PROGRAMMATION EN PHP

PHP : C'est QUOI ?

- PHP : ***P**ersonal **H**ome **Page***
- Un langage de scripts permettant la création d'applications Web
- exécuté côté serveur et non côté client
- Indépendant de la plate-forme utilisée
- Ses principaux atouts sont:
  - La possibilité d'inclure le script PHP au sein d'une page HTML
  - La simplicité d'écriture de scripts
  - La simplicité d'interfaçage avec des bases de données

Un peu d'histoire

- **Historique :**
  - juin 1998 : La version3.0 de PHP
  - fin de l'année 1999: la version PHP4 est apparue
  - Juillet 2004 : La version actuelle PHP5 est annoncée
- **Popularité :**
  - 2002 : PHP est utilisé par plus de 8 millions de sites Web à travers le monde
  - 2007 : plusque20 millions
  - 2013 : par plus de 244 millions
  - sites web célèbres créés par PHP : Facebook, YouTube, Wikipedia, …

Fonctionnement de PHP

= réponse HTTP

Réseau

Client

GET /hello.php HTTP/1.0

teur
- HTML
- JavaScript
- CSS

Protocole HTTP

Exécution d'un programme sur le serveur

Serv

Ser

Module PHP

MySQL

tml
hp
.jpg

```
<html>
<head>
<title>Hello</title>
</head>
<body>Hello world</body>
</html>
```

Hello world

hello.php

```
<html>
 <head>
  <title>Hello</title>
 </head>
 <body>
<?php
Echo "Hello world"
>
</body>
</html>
```

5

syntaxe

Intégration d'un script dans une page

Le code source php est directement insérer dans le fichier html :

<?php … ?>

Exemple:

```
<html>
<body>
<?php
        echo ''Bonjour'';
?>
</body>
</html>
```

Intégration PHP et HTML (4)

- Envoi du code HTML par PHP
  - La fonction echo : echo Expression;
    - echo "Chaine de caracteres";
    - echo (1+2)*87;
  - La fonction print : print(expression);
    - print("Chaine de caracteres");
    - print ((1+2)*87);
  - La fonction printf : printf (chaîne formatée);
    - printf ("Le périmètre du cercle est %d",$Perimetre);

Exemple de script

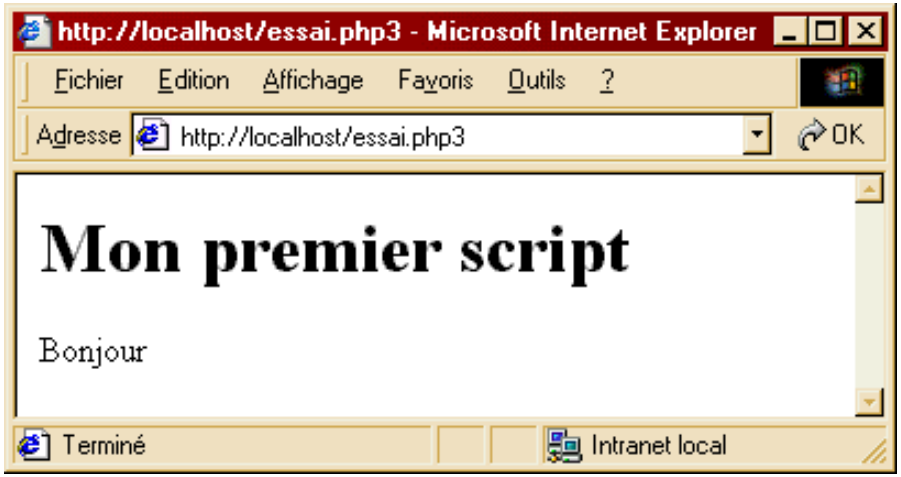
Exemple de script, code source (côté serveur) :

```
<html>
<body>
<h1>Mon premier script</h1>
<?php echo ''Bonjour\n''; ?>
</body>
</html>
```

