BACK #1319 - PHP

1 - Structures de base

1.1 - PHP et HTML

> Le code PHP peut être directement intégré dans les fichiers HTML.

Le code PHP peut figurer à différents endroits de ces fichiers, tout en étant entrecoupé de code HTML (ou réciproquement).

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP</title>
    </head>
    <body>
        <h1>Les structures de base</h1>
        <p>
            <?php echo "Ceci est du code PHP"; ?>
        </p>
        <p>Ceci est du code HTML</p>
        <?php echo "<p>Puis à nouveau du code PHP</p>"; ?>
    </body>
</html>
```

Balises d'ouverture et de fermeture

Le début et la fin des portions de code PHP sont signalés grâce à des **balises d'ouverture et de fermeture**. Seul ce qui est entre ces balises est interprété par PHP, le reste est envoyé tel quel.

Ces balises sont : `<?php` : ouverture et `?>` : fermeture

Balises d'ouverture rapide : Il existe une syntaxe supplémentaire : `<?= valeur; ?>` qui équivaut à `<?php echo valeur; ?>`

Commentaires

```
// Commentaire PHP sur une ligne

/*
Commentaire PHP sur
Plusieurs lignes
*/
```

```
echo 'Quelque chose<br />'; // Commentaire jusqu'à la fin de la ligne
echo 'Quelque chose';       # Commentaire jusqu'à la fin de la ligne

/**
* [une_fontion description]
* @param  [type] $param1 [description]
* @param  [type] $param2 [description]
* @return  [type]         [description]
*/
function une_fontion( $param1, $param2) {
    // Traitement;
    return $valeur;
}
```

Enchaînement des instructions

> Les instructions PHP doivent être placées entre les balises d'ouverture et de fermeture et séparées par des points-virgules.

Seuls les points-virgules `;` séparent les instructions; Les retours à la ligne n'ont aucune influence. Plusieurs instructions peuvent être écrites sur la même ligne Une instruction peut être écrite sur plusieurs lignes

```
$a = 25; echo $a;
$b =
"PHP 8"; echo $b;

/* Equivaut à */
$a  = 25;
echo $a;
$b  = "PHP 8";
echo $b;
```

Erreur courante : Parse error on line X

Cela signifie que PHP, en lisant ligne à ligne le fichier de script, a rencontré une incohérence de syntaxe (point-virgule manquant, guillemet en trop, ...).

Affichage et envoi du code HTML

> La commande `echo` indique au serveur les informations contenues entre les guillemets.

Il est possible de renvoyer du code HTML.

Par exemple des balises

ou encore le code HTML désignant une image :

```
echo "<p>Ceci est du code PHP</p>";
echo "<img src='./image.jpg' alt='texte de remplacement';
```

1.2 - Variables

Syntaxe des variables

En PHP, les variables :

- commencent par le symbole `$`,
- sont constituées de chiffres, lettres et tirets bas (underscore : _),
- le premier caractère du nom de la variable (après le `$`) ne peut pas être un chiffre,
- sont sensibles à la casse

```
$une_1ere_variable = "Ma première variable";
```

Conventions de nommage :

- *Camel case* : **maPremiereVariable** (Javascriopt, Java, C, C++, ...)
- *Pascal case* : **MaPremiereVariable** (PASCAL, noms de classes en PHP)
- *Kebab case* ou *Spinal case* : **ma-premiere-variable** (URL, noms de fichiers, HTML/CSS : noms de classes et d'id) *Underscore case* ou *Snake case* : **ma_premiere_variable** (PHP, Ruby, Python) *Screaming snake case* : **MA_PREMIERE_VARIABLE** (PHP : noms de constantes)

Durée de vie des variables

> Les variables sont temporaires : elles n'existent que le temps du script.

Une fois le traitement de la page terminé, elles cessent d'exister. Il est donc impossible de les relire dans un autre script ou lors d'une nouvelle exécution du même script.

Variables locales et variables globales

Cette partie sera détaillée lors de la présentation des fonctions.

Variables dynamiques

Egalement appelées "*variables variables*", elles reposent sur le fait que le nom de variable peut lui-même être une variable.

