.

count() 함수를 이용을 하시면 배열에 들어있는 데이터의 요소 개수를 확인 할 수 있습니다.

sizeof() 는 count()의 또다른 이름 별칭입니다.

**예제파일) array-42.php**

|  |
| --- |
| <?php  $arr = array("hojin","jiny","james","eric");  echo "배열의 갯수는 = ". count($arr) . " 입니다.<br>";    echo "배열의 갯수는 = ". sizeof($arr) . " 입니다.<br>";  ?> |

**화면출력)**

배열의 갯수는 = 4 입니다.

배열의 갯수는 = 4 입니다.

**|내부함수|**

|  |
| --- |
| array **array\_pad** ( array $array , int $size , mixed $value ) |

내부함수 array\_pad()는 입력된 배열과 같이 지정된 갯수(size)의 배열을 반환합니다.

지정한 배열의 갯수의 값이 양수이면, 입력 배열 뒤쪽에 $value 값으로 채워진 배열을 생성하여 반환을 합니다. 또는 지정한 배열의 갯수의 값이 음수이면, 입력 배열 앞쪽에 $value 값으로 채워진 배열을 생성합니다.

만일 지정한 배열의 갯수가 입력된 배열의 갯수보다 작은 경우에는 패딩 처리를 하지 않습니다. 한번에 최대 생성할 수 있는 배열의 갯수는 1048576 개 입니다.

**예제파일) array-43.php**

|  |
| --- |
| <?php  $input = array(12, 10, 9);  $result = array\_pad($input, 5, 0);  // result is array(**12, 10, 9**, 0, 0)  print\_r($result);  echo "<br>";  $result = array\_pad($input, -7, -1);  // result is array(-1, -1, -1, -1, **12, 10, 9**)  print\_r($result);  echo "<br>";  $result = array\_pad($input, 2, "noop");  // not padded  print\_r($result);  echo "<br>";  ?> |

**화면출력)**

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 )

**03.7.2 범위**

======================

내부함수 range()는 정수값을 가지는 배열이 있을때 값의 범위를 구하여 데이터를 추출할 수 있습니다.

**|내부함수| 정수값의 배열**

|  |
| --- |
| array **range** ( mixed $start , mixed $end [, number $step = 1 ] ) |

**예제파일) array-44.php**

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

**화면출력)**

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

**03.7.3 연산처리**

======================

내부함수 array\_sum()은 숫자값의 배열이 있을때, 배열의 값의 총 합을 출력합니다.

**|내부함수| 배열의합**

|  |
| --- |
| number **array\_sum** ( array $array ) |

**예제파일) 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;  ?> |

**화면출력)**

배열의 합 = 15

**|내부함수| 배열의 곱**

|  |
| --- |
| number **array\_product** ( array $array ) |

내부함수 array\_product()는 입력된 배열의 값을 모두 곱하여 출력합니다.

**예제파일) array-46.php**

|  |
| --- |
| <?php  $a = array(1, 2, 3, 4, 5);  echo "배열의 곱 = " . array\_product($a);  ?> |

**화면출력)**

배열의 곱 = 120

**03.7.4 비정렬**

======================

배열을 처리하는데 있어서 정렬처리하여 사용하는 경우가 많이 있습니다. 이와 반대로 배열의 데이터를 비정렬화 하여 처리할 수 있습니다.

PHP는 데이터의 비정렬화를 할 수 있는 몇 개의 함수들을 제공합니다.

**|내부함수| 배열을 뒤썩음**

|  |
| --- |
| bool **shuffle** ( array &$array ) |

내부함수 shuffle()는 입력된 배열의 순서를 뒤썩어서 출력합니다.

