NOVEI FORMATION

LA MANU

L'ÉCOLE DES MÉTIERS DU NUMÉRIQUE
Manufacture de compétences

SUPPORT APPRENANT

PHP

CONFIDENTIEL

Ce document est strictement confidentiel et ne doit pas être diffusé sans accord préalable écrit

PHP

Ce que l'on a vu…

Le langage PHP est inventé dans les années 1990 par Rasmus LERDORF. Il est d'abord apparu en tant qu'outil en 1994 sous le nom de Personnal Home Page Tools. Il devient un langage en 1996 et c'est à partir de la version 5.0 de PHP en 2004, que le langage devient un langage de programmation orientée objet.
Aujourd'hui, PHP est l'acronyme de **P**HP **H**ypertext **P**reprocessor, et nous utilisons aujourd'hui la version 7.2. C'est un langage très populaire pour le développement web, plus de 300 millions de sites sont réalisés en PHP à travers le monde.

PHP est un langage de Programmation Orientée Objet, mais c'est également un langage serveur. En effet, le code PHP qui est inclus dans des balises **<?php** *code* **?>** est exécuté sur le serveur qui génère du code HTML qui sera envoyé au client. Ainsi, le PHP est transformé en HTML pour le client. Cependant le code PHP n'est pas accessible par le client comme l'est du HTML ou du JavaScript.

Pour utiliser le langage PHP, il faut commencer par installer **WAMP** sur son ordinateur. WAMP permet de mettre en place un serveur web, et est l'acronyme de **W**indows (système d'exploitation), **A**pache (serveur http), **M**ySQL (serveur de bases de données), **P**HP (langage de programmation). Sans cela le code PHP ne pourra pas être exécuté. Après avoir vérifié et installé les packages manquants grâce au logiciel check_vcredist.exe, on peut télécharger WAMP sur le site http://www.wampserver.com/ et l'installer sur son ordinateur.
Une fois WAMP installé, on peut créer un hôte virtuel (**VHost ou VirtualHost**) via la section « Localhost » du menu de WAMP. Lors de la création du VHost, il faut indiquer le nom du VHost tout en minuscule, sans espace, ni underscore. Il faut également renseigner le chemin complet jusqu'au fichier index.php de son projet pour éviter les erreurs. Une fois le VHost créé, il ne reste plus qu'à redémarrer les services de WAMP et tester le VHost dans le navigateur. Lorsque le VHost est fait on peut alors commencer à coder en PHP.

Le PHP peut s'inclure dans le fichier HTML entre des balises **<?php** *code* **?>**. Ces balises peuvent se placer en dehors des balises HTML ou à l'intérieur. On privilégiera d'écrire le code qui n'est pas directement nécessaire à l'affichage en dehors du HTML, soit avant la balise < !DOCTYPE html>, soit dans un autre fichier.

Les Variables

Pour déclarer une variable en PHP on utilise la syntaxe suivante : ***$nomVariable = valeur;***. Les noms de variable commencent par un $ suivi du nom de la variable en anglais et en camelCase. On affiche une variable grâce à un **echo**.
Exemple :

```
<?php $message = ''Hello world'' ; ?>
…
<p><?php echo $message ; ?></p>
…
```

Ou bien en affichage réduit :

```
<p><?= $message ; ?></p>
```

La balise **<?=** ne s'utilise que pour raccourcir la syntaxe d'affichage d'un **echo**.

Le PHP, contrairement à d'autres langages, est assez permissif. Il n'est pas nécessaire de définir le type d'une variable lorsqu'on la déclare, la variable prend le type de la valeur qui lui est attribuée.

Type	Déclaration de la variable
Chaîne de caractères (string)	$string = 'Message' ;
Nombre entier (int)	$integer = 12 ;
Nombre décimal (float)	$float = 12.5 ;
Booléen (bool)	$boolean = true ;

Il est toutefois possible d'affecter un type particulier à une variable en utilisant la fonction php settype().

Comme en JavaScript, il est possible de concaténer plusieurs variables entre elles ou avec des chaînes de caractères en utilisant un point (.). En revanche lorsqu'on doit afficher une variable dans un texte, il est préférable d'écrire le texte dans le code HTML et d'inclure uniquement la variable en PHP.

Exemple :
```
<p>Bonjour <?= $firstname . ' ' . $lastname ; ?>, comment allez-vous ?</p>
```

Pour que le code s'exécute plus rapidement, il est conseillé d'utiliser des simples quotes plutôt que des doubles quotes. En effet, tout ce qui se trouve entre des simples quotes sera interprété par le navigateur comme étant une chaîne de caractères, alors qu'en présence de doubles quotes ce dernier analysera d'abord la chaîne de caractères à la recherche de variables à afficher.

Les Conditions

En PHP on peut utiliser des conditions de la même manière qu'en JavaScript :
```
if(condition){
   instructions
}elseif(condition){
   instructions
}else{
   instructions
}
```

Il est tout à fait possible de faire des conditions multiples en posant plusieurs conditions à la fois dans un *if* ou un *elseif* en séparant les conditions avec des opérateurs de comparaison (&& ou ||). Les instructions peuvent être constituées de code PHP ou HTML.

Il est également possible d'utiliser des ternaires. Les ternaires sont des conditions dont la syntaxe est raccourcie, leur permettant d'être écrites sur une seule ligne contrairement à une condition classique avec un *if…else…*. La syntaxe d'une ternaire est la suivante :
```
(condition) ? 'instructionsSiConditionTrue' : 'instructionSiConditionFalse' ;
```

Les Boucles

Les boucles servent à répéter une instruction un certain nombre de fois ou tant qu'une condition n'est pas atteinte par exemple. Il existe différents types de boucles.

