**PHP**

# Découvrez le fonctionnement d'un site écrit en PHP

## Les sites statiques et dynamiques

Les sites statiques : ce sont des sites réalisés uniquement à l'aide des langages HTML et CSS. Ils fonctionnent très bien mais leur contenu ne peut pas être mis à jour automatiquement.

Les sites dynamiques : plus complexes, ils utilisent d'autres langages en plus de HTML et CSS, tels que PHP et MySQL. Le contenu de ces sites web est dit « dynamique » parce qu'il peut changer sans l'intervention du webmaster ! La plupart des sites web que vous visitez aujourd'hui, y compris OpenClassrooms, sont des sites dynamiques. Le seul prérequis pour apprendre à créer ce type de sites est de déjà savoir réaliser des sites statiques en HTML et CSS.

## Comment fonctionne un site web ?

Il faut savoir qu'Internet est un réseau composé d'ordinateurs. Ceux-ci peuvent être classés en deux catégories.

Les clients : ce sont les ordinateurs des internautes comme vous. Votre ordinateur fait donc partie de la catégorie des clients. Chaque client représente un visiteur d'un site web. Dans les schémas qui vont suivre, l'ordinateur d'un client sera représenté par l'image suivante.

Les serveurs : ce sont des ordinateurs puissants qui stockent et délivrent des sites web aux internautes, c'est-à-dire aux clients. La plupart des internautes n'ont jamais vu un serveur de leur vie. Pourtant, les serveurs sont indispensables au bon fonctionnement du Web. Sur les prochains schémas, un serveur sera représenté par l'image de la figure suivante.

**Cas d'un site statique**

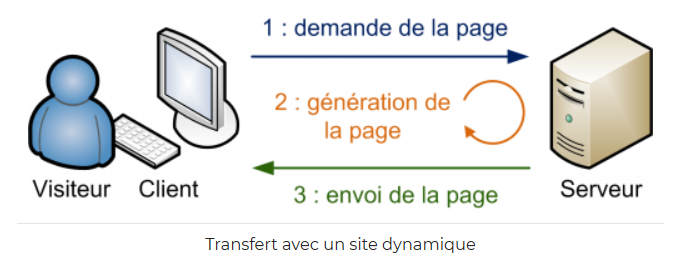
Lorsque le site est statique, le schéma est très simple. Cela se passe en deux temps, ainsi que vous le schématise la figure suivante :

1. le client demande au serveur à voir une page web ;
2. le serveur lui répond en lui envoyant la page réclamée.

**Cas d'un site dynamique**

Lorsque le site est dynamique, il y a une étape intermédiaire : la page est **générée** (figure suivante).

* Le client demande au serveur à voir une page web ;
* le serveur prépare la page spécialement pour le client ;
* le serveur lui envoie la page qu'il vient de générer.



## Les langages du WEB

Pour un site statique : HTML et CSS

Pour un site dynamique : Ajoutez PHP et MySQL

**PHP** est un langage que seuls les serveurs comprennent et qui permet de rendre votre site dynamique. C'est PHP qui « génère » la page web comme on l'a vu sur un des schémas précédents.

**MySQL**  est ce qu'on appelle un SGBD (Système de Gestion de Base de Données). Pour faire simple, son rôle est d'enregistrer des données de manière organisée afin de vous aider à les retrouver facilement plus tard. C'est grâce à MySQL que vous pourrez enregistrer la liste des membres de votre site, les messages postés sur le forum, etc. Le langage qui permet de communiquer avec la base de données s'appelle le SQL.

**PHP génère du HTLM**

Les clients sont incapables de comprendre le code PHP : ils ne connaissent que le HTML et le CSS. Seul le serveur est capable de lire du PHP.

Le rôle de PHP est justement de générer du code HTML (on peut aussi générer du CSS, mais c'est plus rare), code qui est ensuite envoyé au client de la même manière qu'un site statique, comme le montre la figure suivante.

**Les concurrents de PHP**

Parmi les concurrents de PHP, on peut citer les suivants :

ASP .NET : conçu par Microsoft, il exploite le framework (c'est-à-dire un ensemble de bibliothèques qui fournissent des services pour les développeurs) .NET.

Ruby on Rails : très actif, ce framework s'utilise avec le langage Ruby et permet de réaliser des sites dynamiques rapidement en suivant certaines conventions.

Django : il est similaire à Ruby on Rails, mais il s'utilise en langage Python.

Java et les JSP (Java Server Pages) : plus couramment appelé « JEE » ou « Java EE », il est particulièrement utilisé dans le monde professionnel.

**Les concurrents de MySQL**

Parmi les concurrents de MySQL il y a :

Oracle : c'est le SGBD le plus célèbre, le plus complet et le plus puissant. Il est malheureusement payant (et cher)

MariaDB : variante libre de MySQL, qui a été créée depuis que ce dernier a été racheté par... Oracle. MariaDB est une copie (fork) de MySQL qui a la volonté de rester libre et indépendante.

Microsoft SQL Server : édité par Microsoft, on l'utilise souvent en combinaison avec ASP .NET, bien qu'on puisse l'utiliser avec n'importe quel autre langage. Il est payant, mais il existe des versions gratuites limitées.

PostgreSQL : il s'agit d'un SGBD libre et gratuit comme MySQL, qui propose des fonctionnalités plus avancées. Parfois comparé à Oracle, il lui reste cependant du chemin à parcourir. Il dispose d'une communauté un peu moins importante que MySQL et Oracle. OpenClassrooms utilise PostgreSQL.

