Info 303

Les Tableaux PHP

Alin F. Rabat C.

Université de Reims Champagne Ardenne

22 septembre 2018

Sommaire

1. Syntaxe de création d'un tableau
2. Quelques fonctions utiles

Un tableau en PHP est en fait une map ordonnée. Une map est un type qui associe des valeurs à des clés. On peut avoir, comme valeur d'un tableau, d'autres tableaux, multidimensionnels ou non.

Pour créer un tableau on utilise la commande `array()`. Elle prend un nombre illimité de paramètres, chacun séparé par une virgule, sous la forme d'une paire key => value.

```
array(
    key1  => value1,
    key2 => value2,
    key3 => value3,
    ...
)
```

La virgule après le dernier élément d'un tableau est optionnelle et peut ne pas être ajoutée.

Il est également possible d'utiliser la syntaxe courte `[]`.

Listing 1 – Tableau simple

```php
<?php
$array = array(
    "foo" => "bar",
    "bar" => "foo",
);

$array = [
    "foo" => "bar",
    "bar" => "foo",
];
?>
```

- La clé key est un entier, ou une chaîne de caractères. La valeur value peut être de n'importe quel type.
- Si plusieurs éléments dans la déclaration d'un tableau utilisent la même clé, seule la dernière sera utilisée, écrasant ainsi toutes les précédentes.

PHP effectue les modifications de type suivant pour la clé key :

- Les chaînes de caractères contenant un entier valide, sauf si le nombre est précédé d'un signe + seront modifiées en un type entier. I.e. la clé "8" sera actuellement stockée comme l'entier 8. D'un autre côté, "08" ne sera pas modifié, sachant que ce n'est pas un entier décimal valide.
- Les nombres à virgule flottante seront aussi modifiés en entier, ce qui signifie que la partie après la virgule sera tronquée. I.e. la clé 8.7 sera stockée sous l'entier 8.
- Les booléens seront modifiés en entier également, i.e. la clé true sera stockée sous l'entier 1 et la clé false sous l'entier 0.
- La valeur Null sera modifiée en une chaîne vide, i.e. la clé null sera stockée sous la chaîne de caractère "".
- Les tableaux et les objets ne peuvent pas être utilisés comme clé. Si vous le tentez, l'alerte suivante sera émise : Illegal offset type.

La clé key est optionnelle. Si elle n'est pas spécifiée, PHP utilisera un incrément de la dernière clé entière utilisée.

Listing 2 – Indexation automatique des tableaux

```php
<?php
$array = array("foo", "bar", "hello", "world");
?>
```

Pour accéder aux éléments d'un tableau on utilise la syntaxe à base de crochets. Les éléments d'un tableau sont accessibles en utilisant la syntaxe `array[key]`.

Listing 3 – Accès aux éléments d'un tableau

```php
<?php
$array = array(
    "foo" => "bar",
    42    => 24,
    "multi" => array(
         "dimensional" => array(
             "array" => "foo"
         )
    )
);

var_dump($array["foo"]);
var_dump($array[42]);
var_dump($array["multi"]["dimensional"]["array"]);
?>
```

Un tableau existant peut être modifié en y assignant explicitement des valeurs. L'assignation d'une valeur dans un tableau est effectuée en spécifiant la clé, entre crochets. Si aucune clé n'est fournie alors PHP attribue automatique comme clé le premier entier disponible.

```
$arr[cle] = valeur;
$arr[] = valeur;
// cle peut etre un entier ou une chaine de caracteres
// valeur peut etre n'importe quel type
```

Pour modifier une valeur en particulier, il convient d'assigner une valeur en spécifiant sa clé. Pour effacer une paire clé/valeur, il convient d'appeler la fonction `unset()` sur la clé désirée.

```php
<?php
$arr = array(5 => 1, 12 => 2);

$arr[] = 56;     // Identique a $arr[13] = 56;
                 // a cet endroit du script

$arr["x"] = 42; // Ceci ajoute un nouvel element au
                 // tableau avec la cle "x"

unset($arr[5]); // Ceci efface l'element du tableau

unset($arr);     // Ceci efface completement le tableau
?>
```

