MA SOLUTION FORMATION

PHP - Initial

Apprendre les outils de base de PHP

Date de création : 19/03/2025 - Version 1
Date de modification :

Table des matières

Introduction au PHP

1 Histoire

PHP est le descendant du langge **PHP/FI** créé en 1994 par le canadien Rasmus Lerdorf. Il a inventé ce langage dans le but de faire un compteur de visite pour son site web personnel.

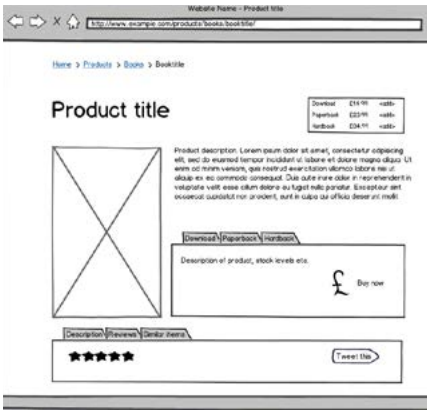
PHP a été par la suite repris par deux étudiants Andi Gutmans & Zeev Suraski pour arriver à PHP 3.0.
Puis le PHP commença à se démocratiser et à prendre en popularité.

2. Pourquoi allons-nous apprendre PHP ?

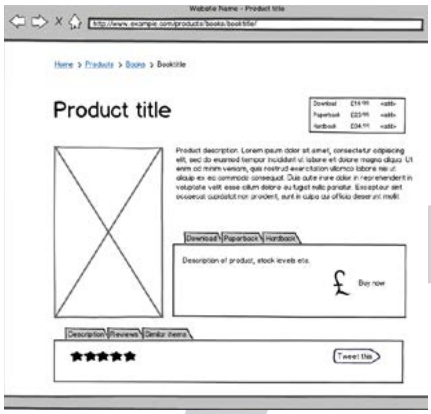
PHP est le langage côté serveur le plus utilisé sur le globe pour les sites web, on note plus de 75% des sites en ligne utilisant PHP. Dont 40% sont des sites wordpress. *Et oui Wordpress est codé en PHP :)*

3. À quoi sert PHP ?

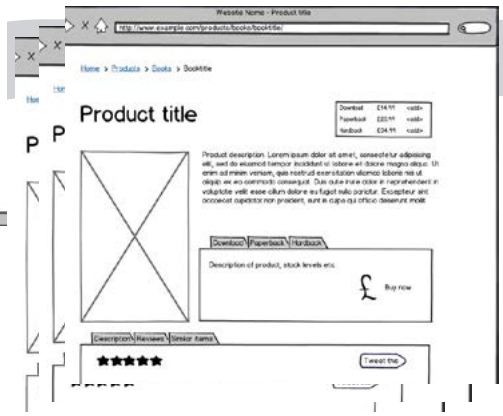
Imaginons que nous travaillons chez Amazon :



page_produit_iphone_1.html



page_produit_iphone_2.html



La solution PHP !!!

Va-t-on créer autant de pages HTML qu'il y a de produit sur amazone ?
(~ 353 Millions de produits)

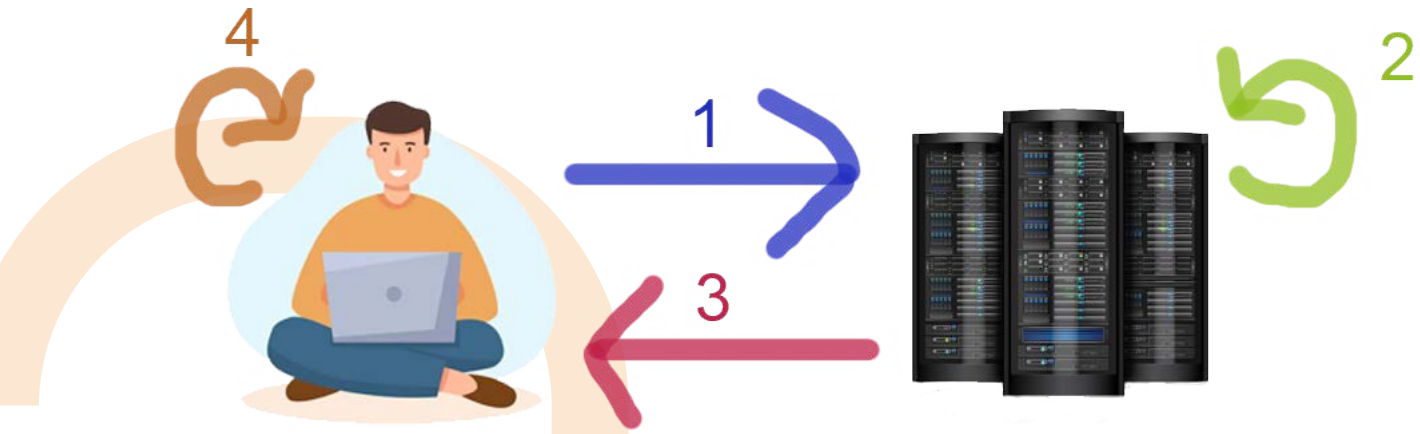
Grâce à PHP nous pouvons écrire des scripts qui récupéreront les informations envoyées par l'utilisateur et généreront un contenu html personnalisé sans réécrire la totalité du fichier à la main.

Définition script : Un script est un ensemble d'instructions ou de commandes écrites dans un langage de programmation.

Définition instruction : Une instruction informatique est une étape dans un programme informatique.

3 Explication

Comment PHP fait pour éviter de générer 353 millions de pages HTML ???



1. L'utilisateur (vous) envoie une requête (HTTP) au serveur, via une URL. (exemple : www.google.com)
2. Le serveur récupère les données à l'intérieur de la requête effectue un traitement en PHP.
3. Le serveur répond à votre requête en vous envoyant une page HTML et les éléments qui y sont attachés (exemple : image, css, javascript, etc...)
4. Le navigateur de l'utilisateur (toujours vous) affiche la page HTML reçu, et exécute le css/javascript s'il y en a.

Base de la syntaxe PHP

Ecrire un fichier PHP

Pour écrire un fichier php, il faut créer un nouveau fichier depuis vs-code, en faisant clic droit > nouveau fichier, puis nommer le fichier avec un nom qui fait sens **ET SURTOUT PRECISER L'EXTENSION DU FICHIER EN .php !!!**

Bien

Pas bien

C'est pas fini

Une fois votre fichier nommé avec le bon nom et la bonne extension, il faut à **l'intérieur** du fichier écrire le début de la balise **<?php** à **la première ligne** et **?>** à la **dernière ligne**. Sans ça votre serveur sera incapable de lire vos commandes PHP.

```php
<?php
    // Le contenu de votre script
?>
```

Voilà, vous êtes prêt pour réaliser vos premiers scripts PHP !

Version 2 date de modification : 05/12/2024

Les variables

Qu'est-ce qu'une variable ?

Définition : C'est un élément qui associe un nom (ou identifiant) à une valeur.

Oui mais encore ?

Pour faire simple une variable est une boîte avec un nom que vous lui choisissez et dans laquelle vous venez stocker une valeur. Et le nom que porte cette action se nomme la «déclaration de variable ».

```php
<?php
    $boiteNumeroUne = 2025;
    $boiteContenantLeMoisEnCours = 3;
    $monPrenom = 'Alexis';
?>
```

20253Alexis

Vous ne remarquez rien ?

Note : En PHP pour signifier à l'ordinateur que nous avons fini d'écrire une instruction, nous écrivons un point-virgule ;

Les types de variable

Qu'est-ce qu'un type de variable ?

Par défaut si vous ne dites rien à un ordinateur il peut lire la valeur «1» de plusieurs manières. Il peut croire que c'est un nombre entier, ou bien une valeur booléenne car oui true ou false peut se lire et s'écrire 1 ou 0, et l'ordinateur peut même croire que vous essayez d'écrire un message textuel avec le caractère 1.

Solution

Dire explicitement à l'ordinateur la façon dont vous voulez que votre variable soit traitée.

Vous ne remarquez rien ?

```php
<?php
    $boiteContenantUnNombre = 23;
    $boiteContenantUnNombreAVirgule = 3.60;
    $boiteContenantUnMessage = "Nous sommes en 2025";
    $boiteContenantUneValeurBooleen = true;
    $boiteContenantRien = null;
    $boiteMystere;
?>
```

Note : En PHP pour signifier à l'ordinateur que nous sommes en train d'écrire une variable, nous écrivons le symbole «$» avant le nombre de la variable.

Note 2 : La façon dont vous écrivez votre valeur définit le type de la boîte qui la contient.

Exercice pratique ! Aller voir le fichier ep_1.php

Assignation de variable

Nous venons de voir qu'il est possible de créer une boîte permettant d'enregistrer une valeur à moment T. Ce qu'on appelle la déclaration ! Mais il est également possible de modifier la valeur d'une boîte déjà existante, cela se nomme l'assignation !

```php
<?php
    $livrePrefere = 'Babare';
    $livrePrefere = 'Tintin et Milou en Belgique';
    echo $livrePrefere;
?>
```

Quelle valeur va s'afficher à l'écran ?

Note : Vous pouvez modifier une boîte avec la valeur à l'intérieur d'une autre boîte
cf (fichier exemple_6)

```php
<?php
    $boiteMagique = "Abracadabra";
    $boiteMagique = 42;
?>
```

Note : Vous pouvez modifier une boîte avec des valeurs qui ne sont pas du même type

Exercice pratique ! Aller voir le fichier ep_2.php

Opérateurs arithmétiques

Super nous savons créer des boîtes, mais qu'allons-nous faire avec ?
Voici donc les opérateurs arithmétiques, comme leurs noms l'indiquent, ils sont là pour faire des opérations arithmétiques entre deux expressions arithmétiques.

Nom	Symbole	Exemple
Soustraction	-	15-2
Addition	+	5+5
Multiplication	*	10*10
Division (sans reste)	/	130/8
Division (avec reste)	%	25%3

```php
<?php
    echo 3 + 3;
    echo '<br>';
    echo 5 * 2;
    echo '<br>';
    echo 10 - 4;
    echo '<br>';
    echo 15 / 2;
    echo '<br>';
    echo 15 % 2;
?>
```

```
6
10
6
7.5
1
```


Version 2 date de modification : 05/12/2024

Mélanger variables et expressions arithmétiques

1. Premier degré de mélange possible

Il est possible de récupérer le résultat d'une opération arithmétique et de le stocker dans une variable.

```php
<?php
    $boiteAddition = 3 + 7;
    $boiteSoustraction = 15 - 7;
    $boiteMultiplication = 3 * 9;
    $boiteDivision = 100 / 10;
    $boiteModulo = 100 % 10;
    // le mot clé echo fonctionne également avec des variables
    echo $boiteAddition;
    echo '<br>';
    echo $boiteSoustraction;
    echo '<br>';
    echo $boiteMultiplication;
    echo '<br>';
    echo $boiteDivision;
    echo '<br>';
    echo $boiteModulo;
?>
```

```
10
8
27
10
0
```

2. Second degré de mélange possible

Comme vu dans l'exemple ci-dessous, nous pouvons commencer à faire des mélanges un peu plus complexes.
Ici nous mélangeons des opérateurs arithmétiques avec des variables contenant des expressions arithmétiques et des expressions arithmétiques.

```php
<?php
    $boiteNumero1 = 30;
    $boiteNumero2 = 25;
    $boiteNumero3 = 10;

    echo $boiteNumero1 + 5 + 10;
    echo '<br>';
    echo $boiteNumero2 * 4 - 10;
    echo '<br>';
    echo $boiteNumero3 / 2 - 5;
?>
```

```
45
90
0
```

