PHP : concepts de base

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

Plan

1. Variables
2. Quelques opérations sur les variables
3. Fonctions utiles pour les chaînes de caractères
4. Fonctions `ctype` et fonctions de conversion
5. Constantes

PHP

Une variable ?

- Un pointeur vers une zone mémoire
- Permettant de stocker une ou plusieurs données
- Pouvant avoir plusieurs valeurs différentes dans un programme

PHP

Une variable ?

- Un pointeur vers une zone mémoire
- Permettant de stocker une ou plusieurs données
- Pouvant avoir plusieurs valeurs différentes dans un programme

Rappel

- **PHP** est un langage faiblement typé.
- Le nom est sensible à la casse.

PHP

Conventions de nommage des variables en **PHP**

- Préfixer le nom de la variable par `$` suivi du nom de la variable dont le premier caractère est soit une lettre soit un underscore (_).
- Utiliser les lettres de l'alphabet (minuscules ou majuscules), les chiffres (0-9) et l'underscore (_).
- Utiliser **snake_case**.
- Choisir des noms courts et significatifs.
- Éviter les conflits avec les mots-clés de **PHP** tels que `if`, `for`, `class`...
- Utiliser des noms explicites pour les booléens comme `est_majeur`...
- Éviter les caractères spéciaux (Bien que **PHP** permette l'utilisation de caractères non **ASCII**, il est généralement préférable d'éviter les lettres accentuées et autres caractères spéciaux).

PHP

Exemple 1

```
$x = 5;
```

PHP

Exemple 1

```
$x = 5;
```

Exemple 2

```
$_x = 5;
```

PHP

Exemple 1

```
$x = 5;
```

Exemple 2

```
$_x = 5;
```

Exemple 3

```
$ma_variable = 5;
```

PHP

Exemple 1

```
$x = 5;
```

Exemple 2

```
$_x = 5;
```


Exemple 3

```
$ma_variable = 5;
```

Exemple 4

```
$_ma_variable = 5;
```

PHP

Quatre types scalaires (selon la valeur affectée)

- `int` (ou `integer`)
- `string`
- `boolean`
- `float` (ou `double`)

PHP

Quatre types scalaires (selon la valeur affectée)

- `int` (ou `integer`)
- `string`
- `boolean`
- `float` (ou `double`)

Types non-scalaires

- tableaux
- objets
- ressources (fichiers, connexion...)

PHP

Exemple avec `integer`

```php
$var = 5;
echo "$var ", gettype($var), "<br>";
/* affiche 5 integer */
```

PHP

Exemple avec `integer`

```php
$var = 5;
echo "$var ", gettype($var), "<br>";
/* affiche 5 integer */
```

Exemple avec `double`

```php
$var = 5.5;
echo "$var ", gettype($var), "<br>";
/* affiche 5.5 double */
```

PHP

Exemple avec `integer`

```php
$var = 5;
echo "$var ", gettype($var), "<br>";
/* affiche 5 integer */
```

Exemple avec `double`

```php
$var = 5.5;
echo "$var ", gettype($var), "<br>";
/* affiche 5.5 double */
```

Exemple avec `boolean`

```php
$var = TRUE;
echo "$var ", gettype($var), "<br>";
/* affiche 1 boolean */
```

PHP

`float` vs `double`

- Historiquement, `float` était utilisé pour représenter les nombres à virgule flottante en simple précision (32 bits), tandis que `double` était utilisé pour la double précision (64 bits).
- Cependant, avec les versions récentes de **PHP**, ils sont traités de la même manière et utilisent généralement une représentation en double précision.

PHP

`float` vs `double`

- Historiquement, `float` était utilisé pour représenter les nombres à virgule flottante en simple précision (32 bits), tandis que `double` était utilisé pour la double précision (64 bits).
- Cependant, avec les versions récentes de **PHP**, ils sont traités de la même manière et utilisent généralement une représentation en double précision.

