PHP 05 - Les fonctions

Définition

Une fonction est une partie de code PHP qui peut remplir n'importe quelle tâche. Dans ce code, peut figurer toute instruction PHP valide.

Une fonction est destinée :

- à effectuer une tâche spécifique prédéfinie.
- à être utilisé plusieurs fois au sein d'une application (factorisation de code)

On rencontre 2 types de fonctions :

- Les fonctions **natives** = celles proposées par défaut par le langage; en PHP il en existe des centaines.
- Les fonctions **utilisateurs** = celles que le développeur écrit, et qui peuvent inclure des fonctions natives.

L'utilisation des fonctions présente de nombreux avantages :

- une réutilisabilité du code,
- la non répétition d'une séquence de code, le contraire occasionnant souvent d'importants dysfonctionnements.
- une meilleure lisibilité du code source, ainsi qu'une maintenance facilitée.
- un gain de productivité

Les fonctions utilisateurs

Déclaration d'une fonction

Pour déclarer (= définir) une fonction, on utilisera le mot-clé `function` suivi du nom de la fonction (à vous de lui donner un nom) et enfin de parenthèses (obligatoires, mêmes si vides).

Exemple :

```
<?php
function bonjour()
{
    echo "Bonjour";
}
```

On appellera donc la fonction de la manière suivante :

```
<?php
bonjour();
```

Arguments

On peut ajouter un nom ou prénom comme ceci :

```
<?php
function bonjour($prenom)
{
    echo "Bonjour ".$prenom;
}
```

On appellera la fonction de la façon suivante :

```
<?php
bonjour("Dave");
```

Cette fonction affichera : *Bonjour Dave.*

Une fonction peut recevoir un ou plusieurs arguments (appelés aussi paramètres) en entrée, écrits dans les parenthèses.

Un paramètres sont des variables transmis en entrée à la fonction qui en a besoin pour effectuer des opérations en son sein.

Exemple :

```
<?php
function bonjour($prenom, $nom)
{
    echo "Bonjour ".$prenom." ".$nom;
}
```

Cette fonction affichera *donc Bonjour Dave LOPER.*

Les paramètres peuvent être de tout type :

- variables simples (chaînes, entiers...)
- tableaux
- objets (que nous n'avons pas encore abordés)

Les arguments peuvent être directement une valeur soit une variable :

```
<?php
$prenom = "Dave";
$nom = "Loper";

bonjour($prenom, $nom);
```

Renvoyer des informations

Les fonctions peuvent retourner un résultat en sortie : valeur, booléen, tableau, objet... On indique ce retour par l'instruction `return` suivi de la variable à retourner.

```
<?php
function addition($entier1, $entier2)
{
    $resultat = $entier1 + $entier2;
    return $resultat;
}
```

Pour *capter* le retour de la fonction, il faut l'*affecter* à une variable lors de l'appel à la fonction :

```
$resultat = addition(1, 2);
echo $resultat; // affichera 3
```

Si on veut juste afficher le retour, on peut écrire directement :

```
echo addition(1, 2);
```

De même, dans la fonction, on peut renvoyer le résultat directement :

```
<?php
function addition($entier1, $entier2)
{
    return $entier1 + $entier2;
}
```

Une fonction peut avoir plusieurs instructions `return`, par exemple lorsqu'il y a des conditions, **mais il faut bien retenir qu'on sort de la fonction dès qu'un `return` est validé, c'est-à-dire que le code situé après n'est pas exécuté :**

```
<?php
$age = 19;
function isMineur($age)
{
    if ($age < 18)
    {
        return "mineur";
    }
    else
    {
        return "majeur";
    }
}
```

Dans cet exemple, la fonction renverra *majeur*.

L'instruction `return` ne peut renvoyer qu'une variable/valeur à la fois. Lorsqu'on souhaite retourner plusieurs valeurs, on peut "fricher" en renvoyant un tableau contenant les différentes valeurs à renvoyer :

```
<?php
function calculPrix($prix_ht, $remise)
{
    $base_ht = $prix_ht - $remise;
    $prix_ttc = $base_ht * 1.20;
    $retour = array($base_ht, $prix_ttc);

    return $retour;
}
$retour = function calculPrix(110, 10);

echo" Base HT : ".$retour["base_ht"]." Euro\n"; // affiche 100 €
echo" Prix TTC : ".$retour["prix_ttc"]." Euro\n"; // affiche 120 €
```

Les librairies de fonctions

Une librairie de fonctions est un ensemble de fonctions regroupées dans un seul fichier.

En fait, les fonctions que nous créons ne sont disponibles que dans le programme dans lequel elles se trouvent. Ainsi, pour les utiliser dans un autre programme, il faudrait les recopier dans celui-ci.

Mais, il existe bien sur un moyen de rendre les fonctions disponibles dans tous nos programmes.

Il suffit pour cela, de regrouper vos fonctions dans un fichier et d'ajouter une ligne à tous vos programmes afin d'inclure ce fichier et ainsi rendre disponible toutes les fonctions pour ce programme.

En pratique, il faut créer un fichier contenant toutes les fonctions.

