# Résumé du cours PHP

28/05/2018 - Matinée

## Qu'est-ce que PHP ?

PHP, ou Hypertext Preprocessor, est un langage de programmation **libre** utilisé pour développer des applications ou des pages sur le web. Il permet **d'interagir avec un serveur** et permet d'exécuter des fonctionnalités complexes dont la rapidité dépend dudit serveur.
On parle de langage back. A l'inverse, pour HTML, on parle de langage front.

## Pourquoi utiliser PHP ?

On va utiliser PHP principalement pour des fonctionnalités. Quelques exemples de réalisations possibles à l'aide de PHP :

- Système d'authentification
- Minichat
- Envoi de mail
- Mise à jour de contenu
- Interface d'administration
- …

Les possibilités sont aussi nombreuses que vous le permet votre créativité.

## Comment créer un fichier PHP ?

Il suffit de créer un fichier avec l'extension .php. Pour pouvoir exécuter ce fichier .php, on va avoir besoin d'un serveur local qui interprétera le code PHP. C'est pour cela que l'on utilise WAMP. Il existe des alternatives à wamp : Xampp, Mamp (pour mac), uwamp et j'en passe.

## Comment avons-nous appris PHP ?

Nous sommes partis de vos connaissance en javascript pour les transposer sur le PHP. En effet, les bases algorithmiques des langages sont les mêmes : variables, conditions, boucles et fonctions ont les mêmes utilités.

# Les variables

## Créer une variable et les types

Pour créer une variable, on joint le signe « $ » à côté du nom de la variable :

```php
<?php

$age         = 18;
$prenom      = "Yohan";
$temperature = 20.2;
$nuit        = true;
$tableau     = array();
```

Tout comme en JavaScript, les variables ont des **types**.

- « age » est une variable de type nombre entier (ou INT)
- « prenom » est une variable de type chaîne de caractères (ou STRING)
- « temperature » est une variable de type flottant/nombre décimal (ou FLOAT)
- « nuit » est une variable de type booléen (ou BOOLEAN)
- « tableau » est une variable de type tableau (ou ARRAY)

## Faire des opérations basiques sur les variables

On peut réaliser des opérations sur les variables grâce aux opérateur arithmétiques.

```php
$a = 5;
$b = 15;
$c = 5.2;
$d = 0;

echo $a + $b; // Affiche 20 à l'écran (5 + 15 = 20)
echo $b - $c; // Affiche 10.2 à l'écran (15 - 5.2 = 10)
echo $d * $a; // Affiche 0 à l'écran (0 * 5 = 0)
echo $b / $a; // Affiche 3 à l'écran (15 / 5 = 3)
echo $c % $a; // Affiche 0.2 à l'écran (le reste de 5.2 / 5 est 0.2 )
```

- « + » sert à faire une addition
- « - » sert à faire une soustraction
- « * » sert à faire une multiplication
- « / » sert à faire une division
- « % » sert à récupérer le reste d'une division euclidienne

Comme en javascript, le principe de concaténation existe également cependant, en PHP, on va utiliser le « . ».

```php
$nom    = "Pradeau";
$prenom = "Yohan";
$ville  = "Oullins";

echo "Bonjour, je m'appelle " . $nom . " " . $prenom . " et j'habite à " . $ville . ". Je suis heureux de vous rencontrer !";
// Affichera : Bonjour, je m'appelle Pradeau Yohan et j'habite à Oullins. Je suis heureux de vous rencontrer !
```

# Les conditions

## Si … Sinon

```php
$age = 18;

if( $age >= 21 ) {
    echo "Tu es majeur dans le monde !";
} elseif( $age > 17 ) {
    echo "Tu es majeur en France !";
} else {
    echo "Tu n'es pas majeur ...";
}

// Affichera "Tu es majeur en France !"
```

La structure utilise les mêmes keywords et la même structure qu'en JavaScript. On Chaque branche de notre structure conditionnelle comporte une condition. Dans le cas où aucune condition ne serait remplie, on peut utiliser une condition par défaut : le **else**. Une fois que le script est « entré » dans une branche d'une structure conditionnelle, elle en ressort immédiatement (et ne regardera pas dans les autres branches suivantes).