**예제파일) array-47.php**

|  |
| --- |
| <?php  // 배열  $array = array('a' => 5, 'b' => 7, 'c' => -1, 'd' => -6, 'e' => 3, 'f' => 2, 'g' => 8, 'h' => -3);  print\_r($array);  echo "<br> === 배열 뒤썩음 === <br>";  shuffle($array);  print\_r($array);  ?> |

**화면출력)**

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

=== 배열 뒤썩음 ===

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

**|내부함수| 배열난수**

|  |
| --- |
| mixed **array\_rand** ( array $array [, int $num = 1 ] ) |

내부함수 array\_rand()는 입력된 배열에서 지정한 수 많큼의 임이의 키를 반한합니다.

**예제파일) array-48.php**

|  |
| --- |
| <?php  $country = array("korea", "usa", "japan", "china", "frach", "canada");    // 배열에서 임의 값 2개를 추출 합니다.  $keys = array\_rand($country, 2);  echo $country[$keys[0]] . "\n";  echo $country[$keys[1]] . "\n";  ?> |

**화면출력)**

japan china

**|내부함수|**

|  |
| --- |
| array **array\_replace** ( array $array1 , array $array2 [, array $... ] ) |

내부함수 array\_replace()는 배열 array1의 값을 다음 입력되는 배열의 동일한 키를 기준으로 값을 바꿉니다. 또한 세번째 배열에서 동일한 키값이 있을 경우 계속 대체됩니다. 배열은 순서대로 이전 값을 덮어 씁니다.

**예제파일) array-49.php**

|  |
| --- |
| <?php  $base = array("orange", "banana", "apple", "raspberry");  $replacements = array(0 => "pineapple", 4 => "cherry");  $basket = array\_replace($base, $replacements);  print\_r($basket);  ?> |

**화면출력)**

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

**|내부함수|**

|  |
| --- |
| array **array\_replace\_recursive** ( array $array1 , array $array2 [, array $... ] ) |

내부함수 array\_replace\_recursive()는 배열의 요소를 재귀 적으로 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. 키값이 첫 번째 배열에만있는 경우 그대로 유지합니다.

**예제파일) 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);  ?> |

**화면출력)**

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

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

**03.7.5 변환**

======================

내부함수 array\_change\_key\_case()는 인텍스키의 대문자 또는 소문자로 변환을 합니다.

**|내부함수| 인덱스크 대소문자 변환**

|  |
| --- |
| array **array\_change\_key\_case** ( array $array [, int $case = CASE\_LOWER ] ) |

**예제파일) array-51.php**

|  |
| --- |
| <?php  $arr = array("FirSt" => 1, "SecOnd" => 4);  // 대문자  $UPPER = array\_change\_key\_case($arr, CASE\_UPPER);  print\_r($UPPER);  echo "<br> ===== <br>";  // 소문자  $LOWER = array\_change\_key\_case($arr, CASE\_LOWER);  print\_r($LOWER);  ?> |

**화면출력)**

Array ( [FIRST] => 1 [SECOND] => 4 )

=====

Array ( [first] => 1 [second] => 4 )

**03.7.7 집합**

======================

내부함수 array\_intersect()는 키 값은 유지하면서 두개의 배열의 교차 부분(교집합)을 계산하여 배열로 반환합니다. 값을 기준으로 비교합니다.

**|내부함수|**

|  |
| --- |
| array **array\_intersect** ( array $array1 , array $array2 [, array $... ] ) |

**예제파일) array-52.php**

|  |
| --- |
| <?php  $array1 = array("a" => "green", "red", "blue");  $array2 = array("b" => "green", "yellow", "red");  $result = array\_intersect($array1, $array2);  print\_r($result);  ?> |

**화면출력)**

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

**|내부함수|**

|  |
| --- |
| array **array\_intersect\_assoc** ( array $array1 , array $array2 [, array $... ] ) |

내부함수 array\_intersect\_assoc ()는 array\_intersect ()와 달리 키값도 같이 비교를 하는데 사용을 합니다. 두개의 배열을 비교하여 교차부분(교집합)만 배열로 반환을 합니다.

