PHP : Les tableaux

Les tableaux sont des structures essentielles en PHP qui te permettent de stocker et de **gérer plusieurs valeurs dans une seule variable**. Grâce aux tableaux, tu peux organiser et traiter des données de manière structurée, que ce soit une **liste** simple ou des informations plus complexes. Dans cet article, tu découvriras les différents types de tableaux, comment les utiliser avec des exemples concrets, ainsi que quelques bonnes pratiques à adopter.

Objectifs de la leçon

- **Comprendre** ce qu'est un tableau en PHP.
- **Savoir déclarer** et utiliser différents types de tableaux (indexés, associatifs, multidimensionnels).
- **Apprendre** à parcourir les tableaux avec des boucles, notamment avec `foreach`.
- **Mettre en pratique** ces notions avec des exemples concrets.
- **Adopter** de bonnes pratiques pour écrire un code clair et maintenable.

Qu'est-ce qu'un tableau en PHP ?

Un tableau est une variable capable de stocker plusieurs valeurs. Il existe plusieurs types de tableaux :

- **Tableaux indexés** : Les éléments sont accessibles via des indices numériques.
- **Tableaux associatifs** : Chaque valeur est associée à une clé (nom) explicite.
- **Tableaux multidimensionnels** : Des tableaux qui contiennent d'autres tableaux, utiles pour organiser des données complexes.

Les tableaux indexés en PHP

Un **tableau indexé** (ou **array numéroté**) est une structure de données qui permet de *stocker plusieurs valeurs dans une seule variable*. Chaque valeur est repérée par un **indice numérique (index)**, qui commence généralement à `0`. C'est comme une liste : on y met des éléments, les uns après les autres.

À quoi ça sert ?

Les tableaux sont très utiles pour **grouper des données** du même type ou de types différents. Par exemple : une liste de nombres, de noms, ou une suite d'objets.

On utilise souvent les tableaux pour répéter des actions sur chacun de ses éléments.

Il sera par la suite facile d'appliquer un traitement ou la une suite d'instructions à chacun des éléments de ce tableau.

Différentes façons de créer un tableau

Il y a **plusieurs manières** de créer un tableau en PHP :

```php
// Avec la syntaxe ancienne (toujours valide)
$notes = array(12, 15, 17);

```

```php
// Avec la syntaxe moderne (recommandée)
$notes = [12, 15, 17];
```

Exemples concrets

✅ Tableau d'entiers

```php
$notes = [10, 12, 15, 19];
// $notes[0] vaut 10
// $notes[3] vaut 19
```

✅ Tableau de chaînes de caractères

```php
$prenoms = ["Alice", "Bob", "Chloé"];
// $prenoms[1] vaut "Bob"
```

Parcourir un tableau indexé avec `foreach`

La **meilleure façon de parcourir un tableau indexé** en PHP est d'utiliser une boucle `foreach`. Elle permet de lire **chaque élément** du tableau sans se soucier de son indice. C'est plus simple, plus lisible, et il y a moins de risques d'erreurs qu'avec une boucle `for`.

Exemple :

```php
$notes = [10, 13, 15, 17];

foreach ($notes as $note) {
    echo "Note : $note<br>";
}
```

Dans cet exemple, PHP va automatiquement lire chaque valeur du tableau `$notes` une par une, et la stocker dans la variable `$note` à chaque tour de boucle.

Tu peux aussi récupérer l'indice (la position) de chaque élément:

```php
$notes = [10, 13, 15, 17];

foreach ($notes as $index => $note) {
    echo "Note n°$index : $note<br>";
}
```

Ce code affichera:

```
Note n°0 : 10
Note n°1 : 13
Note n°2 : 15
Note n°3 : 17
```

🧠 À retenir

- `foreach` est **plus pratique** que `for` pour lire un tableau complet.
- Tu n'as pas besoin de connaître la taille du tableau ni les indices.
- C'est la solution recommandée pour tous les parcours simples.

Ce qu'il faut retenir

- Un **tableau indexé** en PHP est une **liste ordonnée**.
- Chaque élément est accessible par un **indice numérique** (ex. `$tableau[0]`).
- Tu peux stocker des **valeurs de tout type** : nombres, textes, objets…
- On les utilise souvent pour **répéter des actions** (avec une boucle `foreach`, par exemple).

Les tableaux associatifs en PHP

Un **tableau associatif** est un tableau dans lequel **chaque valeur est associée à une clé** (appelée aussi "index"), mais **cette clé est un texte** (une chaîne de caractères), et pas un nombre.

C'est un peu comme un **dictionnaire** : chaque mot (la clé) correspond à une définition (la valeur). Cela permet de nommer les éléments, au lieu d'utiliser des numéros.

Pourquoi les utiliser ?

Les tableaux associatifs sont très utiles quand tu veux **organiser des données avec des étiquettes** claires. Cela rend ton code plus lisible et plus facile à comprendre.

