

# Chapitre 5



318

## Langage PHP



# Plan du chapitre 5

319

- ☐ Introduction
- ☐ Intégration du code PHP
- ☐ Cycle de vie d'une page PHP
- ☐ Commentaires
- ☐ Variables
- ☐ Constantes
- ☐ Types de données
- ☐ Détermination du type d'une variable
- ☐ Conversion de type
- ☐ Contrôle de l'état d'une variable
- ☐ Opérateurs numériques



# Plan du chapitre 5

320

- ☐ Fonctions mathématiques
- ☐ Opérateurs booléens
- ☐ Instructions conditionnelles
- ☐ Instructions de boucle
- ☐ Gestion des erreurs
- ☐ Chaînes de caractères
- ☐ Tableaux
- ☐ Formulaires
- ☐ Fonctions
- ☐ Dates



# Plan du chapitre 5

321

- Fichiers
- Sessions
- L'envoi des e-mails
- Accès à une base MySQL avec PHP



# Partie I :

## Les bases du langage PHP



# Plan du chapitre 5

323

- **Introduction**
- Intégration du code PHP
- Cycle de vie d'une page PHP
- Commentaires
- Variables
- Constantes
- Types de données
- Détermination du type d'une variable
- Conversion de type
- Contrôle de l'état d'une variable
- Opérateurs numériques



# Introduction

324

- ☐ PHP signifiait à l'origine **Personal Home Page**. Aujourd'hui, il signifie **Php Hypertext Preprocessor**.
- ☐ Un langage de script libre dédié à Internet, directement inclus dans les pages Web.
- ☐ PHP permet de créer des pages Web dynamiques et interactives via un serveur HTTP.
- ☐ La syntaxe du PHP provient de celle du langage C, du Perl et de Java.
- ☐ Différentes versions du PHP ont été développées, depuis son apparition en 1994 :
  - 1998 : version 3.0 ;
  - 1999 : version 4.0 ;
  - 2004 : version 5.0 ;
  - Aujourd'hui, on parle de PHP 5.3 ;
- ☐ L'exécution des scripts PHP est déléguée à un composant indépendant installé sur le serveur Web souvent appelé moteur (PHP5 utilise Zend Engine).



# Introduction

325

- Les principaux avantages du PHP sont :
  - Sa facilité d'apprentissage ;
  - Sa simplicité d'écriture ;
  - Sa souplesse d'utilisation ;
  - Sa très grande richesse fonctionnelle vis-à-vis de la connexion à des bases de données ;
  - Sa compatibilité avec différents serveurs Web (Apache, Microsoft IIS, ...) ;
  - Sa disponibilité pour différentes plateformes (Linux, Windows, MAC...) ;
  - Sa gratuité (tous ses logiciels sont open source) ;
- PHP, MySQL et Apache forment le trio ultradominant sur les serveurs Web. On parle de système LAMP pour les serveurs à Linux, WAMP pour Windows et MAMP pour MAC.





# Plan du chapitre 5

326

- Introduction
- **Intégration du code PHP**
- Cycle de vie d'une page PHP
- Commentaires
- Variables
- Constantes
- Types de données
- Détermination du type d'une variable
- Conversion de type
- Contrôle de l'état d'une variable
- Opérateurs numériques



# Intégration du code PHP

327

- ☐ Le code PHP est toujours incorporé dans du code HTML.
- ☐ Les pages Web contenant des scripts PHP sont enregistrées avec l'extension **.php**.
- ☐ Il est possible d'inclure autant de scripts PHP indépendants que l'on souhaite, n'importe où dans du code HTML.
- ☐ Un code PHP est délimité par :
  - `<?php` et `?>` (la méthode la plus utilisée)
  - `<?=` et `?>` (forme courte nécessitant l'activation de la directive **short open tag** dans le fichier de configuration de PHP5 (fichier `php.ini`)).
  - `<script language= "php">` et `</script>` (rarement utilisé)
- ☐ Les scripts PHP peuvent être écrits :
  - directement dans le code HTML ;
  - dans des fichiers externes enregistrés avec l'extension **.inc** ou **.inc.php** et qui seront incorporés par la suite dans le code HTML en fonction des besoins ;



# Intégration du code PHP

328

- ☐ Les fonctions permettant l'inclusion d'un fichier externe dans du code PHP :

Fonction	Description
<code>include("nom_fichier.ext")</code>	Lors de son interprétation par le serveur, cette ligne est remplacée par tout le contenu du fichier précisé en paramètre, dont vous fournissez le nom et éventuellement l'adresse complète. En cas d'erreur, par exemple si le fichier n'est pas trouvé, <code>include()</code> ne génère qu'une alerte, et le script continue.
<code>require("nom_fichier.ext")</code>	A désormais un comportement identique à <code>include()</code> , à la différence près qu'en cas d'erreur, <code>require()</code> provoque une erreur fatale et met fin au script.
<code>include_once("nom_fichier.ext")</code> <code>require_once("nom_fichier.ext")</code>	Contrairement aux deux précédentes, ces fonctions ne sont pas exécutées plusieurs fois, même si elles figurent dans une boucle ou si elles ont déjà été exécutées une fois dans le code qui précède.



# Intégration du code PHP

329

## Exemple

### ☐ Insertion directe du code PHP :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>

<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <title>Une page PHP</title>
</head>
<body>
  <?php
    echo "<h3> Aujourd'hui le ". date('d / M / Y H:m:s')."</h3><hr />";
    echo "<h2>Bienvenue sur le site PHP 5</h2>";
  ?>
</body>
</html>
```



# Intégration du code PHP

330

## Exemple

### ☐ Inclusion d'un code externe :

*fichier corps.inc*

```
<?php
echo "<h3> Aujourd'hui le ". date('d / M / Y H:m:s '). "</ h3><hr />";
echo "<h2>Bienvenue sur le site PHP 5</h2>";
?>
```

*fichier principal.php*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <title>Une page PHP</title>
  </head>
  <body>
    <?php
    include("corps.inc");
    ?>
  </body>
</html>
```



# Plan du chapitre 5

331

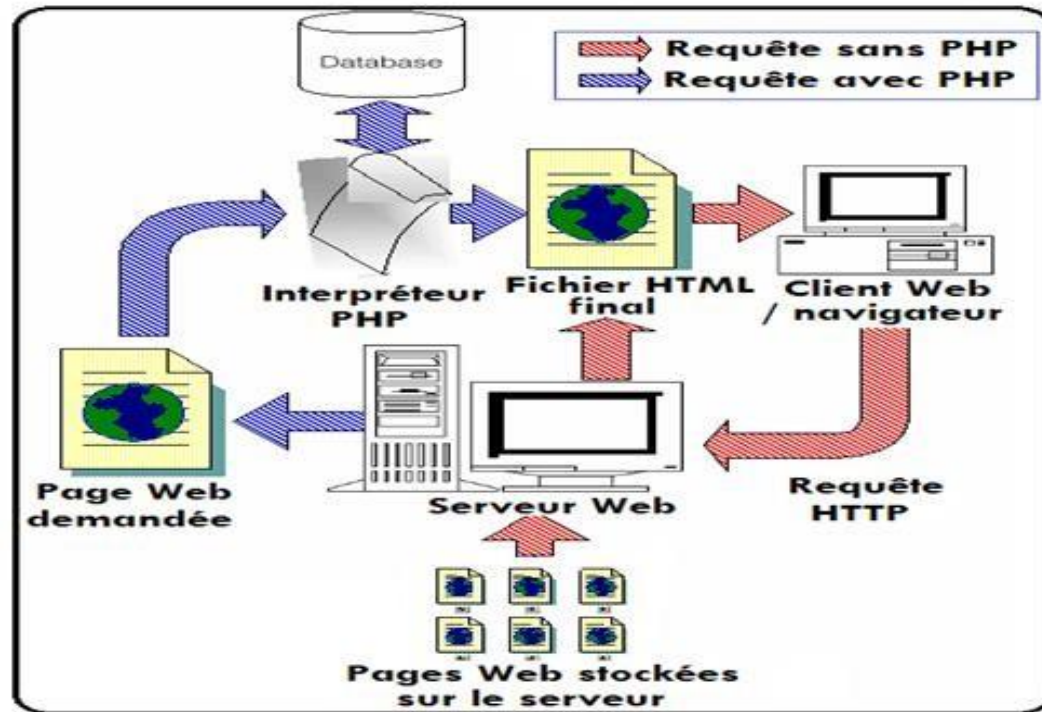
- ☐ Introduction
- ☐ Intégration du code PHP
- ☐ **Cycle de vie d'une page PHP**
- ☐ Commentaires
- ☐ Variables
- ☐ Constantes
- ☐ Types de données
- ☐ Détermination du type d'une variable
- ☐ Conversion de type
- ☐ Contrôle de l'état d'une variable
- ☐ Opérateurs numériques



# Cycle de vie d'une page PHP

332

- ☐ Envoi d'une requête HTTP par le navigateur client vers le serveur.  
<http://www.monserveur.com/codePHP.php>
- ☐ Interprétation par le serveur du code PHP contenu dans la page demandée.
- ☐ Envoi par le serveur Web d'un fichier dont le contenu est purement HTML.





# Plan du chapitre 5

333

- ☐ Introduction
- ☐ Intégration du code PHP
- ☐ Cycle de vie d'une page PHP
- ☐ **Commentaires**
- ☐ Variables
- ☐ Constantes
- ☐ Types de données
- ☐ Détermination du type d'une variable
- ☐ Conversion de type
- ☐ Contrôle de l'état d'une variable
- ☐ Opérateurs numériques





# Commentaires

334

□ PHP supporte les trois syntaxes de commentaires suivantes :

- Commentaires sur une seule ligne introduits par les caractères // :  
// Ceci est un commentaire sur une seule ligne.
- Commentaires sur plusieurs lignes introduits par les caractères /\* et \*/ :

/\* Ceci est un commentaire  
Multi-ligne. \*/

- Commentaires de type UNIX ne comportant qu'une seule ligne introduite par le caractère # :

```
#####  
# Ceci est un commentaire de type UNIX.  
#####
```



# Plan du chapitre 5

335

- ☐ Introduction
- ☐ Intégration du code PHP
- ☐ Cycle de vie d'une page PHP
- ☐ Commentaires
- ☐ **Variables**
- ☐ Constantes
- ☐ Types de données
- ☐ Détermination du type d'une variable
- ☐ Conversion de type
- ☐ Contrôle de l'état d'une variable
- ☐ Opérateurs numériques



# Variables

336

## Déclaration

- ☐ Chaque variable possède un identifiant particulier commençant par le caractère \$ suivi du nom de la variable.
- ☐ Les règles de création des noms de variables sont :
  - Le nom doit commencer par un caractère alphabétique ou par le caractère (\_);
  - La longueur du nom n'est pas limitée ;
  - Le nom de la variable doit être significatif ;
  - Les variables peuvent être déclarées n'importe où dans le script à condition qu'elles soient définies avant d'être appelées ;
  - L'initialisation des variables n'est pas obligatoire et une variable non initialisée n'a pas de type précis.
  - Les noms des variables sont sensibles à la casse.