`int` vs `integer`

En **PHP**, `integer` et `int` sont également des alias du même type de données. Ils représentent tous deux des nombres entiers.

Exemple avec `string`

```
$var = "hello";
echo "$var ", gettype($var), "<br>";
/* affiche hello string */
```

Exemple avec `string`

```
$var = "hello";
echo "$var ", gettype($var), "<br>";
/* affiche hello string */
```

Un seul caractère est aussi de type `string`

```
$var = "h";
echo "$var ", gettype($var), "<br>";
/* affiche h string */
```

Exemple avec `string`

```php
$var = "hello";
echo "$var ", gettype($var), "<br>";
/* affiche hello string */
```

Un seul caractère est aussi de type `string`

```php
$var = "h";
echo "$var ", gettype($var), "<br>";
/* affiche h string */
```


Pour afficher plusieurs informations sur une variable, on utilise `var_dump()`

```php
$var = "hello";
var_dump($var);
// affiche C:\wamp64\www\php-intro\index.php:17:
// string(5) "hello"
```

PHP

Comparaison `echo`, `var_dump` et `var_export`

- `echo 42;` → `42`
- `var_dump(42);` → `int(42)`
- `var_export(42);` → `42`

PHP

Comparaison `echo`, `var_dump` et `var_export`

- `echo 42;` → `42`
- `var_dump(42);` → `int(42)`
- `var_export(42);` → `42`

Rappel

`echo` ne fonctionne qu'avec des chaînes simples ou concaténées.

Remarques

- Pour vérifier un type, on peut utiliser la fonction `is_type($nom_variable)` : type à remplacer par le type que l'on cherche à vérifier
- Pour les booléens, il faut utiliser `is_bool($nom_variable)`.
- Ces fonctions retournent `1` si le type de la valeur de la variable passée en paramètre est vérifié, elles ne retournent rien sinon.

Remarques

- Pour vérifier un type, on peut utiliser la fonction `is_type($nom_variable)` : type à remplacer par le type que l'on cherche à vérifier
- Pour les booléens, il faut utiliser `is_bool($nom_variable)`.
- Ces fonctions retournent `1` si le type de la valeur de la variable passée en paramètre est vérifié, elles ne retournent rien sinon.

Exemple avec `is_string()`

```php
$var = "hello";
echo is_string($var) ,"<br>";
/* affiche 1 */

echo is_bool($var) ,"<br>";
/* n'affiche rien */
```

PHP

Explication

PHP convertit une valeur booléenne `TRUE` en la chaîne ”1” et `FALSE` en ”” (la chaîne vide), par conséquence

- `echo true` affiche `1`
- `echo false` n’affiche rien

PHP

Vérification de type avec `is_*`

Fonction	Vérifie si...	Exemple (retourne `true`)
`is_int()`	C'est un entier	`is_int(5)`
`is_float()`	C'est un flottant	`is_float(3.14)`
`is_string()`	C'est une chaîne	`is_string("abc")`
`is_bool()`	C'est un booléen	`is_bool(false)`
`is_array()`	C'est un tableau	`is_array([1, 2])`
`is_object()`	C'est un objet	`is_object($obj)`
`is_null()`	Valeur nulle	`is_null(null)`
`is_numeric()`	Nombre ou chaîne numérique	`is_numeric("123.45")`
`is_scalar()`	Scalaire (int, float, bool, string)	`is_scalar(true)`
`is_callable()`	Fonction exécutable	`is_callable('strlen')`
`is_iterable()`	Peut être parcouru avec `foreach`	`is_iterable([1,2])`

PHP

La fonction `is_numeric` permet de vérifier si le contenu d'une variable est numérique

```
echo(is_numeric(2));
/* affiche 1 */

echo(is_numeric(2.5));
/* affiche 1 */

echo(is_numeric("2"));
/* affiche 1 */

echo(is_numeric("-2.5"));
/* affiche 1 */

echo(is_numeric(true));
/* n'affiche rien */

echo(is_numeric("2a"));
/* n'affiche rien */

echo(is_numeric("a"));
/* n'affiche rien */
```

