Fonctions PHP

Dans ce chapitre, vous apprendrez à créer vos propres fonctions personnalisées en PHP.

Fonctions intégrées PHP

Une fonction est un bloc de code autonome qui exécute une tâche spécifique.

PHP a une énorme collection de fonctions internes ou intégrées que vous pouvez appeler directement dans vos scripts PHP pour effectuer une tâche spécifique, comme gettype(), print_r(), var_dump, etc.

Fonctions PHP définies par l'utilisateur

En plus des fonctions intégrées, PHP vous permet également de définir vos propres fonctions. C'est un moyen de créer des packages de code réutilisables qui effectuent des tâches spécifiques et peuvent être conservés et maintenus séparément du programme principal. Voici quelques avantages de l'utilisation des fonctions :

- **Les fonctions réduit la répétition de code dans un programme** — Les fonctions vous permet d'extraire un bloc de code couramment utilisé en un seul composant. Vous pouvez maintenant effectuer la même tâche en appelant cette fonction où vous voulez dans votre script sans avoir à copier et coller le même bloc de code encore et encore.
- **Les fonctions rendent le code beaucoup plus facile à maintenir** - Puisqu'une fonction créée une fois peut être utilisée plusieurs fois, toute modification apportée à l'intérieur d'une fonction est automatiquement implémentée à tous les endroits sans toucher aux différents fichiers.
- **Les fonctions facilitent l'élimination des erreurs** - Lorsque le programme est subdivisé en fonctions, si une erreur se produit, vous savez exactement quelle fonction est à l'origine de l'erreur et où la trouver. Par conséquent, la correction des erreurs devient beaucoup plus facile.
- **Les fonctions peuvent être réutilisées dans d'autres applications** — Parce qu'une fonction est séparée du reste du script, il est facile de réutiliser la même fonction dans d'autres applications simplement en incluant le fichier php contenant ces fonctions.

La section suivante vous montrera avec quelle facilité vous pouvez définir votre propre fonction en PHP.

Créer et invoquer des fonctions

La syntaxe de base de la création d'une fonction personnalisée peut être donnée avec :

```
function nom_fonction(){
    // Code à exécuter
}
```

La déclaration d'une fonction définie par l'utilisateur commence par le mot function, suivi du nom de la fonction que vous souhaitez créer suivi de parenthèses, c'est-à-dire () et placez enfin le code de votre fonction entre accolades {}.

Voici un exemple simple d'une fonction définie par l'utilisateur, qui affiche Hello World ! :

```
<?php
//creation de function
function writeMsg() {
  echo "Hello world!";
}
//appel de function
writeMsg();
?>
```

Remarque : Un nom de fonction doit commencer par une lettre ou un caractère de soulignement et non par un chiffre, éventuellement suivi par d'autres lettres, chiffres ou caractères de soulignement.

Fonctions avec paramètres

Vous pouvez spécifier des **paramètres** lorsque vous définissez votre fonction pour accepter des valeurs d'entrée au moment de l'exécution. Les *paramètres* fonctionnent comme des *variables* d'espace réservé dans une fonction ; ils sont remplacés au moment de l'exécution par les valeurs (appelées **arguments**) fournies à la fonction au moment de l'invocation.

```
function myFunc($oneParameter, $anotherParameter){
    // Code à exécuter
}
```

Vous pouvez définir autant de paramètres que vous le souhaitez. Cependant, pour chaque paramètre que vous spécifiez, un argument correspondant doit être passé à la fonction lorsqu'elle est appelée.

La fonction getSum() dans l'exemple suivant prend deux valeurs entières comme arguments, il suffit de les additionner, puis d'afficher le résultat dans le navigateur.

```php
<?php
// Defining function
function getSum($num1, $num2){
  $sum = $num1 + $num2;
  echo "La somme des deux nombres $num1 et $num2 est : $sum";
}

// Calling function
getSum(10, 20);
?>
```

Astuce : Un argument est une valeur que vous transmettez à une fonction, et un paramètre est la variable dans la fonction qui reçoit l'argument. Cependant, dans l'usage courant, ces termes sont interchangeables, c'est-à-dire qu'un argument est un paramètre est un argument.

