

PHP



Historique

- 1995 : Première mouture de PHP/FI créée par Rasmus Lerdorf.
 - ✓ PHP/FI : Personal Home Page/Forms Interpreter, dont l'objectif était, entre autres, de permettre à son auteur de savoir qui venait consulter son CV sur son site personnel.
 - ✓ Bibliothèque de scripts fonctionnant sous Perl
 - ✓ La bibliothèque Perl s'est muée en implémentation directement en langage C => gains de performances et communication avec les bases de données, créer des applications dynamique WEB.
- 2
 - ✓ Rasmus Lerdorf propose son code à la communauté des développeurs.



Historique (suite)

- 1997: PHP/FI 2.0, version bêta
- 1997-1998: PHP 3, entrée sur scène d'Andi Gutschman et Zeev Suraski fondateurs de Zend. Ils décidèrent de réécrire de façon complète PHP/FI. Rasmus rejoint le projet PHP 3.0.
 - PHP devient Hypertext Preprocessor
 - Intégration des extensions grâce à une API modulaire qui donne la possibilité à un développeur de créer ses modules et les partager avec la communauté

Cette version devenait un langage de programmation.
10% du parc mondial utilise PHP comme serveur Web.

- Version stable mais faibles performances.



Historique (suite)

- 1999-2000: Sortie de PHP 4.0 avec un moteur (le Zend Engine) plus puissant.
 - Prise en des sessions HTTP.
 - Prise en charge de nombreux serveurs Web.
 - La mise en tampon des sorties.
 - Une sécurité accrue des informations visiteurs



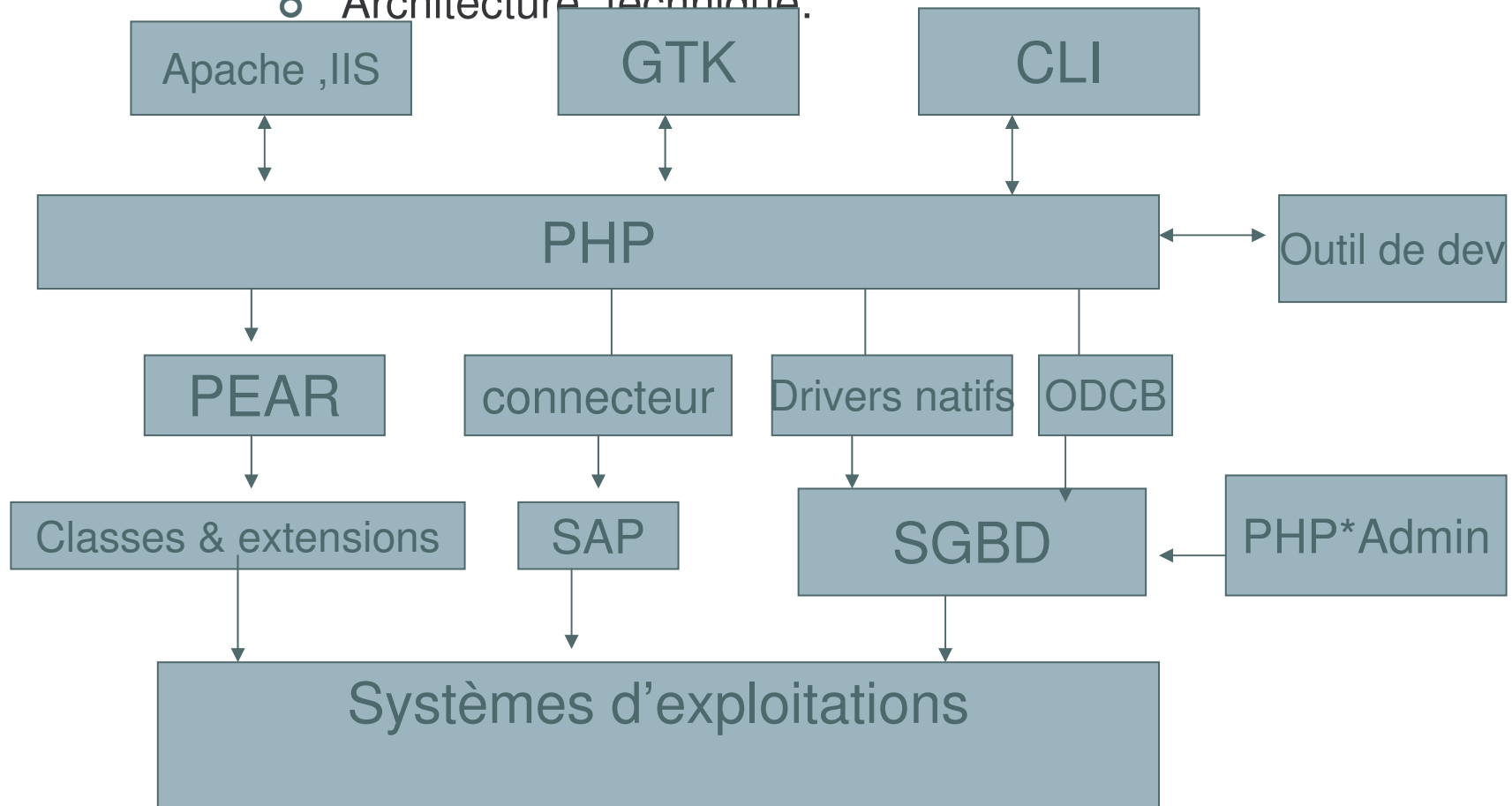
Historique (suite)

- 2002-2003: PHP 5 est en gestation.
- 2004: La sortie de la première version stable de PHP 5.
 - Intégration du Zend Engine 2.
 - Prise en charge de la programmation orientée objet.
 - Simplification des tâches courante.
 - Prise en charge améliorée de XML.
 - Intégration de la base de donnée SQLite.



Architecture et fonctionnement

o Architecture technique:





Architecture et fonctionnement

- Architecture technique (suite)
 - PEAR (PHP Extension and Application Repository): bibliothèque structurée du code open source.
 - Il a été conçu par des développeurs indépendants.
 - ✓ Le code d'un projet est distribué en paquetages.
 - ✓ Les développeurs (extensions, applications) doivent respecter le codage et les méthodes standards de PEAR pour la gestion des erreurs, inclusion de la documentation dans le package.



Architecture et fonctionnement

- Architecture technique (suite)
 - ODBC (Open Database Connectivity)
 - En plus du support de l'ODBC normal, l'ODBC unifié de PHP vous donne accès à diverses bases de données qui ont emprunté la sémantique des API ODBC pour implémenter leur propres API. Au lieu de maintenir de multiples pilotes qui sont similaires, ces pilotes ont été rassemblés dans un jeu de fonctions ODBC uniques.



Architecture et fonctionnement

- Architecture technique (suite)

- SGBD

- Oracle .
 - PostgreSQL.
 - MySQL .
 - Direct MS-SQL.
 - ODBC.
 - Sybase
 - mSQL.
 - Informix.
 - Sybase .
 - Etc.



Architecture et fonctionnement

- Architecture technique (suite)
 - Outils d'administration
 - ✓ PostgreMyadmin.
 - ✓ PhpMyAdmin.
 - ✓ PhpOracleAdmin.
 - ✓ PhpSybaseAdmin.
 - ✓ Etc.

