

# Programmation Web : PHP

Avec l'aimable autorisation de  
**Jérôme CUTRONA**

# PHP: Langage de script pour le Web

- Qu'est-ce que PHP ?
  - Langage de **script**. Utilisé **coté serveur**
  - Acronyme récursif : **PHP**: **H**ypertext **P**reprocessor
  - Créé en 1994-1995 par Rasmus Lerdorf
  - Extension utilisée sur certains serveurs Web (33%)
  - Langage multi plate-forme (UNIX / Windows...)
  - Open Source
  - Versions actuelles (*source nexen.net*) :
    - PHP4 (52% en octobre 2008)
    - PHP5 (48% en octobre 2008)

# Utilité et utilisation de PHP

- Création de pages HTML « dynamiques », fabriquées à la volée, construite à la demande
- Interface entre un serveur Web et des bases de données
- Création d'applications Web

# Principales fonctionnalités de PHP

- Manipulation de chaînes et tableaux
- Calendrier / dates / heures
- Fonctions mathématiques
- Accès au système de fichiers
- Manipulation d'images
- HTTP / FTP / IMAP
- Bases de données (Oracle, MySQL, ...)
- XML
- ...

# Fonctionnement de PHP



# Fonctionnement de PHP

## Client ↔ Serveur

1. Connexion TCP sur le serveur (port 80)
2. Requête HTTP du client (*mon\_fichier.php*)
3. Localisation de la ressource
4. Exécution du code PHP
5. Envoi du résultat de l'exécution au client  
= réponse HTTP
6. Fermeture de la connexion
7. Rendu graphique des données (HTML, image, ...)

# Programme en PHP

Délimitation du code PHP dans le fichier `.php` :

- `<?php Code PHP ?>`

Fermeture optionnelle

- ~~`<script language="PHP">`~~

Code PHP

~~`</script>`~~

Confusion avec JavaScript  
→ à bannir !!

- ~~`<? Code PHP ?>`~~

`short_open_tag`

la configuration

- ~~`<% Code PHP %>`~~

`asp_tags`

du serveur

bannir !!

# Eléments de syntaxe PHP

- La syntaxe de PHP ressemble à celle de famille "C" (C, C++, Java, ...)
- Chaque instruction se termine par ";"
- Commentaires:
  - `/* jusqu'au prochain */`
  - `// jusqu'à la fin de la ligne`
  - `# jusqu'à la fin de la ligne`



# Les variables et les types de données

- Tout identificateur commence par "\$"
- Les affectations sont réalisées grâce à "="
  - Numérique entier: **12** ou réel: **1.54**
  - Chaîne: "Hello" ou 'Bonjour'
  - Booléen: **true**, **false** (PHP 4)
  - Tableau: **\$tab[2]=12**
  - Objet (PHP4, PHP5)
  - Ressource
  - **NULL**
- Le **type** d'une variable est **dynamique** et est **déterminé par la valeur qui lui est affectée**

# Typage faible. Exemple

```
// Pas de déclaration de variable
```

```
$test = 1.5 ; // Réel
```

```
$test = 12 ; // Entier
```

```
$test = array() ; // Tableau
```

```
$test = "10" ; // Chaîne
```

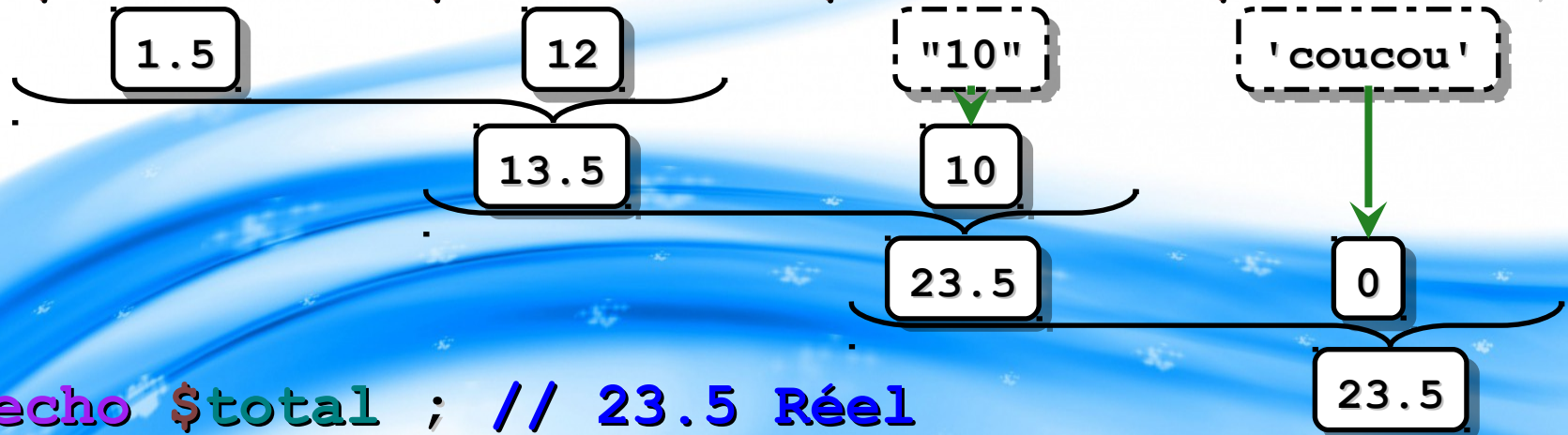
```
echo $test ; // 10
```

# Typage automatique. 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 chaînes de caractères

## Substitution de variables dans les chaînes

- Guillemets simples

- `$a='chaîne' ;`
- `$b='voici une $a' ;`

```
chaîne
```

```
voici une $a
```

- Guillemets doubles

- `$a="chaîne" ;`
- `$b="voici une $a" ;`

```
chaîne
```

```
voici une chaîne
```

- Syntaxe *HereDoc*

- `$a="chaîne" ;`
- `$b=<<  
voici une $a  
sur deux lignes ;-)  
MARQUE_DE_FIN;`

```
chaîne
```

```
voici une chaîne  
sur deux lignes ;-)
```

