

Chapitre 1 : Les variables

Les variables commencent presque systématiquement par un dollar. (sauf constantes).

le nom des variable est sensible à la case, mais pas celui des fonctions, classes et mots-clés

1.1 Déclaration et initialisation

```
1 | $nom = 'antoine';
```

```
1 | isset($var); // verifier si la variable existe et n'est pas nulle
2 | empty($var); // Vrai si Existe pas ou variable de sortie est fausse
3 |
4 | unset($var); // Supprime une variable
```

1.2 Types de variable

On peut récupérer le type d'une variable avec la commande :

```
1 | gettype();
```

Cette fonction peut retourner : "boolean", "integer", "double", "string", "array", "object", "resource" (ressource binaire (donnée générique)), "NULL".

Il est possible de vérifier si une variable est d'un type via les fonctions is_

```
1 | is_bool($var);      //booléen
2 | is_integer($var);   //entier
3 | is_double($var);    //réel
4 | is_numeric($var);   //entier ou réel
5 | is_string($var);    //chaîne de caractère
6 | is_object($var);    //objet
7 | is_resource($var);  //ressource (fichier générique)
8 | is_null($var);      //null
9 | is_nan($var);       //Not a Number
```

note : array est un type à part contrairement au js

1.2.1 Variables Numériques

Contrairement au Js, le php différencie les entiers et les réels.

Il existe aussi NaN.

la division donne TOUJOURS un réel. même si les 2 nombres sont des entiers

1.2.2 NULL

PHP ne possède pas de valeur **undefined** mais elle possède une valeur NULL. Cette valeur représente donc les variables sans valeurs

1.2.3 Booléen

echo TRUE retourne 1 ET echo FALSE ne **produit rien**

1.2.4 String

```
1 | $var = ' ' // littéral de la chaîne de caractère
2 | $var = "" // Chaîne non-littérale (variables remplacées par leurs valeurs)
```

Il existe aussi Here-doc (message Multiline avec mise en page spécifique) et now-doc (idem à here doc mais les variances ne sont pas remplacés)

Concaténation

```
1 | $txt1.$txt2;
```

Accès d'un caractère dans une chaîne de caractères

```
1 | $txt[2];
2 | $txt{3};
```

Affichage formaté

```
1 | printf(format,$var); //comme en c
2 | $reponse = sprintf(format,...);
```

Longueur d'une chaîne

```
1 | strlen($s);
```

1.2.5 Constantes

Les constantes sont des variables qui sont immuables.

Elles sont appelées **sans \$** devant le nom de la variable et les constantes ne sont **pas remplacées** dans le cas de chaînes de caractères non-littérales contrairement aux variables

```
1 | define('MACONSTANTE','Contenu de ma Constante');
2 | echo MACONSTANTE;
```

Les constantes prédéfinies

```
1 | __LINE__ // numéro de la ligne dans le fichier
2 | __FILE__ // fichier courant
3 | __DIR__ // Répertoire courant
```

1.3 Conversion de Types

Attention: Le php, comme le Javascript sont des langages qui pratiquent la conversion implicite. Il est important de faire attention à ces conversions

1.3.1 Caster comme en Java

```
1 $varCaste = (int) $varNonCaste; // si la variable est un réel, on prends sa
  partie entière
2 $varCaste = (integer) $varNonCaste; // si la variable est un réel, on prends
  sa partie entière
3 interval($varNonCaste); // si la variable est un réel, on prends sa partie
  entière
4 //... même principe pour les autres types
```

1.3.2 Settype

```
1 settype($x, 'int');
```

1.3 Fonctions

tentative de transformation de texte en nombre (généralement 0)

Il existe aussi une valeur NULL (undefined n'existe pas)

On peut aussi effectuer des affectations multiples en php

```
1 $a = $b = 0;
```

variables falsy

- FALSE
- 0
- ""
- "0"
- NULL
- []

Attention, en javascript "0" est converti en true

Les Alias

```
1 $b = &$a;
2 $a = 'texte'; //modifie aussi $b
```

Le typage

En php, comme en javascript, le typage est implicite et les conversions sont implicites. Les différents types sont : boolean, integer, double, string, array, object, resource (ressources génériques au format binaire, NULL, et NAN

On peut tester si une variables appartient à un type via les fonctions `is_...($var)` (ex : `is_boolean(x)`)

On peut obtenir le type d'une variable avec `gettype()`

La fonction `var_dump($variable)` est utile pour connaître plus d'informations sur les variables (DEBOGGAGE)

Typecast

On peut forcer la conversion comme en java ou en c via un typecast

type	/	/	/	/
integer	(int)expr	(integer)expr	intval(expr)	
double	(float)expr	(double)expr	(real)expr	floatval(expr)
string	(string)expr	strval(expr)		
boolean	(bool)expr	(boolean)expr		
array	array(expr)			
object	(object)expr			

Pour la conversion vers des nombres entiers,

- Si une variable est un réel on ne prends que sa partie entière
- Si une variable et un string alors on ne prends que le préfixe valide le plus long

