

PHP : Variables, constantes, types

L2 MPCIE - UE Développement Web

David Lesaint

david.lesaint@univ-angers.fr



Janvier 2019

Les variables

Une variable est le conteneur d'une valeur d'un des types utilisés par PHP :

- entiers
- flottants
- chaînes de caractères
- booléens
- tableaux
- objets
- ressources
- nul

Identifiants de variables

- Un identifiant de variable commence toujours par le caractère \$ suivi du nom de la variable.
- Le nom de la variable commence par une lettre de l'alphabet (minuscule ou majuscule) ou bien par le caractère _.
- Les caractères suivants sont des lettres, des chiffres ou _.
- Les identifiants de variables sont sensibles à la casse.

Identifiants corrects

\$maVar

\$_maVar

\$MaVar2

\$_123

Identifiants incorrects

maVar

\$1maVar

\$+maVar

\$maVar*

Déclaration des variables

Déclaration et initialisation

- Il n'est pas obligatoire de déclarer les variables en début de script : on peut les déclarer juste avant leur utilisation.
- Il n'est pas obligatoire d'initialiser les variables mais une variable non initialisée n'a pas de type précis.

non-initialisation.php

```
1 $var;  
2 echo $var; // PHP Notice: Undefined variable: var in /Users/davidlesaint/...  
3 echo gettype($var); // NULL
```

Affectation de variable par valeur et par référence

L'affectation de variable

- Consiste à donner une valeur à une variable.
- Détermine le type de la variable (*typage dynamique*).
- S'effectue par valeur ou par référence.

Affectation par valeur

- Syntaxe : `$var=exp` où `exp` dénote une expression PHP valide : littéral, variable, appel de fonction ...
- Sémantique : `$var` prend la valeur de `exp`.

Affectation par référence avec & (esperluette)

- Syntaxe : `$var1=&$var2` où `$var2` dénote une variable.
- Sémantique : `$var1` est un alias de `$var2`.

Affectation de variable par valeur et par référence

Ré-affectation

On peut réaffecter une même variable au cours d'un script.

- Si `$var=exp` et `exp` est une variable, toute modification ultérieure de `exp` n'aura aucune incidence sur `$var`.
- Si `$var1=&$var2`, toute modification de l'une des deux variables sera répercutée sur l'autre.

exemple2-1.php

L'affectation d'un objet ou d'une ressource à une variable se fait systématiquement par référence. Il est possible de cloner un objet.

Les variables dynamiques

Variables dont on détermine le nom en cours de script !

- Syntaxe : `$$var` où `$var` est une variable dont la valeur est une chaîne de caractères correspondant à un nom de variable bien formé.
- Accès :
 - en dehors d'une chaîne (eg, affectation) : `$$var=exp;`
 - dans une chaîne : `${$var}`

variables-dynamiques.php

```
1 $php="PHP";
2 $$php="Open Source";
3 echo $$php, "<br/>"; echo $PHP, "<br/>";
4 echo "$php est ${$php}", "<br/>";
5 $PHP="un langage";
6 echo "$php est ${$php}", "<br/>";
```

Les variables superglobales

Les superglobales sont des variables prédéfinies

- Contenant des informations sur le serveur et les données transitant entre poste client et serveur : données de formulaires, fichiers, cookies, sessions, ...
- De type tableau.
- Accessibles à partir de tout script.

Superglobales

Identifiant	Description
<code>\$GLOBALS</code>	Contient nom et valeur de toutes les variables globales du script. Les noms des variables sont les clés du tableau. Exemple : <code>\$GLOBALS["var"]</code> récupère la valeur de <code>\$var</code> .
<code>\$_COOKIE</code>	Contient nom et valeur des cookies enregistrés sur le poste client. Les noms des cookies sont les clés du tableau.
<code>\$_ENV</code>	Contient nom et valeur des variables d'environnement.
<code>\$_FILES</code>	Contient les noms des fichiers téléversés à partir du poste client.
<code>\$_GET</code>	Contient nom et valeur des données issues d'un formulaire envoyés par la méthode HTTP GET. Les noms (attribut <code>name</code>) des champs du formulaire sont les clés du tableau.
<code>\$_POST</code>	Contient nom et valeur des données issues d'un formulaire envoyés par la méthode HTTP POST. Les noms (attribut <code>name</code>) des champs du formulaire sont les clés du tableau.
<code>\$_REQUEST</code>	Contient l'ensemble des superglobales <code>\$_GET</code> , <code>\$_POST</code> , <code>\$_COOKIE</code> et <code>\$_FILES</code> .
<code>\$_SERVER</code>	Contient les informations liées au serveur : contenu des en-têtes HTTP, nom du script en cours d'exécution, ...
<code>\$_SESSION</code>	Contient l'ensemble des noms des variables de session et leur valeurs.