# Concaténation de 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"
```

# La commande echo

- Permet **d'envoyer du texte au navigateur** du client (« écrire » la page au format HTML résultant de l'interprétation de PHP)
  - `echo "Bonjour" ;`
  - `$nom="Marcel" ; echo "Bonjour $nom" ;`
- Plus généralement, permet **d'envoyer des octets au navigateur** du client
  - Ficher HTML, XML, CSS, JavaScript, ...
  - Données d'une image
  - Contenu d'un fichier PDF, Flash, etc.

# Hello world !

Interprétation du code PHP sur le serveur  
et transmission du résultat au client

```
<?php
$debut = <<<HTML
<html>
  <head>
    <title>hello</title>
  </head>
  <body>\n
HTML;
$corps = "Hello world!\n";
$fin   = <<<HTML
  </body>
</html>
HTML;
/* Envoi au client */
echo $debut.$corps.$fin ;
```

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

**Impossible de voir le code PHP depuis le navigateur !!**

# 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 d'in- et de dé-crémentation pré- et post-fixés

$\$a++$	Retourne la valeur de $\$a$ puis augmente la valeur de $\$a$ de 1
$++\$a$	Augmente la valeur de $\$a$ de 1 puis retourne la nouvelle valeur de $\$a$
$\$a--$	Retourne la valeur de $\$a$ puis diminue la valeur de $\$a$ de 1
$--\$a$	Diminue la valeur de $\$a$ de 1 puis retourne la nouvelle valeur de $\$a$

# 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 $\$b$
$\$a === \$b$	Vrai si $\$a$ et $\$b$ identiques (valeur et type)
$\$a !== \$b$	Vrai si $\$a$ et $\$b$ différents (valeur ou type)

# Les opérateurs logiques

[Expr1] <b>and</b> [Expr2]	Vrai si [Expr1] et [Expr2] sont vraies
[Expr1] <b>&amp;&amp;</b> [Expr2]	idem
[Expr1] <b>OR</b> [Expr2]	Vrai si [Expr1] ou [Expr2] sont vraies
[Expr1] <b>  </b> [Expr2]	idem
[Expr1] <b>XOR</b> [Expr2]	Vrai si [Expr1] ou [Expr2] sont vraies mais pas les deux
<b>!</b> [Expr1]	Vrai si [Expr1] est non vraie

# Les opérateurs sur bits

$\$a \ \& \ \$b$	ET binaire
$\$a \   \ \$b$	OU binaire
$\$a \ \wedge \ \$b$	XOR binaire
$\sim \ \$a$	Inversion bit à bit
$\$a \ \ll \ \$b$	$\$a$ décalé à gauche de $\$b$ rangs
$\$a \ \gg \ \$b$	$\$a$ décalé à droite de $\$b$ rangs

# Précédence des opérateurs

Opérateurs
<b>new</b>
<b>[</b>
<b>++ --</b>
<b>! ~ - (int) (float) (string) (array) (object) @</b>
<b>* / %</b>
<b>+ - .</b>
<b>&lt;&lt; &gt;&gt;</b>
<b>&lt; &lt;= &gt; &gt;=</b>
<b>== != === !==</b>
<b>&amp;</b>

# Précédence des opérateurs

Opérateurs	
^	
&&	
? :	
= += - = -> <<= >>=	
and	
xor	
or	

En cas de doute,  
utilisez les parenthèses ;-)

# Structure de contrôle Si...Alors...Sinon...

```
if (condition)
{
    /* Bloc d'instructions exécuté si la
    condition est vraie */
}
[else
{
    /* Bloc d'instructions exécuté si la
    condition est fausse */
}]
```

# Structure de contrôle Tant que... faire...

```
while (condition)
{
    /* Bloc d'instructions répété tant que la
    condition est vraie */
}
```

---

```
do {
    /* Bloc d'instructions exécuté une fois
    puis répété tant que la condition est
    vraie */
} while (condition) ;
```



# Structure de contrôle Tant que... faire...

```
for(avant; condition; fin_chaque_itération)  
{ /* Bloc d'instructions répété tant que la  
   condition est vraie */  
}
```

Équivalent à:

```
avant ;  
while (condition)  
{ /* Bloc d'instructions répété tant que la  
   condition est vraie */  
  fin_chaque_itération ;  
}
```

# Structure de contrôle switch...

```
switch (val)
{
  case v1:
    instructions exécutées si val==v1
  case v2:
    instructions exécutées si val==v2
    ou si val==v1
  ...
  default:
    instructions dans tous les cas
}
```

# L'instruction break

Permet de sortir d'une structure de contrôle

```
switch (val)
{
  case v1:
    instructions exécutées si val==v1
    break ; /* On sort du switch si val==v1 */
  case v2:
    instructions exécutées si val==v2
    ou si val==v1
    break ; /* On sort du switch si val==v2 */
  ...
  default:
    instructions exécutées dans tous les cas
    si val!=v1 et val!=v2
}
```

# Les tableaux

- Création / initialisation:

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

```
$tab2[] = 12 ;
```

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

```
$tab2[] = 2.5 ;
```

```
$tab3[0] = 12 ;
```

```
$tab3[1] = "fraise" ;
```

```
$tab3[2] = 2.5 ;
```

Clé	Valeur
0	12
1	"fraise"
2	2.5

# Les tableaux « à trous »

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

```
$tab4[0] = 12 ;  
$tab4[1] = "fraise" ;  
$tab4[2] = 2.5 ;  
$tab4[5] = "e15" ;
```

Clé	Valeur
0	12
1	"fraise"
2	2.5
<del>3</del>	<del></del>
<del>4</del>	<del></del>
5	"e15"

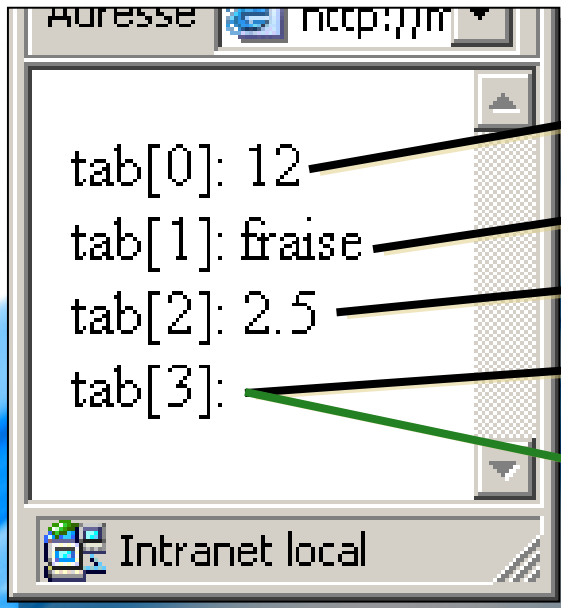
- Comment parcourir de tels tableaux ?

# Les tableaux « à trous » (suite)

Parcours classique :

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

4



Clé	Valeur
0	12
1	"fraise"
2	2.5
3	
4	
5	"e15"

# 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 */
```

```
}
```

