Les Tableaux (Array) en PHP

Les tableaux en PHP sont des structures de données qui permettent de stocker plusieurs valeurs dans une seule variable. Ils peuvent être utilisés pour gérer des collections de données comme des listes, des ensembles ou des cartes (associatives).

Voici une leçon complète sur les tableaux en PHP :

1. Création d'un tableau

En PHP, un tableau peut être créé de deux manières : avec l'ancienne syntaxe (`array()`) ou avec la nouvelle syntaxe plus simple (`[]`).

```php
<?php

// membuat array kosong
$buah = array();
$hobi = [];

// membuat array sekaligus mengisinya
$minuman = array("Kopi", "Teh", "Jus Jeruk");
$makanan = ["Nasi Goreng", "Soto", "Bubur"];

// membuat array dengan mengisi indeks tertentu
$anggota[1] = "Dian";
$anggota[2] = "Muhar";
$anggota[0] = "Petani Kode";
```

Exemple 1 : Tableau indexé

Les tableaux indexés sont des tableaux où les clés sont des indices numériques (par défaut, commençant à 0).

```php
// Ancienne syntaxe
$fruits = array("Pomme", "Banane", "Orange");

// Nouvelle syntaxe
$fruits = ["Pomme", "Banane", "Orange"];
```

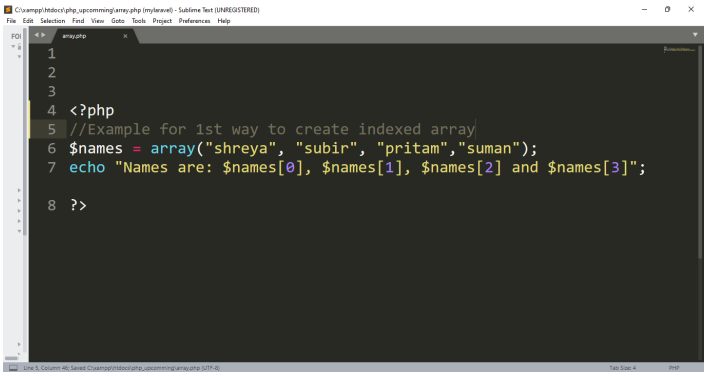
Exemple 2 : Tableau associatif

Les tableaux associatifs utilisent des clés personnalisées pour indexer les valeurs, au lieu de simples indices numériques.

```php
$personne = [
    "nom" => "Jean",
    "prénom" => "Dupont",
    "âge" => 30
];
```

2. Accéder aux éléments d'un tableau

Pour accéder aux éléments d'un tableau, vous utilisez les crochets `[]` et indiquez l'indice ou la clé.



Exemple : Accéder à un tableau indexé

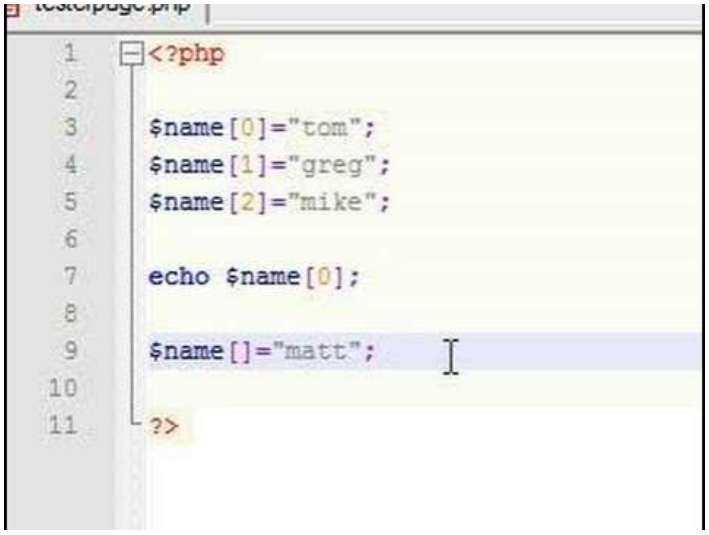
```php
echo $fruits[0];  // Affiche "Pomme"
```

Exemple : Accéder à un tableau associatif

```php
echo $personne["nom"];  // Affiche "Jean"
```

3. Ajouter des éléments dans un tableau

Vous pouvez ajouter des éléments à un tableau indexé ou associatif de plusieurs manières :



Exemple : Ajouter un élément dans un tableau indexé

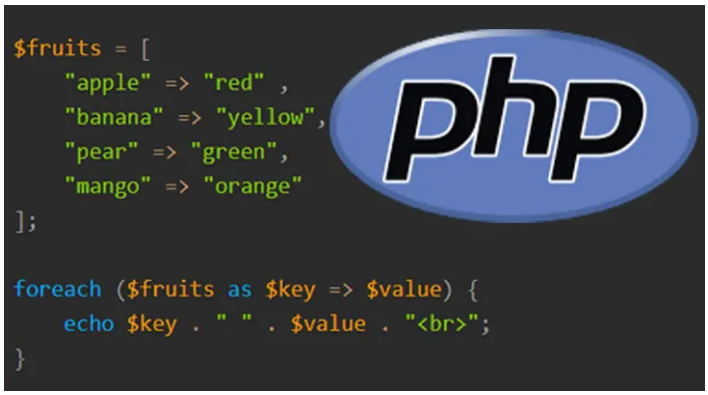
```php
// Ajoute "Mangue" à la fin du tableau
$fruits[] = "Mangue";
```