3.Troisième degré de mélange possible

Ici nous pouvons observer qu'il est possible de stocker dans une variable le résultat entre une variable contenant une expression arithmétique et une expression arithmétique. À partir de ce point vous pouvez réaliser des instructions bien plus complexe en ajoutant un nombre infini d'opérateur et d'expression arithmétique !

```php
<?php
    $boiteA = 4;
    $boiteB = $boiteA * 3;
    $boiteC = $boiteA - 2;
    echo '<br>';
    echo $boiteA;
    echo '<br>';
    echo $boiteB;
    echo '<br>';
    echo $boiteC;
    echo '<br>';
    $boiteC = 4;
    echo $boiteC * 2 + 1;
?>
```

```
4
12
2
9
```

Exercice pratique ! Aller voir le fichier ep_3.php

Opérateur arithmétique + assignation

Contrairement à la réalité, un développeur feignant est une bonne chose, moins il en fait pour réaliser une tâche, mieux c'est ! Le but des opérateurs arithmétique d'assignation est d'écrire moins code pour le même résultat.
TLDR : Faire à la fois des opérations arithmétiques et de l'assignation.

```php
<?php
    $a = 10;
    $a +=1; // Reviens à écrire $a = $a + 1; <=> $a = 10 + 1;
    echo $a."<br>";
    $b = 5;
    $b -=10; // Reviens à écrire $b = $b - 10; <=> $b = 5 - 10;
    echo $b."<br>";
    $c = 10;
    $c *=10; // Reviens à écrire $c = $c * 10; <=> $c = 10 * 10;
    echo $c."<br>";
    $d = 42;
    $d /= 7; // Reviens à écrire $d = $d / 7; <=> $d = 42 / 7;
    echo $d."<br>";
    $e = 37;
    $c %= 6; // Reviens à écrire $c = $c % 6; <=> $d = 36,6 % 6;
    echo $e."<br>";
```

```
11
-5
100
6
1
```

Et il existe également des opérateurs d'incrémentation qui permettent de compter de 1 en 1.

```php
<?php
$compteurDeVue = 99120;
$compteurDeVue ++; // Reviens à écrire $compteurDeVue = $compteurDeVue + 1;
echo $compteurDeVue;
$compteARebour = 180;
$compteARebour --;
```

```
99121
179
```

Exercice pratique ! Aller voir le fichier ep_4.php

Fabriquer des chaînes de caractères

Dans tous les langages de programmation, vous allez être amenés à créer des chaînes de caractères de manière dynamique afin de mélanger plusieurs types d'expressions (arithmétiques, booléennes, textuelles, etc...)
TLDR : Fabriquer des chaînes de caractères avec plusieurs types d'expressions.

Rappel : Une chaîne de caractères est une suite de caractères entourés de guillement double " ou simple '

Question : Quelle est la différence d'entourer notre chaîne de caractères de guillement double ou simple ?

Réponse : La prise en compte des caractères d'échappements ! Le caractère d'échapement «anti-slash» suivit de la lettre n signifie un retour à la ligne. Il en existe d'autres comme par exemple \t qui indique une tabulation.

```php
<?php
$message1 = "Hello \nworld";
$message2 = 'Hello \nworld';
```

```
Hello
world
Hello \nworld
```

Qu'est-ce qu'un caractère d'échappement

Définition : Un caractère d'échappement, en informatique et en télécommunications, est un caractère qui déclenche une interprétation alternative du ou des caractères qui le suivent.

Fabriquer des chaînes de caractères (CONCATENATION)

Note : Le guillemet simple/double est également appelé un délimiteur de string, car il indique à PHP quand la chaîne de caractères (ou string) commence et où elle se termine.

Définition : Dans le domaine de la programmation et de l'informatique, la concaténation est l'acte d'assembler deux ou plusieurs chaînes de caractères (ou des expressions) pour créer une chaîne unique.

En PHP, pour faire de la concaténation on utilise le symbole point «.» entre deux expressions pour créer une nouvelle chaîne de caractères.

```php
<?php
    $prenomUtilisateur = "Pierre ";
    $messageDeBienvenue = "Un nouvel utilisateur à rejoint le forum, bienvenue ";
    $messageFinale = $messageDeBienvenue. $prenomUtilisateur;

    echo $messageFinale;
?>
```

Un nouvel utilisateur à rejoint le forum, bienvenue Pierre

Nous pouvons également utiliser notre nouvelle chaîne de caractères directement dans le echo sans passer par une variable.

```php
<?php
    $prenomUtilisateur = "Pierre ";
    $messageDeBienvenue = "Un nouvel utilisateur à rejoint le forum, bienvenue ";
    echo $messageDeBienvenue.$prenomUtilisateur;
?>
```

Et maintenant nous allons rajouter un peu de complexité en concaténant une variable string (= chaîne de caractères) et une expression string.

```php
<?php
    $prenomUtilisateur = "Pierre";
    echo "Salut je m'appelle ".$prenomUtilisateur.", enchanté"."!";
?>
```

Salut je m'appelle Pierre, enchanté!

Note : À chaque fois que nous devons rajouter une expression à notre chaîne de caractères, il est nécessaire de rajouter un point.

Et enfin, pour construire des chaînes de caractères uniques, on peut mixer plusieurs types d'expression dans notre string.

```php
<?php
    $prenomUtilisateur = "Pierre";
    $ageUtilisateur = 31;
    echo "Salut je m'appelle ".$prenomUtilisateur.", j'ai ".$ageUtilisateur." ans, enchanté"."!";
?>
```

Salut je m'appelle Pierre, j'ai 31 ans, enchanté!

Ce n'est pas le seul moyen de concaténer, voici une autre façon de faire.

```php
<?php
    $prenomUtilisateur = "Pierre";
    $ageUtilisateur = 31;
    echo "Salut je m'appelle $prenomUtilisateur et j'ai $ageUtilisateur ans, enchanté !";
?>
```

Note : Il existe d'autres façons de faire, sprintf, stringinterpolation, mais nous verrons cela plus tard !

Exercice pratique ! Aller voir le fichier ep_5.php

Opérateurs de comparaison

Les opérateurs de comparaison, comme leur nom l'indique, comparent deux expressions. Voici le tableau de tous les opérateurs de comparaison en PHP :

Nom	Symbol	Exemple	Résultat
Égal	==	$a == $b	Vérifie que la valeur de $a est égal à la valeur de $b
Strictement égal	===	$a === $b	Vérifie que la valeur et le type $a est égale à la valeur et le type de $b
Différent	!=	$a != $b	Vérifie que la valeur de $a est différent de la valeur de $b
Strictement différent	!==	$a !== $b	Vérifie que la valeur et le type de $a est différent de la valeur et le type de $b
Inférieur	<	$a < $b	Vérifie que la valeur de $a est inférieur à $b
Inférieur ou égal	<=	$a <= $b	Vérifie que la valeur de $a est inférieur ou égale à $b
Supérieur	>	$a > $b	Vérifie que la valeur de $a est supérieur à $b
Supérieur	>=	$a >= $b	Vérifie que la valeur de $a est supérieur ou égale à $b

Note : Le résultat d'une comparaison ne peut donner qu'un résultat booléen c-à-d true ou false (ou 1 ou 0)

Comparaison simple (on vérifie que la valeur):

```php
<?php
// Note : à partir de maintenant pour vérifier nos executions nous allons utiliser var_dump au lieu de
// Car echo false n'affichera rien sur votre navigateur
var_dump(1 == 2);
var_dump(1 == "1"); // (conversion de type)
var_dump(1 != "1"); // (conversion de type)
var_dump("hello" == "hello");
var_dump("hello" == "HELLO");
var_dump(0 == false);  //(conversion de type)
var_dump("" == false);  //(conversion de type)
```

```
bool(false)
bool(true)
bool(false)
bool(true)
bool(false)
bool(true)
bool(true)
```

Comparaison stricte (on vérifie la valeur et le type):

```php
<?php
var_dump(1 === 1);
var_dump(1 === "1");
var_dump(1 !== "1");
var_dump(true === true);
var_dump(true === 1);
var_dump("hello" === "hello");
var_dump("hello" === "HELLO");
var_dump(0 === false);
var_dump("" === false);
```

```
bool(true)
bool(false)
bool(true)
bool(true)
bool(false)
bool(true)
bool(false)
bool(false)
bool(false)
```

Le choix de l'opérateur de comparaison est crucial lorsque vous travaillez avec des données provenant d'un utilisateur. Il est essentiel de sélectionner l'opérateur adapté à la situation pour garantir des résultats fiables et éviter les erreurs.

Comparaison avec des nombres :

```php
<?php
var_dump(10 > 5);
var_dump(10 < 5);
var_dump(10 >= 10);
var_dump(10 <= 9);
var_dump("10" > 5); // (conversion de type)
var_dump("10" < "5"); // (comparaison lexicographique)
```

```
bool(true)
bool(false)
bool(true)
bool(false)
bool(true)
bool(false)
```

Stocker les résultats des comparaisons dans des variables :

```php
<?php
$ageUtilisateur = 21;
$boiteTestEstCeQueUtilisateurMajeur = $ageUtilisateur >= 18;
$boiteTestEstCeQueUtilisateurMineurAuxUSA = $ageUtilisateur < 21;
var_dump($boiteTestEstCeQueUtilisateurMajeur);
var_dump($boiteTestEstCeQueUtilisateurMineurAuxUSA);
```

```
bool(true) bool(false)
```

Exercice pratique ! Aller voir le fichier ep_6.php

Opérateurs booléens (ou opérateur logique)

Rappel : Une valeur booléenne (ou logique) est soit égale à true (1) soit à false (0).
Les opérateurs booléens ont pour but de créer des expressions de comparaisons plus complexes. Voici le tableau des opérateurs booléens utilisables en PHP :

Nom	Symbol	Exemple	Résultat
ET	&&	$a && $b	Renvoie vrai (ou 1) si $a **ET** $b vallent vrai
OU	||	$a || $b	Renvoie vrai si $a **OU** $b vallent vrai
NOT	!	!$a	Inverse la valeur booléene de $a

```php
<?php
$ilFaitBeau = true;
$jaiFiniLeTravail = false;
var_dump($ilFaitBeau && $jaiFiniLeTravail);

$jaimeLesPates = true;
$jaimeLesHuitres = false;
var_dump($jaimeLesPates || $jaimeLesHuitres);
```

```
bool(false) bool(true)
```

Tableau de vérité

A	B	A && B	A || B	!A
1	1	1	1	0
0	1	0	1	1
1	0	0	1	0
0	0	0	0	1

Le tableau de vérité reflète le comportement de chaque opérateur logique.

Piège à éviter :

```php
var_dump(true && "hello"); // true (PHP convertit "hello" en true)
var_dump(true && ""); // false (PHP convertit "" en false)
```

Mettre true dans une comparaison logique n'est jamais une bonne idée car elle renverra toujours vrai à chaque exécution.
Note : Une chaîne de caractère avec au moins 1 caractère renverra toujours true dans une comparaison logique.

Pour aller plus loin :

On peut créer des expressions booléennes plus complexes grâce à la découvrete des opérateurs logiques

```php
<?php
$ageUtilisateur = 17;
$utilisateurMajeur = $ageUtilisateur >= 18;
$ageDuParentDeLutilisateur = 40;
$parentMajeur = $ageDuParentDeLutilisateur >= 18;

var_dump($utilisateurMajeur || $$ageDuParentDeLutilisateur);
// Revient à écrire $ageUtilisateur >= 18 || $ageDuParentDeLutilisateur >= 18
```

```
bool(true)
```

Exercice pratique ! Aller voir le fichier ep_7.php

Structure de contrôle

Def : En programmation informatique, une structure de contrôle est une instruction particulière d'un langage de programmation. impératif pouvant dévier le flot de contrôle du programme la contenant lorsqu'elle est exécutée.

If - else if - else

Def : Les structures de contrôle conditionnelles qui permettent d'exécuter certaines parties du code si une condition spécifique est remplie.

En PHP, la structure if (qui signifie «si» en français) permet d'exécuter un bloc de code uniquement si une condition est vraie.

Voici comment elle s'écrit :

Mot-clé if :
On commence par écrire le mot-clé if.