# Variables

337

## Affectation

- Le type de la variable est déterminé selon la valeur qui lui est affectée.
- On distingue deux types d'affectation pour une variable donnée :
  - **Affectation par valeur :**
    - `$nomVariable=expression ;`
    - **Exemple :**  
`$age=85 ;`
  - **Affectation par référence :**
    - `$nomVariable2=&$nomVariable1 ;`
    - **Exemple :**  
`$employe1="Ahmed" ;`  
`$employe2=&$employe1 ;`
    - La variable `$employe2` devient un alias de la variable `$employe1`. Ainsi, les modifications opérées sur l'une des deux variables seront répercutées sur l'autre.



# Variables

338

## Opérateurs d'affectation combinée

Opérateur	Description
<b>+=</b>	Addition puis affectation : $x += y$ équivaut à $x = x + y$ $y$ peut être une expression complexe dont la valeur est un nombre.
<b>-=</b>	Soustraction puis affectation : $x -= y$ équivaut à $x = x - y$ $y$ peut être une expression complexe dont la valeur est un nombre.
<b>*=</b>	Multiplication puis affectation : $x *= y$ équivaut à $x = x * y$ $y$ peut être une expression complexe dont la valeur est un nombre.
<b>/=</b>	Division puis affectation : $x /= y$ équivaut à $x = x / y$ $y$ peut être une expression complexe dont la valeur est un nombre différent de 0.
<b>%=</b>	Modulo puis affectation : $x %= y$ équivaut à $x = x \% y$ $y$ peut être une expression complexe dont la valeur est un nombre.
<b>.=</b>	Concaténation puis affectation : $x .= y$ équivaut à $x = x . y$ $y$ peut être une expression littérale dont la valeur est une chaîne de caractères.



# Variables

339

## Variables prédéfinies

- ☐ PHP dispose d'un grand nombre de variables prédéfinies.
- ☐ Les variables prédéfinies stockent des informations sur le serveur et sur toutes les données pouvant transiter entre le client et le serveur Web (exp : les valeurs saisies dans un formulaire, les cookies...).
- ☐ Les variables prédéfinies se présentent sous la forme de tableaux appelés **superglobaux** qui sont accessibles en tout point de n'importe quel script.

[Voir Tableau 1](#)



# Plan du chapitre 5

340

- ☐ Introduction
- ☐ Intégration du code PHP
- ☐ Cycle de vie d'une page PHP
- ☐ Commentaires
- ☐ Variables
- ☐ **Constantes**
- ☐ Types de données
- ☐ Détermination du type d'une variable
- ☐ Conversion de type
- ☐ Contrôle de l'état d'une variable
- ☐ Opérateurs numériques



# Constantes

341

## Constantes personnalisées

- Une constante personnalisée est définie grâce à la fonction **define()** :  

```
boolean define (string nom_cte, divers valeur_cte, boolean casse) ;
```

  - **nom\_cte** : nom de la constante ;
  - **valeur\_cte** : valeur de la constante ;
  - **casse** : vaut **TRUE** si le nom de la constante est insensible à la casse et **FALSE** sinon.
  - La fonction **define()** retourne **TRUE** si la constante a bien été définie et **FALSE** en cas de problème.
- La fonction **defined(string nom\_cte)** permet de vérifier l'existence d'une constante nommée **nom\_cte**. Elle retourne **TRUE** si la constante existe déjà et **FALSE** sinon.





# Constantes

342

## Constantes personnalisées

### Exemple :

```
<?php
//définition insensible à la casse
define("PI",3.1415926535,TRUE);
//Utilisation
echo "La constante PI vaut ",PI,"<br />";
echo "La constante PI vaut ",pi,"<br />";
//Vérification de l'existence
if (defined( "PI")) echo "La constante PI est déjà définie","<br />";
if (defined( "pi")) echo "La constante pi est déjà définie","<br />";
//définition sensible à la casse, vérification de l'existence et utilisation
if(define("site","http://www.funhtml.com",FALSE))
{
    echo "<a href=\" " .site, " \">>Lien vers mon site </ a>";
}
?>
```

## Constantes prédéfinies

### Quelques constantes prédéfinies

PHP_VERSION	Version de PHP installée sur le serveur
PHP_OS	Nom du système d'exploitation du serveur
DEFAULT_INCLUDE_PATH	Chemin d'accès aux fichiers par défaut
__FILE__	Nom du fichier en cours d'exécution
__LINE__	Numéro de la ligne en cours d'exécution



# Plan du chapitre 5

344

- ☐ Introduction
- ☐ Intégration du code PHP
- ☐ Cycle de vie d'une page PHP
- ☐ Commentaires
- ☐ Variables
- ☐ Constantes
- ☐ **Types de données**
- ☐ Détermination du type d'une variable
- ☐ Conversion de type
- ☐ Contrôle de l'état d'une variable
- ☐ Opérateurs numériques



# Types de données

345

- En PHP, il n'existe pas de déclaration explicite du type d'une variable lors de sa création.
- PHP permet la manipulation d'un certain nombre de types de données différents :
  - Les types scalaires de base :
    - Entiers, avec le type **integer** : représentation des nombres entiers dans les bases 10, 8 et 16.  
Les entiers sont codés sur 32 bits. L'intervalle de valeurs des entiers est de  $-2^{31}$  à  $2^{31} - 1$ . Si une opération mathématique sur une variable entière l'amène à contenir une valeur en dehors de cet intervalle, elle est automatiquement convertie en type double et conserve sa nouvelle valeur.
    - Flottants, avec le type **double** ou **float** : représentation des nombres réels.  
Les réels sont codés sur 32 bits. Le type double permet de représenter l'ensemble des nombres décimaux avec une précision de 14 chiffres.
    - Chaînes de caractères, avec le type **string**.
    - Booléens, avec le type **boolean** : contient les valeurs de vérité **TRUE** ou **FALSE**.



# Types de données

346

- Les types composés :
  - Tableaux, avec le type **array**.
  - Objets, avec le type **object**.
- Les types spéciaux :
  - Type **resource** :
    - Représente une référence à des informations présentes sur le serveur.
    - Il est le type retourné par certaines fonctions particulières.

**Exemple :** les fonctions utilisées pour accéder à une base de données lors de la connexion, qui retournent une valeur de type resource permettant d'identifier chaque connexion initiée par un utilisateur afin d'être utilisée pour retourner les données après interrogation de la base par l'utilisateur concerné.
  - Type **NULL** :
    - Le type NULL (ou null) est attribué à une variable qui n'a pas de contenu ou qui a été explicitement initialisée avec la valeur NULL.
    - **N.B :** Une variable contenant une chaîne vide ou la valeur "0" n'a pas le type NULL mais string. De même, une variable contenant la valeur 0 est du type integer.



# Plan du chapitre 5

347

- ☐ Introduction
- ☐ Intégration du code PHP
- ☐ Cycle de vie d'une page PHP
- ☐ Commentaires
- ☐ Variables
- ☐ Constantes
- ☐ Types de données
- ☐ **Détermination du type d'une variable**
- ☐ Conversion de type
- ☐ Contrôle de l'état d'une variable
- ☐ Opérateurs numériques



# Détermination du type d'une variable

348

<b>string</b> <code>gettype(\$nom_variable)</code>	Elle permet de déterminer le type d'une variable. Elle retourne une chaîne de caractères contenant le type de la variable.
<b>boolean</b> <code>is_integer(\$nom_variable)</code> ou <code>is_int(\$nom_variable)</code>	Elle retourne TRUE si la variable est un entier, FALSE sinon.
<b>boolean</b> <code>is_double(\$nom_variable)</code>	Elle retourne TRUE si la variable est un double, FALSE sinon.
<b>boolean</b> <code>is_string(\$nom_variable)</code>	Elle retourne TRUE si la variable est une chaîne de caractères, FALSE sinon.
<b>boolean</b> <code>is_bool(\$nom_variable)</code>	Elle retourne TRUE si la variable est un booléen, FALSE sinon.
<b>boolean</b> <code>is_array(\$nom_variable)</code>	Elle retourne TRUE si la variable est un tableau, FALSE sinon.
<b>boolean</b> <code>is_object(\$nom_variable)</code>	Elle retourne TRUE si la variable est un objet, FALSE sinon.
<b>boolean</b> <code>is_resource(\$nom_variable)</code>	Elle retourne TRUE si la variable est de type resource, FALSE sinon.
<b>boolean</b> <code>is_null(\$nom_variable)</code>	Elle retourne TRUE si la variable est de type null, FALSE sinon.



# Plan du chapitre 5

349

- ☐ Introduction
- ☐ Intégration du code PHP
- ☐ Cycle de vie d'une page PHP
- ☐ Commentaires
- ☐ Variables
- ☐ Constantes
- ☐ Types de données
- ☐ Détermination du type d'une variable
- ☐ **Conversion de type**
- ☐ Contrôle de l'état d'une variable
- ☐ Opérateurs numériques





# Conversion de type

350

- ❑ Convertir une variable d'un type dans un autre :

`$resultat = (type_désiré) $nom_variable`

- ❑ Exemple :

```
<?php
$var="3.52 kilomètres";
$var2 = (double) $var;
echo "\$var2= ",$var2,"<br />";//affiche "$var2=3.52"
$var3 = (integer) $var2;
echo "\$var3= ",$var3,"<br />";//affiche "$var3=3"
?>
```

- ❑ Modifier le type de la variable elle-même :

`Boolean settype ($nom_variable, "type_désiré")`

Elle retourne TRUE si l'opération est réalisée et FALSE dans le cas contraire.

