Programmation du Web : Présentation et description du langage PHP

Jean-Baptiste Vioix
(jean-baptiste.vioix@iut-dijon.u-bourgogne.fr)

IUT de Dijon-Auxerre - LE2I

Historique de PHP

- Rasmus Lerdorf a crée PHP en 1995. A ce moment c'est un ensemble de scripts Perl destinés au Web.
- Le but était de faciliter des tâches répétitives.
- Différentes versions se sont succédées, chacune apportant des améliorations (rapidité, gestion des bases de données,...).
- En 2000, la version 4 apporte de nombreuses évolutions et plusieurs millions de sites utilisent PHP qui devient un concurrent sérieux aux solutions comme les ASP, JSP, ...
- La dernière version majeure est la 5 sortie en 2004.

Historique de PHP

- Rasmus Lerdorf a crée PHP en 1995. A ce moment c'est un ensemble de scripts Perl destinés au Web.
- Le but était de faciliter des tâches répétitives.
- Différentes versions se sont succédées, chacune apportant des améliorations (rapidité, gestion des bases de données,...).
- En 2000, la version 4 apporte de nombreuses évolutions et plusieurs millions de sites utilisent PHP qui devient un concurrent sérieux aux solutions comme les ASP, JSP, ...
- La dernière version majeure est la 5 sortie en 2004.

Historique de PHP

- Rasmus Lerdorf a crée PHP en 1995. A ce moment c'est un ensemble de scripts Perl destinés au Web.
- Le but était de faciliter des tâches répétitives.
- Différentes versions se sont succédées, chacune apportant des améliorations (rapidité, gestion des bases de données,...).
- En 2000, la version 4 apporte de nombreuses évolutions et plusieurs millions de sites utilisent PHP qui devient un concurrent sérieux aux solutions comme les ASP, JSP, ...
- La dernière version majeure est la 5 sortie en 2004.

Historique de PHP

- Rasmus Lerdorf a crée PHP en 1995. A ce moment c'est un ensemble de scripts Perl destinés au Web.
- Le but était de faciliter des tâches répétitives.
- Différentes versions se sont succédées, chacune apportant des améliorations (rapidité, gestion des bases de données,...).
- En 2000, la version 4 apporte de nombreuses évolutions et plusieurs millions de sites utilisent PHP qui devient un concurrent sérieux aux solutions comme les ASP, JSP, ...
- La dernière version majeure est la 5 sortie en 2004.

Historique de PHP

- Rasmus Lerdorf a crée PHP en 1995. A ce moment c'est un ensemble de scripts Perl destinés au Web.
- Le but était de faciliter des tâches répétitives.
- Différentes versions se sont succédées, chacune apportant des améliorations (rapidité, gestion des bases de données,...).
- En 2000, la version 4 apporte de nombreuses évolutions et plusieurs millions de sites utilisent PHP qui devient un concurrent sérieux aux solutions comme les ASP, JSP, ...
- La dernière version majeure est la 5 sortie en 2004.

Caractéristiques de PHP

- Langage facile à apprendre (beaucoup de points communs avec les langages usuels comme C, Java, ...).
- Possibilité de programmer en objet sans être obligatoire.
- Interfaçage aisé avec la plupart des moteurs de bases de données existants.
- API très riche (création d'image, de document PDF, ...).
- Très nombreuses applications PHP libres disponibles (forums, webmail, sondages, ...).
- Communauté de développeurs très dynamique.

Caractéristiques de PHP

- Langage facile à apprendre (beaucoup de points communs avec les langages usuels comme C, Java, ...).
- Possibilité de programmer en objet sans être obligatoire.
- Interfaçage aisé avec la plupart des moteurs de bases de données existants.
- API très riche (création d'image, de document PDF, ...).
- Très nombreuses applications PHP libres disponibles (forums, webmail, sondages, ...).
- Communauté de développeurs très dynamique.

Caractéristiques de PHP

- Langage facile à apprendre (beaucoup de points communs avec les langages usuels comme C, Java, ...).
- Possibilité de programmer en objet sans être obligatoire.
- Interfaçage aisé avec la plupart des moteurs de bases de données existants.
- API très riche (création d'image, de document PDF, ...).
- Très nombreuses applications PHP libres disponibles (forums, webmail, sondages, ...).
- Communauté de développeurs très dynamique.

Caractéristiques de PHP

- Langage facile à apprendre (beaucoup de points communs avec les langages usuels comme C, Java, ...).
- Possibilité de programmer en objet sans être obligatoire.
- Interfaçage aisé avec la plupart des moteurs de bases de données existants.
- API très riche (création d'image, de document PDF, ...).
- Très nombreuses applications PHP libres disponibles (forums, webmail, sondages, ...).
- Communauté de développeurs très dynamique.

Caractéristiques de PHP

- Langage facile à apprendre (beaucoup de points communs avec les langages usuels comme C, Java, ...).
- Possibilité de programmer en objet sans être obligatoire.
- Interfaçage aisé avec la plupart des moteurs de bases de données existants.
- API très riche (création d'image, de document PDF, ...).
- Très nombreuses applications PHP libres disponibles (forums, webmail, sondages, ...).
- Communauté de développeurs très dynamique.

Caractéristiques de PHP

- Langage facile à apprendre (beaucoup de points communs avec les langages usuels comme C, Java, ...).
- Possibilité de programmer en objet sans être obligatoire.
- Interfaçage aisé avec la plupart des moteurs de bases de données existants.
- API très riche (création d'image, de document PDF, ...).
- Très nombreuses applications PHP libres disponibles (forums, webmail, sondages, ...).
- Communauté de développeurs très dynamique.

Utilisation de PHP (livre blanc de PHP - 2005)

- PHP est utilisé par 87 % des entreprises du CAC40.
- En France, parmi les 10 entreprises ayant le plus de visites, 9 utilisent PHP (la 10ème est Microsoft).
- Dans le monde, PHP est utilisé par 22 millions de domaines et 1,2 millions d'adresses IP.
- Développé par la fondation Apache par environ 1000 ingénieurs et utilisé par environ 500 000 développeurs.
- PHP est un des piliers des plate-formes LAMP (*Linux*, *Apache*,*MySQL* et *PHP* ou *Perl* ou *Python*).

Utilisation de PHP (livre blanc de PHP - 2005)

- PHP est utilisé par 87 % des entreprises du CAC40.
- En France, parmi les 10 entreprises ayant le plus de visites, 9 utilisent PHP (la 10ème est Microsoft).
- Dans le monde, PHP est utilisé par 22 millions de domaines et 1,2 millions d'adresses IP.
- Développé par la fondation Apache par environ 1000 ingénieurs et utilisé par environ 500 000 développeurs.
- PHP est un des piliers des plate-formes LAMP (*Linux*, *Apache*,*MySQL* et *PHP* ou *Perl* ou *Python*).

Utilisation de PHP (livre blanc de PHP - 2005)

- PHP est utilisé par 87 % des entreprises du CAC40.
- En France, parmi les 10 entreprises ayant le plus de visites, 9 utilisent PHP (la 10ème est Microsoft).
- Dans le monde, PHP est utilisé par 22 millions de domaines et 1,2 millions d'adresses IP.
- Développé par la fondation Apache par environ 1000 ingénieurs et utilisé par environ 500 000 développeurs.
- PHP est un des piliers des plate-formes LAMP (*Linux*, *Apache*,*MySQL* et *PHP* ou *Perl* ou *Python*).

Utilisation de PHP (livre blanc de PHP - 2005)

- PHP est utilisé par 87 % des entreprises du CAC40.
- En France, parmi les 10 entreprises ayant le plus de visites, 9 utilisent PHP (la 10ème est Microsoft).
- Dans le monde, PHP est utilisé par 22 millions de domaines et 1,2 millions d'adresses IP.
- Développé par la fondation Apache par environ 1000 ingénieurs et utilisé par environ 500 000 développeurs.
- PHP est un des piliers des plate-formes LAMP (*Linux*, *Apache*,*MySQL* et *PHP* ou *Perl* ou *Python*).

Utilisation de PHP (livre blanc de PHP - 2005)

- PHP est utilisé par 87 % des entreprises du CAC40.
- En France, parmi les 10 entreprises ayant le plus de visites, 9 utilisent PHP (la 10ème est Microsoft).
- Dans le monde, PHP est utilisé par 22 millions de domaines et 1,2 millions d'adresses IP.
- Développé par la fondation Apache par environ 1000 ingénieurs et utilisé par environ 500 000 développeurs.
- PHP est un des piliers des plate-formes LAMP (*Linux*, *Apache*,*MySQL* et *PHP* ou *Perl* ou *Python*).

Principe de fonctionnement

- Un fichier PHP est un fichier HTML ou XHTML dans lequel on ajouté des instructions PHP placées entre les balises <?php et ?>
- Il est sauvegardé avec l'extension .php
- Lorsque l'utilisateur (un navigateur) demande un fichier PHP au serveur, celui ci commence par vérifier si il existe.
- Si le fichier existe, le serveur le transmet à l'interpréteur PHP qui exécute le code (avec éventuellement des interactions avec une base de données) et remplace le code PHP par le résultat (donc des balises HTML).
- Le serveur renvoie alors le fichier résultat "débarrassé" des instructions PHP, remplacées par leurs résultats.

