# Les fonctions en PHP

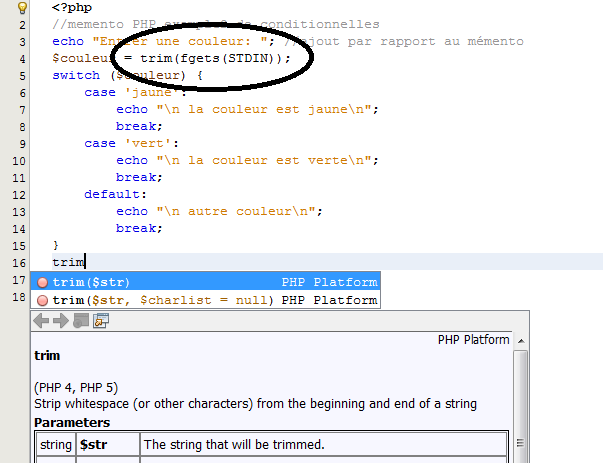
Une fonction est un bloc de code PHP destiné généralement à être réutilisé plusieurs fois.

**Plutôt que d'écrire plusieurs fois le même morceau de code, on met celui-ci dans une fonction** et c'est cette fonction que l'on appellera dès que l'on en aura besoin.

# Fonctions prédéfinies

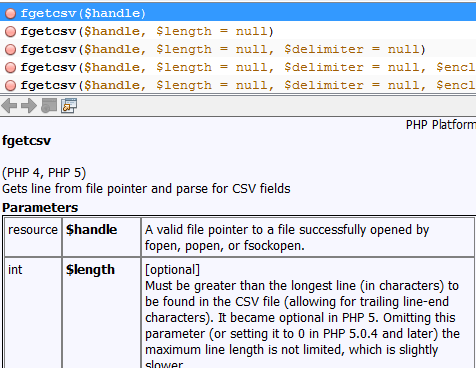
En PHP, comme dans tous les langages de programmation, il y a déjà des **fonctions prédéfinies** que l'on peut utiliser immédiatement.

## Exemple 1



## Exemple 2

<?php   
$ligne = 1; // compteur de lignes  
$fic = fopen("acteurs.csv", "a+");  
while($tab=fgetcsv($fic,1024,';'))  
{  
$champs = count($tab);//nombre de champs dans la ligne en question   
echo "<b> Les " . $champs . " champs de la ligne " . $ligne . " sont :</b><br />";  
$ligne ++;  
//affichage de chaque champ de la ligne en question  
for($i=0; $i<$champs; $i ++)  
{  
echo $tab[$i] . "<br />";  
}  
}  
?>



Pour autant, rien ne nous empêche d'en créer d'autres selon nos besoins.

Et plus encore, on peut se servir des fonctions PHP pour créer nos propres fonctions.

## Exemple 3

*<?php   
// Pour connaître la version de PHP utilisée on écrit :   
echo phpversion();   
?>*

# Création d'une fonction function nom\_fonction ($par1, $par2,…)

**Définition de la fonction**

Lors de la définition d'une fonction, on utilise en premier le mot clé '**function**'. Il est obligatoire de le mettre pour dire que le bloc traité par PHP est une fonction.

Ensuite, on met l’identificateur : le **nom de la fonction**.

**Une fonction peut ne pas retourner une valeur**. Dans ce cas, on parle généralement de **procédure**. C'est juste pour structurer le code et pour alléger le code en évitant les répétitions.

## Exemple de procédure affichant "Bonjour le monde !":

*<?php  
function hello()   
{   
echo "Bonjour le monde !";   
}  
  
hello(); //appelle la fonction "hello" qui va afficher "Bonjour tout le monde !" (sans les guillemets)  
?>*

Pour afficher « Bonjour le monde ! » n'importe où dans le code, il n'y aura plus qu'à taper:

*hello();*

Mais la plus grande utilité des fonctions vient du fait qu'elles peuvent **retourner une valeur**. On peut se servir d'une fonction pour **manipuler des variables**. Au départ, on leur introduit des valeurs qu'on appelle **les arguments ou paramètres de la fonction**.

**Le bloc de fonction est donc devenu un sous programme dans lequel on manipule les paramètres**. Après manipulation, le sous programme (en l'occurrence la fonction) retourne une valeur. Toutes les manipulations se déroulent à l'intérieur du bloc de fonction.

Nous venons de voir l'exemple d'une procédure qui ne contenait aucun paramètre. Une fonction peut avoir un ou plusieurs paramètres. Ces paramètres sont généralement des variables et peuvent être de types différents.

Après avoir écrit le nom de la fonction, on **énonce tous les paramètres entre les deux parenthèses.** En cas de plusieurs paramètres, il faut les séparer par des virgules.

## Exemple de fonction simple qui calcule la division de 2 nombres.

On met les deux nombres en paramètres. Le nombre retourné par la fonction est donc le résultat de la division des deux nombres. Le mot clé **return** permet de retourner le résultat.

*<?php  
function division($nombre1, $nombre2)   
{   
$resultat=$nombre1/$nombre2;  
return $resultat;  
}  
?>*

# Appel d'une fonction … nom\_fonction($par1, $par2,…) ;

Pour appeler une fonction, il suffit de l'appeler par **son nom et de spécifier les valeurs des paramètres**.

Si nous voulons par exemple appeler la fonction division :

*< ?php  
function division($nombre1, $nombre2)   
{   
$resultat=$nombre1/$nombre2;   
return $resultat ;   
}  
  
$valeur=division(100,50);  
echo $valeur;  
//ou encore  
$numerateur=100;  
$denominateur=50;  
$quotient=division($numerateur, $denominateur);  
echo $quotient;  
?>*

Remarquez bien qu'il y a **un ordre** dans la disposition des paramètres dans la définition de la fonction. **Si l'on intervertit les paramètres, on aura un résultat différent.**

# Valeurs par défaut des paramètres d'une fonction

## Exemple

*<?php   
function division ($nombre1 = 10, $nombre2 = 2)   
{   
$resultat=$nombre1/$nombre2;   
return $resultat ;   
}   
echo division();  
?>*

Ici, nous assignons comme valeurs par défaut, les nombres 10 et 2. Si nous ne spécifions pas les valeurs des paramètres lors de l'appel de la fonction, nous obtiendrons 5 comme résultat.

**Lorsque l'on met des valeurs par défaut, les deux paramètres ne sont pas obligatoires lors de l'appel de la fonction.**

# Librairie de fonctions include (« nom\_librairie.php ») ;

Dans nos exemples, on a toujours fait **l'appel** de la fonction suite à sa définition. Tout se passe donc **dans le même fichier**.

Mais lorsque l’on a plusieurs fonctions et que l’on veut utiliser les mêmes fonctions dans différents fichiers, il devient fastidieux de répéter les définitions des fonctions en en-tête de chaque fichier.

PHP nous offre la possibilité de créer **une bibliothèque (ou libraire) de fonctions et de les inclure tout simplement au début de chaque fichier**.

On 'inclut' alors le contenu de la librairie dans les autres fichiers, en l'appelant par la fonction '**include**' et en lui passant en paramètre **le nom du fichier à inclure**.