- ❑ Exemple :

```
<?php
$var="3.52 kilomètres";
settype($var,"double");
echo "\$var= ",$var,"<br />";//affiche "$var=3.52"
settype($var,"integer");
echo "\$var= ",$var,"<br />";//affiche "$var=3"
?>
```



# Plan du chapitre 5

351

- Introduction
- Intégration du code PHP
- Cycle de vie d'une page PHP
- Commentaires
- Variables
- Constantes
- Types de données
- Détermination du type d'une variable
- Conversion de type
- **Contrôle de l'état d'une variable**
- Opérateurs numériques

# Contrôle de l'état d'une variable



352

<b>boolean isset(\$nom_variable)</b>	Retourne la valeur FALSE si la variable n'est pas initialisée ou a la valeur NULL, et la valeur TRUE si elle a une valeur quelconque.
<b>boolean empty(\$nom_variable)</b>	Retourne la valeur TRUE si la variable n'est pas initialisée, a la valeur 0 ou NULL ou la chaîne "0", et la valeur FALSE si elle a une valeur quelconque.

## Exemple :

```
<?php
$a=null;
if(isset($a)){echo "\$a existe déjà<br />";}
else {echo "\$a n'existe pas<br />";}
if(empty($a)){echo "\$a est vide <br />";}
else {echo "\$a a la valeur $a<br />";}
//Affiche "$a n'existe pas" et "$a est vide"
$b=0;
if(isset($b)){echo "\$b existe déjà<br />";}
else {echo "\$b n'existe pas<br />";}
if(empty($b)){echo "\$b est vide <br />";}
else {echo "\$b a la valeur $b<br />";}
//Affiche "$b existe déjà" et "$b est vide"
$c=1;
if(isset($c)){echo "\$c existe déjà<br />";}
else {echo "\$c n'existe pas<br />";}
if(empty($c)){echo "\$b est vide <br />";}
else {echo "\$c a la valeur $c<br />";}
//Affiche "$c existe déjà" et "$c a la valeur 1"
?>
```



# Plan du chapitre 5

353

- ☐ Introduction
- ☐ Intégration du code PHP
- ☐ Cycle de vie d'une page PHP
- ☐ Commentaires
- ☐ Variables
- ☐ Constantes
- ☐ Types de données
- ☐ Détermination du type d'une variable
- ☐ Conversion de type
- ☐ Contrôle de l'état d'une variable
- ☐ **Opérateurs numériques**



# Opérateurs numériques

354

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	<p>Modulo : reste de la division du premier opérande par le deuxième. Fonctionne aussi avec des opérandes décimaux. Dans ce cas, PHP ne tient compte que des parties entières de chacun des opérandes.</p> <pre>\$var = 159; echo \$var%7; //affiche 5 car 159=22x7 + 5. \$var = 10.5; echo \$var%3.5; //affiche 1et non pas 0.</pre>
--	<p>Décrémentement : soustrait une unité à la variable. Il existe deux possibilités, la prédécrémentation, qui soustrait avant d'utiliser la variable, et la postdécrémentation, qui soustrait après avoir utilisé la variable.</p> <pre>\$var=56; echo \$var--; //affiche 56 puis décrémente \$var. echo \$var; //affiche 55. echo --\$var; //décrémente \$var puis affiche 54.</pre>
++	<p>Incrémentement : ajoute une unité à la variable. Il existe deux possibilités, la préincrémentation, qui ajoute 1 avant d'utiliser la variable, et la postincrémentation, qui ajoute 1 après avoir utilisé la variable.</p> <pre>\$var=56; echo \$var++; //affiche 56 puis incrémente \$var. echo \$var; //affiche 57. echo ++\$var; //incrémente \$var puis affiche 58.</pre>



# Plan du chapitre 5

355

- **Fonctions mathématiques**
- Opérateurs booléens
- Instructions conditionnelles
- Instructions de boucle
- Gestion des erreurs
- Chaînes de caractères
- Tableaux
- Formulaires
- Fonctions
- Dates



# Fonctions mathématiques

356

**N.B :** les noms des fonctions ne sont pas sensibles à la casse.

[Voir Tableau 2](#)



# Plan du chapitre 5

357

- Fonctions mathématiques
- **Opérateurs booléens**
- Instructions conditionnelles
- Instructions de boucle
- Gestion des erreurs
- Chaînes de caractères
- Tableaux
- Formulaires
- Fonctions
- Dates





# Opérateurs booléens

358

☐ Les opérateurs booléens servent à écrire des expressions simples ou complexes qui sont évaluées par une valeur booléenne **TRUE** ou **FALSE**.

☐ On distingue deux types d'opérateurs booléens :

- Opérateurs de comparaison ;

[Voir Tableau 3](#)

- Opérateurs logiques ;

[Voir Tableau 4](#)



# Plan du chapitre 5

359

- Fonctions mathématiques
- Opérateurs booléens
- **Instructions conditionnelles**
- Instructions de boucle
- Gestion des erreurs
- Chaînes de caractères
- Tableaux
- Formulaires
- Fonctions
- Dates



# Instructions conditionnelles

360

## Instruction if :

<pre>if(expression) {     //bloc d'instructions ; }</pre>	<pre>&lt;?php \$a=6; if(is_integer(\$a) &amp;&amp; (\$a&lt;10 &amp;&amp; \$a&gt;5) &amp;&amp; (\$a%2==0) ) {     echo "Conditions satisfaites"; } ?&gt;</pre>
---	---

## Instruction if... else :

<pre>if(expression1) {     //Bloc 1 } elseif(expression2) {     //Bloc 2 } else {     //Bloc 3 }</pre>	<pre>&lt;?php \$prix=55; if(\$prix&gt;100) {     echo "la remise est de 10 %"; } else {     echo "la remise est de 5 %"; } ?&gt;</pre>
--	--



# Instructions conditionnelles

361

## Opérateur ?

<code>\$var = expression ? valeur1 : valeur2</code>	<code>\$var = (\$prix&gt;100)? "la remise est de 10 %":"la remise est de 5 %";</code>
---	---

## Instruction switch... case :

<pre>switch(expression) {     case valeur1:         //bloc d'instructions 1;         break;     case valeur2:         //bloc d'instructions 2;         break;     .....     case valeurN:         //bloc d'instructions N;         break;     default:         //bloc d'instructions par défaut;         break; }</pre>	<pre>&lt;?php \$dept=75; switch(\$dept) {     //Premier cas     case 75:         echo "Paris";         break;     //Deuxième cas     case 78:         echo "Hauts de Seine";         break;     //Troisième cas     case 93:         echo "Seine Saint Denis";         break;     //Cas par défaut     default:         echo "Département inconnu en Ile de France";         break; } ?&gt;</pre>
---	---



# Plan du chapitre 5

362

- Fonctions mathématiques
- Opérateurs booléens
- Instructions conditionnelles
- **Instructions de boucle**
- Gestion des erreurs
- Chaînes de caractères
- Tableaux
- Formulaires
- Fonctions
- Dates



# Instructions de boucle

363

## Boucle for :

```
for(expression1; expression2; expression3)
{
    //instruction ou bloc;
}
```

```
<?php
for($i=1;$i<7;$i++)
{
    echo "<h$i> $i :Titre de niveau $i </h$i>";
}
?>
```

## Boucle while :

```
while(expression)
{
    //Bloc d'instructions à répéter;
}
```

```
<?php
$n=1;
while($n%7!=0 )
{
    $n = rand(1,100);
    echo $n,"&nbsp; / ";
}
?>
```

## Boucle do... while :

```
do
{
    //bloc d'instructions ;
}
while(expression);
```

```
<?php
do
{
    $n = rand(1,100);
    echo $n,"&nbsp; / ";
}
while($n%7!=0);
?>
```



# Instructions de boucle

364

## Boucle foreach :

```
foreach($tableau as $valeur)
{
    //bloc utilisant la valeur de l'élément courant
}
```

```
<?php
//Création du tableau de 9 éléments
for($i=0;$i<=8;$i++)
{
    $tab[$i] = pow(2,$i);
}
//Lecture des valeurs du tableau
echo"Les puissances de 2 sont :";
foreach($tab as $val)
{
    echo $val." : ";
}
?>

/* résultat affiché :
Les puissances de 2 sont : 1 : 2 : 4 : 8 : 16 :
32 : 64 : 128 : 256:
*/
```



# Instructions de boucle

365

## Boucle foreach :

```
foreach($tableau as $indice=>$valeur)
{
    //bloc utilisant la valeur et l'indice de
    l'élément courant
}
```

```
<?php
//Création du tableau
for($i=0;$i<=8;$i++)
{
    $tab[$i] = pow(2,$i);
}
//Lecture des indices et des valeurs
foreach($tab as $ind=>$val)
{
    echo " 2 puissance $ind vaut $val <br
/>";
}
?>

/* résultat affiché :
2 puissance 0 vaut 1
2 puissance 1 vaut 2
.....
2 puissance 7 vaut 128
2 puissance 8 vaut 256
*/
```





# Instructions de boucle

366

## Sortie anticipée des boucles

- ❑ **Instruction break** : permet d'arrêter complètement une boucle **for**, **foreach** ou **while** avant son terme normal si une condition particulière est vérifiée.

```
<?php
    //Création d'un tableau de noms
    $tab[1]="Basile";
    $tab[2]="Conan";
    $tab[3]="Albert";
    $tab[4]="Vincent";
    //Boucle de lecture du tableau
    for($i=1;$i<count($tab);$i++)
    {
        if ($tab[$i][0]=="A")
        {
            echo "Le premier nom commençant par A est le n° $i: ",$tab[$i];
            break;
        }
    }
?>
```



# Instructions de boucle

367

## Sortie anticipée des boucles

- **Instruction continue** : n'arrête pas la boucle en cours mais seulement l'itération en cours. La variable compteur est incrémentée immédiatement, et toutes les instructions qui suivent le mot-clé **continue** ne sont pas exécutées lors de l'itération en cours.

```
<?php
//Interruption d'une boucle for
for($i=0;$i<20;$i++)
{
    if($i%5==0)
    {
        continue;
    }
    echo $i,"<br />";
}
?>
```



# Plan du chapitre 5

368

- Fonctions mathématiques
- Opérateurs booléens
- Instructions conditionnelles
- Instructions de boucle
- **Gestion des erreurs**
- Chaînes de caractères
- Tableaux
- Formulaires
- Fonctions
- Dates



# Gestion des erreurs

369

- Il existe différents types d'erreurs :
  - Erreur de syntaxe lors de la compilation du programme ;
  - Erreur d'exécution lors de l'exécution du programme ;
  - Erreur de données : donnée inattendue, incompatible avec les routines développées pour son traitement (lettre dans un nombre représentant une quantité, ... ) ;
- Lorsqu'une erreur est rencontrée dans un script PHP :
  - un message s'affiche indiquant la nature de l'erreur, sa cause, le nom du fichier de script et la ligne du script où l'erreur s'est produite ;
  - selon l'erreur, l'exécution du script se termine à l'endroit de l'erreur, ou simplement une ligne du script ne s'exécute pas, ou toutes les lignes peuvent néanmoins s'exécuter.