Principe de fonctionnement

- Un fichier PHP est un fichier HTML ou XHTML dans lequel on ajouté des instructions PHP placées entre les balises `<?php` et `?>`
- Il est sauvegardé avec l'extension `.php`
- Lorsque l'utilisateur (un navigateur) demande un fichier PHP au serveur, celui ci commence par vérifier si il existe.
- Si le fichier existe, le serveur le transmet à l'interpréteur PHP qui exécute le code (avec éventuellement des interactions avec une base de données) et remplace le code PHP par le résultat (donc des balises HTML).
- Le serveur renvoie alors le fichier résultat "débarrassé" des instructions PHP, remplacées par leurs résultats.

Principe de fonctionnement

- Un fichier PHP est un fichier HTML ou XHTML dans lequel on ajouté des instructions PHP placées entre les balises `<?php` et `?>`
- Il est sauvegardé avec l'extension `.php`
- Lorsque l'utilisateur (un navigateur) demande un fichier PHP au serveur, celui ci commence par vérifier si il existe.
- Si le fichier existe, le serveur le transmet à l'interpréteur PHP qui exécute le code (avec éventuellement des interactions avec une base de données) et remplace le code PHP par le résultat (donc des balises HTML).
- Le serveur renvoie alors le fichier résultat "débarrassé" des instructions PHP, remplacées par leurs résultats.

Principe de fonctionnement

- Un fichier PHP est un fichier HTML ou XHTML dans lequel on ajouté des instructions PHP placées entre les balises `<?php` et `?>`
- Il est sauvegardé avec l'extension `.php`
- Lorsque l'utilisateur (un navigateur) demande un fichier PHP au serveur, celui ci commence par vérifier si il existe.
- Si le fichier existe, le serveur le transmet à l'interpréteur PHP qui exécute le code (avec éventuellement des interactions avec une base de données) et remplace le code PHP par le résultat (donc des balises HTML).
- Le serveur renvoie alors le fichier résultat "débarrassé" des instructions PHP, remplacées par leurs résultats.

Principe de fonctionnement

- Un fichier PHP est un fichier HTML ou XHTML dans lequel on ajouté des instructions PHP placées entre les balises `<?php` et `?>`
- Il est sauvegardé avec l'extension `.php`
- Lorsque l'utilisateur (un navigateur) demande un fichier PHP au serveur, celui ci commence par vérifier si il existe.
- Si le fichier existe, le serveur le transmet à l'interpréteur PHP qui exécute le code (avec éventuellement des interactions avec une base de données) et remplace le code PHP par le résultat (donc des balises HTML).
- Le serveur renvoie alors le fichier résultat "débarrassé" des instructions PHP, remplacées par leurs résultats.

Principe de fonctionnement

Source wikipedia

Premier programme

- Voici un premier programme en PHP :

```
<!DOCTYPE ...
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Bonjour</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
</head>
<body>
<?php
  echo "<h1>Bonjour ! </h1>";
?>
</body>
</html>
```

- Le code suivant est renvoyé au navigateur :

Premier programme

- Voici un premier programme en PHP :

```
<!DOCTYPE ...
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Bonjour</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
</head>
<body>
<?php
  echo "<h1>Bonjour ! </h1>";
?>
</body>
</html>
```

- Le code suivant est renvoyé au navigateur :

```
<!DOCTYPE ...
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Bonjour</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
</head>
<body>
<h1>Bonjour ! </h1></body>
</html>
```

Généralités

- Le code est compris entre les balises `<?php` et `?>`
- Les instructions sont séparées par un point-virgule ; comme en C, Java, ...
- Les commentaires sont placés entre les signes `/*` et `*/` ou sur une seule ligne précédée de `//` ou de `#`
- PHP est sensible à la casse (majuscules et minuscules).
- Les blocs de code sont délimités par des accolades { et }

Généralités

- Le code est compris entre les balises <?php et ?>
- Les instructions sont séparées par un point-virgule ; comme en C, Java, ...
- Les commentaires sont placés entre les signes /* et */ ou sur une seule ligne précédée de // ou de #
- PHP est sensible à la casse (majuscules et minuscules).
- Les blocs de code sont délimités par des accolades { et }

Généralités

- Le code est compris entre les balises `<?php` et `?>`
- Les instructions sont séparées par un point-virgule `;` comme en C, Java, ...
- Les commentaires sont placés entre les signes `/*` et `*/` ou sur une seule ligne précédée de `//` ou de `#`
- PHP est sensible à la casse (majuscules et minuscules).
- Les blocs de code sont délimités par des accolades { et }

Généralités

- Le code est compris entre les balises `<?php` et `?>`
- Les instructions sont séparées par un point-virgule `;` comme en C, Java, ...
- Les commentaires sont placés entre les signes `/*` et `*/` ou sur une seule ligne précédée de `//` ou de `#`
- PHP est sensible à la casse (majuscules et minuscules).
- Les blocs de code sont délimités par des accolades { et }

Généralités

- Le code est compris entre les balises `<?php` et `?>`
- Les instructions sont séparées par un point-virgule ; comme en C, Java, ...
- Les commentaires sont placés entre les signes `/*` et `*/` ou sur une seule ligne précédée de `//` ou de `#`
- PHP est sensible à la casse (majuscules et minuscules).
- Les blocs de code sont délimités par des accolades { et }

Les variables : présentation

- Les noms de variables sont précédés du signe $
- Les noms doivent commencer par un caractère non numérique ensuite, ce peut être une combinaison quelconque de chiffres et de lettres.
- PHP reconnaît plusieurs formes de variables : les variables scalaires et les variables agrégats (tableaux et tables de hachages).
- L'opérateur d'affectation est le signe =
- Le typage est implicite, les variables ne sont pas déclarées avant l'utilisation.

Les variables : présentation

- Les noms de variables sont précédés du signe $
- Les noms doivent commencer par un caractère non numérique ensuite, ce peut être une combinaison quelconque de chiffres et de lettres.
- PHP reconnaît plusieurs formes de variables : les variables scalaires et les variables agrégats (tableaux et tables de hachages).
- L'opérateur d'affectation est le signe =
- Le typage est implicite, les variables ne sont pas déclarées avant l'utilisation.

Les variables : présentation

- Les noms de variables sont précédés du signe $
- Les noms doivent commencer par un caractère non numérique ensuite, ce peut être une combinaison quelconque de chiffres et de lettres.
- PHP reconnaît plusieurs formes de variables : les variables scalaires et les variables agrégats (tableaux et tables de hachages).
- L'opérateur d'affectation est le signe =
- Le typage est implicite, les variables ne sont pas déclarées avant l'utilisation.

Les variables : présentation

- Les noms de variables sont précédés du signe $
- Les noms doivent commencer par un caractère non numérique ensuite, ce peut être une combinaison quelconque de chiffres et de lettres.
- PHP reconnaît plusieurs formes de variables : les variables scalaires et les variables agrégats (tableaux et tables de hachages).
- L'opérateur d'affectation est le signe =
- Le typage est implicite, les variables ne sont pas déclarées avant l'utilisation.

Les variables : présentation

- Les noms de variables sont précédés du signe $
- Les noms doivent commencer par un caractère non numérique ensuite, ce peut être une combinaison quelconque de chiffres et de lettres.
- PHP reconnaît plusieurs formes de variables : les variables scalaires et les variables agrégats (tableaux et tables de hachages).
- L'opérateur d'affectation est le signe =
- Le typage est implicite, les variables ne sont pas déclarées avant l'utilisation.

Les variables : les variables numériques

- Deux types numériques sont possibles : les entiers et les flottants.
- Pour les flottants, le symbole . sépare la partie entière de la partie réelle.
- La notation scientifique est reconnue par l'opérateur e
- Le type booléen existe, les deux variables booléennes sont `FALSE` et `TRUE` et sont insensibles à la casse.
- Les opérations usuelles sont reconnues : +, -, /, *, % ainsi que les incrémentations ++ et --

Les variables : les variables numériques

- Deux types numériques sont possibles : les entiers et les flottants.
- Pour les flottants, le symbole . sépare la partie entière de la partie réelle.
- La notation scientifique est reconnue par l'opérateur e
- Le type booléen existe, les deux variables booléennes sont `FALSE` et `TRUE` et sont insensibles à la casse.
- Les opérations usuelles sont reconnues : +, -, /, *, % ainsi que les incrémentations ++ et --

Les variables : les variables numériques

- Deux types numériques sont possibles : les entiers et les flottants.
- Pour les flottants, le symbole . sépare la partie entière de la partie réelle.
- La notation scientifique est reconnue par l'opérateur e
- Le type booléen existe, les deux variables booléennes sont `FALSE` et `TRUE` et sont insensibles à la casse.
- Les opérations usuelles sont reconnues : +, −, /, *, % ainsi que les incrémentations ++ et −−