Concrètement :

```
<?php
function 1 ...

function 2 ...

?>
```

Enregistrer le fichier sous un nom quelconque que vous choisirez, *par exemple : lib.inc.php*

Vous pourrez ajouter à ce fichier autant de fonctions que vous aurez besoin pour vos applications. Ensuite, nous ajouterons l'instruction require dans chaque programme où nous aurons besoin des fonctions :

```
<?php
require('lib.inc.php');
... suite de l'application...
```

Les fonctions natives PHP

Les fonctions arithmétiques

Le langage PHP offre un jeu de fonctions mathématiques assez complet qui couvre largement les besoins que l'on peut avoir pour réaliser une application. Composition :

- opérateurs mathématiques (addition, soustraction, multiplication, division, pussarnce...)
- arrondis, valeur absolue, minimum, maximum, ...
- cosinus, sinus, tangente, logarithmes...
- génération de nombres aléatoires
- conversions (binaire, décimal, hexadécimal...)
- formatage de nombres (en chaîne de caractères, ...)

La gestion des chaînes de caractères

PHP fournit un ensemble complet de fonctions pour gérer les chaînes de caractères :

- comparer des chaînes de caractères
- définir la longueur d'une chaîne de caractères
- découper une chaîne, extraire des parties
- crypter des chaînes/données
- remplacer des parties de chaînes
- rechercher des composants d'une chaîne de caractères
- convertir des caractères spécifiques en code HTML

La gestion des fichiers et répertoires

Les fonctions de gestion de fichiers et de répertoires fournies par PHP vous permettront :

- de créer, modifier, supprimer, renommer et déplacer des fichiers de type (texte, csv, images, PDF,...)
- de créer, modifier (nom), supprimer, renommer et déplacer des répertoires
- de lire, ajouter, modifier et supprimer du contenu d'un fichier
- de lire et modifier les informations de métadonnées (dates de création/modification/accès, nom, type, poids, droits d'exécution...) d'un fichier ou d'un répertoire

Les fonctions de génération d'images dynamiques

Une librairie, nommée GD (pour *Gif Draw*), de fonctions de création et de manipulation d'images est fournie avec PHP.

GD permet de nombreuses possibilités :

- générer dynamiquement des images (formes géométriques, miniatures, graphiques...)
- gestion des couleurs dans une image
- transmettre des informations sur une image (largeur x hauteur),
- découper et faire pivoter une image (rotation selon un angle précis)
- incrustation de texte, superposition/fusion d'images (par exemple, ajouter un logo dans une image pour montrer qu'elle provient d'un site)
- manipuler les différents formats d'images (gif, png, jpeg, tiff, iaw...)

Les bases de GD et quelques exemples

Les fonctions spécifiques à Internet

Le PHP, conçu pour Internet, propose bien sûr un groupe de fonctions spécifiques à celui-ci :

- fonctions de traitement d'URLs
- fonctions de lecture d'une page HTML (avec la librairie *CURL* notamment)
- la protection d'accès à une page web : sessions, cryptage de mots de passe que nous aborderons plus tard dans le programme
- envoi de mails (séquence complète)
- fonctions de gestion de cookies

Quelques fonctions utiles

Fonction	Description	Code PHP	Résultat
`addslashes()`	Ajoute des anti-slashes devant les caractères spéciaux	`$s = addslashes("L'a");`	*L'a*
`stripslashes()`	Retire les antislashes devant les caractères spéciaux	`$s = stripslashes("L\'a");`	*L'a*
`ceil()`	Retourne le nombre entier supérieur ici (12.1)	`$i = ceil("12.1");`	13
`chunk_split()`	Permet de scinder une chaîne en plusieurs morceaux	`$s = chunk_split("DGF0F","2","-");`	*DG-DF-F-*
`htmlentities()`	Remplace les caractères par leur équivalent HTML (si ils existent)	`$s = htmlentities("&");`	*&*
`strtr()`	Recherche le premier caractère 'p' dans la chaîne et affiche le reste de la chaîne y compris le 'p'	`$s = strtr ("webmaster@php.cdsi.fr.st","p");`	*php.cdsi.fr.st*
`strlen()`	Retourne la longueur de la chaîne	`$s = strlen("Lachaînedecaractères");`	*20*
`strtolower()`	Passe tout les caractères en minuscules	`$s = strtolower("LA CHAINE DE CARACTERES");`	*la chaine de caractères*
`strtoupper()`	Passe tout les caractères en majuscules	`$s = strtoupper("LA CHAINE DE CARACTERES");`	*LA CHAINE DE CARACTERES*
`str_replace()`	Remplace un caractère par un autre dans une chaîne (tient compte de la casse)	`$s = str_replace("a","b","Lalala");`	*Lolalo*
`trim()`	Efface les espaces dits blancs (espaces, sauts et retours de lignes au début et à la fin d'une chaîne	`$s = trim(" Bonjour le monde ");`	*Bonjour le monde*
`ucfirst()`	Met la première lettre d'une chaîne en majuscule	`$s = ucfirst("bonjour le monde");`	*Bonjour le monde*
`ucwords()`	Met la première lettre de chaque mot d'une chaîne en majuscule	`$s = ucwords("bonjour le monde");`	*Bonjour Le Monde*
`strpos()`	Recherche la position du premier caractère trouvé. Retourne le nombre de caractères placés avant lui (ici 4).	`$s = strpos("ABCDEF","E");`	*4*
`preg_match()`	Recherche si une chaîne de caractère est contenue dans une autre (ex. recherche si ABCDE contient BCD). Peut rechercher via des expressions régulières	`if (preg_match ("BCD","ABCDEF")) { echo "Trouvé"; }`	Renvoie 1 (TRUE) si trouvé, 0 (FALSE) sinon

Exercice

Ecrivez la fonction `calculator()` traitant des opérations d'addition, de soustraction, de multiplication et de division suite à une saisie utilisateur.