# Gestion des erreurs

370

❑ Les messages d'erreur affichés sont de trois types :

- **Notices:** Erreurs non critiques, par défaut non affichées. Toutes les instructions ont néanmoins pu être exécutées ;

```
<?php
error_reporting(E_ALL);
$message_2 = "c'est la rose";
$message = $message_1.$message_2 ;
echo $message ;
?>
```

**Notice:** Undefined variable: message\_1 in  
/homez.60/poulhes/atelierphp\_net/exercis  
es/lesson\_11\_a0\_notice.php on line 4  
c'est la rose

- **Warnings:** Une instruction n'a pu être correctement exécutée (exp: fichier manquant...), néanmoins le script peut continuer son exécution ;

```
<?php
$a=10; $b=0; echo $a/$b;
?>
```

**Warning:** Division by zero in  
c:\wamp5\www\php5\c3instructions\instruct3.15a.php  
on line 4

- **Fatal errors:** Erreurs fatales : le script s'arrête : erreur de syntaxe...

```
<?php
echo "-- begin --";
// Appel d'une fonction non existante
// qui va générer une E_ERROR (fatale)
someFunction();
// La ligne suivante ne sera jamais exécutée
echo "-- end --";
function somefonction()
{
    echo "Ce message est à l'intérieur de la fonction";
}
?>
```

-- begin --  
**Fatal error:** Call to undefined  
function: somefunction() in  
/homez.60/poulhes/atelierph  
p\_net/exercises/lesson\_11\_b\_  
fatal.php on line 5



# Gestion des erreurs

371

- ☐ Le but de la gestion des erreurs consiste à signaler « proprement » les problèmes au visiteur afin d'éviter l'affichage des messages d'erreur bruts tels que envoyés par PHP au navigateur.





# Gestion des erreurs

372

## Suppression des messages d'erreur

- Deux méthodes pour éviter l'affichage des messages d'erreur de PHP dans le navigateur, à savoir :
  - Faire précéder l'appel d'une fonction du caractère @ en écrivant.
    - **Exemple** : @fopen ("fichier.txt","r").
  - Utiliser la fonction error\_reporting(), définit le niveau d'erreur pour lequel le serveur doit renvoyer une erreur.
    - **Syntaxe** : int error\_reporting ( [int niveau]).
    - Le paramètre **niveau** permet de choisir le niveau d'affichage des messages d'erreur. Ses valeurs possibles sont :

Constante	Valeur	Niveau d'affichage
E_ERROR	1	Erreur fatale qui provoque l'arrêt du script, par exemple, l'appel d'une fonction qui n'existe pas.
E_WARNING	2	Avertissement ne provoquant pas l'arrêt du script, par exemple, une division par 0.
E_PARSE	4	Erreur de syntaxe détectée par l'analyseur PHP et provoquant l'arrêt du script, par exemple l'oubli du point-virgule en fin de ligne.
E_NOTICE	8	Avis que le script a rencontré un problème simple qui peut ne pas être une erreur.
E_ALL	4095	Toutes les erreurs



# Gestion des erreurs

373

## Suppression des messages d'erreur

### □ Exemple :

```
<?php
// set on ne veut voir que les erreurs notice
error_reporting(8);
//une erreur notice, variable non définie
echo $myVar ;
$string = "L'important, c'est la rose";
/*On sait que l'instruction suivante génère un warning car sting n'est pas
un tableau*/
print (join(", $string));
/*mais comme ce n'est pas une erreur fatale, l'instruction suivante est
exécutée */
echo "<br/>-- end --<br/>";
?>
```

```
Notice:Undefined variable:
myVarin
/homez.60/poulhes/atelier
php_net/exercises/lesson_
11_co_error_reporting_fun
ction.php on line 5

-- end --
```

□ **N.B:** `error_reporting(0);` // Empêche tout affichage d'erreur





# Plan du chapitre 5

374

- Fonctions mathématiques
- Opérateurs booléens
- Instructions conditionnelles
- Instructions de boucle
- Gestion des erreurs
- **Chaînes de caractères**
- Tableaux
- Formulaires
- Fonctions
- Dates



# Chaînes de caractères

375

## Définition d'une chaîne de caractères

- Une chaîne de caractères est une suite de caractères alphanumériques contenus entre des guillemets simples ou doubles.

```
$chaine='Bonjour tout le monde' ;  
Ou  
$chaine="Bonjour tout le monde" ;
```

- Si une chaîne contient une variable, celle-ci est évaluée et sa valeur est incorporée dans la chaîne uniquement si les guillemets doubles sont utilisés.

```
$chaine1='Bonjour' ;  
$chaine2=" $chaine1 tout le monde" ; // affiche : Bonjour tout le monde  
$chaine3=' $chaine1 tout le monde' ; // affiche : $chaine1 tout le monde
```



# Chaînes de caractères

376

## Séquences d'échappement

Séquence	Signification
\'	Affiche une apostrophe.
\"	Affiche des guillemets.
\\$	Affiche le signe \$.
\\	Affiche un antislash.
\n	Nouvelle ligne (code ASCII 0x0A).
\r	Retour chariot (code ASCII 0x0D).
\t	Tabulation horizontale (code ASCII 0x09).
\[0-7] {1,3}	Séquence de caractères désignant un nombre octal (de 1 à 3 caractères 0 à 7) et affichant le caractère correspondant : echo '\115\171\123\121\114'; //Affiche MySQL.
\x[0-9 A-F a-f] {1,2}	Séquence de caractères désignant un nombre hexadécimal (de 1 à 2 caractères 0 à 9 et A à F ou a à f) et affichant le caractère correspondant : echo '\x40\x79\x53\x51\x4C'; //Affiche MySQL.



# Chaînes de caractères

377

## Séquences d'échappement

Fonctions	Exemples
string addslashes (string \$ch)	<pre>\$ch="Le répertoire est : 'C:\PHP_doc\php5'; \$ch = addslashes(\$ch); echo \$ch; \$ch=stripslashes(\$ch); echo \$ch;</pre>
string stripslashes (string \$ch)	<pre>/* Affiche : Le répertoire est : \C:\\PHP_doc\\php5\ Le répertoire est : 'C:\PHP_doc\php5' */</pre>



# Chaînes de caractères

378

## Concaténation des chaînes

- ☐ L'opérateur PHP de concaténation est le point (.), qui fusionne deux chaînes littérales ou contenues dans des variables en une seule chaîne.
- ☐ **Exemple :**

```
$chaine="Bonjour"." tout le monde";  
echo $chaine ; // affiche : Bonjour tout le monde  
ou  
print ($chaine) ; // affiche : Bonjour tout le monde
```

```
$chaine1='Bonjour' ;  
$chaine2=" tout le monde" ;  
$chaine3=$chaine1. $chaine2;  
echo $chaine3 ; // affiche : Bonjour tout le monde  
ou  
print ($chaine3) ; // affiche : Bonjour tout le monde
```

```
$chaine1='Bonjour' ;  
echo $chaine1." tout le monde" ; // affiche : Bonjour tout le monde  
ou  
echo $chaine1," tout le monde" ; // affiche : Bonjour tout le monde  
ou  
print ($chaine1." tout le monde") ; // affiche : Bonjour tout le monde
```



# Chaînes de caractères

379

## Affichage formaté des chaînes

<code>void printf(string "format", string \$ch1, ..., string \$chN)</code>	Affiche directement le contenu des chaînes \$ch1, ..., \$chN, selon le format spécifié dans la chaîne "format".
<code>string sprintf(string "format", string \$ch1, ..., string \$chN)</code>	Affiche et retourne une chaîne composée des chaînes \$ch1, ..., \$chN, formatées selon le format spécifié dans la chaîne "format".
<code>void vprintf(string "format", array \$tab)</code>	Joue le même rôle que printf mais pour des chaînes passées en argument dans un tableau.
<code>string vsprintf(string "format", array \$tab)</code>	Joue le même rôle que sprintf mais pour des chaînes passées en argument dans un tableau.



# Chaînes de caractères

380

## Affichage formaté des chaînes

- La chaîne de formatage **"format"** est constituée de directives d'affichage qui indiquent la manière dont les variables passées en paramètres doivent être incorporées dans la chaîne.
- Les directives d'affichage sont composées, dans l'ordre, du caractère % suivi de :
  - Un caractère de remplissage utilisé pour compléter la chaîne quand on lui impose une longueur fixe. Le caractère de remplissage doit être précédé d'une apostrophe (').
  - Un caractère (-), pour indiquer un alignement à droite. L'alignement par défaut se fait à gauche.
  - Un nombre indiquant le nombre de caractères pour la chaîne formatée.
  - Un point suivi d'un entier indiquant le nombre de décimales à afficher pour les décimaux.
  - Une lettre indiquant la spécification de type de la valeur à afficher.