Les variables : les variables numériques

- Deux types numériques sont possibles : les entiers et les flottants.
- Pour les flottants, le symbole . sépare la partie entière de la partie réelle.
- La notation scientifique est reconnue par l'opérateur e
- Le type booléen existe, les deux variables booléennes sont `FALSE` et `TRUE` et sont insensibles à la casse.
- Les opérations usuelles sont reconnues : +, −, /, *, % ainsi que les incrémentations ++ et −−

Les variables : les variables numériques

- Deux types numériques sont possibles : les entiers et les flottants.
- Pour les flottants, le symbole . sépare la partie entière de la partie réelle.
- La notation scientifique est reconnue par l'opérateur e
- Le type booléen existe, les deux variables booléennes sont `FALSE` et `TRUE` et sont insensibles à la casse.
- Les opérations usuelles sont reconnues : +, -, /, *, % ainsi que les incrémentations ++ et --

```
...
$a=12;
$g=9.81;
$micro=1e-6;
$a++;
$b=15;
$c=$a+$b;          // c contient 28
...
```

Les variables : les chaînes de caractères (1)

- Les chaînes de caractères sont encadrées par des guillemets simples (') ou doubles (").
- Les variables présentent à l'intérieur des guillemets doubles sont interprétées.
- Dans le cas d'ambiguïté lors de l'interprétation des variables, il est possible de les encadrer d'accolades { et }
- Les principaux caractères d'échappement sont reconnus (`\t`, `\n`, `\r`) de plus, les signes $, \ et " doivent être protégés : `\$`, `\\` et `\"`
- L'opérateur . permet de concaténer des valeurs.

Les variables : les chaînes de caractères (1)

- Les chaînes de caractères sont encadrées par des guillemets simples (') ou doubles (").
- Les variables présentent à l'intérieur des guillemets doubles sont interprétées.
- Dans le cas d'ambiguïté lors de l'interprétation des variables, il est possible de les encadrer d'accolades { et }
- Les principaux caractères d'échappement sont reconnus (`\t`, `\n`, `\r`) de plus, les signes $, \ et " doivent être protégés : `\$`, `\\` et `\"`
- L'opérateur . permet de concaténer des valeurs.

Les variables : les chaînes de caractères (1)

- Les chaînes de caractères sont encadrées par des guillemets simples (') ou doubles (").
- Les variables présentent à l'intérieur des guillemets doubles sont interprétées.
- Dans le cas d'ambiguïté lors de l'interprétation des variables, il est possible de les encadrer d'accolades { et }
- Les principaux caractères d'échappement sont reconnus (`\t`, `\n`, `\r`) de plus, les signes $, \ et " doivent être protégés : `\$`, `\\` et `\"`
- L'opérateur . permet de concaténer des valeurs.

Les variables : les chaînes de caractères (1)

- Les chaînes de caractères sont encadrées par des guillemets simples (') ou doubles (").
- Les variables présentent à l'intérieur des guillemets doubles sont interprétées.
- Dans le cas d'ambiguïté lors de l'interprétation des variables, il est possible de les encadrer d'accolades { et }
- Les principaux caractères d'échappement sont reconnus (\t, \n, \r) de plus, les signes $, \ et " doivent être protégés : \$, \\ et \"
- L'opérateur . permet de concaténer des valeurs.

Les variables : les chaînes de caractères (1)

- Les chaînes de caractères sont encadrées par des guillemets simples (') ou doubles (").
- Les variables présentent à l'intérieur des guillemets doubles sont interprétées.
- Dans le cas d'ambiguïté lors de l'interprétation des variables, il est possible de les encadrer d'accolades { et }
- Les principaux caractères d'échappement sont reconnus (\t, \n, \r) de plus, les signes $, \ et " doivent être protégés : \$, \\ et \"
- L'opérateur . permet de concaténer des valeurs.

```
...
$nom = "tux";
$serveur = "@mail.com";
$adresse_mail = $nom.$serveur;
echo "Adresse mail : $adresse_mail";
...
```

Les variables : les chaînes de caractères (2)

- Chaque caractère composant la chaîne peut être accédé en utilisant le nom de la chaîne suivit de l'indice (à partir de 0) entre accolades :

```
$b=$chaine{3}
```

- La longueur d'une chaîne est renvoyée par la fonction `strlen` :

```
$l = strlen($s)
```

- Les chaînes de caractères peuvent être affichées avec les fonctions `echo` et `print` (il existe une petite différence entre les deux)[1].
- La fonction `print_r` permet d'afficher le contenu d'une variable, d'un tableau, d'un objet ou de le copier vers une chaîne de caractères.

[1] http://www.estvideo.com/dew/index/page/phpbench

Les variables : les chaînes de caractères (2)

- Chaque caractère composant la chaîne peut être accédé en utilisant le nom de la chaîne suivit de l'indice (à partir de 0) entre accolades :

```
$b=$chaine{3}
```

- La longueur d'une chaîne est renvoyée par la fonction `strlen` :

```
$l = strlen($s)
```

- Les chaînes de caractères peuvent être affichées avec les fonctions `echo` et `print` (il existe une petite différence entre les deux)[1].
- La fonction `print_r` permet d'afficher le contenu d'une variable, d'un tableau, d'un objet ou de le copier vers une chaîne de caractères.

[1] http://www.estvideo.com/dew/index/page/phpbench

Les variables : les chaînes de caractères (2)

- Chaque caractère composant la chaîne peut être accédé en utilisant le nom de la chaîne suivit de l'indice (à partir de 0) entre accolades :

```
$b=$chaine{3}
```

- La longueur d'une chaîne est renvoyée par la fonction `strlen` :

```
$l = strlen($s)
```

- Les chaînes de caractères peuvent être affichées avec les fonctions `echo` et `print` (il existe une petite différence entre les deux)[1].
- La fonction `print_r` permet d'afficher le contenu d'une variable, d'un tableau, d'un objet ou de le copier vers une chaîne de caractères.

[1] http://www.estvideo.com/dew/index/page/phpbench

Les variables : les chaînes de caractères (2)

- Chaque caractère composant la chaîne peut être accédé en utilisant le nom de la chaîne suivit de l'indice (à partir de 0) entre accolades :

```
$b=$chaine{3}
```

- La longueur d'une chaîne est renvoyée par la fonction `strlen` :

```
$l = strlen($s)
```

- Les chaînes de caractères peuvent être affichées avec les fonctions `echo` et `print` (il existe une petite différence entre les deux)[1].
- La fonction `print_r` permet d'afficher le contenu d'une variable, d'un tableau, d'un objet ou de le copier vers une chaîne de caractères.

[1] http://www.estvideo.com/dew/index/page/phpbench

Les variables : les tableaux indiciés

- Les données peuvent être regroupées en tableaux, les indices sont placés dans des crochets [et]. Les tableaux peuvent être multidimensionnels.
- Le premier élément porte l'indice 0 (comme en C, Java, ...)
- Les données stockées dans un tableau peuvent être hétérogènes (types chaînes et types numériques).
- Lors d'une affectation si aucun indice n'est spécifié entre les crochets, l'élément est placé à la fin du tableau.
- La fonction array peut aussi être utilisée pour initialiser un tableau.

Les variables : les tableaux indiciés

- Les données peuvent être regroupées en tableaux, les indices sont placés dans des crochets [et]. Les tableaux peuvent être multidimensionnels.
- Le premier élément porte l'indice 0 (comme en C, Java, ...)
- Les données stockées dans un tableau peuvent être hétérogènes (types chaînes et types numériques).
- Lors d'une affectation si aucun indice n'est spécifié entre les crochets, l'élément est placé à la fin du tableau.
- La fonction `array` peut aussi être utilisée pour initialiser un tableau.

Les variables : les tableaux indiciés

- Les données peuvent être regroupées en tableaux, les indices sont placés dans des crochets [et]. Les tableaux peuvent être multidimensionnels.
- Le premier élément porte l'indice 0 (comme en C, Java, ...)
- Les données stockées dans un tableau peuvent être hétérogènes (types chaînes et types numériques).
- Lors d'une affectation si aucun indice n'est spécifié entre les crochets, l'élément est placé à la fin du tableau.
- La fonction `array` peut aussi être utilisée pour initialiser un tableau.

Les variables : les tableaux indiciés

- Les données peuvent être regroupées en tableaux, les indices sont placés dans des crochets [et]. Les tableaux peuvent être multidimensionnels.
- Le premier élément porte l'indice 0 (comme en C, Java, ...)
- Les données stockées dans un tableau peuvent être hétérogènes (types chaînes et types numériques).
- Lors d'une affectation si aucun indice n'est spécifié entre les crochets, l'élément est placé à la fin du tableau.
- La fonction `array` peut aussi être utilisée pour initialiser un tableau.

Les variables : les tableaux indiciés

- Les données peuvent être regroupées en tableaux, les indices sont placés dans des crochets `[` et `]`. Les tableaux peuvent être multidimensionnels.
- Le premier élément porte l'indice 0 (comme en C, Java, ...)
- Les données stockées dans un tableau peuvent être hétérogènes (types chaînes et types numériques).
- Lors d'une affectation si aucun indice n'est spécifié entre les crochets, l'élément est placé à la fin du tableau.
- La fonction `array` peut aussi être utilisée pour initialiser un tableau.