Comment les créer ?

```php
// Syntaxe avec l'ancienne méthode
$eleve = array(
  "nom" => "Emma",
  "age" => 16,
  "classe" => "5e"
);

// Syntaxe moderne (recommandée)
$eleve = [
  "nom" => "Emma",
  "age" => 16,
  "classe" => "5e"
];
```

Exemples concrets

✅ Fiche d'un élève

```php
$eleve = [
  "nom" => "Alex",
  "age" => 17,
  "section" => "Informatique"
];

// $eleve["nom"] vaut "Alex"
// $eleve["age"] vaut 17
```

✅ Configuration d'un personnage de jeu

```php
$personnage = [
  "pseudo" => "DragonSlayer",
  "pv" => 100,
  "niveau" => 5
];
```

✅ Répertoire téléphonique

```php
$contacts = [
  "maman" => "0486 12 34 56",
  "papa" => "0475 98 76 54",
  "ami" => "0491 11 22 33"
];

// $contacts["papa"] vaut "0475 98 76 54"
```

Parcourir un tableau associatif avec `foreach`

Lorsque tu utilises un **tableau associatif**, chaque élément possède une **clé** (souvent un mot) associée à une **valeur**. Pour parcourir ce type de tableau, on utilise la boucle `foreach` en récupérant **la clé et la valeur** à chaque tour.

Exemple :

```php
$eleve = [
  "nom" => "Alex",
  "age" => 17,
  "section" => "Informatique"
];

foreach ($eleve as $cle => $valeur) {
    echo "$cle : $valeur<br>";
}
```

Ce code va afficher :

```
nom : Alex
age : 17
section : Informatique
```

🧠 À retenir

- `$cle` contient le **nom du champ** (ex. : "nom", "age", "section").
- `$valeur` contient la **valeur associée** (ex. : "Alex", 17...).
- Ce type de boucle est idéal pour **lire ou afficher** les contenus d'un objet sous forme de tableau.

Ce qu'il faut retenir

- Un **tableau associatif** utilise des **clés textuelles** pour accéder aux données.
- C'est utile quand tu veux **donner un sens à chaque donnée** (ex. : "nom", "age", "score").
- Tu peux mélanger différents types de données dans un même tableau.
- C'est très courant quand tu manipules **du JSON ou des objets** transformés en tableaux.

Tableau comparatif : indexés vs associatifs

Caractéristique	Tableau indexé	Tableau associatif
Type d'index	Nombres entiers (0, 1, 2, ...)	Chaînes de caractères ("nom", "age", ...)
Ordre des éléments	Important et conservé	Important, mais l'accès se fait par **clé**
Accès à une valeur	`$tableau[0]`	`$tableau["nom"]`
Création (syntaxe moderne)	`$t = [1, 2, 3];`	`$t = ["clé1" => "val1", "clé2" => "val2"];`
Utilisation typique	Listes, séries, boucles numérotées	Objets, fiches, données avec étiquettes
Exemple	`$nombres = [10, 20, 30];`	`$eleve = ["nom" => "Emma", "age" => 16];`
Lisibilité	Moins explicite (indices numériques)	Plus lisible grâce aux noms de clés
Boucle recommandée	`foreach ($t as $val)`	`foreach ($t as $cle => $val)`

🧠 À retenir

- Utilise un **tableau indexé** quand tu veux juste stocker des valeurs dans un ordre précis.
- Utilise un **tableau associatif** quand tu veux **nommer** les éléments pour mieux comprendre et organiser tes données.

Souhaites-tu une fiche d'exercice pour que les élèves s'entraînent à utiliser les deux types ?

Exemples Concrets

Exemple 1 : Liste de Courses (Tableau Indexé)

Imagine que tu prépares une liste de courses pour la semaine. Tu peux stocker chaque article dans un tableau indexé.

```php
<?php
$listeCourses = ["Pain", "Lait", "Beurre", "Œufs", "Fromage"];

echo "Liste de courses :<br>";
foreach ($listeCourses as $article) {
    echo "- $article<br>";
}
?>
```

Explication :
Chaque élément de la liste est stocké dans le tableau `$listeCourses` et affiché grâce à une boucle `foreach`.

Exemple 2 : Les Notes d'un Élève (Tableau Associatif)

Supposons que tu veuilles stocker les notes d'un élève pour différentes matières. Tu utiliseras un tableau associatif où la clé représente la matière.

```php
<?php
$notesEleve = [
    "Mathématiques" => 15,
    "Français"      => 13,
    "Histoire"      => 14,
    "Sciences"      => 16,
```

```php
    "Anglais"       => 12
];

echo "Notes de l'élève :<br>";
foreach ($notesEleve as $matiere => $note) {
    echo "$matiere : $note/20<br>";
}
?>
```

Explication :
Ce tableau associe chaque matière à sa note, ce qui facilite le calcul d'une moyenne ou la recherche d'une note spécifique.

