II) Variables et constantes

PHP - Initiation

Variables et constantes

II.1) Déclaration

La déclaration d'une variable se fait avec :

- Un **nom** commençant impérativement par **$** et composé de caractères alphanumériques et **_** ou **-**.
- Une **valeur** (dont le type sera déterminé par la syntaxe).
- L'**association** entre le nom et la valeur est faite par le signe **=**.

Syntaxe

***$ma_variable = valeur*;**

PHP - Initiation

Variables et constantes

Exemples

```
$var_1 = 4.56;                  // valeur numérique de type flottant
$var_2 = 12;                    // valeur numérique de type entier
$var_3 = 'Pascal';              // valeur de type chaine
$var_4 = true;                  // valeur de type booléen
```

Une **valeur de variable** peut être affectée à une autre variable :

```
$taxe = $tva;
```

$taxe contient la valeur qu'avait ***$tva*** au moment de l'affectation.

Une **référence de variable** peut être affectée à une autre variable :

```
$taxe = &$tva;
```

$taxe contiendra toujours la valeur de ***$tva*** même si celle-ci est modifiée.

PHP - Initiation

Variables et constantes

II.2) Les types

PHP supporte les types suivants :
boolean, **integer**, **float**, **string**, **array**, **object**, **resource** et **NULL**.

C'est **PHP qui décide** à l'exécution, en fonction de son contenu, quel type est le plus indiqué pour la variable.

Par exemple, dans un contexte de test, une valeur sera évaluée en *booléen* en appliquant les règles de conversion qui sont :

- toute valeur numérique non nulle → *true*, 0 → *false*
- toute chaîne de caractère non vide → *true*, sinon → *false*
- la valeur *NULL* → *false*
- toutes les autres valeurs → *true*

Ce comportement est celui des langages faiblement typés.

PHP - Initiation

Variables et constantes

PHP n'impose donc pas de déclaration explicite des variables.

Il est cependant possible de forcer le **transtypage** en "castant" les variables vers le type choisi. Les "casts" autorisés sont :

- **(int)** ou **(integer)** : conversion en entier
- **(bool)** ou **(boolean)** : conversion en booléen
- **(double)**, **(float)** : conversion en flottant double précision
- **(string)** : conversion en chaîne de caractères
- **(array)** : conversion en tableau
- **(object)** : conversion en objet
- **(unset)** : conversion en NULL

Lors de la conversion entre un nombre décimal et un entier, le nombre sera arrondi à la valeur inférieure s'il est positif, et supérieure s'il est négatif (conversion dite "vers zéro").

PHP - Initiation

Variables et constantes

Syntaxe

***$var* = (type)$*var*;**

Exemple

```
$var_1 = 4.56;                  // valeur numérique de type flottant
var_dump($var_1);               // affiche : float(4.56)

echo '<hr>';

$var_1 = (string)$var_1;        // valeur désormais transformée pour être de type string
var_dump($var_1);               // affiche : string(4)"4.56"
```

La fonction **var_dump()** permet d'afficher les informations d'une variable, y compris son type et sa valeur. Elle fonctionne également sur les tableaux et les objets et est **très utile pour "débugger"**.

PHP - Initiation

Variables et constantes

II.3) Existence et destruction

En mode d'erreur strict (**error_reporting = E_STRICT** dans **php.ini**) l'utilisation d'une variable non déclarée enverra une alerte.

La fonction **isset()** permet de vérifier si la variable passée en argument existe et est différente de NULL.

La fonction **unset()** détruit l'ensemble des variables passées en argument. (Une variable peut être masquée par affectation à NULL : ***$ma_var*** **= NULL;**)

Remarque
Il existe un différence entre une variable dont la valeur est NULL et une variable non déclarée. Cela concerne la quantité de **mémoire utilisée**.
Il sera préférable d'utiliser **unset()** dans le cas ou la variable ne sera plus utilisée par la suite car à NULL elle conserve son espace mémoire

PHP - Initiation

Variables et constantes

II.4) Les constantes

Il est possible de déclarer des constantes en PHP. Ce sont tout simplement des valeurs que l'on va enregistrer en mémoire et qui ne vont pas changer au cours du temps. (Une constante à valeur égale d'une variable prendra moins de place en mémoire).

Syntaxe

define(*chaîne*, *value*); permet d'affecter *valeur* à la constante *chaîne*.
defined(*chaîne*); permet de vérifier si la constante *chaîne* est définie.

Exemple

```
define('MA_CONSTANTE',  '500');
var_dump(defined('MA_CONSTANTE'));          // affiche : bool(true)
```

Remarque

Par convention, il est conseillé d'utiliser des noms en majuscules.