[Voir Tableau 5](#)





# Chaînes de caractères

382

## Ordre de passage des paramètres à afficher

```
$ch1 = "Monsieur " ;  
$ch2 = " Rasmus" ;  
echo sprintf ("Bonjour %s %s, bravo !",$ch1,$ch2);  
//Affiche: Bonjour Monsieur Rasmus, bravo !  
echo sprintf ("Bonjour %2$s , bravo %1$s!", $ch1,$ch2);  
//Affiche: Bonjour Rasmus , bravo Monsieur !
```





# Chaînes de caractères

383

## Longueur d'une chaîne et codes des caractères

<code>int strlen(string \$chaine)</code>	Détermine le nombre de caractères d'une chaîne.
<code>int ord(string caractere)</code>	Retrouve le code d'un caractère.
<code>string chr(int code)</code>	Retrouve le caractère à partir de son code.



# Chaînes de caractères

384

## Mise en forme des chaînes

- Les principales fonctions de mise en forme des chaînes en PHP sont :

[Voir Tableau 6](#)

- Exemples :

[Voir Tableau 7](#)



# Chaînes de caractères

385

## Recherche de sous-chaînes

- ☐ Les principales fonctions permettant la recherche de sous-chaînes en PHP sont :

[Voir Tableau 8](#)

- ☐ Exemples :

[Voir Tableau 9](#)



# Chaînes de caractères

386

## Comparaison de chaînes

- ☐ Les opérateurs de comparaison usuels sont utilisables avec les chaînes.

[Voir Tableau 10](#)

- ☐ Les principales fonctions permettant la comparaison des chaînes sont :

[Voir Tableau 11](#)



# Chaînes de caractères

387

## Transformation de chaînes en tableaux

- ☐ Les principales fonctions permettant la transformation de chaînes en tableaux sont :

[Voir Tableau 12](#)



# Plan du chapitre 5

388

- Fonctions mathématiques
- Opérateurs booléens
- Instructions conditionnelles
- Instructions de boucle
- Gestion des erreurs
- Chaînes de caractères
- **Tableaux**
- Formulaires
- Fonctions
- Dates



# Tableaux

389

## Création d'un tableau

□ La fonction permettant de créer des tableaux est la fonction **array()**.

□ On distingue deux types de tableaux :

■ **Les tableaux indicés :**

<code>\$tab[n] = valeur;</code>
<code>\$tab = array(valeuro,valeur1,...,valeurN);</code>

**N.B :**

- Le premier élément d'un tableau indicé est repéré par l'indice 0.
- Les éléments d'un tableau peuvent appartenir à des types distincts.
- Les éléments d'un tableau peuvent avoir des indices négatifs. Un indice négatif permet d'accéder aux éléments à partir de la fin du tableau en comptant à rebours. Le dernier élément du tableau non vide est toujours `$tab [-1]`.

■ **Les tableaux associatifs :**

<code>\$tabasso = array("cléA"=&gt;valeurA, "cléB"=&gt;valeurB,... "cléZ"=&gt;valeurZ);</code>
--

**N.B :** Dans un tableau associatif, la notion d'ordre des éléments perd la valeur qu'elle peut avoir dans un tableau indicé.



# Tableaux

390

## Création d'un tableau multidimensionnel

### Script PHP :

```
<? php
    $tabmulti=array(array("ligne 0-colonne 0","ligne 0-colonne
    1","ligne 0-colonne 2"),
    array("ligne 1-colonne 0","ligne 1-colonne 1","ligne 1-colonne 2"),
    array("ligne 2-colonne 0","ligne 2-colonne 1","ligne 2-colonne 2"),
    array("ligne 3-colonne 0","ligne 3-colonne 1","ligne 3-colonne 2"));

    echo "<h3>Tableau multidimensionnel</h3><table border='1'
    width=\"100%\"> <tbody>";
    for ($i=0;$i<count($tabmulti);$i++)
    {
        echo "<tr>";
        for($j=0;$j<count($tabmulti[$i]);$j++)
        {
            echo "<td><h3> .. ",$tabmulti[$i][$j]," .. </h3></td>";
        }
        echo "</tr>";
    }
    echo " </tbody> </table> ";
?>
```



## Création d'un tableau multidimensionnel

Résultat affiché :

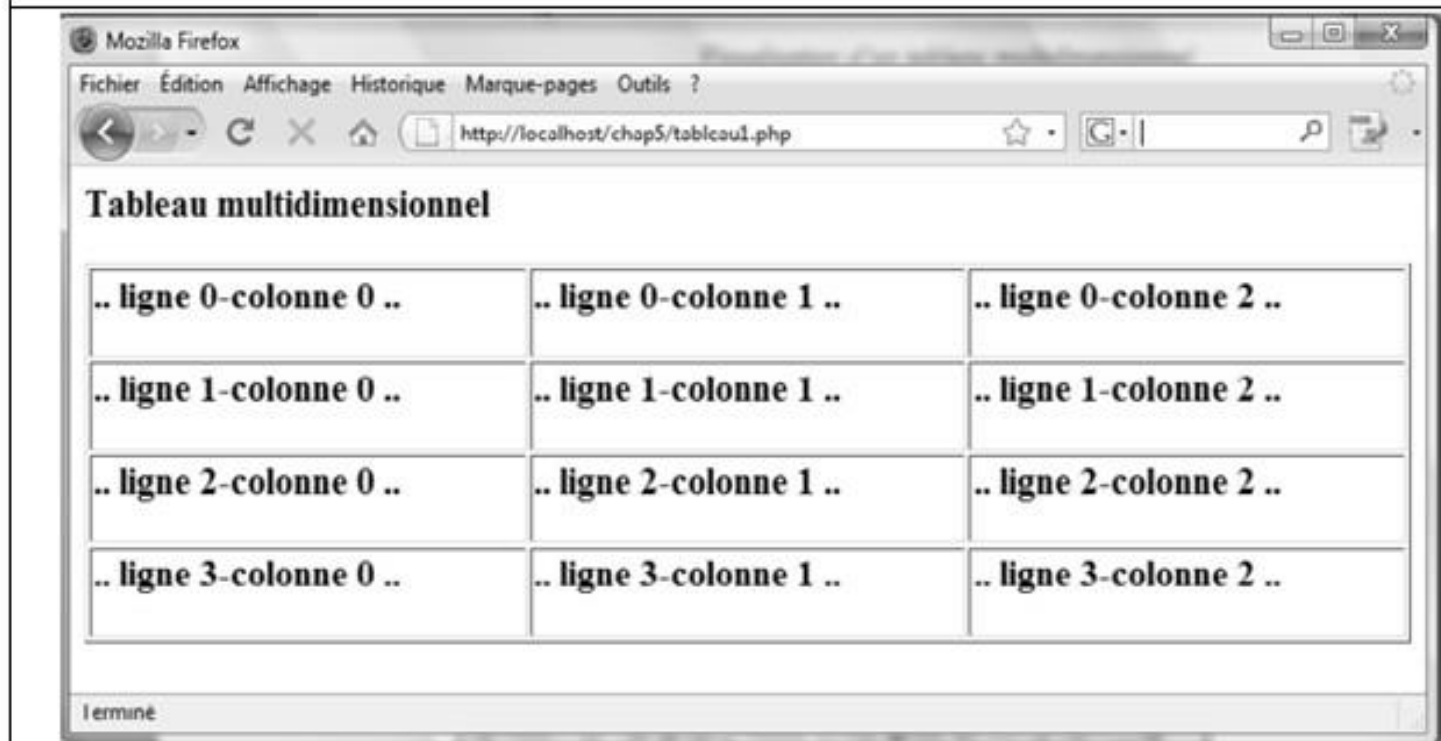


Tableau multidimensionnel

.. ligne 0-colonne 0 ..	.. ligne 0-colonne 1 ..	.. ligne 0-colonne 2 ..
.. ligne 1-colonne 0 ..	.. ligne 1-colonne 1 ..	.. ligne 1-colonne 2 ..
.. ligne 2-colonne 0 ..	.. ligne 2-colonne 1 ..	.. ligne 2-colonne 2 ..
.. ligne 3-colonne 0 ..	.. ligne 3-colonne 1 ..	.. ligne 3-colonne 2 ..

Terminé



# Tableaux

392

## Détermination du nombre d'éléments dans un tableau

- La fonction `count()` permet de déterminer le nombre d'éléments d'un tableau.
- Exemples :

### Exemple 1

```
<?php
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$result = count($a);
// $result == 3
?>
```

### Exemple 2

```
<?php
$b[0] = 7;
$b[5] = 9;
$b[10] = 11;
$result = count($b);
// $result == 3
?>
```

### Exemple 3

```
<?php
$result = count(null);
// $result == 0

$result = count(false);
// $result == 1
?>
```

### Exemple 4

```
<?php
$food = array('fruits' => array('orange', 'banana', 'apple'),
              'veggie' => array('carrot', 'collard', 'pea'));

// recursive count
echo count($food, COUNT_RECURSIVE); // output 6
// normal count
echo count($food); // output 2
?>
```



# Tableaux

393

## Lecture et affichage des éléments d'un tableau

☐ Affichage avec la fonction `print_r()`.

☐ Exemple :

### Script PHP :

```
<?php
$tabmulti=array(array("ligne 0-colonne 0","ligne 0-colonne 1","ligne 0-colonne 2"),
array("ligne 1-colonne 0","ligne 1-colonne 1","ligne 1-colonne 2"),
array("ligne 2-colonne 0","ligne 2-colonne 1","ligne 2-colonne 2"),
array("ligne 3-colonne 0","ligne 3-colonne 1","ligne 3-colonne 2"));
print_r($tabmulti);
?>
```

### Résultat affiché :

```
Array (
[0] => Array ( [0] => ligne 0-colonne 0 [1] => ligne 0-colonne 1 [2] => ligne 0-colonne 2 )
[1] => Array ( [0] => ligne 1-colonne 0 [1] => ligne 1-colonne 1 [2] => ligne 1-colonne 2 )
[2] => Array ( [0] => ligne 2-colonne 0 [1] => ligne 2-colonne 1 [2] => ligne 2-colonne 2 )
[3] => Array ( [0] => ligne 3-colonne 0 [1] => ligne 3-colonne 1 [2] => ligne 3-colonne 2 ) )
```



# Tableaux

394

## Lecture et affichage des éléments d'un tableau

☐ Lecture et affichage avec la boucle **for**.