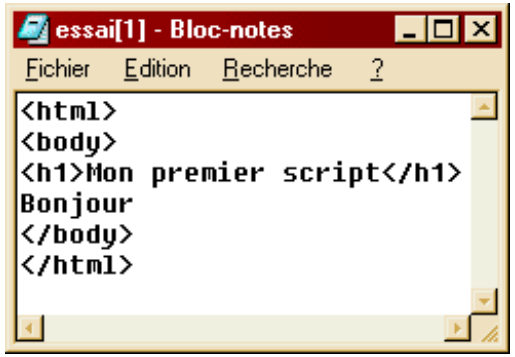
Autre écriture du même script :

```
<?php
echo ''<html>\n<body>\n'';
echo ''<h1>Mon premier script</h1>\n'';
echo ''Bonjour\n'';
echo ''</body>\n</html>\n'';
?>
```

Résultat affiché par le navigateur :



Code source (côté client) de la page essai.ph3 résultant du script



Eléments de syntaxe PHP

- Chaque instruction se termine par " ; "
- Tabulation :
  - \car échappe un caractère spécifique.
  - \n nouvelle ligne
- Commentaires:

```
/* jusqu'au prochain */
// jusqu'à lafin de la ligne
# jusqu'à la fin de la ligne
```

Syntaxe de base : Les constantes

- Les constantes
  - **Define**("nom_constante", valeur_constante)
    - define (“ma_const”, “Vive PHP4”) ;
    - define (“an”, 2002) ;
  - Les constantes prédéfinies
    - NULL
    - _FILE_
    - _LINE_
    - PHP_VERSION
    - PHP_OS
    - TRUE et FALSE
    - E_ERROR

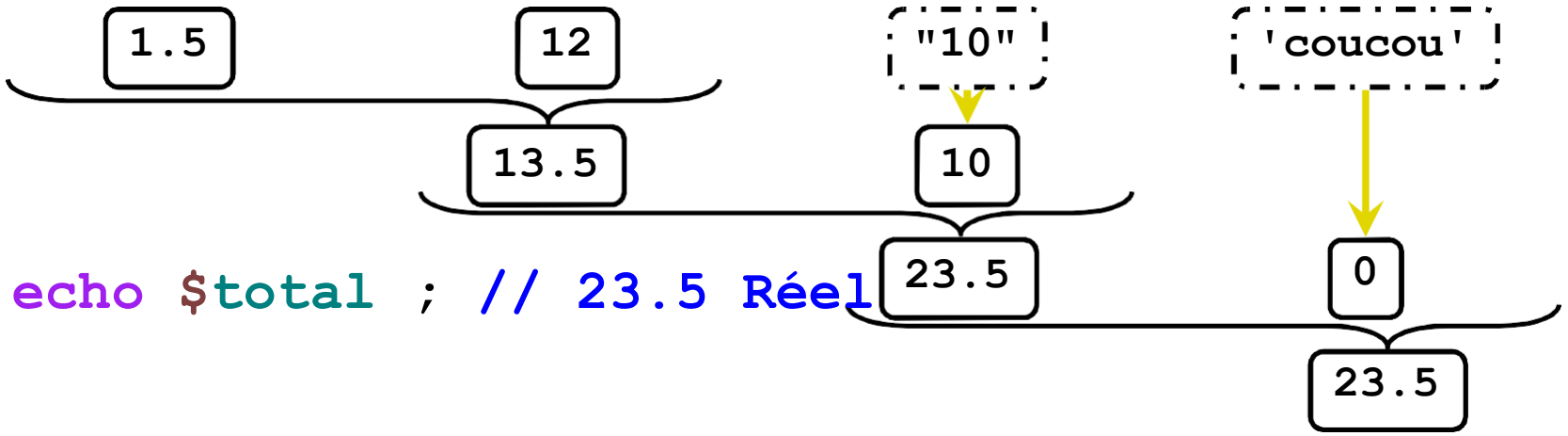
Syntaxe de base : Les variables (1)

