Formation PHP X75-1

Table des matières

De quoi traite ce cours ?

Dans ce cours, nous allons explorer et apprendre à utiliser le langage de programmation PHP ainsi que le système de gestion de bases de données MySQL, basé sur le langage SQL. PHP et MySQL forment un duo réputé pour leur puissance et leur simplicité d'utilisation. Ils sont principalement utilisés dans un contexte web, notamment pour manipuler les données envoyées par les utilisateurs et rendre un site dynamique (PHP), ainsi que pour stocker ces données (MySQL).

Quels sont les objectifs du cours?

Ce cours a deux objectifs principaux : présenter les notions et fonctionnalités importantes de PHP et MySQL, afin que vous disposiez des outils et compétences nécessaires pour commencer à programmer dès la fin du cours, et vous rendre le plus autonome possible.

Pour atteindre ces objectifs, nous ferons un tour d'horizon complet des langages, où chaque grande notion sera présentée, étudiée et illustrée. Je vous encouragerai à être proactif. Plutôt que d'éviter les notions complexes, nous les aborderons en profondeur pour que vous compreniez véritablement leur fonctionnement.

De plus, de nombreux exemples et exercices vous seront proposés pour vous confronter aux difficultés et vous assurer de bien comprendre chaque concept. Cette méthode est, selon moi, la meilleure pour vous rendre rapidement autonome. Si vous prenez le temps de refaire les exemples et exercices, vous devriez être capable de réaliser la plupart de vos projets à la fin du cours.

Introduction au cours

Définition et rôle du PHP

Le terme PHP signifie « PHP : Hypertext Preprocessor ». Le premier « P » de PHP est lui-même l'abréviation de « PHP », une particularité qui n'a pas une grande importance pour nous.

Créé en 1994, PHP est un langage de programmation dont la version stable la plus récente est la 8.3, sortie le 24 octobre 2024. C'est sur cette version que se base ce cours. Si une nouvelle version venait à sortir, cela ne rendrait pas ce cours obsolète, car les changements entre deux versions mineures sont limités et la rétrocompatibilité est assurée pendant plusieurs années.

PHP permet de créer des pages web générées dynamiquement. En d'autres termes, grâce à PHP, nous pouvons afficher des contenus différents sur une même page en fonction de certaines variables, comme l'heure de la journée ou si l'utilisateur est connecté ou non.

Prenons l'exemple d'un espace client sur un site e-commerce. Un utilisateur qui a déjà commandé sur le site et créé un compte dispose d'un formulaire de connexion à son espace client. En entrant ses informations (généralement un pseudonyme et un mot de passe), celles-ci sont traitées et analysées par PHP. Si les informations sont correctes, PHP récupère des données spécifiques à cet utilisateur et génère dynamiquement les pages de son espace client, incluant l'historique des commandes, le profil, les informations de facturation et l'adresse de livraison.

Lorsque l'utilisateur fournit des informations comme une adresse ou passe une commande, ces données sont généralement enregistrées dans une base de données. PHP permet également de récupérer ces données pour les réutiliser.

PHP s'exécute côté serveur, ce qui le classe parmi les langages « server side », contrairement aux langages « client side » qui s'exécutent côté client. Nous expliquerons ces notions en détail dans la prochaine leçon.

Sites statiques et sites dynamiques

Les langages de programmation web peuvent être classés en deux grandes catégories :

les langages statiques et les langages dynamiques,

Les sites dits statiques sont caractérisés par leur absence d'interaction et d'adaptation aux visiteurs. Le code des différentes pages reste inchangé, quel que soit l'utilisateur ou toute autre variable. Par exemple, un site de type « CV » ou un site informatif sont généralement statiques, car ils ne nécessitent aucune interaction dynamique avec le visiteur. Un site créé uniquement en HTML et CSS sera toujours statique.

En revanche, les sites dynamiques peuvent fournir des pages différentes pour chaque visiteur ou selon diverses contraintes, permettant ainsi des interactions avec l'utilisateur, comme l'envoi de données. De

nombreux langages permettent de créer des sites dynamiques, chacun ayant ses points forts, ses faiblesses et son champ d’application.

Dans ce cours, nous nous concentrons sur le duo le plus connu : PHP, utilisé pour le calcul et le traitement des données, et MySQL, utilisé pour la gestion des bases de données.

Définition et rôle du MySQL

MySQL est un système de gestion de bases de données relationnelles. Une base de données est un ensemble structuré de données, telles que des informations clients (nom, adresse, mot de passe), des commentaires de blog, ou le texte d’articles.

Les données dans une base de données ne sont pas directement lisibles par un humain. Pour les manipuler, nous utilisons un langage de bases de données, le plus célèbre étant le SQL (Structured Query Language).

MySQL utilise le langage SQL pour manipuler les données des bases de données. Ses avantages incluent sa simplicité d’utilisation, sa fiabilité, et ses performances. MySQL permet également de gérer différents types de bases de données et peut être utilisé conjointement avec PHP.

Prenons un exemple concret : imaginons que nous voulons créer un site où les utilisateurs peuvent s’inscrire et s’identifier. Nous créons des formulaires d’inscription en HTML et récupérons les données en PHP. Ces données doivent être enregistrées dans une base de données, qui n’est rien d’autre qu’un fichier structuré.

Pour créer cette base de données, nous pouvons utiliser une application spécialisée comme phpMyAdmin, dbeaver ou envoyer des requêtes SQL depuis un fichier de code. Nous utilisons une extension PHP (comme PDO ou MySQLi) pour coder en MySQL. Dans notre code, nous écrivons des requêtes SQL pour créer la base de données et y enregistrer les données.

Une base de données créée avec MySQL est une base MySQL(ou mariaDb...), ce qui signifie que ce système de gestion s’occupe de créer le fichier, d’ordonner les données et de les sécuriser. MySQL est un système de gestion de bases de données relationnelles, ce qui signifie que les informations sont stockées dans plusieurs compartiments appelés « tables » qui peuvent communiquer entre elles.

L’idée principale à retenir est que nous ne pouvons pas créer ni manipuler de bases de données sans un système de gestion de bases de données.

MariaDB et MySQL

Un abus de langage existe souvent. On parle souvent de base de donnée mysql pour des bases de données MariaDB.

Un peu d’histoire :

L'histoire de MariaDB est étroitement liée à l'acquisition de MySQL par Oracle.

1. Création deMySQL: MySQL a été créé en 1995 par Michael "Monty" Widenius, David Axmark et Allan Larsson. Ce système de gestion de bases de données relationnelles est rapidement devenu populaire grâce à sa simplicité et son efficacité.
2. Acquisition par Sun Microsystems: En 2008, Sun Microsystems a acquis MySQL pour environ 1 milliard de dollars. Cette acquisition visait à renforcer la position de Sun dans le domaine des logiciels open-source.
3. Rachat par Oracle: En 2010, Oracle a racheté Sun Microsystems, incluant MySQL dans la transaction. Cette acquisition a suscité des inquiétudes parmi les utilisateurs et développeurs de MySQL, craignant qu'Oracle, un concurrent direct, ne compromette l'avenir open-source de MySQL.
4. Création de MariaDB: En réponse à ces préoccupations, Michael "Monty" Widenius a décidé de créer un fork de MySQL en 2009, qu'il a nommé MariaDB, d'après le prénom de sa fille. MariaDB a été conçu pour rester libre et open-source, tout en offrant une compatibilité maximale avec MySQL.
5. Évolution de MariaDB: Depuis sa création, MariaDB a gagné en popularité et est devenu une alternative fiable à MySQL. De nombreuses grandes entreprises et projets, comme Wikipedia et Google, ont migré de MySQL vers MariaDB.

En résumé, MariaDB est né des préoccupations concernant l'avenir de MySQL après son acquisition par Oracle, et il continue de prospérer en tant qu'alternative open-source robuste et compatible.

Les différences :

MariaDB et MySQL sont tous deux des systèmes de gestion de bases de données relationnelles, mais ils présentent quelques différences clés :

1. Licence: MariaDB est entièrement open-source, tandis que MySQL propose des versions open-source et commerciales.
2. Performance: MariaDB est souvent considéré comme plus rapide grâce à un plus grand nombre de moteurs de stockage et une meilleure gestion des connexions.
3. Fonctionnalités: MySQL supporte des fonctionnalités comme le masquage des données et les colonnes dynamiques, ce qui n'est pas le cas de MariaDB.
4. Compatibilité: MariaDB est conçu pour être compatible avec MySQL, facilitant la migration entre les deux.

En résumé, MariaDB offre plus de flexibilité et d'innovation sans verrouillage par le fournisseur, tandis que MySQL est largement utilisé et supporte des fonctionnalités avancées pour la gestion des données.

Pourquoi utiliser le PHP et MySQL ?

Contrairement au HTML et au CSS, qui sont des standards établis, PHP et MariaDB ont de nombreux concurrents : Python, Ruby, JavaScript,... pour PHP, et PostgreSQL, Microsoft SQL Server, SQLite,... pour MySQL, pour ne citer qu'eux.

Pourquoi alors préférer le couple PHP/MySQL aux autres langages ? Il n'y a pas de raison absolue, car les alternatives mentionnées sont également performantes et présentent certains avantages et inconvénients par rapport à PHP et MySQL.

Cependant, le couple PHP/MySQL reste de loin le plus célèbre et le choix de référence pour créer des sites dynamiques et stocker des données, et ce pour de bonnes raisons.

Le premier avantage de PHP réside dans sa structure : c'est un langage très accessible pour les débutants, qui peuvent rapidement en comprendre la syntaxe de base et réaliser leurs premiers scripts, tout en supportant des structures très complexes pour les utilisateurs avancés.

De plus, PHP est un langage open source et donc gratuit, ce qui n'est pas toujours le cas dans le monde du web. Il est universellement reconnu et supporté, fonctionnant sur la majorité des architectures techniques.

PHP se distingue également par ses performances et sa solidité. Étant open source, il bénéficie des contributions de nombreux développeurs, ce qui le rend constamment amélioré et jamais abandonné. Il offre de bonnes performances d'exécution en termes de rapidité et est un langage sûr : les rares failles détectées sont corrigées rapidement.

Les systèmes de gestion de bases de données sont nombreux et la plupart utilisent le SQL standard. J'ai choisi d'utiliser MySQL dans ce cours car c'est le choix le plus populaire parmi les développeurs, et ce pour de bonnes raisons.

MySQL est totalement compatible avec PHP et utilise une syntaxe SQL standard, facilitant les opérations si vous devez changer de système de gestion de bases de données. De plus, MySQL est simple d'utilisation, très robuste et offre d'excellentes performances, que ce soit pour une petite ou une grande structure.

Client et serveur : définitions et interactions

Nous allons définir ce qu'est un « client » et ce qu'est un « serveur » et examiner les grandes différences entre les langages dits « client side » et « server side ».

Le fonctionnement d’Internet et du Web

Internet est un système conçu pour transporter de l’information. C’est un réseau de réseaux utilisant divers protocoles (règles définissant comment formater, transmettre et recevoir des données) pour envoyer des informations.

Le World Wide Web, ou simplement « Web », est l’un des réseaux d’Internet. Le Web n’est donc qu’une partie d’Internet. Plus précisément, le Web est un réseau de machines interconnectées qui stockent des sites. Lorsqu’une machine est connectée au Web et fournit un accès à un site web, on l’appelle un serveur car elle « sert » le site web.

Un serveur est une sorte de super ordinateur, fonctionnant 24h/24 et 7j/7 (en théorie). Un serveur dispose de logiciels spécifiques et son rôle est de stocker toutes sortes de médias composant les sites (fichiers, images, etc.), et de les rendre accessibles à tout utilisateur à tout moment, où qu’il soit.

Pour pouvoir se comprendre et échanger des données, toutes ces machines doivent parler la même langue, c’est-à-dire utiliser le même protocole. Le Web repose ainsi sur le protocole HTTP (HyperText Transfer Protocol) et sur son frère sécurisé, le HTTPS (Secure HTTP). Pour utiliser ce protocole, nous devons passer par un navigateur web (Chrome, Safari, etc.) que l’on appelle alors un « client HTTP ».

Pour accéder directement à une page web, on passe par un navigateur en utilisant le protocole HTTP. On fournit une adresse au format spécial à notre navigateur : une URL (Uniform Resource Locator) qui sert à identifier une page web de manière unique. Ici, notre navigateur (et nous) sommes des « clients » car nous demandons à accéder à la page web.

Le navigateur va alors chercher où se trouve le serveur hébergeant la page demandée. Pour cela, il utilise un service de DNS (Domain Name Server), des serveurs permettant d’associer un nom de domaine (mon-site-web.com par exemple) à une adresse IP unique. Chaque fournisseur de services Internet fournit une liste d’adresses de DNS à contacter. Si le premier DNS ne reconnaît pas le site, il envoie la demande à d’autres DNS, et ainsi de suite jusqu’à ce qu’un DNS possède le site dans sa liste de noms.

L’adresse IP liée au site est alors renvoyée au serveur. L’IP (Internet Protocol) est une suite de nombres permettant d’identifier de manière unique une machine connectée à Internet. Chaque machine possède sa propre IP, qui change en fonction du réseau sur lequel elle est connectée (l’IP est attribuée par le fournisseur de services Internet).

Le navigateur possède maintenant l’adresse IP de notre site et donc l’adresse de la machine (le fameux « serveur ») sur laquelle il est stocké. Il peut ainsi contacter directement le serveur en utilisant le protocole HTTP pour lui demander de renvoyer la page en question et envoie également notre IP pour que le serveur sache à quelle adresse renvoyer la page demandée.