Exemple : Ajouter un élément dans un tableau associatif

```php
// Ajoute la clé "profession" avec la valeur "Développeur"
$personne["profession"] = "Développeur";
```

4. Parcourir un tableau

PHP fournit plusieurs boucles pour parcourir les tableaux, comme `foreach`, `for`, ou `while`.



Exemple : Utilisation de `foreach` pour parcourir un tableau indexé

```php
$fruits = ["Pomme", "Banane", "Orange"];
foreach ($fruits as $fruit) {
    echo $fruit . "<br>";  // Affiche chaque fruit
}
```

Exemple : Utilisation de `foreach` pour parcourir un tableau associatif

```php
$personne = [
    "nom" => "Jean",
    "prénom" => "Dupont",
    "âge" => 30
];
foreach ($personne as $clé => $valeur) {
    // Affiche chaque clé et valeur
    echo $clé . " : " . $valeur . "<br>";
}
```

5. Fonctions utiles pour les tableaux

```php
// Returns a new array, appending input arrays with array unpacking
$arr = [1,2,3,4,5];
$output = [...$arr, ...[10,11,12]];
print_result($output);

// Returns a new array, appending input arrays with array_merge
$arr = [1,2,3,4,5];
$output = array_merge($arr, [10,11,12]);
print_result($output);
// ARRAY   : 1,2,3,4,5,10,11,12

// Joins all elements into a string with a separator
$arr = [1,2,3,4,5];
$output = implode(",", $arr);
print_result($output);
// STRING  : 1,2,3,4,5

// Returns a section of an array from $start to $end
$start = 0;
$end = 3;
$arr = [1,2,3,4,5];
$output = array_slice($arr, $start, $end);
print_result($output);
// ARRAY   : 1,2,3

// Returns the index of the first occurrence of element in array
$arr = [1,2,3,4,5, 1,2,3,4,5];
$element = 4;
$output = array_search($element, $arr);
print_result($output);
// INTEGER : 3

// Returns the index of the last occurrence of element in array
$arr = [1,2,3,4,5, 1,2,3,4,5];
$element = 4;
$output = array_search(4, array_reverse($arr, true));
print_result($output);
// INTEGER : 8

// Calls function for each element of array (the array is unchanged)
$arr = [1,2,3,4,5];
$output = array_walk($arr, function ($element) {
    $double = $element * 2;
});
print_result($output);
print_result($arr);
// BOOLEAN : true
// ARRAY   : 1,2,3,4,5

// Returns true if all elements return true
$arr = [1,2,3,4,5];
$result = array_reduce($arr, fn($result, $element) => $result && ($element > 0), true);
print_result($result);
// BOOLEAN : true

// Returns true if at least one element return true
$arr = [1,2,3,4,5];
$result = array_reduce($arr, fn($result, $element) => $result || ($element >= 5), false);
print_result($result);
// BOOLEAN : true


// Returns new array, with element that matches the fn() function
$arr = [1,2,3,4,5];
$result = array_filter($arr, fn($element) => $element > 3);
print_result($result);
// ARRAY   : 4,5

// Returns new array with the results of running fn on every element
$arr = [1,2,3,4,5];
$result = array_map(fn($element) => $element + 3, $arr);
print_result($result);
// ARRAY   : 4,5,6,7,8

// Returns a flatten array of a nested array
$nestedArray = [1, [2,3], [4,5,6], 7];
$result = array_reduce($nestedArray, fn($result, $element) => array_merge($result, is_array($element) ?
[...$element] : [$element]), []);
print_result($result);
// ARRAY   : 1,2,3,4,5,6,7

// Changes all elements in range to a the specified value
$arr = [1,2,3,4,5,6,7,8,9];
$start = 2;
$count = 6;
$result = array_replace($arr, array_fill($start, $count, 88));
print_result($result);
// ARRAY   : 1,2,88,88,88,88,88,88,9

// Returns a single value which is the function's accumulated result L2R
$arr = [1,2,3,4,5,6,7,8,9];
$result = array_reduce($arr, fn($result, $element) => $result + $element, 0);
print_result($result);
// INTEGER : 45

// Returns a single value which is the function's accumulated result R2L
$arr = [1,2,3,4,5,6,7,8,9];
$result = array_reduce(array_reverse($arr), fn($result, $element) => $result + $element, 0);
print_result($result);
// INTEGER : 45

// Removes and returns last element from arr
$arr = [1,2,3,4,5,6,7,8,9];
$result = array_pop($arr);
print_result($result);
print_result($arr);
// INTEGER : 9
// ARRAY   : 1,2,3,4,5,6,7,8

// Add element to start of arr and return new length
$arr = [1,2,3,4,5,6,7,8,9];
$newElement = 99;
$result = array_unshift($arr, $newElement);
print_result($result);
print_result($arr);
// INTEGER : 10
// ARRAY   : 99,1,2,3,4,5,6,7,8,9

// Adds element to the end of arr and returns new length
$arr = [1,2,3,4,5,6,7,8,9];
$newElement = 99;
$result = array_push($arr, 99);
print_result($result);
print_result($arr);
// INTEGER : 10
// ARRAY   : 1,2,3,4,5,6,7,8,9,99

// Reverse order of arr
$arr = [1,2,3,4,5,6,7,8,9];
$result = array_reverse($arr);
print_result($result);
$result = sort($arr);
print_result($arr);
// ARRAY   : 1,2,3,4,5,6,7,8,9

// Changes content of arr removing, replacing and adding elements
// at position $start, it removes $count elements and insert
// it returns the elements removed, and changes the $arr
$arr = [1,2,3,4,5,6,7,8,9];
$start = 4;
$count = 5;
$newElements = [70,71];
$result = array_splice($arr, $start, $count, $newElements);
print_result($result); // removed elements
print_result($arr); // changed array
// ARRAY   : 5,6,7
// ARRAY   : 1,2,3,4,70,71,8,9

// Returns a string representing arr its elements (same as arr.join(','))
$arr = [1,2,3,4,5,6,7,8,9];
$result = implode("", $arr);
print_result($result);
// STRING  : 123456789


// Returns length of arr
$arr = [1,2,3,4,5,6,7,8,9];
$result = count($arr);
print_result($result);
// INTEGER : 9

// Returns true if $arr is an array
$arr = [1,2,3,4,5,6,7,8,9];
$result = is_array($arr);
print_result($result);
// BOOLEAN : true
```

