Les Arrays

# Introduction

Nous avons vu qu’une variable peut stocker une valeur.

Pour le moment si on veut stocker plusieurs valeurs il nous faut plusieurs variables.

Pour stoker plusieurs fruits en string :

$fruit1="pomme";

$fruit2="poire";

$fruit3="cerise";

Pour stoker plusieurs nombres :

$nb1=15;

$nb2=2;

$nb3=6;

$nb4=50;

En php et en js il est possible dans une variable de stocker plusieurs valeurs en utilisant en les regroupant dans un array.

$fruits=["pomme","poire","cerise"];

$nombres=[15,2,6,50];

Pour créer un array on affecte un array représenter par une paire de [ ] **Attention [ ] crochets pas { } accolades** . On peut placer les valeurs directement dans l’array séparé par des virgules.

Dans un array on peut stocker n’importe quel type de valeur (même un autre array).

$melange=["machin",50,true,null,["rouge","vert"]];

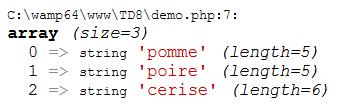
# Mieux comprendre les arrays et lire les valeurs d’un array

Pour mieux comprendre comment fonctionne un array nous allons faire un var\_dump() d’un array.

$fruits=["pomme","poire","cerise"];

var\_dump($fruits);

On obtient ceci :

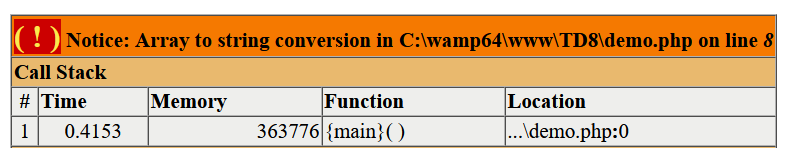


Sur la gauche on remarque des nombre 0,1,2 on appelle c’est nombre **des clés ou index**. On voit à chaque clé il est associé **une valeur** a 0 il est associé « pomme » a 1 il est associé « poire » ici pomme, poire et cerise sont des valeurs. Dans un array a chaque valeur il est associé une clé.

La clé nous permet de retrouver la valeur plus facilement pour pouvoir par exemple la récupérer pour l’afficher, la modifier ou la supprimer.

Si on essaye de faire directement un echo d’un array cela va déclencher une erreur.

echo $fruits;



Cela nous indique qu’il n’est pas possible d’afficher un array via echo car on ne peut pas le convertir en string (en tout cas de cette manière).

Si on souhaite afficher une valeur de l’array par exemple pomme il faut faire ceci :

echo $fruits[0];

On peut récupérer une des valeurs stocker dans l’array en utilisant la clé.

On peut également utiliser la même méthode pour stocker pomme dans une nouvelle variable.

$fruit1=$fruits[0];

Si on souhaite afficher poire :

echo $fruits[1];

Et pour cerise :

echo $fruits[2];

# Les clés

**Dans un array les clés peuvent être des nombres ou les clés peuvent être de string.**

Dans les cas ou les clés sont de string on parle d’array associatif.

En général, on utilise des clés numériques quand les arrays contiennent de données homogènes.

Et on utilise des chaines quand les données sont hétérogènes.

Exemple : une liste de fruits sont de données homogènes. En revanche les informations d’un utilisateur (nom,prenom,email,ville, age …) sont de données hétérogène.

Dans le cas ou créer un array qui contient lui-même d’autre array on parle d’array multidimensionnelle. Par exemple un array avec des lignes et de colonnes.

( En JS les arrays on seulement des clés numérique)

# Création d’un array avec clé numérique

Comme on l’a vu précédemment il suffit d’indiquer les valeurs comme ceci :

$fruits=["pomme","poire","cerise"];

Les clés sont automatiquement générées 0 pour pomme, 1 pour poire …

# Création d’un array associatif

$utilisateur=["nom"=>"Moulin","prenom"=>"Timothée","id"=>23,"email"=>"timomoulin@msn.com"];

Contrairement a un array classique il faut indiquer la clé pour chaque valeur.

La clé nom est associé à la valeur Moulin ….

# Création d’un array multidimensionnelle

$morpion=[["O","X","O"],["X","X","O"],["O","X",""]];

$articles = [

 ["id" => 1, "nom" => "t-shirt", "prix" => 10],

 ["id" => 2, "nom" => "robe", "prix" => 50]

];

# Consulter une valeur dans un array standard :

Pour récupérer une valeur dans un array on la récupère par son index.