PHP

TP - Vérification de types en **PHP**

Objectif : pour chaque variable, indiquez le ou les `is_...()` qui retournent `true`.

```php
$a = 42;
$b = 3.14;
$c = "42";
$d = null;
$e = [1, 2, 3];
$f = function() { return 1; };
$g = true;
```

- 1. Quelles fonctions `is_...()` retournent `true` pour `$a` ?
- 2. Et pour `$c` ? et `$b` ?
- 3. Quelles variables sont considérées comme `is_numeric()` ?
- 4. Quelles fonctions tester pour savoir si une variable peut être utilisée dans un `foreach` ?
- 5. Bonus : quelles variables sont `is_callable()` ?

PHP

Corrigé

- $a → `is_int()`, `is_numeric()`, `is_scalar()`
- $b → `is_float()`, `is_numeric()`, `is_scalar()`
- $c → `is_string()`, `is_numeric()`, `is_scalar()`
- $d → `is_null()`
- $e → `is_array()`, `is_iterable()`
- $f → `is_callable()`, `is_object()`
- $g → `is_bool()`, `is_scalar()`

PHP

Utiliser une variable non-déclarée et non-initialisée $\Rightarrow$ Undefined variable

```
$x = $y + 1;

echo  $x;
```

PHP

La constante `NULL` peut-être utilisée pour créer une variable sans l'initialiser. Cette dernière sera de type `null`

```php
$var = NULL;

echo $var;
/* n'affiche rien */

echo gettype($var);
/* affiche NULL */
```

PHP

La fonction `isset` peut-être utilisée pour vérifier si une variable est déclarée et est différente de `NULL`

```php
$var = NULL;

echo isset($var);
/* n'affiche rien */

$var = 2;
echo isset($var);
/* affiche 1 */
```

PHP

La fonction `isset` peut-être utilisée pour vérifier si une variable est déclarée et est différente de `NULL`

```php
$var = NULL;

echo isset($var);
/* n'affiche rien */

$var = 2;
echo isset($var);
/* affiche 1 */
```


`empty()` **vs** `isset()` **vs** `is_null()`

	""	"wick"	NULL	false	true	0
empty()	true	false	true	true	false	true
isset()	true	true	false	true	true	true
is_null()	false	false	true	false	false	false

PHP

La fonction `unset` peut-être utilisée pour détruire la variable passée en paramètre

```php
$var = 2;
echo isset($var);
/* affiche 1 */

unset($var);
echo isset($var);
/* n'affiche rien */
```

PHP

Différence entre ”double quote” et ’simple quote’ avec `echo`

```
$var = 2;
echo "Bonjour. \nContenu de ma variable : $var"
/* affiche Bonjour.
Contenu de ma variable : 2 */

echo 'Bonjour. \nContenu de ma variable : $var'
/* affiche Bonjour. \nContenu de ma variable : $var */
```

PHP

Différence entre ”double quote” et ’simple quote’ avec `echo`

```php
$var = 2;
echo "Bonjour. \nContenu de ma variable : $var"
/* affiche Bonjour.
Contenu de ma variable : 2 */

echo 'Bonjour. \nContenu de ma variable : $var'
/* affiche Bonjour. \nContenu de ma variable : $var */
```


'' vs ""

- Les '' sont légèrement plus performantes que les "" parce que **PHP** n’a pas besoin de vérifier si, par exemple, des variables doivent être interpolées.
- Cependant, la différence de performance est généralement négligeable et ne pose pas de problème pour la plupart des applications.

PHP

Conclusion

- Utilisez les `""` lorsque vous avez besoin d'interpoler des variables ou d'inclure des séquences d'échappement.
- Utilisez les `''` pour des chaînes simples sans interpolation ni échappement complexe.