Lorsque le serveur reçoit la requête, il recherche immédiatement le fichier demandé, effectue éventuellement certaines opérations dessus et le renvoie au navigateur, ou renvoie un code d’erreur si le fichier demandé est introuvable ou ne peut pas être envoyé.

Langages client-side et server-side

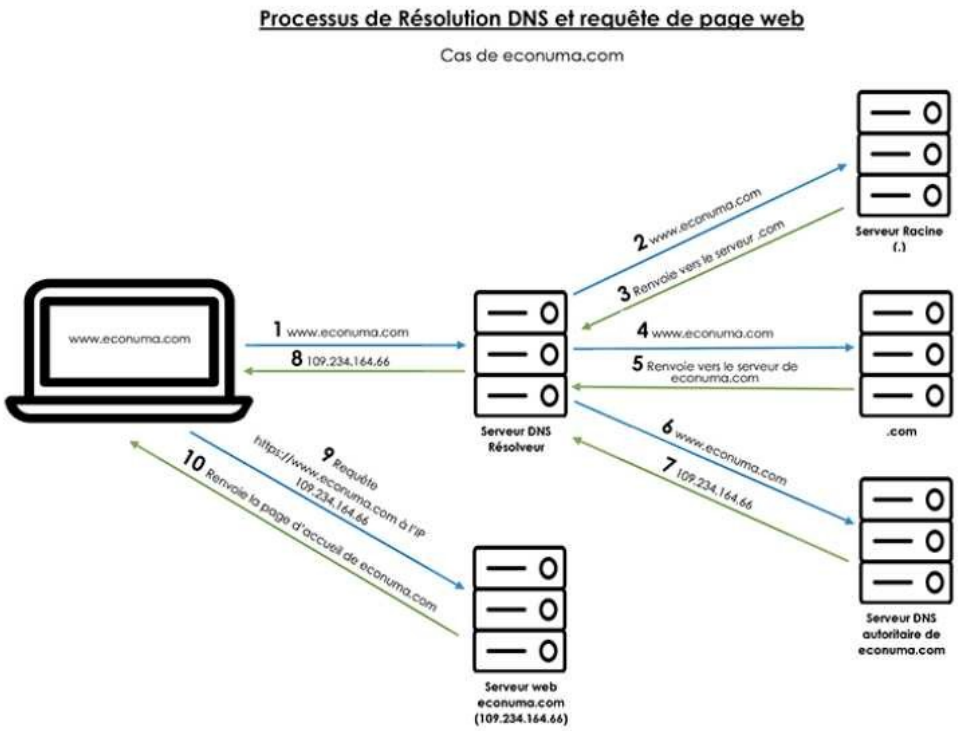
Un site Internet est un ensemble de fichiers de code liés entre eux, faisant éventuellement appel à des ressources ou médias comme des images. Le code écrit dans ces fichiers, appelé « script », peut être exécuté soit côté client (client-side), c'est-à-dire directement dans le navigateur de l'utilisateur, soit côté serveur (server-side).

Il est important de comprendre que les navigateurs (côté client) et les serveurs (côté serveur) ne peuvent effectuer que certaines opérations et lire certains langages. La majorité des navigateurs ne comprennent et n'exécutent que du code HTML, CSS et JavaScript. Un navigateur est donc incapable de comprendre du code PHP.

Lorsqu'un navigateur demande à un serveur de lui fournir une page, le serveur exécute tout code incompréhensible pour le navigateur (en utilisant au besoin un interpréteur). Une fois ces opérations effectuées, le serveur renvoie le résultat sous forme de code compréhensible par le navigateur, principalement en HTML. Le navigateur affiche alors la page au visiteur, unique car générée par le serveur pour un utilisateur spécifique en tenant compte de données particulières.

Si la page demandée ne contient aucun code nécessitant l'intervention du serveur, le serveur la renvoie telle quelle au navigateur, qui l'affiche au visiteur.

Les opérations réalisées côté serveur sont transparentes pour le visiteur, qui n'a jamais accès au code exécuté côté serveur. Une fois les opérations terminées, le serveur ne renvoie que du code compréhensible par le navigateur. C'est pourquoi, en analysant le code d'une page, vous ne trouverez jamais d'instructions PHP, mais seulement du code HTML, CSS et JavaScript, des langages exécutés côté client.



Mise en place de notre environnement de travail

Pour coder en HTML et en CSS et afficher le résultat, un simple éditeur de texte et un navigateur suffisent. Pour le PHP et MySQL, cependant, c'est un peu plus complexe car le code s'exécute côté serveur.

Dans cette leçon, nous allons explorer les différentes options pour coder en PHP et MySQL, et voir quels logiciels installer et pourquoi.

Le travail en local et le travail en production

Lorsque l'on code, on peut travailler soit en local, c'est-à-dire en hébergeant tous les fichiers sur nos propres machines ou hors ligne, soit en production (ou en préproduction), c'est-à-dire sur des fichiers hébergés sur un serveur distant et potentiellement accessibles à tous, en direct.

Lors de la phase de développement d'un site, ou pendant les phases de test ou de débogage, un bon développeur préfère toujours travailler en local ou en préproduction pour ne pas perturber le fonctionnement normal d'un site web.

La préproduction est une copie d'un site également hébergée sur serveur, généralement accessible uniquement aux développeurs pour tester leurs modifications en toute sécurité et en conditions réelles.

Dans cette formation nous travaillerons en local. Cependant, rappelez-vous que PHP et MySQL s'exécutent côté serveur. Il nous faut donc recréer une architecture serveur sur nos machines avec les logiciels adaptés pour tester nos codes PHP. Rassurez-vous, c'est très simple et totalement gratuit.

Recréer une architecture serveur sur son ordinateur

Un serveur dispose de différents programmes lui permettant de lire et de comprendre certains langages informatiques que des ordinateurs « normaux » ne peuvent pas lire. Nous allons donc devoir installer des programmes similaires afin de pouvoir tester nos codes PHP et MySQL.

Vous pouvez installer MAMP, disponible à l'adresse http://www.mamp.info/

Mamp est un serveur web de test complet (LAMP = Linux, Apache, Mysql, Php)

Si vous êtes sous Mac Os ou Linux. Il est possible d'installer Apache2, Mysql et Php directement.

Si vous utilisez un autre logiciel LAMP (Wamp, Xamp,..) C'est très bien, inutile d'installer Mamp.

Où écrire le code PHP ?

Nous allons pouvoir écrire nos scripts PHP soit dans des fichiers dédiés, c'est-à-dire des fichiers qui ne vont contenir que du PHP, soit intégrer le PHP au sein de nos fichiers HTML mais qui devrons être enregistré avec l'extension .php et pas html.

Soyons clair, melanger du HTML et du CSS est un non sens aujourd'hui et ne se pratique plus depuis 15ans. Aujourd'hui il n'est plus envisageable de développer un projet en « php procédurale », on ne fait plus que du « php objet » en respectant le modèle MVC.

Malgré cela, nous ferons du procédurale le temps de l'initiation de base.

Le modèle MVC ?

Le modèle MVC (Modèle-Vue-Contrôleur) est une architecture utilisée pour organiser et structurer le code dans les applications web. Il permet de séparer les préoccupations en trois composants distincts, rendant les applications plus modulaires, plus faciles à maintenir et à étendre. Voici une explication de chaque composant dans le contexte de PHP :

1. Modèle (Model):

    1.1. Rôle: Le modèle représente la couche des données et la logique métier de l'application. Il gère l'accès aux données et les interactions avec la base de données.

    1.2. Exemple** : En PHP, le modèle pourrait être un fichier de code qui interagit avec une base de données MySQL pour récupérer, ajouter ou supprimer des données.

2. Vue (View):

2.1. Rôle: La vue est responsable de l'affichage des données aux utilisateurs. Elle récupère les données du modèle et les présente dans un format compréhensible, souvent en HTML.

2.2. Exemple: Une vue en PHP pourrait être un fichier HTML avec des balises PHP intégrées pour afficher dynamiquement les données récupérées par le modèle.

3. Contrôleur (Controller):

3.1. Rôle: Le contrôleur agit comme un intermédiaire entre le modèle et la vue. Il reçoit les requêtes du client, interagit avec le modèle pour récupérer les données nécessaires, et transmet ces données à la vue pour l'affichage.

3.2. Exemple** : Un contrôleur en PHP pourrait être une classe qui traite les requêtes HTTP, appelle les méthodes du modèle pour manipuler les données, et sélectionne la vue appropriée pour afficher les résultats.

Pourquoi utiliser l'architecture MVC en PHP ?

- Séparation des préoccupations: En séparant la logique métier, l'affichage et le contrôle en composants distincts, le MVC améliore la lisibilité et la maintenabilité du code.
- Réutilisation du code: Le MVC permet de réutiliser les modèles et les vues dans différentes parties de l'application.
- Facilité de test: Le MVC permet de tester chaque composant individuellement, simplifiant le débogage et l'amélioration continue de l'application.
- Collaboration efficace: Les développeurs peuvent travailler sur différents composants sans interférer avec le travail des autres, améliorant ainsi la productivité.

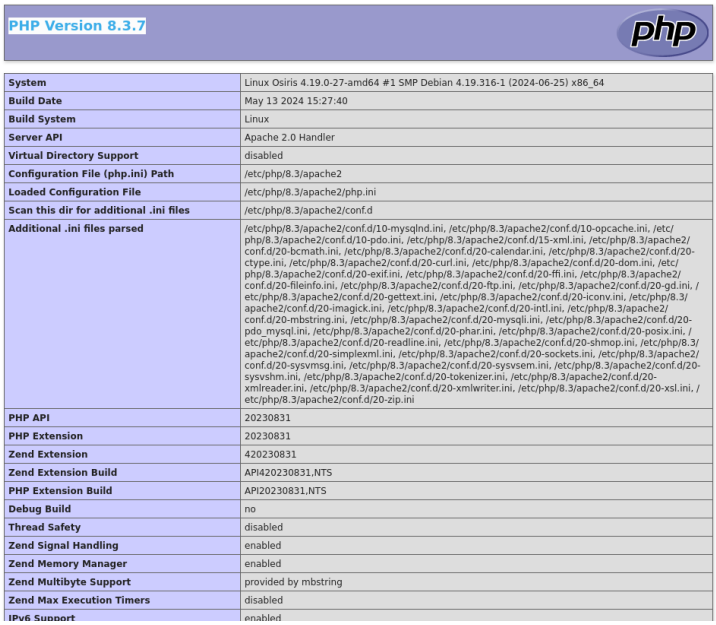
Testons notre serveur

Attention que les fichiers php devront être dans un répertoire dont le parent est le répertoire de Mamp. Il est possible de le connaître en allant dans la configuration de MAMP.

Créons notre premier fichier php. A la racine du dossier de Mamp créons le fichier phpInfo.php

```php
<?php
phpinfo();
```

2 lignes pour un résultat bien long…



PHP Version 8.3.7

System	Linux Osiris 4.19.0-27-amd64 #1 SMP Debian 4.19.316-1 (2024-06-25) x86_64
Build Date	May 13 2024 15:27:40
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.3/apache2
Loaded Configuration File	/etc/php/8.3/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.3/apache2/conf.d
Additional .ini files parsed	/etc/php/8.3/apache2/conf.d/10-mysqlnd.ini, /etc/php/8.3/apache2/conf.d/10-opcache.ini, /etc/php/8.3/apache2/conf.d/10-pdo.ini, /etc/php/8.3/apache2/conf.d/15-xml.ini, /etc/php/8.3/apache2/conf.d/20-bcmath.ini, /etc/php/8.3/apache2/conf.d/20-calendar.ini, /etc/php/8.3/apache2/conf.d/20-ctype.ini, /etc/php/8.3/apache2/conf.d/20-curl.ini, /etc/php/8.3/apache2/conf.d/20-dom.ini, /etc/php/8.3/apache2/conf.d/20-exif.ini, /etc/php/8.3/apache2/conf.d/20-ffi.ini, /etc/php/8.3/apache2/conf.d/20-fileinfo.ini, /etc/php/8.3/apache2/conf.d/20-ftp.ini, /etc/php/8.3/apache2/conf.d/20-gd.ini, /etc/php/8.3/apache2/conf.d/20-gettext.ini, /etc/php/8.3/apache2/conf.d/20-iconv.ini, /etc/php/8.3/apache2/conf.d/20-imagick.ini, /etc/php/8.3/apache2/conf.d/20-intl.ini, /etc/php/8.3/apache2/conf.d/20-mbstring.ini, /etc/php/8.3/apache2/conf.d/20-mysqli.ini, /etc/php/8.3/apache2/conf.d/20-pdo_mysql.ini, /etc/php/8.3/apache2/conf.d/20-phar.ini, /etc/php/8.3/apache2/conf.d/20-posix.ini, /etc/php/8.3/apache2/conf.d/20-readline.ini, /etc/php/8.3/apache2/conf.d/20-shmop.ini, /etc/php/8.3/apache2/conf.d/20-simplexml.ini, /etc/php/8.3/apache2/conf.d/20-sockets.ini, /etc/php/8.3/apache2/conf.d/20-sysvmsg.ini, /etc/php/8.3/apache2/conf.d/20-sysvsem.ini, /etc/php/8.3/apache2/conf.d/20-sysvshm.ini, /etc/php/8.3/apache2/conf.d/20-tokenizer.ini, /etc/php/8.3/apache2/conf.d/20-xmlreader.ini, /etc/php/8.3/apache2/conf.d/20-xmlwriter.ini, /etc/php/8.3/apache2/conf.d/20-xsl.ini, /etc/php/8.3/apache2/conf.d/20-zip.ini
PHP API	20230831
PHP Extension	20230831
Zend Extension	420230831
Zend Extension Build	API420230831,NTS
PHP Extension Build	API20230831,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
Zend Max Execution Timers	disabled
IPv6 Support	enabled

la ligne phpinfo() ; est une méthode qui vous renvoi la configuration de php sur votre machine. Verifiez la version (8.3 ou suivante). Pour l'instant c'est suffisant. La majorité des données viennent du fichier de configuration de php > php.ini. Il vous faudra certainement modifier des choses dedans sur des projets plus complexes.