Condition entre parenthèses :
Après le if, on ajoute des parenthèses () qui contiennent une expression booléenne. Cette expression doit retourner true (vrai) ou false (faux).

Bloc de code entre accolades :
Ensuite, on écrit des accolades { }. À l'intérieur de ces accolades, on place le code à exécuter si la condition est vraie (true).

Pourquoi utiliser if ?

Le if permet de prendre des décisions dans votre programme, et de faire un traitement différent en fonction des données entrant dans le script.

Condition entre parenthèses

Mot clé if

```php
$ageUtilisateur = 99;
if ($ageUtilisateur >= 18) {
    echo "Vous êtes majeur";
}
```

Vous êtes majeur

Code entre accolage

```php
$meteo = "Pluie";
if ($meteo == 'soleil'){
    echo "Je vais courir";
}
```

Qu'est-ce que le script va afficher dans mon navigateur

Else (sinon)

Maintenant que nous savons comment exécuter du code si une condition est remplie, nous allons apprendre à ajouter un cas par défaut. Dans une structure de contrôle if nous pouvons ajouter le mot clé «else» (en français sinon) après les accolades de notre if afin d'ajouter du code à exécuter si la condition du if n'est pas remplie.

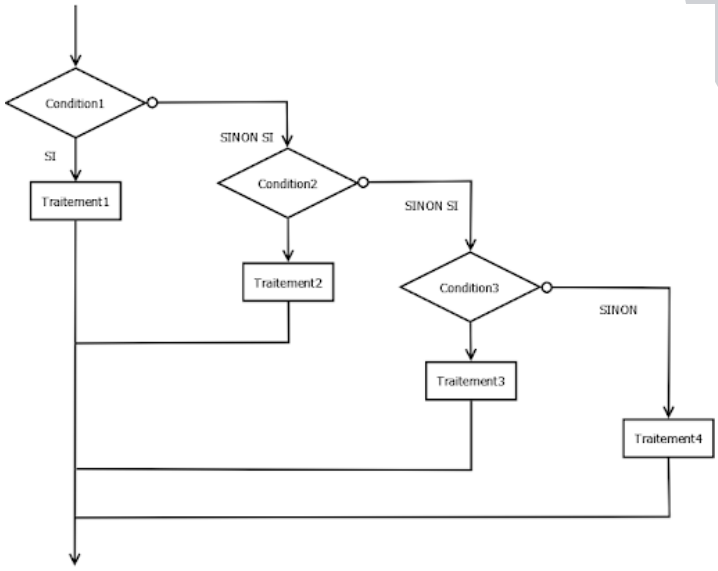
```php
<?php
$meteo = "Pluie";
if ($meteo == 'soleil'){
    echo "Je vais courir";
}else {
    echo "Je regarde un film sur le canapé";
}
echo "Je vais passer une bonne journée";
```

Je regarde un film sur le canapé
Je vais passer une bonne journée

Note : Si la condition if est vrai alors on n'exécutera pas le code présent dans les accolades else.
Note : Il n'a pas besoin d'ajouter de parenthèses après le mot clé «else» car else sert juste à indiquer du code à exécuter dans le cas où la condition de votre if n'est pas remplie.

Else if (sinon si)

Le else if donne la possibilité à votre code de tester plusieurs cas de façon séquentielle, si la première condition (if) est fausse alors on testera la condition à l'intérieur du else if.



Note : Dans une structure de contrôle if , il est obligatoire de commencer par if.
Note : Dans une structure de contrôle if, le else n'est pas obligatoire, mais s'il est présent alors il doit être à la fin du if.
Note : Dans une structure de contrôle if, vous pouvez mettre entre zéro et une infinité de else if, ces derniers doivent se situer entre le if et le else.

```php
<?php
$temperature = 22;
if ($temperature > 30){
    echo "Je vais à la plage";
} else if ($temperature > 20 ){
    echo "Je vais à la piscine";
} else if ($temperature > 10 ){
    echo "Je vais en forêt";
} else {
    echo "Je reste chez moi";
}
```

Je vais à la piscine

Switch

Le switch est une alternative au if else if else, utilisable uniquement pour tester une seule variable, elle permet d'écrire moins de code qu'un if else if.

Sturcture :

Mot-clé switch :

On commence par écrire le mot-clé switch.

Variable à tester :

Après switch, on ajoute des parenthèses () qui contiennent le nom de la variable à tester.

Accolades { } :

Ensuite, on écrit des accolades { } pour délimiter le bloc du switch.

Mot-clé case :

À l'intérieur des accolades, on utilise le mot-clé case pour indiquer une valeur possible de la variable.

Si la variable correspond à cette valeur, le code qui suit est exécuté.

Chaque case doit se terminer par un break pour éviter que PHP n'exécute les cases suivants.

Mot-clé default (optionnel) :

Le bloc default est exécuté si aucun case ne correspond à la variable.

Il sert à gérer les cas non prévus.

```php
<?php
$couleur = "rouge";
switch ($couleur) {
    case "rouge":
        echo "Votre starter pour cette aventure sera salamèche";
        break;
    case "bleu":
        echo "Votre starter pour cette aventure sera carapuce";
        break;
    case "vert":
        echo "Votre starter pour cette aventure sera bulbizar";
        break;
    default:
        echo "Votre couleur est trop compliquée. Votre starter pour cette aventure sera chenipotte";
        break;
}

$jour = "Samedi";
switch ($jour) {
    case 'Lundi':
        echo "Super c'est lundi";
        break;
    case 'Mardi':
        echo "Salut everybody tout le monde c'est mardi, c'est bientôt le week-end";
        break;
    case 'Mercredi':
        echo "Mercredi c'est ravioli";
        break;
    case 'Jeudi':
        echo "Jeudi pluvieux, jeudi heureux";
        break;
    case 'Vendredi':
        echo "C'est le dernière jour avant le week-end vivre le vendredi !";
        break;
    default:
        echo "C'est le week-end !!!!!";
        break;
}
```

Votre starter pour cette aventure sera salamèche
C'est le week-end !!!!!

Exercice pratique ! Aller voir le fichier ep_8.php

Boucle

Déf : La structure de contrôle boucle permet de répéter l'exécution d'une séquence d'instructions.

Boucle for (pour)

La boucle for (ou pour en français) permet de répéter une séquence d'instructions un nombre N de fois.

Structure de la boucle for

Mot-clé for :

On commence par écrire le mot-clé for.

Paramètres entre parenthèses :

Après for, on ajoute des parenthèses () qui contiennent trois expressions séparées par des points-virgules ; :

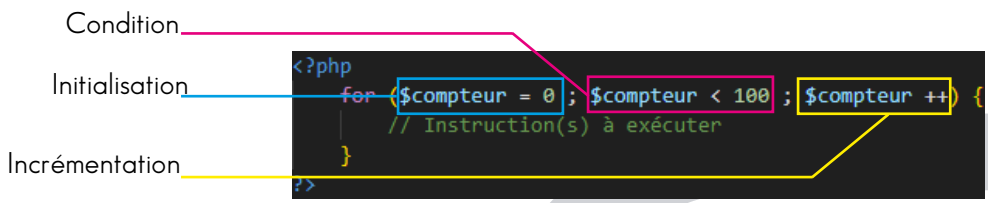
Initialisation : Une instruction exécutée une seule fois avant le début de la boucle **(ex : $i = 0).**

Condition : Une expression booléenne testée avant chaque itération. Si elle est vraie, la boucle continue ; sinon, elle s'arrête **(ex : $i < 10)**.

Incrémentation : Une instruction exécutée à la fin de chaque itération **(ex : $i++).**

Accolades { } :

Ensuite, on écrit des accolades **{ }** pour délimiter le bloc de code à répéter.



Exemple :

```php
<?php
    for ($nombre = 0 ; $nombre < 20 ; $nombre ++) {
        echo $nombre."<br>";
    }
?>
```

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

```php
<?php
    for ($rebour = 20 ; $rebour > 1 ; $rebour --) {
        echo $rebour." <br>";
    }
?>
```

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

Question : Pourquoi dans ces deux cas la boucle n'affiche pas le dernier numéro de chaque série ?

Boucle while (tant que)

La boucle while (qui signifie «tant que» en français) permet de répéter un bloc de code tant qu'une condition donnée est vraie (true). Elle est particulièrement utile lorsque vous ne connaissez pas à l'avance le nombre d'itérations nécessaires.

Structure de la boucle while

Mot-clé while :

Après while, on ajoute des parenthèses () qui contiennent une expression booléenne.

Cette condition est testée avant chaque itération.

Si elle est vraie (true), la boucle continue.

Si elle est fausse (false), la boucle s'arrête.

Accolades { } :

Ensuite, on écrit des accolades **{ }** pour délimiter le bloc de code à répéter.

```php
<?php
$compteur = 1;
while ($compteur <= 20 ){
    echo $compteur." ";
    $compteur++;
}
// ça ressemble à une boucle for nan ?
```

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

```php
<?php
$rebour = 20;
while ($rebour >= 1 ){
    echo $rebour." ";
    $rebour--;
}
```

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

Question : Qu'est-ce que le code ci-dessous va m'afficher dans le navigateur ?

```php
<?php
$compteurBis = 0;
while ($compteurBis > 10){
    echo $compteurBis." ";
    $compteurBis ++;
}
echo "Alors qu'est-ce que le programme va afficher ????";
```

Boucle do while (faire tant que)

La boucle do-while (ou faire tant que en français) ressemble beaucoup à la boucle while, la seule différence entre ces deux boucles, est que dans une boucle do while le bloc de code entre accolade est exécuté au moins une fois même si la condition entre parenthèses est fausse.

Mot-clé do :
On commence par écrire le mot-clé do.
Accolades { } :
Ensuite, on écrit des accolades { } pour délimiter le bloc de code à répéter.
Mot-clé while :
Après le bloc de code, on écrit le mot-clé while.
Condition d'arrêt :
Après while, on ajoute des parenthèses () qui contiennent une expression booléenne.
Cette condition est testée après chaque itération.
Si elle est vraie (true), la boucle continue.
Si elle est fausse (false), la boucle s'arrête.

Question : Qu'est-ce que le code ci-dessous va m'afficher dans le navigateur ?

```php
<?php
$compteurBis = 0;
do {
    echo $compteurBis." ";
    $compteurBis ++;
} while ($compteurBis > 10);
echo "Alors qu'est-ce que le programme va afficher ????";
```

Exercice pratique ! Aller voir le fichier ep_9.php

Les tableaux

Def: Les tableaux, aussi appelés arrays en anglais, sont des types de données structurés permettant de grouper des informations ensemble. A la différence des types primitifs (entiers, réels, flottants, booléens, chaînes de caractères), les tableaux peuvent stocker une ou plusieurs valeurs à la fois (de types différents).

Quelques points clefs:

Initialisation : On peut initialiser un tableau en utilisant l'opérateur d'affectation = et en entourant les valeurs séparées par des virgules avec des crochets [].

Indexation : Les éléments d'un tableau sont identifiés par leur index, qui est un entier et commence à zéro par défaut. On peut accéder à un élément particulier en utilisant les crochets avec l'index souhaité.

Boucle : On peut parcourir les éléments d'un tableau en utilisant une boucle for, while, do while ou une boucle foreach.

Taille : La fonction count permet de déterminer la taille d'un tableau, c'est-à-dire le nombre d'éléments qu'il contient.

Ajout/Suppression : On peut ajouter ou supprimer des éléments d'un tableau en utilisant les fonctions array_push, array_pop, array_unshift et array_shift.

Note : Il est bon de préciser qu'un tableau PHP et un tableau HTML sont deux choses complètement différentes. Un tableau PHP a pour fonction de stocker et manipuler des informations tandis qu'un tableau HTML sert à présenter des données sur un écran.