PHP

Conditions sur l'écriture `"$x"`

- `$x` doit être défini avant d'être utilisé, sinon un avertissement (**Notice : Undefined variable**) sera affiché.
- Si `$x` est suivi d'un caractère alphanumérique ou _, utilisez {} pour éviter toute ambiguïté. Exemple : `echo "contenu de x : $xvalue";`
- Valeur affichable de `$x`
  - Un nombre (entier ou flottant)
  - Une chaîne de caractères
  - Un booléen (`true` sera affiché comme 1, `false` comme une chaîne vide) `null` (affichera une chaîne vide)
  - Un tableau ou un objet déclenchera une erreur (**Array to string conversion**).

PHP

Opérateurs arithmétiques

- $=$: affectation
- $+$: addition
- $-$: soustraction
- $*$: multiplication
- $/$: division
- $\%$: reste de la division
- $**$: exponentiel

PHP

Quelques exemples avec l'addition

```php
$x = 1;
$y = 3;
$z = '8';
$t = "2";
$n = 2.5;
$m = 'a';

echo ($x + $y ."<br>"); /* 4 */
echo ($x + $z ."<br>"); /* 9 */
echo ($x + $t ."<br>"); /* 3 */
echo ($z + $t . "<br>"); /* 10 */
echo ($x + $y + $z ."<br>"); /* 12 */
echo ($z + $y + $x ."<br>"); /* 12 */
echo ($x + $n ."<br>"); /* 3.5 */
```

PHP

Quelques exemples avec l'addition

```
$x = 1;
$y = 3;
$z = '8';
$t = "2";
$n = 2.5;
$m = 'a';

echo ($x + $y ."<br>"); /* 4 */
echo ($x + $z ."<br>"); /* 9 */
echo ($x + $t ."<br>"); /* 3 */
echo ($z + $t . "<br>"); /* 10 */
echo ($x + $y + $z ."<br>"); /* 12 */
echo ($z + $y + $x ."<br>"); /* 12 */
echo ($x + $n ."<br>"); /* 3.5 */
```

Si la chaîne contient une valeur non-numérique, alors on aura une erreur

```
echo ($x + $m ."<br>");
```

PHP

Quelques raccourcis

- `$i++;` ≡ `$i = $i + 1;`
- `$i--;` ≡ `$i = $i - 1;`
- `$i += 2;` ≡ `$i = $i + 2;`
- `$i -= 3;` ≡ `$i = $i - 3;`
- `$i *= 2;` ≡ `$i = $i * 2;`
- `$i /= 3;` ≡ `$i = $i / 3;`
- `$i %= 5;` ≡ `$i = $i % 5;`

Exemple de post-incrémentation

```
$i = 2;
$j = $i++;
echo($i); /* affiche 3 */
echo($j); /* affiche 2 */
```

PHP

Exemple de post-incrémentation

```
$i = 2;
$j = $i++;
echo($i); /* affiche 3 */
echo($j); /* affiche 2 */
```


Exemple de pre-incrémentation

```
$i = 2;
$j = ++$i;
echo($i); /* affiche 3 */
echo($j); /* affiche 3 */
```

PHP

Exercice

Écrire un script **PHP** qui permet de permuter le contenu de deux variables.

PHP

Solution

```php
$a = 2;
$b = 0;
[$a, $b] = [$b, $a];
echo $a . " " . $b;
/* affiche 0 2 */
```

PHP

L'opérateur . pour concaténer deux chaînes de caractères

```
$string = "bon";
$string2 = "jour";
$str_concat = $string . $string2;
echo  $str_concat;
/* affiche bonjour */
```

PHP

L'opérateur . pour concaténer deux chaînes de caractères

```
$string = "bon";
$string2 = "jour";
$str_concat = $string . $string2;
echo  $str_concat;
/* affiche bonjour */
```

L'opérateur .= permet de faire concaténation + affectation

```
$string = "bon";
$string2 = "jour";
$string .= $string2;
echo  $string;
/* affiche bonjour */
```