La balise PHP

Le serveur, pour être en mesure d'exécuter le code PHP, va devoir le reconnaître. Pour lui indiquer qu'un script ou que telle partie d'un code est écrit en PHP, nous allons entourer ce code avec une balise PHP qui a la forme suivante : <?php ?>. Lorsqu'on intègre du PHP dans du code HTML, on va pouvoir placer cette balise et du code PHP à n'importe quel endroit dans notre fichier. On va même pouvoir placer la balise PHP en dehors de notre élément html. De plus, on va pouvoir déclarer plusieurs balises PHP à différents endroits dans un fichier.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Premiere page php</title>
</head>
<body>
<?php


?>
<h1>Ceci est une page dans un fichier .php</h1>
<?php


?>
<p class="relatif gauche">Bonjour à tous,</p>
<p>On ne lâche rien<p>
<hr>
<p>Bonne continuation...</p>
</body>
</html>
```

Ceci est une page dans un fichier .php

Bonjour à tous,

On ne lâche rien

Bonne continuation...

Évidemment, si vous enlever les 2 balises php, vous aurez le meme résultat. C'est juste du HTML.

Syntaxe PHP de base

Écrivons une première instruction PHP afin d'analyser et de comprendre la syntaxe générale du PHP.

```
<h1>Ceci est une page dans un fichier .php</h1>

<p>
<!--    ouvrons la balise PHP-->
    <?php
//    Bonjour à tous, **** On garde cette partie de code au cas ou...
    echo"Hello World,";
    /*
     Changement du texte pour voir une difference
    Avec l'affichage d'avant
     */
    ?>
</p>
<p>On ne lâche rien<p>
<hr>
<p>Bonne continuation...</p>
```

Analysons cela :

A l'affichage, rien n'a fort changé. Un des texte à été modifié mais rien de plus.

- L'instruction ECHO demande à php de renvoyer le contenu à l'explorateur.
- Les instructions PHP sont TOUJOURS suivies par un point virgule ;-)

Notez les 3 commentaires.

- Le premier est en dehors des balises php. C'est un commentaire HTML.
- Le deuxième est un commentaire mono-ligne php
- Le troisième un commentaire multi-lignes php

Demandons à notre explorateur de voir le code source de la page :

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Premiere page php</title>
6  </head>
7  <body>
8
9  <h1>Ceci est une page dans un fichier .php</h1>
10
11 <p>
12 <!--    ouvrons la balise PHP-->
13     Hello World,</p>
14 <p>On ne lâche rien<p>
15 <hr>
16 <p>Bonne continuation...</p>
17 </body>
18 </html>
```

On voit bien le commentaire HTML mais pas les commentaires PHP ! Les commentaires php ne sont pas envoyés à l'explorateur.

(Chrome ne montre pas les commentaires HTML non plus… Mais ils sont bien là!)

ECHO et PRINT

Les 2 sont fortement identiques. L’usage d’echo est préféré

```php
<?php

echo '<p>Bonjour à tous</p>';
echo "<p>On ne lâche rien<p>";
echo 12;
echo " ou ";
echo "13";
echo " personnes sur ", 24, " sont présentes";
echo "<p>On ne lâche rien<p>";

?>
```

Bonjour à tous

On ne lâche rien

12 ou 13 personnes sur 24 sont présentes

On ne lâche rien

Première et deuxième ligne : le contenu peut être entouré d’une simple ou d’un double guillemet.

Troisième et cinquième ligne : un nombre peut être entouré ou pas de guillemets.

Sixième ligne : les contenus peuvent être séparés par des virgules.

Les variables Php

Qu’est-ce qu’une variable ?

Une variable est un conteneur utilisé pour stocker temporairement des informations, telles qu’une chaîne de caractères (texte) ou un nombre. La particularité d’une variable est qu’elle peut changer, c’est-à-dire qu’elle peut contenir différentes valeurs au fil du temps.

En PHP, les variables ne stockent les informations que temporairement. Plus précisément, une variable n’existe que pendant l’exécution du script qui l’utilise. Ainsi, les variables ne permettent pas de stocker des informations de manière durable. Pour cela, nous utiliserons des fichiers, des cookies ou des bases de données, que nous aborderons plus tard dans ce cours.

Note : Au début de ce cours, nous définirons nous-mêmes les valeurs à stocker dans nos variables, ce qui peut sembler peu utile en pratique. Il est donc normal que vous ne voyiez pas immédiatement l’intérêt d’utiliser des variables. Retenez que les variables deviennent vraiment intéressantes lorsqu’elles servent à stocker des données envoyées par les utilisateurs (via des formulaires, par exemple), pour effectuer les comparaisons, des calculs car nous pourrons ensuite manipuler ces données.

Les règles de déclaration des variables en PHP

Une variable est donc un conteneur ou un espace de stockage temporaire auquel nous pouvons assigner une valeur. Nous pouvons créer différentes variables dans un script pour stocker diverses valeurs et les réutiliser facilement par la suite. Lorsqu’on crée une variable en PHP, on dit qu’on la « déclare ».

Nous pouvons choisir le nom de nos variables, mais il y a quelques règles à respecter lors de leur déclaration :

- Toute variable en PHP doit commencer par le signe $ suivi du nom de la variable.
- Le nom d’une variable doit obligatoirement commencer par une lettre ou un underscore (_) et ne doit pas commencer par un chiffre.
- Le nom d’une variable ne doit contenir que des lettres, des chiffres et des underscores, mais pas de caractères spéciaux.
- Le nom d’une variable ne doit pas contenir d’espace.

Bonnes pratiques :

Une bonne pratique à prendre pour nommer les variables est d’indiquer clairement ce qu’elle représente en nom de variable et d’écrire cela en « camelCase ». Cela consiste à écrire un ensemble de mots sans espaces ni ponctuation, où chaque nouveau mot débute par une majuscule, à l'exception du premier mot qui commence par une minuscule.

Exemple : une variable qui represente l’age du capitaine : $ageDuCapitaine. Bien que, en pratique, l’usage de l’anglais est préféré $ageOfTheCaptain".

es sont sensibles à la casse en PHP. Cela signifie que l'utilisation de majuscules ou de minuscules crée des variables différentes. Par exemple, les variables $texte, $TEXTE et $tEXTe sont distinctes.

Enfin, sachez qu'il existe des noms « réservés » en PHP. Vous ne pouvez pas utiliser ces noms pour vos variables, car le langage PHP les utilise déjà pour désigner différents objets intégrés.

Déclaration de variables en php (à noter que cet exemple n'affiche rien!)

```php
<?php
    $prenom = "Pierre";
    $age = 28;
?>
```

Affichons une variable :

```php
<?php

$onLacheRien= 'On ne lâche rien';

echo '<p>Bonjour à tous</p>';
echo "<p>";

echo $onLacheRien;

echo "<p>";

?>
```

Bonjour à tous

On ne lâche rien

Utilisation concrète des variables en PHP

Les variables en PHP sont extrêmement utiles dans de nombreuses situations. Elles permettent de manipuler des données dont les valeurs ne sont pas connues à l'avance. Par exemple, si vous souhaitez manipuler (afficher, stocker, etc.) des données récoltées via un formulaire rempli par vos visiteurs, vous ne connaissez pas les valeurs qui seront envoyées à l'avance.

Pour manipuler ces données, nous utilisons des variables. Nous créons un script qui traite les données envoyées par les utilisateurs. Les valeurs envoyées sont stockées dans des variables, que nous pouvons ensuite manipuler. Ainsi, chaque fois qu'un visiteur envoie le formulaire, notre script se déclenche, les données sont placées dans des variables prédéfinies, et le script manipule ces variables pour les afficher, les stocker, etc.

Les variables sont également essentielles pour dynamiser notre site grâce à leur capacité à stocker différentes valeurs. Par exemple, si nous souhaitons afficher une horloge sur notre site, nous créons un

script qui recalcule l'heure actuelle toutes les secondes. Cette heure est placée dans une variable `$heure`. Le script met à jour cette variable chaque seconde (la variable stocke alors l'heure actuelle) et affiche son contenu.

Nous pouvons utiliser les variables dans de nombreuses autres situations pour créer des scripts plus complexes. Cependant, il serait difficile d'illustrer l'intérêt des variables avec des exemples trop avancés à ce stade.

Pour l'instant, faites confiance au processus et concentrez-vous sur la compréhension de chaque notion présentée. Ces concepts seront très utiles par la suite.

Les types de données PHP

Les variables PHP vont pouvoir stocker différents types de valeurs, comme du texte ou un nombre par exemple. Par abus de langage, nous parlerons souvent de « types de variables » PHP.

En PHP, contrairement à d'autres langages de programmation, nous n'avons pas besoin de préciser à priori le type de valeur qu'une variable va pouvoir stocker. Le PHP va en effet automatiquement détecter quel est le type de la valeur stockée dans telle ou telle variable, et nous allons ensuite pouvoir performer différentes opérations selon le type de la variable, ce qui va s'avérer très pratique pour nous !

Une conséquence directe de cela est qu'on va pouvoir stocker différents types de valeurs dans une variable au fil du temps sans se préoccuper d'une quelconque compatibilité. Par exemple, une variable va pouvoir stocker une valeur textuelle à un moment dans un script puis un nombre à un autre moment.

Les variables en PHP vont pouvoir stocker 8 grands types de données différents :

- Le type « chaine de caractères » ou String en anglais ;
- Le type « nombre entier » ou Integer en anglais ;
- Le type « nombre décimal » ou Float en anglais ;
- Le type « booléen » ou Boolean(bool) en anglais ;
- Le type « tableau » ou Array en anglais ;
- Le type « objet » ou Object en anglais ;
- Le type « NULL » qui se dit également NULL en anglais ;
- Le type « ressource » ou Resource en anglais ;

Nous allons pour le moment nous concentrer sur les types simples de valeurs.

Le type chaîne de caractères ou String

Le premier type de données qu'une variable va pouvoir stocker est le type String ou chaîne de caractères. Une chaine de caractères est une séquence de caractères, ou ce qu'on appelle communément un texte.

Notez que toute valeur stockée dans une variable en utilisant des guillemets ou des apostrophes sera considérée comme une chaine de caractères, et ceci même dans le cas où nos caractères sont à priori des chiffres comme « 28 » par exemple. Il serait impossible de multiplier 28 par deux si 28 est de type string.

```php
$string = "texte";
$string = "10";
```

Les types de données nombre entier (Integer) et nombre décimal (Float ou Double)

En PHP, on va pouvoir stocker deux types différents de donnée numéraires dans nos variables : le type Integer, qui contient tous les nombres entiers positifs ou négatifs et le type Float ou Double, qui contient les nombres décimaux (nombres à virgule) positifs ou négatifs.

```php
$nombre = 10;
```

Le type de données booléen (Boolean)

Une variable en PHP peut encore stocker une valeur de type booléen (Boolean en anglais). Le type booléen est un type qui ne contient que deux valeurs : les valeurs true (vrai) et false (faux). Ce type n'est pas courant dans la vie de tous les jours mais est très (très) utilisé en informatique.

```php
/* SANS GUILEMETS*/
$bool = true;
$bool = false;
```

Le type de données Null

Le type de données Null est un type un peu particulier puisqu'il correspond à l'absence de valeur et sert donc à représenter des variables vides en PHP. Ce type de valeur ne contient qu'une seule valeur : la valeur NULL qui correspond elle-même à l'absence de valeur.

Par exemple cela permet de faire un distinction entre une chaine de caractère vide et l'absence de valeur.

```php
/* SANS GUILEMETS*/
$null = null;
```

Les types de données PHP tableau (Array) et objet (Object)

Les types de données Array et Object sont des types de données complexes particuliers qui méritent de faire chacun l'objet de chapitres séparés.

Pour l'instant sachez simplement que l'on va pouvoir stocker plusieurs valeurs d'un coup à l'intérieur d'une variable en lui assignant des valeurs de type Array (tableau) ou Object (objet).

Le type de données ressource (Resource)

Une ressource est une variable particulière qui contient une référence vers une ressource externe au PHP, comme dans le cas d'une variable qui représente la connexion vers une base de données par exemple.

Les opérateurs de chaines et la concaténation en PHP

Concaténer signifie littéralement « mettre bout à bout ». Pour indiquer qu'on souhaite concaténer, c'est-à-dire qu'on souhaite mettre bout à bout deux chaines de caractères en PHP on utilise le point (.) qu'on appelle également « opérateur de concaténation ».

Pour bien comprendre comment fonctionne l'opérateur de concaténation et son intérêt, il me semble nécessaire de connaitre les différences entre l'utilisation des guillemets et des apostrophes lorsqu'on manipule une chaine de caractères en PHP.