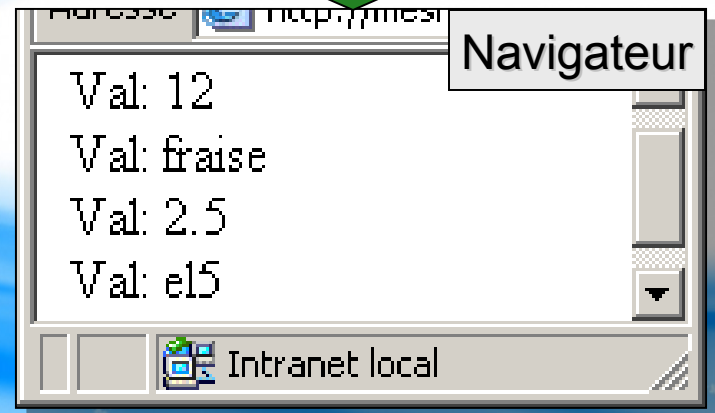
# Parcours de tableau : foreach

PHP

```
...  
$tab4[0] = 12 ;  
$tab4[1] = "fraise" ;  
$tab4[2] = 2.5 ;  
$tab4[5] = "e15" ;  
foreach($tab4 as $v)  
{  
    echo "Val: $v<br>\n";  
}  
...
```

HTML

```
...  
Val:12<br>\n  
Val:fraise<br>\n  
Val:2.5<br>\n  
Val:e15<br>\n  
...
```





# 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

```
$tab5['un']      = 12 ;  
$tab5['trois']   = "fraise" ;  
$tab5["deux"]   = 2.5 ;  
$tab5[42]       = "e15" ;
```

Clé	Valeur
"un"	12
"trois"	"fraise"
"deux"	2.5
42	"e15"

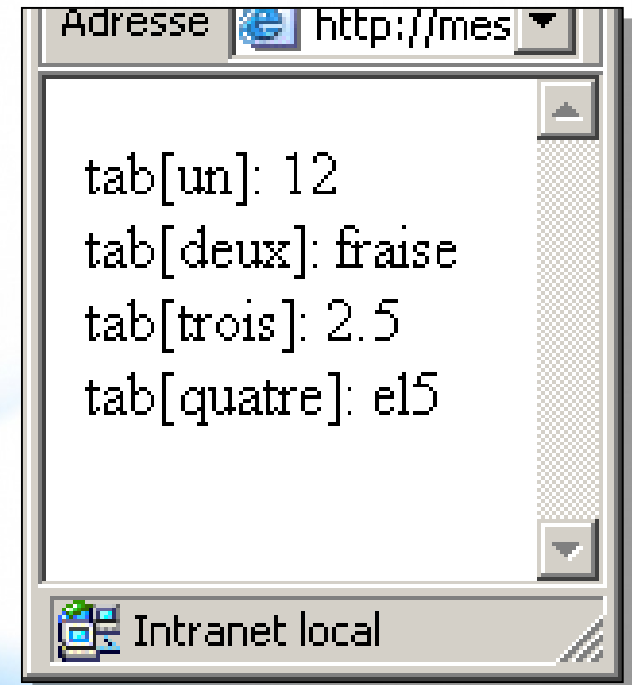
```
$tab6 = array('un'      => 12,  
              'trois'   => "fraise",  
              "deux"    => 2.5,  
              42        => "e15") ;
```