- Si aucune clé n'est spécifiée, l'indice maximal existant est repris, et la nouvelle clé sera ce nombre, plus 1 (mais au moins 0).
- Si aucun indice entier n'existe, la clé sera 0 (zéro).
- Notez que la clé entière maximale pour cette opération n'a pas besoin d'exister dans le tableau au moment de la manipulation. Elle doit seulement avoir existé dans le tableau à un moment ou un autre depuis la dernière fois où le tableau a été ré-indexé.

```php
<?php
// Creation d'un tableau simple.
$array = array(1, 2, 3, 4, 5);
print_r($array);

// Maintennant, on efface tous les elements,
// mais on conserve le tableau :
foreach ($array as $i => $value) {
    unset($array[$i]);
}
print_r($array);

// Ajout d'un element
// (notez que la nouvelle cle est 5, et non 0).
$array[] = 6;
print_r($array);

// Re-indexation :
$array = array_values($array);
$array[] = 7;
print_r($array);
?>
```

array_change_key_case()

- Permet de modifier la case de la clé d'un tableau.
- Renvoie un array dont les clés sont en minuscule ou en majuscule.
- Renvoie `FALSE` si la variable passée n'est pas un `array`.

Syntaxe :

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

Exemple :

```
$test_array = array ('nom' => 'ALIN', 'prenom' => 'Francois");
print_r (array_change_key_case ($test_array, CASE_UPPER));
```

array_fill()

- Permet de remplir un tableau sans fournir de clé d'index.
- Remplit un tableau de *num* cases avec la valeur *value*, la valeur de l'index du tableau débutant à la valeur *start_index*.

Syntaxe :

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

Exemple :

```
$default_array = array_fill (1,8,'default');
print_r($default_array);
```

array_fill_keys()

- Permet de remplir un tableau en précisant les clés d'index.
- Remplit une variable de type array avec la valeur du paramètre *value* en utilisant les valeurs du tableau *keys* comme clé d'index.

Syntaxe :

```
array array_fill_key (array $keys, mixed $value)
```

Exemple :

```
$keys = array ('nom','prenom','adresse','ville','pays','codePostal');
$default_array = array_fill_keys ($keys,'default');
print_r($default_array);
```

array_filter()

- Filtre les éléments d'un tableau grâce à une fonction utilisateur.
- Évalue chaque valeur du tableau array en les passant à la fonction utilisateur. Si la fonction utilisateur retourne TRUE, la valeur courante du tableau array est retournée dans le tableau résultant. Les clés du tableau sont préservées.

Syntaxe :

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

Liste des paramètres

- **array** : Le tableau à évaluer
- **callback** : La fonction utilisateur à utiliser. Si aucune fonction utilisateur n'est fournie, toutes les entrées du tableau array valant FALSE (voir la conversion en booléen) seront effacées.
- **flag** : Drapeau indiquant quels sont les arguments à envoyer au paramètre callback :
  - ARRAY_FILTER_USE_KEY : ne passer que la clé comme seul argument à callback au lieu de la valeur
  - ARRAY_FILTER_USE_BOTH : passer à la fois la valeur et la clé comme arguments de callback au lieu de la valeur

Valeur de retour Retourne le tableau filtré.

Exemple

```php
<?php
function odd($var)
{
    // retourne lorsque l'entree est impaire
    return($var & 1);
}

function even($var)
{
    // retourne lorsque l'entree est paire
    return(!($var & 1));
}

$array1 = array("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);
$array2 = array(6, 7, 8, 9, 10, 11, 12);

echo "Impair :\n";
print_r(array_filter($array1, "odd"));
echo "Pair :\n";
print_r(array_filter($array2, "even"));
?>
```

array_flip()

Remplace les clés par les valeurs, et les valeurs par les clés

Syntaxe :

```
array array_flip ( array $array )
```

`array_flip()` retourne un tableau dont les clés sont les valeurs du précédent tableau array, et les valeurs sont les clés.

- Les valeurs de array doivent être des clés valides, c'est-à-dire qu'elles doivent être des entiers (entier) ou des chaînes de caractères (chaîne de caractères).
- Une alerte sera émise si une valeur est d'un type qui ne convient pas et la paire en question ne sera pas incluse dans le résultat.
- Si une valeur n'est pas unique, seule la dernière clé sera utilisée comme valeur, et toutes les autres seront perdues.