● Architecture et fonctionnement

○ Architecture technique (suite)

▪ Capacités de PHP

Il est capable de générer:

- Des images
- des fichiers PDF
- des animations Flash
- texte, du code XML ou XHTML



Architecture et fonctionnement

- Architecture technique (suite)

PHP supporte de nombreux protocoles:

- LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM
- PHP supporte le format complexe WDDX, qui permet de communiquer entre tous les langages web
- PHP supporte aussi les instanciations d'objets Java, et les utilise de manière transparente comme objets intégrés. Vous pouvez aussi exploiter les objets distants



Architecture et fonctionnement

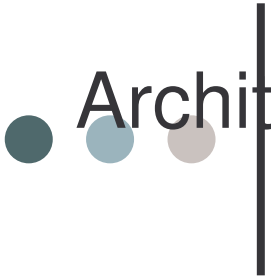
- Architecture technique (suite)

Commerce électronique:

- outils de paiement intégrés comme Cybercash, CyberMut, VeriSign Payflow Pro et C CVS, pour réaliser des paiements en ligne.

- Et d'autres choses:

PHP dispose d'extensions très pratiques comme le moteur de recherche mnoGoSearch, la passerelle avec IRC, des outils de compression (gzip, bz2) et de conversion calendaire, de traduction



Architecture et fonctionnement

- Architecture technique (suite)

Les plates-formes en production reposent, généralement, sur le quatuor Linux, Apache, MySQL et PHP (LAMP:Linux, Apache,MySQL, PHP)

- Serveur HTTP, généralement Apache. IIS rarement.
- GTK (Graphic Tool Kit): client riche.
- CLI (Command Line Interpreter ou interface).
- Les principaux outils de développement: PHPEdit, PHP Editor, Zend Studio, Glade (GTK) Dreamweaver Mx...



Différence entre PHP et HTML

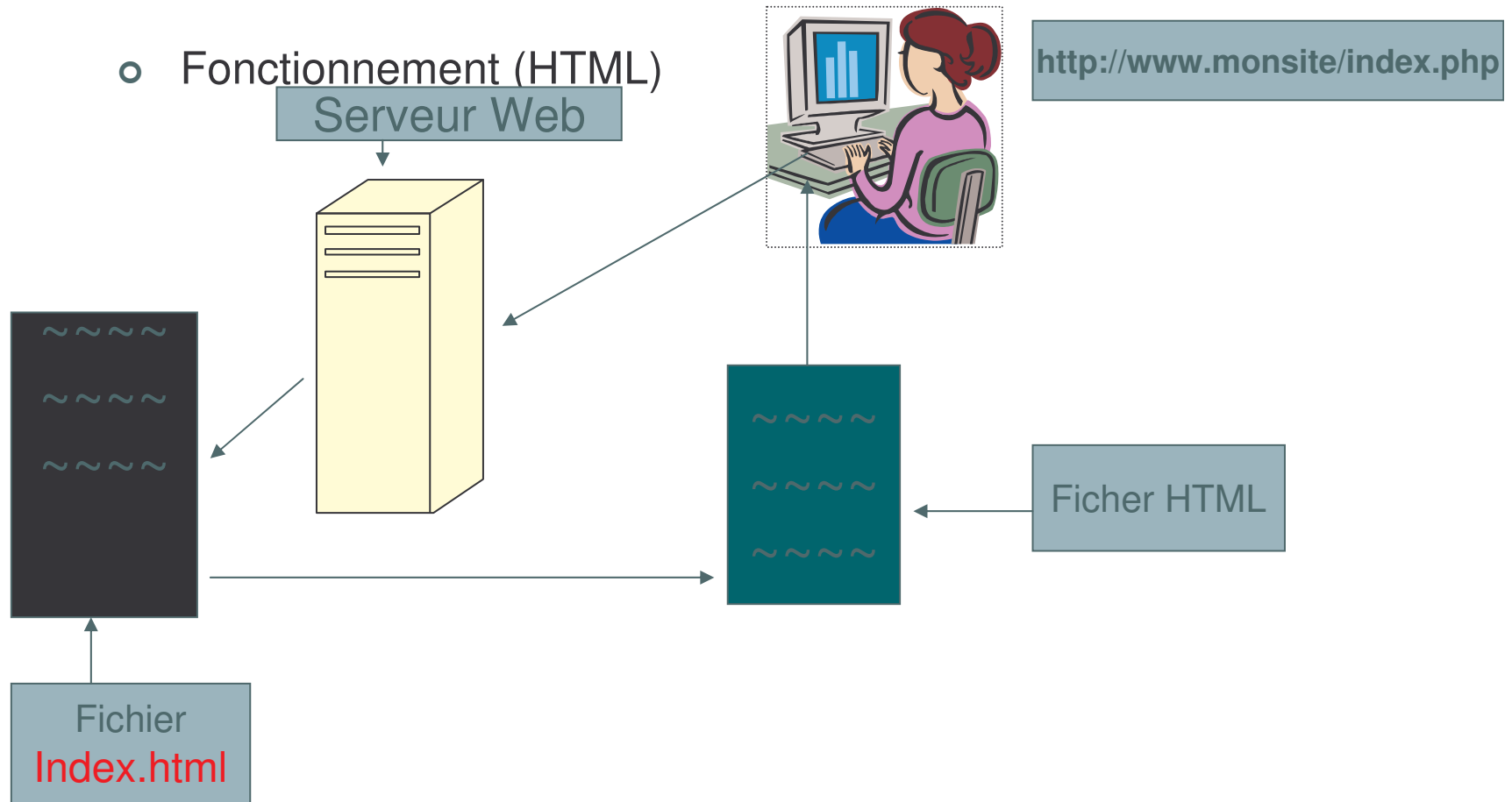
PHP est exécuté côté serveur.

HTML est exécuté coté client par le navigateur.

On va pouvoir générer des pages dynamiques côté serveur.