$nombres = [15, 2, 6, 50];

 // recupere la valeur a la position 0 et l'affiche (ici 15)

echo $nombres[0];

// recupere la valeur a la position 3 et l'affiche (ici 50)

echo $nombres[3];

// on recupere la valeur et la stocke dans une variable

$nb1=$nombres[0];

# Consulter une valeur dans un array associatif :

De manière similaire on récupère les valeurs par leurs clés.

$utilisateur = ["nom" => "Moulin", "prenom" => "Timothée", "id" => 23, "email" => "timomoulin@msn.com"];

//on affiche "Moulin"

echo $utilisateur["nom"];

//on affiche 23

echo $utilisateur["id"];

//on stocke prenom dans une variable

$unPrenom=$utilisateur["prenom"];

# Consulter une valeur dans un array multidimensionnelle :

Pour consulter une valeur dans un array multidimensionnelle il faut indiquer en 2 clés.

En premier on indique la clé de l’array que l’on veut consulter. Puis on indique la clé de la valeur que l’on veut consulter.

$articles = [

 ["id" => 1, "nom" => "t-shirt", "prix" => 10],

 ["id" => 2, "nom" => "robe", "prix" => 50]

];

Par exemple si je veux afficher le prix de la robe :

Je veux consulter l’array a la position 1 qui représente la robe car l’array a la position 0 représente le t-shirt.

Dans l’array a la position 1 je veux afficher la valeur qui correspond à la clé prix.

Cela donne donc :

echo $articles[1]["prix"]; //50

Si je veux afficher l’id du t-shirt :

echo $articles[0]["id"];//1

# Modifier une valeur dans un array

Pour modifier une valeur dans un array il suffit de cibler une valeur par sa clé puis de lui affecter une nouvelle valeur.

Note : si la clé n’existe pas dans l’array alors la valeur est ajouté.

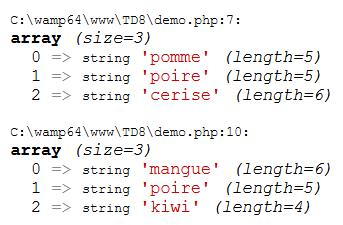
$fruits = ["pomme", "poire", "cerise"];

var\_dump($fruits);

$fruits[0]="mangue";

$fruits[2]="kiwi";

var\_dump($fruits);



De même pour un array associatif ou dimensionnelle :

$utilisateur = ["nom" => "Moulin", "prenom" => "Timothée", "id" => 23, "email" => "timomoulin@msn.com"];

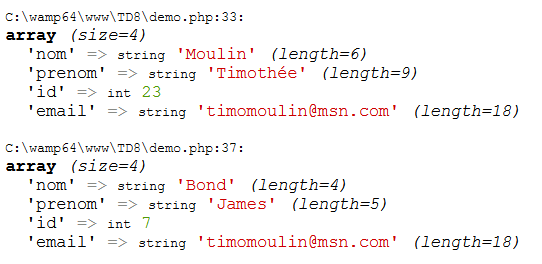
var\_dump($utilisateur);

$utilisateur["nom"]="Bond";

$utilisateur["prenom"]="James";

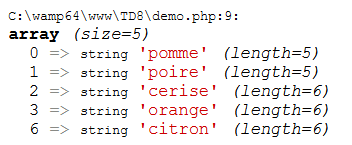
$utilisateur["id"]=007;

var\_dump($utilisateur);

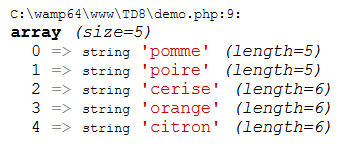


# Ajouter une valeur dans un array :

* On peut ajouter une valeur en utilisant la clé et en lui affectant une valeur comme pour la modification.
* $fruits = ["pomme", "poire", "cerise"];
* $fruits[3]="orange";
* $fruits[6]="citron";
* var\_dump($fruits);



* On peut aussi dans le cas d’un array avec des clés numériques utilisé le code suivant pour ajouter une valeur a la fin de l’array
* $fruits = ["pomme", "poire", "cerise"];
* $fruits[]="orange";
* $fruits[]="citron";
* var\_dump($fruits);



# Supprimer une valeur

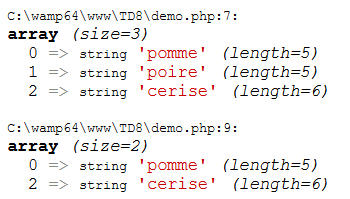
Pour supprimer une valeur dans un array on peut utiliser la fonction **unset()**.