SQlite

# Préparez votre environnement de travail

Pour que votre ordinateur puisse lire du PHP, il faut qu'il se comporte comme un serveur. Il suffit simplement d'installer les mêmes programmes que ceux que l'on trouve sur les serveurs qui délivrent les sites web aux internautes.

**Apache** : c'est ce qu'on appelle un serveur web. Il s'agit du plus important de tous les programmes, car c'est lui qui est chargé de délivrer les pages web aux visiteurs. Cependant, Apache ne gère que les sites web statiques (il ne peut traiter que des pages HTML). Il faut donc le compléter avec d'autres programmes.

**PHP** : c'est un plug-in pour Apache qui le rend capable de traiter des pages web dynamiques en PHP. En clair, en combinant Apache et PHP, notre ordinateur sera capable de lire des pages web en PHP.

**MySQL** : c'est le logiciel de gestion de bases de données dont je vous ai parlé en introduction. Il permet d'enregistrer des données de manière organisée (comme la liste des membres de votre site). Nous n'en aurons pas besoin immédiatement, mais autant l'installer de suite.

Il y a des packs qui renferment ces trois programmes :

**WAMP** sous Windows

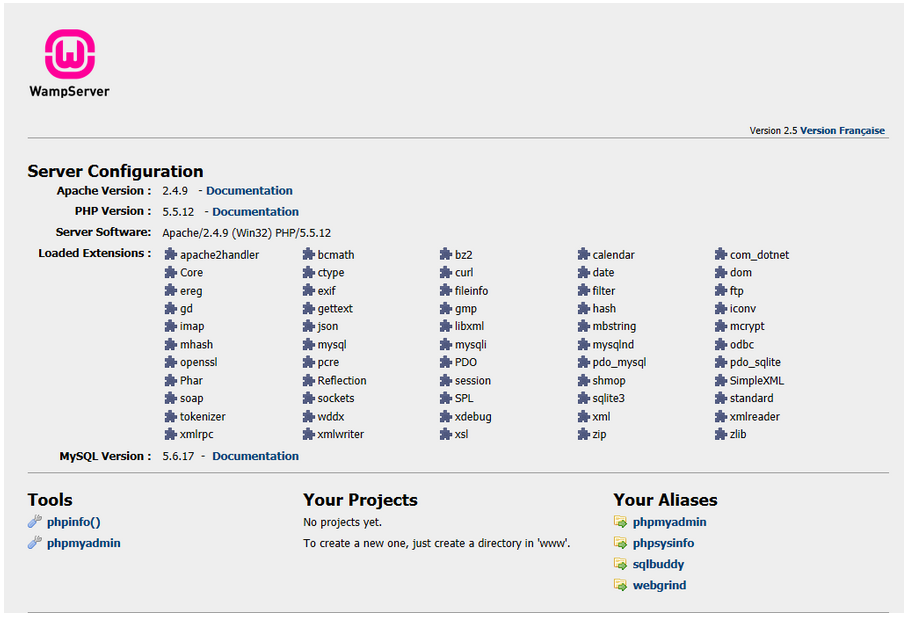
**MAMP** sous MAC

**XAMPP** sous Linux

Il faut un éditeur de texte comme **Sublime Text** ou **Brackets**

Lorsque vous lancez WAMP, une icône doit apparaître en bas à droite de la barre des tâches, à côté de l'horloge, elle passera du rouge au orange puis vert. Tant qu’elle n’est pas verte ça ne fonctionnera pas.

En allant sur <http://localhost> vous verrez la page web envoyée par votre propre serveur Apache que vous avez installé en même temps que WAMP. Vous êtes en train de simuler le fonctionnement d'un serveur web sur votre propre machine. Pour le moment, vous êtes le seul internaute à pouvoir y accéder. On dit que l'on travaille « **en local** ». Vous naviguez sur un site web situé sur votre propre ordinateur.



Ensuite il faudra aller dans C:\wamp64\www et créer un dossier par exemple tests où les codes seront mis, tu y accèderas en tapant : <http://localhost/tests>.

Sous MAC c’est dans Applications 🡪 MAMP 🡪 htdocs

Ensuite pour y accéder après avoir créé un dossier tests , vous pouvez visualiser son contenu en vous rendant à l'adressehttp://localhost:8888/tests/

**Installation sous Linux**

Sous Linux, il est courant d'installer Apache, PHP et MySQL séparément. Toutefois, il existe aussi des packs tout prêts comme XAMPP (X Apache MySQL Perl PHP), anciennement connu sous le nom de LAMPP.

Ce pack est plus complet que WAMP pour Windows ou MAMP pour Mac OS X

Sur le [site officiel de XAMPP](https://www.apachefriends.org/fr/index.html), recherchez le lien XAMPP pour Linux.

Une fois le téléchargement terminé, rendez-vous dans le dossier dans lequel vous avez téléchargé XAMPP.

Vous devez passer root pour installer et lancer XAMPP.  
root est le compte administrateur de la machine qui a notamment le droit d'installer des programmes. Normalement, il suffit de taper su et de rentrer le mot de passe root. Sous Ubuntu, il faudra taper sudo su et taper votre mot de passe habituel.

Donnez les droits d'exécution au fichier que vous venez de télécharger et lancez le programme d’installation ex : ./xampp-linux…

Pour démarrer XAMPP (et donc Apache, PHP et MySQL), tapez la commande suivante :

/opt/lampp/lampp start

Si vous désirez plus tard arrêter XAMPP, tapez :

/opt/lampp/lampp stop

Vous pouvez maintenant tester XAMPP en ouvrant votre navigateur favori et en tapant l'adresse suivante : <http://localhost>.