Création d'un tableau vide :

```php
<?php
// Création d'un tableau vide
$tableauVide = [];
// Utiliser la fonction print_r au lieu d'echo pour éviter une erreur
print_r($tableauVide);
```

```
Array ( )
```

Note : Lors de la phase de développement, vous aurez parfois besoin de savoir ce que contient un tableau, et la solution la plus simple est d'utiliser la fonction print_r (fonction créée par les développeurs de PHP) suivie de parenthèses dans lesquels vous inscrirez le nom de votre variable contenant votre tableau. Cette fonction vous affichera la structure de votre tableau comprenant index et valeurs.

Création d'un tableau avec des valeurs initiales :

```php
<?php

$groupeDeNombres = [148, 9955, 151151, 115115];

$panierDeFruits = ['Durian', 'Kiwano', 'Pitaya', 'Rambutan'];

echo '<pre>';
print_r($groupeDeNombres);
print_r($panierDeFruits);
echo '</pre>';
```

```
Array
(
    [0] => 148
    [1] => 9955
    [2] => 151151
    [3] => 115115
)
Array
(
    [0] => Durian
    [1] => Kiwano
    [2] => Pitaya
    [3] => Rambutan
)
```

index	0	1	2	3
valeur	148	9955	151151	115115

Ajouter une valeur à un tableau

Pour ajouter une valeur à un tableau déjà déclaré vous avez deux syntaxes possibles :

```php
<?php

$panierDeFruits = ['Durian', 'Kiwano', 'Pitaya', 'Rambutan'];

// Solution n°1 les brackets vide
$panierDeFruits[] = 'Fraise';

// Solution n°2 la fonction array_push
array_push($panierDeFruits, 'Poire');

echo "<pre>";
print_r($panierDeFruits);
echo"</pre>";
```

```
Array
(
    [0] => Durian
    [1] => Kiwano
    [2] => Pitaya
    [3] => Rambutan
    [4] => Fraise
    [5] => Poire
)
```

Cette manière d'ajouter un élément au tableau se nomme **l'empilement**, car on vient ajouter (ou empiler) un élément au bout du tableau.

Supprimer une valeur d'un tableau

Pour supprimer une valeur d'un tableau déjà déclaré vous avez deux syntaxes possibles:

```php
<?php

$panierDeFruits = ['Durian', 'Kiwano', 'Pitaya', 'Rambutan', 'Fraise', 'Poire'];

// Solution n°1 les brackets vide
array_splice($panierDeFruits, 2, 1);
// Solution n°2 la fonction array_push
unset($panierDeFruits[2]);

echo "<pre>";
print_r($panierDeFruits);
echo"</pre>";
```

```
Array
(
    [0] => Durian
    [1] => Kiwano
    [3] => Fraise
    [4] => Poire
)
```

Dans la première solution nous appelons la fonction array_splice(fonction créée par les développeurs chez PHP) qui a besoin de **3 éléments** pour fonctionner : **votre variable contenant le tableau à cibler**, **l'index de la cellule à partir duquel vous voulez commencer la suppression**, **et le nombre d'élements à supprimer à partir de cet index**.

Dans la seconde solution nous utilisons la fonction unset (encore créée par PHP) qui n'a besoin que d'un seul élément pour fonctionner, la case de votre tableau à supprimer on reviendra sur la notation *$panierDeFruits[2]* juste après.

Accéder à une valeur de votre tableau

Comme mentionné plus haut, vous ne pouvez plus utiliser echo pour **afficher l'entièreté** d'un tableau, mais vous pouvez toujours l'utiliser pour afficher un élément ciblé. La question est comment cibler un seul élément ? Avec la syntaxe suivante :

```php
<?php

$panierDeFruits = ['Durian', 'Kiwano', 'Pitaya', 'Rambutan', 'Fraise', 'Poire'];
echo $panierDeFruits[0];
```

```
Durian
```

On écrit le nom de la variable contenant le tableau suivi de «brackets» à l'intérieur desquels on vient inscrire l'index de l'élément à sélectionner. Dans l'exemple ci-dessus on utilise cette syntaxe pour afficher le premier élément de notre tableau $panierDeFruits.

Modifier un élément du tableau

Une fois que l'on sait comment accéder à un élément du tableau, le modifier devient trivial. Voici la syntaxe à appeler, imaginons que nous souhaitons modifier le 4ème élément de notre panier de fruits :

```php
<?php
$panierDeFruits = [
    'Durian',
    'Kiwano',
    'Pitaya',
    'Rambutan',
    'Fraise',
    'Poire'
];
echo "<pre>";
print_r($panierDeFruits);
$panierDeFruits[3] = 'Framboise';
print_r($panierDeFruits);
echo "</pre>";
```

```
Array
(
    [0] => Durian
    [1] => Kiwano
    [2] => Pitaya
    [3] => Rambutan
    [4] => Fraise
    [5] => Poire
)
Array
(
    [0] => Durian
    [1] => Kiwano
    [2] => Pitaya
    [3] => Framboise
    [4] => Fraise
    [5] => Poire
)
```

Parcourir tout les éléments d'un tableau

Dans le monde du développement, il vous arrivera souvent de vouloir effectuer un traitement pour chaque élément d'un tableau et la boucle est un élément essentiel pour effectuer ce genre de tâche. Imaginons un tableau «notesEleve» représentant toutes les notes d'un élève sur une période. Nous pouvons, grâce aux boucles, créer un script pour calculer la moyenne de cet élève ainsi qu'un message pour indiquer chaque note de cet élève.

```php
<?php
$notesEleve = [
    15,
    13,
    17,
    8,
    4,
    19,
    11,
    10.5,
];
// Contient la valeur numérique 8
$tailleDuTableauNotesEleve = count($notesEleve);
$cumulDesNotes = 0;
for ($indexNote = 0 ; $indexNote < $tailleDuTableauNotesEleve ; $indexNote++){
    $cumulDesNotes += $notesEleve[$indexNote];
    echo "La note numéro $indexNote est : ".$notesEleve[$indexNote]."<br>";
}
$moyenneGenerale = $cumulDesNotes / $tailleDuTableauNotesEleve;
echo "La moyenne générale de cet élève est de : ".$moyenneGenerale;
```

```
La note numéro 0 est : 15
La note numéro 1 est : 13
La note numéro 2 est : 17
La note numéro 3 est : 8
La note numéro 4 est : 4
La note numéro 5 est : 19
La note numéro 6 est : 11
La note numéro 7 est : 10.5
La moyenne générale de cet élève est de : 12.1875
```

Explication :

1. On crée le tableau contenant les notes.
2. On crée une variable qui contient le nombre d'éléments dans notre tableau.
3. On crée une variable (appelée en informatique un «accumulator») qui va contenir la somme des notes de notre étudiant.
4. On déclare notre boucle for.
5. On additionne les notes dans l'accumulator + on affiche un message pour l'utilisateur
6. On calcule la moyenne générale (somme des notes / nombre de notes).
7. On affiche la moyenne dans un message pour l'utilisateur.

La boucle foreach (pour chaque)

La boucle foreach est un outil en PHP conçu spécifiquement pour parcourir les tableaux (et les objets itérables) de manière simple et intuitive. Elle permet d'accéder à chaque élément d'un tableau sans avoir à gérer manuellement un index, ce qui réduit le risque d'erreurs et améliore la lisibilité du code.

Mot-clé foreach :
On commence par écrire le mot-clé foreach, qui indique à PHP qu'on veut parcourir un tableau.
Parenthèses ( ) :
Le tableau à parcourir ($tableau).
Le mot-clé as.
La variable qui stockera la valeur de l'élément courant ($valeur).
(Optionnel) Pour avoir aussi l'index : $cle => $valeur.
Accolades { } :
Ensuite, on écrit des accolades { } pour délimiter le bloc de code à répéter.

```php
<?php
foreach ($tableau as $index => $valeur) {
    // $index = clé (numérique ou associative)
    // $valeur = valeur correspondante
}

$notesEleve = [15, 13, 17, 8, 4, 19, 11, 10.5];

foreach ($notesEleve as $index => $note) {
    echo "La note n°$index est : $note<br>";
}
```

La note n°0 est : 15
La note n°1 est : 13
La note n°2 est : 17
La note n°3 est : 8
La note n°4 est : 4
La note n°5 est : 19
La note n°6 est : 11
La note n°7 est : 10.5

On préfère souvent la boucle foreach car elle est moins verbeuse que la boucle for, si on reprend notre histoire de moyenne d'élève voici ce que ce code donne avec une boucle foreach.

```php
<?php
$notesEleve = [
    15,
    13,
    17,
    8,
    4,
    19,
    11,
    10.5,
];

$cumulDesNotes = 0;
foreach ($notesEleve as $indexNote => $note) {
    $cumulDesNotes += $note;
    echo "La note numéro $indexNote est : ".$note."<br>";
}

$moyenneGenerale = $cumulDesNotes / count($notesEleve);
echo "La moyenne générale de cet élève est de : ".$moyenneGenerale;
```

La note numéro 0 est : 15
La note numéro 1 est : 13
La note numéro 2 est : 17
La note numéro 3 est : 8
La note numéro 4 est : 4
La note numéro 5 est : 19
La note numéro 6 est : 11
La note numéro 7 est : 10.5
La moyenne générale de cet élève est de : 12.1875

Exercice pratique ! Aller voir le fichier ep_10.php

Clé associative

En PHP les index (ou addresse pour simplifier) peuvent être également des chaînes de caractères c'est ce qu'on appelle des **«clés associatives»**. Pour indiquer une **«paire clé-valeur»** on écrit le nom de la clé comme une expression string suivi d'une flèche => puis la valeur associé à cette clé.

```php
<?php
$infoUtilisateur = [
    'pseudo' => 'Alex',
    'email' => 'alexis.legras@masolutionformation.com',
    'age' => 25,
];
echo '<pre>';
print_r($infoUtilisateur);
// Ajouter une clé
$infoUtilisateur['motDePasse'] = 'motdepassesupersecretdelamortquitue76000!!!';
// Modifier une clé
$infoUtilisateur['pseudo'] = 'Alexxela';
// Supprimer une clé
unset($infoUtilisateur['age']);
print_r($infoUtilisateur);
echo '</pre>';
```

```
Array
(
    [pseudo] => Alex
    [email] -> alexis.legras@masolutionformation.com
    [age] => 25
)
Array
(
    [pseudo] => Alexxela
    [email] => alexis.legras@masolutionformation.com
    [motDePasse] => motdepassesupersecretdelamortquitue76000!!!
)
```

Note : La boucle foreach est idéeale ici pour parcourir le tableau puisque l'on ne connaît pas l'index du prochain élément.

```php
<?php
$infoUtilisateur = [
    'pseudo' => 'Alex',
    'email' => 'alexis.legras@masolutionformation.com',
    'age' => 25,
];

foreach ($infoUtilisateur as $clef => $valeur) {
    echo "<b>".$clef."</b> de mon utilisateur est utilisateur <b>".$valeur."</b><br>";
}
```

pseudo de mon utilisateur est utilisateur **Alex**
email de mon utilisateur est utilisateur **alexis.legras@masolutionformation.com**
age de mon utilisateur est utilisateur **25**

Tableau multidimensionnels

Lors de l'explication des tableaux, on a vu qu'on pouvait stocker des variables appelées primitives (string, int, bool, float), mais les tableaux peuvent également stocker d'autres tableaux. C'est ce qu'on appelle un tableau multidimensionnels.

```php
<?php
$tiroir = [ 'chaussettes', 'gants', 'écharpe'];

$commode = [
    [ 'chaussettes', 'gants' ], // tiroir 1
    [ 'pull', 'chemise' ], // tiroir 2
    [ 'ceinture', 'cravate', 'bonnet'] // tiroir 3
];
```

```
Array
(
    [0] => Array
        (
            [0] => chaussettes
            [1] => gants
        )

    [1] => Array
        (
            [0] => pull
            [1] => chemise
        )

    [2] => Array
        (
            [0] => ceinture
            [1] => cravate
            [2] => bonnet
        )

)
```

```php
<?php

$classes = [
    ['Jonathan', 13, 10 , 6],
    ['Anthonin', 12, 20, 4],
    ['Julien', 8, 6, 5],
    ['Carlo', 14, 17, 10]
];

echo $classes[0][0]."<br>"; // Affiche Jonathan
echo $classes[0][2]."<br>"; // Affiche 10 (la deuxieme note de Jonathan)
echo $classes[3][0]."<br>"; // Affiche Carlo
echo $classes[2][3]."<br>"; // Affiche 5 (La derniere note de Julien)
```

Jonathan
10
Carlo
5

```php
<?php

$classes = [
    ['Jonathan', 13, 10 , 6],
    ['Anthonin', 12, 20, 4],
    ['Julien', 8, 6, 5],
    ['Carlo', 14, 17, 10]
];

echo '<ul>';
foreach ($classes as $index => $eleve) {
    foreach ($eleve as $cle => $valeur){
        echo '<li>'.$cle.'=>'.$valeur.'</li>';
    }
}
echo '</ul>';
```

- 0=>Jonathan
- 1=>13
- 2=>10
- 3=>6
- 0=>Anthonin
- 1=>12
- 2=>20
- 3=>4
- 0=>Julien
- 1=>8
- 2=>6
- 3=>5
- 0=>Carlo
- 1=>14
- 2=>17
- 3=>10

Ici on utilise une **«boucle imbriqué»**, soit deux boucles, l'une dans l'autre, pour passer sur chaque élément de chaque tableau.