- Principe
  - Commencent par le caractère $
  - N'ont pas besoin d'être déclarées
- Les types :
  - Numérique entier: **12** ou réel: **1.54**
  - Chaîne: "Hello" ou 'Bonjour'
  - Booléen: **true, false**
- Le type d'une variable : déterminé par la valeur qui lui est affectée
- Affectation par valeur et par référence
  - Affectation par valeur : $b=$a
  - Affectation par (référence) variable : $c = &$a

Exemple :


```
$nombre1 = 1.5 ;        // Réel
$nombre2 = 12 ;         // Entier
$chaine1 = "10" ;       // Chaîne
$chaine2 = 'coucou' ;   // Chaîne

$total =
 $nombre1 + $nombre2 + $chaine1 + $chaine2 ;
```




```
echo $total ; // 23.5 Réel
```

Les variables et les types de données

- Fonctions de vérifications de variables
  - Test sur le type de la valeur :
    is_array(), is_bool(), is_double()
    is_float(), is_int(), is_long()
    is_object(), is_numeric(), is_string()
  - Test sur le contenu de la valeur :
    empty(), gettype(), Isset(), settype()

Les opérateurs arithmétiques

`$a + $b`	Somme
`$a - $b`	Différence
`$a * $b`	Multiplication
`$a / $b`	Division
`$a % $b`	Modulo (Reste de la division entière)

Les opérateurs de comparaison

`$a == $b`	Vrai si égalité entre les valeurs de `$a` et `$b`
`$a != $b`	Vrai si différence entre les valeurs de `$a` et `$b`
`$a < $b`	Vrai si `$a` inférieur à `$b`
`$a > $b`	Vrai si `$a` supérieur à `$b`
`$a <= $b`	Vrai si `$a` inférieur ou égal à `$b`
`$a >= $b`	Vrai si `$a` supérieur ou égal à `$b`

Les opérateurs booléens

Opérateur	Dénomination	Effet
|| ou OR	OU logique	Vérifie qu'une des conditions est réalisée
&& ou AND	ET logique	Vérifie que toutes les conditions sont réalisées
XOR	OU exclusif	Opposé du OU logique
!	NON logique	Inverse l'état d'une variable booléenne (retourne la valeur 1 si la variable vaut 0, 0 si elle vaut 1)

Opérations sur les chaînes

- Permet d'assembler plusieurs chaînes
- Réalisé grâce à l'opérateur point : .

```
"Bonjour " . "Marcel"
```

→ vaut **"Bonjour Marcel"**

```
$nb = 6*2 ;
"Acheter " . $nb . " oeufs"
```

→ vaut **"Acheter 12 oeufs "**

- $longueur = strlen($phrase) ;
- $ma_variable = str_replace(' b' , ' p' , ' bim bam boum') ;
- $chaine = strtolower($chaine) ;
- De même pour strtoupper

Les chaînes de caractères

Substitution de variables dans les chaînes

- Guillemets simples
  - `$a='chaîne';`
  - `$b='voiciune $a';`

  chaîne

  voici une $a
- Guillemets doubles
  - `$a="chaîne";`
  - `$b="voiciune $a";`

  chaîne

  voici une chaîne

19

Les tableaux

Les tableaux

- Création / initialisation:

```
$tab1=array(12, "fraise", 2.5) ;
```

```
$tab2[] = 12 ;
$tab2[] = "fraise" ;
$tab2[] = 2.5 ;
```

Clé	Valeur
0	12
1	"fraise"
2	2.5

```
$tab3[0] = 12 ;
$tab3[1] = "fraise" ;
$tab3[2] = 2.5 ;
```

Les tableaux « à trous »

- Les éléments du tableaux ne sont pas forcement d'indices consécutifs :

```
$tab4[0] = 12 ;
$tab4[1] = "fraise" ;
$tab4[2] = 2.5 ;
$tab4[5] = "el5" ;
```

Clé	Valeur
0	12
1	"fraise"
2	2.5
3	
4	
5	"el5"

