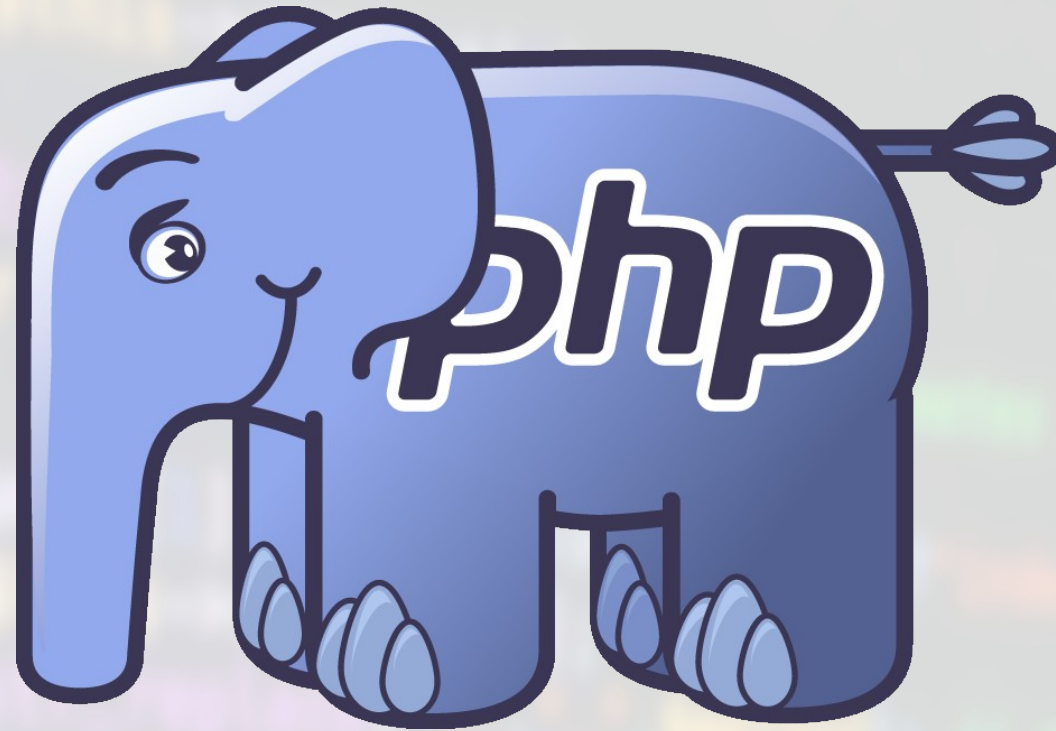


Programmation Web coté serveur : PHP



INTRODUCTION

Qu'est-ce que PHP ?

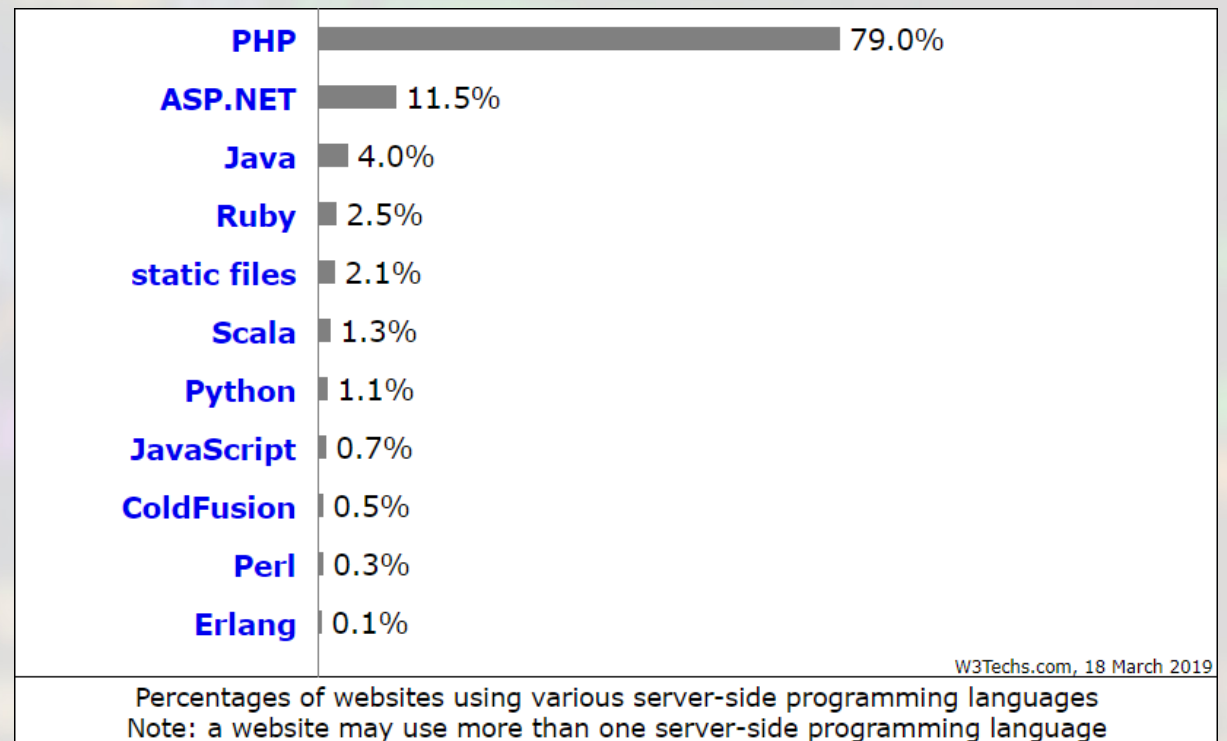
- Langage de **script**. Principalement utilisé **côté serveur**
- Créé en 1994-1995 par Rasmus Lerdorf
- Acronyme initial : **P**ersonal **H**ome **P**age
- Acronyme récursif : **P**HP: **H**ypertext **P**reprocessor
- Langage multi plate-forme (UNIX / Windows...)
- Open Source
- ≈80% des serveurs Web (<https://w3techs.com>, 03/2019)
- Versions actuelles : 7.2.x et 7.3.x

Langage de script ?

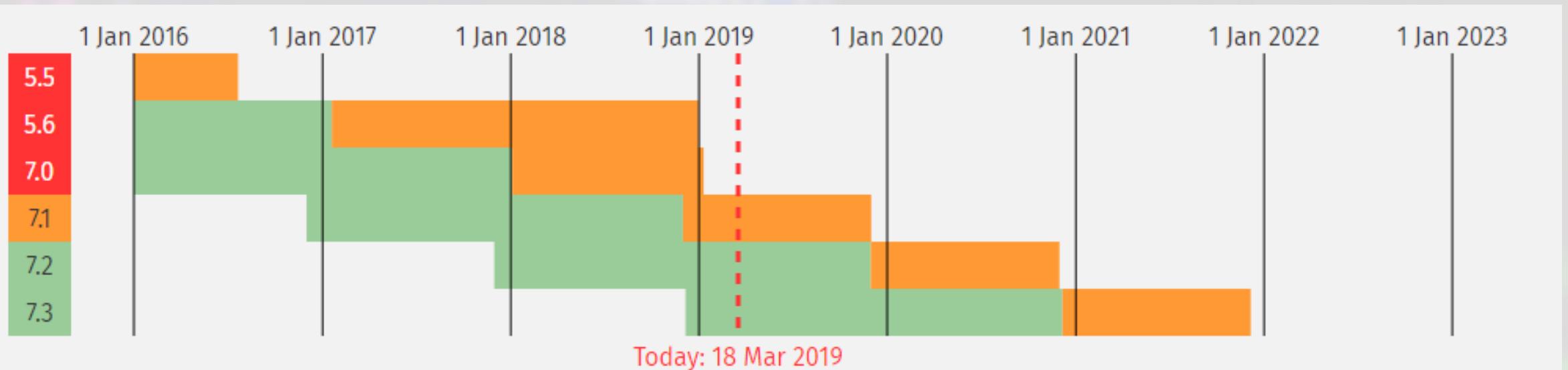
- Langage impératif
- Langage interprété
- Pas de compilation
- Le code source « est » le programme
- Typage faible
- Programmation orientée objet possible

Utilité et utilisation de PHP

- Création de contenus « dynamiques », fabriqués à la volée, construits à la demande (pages Web)
- Interface entre un serveur Web et des bases de données
- Création d'applications Web
- Le plus utilisé sur serveur



Cycle de vie des versions de PHP



Key

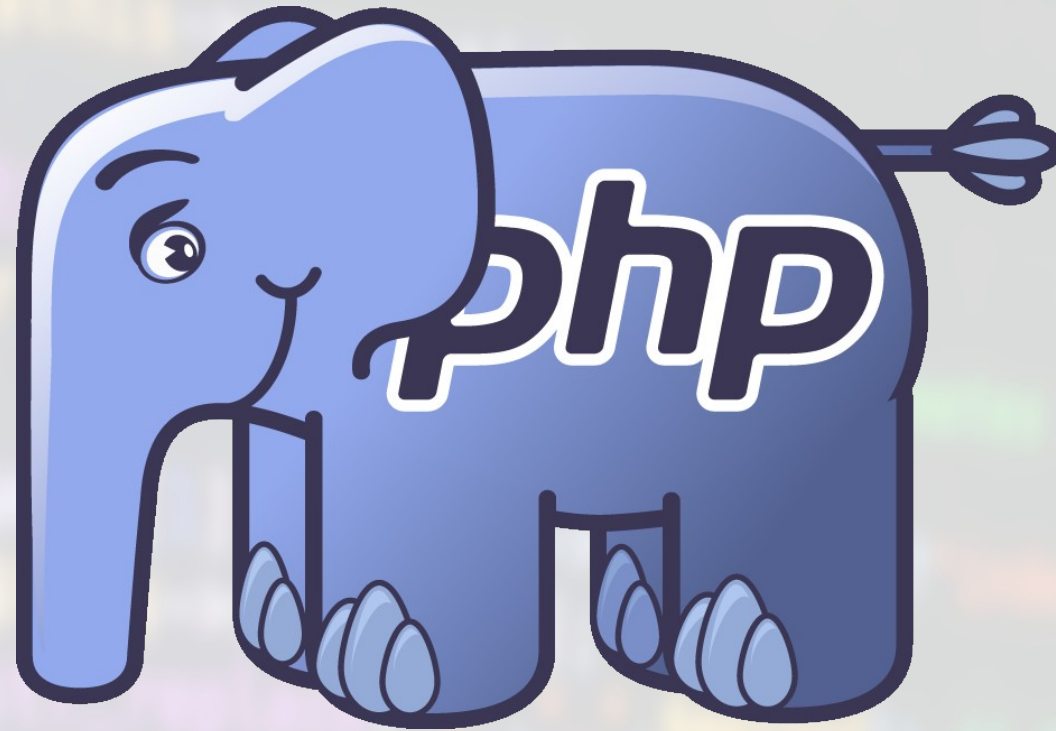
Active support	A release that is being actively supported. Reported bugs and security issues are fixed and regular point releases are made.
Security fixes only	A release that is supported for critical security issues only. Releases are only made on an as-needed basis.
End of life	A release that is no longer supported. Users of this release should upgrade as soon as possible, as they may be exposed to unpatched security vulnerabilities.