Architecture et fonctionnement

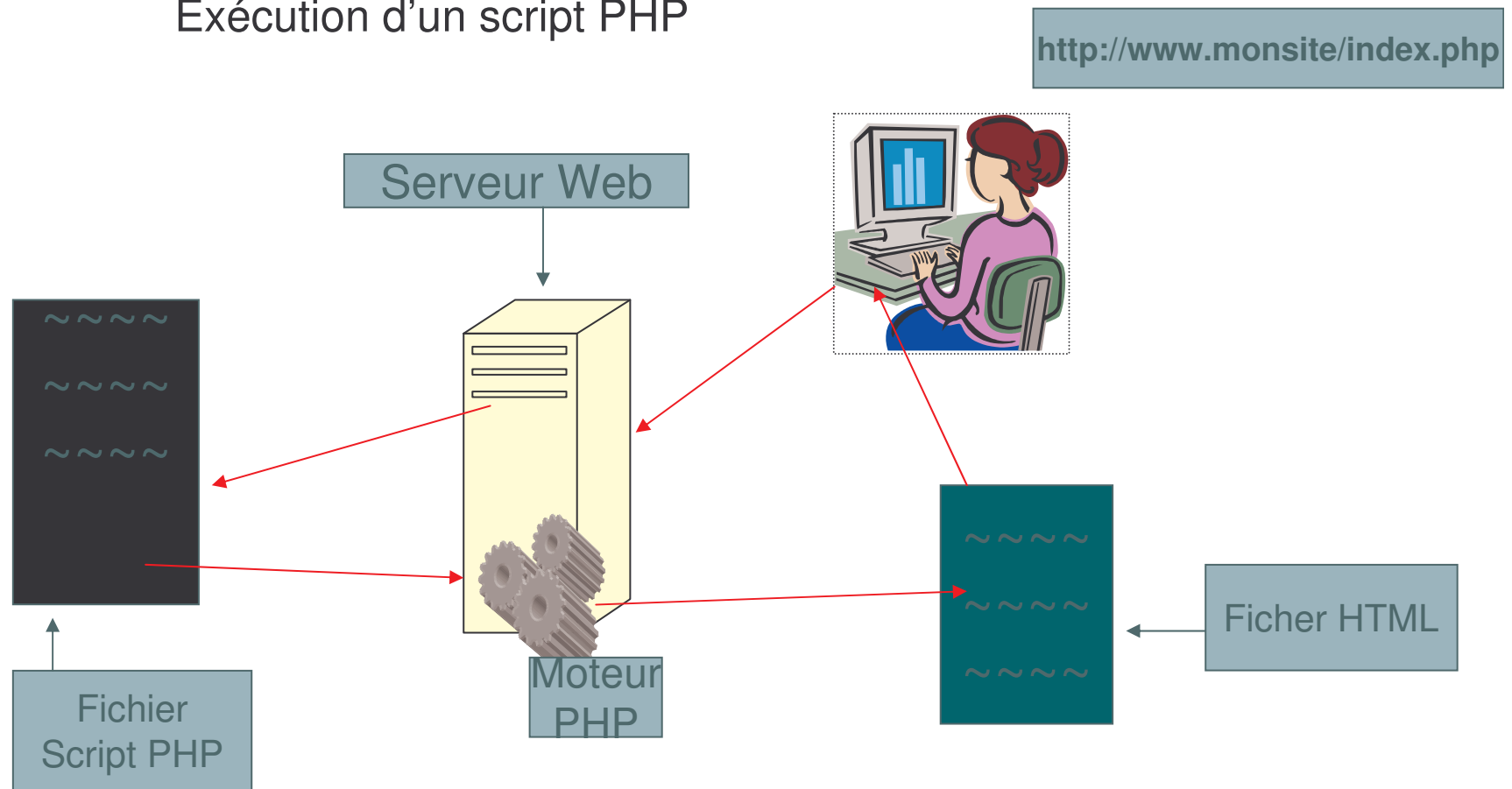
- Fonctionnement (HTML)



Architecture et fonctionnement

- Fonctionnement

 - Exécution d'un script PHP



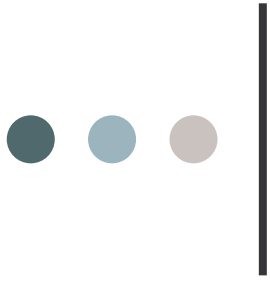
Les structures de base

- Insertion de PHP dans HTML

Le code PHP peut être directement intégré dans les fichiers HTML. Il figure à différents endroits de ces fichiers, tout en étant entrecoupé de code HTML.

Exemple:

```
<html>
  <head>
    <title>Mon script PHP</title>
  </head>
  <body>
    <?php
      echo "Hello world !>";
    ?>
  </body>
</html>
```



LES STRUCTURES

DE BASE



Les structures de base

- Insertion de PHP dans HTML (suite)

l'exécution de l'exemple produit:

```
<html>  
  <head>  
    <title>Mon script PHP</title>  
  </head>  
  
  <body>  
    Hello world !  
  
  </body>  
</html>
```



Les structures de base

- Balise d'ouverture et de fermeture

Ouverture	Fermeture
<?	?>
<?php	?>
<script language= " php " >	</script>



Les structures de base

- Les commentaires

Comme avec le langage C, on peut utiliser deux façons:

`/* commentaire*/` : plusieurs lignes

✓ `// commentaire` ou `# commentaire` : une seule ligne.

Rem: l'utilisation de `#` est possible mais désuète .

Exemple:

```
<?php
```

```
/*addition
```

```
De deux nombres */
```

```
<p> cinq plus cinq =
```

```
print 5+5;
```

22

```
</p>
```

```
// le résultat doit être 10 !
```



Les structures de base

- Séquences d'échappement pour chaîne de caractères entre guillemets doubles.

Séquence d'échappement

`\n`

`\r`

`\t`

`\\`

`\$`

`\“`

`\{`

`\}`

`\[`

`\]`

`\0 à \777`

`\x0 à \xFF`

caractères

saut de ligne (ASCII 10)

retour chariot (ASCII 13)

tabulation (ASCII 9)

back slash

signe dollar

guillemet

accolade à gauche

accolade à droite

crochet à gauche

crochet à droite

nombre octal

nombre hexadécimal

● ● ● | Les structures de base

○ Les variables

- ✓ une variable, pour simplifier, peut être représentée comme un récipient disponible pendant toute l'exécution du programme (case mémoire).
- ✓ Elle peut recevoir des valeurs, les modifier et les utiliser.
- ✓ Affectation d'une valeur par le symbole =.

Syntaxe:

```
$nom_var i a b l e
```

Exemple :

```
<?php
```

```
$temps = "le ciel est bleu !"
```

```
?>
```




Les structures de base

- Les variables (suite)

PHP n'est pas un langage déclaratif !. Donc ...

Règles à respecter:

- ✓ Tous les noms de variables commencent par un signe \$.
- ✓ Le nom de variable peut être de n'importe quelle longueur.
- ✓ Un nom de variable peut comporter des lettres, des chiffres et des underscores.
- ✓ Un nom de variable doit commencer par une lettre ou underscore, mais jamais par un chiffre!.
- ✓ Les minuscules et les majuscules ne sont pas considérées comme identiques



Les structures de base

- Les variables (suite)

Créer des variables:

Correct	Incorrect	explication
<code>\$variable</code>	<code>\$variable 1</code>	Contient des espaces
<code>\$variable</code>	<code>Variable</code>	Une variable doit commencer par \$
<code>\$variable_ d</code>	<code>\$variable-d</code>	Le signe – est interdit
<code>\$var_email</code>	<code>\$var@yahoo.fr</code>	Les caractères @ et . Sont interdits
<code>\$var</code>	<code>\$2var</code>	Une variable ne commence pas par un chiffre



Les structures de base

- Les variables dynamiques ou variable de variables

Le d'une variable peut lui-même être une variable.

Exemple:

```
<?php
    $nom="ville";
    $$nom="Paris";// remarquer le signe $ supplémentaire
    // $ville = " Paris" mais $nom ne change pas.
    $cd = " 15 £" ;
    $dvd = " 30 £" ;
    $produit= "dvd" ; // on choisit comme produit le dvd
    echo $$produit; // affiche 30£

// on peut référencer un nom dynamique en utilisant les accolades
    echo ${"dvd"};// affiche 30£
// Il est possible de faire des opérations à l'intérieur des accolades
    echo ${"d"."vd"}; //affiche 30£
?>
```



Les structures de base

- Variables placées dans des chaînes de caractères

- ✓ Interpolation de variable.