La différence majeure entre l'utilisation des guillemets et des apostrophes est que tout ce qui est entre guillemets va être interprété tandis que quasiment tout ce qui est entre apostrophes va être considéré comme une chaine de caractères. Ici, « interprété » signifie « être remplacé par sa valeur ». Ainsi, lorsqu'on inclut une variable au sein d'une chaine de caractères et qu'on cherche à afficher le tout avec un echo par exemple en entourant la chaine complète avec des guillemets, la variable va être remplacée par sa valeur lors de l'affichage.

En revanche, lorsqu'on utilise des apostrophes, les variables ne vont pas être interprétées mais leur nom va être considéré comme faisant partie de la chaine de caractères.

Regardez plutôt l'exemple suivant :

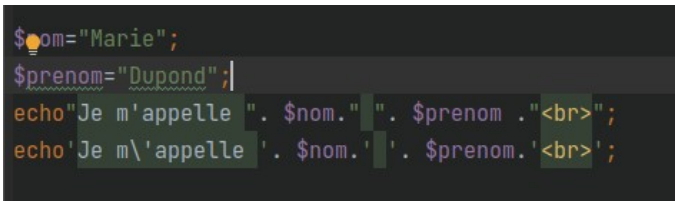
```php
$nom="Marie";
$prenom="Dupond";
echo"Je m'appelle $nom $prenom <br>";
echo"Je m'appelle {$nom} {$prenom} <br>";
echo'Je m\'appelle $nom $prenom<br>';
echo'Je m\'appelle {$nom} {$prenom}';
```

```
Je m'appelle Marie Dupond
Je m'appelle Marie Dupond
Je m'appelle $nom $prenom
Je m'appelle {$nom} {$prenom}
```

Pour concaténer correctement avec l'opérateur de concaténation, la règle est de séparer les différentes variables avec l'opérateur de concaténation (le point) des textes autour.

Chaque texte devra être entouré de guillemets ou d'apostrophes selon ce qu'on a choisi.

L'utilisation de la concaténation peut sembler inutile mais son usage est prioritaire.



Je m'appelle Marie Dupond
Je m'appelle Marie Dupond

Les opérateurs arithmétiques

Les opérateurs arithmétiques vont nous permettre d'effectuer toutes sortes d'opérations mathématiques entre les valeurs contenues dans différentes variables lorsque ces valeurs sont des nombres.

Le fait de pouvoir réaliser des opérations entre variables va être très utile dans de nombreuses situations. Par exemple, si un utilisateur commande plusieurs produits sur notre site ou plusieurs fois un même produit et utilise un code de réduction, il faudra utiliser des opérations mathématiques pour calculer le prix total de la commande.

En PHP, nous allons pouvoir utiliser les opérateurs arithmétiques suivants :

Opérateur	Nom de l'opération associée
+	Addition
–	Soustraction
*	Multiplication
/	Division
%	Modulo (reste d'une division euclidienne)
**	Exponentielle (élévation à la puissance d'un nombre par un autre)

Pour info :

- Le modulo correspond au reste entier d'une division euclidienne. Par exemple, lorsqu'on divise 5 par 3, le résultat est 1 et il reste 2 dans le cas d'une division euclidienne. Le reste, 2, correspond justement au modulo.
- L'exponentielle correspond à l'élévation à la puissance d'un nombre par un autre nombre. La puissance d'un nombre est le résultat d'une multiplication répétée de ce nombre par lui-même. Par exemple, lorsqu'on souhaite calculer 2 à la puissance de 3 (qu'on appelle également « 2

exposant 3 »), on cherche en fait le résultat de 2 multiplié 3 fois par lui-même c’est-à-dire 2*2*2 = 8.

By the way : Petit rappel utile sur l’ordre des operations en mathématique

Voici l'ordre de priorité des opérations qu'il faut respecter :

1. Les Parenthèses
2. Les Exposants
3. Les Multiplications et les Divisions (de la gauche vers la droite)
4. Les Additions et les Soustractions (de la gauche vers la droite)

Exercices variables

Utilisez la fonction var_dump() Pour afficher les détails de vos variables

```php
$variable1 = 5;
$variable2 = "chaise";
$variable3 = true;
$variable4 = null;
echo '<br> variable1 vaut: ';
var_dump($variable1);
echo '<br> variable2 vaut: ';
var_dump($variable2);
echo '<br> variable3 vaut: ';
var_dump($variable3);
echo '<br> variable4 vaut: ';
var_dump($variable4);
```

variable1 vaut: int(5)
variable2 vaut: string(6) "chaise"
variable3 vaut: bool(true)
variable4 vaut: NULL

variable1 vaut : un entier de valeur 5

variable2 vaut : une chaine de caractère de 6 caractères et de valeur « chaise »

variable3 vaut : un boolean de valeur « vrai » (ou 1)

variable4 vaut : « rien » pas de type défini

Exercice 1: Créer une variable name et l'initialiser avec la valeur de votre choix. Afficher son contenu.

Exercice 2: Créer trois variables lastname, firstname et age et les initialiser avec les valeurs de votre choix. Attention age est de type entier. Afficher leur contenu.

Exercice 3 :Créer une variable km. L'initialiser à 1. Afficher son contenu. Changer sa valeur par 3. afficher son contenu. Changer sa valeur par 125. Afficher son contenu.

Exercice 4 :Créer une variable de type string, une variable de type int, une variable de type float, une variable de type booléan et les initialiser avec une valeur de votre choix. Les afficher.

Exercice 5 : Créer trois variables lastname , firstname et age et les initialiser avec les valeurs de votre choix. Attention age est de type entier. Afficher : "Bonjour" + lastname + firstname + ",tu as" + age + "ans".

Exercice 6 :Créer 3 variables. La 1er vaut 3, la 2eme vaut 4, la 3eme vaut 5. Afficher le résultat que vaut 3+4, 5/4 et 3+5*4

Exercice 7 : Afficher le résultat de (8+2×2)÷(12÷4+3)

Solutions : variables_resolu.php

Les opérateurs d’affection et de comparaison (introduction)

Une nouvelle fois, vous devez bien comprendre que le signe égal simple utilisé ci-dessus n’est pas un opérateur de comparaison mais bien un opérateur d’affection (ou d’assignation) : il sert à affecter une valeur à une variable. Cela signifie que l’opérateur = ne représente pas l’égalité d’un point de vue mathématique.

L’égalité en termes de valeurs simples est symbolisée en PHP par le double signe égal : ==. L’égalité en termes de valeurs et de types de données, c’est-à-dire l’identité, va être représentée en PHP par le triple signe égal : ===.

En effet, nous allons voir plus tard que les variables peuvent stocker différents types de données : des chaînes de caractères, des nombres entiers, des nombres décimaux, etc. En utilisant des guillemets ou des apostrophes, on indique que la valeur stockée par la variable est une chaîne de caractères.

Exemple

$a = « 3 »	Assigne la valeur à « 3 » type string
$a == 3	Vaut true
$a === 3	Vaut false (la chaine de caractère 3 ne vaut pas le chiffre 3)

Structures de contrôle

Les conditions et des opérateurs de comparaison

Une structure de contrôle est un ensemble d'instructions qui permet de contrôler l'exécution du code.

Il existe différents types de structures de contrôle. Les deux types les plus connus et les plus utilisés sont les structures de contrôle conditionnelles qui permettent d'exécuter un bloc de code si une certaine condition est vérifiée et les structures de contrôle de boucle qui permettent d'exécuter un bloc de code en boucle tant qu'une condition est vérifiée.

Présentation des opérateurs de comparaison

Nous allons souvent construire nos conditions autour de variables : selon la valeur d'une variable, nous allons exécuter tel bloc de code ou pas. En pratique, nous allons donc comparer la valeur d'une variable à une certaine autre valeur donnée et selon le résultat de la comparaison exécuter un bloc de code ou pas. Pour comparer des valeurs, nous allons devoir utiliser des opérateurs de comparaison.

Opérateur Définition :

==	Permet de tester l'égalité sur les valeurs
===	Permet de tester l'égalité en termes de valeurs et de types
!=	Permet de tester la différence en valeurs
<>	Permet également de tester la différence en valeurs
!==	Permet de tester la différence en valeurs ou en types
<	Permet de tester si une valeur est strictement inférieure à une autre
>	Permet de tester si une valeur est strictement supérieure à une autre
< =	Permet de tester si une valeur est inférieure ou égale à une autre
> =	Permet de tester si une valeur est supérieure ou égale à une autre

exercices :

Exercice1

$a = 3;

il faut que $a ... 3 soit true. Trouvez 4 comparaisons renvoyant la valeur true

var_dump($a ... 3);

var_dump($a ... 3);

var_dump($a ... 3);

var_dump($a ... 3);

Exercice 2

$a = "3";

il faut que $a … 3 soit true

var_dump($a … 3);

var_dump($a ... 3);

il faut que $a …. 3 soit false

var_dump($a ... 3);

var_dump($a ... 3);

var_dump($a ... 3);

var_dump($a ... 3);

var_dump($a ... 3);

Exercice 3

Répondez à la question avant de tester votre réponse avec php

$a = true ;

var_dump($a == true); affiche …

var_dump($a == « true »); affiche ...

var_dump($a < 3); affiche ...

var_dump($a === true); affiche ...

var_dump($a == 1); affiche …

var_dump($a === 1); affiche ...

var_dump($a <> 3); affiche ...

var_dump($a == "coucou"); affiche ...

var_dump($a == "null"); affiche ...

var_dump($a == 110); affiche ...

var_dump($a != 3); affiche ...

var_dump($a !== 3); affiche ...

var_dump($a != null); affiche ...

var_dump($a !== null); affiche ...

Solutions :operateur_resolu.php

Présentation des conditions en PHP

Les structures de contrôle conditionnelles (ou plus simplement conditions) vont nous permettre d'exécuter différents blocs de code selon qu'une condition spécifique soit vérifiée ou pas.

Par exemple, on va pouvoir utiliser les conditions pour afficher un message de bienvenue différent en PHP sur notre site selon que l'utilisateur soit connu ou un simple visiteur qui ne s'est jamais inscrit sur notre site.

Nous allons très souvent utiliser les conditions avec des variables : selon la valeur stockée dans une variable, nous allons vouloir exécuter un bloc de code plutôt qu'un autre.

Les conditions vont ainsi être un passage incontournable pour rentre un site dynamique puisqu'elles vont nous permettre d'exécuter différents codes et ainsi afficher différents résultats selon le contexte.

Les conditions if, if...else et if...elseif...else PHP

La condition if en PHP

La condition if est l'une des plus importantes et des plus utilisées dans l'ensemble des langages de programmation utilisant les structures de contrôle et en particulier en PHP.

La condition if est également la plus simple, puisqu'elle va nous permettre d'exécuter un bloc de code si et seulement si le résultat d'un test vaut true.

```php
<?php
$a = 2;
$b = 4;

if ($a < $b) {
    echo '$a est plus petit que b';
}
if ($a > $b) {
    echo '$a est plus grand que b';
}
if ($a == $b) {
    echo '$a est égale à  b';
}
```

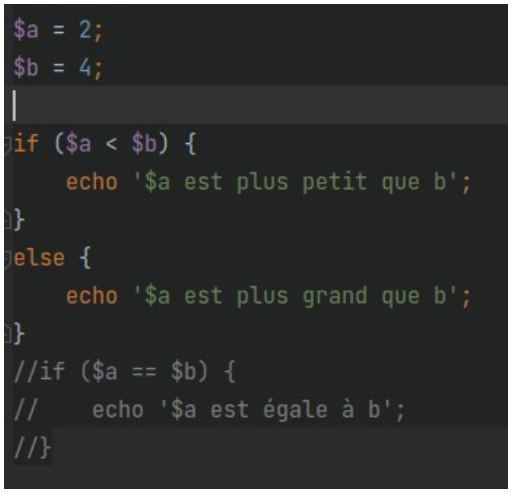
Exemple :

$a est plus petit que b

La condition if...else en PHP

L’exemple précédent serait parfaitement fonctionnel mais n’est pas efficace. En effet, le programme va tester si $a est plus petit que $b, définir que oui et donc afficher la phrase prévue. Mais le programme n’est pas fini… Il va donc tester si $a >$b et puis que $a == $b. Cela n’a aucun sens… Il n’y a aucun cas de figure ou plus d’une phrase serait affichée.

C’est là que vient à notre aide if...else. Pour ne pas embrouiller j’ai supprimé un des tests



L’affichage est bien entendu identique.

Comme vous pouvez le constater, il n’y a pas de condition sur le else. Si la condition du if est false, le else est d’office exécuter. Ce qui crée un bug dans notre programme si $a == $b

La condition if...elseif...else en PHP

La condition if...elseif...else (« si...sinon si...sinon ») est une structure conditionnelle encore plus complète que la condition if...else puisqu’elle va nous permettre cette fois-ci de générer autant de cas que l’on souhaite et d’arranger notre bug.

Notez que contrairement au else, on peut faire suivre autant de elseif que l’on veut dans notre condition if...elseif...else, chacun avec un test différent.

```php
if($nom == "Laurent") {echo'coucou Laurent';}
elseif($nom == "Pierre") {echo'coucou Pierre';}
elseif($nom == "Sophie") {echo'coucou Sophie';}
elseif($nom == "...") {echo'coucou ...';}
```

Revenons à notre exemple :

```php
if ($a < $b) {
    echo '$a est plus petit que b';
}
elseif ($a > $b) {
    echo '$a est plus grand que b';
}
else {
    echo '$a est égale à b';
}
```

Le else final n'est en rien obligatoire. Nous aurions pu écrire le code cette façon pour toujours le même résultat.

```php
$a = 2;
$b = 4;
$txt = '$a est égale à b';

if ($a < $b) {
    $txt = '$a est plus petit que b';
}
elseif ($a > $b) {
    $txt = '$a est plus grand que b';
}
echo $txt;
```

Vous comprendrez que si cela fonctionne ce n'est pas le plus efficace. Le deuxième test ne servant absolument à rien.