Principales fonctionnalités de PHP

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

Principales extensions de PHP

- **Apache**
- APC
- APCu
- APD
- **Arrays**
- BBCode
- BC Math
- bcompiler
- BLENC
- BLENC
- Bzip2
- Cairo
- **Calendar**
- chdb
- Classkit
- **Classes /Objects**
- COM
- Crack
- CSPRNG
- **Ctype**
- CUBRID
- cURL
- Cyrus
- **Date/Time**
- DBA
- dBase
- DB++
- dbx
- Direct IO
- **Directories**
- **DOM**
- **Data Structures**
- Eio
- Enchant
- **Error Handling**
- Ev
- Event
- Program execution
- Exif
- Expect
- FAM
- FANN
- FANN
- FrontBase
- FDF
- **Fileinfo**
- **Filesystem**
- Filter
- FastCGI
- Process Manager
- FriBiDi
- FTP
- **Function Handling**
- Gearman
- Gender
- Gettext
- Gmagick
- GMP
- GnuPG
- Gupnp
- haru
- **Hash**
- HRTIME
- htscanner
- Hyperwave API
- Firebird/InterBase
- IBM DB2
- **iconv**
- ID3
- Informix
- IIS
- **GD**
- ImageMagick
- IMAP
- included
- **PHP Options/Info**
- Ingres
- Inotify
- **intl**
- **JSON**
- Judy
- KADM5
- KTaglib
- Lapack
- **LDAP**
- Libevent
- **libxml**
- Lua
- LZF
- **Mail**
- Mailparse
- Math
- MaxDB
- **Multibyte String**
- Mcrypt
- MCVE
- Memcache
- Memcached
- Memtrack
- Mhash
- **Mimetype**
- Ming
- Misc.
- mnoGoSearch
- Mongo
- MongoDB\BSON
- MongoDB\Driver
- mqseries
- Msession
- mSQL
- Mssql
- MySQL (Original)
- MySQLi
- Mysqlnd
- mysqlnd_mem...
- mysqlnd_ms
- mysqlnd_mu x
- mysqlnd_qc
- mysqlnd_uh
- Ncurses
- Gopher
- Network
- Newt
- YP/NIS
- NSAPI
- OAuth
- OCI8
- oggvorbis
- OPcache
- OpenAL
- OpenSSL
- Output Control
- Paradox
- Parle
- Parsekit
- Password Hashing
- PCNTL
- **PCRE**
- PDF
- **PDO**
- 4D (PDO)
- CUBRID (PDO)
- MS SQL Server(PDO)
- Firebird(PDO)
- IBM (PDO)
- Informix (PDO)
- **MySQL (PDO)**
- Oracle (PDO)
- ODBC and DB2 (PDO)
- **PostgreSQL (PDO)**
- SQLite (PDO)
- MS SQL Server (PDO)
- PostgreSQL
- Phar
- Phdfs
- POSIX
- Proctitle
- PS
- Pspell
- pthreads
- Quickhash
- Radius
- **Rar**
- **Readline**
- RPM Reader
- RRD
- runkit
- SAM
- SCA
- scream
- SDO
- SDO DAS XML
- SDO-DAS-Relational
- Seaslog
- Semaphore
- **Sessions**
- Session PgSQL
- Shared Memory
- SimpleXML
- SNMP
- SOAP
- Sockets
- Sodium
- Solr
- Sphinx
- SPL
- SPL Types
- SQLite
- SQLite3
- SQLSRV
- ssdeep
- SSH2
- Statistics
- Stomp
- Streams
- Strings
- SVM
- SVN
- Swish
- Swoole
- Sybase
- Sync
- Taint
- TCP
- Tidy
- Tokenizer
- tokyo_tyrant
- Trader
- UI
- ODBC
- uopz
- URLs
- V8js
- **Variable handling**
- Varnish
- vpopmail
- WDDX
- Weakref
- win32ps
- win32service
- WinCache
- wkhtmltox
- xattr
- xdiff
- Xhprof
- XML Parser
- XMLDiff
- XMLReader
- XML-RPC
- XMLWriter
- XSL
- Yacnf
- Yaf
- Yaml
- Yar
- YAZ
- **Zip**
- Zlib
- 0MQ messaging
- ZooKeeper

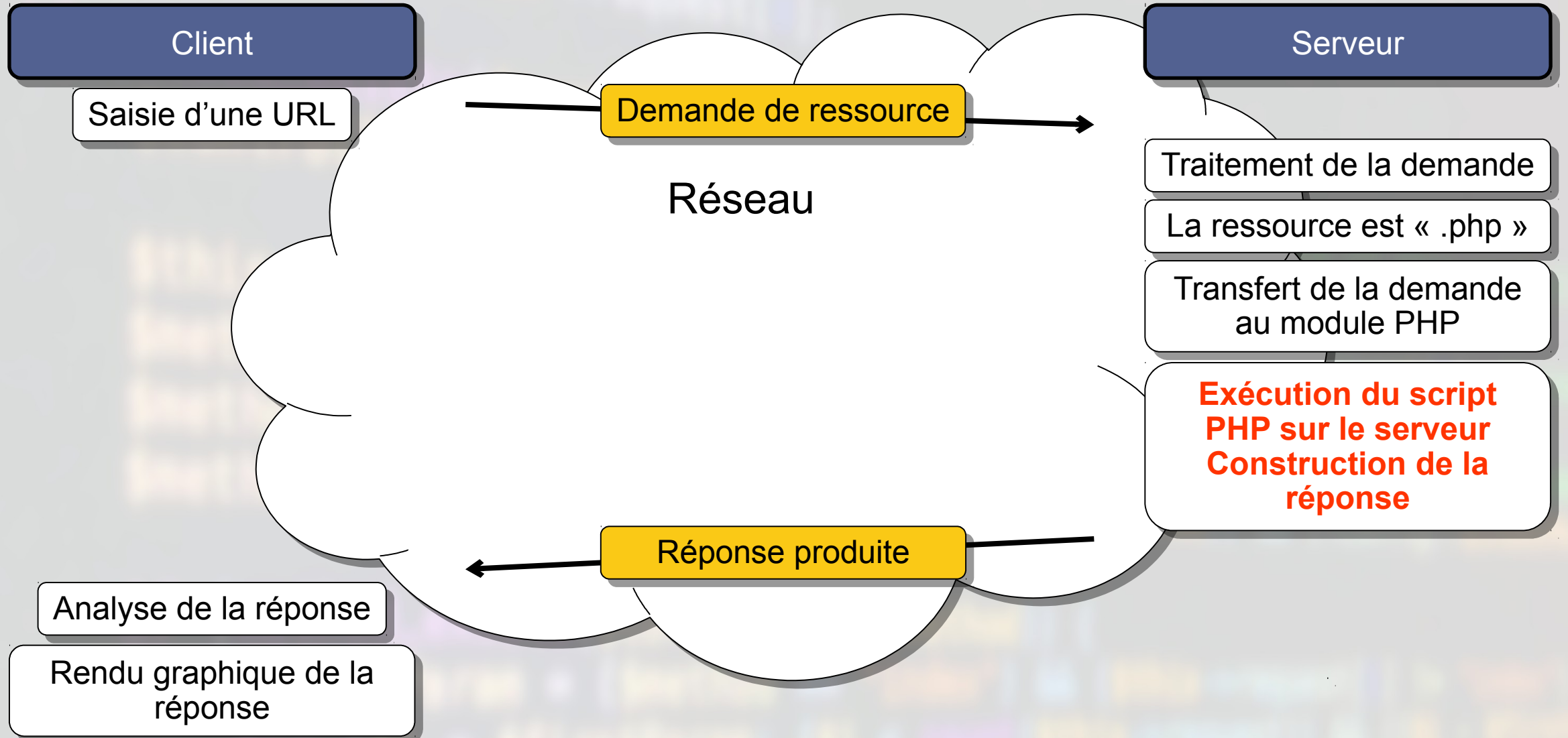


FONCTIONNEMENT DE PHP EN MODE WEB

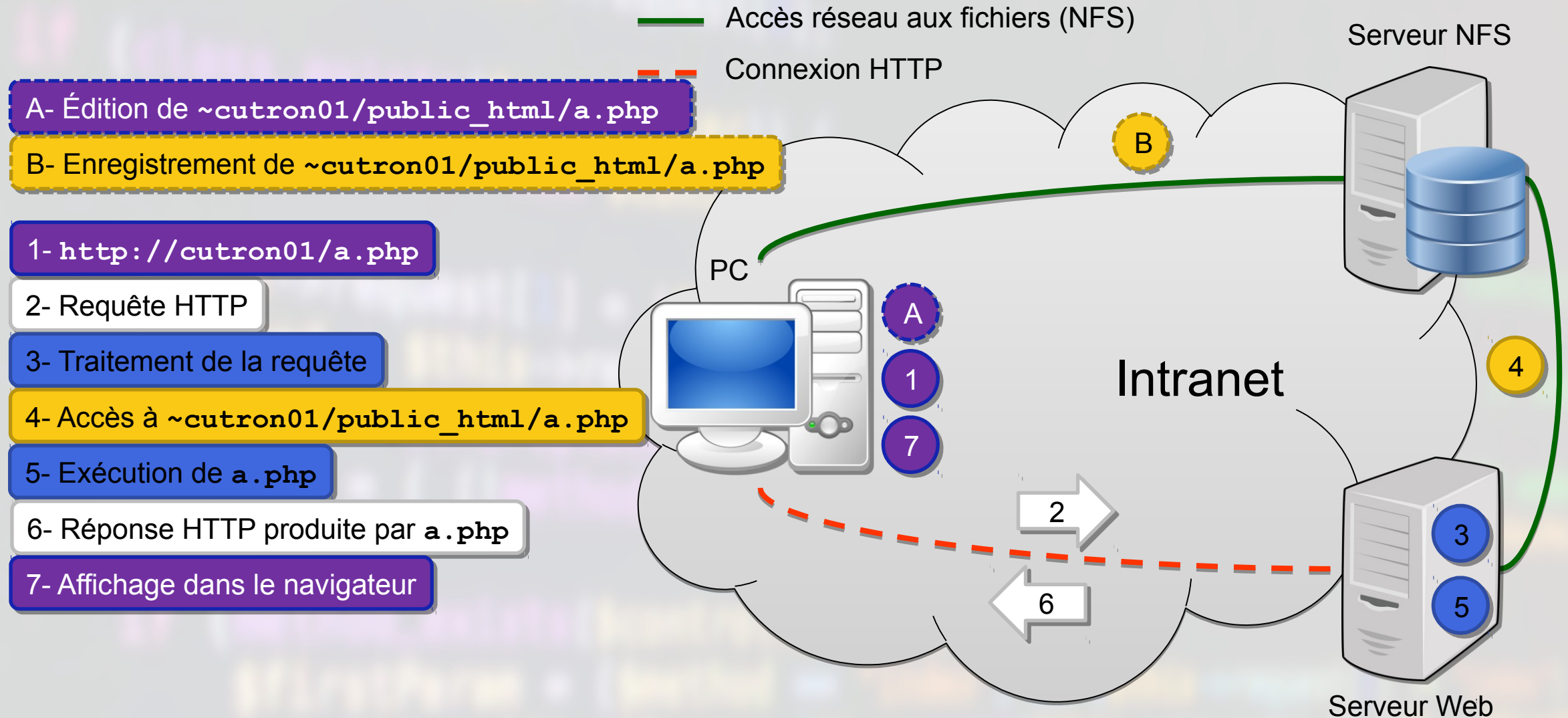
Fonctionnement de PHP en mode serveur



Fonctionnement de PHP en mode serveur



Fonctionnement de PHP en mode serveur au dpt INFO



Programme en PHP

Délimitation du code PHP dans le fichier .php :

- `<?php` Code PHP `?>`
- `<script language="PHP">`
Code PHP
`</script>`
- ~~`<? Code PHP ?>`~~
- ~~`<% Code PHP %>`~~

Fermeture optionnelle
et déconseillée

Confusion avec JavaScript
Ⓢ à bannir !!