Les variables : les tableaux indiciés

- Les données peuvent être regroupées en tableaux, les indices sont placés dans des crochets [et]. Les tableaux peuvent être multidimensionnels.
- Le premier élément porte l'indice 0 (comme en C, Java, ...)
- Les données stockées dans un tableau peuvent être hétérogènes (types chaînes et types numériques).
- Lors d'une affectation si aucun indice n'est spécifié entre les crochets, l'élément est placé à la fin du tableau.
- La fonction `array` peut aussi être utilisée pour initialiser un tableau.

```
...
$tab[0] = "un";
$tab[1] = "deux" ;
$tab[] = 3 ;              // équivalent à $tab[2] = 3
$a = $tab[2]+5;           // a contient 8
$autre_tab = array(1,2,3);
$point[12][32] = "rouge";
...
```

Les variables : les tableaux associatifs

- Les tableaux associatifs (ou tables de hachage) sont des tableaux où la référence n'est pas un indice mais une clef (chaîne ou type numérique).
- Pour le langage, il n'y a pas de différence entre les tableaux indiciés et les tableaux associatifs.
- Ils utilisent la même syntaxe que les tableaux, la fonction `array` peut aussi être utilisée pour construire un tableau associatif, l'opérateur => définie la relation clef=>valeur.

Les variables : les tableaux associatifs

- Les tableaux associatifs (ou tables de hachage) sont des tableaux où la référence n'est pas un indice mais une clef (chaîne ou type numérique).
- Pour le langage, il n'y a pas de différence entre les tableaux indiciés et les tableaux associatifs.
- Ils utilisent la même syntaxe que les tableaux, la fonction `array` peut aussi être utilisée pour construire un tableau associatif, l'opérateur => définie la relation clef=>valeur.

Les variables : les tableaux associatifs

- Les tableaux associatifs (ou tables de hachage) sont des tableaux où la référence n'est pas un indice mais une clef (chaîne ou type numérique).
- Pour le langage, il n'y a pas de différence entre les tableaux indiciés et les tableaux associatifs.
- Ils utilisent la même syntaxe que les tableaux, la fonction `array` peut aussi être utilisée pour construire un tableau associatif, l'opérateur => définie la relation clef=>valeur.

```
$livres = array ( "Le seigneur des anneaux" => "Tolkien" ,
                  "L'appel de Cthulhu" => "Lovecraft" ,
                  "Les fourmis" => "Werber" ,
                  "La ligne verte" => "King");
...
$livres["Harry Potter à l'école des sorciers"] = "Rowling";
...
$titre = "Les fourmis";
echo "L'auteur du livre ".$titre." est : ".$livres[$titre];
```

Les variables : les constantes

- Il est possible d'associer un symbole à une constante inaltérable.
- Ces constantes sont souvent utilisées pour stocker des paramètres dans un programme.
- Par convention, les noms de constantes sont en majuscules pour les différencier des variables.
- La commande `define` permet de définir une constante que ce soit un type numérique ou une chaîne de caractères

Les variables : les constantes

- Il est possible d'associer un symbole à une constante inaltérable.
- Ces constantes sont souvent utilisées pour stocker des paramètres dans un programme.
- Par convention, les noms de constantes sont en majuscules pour les différencier des variables.
- La commande `define` permet de définir une constante que ce soit un type numérique ou une chaîne de caractères

Les variables : les constantes

- Il est possible d'associer un symbole à une constante inaltérable.
- Ces constantes sont souvent utilisées pour stocker des paramètres dans un programme.
- Par convention, les noms de constantes sont en majuscules pour les différencier des variables.
- La commande `define` permet de définir une constante que ce soit un type numérique ou une chaîne de caractères

Les variables : les constantes

- Il est possible d'associer un symbole à une constante inaltérable.
- Ces constantes sont souvent utilisées pour stocker des paramètres dans un programme.
- Par convention, les noms de constantes sont en majuscules pour les différencier des variables.
- La commande `define` permet de définir une constante que ce soit un type numérique ou une chaîne de caractères

Les variables : les constantes

- Il est possible d'associer un symbole à une constante inaltérable.
- Ces constantes sont souvent utilisées pour stocker des paramètres dans un programme.
- Par convention, les noms de constantes sont en majuscules pour les différencier des variables.
- La commande `define` permet de définir une constante que ce soit un type numérique ou une chaîne de caractères

```
define("PI", 3.14159);
define("SERVEUR", "serveur.fr");
```

Les variables : fonctions utiles

- La fonction `isset` est utilisée pour tester l'existence d'une variable.
- La fonction `empty` permet de savoir si une variable contient une valeur non nulle (`''`, `""`, `"0"`, `NULL`, `FALSE`, un tableau vide...sont des éléments vides).
- La fonction `unset` permet de détruire une variable.
- Les fonctions `is_XXXX` (`is_int`,`is_float`, `...`) renseignent sur le type d'une variable.

Les variables : fonctions utiles

- La fonction `isset` est utilisée pour tester l'existence d'une variable.
- La fonction `empty` permet de savoir si une variable contient une valeur non nulle (`''`, `""`, `"0"`, `NULL`, `FALSE`, un tableau vide. . . sont des éléments vides).
- La fonction `unset` permet de détruire une variable.
- Les fonctions `is_XXXX` (`is_int`,`is_float`, `...`) renseignent sur le type d'une variable.

Les variables : fonctions utiles

- La fonction `isset` est utilisée pour tester l'existence d'une variable.
- La fonction `empty` permet de savoir si une variable contient une valeur non nulle (‘‘’’, ’’, ‘‘0’’, `NULL`, `FALSE`, un tableau vide. . . sont des éléments vides).
- La fonction `unset` permet de détruire une variable.
- Les fonctions `is_XXXX` (`is_int`,`is_float`, `...`) renseignent sur le type d'une variable.

Les variables : fonctions utiles

- La fonction `isset` est utilisée pour tester l'existence d'une variable.
- La fonction `empty` permet de savoir si une variable contient une valeur non nulle (‘‘’’, ’’, ‘‘0’’, `NULL`, `FALSE`, un tableau vide. . . sont des éléments vides).
- La fonction `unset` permet de détruire une variable.
- Les fonctions `is_XXXX` (`is_int`,`is_float`, `...`) renseignent sur le type d'une variable.

Les structures du langage : les instructions de test `if...else`

- La structure `if...else` permet de réaliser des tests.
- Les tests usuels sont possibles (==, >, <, >=, <=, !=) et peuvent être combinés avec les opérateurs booléens ET (&&), OU (||), OU-EXCLUSIF (`xor`)
- Il est possible de tester l'égalité de type avant de l'égalité de valeur avec l'opérateur identité ===
- Des tests imbriqués (`if...else...if...else`) peuvent être écrits `elseif`

Les structures du langage : les instructions de test `if...else`

- La structure `if...else` permet de réaliser des tests.
- Les tests usuels sont possibles (==, $>$, $<$, $>=$, $<=$, !=) et peuvent être combinés avec les opérateurs booléens ET (&&), OU (||), OU-EXCLUSIF (`xor`)
- Il est possible de tester l'égalité de type avant de l'égalité de valeur avec l'opérateur identité ===
- Des tests imbriqués (`if...else...if...else`) peuvent être écrits `elseif`

Les structures du langage : les instructions de test `if...else`

- La structure `if...else` permet de réaliser des tests.
- Les tests usuels sont possibles (==, $>$, $<$, $>=$, $<=$, !=) et peuvent être combinés avec les opérateurs booléens ET (&&), OU (||), OU-EXCLUSIF (`xor`)
- Il est possible de tester l'égalité de type avant de l'égalité de valeur avec l'opérateur identité ===
- Des tests imbriqués (`if...else...if...else`) peuvent être écrits `elseif`

Les structures du langage : les instructions de test `if...else`

- La structure `if...else` permet de réaliser des tests.
- Les tests usuels sont possibles (==, $>$, $<$, $>=$, $<=$, !=) et peuvent être combinés avec les opérateurs booléens ET (&&), OU (||), OU-EXCLUSIF (`xor`)
- Il est possible de tester l'égalité de type avant de l'égalité de valeur avec l'opérateur identité ===
- Des tests imbriqués (`if...else...if...else`) peuvent être écrits `elseif`

Les structures du langage : les instructions de test `if...else`

- La structure `if...else` permet de réaliser des tests.
- Les tests usuels sont possibles (==, >, <, >=, <=, !=) et peuvent être combinés avec les opérateurs booléens ET (&&), OU (||), OU-EXCLUSIF (`xor`)
- Il est possible de tester l'égalité de type avant de l'égalité de valeur avec l'opérateur identité ===
- Des tests imbriqués (`if...else...if...else`) peuvent être écrits `elseif`

```
$note=12.9;
if ($note<10){
   echo "<p>Vous êtes recalé</p>";
}elseif ( ($note>=12) && ($note<14) ){
   echo "<p>Mention bien</p>";
}elseif ( ($note>=14) && ($note<16) ){
   echo "<p>Mention très bien</p>";
}elseif ( $note>=16 ){
   echo "<p>Mention très bien avec félicications du jury</p>";
}else{
   echo "<p>Mention passable<p>";
}
```

Les structures du langage : les instructions de test `switch...case`

- Les instructions de tests imbriquées peuvent être écrites en utilisant une structure `switch...case`.
- Cette structure permet de faire des choix parmi des valeurs prédéfinies, ce peut être des valeurs numériques ou des chaînes de caractères.