Vous pouvez vérifier que tout fonctionne correctement en allant dans le menu Statut

Au minimum, les modules MySQL et PHP doivent être en vert.

Les fichiers PHP devront être placés dans le répertoire/opt/lampp/htdocs. Vous pouvez y créer un sous-répertoire tests pour vos premiers tests.

Une fois le dossier créé, vous pouvez y accéder depuis votre navigateur à l'adresse suivante : <http://localhost/tests>

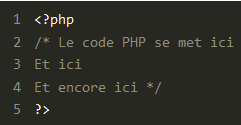
# Ecrivez votre premier script

## Les balises PHP

Le code PHP vient s'insérer au milieu du code HTML. On va progressivement placer dans nos pages web des morceaux de code PHP à l'intérieur du HTML.

**La forme d’une balise PHP**

Elle commence par <?php et se termine par ?>



## Afficher du texte

On utilise l'instruction **echo** pour afficher du texte, ex :



Enfin, l'instruction se termine par un point-virgule comme je vous l'avais annoncé, ce qui signifie **Fin de l'instruction**.

Le code suivant fonctionne :



Le mot « texte » sera affiché en gras grâce à la présence des balises<strong>et</strong>.

Comment faire pour afficher un guillemet ?

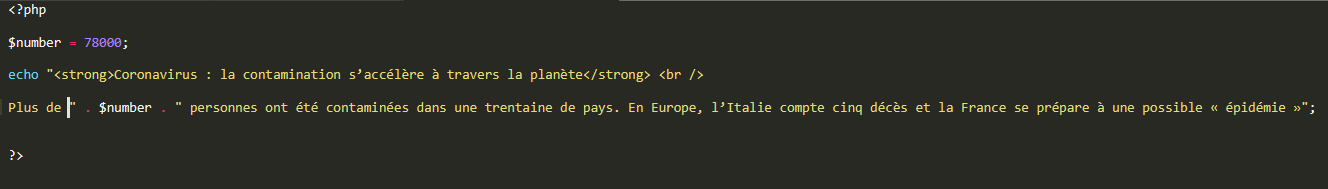
Bonne question. Si vous mettez un guillemet, ça veut dire pour l'ordinateur que le texte à afficher s'arrête là. Vous risquez au mieux de faire planter votre beau code et d'avoir une terrible « Parse error ».

La solution consiste à faire précéder le guillemet d'un antislash\



**Enregistrer une page PHP**

Il faut lui donner l’extension .php et que le fichier soit enregistré dans le dossier www ou l’un de ses sous-dossiers.



**Comment PHP génère du code HTML**

L'instruction **echo** demande à PHP d'insérer à cet endroit le texte que vous demandez

Rappelez-vous, seul le serveur peut exécuter du PHP. Le PHP n'est **jamais** envoyé au visiteur. Pour que nous puissions exécuter du PHP sur notre ordinateur (afin de faire nos tests), nous avons dû le transformer en mini-serveur en installant un programme tel que WAMP.

## Les commentaires

**Les commentaires monolignes**

Pour indiquer que vous écrivez un commentaire sur une seule ligne, vous devez taper deux slashs : «//»

**Les commentaires multilignes**

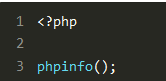
Ce sont les plus pratiques si vous pensez écrire un commentaire sur plusieurs lignes, mais on peut aussi s'en servir pour écrire des commentaires d'une seule ligne. Il faut commencer par écrire/\*puis refermer par\*/:

Configurez PHP pour visualiser les erreurs

**Localiser le fichier de configuration PHP du serveur web**

Pour connaitre l'ensemble des informations relatives au PHP utilisé par le serveur web, il existe une fonction PHP  phpinfo(). Nous allons l'utiliser pour localiser le fichier de configuration pour que nous puissions le modifier.

Je vous invite donc à créer un fichier PHP avec simplement le code qui suit :



Enregistrez-le sous le nom  info.php  dans le dossier qui est lu par votre serveur web. Et enfin, affichez la page.

Cette page contient toute une flopée d'informations dont la version de PHP, le type de serveur web et la localisation du (ou des fichiers) de configuration pour PHP.

Retrouvez la ligne "Loaded Configuration File" (ce qui signifie "fichier de configuration chargé" en anglais), et regardez la valeur. Dans mon cas, on peut voir ceci par exemple :



Il faut s'assurer que les clés de configuration  error\_reporting  et  display\_errors  ont respectivement les valeurs  E\_ALL  &  on puis si ce n’est pas le cas modifier, enregistrer et redémarrer WAMP

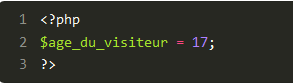
# Les variables

## Qu’est-ce qu’une variable ?

Une **variable**, c'est une petite information stockée en mémoire **temporairement**.

## Affecter une valeur à une variable

Création d’une variable



Analysons dans le détail le code qu'on vient de voir.

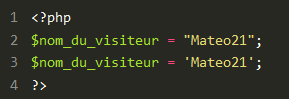
* D'abord, on écrit le symbole « dollar » ($) : il précède toujours le nom d'une variable. C'est comme un signe de reconnaissance si vous préférez : ça permet de dire à PHP « J'utilise une variable ». Vous reconnaîtrez toujours qu'il y a une variable par la présence du symbole « dollar » ($).
* Ensuite, il y a le signe « égal » (=) : celui-là c'est logique, c'est pour dire que $age\_du\_visiteur est égal à…
* À la suite, il y a la valeur de la variable, ici 17.
* Enfin, il y a l'incontournable point-virgule (;) qui permet de terminer l'instruction.