Superglobale \$_SERVER : quelques éléments

Identifiant	Description
<code>\$_SERVER["DOCUMENT_ROOT"]</code>	La racine web sous laquelle le script est exécuté (eg. /var/www).
<code>\$_SERVER["PHP_SELF"]</code>	Chemin du script en cours par rapport à la racine web.
<code>\$_SERVER["REQUEST_METHOD"]</code>	Méthode de requête HTTP utilisée pour accéder à la page (GET, ...).
<code>\$_SERVER["QUERY_STRING"]</code>	Chaîne de requête utilisée, si elle existe, pour accéder au script.
<code>\$_SERVER["HTTP_ACCEPT"]</code>	Contenu de l'en-tête Accept : de la requête courante.
<code>\$_SERVER["HTTP_ACCEPT_CHARSET"]</code>	Contenu de l'en-tête Accept-Charset : de la requête courante (eg. 'iso-8859-1,*utf-8').
<code>\$_SERVER["HTTP_ACCEPT_ENCODING"]</code>	Contenu de l'en-tête Accept-Encoding : de la requête courante (eg. 'gzip').
<code>\$_SERVER["HTTP_ACCEPT_LANGUAGE"]</code>	Contenu de l'en-tête Accept-Language : de la requête courante (eg. 'fr').
<code>\$_SERVER["HTTP_USER_AGENT"]</code>	Contenu de l'en-tête User-Agent : de la requête courante.
<code>\$_SERVER["HTTPS"]</code>	Valeur vide si le script n' a pas été appelé par HTTPS.

Les opérateurs d'affectation combinée

Réalisent une opération entre deux opérandes et affectent le résultat à l'opérande de gauche.

Opérateur	Description
<code>+=</code>	$x += y$ équivaut à $x = x + y$. y peut être une expression complexe dont la valeur est un nombre.
<code>-=</code>	$x -= y$ équivaut à $x = x - y$. y peut être une expression complexe dont la valeur est un nombre.
<code>*=</code>	$x *= y$ équivaut à $x = x * y$. y peut être une expression complexe dont la valeur est un nombre.
<code>**=</code>	$x **= y$ équivaut à $x = x^{y}$. y peut être une expression complexe dont la valeur est un nombre.
<code>/=</code>	$x /= y$ équivaut à $x = x / y$. y peut être une expression complexe dont la valeur est un nombre différent de 0.
<code>%=</code>	$x \% = y$ équivaut à $x = x \% y$. y peut être une expression complexe dont la valeur est un nombre différent de 0.
<code>.=</code>	$x .= y$ équivaut à $x = x . y$. y peut être une expression complexe dont la valeur est une chaîne de caractères.

Les constantes personnalisées

Se définissent par appel à `define` ou avec le mot-clé `const` :

- `define(string nom, valeur, boolean casse)`
 - crée la constante de nom `nom` et valeur `valeur`
 - son nom est insensible à la casse ssi `casse` vaut `true`.
 - `const nom = valeur;`
 - crée la constante de nom `nom` et valeur `valeur`.
-
- La fonction `defined(string nom)` teste si la constante de nom `nom` existe.
 - Insérer une constante dans une chaîne ne peut se faire que par concaténation.

Les constantes personnalisées

exemple2-2.php

```
1 // Définition insensible a la casse
2 define("PI", 3.1415926535, TRUE);
3 // Utilisation
4 echo "La constante PI vaut ",PI,"<br />";
5 echo "La constante PI vaut ",pi,"<br />";
6 // Vérification de l'existence
7 if (defined( "PI")) echo "La constante PI est d&#233;ja
d&#233;finie", "<br />";
8 if (defined( "pi")) echo "La constante pi est d&#233;ja
d&#233;finie", "<br />";
9 // Définition sensible à la casse, vérification de l'existence et utilisation
10 if(define("site", "http://www.funhtml.com", FALSE))
11 {
12     echo "<a href=",site,">Lien vers mon site</a>";
13 }
```

Les constantes prédéfinies

Les constantes prédéfinies sont nombreuses !

```
<?php print_r(get_defined_constants()); ?>
```

Quelques exemples :

Nom	Description
PHP_VERSION	Version de PHP installée sur le serveur.
PHP_OS	Nom du système d'exploitation du serveur.
DEFAULT_INCLUDE_PATH	Chemin d'accès aux fichiers par défaut.
__FILE__	Nom du fichier en cours d'exécution.
__LINE__	Numéro de ligne du fichier en cours d'exécution.