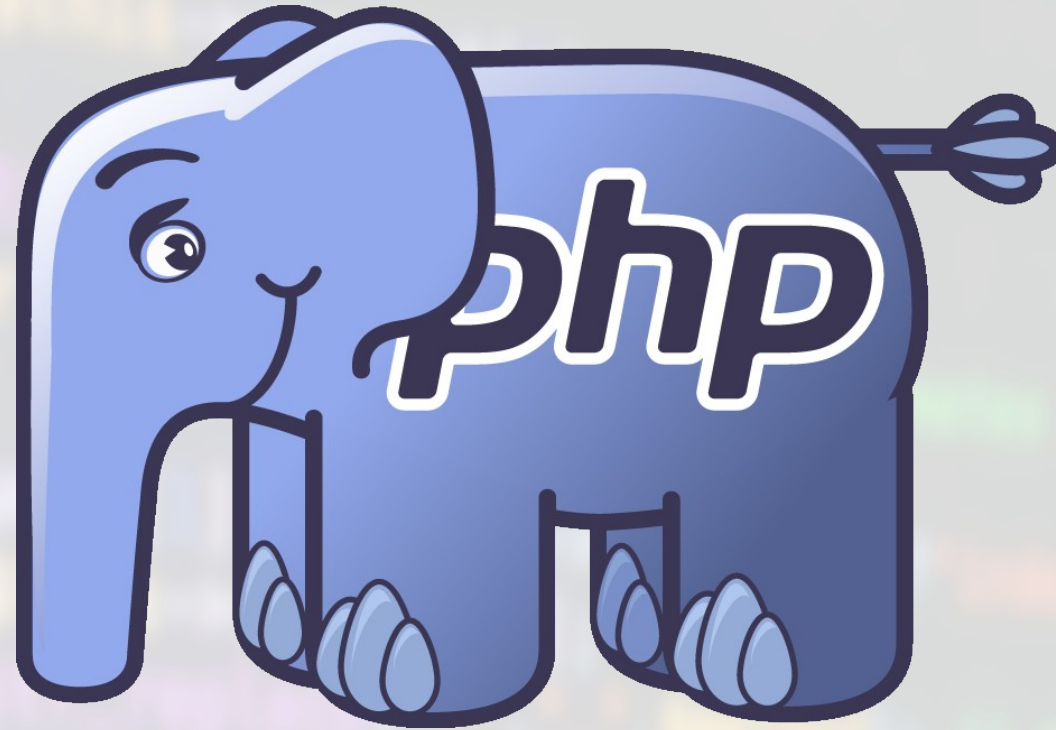
Dépend de la configuration
du serveur
Ⓢ à bannir !!

`short_open_tag`

`asp_tags`

Éléments de syntaxe PHP

- La syntaxe de PHP est celle de la 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`



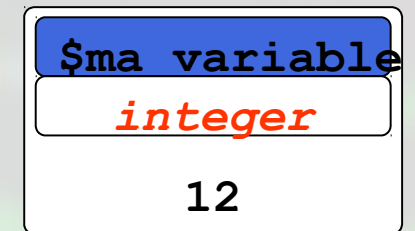
VARIABLES ET TYPES

Les variables et les types de données

- Tout identificateur commence par « \$ »
- Les affectations sont réalisées grâce à « = »
- Les types sont :
 - Numérique entier : 12 ou réel : 1.54
 - Chaîne: "Hello" ou 'Bonjour'
 - Booléen: true, false (PHP 4)
 - Tableau: \$tab[2]=12
 - Objet
 - Ressource
 - NULL
 - Callable
- Les variables ne sont pas explicitement déclarées

Les variables et les types de données

- La « déclaration » d'une variable correspond à sa **première affectation**
- Le **type** d'une variable est **dynamique** et est **déterminé par la valeur qui lui est affectée**
- Une variable possède donc :
 - Un **nom** (commençant par **\$**)
 - Un **type dynamique**
 - Une **valeur**
- **Adaptation possible du type selon le contexte** sans modification du type intrinsèque



Typage à l'affectation. Exemple

// Pas de déclaration préalable de variable

➔ `$test = 1.5 ;`

➔ `$test = 12 ;`

➔ `$test = array() ;`

➔ `$test = "10" ;`

<code>\$test</code>
<i>string</i>
"10"

Typage en fonction du contexte. Exemple

➔ \$nombre1 = 1.5 ;
➔ \$nombre2 = 12 ;
➔ \$chaine1 = "10" ;
➔ \$chaine2 = 'coucou' ;

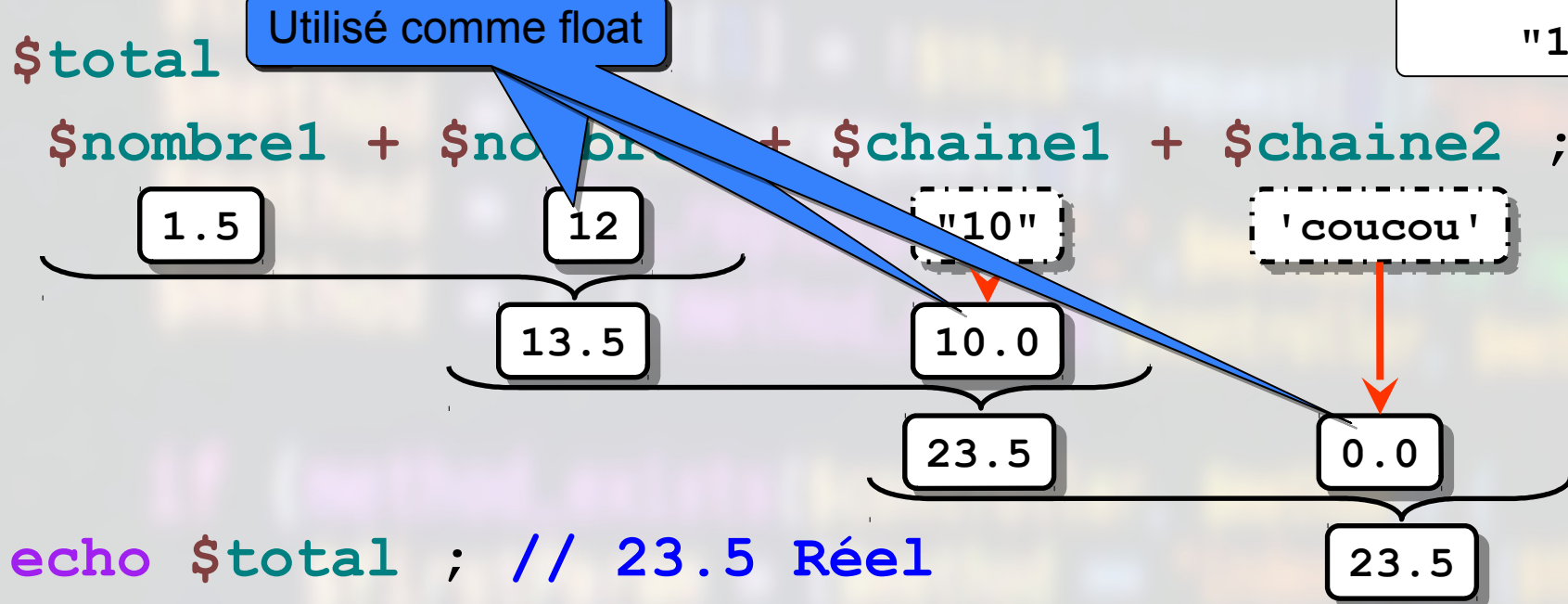
\$total
<i>float</i>
23.5

\$nombre1
<i>float</i>
1.5

\$nombre2
<i>integer</i>
12

\$chaine1
<i>string</i>
"10"

\$chaine2
<i>string</i>
"coucou"



Modification de types

- Le type désiré peut être précisé : cast
- `$variable = (nom_du_type) valeur`

```
$a = (integer) "10" ;
```

```
$b = (string) 12 ;
```

```
$c = (float) $b ;
```

<code>\$a</code>
<i>integer</i>
10

<code>\$b</code>
<i>string</i>
"12"

<code>\$c</code>
<i>float</i>
12.

Définition de constantes

```
<?php
```

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

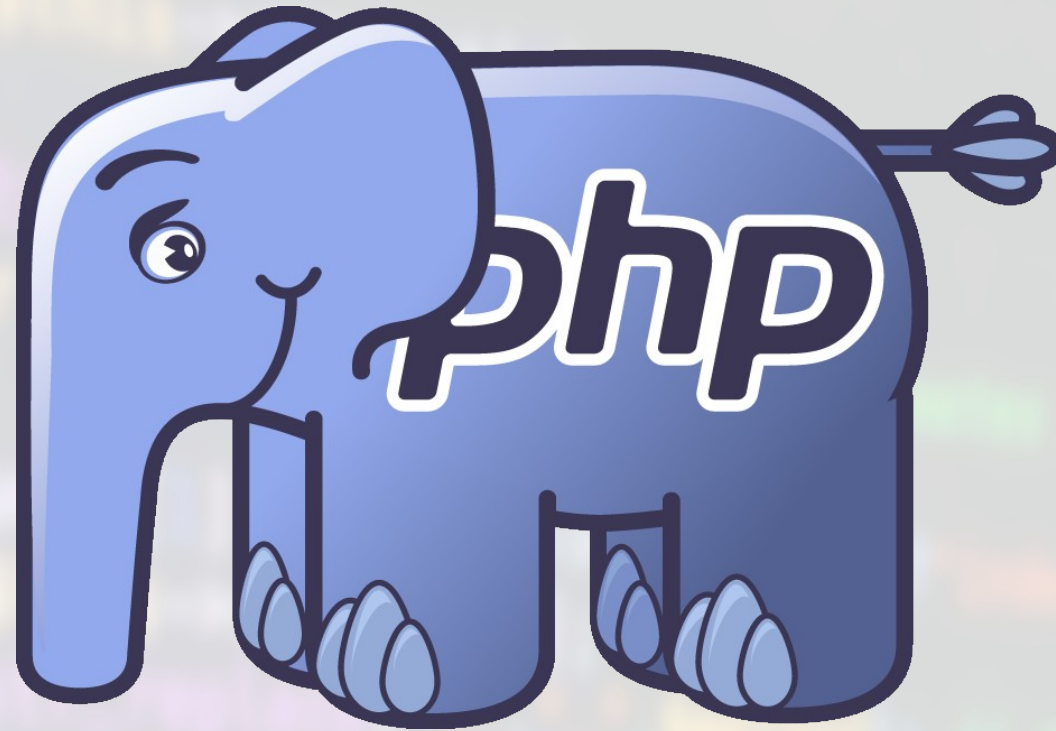
nom

valeur

Définition d'une constante

```
echo ma_constante ;
```

Utilisation de la constante



CHAÎNES DE CARACTÈRES

Substitution de variables dans les chaînes

- Guillemets doubles

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

chaîne

voici une chaîne

- Guillemets simples

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

chaîne

voici une \$a

Substitution de variables dans les chaînes

- Syntaxe *HereDoc*

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

chaîne

voici une chaîne
sur deux lignes ;-)

- Syntaxe *NowDoc* (PHP 5.3)

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

chaîne

voici une \$a
sur deux lignes ;-)

- **MARQUEDEFIN** au choix, après <<< et seule sur la ligne pour la fin

La commande echo

- Naïvement, 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" ;`
- Exactement, permet d'envoyer des octets au navigateur du client
 - Contenu HTML, XML, CSS, JavaScript, ...
 - Données d'une image
 - Contenu d'un fichier PDF, Flash, etc.
- Précisément, remplit le corps de la réponse HTTP (la charge utile) et engendre la production et l'envoi de la réponse HTTP complète

Hello world !