# 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
$html = <<<HTML
<html>
  <head><title>foreach clé</title>
  </head>
<body>
HTML;
$tab6 = array('un'      => 12,
              'deux'   => "fraise",
              "trois"  => 2.5,
              "quatre" => "e15") ;
foreach ($tab6 as $cle => $val)
{
  $html .= "tab[$cle]: $val<br>\n" ;
}
echo $html . "</body>\n</html>" ;
```



# Exemple de génération de code HTML

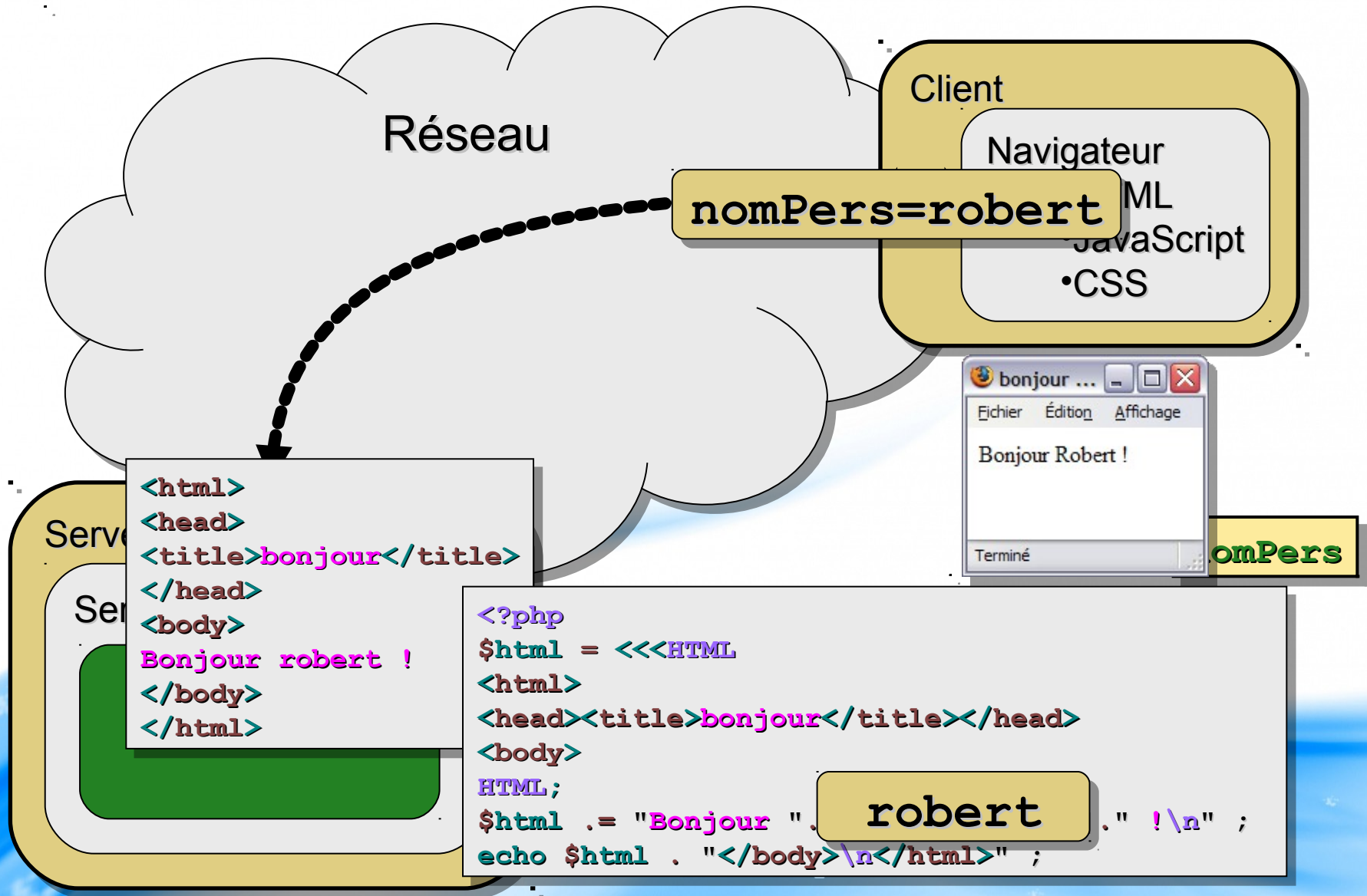
```
<?php
$html = <<<HTML
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
  Transitional//EN">
<html>
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=iso-8859-1">
    <title>Boucle</title>
  </head>
  <body>
HTML;
for ($i=1; $i<20; $i++) {
  $html .= "Le serveur compte... "
    . $i . "<br>\n" ;
}
$html .= <<<HTML
  </body>
</html>
HTML;
echo $html ;
```

Le serveur compte... 1  
Le serveur compte... 2  
Le serveur compte... 3  
Le serveur compte... 4  
Le serveur compte... 5  
Le serveur compte... 6  
Le serveur compte... 7  
Le serveur compte... 8  
Le serveur compte... 9  
Le serveur compte... 10  
Le serveur compte... 11  
Le serveur compte... 12  
Le serveur compte... 13  
Le serveur compte... 14  
Le serveur compte... 15  
Le serveur compte... 16  
Le serveur compte... 17  
Le serveur compte... 18  
Le serveur compte... 19

# Traitement des données de formulaires

- PHP permet de **traiter les données** saisies grâce à un formulaire HTML si le champ **ACTION** du formulaire désigne une page PHP du serveur.
- Après récupération par le serveur Web, les données sont contenues dans l'une des variables superglobales de type tableau associatif **\$\_GET** ou **\$\_POST** (ou **\$\_REQUEST**).
- La valeur peut être trouvée grâce à une clé **qui porte le même nom** que le champs du formulaire de la page HTML de saisie.

# Traitement des données de formulaires



# Exemple – Formulaire HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>formulaire</title>
</head>
<body>
  <form action="valide1.php" method="get">
    Nom: <input type="text" name="nomPers">
    <input type="submit" value="Envoyer">
  </form>
</body>
</html>
```



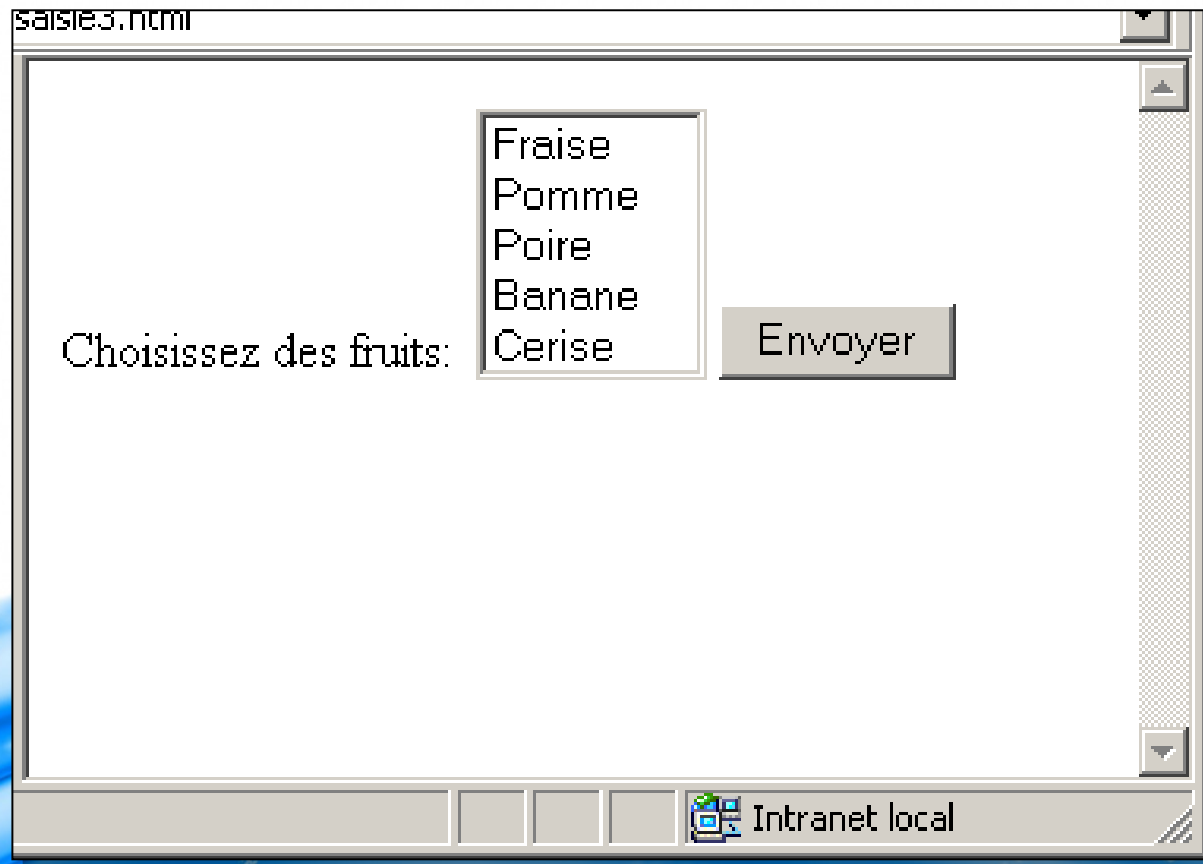
# Exemple – Traitement en PHP

```
<?php
$html = <<<HTML
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Validation</title>
</head>
<body>
HTML,
  if (isset($_GET['nomPers']))
  {
    if (!empty($_GET['nomPers']))
    {
      $html .= "Vous avez saisi '"
              . $_GET['nomPers'] . "'\n" ;
    }
    else
      $html .= "Aucune valeur saisie\n";
  }
  else
    $html .= "Utilisation incorrecte\n" ;
echo $html . "</body>\n</html>" ;
```