Les structures du langage : les instructions de test `switch...case`

- Les instructions de tests imbriquées peuvent être écrites en utilisant une structure `switch...case`.
- Cette structure permet de faire des choix parmi des valeurs prédéfinies, ce peut être des valeurs numériques ou des chaînes de caractères.

Les structures du langage : les instructions de test `switch...case`

- Les instructions de tests imbriquées peuvent être écrites en utilisant une structure `switch...case`.
- Cette structure permet de faire des choix parmi des valeurs prédéfinies, ce peut être des valeurs numériques ou des chaînes de caractères.

```
...
   switch ($menu) {
   case "entrée":
     echo "Choisissez une entrée : <br />";
     break;
   case "plat":
     echo "Choisissez un plat <br />";
     break;
   case "dessert":
     echo "Choisissez un dessert <br />";
     break;
   default:
         echo "Vous allez bientot être servi ... <br />";
   }
...
```

Les structures du langage : les boucles `for`

- La boucle la plus simple est la boucle `for`.
- Elle a exactement la même syntaxe qu'en C, Java, ... : `for(exp1 ;exp2 ;exp3){...}`
- Au départ de la boucle l'expression `exp1` est traitée. A chaque début d'itération, le contenu de l'expression `exp2` est évalué, s'il est vrai, le contenu des accolades est effectué puis l'expression `exp3` est traitée.

Les structures du langage : les boucles `for`

- La boucle la plus simple est la boucle `for`.
- Elle a exactement la même syntaxe qu'en C, Java, ... : `for(exp1 ;exp2 ;exp3){...}`
- Au départ de la boucle l'expression `exp1` est traitée. A chaque début d'itération, le contenu de l'expression `exp2` est évalué, s'il est vrai, le contenu des accolades est effectué puis l'expression `exp3` est traitée.

Les structures du langage : les boucles `for`

- La boucle la plus simple est la boucle `for`.
- Elle a exactement la même syntaxe qu'en C, Java, ... : `for(exp1 ;exp2 ;exp3){...}`
- Au départ de la boucle l'expression `exp1` est traitée. A chaque début d'itération, le contenu de l'expression `exp2` est évalué, s'il est vrai, le contenu des accolades est effectué puis l'expression `exp3` est traitée.

```
...
   for($i=0 ; $i<8; $i++){
      $puissance = pow(2, $i);
      echo "2 puissance $i vaut : $puissance <br />";
   }
...
```

Les structures du langage : les boucles `foreach`

- La boucle `foreach($tab as $value){...}` permet de parcourir tous les éléments d'un tableau indicié de manière rapide.
- La boucle commence par le premier élément du tableau, accessible par la variable `$value`, puis, à chaque itération, l'élément suivant du tableau est pris en compte. La boucle s'arrête quand tous les éléments ont été parcourus.
- Pour un tableau associatif la syntaxe est `foreach($tab as $key=>$value){...}`

Les structures du langage : les boucles `foreach`

- La boucle `foreach($tab as $value){...}` permet de parcourir tous les éléments d'un tableau indicié de manière rapide.
- La boucle commence par le premier élément du tableau, accessible par la variable `$value`, puis, à chaque itération, l'élément suivant du tableau est pris en compte. La boucle s'arrête quand tous les éléments ont été parcourus.
- Pour un tableau associatif la syntaxe est `foreach($tab as $key=>$value){...}`

Les structures du langage : les boucles `foreach`

- La boucle `foreach($tab as $value){...}` permet de parcourir tous les éléments d'un tableau indicié de manière rapide.
- La boucle commence par le premier élément du tableau, accessible par la variable `$value`, puis, à chaque itération, l'élément suivant du tableau est pris en compte. La boucle s'arrête quand tous les éléments ont été parcourus.
- Pour un tableau associatif la syntaxe est `foreach($tab as $key=>$value){...}`

Les structures du langage : les boucles `foreach`

- La boucle `foreach($tab as $value){...}` permet de parcourir tous les éléments d'un tableau indicié de manière rapide.
- La boucle commence par le premier élément du tableau, accessible par la variable `$value`, puis, à chaque itération, l'élément suivant du tableau est pris en compte. La boucle s'arrête quand tous les éléments ont été parcourus.
- Pour un tableau associatif la syntaxe est `foreach($tab as $key=>$value){...}`

```
...
$livres = array ( "Le seigneur des anneaux" => "Tolkien" ,
                  "L'appel de Cthulhu" => "Lovecraft" ,
                  "Les fourmis" => "Werber" ,
                  "La ligne verte" => "King");
foreach($livres as $titre=>$auteur){
   echo "<strong>$auteur</strong> a écrit <em>$titre</em> <br />";
}
...
```

Les structures du langage : les boucles `foreach`

- Si on modifie la variable `$value`, les modifications ne se répercutent pas sur le tableau.
- Il faut faire un passage par référence pour pouvoir modifier le tableau en ajoutant & lors de la déclaration : `foreach($tab as &$value){...}` ou `foreach($tab as $key=>&$value){...}`

Les structures du langage : les boucles `foreach`

- Si on modifie la variable `$value`, les modifications ne se répercutent pas sur le tableau.
- Il faut faire un passage par référence pour pouvoir modifier le tableau en ajoutant `&` lors de la déclaration : `foreach($tab as &$value){...}` ou `foreach($tab as $key=>&$value){...}`

Les structures du langage : les boucles `foreach`

- Si on modifie la variable `$value`, les modifications ne se répercutent pas sur le tableau.
- Il faut faire un passage par référence pour pouvoir modifier le tableau en ajoutant & lors de la déclaration : `foreach($tab as &$value){...}` ou `foreach($tab as $key=>&$value){...}`

```
...
$tableau = array(1, 2, 3, 4, 5, 6);
foreach($tableau as $valeur) {
   echo "$valeur   ";
}
echo "<br />";
foreach ($tableau as &$val) {
    $val = $val * 2;
}
foreach($tableau as $valeur) {
   echo "$valeur    ";
}
echo "<br />";
...
```

Les structures du langage : les boucles `while` et `do...while`

- Pour répéter un bloc d'instruction tant qu'une expression est vraie, on utilise les boucles `while` et `do...while`
- Dans la boucle `while(expression){...}`, l'expression est testée avant chaque itération, dans la boucle `do{...}while(expression)`, l'expression est testée après chaque itération.

Les structures du langage : les boucles `while` et `do...while`

- Pour répéter un bloc d'instruction tant qu'une expression est vraie, on utilise les boucles `while` et `do...while`
- Dans la boucle `while(expression){...}`, l'expression est testée avant chaque itération, dans la boucle `do{...}while(expression)`, l'expression est testée après chaque itération.

Les structures du langage : les boucles `while` et `do...while`

- Pour répéter un bloc d'instruction tant qu'une expression est vraie, on utilise les boucles `while` et `do...while`
- Dans la boucle `while(expression){...}`, l'expression est testée avant chaque itération, dans la boucle `do{...}while(expression)`, l'expression est testée après chaque itération.

```
...
do{
   $i=rand();
}while($i%2==1);
echo "<p>Un nombre aléatoire paire : $i </p>";
...
```

Les fonctions : présentation

- Une fonction est déclarée avec le mot-clef `function` suivit d'un bloc d'instructions.
- Il est possible de passer des paramètres et de renvoyer une valeur.
- Pour retourner une valeur, on utilise le mot-clef `return`. Le type de retour d'une fonction n'est pas spécifié dans sa déclaration.

Les fonctions : présentation

- Une fonction est déclarée avec le mot-clef `function` suivit d'un bloc d'instructions.
- Il est possible de passer des paramètres et de renvoyer une valeur.
- Pour retourner une valeur, on utilise le mot-clef `return`. Le type de retour d'une fonction n'est pas spécifié dans sa déclaration.

Les fonctions : présentation

- Une fonction est déclarée avec le mot-clef `function` suivit d'un bloc d'instructions.
- Il est possible de passer des paramètres et de renvoyer une valeur.
- Pour retourner une valeur, on utilise le mot-clef `return`. Le type de retour d'une fonction n'est pas spécifié dans sa déclaration.

Les fonctions : présentation

- Une fonction est déclarée avec le mot-clef `function` suivit d'un bloc d'instructions.
- Il est possible de passer des paramètres et de renvoyer une valeur.
- Pour retourner une valeur, on utilise le mot-clef `return`. Le type de retour d'une fonction n'est pas spécifié dans sa déclaration.

```
function addition ($a, $b)
{
   $somme = $a + $b ;
   return $somme;
}
// ...
$u=1;
$v=2;
$w = addition($u, $v);
// ...
```

Les fonctions : les paramètres

- Par défaut, les variables sont passées par valeur, si elles sont altérées à l'intérieur de la fonction, elles gardent leurs valeurs hors de la fonction.
- Pour pouvoir modifier une variable, elle doit être passée par référence. Lors de la déclaration, on précède le nom de la variable du signe & dans la déclaration de fonction.
- Il est possible de prédéfinir une valeur pour chaque paramètre.