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

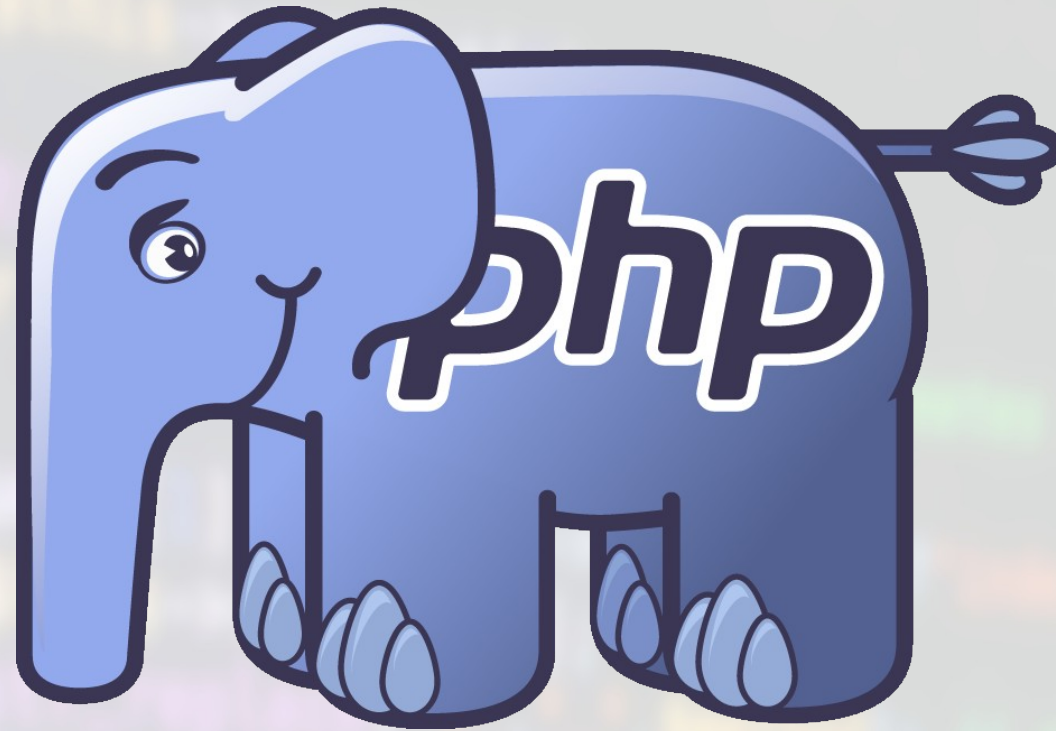
Serveur

```
<?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 ;
```

Navigateur

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

Impossible de voir le code PHP depuis le navigateur !!



OPÉRATEURS

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"

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-crémentement 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)

Comparaison large / stricte

```
$a = 12 ; $b = "12" ; $c = 12.0 ;
```

```
var_dump($a == $b) ;
```

boolean true

```
var_dump($a == $c) ;
```

boolean true

```
var_dump($c == $b) ;
```

boolean true

```
var_dump($a != $b) ;
```

boolean false

```
var_dump($a != $c) ;
```

boolean false

```
var_dump($c != $b) ;
```

boolean false

```
var_dump($a === $b) ;
```

boolean false

```
var_dump($a === $c) ;
```

boolean false

```
var_dump($c === $b) ;
```

boolean false

```
var_dump($a !== $b) ;
```

boolean true

```
var_dump($a !== $c) ;
```

boolean true

```
var_dump($c !== $b) ;
```

boolean true

\$a
<i>integer</i>
12

\$b
<i>string</i>
"12"

\$c
<i>float</i>
12

Les opérateurs logiques

[Expr1] and [Expr2]	Vrai si [Expr1] et [Expr2] sont vraies
[Expr1] && [Expr2]	idem
[Expr1] or [Expr2]	Vrai si [Expr1] ou [Expr2] sont vraies
[Expr1] [Expr2]	idem
[Expr1] xor [Expr2]	Vrai si [Expr1] ou [Expr2] sont vraies mais pas les deux
! [Expr1]	Vrai si [Expr1] est non vraie

Les opérateurs sur bits

$\$a \ \& \ \b	ET binaire
$\$a \ \ \b	OU binaire
$\$a \ ^ \ \b	XOR binaire
$\sim \ \$a$	Inversion bit à bit
$\$a \ << \ \b	$\$a$ décalé à gauche de $\$b$ rangs
$\$a \ >> \ \b	$\$a$ décalé à droite de $\$b$ rangs

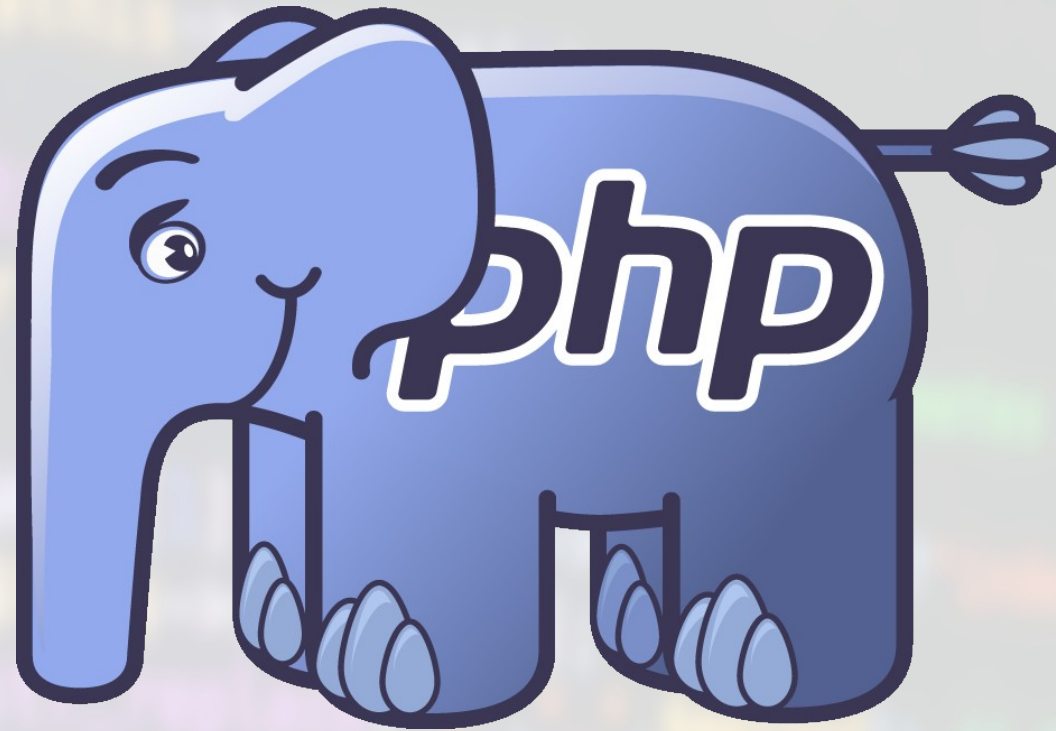
Précédence des opérateurs

Opérateurs
<code>new</code>
<code>[</code>
<code>++ --</code>
<code>! ~ - (int) (float) (string) (array) (object) @</code>
<code>* / %</code>
<code>+ - .</code>
<code><< >></code>
<code>< <= > >=</code>
<code>== != === !==</code>
<code>&</code>
...

Précédence des opérateurs

Opérateurs	
...	
^	
&&	
? :	
= += -= *=	
and	
xor	
or	

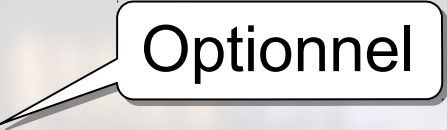
En cas de doute,
utilisez les parenthèses ;-)



STRUCTURES DE CONTRÔLE

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 selon...

```
switch (val)
{
    case v1:
        instructions exécutées si val==v1
    case v2:
    case v3:
        instructions exécutées si val==v2
        ou si val==v3
        ou si val==v1
    ...
    default:
        instructions dans tous les cas
}
```

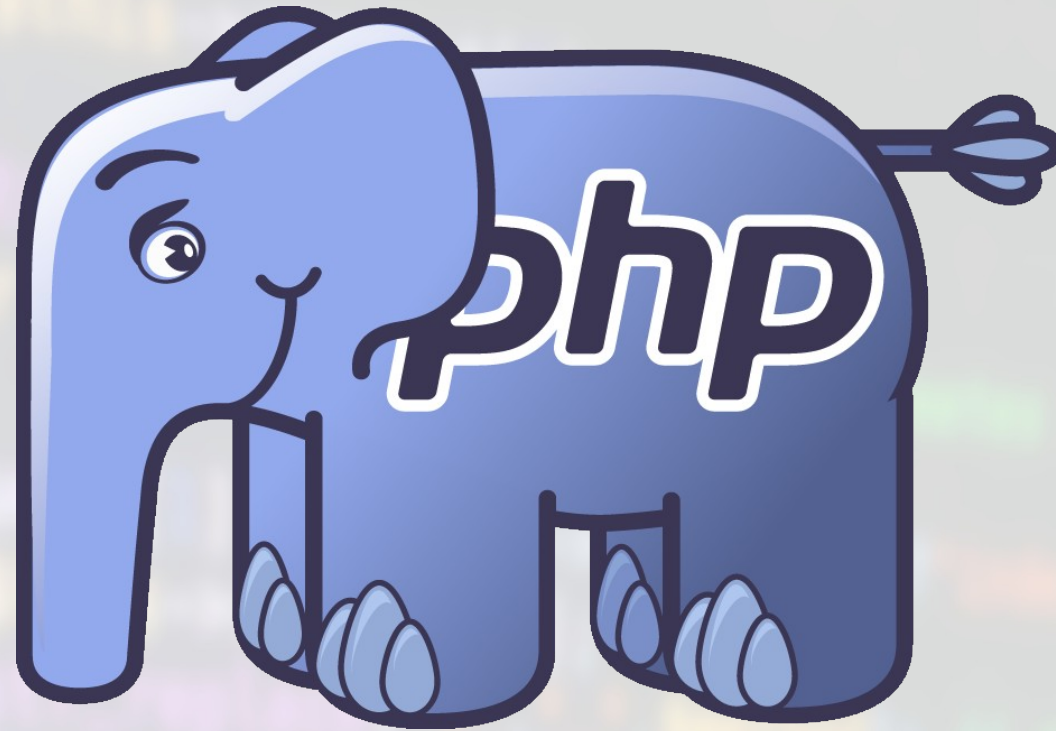
- Fonctionne exactement comme une série de `if/else` avec comparaison large
- `val`, `v1`, `v2`, ... peut être de type `integer`, `float`, `string`, `NULL`

L'instruction **break**

Permet de sortir d'une structure de contrôle

```
switch (val)
{
    case v1:
        instructions exécutées si val==v1
        break ; /* Sortie du switch si val==v1 */
    case v2:
    case v3:
        instructions exécutées si val==v2 ou si val==v3
        ou si val==v1
        break ; /* Sortie du switch si val==v2 ou val==v3 */
    default:
        instructions exécutées dans tous les cas
        si val!=v1 et val!=v2 et val!=v3
}
```

Cas rendus exclusifs si **break** présent pour chaque **case**



TABLEAUX

Principes généraux

- Un tableau PHP est en réalité une **carte ordonnée**
- Une carte associe des valeurs à des clés
- Un tableau PHP peut être vu comme une série ordonnées de valeurs (peu importe leur type)
- Chaque valeur est étiquetée, repérée par une clé **int** ou **string**
- Ils sont dits **associatifs**

Clé	0	1	2	3
Valeur	12	15.2	"42"	NULL

Clé	"début"	12	"a"	"valeur"
Valeur	false	3	16	"bonjour"

Tableaux « classiques » : indexés

- Création / initialisation:

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

```
$tab1b = [ 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

Tableaux associatifs : syntaxe

```
$tab5[ 'un' ] = 12 ;
```

```
$tab5[ 'trois' ] = "fraise" ;
```

```
$tab5[ "deux" ] = 2.5 ;
```

```
$tab5[ 42 ] = "e15" ;
```

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

```
$tab7 = [ 'un' => 12, 'trois' => "fraise",  
          "deux" => 2.5, 42 => "e15" ] ;
```

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

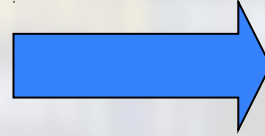
Structure de contrôle « Pour chaque... »

Des tableaux associatifs ne peuvent PAS être parcourus grâce aux indices des cases contenant les éléments...