```php
$cd  = "15 €";
$dvd = "30 €";

$produit = "dvd";  // On choisit le DVD comme produit

echo $$produit;    // Equivaut à $dvd, soit 30 €
echo ${$produit};  // Autre notation
```

1.3 - Constantes

PHP définit les constantes à l'aide du mot clé `const` ou de la fonction `define()`.

Leur valeur ne peut plus être modifiée par la suite.

Par convention, les constantes sont écrites en majuscules (*Screaming snake case*).

Attention : les constantes n'ont pas de `$` devant leur nom.

```php
const NOM = "PHP";
const VERSION = "8";

define("HOME_URL", "http://monsite.fr");
define("NOMBRE_MAX_TENTATIVES", 3);

echo '<h1>';
echo NOM;
echo ' ';
echo VERSION;
echo '</h1>';
var_dump(HOME_URL);
var_dump(NOMBRE_MAX_TENTATIVES);
```

Constantes magiques

> Il existe neufs constantes qui changent suivant l'emplacement où elles sont utilisées

Nom	Description
__LINE__	La ligne courante dans le fichier
__FILE__	Le chemin complet et le nom du fichier courant
__DIR__	Le dossier du fichier
__FUNCTION__	Le nom de la fonction
__CLASS__	Le nom de la classe courante. Le nom de la
__TRAIT__	Le nom du trait. Le nom du trait inclut l'espace de nom dans lequel il a été déclaré
__METHOD__	Le nom de la méthode courante
__NAMESPACE__	Le nom de l'espace de noms courant
ClassName::class	Le nom entièrement qualifié de la classe

1.4 - Types de données

> PHP définit le type d'une variable en fonction de la valeur qui lui est assignée.

PHP fournit 4 types de données simples (scalaires/scalar) : **booléen**, **entier**, **nombres à virgule flottante** et **chaînes de caractères**.

La fonction `gettype()` permet de connaître le type d'une variable

Le type booléen (*boolean*)

> Un booléen est une valeur pouvant être soit vraie, soit fausse.

Le mot-clé `true` désigne un booléen vrai et `false` un booléen faux.

Ces mots sont insensibles à la casse.

Le type entier (*integer*)

Un nombre entier négatif est précédé du signe `-` Un nombre commençant par un chiffre de 1 à 9 est interprété selon la base décimale Un nombre commençant par `0` est interprété selon la base octale Un nombre commençant par `0x` est interprété selon la base hexadécimale

Le type nombre à virgule flottante (*double*, *float*)

Un nombre à virgule flottante peut comporter une partie décimale Le séparateur décimal est le point `.` Un nombre à virgule flottante peut également être écrit sous sa forme exponentielle

Le type chaine de caractères (*string*)

> Les chaînes de caractères sont délimitées par des guillemets (`"`) ou des apostrophes (`'`).

Interprétation des variables

> à l'intérieur d'une chaîne délimitée par des guillemets (`"`), une variable est remplacée par sa valeur. Les variables peuvent être délimitées par des accolades.

> à l'intérieur d'une chaîne délimitée par des apostrophes (`'`), une variable n'est pas remplacée.

Caractère d'échappement (ou de protection)

Utiliser des guillemets (resp. apostrophes) dans une chaîne délimitée par des guillemets (resp. apostrophes) provoque une erreur.

Le caractère d'échappement `\` indique à PHP de considérer que le caractère qui suit est à traiter comme tout autre caractère et n'a pas de fonction particulière.

Le type tableau (*array*)

En plus des types de données simples, PHP propose une façon de les grouper et autorise deux types de tableaux (type de données composées) :

- les tableaux indexés numériquement
- les tableaux associatifs

> Un tableau PHP est un type de données qui associe une clé à une valeur.

Cette clé est définie par une valeur numérique (clé à indexation numérique) ou une chaîne de caractères (clé associative).

Un tableau est une liste d'éléments créée grâce au mot clé `array()` ou directement avec des crochets `[]`.

La valeur est associée de la clé par l'expression `=>`.

Chaque paire clé/valeur y figurant doit être séparée des autres par une virgule (`,`).

Tableaux à indexation numérique