Boucle while

La boucle **while** peut être traduit en français par « tant que ». Sa syntaxe est la suivante :

```
initialisation ;
while(condition){
   instructions
   pas
}
```

La boucle *while* est simple et flexible, elle s'utilise plutôt lorsque le nombre d'itération de la boucle est inconnu. Cependant leur syntaxe fait qu'il y a plus de risque d'oublier un élément et ainsi faire une boucle infinie, surtout lorsqu'on manipule des valeurs numériques.

Boucle for

La boucle **for** peut être traduit en français par « pour ». Elle s'utilise principalement lorsque l'on connait le nombre de fois que l'on souhaite réaliser une instruction. Sa syntaxe est la suivante :

```
for(initialisation ; condition ; pas){
   instructions
}
```

L'avantage d'une boucle *for* c'est que l'on retrouve toutes les informations importantes, à savoir la valeur initiale, la condition et le pas, au même niveau. Ce qui n'est pas le cas avec une boucle *while* avec laquelle ces informations sont plus dispersées. Il y a donc plus de risques d'oublier un élément (initialisation ou pas) et de faire une boucle infinie. Autant que possible on préfèrera privilégier les boucles *for* aux boucles *while* pour limiter les risques de boucles infinies.

Boucle foreach

La boucle **foreach** peut être traduit en français par « pour chaque élément dans ce tableau ». Il s'agit de la boucle la plus adaptée pour parcourir un tableau. Sa syntaxe est la suivante :

```
foreach($nomTableau as $élément){
   instructions (par ex : echo $élément ;)
}
```

Le nom de la variable *$élément* suivant le « as » est un nom que l'on donne de façon arbitraire et qui contiendra consécutivement la valeur de chaque élément du tableau parcouru. La portée de cette variable se limite à la boucle *foreach*. Pour parcourir un tableau associatif la syntaxe de la boucle est légèrement différente :

```
foreach($nomTableau as $clé => $élément){
   instructions (par ex : echo $clé . ' vaut ' . $élément ;)
}
```

Cette syntaxe permet d'afficher à la fois la clé du tableau associatif et sa valeur.

Les Tableaux

Il existe 2 types de tableaux : les tableaux classiques et les tableaux associatifs. Ils se définissent grâce à la fonction array() et s'initialisent ainsi : **$*nomTableau* = array(*valeur1*, *valeur2*, *valeur3*,…);** ou bien **$*nomTableauAssociatif* = array(*clé1* => *valeur1*, *clé2* => *valeur2*, *clé3* => *valeur3*,…);**.
Dans un tableau classique, un index ou clé est associé par défaut à chaque valeur du tableau. Cet index est un nombre entier. La 1ère valeur du tableau a toujours l'index 0, la 2ème valeur a l'index 1, la 3ème valeur l'index 2, et ainsi de suite. Il est également possible de déclarer un tableau vide et de lui attribuer des valeurs par la suite avec **$*nomTableau*[*index*] = *valeur*;**. La même syntaxe peut être utilisée pour attribuer des valeurs à un tableau associatif : **$*nomTableauAssociatif*[*clé*] = *valeur*;**.

Les Fonctions

En PHP les fonctions sont souvent utilisées. Une fonction est une série d'instructions qui effectue des actions et qui retourne une valeur. Il existe de nombreuses fonctions PHP prédéfinies mais il est également possible de créer ses propres fonctions. Elles se déclarent de la façon suivante :

```
function nomFonction(paramètreFacultatif){
   instructions
}
```

On utilise des fonctions notamment pour factoriser son code et ainsi éviter la redondance lorsque l'on a besoin de répéter du code plusieurs fois. Là où l'on répétait plusieurs fois le même morceau de code pour réaliser une même opération avec des variables différentes par exemple, il sera possible d'écrire une seule fois l'opération et factoriser son code à l'aide d'une fonction. On peut ensuite appeler cette fonction autant de fois que nécessaire. Pour cela il suffit de nommer la fonction que l'on souhaite appeler en lui passant des éventuels paramètres entre les parenthèses : ***nomFonction(paramètreFacultatif);***. Si on veut récupérer le résultat retourné par une fonction il suffit de déclarer une variable et lui attribuer comme valeur l'appel de la fonction : ***$nomVariable = nomFonction(paramètreFacultatif);***. Les fonctions doivent contenir qu'une seule instruction *return*, mais en revanche elles ne doivent pas contenir de *echo*.

Quelques fonctions PHP…

Fonction	Description
echo	Permet l'affichage de variables, chaînes de caractères, nombres…
settype()	Affecte un type à une variable.
array()	Crée un tableau.
isset()	Permet de déterminer l'existence d'une variable.
empty()	Permet de déterminer si une variable est vide.
preg_match()	Recherche une correspondance entre une variable et une expression régulière (RegEx).
time()	Permet d'obtenir le timestamp actuel.
date()	Retourne une date sous un format donné.

Liens utiles

https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql
http://php.net/
https://www.w3schools.com/php7/
https://www.grafikart.fr/tutoriels/php
https://www.grafikart.fr/tutoriels/php-langage-merde-1001
https://pear.php.net/manual/en/standards.php

Bonnes pratiques

Rappel des principales bonnes pratiques	
☐	L'utilisation des simples quotes.
☐	Pas d'HTML dans la fonction echo. Préférer fermer le php, écrire du HTML et rouvrir le PHP.
☐	Indenter son code pour une meilleur compréhension du code.
☐	Utilisation des && et || plutôt que AND et OR.
☐	Un seul return par fonction.
☐	L'utilisation de l'anglais est obligatoire pour les noms de variables, des class, des id…
☐	Utiliser les commentaires.
☐	Vocabulaire spécifique : • Fonction • Variable • Tableau • Boucle • Condition • Instruction • Apache • VHost