Liste de paramètres

- **array** : Un tableau de paire clés/valeurs à inverser.

Valeurs de retour Retourne un tableau inversé en cas de succès, NULL si une erreur survient.

Exemple

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

print_r($flipped);
?>
```

array_pad()

Complète un tableau avec une valeur jusqu'à la longueur spécifiée

Syntaxe :

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

- `array_pad()` retourne une copie du tableau array complétée jusqu'à la taille de size avec la valeur value.
- Si size est positif, alors le tableau est complété à droite, s'il est négatif, il est complété à gauche.
- Si la valeur absolue de size est plus petite que la taille du tableau array, alors le tableau n'est pas complété.
- Il est possible d'ajouter au maximum 1048576 éléments d'un seul coup.

Liste de paramètres

- **array** : Tableau initial de valeurs à compléter ;
- **size** : Nouvelle taille du tableau.
- **value** : Valeur à insérer si l'argument array est plus petit que l'argument size.

Valeurs de retour

Retourne une copie du tableau array complétée jusqu'à la taille de size avec la valeur value.

Exemple

```php
<?php
$input = array(12, 10, 9);

$result = array_pad($input, 5, 0);
// Le resultat est : array(12, 10, 9, 0, 0)

$result = array_pad($input, -7, -1);
// Le resultat est : array(-1, -1, -1, -1, 12, 10, 9)

$result = array_pad($input, 2, "noop");
// pas complete
?>
```

array_pop()

Dépile un élément de la fin d'un tableau

Syntaxe :

```
mixed array_pop ( array &$array )
```

`array_pop()` dépile et retourne le dernier élément du tableau array, le raccourcissant d'un élément.

Liste de paramètres

- **array** : Le tableau duquel on récupère la valeur.

Valeurs de retour

Retourne la dernière valeur du tableau `array`. Si `array` est vide (ou n'est pas un tableau), `NULL` sera retourné.

Exemple

```php
<?php
$stack = array("orange", "banana", "apple", "raspberry");
$fruit = array_pop($stack);
print_r($stack);
?>
```

array_push()

Empile un ou plusieurs éléments à la fin d'un tableau

Syntaxe :

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

`array_push()` considère array comme une pile, et empile les variables `var`, ... à la fin de `array`. La longueur du tableau `array` augmente d'autant. Cela a le même effet que :

```
<?php
$array[] = $var;
?>
```

répété pour chaque valeur.

Liste de paramètres

- **array** Le tableau d'entrée ;
- **value1** La première valeur à insérer à la fin du tableau array.

Valeurs de retour Retourne le nouveau nombre d'éléments dans le tableau.

Exemple

```php
<?php
$stack = array("orange", "banana");
array_push($stack, "apple", "raspberry");
print_r($stack);
?>
```

array_shift()

Dépile un élément au début d'un tableau

Syntaxe :

```
mixed array_shift ( array &$array )
```

`array_shift()` extrait la première valeur d'un tableau et la retourne, en raccourcissant le tableau d'un élément, et en déplaçant tous les éléments vers le bas. Toutes les clés numériques seront modifiées pour commencer à zéro.

Cette fonction remet le pointeur au début du tableau d'entrée.

Liste de paramètres

- **array** Le tableau d'entrée.

Valeurs de retour Retourne la valeur dépilée, ou `NULL` si le tableau est vide ou si la valeur d'entrée n'est pas un tableau.

Exemple

```php
<?php
$stack = array("orange", "banana", "apple", "raspberry");
$fruit = array_shift($stack);
print_r($stack);
?>
```

array_unshift()

Empile un ou plusieurs éléments au début d'un tableau

Syntaxe :

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

- `array_unshift()` ajoute les éléments `value1`, `...`, passés en argument au début du tableau `array`.
- Les éléments sont ajoutés comme un tout. Ils restent dans le même ordre.
- Les clés numériques sont modifiées pour commencer à zéro.
- Les clés littérales ne sont pas touchées.

Liste de paramètres

- **array** : Le tableau d'entrée.
- **value1** : Première valeur à empiler.

Valeurs de retour

- Retourne le nouveau nombre d'éléments du tableau `array`.

compact()

Crée un tableau à partir de variables et de leur valeur

Syntaxe :

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

Pour chacun des arguments `varname`, `...`, `compact()` recherche une variable avec un même nom dans la table courante des symboles, et l'ajoute dans le tableau, de manière à avoir la relation `nom => 'valeur de variable'`. Toute chaîne non reconnue dans la table des symboles sera tout simplement ignorée.

Liste de paramètres

- **varname1** : `compact()` accepte différents paramètres `varname`. Les paramètres peuvent être des variables contenant des chaînes, ou un tableau de chaînes, qui peut contenir d'autres tableaux de noms de variables, que `compact()` traitera récursivement.

Valeurs de retour

- Retourne le tableau de sortie contenant toutes les variables ajoutées.

Exemple