Les types de données

Les types prédéfinis

Type abstrait	Type	Description
Scalaire	integer	entiers en base 2, 8, 10 ou 16
	float ou double	nombres décimaux
	string	chaînes de caractères
	boolean	les booléens TRUE et FALSE
Composés	array	tableaux
	object	objets
Spéciaux	resource	(références à) ressources externes
	null	type nul

Les pseudo-types de données

Mots-clés utilisés **UNIQUEMENT** pour documenter prototypes de fonctions, constructeurs du langage, mots-clés, ...

- Type d'arguments ou de valeur-retour attendus.
- Absence d'arguments ou de valeur-retour.
- Nombre d'arguments variables ...

Pseudo-type	Description
<code>mixed</code>	le paramètre peut accepter plusieurs types
<code>number</code>	le paramètre est de type <code>integer</code> ou <code>float</code>
<code>callable</code>	le paramètre est une fonction de rappel (alias callback)
<code>array object</code>	le paramètre est de type <code>array</code> ou <code>object</code>
<code>\$...</code>	nombre indéfini d'arguments
<code>void</code>	pas d'arguments (depuis PHP 7.1, <code>void</code> est un type de valeur-retour valide)

Détermination du type d'une variable

```
string get_type($var)
```

Retourne le type de `$var` sous forme de `string`.

Fonctions de vérification (de valeur retour booléenne) :

- `is_scalar($var)` : teste si `$var` est un scalaire.
- `is_numeric($var)` : teste si `$var` est un nombre (integer ou float).
- `is_integer($var)` ou `is_int($var)`,
`is_float($var)`, `is_string($var)`,
`is_bool($var)`.
- `is_array($var)`, `is_object($var)`.
- `is_resource($var)`, `is_null($var)`.

`var_dump($var)` : affiche type et valeur de `$var`.

Conversion de type (alias transtypage, coercion, cast)

Utile en particulier pour traiter les données de formulaire
(toutes de type `string`)

Deux méthodes possibles :

- par affectation combinée à une coercion

```
$var2 = (type_désiré) $var1;
```

- en utilisant la fonction `settype()`

```
boolean settype($var,type_désiré)
```

coercition.php

```
1 $x="3.14 radians";  
2 echo "\$x is string ".$x."<br/>"; // $x is string 3.14 radians  
3 settype($x, "double");  
4 echo "\$x is double ".$x."<br/>"; // $x is double 3.14  
5 $y = (integer) $x;  
6 echo "\$y is integer ".$y."<br/>"; // $y is integer 3  
7 settype($y, "boolean");  
8 echo "\$y is boolean ".$y."<br/>"; // $y is boolean 1
```

Contrôler l'état d'une variable

Utile pour tester les champs de formulaire

Fonctions booléennes à disposition :

- `isset($var)` : FALSE ssi \$var n'existe pas, ou n'est pas initialisée, ou a la valeur NULL.
- `empty($var)` : TRUE ssi \$var n'existe pas, ou n'est pas initialisée, ou vaut NULL, FALSE, 0, "", "0", ou [].

controle-etat.php

```
1 $unini; $null=NULL;
2 echo (int) isset($undef) . "<br/>"; // 0
3 echo (int) isset($unini) . "<br/>"; // 0
4 echo (int) isset($null) . "<br/>"; // 0
5 echo (int) !empty(NULL) . "<br/>"; // 0
6 echo (int) !empty(FALSE) . "<br/>"; // 0
7 echo (int) !empty(0) . "<br/>"; // 0
8 echo (int) !empty(0.0) . "<br/>"; // 0
9 echo (int) !empty("") . "<br/>"; // 0
10 echo (int) !empty('0') . "<br/>"; // 0
11 echo (int) !empty([]) . "<br/>"; // 0
```

Contrôler l'état d'une variable : opérateur ??