```
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

```
...  
$tab4[0] = 12 ;  
$tab4[6] = "fraise" ;  
$tab4[2] = 2.5 ;  
$tab4[5] = "e15" ;  
foreach($tab4 as $v)  
{  
    echo "<p>Val: $v\n";  
}  
...
```



```
...  
<p>Val: 12\n<p>Val: fraise\n<p>Val: 2.5\n<p>Val: e15\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, ... );
```

```
$tab = [ cle1 => val1, cle2 => val2, ... ];
```

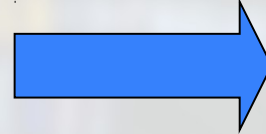
Structure de contrôle « Pour chaque... »

```
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 */
}
```

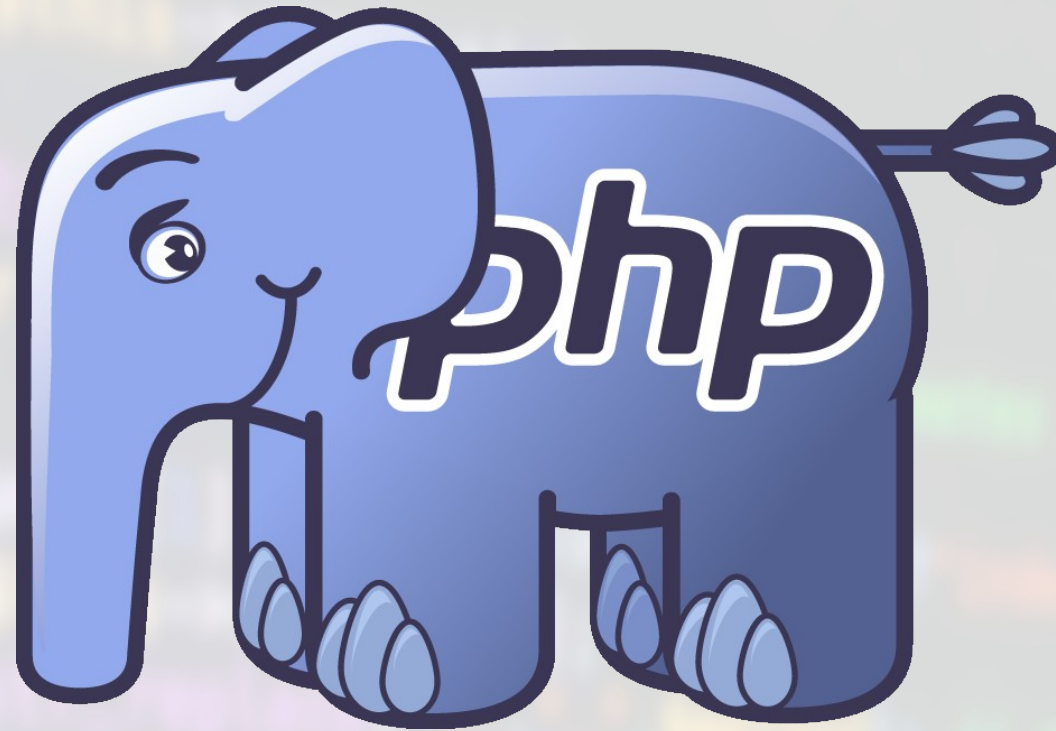

Parcours de tableau

```
<?php
$html = <<<HTML
<!doctype html>
<html lang="fr">
  <head><title>foreach clé</title>
  </head>
<body>
HTML;
$tab6 = array('un'      => 12,
              'trois'   => "fraise",
              "deux"    => 2.5,
              42        => "e15") ;

foreach ($tab6 as $cle => $val)
{
    $html .= "<p>tab[$cle]: $val\n" ;
}
echo $html . "</body>\n</html>" ;
```



```
...
<p>tab[un]:12\n
<p>tab[trois]:fraise\n
<p>tab[deux]:2.5\n
<p>tab[42]:e15\n
...
```

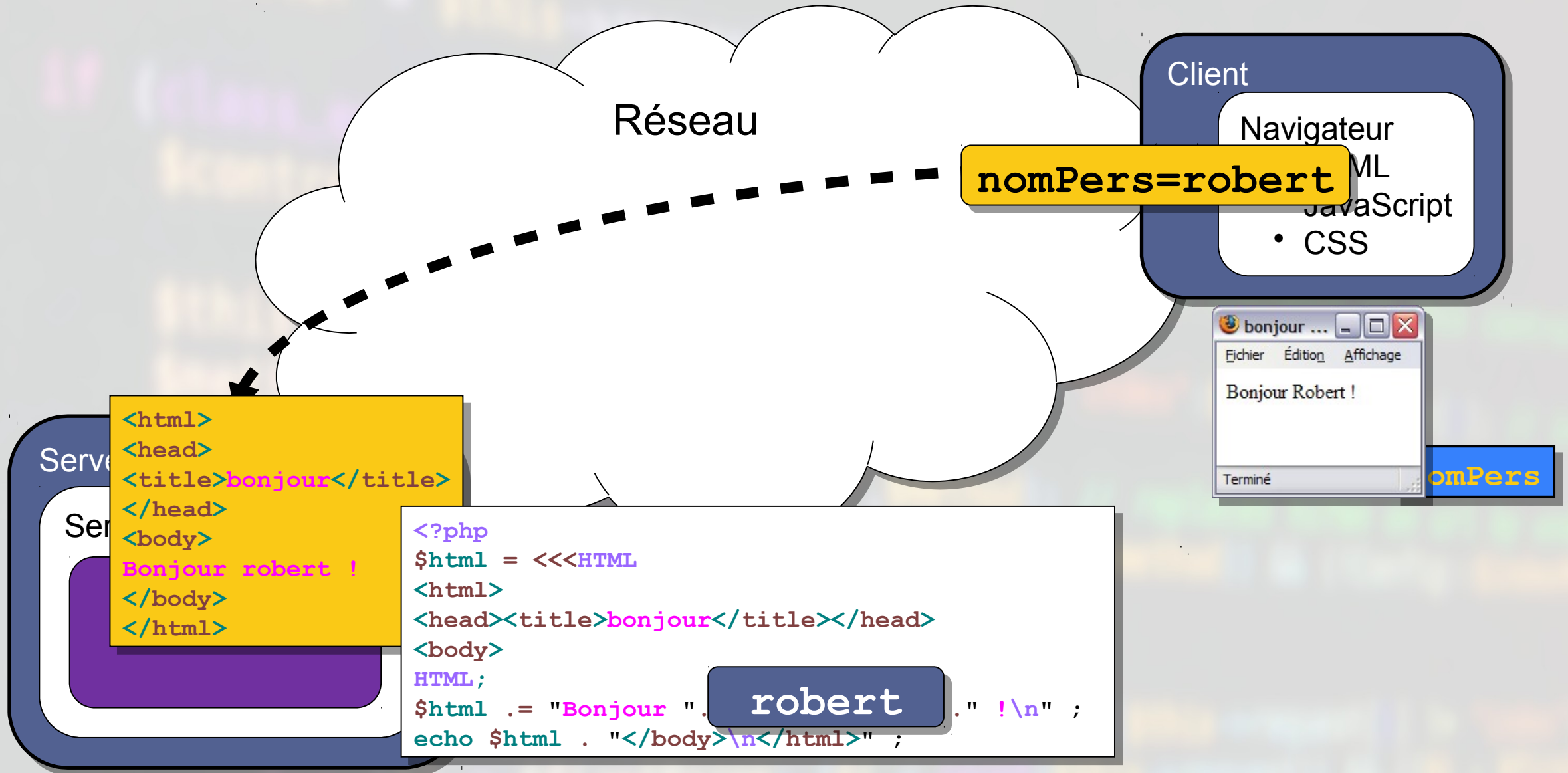


DONNÉES DE FORMULAIRES

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** selon la méthode de la requête (ou **\$_REQUEST** qui reçoit le contenu de **\$_GET** et **\$_POST**).
- La valeur peut être trouvée grâce à une clé **qui porte le même nom** que le champ du formulaire de la page HTML de saisie.

Traitement des données de formulaires



Exemple – Formulaire HTML

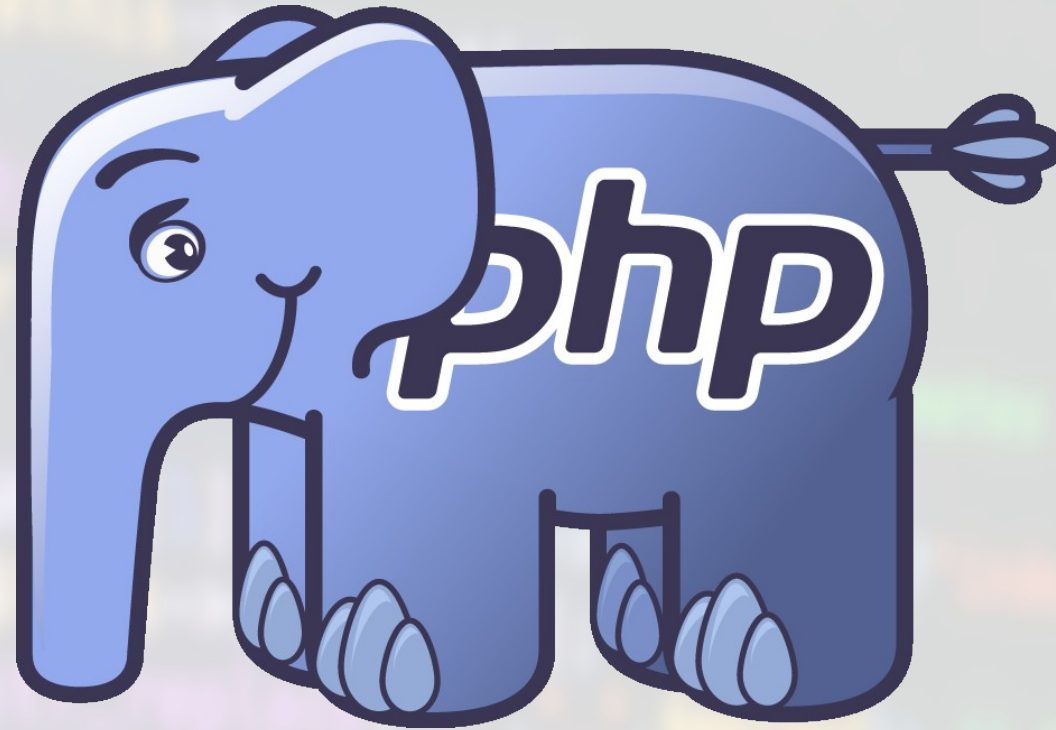
```
<!doctype html>
<html lang="fr">
  <head>
    <title>formulaire</title>
  </head>
  <body>
    <form action="valide1.php" method="get">
      Nom: <input type="text" name="nomPers">
      <input type="submit" value="Valider">
    </form>
  </body>
</html>
```