Simplifions :

```php
$a = 2;
$b = 4;
$txt = '$a est égale à b';

if ($a < $b) {
    $txt = '$a est plus petit que b';
}
else {
    $txt = '$a est plus grand que b';
}
echo $txt;
```

Je vous invite à vous renseigner sur « PHP Shorthand if Statements » qui permet de réécrire de cette façon :

```php
$a = 2;
$b = 4;
$txt = '$a est égale à b';

$a < $b ? $txt = '$a est plus petit que b' : $txt = '$a est plus grand que b';

echo $txt;
```

Imbriquer plusieurs conditions

Bien souvent, il faudra effectuer des tests plus complexe que ce que l'on a vu précédement. Il est souvent utile d'avoir une sous condition dans une condition.

Exemple : Vérifions si une personne peut acheter une boisson. En Belgique il faut 18 ans pour acheter de l'alcool fort et 16 ans de la bière.

```php
$type = "biere"; // valeur biere ou soft ou alcool
$age = 17;
$ok = false; //par defaut on interdit la vente. C'est une bonne pratique

if ($age < 18) {
    if ($type == "biere") {
        if($age >= 16) $ok = true;
    }
    if($type == 'soft') $ok = true;
}
else $ok = true;

if($ok) echo'Vous pouvez en avoir';
else echo 'vous ne pouvez pas en avoir';
```

C'est inutilement compliqué dans ce cas… Améliorons la lisibilité avec :

Les opérateurs logiques

Les opérateurs logiques vont être principalement utilisés avec les conditions puisqu'ils vont nous permettre d'écrire plusieurs comparaisons au sein d'une même condition ou encore d'inverser la valeur logique d'un test.

Opérateur	Définition
AND ou &&	Renvoie true si toutes les comparaisons valent true
OR ou ||	Renvoie true si une des comparaisons vaut true
!	Renvoie true si la comparaison vaut false (et inversement)

Réécrivons le code précédent :

```php
if ($type == 'soft') $ok = true;
elseif ($age >= 18) $ok = true;
elseif ($type == 'biere' && $age >= 16) $ok = true;
```

Déjà plus simple… Mais on peut encore simplifier :

```php
if ($age >= 18 || $type == 'soft' || ($type == 'biere' && $age >= 16)) $ok = true;
```

Si le client à au minimum 18ans ou que la boisson est un soft ou que c'est une bière mais que le client à minimum 16 ans alors il peut en acheter…

Notez que par défaut $ok vaut false. C'est une bonne méthode d'un point de vue sécurité. Pourquoi ?

Exercices

Exercice 1 :

Remplissez le if de la conditionnelle suivante qui teste si le résultat atteint la moyenne /20

```php
$resultat = 15;
if () {
    echo 'C\'est bien vous avez eu la moyenne ou plus au test';
} else {
    echo 'Vous n\'avez pas eu la moyenne au test';
}
```

Exercice 2 :

Déclarer une variable `$budget` qui contient la somme de 1 553,89 €. Déclarer une variable `$achats` qui contient la somme de 1 554,76 €. Afficher si le budget permet de payer les achats.

Exercice 3 :

Si la variable age est inférieur à 18 ans, afficher "Tu es mineur ", sinon "Vous êtes majeur".

Exercice 4

En fonction de l'heure, afficher ces messages :

avant 12h: Bonne matinée
de 12h à 14h: Bonne appétit
de 14h à 19h: Bonne aprés midi

sinon Bonne soirée

Exercice 5 :

Compléter la conditionnelle suivante qui teste l'age et l'origine de la personne.

$age est un nombre entier.

$origin peut prendre la valeur : "Belge" ou "belge" (ou « francais »,…). Attention que l'objectif est de tester les 2 possibilités. Donc pas de fonction pour mettre en minuscule le texte.

Le code ne peut pas changer en dehors des 2 parenthèses.

```php
echo '<h2>Exercice 5</h2>';

$age = 15;
$origin = 'français';
if () {
    echo 'Vous avez 18 ans ou plus et êtes Belge';
} elseif () {
    echo 'Vous avez 18 ans ou plus et êtes étranger';
} else {
    echo 'Vous n\'êtes pas majeur';
}
```

Exercice 6 : Réécrire le code précédent en ne testant qu'une fois l'age.

Exercice 7 : Rédigez une expression conditionnelle pour tester si un nombre est à la fois un multiple de 3 et de 5. (aide : Modulo - reste d'une division euclidienne)

Correctif : conditions.php

Projet jeu du pendu V0.1

Nous allons créer, petit à petit, un jeux du pendu. Ceci uniquement en PHP, HTML et CSS.

Télécharger le fichier de base 0-fichierBase.php et le fichier CSS style.css

Vous avez à disposition plusieurs choses :

- le code HTML et CSS du pendu
- un bloc « constante d'environnement ». Il contient une constante qui défini le nombre maximum d'échec possible avant le Game Over. Une constante est une sorte de variable mais la valeur ne peut jamais changer contrairement à une variable. Par convention elles sont écrites en majuscules. Pour l'afficher par exemple :

  ```
  echo NBR_CHANCE;
  ```

- un bloc « variables qui viennent du formulaire ». A terme certaines données seront envoyées par un formulaire html. Le joueur devra par exemple indiquer la lettre à jouer.

  Pour l'instant, on « force » la valeur n'ayant pas de formulaire.

- Un bloc « variables externes ». C'est le cas du mot à trouver. Externe car à terme, il viendra d'un service externe au jeu en lui même.
- Un bloc « variables qui seront calculées ». Dans un premier temps, on force toujours les valeurs. Evidemnt le nombre d'erreur devra etre calculé en incrémentant la valeur à chaque erreur.

Objectif de la V0.1

1. Créer un dossier dans votre serveur web « pendu » et y placer les 2 fichiers de base
2. Créer un repository pour le jeu sur github.
3. Afficher les morceaux du pendu suivant le nombre d'erreur (compris entre 0 et NBR_CHANCE). On commence par la tête
4. Afficher les lettres qui ont déjà été trouvées. (a,b,c)
5. Afficher un message si la lettre proposée est bonne ou pas(il faudra créer une nouvelle variable)
6. Afficher un message quand c'est game over.
7. Afficher un message quand le mot est trouvé (le formulaire contiendra un champs ou l'on peut proposer un mot en cours de jeux). Une nouvelle variable devra être définie.

Attention que pour l'instant il ne faut pas vérifier si une lettre est dans le mot mystère ou autre . On forces les variables pour tester.

Tester tous les cas en changeant les divers variables.

L’instruction switch en PHP

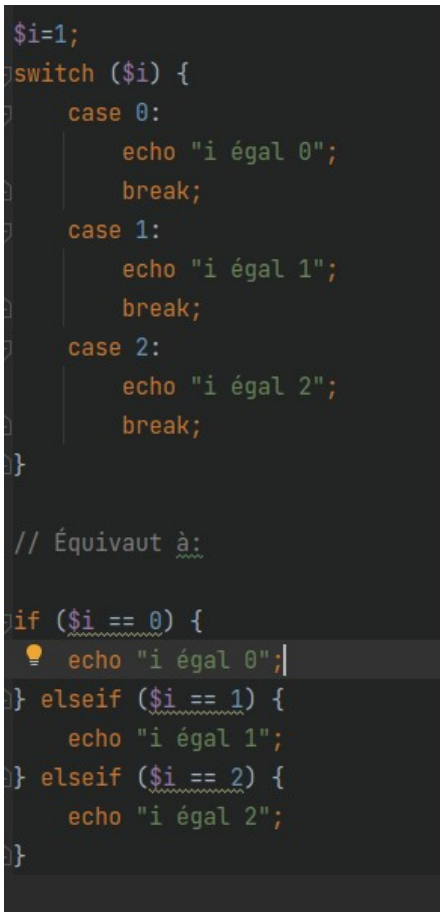
Dans la vie de développeur il faut de temps à autre tester une valeur de nombreuses fois. Cela nous amène à des conditions du type

if ($a=…)... elseif($a=…)….elseif($a=…)….elseif($a=…)….elseif($a=…)….elseif($a=…)
….elseif($a=…)….elseif($a=…)….elseif($a=…)

seif($a=…)….elseif($a=…)….elseif($a=…)….elseif($a=…)….elseif($a=…)….elseif($a=…)
….elseif($a=…)….elseif($a=…)….elseif($a=…)….elseif($a=…)….elseif($a=…)….elseif($a=…)….

switch permet d’etre plus lisible et un peu plus rapide. Sa limite est que c’est une comparaison « pauvre » (== et pas = = =)

La syntaxe de switch est assez simple. On indique la variable à tester, les valeurs possibles et le code à executer si c’est vrai. L’instruction break, permet de quitter le switch. Sans cela le test suivant est effectué.



```php
$i=1;
switch ($i) {
    case 0:
        echo "i égal 0";
        break;
    case 1:
        echo "i égal 1";
        break;
    case 2:
        echo "i égal 2";
        break;
}

// Équivaut à:

if ($i == 0) {
    echo "i égal 0";
} elseif ($i == 1) {
    echo "i égal 1";
} elseif ($i == 2) {
    echo "i égal 2";
}
```

L’instruction match

Cette instruction n’est disponible que depuis PHP 8.0. Elle complète switch car la comparaison effectuée est riche (= = = et pas ==)

Utilisation de base :

```php
$return_value = match ($food) {
    'apple' => 'This food is an apple',
    'bar' => 'This food is a bar',
    'cake' => 'This food is a cake',
};
```

Utilisation avec des opérations de comparaison :

```php
$age = 18;

$output = match (true) {
    $age < 2 => "Bébé",
    $age < 13 => "Enfant",
    $age <= 19 => "Adolescent",
    $age > 19 => "Jeune adulte",
    $age >= 99 => "Adulte âgé"
};
```

« Read the fucking manual » ou l'art de lire la doc :-)

Si un jour vous posez une question à un autre développeur et qu'il vous répond RTFM. Le message sera passé…

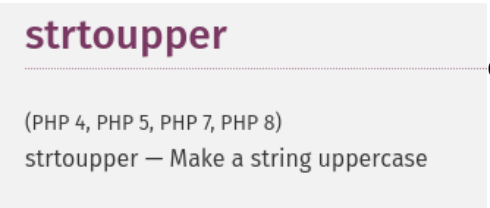
Comme l'ensemble des langages, php à une documentation exhaustive. Avant de foncer sur un forum ou de demander à une intelligence artificielle cherchez dedans ! Vous aurez accès aux subtilités, aux limites,… de la fonction recherchée.

Php regorge de fonctions pré-établies. Une fonction est un « programme » qui va effectuer un tâche spécifique. Vous connaissez echo, if et d'autres fonctions mais il en existe tant d'autre…

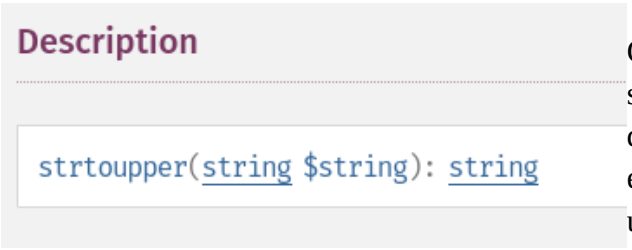
Les fonctions php vont vous permettre d'avoir une solution à un de vos besoins. Les fonctions vont renvoyer un type de réponse et vous devrez lui donner des informations pour que cela fonctionne.

Je vous propose de chercher dans votre moteur de recherche « strtoupper php » Le premier résultat devrait renvoyer vers la doc php.org.

Que voyons nous dans celle-ci



Vous trouvez le nom de la fonction et ce qu'elle fait. Mais également les versions de php compatible !



Dans description, cette partie est importante. Comment utilisons t'on cette fonction. On y indique son nom et entre parenthèse on lui donne la variable qui DOIT être de type string et qui est le mot à mettre en majuscule. Derriere les « : » ce qu'elle répond. Ici un string

Vous trouverez plus bas des exemples d'utilisation et même des commentaires de contributeurs qui explique des subtilités, des trucs et stuces, des limites ou bugs…

La variable tableau

Elle vaut bien un chapitre...

C’est quoi un tableau ?

Que cela soit en Front ou en back, difficile de passer à coter…

En gros, un tableau est une variable qui contient un ensemble de variable

Imaginez que vous vouliez stocker dans une variable vos compétences en programmation. Si vous n’avez qu’une compétence c’est simple

```
$competences= "php";
```

Mais cela se complique si vous en avez plusieurs…

```
$competences= "php & html & javascript";
```

Cela serait une solution, mais si vous voulez faire un code php qui donnera ceci en html

```
<ul>
        <li>php</li>
        <li>html</li>
        <li>javascript</li>
</ul>
```

C’est pas gagné. Heureusement les tableaux sont là pour ca.
On va pouvoir créer une variable qui pourait etre représentée de la sorte (attention, ce n’est qu’une représentation pour comprendre, cela ne fonctionne pas comme ca)
$competences = ($competence1 = « php, $compentence2 = «html », $competence3 = « javascript »)

Avec un truc du genre nous pouvons faire notre affichage précédent en récupérent les 3 variables compétence.

