# Exercices callbacks et fonctions anonymes (variables et paramètres)

1. Créez une fonction anonyme qui affiche "bonjour" sur l'écran. Stockez-la dans une variable. Testez-la.
2. Créez une fonction anonyme qui **affiche** la somme de deux éléments envoyées en paramètre. Stockez le dans une variable. Testez-la.
3. Créez une fonction anonyme qui reçoit un array de strings et affiche chaque string en majuscules (utilisez mb\_strtoupper à l'intérieur de votre fonction). Stockez le dans une variable. Testez-la.
4. Créez une fonction anonyme qui reçoit deux arrays et crée un array associatif où les clés sont les éléments du premier array et les valeurs sont les éléments du deuxième.

Si les tableaux n'ont pas la même taille la fonction renvoie **null**. Exemple :

    $tabCles = ['nom', 'prénom', 'profession', 'hobby'];

    $tabVals = ['Groening', 'Matt', 'scénariste', 'dessiner'];

    $tabAssoc = $creerAssoc($tabCles, $tabVals);

Doit créer :

**array** *(size=4)*

'nom' => string 'Groening' *(length=8)*

'prénom' => string 'Matt' *(length=4)*

'profession' => string 'scénariste' *(length=11)*

'hobby' => string 'dessiner' *(length=8)*

1. Créez une fonction anonyme qui reçoit une valeur et renvoie true si la valeur est paire et faux autrement
2. Créez une fonction anonyme qui **renvoie** la somme des éléments d'un array envoyé en paramètre
3. Créez une fonction **isImageType (string lien, string type) : bool** capable de nous dire si un lien vers une image (string) correspond à un type d'image. Ex :

$isImageType ('Yevgeny Zamiatin.jpg', '.jpg') // renvoie true

1. Créez une simple fonction capable d'afficher un array reçu en paramètre. Utilisez une boucle foreach pour parcourir les éléments de l'array reçu et les afficher.
2. Modifiez la fonction précédente pour qu'elle ne sert pas uniquement à afficher. Commencez par rajouter un deuxième paramètre qui sera une fonction. Dans la boucle foreach, faites appel à la fonction reçue en paramètre sur chaque élément de l'array (n'oubliez pas le $). Cet exemple à été fait pendant le cours.

Pour tester son bon fonctionnement créez deux nouvelles fonctions callback :

1. Fonction qui reçoit un chiffre et **affiche** le chiffre fois 10
2. Fonction qui reçoit un string et **l'affiche** en mettant la première lettre en majuscule (utilisez la fonction ucfirst)

Puis faites appel à la fonction qui parcourt l'array en envoyant chaque callback.

    parcourirArray([10, 30, 40], $fois10);

    parcourirArray(['vin', 'fromage', 'chocolat'], $capitalize);

1. Créez une nouvelle version de la fonction qui parcourt l'array. Cette nouvelle version sera capable de créer un nouvel array, qui sera le résultat d'avoir appliquer le callback à chaque élément de l'array envoyé en paramètre. Avant de sauter par la fenêtre ou m'insulter, essayez de comprendre :

Nous avons deux nouveaux callbacks:

1. Fonction qui reçoit un chiffre et **renvoie** le chiffre fois 10
2. Fonction qui reçoit un string et **le renvoie** après avoir mise la première lettre en majuscule (utilisez la fonction ucfirst)

Nous voulons que la fonction qui parcourt l'array puisse générer pendant son parcours un nouvel array en appliquant le callback. Cet array sera renvoyé avec return.

A la fin, on doit pouvoir lancer ce code :

    $tabFois10 = parcourirGenereArray([10, 30, 40], $fois10);

    $tabCapitalize = parcourirGenereArray(['vin', 'fromage', 'chocolat'],  $capitalize);

    var\_dump ($tabFois10); // affichera l'array modifié

    var\_dump ($tabCapitalize); // affichera l'array modifié

Si vous arrivez à réaliser cette fonction, félicitations car vous venez d'implémenter le noyau de la fonction array\_map!

1. Utilisez la fonction array\_map avec les callbacks que vous venez de créer sur ces arrays. Vous devriez obtenir exactement les mêmes résultats
2. Utilisez la fonction array\_map pour créer un array associatif qui sera la combinaison de ces deux arrays :

    $tabPrenoms = ['Eduard', 'Ada', 'Nikola', 'Marian'];

    $tabNoms = ['Snowden', 'Lovelace', 'Tesla', 'Rejewski'];

1. Vous avez cet array :

$tab = [ 22, -21, -45, 2, -8, 11 ];

Utilisez array\_filter pour obtenir un nouvel array contenant uniquement les chiffres paires. Utilisez comme callback la fonction de l'exercice 5)

1. Modifiez l'exercice précédant pour qu'on puisse obtenir uniquement les chiffres paires et négatives
2. Vous avez un array contenant de liens vers des images .jpg, .png, .gif, etc… reçu d'une base de données :

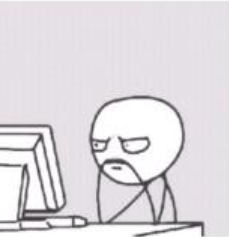
$tab = ['biere.jpg','miammiam.jpg','loup.png','souris.png','alien.jpg',

'demogorgon.gif','BillyJoel.jpg','Santa Claus.gif'];

Créez un callback $isImageJpg et utilisez-le avec array\_filter pour générer un nouvel array contenant uniquement les images .jpg. Exemple :

$tabJPG = array\_filter ($tab, $isImageJpg); // tabJPG contient uniquement // les .jpgs

**A partir de maintenant …. il va falloir se raccrocher!!**



1. Cet exercice consiste uniquement à comprendre le code de Ex16.php (la solution). On crée le callback inline, sans passer par la variable.
2. Modifiez l'exercice 15 pour qu'on puisse choisir le type d'image qu'on veut sélectionner.

Nous avions déjà, dans l'exercice 15 :

$isImageJpg = function ($lien)

Et alors vous pensez à créer quelque chose comme :

$isImageMonType = function ($lien, **$type**)

Ou (en modifiant ex16.php)

$tabJPG = array\_filter ($tab, function ($lien, **$type**) {

**Mais** **array\_filter l'appelle elle envoie uniquement un paramètre** (l'élément de l'array). Quoi faire alors?

Regardez la solution expliquée dans Ex17.php

1. Encore plus loin! Et si on ne veut pas avoir une variable globale? Une solution possible se trouve dans Ex18.php : mettre tout le code dans une autre fonction et lui envoyer le type recherché en paramètre.
2. Et encore plus loin : création d'une usine à fonctions qui repondront à nos critères. L'usine fait **return** d'une fonction qui a été crééé selon nos besoins (les paramètres qui reçoit l'usine, ici uniquement le type à rechercher)
3. J'invite au repas de midi à la personne qui arrive à nous expliquer la totalité de cet exemple (ex20.php)