**Les différents types de variables**

Le type **string** (chaîne de caractères)

Ce type permet de stocker du texte. Pour cela, vous devez entourer votre texte de guillemets doubles "" ou de guillemets simples''(attention, ce sont des apostrophes).

Voici deux exemples, l'un avec des guillemets simples et l'autre avec des guillemets doubles :



**Le type int(nombre entier)**

On vient de l'utiliser pour nos exemples précédents. Il suffit tout simplement d'écrire le nombre que vous voulez stocker, sans guillemets.

<?php

$age\_du\_visiteur = 17;

?>

**Le type float(nombre décimal)**

Vous devez écrire votre nombre avec un point au lieu d'une virgule. C'est la notation anglaise.

<?php

$poids = 57.3;

?>

**Le type bool(booléen)**

Pour dire si une variable vaut vrai ou faux, vous devez écrire le mottrueoufalsesans guillemets autour (ce n'est pas une chaîne de caractères !). Je vous conseille de bien choisir le nom de votre variable pour que l'on comprenne ce que ça signifie. Voyez vous-mêmes :

<?php

$je\_suis\_un\_zero = true;

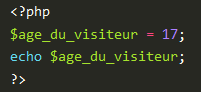
$je\_suis\_bon\_en\_php = false;

?>

## Afficher et concaténer des variables

**Afficher le contenu d'une variable**

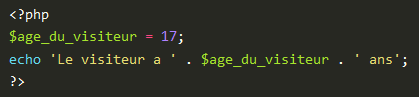
Vous vous souvenez que l'on peut afficher du texte avec **echo** ? On peut aussi s'en servir pour afficher la valeur d'une variable !



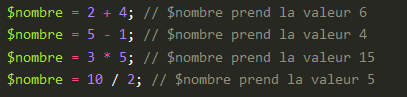
**La concaténation**

La concaténation est justement un moyen d'assembler du texte et des variables.

On utilise le point.



## Faire des calculs simples

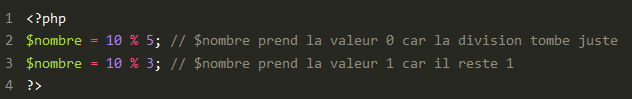


**Le modulo**

Il est possible de faire un autre type d'opération un peu moins connu : le **modulo**. Cela représente le reste de la division entière.

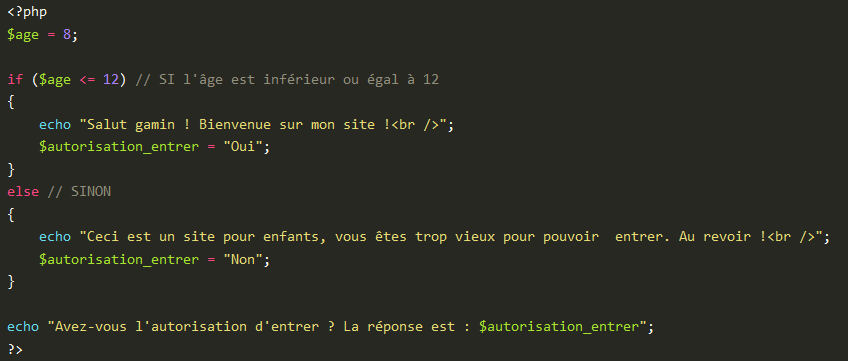
Par exemple, 6 / 3 = 2 et il n'y a pas de reste. En revanche, 7 / 3 = 2 (car le nombre 3 « rentre » 2 fois dans le nombre 7) et il reste 1. Vous avez fait ce type de calculs à l'école primaire, souvenez-vous !

Le modulo permet justement de récupérer ce « reste ». Pour faire un calcul avec un modulo, on utilise le symbole%.