☐ Exemple :

### Script PHP

```
<?php
$montab=array("Paris","London","Brüssel");
for ($i=0;$i<count($montab);$i++)
{
    echo "L'élément $i est $montab[$i]<br />";
}
?>
```

### Résultat affiché

```
L'élément 0 est Paris
L'élément 1 est London
L'élément 2 est Brüssel
```



# Tableaux

395

## Lecture et affichage des éléments d'un tableau

☐ Lecture et affichage avec la boucle `while`.

☐ Exemple :

### Script PHP

```
<?php
$montab=array("Paris","London","Brüssel");
$i=0;
// isset($montab[$i]) retourne FALSE lorsque
// $i dépasse le nombre d'éléments du tableau
while(isset($montab[$i]) )
{
    echo "L'élément $i est $montab[$i]<br />";
    $i++;
}
?>
```

### Résultat affiché

```
L'élément 0 est Paris
L'élément 1 est London
L'élément 2 est Brüssel
```



# Tableaux

396

## Lecture et affichage des éléments d'un tableau

- Lecture et affichage avec la fonction **each()**.
  - **N.B :** la fonction **each()** est utilisée lorsque les indices des différents éléments d'un tableau donné ne sont pas consécutifs.
  - **Syntaxe :**  
\$element = each(\$tab)
    - \$element est un tableau à 4 éléments :
    - \$element[0], qui contient l'indice de l'élément courant.
    - \$element[1], qui contient la valeur de l'élément courant.
    - \$element["key"], qui contient la clé de l'élément courant.
    - \$element["value"], qui contient la valeur de l'élément courant.
  - L'expression **\$element = each(\$tab)** retourne TRUE tant que le tableau contient des éléments.

□ **Exemple :**

[Voir Tableau 13](#)

## Lecture et affichage des éléments d'un tableau

- ☐ Lecture et affichage avec l'instruction `foreach`.
- ☐ Exemple :

[Voir Tableau 14](#)

## Extraction d'une partie d'un tableau

□ Il est possible de créer un nouveau tableau comme sous-ensemble du tableau initial et ne contenant qu'un nombre déterminé de ses éléments, grâce à la fonction `array_slice()`.

□ Syntaxe :

```
$sous_tab = array_slice(array $tab,int ind, int nb)
```





# Tableaux

399

## Extraction d'une partie d'un tableau

- Si `ind` et `nb` sont positifs, le tableau `$sous_tab` contient `nb` éléments du tableau initial extraits en commençant à l'indice `ind`.

**Exemple :**

`array_slice($tab,2,3)` retourne un tableau comprenant 3 éléments extraits à partir de l'indice 2. Il contient donc les éléments d'indice 2, 3 et 4 du tableau `$tab`.

- Si `ind` est négatif et que `nb` est positif, le compte des éléments se fait en partant de la fin du tableau `$tab` (le dernier élément du tableau est affecté virtuellement de l'indice -1). Le paramètre `nb` désigne encore le nombre d'élément à extraire.

**Exemple :**

`array_slice($tab,-5,4)` retourne quatre éléments de `$tab` extraits en commençant au cinquième à partir de la fin.



# Tableaux

400

## Extraction d'une partie d'un tableau

- Si `ind` est positif et `nb` négatif, le tableau `$sous_tab` contient les éléments de `$tab` extraits en commençant à l'indice `ind` et en s'arrêtant à celui qui a l'indice négatif virtuel `nb` (toujours en commençant par la fin).

**Exemple :**

`array_slice($tab,2,-4)` retourne tous les éléments à partir de l'indice 2 jusqu'à la fin, sauf les quatre derniers.

- Si `ind` et `nb` sont négatifs, le tableau `$sous_tab` contient les éléments de `$tab` extraits en commençant à l'indice négatif `ind` et en s'arrêtant à celui d'indice négatif `nb`.

**Exemple :**

`array_slice($tab,-5,-2)` retourne trois éléments compris entre les indices virtuels `-5` compris et `-2` non compris.

## Ajout et suppression d'éléments dans un tableau

<code>int array_push(\$tab, valeur1,..., valeurN)</code>	ajoute en une seule opération les N éléments passés en paramètres à la fin du tableau désigné par la variable \$tab.
<code>int array_unshift(\$tab, valeur1,..., valeurN)</code>	ajoute au tableau \$tab les N éléments passés en paramètres au début du tableau.
<code>array_pop(\$tab)</code>	supprime le dernier élément du tableau \$tab et retourne cet élément s'il existe ou la valeur NULL dans le cas contraire.
<code>array_shift(\$tab)</code>	supprime le premier élément du tableau \$tab et retourne cet élément s'il existe ou la valeur NULL dans le cas contraire.
<code>unset(element_a_supprimer)</code>	supprime un élément d'indice ou de clé quelconque du tableau \$tab. Cette fonction n'a pas d'effet sur les autres indices du tableau, qui conservent tous la valeur qu'ils avaient avant la suppression.



## Ajout et suppression d'éléments dans un tableau

Exemple :

[Voir Tableau 15](#)



# Tableaux

403

## Élimination des éléments faisant double emploi dans un tableau

`array array_unique($tab)`

retourne un nouveau tableau ne contenant que la dernière occurrence de chaque valeur présente plusieurs fois dans le tableau \$tab. Les indices ou les clés associées à chaque élément sont conservés, et le tableau retourné comporte des « trous » dans la suite des indices si ces derniers sont numériques.

**Exemple :**

### Script PHP :

```
<?php
$tab = array("Jacques","Paul","Pierre","Alban","Paul","Jack","Paul");
$tab2 = array_unique($tab);
print_r($tab2);
?>
```

### Résultat affiché :

```
Array ( [0] => Jacques [2] => Pierre [3] => Alban [5] => Jack [6] => Paul )
```

## Fusion des tableaux

**`$tab = array_merge($tab1,$tab2,...,$tabN)`**

retourne dans \$tab l'ensemble des éléments présents dans les tableaux \$tab1, \$tab2, ..., \$tabN. Si les tableaux à fusionner sont indicés, les éléments du tableau passé en premier paramètre sont conservés, ceux des autres paramètres ayant les indices suivants. Les éléments présents dans plusieurs des paramètres sont présents en double dans le tableau final. Si les tableaux à fusionner sont associatifs, les clés et les associations clé-valeur sont préservées. Par contre, si plusieurs des paramètres ont des clés communes, seule l'association clé-valeur du dernier paramètre est conservée, et celle du tableau précédent est perdue.

**`$tab = array_merge_recursive()`**

Cette fonction n'efface pas la première valeur associée à une clé double mais associe à chaque clé présente plusieurs fois un tableau indicé contenant toutes les valeurs ayant la même clé.

## Intersection et différence de deux tableaux

Fonction	Description
<code>array array_intersect(\$tab1,\$tab2)</code>	Cette fonction retourne un tableau contenant tous les éléments communs aux tableaux \$tab1 et \$tab2. Les indices associés aux valeurs du tableau retourné comme résultat correspondent à ceux du tableau passé en premier paramètre.
<code>array array_diff(\$tab1,\$tab2)</code>	Elle retourne un tableau contenant les éléments présents dans le premier paramètre mais pas dans le second. Les indices associés aux valeurs dans les tableaux d'origine sont conservés.



## Intersection et différence de deux tableaux

Exemple :

[Voir Tableau 16](#)



## Tri d'un tableau indicé

### Tri selon l'ordre ASCII

`array sort($tab)`

Trie les valeurs du tableau \$tab en ordre croissant des codes ASCII des caractères qui les composent (donc en tenant compte de la casse des caractères). Les correspondances entre les indices et les valeurs des éléments sont perdues après le tri.

`array rsort($tab)`

Trie les valeurs du tableau \$tab en ordre décroissant des codes ASCII des caractères qui les composent. Les correspondances entre les indices et les valeurs des éléments sont perdues après le tri.

`array array_reverse($tab)`

Inverse l'ordre des valeurs des éléments de \$tab. Les indices sont perdus.



## Tri d'un tableau indicé

Exemple :

[Voir Tableau 17](#)

## Tri d'un tableau indicé

### Tri selon l'ordre naturel

`array natsort($tab)`

Trie les valeurs du tableau `$tab` selon l'ordre naturel croissant des caractères qui les composent. Le tri étant effectué en tenant compte de la casse, les majuscules sont placées avant les minuscules. Les correspondances entre les indices ou les clés et les valeurs des éléments sont sauvegardées après le tri, ce qui rend la fonction également applicable aux tableaux associatifs.

`array natcasesort($tab)`

Trie les valeurs du tableau `$tab` selon l'ordre naturel croissant, sans tenir compte de la casse, ce qui correspond davantage à l'ordre courant du dictionnaire. Les correspondances entre les indices ou les clés et les valeurs des éléments sont sauvegardées après le tri.



## Tri d'un tableau indicé

Exemple :

[Voir Tableau 18](#)



## Tri d'un tableau indicé

**N.B :** Il est déconseillé d'utiliser une boucle **for** pour lire l'ensemble des données, au risque de perdre l'ordre créé par le tri vu que les fonctions **natsort()** et **natcasesort()** conservent les correspondances entre les indices ou les clés et les valeurs. Une boucle **foreach** est indispensable, même pour des tableaux indicés.

## Tri d'un tableau associatif

### ☐ Tri des valeurs :

Tri selon le code ASCII	
<code>void asort(array \$tab)</code>	Trie les valeurs du tableau \$tab selon l'ordre croissant des codes ASCII des caractères qui les composent en préservant les associations clé-valeur.
<code>void arsort(array \$tab)</code>	Trie les valeurs du tableau \$tab selon l'ordre décroissant des codes ASCII des caractères qui les composent en préservant les associations clé-valeur.