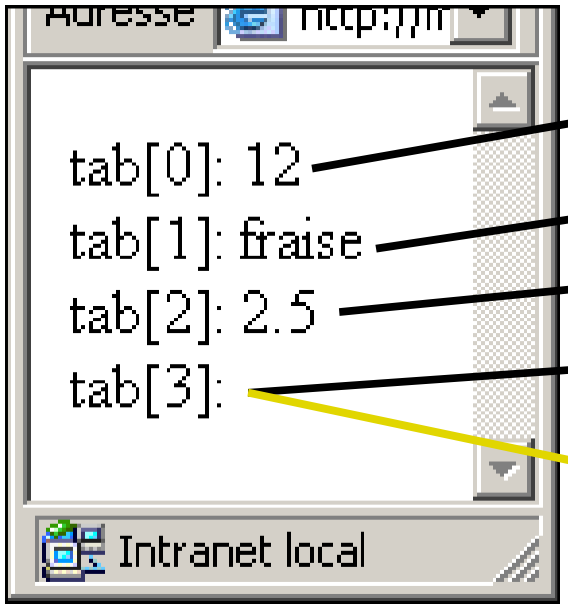
- Comment parcourir de tels tableaux ?

Les tableaux « à trous » (suite)

4

Parcours classique :

```
for ($i=0; $i < sizeof($tab4); $i++)
{ echo "tab4[$i]: "
        . $tab4[$i] . "<BR>\n";
}
```



Clé	Valeur
0	12
1	"fraise"
2	2.5
3	
4	
5	"el5"

Structure de contrôle Pour chaque…


```
foreach ($tableau as $element)
{
  /* Bloc d'instructions répété pour
  chaque élément de $tableau */
  /* Chaque élément de $tableau est
  accessible grâce à $element */
}
```

Parcours de tableau : foreach

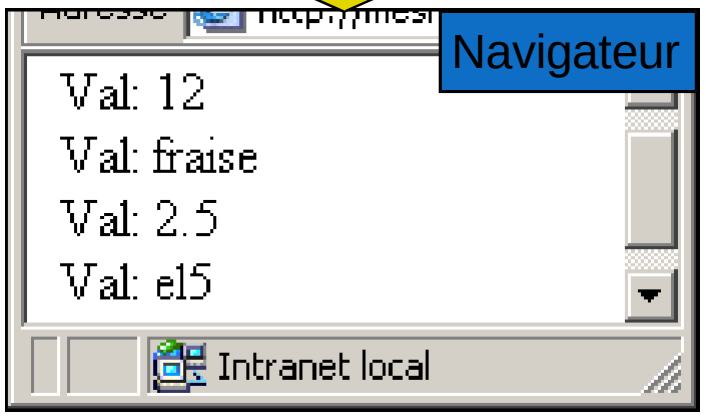
PHP

```
…
$tab4[0] = 12 ;
$tab4[1] = "fraise" ;
$tab4[2] = 2.5 ;
$tab4[5] = "el5" ;
foreach($tab4 as $v)
{
   echo "Val: $v<br>\n";
}
…
```

HTML

```
…
Val:12<br>\n
Val:fraise<br>\n
Val:2.5<br>\n
Val:el5<br>\n
…
```

Navigateur



Tableaux associatifs

- Tableaux dont l'accès aux éléments n'est plus réalisé grâce à un index (0,1,…) mais grâce à une clé de type entier ou chaîne.
- Exemples de clés:

```
$tab['un']          = 12 ;
$tab[205]           = "bonjour" ;
$tab["la valeur"]   = 3.0 ;
```

- Création

```
$tab = array(cle1 => val1,
             cle2 => val2,
             …);
```

Tableaux associatifs - Exemples


```php
$tab5['un']     = 12 ;
$tab5['trois'] = "fraise" ;
$tab5["deux"]   = 2.5 ;
$tab5[42]       = "el5" ;
```

Clé	Valeur
"un"	12
"trois"	"fraise"
"deux"	2.5
42	"el5"