**예제파일) 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);  ?> |

**화면출력)**

Array ( [a] => green )

**|내부함수|**

|  |
| --- |
| array **array\_intersect\_key** ( array $array1 , array $array2 [, array $... ] ) |

내부함수 array\_intersect\_key()는 키값을 사용하여 배열의 교차 부분을 계산합니다. array\_intersect\_key ()는 array1의 키를 기준으로 교차되는 모든 항목을 배열을 반환합니다.

**예제파일) 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));  ?> |

**화면출력)**

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

**|내부함수|**

|  |
| --- |
| array **array\_intersect\_uassoc** ( array $array1 , array $array2 [, array $... ], callable $key\_compare\_func ) |

내부함수 array\_intersect\_uassoc()는 콜백 함수를 사용하여 인덱스를 비교합니다. array\_intersect\_uassoc()는 array1을 기준으로 배열의 교차부분을 비교합니다.

**예제파일) array-55.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"));  ?> |

**화면출력)**

Array ( [b] => brown )

**|내부함수|**

|  |
| --- |
| array **array\_intersect\_ukey** ( array $array1 , array $array2 [, array $... ], callable $key\_compare\_func ) |

내부함수 array\_intersect\_ukey()는 외부 콜백함수를 통하여 두 배열의 교차 부분(교집합)을 계산합니다. array\_intersect\_ukey ()는 일치하는 키를 가진 array1을 기준으로 배열을 반환합니다.

**예제파일) array-56.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);  $array2 = array(**'green' => 5**, **'blue' => 6**, 'yellow' => 7, 'cyan' => 8);  var\_dump(array\_intersect\_ukey($array1, $array2, 'key\_compare\_func'));  ?> |

**화면출력)**

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

**|내부함수|**

|  |
| --- |
| array **array\_uintersect** ( array $array1 , array $array2 [, array $... ], callable $value\_compare\_func ) |

내부함수 array\_uintersect()는 콜백 함수를 통하여 배열의 교차점을 비교합니다.

**예제파일) array-57.php**

|  |
| --- |
| <?php  $array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");  $array2 = array("a" => "GREEN", "B" => "brown", "yellow", "red");  print\_r(array\_uintersect($array1, $array2, "strcasecmp"));  ?> |

**화면출력)**

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

**|내부함수|**

|  |
| --- |
| array **array\_uintersect\_assoc** ( array $array1 , array $array2 [, array $... ], callable $value\_compare\_func ) |

내부함수 array\_uintersect\_assoc()는 인덱스 검사를 통해 배열의 교차점을 계산후에 콜백 함수를 통하여 데이터를 비교합니다.

**예제파일) 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"));  ?> |

**화면출력)**

Array ( [a] => green )

**|내부함수|**

|  |
| --- |
| array **array\_uintersect\_uassoc** ( array $array1 , array $array2 [, array $... ], callable $**value**\_compare\_func , callable $**key**\_compare\_func ) |

내부함수 array\_uintersect\_uassoc()는 인덱스 검사를 통해 배열의 교차점을 찾은후에 별도의 콜백 함수를 통하여 데이터와 인덱스를 비교합니다.

**예제파일) 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"));  ?> |

**화면출력)**

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

**03.8 비교**

======================

배열의 데이터값, 키를 비교하여 처리를 할 수 있습니다.

**03.8.1 값의 비교**

======================

**|내부함수|**

|  |
| --- |
| array **array\_diff** ( array $array1 , array $array2 [, array $... ] ) |

내부함수 array\_diff()는 배열의 차이를 계산합니다. array1을 다른 배열과 비교합니다. 비교후에 array1에 있고 다른 배열에는 없는 값들을 반환합니다.