## Tri d'un tableau associatif

### ☐ Tri des valeurs :

Tri selon l'ordre naturel	
<code>array natsort(\$tab)</code>	Trie les valeurs du tableau <code>\$tab</code> selon l'ordre naturel croissant des caractères qui les composent. Le tri étant effectué en tenant compte de la casse, les majuscules sont placées avant les minuscules. Les correspondances entre les indices ou les clés et les valeurs des éléments sont sauvegardées après le tri, ce qui rend la fonction également applicable aux tableaux associatifs.
<code>array natcasesort(\$tab)</code>	Trie les valeurs du tableau <code>\$tab</code> selon l'ordre naturel croissant, sans tenir compte de la casse, ce qui correspond davantage à l'ordre courant du dictionnaire. Les correspondances entre les indices ou les clés et les valeurs des éléments sont sauvegardées après le tri.



# Tableaux

414

## Tri d'un tableau associatif

### ☐ Tri des clés :

<code>boolean ksort(array \$tab)</code>	trie les clés de <code>\$tab</code> selon l'ordre croissant des codes ASCII des caractères. Les associations clé-valeur sont conservées. Cette fonction retourne une valeur booléenne indiquant si l'opération de tri a réussi ou non.
<code>boolean krsort(array \$tab)</code>	trie les clés de <code>\$tab</code> selon l'ordre décroissant des codes ASCII des caractères. Les associations clé-valeur sont conservées. Cette fonction retourne une valeur booléenne indiquant si l'opération de tri a réussi ou non.

**N.B :** Il est possible de transformer la casse des clés avant le tri en appliquant au tableau la fonction `array_change_key_case()` :

`array array_change_key_case (array $tab, int CTE)`

Cette fonction transforme toutes les clés du tableau `$tab` en minuscules si la constante `CTE` vaut `CASE_LOWER` (valeur par défaut) ou en majuscules si elle vaut `CASE_UPPER`.





## Tri d'un tableau associatif

Exemple :

[Voir Tableau 19](#)

## Sélection des éléments

```
array array_filter(array $tab, string "nom_fonction")
```

Elle retourne un nouveau tableau ne contenant que les éléments de \$tab qui répondent à la condition définie dans la fonction dont le nom est passé en second paramètre. Le tableau initial est conservé.

Exemple :

[Voir Tableau 20](#)



# Tableaux

417

## Application d'une fonction aux éléments d'un tableau

**int array\_walk(\$tab,"nom\_fonction")**

Applique la fonction dont le nom est passé en paramètre à tous les éléments du tableau qu'il soit indicé ou associatif. La fonction appliquée aux valeurs doit être une fonction personnalisée et non une fonction native de PHP.

**divers array\_reduce(array \$tab, string "nom\_fonction")**

Applique la fonction dont le nom est passé en paramètre pour retourner un seul résultat à partir de l'ensemble des valeurs contenues dans le tableau (exp : somme ou produit des éléments du tableau).

**Exemple :**

[Voir Tableau 21](#)



# Plan du chapitre 5

418

- Fonctions mathématiques
- Opérateurs booléens
- Instructions conditionnelles
- Instructions de boucle
- Gestion des erreurs
- Chaînes de caractères
- Tableaux
- **Formulaires**
- Fonctions
- Dates



# Formulaires

419

## Récupération des données d'un formulaire

- ☐ Lorsque l'utilisateur clique sur le bouton d'envoi après avoir rempli les différents champs du formulaire, une requête HTTP est envoyée au serveur à destination du script désigné par l'attribut **action** de l'élément **<form>**.
- ☐ La requête contient toutes les associations entre les noms des champs et leurs valeurs. Ces associations se trouvent dans l'en-tête HTTP si la méthode **POST** est utilisée et dans l'URL s'il s'agit de la méthode **GET**.



# Formulaires

420

## Récupération des données d'un formulaire

### Valeurs uniques :

- ☐ Les valeurs uniques proviennent des champs de formulaire dans lesquels l'utilisateur ne peut entrer qu'une seule valeur (exp: un texte), ou ne peut faire qu'un seul choix (bouton radio, liste de sélection à choix unique).
- ☐ Les valeurs lues sont contenues sur le serveur dans des tableaux associatifs dits **superglobaux** appelés `$_POST` et `$_GET`, selon la méthode choisie (POST ou GET) et dont les clés sont les noms associés aux champs par l'attribut **name**.
- ☐ Exemple 1 : [Voir Tableau 22-1](#)
- ☐ Exemple 2 : [Voir Tableau 22-2](#)



# Formulaires

421

## Récupération des données d'un formulaire

### Valeurs multiples :

- ☐ Pour permettre à l'utilisateur de saisir plusieurs valeurs sous un même nom de composant (exp : un groupe de cases à cocher ayant le même attribut **name**), il faut définir l'attribut **multiple** dans le composant en question.
- ☐ Dans le cas de valeurs multiples, les données sont récupérées côté serveur sous la forme d'un tableau. Il est donc nécessaire de faire suivre le nom du composant de crochets tel est le cas lors de la création d'une variable de type **array**.
- ☐ Exemple :

[Voir Tableau 22-3](#)



# Formulaires

422

## Transfert de fichiers vers le serveur

- ☐ Comparé au transfert de données, le transfert de fichiers présente un problème de sécurité pour les sites web puisque des fichiers vont être écrits et éventuellement exécutés sur le serveur.
- ☐ Il est ainsi incontournable d'utiliser l'attribut **accept** de l'élément `<input />` qui détermine le type de fichiers acceptés (exp : image/gif, text/html...).
- ☐ Il est aussi possible de limiter la taille maximale des fichiers à transmettre via un champ caché nommé **MAX\_FILE\_SIZE** dont l'attribut **value** contient la taille maximale admise en octet.

```
<input type="hidden" name="MAX_FILE_SIZE" value="100000" />
```





# Formulaires

423

## Transfert de fichiers vers le serveur

- ☐ Dans le cas de transfert de fichiers dans un formulaire, l'élément `<form>` doit avoir l'attribut **method** à la valeur **post** et l'attribut **enctype** à la valeur **multipart/form-data**.
- ☐ Un clic sur le bouton **Submit** provoque l'envoi du fichier sélectionné au serveur et son traitement par un script. Le fichier est donc temporairement mis dans un répertoire tampon défini par la directive `"upload_tmp_dir"` du fichier `php.ini` et est enregistré sous un nom différent de celui qu'il avait sur le poste client.
- ☐ Si le fichier transféré ne subit aucun traitement, il est perdu lors de la déconnexion du client.
- ☐ Exemple : [Voir Tableau 23](#)

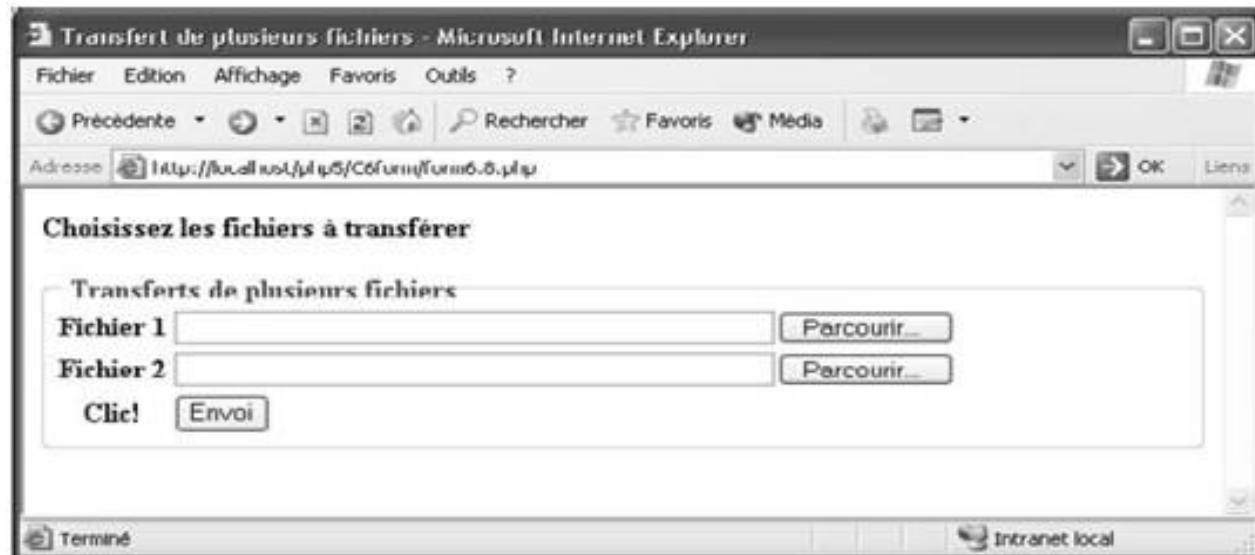
# Formulaires

424

## Transfert de fichiers vers le serveur

- Il est possible de transférer plusieurs fichiers simultanément en utilisant la syntaxe suivante (suite de l'exemple) :

```
<input type="file" name="fich[]" accept="image/gif" size="50"/>
```





# Formulaires

425

## Gestion de boutons d'envoi multiples

☐ Il est possible d'avoir plusieurs boutons **submit** dans un même formulaire dont l'activation d'un d'entre eux par l'utilisateur déclenche une action différente définie par l'attribut **value**.

☐ Exemple :

[Voir Tableau 24](#)



# Plan du chapitre 5

426

- Fonctions mathématiques
- Opérateurs booléens
- Instructions conditionnelles
- Instructions de boucle
- Gestion des erreurs
- Chaînes de caractères
- Tableaux
- Formulaires
- **Fonctions**
- Dates



# Fonctions

427

## Déclaration d'une fonction

```
function mafonction($x,$y,...)
{
    //code de définition de la fonction
    return $var; /* dans le cas où la fonction retourne une valeur, ce qui
n'est pas obligatoire */
}
```

- ☐ Depuis PHP 4, la position de la déclaration d'une fonction dans le script n'a pas d'importance. Il est ainsi possible d'appeler une fonction au début du script alors qu'elle n'est définie qu'en fin de script.
- ☐ Pour les fonctions définies dans des scripts séparés, il est préférable de les inclure dès le début du script qui les utilise via l'instruction **include()** ou **require()**.