`$_GET['nomPers']` est-il défini ?

`$_GET['nomPers']` est-il vide ?

# Formulaires contenant des champs « SELECT »



# Formulaires contenant des champs « SELECT unique »

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Formulaire de saisie des fruits</title>
</head>
<body>
  <form action="valide3.php" method="get">
    Choisissez des fruits:&nbsp;
    <select name="sel">
      <option>Fraise
      <option>Pomme
      <option>Poire
      <option>Banane
      <option>Cerise
    </select>
    <input type="submit" value="envoyer">
  </form>
</body>
</html>
```

valide3.php?sel=Pomme

# Formulaires contenant des champs « SELECT multiple »

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Formulaire de saisie des fruits</title>
</head>
<body>
  <form action="valide3.php" method="get">
    Choisissez des fruits:&nbsp;
    <select name="sel" multiple>
      <option>Fraise
      <option>Pomme
      <option>Poire
      <option>Banane
      <option>Cerise
    </select>
    <input type="submit" value="envoyer">
  </form>
</body>
</html>
```

Envoyer

valide3.php?sel=Pomme&sel=Poire

???

# Formulaires contenant des champs « SELECT multiple »

```
<html>
<head>
  <title>Formulaire de saisie des fruits</title>
</head>
<body>
  <form action="valide3.php" method="get">
    Choisissez des fruits:&nbsp;  
    <select name="sel[]" multiple>
      <option>Fraise
      <option>Pomme
      Envoyer >Poire
      <option>Banane
      <option>Cerise
    </select>
    <input type="submit" value="envoyer">
  </form>
</body>
</html>
```

valide3.php?sel%5B%5D=Pomme&sel%5B%5D=Poire

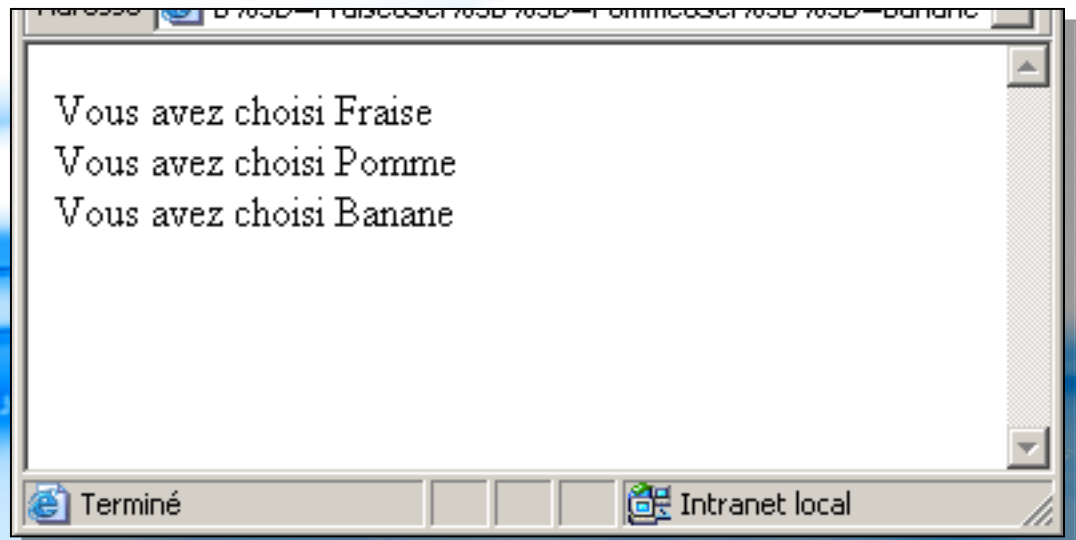
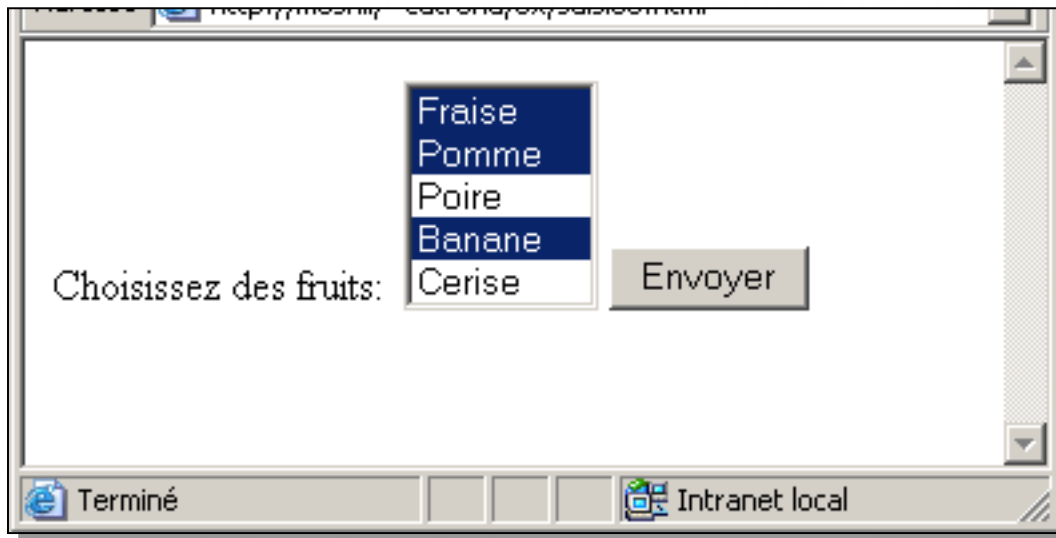
valide3.php?sel[]=Pomme&sel[]=Poire

