



Introduction au langage de programmation PHP

[Accueil](#) ► [Mes cours](#) ► [Développement logiciel](#) ► [Intro PHP](#) ► [Les fonctions](#) ► [Les fonctions utilisateurs](#)

Les fonctions utilisateurs

PHP nous permet de construire nos propres fonctions. Une fonction se déclare avec le mot clé "function". Ce mot clé sera suivi du nom de la fonction (par exemple `mafonction` ou `ma_function`) puis des paramètres toujours écrits entre parenthèses. S'il n'y a pas de paramètres, les parenthèses resteront vides mais seront toujours présentes. On ne pourra pas créer deux fonctions portant le même nom. Enfin les instructions à réaliser seront écrites entre `{}` dans le bloc d'instructions :

```
function ma_fonction($param1, $param2){  
    instruction_1 ;  
    instruction_2 ;  
    ...  
}
```

Une fois cette fonction créée, nous l'appellerons comme une fonction "normale" c'est-à-dire via son nom et en lui fournissant les arguments nécessaires. Par exemple la fonction `verifier_majorite()` pourrait être implémentée comme cela :

```
1 <?php  
2  
3 // fonctions_utilisateurs_1.php  
4 function verifier_majorite($age) {  
5     if ($age >= 18) {  
6         echo 'Vous êtes majeur';  
7     } else {  
8         echo 'Vous êtes mineur';  
9     }  
10 }  
11  
12 verifier_majorite(25); // appel de la fonction  
13 echo PHP_EOL;
```

Ou encore `additionner()` qui prendra en paramètre un nombre `nb1` et un nombre `nb2` :

```
1 <?php  
2  
3 // fonctions_utilisateurs_2.php  
4 function additionner($nb1, $nb2) {  
5     echo "$nb1 + $nb2 = " . ($nb1 + $nb2);  
6 }  
7  
8 additionner(10, 15);  
9 echo PHP_EOL;
```

Pour renvoyer une valeur à la fin de la fonction nous utiliserons le mot clé "return". Il est possible de mettre plusieurs `return` dans une fonction mais l'exécution d'un `return` met toujours fin à la fonction. Par exemple si nous souhaitons transformer la fonction `verifier_majorite()` pour qu'elle renvoie le résultat :

```
1 <?php  
2  
3 // fonctions_utilisateurs_3.php  
4 function verifier_majorite($age) {  
5     $resultat = '';
```

```

6     if ($age >= 18) {
7         $resultat = 'Vous êtes majeur';
8     } else {
9         $resultat = 'Vous êtes mineur';
10    }
11    return $resultat;
12 }
13
14 echo verifier_majorite(25) . PHP_EOL;

```

Nous pourrions également écrire :

```

1 <?php
2
3 // fonctions_utilisateurs_4.php
4 function verifier_majorite($age) {
5     if ($age >= 18) {
6         return 'Vous êtes majeur';
7     } else {
8         return 'Vous êtes mineur';
9     }
10 }
11
12 echo verifier_majorite(25) . PHP_EOL;

```

Si vous créez une variable dans une fonction, cette variable n'existera plus une fois la fonction exécutée. Cette variable est dite "locale" à la fonction.

```

1 <?php
2
3 // fonctions_utilisateurs_5.php
4 function do_truc() {
5     $message = 'azerty'; // création d'une variable dans une fonction
6     echo 'do_truc() est exécutée !';
7 }
8
9 do_truc();
10 echo $message; // renvoie une erreur de type NOTICE : undefined variable
11 echo PHP_EOL;

```

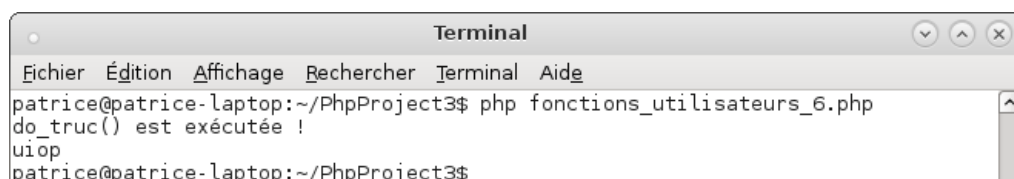
Le mot clé global permet d'utiliser au sein d'une fonction une variable définie hors de la fonction :

```

1 <?php
2
3 // fonctions_utilisateurs_6.php
4 $message = 'azerty';
5
6 function do_truc() {
7     global $message;
8     $message = 'uiop';
9     echo 'do_truc() est exécutée !' . PHP_EOL;
10 }
11
12 do_truc();
13 echo $message;
14 echo PHP_EOL;

```

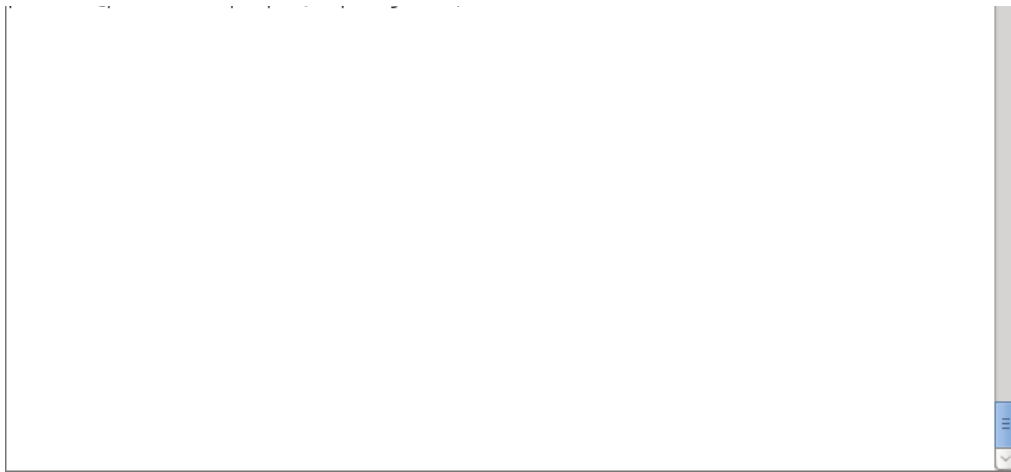
Cela affiche :



```

Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php fonctions_utilisateurs_6.php
do_truc() est exécutée !
uiop
patrice@patrice-laptop:~/PhpProject3$