```php
<?php
$city  = "San Francisco";
$state = "CA";
$event = "SIGGRAPH";

$location_vars = array("city", "state");

$result = compact("event", "nothing_here", $location_vars);
print_r($result);
?>
```

range()

Crée un tableau contenant un intervalle d'éléments

Syntaxe :

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

Liste de paramètres

- **start** : Première valeur de la séquence.
- **end** : La séquence se termine lorsque la valeur **end** est atteinte.
- **step** : Si une valeur est donnée au paramètre **step**, il sera utilisé comme valeur incrémentale entre les éléments de la séquence. **step** doit être exprimé comme un nombre entier positif. S'il n'est pas spécifié, **step** vaut par défaut 1.

Valeurs de retour

- Retourne un tableau d'éléments depuis **start** jusqu'à **end**, inclusif.

Exemple

```php
<?php
// array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)
foreach (range(0, 12) as $number) {
    echo $number;
}

// La parametre de pas (step)
// array(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
foreach (range(0, 100, 10) as $number) {
    echo $number;
}

// Utilisation des caracteres
// array('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i');
foreach (range('a', 'i') as $letter) {
    echo $letter;
}
// array('c', 'b', 'a');
foreach (range('c', 'a') as $letter) {
    echo $letter;
}
?>
```

Les valeurs des caractères de séquence sont limitées à une longueur d'un. Si une longueur supérieure à un est entrée, seul le premier caractère est utilisé.

array_chunks()

Sépare un tableau en tableaux de taille inférieure

Syntaxe :

```
array array_chunk ( array $array , int $size [, bool $preserve_keys = FALSE ] )
```

Sépare le tableau array en plusieurs tableaux comptant `size` éléments. Il est aussi possible que le dernier tableau contienne moins de valeurs. **Liste de paramètres**

- **array** : Le tableau à traiter
- **size** : La taille de chaque tableau
- **preserve_keys** : Lorsque défini à `TRUE`, les clés seront préservées. Par défaut, vaut `FALSE` ce qui réindexera le tableau résultant numériquement

Valeurs de retour

- Retourne un tableau multidimensionnel indexé numériquement, commençant à zéro, dont chaque dimension contient size éléments

Exemples

```
Exemple #1 Exemple avec array_chunk()
<?php
$input_array = array('a', 'b', 'c', 'd', 'e');
print_r(array_chunk($input_array, 2));
print_r(array_chunk($input_array, 2, true));
?>
```

array_slice()

Extrait une portion de tableau

Syntaxe :

```
array array_slice ( array $array , int $offset [, int $length = NULL [, bool $preserve_keys = FALSE ]] )
```

`array_slice()` retourne une série d'éléments du tableau `array` commençant à l'offset `offset` et représentant `length` éléments.

Liste de paramètres

- **array** : Le tableau d'entrée ;
- **offset** : Si offset est non-négatif, la série commencera à cet offset dans le tableau array. Si offset est négatif, cette série commencera à l'offset offset, mais en commençant à la fin du tableau array ;
- **length** : Si length est fourni et positif, alors la série retournée aura autant d'éléments. Si le tableau est moins long que length, alors seuls les éléments de tableau disponibles seront présents. Si length est fourni et négatif, alors la série contiendra les éléments depuis l'offset offset jusqu'à length éléments en partant de la fin. Si length est omis, la séquence lira tous les éléments du tableau, depuis l'offset précisé jusqu'à la fin du tableau ;
- **preserve_keys** : Notez que, par défaut, la fonction `array_slice()` va réordonner et réinitialiser les indices numériques du tableau. Vous pouvez modifier ce comportement en définissant le paramètre `preserve_keys` à `TRUE`.

Valeurs de retour Retourne la portion du tableau. Si l'offset est plus grand que la taille du tableau, alors la fonction retourne un tableau vide

Exemple