Tableau multidimensionnels avec clé associative

On peut mélanger les notions précédentes dans un seul tableau afin de créer une structure d'éléments ordonnés.

```php
<?php
$classe = [
    [
        'prenom' => 'Jonathan',
        'notes' => [
            13, 10, 6
        ],
    ],
    [
        'prenom' => 'Anthonin',
        'notes' => [
            12, 20, 4
        ],
    ],
    [
        'prenom' => 'Julien',
        'notes' => [
            8, 6, 5
        ],
    ],
    [
        'prenom' => 'Carlo',
        'notes' => [
            14, 17, 10
        ],
    ],
];
// On boucle sur chaque élève disponnible dans le tableau parent
foreach ($classe as $index => $eleve) {
    $noteAccumulator = 0;
    // On boucle sur chaque note
    foreach ($eleve['notes'] as $note) {
        $noteAccumulator+= $note;
    }
    // On ajoute une nouvel clé dans chaquel élève pour ajouter la moyenne
    $classe[$index]['moyenne'] = $noteAccumulator / count($eleve['notes']);
}
echo '<pre>';
print_r($classe);
echo '</pre>';
?>


<table>
    <tr>
        <th>Prénom</th>
        <th>Notes</th>
        <th>Moyenne</th>
    </tr>
    <tbody>
    <?php
        foreach ($classe as $eleve) {
            echo '<tr>';
            echo '<td>'.$eleve['prenom'].'</td>';
            echo '<td>';
            foreach ($eleve['notes'] as $note) {
                echo $note." ";
            }
            echo '</td>';
            echo '<td>'.$eleve['moyenne'].'</td>';
            echo '</tr>';
        }
    ?>
    </tbody>
</table>
```

```
Array
(
    [0] => Array
        (
            [prenom] => Jonathan
            [notes] => Array
                (
                    [0] => 13
                    [1] => 10
                    [2] => 6
                )

            [moyenne] => 9.6666666666667
        )

    [1] => Array
        (
            [prenom] => Anthonin
            [notes] => Array
                (
                    [0] => 12
                    [1] => 20
                    [2] => 4
                )

            [moyenne] => 12
        )

    [2] => Array
        (
            [prenom] => Julien
            [notes] => Array
                (
                    [0] => 8
                    [1] => 6
                    [2] => 5
                )

            [moyenne] => 6.3333333333333
        )

    [3] => Array
        (
            [prenom] => Carlo
            [notes] => Array
                (
                    [0] => 14
                    [1] => 17
                    [2] => 10
                )

            [moyenne] => 13.666666666667
        )

)
```

Prénom	Notes	Moyenne
Jonathan	13 10 6	9.6666666666667
Anthonin	12 20 4	12
Julien	8 6 5	6.3333333333333
Carlo	14 17 10	13.666666666667

Le code ci-dessus peut être décomposé en 3 étapes, la déclaration du tableau multi-dimensionnel associatif, la boucle foreach pour calculer la moyenne et l'ajouter au tableau, puis l'affichage HTML pour l'utilisateur.

Note : Première fois dans toute la formation que l'on mélange du code HTML et PHP dans le même fichier.

Exercice pratique ! Aller voir le fichier ep_11.php

Les fonctions

Les bases

Def : Une fonction correspond à une série d'instructions qui ont été créées pour effectuer une tâche précise.

À quoi peut servir une fonction ?

Les fonctions permettent d'empaqueter du code et de le réutiliser sans avoir à réécrire ce même code.

Mot-clé function :

On commence par écrire le mot-clé function, qui indique à PHP qu'on veut créer une fonction personnalisée.
Suivie d'un nom personnalisé qui a du sens choisi par le développeur. (vous)

Parenthèses ( ) :

Après le nom de la fonction, on ajoute des parenthèses () qui peuvent contenir :
Des paramètres séparés par des virgules ($param1, $param2)
Rien si la fonction n'a pas besoin de paramètres

Accolades { } :

On écrit des accolades {} pour délimiter :
Le corps de la fonction (instructions à exécuter)
L'instruction return (optionnelle) pour renvoyer un résultat

On peut comparer une fonction à une recette de cuisine, cela se passe en deux temps. Premier temps on écrit la recette de cuisine. Dans un second temps on applique la recette, mais on peut varier la quantité en fonction du nombre de convives.

Commençons par un exemple simple, faire une fonction qui permet de faire la somme de deux nombres.

```php
<?php
    // Premier temps on écrit la recette
    function addition($nombre1, $nombre2){
        $resultat = $nombre1 + $nombre2;
        return $resultat;
    }

    // Second temps on appllique la recette avec nos quantités
    echo addition(55, 33);
?>
```

88

```php
<?php

function helloWorld(){
    echo 'Hello world';
}

helloWorld();
```

Hello world

Note : Une fonction n'est pas obligée de rendre une valeur de retour, mais si elle doit le faire alors il faut utiliser le mot clé «return» (retour en français) suivie de l'expression à renvoyer. Dans notre exemple on renvoie la valeur à l'intérieur de la variable $resultat.
Note [2]: Une fonction n'est pas obligée d'avoir de paramètres également.
Bien sûr on peut utiliser des variables en paramètres, et on peut aussi récupérer la valeur de retour d'une fonction dans une variable.

```php
<?php
function multiplicationDeAParB($a, $b) {
    return $a * $b;
}

$premierNombre = 199;
$secondNombre = 3;

$resultat = multiplicationDeAParB($premierNombre, $secondNombre);
echo $resultat;
```

597

Vous pouvez même appeler une de vos fonctions à l'intérieur d'une autre :

```php
<?php

function multiplicationDeAParB($a, $b) {
    return $a * $b;
}

function mettreUnNombreAuCarre($nombreAMettreAuCarre) {
    return multiplicationDeAParB($nombreAMettreAuCarre, $nombreAMettreAuCarre);
}

echo mettreUnNombreAuCarre(4);
```

16

D'ailleurs vous avez déjà utilisé des fonctions depuis le début de cette session PHP :
- echo(string)
- print_r(tableau)
- unset(variable)
-array_splice(tableau, number, number).
PHP possède un grand nombre de fonctions que vous pouvez appeler lors de vos scripts.
Vous pouvez les consulter sur la documentation officielle de PHP.

Exercice pratique ! Aller voir le fichier ep_12.php

Inclusion de fichiers externes

Def: L'inclusion de fichiers en PHP est un moyen d'inclure du code PHP stocké dans un fichier séparé dans une page PHP. Cela permet de séparer le code en plusieurs parties et de les réutiliser facilement sur plusieurs pages.

Il existe deux principales façons d'inclure des fichiers en PHP:
include: Cette fonction permet d'inclure un fichier et continue d'exécuter le reste du code même en cas d'erreur dans l'inclusion.

«include_once» est similaire à **«include»**, sauf qu'il vérifie si le fichier a déjà été inclus. Si le fichier a déjà été inclus, il n'est pas inclus une seconde fois, ce qui évite les erreurs de définition multiple de variables ou de fonctions.

require: Cette fonction est similaire à include, mais arrête l'exécution du script en cas d'erreur dans l'inclusion.

La fonction **require_once**, quant à elle, inclura le fichier une seule fois, même si elle est appelée plusieurs fois dans le même script. Cela évite les erreurs en cas de redéfinition de fonctions ou de variables. Si le fichier n'est pas trouvé ou si une erreur se produit, l'exécution du script s'arrêtera.

Dans cet exemple : le fichier partie 1 appelle le fichier partie 2.

Fichier : /exemple_54_fonction_5_partie_1.php

```php
<?php
require_once('./exemple_54_fonction_5_partie_2.php');
echo nombreAEuros(10);
```

Fichier : /exemple_54_fonction_5_partie_2.php

```php
<?php
function nombreAEuros($nombre){
    return $nombre." €";
}
```

Note: Include ou require un fichier depuis un autre revient à exécuter le fichier inclus à l'intérieur du fichier qui l'appelle.

```php
<?php
require_once('./exemple_55_fonction_6_partie_2.php');
echo "Hello world <br>";
```

```php
<?php
echo "Bonjour le monde !!!!";
```

Bonjour le monde !!!!Hello world

Exemple pratique : Voir fichier exemple_56_fonction_7_partie_1.php dans lequel il y a une navbar designer ainsi que le head de ma page html, ce bout de code pourra désormais être appelé par d'autres fichiers comme le exemple_56_fonction_7_partie_2.php et exemple_56_fonction_7_partie_3.php afin de générer à chaque fois le même header mais pas le même contenu de body.

Exercice pratique ! Aller voir le fichier ep_13.php

PHP & HTML

Le but de cette section est de vous montrer comment utiliser tous les outils vus précédemment (depuis les variables jusqu'aux fonctions) pour générer du contenu HTML en fonction des données fournies à l'entrée de notre script.

Syntaxe #1

Il existe plusieurs moyens de mélanger du PHP avec du HTML. La première méthode consiste à alterner les balises HTML et PHP en utilisant la fonction echo pour générer du HTML.

Note ! : Utiliser la fonction echo avec une chaîne de caractères contenant des balises HTML revient à écrire du code HTML capable d'être interprété par votre navigateur. (On l'a déjà vu dans certains exemples)

Note !!: Les fichiers PHP peuvent interpréter le HTML.

```php
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple 57 - Methode 1</title>
</head>
<body>
    <h1>Méthode numéro alterner des balises php et html</h1>
    <p>Balise écrit en HTML</p>
    <?php echo '<p>Balise écrit en PHP</p>'  ?>
</body>
</html>
```

Méthode numéro alterner des balises php et html

Balise écrit en HTML

Balise écrit en PHP

Vous pouvez exécuter l'exemple depuis votre poste et inspecter la page via l'inspecteur d'éléments de votre navigateur. Vous remarquerez que votre navigateur interprète bien la balise p écrite via le echo.

Syntaxe #2

Cette syntaxe est plus courte et s'écrit avec une balise PHP différente, elle peut être utilisée uniquement avec une seule expression par balise. On utilise les symboles **<?= pour ouvrir la balise et ?> pour la fermer** et ce qu'il y a entre les deux doit être une chaîne de caractères.

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple 57 - Methode 1</title>
</head>
<body>
    <h1>Méthode numéro alterner des balises php et html</h1>
    <p>Balise écrit en HTML</p>
    <?= '<p>Balise écrit en PHP</p>'  ?>
</body>
</html>
```

Méthode numéro alterner des balises php et html

Balise écrit en HTML

Balise écrit en PHP

Cette syntaxe revient à écrire **<?php echo '...' ?>**.
Point fort de cette syntaxe, elle permet d'écrire une expression plus lisible et plus simple à comprendre.
Point faible vous ne pouvez utiliser qu'un seul affichage par balise.