Idiome classique (patron-is-set.php)

```
1 if (isset($_GET['user'])) {  
2     $nom = $_GET['user'];  
3 } else {  
4     $nom = 'nobody';  
5 }  
6 // équivalent à  
7 $nom = isset($_GET['user']) ? $_GET['user'] : 'nobody';
```

Opérateur ?? "null coalescing" (depuis PHP 7) (opérateur-coalescing.php)

```
1 $nom = $_GET['user'] ?? 'nobody';  
2 // équivalent à  
3 $nom = isset($_GET['user']) ? $_GET['user'] : 'nobody';
```

Les entiers (type `integer`)

Codage

- Généralement sur 32 bits (varie selon les plateformes).
- Valeur comprise dans $\{-2^{31}, \dots, 2^{31} - 1\}$.
- Une variable entière est automatiquement convertie en `double` si sa valeur est en dehors de l'intervalle.

Représentations littérales

- Décimale : `$x=1789; $y=-476;`
- Binaire : `$x=0b1101111101; $y=-0b111011100;`
- Octale : `$x=03375; $y=-0734;`
- Hexadécimale : `$x=0x6FD; $y=-0x1DC;`

Les entiers sont généralement affichés en base 10 (eg. `echo`).

Les flottants (type `double`)

Le type `double`

Représente les nombres décimaux avec une précision de 14 chiffres.

- Voir librairie `BCMath` pour permettre des calculs plus précis.

Représentations littérales

- Notation décimale avec `.` : `$x=101.23;`
- Notation scientifique avec `E` ou `e` : `$x=1.0123E2;`

Les flottants sont affichés sous forme décimale si le nombre a moins de 15 chiffres, sous forme exponentielle sinon.

Les opérateurs numériques

S'appliquent aux opérandes de type numérique

Opérateur	Description
+	addition
-	soustraction
*	multiplication
**	exponentiation/puissance (associatif à droite)
/	division
%	modulo (s'applique aux flottants par restriction aux parties entières)
--	pré-décrementation (--\$x) ou post-décrementation (\$x--)
++	pré-incrémentation (++\$x) ou post-incrémentation (\$x++)

Les fonctions mathématiques

De nombreuses fonctions sont disponibles par défaut :

- **Conversions** : `integer hexdec(string) ...`
- **Arrondis** : `double ceil(double) ...`
- **Trigonométrie** : `double cos(double) ...`
- **Logarithmes** : `double log(double X, double B) ...`
- **Génération de nombres aléatoires** : `integer rand(integer min, integer max) ...`
- ...

Les booléens (type `boolean`)

Le type `boolean`

- Contient uniquement les valeurs `TRUE` et `FALSE`.
- Est à la base des instructions conditionnelles.

Pour l'affichage, PHP assimile `TRUE` à "1" et `FALSE` à "".

Evaluation en contexte booléen d'expression

- à valeur booléenne
 - `$x < 10` vaut `TRUE` ssi `$x` est de valeur inférieure à 10.
- à valeur non booléenne :
 - `$x` vaut `TRUE` ssi `$x` est initialisée à une valeur *non nulle*.

Règles d'évaluation booléenne d'expressions

Expressions évaluées à FALSE	Une expression logique fausse utilisant un ou plusieurs opérateurs de comparaison
	Le mot-clé <code>FALSE</code>
	La valeur entière 0
	La valeur décimale 0.0
	Les chaînes "" et "0"
	Une variable de type <code>null</code>
	Une variable non initialisée
	Un tableau vide
	Un objet sans propriétés ni méthodes
Expressions évaluées à TRUE	<p>Toute autre possibilité dont :</p> <ul style="list-style-type: none">- Les entiers strictement positifs ou négatifs- La chaîne <code>"FALSE"</code>- Les variables de type <code>resource</code>

Règles d'évaluation booléenne d'expressions

evaluation-booleenne.php

```
1 // Expressions fausses
2 if (!FALSE)      var_dump (FALSE);      // bool(false)
3 if (!0)          var_dump (0);           // int(0)
4 if (!0.0)        var_dump (0.0);         // float(0)
5 if (!" ")        var_dump (" ");         // string(0) ""
6 if (!"0")        var_dump ("0");         // string(1) "0"
7 if (!null)       var_dump (null);        // NULL
8 if (!isset($x)) echo "1 ";              // 1
9 if (![ ])       var_dump ([ ]);          // array(0) {}
10 // Expressions vraies
11 if (-1)          var_dump (-1);          // int(-1)
12 if ("FALSE")     var_dump ("FALSE");     // string(5) "FALSE"
13 $f = fopen(__FILE__, "r");
14 if ($f)          var_dump ($f);          // resource(5) of type (stream)
```

Les opérateurs booléens

Se divisent en

- Opérateurs de comparaison (binaires).
- Opérateurs logiques de composition (unaires/binaires).