```



Dans les fonctions PHP, il est possible de donner aux paramètres une valeur par défaut. Les paramètres qui possèdent une valeur par défaut deviennent facultatifs. On peut mélanger les paramètres "normaux" et les paramètres avec une valeur par défaut, mais les paramètres avec une valeur par défaut doivent toujours figurés à la fin. Quelques exemples :

```
1 <?php
2
3 // fonctions_utilisateurs_7.php
4 function se_presenter($age, $nom = 'inconnu', $prenom = 'inconnu') {
5     echo "votre nom est $nom, votre prénom est $prenom et vous avez $age ans";
6     echo PHP_EOL;
7 }
8
9 se_presenter(18);
10 se_presenter(45, 'Sparrow');
11 se_presenter(42, 'Wayne', 'Bruce');
```

Ce code affiche :

```
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
patrice@patrice-laptop:~/PhpProject3$ php fonctions_utilisateurs_7.php
votre nom est inconnu, votre prénom est inconnu et vous avez 18 ans
votre nom est Sparrow, votre prénom est inconnu et vous avez 45 ans
votre nom est Wayne, votre prénom est Bruce et vous avez 42 ans
patrice@patrice-laptop:~/PhpProject3$
```

Il est possible en PHP de gérer un nombre indéfini d'arguments. La fonction `func_num_args()` permet à l'intérieur d'une fonction, de récupérer le nombre d'arguments qui lui a été passé. La fonction `func_get_args()` renvoie un tableau contenant tous les arguments qui ont été passés à la fonction. Enfin la fonction `func_get_arg()` permet de récupérer un argument selon son index.

```
1 <?php
2
3 // fonctions_utilisateurs_8.php
4 function do_truc() {
5     $nb_arg = func_num_args();
```

```

6     echo "vous avez passé $nb_arg arguments à cette fonction";
7 }
8
9 function calculer_moyenne() {
10     $somme = 0;
11     $args = func_get_args();
12     foreach ($args as $value) {
13         $somme += $value;
14     }
15     $moyenne = $somme / func_num_args();
16     echo "la moyenne des arguments est $moyenne";
17 }
18
19 do_truc('Sparrow', 45645, [1, 2, 3], 12.5);
20 echo PHP_EOL;
21 calculer_moyenne(3, 4, 5);
22 echo PHP_EOL;

```

Dans un script les fonctions peuvent être déclarées n'importe où, même après leurs appels :

```

1 <?php
2
3 // fonctions_utilisateurs_9.php
4 do_truc('Sparrow', 45645, [1, 2, 3], 12.5);
5 echo PHP_EOL;
6 calculer_moyenne(3, 4, 5);
7 echo PHP_EOL;
8
9 function do_truc() {
10     $nb_arg = func_num_args();
11     echo "vous avez passé $nb_arg arguments à cette fonction";
12 }
13
14 function calculer_moyenne() {
15     $somme = 0;
16     $args = func_get_args();
17     foreach ($args as $value) {
18         $somme += $value;
19     }
20     $moyenne = $somme / func_num_args();
21     echo "la moyenne des arguments est $moyenne";
22 }

```

Les fonctions peuvent aussi être déclarées dans des structures de contrôle, mais dans ce cas, elles doivent être déclarées avant d'être appelées :

```

1 <?php
2
3 // fonctions_utilisateurs_10.php
4 $nom = 'Wayne';
5 if ($nom == 'Sparrow') {
6
7     function direBonjour() {
8         echo 'Bonjour Sparrow';
9     }
10
11 } else {
12
13     function direBonjour() {
14         echo 'Bonjour inconnu';
15     }
16
17 }
18 direBonjour();

```

[Fin](#)

NAVIGATION



Accueil

■ [Ma page](#)

[Pages du site](#)[Mon profil](#)[Cours actuel](#)

[Intro PHP](#)

[Participants](#)[Le langage PHP : introduction](#)[Les types et les variables](#)[Les opérateurs](#)[Les structures de contrôle](#)[Les structure de données](#)[Les fonctions](#)[Les fonctions prédéfinies](#)[Opérations sur les types et variables](#)[Opérations sur les chaînes](#)[T.P. voyelles et consonnes](#)[Opérations sur les tableaux](#)[T.P. occurrences](#)[Les fonctions utilisateurs](#)[T.P. fonction d'étoiles](#)[Valeurs et références](#)[T.P. fonction doubler](#)[T.P. fonction calculer](#)[Les erreurs](#)[Les fichiers](#)[Les expressions rationnelles](#)[PHP et HTML](#)[Petite application](#)[Mes cours](#)

ADMINISTRATION

[Administration du cours](#)[Réglages de mon profil](#)

Connecté sous le nom « [Arnaud Lemais](#) » ([Déconnexion](#))

[Intro PHP](#)