Exemple:

```
<?php
```

```
$email="toto@yahoo.fr";
```

```
Print (" Envoyer les réponses à :$mail");
```

```
// on obtient : Envoyer les réponses à : toto@yahoo.fr
```

```
?>
```

- ✓ Interpolation avec accolades

```
<?php
```

```
    $cuisson = 'flambé';
```

```
    $ingredients='Banane';
```

```
Print "$ingredients {$cuisson}es à la chantilly";// donne bananes flambées ..
```

```
?>
```



Les structures de base

- Portée des variables
 - ✓ La durée de vie d'une variables est égale au plus à celle du script où elle est définie. La fin du script entraîne la perte de la valeur des variables.
 - ✓ Il est donc impossible de stocker une valeur dans une variable pour la relire dans un autre script.
 - ✓ Variables locales: ne sont accessibles qu'au module où elle sont définie.
 - ✓ Variable globales: une variable définie à l'extérieur d'un module n'est pas accessible à ce dernier.
 - La solution consiste à utiliser le tableau `$GLOBALS[]` ou le type global.
 - Voir cours sur les fonctions.



Les structures de base

- Quelques variables prédéfinies (voir doc)
 - ✓ Se présentent sous forme de tableaux associatifs.
 - ✓ Les éléments de ces tableaux sont accessibles par des indices qui sont des variables prédéfinies.



Les structures de base

- ***constante***

Est définie à l'aide de la fonction [define\(\)](#). Son type est soit [bool](#), [integer](#), [double](#) ou [string](#).

Exemple :

```
<?
```

```
define ("maconstante" , "PHP EST  
SIMLE !");
```

```
echo maconstante;
```

```
?>
```



Les structures de base

- Constante (suite)
 - ✓ Les constantes ne commencent pas par le signe (\$);
 - ✓ Les constantes sont définies et accessibles à tout endroit du code, globalement.
 - ✓ Les constantes ne peuvent pas être redéfinies ou indéfinies une fois qu'elles ont été définies.
 - ✓ Les constantes **ne peuvent contenir que des scalaires.**
 - ✓ Pour accéder au contenu d'une constante on utilise le nom de la constante sans \$
 - ✓ pour connaître la liste de toutes les constantes définies **get defined constants()** .



Les structures de base

- Opérateurs arithmétiques

opérateur	opération	exemple
+	Addition	$\$a+\b
-	Soustraction	$\$a-\b
*	Multiplication	$\$a*\b
/	division	$\$a/\b
%	modulo	$\$a/\b

Les structures de base

- Opérateurs relationnels ()

- ✓ résultat est vrai (TRUE) ou faux (FALSE).
- ✓ Utilisés dans des expressions de test.

opérateur	signification	exemple
==	égal	\$a == \$b
===	identique	\$a === \$b
!=	Égal à, en type et en valeur	\$a != \$b
!==	Différent en valeur mais égal en type	\$a !== \$b
<	inférieur	\$a < \$b
>	supérieur	\$a > \$b
<=	Inférieur ou égal	\$a <= \$b
>=	Supérieur ou égal	\$a >= \$b

Les structures de base

- Opérateurs logiques
 - ✓ Connecteurs d'expressions logiques

opérateur	signification	exemple
&&	Et	\$a && \$b
AND	Et	\$a AND \$b (*)
	Ou	\$a \$b
OR	Ou	\$a OR \$b (*)
XOR	Ou exclusif	\$a XOR \$b
!	négation	!\$a ne renvoie pas TRUE

(*) AND ET OR ont une priorité faible, && et || ont une priorité forte

Les structures de base

- Opérateurs binaires

opérateur	signification	exemple
&	ET	$\$a \& \b
	OU	$\$a \b
<<	Décalage à gauche	$\$a \ll \b
>>	Décalage à droite	$\$a \gg \b
^	XOR	$\$a \wedge \b
~	Négation	$\sim \$a$



Les structures de base

- Opérateurs d'incrémentation, décrémentation.

++\$a: la variable est incrémenté puis évaluée.

\$a++: la variable est évaluée puis incrémentée.

--\$a: la variable est décrémentation puis évaluée.

\$a--: la variable est évaluée puis décrémentation.

opérateur	opération	exemple
++	incrémentati on	\$a++ ou ++\$a
--	décrémentat ion	\$a-- ou --\$a



Les structures de base

- Opérateurs combinés

X un opérateur de $\{+, -, *, /, \&, \%, |, \ll, \gg, \sim, \cdot\}$
 $a X b$ est équivalent à $a = a X b$;

Exemple:

$a * b$; est équivalent à $a = a * b$;

$a \cdot b$; est équivalent à $a = a \cdot b$;



Les structures de base

- o La concaténation

opération qui permet d'empiler des informations dans une variable.

opérateur	Opération	exemple
.	concaténation	Echo \$a.\$b
.=	Concaténation et assignation	\$a.= \$b



Les structures de base

- Concaténation (suite)

exemple

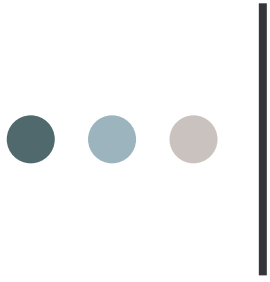
```
<?php
```

```
    $a='salut';
```

```
    $a.=' '. 'toto';
```

```
    echo $a;
```

```
?>
```

LES TYPES DE DONNEES



Les types de données

- **Entiers**

Entiers naturels ou relatifs.

Exemple:

<?

```
$a = 1234; // nombre entier en base 10
```

```
$a = -123; // nombre entier négatif
```

```
$a = 0123; // nombre entier en base 8,
```

```
$a = 0x1A; // nombre entier en base 16,
```

...

?>



Les types de données

○ REELS

Les nombres à virgule flottante, la virgule flote pour préciser le nombre de chiffres après la virgule.

Exemple:

```
<?php
    $r = 1.234;
    $T = 1.2e3;
?>
```



Les types de données

- REELS (suite)

Conversion automatique de type

Exemple:

```
<?php
    $var = "0"; // $var est une chaîne de caractères (ASCII
48)
    $var += 3; // $var est maintenant du type entier (3)
    $var = $var + 1.3; // $var est maintenant du type
double (3.3)
    $var = 5 + "10 canards sauvages"; // $var est du
type entier (15)
)
?>
```



Les types de données

○ **Booleen**

- ✓ Résultat d'une évaluation d'une expression logique (oui/non)
- ✓ les constantes '**TRUE**' et '**FALSE**' pour spécifier une valeur de type **booleen**. Ces constantes sont insensibles à la casse.



Exemple:

```
<?php
    $logique = True;
    // assigne la valeur TRUE à la variable $logique
```

45

```
?>
```



Les types de données

- Chaînes de caractères

Une chaîne peut être spécifiée entre guillemets simple ou double.

Exemple:

```
<?php
    echo 'Ceci est une chaîne simple';
    echo ' Quand reviens-tu? : I'll be back next
week!';
?>
```



Les types de données

- Chaînes de caractères (suite)
- ✓ Interprétation des variables: à l'intérieur d'une chaîne entre guillemets, les variables sont automatiquement remplacées par leur valeur.