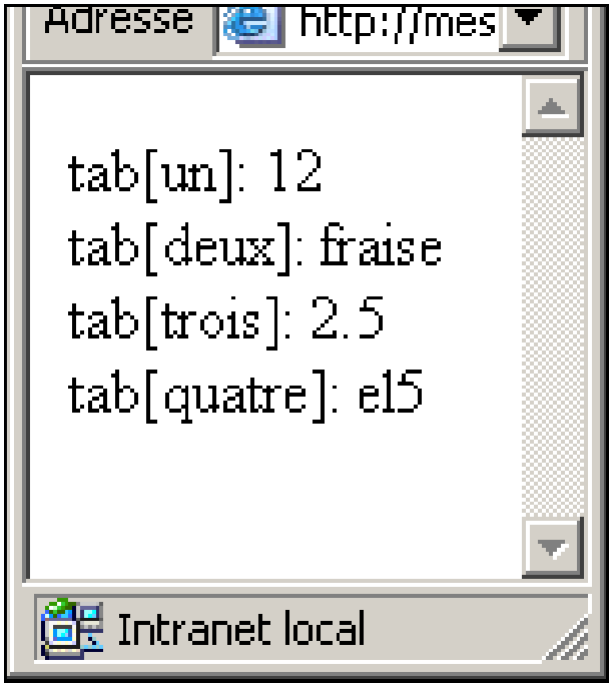
```php
$tab6 = array('un'     => 12,
              'trois' => "fraise",
              "deux"   => 2.5,
              42       => "el5") ;
```

Structure de contrôle Pour chaque…


```
foreach($tableau as $cle => $element)
{
  /* Bloc d'instructions répété pour
     chaque élément de $tableau */
  /* Chaque élément de $tableau est
     accessible grâce à $element */
  /* La clé d'accès à chaque élément est
     donnée par $cle */
}
```

Parcours de tableau


```php
<?php
$html = <<<HTML
<html>
  <head><title>foreach clé</title>
  </head>
<body>
HTML;
$tab6 = array('un'      => 12,
              'deux'    => "fraise",
              "trois"   => 2.5,
              "quatre"  => "el5") ;
foreach ($tab6 as $cle => $val)
{
  $html .= "tab[$cle]: $val<br>\n" ;
}
echo $html . "</body>\n</html>" ;
```



Fonctions sur les tableaux

- array_key_exists : pour vérifier si une clé existe dans l'array ;
  - array_key_exists('cle' , $array) ;
- in_array : pour vérifier si une valeur existe dans l'array ;
  - in_array(' Myrtille' , $fruits)
- array_search : pour récupérer la clé d'une valeur dans l'array
  - $position = array_search(' Fraise' , $fruits) ;
- count($tab), sizeof : retournent le nombre d'éléments du tableau
- sort($tab) : trie alphanumérique les éléments du tableau
- rsort($tab) : trie alphanumérique inverse les éléments du tableau
- array_merge($tab1,$tab2,$tab3…) : concatène les tableaux passés en arguments
- array_rand($tab) : retourne un élément du tableau au hasard

30

Les fonctions

Fonctions utilisateur

- Description d'une fonctionnalité dépendant éventuellement de paramètres et retournant éventuellement un résultat
- Définition

```
function moyenne($a,$b)
{
    return ($a+$b)/2. ;
}
```

- Utilisation

```
$resultat = moyenne(2,4) ;
echo $resultat ; // vaut 3
```

Déclaration fonction

- **Sans paramètre de retour:**
  - **function** DireBonjour($nom)

```
{
echo ' Bonjour ' . $nom . ' ! <br />' ;
}
```

- **Avec paramètre de retour:**
  - **function** VolumeCone($rayon, $hauteur)

```
{
$v=$rayon * $rayon * 3.14 * $hauteur *(1/3);
// calcul du volume
return $volume;
}
```

Mode de passage des arguments
(types natifs)