PHP

La fonction `strcmp()` pour comparer deux chaînes de caractères

```php
$string = "bon";
$string2 = "jour";

echo strcmp($string, $string2);
/* affiche -1 */

echo strcmp($string2, $string);
/* affiche 1 */

echo strcmp($string, $string);
/* affiche 0 */
```

PHP

La fonction `strcmp()` pour comparer deux chaînes de caractères

```
$string = "bon";
$string2 = "jour";

echo strcmp($string, $string2);
/* affiche -1 */

echo strcmp($string2, $string);
/* affiche 1 */

echo strcmp($string, $string);
/* affiche 0 */
```


Pour comparer deux chaînes de caractères, on peut aussi utiliser l'opérateur ==

```
echo $string == $string;
/* affiche 1 */

echo $string == $string2;
/* n'affiche rien */
```

Fonctions utiles pour les chaînes de caractères

- `strlen()` : retourne la longueur de la chaîne de caractères passée en paramètre
- `strtoupper()` : convertit la chaîne de caractères passée en paramètre en majuscule
- `strtolower()` : convertit la chaîne de caractères passée en paramètre en minuscule
- `trim()` : supprime les espaces au début et à la fin (autres variantes : `ltrim()` et `rtrim()`)
- `substr()` : extrait une sous-chaîne de caractères
- `strpos($str, $str2)` : retourne la position de `str2` dans `str`. une chaîne, `false` sinon.
- `str_contains($str, $str2)` (**PHP 8**) : retourne `true` si `str` contient `str2`, `false` sinon.
- `substr_count($str, $str2)` : retourne le nombre d'occurrence de `str2` dans `str`.
- `str_split($str)` : transforme la chaîne de caractères passée en paramètre en tableau de sous-chaînes de caractères
- ...

PHP

Pour connaître la longueur d'une chaîne

```php
$str = "bonjour";
echo(strlen($str));
/* affiche 7 */
```

PHP

Pour connaître la longueur d'une chaîne

```
$str = "bonjour";
echo(strlen($str));
/* affiche 7 */
```

Pour supprimer les espaces au début et à la fin de la chaîne

```
$str = "  bon jour  ";
echo(strlen($str));
/* affiche 12 */

$sans_espace = trim($str);
echo(strlen($sans_espace));
/* affiche 8 */
```

Pour extraire une sous-chaîne à partir de l'indice 3 jusqu'à la fin

```php
$str = "bonjour";
echo(substr($str, 3));
/* affiche jour */
```

Pour extraire une sous-chaîne à partir de l'indice 3 jusqu'à la fin

```
$str = "bonjour";
echo(substr($str, 3));
/* affiche jour */
```

On peut aussi préciser le nombre de caractère à extraire

```
$str = "bonjour";
echo(substr($str, 3, 2));
/* affiche jo */
```

Pour extraire une sous-chaîne à partir de l'indice 3 jusqu'à la fin

```
$str = "bonjour";
echo(substr($str, 3));
/* affiche jour */
```

On peut aussi préciser le nombre de caractère à extraire

```
$str = "bonjour";
echo(substr($str, 3, 2));
/* affiche jo */
```


Pour extraire les trois derniers caractères, on utilise une valeur négative

```
$str = "bonjour";
echo(substr($str, -3)); /* eq substr($str, 4) avec 4 = length - 3 */
/* affiche our */
```

PHP

Pour déterminer l'indice d'une sous-chaîne dans une chaîne de caractères

```
$str = "Bonjour les bons jours";
echo(strpos($str, "bon"));
/* affiche 12 */
```

PHP

Pour déterminer l'indice d'une sous-chaîne dans une chaîne de caractères

```
$str = "Bonjour les bons jours";
echo(strpos($str, "bon"));
/* affiche 12 */
```

Pour une recherche insensible à la casse

```
$str = "Bonjour les bons jours";
echo(stripos($str, "bon"));
/* affiche 0 */
```