On peut aussi utiliser :

```
1 | settype($var, 'int');
```

php est différencie les entiers et les réels

Portée des variables

les variables globales ne sont pas forcément accessibles depuis une fonction (il faut mettre global devant pour les rendre accessibles.

```
1 | $intro = 'Le résultat est ';
2 | $outro= '<br/>';
3 | function afficheDouble($v) {
4 |     global $intro, $outro;
5 |     ...
6 | }
```

Si une variable globale est passée par référence en argument de la fonction(cfr chapitre fonctions/références de variables), le résultat sera le même

Conversions implicites

exemples

```
1 echo '13' + '17 vaches'; // affiche 30 !
2 $txt1 = '12';
3 $txt2 = '+12';
4 $txt3 = '12pommes';
5 echo ($txt1 == $txt2); // affiche 1 (= vrai) !
6 echo ($txt1 === $txt2); // affiche <rien> (= faux) !
7 echo ($txt1 == $txt3); // affiche <rien> (= faux) !
8 echo (12 == $txt3); //affiche 1 (= vrai) !
```

Strings

1. chaîne littérale (' ... ' -> non interprété)
 - ' et \
2. chaîne non littérale (" ... " -> interprété)
 - " \n \t \$
 - ou la notation octale ou hexa (ex : \123 ou \xA5)
3. here document (écriture sur plusieurs lignes)(remplace les chaînes)
 - [lien documentation](#)

```
1 $msg = <<<EXTRAIT
2 Voici le texte "contenu" dans cette chaîne
3 de caractères sans qu'il soit
4 nécessaire d'échapper les " et les '
5 EXTRAIT;
```

4. nowdoc (comme heredoc sauf non interprété)
 - [lien documentation](#)

```
1 $msg = <<<'EXTRAIT'
2 Voici le texte "contenu" dans cette chaîne
3 de caractères sans qu'il soit
4 nécessaire d'échapper les " et les '
5 EXTRAIT;
```

concaténation de strings

Attention, en JS, la concaténation s'écrit +

```
1 $prenom = 'antoine';
2 $nom = 'Lambert';
3
4 echo $prenom. ' '. $nom;
5 $user = $prenom;
6 $user .= ' '; // le .= est le signe de la concaténation
7 $user .= $nom;
```

\n pour saut de ligne :

attention, les guillemets simples n'interprètent pas les variables et caractères spéciaux contrairement aux guillemets doubles.

en cas de calcul à afficher : entourer les calculs de ()

Longueur d'une chaîne de caractères

```
1 | strlen($s)
```

Définition de Constantes

```
1 | define('MACONSTANTE', 'contenu de la constante');  
2 |  
3 | echo MACONSTANTE // Les constantes s'utilisent sans $
```

clean code : les constantes sont définies en MAJUSCULES

Remarques

- Une fois définie, une constante est visible partout.
- Si on utilise `MACONSTANTE` sans la définir d'abord, PHP suppose qu'elle contient la chaîne "MACONSTANTE".

```
1 | __LINE__ \\ numéro de la ligne courante dans le fichier  
2 | __FILE__ \\ fichier courant (__DIR__ pour son répertoire)
```

/	Constantes	Variables
Utilisation	Sans \$	Avec \$
Déclaration	via define	Lors de la première affectation
Visibilité	Partout	Locale ou globale
Valeur	Doit être Scalaire	Quelconque
Modification	Impossible	possible, ainsi qu'unset

Récupérer le type d'une variable

```
1 | gettype($var)
```

`var_export` qui donne une écriture de la valeur correspondant à un littéral PHP
`var_dump` donne non seulement la valeur mais également le type de celle-ci

Variables superglobales

Variables superglobales(= prédéfinies et accessibles partout !)

- **\$GLOBALS** : toutes les variables définies dans le contexte global(dont les variables globales déclarées : **\$_GLOBALS['mavar']**)
- **\$_SERVER** : informations provenant du serveur web
- **\$_ENV** : informations sur l'environnement et l'OS
- **\$_GET, \$_POST** : variables reçues dans la requête http
- **\$_FILES** : fichiers attachés à la requête http
- **\$_COOKIE** : variables passées sous la forme de cookies http
- **\$_REQUEST** : l'ensemble des informations attachées à la requête http (redondant avec *\$_GET, \$_POST, \$_COOKIE* et *\$_FILES*).
- **\$_SESSION** : variables de session

ces variables prennent la forme de tableaux associatifs

Adresses relatives ou absolues

En HTML et en PHP, les adresses peuvent être relatives (`../inc/file.php` : fichier contenu dans le dossier frère)

Par contre, les adresses absolues sont différentes en html et en php : pour les adresses en html, le `"/"` représente la racine du projet web, tandis que en php, ce même `"/"` représente la racine du Système.

Envoi par GET et POST

GET est utilisé pour envoyer des données par l'url, tandis que **POST** se chargera des données d'un formulaire.

En php, on récupère les données issues d'un **GET** en utilisant **\$_GET**

exemple d'url

```
1 | commande.php?article=montre
```

exemple de code avec la méthode GET

```
1 | <?php
2 | if (isset($_GET) && isset($_GET['article']))
3 |     echo $_GET['article'];
4 | else
5 |     echo 'aucune commande';
6 | ?>
```

Pour rappel, la commande Javascript pour se rendre vers une autre url est `location.href=nouvelleURL;`