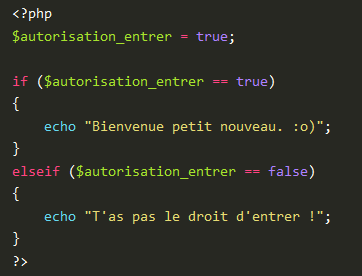


# 

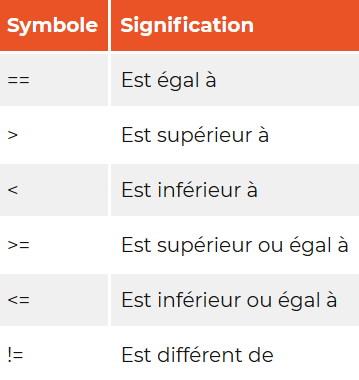
**La structure de base : if… else**



## Le cas des booléens



## Les différents symboles

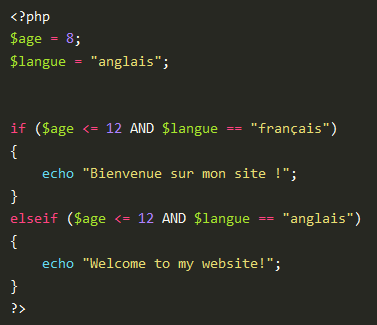


## Les conditions multiples

AND 🡪 &&

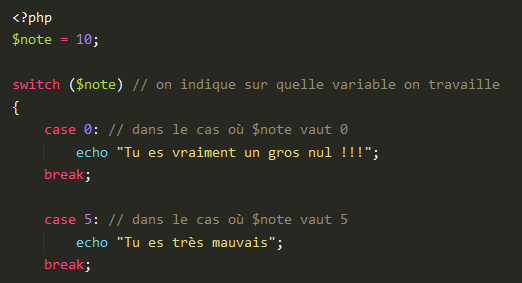
OR 🡪 ||

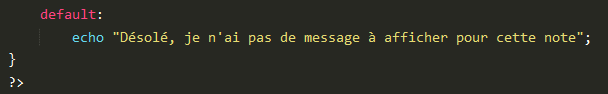
## If et elseif



## Le switch

Il permet de se passer de faire plusieurs elseif

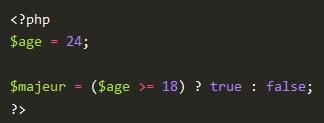




## Les ternaires : des conditions condensées

Un ternaire est une condition condensée qui fait deux choses sur une seule ligne :

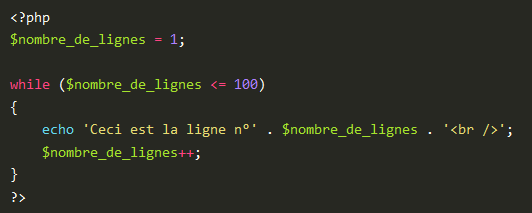
* on teste la valeur d'une variable dans une condition ;
* on affecte une valeur à une variable selon que la condition est vraie ou non.



La condition testée est $age >= 18. Si c'est vrai, alors la valeur indiquée après le point d'interrogation (ici true) sera affectée à la variable $majeur. Sinon, c'est la valeur qui suit le symbole « deux-points » qui sera affectée à $majeur.

# Les boucles

## Une boule simple : while



À noter qu'il y a une balise HTML<br />à la fin : cela permet d'aller à la ligne.

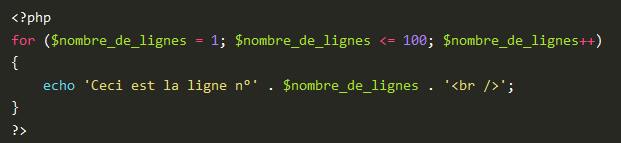
$nombre\_de\_lignes++; est **l'incrémentation**

Il faut TOUJOURS s'assurer que la condition sera fausse au moins une fois. Si elle ne l'est jamais, alors la boucle s'exécutera à l'infini !  
PHP refuse normalement de travailler plus d'une quinzaine de secondes. Il s'arrêtera tout seul s'il voit que son travail dure trop longtemps et affichera un message d'erreur.

Astuce :

Vous pouvez fermer la balise PHP?>, écrire du texte en HTML, puis rouvrir la balise PHP<?php. Cela vous évite d'utiliser une ou plusieurs instructions echo au milieu. On aura l'occasion d'utiliser cette astuce de nombreuses fois dans la suite du cours.

## Une boule plus complexe : for



Après le mot for, il y a des parenthèses qui contiennent trois éléments, séparés par des points-virgules;.

Décrivons chacun de ces éléments.

* Le premier sert à l'**initialisation**. C'est la valeur que l'on donne au départ à la variable (ici, elle vaut 1).
* Le second, c'est la **condition**. Comme pour le while, tant que la condition est remplie, la boucle est réexécutée. Dès que la condition ne l'est plus, on en sort.
* Enfin, le troisième c'est l'**incrémentation**, qui vous permet d'ajouter 1 à la variable à chaque tour de boucle.

Si on sait le nombre de fois que l’on veut répéter une instruction alors il vaut mieux utiliser la boucle for.

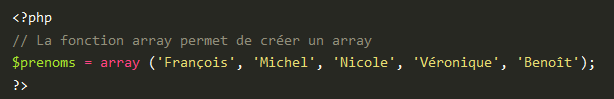
# Les tableaux

Les arrays sont des variables composées, elles ont plusieurs valeurs

## Les deux types de tableaux

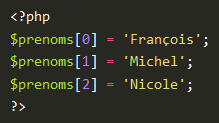
On en distingue deux types :

**Les tableaux numérotés**



Le premier élément (« François ») aura le n°0, ensuite Michel le n°1, etc.

Vous pouvez aussi créer manuellement le tableau case par case :



**Afficher un tableau numéroté**

Pour afficher un élément, il faut donner sa position entre crochets après $prenoms. Pour afficher « Michel », on doit donc écrire :

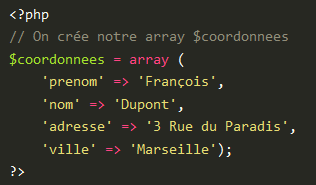


**Les tableaux associatifs**

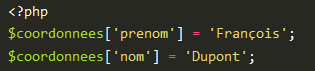
Les tableaux associatifs fonctionnent sur le même principe, sauf qu'au lieu de numéroter les cases, on va les étiqueter en leur donnant à chacune un nom différent.

**Construire un tableau associatif**

Pour en créer un, on utilisera la fonction array comme tout à l'heure, mais on va mettre « l'étiquette » devant chaque information



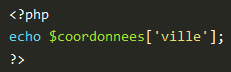
Il est aussi possible de créer le tableau case par case, comme ceci :



**Afficher un tableau associatif**

Pour afficher un élément, il suffit d'indiquer le nom de cet élément entre crochets, ainsi qu'entre guillemets ou apostrophes puisque l'étiquette du tableau associatif est un texte.