Présentation des tableaux en PHP

Les tableaux en PHP sont donc des variables spéciales qui peuvent stocker plusieurs valeurs en même temps.
Dans un tableau, chaque valeur va être associée à une clef unique. Cette clef va nous permettre notamment de récupérer la valeur associée. Nous allons pouvoir définir les différentes clefs ou laisser le PHP générer automatiquement les clefs pour les différentes valeurs d’un tableau.

On va pouvoir créer trois types de tableaux différents en PHP :

- Des tableaux numérotés ou indexés (les clefs vont être des nombres) ;
- Des tableaux associatifs (nous allons définir la valeur que l’on souhaite pour chaque clef) ;
- Des tableaux multidimensionnels (tableaux qui stockent d’autres tableaux en valeur).

Pour créer un tableau, on utilise la syntaxe [], mais vous pouvez croiser encore l’ancienne notation array().

On va pouvoir passer autant d'argument qu'on souhaite stocker de valeurs dans notre tableau. Les arguments vont pouvoir être soit des valeurs simples (auquel cas les clefs seront des entiers générés automatiquement), soit des paires clef => valeur.

Création d'un tableau numéroté ou indexé en PHP

Les tableaux numérotés sont le type de tableaux le plus simple à créer en PHP puisque les clefs vont être générées automatiquement par le PHP.
Pour créer un tableau numéroté en PHP, il suffit en fait d'indiquer une série de valeurs et le PHP associera automatiquement une clef unique à chaque valeur, en commençant avec la clef 0 pour la première valeur, la clef 1 pour la deuxième valeur, la clef 2 pour la troisième valeur, etc.

```php
$competences = ["php", "html", "javascript"];
```

Rien de plus simple. Regardons ce qu'il se cache dans cette variable grâce à var_dump().

```
array(3) {
  [0]=>
  string(3) "php"
  [1]=>
  string(4) "html"
  [2]=>
  string(10) "javascript"
}
```

Cela nous renvoi bien un tableau de 3 valeurs.

- La première à la clé 0 et est un string de 3 caractères et valeur php
- La deuxième à la clé 1 et est un string de 4 caractères et valeur html
- La troisième à la clé 3 et est un string de 10 caractères et valeur javascript

Le clés ont étés définies par php, car elles n'ont pas été indiquées au départ. Nous aurions pu le faire nous même :

Façon 1 :

```php
$competences[0] = "php";
$competences[1] = "html";
$competences[2] = "javascript";
```

Façon 2 :

```php
$competences = [0 => "php", 1 => "html", 2 => "javascript"];
```

Dans tous les cas le var_dump() renverra la même chose.

Évidement, dans le cas présent cela n'a pas de sens de donner les clés, mais cela sera nécessaire dans beaucoup de cas.

Pour afficher les valeurs d'un tableau numéroté une à une, il suffit d'echo notre variable tableau en précisant l'indice (entre crochets) correspondant à la valeur que l'on souhaite afficher.

```php
$competences = ["php", "html", "javascript"];

echo $competences[0]; //retournera php
echo $competences[1]; //retournera html
echo $competences[2]; //retournera javascript
```

Les tableaux associatifs PHP

Un tableau associatif est un tableau qui va utiliser des clefs textuelles qu'on va associer à chaque valeur.

Les tableaux associatifs vont s'avérer intéressant lorsqu'on voudra donner du sens à nos clefs, c'est-à-dire créer une association forte entre les clefs et les valeurs d'un tableau.

Les tableaux associatifs vont être différents des tableaux numérotés au sens où nous allons devoir définir chacune des clefs : PHP ne va pas ici pouvoir nommer automatiquement nos clés. Attention que les clés doivent être uniques dans le tableau.
Tout comme pour les tableaux numérotés, on va pouvoir créer notre tableau en une fois en utilisant la syntaxe [] ou le construire clef par clef et valeur par valeur.
"co

Imaginons le tableau indexé suivant qui contient les informations d'une personne.

Vous savez que cela comprend la profession, l'âge, le nom, le prénom et le nombre de commande que l'utilisateur à passé sur le site…

A part la profession, impossible de savoir ce que représente quoi... Même si être médecin est rare à 24 ans :-)

Réécrivons cela avec un tableau associatifs

Plus simple à comprendre et à utiliser :

```php
echo 'Bonjour ' . $user['prenom'] . ' ' . $user['nom'] . ', vous avez ' . $user['age'] . ' ans.
Vous avez effectué ' . $user["nbrCommande"] . "commandes sur le site";
```

Les tableaux multidimensionnels

Un tableau multidimensionnel est un tableau qui va lui-même contenir d'autres tableaux en valeurs. C'est donc un tableau de tableaux.

On appelle tableau à deux dimensions un tableau qui contient un ou plusieurs tableaux en valeurs, tableau à trois dimensions un tableau qui contient un ou plusieurs tableaux en valeurs qui contiennent eux-mêmes d'autres tableaux en valeurs et etc.

Les « sous » tableaux vont pouvoir être des tableaux numérotés ou des tableaux associatifs ou un mélange des deux.

Pour illustrer cela, rajoutons un tableau de compétence à notre utilisateur.

```php
$user = [
    "nbrCommande" => 24,
    "profession" => "médecin",
    "competences" => [
        "chirurgie",
        "ophtalmologie",
    ],
    "prenom" => "François",
    "nom" => "Bernard",
    "age" => 32
];
```

```
array(6) {
  ["nbrCommande"]=>
  int(24)
  ["profession"]=>
  string(8) "médecin"
  ["competences"]=>
  array(2) {
    [0]=>
    string(9) "chirurgie"
    [1]=>
    string(13) "ophtalmologie"
  }
  ["prenom"]=>
  string(9) "François"
  ["nom"]=>
  string(7) "Bernard"
  ["age"]=>
  int(32)
}
```

Notre tableau associatif comprend maintenant une clé « compétence » qui est un tableau indexé.

Autre exemple, sur notre site fictif, il n'y a pas qu'un utilisateur mais plusieurs. Imaginons que pour l'instant il y en a deux.

```php
$users = [
    [
        "nbrCommande" => 24,
        "profession" => "médecin",
        "competences" => [
            "chirurgie",
            "ophtalmologie",
        ],
        "prenom" => "François",
        "nom" => "Bernard",
        "age" => 32
    ],
    [
        "nbrCommande" => 7,
        "profession" => "ingénieur",
        "competences" => [
            "construction de routes et pont",
            "construction métallique",
            "experte en matériaux durables"
        ],
        "prenom" => "Sophie",
        "nom" => "Plakbala",
        "age" => 41
    ]
];
```

```
array(2) {
  [0]=>
  array(6) {
    ["nbrCommande"]=>
    int(24)
    ["profession"]=>
    string(8) "médecin"
    ["competences"]=>
    array(2) {
      [0]=>
      string(9) "chirurgie"
      [1]=>
      string(13) "ophtalmologie"
    }
    ["prenom"]=>
    string(9) "François"
    ["nom"]=>
    string(7) "Bernard"
    ["age"]=>
    int(32)
  }
  [1]=>
  array(6) {
    ["nbrCommande"]=>
    int(7)
    ["profession"]=>
    string(10) "ingénieur"
    ["competences"]=>
    array(3) {
      [0]=>
      string(30) "construction de routes et pont"
      [1]=>
      string(24) "construction métallique"
      [2]=>
      string(30) "experte en matériaux durables"
    }
    ["prenom"]=>
    string(6) "Sophie"
    ["nom"]=>
    string(8) "Plakbala"
    ["age"]=>
    int(41)
  }
}
```

Affichage des tableaux multidimensionnels

Nous voilà avec de beaux tableaux, mais comment afficher cela ?

Notre premier tableau dans la hierarchie est un tableau indexé. On sait donc que le premier utilisateur à la clé 0, le deuxième la clé 1 …

Donc si nous désirons afficher le prénom du premier utilisateur, il nous suffira d’écrire

```php
echo $users[0]["prenom"];
```

De la même façon si on désire la première compétence de cet utilisateur :

```php
echo $users[0]["competences"][0];
```

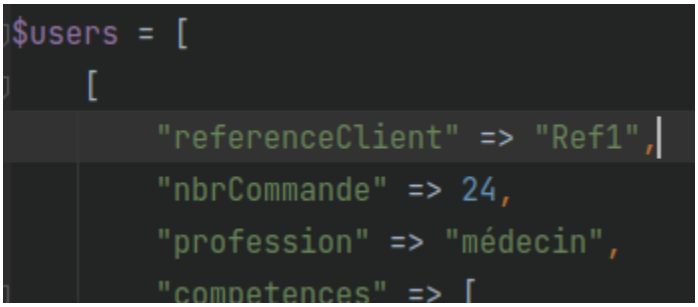
Tableau associatif ou indexé, comment choisir…

En règle générale on préfère un tableau indexé dans le cas d’une liste dont la longueur n’est pas définie et peu changer. Dans l’exemple des compétences des utilisateurs il serait difficile (et inutile...) de faire un tableau associatif. Des utilisateurs peuvent avoir une ou deux ou 200 compétences.

Dans le même exemple nous pourrions nous dire que c’est la meme chose pour les utilisateurs. Il peut y en avoir un ou bien plus. Mais ce n’est pas parfaitement exacte en pratique.

Il est évident que dans un vrai site, la liste des utilisateurs est stockée dans une base de donnée. Cela signifie qu’il existe obligatoirement un identifiant unique par utilisateur. Une sorte de référence qui permet de le trouver facilement dans la base de donnée.

On pourrait imaginé mettre cette référence dans le tableau de valeurs de l’utilisateur comme ceci :



Mais une façon bien plus performante serait de mettre cette référence en clé. Comme elle est unique en base de donnée on peut le faire. Cela sera bien plus efficace pour afficher les données d’un utilisateur si on a sa référence.

Exercice :

Transformer le fichier excell (tableau.php) en un tableau php qui doit être exploitable au maximum. Il faut donc pouvoir avoir à la fin une solution pour sortir une information très précise. Pensez à l'ensemble des besoins que vous pourriez avoir après. Avoir une info, faire des calculs, …
Vous aurez remarqué que les informations sont parfois differentes pour les 2 pays. Soit l'info est manquant, soit elle est incomplète. Malgré cela la structure des 2 tableaux pays doit être la même. Le contenu seul changera.

Correctif : tableau_corrige.php

Les boucles et les opérateurs d’incrémentation et de décrémentation

C’est quoi une boucle

Les boucles vont nous permettre d’exécuter plusieurs fois un bloc de code, c’est-à-dire d’exécuter un code « en boucle » tant qu’une condition donnée est vérifiée.

Lorsqu’on code, on va en effet souvent devoir exécuter plusieurs fois un même code. Imaginez vous devoir afficher toute les information contenues dans un tableau php. Le meme code sera exploité (un simple echo ici) pour toutes les lignes.

Ou encore récupérer la liste des produits achetés dans une commande, afficher le prénom de tous les utilisateurs de notre site, ...

Nous disposons de quatre boucles différentes en PHP :

- La boucle while (« tant que ») ;
- La boucle do... while (« faire... tant que ») ;
- La boucle for (« pour ») ;
- La boucle foreach (« pour chaque »).

Le fonctionnement général des boucles sera toujours le même : on pose une condition qui sera généralement liée à la valeur d’une variable et on exécute le code de la boucle « en boucle » tant que la condition est vérifiée.

Pour éviter de rester bloqué à l’infini dans une boucle, vous pouvez donc déjà noter qu’il faudra que la condition donnée soit fausse à un moment donné (pour pouvoir sortir de la boucle).

Pour que notre condition devienne fausse à un moment, on incrémentera ou on décrémentera la valeur de notre variable à chaque nouveau passage dans la boucle.

Les opérateurs d’incrémentation et de décrémentation

Incrémenter une valeur signifie ajouter 1 à cette valeur tandis que décrémenter signifie enlever 1.

Les opérations d’incrémentation et de décrémentation vont principalement être utilisées avec les boucles en PHP. Elles vont pouvoir être réalisées grâce aux opérateurs d’incrémentation ++ et de décrémentation --.

Donc en gros :

$x++ équivaut à $x = $x + 1

$x-- équivaut à $x = $x – 1

Il existe également les opérateurs ++$x et --$x La seule difference est dans le cas ou vous traitez la valeur de $x au moment de l’incrémentation :

```php
$x = 0;
echo ++$x; // affiche 1
$x = 0;
echo $x++; // affiche 0
echo $x; // affiche 1
```

La boucle PHP while (rarement utilisée)

La boucle while (« tant que » en français) est la boucle PHP la plus simple à appréhender.

La boucle while va nous permettre d'exécuter un certain bloc de code « tant qu'une » condition donnée est vérifiée.

```php
$i = 0;
while ($i == 10) {
    echo '$i vaut: ' . $i . '<br>';
    $i++;
}
```

```
$i vaut: 0
$i vaut: 1
$i vaut: 2
$i vaut: 3
$i vaut: 4
$i vaut: 5
$i vaut: 6
$i vaut: 7
$i vaut: 8
$i vaut: 9
```

Dans l'exemple ci-dessus, on commence par stocker la valeur 0 dans notre variable $i. On dit également qu'on « initialise » notre variable.

Ensuite, on crée notre boucle while, qui va réutiliser $i puisque nous allons boucler sur les valeurs successives stockées dans $i jusqu'à ce que $i stocke une certaine valeur.