Fonctions avec paramètres facultatifs et valeurs par défaut

Vous pouvez également créer des fonctions avec des paramètres facultatifs - insérez simplement le nom du paramètre, suivi d'un = signe égal (), suivi d'une valeur par défaut, comme ceci.

```php
<?php
// Defining function
function customFont($font, $size=1.5){
    echo "<p style='font-family: $font; font-size: {$size}em;'>Hello, world!</p>";
}

// Calling function
customFont("Arial", 2);
customFont("Times", 3);
customFont("Courier");
?>
```

Comme vous pouvez le voir, le troisième appel à customFont() n'inclut pas le deuxième argument. Cela amène le moteur PHP à utiliser la valeur par défaut pour le paramètre $size qui est 1.5.

Renvoyer des valeurs à partir d'une fonction

Une fonction peut renvoyer une valeur au script qui a appelé la fonction à l'aide de l'instruction return. La valeur peut être de n'importe quel type, y compris des tableaux et des objets.

```php
<?php
// Defining function
function getSum($num1, $num2){
    $total = $num1 + $num2;
    return $total;
}

// Printing returned value
echo getSum(5, 10); // Outputs: 15
?>
```

Une fonction ne peut pas renvoyer plusieurs valeurs. Cependant, vous pouvez obtenir des résultats similaires en retournant un tableau, comme illustré dans l'exemple suivant.

```php
<?php
// Defining function
function divideNumbers($dividend, $divisor){
    $quotient = $dividend / $divisor;
    $array = array($dividend, $divisor, $quotient);
    return $array;
}

// Assigner les variables comme s'il s'agissait d'un tableau
list($dividend, $divisor, $quotient) = divideNumbers(10, 2);
echo $dividend . "<br>";  // Outputs: 10
echo $divisor  . "<br>";   // Outputs: 2
echo $quotient . "<br>";  // Outputs: 5
?>
```

Comprendre la portée variable

Vous pouvez déclarer les variables n'importe où dans un script PHP. Mais, l'emplacement de la déclaration détermine l'étendue de la visibilité d'une variable dans le programme PHP, c'est-à-dire l'endroit où la variable peut être utilisée ou accessible. Cette accessibilité est connue sous le nom de portée variable .

Par défaut, les variables déclarées dans une fonction sont locales et ne peuvent pas être visualisées ou manipulées depuis l'extérieur de cette fonction, comme illustré dans l'exemple ci-dessous :

```php
<?php
// Defining function
function test(){
    $greet = "Hello World!";
    echo $greet;
}
test(); // Outputs: Hello World!
echo $greet; // Generate undefined variable error
?>
```

De même, si vous essayez d'accéder ou d'importer une variable externe à l'intérieur de la fonction, vous obtiendrez une erreur de variable indéfinie, comme illustré dans l'exemple suivant :

```php
<?php
$greet = "Hello World!";
// Defining function
function test(){
    echo $greet;
}
test();  // Generate undefined variable error
echo $greet; // Outputs: Hello World!
?>
```

Comme vous pouvez le voir dans les exemples ci-dessus, la variable déclarée à l'intérieur de la fonction n'est pas accessible de l'extérieur, de même la variable déclarée à l'extérieur de la fonction n'est pas accessible à l'intérieur de la fonction. Cette séparation réduit les chances que les variables d'une fonction soient affectées par les variables du programme principal.

Astuce : Il est possible de réutiliser le même nom pour une variable dans différentes fonctions, car les variables locales ne sont reconnues que par la fonction dans laquelle elles sont déclarées.

Le mot-clé global

Il peut arriver que vous ayez besoin d'importer une variable du programme principal dans une fonction, ou vice versa. Dans de tels cas, vous pouvez utiliser le mot-clé **global** avant les variables à l'intérieur d'une fonction. Ce mot clé transforme la variable en variable globale, la rendant visible ou accessible à la fois à l'intérieur et à l'extérieur de la fonction, comme le montre l'exemple ci-dessous :

```php
<?php
$greet = "Hello World!";
// Defining function
function test(){
    global $greet;
    echo '<p>$greet l\'intérieur de la fonction est: ' . $greet . '</p>';
}
test();
echo '<p>$greet l\'extérieur de la fonction est: ' . $greet . '</p>';
// Assign a new value to variable
$greet = "Goodbye";
test();
echo '<p>$greet à l\'extérieur de la fonction est: ' . $greet . '</p>';
?>
```