# Traitement des données des champs « SELECT »

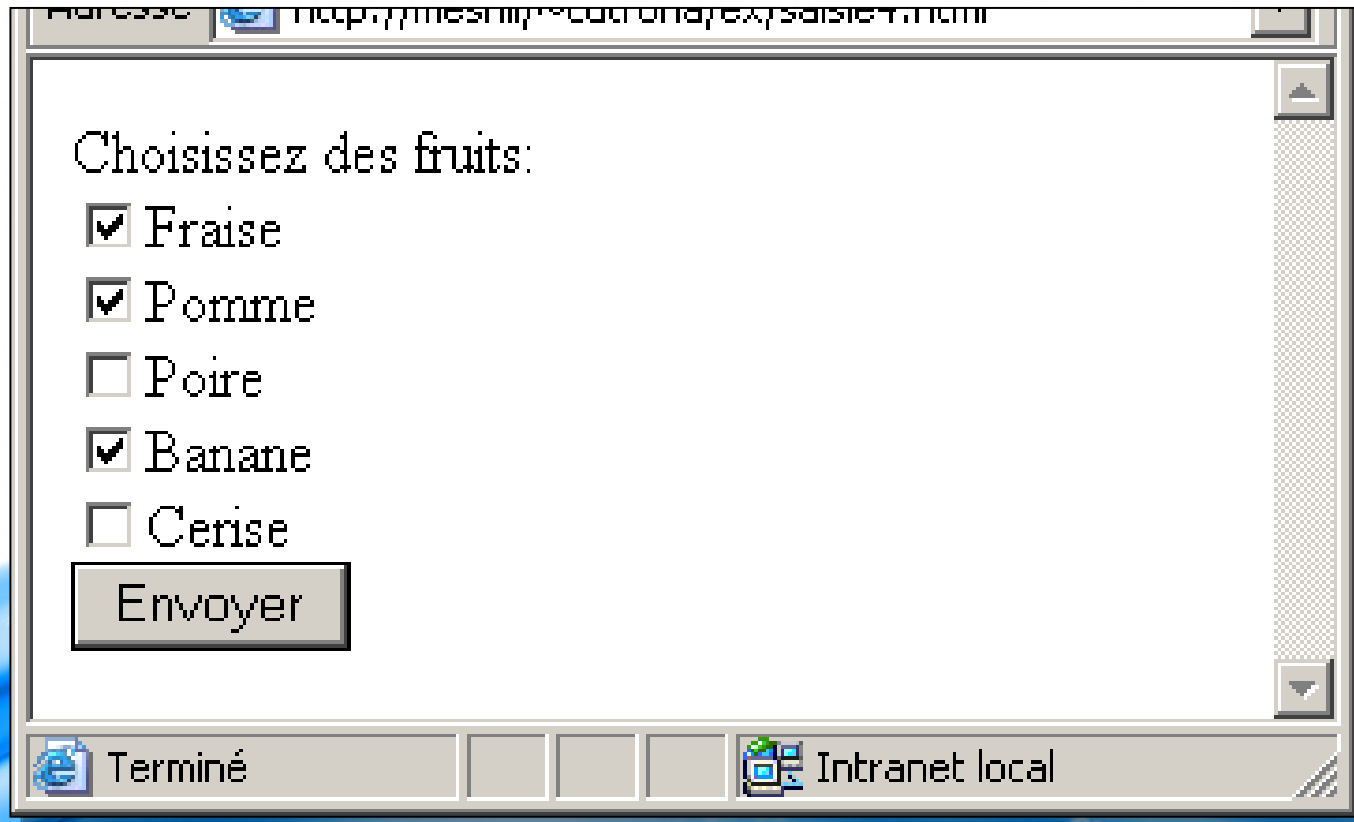
```
<?php
$html = <<<HTML
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Liste de fruits</title>
</head>
<body>
HTML;
  if (isset($_GET['sel']) && !empty($_GET['sel']))
  { /* La variable $_GET['sel'] est définie
    et elle n'est pas vide */
    foreach($_GET['sel'] as $fruit)
      $html .= "Vous avez choisi $fruit<br>\n" ;
    }
  else
    $html .= "Vous n'avez pas choisi de fruit\n" ;
echo $html . "</body>\n</html>" ;
```

`$_GET['sel']`  
est un tableau

# Résultat



# Formulaires contenant des champs « CHECKBOX »



A screenshot of a web browser window displaying a form. The browser's address bar shows the URL `http://mesnlp-edu.chrono.fr/ex/saisie.html`. The form content is as follows:

Choisissez des fruits:

- Fraise
- Pomme
- Poire
- Banane
- Cerise

Below the list is a button labeled "Envoyer".