Les fonctions : les paramètres

- Par défaut, les variables sont passées par valeur, si elles sont altérées à l'intérieur de la fonction, elles gardent leurs valeurs hors de la fonction.
- Pour pouvoir modifier une variable, elle doit être passée par référence. Lors de la déclaration, on précède le nom de la variable du signe & dans la déclaration de fonction.
- Il est possible de prédéfinir une valeur pour chaque paramètre.

Les fonctions : les paramètres

- Par défaut, les variables sont passées par valeur, si elles sont altérées à l'intérieur de la fonction, elles gardent leurs valeurs hors de la fonction.
- Pour pouvoir modifier une variable, elle doit être passée par référence. Lors de la déclaration, on précède le nom de la variable du signe & dans la déclaration de fonction.
- Il est possible de prédéfinir une valeur pour chaque paramètre.

```
function addition ($a, $b, &$s)
{
   $s = $a + $b ;
}
// ...
$u=1;
$v=2;
$w=5;
addition($u, $v, $w);       // $w vaut maintenant 3
// ...
```

Les fonctions : les paramètres

- Par défaut, les variables sont passées par valeur, si elles sont altérées à l'intérieur de la fonction, elles gardent leurs valeurs hors de la fonction.
- Pour pouvoir modifier une variable, elle doit être passée par référence. Lors de la déclaration, on précède le nom de la variable du signe & dans la déclaration de fonction.
- Il est possible de prédéfinir une valeur pour chaque paramètre.

Les fonctions : les paramètres

- Par défaut, les variables sont passées par valeur, si elles sont altérées à l'intérieur de la fonction, elles gardent leurs valeurs hors de la fonction.
- Pour pouvoir modifier une variable, elle doit être passée par référence. Lors de la déclaration, on précède le nom de la variable du signe & dans la déclaration de fonction.
- Il est possible de prédéfinir une valeur pour chaque paramètre.

```
function Connexion ($user, $pass, $serveur="chezmoi.com")
{
   // ...
}
//...
// Connexion au serveur monautreserrveur.org
Connexion("jb","toto","monautreserrveur.org");
//...
// Connexion au serveur chezmoi.com
Connexion("jb","toto");
```

Les fonctions : les variables statiques

- A chaque appel d'une fonction les variables internes sont réinitialisées.
- Les variables statiques conservent leurs valeurs entre chaque appels de la fonction.
- Elles ne sont visibles qu'à l'intérieur de la fonction dans laquelle elles ont été déclarées.
- Avant d'utiliser une variable statique on doit la déclarer à l'aide du mot-clef `static` :

Les fonctions : les variables statiques

- A chaque appel d'une fonction les variables internes sont réinitialisées.
- Les variables statiques conservent leurs valeurs entre chaque appels de la fonction.
- Elles ne sont visibles qu'à l'intérieur de la fonction dans laquelle elles ont été déclarées.
- Avant d'utiliser une variable statique on doit la déclarer à l'aide du mot-clef `static` :

Les fonctions : les variables statiques

- A chaque appel d'une fonction les variables internes sont réinitialisées.
- Les variables statiques conservent leurs valeurs entre chaque appels de la fonction.
- Elles ne sont visibles qu'à l'intérieur de la fonction dans laquelle elles ont été déclarées.
- Avant d'utiliser une variable statique on doit la déclarer à l'aide du mot-clef `static` :

Les fonctions : les variables statiques

- A chaque appel d'une fonction les variables internes sont réinitialisées.
- Les variables statiques conservent leurs valeurs entre chaque appels de la fonction.
- Elles ne sont visibles qu'à l'intérieur de la fonction dans laquelle elles ont été déclarées.
- Avant d'utiliser une variable statique on doit la déclarer à l'aide du mot-clef `static` :

Les fonctions : les variables statiques

- A chaque appel d'une fonction les variables internes sont réinitialisées.
- Les variables statiques conservent leurs valeurs entre chaque appels de la fonction.
- Elles ne sont visibles qu'à l'intérieur de la fonction dans laquelle elles ont été déclarées.
- Avant d'utiliser une variable statique on doit la déclarer à l'aide du mot-clef `static` :

```
function compteur(){
   static $n=0;
   $n++;
   echo "La fonction a été appelée : ".$n." fois. <br />";
}
```

Les fonctions : les variables globales

- La visibilité réduite des variables peut être modifiée en utilisant des variables globales.
- Le mot clef `global` permet de définir une variable globale.
- Les variables globales doivent être utilisées de manière parcimonieuse. Elles rendent les scripts peu lisibles et peu sécurisés.

Les fonctions : les variables globales

- La visibilité réduite des variables peut être modifiée en utilisant des variables globales.
- Le mot clef `global` permet de définir une variable globale.
- Les variables globales doivent être utilisées de manière parcimonieuse. Elles rendent les scripts peu lisibles et peu sécurisés.

```
...
global $user;
...
```

Les fonctions : les variables globales

- La visibilité réduite des variables peut être modifiée en utilisant des variables globales.
- Le mot clef `global` permet de définir une variable globale.
- Les variables globales doivent être utilisées de manière parcimonieuse. Elles rendent les scripts peu lisibles et peu sécurisés.

```
...
global $user;
...
```

Les fonctions : inclusion de scripts

- Pour organiser un site, on sépare les fonctions en différents fichiers.
- Quatre fonctions permettent d'inclure des fichiers : `require`, `require_once`, `include` et `include_once`.
- Les instructions `require` et `require_once` conduisent à l'arrêt du script en cas d'erreur dans un script inclus alors que les instructions `include` et `include_once` ne provoquent qu'une alerte (donc le script continu).
- Lors d'imbrications multiples le problème de la redéclaration peut se poser. Pour éviter ceci, les fonctions `require_once` et `include_once` vérifient que le fichier n'a pas déjà été inclus avant de l'inclure.

Les fonctions : inclusion de scripts

- Pour organiser un site, on sépare les fonctions en différents fichiers.
- Quatre fonctions permettent d'inclure des fichiers : `require`, `require_once`, `include` et `include_once`.
- Les instructions `require` et `require_once` conduisent à l'arrêt du script en cas d'erreur dans un script inclus alors que les instructions `include` et `include_once` ne provoquent qu'une alerte (donc le script continu).
- Lors d'imbrications multiples le problème de la redéclaration peut se poser. Pour éviter ceci, les fonctions `require_once` et `include_once` vérifient que le fichier n'a pas déjà été inclus avant de l'inclure.

Les fonctions : inclusion de scripts

- Pour organiser un site, on sépare les fonctions en différents fichiers.
- Quatre fonctions permettent d'inclure des fichiers : `require`, `require_once`, `include` et `include_once`.
- Les instructions `require` et `require_once` conduisent à l'arrêt du script en cas d'erreur dans un script inclus alors que les instructions `include` et `include_once` ne provoquent qu'une alerte (donc le script continu).
- Lors d'imbrications multiples le problème de la redéclaration peut se poser. Pour éviter ceci, les fonctions `require_once` et `include_once` vérifient que le fichier n'a pas déjà été inclus avant de l'inclure.

Les fonctions : inclusion de scripts

- Pour organiser un site, on sépare les fonctions en différents fichiers.
- Quatre fonctions permettent d'inclure des fichiers : `require`, `require_once`, `include` et `include_once`.
- Les instructions `require` et `require_once` conduisent à l'arrêt du script en cas d'erreur dans un script inclus alors que les instructions `include` et `include_once` ne provoquent qu'une alerte (donc le script continu).
- Lors d'imbrications multiples le problème de la redéclaration peut se poser. Pour éviter ceci, les fonctions `require_once` et `include_once` vérifient que le fichier n'a pas déjà été inclus avant de l'inclure.

Les fonctions : inclusion de scripts

- Pour organiser un site, on sépare les fonctions en différents fichiers.
- Quatre fonctions permettent d'inclure des fichiers : `require`, `require_once`, `include` et `include_once`.
- Les instructions `require` et `require_once` conduisent à l'arrêt du script en cas d'erreur dans un script inclus alors que les instructions `include` et `include_once` ne provoquent qu'une alerte (donc le script continu).
- Lors d'imbrications multiples le problème de la redéclaration peut se poser. Pour éviter ceci, les fonctions `require_once` et `include_once` vérifient que le fichier n'a pas déjà été inclus avant de l'inclure.

```
...
include("Menu.php");
...
```

Les formulaires

- Les tableaux super-globaux `$_GET` et `$_POST` sont des tableaux associatifs contenant les variables définies dans les formulaires.
- Chaque tableau est associé à la méthode d'envoi du même nom.
- Pour les éléments homogènes à des tableaux le nom doit se terminer par `[]` pour être transformé en tableau.
- Il existe un tableau `$_FILES` qui contient les fichiers téléchargés via la méthode POST du client vers le serveur.