$fruits = ["pomme", "poire", "cerise"];

var\_dump($fruits);

unset($fruits[1]);

var\_dump($fruits);



On remarque que la cerise est toujours à la position 2.

Pour un array avec des clés numériques on peut les redistribuer en utilisant la fonction **array\_values().**

$fruits = ["pomme", "poire", "cerise"];

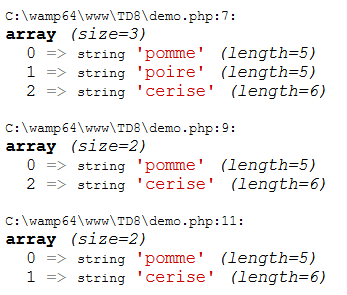
var\_dump($fruits);

unset($fruits[1]);

var\_dump($fruits);

$fruits=array\_values($fruits);

var\_dump($fruits);



# Connaitre la taille d’un array

Pour connaitre la taille cad le nombre d’élément dans le tableau on peut utiliser la fonction **count()**

$fruits = ["pomme", "poire", "cerise"];

echo count($fruits);//3

$fruits[]="kiwi";

$fruits[]="mangue";

echo count($fruits);//5

Note : le dernier élément d’un tableau numérique sera à la position longeur-1.

Si le tableau a 3 valeurs (la longueur est 3) le dernier élément est à l’index 2.

Si le tableau a n valeurs (la longueur est n) le dernier élément est à l’index n-1.

# Parcourir un array :

Si on souhaite par exemple afficher ou effectuer un traitement sur toutes les valeurs d’un tableau il peut être fastidieux de devoir ciblé chaque valeur par leurs clés une à une.

Pour éviter cela on peut utiliser des boucles.

* Si le tableau est array avec des clés numériques alors on peut utiliser une boucle for (ou éventuellement while) pour parcourir toutes les valeurs de l’array **(Ne marche que si le tableau a des clés numériques)**.

Exemple avec une boucle for :

$fruits = ["pomme", "poire", "cerise"];

for($i=0;$i<count($fruits);$i++){

    $unFruit=$fruits[$i];

    echo $unFruit."<br>";

}

* On peut utiliser une autre boucle pour parcourir les tableaux **même pour les tableaux associatifs** il s’agit de la boucle **foreach**.

$fruits = ["pomme", "poire", "cerise"];

foreach($fruits as $unFruit){

    echo "$unFruit <br>";

}

$fruits correspond a l’array que je souhaite parcourir $unFruit correspond a une nouvelle variable qui va stocke a chaque iteration la valeur que l’on est entrain de parcourir ( pomme puis poire puis cerise)

Contrairement a la boucle for classique la boucle foreach fonctionne également sur un array associatif.

$utilisateur = ["nom" => "Moulin", "prenom" => "Timothée", "id" => 23, "email" => "timomoulin@msn.com"];

foreach($utilisateur as $uneValeur){

    echo "$uneValeur <br>";

}

Si on souhaite afficher également les clés on peut modifier la boucle foreach comme ceci :

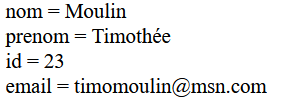
foreach($utilisateur as $cle=>$uneValeur){

    echo "$cle = $uneValeur <br>";

}

$cle prendra tour a tour les valeur de nom, prenom,id, email.

Tandis que $uneValeur prendrat tour a tour les valeurs de Moulin, Timothée, 23, [timomoulin@msn.com](mailto:timomoulin@msn.com)



# Parcourir un array dimensionnelle :

Le plus simple est de partir du principe qu’un array dimensionnelle est première composé de ligne et que dans chaque ligne on trouve plusieurs colonnes.

Pour parcourir un array dimensionnelle il faut 2 boucles imbriqué l’une dans l’autre. La première parcours les lignes et la seconde parcours les colonnes de chaque ligne.

$morpion = [["O", "X", "O"], ["X", "X", "O"], ["O", "X", ""]];

foreach($morpion as $ligne){

    foreach($ligne as $col){

        echo $col;

    }

    echo "<br>";

}



$articles = [

 ["id" => 1, "nom" => "t-shirt", "prix" => 10],

 ["id" => 2, "nom" => "robe", "prix" => 50]

];

foreach($articles as $unArticle){

    foreach($unArticle as $cle=>$uneValeur)

    {

        echo "$cle = $uneValeur <br>";

    }

    echo "<br>";

}