Que se passe-t-il exactement dans cette boucle ? Dans le cas présent, on demande à notre boucle d'afficher $i vaut : suivi de la valeur de $i. Ceci jusque quand la valeur stockée dans $i soit égale à 10.

A la fin de chaque passage dans la boucle, on ajoute 1 à la valeur précédente contenue dans $i grâce à l'opérateur d'incrémentation ++. Cette étape d'incrémentation est indispensable. Sans celle-ci, $i contiendrait toujours la valeur 0 (sa valeur de départ)on ne pourrait jamais sortir de la boucle.

La boucle PHP do...while (rarement utilisée)

La boucle PHP do... while (« faire... tant que » en français) ressemble à priori à la boucle while mais va fonctionner « en sens inverse » par rapport à while.

En effet, la boucle PHP do... while va également nous permettre d'exécuter un code tant qu'une condition donnée est vraie, mais cette fois-ci le code dans la condition sera exécuté avant que la condition soit vérifiée.

Ainsi, même si une condition est fausse dès le départ, on effectuera toujours au moins un passage au sein d'une boucle do...while, ce qui n'est pas le cas avec une boucle while.

```php
$i = 0;
$y = 10;
echo'<br>while $i<br>';
while ($i < 10) {
    echo 'Avec while $i vaut: ' . $i . '<br>';
    $i++;
}
echo'<br>while $y<br>';

while ($y < 10) {
    echo 'Avec while $y vaut: ' . $y . '<br>';
    $y++;
}

$i = 0;
$y = 10;
echo'<br>doWhile $i<br>';
do {
    echo 'avec doWhile $i vaut: ' . $i . '<br>';
    $i++;
} while (
    $i < 10
);
echo'<br>doWhile $y<br>';
do {
    echo 'avec doWhile $y vaut: ' . $y . '<br>';
    $y++;
} while (
    $y < 10
);
```

```
while $i
Avec while $i vaut: 0
Avec while $i vaut: 1
Avec while $i vaut: 2
Avec while $i vaut: 3
Avec while $i vaut: 4
Avec while $i vaut: 5
Avec while $i vaut: 6
Avec while $i vaut: 7
Avec while $i vaut: 8
Avec while $i vaut: 9

while $y

doWhile $i
avec doWhile $i vaut: 0
avec doWhile $i vaut: 1
avec doWhile $i vaut: 2
avec doWhile $i vaut: 3
avec doWhile $i vaut: 4
avec doWhile $i vaut: 5
avec doWhile $i vaut: 6
avec doWhile $i vaut: 7
avec doWhile $i vaut: 8
avec doWhile $i vaut: 9

doWhile $y
avec doWhile $y vaut: 10
```

La boucle for

La boucle PHP for (« pour » en français) est la boucle utilisée majoritairement quand un test est nécessaire.

Nous pouvons décomposer le fonctionnement d'une boucle for selon trois phases :

- Une phase d'initialisation :

  Nous allons tout simplement initialiser la variable que nous allons utiliser dans la boucle.

- Une phase de test :

  Nous allons préciser la condition qui doit être vérifiée pour rester dans la boucle.

- Une phase d'incrémentation : Nous allons pouvoir incrémenter ou décrémenter notre variable afin que notre condition soit fausse à un moment donné.

Concrètement, reprenons l'exemple précédent avec for :

```php
for ($i=0; $i<10; $i++){
    echo 'avec for $i vaut: ' . $i . '<br>';
}
```

```
$i vaut: 0
$i vaut: 1
$i vaut: 2
$i vaut: 3
$i vaut: 4
$i vaut: 5
$i vaut: 6
$i vaut: 7
$i vaut: 8
$i vaut: 9
```

Meme résultat qu'avec while ou doWhile.

La boucle PHP foreach

La boucle PHP foreach est un peu particulière puisqu'elle a été créée pour fonctionner avec des variables tableaux . Elles est certainement la plus utilisée aujourd'hui.

La boucle foreach va parcourir un tableau du début à la fin. Il n'y a pas de test pour sortir de la boucle.

Exemple :

```php
$array = ['pierre', 'papier', 'ciseau'];
echo'<br>foreach<br>';

foreach ($array as $value){
    echo 'avec foreach la valeur vaut: ' . $value . '<br>';
}
```

```
foreach
avec foreach la valeur vaut: pierre
avec foreach la valeur vaut: papier
avec foreach la valeur vaut: ciseau
```

N’oublions pas qu’un tableau est une association entre une clé et une valeur. La boucle foreach va nous permettre d’avoir accès aux 2.

```php
foreach ($array as $key => $value){
    echo 'avec foreach la valeur de la clé '. $key.' vaut: ' . $value . '<br>';
}
```

```
avec foreach la valeur de la clé 0 vaut: pierre
avec foreach la valeur de la clé 1 vaut: papier
avec foreach la valeur de la clé 2 vaut: ciseau
```

Break et continue, gagnons du temps d’exécution

Imaginez un tableau qui contient le dictionnaire. Le tableau aura donc au moins 60.000 valeurs avec une structure semblable à ceci : `$dico['table'] = "La table est un objet ....."`;

Créons un code qui va trouver la définition d’un mot en particulier :

```php
$dico['abris'] = "Un abris est... .....";
$dico['table'] = "La table est  .....";
$dico['croquette'] = "Une croquette est.....";
$dico['....'] = "....."; //60000 X donc...

$motRecherche = 'table';

foreach ($dico as $key => $value){
    if($key == $motRecherche) echo'La définition de ' . $motRecherche . ' est: '. $value;
}
```

Cela va fonctionner mais que va faire le programme.
Il va boucler sur le tableau. Tres vite il tombera sur le mot « table » et affichera la définition.
Mais le programme ne sera pas fini… Il va continuer sur les miliers d'autres mots à la recherche d'une autre concordance qu'il ne trouvera jamais.

C'est là que **l'instruction break;** est utilisée. Break va simplement arrêter la boucle.

```php
foreach ($dico as $key => $value){
    if($key == $motRecherche) {
        echo'La définition de ' . $motRecherche . ' est: '. $value;
        break;
    }
}
```

En pratique, dés que la clé vaut table le programme affichera la phrase et terminera de boucler pour continuer le reste du code éventuellement.

L'instruction continue; ne va pas arrêter la boucle mais passer à l'occurrence suivante.

Modifions notre programme pour rechercher 2 définitions :

```php
$motRecherche = 'table';
$motRecherche2 = 'panier';

foreach ($dico as $key => $value) {
    if ($key != $motRecherche && $key != $motRecherche2) continue;
    echo 'La définition de ' . $motRecherche . ' est: ' . $value;
}
```

Il faut bien comprendre le fonctionnement. Si $key ne vaut ni « table », ni « panier », le programme tombe sur l'instruction continue; Il ne va donc pas traiter la suite du code de la boucle et donc le echo dans cet exemple. Il passera au mot suivant. Quand il tombera sur un des 2 mots. Le test sera false, donc l'instruction continue; ne sera pas exécutée contrairement au echo.

Par contre, le script a des problèmes qu'il va falloir arranger…

- Le script affichera 2 lignes avec le mot de $motRecherche(la clé) dans les deux, et jamais celle de $motRecherche2
- Le script ne s'arrete pas quand les 2 définitions sont trouvées.

Cela sera votre mission de corriger cela dans les exercices 10 et 11 qui suivent.

Exercices

Niveau Bronze

Exercice 1 : Affichez une liste de course contenant 10 articles. Pour cela créez un tableau $list indexé. En utilisant la boucle foreach, affichez la liste sous forme de liste à point

Exercice 2 : Réutilisez le code précédent mais affichez le numéro d'ordre de 1 à 10 au début de la ligne. Ceci en récupérant le numéro de la clé.

Exercice 3 : Réutilisez le code précédent. N'affichez pas l'item 2 et 3 et n'affichez que 4 produits. Pour cela il faut utiliser exclusivement les instructions continue; et break;

Exercice 4 : Réutilisez le code de l'exercice 2. Affichez que les produits impaires. Pour cela créez une $variable $afficher = true ; avant le foreach. Changez la valeur de celle ci pour arriver à l'objectif.

Exercice 5 : Réutilisez la variable $list de l'exercice 1. affichez la liste à l'envers (de 10 à 0). N'utilisez plus foreach mais for pour y arriver. Pensez aux clés pour y arriver.

Exercice 6 : Déclarez un tableau multi-dimentionnel, qui contient les 4 saisons, lesquelles contiennent des légumes (2/saison), lesquels contiennent les clés "quantité" et "prix". Profitez des clés pour vous faciliter la vie :-) Les saisons sont uniques et les légumes le sont également (unicité par saison)

Affichez la liste de cette facon :

```
<ul>
        <li>Ete</li>
        <li>
                <ul>
                        <li>Fruit 1 : (quantité : 3 , prix : 10€)</li>
                        <li>Fruit 2 : (quantité : 3 , prix : 10€)</li>
                </ul>
        </li>
</ul>
…..
```

Niveau Argent

Exercice 7: Sur base de l'exercice 6 des conditions. Rédigez un programme qui affichera tous les nombres entre 1 et 1000 qui sont multiples de 3 ou de 5. (Modulo)

Exercice 8 : Effectuez une suite de tirages de nombres aléatoires (entre 1 et 100) jusqu'à obtenir une suite composée d'un nombre pair suivi de deux nombres impairs.

Coup de pouce : La fonction php rand() permet d’avoir un nombre aléatoire. Utilisez la boucle doWhile (et oui, de temps à autre elle est utile).

Exercice 9 : Choisissez un nombre de trois chiffres. Effectuez ensuite des tirages aléatoires, et comptez le nombre de tirages nécessaire pour obtenir le nombre initial. Arrêtez les tirages, et affichez le nombre de coups réalisés. Réalisez ce script d’abord avec l’instruction while puis avec l’instruction for.

Exercice 10 : Corrigez notre script dico pour corriger .

Exercice 11: Corrigez notre script dico qui a servi d'exemple à l'instruction continue; pour que la boucle s’arrête quand les 2 mots sont trouvés.

$dico = […….];

$motRecherche1 = 'table';

$motRecherche2 = 'panier';

$compteur = 2;

Niveau Or :

Exercice 11 : Affichez ceci :

```
0000000
1111111
2222222
3333333
4444444
5555555
6666666
7777777
8888888
9999999
```

Vous avez 2 variables :

$colonnes = 10; $lignes = 7; et une ou plusieurs boucles for.

Exercice 12 : Affichez ceci :

```
1
22
333
4444
55555
666666
7777777
```

Vous avez une variable $lignes = 7 ; et une ou plusieurs boucles for.

Exercice 13 : Affichez ceci :

```
0000000
1000000
```

```
2200000
3330000
4444000
5555500
6666660
7777777
```

Vous avez 2 variables : $lignes = 8; $colonnes = 6; et une ou plusieurs boucles for.

Exercice 14 : Affichez ceci :

```
20 19 18
17 16 15
14 13 12
11 10 9
8 7 6
5 4 3
2 1
```

Avec 2 variables et une boucle while

Inclure des fichiers dans d’autres en PHP avec include et require

Il existe quatre structures de contrôle qui vont se révéler très utiles et pratiques lorsqu’on voudra créer un site : include, require, include_once et require_once.

Présentation de include et de require et cas d’utilisation

Les instructions PHP include et require vont nous permettre toutes deux d’inclure des fichiers de code (ou plus exactement le contenu de ces fichiers) à l’intérieur d’autres fichiers de code.

Ces deux instructions sont très puissantes et vont s’avérer extrêmement utiles lors de la création d’un site web.

En effet, imaginions que vous vouliez créer un site personnel de plusieurs pages. Sur votre site, il y a de fortes chances que l’en-tête, le menu et le pied de page soient identiques pour toutes les pages.

Jusqu’à présent, en HTML, nous étions obligés de réécrire le même code correspondant à l’en-tête, au menu et au pied de page sur chacune des pages de notre site.Cela nous fait perdre beaucoup de temps et va poser de sérieux problèmes le jour où l’on souhaite modifier le texte dans notre pied de page par exemple pour tout notre site.

Plutôt que de réécrire tout le code relatif à ces différentes parties sur chacune de nos pages, pourquoi ne pas plutôt enregistrer le code de notre menu dans un fichier séparé que l’on appellera par exemple menu.php, et faire de même pour le code relatif à notre en-tête et à notre pied de page puis ensuite inclure directement ces fichiers sur chacune de nos pages ?

Nous allons pouvoir faire cela grâce aux instructions include et require.

Ainsi, nous n’avons plus qu’à écrire le code relatif à notre menu une bonne fois pour toutes et à l’inclure dans chaque page par exemple. Cela représente une considérable simplification pour la maintenance de notre code puisque nous n’avons qu’à modifier le code de notre menu dans le fichier menu.php plutôt que de le modifier dans toutes les pages si nous avions copié-collé le code du menu dans chaque page.

Notez déjà que pour inclure un fichier dans un autre fichier, il faudra préciser son emplacement par rapport au fichier qui contient l’instruction include ou require de la même façon qu’on pourrait le faire pour faire un lien en ou pour inclure une image en HTML.

Les différences entre include, include_once, require et require_once

La seule et unique différence entre les instructions include et require va se situer dans la réponse du PHP dans le cas où le fichier ne peut pas être inclus pour une quelconque raison (fichier introuvable, indisponible, etc.).

Dans ce cas-là, si l'inclusion a été tentée **avec include**, le **PHP renverra un simple avertissement** et le reste du script s'exécutera quand même tandis que si la même chose se produit **avec require**, **une erreur fatale sera retournée par PHP** et l'exécution du script s'arrêtera immédiatement.