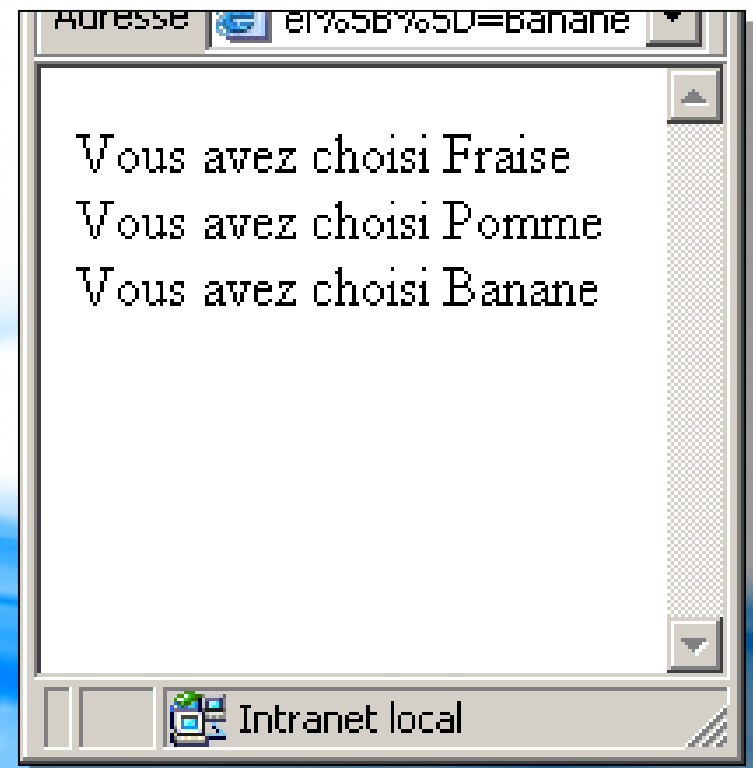
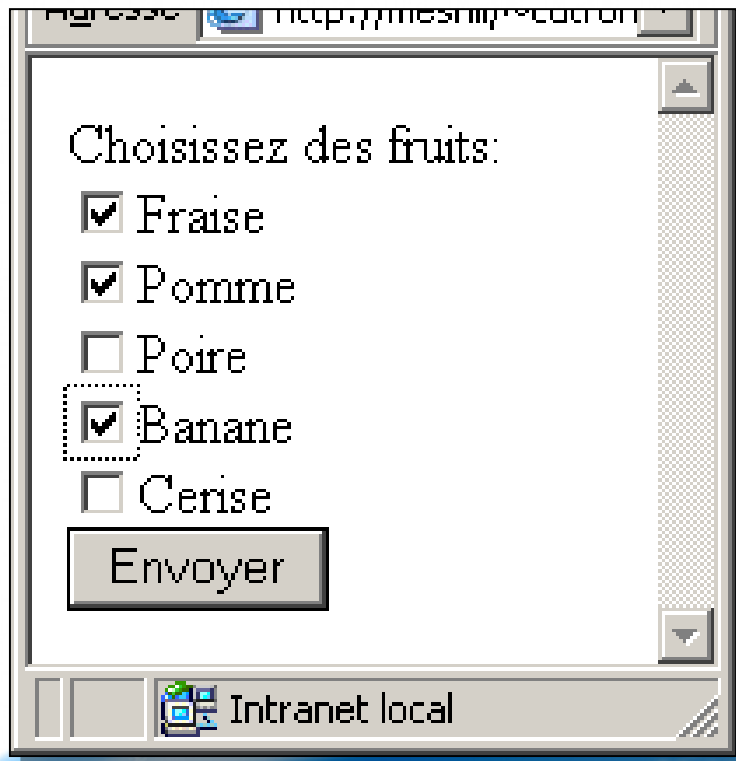
The browser's taskbar at the bottom shows several open windows: "Terminé" (with an Internet Explorer icon), two empty windows, and "Intranet local" (with a folder icon).



# Formulaires contenant des champs « CHECKBOX »

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Formulaire de saisie des fruits</title>
</head>
<body>
  <form name="formu" action="valide3.php" method="get">
    Choisissez des fruits    :<br>
    <input type="checkbox" name="sel[]" value="Fraise">Fraise<br>
    <input type="checkbox" name="sel[]" value="Pomme" >Pomme <br>
    <input type="checkbox" name="sel[]" value="Poire" >Poire <br>
    <input type="checkbox" name="sel[]" value="Banane">Banane<br>
    <input type="checkbox" name="sel[]" value="Cerise">Cerise<br>
    <input type="submit" value="Envoyer">
  </form>
</body>
</html>
```

# Résultat



# Références

```
$a = 12 ;
```

```
$b = $a ;
```

```
$c = &$a ;
```

```
$b = "coucou" ;
```

```
$c = 84 ;
```

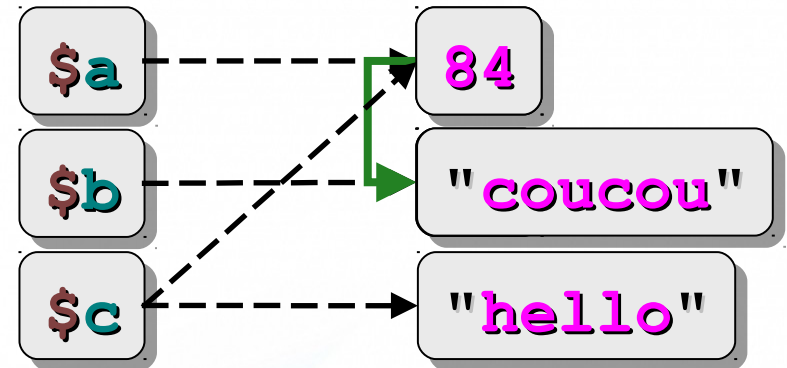
```
$a : 84
```

```
$b : coucou
```

```
$c : 84
```

```
unset($c) ;
```

```
$c = "hello" ;
```



# 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
```

# Fonctions utilisateur

- Valeur de retour

 **function** moyenne (\$a, \$b)

{ ... }

Typage faible de PHP :  
Aucune information

- Arguments

**function** moyenne (  \$a,  \$b )

{ ... }

Typage faible de PHP :  
Aucune information

# Mode de passage des arguments (types natifs)

```
<?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
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  
réussie

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

# 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
```



# Définition de fonctions fréquemment utilisées

- Certaines fonctions sont utilisées dans plusieurs scripts PHP
- Comment faire pour ne pas les définir dans chacune des pages ?
- Utilisation de :
  - `include("fichier") ;`
  - `require("fichier") ;`
  - `include_once("fichier") ;`
  - `require_once("fichier") ;`
- Permet d'inclure le contenu de *fichier* dans le script courant