Exemple – Traitement en PHP

```
<?php
$html = <<<HTML
<!doctype html>
<html lang="fr">
  <head><title>bonjour</title></head>
  <body>
HTML;
  if (isset($_GET['nomPers']))
    if (!empty($_GET['nomPers']))
      $html .= "Bonjour " . $_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 ?



DONNÉES DE FORMULAIRES : QUELQUES EXEMPLES

Formulaires contenant des champs « SELECT »



saisie3.html

Choisissez des fruits:

- Fraise
- Pomme
- Poire
- Banane
- Cerise

Envoyer

Intranet local

Formulaires contenant des champs « SELECT unique »

```
<!doctype html>
<html lang="fr">
<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>
```

The diagram illustrates the form's action and the selected value. A dotted arrow points from the `name="sel"` attribute of the `<select>` tag to a box containing the URL `valide3.php?sel=Pomme`. Above this box is a button labeled "Envoyer", which corresponds to the `<input type="submit" value="envoyer">` in the code.

Formulaires contenant des champs « SELECT multiple »

```
<!doctype html>
<html lang="fr">
<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 »

```
<!doctype html>
<html lang="fr">
<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%5B%5D=Pomme&sel%5B%5D=Poire

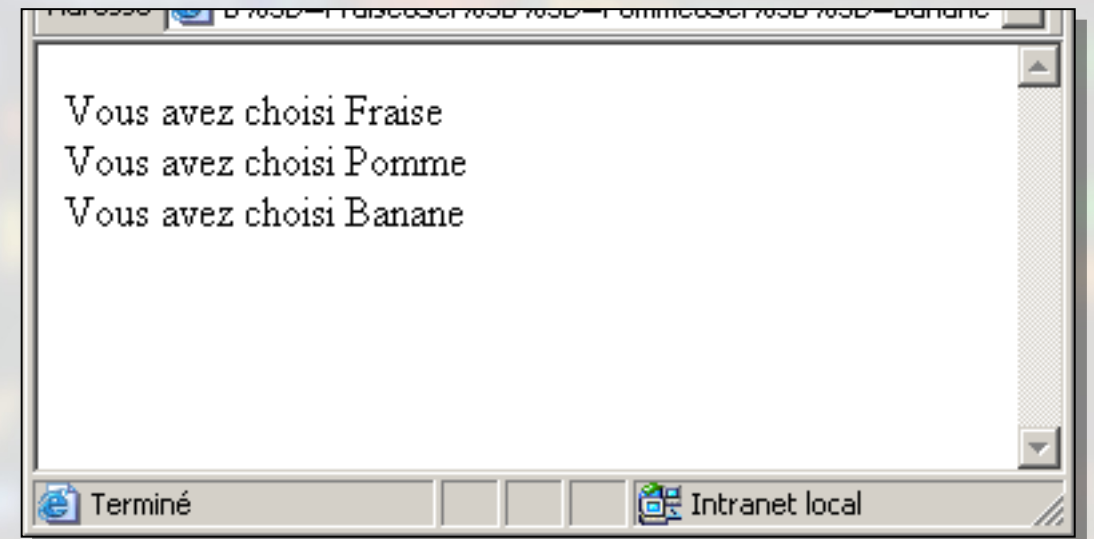
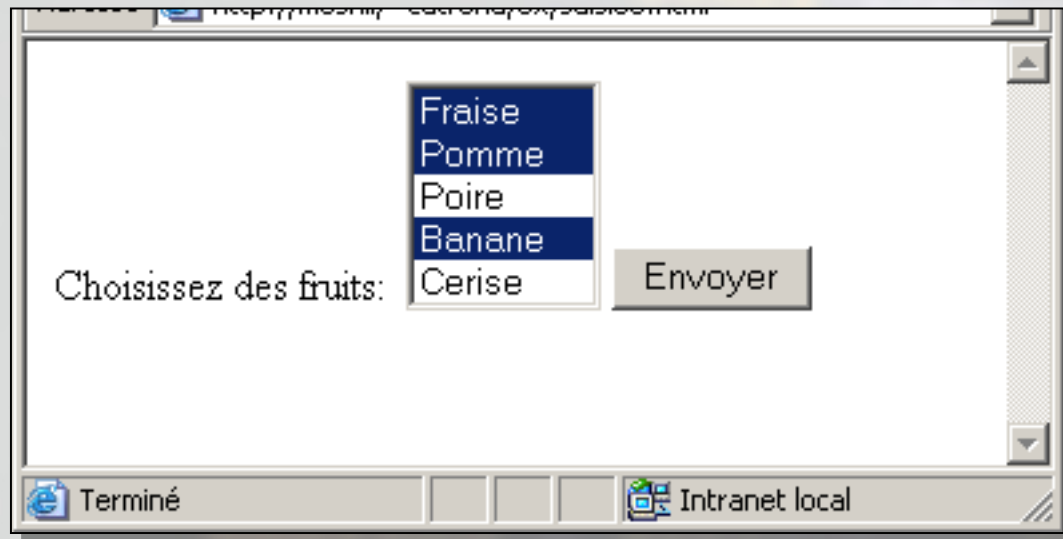
valide3.php?sel[]=Pomme&sel[]=Poire

Traitement des données des champs « SELECT »

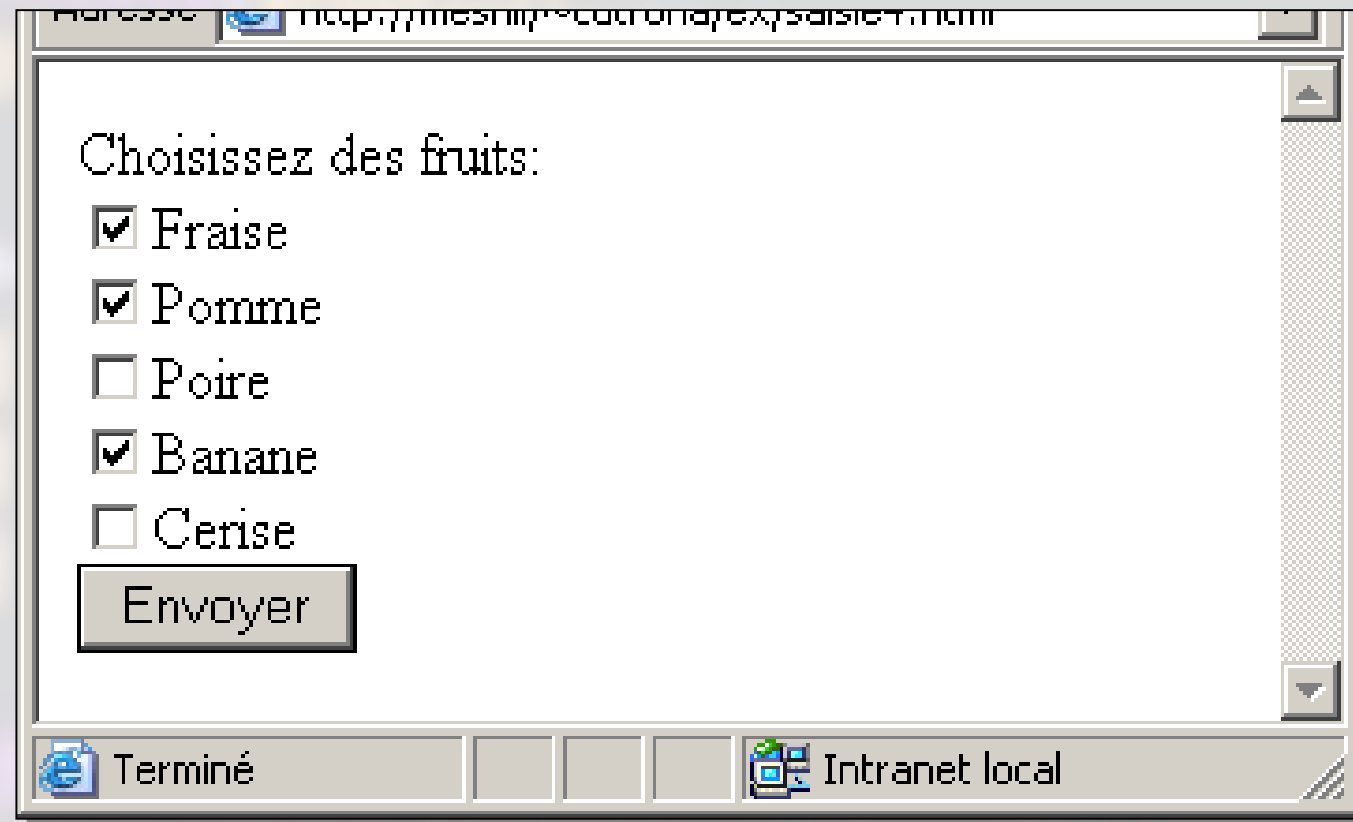
```
<?php
$html = <<<HTML
<!doctype html>
<html lang="fr">
<head>
    <title>Liste de fruits</title>
</head>
<body>
HTML;
    if (isset($_GET['sel']) && is_array($_GET['sel']))
    { /* La variable $_GET['sel'] est définie et elle est un tableau */
        foreach($_GET['sel'] as $fruit)
            $html .= "<p>Vous avez choisi $fruit\n" ;
        }
    else
        $html .= "<p>Vous n'avez pas choisi de fruit\n" ;
$html .= "</body>\n</html>" ;
echo $html ;
```

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

Résultat



Formulaires contenant des champs « CHECKBOX »



A screenshot of a web browser window displaying a form titled "Choisissez des fruits:". The form contains five checkboxes, each followed by a fruit name: "Fraise" (checked), "Pomme" (checked), "Poire" (unchecked), "Banane" (checked), and "Cerise" (unchecked). Below the checkboxes is a button labeled "Envoyer". The browser's address bar shows the URL "http://mesnlp-edu.chonay.fr/saisie.html". The taskbar at the bottom shows a "Terminé" icon and an "Intranet local" icon.

Choisissez des fruits:

- ☒ Fraise
- ☒ Pomme
- ☐ Poire
- ☒ Banane
- ☐ Cerise

Envoyer

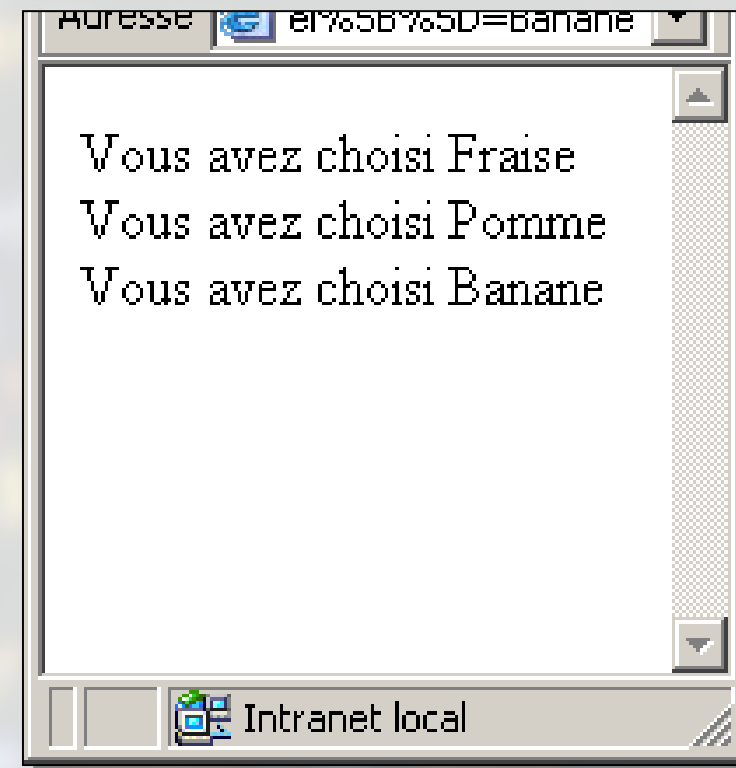
Formulaires contenant des champs « CHECKBOX »