# Fonctions

428

## Appel d'une fonction

### Fonction ne retournant pas de valeur :

```
mafonction($variable1,$variable2,...) ;
```

ou

```
mafonction(valeur1, valeur2,...);
```

### Fonction retournant une valeur :

```
$valeurRetournee=mafonction(valeur1, valeur2,...);
```

□ Exemple :

[Voir Tableau 25](#)



# Fonctions

429

## Fonction retournant plusieurs valeurs

- ☐ Pour qu'une fonction en PHP puisse retourner plusieurs variables, on a recours au type `array`.
- ☐ Exemple :

[Voir Tableau 26](#)



## Fonction avec paramètres par défaut

☐ Dans la définition d'une fonction, tous les paramètres qui ont une valeur par défaut doivent figurer en dernier dans la liste des variables.

☐ Exemple :

[Voir Tableau 27](#)





# Fonctions

431

## Fonctions avec un nombre variable de paramètres

- Il existe deux méthodes pour définir des fonctions dont le nombre de paramètres à passer n'est pas connu à l'avance :
  - Utiliser le type **array** pour les paramètres ;
  - Utiliser les fonctions particulières de PHP :

### **integer func\_num\_args()**

s'utilise sans argument et seulement dans le corps même d'une fonction. Elle retourne le nombre d'arguments passés à cette fonction.

### **divers func\_get\_arg(integer \$N)**

retourne la valeur du paramètre passé à la position \$N. Comme dans les tableaux, le premier paramètre a l'indice 0.

### **array func\_get\_args()**

s'utilise sans paramètre et retourne un tableau indicé contenant tous les paramètres passés à la fonction.

**Exemples :**

[Voir Tableau 28-1](#)

[Voir Tableau 28-2](#)



# Fonctions

432

## Portée des variables

### Variables locales et globales

- ☐ Toute variable utilisée dans la déclaration d'une fonction est, sauf indication contraire, locale au bloc de définition de la fonction.
- ☐ Toute variable définie en dehors d'une fonction ou d'une classe est globale et accessible partout dans le script qui l'a créée.
- ☐ **N.B :** toute modification d'une variable locale opérée dans le corps d'une fonction n'a aucun effet sur une variable externe à la fonction et portant le même nom.
- ☐ Pour utiliser la valeur d'une variable globale dans une fonction, il faut la déclarer dans le corps de la fonction avec le mot-clé **global**.
- ☐ **Exemple :**

[Voir Tableau 29](#)



# Fonctions

433

## Portée des variables

### Les variables statiques :

- ☐ Pour conserver la valeur précédemment affectée à une variable locale entre deux appels d'une même fonction, il faut déclarer la variable comme statique avec le mot-clé **static**, et ce avant de l'utiliser dans le corps de la fonction.
- ☐ L'utilisation typique des variables statiques concerne les fonctions qui effectuent des opérations de cumul.
- ☐ **N.B :** Une variable déclarée comme statique ne conserve toutefois sa valeur que pendant la durée du script (i.e. exécution d'une page).
- ☐ Exemple :

[Voir Tableau 30](#)



# Fonctions

434

## Passage de paramètres par référence

- ☐ Lorsque les paramètres sont passés par valeur, une copie des variables est utilisée par la fonction. Ainsi, les modifications apportées aux valeurs des paramètres passés ne sont pas visibles à l'extérieur de la fonction.
- ☐ Grâce au passage des paramètres par référence, les modifications effectuées dans le corps d'une fonction sont répercutées à l'extérieur.
- ☐ Le passage de paramètres par référence peut être :
  - Systématique ;
  - occasionnel.
- ☐ Exemple :

[Voir Tableau 31](#)

## Fonctions dynamiques

☐ PHP offre la possibilité de travailler avec des noms de fonctions dynamiques qui peuvent être variables et donc dépendants de l'utilisateur du site ou de l'interrogation d'une base de données.

☐ Exemple :

[Voir Tableau 32](#)



# Fonctions

436

## Fonctions conditionnelles

- ☐ Une fonction est conditionnelle est une fonction définie à l'intérieur d'un bloc **if**.
- ☐ Une fonction conditionnelle n'est utilisable que si l'instruction **if** qui la contient a été exécutée et l'expression conditionnelle contenue dans **if** a la valeur booléenne **TRUE**.
- ☐ Exemple :

[Voir Tableau 33](#)



# Fonctions

437

## Fonctions récursives

- ☐ Une fonction est dite récursive si, à l'intérieur de son corps, elle s'appelle elle-même avec une valeur de paramètre différent (sinon elle boucle).
- ☐ Chaque appel constitue un niveau de récursivité.
- ☐ **Exemple :** la fonction qui retourne la factorielle d'un nombre entier  $n$ .

[Voir Tableau 34](#)



# Plan du chapitre 5

438

- Fonctions mathématiques
- Opérateurs booléens
- Instructions conditionnelles
- Instructions de boucle
- Gestion des erreurs
- Chaînes de caractères
- Tableaux
- Formulaires
- Fonctions
- **Dates**



## Définition d'une date

**int time() ;**

retourne le timestamp de l'instant présent en secondes par rapport à une date d'origine arbitraire, correspondant au 1er janvier 1970 00 h 00 m 00 s.

**string microtime() ;**

retourne une chaîne de caractères commençant par le nombre de microsecondes suivi du nombre de secondes.

**int mktime(int heure, int minute, int seconde, int mois, int jour, int année, int été)**

retourne le timestamp correspondant à la date définie par les valeurs entières passées en paramètres. Le dernier paramètre doit valoir 1 pour l'heure d'hiver, 0 pour l'heure d'été et - 1, valeur par défaut, si vous ne savez pas.

**int gmmktime(int heure, int minute, int seconde, int mois, int jour, int année, int été)**

retourne le timestamp correspondant à la date GMT.

**Exemple :**

[Voir Tableau 35](#)

## Vérification d'une date

**boolean checkdate(int mois, int jour, int année) ;**

retourne une valeur booléenne TRUE si la date existe et FALSE dans le cas contraire.

□ Exemple :

[Voir Tableau 36](#)



# Dates

441

## Affichage d'une date

**string date(string format\_de\_date,[int timestamp]) ;**

retourne une chaîne contenant des informations de date dont la mise en forme est définie par des caractères spéciaux. La date retournée correspond à celle du timestamp passé en deuxième paramètre ou, si ce dernier est omis, à celle de l'instant en cours. Pour afficher un des caractères spéciaux indépendamment de sa fonction de formatage, il faut le faire précéder d'un antislash. Exemple, \h affiche le caractère « h » et non le nombre d'heure. Pour afficher les caractères « n » et « t », il faut écrire \\n et \\t car \n et \t sont employés pour le saut de ligne et la tabulation.

### □ Exemple :

```
echo "Aujourd'hui ",date("l, d F Y |i| |e|s\\t H:i:s ");  
$numjour = date("w");  
echo $numjour ;
```

```
Aujourd'hui Monday, 20 October 2008 il est 23:36:27  
1
```



# Dates

442

## Affichage d'une date

- Caractères de définition du format d'affichage :

[Voir Tableau 37](#)

**array getdate([int timestamp]) ;**

retourne un tableau contenant toutes les informations de date. Si le paramètre timestamp est omis, la fonction getdate() retourne les informations sur la date en cours.

**array localtime([nombre][, tab\_assocatif]) ;**

retourne l'heure locale dans un tableau indicé par défaut ou associatif.

- Exemple :

```
$jour = getdate();  
echo "Aujourd'hui {$jour["weekday"]} {$jour["mday"]} {$jour["month"]} {$jour["year"]}";  
Aujourd'hui Monday 20 October 2008
```

- Clés du tableau retourné par la fonction getdate() :

[Voir Tableau 38](#)



# Dates

443

## Affichage d'une date

**`string strftime(string format_de_date, int timestamp) ;`**

Similaire à la fonction `date()`, elle permet d'afficher, en anglais, les informations de date composées à l'aide des caractères spéciaux du tableau 37.

**`string gmstrftime (string format_de_date, int timestamp) ;`**

fournit les mêmes résultats que la fonction `strftime()` mais en heure GMT.

**`string setlocale(int constante, string lang) ;`**

affiche l'équivalent des dates récupérées par les fonctions `strftime()` et `gmstrftime()` dans la langue indiquée en paramètre. La constante prend les valeurs `LC_ALL` ou `LC_TIME` dans le contexte temporel.



# Dates

444

## Affichage d'une date

### ☐ Exemple :

#### Script PHP :

```
<?php
echo "fonction strftime() et setlocale() <br />";
setlocale (LC_ALL, "fr");
echo "Français : Aujourd'hui",strftime(" %A %d %B %Y %H h %M m %S s
%Z",time()),"<br />";
echo "Français GMT : Aujourd'hui",gmstrftime(" %A %d %B %Y %H h %M m
%S s %Z",time()),"<br />";
?>
```

#### Résultat affiché :

```
Fonctions strftime() et setlocale()
Français : Aujourd'hui vendredi 24 octobre 2008 23 h 53 m 46 s Paris,
Madrid (heure d'été)
Français GMT : Aujourd'hui vendredi 24 octobre 2008 21 h 53 m 46 s
Paris, Madrid
```

### ☐ Caractères de formatage de la fonction strftime() :

[Voir Tableau 39](#)



# Dates

445

## Affichage d'une date en français

### Script PHP

```
<?php
//Date en français
$jour = getdate();
echo "Anglais: Aujourd'hui {".$jour["weekday"]."} {".$jour["mday"]."}
{".$jour["month"]."}
{".$jour["year"]."}<br />";
$semaine = array(" dimanche "," lundi "," mardi "," mercredi "," jeudi ","
vendredi "," samedi ");
$mois = array(1=>" janvier "," février "," mars "," avril "," mai "," juin ","
juillet "," août "," septembre "," octobre "," novembre "," décembre ");
//Avec getdate()
echo "Français: Avec getdate(): Aujourd'hui ", $semaine[$jour['wday']]
,$jour['mday'], $mois[$jour['mon']], $jour['year'], "<br />";
//Avec date()
echo " Français: Avec date(): Aujourd'hui ", $semaine[date('w')], "
", date('j'), " ", $mois[date('n')], date('Y'), "<br />";
?>
```

### Résultat affiché

```
Anglais: Aujourd'hui Monday 20 October 2008
Français: Avec getdate(): Aujourd'hui lundi 20 octobre 2008
Français: Avec date(): Aujourd'hui lundi 20 octobre 2008
```