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

```
$nom = 'antoine';
```

opération numérique

```
echo (2+3)/2;
```

> PHP possède une valeur nan (checker via is_nan($var))

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

> tentative de transformation de texte en nombre (généralement 0)

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

> isset($var) vrai si la variable existe et ne vaut pas null.
> empty($var) vrai si $v n'**existe pas** ou **vaut falsy**

On peut aussi effectuer des affectations multiples en php

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

variables falsy

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

> Attention, en javascript "0" est converti en true

Les Alias

```
$b = &$a;
$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 :

```
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.

```
$intro = 'Le résultat est ';
$outro= '.<br/>';
function afficheDouble($v) {
global $intro, $outro;
...
}
```

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

```
echo '13' + '17 vaches';    // affiche 30 !
$txt1 = '12';
$txt2 = '+12';
$txt3 = '12pommes';
echo ($txt1 == $txt2);  // affiche 1 (= vrai) !
echo ($txt1 === $txt2);     // affiche <rien> (= faux) !
echo ($txt1 == $txt3);  // affiche <rien> (= faux) !
echo (12 == $txt3);     //affiche 1 (= vrai) !
```

Strings

1. chaine litteral (' ... ' -> non interpreté)
    - ' et \
2. chaine non litteral ("... " -> interpreté)
    - " \ \n \t $
    - ou la notation octale ou hexa (ex : \123 ou \xA5)
3. here document (écriture sur plusieurs lignes)(remplace les chaines)
    - lien documentation

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

4. nowdoc (comme heredoc sauf non interpreté)
    - lien documentation

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

concaténation de strings

> Attention, en JS, la concaténation s'écrit +

```
$prenom = 'antoine';
$nom = 'Lambert';

echo $prenom. '  '.$nom;
$user = $prenom;
$user .= ' '; // le .= est le signe de la concaténation
$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 a afficher : entourer les calculs de ()

Longueur d'une chaîne de caractères

```
strlen($s)
```

Définition de Constantes

```
define('MACONSTANTE', 'contenu de la constante');

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 utiliseMACONSTANTEsans la définir d'abord, PHP suppose qu'elle contient la chaîne"MACONSTANTE".

```
__LINE__ \\ numéro de la ligne courante dans le fichier
__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

```
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 êtres 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

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

exemple de code avec la méthode GET

```
<?php
if (isset($_GET) && isset($_GET['article']))
echo $_GET['article'];
else
echo 'aucune commande';
?>
```

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