PHP

Pour déterminer l'indice d'une sous-chaîne dans une chaîne de caractères

```
$str = "Bonjour les bons jours";
echo(strpos($str, "bon"));
/* affiche 12 */
```

Pour une recherche insensible à la casse

```
$str = "Bonjour les bons jours";
echo(stripos($str, "bon"));
/* affiche 0 */
```

S'il n'y a aucune occurrence, elle ne retourne rien

```
$str = "Bonjour les bons jours";
echo(strpos($str, "soir"));
/* n'affiche rien */
```

PHP

Pour déterminer la dernière occurrence d'une sous-chaîne dans une chaîne de caractères

```php
$str = "Bonjour les bons jours";
echo(strrpos($str, "jour"));
/* affiche 17 */
```

PHP

Pour déterminer la dernière occurrence d'une sous-chaîne dans une chaîne de caractères

```
$str = "Bonjour les bons jours";
echo(strrpos($str, "jour"));
/* affiche 17 */
```

Pour déterminer la dernière occurrence d'une sous-chaîne dans une chaîne de caractères (insensible à la casse)

```
$str = "Bonjour les bons jours";
echo(strripos($str, "bon"));
/* affiche 12 */
```

PHP

Pour accéder à un caractère d'indice `i` dans une chaîne de caractères

```
// soit directement via l'indice
echo($str[i]);
```

PHP

Pour accéder à un caractère d'indice `i` dans une chaîne de caractères

```
// soit directement via l'indice
echo($str[i]);
```

Ou

```
/* soit en faisant l'extraction d'une sous chaine de caractères
   */
echo(substr($str, i, 1));
```

PHP

Pour accéder à un caractère d'indice `i` dans une chaîne de caractères

```
// soit directement via l'indice
echo($str[i]);
```

Ou

```
/* soit en faisant l'extraction d'une sous chaine de caractères
   */
echo(substr($str, i, 1));
```

Ou via un indice négatif

```
echo $str[-1];
echo $str[-5];
```

PHP

Étant données les deux chaînes de caractères suivantes

```
$ma_chaine = "Hello les holoulos";
$motif = "lo";
```

PHP

Étant données les deux chaînes de caractères suivantes

```
$ma_chaine = "Hello les holoulos";
$motif = "lo";
```

Exercice

En utilisant les fonctions sur les chaînes de caractères, écrire un script **PHP** qui permet de retourner la position de l'avant dernière occurrence de `$motif` dans `$ma_chaine`.

PHP

Correction

```
<?php
    $ma_chaine = "Hello les holoulos";
    $motif = 'lo';
    $last_position = strrpos($str, $motif);
    $without_last = substr($str, 0, $last_position);
    echo strrpos($without_last, $motif);
?>
```

PHP

Étant données les deux chaînes de caractères suivantes

```
$ma_chaine = "Hello les holoulos";
$motif = "lo";
```

PHP

Étant données les deux chaînes de caractères suivantes

```
$ma_chaine = "Hello les holoulos";
$motif = "lo";
```

Exercice

En utilisant les fonctions sur les chaînes de caractères, écrire un script **PHP** qui permet de supprimer l'avant dernière occurrence de `$motif` dans `$ma_chaine`.

PHP

Correction

```
<?php
  $ma_chaine = "Hello les holoulos";
  $motif = 'lo';
  $sans_dernier_motif = substr($ma_chaine, 0, strrpos($ma_chaine,
     $motif));

  $pos_avant_dernier =  strrpos($sans_dernier_motif, $motif);
  $avant = substr($ma_chaine, 0, $pos_avant_dernier);
  $apres =  substr($ma_chaine, $pos_avant_dernier + strlen($motif));
  $ma_chaine = $avant . $apres;
  echo  $ma_chaine;
?>
```

PHP

La fonction `nl2br` permet de remplacer `\n` et `\r` par la balise `<br>`

```
$bonjour = "bonjour \n john \n wick \n\r ou \r\n
   Keanu Reeves" ;

echo nl2br($bonjour);

/* le code source généré
bonjour <br />
 john <br />
 wick <br />
 ou <br />
 Keanu Reeves
 */
```