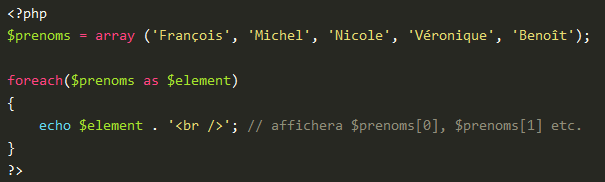
Par exemple, pour extraire la ville, on devra taper :



## Parcourir un tableau

Il est préférable d’utiliser foreach

Foreach va passer en revue chaque ligne du tableau, et lors de chaque passage, elle va mettre la valeur de cette ligne dans une variable temporaire (par exemple $element) ex :

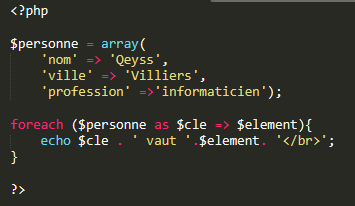


L'avantage de foreach est qu'il permet aussi de parcourir les tableaux associatifs.



Toutefois, avec cet exemple, on ne récupère que la valeur. Or, on peut aussi récupérer la clé de l'élément. On doit dans ce cas écrire foreach comme ceci :





## Rechercher dans un tableau

Nous allons voir trois types de recherches, basées sur des fonctions PHP :

* array\_key\_exists : pour vérifier si une clé existe dans l'array ;
* in\_array : pour vérifier si une valeur existe dans l'array ;
* array\_search : pour récupérer la clé d'une valeur dans l'array.

**Vérifier si une clé existe dans l'array : array\_key\_exists**

Voici notre problème : on a un array, mais on ne sait pas si la clé qu'on cherche s'y trouve.  
Pour vérifier ça, on va utiliser la fonctionarray\_key\_existsqui va parcourir le tableau pour nous et nous dire s'il contient cette clé.

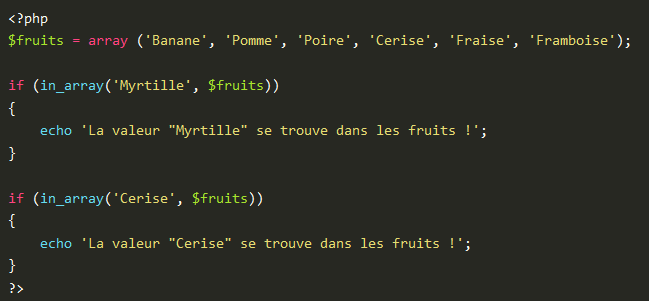
On doit d'abord lui donner le nom de la clé à rechercher, puis le nom de l'array dans lequel on fait la recherche :





**Vérifier si une valeur existe dans l'array :in\_array**

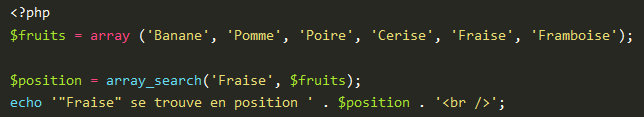
Le principe est le même quearray\_key\_exists… mais cette fois on recherche dans les valeurs.in\_arrayrenvoietruesi la valeur se trouve dans l'array,falsesi elle ne s'y trouve pas.



**Récupérer la clé d'une valeur dans l'array :array\_search**

array\_searchfonctionne commein\_array : il travaille sur les valeurs d'un array. Voici ce que renvoie la fonction :

* si elle a trouvé la valeur,array\_searchrenvoie la clé correspondante (c'est-à-dire le numéro si c'est un array *numéroté*, ou le nom de la clé si c'est un array *associatif*);
* si elle n'a pas trouvé la valeur,array\_searchrenvoiefalse



Il affichera : "Fraise" se trouve en position 4

# Les fonctions

## Qu’est-ce qu’une fonction ?

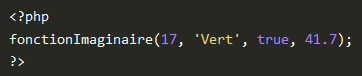
Une fonction est une série d'instructions qui effectue des actions et qui retourne une valeur.

Pour l’écrire on doit utiliser le mot clé **function**, ex : function Bonjour()

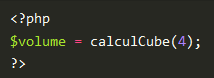
Ensuite pour l’appeler il suffit d’écrire son nom ex : Bonjour() ;

Elle peut avoir des paramètres, ex : Bonjour(nom) ;

Souvent, les fonctions acceptent plusieurs paramètres. Vous devez dans ce cas les séparer par des virgules :



Si la fonction retourne une valeur, on la récupère dans une variable, comme ceci



## Les fonctions prêtes à l'emploi de PHP

PHP propose des centaines et des centaines de fonctions prêtes à l'emploi. Sur le site officiel, la documentation PHP les répertorie toutes, classé par catégories.

Ces fonctions sont très pratiques et très nombreuses. En fait, c'est en partie là que réside la force de PHP : ses fonctions sont vraiment excellentes car elles couvrent la quasi-totalité de nos besoins.

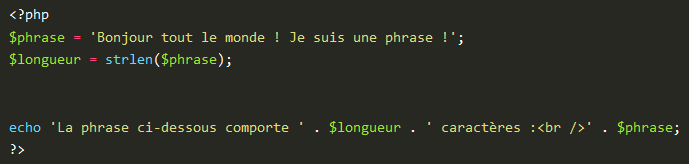
Voici un petit aperçu des fonctions qui existent pour vous mettre l'eau à la bouche :

* Une fonction qui permet de rechercher et de remplacer des mots dans une variable.
* Une fonction qui envoie un fichier sur un serveur.
* Une fonction qui permet de créer des images miniatures (aussi appeléesthumbnails).
* Une fonction qui envoie un mail avec PHP (très pratique pour faire une newsletter !).
* Une fonction qui permet de modifier des images, y écrire du texte, tracer des lignes, des rectangles, etc.
* Une fonction qui crypte des mots de passe.
* Une fonction qui renvoie l'heure, la date…
* Etc.