Exemple:

```
<?php
```

```
$texte="roman";
```

```
$auteur="Popov";
```

```
$annonce=" $nom son $texte est célèbre!";
```

```
Echo $annonce;
```

```
// affiche: Popov son roman est célèbre
```

```
?>
```



Les types de données

- Chaînes de caractères (suite)
 - ✓ Le caractère d'échappement (\)
Permet de protéger les caractères interpréter par PHP.

Exemple:

```
<?php
$string="Le film \"Anna et ses soeurs\" est...";
echo $string;
// affiche: Le film "Anna et ses soeurs est ...
$monnaie="Le \$ Australien est différent du \$ USA";
// affiche: Le $ Australien est différent du $ USA
echo $monnaie;
Echo " Pour afficher un antislash (\\) il faut le protéger!";
?>
```




Les types de données

- Chaînes de caractères (suite)

- ✓ Délimitation par apostrophes

En utilisant l'apostrophe (') comme délimiteur de chaîne de caractères, tous les autres caractères peuvent être mis directement dans la chaîne et seul le caractère (') doit être protégé

Exemple

```
<?php
echo 'le film "Anna es ses soeurs" a coûté 250.000 $';
//affiche: le film "Anna es ses soeurs" a coûté 250.000 $
$objet="toto";
echo "\n $objet";
//affiche: \n $objet
echo 'l\'apostrophe';
//affiche: l'apostrophe
echo 'L\'antislash: \'';
// affiche: L'antislash: \
?>
```



Les types de données

- Chaînes de caractères (suite)

- ✓ La syntaxe heredoc:

Cette syntaxe se comporte exactement comme une chaîne à guillemet.

Pas d'échappement des caractères,

Les variables sont remplacées par leur valeur.

Exemple:

```
<?php
```

```
  $var = "PHP";
```

```
  $texte=<<<fin
```

dans la syntaxe heredoc le texte

est délimité à l'ouverture par 3 symboles < et un identifiant
.

le texte est fermé par une ligne ne contenant que l'identifiant

avec la syntaxe heredoc "\$var" plus de guillemets et apostrophes
comme délimiteurs!

le délimiteur de fin ne doit pas contenir d'espace ou d'indentation, et
seul avec point virgule!

```
fin;
```

- echo \$texte;

- ?>



Les types de données

- Chaînes de caractères (suite)
 - ✓ Un caractère de la chaîne est accessible en le référençant par sa position dans la chaîne entre accolade et le nom de la chaîne.

Exemple:

```
<?php
$texte="php est simple!";
Echo $texte{1};
// affiche :h
Les index commence à 0.
?>
```



Les types de données

- Les tableaux (array)

Regroupement de données sous le même nom.

Deux types:

- ✓ Indicés (numérique);
- ✓ Associatifs (chaîne).
- ✓ La déclaration de la taille n'est pas nécessaire. Elle est gérée par PHP.
- ✓ Le contenu de chaque case est accessible en utilisant Le nom du tableau suivi de son indice entre crochets.



Les types de données

- Les tableaux (suite)
- ✓ Un tableau en PHP est ,donc,une association ordonnée qui fait correspondre des valeurs à des *clés*.
- ✓ Une valeur peut elle-même être un tableau.
- ✓ Un tableau peut être crée à l'aide de la fonction **ARRAY()**
- ✓ Cette fonction prend en argument des structures *clé* => *value*, séparées par des virgules.
`array([cle =>] valeur , ...)`
key est soit une **chaîne** , soit un **entier** positif

Les types de données

- Les tableaux (suite)

✓ Les tableaux indexés numériquement.

Exemple:

```
<?php
    $var="toto";
    $semaine=array(lundi,mardi,mercredi,jeudi,vendredi,samedi);
    echo $semaine[0]."<br>",$semaine[5]."<br>";
    //affiche lundi et samedi
    $tableau=array("orange",2005,$var,1235.12);
    $ref= $tableau[2]; // $ref=$var
    echo $tableau[0]."<br>";//affiche :orange
    echo $tableau[2]."<br>";//affiche :toto et non $var
    echo $ref; //affiche:toto
?>
```

Les types de données



- Tableau indexé numériquement (suite)

Il est possible d'écrire directement dans le tableau.

Exemple:

```
<?php
```

```
$tableau[0]="langage php";
```

```
$tableau[5]="lagage java";//
```

```
    echo $tableau[5];//affiche: langage java
```

```
    echo $tableau[1];//affiche: vide
```

```
?>
```

Les types de données

- Tableau indexé numériquement (suite)

Il existe une syntaxe réduite pour ajouter les éléments sans manipuler les index.

Exemple:

```
<?php
```

```
    //ces trois codes sont équivalents:
```

```
    $tableau=array(1,2,3);
```

```
    $tableau[0]=1; $tableau[]=1;
```

```
    $tableau[1]=2; $tableau[]=2;
```

```
    $tableau[2]=3; $tableau[]=3;
```

```
    $tableau[4]=5; $tableau[]=5;
```

```
    echo $tableau[4]; //affiche: 5! et pourtant dans la déclaration --> 3  
valeurs
```

```
?>
```


Les types de données

- Tableaux associatifs

- ✓ Les index sont des chaînes de caractères.
- ✓ Très utiles dans le traitement des requêtes SQL.

EXEMPLE (tableau mono-dimension)

```
<?php
```

```
$a=array('couleur'=>'rouge',4,'gout'=>'sucré','forme'=>'rond',  
        'nom'=>'pomme',6,7);
```

```
echo $a[0].'<br>',$a[1].'<br>',$a[2].'<br>';
```

```
// affiche: 4 6 et 7
```

```
echo "une $a[nom] de forme $a[forme]et de couleur  
    $a[couleur]";
```

```
// affiche: une pomme de forme rond et de couleur rouge
```

```
?>
```

Les types de données



- Tableaux associatifs (suite)

Exemple (tableau bidimensions)

```
<?php
```

```
$fruits = array ( "fruits" => array ("a"=>"orange",  
    "b"=>"banane", "c"=>"pomme"),  
    "nombres" => array (1, 2, 3, 4, 5, 6),  
    "trous" => array ("premier", 5 => "second",  
    "troisième"));  
echo $fruits[fruits][a]; //affiche: orange,  
echo $fruits[0][0]; //affiche :rien  
?>
```

Les types de données



Array ([fruits] => Array ([a] => orange [b] => banane
[c] => pomme)

[nombres] => Array ([0] => 1 [1] => 2 [2] => 3 [3] => 4
[4] => 5 [5] => 6)

[trous] => Array ([0] => premier [5] => second
[6] => troisième))

Les types de données

- Tableaux associatifs (suite)

Indexation automatique.

Exemple.

```
<?php
```

```
$array = array( 1, 1, 1, 1, 1, 8=>1, 4=>1, 19, 3=>13,4=>5);
```

```
// la case 4 est déclarée DEUX fois mais seule la dernière est valide
```

```
print_r($array);
```

```
?>
```

On obtient:

```
Array ( [0] => 1 [1] => 1 [2] => 1 [3] => 13 [4] => 5  
[8] => 1 [9] => 19 )
```



Les types de données

- OBJET

Programmation objet.



Les types de données

Les ressources

- ✓ Une ressource ("resource" en anglais), est un type spécial, qui représente une référence sur une ressource externe.

Exemple:

Connexion à une base de données,
fichier, image...