```php
<?php
    $prenom = 'Alexis';
    $dateDuJour = date('d/m/Y  H:i:s');
?>

<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple 57 - Methode 1</title>
</head>
<body>
    <h1>Méthode numéro alterner des balises php et html</h1>
    <!-- Methode 1 -->
    <?php
        echo "<p>Bonjour ".$prenom." bienvenue sur mon site</p>";
        echo "<p>Ajourd'hui nous somme le : ".$dateDuJour."</p>";
    ?>
    <hr>
    <!-- Methode 2 -->
    <?= "<p>Bonjour ".$prenom." bienvenue sur mon site</p>"  ?>
    <?= "<p>Ajourd'hui nous somme le : ".$dateDuJour."</p>" ?>
    <hr>
    <!-- Methode 3 -->
    <?= "<p>Bonjour ".$prenom." bienvenue sur mon site</p><p>Ajourd'hui nous somme le : ".$dateDuJour."</p>"  ?>
</body>
</html>
```

Bonjour Alexis bienvenue sur mon site

Ajourd'hui nous somme le : 25/03/2025 - 09:49:32

Bonjour Alexis bienvenue sur mon site

Ajourd'hui nous somme le : 25/03/2025 - 09:49:32

Bonjour Alexis bienvenue sur mon site

Ajourd'hui nous somme le : 25/03/2025 - 09:49:32

Rendu conditionel

Le rendu conditionnel permet d'afficher du contenu seulement si une ou plusieurs conditions sont présentes. Nous allons donc utiliser le if avec les syntaxes vues dans la notion précédente.

Syntaxe #1

```php
<?php
   $ageUtilisateur = 16;
?>

<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple 60</title>
</head>
<body>
    <h1>Casino</h1>
    <?php
        if ($ageUtilisateur > 17) {
            echo '<h4> Bienevnue sur le site du casino </h4>';
        } else {
            echo "<p>Vous n'avez pas l'âge requis pour venir jouer ici</p>";
            echo "<a href='https://www.google.com'> Faire demi-tour</a>";
        }
    ?>
</body>
</html>
```

Casino

Vous n'avez pas l'âge requis pour venir jouer ici

Faire demi-tour

Syntaxe #2

```php
<?php
   $ageUtilisateur = 16;
?>

<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple 60</title>
</head>
<body>
    <h1>Casino</h1>
    <?php if ($ageUtilisateur > 17) : ?>
        <h4> Bienevnue sur le site du casino </h4>
    <?php else: ?>
        <p>Vous n'avez pas l'âge requis pour venir jouer ici</p>
        <a href='https://www.google.com'> Faire demi-tour</a>
    <?php endif; ?>
</body>
</html>
```

Pour faire un rendu conditionnel avec cette syntaxe plusieurs choses importantes sont à retenir :

- Pour écrire un if imbriqué, on utilise une balise php pour le if, une pour chaque elseif, une pour le else et enfin une pour le endif qui indique la fin de notre arbre de condition.
- Pour chaque condition (if, elseif, else) il est important d'ajouter le symbole «:» avant de fermer la balise.
- Il faut fermer le if avec une balise endif qui doit contenir le symbole «;» avant de fermer la balise.

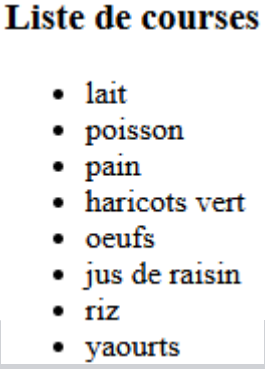
Cette syntaxe permet une meilleure lisibilité, sépare clairement le HTML du PHP et permet une meilleure gestion des balises. (fermeture, ouverture de balise)

```php
<?php
$couleur = 'red';
?>

<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple 62</title>
</head>

<body>
    <?php if ($couleur == 'green') : ?>
        <h4 style="color: green"> Hello world </h4>
    <?php elseif ($couleur == 'blue'): ?>
        <h4 style="color: blue"> Hello world </h4>
    <?php elseif ($couleur == 'purple'): ?>
        <h4 style="color: purple"> Hello world </h4>
    <?php else : ?>
        <h4 style="color:red;">Hello world</h4>
    <?php endif; ?>
</body>

</html>
```

Rendu boucle

Syntaxe #1

```php
<body>
    <h3>Liste de courses</h3>
    <ul>
        <?php
            foreach ($listeDeCourse as $itemDeCourse) {
                echo '<li>'.$itemDeCourse.'</li>';
            }
        ?>
    </ul>
</body>
```



Syntaxe #2

Points à retenir pour faire du rendu de boucle avec cette syntaxe :
- Une balise pour déclarer le début de la boucle.
- Une balise pour déclarer la fin de la boucle.
- Ne pas oublier le symbole «:» après la déclaration de la boucle.
- Ne pas oublier le symbole «;» à la fin de la fermeture de la boucle.

```php
<body>
    <h3>Liste de courses</h3>
    <ul>
        <?php foreach($listeDeCourse as $itemDeCourse): ?>
            <li><?= $itemDeCourse ?></li>
        <?php endforeach ?>
    </ul>
</body>
```

Liste de courses

- lait
- poisson
- pain
- haricots vert
- oeufs
- jus de raisin
- riz
- yaourts

Fonctionne également avec d'autres types de boucle :

```php
<?php
$listeDeCourse = [
    'lait',
    'poisson',
    'pain',
    'haricots vert',
    'oeufs',
    'jus de raisin',
    'riz',
    'yaourts'
];
?>

<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple 65</title>
</head>

<body>
    <h3>Liste de courses</h3>
    <ul>
        <?php for($i = 0 ; $i < count($listeDeCourse) ; $i++): ?>
            <li><?= $listeDeCourse[$i] ?></li>
        <?php endfor; ?>
    </ul>
</body>

</html>
```

Boucle for

```php
<?php
$listeDeCourse = [
    'lait',
    'poisson',
    'pain',
    'haricots vert',
    'oeufs',
    'jus de raisin',
    'riz',
    'yaourts'
];
$index = 0;
?>

<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple 65</title>
</head>

<body>
    <h3>Liste de courses</h3>
    <ul>
        <?php while($index < count($listeDeCourse)): ?>
            <li><?= $listeDeCourse[$index] ?></li>
            <?php $index++; ?>
        <?php endwhile; ?>
    </ul>
</body>

</html>
```

Boucle while

Note : Il existe d'autres fonctionnalités d'imbrication PHP/HTML comme **<?php include 'nomdunfichier.php'?>, les fonctions de rendu, l'héritage de «layout», mais on abordera ça un peu plus tard.**

Exercice pratique ! Aller voir le fichier ep_1 4.php

Formulaire

Cette section va traiter de la gestion des données envoyées par un utilisateur via un formulaire. Le but ici est de traiter, vérifier, et néttoyer les données avant un envoi en base de données ou sur des plateformes tierce. La règle d'or sur cette notion : «Quand on travaille avec des données utilisateurs on **NE FAIT JAMAIS CONFIANCE À UN UTILISATEUR, JAMAIS»!**

Voici le scénario que vous appliquerez quand vous travaillerez avec des formulaires :

1. Écrire/Modifier le formulaire HTML avec les bons attributs **action** et **method.**
2. Nettoyer les données de caractères indésirables.
3. Vérifier si la donnée envoyée respecte le format désiré.
4. Afficher un feedback (retour) visuel à l'utilisateur une fois le formulaire traité.

Etape #1

La balise HTML form permet d'indiquer un formulaire sur votre page et regroupera toutes vos entrées de formulaire (ou input en anglais) dans un seul envoi de données lors de la soumissions. Les premiers éléments à renseigner dans un formulaire sont les attributs **action** et **method**.

action : Représente l'**URL ou le chemin** du fichier à qui envoyer les données pour qu'il fasse le traitement.
method : Représente la façon dont vous voulez envoyer vos données. Il en existe deux (pour l'instant) **GET** et **POST**.

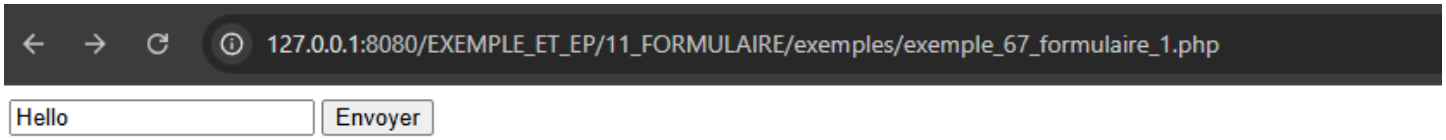
Method

Pour vous montrez la différence entre POST et GET nous allons faire un exemple avec GET en premier puis POST en second pour voir ce qu'il se passe à l'écran.

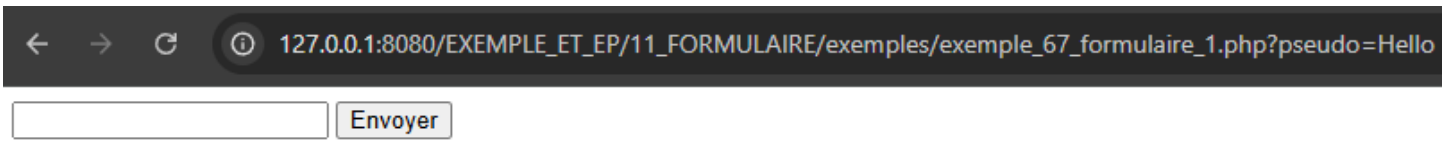
```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple 67</title>
</head>
<body>
    <form action="" method="GET">
        <input type="text" name="pseudo">
        <input type="submit" value="Envoyer">
    </form>
</body>
</html>
```

Note : Lorsque l'on écrit un attribut action vide cela revient à dire que le fichier qui va traiter les données et le même fichier qui envoie les données, soit le fichier dans lequel le formulaire est écrit.

Avant validation :



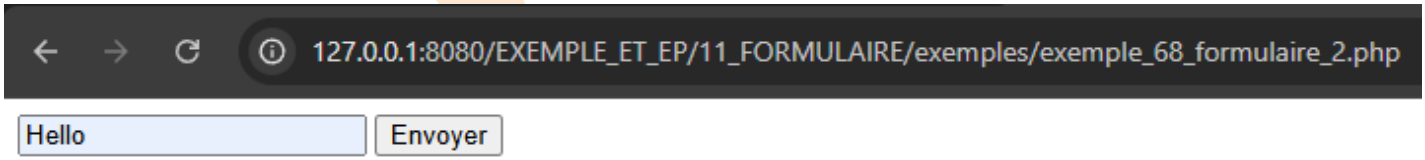
Après validation :



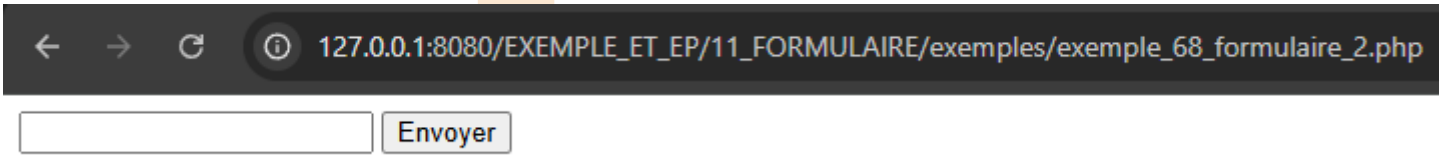
Maintenant même chose mais avec la method POST :

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple 67</title>
</head>
<body>
    <form action="" method="POST">
        <input type="text" name="pseudo">
        <input type="submit" value="Envoyer">
    </form>
</body>
</html>
```