Quelques exemples :

**strlen**: longueur d'une chaîne

Cette fonction retourne la longueur d'une chaîne de caractères, c'est-à-dire le nombre de lettres et de chiffres dont elle est constituée (espaces compris).



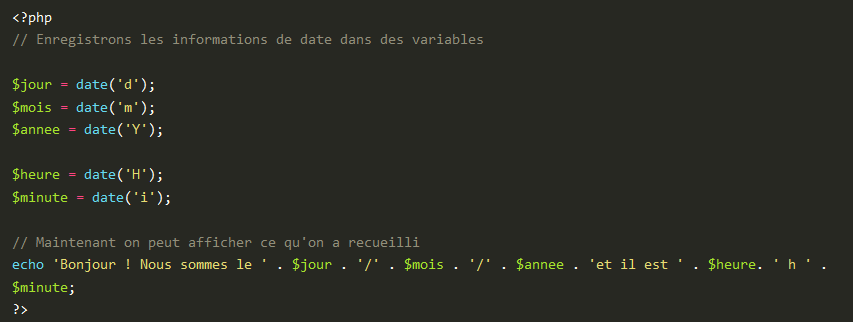
Cela va afficher 45

Méfiez-vous, il se peut que le nombre de caractères soit parfois inexact. Ceci est dû à un bug de PHP dans la gestion des encodages de caractères.

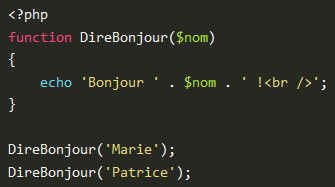
**strtolower: écrire en minuscules**

strtolower met tous les caractères d'une chaîne en minuscules.

**Récupérer la date**



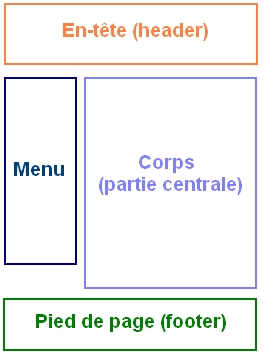
## Créer ses propres fonctions



Vous avez peut-être remarqué que la première ligne est la seule à ne pas se terminer par un point-virgule. C'est normal, il ne s'agit pas d'une instruction mais juste d'une « carte d'identité » de la fonction (son nom, ses paramètres…).

# Inclure des portions de page

La plupart des sites web sont généralement découpés selon le schéma suivant :



Au lieu d’avoir une page ainsi :

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Mon super site</title>

</head>

<body>

<!-- L'en-tête -->

<header>

</header>

<!-- Le menu -->

<nav id="menu">

<div class="element\_menu">

<h3>Titre menu</h3>

<ul>

<li><a href="page1.html">Lien</a></li>

<li><a href="page2.html">Lien</a></li>

<li><a href="page3.html">Lien</a></li>

</ul>

</div>

</nav>

<!-- Le corps -->

<div id="corps">

<h1>Mon super site</h1>

<p>

Bienvenue sur mon super site !<br />

Vous allez adorer ici, c'est un site génial qui va parler de... euh... Je cherche encore un peu le thème de mon site. :-D

</p>

</div>

<!-- Le pied de page -->

<footer id="pied\_de\_page">

<p>Copyright moi, tous droits réservés</p>

</footer>

</body>

</html>

D'une page à l'autre, ce site contiendra à chaque fois le même code pour l'en-tête, le menu et le pied de page ! En effet, seul le contenu du corps change en temps normal. Au lieu de faire ça on peut faire une page pour chaque bloc, ex pour la partie menu, on aura cette page menu.php :

<nav id="menu">

<div class="element\_menu">

<h3>Titre menu</h3>

<ul>

<li><a href="page1.html">Lien</a></li>

<li><a href="page2.html">Lien</a></li>

<li><a href="page3.html">Lien</a></li>

</ul>

</div>

</nav>

Ensuite on fait de même pour toutes les autres portions et on créé donc ainsi la page index.php

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Mon super site</title>

</head>

<body>

<?php include("entete.php"); ?>

<?php include("menus.php"); ?>

<!-- Le corps -->

<div id="corps">

<h1>Mon super site</h1>

<p>

Bienvenue sur mon super site !<br />

Vous allez adorer ici, c'est un site génial qui va parler de... euh... Je cherche encore un peu le thème de mon site. :-D

</p>

</div>

<!-- Le pied de page -->

<?php include("pied\_de\_page.php"); ?>

</body>

</html>

Ce code suppose que votre page index.php et celles qui sont incluses (comme menus.php) sont dans le même dossier. Si le menu était dans un sous-dossier appelé includes, il aurait fallu écrire

<?php include("includes/menus.php"); ?>

# Codez proprement

Des noms clairs pour les balises

Indentation du code

Un code correctement commenté

# Transmettez des données avec l’url

### Envoyez des paramètres dans l’URL

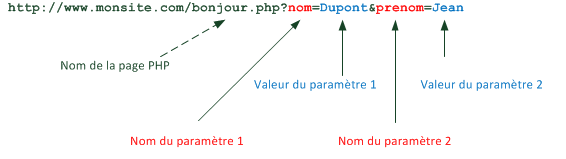
Imaginons que votre site s'appelle monsite.com et que vous avez une page PHP intitulée bonjour.php. Pour accéder à cette page, vous devez aller à l'URL suivante :

http://www.monsite.com/bonjour.php

Si on souhaite **envoyer** des informations à la page bonjour.php, on va ajouter des informations à la fin de l'URL, comme ceci :

http://www.monsite.com/bonjour.php?nom=Dupont&prenom=Jean

Ce que vous voyez après le point d'interrogation, ce sont des **paramètres** que l'on envoie à la page PHP. Celle-ci peut récupérer ces informations dans des variables. Voyez sur la figure suivante comment on peut découper cette URL.