- ✓ Les ressources sont créées par des fonctions dédiées.



Les types de données

- **Transtypage**

Idem qu'en C.

- ✓ Les conversions autorisées sont:
- ✓ (int), (integer) -type entier
- ✓ (bool), (boolean) - booléen
- ✓ (real), (double), (float) - type double
- ✓ (string) - type chaîne
- ✓ (array) - type tableau
- ✓ (object) - type objet

Exemple:

```
<?php
    $var= 10; // $var est un entier
    $bar = (double) $var; // $var est un double
?>
```



Les types de données

o Exercice:

Soit "Drapeau" un tableau contenant les trois couleurs: Blanc, Bleu et Rouge.

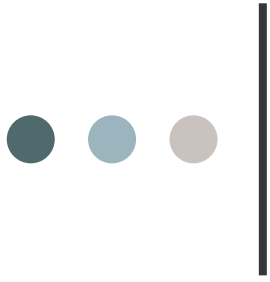
Écrire un script PHP qui affiche les couleurs du drapeau Français.



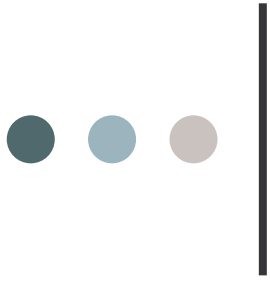
Les types de données

- Solution possible:

```
< ?php
    // « Ceci est un tableau »
    $drapeau=array('Bleu','Blanc','Rouge');
    echo "<HR>";
    echo "Affichage :<br>";
    echo
    $drapeau[0].$drapeau[1].$drapeau[2];
    echo "<hr>";
    ?>
```



LES STRUCTURES DE CONTRÔLE



LES CONDITIONS



LES STRUCTURES DE CONTRÔLE

- LE TEST SIMPLE (le SI)

Syntaxe:

```
<?  
If (expression)  
    {Instructions;}
```

```
?>
```

Exemple:

```
<?  
$a=6;  
$b=7;  
if ($a > $b)  
    {print ("$a > $b");}
```

```
?>
```



LES STRUCTURES DE CONTRÔLE

- L'ALTERNATIVE (si ...alors...Sinon ...)

Syntaxe:

```
Si (condition) {instruction.1;}  
                Else {instruction.2;}
```

Exemple :

```
<?  
$a=5;  
$b=6;  
if ($a > $b) {echo "$a > $b";}  
    else  
{echo "$a <= $b";}  
?>
```



LES STRUCTURES DE CONTRÔLE

- ALTERNATIVE (suite)

Syntaxe

Si (condition) {instruction.1;}

Elseif (condition.2) {instructioun.2 ;}

Else {{instruction.3 ;}

Exemple:

<?

\$a=7;

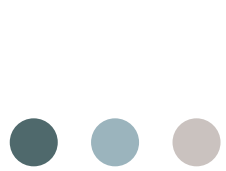
\$b=8;

if (\$a>\$b) {echo "\$a > \$b";}

 elseif (\$a=\$b) {echo "\$a=\$b";}

 else {echo "\$a < \$b";}

?>



LES STRUCTURES DE CONTRÔLE

- L'opérateur ?

L'opérateur ? peut remplacer

If --- else

\$resultat= condition ? expression1:expresssion2

Exemple :

```
<?
```

```
$a=19;
```

```
$b=10;
```

```
$resultat=($a==$b)? "Egaux" : "Différents";
```

```
echo $resultat;
```

```
?>
```



LES STRUCTURES DE CONTRÔLE

- Le selon

Généralisation de si.

Switch (\$val)

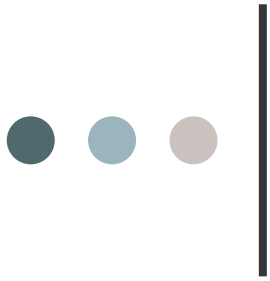
```
{  
    case val1 : commande1;  
        break;  
    case val2 : commande2;  
        break;  
    case valn : commanden;  
        break;  
    default :  
        commande.par.default;  
        break;  
}
```




LES STRUCTURES DE CONTRÔLES

Exemple:

```
<?
$telephone="0862828282";
$region=$telephone[1];
switch ($region)
  {case "1" :
    echo "Paris";
    break;
  case "2" :
    echo "Ouest";
    break;
  case "3" :
    echo "Nord";
    break;
  case "4" :
    echo "Sud-Est";
    break;
  case "5" :
    echo "Sud-Ouest";
    break;
  case "6" :
    echo "Portable";
    break;
  default :
    echo "Il y a une erreur au numéro !";
    break;
  }
?>
```



LES BOUCLES



LES BOUCLES

- o Tant que --- faire (While)

idem que dans C

Syntaxe:

```
While (expression)
    {commandes;}
```

Exemple :

```
<?
$i=0;
while ($i <= 10) {print ($i++); print("\t");}
```

?>



LES BOUC LES

- Tant Que (suite)

Variante de Tant que (while...endwhile).

Exemple:

```
<?  
$i=0;  
while ($i >=10):  
print $i;  
$i++;  
endwhile  
>?
```

LES BOUCLES



- o Exemple de génération dynamique d'une table :

On utilise :

La fonction `key()` qui retourne la clé de l'élément en cours.

La fonction `next()` qui positionne le pointeur interne du tableau sur l'élément suivant.

<?

```
Echo "<center><b> Création dynamique de tableau </b></center>";
```

```
$tableau=array("un" => 'PHP', 'Deux' => 'MySQL', 'Trois' => 'Javascript', 'Quatre' => 'Perl');
```

```
echo "<table width=400 border=\"2\">";
```

```
while ($ligne=key($tableau))
```

```
{echo "<tr>";
```

```
echo "<td width=100> ligne $ligne </td>";
```

```
echo "<td> livre $ligne=".$tableau[$ligne]. "</td>";
```

```
echo "</tr>";
```

```
next($tableau);}
```

```
echo "</table>";
```

```
?>
```

LES BOUCLES



- Faire ... tant que (do ... while)

Le corps de la boucle est exécuté au moins une fois.

Exemple :

```
<?  
$i=10;  
do  
    { print $i;  
      $i--;}  
while ($i > 0);  
>
```



LES BOUCLES

- o La boucle Pour (for)

Syntaxe:

```
For (exp1; exp2; exp3)
    {instructions;}
```

Exemple :

```
<?
```

```
For ($i=0; $i<10;$i++)
    {print $i;}
```

```
?>
```

```
<?
```

```
For ($i; $i<10; print $i, $i++);
```

```
?>
```



LES BOUCLES

- **Pour chaque (ForEach)**

Moyen simple pour passer en revue un tableau.

Deux syntaxes :

- `ForEach (array.expression as $value) {commande;}`
- `ForEach (array.expression as $key => $value) {commande;}`



LES BOUCLES

- **Pour chaque(suite)**

Exemple :

```
<?
```

```
Echo "<center><b> Lecture d'un tableau </b></center>";
```

```
    $matiere=array("Math", "Informatique", "Gestion", "Comptabilité");
```

```
    $indice=0;
```

```
    foreach($matiere as $valeur)
```

```
    {echo "indice=', $indice, "valeur=", $valeur, " <br>";
```

```
    $indice++;}
```

```
?>
```



LES BOUCLES

○ Break et continu

- ✓ Break : permet de stopper une boucle, si une condition est vérifiée.
- ✓ Continue : permet aussi de stopper une boucle mais uniquement pour la valeur du compteur en cours.