```php
<?php
$input = array("a", "b", "c", "d", "e");

$output = array_slice($input, 2);          // retourne "c", "d", et "e"
$output = array_slice($input, -2, 1);      // retourne "d"
$output = array_slice($input, 0, 3);       // retourne "a", "b", et "c"

// notez les cles d'index differentes
print_r(array_slice($input, 2, -1));
print_r(array_slice($input, 2, -1, true));
?>
```

array_splice()

Efface et remplace une portion de tableau

Syntaxe :

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

`array_splice()` supprime les éléments désignés par `offset` et `length` du tableau `input` et les remplace par les éléments du tableau replacement, si ce dernier est présent.
Notez que les clés numériques de `input` ne sont pas préservées.

S

i `replacement` n'est pas un tableau, il en deviendra un par transtypage (i.e. `(array) $replacement`). Cela peut conduire en un résultat non prévu lors de l'utilisation d'un objet ou `NULL` comme paramètre replacement.

Liste de paramètres

- **input** : Le tableau d'entrée ;
- **offset** : Si `offset` est positif, la série commencera à cet `offset` dans le tableau `input`. Si `offset` est négatif, cette série commencera à l'offset `offset`, mais en commençant à la fin du tableau `input` ;
- **length** : Si `length` est donné et positif, alors la série aura autant d'éléments. Si `length` est donné et négatif, les éléments seront pris dans l'ordre inverse. Si `length` est omis, la séquence supprimera tous les éléments du tableau, depuis l'offset `offset` jusqu'à la fin du tableau. Si le paramètre `length` est spécifié et vaut zéro, aucun élément ne sera supprimé.
  Conseil : pour supprimer tous les éléments du tableau depuis `offset` jusqu'à la fin, même si un tableau de remplacement replacement est spécifié, utilisez `count($input)` à la place de `length` ;
- **replacement** : Si replacement est précisé, alors les éléments supprimés sont remplacés par les éléments de ce tableau.
  Si l'offset et `length` sont tels que la taille du tableau ne change pas, alors les éléments du tableau de remplacement replacement sont insérés à partir de l'offset `offset`. Notez que les clés numériques de input ne sont pas préservées.
  Si le tableau de remplacement replacement ne contient qu'un seul élément, il n'est pas obligatoire de forcer le type en tableau avec `array()`, à moins que cette variable ne soit elle-même un tableau, un objet ou `NULL`.

Valeurs de retour Retourne le tableau contenant les éléments supprimés.

Exemple

```php
<?php
$input = array("red", "green", "blue", "yellow");
array_splice($input, 2);
// $input is now array("red", "green")

$input = array("red", "green", "blue", "yellow");
array_splice($input, 1, -1);
// $input is now array("red", "yellow")

$input = array("red", "green", "blue", "yellow");
array_splice($input, 1, count($input), "orange");
// $input is now array("red", "orange")

$input = array("red", "green", "blue", "yellow");
array_splice($input, -1, 1, array("black", "maroon"));
// $input is now array("red", "green",
//          "blue", "black", "maroon")

$input = array("red", "green", "blue", "yellow");
array_splice($input, 3, 0, "purple");
// $input is now array("red", "green",
//          "blue", "purple", "yellow");
?>
```

array_multisort()

Trie les tableaux multidimensionnels

Syntaxe :