Maintenant que nous savons cela, nous pouvons créer des liens en HTML qui transmettent des paramètres d'une page vers une autre.

Imaginons que vous avez deux fichiers sur votre site :

* index.php(l'accueil) ;
* bonjour.php.

Nous voulons faire un lien de index.php qui mène à bonjour.php et qui lui transmet des informations dans l'URL

Pour cela, ouvrez index.php(puisque c'est lui qui contiendra le lien) et insérez-y par exemple le code suivant :

<a href="bonjour.php?nom=Dupont&amp;prenom=Jean">Dis-moi bonjour !</a>

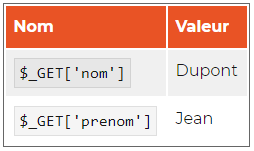
Comme vous le voyez, le & dans le code a été remplacé par &amp;dans le lien. Ça n'a rien à voir avec PHP : simplement, en HTML, on demande à ce que les & soient écrits &amp;dans le code source. Si vous ne le faites pas, le code ne passera pas la validation W3C.

Ce lien appelle la page bonjour.php et lui envoie deux paramètres :

* nom : Dupont ;
* prenom : Jean.

### Récupérer les paramètres en PHP

Intéressons-nous maintenant à la page qui réceptionne les paramètres. Dans notre exemple, il s'agit de la page bonjour.php. Celle-ci va automatiquement créer un array au nom un peu spécial :$\_GET. Il s'agit d'un array associatif dont les clés correspondent aux noms des paramètres envoyés en URL. Cela signifie que nous aurons accès aux variables suivantes :



Dans la page bonjour.php nous aurons donc ce code :

<p>Bonjour <?php echo $\_GET['prenom'] . ' ' . $\_GET['nom']; ?> !</p>

### Ne faites jamais confiance aux données reçues !

**Tester la présence d'un paramètre**

On utilise une fonction un peu spéciale :isset(). Cette fonction teste si une variable existe. Nous allons nous en servir pour afficher un message spécifique si le nom ou le prénom sont absents.

<?php

if (isset($\_GET['prenom']) AND isset($\_GET['nom'])) // On a le nom et le prénom

{

echo 'Bonjour ' . $\_GET['prenom'] . ' ' . $\_GET['nom'] . ' !';

}

else // Il manque des paramètres, on avertit le visiteur

{

echo 'Il faut renseigner un nom et un prénom !';

}

?>

# Transmettez des données avec les formulaires

On a besoin de formulaires pour échanger des informations avec nos visiteurs.

## Créer la base du formulaire

En HTML, pour insérer un formulaire, on se sert de la balise<form>. On l'utilise de la manière suivante :

<form method="post" action="cible.php">

<p>

On insèrera ici les éléments de notre formulaire.

</p>

</form>

**La méthode**

Il faut savoir qu'il existe plusieurs moyens d'envoyer les données du formulaire (plusieurs « méthodes »). Vous pouvez en employer deux.

* get : les données transiteront par l'URL comme on l'a appris précédemment. On pourra les récupérer grâce à l'array$\_GET. Cette méthode est assez peu utilisée car on ne peut pas envoyer beaucoup d'informations dans l'URL (je vous disais dans le chapitre précédent qu'il était préférable de ne pas dépasser 256 caractères).
* post : les données ne transiteront pas par l'URL, l'utilisateur ne les verra donc pas passer dans la barre d'adresse. Cette méthode permet d'envoyer autant de données que l'on veut, ce qui fait qu'on la privilégie le plus souvent. Néanmoins, les données ne sont pas plus sécurisées qu'avec la méthode GET et il faudra toujours vérifier si tous les paramètres sont bien présents et valides, comme on l'a fait dans le chapitre précédent. **On ne doit pas plus faire confiance aux formulaires qu'aux URL**.

C'est à vous de choisir par quelle méthode vous souhaitez que les données du formulaire soient envoyées. Si vous hésitez, sachez que dans 99 % des cas la méthode que l'on utilise est post, vous écrirez donc method="post" comme je l'ai fait.

**La cible**

L'attribut action sert à définir la page appelée par le formulaire. C'est cette page qui recevra les données du formulaire et qui sera chargée de les traiter.

La page cible ne doit pas forcément s'appeler cible.php. J'utilise ce nom dans mes exemples simplement pour que vous compreniez bien que c'est la page qui est appelée.  
Remarquez qu'en théorie rien n'empêche le formulaire de s'appeler lui-même. Il suffirait d'écrire action="formulaire.php". Dans ce cas, la page du formulaire doit être capable aussi bien d'afficher le formulaire que de traiter les données.

## Les éléments du formulaire

1

## Ne faites jamais confiance aux données reçues : la faille XSS

1

# Variables superglobales

1

# Session et cookies

1

1

p