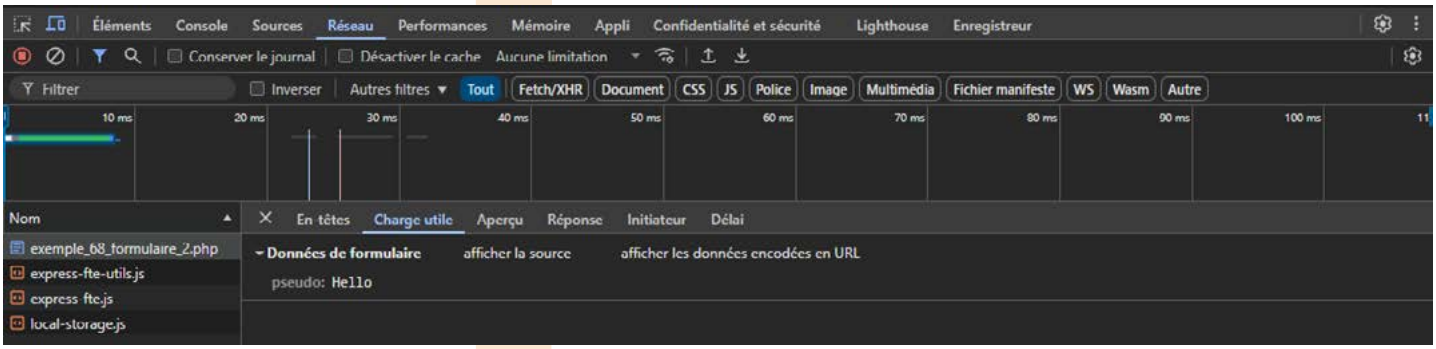
Avant validation :



Avant validation :



On peut penser que rien n'a été envoyé, mais l'envoi des données a juste été camouflé à l'oeil de l'utilisateur, on peut néanmoins retrouver ce qu'on a envoyé dans l'onglet réseau de la console développeur.



Dans la majorité de vos formulaires il est donc préférable d'utiliser la method post car elle évite l'affichage de données sensibles dans l'URL.

Action

Maintenant qu'on a vu avec quelle method on a envoyé nos données il est temps de savoir à quel fichier l'envoyer!. C'est là que notre attribut action entre en jeu. On a observé dans notre exemple précédent que si l'action était vide alors le fichier qui allait recevoir les données du formulaire est le même fichier qui les a envoyées. Cela peut nous arranger dans certains cas mais pas tout le temps. Nous allons donc voir comment définir une action capable de rediriger nos données vers un fichier de traitement.

Pour cet exemple nous allons utiliser deux fichiers, le premier affichera le formulaire le second affichera simplement les données envoyées, on le modifiera plus tard pour faire un réel traitement.

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple 68</title>
</head>
<body>
    <form action="./traitement.php" method="POST">
        <input type="text" name="pseudo">
        <input type="submit" value="Envoyer">
    </form>
</body>
</html>
```

formulaire.php

```php
<?php
    var_dump($_POST);
?>
```

traitement.php

Dans notre formulaire.php on a notre action définie à «./traitement.php» cela veut dire que lorsque notre utilisateur appuiera sur le bouton «Envoyer» les données du formulaire iront directement au fichier traitement.php.

Note : Cette notation «./traitement.php» ne fonctionne uniquement que si formulaire.php et traitement.php sont dans le même dossier.
Note : Action peut également contenir une URL cela signifie que l'on envoie les données à l'URL indiquée.

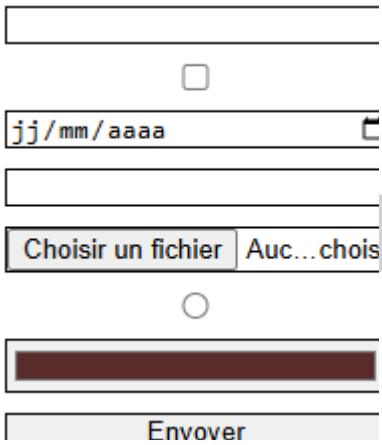
Maintenant, attardons-nous sur le fichier traitement.php. Son rôle est de simplement afficher ce que contient le formulaire et grâce à notre variable $_POST (qui est déjà déclarée par PHP) nous pouvons lire les données avec notre var_dump.
Note : Ce genre de variable ($_POST, $_GET, $_FILES, etc...) se nomment des variables super global et elles sont déjà définies par défaut dans chacun de vos fichiers PHP.

Définir les champs de formulaire

Maintenant que l'on a compris comment préparer notre envoi des données avec la method et l'action, il est temps de définir les données à envoyer. Pour ce faire, nous utilisons les éléments HTML input pour créer des champs de formulaires.

formulaire.php

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple 68</title>
    <style>···
    </style>
</head>
<body>
    <form action="./traitement.php" method="POST">
        <input type="text" name="text_input">
        <input type="checkbox" name="check_box">
        <input type="date" name="date_input">
        <input type="email" name="email_input">
        <input type="file" name="file_input">
        <input type="radio" name="radio_input">
        <input type="color" name="color_input">
        <input type="submit" value="Envoyer">
    </form>
</body>
</html>
```



traitement.php

```php
<pre>
<?php
    print_r($_POST);
?>
</pre>
```

```
Array
(
    [text_input] =>
    [date_input] =>
    [email_input] =>
    [file_input] =>
    [color_input] => #5a2b2b
)
```

Il est important de choisir le bon type de champ de formulaire pour chaque champ. L'élément input possède l'attribut type qui vous permet de sélectionner un type de données. Il y a un certain nombre de types présents dans l'exemple mais il en existe encore plus. (Lien liste types d'input possible)

Une fois le type défini dans votre input, vous devez lui définir un nom (name en anglais) pour récupérer cette donnée dans le fichier traitment.php via la variable $_POST. À chaque fois qu'un champ input est rajouté avec un attribut name, une paire clé => valeur est ajoutée dans la variable $_POST. Sauf rare exceptions comme la checkbox qui n'apparaît pas si elle n'est pas cochée et pareil pour le champ de formulaire de type radio.

Il est important de mettre l'attribut name à un élément input, sans ça vous ne pouvez le récupérer du côté traitement !

Etape #2 Traitement des données

Récupérer la donnée

Une fois le formulaire soumis, c'est au fichier de traitement de récupérer et d'effectuer des traitements.

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scal
    <title>Exemple 71</title>
    <style>···
    </style>
</head>
<body>
    <form action="./traitement.php" method="POST">
        <label for="">Pseudo</label>
        <input type="text" name="pseudo" required>
        <label for="">Email</label>
        <input type="email" name="mail_utilisateur" required>
        <label for="">Mot de passe</label>
        <input type="password" name="mot_de_passe" required>
        <input type="submit" value="Envoyer" name="submit_button">
    </form>
</body>
</html>
```

```php
<?php
    print_r($_POST);
    $pseudoUtilisateur = $_POST['pseudo'];
    echo $pseudoUtilisateur;
?>
```

Quand un visiteur remplit un formulaire sur notre site, toutes les informations qu'il a saisies sont envoyées à votre script PHP. Ces données sont stockées dans une variable spéciale appelée $_POST.

Comment $_POST fonctionne ?

C'est un tableau associatif
Chaque information est rangée dans une case avec une clé

La clé correspond au nom (name=) que vous avez donné au champ dans votre formulaire HTML

Pour accéder à une information précise
Vous utilisez sa clé (le nom du champ) entre crochets :
$_POST['nom_du_champ']

Dans l'exemple ci-dessus, dans le fichier traitement, on récupère le pseudo du formulaire pour le stocker dans une variable.
Note: On préfère d'abord stocker les données dans des variables pour les modifier par la suite si besoin. Car il est déconseillé de modifier une variable superglobal comme $_POST.

Vérifier la présence de la donnée

Dans l'écrasante majorité des sites lorsque vous essayez de vous connecter sans email, le site va vous empêcher de vous connecter. Ici on veut faire la même chose et apprendre à vérifier si les données sont présententes.

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple 72</title>
    <style>
        input {
            border : 1px solid;
            display: block;
            width: 10%;
            margin: 10px 0px;
        }
    </style>
</head>
<body>
    <form action="./traitement.php" method="POST">
        <label for="">Pseudo</label>
        <input type="text" name="pseudo">
        <label for="">Email</label>
        <input type="email" name="mail_utilisateur" required>
        <label for="">Mot de passe</label>
        <input type="password" name="mot_de_passe" required>
        <input type="submit" value="Envoyer" name="submit_button">
    </form>
</body>
</html>
```

```php
<?php
    $donneesFormulaire = $_POST;
    $pseudo = null;
    if (isset($donneesFormulaire['pseudo']) && $donneesFormulaire['pseudo'] !== ''){
        $pseudo = $donneesFormulaire['pseudo'];
    }
?>

<!DOCTYPE html>
<html lang="72">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemple 72</title>
</head>
<body>
    <p>
        <?php
            if ($pseudo !== null) {
                echo "Votre pseudo ".$pseudo." a bien été enregistré !";
            }else{
                echo "Formulaire incomplet pseudo manquant !";
            }
        ?>
    </p>
</body>
</html>
```

Dans notre fichier de traitement, on découvre la fonction isset (créée par PHP) qui a pour but de tester si une variable/ ou une cellule de tableau à une clé donnée existe. Elle renvoie true si elle existe et false si elle n'existe pas. Dans cet exemple on teste **si notre donnée existe dans le formulaire et également si elle n'est pas vide !!! Car une donnée peut exister mais être une chaîne de caractères vide ''.**

Nettoyer la donnée (sanitizing)

Quand notre site reçoit des données (formulaires, URLs, etc.), des utilisateurs malveillants peuvent essayer d'y injecter du code ou des caractères dangereux.
Le sanitizing (nettoyage) permet de :

- Supprimer les risques de piratage (comme les attaques XSS)
- Garantir que les données sont «propres» avant utilisation.

PHP propose tout un tas de fonctions pour nettoyer vos données (sanitize en anglais)

```php
<?php
$texte1 = "<script>alert('Hack!')</script>";
$cleanTexte1 = htmlspecialchars($texte1);
// Affiche : &lt;script&gt;alert(&#039;Hack!&#039;)&lt;/script&gt;
$texte2 = "<p>Bonjour <strong>à tous</strong></p>";
$cleanTexte2 = strip_tags($texte2);
// Résultat : "Bonjour à tous"
$texte3 = "   Bonjour   ";
$cleanTexte3 = trim($text3);
// Résultat : "Bonjour"
```

htmlspecialchars
Convertit les caractères spéciaux en entités HTML

strip_tags
Supprime toutes les balises HTML/PHP

trim
Enlève les espaces inutiles en début/fin de chaîne

```php
<?php
$donneesFormulaire = $_POST;
$email = null;
if (isset($donneesFormulaire['mail_utilisateur']) && $donneesFormulaire['mail_utilisateur'] !== '') {

    $email = htmlspecialchars($donneesFormulaire['mail_utilisateur']);
    $email = strip_tags($email);
    $email = trim($email);
}
?>
```

Vérifier le format de la donnée

Avant de traiter des données (emails, numéros, etc.), il est crucial de vérifier qu'elles correspondent au format attendu. Cela permet d'éviter :

- Les erreurs dans votre application.
- L'envoi de données incohérentes.
- Les tentatives de piratage.

Encore une fois PHP nous propose plein d'outils déjà faits ! Comme la fonction filter_var, ou bien is_numeric.

```php
<?php
$email = "utilisateur@example.com";

if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Email valide !";
} else {
    echo "Email invalide !";
}

$age = "25";

if (is_numeric($age)) {
    echo "Nombre valide !";
} else {
    echo "Ce n'est pas un nombre !";
}

$url = "https://example.com";

if (filter_var($url, FILTER_VALIDATE_URL)) {
    echo "URL valide !";
} else {
    echo "URL invalide !";
}
```

filter_var

est une fonction intégrée à PHP qui permet de :

- Valider si une donnée respecte un format spécifique. (email, URL, nombre entier, etc.)
- Nettoyer une valeur en supprimant les caractères indésirables.

is_numeric

vérifie si une variable est :

- Un nombre (entier ou décimal)
- Une chaîne numérique (ex: «123»)

Note : la fonction filter_var utilise ce que l'on appelle un flag (drapeau en français), le fameux FILTER_VALIDATE_EMAIL, un flag est simplement une variable «constante» définie par PHP. Ici, elle indique le type de filtrage à faire sur la variable. Voici d'autres flags disponibles pour la fonction filter_var.