```
bool array_multisort ( array &$array1 [, mixed $array1_sort_order = SORT_ASC [, mixed $array1_sort_flags =
```

`array_multisort()` sert à trier simultanément plusieurs tableaux, ou bien à trier un tableau multidimensionnel, suivant l'une ou l'autre de ses dimensions.
Les clés associatives (chaîne de caractères) seront maintenues, mais les clés numériques seront réindexées.

Si deux membres se comparent comme égaux, leur ordre relatif dans le tableau trié n'est pas défini.

Liste de paramètres

- **array1** : Un tableau à trier.
- **array1_sort_order** : L'ordre utilisé pour trier le précédent argument array. Soit la constante `SORT_ASC` pour trier de façon croissante, soit la constante `SORT_DESC` pour trier de façon décroissante ;
  Cet argument peut être associé avec le paramètre `array1_sort_flags` ou simplement omis, auquel cas, la constante `SORT_ASC` sera utilisée.
- **array1_sort_flags** : Options de tri du précédent argument array :

  Type d'options de tri :
  - `SORT_REGULAR` - compare les éléments normalement (pas de changement de type)
  - `SORT_NUMERIC` - compare les éléments numériquement
  - `SORT_STRING` - compare les éléments sous forme de chaînes de caractères
  - `SORT_LOCALE_STRING` - compare les éléments sous forme de chaînes de caractères, en se basant sur la locale courante. La fonction utilise les locales, et elles peuvent être modifiées en utilisant la fonction `setlocale()`
  - `SORT_NATURAL` - compare les éléments sous forme de chaînes de caractères, en utilisant le "tri naturel", comme le fait la fonction `natsort()`
  - `SORT_FLAG_CASE` - peut être combiné (avec le mot clé OR) avec `SORT_STRING` ou `SORT_NATURAL` pour trier les chaînes sans tenir compte de la casse

  Cet argument peut être associé avec le paramètre `array1_sort_order` ou simplement omis, auquel cas, la constante `SORT_REGULAR` sera utilisée.
- ... : Plus d'arguments, optionnellement suivis par des façons de trier et des drapeaux. Seuls les éléments équivalents dans les tableaux précédents sont comparés. En d'autres termes, le tri est lexicographique.

Valeurs de retour

- Cette fonction retourne `TRUE` en cas de succès ou `FALSE` si une erreur survient.

Exemple

```php
<?php
$ar1 = array(10, 100, 100, 0);
$ar2 = array(1, 3, 2, 4);
array_multisort($ar1, $ar2);

var_dump($ar1);
var_dump($ar2);
?>
```

Dans cet exemple, après le tri, le premier tableau contient 0, 10, 100, 100. Le deuxième tableau contient 4, 1, 2, 3. Les entrées du second tableau correspondant aux valeurs jumelles du premier tableau (100 et 100), sont aussi triées.

array_reverse()

Inverse l'ordre des éléments d'un tableau

Syntaxe :

```
array array_reverse ( array $array [, bool $preserve_keys = FALSE ] )
```

`array_reverse()` retourne un nouveau tableau qui contient les mêmes éléments que `array`, mais dans l'ordre inverse. **Liste de paramètres**

- **array** : Le tableau d'entrée ;
- **preserve_keys** : Si défini à `TRUE`, les clés numériques seront préservées. Les clés non-numériques ne seront pas affectées par cette configuration, et seront toujours préservées.

Valeurs de retour Retourne le tableau dans l'ordre inverse.

Exemple

```php
<?php
$input    = array("php", 4.0, array("green", "red"));
$reversed = array_reverse($input);
$preserved = array_reverse($input, true);

print_r($input);
print_r($reversed);
print_r($preserved);
?>
```

arsort()

Trie un tableau en ordre inverse et conserve l'association des index

Syntaxe :

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

`arsort()` trie le tableau `array` de telle manière que la corrélation entre les index et les valeurs soit conservée.

Si deux membres se comparent comme égaux, leur ordre relatif dans le tableau trié n'est pas défini.

Liste de paramètres

- **array** : Le tableau d'entrée.
- `sort_flags` : Modifie le comportement de la fonction en utilisant le paramètre optionnel `sort_flags`.

Valeurs de retour

- `TRUE` en cas de succès ou `FALSE` si une erreur survient.

Exemples

```php
<?php
$fruits = array("d" => "lemon", "a" => "orange",
        "b" => "banana", "c" => "apple");
arsort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val\n";
}
?>
```

Trie un tableau et conserve l'association des index

Syntaxe :

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

`asort()` trie le tableau array de telle manière que la corrélation entre les index et les valeurs soit conservée. L'usage principal est lors de tri de tableaux associatifs où l'ordre des éléments est important.

Si deux membres se comparent comme égaux, leur ordre relatif dans le tableau trié n'est pas défini.

Liste de paramètres

- **array** : Le tableau d'entrée.
- **sort_flags** : Modifie le comportement de la fonction. Pour plus de détails, voyez le manuel pour la fonction `sort()`.

Valeurs de retour

- `TRUE` en cas de succès ou `FALSE` si une erreur survient.

Exemples