L'instruction require est donc plus « stricte » que include.

La différence entre les instructions **include** et **require** et leurs variantes **include_once** et **require_once** est qu'on va pouvoir inclure plusieurs fois un même fichier avec include et require tandis qu'**en utilisant include_once et require_once** cela ne sera pas possible : **un même fichier ne pourra être inclus qu'une seule fois** dans un autre fichier.

Notez également qu'une variable ou fonction définie dans un fichier sera disponible dans le fichier qui l'inclus (include ou require) après cette inclusion. Il faut imaginer les inclusions comme des copié collé d'un fichier dans un autre.

Réécrivons le projet chocolat ensemble de cette façon.

C'est une bonne pratique de programmation de spécialiser à outrance les fichiers, les class, les fonctions,...

Dans le cas présent, le découpage du code va permettre d'avoir des fichiers bien plus petits qui seront plus simple à lire et à modifier.

Exercice Reprenez le projet Agence de voyage et séparez les divers éléments.

Dans cet exercice, il sera possible de récupérer le header, le footer, le menu sur toutes les pages mais le découpage du code va permettre également d'avoir des fichiers bien plus petits qui seront plus simple à lire et à modifier.

Télécharger le fichier de base (la structure a changée).

Créer un dossier contents qui contiendra les fichiers dont le contenu est partagé.

Le fichier index à la racine est « vide ». Cela va devenir l'unique fichier ou aboutissent les requêtes http. Cela signifie qu'il aura le rôle d'un router et va faire un include/require du bon fichier dans le dossier « pages » suivant l'url. A vous d'imaginer comment faire.

Les fonctions en PHP

Introduction aux fonctions PHP

Les fonctions sont un outil très puissants et pratiques en PHP et un préliminaire à la programmation object. C'est donc incontournable. Cela va encore nous permettre d'avoir un code clair et consis mais également augmenter la possibilité de partager du code entre nos pages.

Définition des fonctions et fonctions internes ou prêtes à l'emploi

Une fonction correspond à une série cohérente d'instructions qui ont été créées pour effectuer une tâche précise. Pour exécuter le code contenu dans une fonction, il va falloir appeler la fonction.

Nous avons déjà croisé des fonctions dans ce cours avec notamment la fonction strtoupper() dont le rôle est de mettre en majuscule une chaine de caractère.

La fonction strtoupper() fait partie des fonctions dites « internes » au PHP ou « prêtes à l'emploi » tout simplement car sa définition fait partie du langage PHP.

En effet, vous devez bien comprendre ici que la fonction strtoupper() contient une série d'instructions qui permettent de mettre notre chaîne de caractère en majuscule.

Comme cette fonction a été créée et définie par le PHP lui-même, nous n'avons pas à nous soucier des instructions qu'elle contient : nous n'avons qu'à appeler strtoupper() en précisant son nom et le nom d'une variable à mettre en couple de parenthèses pour que la série d'instructions qu'elle contient soient exécutées et pour obtenir la chaine en majuscule.

Les fonctions définies par l'utilisateur en PHP

En plus des fonctions internes, le PHP nous laisse la possibilité de définir nos propres fonctions. Cela va s'avérer très pratique dans le cas où nous travaillons sur un projet avec des besoins très spécifiques et pour lequel on va souvent devoir répéter les mêmes opérations.

Pour utiliser nos propres fonctions, nous allons déjà devoir les définir, c'est-à-dire préciser une première fois la série d'instructions que chaque fonction devra exécuter lors de son appel.

Pour déclarer une fonction, il faut déjà commencer par préciser le mot clef function qui indique à PHP qu'on va définir une fonction personnalisée.

Ensuite, nous allons devoir préciser le nom de notre fonction . Le nom des fonctions va suivre les mêmes règles que celles des variables, à savoir qu'il devra commencer par une lettre ou un underscore et ne devra pas être un nom déjà pris par le langage PHP (comme le nom d'une fonction déjà existante par exemple).

En plus de cela, notez qu'à la différence des variables le nom des fonctions est insensible à la casse. Cela signifie que l'utilisation de majuscules et des minuscules ne servent pas à différencier une fonction d'une autre.

Par exemple, si on crée une fonction bonjour(), les notations BONJOUR(), bonJOUR(),… feront référence à la même fonction.

Après le nom de notre fonction, nous devons obligatoirement mentionner un couple de parenthèses puis ensuite placer les instructions que la fonction devra exécuter lors de son appel entre des crochets.

Créons une première fonction très simple aura comme rôle d'afficher le texte « Bonjour » :

```
function direBonjour(){
   echo'Bonjour';
}
```

Si l'on test notre code, cela n'affiche rien. C'est entieremement normal car la fonction n'est pas appelée. Et donc php ne l'utilise pas.

Il ne reste plus qu'a en faire appel pour afficher notre texte à l'écran

```
function direBonjour(){
   echo'Bonjour';
   echo'<br>';
}
direBonjour();
direBonjour();
```

Bonjour
Bonjour

Le texte est visible deux fois car la fonction a été appelé deux fois.

Les paramètres et arguments des fonctions PHP

Il est évident que l'intérêt de la fonction précédente ne saute pas aux yeux… Par contre, imaginez une fonction qui envoi un mail à l'utilisateur après inscription sur un site. Cette fonction va s'occuper de toutes les tâches utiles à cette mission (en appelant elle même d'autres fonctions spécialisées). Elle devra créer le contenu du mail, rajouter des pièces jointes éventuelles, indiquer l'expéditeur, le destinataire, rajouter les clés de sécurité et enfin l'envoyer…

Pour pouvoir fonctionner cette fonction de mailing aura besoin au minimum de l'adresse du destinataire qui changera à chaque fois évidement.

En réalité, nous avons déjà vu comment cela fonctionne avec strtoupper($string). Ici on passe une chaine de caractère à la fonction. Ces informations (la valeure de la chaîne de caractère) sont appelées des **arguments**. Nous allons toujours passer les arguments dans les parenthèses de notre fonction.

Certaines fonctions ne vont pas avoir besoin d'argument pour fonctionner, mais la majorité vont en avoir besoin d'un ou de plusieurs. Par ailleurs, certains arguments vont parfois être facultatifs tandis que d'autres seront obligatoires.

Imodifions notre fonction pour que son rôle soit de renvoyer « Bonjour + un prénom ». Cette fonction va avoir besoin qu'on lui passe un prénom pour fonctionner correctement.

Dans sa définition, on va donc indiquer le fait que notre fonction va avoir besoin d'une donnée pour fonctionner en précisant **un paramètre** entre les parenthèses.

Ensuite, lorsqu'on appelle notre fonction, nous allons lui passer un prénom effectif, soit en lui passant une variable qui contient un prénom, soit en lui passant directement une chaîne de caractères. La valeur effective va alors ici remplacer le paramètre de la définition de la fonction c'est **l'argument**.

```php
function direBonjour($prenom){
   echo'Bonjour '.$prenom;
   echo'<br>';
}

direBonjour('Pierre');
?>
```

Bonjour Pierre

Définir des valeurs par défaut pour les paramètres de nos fonctions

Le PHP va également nous permettre de définir une valeur par défaut pour un paramètre d'une fonction. Cette valeur sera utilisée si aucun argument n'est fourni lors de l'appel de la fonction.

Si on définit une fonction avec plusieurs paramètres et qu'on choisit de donner des valeurs par défaut à seulement certains d'entre eux, alors il faudra placer les paramètres qui ont une valeur par défaut après ceux qui n'en possèdent pas dans la définition de la fonction.

Dans le cas contraire, le PHP renverra une erreur et notre fonction ne pourra pas être exécutée.

Reprenons l'exemple et rajoutons le rôle de l'utilisateur dans l'affichage. Il est probable que 99 % du temps cela soit un utilisateur, le signifier n'est pas nécessaire.

```php
function direBonjour($prenom, $role = "utilisateur"){
   echo'Bonjour '. $role. ' ' .$prenom;
   echo'<br>';
}
direBonjour('Pierre');
direBonjour("chef", "administrateur");
```

Bonjour utilisateur Pierre
Bonjour administrateur chef

Portées des variables avec les fonctions

Retenez que ce qui se passe dans une fonction reste dans la fonction.

Que va afficher le code suivant ?

```php
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);
$a = 5;
$b = 10;
echo '<br>1-> $a vaut: ' . @$a . '<br>';
echo '1-> $b vaut: ' . @$b . '<br>';
echo '1-> $x vaut: ' . @$x . '<br><br>';
function multiplier($x, $b)
{
   echo 'Résultat du calcul: ' .$x * $b;
   echo '<br>';
   echo '<br>2-> $a vaut: ' . @$a . '<br>';
   echo '2-> $b vaut: ' . @$b . '<br>';
   echo '2-> $x vaut: ' . @$x . '<br><br>';
}
multiplier($a, $b + 1);
echo '<br>3-> $a vaut: ' . @$a . '<br>';
echo '3-> $b vaut: ' . @$b . '<br>';
echo '3-> $x vaut: ' . @$x . '<br><br>';
```

Le retour d’une fonction

Notre fonction de multiplication pourrait être utilisée pour faire autre chose que d’imprimer le résultat. Par exemple stocker le résultat en base de donnée. C’est pour cela que très souvent une fonction renvoi un donnée que nous pourrons stocker dans une variable pour être utilisée plus tard.

Pour cela nous aurons besoin de la fonction return, le code suivant fait la même chose que précédemment. Afficher le résultat :

```php
function multiplier(float $x, float $y)
{
   return $x * $y;
}
$result = multiplier($a, $b);
echo 'Résultat du calcul: ' . $result;
```

Ou encore plus facilement :

```
echo 'Résultat du calcul: ' . multiplier($a, $b);
```

Le PHP, un langage au typage faible

A la base le langage PHP est un langage qui ne possède pas un typage fort.

Cela signifie de manière concrète qu'on n'a pas besoin de spécifier le type de données attendues lorsqu'on définit des paramètres pour une fonction car PHP va déterminer lui-même le type des données passées à notre fonction en fonction de leur valeur.

Une conséquence de cela est qu'il va être possible de passer des arguments qui n'ont aucun sens à une fonction sans déclencher d'erreur. On va par exemple pouvoir parfaitement passer deux chaînes de caractères à notre fonction multiplier() sans déclencher d'erreur.

Le typage des arguments

Depuis la version 7 de PHP il est possible de faire du typage en indiquant le type de données attendues lorsqu'on définit une fonction. Si une donnée passée ne correspond pas au type attendu une erreur sera renvoyée.

Cela va permettre à nos fonctions de ne s'exécuter que si les valeurs passées en argument sont valides et donc d'obtenir toujours un résultat cohérent par rapport à nos attentes.

C'est extrèmement important de prendre très vite l'habitude de la faire. En objet, c'est indispensable pour avoir une chance de déboguer facilement et pour éviter des problèmes.

Les types valides les plus importants sont les suivants :

- string L'argument passé doit être de type string (chaine de caractères)
- int L'argument passé doit être de type integer (nombre entier)
- float L'argument passé doit être de type float ou double (nombre décimal ou nombre entier...)
- bool L'argument passé doit être de type boolean (booléen)
- array L'argument passé doit être de type array (tableau)
- object L'argument passé doit être de type object (objet)

```
$a = "cinq";
$b = 10;
$c = 5;
function multiplier(float $x, float $y)
{
   echo 'Résultat du calcul: ' . $x * $y;
   echo '<br>';
```

```
}
multiplier($a, $b);
multiplier($c, $b);
```

Résultat du calcul: 50

Fatal error: Uncaught TypeError: multiplier(): Argument #1 ($x) must be of type float, string given, called in /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php on line 15 and defined in /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php:9 Stack trace: #0 /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php(15): multiplier() #1 {main} thrown in **/home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php** on line **9**

Typage strict

Essayons ceci :

```
$a = 5;
$b = 10;
function multiplier(string $x, float$y)
{
   echo 'Résultat du calcul: ' .$x * $y;
   echo '<br>';
}
multiplier($a, $b);
```

Résultat du calcul: 50

Mais pourquoi cela fonctionne… Et bien simplement car PHP va essayer de convertir $x en string. Comme il y parvient, il n'y a pas d'erreur. Si vous faites un var_dump de $x au début de la fonction, cela renvoi `string(1) "5"`

Pour éviter cela, il existe le typage strict. C'est celui qui sera utilisé en Objet. Rajoutons une ligne de paramétrage à notre fichier PHP. Attention elle doit être juste après l'ouverture de la balise <?php

```
<?php
declare(strict_types=1);
```

Maintenant, notre code précédent renvoi une belle erreur :

Fatal error: Uncaught TypeError: multiplier(): Argument #1 ($x) must be of type string, int given, called in /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php on line 19 and defined in /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php:10 Stack trace: #0 /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php(19): multiplier() #1 {main} thrown in **/home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php** on line **10**

Typage du retour d’une fonction

Comme nous l’avons vu, une fonction renvoi presque toujours une donnée. Celle ci peut également être contrôlée. Ceci en indiquant le type juste après les parenthèses et précédé de deux points

```php
function multiplier(float $x, float $y): array
```

Fatal error: Uncaught TypeError: multiplier(): Return value must be of type array, float returned in /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php:12 Stack trace: #0 /home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php(15): multiplier() #1 {main} thrown in **/home/cedric/www/ifapme-cours/X75-1/X75-1COMPLET/php/test.php** on line **12**