LES BOUCLES

- Continue(suite)

Continue accepte un argument numérique optionnel qui vous indiquera combien de structures emboîtées ont été ignorées.

Exemple:

```
<?php
    $i = 0;
    while ($i++ < 5) {
        echo "Dehors<br>\n";
        while (1) {
            echo "&nbsp;&nbsp;&nbsp;Milieu<br>\n";
            while (1) {
                echo "&nbsp;&nbsp;&nbsp;Intérieur<br>\n";
                continue 3 ;// SAUT VERS LA BOUCLE DE DEPART
            }
            echo "Ceci n'est jamais atteint.<br>\n";
        }
        echo "Ceci non plus.<br>\n";
    }
?>
```

● LES BOUCLES

- L'instruction break.

- ✓ permet de sortir d'une structure for, while, foreach ou switch.
- ✓ break accepte un argument numérique optionnel qui vous indiquera combien de structures emboîtées ont été interrompues.



LES BOUCLES

- L'instruction continue.

est utilisée dans une boucle afin d'éviter les instructions de l'itération courante et de passer directement à l'itération suivante.

Exemple:

```
<?php
    for ($i = 0; $i < 5; ++$i) {
        if ($i == 2)
            continue
        print "$i\n";
    }
?>
```

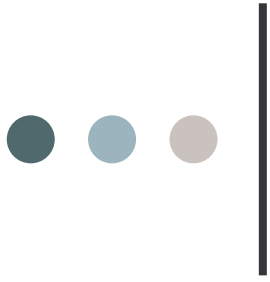


LES BOUCLES

- Break (suite)

Exemple:

```
/* Utilisation de l'argument optionnel. */
$i = 0;
while (++$i) {
  switch ($i) {
    case 5:
      echo "A 5<br>\n";
      break 1; /* Termine uniquement le switch. */
    case 10:
      echo "At 10; quitting<br>\n";
      break 2; /* Termine le switch et la boucle while. */
    default:
      break;
  }
}
?>
```



LES SOUS -PROGRAMMES



LES FONCTIONS

- **Déclaration et appel.**

Syntaxe:

```
<?php
// déclaration
Function nom_de_fonction($arg1, $arg2, ..., $argn)
{
instructions;
}
//Appel de la fonction
nom_de_fonction($arg1, $arg2, ..., $argn);// procedure
//ou
$var= nom_de_fonction($arg1, $arg2, ..., $argn);/*fonction retournant
une valeur*/
?>
```

Remarque: la liste des arguments peut être vide.



LES FONCTIONS

- Valeur par défaut

Il est possible d'assigner une valeur par défaut aux différents arguments dans l'entête de la fonction.

Syntaxe:

```
<?php
```

```
Fonction nom_de_la_fonction($arg1='val1',$arg2='val2,...)
```

```
{
```

```
Instructions;
```

```
}
```



LES FONCTIONS

- Valeur de retour

Grâce au mot clé ***return***, une fonction peut retourner une valeur au programme appelant.

- ✓ Une fonction peut contenir plusieurs instructions de retour (return), mais l'exécution de la fonction s'arrêtera au premier retour rencontré.
- ✓ Une fonction qui ne retourne aucune valeur, ou plusieurs valeurs est une procédure.



LES FONCTIONS

- Fonctions (suite).

Exemple de fonction retournant une seule valeur.

```
<?
```

```
Function somme ($a=2, $b=3);//déclaration
```

```
{ $s=0;
```

```
$s=$a+$b;
```

```
return $s;}
```

```
echo somme(5,7);//appel de la fonction
```

```
?>
```



LES FONCTIONS

- Fonctions (suite).

Exemple de fonction retournant ou non de valeurs.

<?

Function somme (\$a, \$b); //déclaration

```
{ $s=0;
```

```
$s=$a+$b;
```

```
echo " $a + $b = $s; }
```

```
somme(5,7); //appel de la fonction
```

?>



LES FONCTIONS

- Passage d'argument

- ✓ Par valeur : la fonction utilise une copie de la valeur. Toute modification à l'intérieur de la fonction n'aura pas d'effet à l'extérieur.

Appel: `nom_fonction($arg1, $arg2, ..., argn);`

- ✓ Par référence: la fonction utilise l'adresse de la variable. Toute modification à l'intérieur de la fonction n'aura pas d'effet à l'extérieur.

Appel: `nom_fonction(&$arg1, &$arg2, &$argn);`



LES FONCTIONS

- Exemple: appel par valeur.

```
<?php
function somme ($prix1=2,$prix2=3)
{
    $total=$prix1+$prix2;
    return $total;
}
print somme(); //affiche :5
print somme(7,8); //affiche :15
print somme(10); //affiche :13
print soome(,10); // affiche une erreur
?>
```



LES FONCTIONS

- Exemple: appel par valeur.

```
<?php
function incrementation($nombre=7)
{
    $nombre++;
    print $nombre;
}
$val=8;
incrementation($val); // affiche: 9
print('<br>');
print $val ;//affiche 8
?>
```



LES FONCTIONS

- Exemple : appel par référence (procédure).

```
<?php
```

```
function calcul ($prix1=2,$prix2=3,&$totalht=0,&$totalttc)
```

```
{    $prix1*=2; $prix2*=2;
```

```
    print ("$prix1,$prix2,<br>");
```

```
    $totalht=$prix1+$prix2;
```

```
    $totalttc= $totalht*1.9;
```

```
}
```

```
$P1=5;      $P2=8; $HT=0;      $TTC=0;
```

```
print ("$P1,$P2,$HT,$TTC,<br>"); //affiche : 5 8 0 0
```

```
calcul($P1,$P2,&$HT,&$TTC); //affiche: 10 16
```

```
print ("$P1,$P2,$HT,$TTC"); //affiche 5 8 26 49.4
```

```
?>
```




LES FONCTIONS

- Portée des variables

Une variable, selon l'endroit où elle est définie, a une portée plus au moins grande.

- ✓ Le niveau *local*: Utilisé par défaut.
- ✓ Le niveau *global*: Dans ce cas, la variable est visible à l'intérieur et à l'extérieur de la fonction.
- ✓ Le niveau *static*: *variable locale à la fonction, sa valeur reste persistante pendant tout le script et conserve sa valeur entre différents appel de la fonction.*

LES FONCTIONS

- - Portée (suite).

Exemple: Variable locale.

```
<?php
$viande='Coq';
function menu_midi()
{
    print "la viande ce midi est le: $viande ou";
    $viande='canard sauvage';
    print $viande;
    print"\n";
}
function menu_soir()
{
    print "la viande ce soir est le: $viande ou";
    $viande="poisson";
    print $viande;
    print"\n";
}
menu_midi();// affiche: la viande ce midi est le: ou canard sauvage
menu_soir();//affiche:la viande ce midi est le: ou poisson
print "La viande du chef est: $viande"; //affiche: La viande du chef est: Coq
?>
```

LES FONCTIONS

Portée (suite).