```php
<?php
$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" => "apple");
asort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val\n";
}
?>
```

krsort()

Trie un tableau en sens inverse et suivant les clés

Syntaxe :

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

`krsort()` trie le tableau array en ordre inverse et suivant les clés, en maintenant la correspondance entre les clés et les valeurs. Cette fonction est pratique pour les tableaux associatifs. **Liste de paramètres**

- **array** : Le tableau d'entrée.
- **sort_flags** : Vous pouvez modifier le comportement de cette fonction en utilisant le paramètre optionnel `sort_flags`. Pour plus de détails, voyez le manuel pour la fonction `sort()`.

Valeurs de retour

- Cette fonction retourne `TRUE` en cas de succès ou `FALSE` si une erreur survient.

Exemples

```php
<?php
$fruits = array("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
krsort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val\n";
}
?>
```

ksort()

Trie un tableau suivant les clés

Syntaxe :

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

Trie le tableau array suivant les clés, en maintenant la correspondance entre les clés et les valeurs. Cette fonction est pratique pour les tableaux associatifs. **Liste de paramètres**

- **array** : Le tableau d'entrée.
- **sort_flags** : Vous pouvez modifier le comportement de cette fonction en utilisant le paramètre optionnel `sort_flags`. Pour plus de détails, voyez le manuel pour la fonction `sort()`.

Valeurs de retour Cette fonction retourne `TRUE` en cas de succès ou `FALSE` si une erreur survient.

Exemples

```php
<?php
$fruits = array("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
ksort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val\n";
}
?>
```

natecasesort()

Trie un tableau dans l'ordre naturel, insensible à la casse

Syntaxe :

```
bool natcasesort ( array &$array )
```

`natcasesort()` est la version insensible à la casse de `natsort()`. Cette fonction implémente un algorithme de tri qui traite les chaînes alphanumériques du tableau `array` comme un être humain tout en conservant la relation clé/valeur.

Si deux membres se comparent comme égaux, leur ordre relatif dans le tableau trié n'est pas défini.

Liste de paramètres

- **array** : Le tableau d'entrée.

Valeurs de retour `TRUE` en cas de succès ou `FALSE` si une erreur survient.

Exemples

```php
<?php
$array1 = $array2 = array('IMG0.png', 'img12.png', 'img10.png', 'img2.png',
                          'img1.png', 'IMG3.png');

sort($array1);
echo "Tri standard\n";
print_r($array1);

natcasesort($array2);
echo "\nTri en ordre naturel (insensible a la casse)\n";
print_r($array2);
?>
```

natsort()

Trie un tableau avec l'algorithme à "ordre naturel"

Syntaxe :

```
bool natsort ( array &$array )
```

`natsort()` implémente un algorithme de tri qui traite les chaînes alphanumériques du tableau `array` comme un être humain tout en conservant la relation clé/valeur.

Liste de paramètres

- **array** : Le tableau d'entrée.

Valeurs de retour

- `TRUE` en cas de succès ou `FALSE` si une erreur survient.

Exemple

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

asort($array1);
echo "Standard sorting\n";
print_r($array1);

natsort($array2);
echo "\nNatural order sorting\n";
print_r($array2);
?>
```

shuffle()

Mélange les éléments d'un tableau

Syntaxe :

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

Mélange les éléments du tableau `array`. Cette fonction utilise un pseudo générateur de nombre aléatoire qu'il n'est pas conseillé d'utiliser pour de la cryptographie.

Si deux membres se comparent comme égaux, leur ordre relatif dans le tableau trié n'est pas défini.

Liste de paramètres

- **array** : Le tableau.

Valeurs de retour

- TRUE en cas de succès ou FALSE si une erreur survient.

Exemple

```php
<?php
$numbers = range(1, 20);
shuffle($numbers);
foreach ($numbers as $number) {
    echo "$number ";
}
?>
```

Cette fonction assigne de nouvelles clés pour les éléments du paramètre array. Elle effacera toutes les clés existantes que vous aviez pu assigner, plutôt que de les trier.

sort()

Trie un tableau

Syntaxe :

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

Cette fonction trie le tableau `array`. Les éléments seront triés du plus petit au plus grand.