Les *tableaux indexés numériquement* sont des tableaux dont chaque clé est une valeur entière positive ou négative.

Chaque clé est unique.

Lors de la déclaration du tableau, la valeur de la clé est facultative.

Si la valeur de la clé n'est pas fournie, PHP utilisera une clé qui correspond à la plus grande valeur existante, incrémentée de 1. **Les index commencent à 0.**

> Si plusieurs clés identiques sont fournies, PHP résout le conflit en ne conservant que la dernière valeur fournie pour cette clé.

Tableaux associatifs

Les tableaux associatifs sont des tableaux dont chaque clé est représentée par une chaîne de caractères (même vide).

Chaque clé est optionnelle et unique.

Lors de la déclaration du tableau, la valeur de la clé est facultative.

Si la valeur de la clé n'est pas fournie, PHP utilisera une clé qui correspond à la plus grande valeur numérique existante, incrémentée de 1. Les index commencent à 0.

Si plusieurs clés identiques sont fournies, PHP résout le conflit en ne conservant que la dernière valeur fournie pour cette clé.

Tableaux multidimensionnels

> En PHP, les tableaux multidimensionnels (matrices) sont des tableaux de tableaux.

Lors de la déclaration du tableau, la valeur de la clé est facultative.

Si la valeur de la clé n'est pas fournie, PHP utilisera une clé qui correspond à la plus grande valeur numérique existante, incrémentée de 1. Les index commencent à 0.

Si plusieurs clés identiques sont fournies, PHP résout le conflit en ne conservant que la dernière valeur fournie pour cette clé.

Vérification des types de données

PHP propose des fonctions permettant de vérifier si une variable est d'un type donné.

Ces fonctions renvoient `true` si la variable est du type testé, `false` sinon.

Fonction	Description
is_bool()	Détermine si une variable est de booléen
is_int() / is_integer() / is_long()	Détermine si une variable est de type nombre entier
is_float() / is_double() / is_real()	Détermine si une variable est de type nombre décimal
is_string()	Détermine si une variable est de type chaîne de caractères
is_array()	Détermine si une variable est un tableau
is_scalar()	Indique si une variable est un scalaire (entier, nombre décimal, chaîne de caractères, booléen)
is_numeric()	Détermine si une variable st un type numérique

En complément les fonctions `isset()`, `is_empty()` et `is_null()` permettent de vérifier respectivement si une variable est définie, vide ou nulle (`null`)

Transtypage

> PHP permet de manipuler les données sans déclarer leur type. Quand une donnée d'un certain type est attendue et qu'une donnée d'un autre type est fournie, PHP tente d'effectuer une conversion automatique.

Règles de conversion

Chaîne de caractères vers nombre : PHP parcourt la chaîne de gauche à droite :

- si le caractère n'est pas un chiffre (ou le signe -), la chaîne est convertie en la valeur 0,
- si le caractère est un chiffre (ou le signe -), PHP passe au caractère suivant jusqu'à trouver un caractère non numérique ou la fin de la chaîne.

Booléen vers nombre :

- `false` est converti en un entier de valeur 0
- `true` est converti en un entier de valeur 1

Forcer une conversion : pour forcer la conversion d'un type de données vers un autre, on indique le type souhaité entre parenthèses ou on utilise la fonction `settype()` :

```
$a = (string) 1234;
$a = 1234; settype( $a, "string" );
```

Il existe également quelques fonctions permettant d'obtenir la valeur d'une variable dans un format donné : `strval()`, `boolval()`, `floatval()`,`intval()`

Vers chaîne de caractères :

- `null` et `false` sont convertis en une chaîne vide,
- `true` est converti en '1'
- Les valeurs numériques numériques sont convertis en leurs équivalents décimaux

Vers un booléen : les conversions en booléen des valeurs suivantes seront considérées comme `false` :

- valeur vide ou tableau vide,
- false et null
- les entiers et nombres numériques 0
- une chaîne de caractères comportant uniquement le caractère 0