Opérateurs de comparaison

Opérateur	Description
==	égalité
!= ou <>	inégalité
<	strictement inférieur
<=	inférieur ou égal
>	strictement supérieur
>=	supérieur ou égal
<=>	$\$x<=>\y vaut -1 ssi $\$x<\y , 0 ssi $\$x=\y , 1 ssi $\$x>\y ("spaceship operator" introduit en PHP 7)
===	égalité des valeurs et des types
!==	inégalité des valeurs ou des types

Comparaison et coercion implicite

L'évaluation booléenne d'expressions met en jeu des règles de coercion implicites.

operateur-comparaison.php

```
1 var_dump(FALSE==0); // TRUE
2 var_dump(FALSE==0.0); // TRUE
3 var_dump(FALSE==""); // TRUE
4 var_dump(FALSE=="0"); // TRUE
5 var_dump(FALSE==NULL); // TRUE
6 var_dump(FALSE==[]); // TRUE
7 var_dump(0.0==0); // TRUE - conversion (double) 0
8 var_dump(0==""); // TRUE - conversion (integer) ""
9 var_dump(41=="41ok12"); // TRUE - conversion (integer) "41ok12"
10 var_dump(41==" 41ok12"); // TRUE - conversion (integer) " 41ok12"
11 var_dump(0=="a41ok12"); // TRUE - conversion (integer) "a41ok12"
12 var_dump(NULL==""); // TRUE
13 var_dump(NULL==[]); // TRUE
14 ////
15 var_dump(""=="0"); // FALSE
16 var_dump(""==[]); // FALSE
17 var_dump(0==[]); // FALSE
```

Comparaison stricte

Pour éviter les désagréments, privilégier les opérateurs `===` et `!==`.

operateur-comparaison-strict.php

```
1 var_dump(TRUE===1); // FALSE
2 var_dump(" "===NULL); // FALSE
3 var_dump(1e2==="100"); // FALSE
```

Les opérateurs booléens

Opérateurs logiques

Opérateur	Description
OR	disjonction
	équivalent à OR mais n'a pas la même priorité
XOR	disjonction exclusive
AND	conjonction
&&	équivalent à AND mais n'a pas la même priorité
!	négation

Priorité des opérateurs en PHP

Les chaînes de caractères

Suites de caractères alphanumériques encadrées par

- des apostrophes : 'PHP7 et MySQL'
- des guillemets : "PHP7 et MySQL"

Traitement d'une variable apparaissant dans une chaîne

- Son identifiant est affiché si la chaîne est encadrée par des apostrophes.
 - `<?php $x="A"; echo ' $x' ; ?>` affiche \$x.
- Sa valeur est affichée si la chaîne est encadrée par des guillemets.
 - `<?php $x="A"; echo " $x" ; ?>` affiche A.

Affichage mixte à l'aide de séquences d'échappement

```
<?php $x="A"; echo "\$x vaut $x"; ?> affiche $x vaut A.
```


Séquences d'échappement

Pour afficher des caractères

- Protégés du langage (eg, \$).
- De contrôle (eg. un retour à la ligne).
- A partir de leur code ASCII octal ou hexadécimal.