PHP

Pour convertir une chaîne en entier (première solution avec le cast)

```php
$a = '4';
$b = (int)$a;
echo $b . " " . gettype($b);
/* affiche 4 integer */
```

PHP

Pour convertir une chaîne en entier (première solution avec le cast)

```
$a = '4';
$b = (int)$a;
echo $b . " " . gettype($b);
/* affiche 4 integer */
```

Une deuxième solution avec la fonction `intval()`

```
$a = '4';
$b = intval($a);
echo $b . " " . gettype($b);
/* affiche 4 integer */
```

Pour convertir une chaîne en double (première solution avec le cast)

```php
$a = '4.5';
$b = (float)$a;
echo $b . " " . gettype($b);
/* affiche 4.5 double */
```

Pour convertir une chaîne en double (première solution avec le cast)

```php
$a = '4.5';
$b = (float)$a;
echo $b . " " . gettype($b);
/* affiche 4.5 double */
```

Pour faire le cast, on peut aussi utiliser `(double)`

```php
$a = '4.5';
$b = (double)$a;
echo $b . " " . gettype($b);
/* affiche 4.5 double */
```

Pour convertir une chaîne en double (première solution avec le cast)

```
$a = '4.5';
$b = (float)$a;
echo $b . " " . gettype($b);
/* affiche 4.5 double */
```

Pour faire le cast, on peut aussi utiliser `(double)`

```
$a = '4.5';
$b = (double)$a;
echo $b . " " . gettype($b);
/* affiche 4.5 double */
```

Une deuxième solution avec la fonction `floatval()` **ou** `doubleval()`

```
$a = '4.5';
$b = floatval($a);
echo $b . " " . gettype($b);
/* affiche 4 double */
```

PHP

La fonction `eval` exécute une chaîne de caractère passée en paramètre comme un script PHP

```php
$str = "echo 2 * 3;";
eval($str);
/* affiche 6 */
```

PHP

Fonctions ctype_ les plus courantes

Fonction	Description	Exemple (retourne true)
ctype_alnum()	Lettres ou chiffres	"abc123"
ctype_alpha()	Lettres uniquement	"Hello"
ctype_digit()	Chiffres uniquement	"123456"
ctype_space()	Espaces blancs (\n, \t, espace...)	"\t \n"
ctype_upper()	Lettres majuscules uniquement	"PHP"
ctype_lower()	Lettres minuscules uniquement	"php"
ctype_xdigit()	Caractères hexadécimaux	"1A3f"
ctype_cntrl()	Caractères de contrôle	"\n"
ctype_punct()	Ponctuation uniquement	"!?@#"
ctype_print()	Caractères imprimables (espace inclus)	"Hello 123!"
ctype_graph()	Caractères imprimables (sans espace)	"Hello!"

PHP

Remarques

- Ces fonctions ne retournent `true` que si **tous les caractères** respectent la condition.
- Elles ne retournent `true` que si la chaîne est non vide.

```
ctype_digit("123");    // true
ctype_digit("123abc"); // false
ctype_alpha("");       // false
```

- Très utile pour valider des formats simples sans **REGEX**.

PHP

Fonctions de conversion de type en **PHP**

- `intval(mixte $val)` : convertit en entier
- `floatval(mixte $val)` : convertit en nombre à virgule
- `strval(mixte $val)` : convertit en chaîne
- `boolval(mixte $val)` : convertit en booléen
- `settype($var, "type")` : modifie le type d'une variable

PHP

Fonctions de conversion de type en **PHP**

- `intval(mixte $val)` : convertit en entier
- `floatval(mixte $val)` : convertit en nombre à virgule
- `strval(mixte $val)` : convertit en chaîne
- `boolval(mixte $val)` : convertit en booléen
- `settype($var, "type")` : modifie le type d'une variable


```php
intval("123abc");    // 123
floatval("12.34px"); // 12.34
boolval(0);          // false
strval(true);        // "1"
```

PHP

Casting explicite en **PHP**

PHP permet aussi le cast direct : `(int)`, `(string)`, `(bool)`, `(float)`.