## Opérateurs de Comparaison

Les conditions nous mettent donc à disposition les opérateurs de comparaison :

http://php.net/manual/fr/language.operators.comparison.php

- « === » Test si deux variables sont égales en valeur et en type
- « == » Test si deux variables sont égales
- « != » Test si deux variables sont différentes
- « !== » Test si deux variables sont différentes en valeur et en type
- « > » Test si une variable est supérieure à une autre
- « < » Test si une variable est inférieure à une autre
- « >= » Test si une variable est supérieure ou égale à une autre
- « <= » Test si une variable est inférieure ou égale à une autre

Deux petits nouveaux :

- « <> » Test si deux variables sont différentes (un peu comme !=)
- « <=> » : « Un entier inférieur, égal ou supérieur à zéro lorsque *$a* est respectivement inférieur, égal, ou supérieur à *$b*. »

## Opérateurs de Logique

Elles nous mettent également à disposition les opérateurs de logique :

- && (ou AND)
- || (ou OR)

# Les boucles

Nous avons mis en évidence 4 boucles différentes :

- La boucle **for**
- La boucle **while**
- La boucle **do..while**
- Et la boucle **foreach**

## La boucle FOR

La boucle for se construit comme en JavaScript :

```php
for($i = 0; $i < 20; $i++) {
    echo $i;
}
```

On a d'abord l'initialisation avec $i=0, la condition avec $i < 20 et enfin l'incrémentation avec $i++.

On utilise une boucle for quand on veut qu'un bout de code s'exécute un nombre de fois précis.

## La boucle WHILE

La boucle while se traduit par « tant que » en français. Elle se construit comme en JavaScript :

```php
$compteur = 0;

while($compteur < 10) {

    echo $compteur;

    $compteur++;

}
```

Cette boucle implique d'avoir une variable qui va varier tout au long de tours de boucle. Elle peut être risquée car souvent la cible de boucles infinies (et ça on aime pas du tout !).

## La boucle DO..WHILE

Cette boucle fonctionne pratiquement comme while. La différence, c'est que le code va être exécuté AU MOINS une fois. En réalité, on peut dire : Tu exécutes ce bout de code et tant que la condition que je te donne est vraie, tu recommences.

```php
$compteur = 10;

do {

    echo $compteur;

    $compteur++;

} while($compteur < 10);
```

## La boucle FOREACH

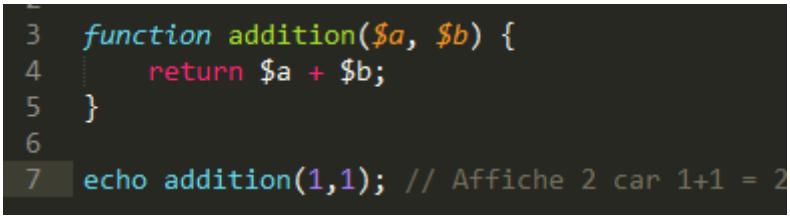
Elle a besoin d'un tableau à parcourir. Elle va parcourir tous les index du tableau et s'arrêter lorsqu'elle n'en trouve plus. A chaque tour de boucle, elle va nous mettre à disposition une « variable temporaire » qui contiendra l'index en train d'être parcouru ainsi que l'index actuel.

```php
$eleves = array(
    0 => array(
        "prenom" => "Jean"
    ),
    1 => array(
        "prenom" => "Mickael"
    ),
    2 => array(
        "prenom" => "Henry"
    )
);

foreach($eleves as $index => $eleve) {
    echo "Dans l'index " . $index . " de $eleves, on trouve l'élève " . $eleve['prenom'];
}
```

# Fonctions

Pour créer une fonction en PHP, il suffit de faire comme en JavaScript.




Attention à la portée des variables ! Toutes les variables créées à l'extérieur d'une fonction ne sont pas accessibles dans la fonction. De la même manière, toutes les variables créées à l'intérieur d'une fonction n'existent qu'uniquement dans cette fonction et pas dans le contexte global du script.

Si je veux utiliser une variable globale dans une fonction, je peux utiliser le mot clef « **global** ».

En savoir + : http://php.net/manual/en/language.variables.scope.php

# Savoir debug son script

Pour pouvoir débuger mon script (et comprendre ses affreuses erreurs oranges), on nous met à disposition de nombreux outils. Ceux qu'on utilise le plus fréquemment sont les suivants :

- La fonction print_r(), elle affiche des informations visibles d'une variable
- La fonction var_dump(), elle affiche un peu plus d'informations sur une variable
- Les fonctions exit() et die(), elles permettent de couper le script au moment où on les appelle. Ceci peut servir pour isoler un var_dump ou un comportement dans mon script et le tester.