Exemple 3 : Planning de la Semaine (Tableau Multidimensionnel)

Tu peux organiser ton planning hebdomadaire en stockant les activités de chaque jour dans un tableau multidimensionnel.

```php
<?php
$planningSemaine = [
    "Lundi"    => ["08:00" => "Cours de Maths", "10:00" => "Récréation", "11:00" => "Cours d'A
    "Mardi"    => ["09:00" => "Cours de Sciences", "11:00" => "Atelier Informatique"],
    "Mercredi" => ["08:00" => "Cours d'Histoire", "10:00" => "Sport"],
    "Jeudi"    => ["09:00" => "Cours de Français", "11:00" => "Cours de Géographie"],
    "Vendredi" => ["08:00" => "Cours de Musique", "10:00" => "Réunion de classe"]
];

echo "Planning de la semaine :<br>";
foreach ($planningSemaine as $jour => $activites) {
    echo "<strong>$jour</strong><br>";
    foreach ($activites as $heure => $activite) {
        echo "$heure - $activite<br>";
    }
    echo "<br>";
}
?>
```

Explication :
Ici, le tableau `$planningSemaine` contient pour chaque jour un autre tableau avec les heures et les activités. Cela te permet d'organiser des données complexes de manière structurée.

Exemple 4 : Inventaire de Produits (Tableau Associatif)

Dans une boutique, tu peux suivre le stock des produits en utilisant un tableau associatif.

```php
<?php
$inventaire = [
    "Stylo"  => 150,
    "Cahier" => 80,
    "Gomme"  => 200,
    "Règle"  => 50
];
```

```php

echo "Inventaire des produits :<br>";
foreach ($inventaire as $produit => $quantite) {
    echo "$produit : $quantite en stock<br>";
}
?>
```

Explication :
Chaque produit est associé à une quantité, ce qui facilite la gestion et le réapprovisionnement.

Exemple 5 : Carnet d'Adresses (Tableau Multidimensionnel)

Tu peux créer un carnet d'adresses où chaque contact est représenté par un tableau associatif contenant son nom, son numéro et son email.

```php
<?php
$carnetAdresses = [
    [
        "nom"    => "Dupont",
        "tel"    => "0123456789",
        "email"  => "dupont@example.com"
    ],
    [
        "nom"    => "Martin",
        "tel"    => "0987654321",
        "email"  => "martin@example.com"
    ],
    [
        "nom"    => "Durand",
        "tel"    => "0147258369",
        "email"  => "durand@example.com"
    ]
];

echo "Carnet d'adresses :<br>";
foreach ($carnetAdresses as $contact) {
    echo "Nom : " . $contact["nom"] . "<br>";
    echo "Téléphone : " . $contact["tel"] . "<br>";
    echo "Email : " . $contact["email"] . "<br><br>";
}
?>
```

Explication :
Ce tableau multidimensionnel te permet de gérer une collection de contacts, chaque contact étant lui-même un tableau associatif.

Bonnes Pratiques

- **Déclaration et initialisation :**
  Utilise la syntaxe courte `[]` pour déclarer tes tableaux, sauf si tu as besoin de compatibilité avec d'anciennes

versions de PHP.

- **Utilisation des boucles :**
  La boucle `foreach` est très adaptée pour parcourir les tableaux, qu'ils soient simples ou multidimensionnels.

- **Fonctions utiles :**
  Familiarise-toi avec des fonctions comme `count()`, `array_push()`, `in_array()`, etc., pour manipuler tes tableaux plus efficacement.

- **Organisation des données :**
  Choisis le type de tableau (indexé, associatif, multidimensionnel) en fonction de la structure de tes données et de l'opération à effectuer.

- **Lisibilité du code :**
  N'oublie pas de commenter ton code pour expliquer la structure et le rôle de chaque tableau, surtout dans des projets collaboratifs.

Ce qu'il faut retenir

1. **Un tableau permet de stocker plusieurs valeurs dans une seule variable.**
2. **Les tableaux indexés utilisent des indices numériques, tandis que les tableaux associatifs utilisent des clés explicites.**
3. **Les tableaux multidimensionnels organisent des données complexes en imbriquant plusieurs tableaux.**
4. **La boucle `foreach` est idéale pour parcourir les tableaux de manière simple et lisible.**
5. **L'utilisation de fonctions dédiées (comme `count()`) améliore la manipulation des tableaux.**

Conclusion

En maîtrisant les tableaux en PHP, tu pourras organiser tes données de façon claire et structurée, ce qui est une compétence essentielle en programmation. Bonne pratique et n'hésite pas à expérimenter avec différents types de tableaux !

Vidéo à la une à regarder sur Youtube: http://youtu.be/vZs202lVwJQ