```
<!doctype html>
<html lang="fr">
  <head>
    <title>Formulaire de saisie des fruits</title>
  </head>
  <body>
    <form name="formu" action="valide3.php" method="get">
      Choisissez des fruits :
      <label><input type="checkbox" name="sel[]" value="Fraise">Fraise</label>
      <label><input type="checkbox" name="sel[]" value="Pomme">Pomme</label>
      <label><input type="checkbox" name="sel[]" value="Poire">Poire</label>
      <label><input type="checkbox" name="sel[]" value="Banane">Banane</label>
      <label><input type="checkbox" name="sel[]" value="Cerise">Cerise</label>
      <input type="submit" value="Envoyer">
    </form>
  </body>
</html>
```

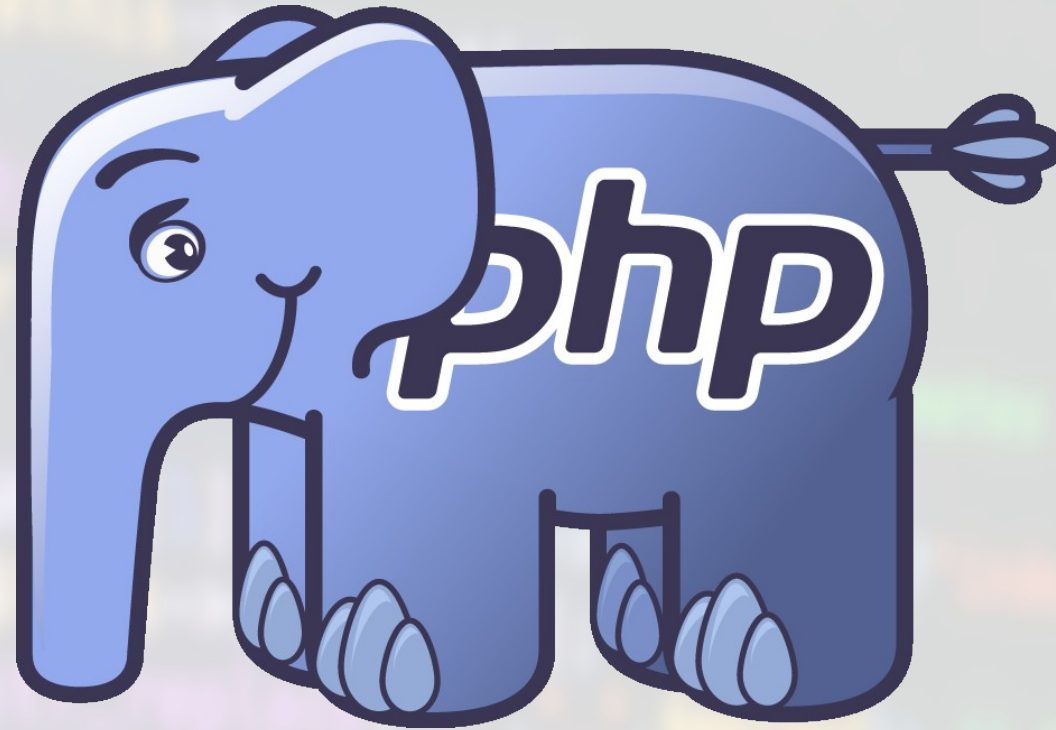

Résultat



A screenshot of a web browser window. The address bar shows a URL starting with 'http://'. The main content area displays the text 'Choisissez des fruits:' followed by a list of fruits with checkboxes: 'Fraise' (checked), 'Pomme' (checked), 'Poire' (unchecked), 'Banane' (checked and highlighted with a dashed border), and 'Cerise' (unchecked). Below the list is a button labeled 'Envoyer'. The status bar at the bottom shows 'Intranet local'.



A screenshot of a web browser window. The address bar shows a URL ending with '%5D=Banane'. The main content area displays three lines of text: 'Vous avez choisi Fraise', 'Vous avez choisi Pomme', and 'Vous avez choisi Banane'. The status bar at the bottom shows 'Intranet local'.



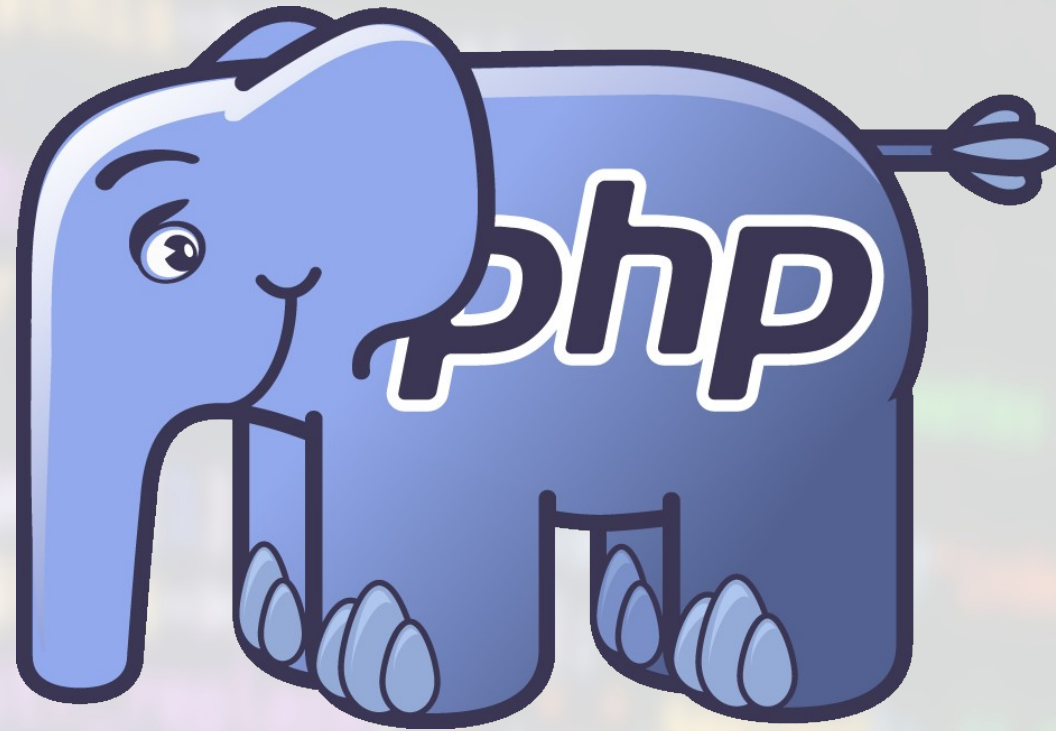
DONNÉES DE FORMULAIRES : VUE UTILISATEUR / VALEUR TRANSMISE

Formulaires contenant des champs « SELECT unique »

```
<!doctype html>
<html lang="fr">
<head>
  <title>Formulaire de saisie des fruits</title>
</head>
<body>
  <form action="valide3.php" method="get">
    Choisissez des fruits:&nbsp;
    <select name="sel">
      <option value="1">Fraise
      <option value="2">Pomme
      <option value="3">Poire
      <option value="4">Banane
      <option value="5">Cerise
    </select>
    <input type="submit" value="envoyer">
  </form>
</body>
</html>
```

Formulaires contenant des champs « CHECKBOX »

```
<!doctype html>
<html lang="fr">
  <head>
    <title>Formulaire de saisie des fruits</title>
  </head>
  <body>
    <form name="formu" action="valide3.php" method="get">
      Choisissez des fruits :
      <label><input type="checkbox" name="sel[]" value="1">Fraise</label>
      <label><input type="checkbox" name="sel[]" value="2">Pomme</label>
      <label><input type="checkbox" name="sel[]" value="3">Poire</label>
      <label><input type="checkbox" name="sel[]" value="4">Banane</label>
      <label><input type="checkbox" name="sel[]" value="5">Cerise</label>
      <input type="submit" value="Envoyer">
    </form>
  </body>
</html>
```



RÉFÉRENCES

Références

```
$a = 12 ;
```

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

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

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

```
$c = 84 ;
```

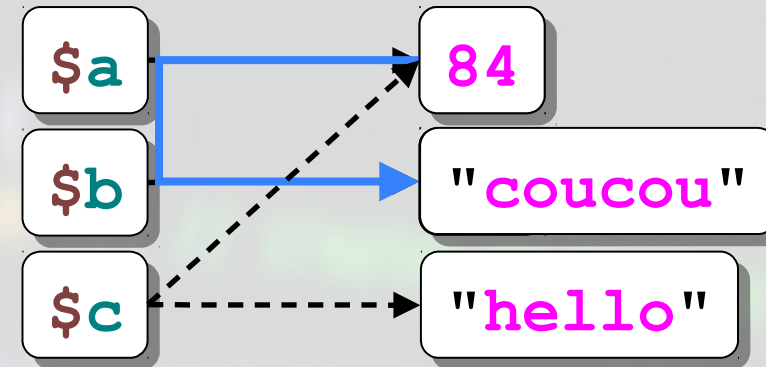
```
$a : 84
```

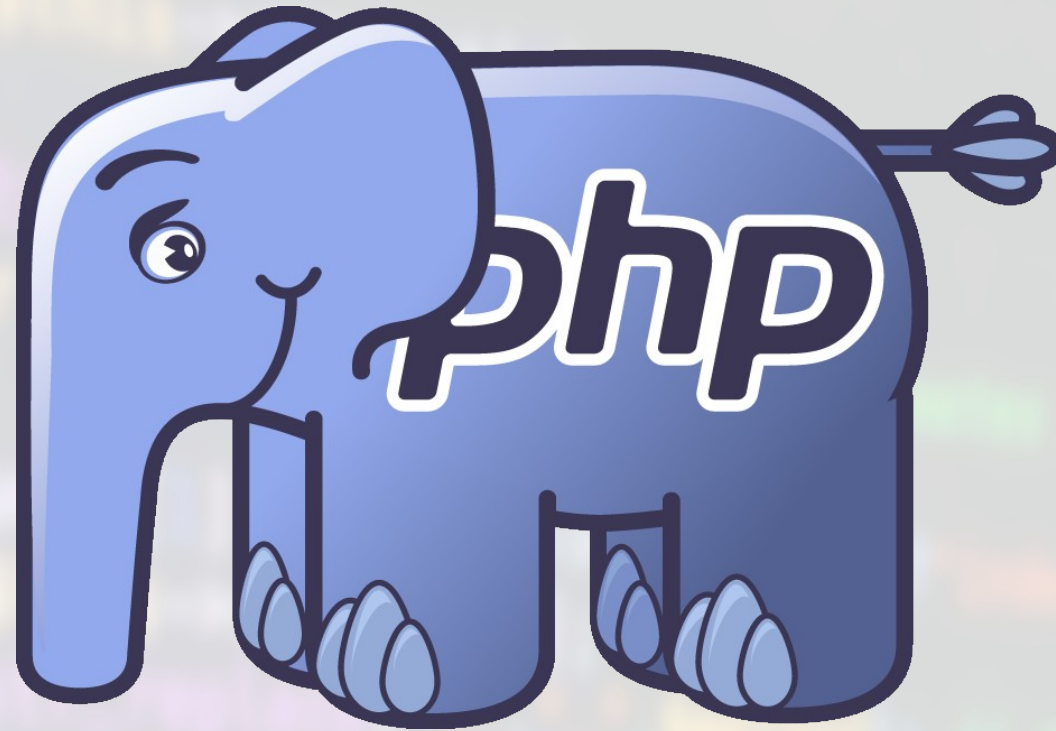
```
$b : coucou
```