Exemple: variable globale

```
<?php
$viande='Coq';
function menu_midi()
{ global $viande;
  print "la viande ce midi est le: $viande ou";
  $viande='canard sauvage';
  print $viande;
  print"\n";
}
function menu_soir()
{global $viande;
  print "la viande ce soir est le: $viande ou";
  $viande="poisson";
  print $viande;
  print"\n";
}
menu_midi();// affiche: la viande ce midi est le: Coq ou canard sauvage
menu_soir();//affiche:la viande ce midi est le: canard sauvage ou poisson
print "La viande du chef est: $viande"; //affiche: La viande du chef est: poisson
```

LES FONCTIONS

- **Portée (suite).**

- Exemple: variable static.**

```
<?php
```

- `$credit = 'Total crédits obtenus:';`
`function ajout_credit($affiche_total=)`

```
{  
    global $credit;  
    static $total=0;  
    $total++;  
    if ( $affiche_total=='affiche')  
    {  
        echo $credit.$total;  
    }  
}
```

- `ajout_credit(); // $total =`

- `ajout_credit(); //$total =2`

- `ajout_credit(); //$total=3`

- `ajout_credit("affiche");// affiche Total Crédits obtenus:4`

- `?>`

● ● ●
○ **Portée (suite)**

Exemple:

```
<?php
    $a = 1;
    $b = 2;
    function somme() {
        global $a, $b;
        $b = $a + $b;
    }
    somme();
    echo $b; // affiche la valeur 3
?>
```

LES FONCTIONS

- **Portée (suite).**

Une deuxième méthode pour accéder aux variables globales est d'utiliser le tableau associatif prédéfini `$GLOBALS`.

Exemple:

```
<?php
    $a = 1;
    $b = 2;
    function somme() {
        $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
    }
    somme();
    echo $b ; // affiche la valeur 3
?>
```

LES FONCTIONS

Fonction retournant plusieurs valeurs.

La méthode la plus simple consiste à stocker les valeurs à retourner, dans un tableau.

Exemple:

```
<?php
function calcul_prix($pu,$ttva,$treduction)
{
    $tva=$pu*$ttva/100;
    $ttc=$pu+$tva;
    $reduction=$ttc*$treduction/100;
    return array($pu,$tva,$ttc,$reduction);
}
$facture=calcul_prix(100,19.6,5);
for($i=0;$i<4;$i++)
{
    print $facture[$i];
    print '<br>';
}
?>
```

LES FONCTIONS

- ○ Inclusion de fichiers (**require()** & **include()**).
 - ✓ La fonction **include()** inclut et exécute le fichier spécifié en argument
 - ✓ La commande **require()** se remplace elle-même par le contenu du fichier spécifié.
 - ✓ **require()** et **include()** sont identiques, sauf dans leur façon de gérer les erreurs.
 - ✓ **include()** produit une Alerte (warning) tandis que **require()** génère une erreur fatale.
- Avec **require()** si le fichier d'inclusion est manquant le script est interrompu. Par contre avec **include()** le script continuera son exécution.

LES FONCTIONS

- ● ● Exemple

```
vars.php  
<?php
```

```
$couleur = 'verte';  
$fruit = 'pomme';
```

```
?>
```

```
test.php  
<?php
```

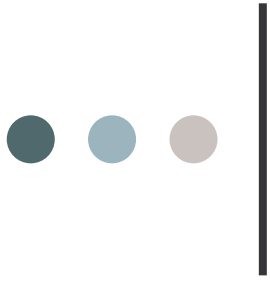
```
include 'vars.php';
```

```
echo "Une $couleur $fruit"; // Une verte pomme
```

```
?>
```

LES FONCTIONS

- ○ `include once()`, `require once()`
- La commande **include once()** inclut et évalue le fichier spécifié durant l'exécution du script. Le comportement est similaire à **include()**, mais la différence est que si le code a déjà été inclus, il ne le sera pas une seconde fois.
- La commande **require once()** se remplace elle-même par le fichier spécifié, un peu comme les commandes de préprocesseur C `#include`.



Les fichiers

principes d'accès aux fichiers

Le système d'utilisation des fichiers repose sur la création de ressources système.

- ✓ création de la ressource.
L'accès: lecture, écriture
- ✓ La destruction



principes d'accès aux fichiers

- DROIT D' ACCES

'r' Lecture seule

'r+' Lecture, écriture pointeur en début.

'w' Ouverture en écriture seule et le pointeur est placé en début.

'w+' Lecture, écriture, pointeur en début mais tronquage (remplacement) du fichier existant.



principes d'accès aux fichiers

- o DROIT D' ACCES (suite).

'a' Ouverture en Ecriture seule et place le pointeur en fin de fichier. Si le fichier n'existe pas une tentative de création aura lieu

'a+' Lecture, écriture et place le pointeur de fichier en fin de fichier. Si le fichier n'existe pas PHP crée le fichier.

fopen accepte les URL – Permet d'ouvrir et de lire un fichier. Si la fonction réussit, elle retourne un entier (VRAI), sinon retourne

110 FALSE.

○ Exemple :

```
<?
//ouverture du fichier en lecture
$fichier="donnees.txt";
if ( !($monfichier=fopen($fichier,'r')))
{
print("erreur !");
    print ("Ce fichier ne peut pas être accédé ! \n");
    exit;
}
while (!feof($monfichier))
{
    $ligne=fgets($monfichier,260);
    print("$ligne <br> \n");
}
fclose($monfichier);
?>
```

FONCTIONS D' ACCES AUX FICHIERS

- Fopen (\$nom.de.fichier, \$ mode);
- Fwrite (\$descripteur.de.fichier, \$chaine, [\$taille]);
- Fputs (\$descripteur.de.fichier, \$chaine);
- Fgets (\$descripteur.de.fichier, \$taille);
- Ftell (\$descripteur.de.fichier);
- Fseek (\$descripteur.de.fichier, offset) → se positionne sur un caractère
- Si ofset = 50 alors position sur le 50èmer caractère.
- Fclose (\$descripteur.de.fichier);
- Autres fonctions
- -basename : sépare le nom du fichier et le nom du répertoire.

Exemple :

<?

```
$chemin="c:\toto\essai\index.php";  
$fichier=basename($chemin);  
print ("Le nom du fichier est : $fichier");
```

?>

- Copy : Copie de fichier

<?

```
$fichier="donnees.txt";  
if (!copy($fichier,$fichier.'bis'))  
{  
    print ("La copie du fichier : $fichier n'a pas réussie...  
<br> \n");
```

- delete : efface un fichier

- - dirname : le nom du dossier

Exemple :

```
<?
```

```
$chemin="/www/phpindex.php";
```

```
$fichier=dirname($chemin);
```

```
print("Le répertoire est : $fichier");
```

```
?>
```

- diskfree : renvoie l'espace disque disponible dans le répertoire.

Exemple :

<?

```
$df=du -skfree ("c:\toto\");  
echo $df." Nombres d'octets libres sur c:\toto";
```

?>

- fflush : vide le buffer
- file_exists : Pour savoir si le fichier existe
- file_type : Pour connaître le type de fichier
- dir : créer un répertoire

Exemple :

<?

```
Print ("<ul> \n");  
$rep=dir(".");  
while ($nom=$rep->read())  
{  
    print("<li>$nom \n");  
}  
$rep->close();  
print ("<ul> \n");
```

?>