Si deux membres se comparent comme égaux, leur ordre relatif dans le tableau trié n'est pas défini.

Liste de paramètres

- **array** : Le tableau d'entrée.
- **sort_flags** : Le paramètre optionnel **sort_flags** peut être utilisé pour modifier le comportement de tri en utilisant ces valeurs :
  Constantes de type de tri :
  - **SORT_REGULAR** : compare les éléments normalement (ne modifie pas les types)
  - **SORT_NUMERIC** : compare les éléments numériquement
  - **SORT_STRING** : compare les éléments comme des chaînes de caractères
  - **SORT_LOCALE_STRING** : compare les éléments en utilisant la configuration locale. La locale courante est utilisée, elle peut être changée au moyen de `setlocale()`.
  - **SORT_NATURAL** - compare les éléments comme des chaînes de caractères en utilisant l'ordre naturel comme le fait la fonction `natsort()`.
  - **SORT_FLAG_CASE** - peut être combiné (grâce à l'opérateur OR) avec **SORT_STRING** ou **SORT_NATURAL** pour trier les chaînes sans tenir compte de la casse.

Valeurs de retour

- **TRUE** en cas de succès ou **FALSE** si une erreur survient.

uasort()

Trie un tableau en utilisant une fonction de rappel

Syntaxe :

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

Trie le tableau `array` en conservant la correspondance entre les index et leurs valeurs. `uasort()` sert essentiellement lors de tri de tableaux associatifs où l'ordre des éléments est significatif. La fonction de comparaison utilisée `cmp_function` est définie par l'utilisateur.
Utilisé habituellement lors du tri de tableaux associatifs où l'ordre actuel des éléments est significatif.

Si deux membres se comparent comme égaux, leur ordre relatif dans le tableau trié n'est pas défini.

Liste de paramètres

- **array** : Le tableau d'entrée.
- **value_compare_func** : Voyez les fonctions `usort()` et `uksort()` pour des exemples de tri avec utilisation de fonction personnalisée.

Valeurs de retour

- Cette fonction retourne **TRUE** en cas de succès ou **FALSE** si une erreur survient.

Exemples

```php
<?php
// Fonction de comparaison
function cmp($a, $b) {
    if ($a == $b) {
        return 0;
    }
    return ($a < $b) ? -1 : 1;
}

// Tableau a trier
$array = array('a' => 4, 'b' => 8, 'c' => -1,
        'd' => -9, 'e' => 2, 'f' => 5, 'g' => 3, 'h' => -4);
print_r($array);

// Trie et affiche le tableau resultant
uasort($array, 'cmp');
print_r($array);
?>
```

usort()

Trie un tableau en utilisant une fonction de comparaison

Syntaxe :

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

`usort()` va trier le tableau array avec ses valeurs, en utilisant une fonction définie par l'utilisateur. Si un tableau doit être trié avec un critère complexe, il est préférable d'utiliser cette fonction.

Si deux membres se comparent comme égaux, leur ordre relatif dans le tableau trié n'est pas défini.
Cette fonction assigne de nouvelles clés pour les éléments du paramètre array. Elle effacera toutes les clés existantes que vous aviez pu assigner, plutôt que de les trier.

Liste de paramètres

- **array** : Le tableau d'entrée.
- **value_compare_func** : La fonction de comparaison doit retourner un entier inférieur à, égal à, ou supérieur à 0 si le premier argument est considéré comme, respectivement, inférieur à, égal à, ou supérieur au second.

```
int callback ( mixed $a, mixed $b )
```

Retourner des valeurs non entières depuis la fonction de comparaison, comme des valeurs de type nombre décimal fera que l'intervalle sera transtypé en entierl. Des valeurs comme 0.99 et 0.1 seront toutes les deux transformées en la valeur 0, et leur comparaison sera égale.

Valeurs de retour

TRUE en cas de succès ou **FALSE** si une erreur survient.

Exemples

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

$a = array(3, 2, 5, 6, 1);

usort($a, "cmp");

foreach ($a as $key => $value) {
    echo "$key: $value\n";
}
?>
```