Les formulaires

- Les tableaux super-globaux `$_GET` et `$_POST` sont des tableaux associatifs contenant les variables définies dans les formulaires.
- Chaque tableau est associé à la méthode d'envoi du même nom.
- Pour les éléments homogènes à des tableaux le nom doit se terminer par `[]` pour être transformé en tableau.
- Il existe un tableau `$_FILES` qui contient les fichiers téléchargés via la méthode POST du client vers le serveur.

Les formulaires

- Les tableaux super-globaux `$_GET` et `$_POST` sont des tableaux associatifs contenant les variables définies dans les formulaires.
- Chaque tableau est associé à la méthode d'envoi du même nom.
- Pour les éléments homogènes à des tableaux le nom doit se terminer par `[]` pour être transformé en tableau.
- Il existe un tableau `$_FILES` qui contient les fichiers téléchargés via la méthode POST du client vers le serveur.

```
...
<form action="result.php" method="get">
   <input type="text" name="Nom" value="Saisissez votre nom" size="25" maxlength="25" />
   <p>Quels sont vos loisirs ?<br />
   <input type="checkbox" name="Loisir[]" value="Cinema" />Cinéma<br />
   <input type="checkbox" name="Loisir[]" value="Lecture" />Lecture<br />
   <input type="checkbox" name="Loisir[]" value="Musique" />Musique<br />
   <input type="checkbox" name="Loisir[]" value="Sport" />Sport<br />
   </p>
   <input type="submit" />
</form>
...
```

Les formulaires

- Les tableaux super-globaux `$_GET` et `$_POST` sont des tableaux associatifs contenant les variables définies dans les formulaires.
- Chaque tableau est associé à la méthode d'envoi du même nom.
- Pour les éléments homogènes à des tableaux le nom doit se terminer par `[]` pour être transformé en tableau.
- Il existe un tableau `$_FILES` qui contient les fichiers téléchargés via la méthode POST du client vers le serveur.

```
...
<p>
<h1>Fiche de renseignements</h1>
<?php
   $nom = $_GET["Nom"];
   echo("<p>");
   echo "Bonjour ".$nom." !<br />";
   echo "Vos loisir sont : <br />";
   foreach($_GET["Loisir"] as $valeur){
      echo "$valeur <br />";
   }
?>
</p>
...
```

Les formulaires

- Les tableaux super-globaux `$_GET` et `$_POST` sont des tableaux associatifs contenant les variables définies dans les formulaires.
- Chaque tableau est associé à la méthode d'envoi du même nom.
- Pour les éléments homogènes à des tableaux le nom doit se terminer par `[]` pour être transformé en tableau.
- Il existe un tableau `$_FILES` qui contient les fichiers téléchargés via la méthode POST du client vers le serveur.

Présentation générale

- Le protocole HTTP impose une déconnexion après l'envoi d'une page.
- Il est nécessaire de pouvoir passer des informations d'une page à l'autre.
- Deux mécanismes sont possibles en PHP :
  1. Les cookies stockés sur le poste client,
  2. [illegible]

Présentation générale

- Le protocole HTTP impose une déconnexion après l'envoi d'une page.
- Il est nécessaire de pouvoir passer des informations d'une page à l'autre.
- Deux mécanismes sont possibles en PHP :
  - Les cookies stockés sur le poste client,
  - Les sessions supportées par le serveur.

Présentation générale

- Le protocole HTTP impose une déconnexion après l'envoi d'une page.
- Il est nécessaire de pouvoir passer des informations d'une page à l'autre.
- Deux mécanismes sont possibles en PHP :
  - Les cookies stockés sur le poste client,
  - Les sessions supportées par le serveur.

Présentation générale

- Le protocole HTTP impose une déconnexion après l'envoi d'une page.
- Il est nécessaire de pouvoir passer des informations d'une page à l'autre.
- Deux mécanismes sont possibles en PHP :
  - Les cookies stockés sur le poste client,
  - Les sessions supportées par le serveur.

Présentation générale

- Le protocole HTTP impose une déconnexion après l'envoi d'une page.
- Il est nécessaire de pouvoir passer des informations d'une page à l'autre.
- Deux mécanismes sont possibles en PHP :
  - Les cookies stockés sur le poste client,
  - Les sessions supportées par le serveur.

Les cookies : présentation

- Un cookie est un petit fichier texte (ou la partie d'un fichier texte) qui permet de stocker des informations simples sur le poste client.
- Les cookies sont envoyés par le serveur dans les entêtes HTTP à l'aide de la directive `Set-Cookie`.
- Un cookie a normalement une date de validité qui assure sa suppression par le navigateur.
- Les cookies sont souvent utilisés pour garder les préférences de l'utilisateur entre deux visites du site.

Les cookies : présentation

- Un cookie est un petit fichier texte (ou la partie d'un fichier texte) qui permet de stocker des informations simples sur le poste client.
- Les cookies sont envoyés par le serveur dans les entêtes HTTP à l'aide de la directive `Set-Cookie`.
- Un cookie a normalement une date de validité qui assure sa suppression par le navigateur.
- Les cookies sont souvent utilisés pour garder les préférences de l'utilisateur entre deux visites du site.

Les cookies : présentation

- Un cookie est un petit fichier texte (ou la partie d'un fichier texte) qui permet de stocker des informations simples sur le poste client.
- Les cookies sont envoyés par le serveur dans les entêtes HTTP à l'aide de la directive `Set-Cookie`.
- Un cookie a normalement une date de validité qui assure sa suppression par le navigateur.
- Les cookies sont souvent utilisés pour garder les préférences de l'utilisateur entre deux visites du site.

Les cookies : présentation

- Un cookie est un petit fichier texte (ou la partie d'un fichier texte) qui permet de stocker des informations simples sur le poste client.
- Les cookies sont envoyés par le serveur dans les entêtes HTTP à l'aide de la directive `Set-Cookie`.
- Un cookie a normalement une date de validité qui assure sa suppression par le navigateur.
- Les cookies sont souvent utilisés pour garder les préférences de l'utilisateur entre deux visites du site.

Les cookies : utilisation avec PHP

- La fonction `setcookie` permet de créer un cookie, elle doit être appelée avant les entêtes HTTP.
- Cette fonction a 6 paramètres tous optionnels sauf le premier :
  - `$name` : le nom du cookie,
  - `$value` : la valeur du cookie,
  - `$expires` : la date d'expiration du cookie au format timestamp Unix,
  - `$path` : le domaine de validité du cookie (répertoire ou sous-répertoire possible),
  - `$domain` : le nom de domaine de validité du site,
  - `$secure` : permet de crypter l'envoi cookie à l'aide du protocole HTTPS si égale à 1.

Les cookies : utilisation avec PHP

- La fonction `setcookie` permet de créer un cookie, elle doit être appelée avant les entêtes HTTP.
- Cette fonction a 6 paramètres tous optionnels sauf le premier :
  - `$name` : le nom du cookie,
  - `$value` : la valeur du cookie,
  - `$expires` : la date d'expiration du cookie au format timestamp Unix,
  - `$path` : le domaine de validité du cookie (répertoire ou sous répertoire possible),
  - `$domain` : le nom de domaine de validité du site,
  - `$secure` : permet de crypter l'envoi cookie à l'aide du protocole HTTPS si égale à 1.

Les cookies : utilisation avec PHP

- La fonction `setcookie` permet de créer un cookie, elle doit être appelée avant les entêtes HTTP.
- Cette fonction a 6 paramètres tous optionnels sauf le premier :
  - `$name` : le nom du cookie,
  - `$value` : la valeur du cookie,
  - `$expires` : la date d'expiration du cookie au format timestamp Unix,
  - `$path` : le domaine de validité du cookie (répertoire ou sous répertoire possible),
  - `$domain` : le nom de domaine de validité du site,
  - `$secure` : permet de crypter l'envoi cookie à l'aide du protocole HTTPS si égale à 1.

Les cookies : utilisation avec PHP

- La fonction `setcookie` permet de créer un cookie, elle doit être appelée avant les entêtes HTTP.
- Cette fonction a 6 paramètres tous optionnels sauf le premier :
  - `$name` : le nom du cookie,
  - `$value` : la valeur du cookie,
  - `$expires` : la date d'expiration du cookie au format timestamp Unix,
  - `$path` : le domaine de validité du cookie (répertoire ou sous répertoire possible),
  - `$domain` : le nom de domaine de validité du site,
  - `$secure` : permet de crypter l'envoi cookie à l'aide du protocole HTTPS si égale à 1.

Les cookies : utilisation avec PHP

- La fonction `setcookie` permet de créer un cookie, elle doit être appelée avant les entêtes HTTP.
- Cette fonction a 6 paramètres tous optionnels sauf le premier :
  - `$name` : le nom du cookie,
  - `$value` : la valeur du cookie,
  - `$expires` : la date d'expiration du cookie au format timestamp Unix,
  - `$path` : le domaine de validité du cookie (répertoire ou sous répertoire possible),
  - `$domain` : le nom de domaine de validité du site,
  - `$secure` : permet de crypter l'envoi cookie à l'aide du protocole HTTPS si égale à 1.