Exercice pratique ! Aller voir le fichier ep_15.php

MYSQL PDO

Se connecter à la BDD

```php
<?php
$connexionBDD = null;
$hote = "mysql:host=learning-env-mariadb-dev-1;dbname=main-database";
$utilisateur = "root";
$motDePasse = "my-secret-pw";
try {
    $connexionBDD = new PDO($hote, $utilisateur, $motDePasse);
} catch (\Throwable $th) {
    die($th->getMessage());
}
```

Pour permettre à PHP de se connecter à une base données, il faut renseigner plusieurs éléments :

- Le dsn (ou Data Source Name) ici contenu dans notre variable $hote, c'est une chaîne de caractères qui regroupe plusieurs éléments indiquant comment et où se connecter :
  - mysql : représente le driver, pour faire simple on indique avec quel langage on veut communiquer avec notre BDD.
  - host : représente l'url/l'adresse de notre bdd. (peut également être une IP ex : 127.0.0.1 ou un nom de domaine)
  - dbname : le nom de la base de données ciblée. (car oui un serveur de BDD peut regrouper plusieurs base de données)
- l'utilisateur : nom/pseudo de l'utilisateur autorisé à se connecter à la BDD.

Note : Ici on découvre une nouvelle syntaxe nommée try catch, cette structure permet d'intercepter les erreurs à l'intérieur des accolades «try» et de faire un message à destination du développeur pour l'aider à corriger les erreurs. (explication simplifiée, pour plus d'informations faites une «recherches gestion des erreurs en PHP try catch»)

Désormais notre variable $connexionBDD contient notre connexion avec notre BDD et nous allons pouvoir l'utiliser pour faire des requêtes.

Requête SELECT

```php
<?php
$connexionBDD = null;
$hote = "mysql:host=learning-env-mariadb-dev-1;dbname=main-database";
$utilisateur = "root";
$motDePasse = "my-secret-pw";
try {
    $connexionBDD = new PDO($hote, $utilisateur, $motDePasse);
} catch (\Throwable $th) {
    die($th->getMessage());
}

$delcaration = $connexionBDD->query("SELECT * FROM users");
$users = $delcaration->fetchAll(PDO::FETCH_ASSOC);
echo "<pre>";
foreach($users as $cle => $valeur){
    print_r($valeur);
}
echo "</pre>";
```

```
Array
(
    [id] -> 1
    [username] => jdupont
    [email] => jean.dupont@example.com
    [password_hash] => $2y$10$N7B7eG5JvW1eLq1kQ9Zr0uJf7KQZ
    [created_at] => 2025-03-26 09:07:03
)
Array
(
    [id] => 2
    [username] => mleclerc
    [email] -> marie.leclerc@example.com
    [password_hash] => $2y$10$V/Ka5YJvW1eLq1kQ9Zr0uJf7KQZJ
    [created_at] => 2025-03-26 09:07:03
)
Array
(
    [id] -> 3
    [username] => admin
    [email] => admin@boutique.com
    [password_hash] => $2y$10$XcR7eG5JvW1eLq1kQ9Zr0uJf7KQZ
    [created_at] => 2025-03-26 09:07:03
)
```

1. **Contexte:**
   Toute interaction avec la BDD passe par l'objet $connexionBDD que vous avez créé via PDO
   PDO (PHP Data Objects) est une interface standardisée pour communiquer avec différents types de bases de données
2. **Phase de préparation :**
   query() envoie une requête SQL brute au serveur de base de données
   Le résultat n'est pas directement exploitable, c'est un «statement» (déclaration) que nous stockons dans $declaration
   Analogie : Comme donner une mission à un assistant - vous lui dictez la tâche (query()), mais il ne vous rapporte pas encore les résultats
3. **Récupération des données :**
   fetchAll() est l'instruction qui demande concrètement les données au serveur
   Le paramètre PDO::FETCH_ASSOC est un mode de récupération standardisé :
   Transforme chaque ligne en tableau associatif (clé = nom colonne)
   Alternative à éviter en début d'apprentissage : PDO::FETCH_NUM (indices numériques) ou PDO::FETCH_BOTH (les deux)
4. **Sécurité implicite :**
   Même avec query(), PDO offre une protection basique contre les injections SQL
   Mais attention : cette protection est incomplète sans requêtes préparées.

Requête SELECT avec des paramètres

Dans certains cas vous aurez besoin de faire des requêtes avec des contraintes where, comme par exemple where email = quelque chose. Et PDO vous permet d'ajouter des paramètres à vos requêtes de manière SECURISEE !
Pour un peu plus de contexte voici la table users sur laquelle l'exemple va se baser :

#	Nom	Type de données	Taille/Ensem...	Non si...	NULL a...	ZERO...	Par défaut	Commentaire	Collation	Expression	Virtualité	SRID	Invisi...
1	id	INT	11	☐	☐	☐	AUTO_INCREME...						☐
2	username	VARCHAR	50	☐	☐	☐	Pas de défaut		utf8mb4_unicode_ci				☐
3	email	VARCHAR	100	☐	☐	☐	Pas de défaut		utf8mb4_unicode_ci				☐
4	password_hash	VARCHAR	255	☐	☐	☐	Pas de défaut		utf8mb4_unicode_ci				☐
5	created_at	TIMESTAMP		☐	☑	☐	current_timestam...						☐

#	id	username	email	password_hash	created_at
1	1	jdupont	jean.dupont@example.com	$2y$10$N7B7eG5JvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6W...	2025-03-26 09:07:03
2	2	mleclerc	marie.leclerc@example.com	$2y$10$V/Ka5YJvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6WYc...	2025-03-26 09:07:03
3	3	admin	admin@boutique.com	$2y$10$XcR7eG5JvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6W...	2025-03-26 09:07:03

```php
$delcaration = $connexionBDD->prepare("SELECT * FROM users WHERE username = :nom_a_chercher;");
$delcaration->execute([
    'nom a chercher' => 'admin'
]);
$users = $delcaration->fetchAll(PDO::FETCH_ASSOC);
echo "<pre>";
foreach($users as $cle => $valeur){
    print_r($valeur);
}
echo "</pre>";
```

```
Array
(
    [id] => 3
    [username] => admin
    [email] => admin@boutique.com
    [password_hash] => $2y$10$XcR7eG5JvW1eLq1kQ9Zr0uJf7K(
    [created_at] => 2025-03-26 09:07:03
)
```

1. **Contexte global**

   Les requêtes préparées sont la méthode recommandée pour dialoguer avec une base de données en PHP.
   Elles offrent une protection intégrée contre les injections SQL, une attaque où un pirate pourrait détourner votre requête.
   Elles séparent clairement la structure SQL des données, ce qui rend le code plus lisible et plus sûr.

2. **Étapes détaillées**

   - **Préparation de la requête**

     On utilise la méthode prepare() sur l'objet de connexion ($connexionBDD).
     La requête SQL contient des paramètres nommés (comme :nom_a_chercher) qui servent de placeholders.
     À ce stade, la requête est vérifiée par le serveur SQL mais n'est pas encore exécutée.
   - **Exécution avec les valeurs**

     La méthode execute() remplace les paramètres nommés par les valeurs réelles (ici, 'admin').
     PDO s'occupe automatiquement de sécuriser les données (pas besoin de les échapper manuellement).
     C'est cette étape qui envoie la requête finale au serveur de base de données.
   - **Récupération des résultats**

     fetchAll() transforme les résultats en un tableau PHP structuré.
     L'option PDO::FETCH_ASSOC garantit que les données sont organisées sous forme de tableau associatif (noms de colonnes comme clés).
     Sans cette option, les résultats pourraient être moins lisibles (indices numériques).

3. **Pourquoi cette méthode est plus sûre ?**

   Sécurité : Les données sont automatiquement protégées contre les injections.
   Performance : La requête est précompilée côté serveur, ce qui est plus efficace si elle est réutilisée.
   Lisibilité : La séparation entre la requête et les données rend le code plus facile à comprendre.

4. **Bonnes pratiques à retenir**

   Toujours utiliser prepare() dès qu'une variable est intégrée dans une requête SQL.
   Vérifier si les résultats existent avant de les utiliser pour éviter des erreurs.

Requête d'ajout/modification (insert/update/delete)

Pour réaliser des requêtes insert/update/delete, nous utiliserons des requêtes préparées comme vu à l'exemple précédent, à défaut qu'ici nous n'utiliserons pas la fonction fetchAll car il n'y a rien à afficher dans ce genre de requête.

```php
try {
    $requete = "INSERT INTO users (username, email, password_hash) values (:pseudonyme, :mail, :mdp_hashe)";
    $delcaration = $connexionBDD->prepare($requete);
    $delcaration->execute([
        'pseudonyme' => 'alexl',
        'mail' => 'alexis.l.msf@gmail.com',
        'mdp_hashe' => '$2y$10$V/Ka5YJvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6WYc5RvJ1XrV2YhL6',
    ]);
} catch (\Throwable $th) {
    die($th->getMessage());
}
```

#	id	username	email	password_hash	created_at
1	1	jdupont	jean.dupont@example.com	$2y$10$N7B7eG5JvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6W...	2025-03-26 09:07:03
2	2	mleclerc	marie.leclerc@example.com	$2y$10$V/Ka5YJvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6WYc...	2025-03-26 09:07:03
3	3	admin	admin@boutique.com	$2y$10$XcR7eG5JvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6W...	2025-03-26 09:07:03
4	4	alexl	alexis.l.msg@gmail.com	$2y$10$V/Ka5YJvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6WYc...	2025-03-26 10:26:53

Exemple pour update

```php
try {
    $requete = "UPDATE users SET username = :username_value WHERE id = :id_value";
    $delcaration = $connexionBDD->prepare($requete);
    $delcaration->execute([
        'username_value' => 'alexisl-76000',
        'id_value' => 4
    ]);
} catch (\Throwable $th) {
    die($th->getMessage());
}
```

#	id	username	email	password_hash	created_at
1	1	jdupont	jean.dupont@example.com	$2y$10$N7B7eG5JvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6W...	2025-03-26 09:07:03
2	2	mleclerc	marie.leclerc@example.com	$2y$10$V/Ka5YJvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6WYc...	2025-03-26 09:07:03
3	3	admin	admin@boutique.com	$2y$10$XcR7eG5JvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6W...	2025-03-26 09:07:03
4	4	alexisl-76000	alexis.l.msg@gmail.com	$2y$10$V/Ka5YJvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6WYc...	2025-03-26 10:26:53

Exemple pour delete

```php
try {
    $requete = "DELETE FROM users WHERE id = :id_value";
    $delcaration = $connexionBDD->prepare($requete);
    $delcaration->execute([
        'id_value' => 4
    ]);
} catch (\Throwable $th) {
    die($th->getMessage());
}
```

#	id	username	email	password_hash	created_at
1	1	jdupont	jean.dupont@example.com	$2y$10$N7B7eG5JvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6W...	2025-03-26 09:07:03
2	2	mleclerc	marie.leclerc@example.com	$2y$10$V/Ka5YJvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6WYc...	2025-03-26 09:07:03
3	3	admin	admin@boutique.com	$2y$10$XcR7eG5JvW1eLq1kQ9Zr0uJf7KQZJ9Xl3dYb6W...	2025-03-26 09:07:03

Exercice pratique ! Aller voir le fichier ep_16.php