**예제파일) 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);  ?> |

**화면출력)**

Array ( [1] => blue )

**|내부함수|**

|  |
| --- |
| array **array\_udiff** ( array $array1 , array $array2 [, array $... ], callable $value\_compare\_func ) |

내부함수 array\_udiff()는 콜백 함수를 사용하여 배열의 차이를 계산합니다. 데이터 비교를위한 내부 함수를 사용하는 array\_diff ()와는 차이가 있습니다.

**예제파일) array-61.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'));  ?> |

**화면출력)**

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

**|내부함수|**

|  |
| --- |
| array **array\_diff\_assoc** ( array $array1 , array $array2 [, array $... ] ) |

내부함수 array\_diff\_assoc()는 인덱스를 기준으로 배열의 차이를 계산합니다.

array1과 array2를 비교하여 차이를 반환합니다. 비교후에 array1에 있고 다른 배열에는 없는 값들을 반환합니다. array\_diff ()와 달리 배열 키도 비교에 사용합니다.

**예제파일) 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);  ?> |

**화면출력)**

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

**|내부함수|**

|  |
| --- |
| array **array\_udiff\_assoc** ( array $array1 , array $array2 [, array $... ], callable $value\_compare\_func ) |

내부함수 array\_udiff\_assoc()는 콜백 함수를 통하여 인덱스 검사하고 배열의 데이터를 비교합니다.

**예제파일) 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);  ?> |

**화면출력)**

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 ) )

**|내부함수|**

|  |
| --- |
| array **array\_udiff\_uassoc** ( array $array1 , array $array2 [, array $... ],  callable $value\_compare\_func , callable $key\_compare\_func ) |

내부함수 array\_udiff\_uassoc()는 콜백 함수로 데이터와 인덱스를 비교합니다.

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

**예제파일) 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),);  $result = array\_udiff\_uassoc($a, $b, array("cr", "**comp\_func\_cr**"), array("cr", "**comp\_func\_key**"));  print\_r($result);  ?> |

**화면출력)**

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 ) )

**|내부함수|**

|  |
| --- |
| array **array\_diff\_uassoc** ( array $array1 , array $array2 [, array $... ], callable $key\_compare\_func ) |

내부함수 array\_diff\_uassoc()는 사용자 지정 콜백함수를 통하여 배열을 검사합니다. array1과 array2를 비교하여 차이를 반환합니다.

사용자가 제공하는 콜백 함수는 인덱스 비교에 사용됩니다.

**예제파일) array-65.php**

|  |
| --- |
| <?php  function key\_compare\_func($a, $b)  {  if ($a === $b) {  return 0;  }  return ($a > $b)? 1:-1;  }  $array1 = array("a" => "green", "b" => "brown", "c" => "blue", "red");  $array2 = array("a" => "green", "yellow", "red");  $result = array\_diff\_uassoc($array1, $array2, "key\_compare\_func");  print\_r($result);  ?> |

**화면출력)**

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

**03.8.2 키의 비교**

======================

**|내부함수|**

|  |
| --- |
| array **array\_diff\_key** ( array $array1 , array $array2 [, array $... ] ) |

내부함수 array\_diff\_key()는 키를 사용하여 배열의 차이점을 계산합니다.

array1의 키와 array2의 키를 비교하여 차이를 배열로 반환합니다. 이 함수는 array\_diff ()와 비슷하지만 비교값이 키에서 수행된다는 점이 다릅니다.

**예제파일) array-66.php**

|  |
| --- |
| <?php  $array1 = array('blue' => 1, 'red' => 2, 'green' => 3, 'purple' => 4);  $array2 = array('green' => 5, 'yellow' => 7, 'cyan' => 8);  var\_dump(array\_diff\_key($array1, $array2));  ?> |

**화면출력)**

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

**|내부함수|**

|  |
| --- |
| array **array\_diff\_ukey** ( array $array1 , array $array2 [, array $... ], callable $key\_compare\_func ) |

내부함수 array\_diff\_ukey()는 사용자 콜백함수를 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.

**예제파일) array-67.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'));  ?> |

**화면출력)**

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