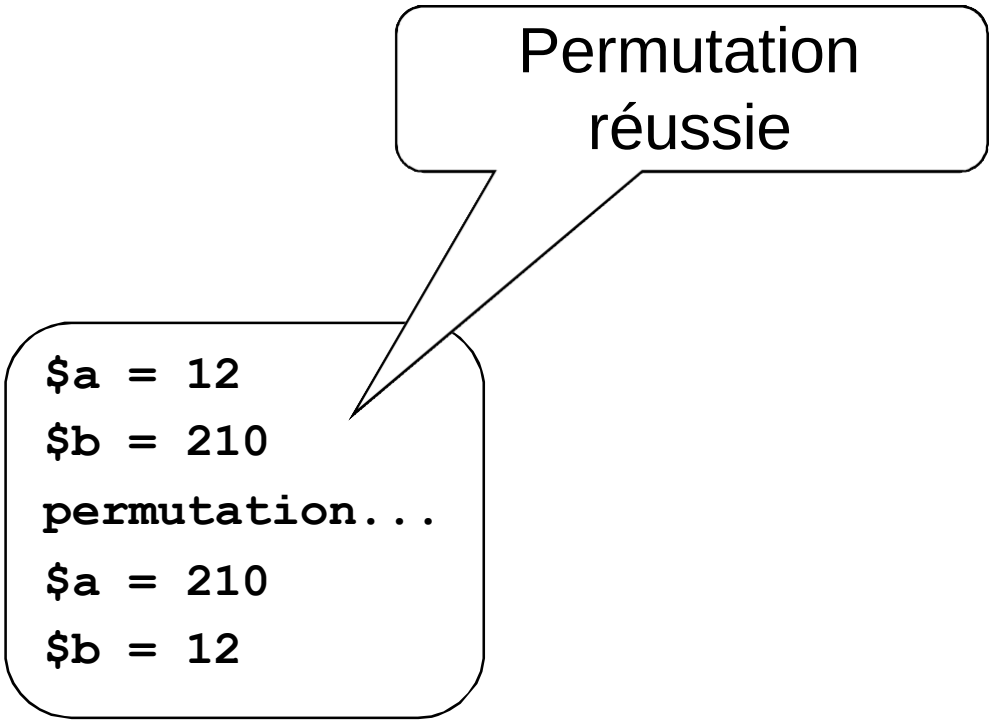

```php
<?php
function permutation($x, $y) {
   echo "permutation..." ;
   $t = $x ;
   $x = $y ;
   $y = $t ;
}
$a = 12 ;
$b = 210 ;
echo "\$a = $a" ;
echo "\$b = $b" ;
permutation($a, $b) ;
echo "\$a = $a" ;
echo "\$b = $b" ;
?>
```

Permutation impossible :
Passage des arguments
des fonctions par valeur

```
$a = 12
$b = 210
permutation...
$a = 12
$b = 210
```

Mode de passage des arguments
(types natifs)


```php
<?php
function permutation(&$x, &$y) {
   echo "permutation..." ;
   $t = $x ;
   $x = $y ;
   $y = $t ;
}
$a = 12 ;
$b = 210 ;
echo "\$a = $a" ;
echo "\$b = $b" ;
permutation($a, $b) ;
echo "\$a = $a" ;
echo "\$b = $b" ;
?>
```



Arguments par défaut des fonctions

- Valeur par défaut d'un argument s'il n'a pas été défini lors de l'appel de la fonction

```
function bonjour($nom="inconnu")
{ echo "Bonjour cher $nom" ; }
```

- Utilisation

```
bonjour() ;
```

Bonjour cher inconnu

```
bonjour("Marcel") ;
```

Bonjour cher Marcel

Transmission de données

Envoyer des paramètres dans l'URL

- Écrire les paramètres dans l'url
- Syntaxe :
  - page.php?param1=valeur1&param2=valeur2&param3=valeur3...
- Exemple :
  - `<a href="bonjour.php?nom=Dupont& prenom=Jean" ></a>`
- **Récupérer les paramètres dans la page destinataire à travers le tableau $_GET**
- **Exemple :**
  - `echo $_GET[ ' prenom' ] ;`

propriétés

- Test d'existance du paramètre :
  - isset($_GET[' prenom']
- Conversion de type:
  - $_GET['repeter'] =(int) $_GET[`repeter']