Les cookies : utilisation avec PHP

- La fonction `setcookie` permet de créer un cookie, elle doit être appelée avant les entêtes HTTP.
- Cette fonction a 6 paramètres tous optionnels sauf le premier :
  - `$name` : le nom du cookie,
  - `$value` : la valeur du cookie,
  - `$expires` : la date d'expiration du cookie au format timestamp Unix,
  - `$path` : le domaine de validité du cookie (répertoire ou sous répertoire possible),
  - `$domain` : le nom de domaine de validité du site,
  - `$secure` : permet de crypter l'envoi cookie à l'aide du protocole HTTPS si égale à 1.

Les cookies : utilisation avec PHP

- La fonction `setcookie` permet de créer un cookie, elle doit être appelée avant les entêtes HTTP.
- Cette fonction a 6 paramètres tous optionnels sauf le premier :
  - `$name` : le nom du cookie,
  - `$value` : la valeur du cookie,
  - `$expires` : la date d'expiration du cookie au format timestamp Unix,
  - `$path` : le domaine de validité du cookie (répertoire ou sous répertoire possible),
  - `$domain` : le nom de domaine de validité du site,
  - `$secure` : permet de crypter l'envoi cookie à l'aide du protocole HTTPS si égale à 1.

Les cookies : utilisation avec PHP

- La fonction `setcookie` permet de créer un cookie, elle doit être appelée avant les entêtes HTTP.
- Cette fonction a 6 paramètres tous optionnels sauf le premier :
  - `$name` : le nom du cookie,
  - `$value` : la valeur du cookie,
  - `$expires` : la date d'expiration du cookie au format timestamp Unix,
  - `$path` : le domaine de validité du cookie (répertoire ou sous répertoire possible),
  - `$domain` : le nom de domaine de validité du site,
  - `$secure` : permet de crypter l'envoi cookie à l'aide du protocole HTTPS si égale à 1.

```
...
   setcookie("themes", "rouge", time()+30*24*60*60, "", "", 0);    // cookie valable 30 jours
...
```

Les cookies : utilisation avec PHP

- La lecture des cookies est très facile, PHP construit un tableau super global `$_COOKIE` qui contient tous les cookies associés à la page en cours.
- Pour supprimer un cookie il suffit de le renvoyer avec une date dépassée.

Les cookies : utilisation avec PHP

- La lecture des cookies est très facile, PHP construit un tableau super global `$_COOKIE` qui contient tous les cookies associés à la page en cours.
- Pour supprimer un cookie il suffit de le renvoyer avec une date dépassée.

```
...
   echo $_COOKIE["themes"];       // affiche "rouge"
...
```

Les cookies : utilisation avec PHP

- La lecture des cookies est très facile, PHP construit un tableau super global `$_COOKIE` qui contient tous les cookies associés à la page en cours.
- Pour supprimer un cookie il suffit de le renvoyer avec une date dépassée.

Les cookies : utilisation avec PHP

- La lecture des cookies est très facile, PHP construit un tableau super global `$_COOKIE` qui contient tous les cookies associés à la page en cours.
- Pour supprimer un cookie il suffit de le renvoyer avec une date dépassée.

```
...
   setcookie("themes", "rouge", time()-200); // cookie valable 30 jours
...
```

Les cookies : avantages et limitations

- Les cookies sont faciles d'emploi pour stocker rapidement des informations.
- Les informations sont stockées en clair, des informations "sensibles" ne doivent donc pas être stockées à l'aide de cookies.
- De même, les cookies peuvent être copiés d'un poste à un autre ce qui peut conduire à des usurpations d'identités.
- L'utilisateur peut configurer son navigateur pour refuser les cookies.

Les cookies : avantages et limitations

- Les cookies sont faciles d'emploi pour stocker rapidement des informations.
- Les informations sont stockées en clair, des informations "sensibles" ne doivent donc pas être stockées à l'aide de cookies.
- De même, les cookies peuvent être copiés d'un poste à un autre ce qui peut conduire à des usurpations d'identités.
- L'utilisateur peut configurer son navigateur pour refuser les cookies.

Les cookies : avantages et limitations

- Les cookies sont faciles d'emploi pour stocker rapidement des informations.
- Les informations sont stockées en clair, des informations "sensibles" ne doivent donc pas être stockées à l'aide de cookies.
- De même, les cookies peuvent être copiés d'un poste à un autre ce qui peut conduire à des usurpations d'identités.
- L'utilisateur peut configurer son navigateur pour refuser les cookies.

Les cookies : avantages et limitations

- Les cookies sont faciles d'emploi pour stocker rapidement des informations.
- Les informations sont stockées en clair, des informations "sensibles" ne doivent donc pas être stockées à l'aide de cookies.
- De même, les cookies peuvent être copiés d'un poste à un autre ce qui peut conduire à des usurpations d'identités.
- L'utilisateur peut configurer son navigateur pour refuser les cookies.

Les sessions : généralités

- Les sessions sont un mécanisme plus sûr de stockage des informations.
- Les informations sont stockées sur le serveur et un identifiant de session unique est associé au client
- L'identifiant est envoyé sous la forme d'un cookie si le navigateur les accepte.
- Sinon, l'identifiant est passé comme paramètre dans l'adresse (méthode GET, variable `PHPSESSID`).

Les sessions : généralités

- Les sessions sont un mécanisme plus sûr de stockage des informations.
- Les informations sont stockées sur le serveur et un identifiant de session unique est associé au client
- L'identifiant est envoyé sous la forme d'un cookie si le navigateur les accepte.
- Sinon, l'identifiant est passé comme paramètre dans l'adresse (méthode GET, variable `PHPSESSID`).

Les sessions : généralités

- Les sessions sont un mécanisme plus sûr de stockage des informations.
- Les informations sont stockées sur le serveur et un identifiant de session unique est associé au client
- L'identifiant est envoyé sous la forme d'un cookie si le navigateur les accepte.
- Sinon, l'identifiant est passé comme paramètre dans l'adresse (méthode GET, variable `PHPSESSID`).

Les sessions : généralités

- Les sessions sont un mécanisme plus sûr de stockage des informations.
- Les informations sont stockées sur le serveur et un identifiant de session unique est associé au client
- L'identifiant est envoyé sous la forme d'un cookie si le navigateur les accepte.
- Sinon, l'identifiant est passé comme paramètre dans l'adresse (méthode GET, variable `PHPSESSID`).

Les sessions : utilisation avec PHP

- Une session dure tant que le navigateur est ouvert ou en fonction de la valeur `session.lifetime` dans le fichier de configuration de PHP.
- Pour commencer une session on appelle la fonction `session_start()` avant les entêtes HTTP dans chaque page où l'on doit accéder aux valeurs stockées.
- Les variables sont ensuite accessibles via un tableau super global `$_SESSION` accessible dans chaque fichier.
- Une session peut être détruite en utilisant la méthode `session_destroy()`

Les sessions : utilisation avec PHP

- Une session dure tant que le navigateur est ouvert ou en fonction de la valeur `session.lifetime` dans le fichier de configuration de PHP.
- Pour commencer une session on appelle la fonction `session_start()` avant les entêtes HTTP dans chaque page où l'on doit accéder aux valeurs stockées.
- Les variables sont ensuite accessibles via un tableau super global `$_SESSION` accessible dans chaque fichier.
- Une session peut être détruite en utilisant la méthode `session_destroy()`

Les sessions : utilisation avec PHP

- Une session dure tant que le navigateur est ouvert ou en fonction de la valeur `session.lifetime` dans le fichier de configuration de PHP.
- Pour commencer une session on appelle la fonction `session_start()` avant les entêtes HTTP dans chaque page où l'on doit accéder aux valeurs stockées.
- Les variables sont ensuite accessibles via un tableau super global `$_SESSION` accessible dans chaque fichier.
- Une session peut être détruite en utilisant la méthode `session_destroy()`

Les sessions : utilisation avec PHP

- Une session dure tant que le navigateur est ouvert ou en fonction de la valeur `session.lifetime` dans le fichier de configuration de PHP.
- Pour commencer une session on appelle la fonction `session_start()` avant les entêtes HTTP dans chaque page où l'on doit accéder aux valeurs stockées.
- Les variables sont ensuite accessibles via un tableau super global `$_SESSION` accessible dans chaque fichier.
- Une session peut être détruite en utilisant la méthode `session_destroy()`

Les sessions : exemple

- Réalisation d'un *style switcher* à partir de sessions :

```
<?php
session_start();
...
if ($_SESSION["themes"] = ""){
   $_SESSION["themes"] = "red";
}
...
?>
<head>
...
<?php
if ($_SESSION["themes"] === "red"){
   echo '<link href="ThemeRouge.css" type="text/css"
   rel="stylesheet" media="screen"/>';
}elseif ($_SESSION["themes"] === "blue"){
   echo '<link href="ThemeBleu.css" type="text/css"
   rel="stylesheet" media="screen"/>';
}...
?>
</head>
...
```