Séquence	Signification
\'	affiche une apostrophe
\"	affiche un guillemet droit
\\$	affiche le symbole \$
\\	affiche une barre oblique inversée \ (alias backslash)
\n	nouvelle ligne (code ASCII 0x0A)
\r	retour chariot (code ASCII 0x0D)
\t	tabulation (code ASCII 0x09)
\[0-7] {1,3}	affiche le caractère dont le code ASCII en octal correspond à la séquence de caractères (séquence de 1 à 3 chiffres pris entre 0 et 7). <code>echo "\101";</code> affiche A
\x[0-9 A-F a-f] {1,2}	affiche le caractère dont le code ASCII en hexadécimal correspond à la séquence de caractères (séquence de 1 à 2 caractères). <code>echo "\x4A";</code> affiche J

Concaténation de chaînes

Avec l'opérateur . (concatenation.php)

```
1 $str1 = "AA";  
2 $str2 = "BB";  
3 $str3 = $str1.$str2."<br/>";  
4 echo $str3; // affiche AABB<br/>
```

Alternative : avec la , pour echo (concatenation-echo.php)

```
1 $str1 = "AA";  
2 $str2 = "BB";  
3 echo $str1,$str2,"<br/>"; // affiche AABB<br/>
```

Les tableaux

Un tableau stocke des *éléments* (valeurs) indépendants

- Les éléments peuvent prendre n'importe quel type.
- Les éléments peuvent être de types différents.

On peut créer implicitement des tableaux multi-dimensionnels avec des tableaux de tableaux.

Deux types de tableaux :

- Tableau *indiqué* : les éléments sont repérés par des indices numériques.
- Tableau *associatif* : les éléments sont repérés par des *clés* (chaîne ou variable de type chaîne).

Les tableaux indicés

Création et accès aux éléments avec la notation []

- `$tab[n]=exp;` initialise/remplace l'élément d'indice `n` du tableau `$tab` à/avec l'expression `exp`.
- `$tab[]=exp;` ajoute l'élément `exp` en fin du tableau `$tab`, c.a.d. à l'indice qui suit l'indice maximum ou 0 si `$tab` est vide.
- `print_r` formate l'affichage de tableaux.

tableau-indice-creation1.php

```
1 $tab[0]="UFR"; // équivalent ICI à $tab[]="UFR";
2 $tab[1]="Sciences"; // équivalent ICI à $tab[]="Sciences";
3 $tab[49]="Angers";
4 $tab[]="janvier"; // équivalent ICI à $tab[50]="janvier";
5 $i = 3;
6 $tab[$i]="Université"; // équivalent ICI à $tab[3]="Université";
7 $tab[]="2017"; // équivalent ICI à $tab[51]="2017";
8 echo count($tab), "<br/>"; // affiche 6
9 print_r($tab);
```

Les tableaux associatifs

Création et accès aux éléments avec la notation []

Les clés (littéral ou variable `string`) remplacent les indices.

- Les clés ne comportent pas d'espace, sont encadrées par apostrophes/guillemets, et sont sensibles à la casse.
- Encadrer tout élément de tableau utilisé dans une chaîne par des accolades
 - `"{$stab['alpha']}"` et non pas `"$stab['alpha']"`.

tableau-associatif-creation1.php

```
1 $stab["zero"]="UFR";
2 $stab[1]="Sciences";
3 $stab["forty-nine"]="Angers";
4 $stab[]="janvier"; // équivalent ICI à $stab[2]="janvier";
5 $stab["3"]="Université";
6 $stab[51]="2017";
7 echo count($stab), "<br/>";
8 print_r($stab);
```

Les tableaux associatifs

exemple2-3.php

```
1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4 <meta charset="UTF-8" />
5 </head>
6 <body>
7 <?php
8 //création des éléments du tableau
9 $stab["php"] = "php.net";
10 $stab["mysql"] = "mysql.com";
11 $stab["html"] = "w3.org";
12 //création des liens
13 echo "<h2> Mes liens préférés </h2>";
14 echo "<ul><li><a href=\" http://www.{\$stab['php']}\" title=\"Le site
php.net\">&nbsp; PHP </a> </li>";
15 echo "<li><a href=\" http://www.{\$stab['mysql']}\" title=\"Le site
mysql.com\">&nbsp; MySQL </a> </li>";
16 echo "<li><a href=\" http://www.{\$stab['html']}\" title=\"Le site du
W3C\">&nbsp; HTML </a> </li></ul>";
17 ?>
18 </body>
19 </html>
```

Les types spéciaux

Le type `resource`

- Une valeur de type `resource` est une référence vers une ressource externe : fichier ouvert, connexion à base de données (BDD), image, ...
- Libérées automatiquement par le ramasse-miettes SAUF cas des connexions persistantes à BDD.

Le type `null` (ou `NULL`)

Attribué à une variable sans contenu ou qui a été explicitement initialisée à la valeur `NULL`.

- "" et "0" ont le type `string`.
- 0 a le type `integer`.