# include et require

Fichier mafunction.php

```
<?
function mafunction($arg)
{
    if (isset($arg))
    {
        echo ("Vrai") ;
    }
    else
    {
        echo ("Faux") ;
    }
}
?>
```

Fichier utilisation1.php

```
...
require("mafunction.php")
mafunction(true) ;
...
```

Fichier utilisation2.php

```
...
include("mafunction.php")
...
$var=false ;
mafunction($var) ;
...
```

Fichier utilisation3.php

```
...
require("mafunction.php")
...
```

# Définition de constantes

```
<?php
```

```
define("ma_constante", "Bonjour à tous") ;
```

nom

valeur

Définition d'une constante

```
echo ma_constante ;
```

```
?>
```

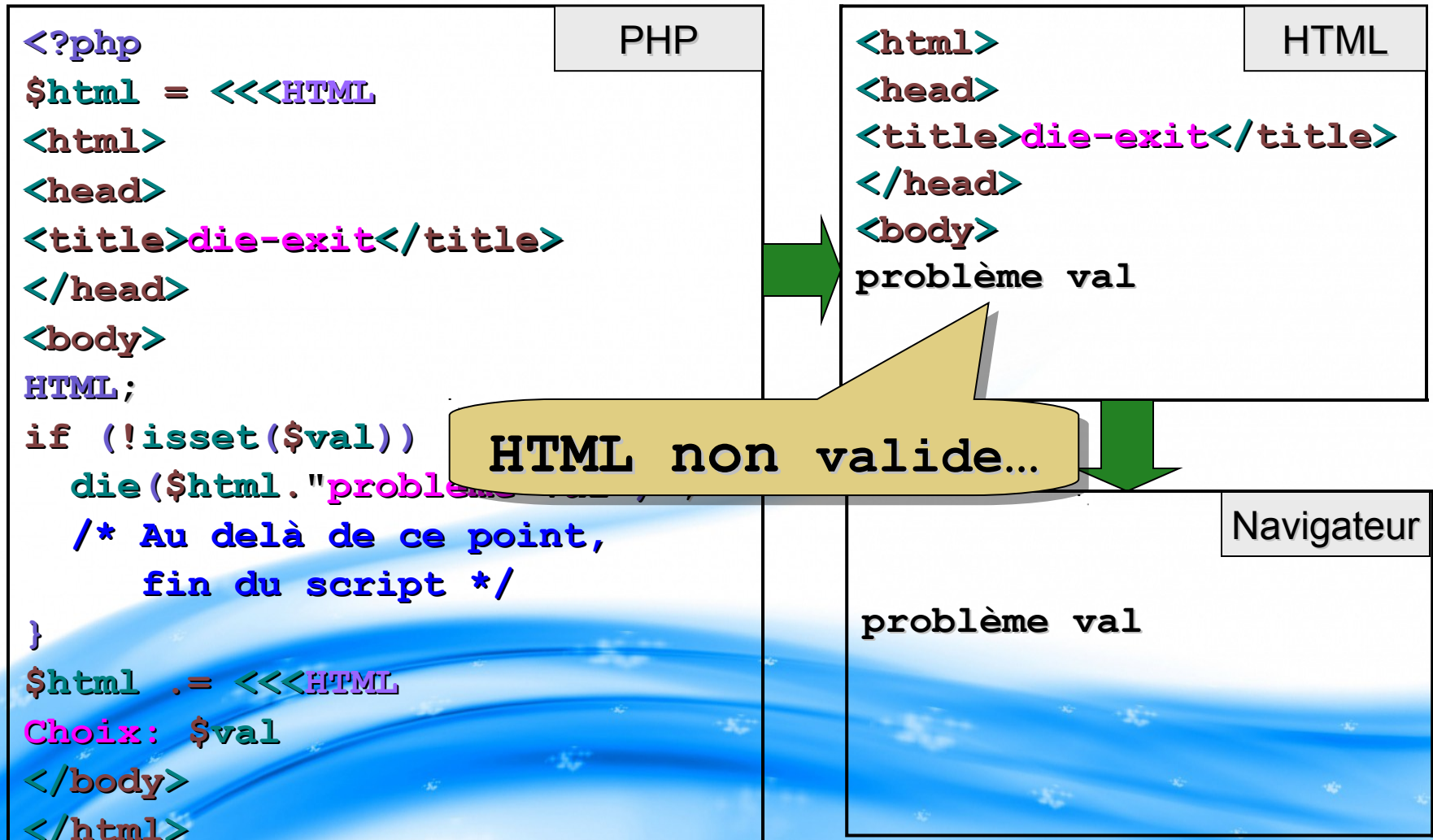
Utilisation de la constante

# Gestion des erreurs

- Dans certains cas, il n'est ni possible ni utile de poursuivre l'exécution du code PHP (variables non définies, valeurs erronées, échec de connexion, ...)
- Arrêt brutal de l'exécution du code:
  - **die** (*message*)
  - **exit** (*message*)

Envoie *message* au navigateur et termine l'exécution du script courant

# Gestion des erreurs – (Mauvais) Exemple



# Gestion de l'affichage des erreurs

■ `int error_reporting ( [int level] )`

Ancien niveau d'erreur

Sur un serveur en production, **toute erreur affichée** donne des indices sur les scripts et **rend le site vulnérable**

`php.ini`

`display_errors boolean`

## Constante

`E_ERROR`

`E_WARNING`

`E_PARSE`

`E_NOTICE`

`E_CORE_ERROR`

`E_CORE_WARNING`

`E_COMPILE_ERROR`

`E_COMPILE_WARNING`

`E_USER_ERROR`

Débogage

# Opérateur de contrôle d'erreur

```
$v = file("dummy.txt") Fichier absent  
or die("Problème de lecture") ;
```

```
Warning: file(dummy.txt): failed to open  
stream: No such file or directory in  
dummy.php on line 68  
Problème de lecture
```

```
$v = @file("dummy.txt")  
or die("Problème de lecture") ;
```

```
Problème de lecture
```