PHP

Casting explicite en **PHP**

PHP permet aussi le cast direct : `(int)`, `(string)`, `(bool)`, `(float)`.

Exemple

```php
$val = "42.7px";
$entier = (int) $val;      // 42
$flottant = (float) $val; // 42.7
$texte = (string) true;    // "1"
$bool = (bool) 0;          // false
```

PHP

Résumé

- Les fonctions `ctype_` servent à vérifier le contenu d'une chaîne caractère par caractère.
- Les fonctions de conversion servent à changer le type d'une variable.
- Toujours vérifier le besoin : valider une chaîne ? ou convertir une donnée ?

PHP

is_x()

- Vérifie le **type** ou une propriété simple
- Accepte tout type de variable
- Exemples
  - `is_numeric("123")` ⇒ `true`
  - `is_numeric("123.45")` ⇒ `true`

ctype_x()

- Vérifie le **contenu** d'une chaîne
- Nécessite un `string`
- Exemples
  - `ctype_digit("123")` ⇒ `true`
  - `ctype_digit("123.45")` ⇒ `false`

PHP

is_x()

- Vérifie le **type** ou une propriété simple
- Accepte tout type de variable
- Exemples
  - is_numeric("123") ⇒ true
  - is_numeric("123.45") ⇒ true

ctype_x()

- Vérifie le **contenu** d'une chaîne
- Nécessite un string
- Exemples
  - ctype_digit("123") ⇒ true
  - ctype_digit("123.45") ⇒ false

Résumé

- is_x() : vérifie le type (numérique, tableau...)
- ctype_x() : vérifie si une chaîne contient uniquement certains caractères

PHP

Constante

élément qui ne peut changer de valeur

PHP

Constante

élément qui ne peut changer de valeur

Remarques

- Le nom d'une constante ne doit pas commencer par $.
- On utilise _ pour séparer les mots dans le nom d'une constante.
- Par convention, les constantes sont toujours écrites en majuscules.
- On peut déclarer une constante avec la fonction `define()` ou avec le mot-clé `const`.

PHP

Remarques

- La fonction `define(nom_constante, valeur, insensibilité à la casse)` permet de définir une constante.
- Le dernier paramètre est facultatif et sa valeur par défaut est `false`.

PHP

Remarques

- La fonction `define(nom_constante, valeur, insensibilité à la casse)` permet de définir une constante.
- Le dernier paramètre est facultatif et sa valeur par défaut est `false`.

Exemple

```
define('PI', 3.1415);
```

PHP

Remarques

- La fonction `define(nom_constante, valeur, insensibilité à la casse)` permet de définir une constante.
- Le dernier paramètre est facultatif et sa valeur par défaut est `false`.

Exemple

```
define('PI', 3.1415);
```

L'instruction suivante ne peut être acceptée

```
PI = 5;
```

PHP

On peut aussi déclarer une constante avec `const`

```
const PI = 3.1415;
```

PHP

On peut aussi déclarer une constante avec `const`

```
const PI = 3.1415;
```

L'instruction suivante ne peut être acceptée

```
PI = 5;
```

PHP

Remarques

- Les constantes définies avec le mot-clé `const` doivent être déclarées dans le contexte global ou à l'intérieur des classes.
- `const` ne doit pas être utilisé à l'intérieur de fonctions, boucles, instructions `if` ou blocs `try/catch`.
- `define()` peut être utilisé de manière conditionnelle (à l'intérieur de blocs de code conditionnels).
- `define()` est utilisée pour déclarer des constantes globales et ne peut pas être utilisée à l'intérieur d'une classe.
- `const` est plus rapide, donc il est souvent préféré pour des déclarations plus modernes.