PHP offre de nombreuses fonctions pour manipuler les tableaux. Voici quelques-unes des plus courantes :

- `count($array)` : Renvoie le nombre d'éléments dans un tableau.

```php
echo count($fruits);  // Affiche le nombre d'éléments dans $fruits
```

- `array_push($array, $value)` : Ajoute un ou plusieurs éléments à la fin d'un tableau.

```php
array_push($fruits, "Mangue");
```

- `array_pop($array)` : Retire le dernier élément d'un tableau.

```php
array_pop($fruits);  // Retire "Mangue"
```

- `array_shift($array)` : Retire le premier élément d'un tableau.

```php
array_shift($fruits);  // Retire "Pomme"
```

- `array_unshift($array, $value)` : Ajoute un ou plusieurs éléments au début d'un tableau.

```php
array_unshift($fruits, "Fraise");  // Ajoute "Fraise" au début
```

- `in_array($value, $array)` : Vérifie si une valeur existe dans un tableau.

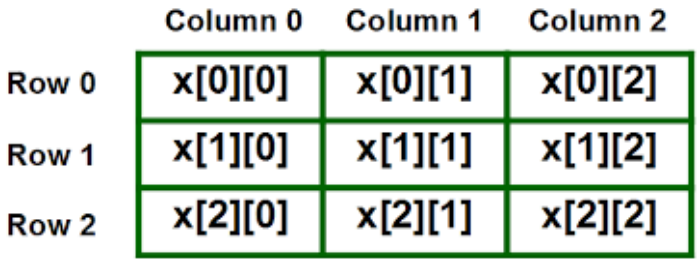
```php
if (in_array("Banane", $fruits)) {
    echo "Banane est dans le tableau.";
}
```

- `array_key_exists($key, $array)` : Vérifie si une clé existe dans un tableau associatif.

```php
if (array_key_exists("âge", $personne)) {
    echo "La clé 'âge' existe.";
}
```

6. Tableaux multidimensionnels

Un tableau multidimensionnel est un tableau qui contient d'autres tableaux. Cela peut être utile pour stocker des données complexes, comme des matrices ou des tableaux de tableaux.



Exemple : Tableau multidimensionnel indexé

```php
$équipe = [
    ["nom" => "Jean", "rôle" => "Développeur"],
    ["nom" => "Alice", "rôle" => "Designer"]
];

echo $équipe[0]["nom"];  // Affiche "Jean"
```

Exemple : Tableau multidimensionnel associatif

```php
$équipe = [
    "développeurs" => [
        ["nom" => "Jean", "rôle" => "Développeur"],
        ["nom" => "Paul", "rôle" => "Développeur"]
    ],
    "designers" => [
        ["nom" => "Alice", "rôle" => "Designer"]
    ]
];

echo $équipe["développeurs"][0]["nom"];  // Affiche "Jean"
```

7. Conclusion

Les tableaux en PHP sont des outils puissants pour organiser et manipuler des collections de données. Que ce soit pour stocker des valeurs simples avec des indices numériques ou gérer des ensembles complexes de données avec des clés personnalisées, les tableaux offrent une flexibilité importante pour travailler avec des données.