```
$c : 84
```

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

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





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
```

Fonctions utilisateur PHP ≥ 4

- Valeur de retour

```
function moyenne ($a, $b)   
{ ... }
```

Typage faible de PHP :
Aucune information

- Arguments

```
function moyenne (  $a,  $b )  
{ ... }
```

Typage faible de PHP :
Aucune information

- Variables définies dans une fonction

Ⓟ Visibles et accessibles dans la fonction

Fonctions utilisateur PHP ≥ 5

- Valeur de retour

```
function moyenne($a, $b) : int  
{ ... }
```



Typage possible

- Arguments

```
function moyenne(int $a, int $b)  
{ ... }
```

- Fonctionnement identique pour les méthodes
- Possible de typer paramètre et valeur de retour

Typage des paramètres / retours PHP ≥ 5

Type	Description	Version PHP minimum
Nom de la Classe/interface	Le paramètre doit être une <i>instanceof</i> de la classe ou interface donnée.	PHP 5.0.0
<i>self</i>	Le paramètre doit être une <i>instanceof</i> de la même classe qui a défini la méthode. Ceci ne peut être utilisé que sur les méthodes des classes et des instances.	PHP 5.0.0
<i>array</i>	Le paramètre doit être un <i>array</i> .	PHP 5.1.0
<i>callable</i>	Le paramètre doit être un <i>callable</i> valide.	PHP 5.4.0
<i>bool</i>	Le paramètre doit être un <i>boolean</i> .	PHP 7.0.0
<i>float</i>	Le paramètre doit être un nombre flottant (<i>float</i>).	PHP 7.0.0
<i>int</i>	Le paramètre doit être un <i>integer</i> .	PHP 7.0.0
<i>string</i>	Le paramètre doit être une <i>string</i> .	PHP 7.0.0
<i>iterable</i>	Le paramètre doit être soit un array ou une <i>instanceof Traversable</i> .	PHP 7.1.0
<i>object</i>	Le paramètre doit être un <i>object</i> .	PHP 7.2.0

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
- `require` erreur bloquante, `include` erreur non bloquante
- `..._once` pour les inclusions uniques

include et require

Fichier mafunction.php

```
<?
function mafunction($arg)
{
    ...
}
```

Problème avec **include** :

- produit un **warning**
- le **script continue**

Problème avec **require** :

- produit un **fatal error**
- le **script s'arrête**

Fichier utilisation1.php

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

Fichier utilisation2.php

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

Fichier utilisation3.php

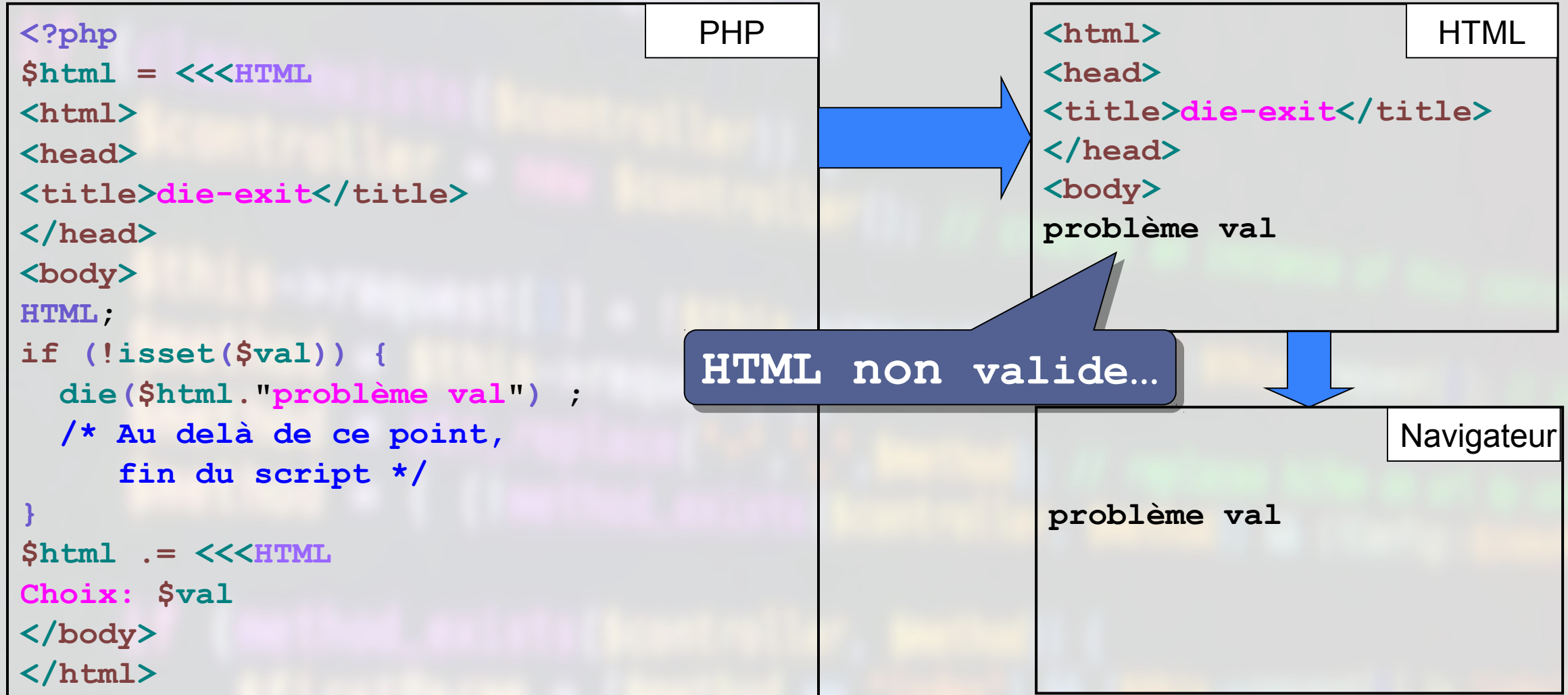
```
...
require("mafunction.php")
...
```

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

Nouveau niveau

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
<code>E_ERROR</code>
<code>E_WARNING</code>
<code>E_PARSE</code>
<code>E_NOTICE</code>
<code>E_CORE_ERROR</code>
<code>E_CORE_WARNING</code>
<code>E_COMPILE_ERROR</code>
<code>E_COMPILE_WARNING</code>
<code>E_USER_ERROR</code>
<code>E_USER_WARNING</code>
<code>E_USER_NOTICE</code>
<code>E_ALL</code>
<code>E_STRICT</code>

Debugage

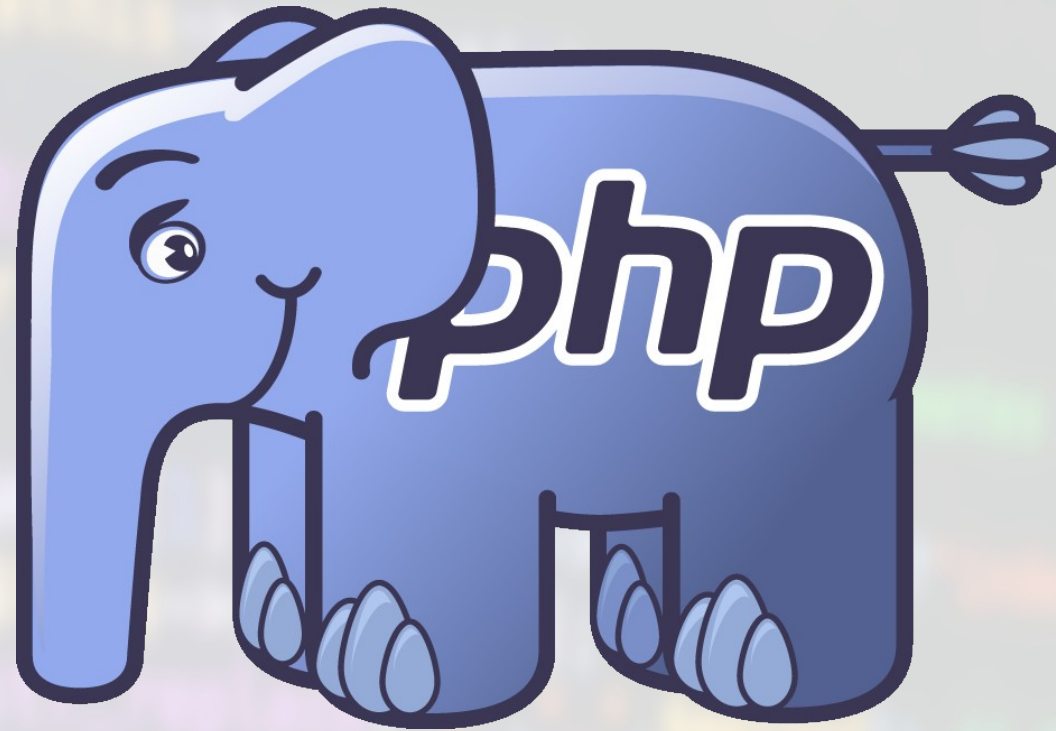
Opérateur de contrôle d'erreur en phase de développement

```
$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") Fichier absent  
    or die("Problème de lecture") ;
```

Problème de lecture



SAVOIR UTILISER LA DOCUMENTATION

Utiliser la documentation

Comment lire la définition d'une fonction (prototype)

Chaque fonction dans le manuel est documentée pour permettre une compréhension rapide. Savoir décoder le texte rendra votre apprentissage plus facile. Plutôt que de dépendre d'exemples prêts en copier/coller, il est plus utile de savoir lire la définition d'une fonction (prototype).

Source : <http://www.php.net>

Exemple : mb_ereg

The image shows a screenshot of the PHP documentation page for the `array_rand` function. The page is in French and includes a search bar at the top right. Several callout boxes are overlaid on the page, pointing to specific sections:

- Fonction courante**: Points to the function name `array_rand`.
- Version de PHP**: Points to the text "(PHP 4, PHP 5, PHP 7)".
- Descriptif**: Points to the description of the function.
- Valeur de retour**: Points to the "Valeurs de retour" section.
- Notes utiles**: Points to the "Notes utiles" section.
- Exemples**: Points to the "Exemples" section.
- À voir aussi**: Points to the "Voir aussi" section.
- Notes des utilisateurs**: Points to the "User Contributed Notes" section.
- Moteur de recherche**: Points to the search bar.
- Liste et description des paramètres**: Points to the "Liste de paramètres" section.
- Fonctions de la même extension**: Points to the sidebar list of functions.

The sidebar on the right lists various array functions, including `array_diff_uassoc`, `array_diff_ukey`, `array_key_exists`, `array_key_first`, `array_key_last`, `array_keys`, `array_map`, `array_merge_recursive`, `array_merge`, `array_multisort`, `array_pad`, `array_pop`, `array_product`, `array_push`, `array_rand`, `array_reduce`, `array_replace_recursive`, `array_replace`, `array_reverse`, `array_search`, `array_shift`, `array_slice`, `array_splice`, `array_unserialize`, `array_walk`, `array_walk_recursive`, `array_walk`, `array`, `arsort`, and `asort`.

Lecture du prototype d'une fonction

```
mb_ereg ( string $pattern ,  
         string $string
```

Nom de la f « Type » de \$pattern) : int

Les **types** et **pseudo-types** sont présents à des fins de
documenta

Paramètre optionnel Paramètres

« Type » de retour

boolean, integer, float, string, array, object,
resource, NULL, callable

types

mixed, number, void, array|object

pseudo-types

\$...

pseudo-variable