PHP
Fonctions associées aux tableaux

Rappels

- Tableaux associatifs
  - fait correspondre des valeurs à des clés
  - clés : entier ou chaîne de caractères
  - valeurs : ce que l'on souhaite, pas forcément homogène
- Syntaxe
  - `$t = array(1, 'a'); $t = array('b'=>1, 2=>'a');`
  - `$t[] = 'b'; $t['z'] = 's';`
- Parcours
  - `foreach ($t as $val) { ... }`
  - `foreach ($t as $cle => $val) { ... }`

Tableaux vus comme des piles/files

- Empiler

int **array_push** (array &*tab*, mixed *var* [, mixed ...])

considère *tab* comme une pile, et empile les variables *var*, ... à la fin de *tab*

retourne le nouveau nombre d'éléments du tableau.

- Dépiler

mixed **array_pop** (array &*tab*)

dépile et retourne le dernier élément du tableau *tab*

Tableaux vus comme des piles/files

```
$stack = arr
array_push (
print_r($sta
```

```
Array
(
    [0] => orange
    [1] => banane
    [2] => pomme
)
```

```
$stack = ar
```

```
Array
(
    [0] => orange
    [1] => banane
    [2] => pomme
    [3] => framboise
)
```

```
$fruit = ar
print_r($st
echo $fruit
```

Tableaux vus comme des piles/files

- Enfiler

int **array_unshift** (array &*tab*, mixed *var* [, mixed ...])

considère *tab* comme une file, et enfile les variables *var*, ... au début de *tab*

retourne le nouveau nombre d'éléments du tableau.

- Défiler

mixed **array_shift** (array &*tab*)

défile et retourne le premier élément du tableau *tab*

Tableaux vus comme des piles/files

```
$queue = ar
array_unshi

print_r($qu

$stack = ar

$fruit = ar
print_r($st
echo $fruit.
```

```
Array
(
    [0] => banane
    [1] => pomme
    [2] => framboise
)
```

```
Array
(
    [0] => pomme
    [1] => framboise
    [2] => orange
    [3] => banane
)
```

Tris sur les tableaux

```
$fruits = array("lemon", "orange",
                "banana", "apple");
```

- bool sort (a … gs])

```
sort($fruits
foreach ($fr                ) {
  echo "frui                l."\n"; }
```

```
fruits[0] = orange
fruits[1] = lemon
fruits[2] = banana
fruits[3] = apple
```

- bool rsort (… gs])

```
rsort($fruit
foreach ($fr                ) {
  echo "frui                l."\n"; }
```

```
fruits[0] = apple
fruits[1] = banana
fruits[2] = lemon
fruits[3] = orange
```

Tris sur les tableaux

```
$fruits = array("lemon", "orange",
                "banana", "apple");
```

- bool asort (... [, flags])

```
asort($fruit
foreach ($fr                ) {
  echo "frui                l."\n"; }
```

```
fruits[1] = orange
fruits[0] = lemon
fruits[2] = banana
fruits[3] = apple
```

- bool arsort (... [, flags])

```
arsort($frui
foreach ($fr                ) {
  echo "frui                l."\n"; }
```

```
fruits[3] = apple
fruits[2] = banana
fruits[0] = lemon
fruits[1] = orange
```

Tris sur les tableaux

```
$fruits= array("d"=>"lemon", "a"=>"orange",
               "b"=>"banana", "c"=>"apple");
```

- bool **ksort** (*array &$array [, int $sort_flags]*)

```
ksort($fruit
foreach ($fr                    ) {
  echo "frui                    l."\n"; }
```

```
fruits[d] = lemon
fruits[c] = apple
fruits[b] = banana
fruits[a] = orange
```

- bool **krsort** (*array &$array [, int $sort_flags]*)

```
krsort($frui
foreach ($fr                    ) {
  echo "frui                    l."\n"; }
```

```
fruits[a] = orange
fruits[b] = banana
fruits[c] = apple
fruits[d] = lemon
```

Mélanges aléatoires sur les tableaux

```php
$numbers = range(1, 10);
```

- bool **shuffle** (array &*tab*)

```php
shuffle($numbers);
foreach ($numbers as $number) {
  echo "$number ";
}
```

```
9 7 6 5 2 4 1 3 10 8
```

Fonctions utiles sur les tableaux

- int **count** (mixed *var* [, int *mode*])
- bool **array_key_exists** (mixed *key*, array *search*)
- array **array_keys** (array *input* [, mixed *search_value* [, bool *strict*]])
- array **array_values** (array *input*)
- mixed **array_search** (mixed *needle*, array *haystack* [, bool *strict*])
- bool **in_array** (mixed *needle*, array *haystack* [, bool *strict*])

Implosion / explosion

- array **explode** (string *delimiter*, string *string* [, int *limit*])

```
$pizza =
"piece1 piece2 piece                ce4 piece5 piece6";
$pieces = explode("                 izza
print_r($pieces);

$data = "foo:*:1023:1000::/
list($user, $pass, $uid, $g
     $home, $shell) = explo
echo $user."\n"; echo $pass
```

```
foo
*
1023
```

```
Array (
    [0] => piece1
    [1] => piece2
    [2] => piece3
    [3] => piece4
    [4] => piece5
    [5] => piece6
)
```

Implosion / explosion

- string **implode** (string *glue*, array *pieces*)

```
$array =
      array('lastname', 'email', 'phone');
$comma_separated = implode(",", $array);
echo $comma_separated."\n";
```

```
lastname,email,phone
```

Récupération des clés / valeurs

- array **array_keys** (array *input*

```php
$array =
      array(0 => 100, "color" => "red");
print_r(array_keys($array));
```

```
Array (
   [0] => 0
   [1] => color
)
```

- array **array_values** (array *input*)

```php
 $array =
     array(0 => 100, "color" => "red");
print_r(array_values($array));
```

```
Array (
   [0] => 100
   [1] => red
)
```

Toujours plus de fonctionnalités

Consulter la documentation officielle

http://fr.php.net/manual/fr/book.array.php