

Consolidation des compétences de programmation PHP

MODULE D12

- ❖ Consolidation des compétences de programmation PHP
- ❖ Mercredi 30 octobre 2019 & Jeudi 31 octobre 2019
- ❖ Répartition du cours
 - ❖ Théorie 15 -20 % du temps
 - ❖ Pratique : 75 à 80 % du temps
- ❖ Evaluation : QCM (Partie Lionel & Jihane)
 - ❖ Jeudi 31 octobre
 - ❖ 1h

Module D12

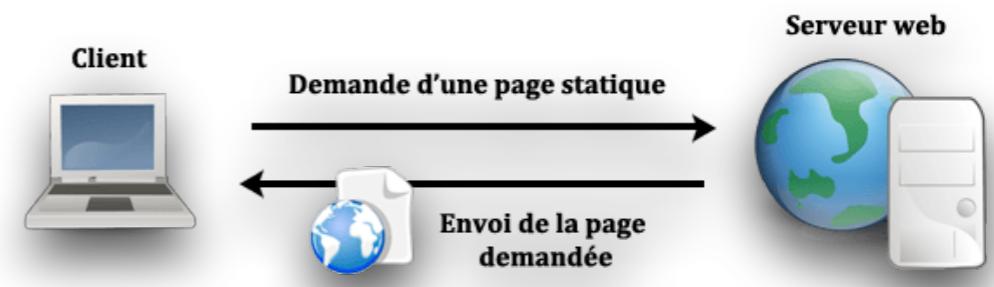
- ❖ Consolidation des compétences de programmation PHP
- ❖ Objectif : Avoir un socle commun dans le langage PHP

Introduction

Page statique / Page dynamique

❖ Page statique

- ❖ Le contenu ne varie pas en fonction des caractéristiques de la demande
- ❖ Tous les internautes qui demandent la page reçoivent le même contenu.
- ❖ HTML, CSS



❖ Page dynamique

- ❖ Son contenu varie en fonction des caractéristiques de la demande
- ❖ Générée à la demande
- ❖ Langage interprété par le serveur

❖ PHP

❖ Java

❖



Langage PHP

- ❖ Créé en 1994 par Rasmus Lerdorf
- ❖ A l'origine c'était une bibliothèque logicielle en C dont Rasmus se servait pour conserver une trace des visiteurs qui venaient consulter son CV
- ❖ Rasmus a transformé la bibliothèque en une implémentation capable de communiquer avec des bases de données et de créer des applications dynamiques et simples pour le Web.
- ❖ En 1995, Il décide de publier son code, pour que tout le monde puisse l'utiliser et en profiter
- ❖ PHP s'appelait alors PHP/FI
 - ❖ Personal Home Page Tools/Form Interpreter



Rasmus Lerdorf

Langage PHP

- ❖ En 1997 : Andi Gutmans et Zeev Suraski redéveloppent le cœur de PHP/FI.

- ❖ Version 3 de PHP

- ❖ PHP: Hypertext Preprocessor

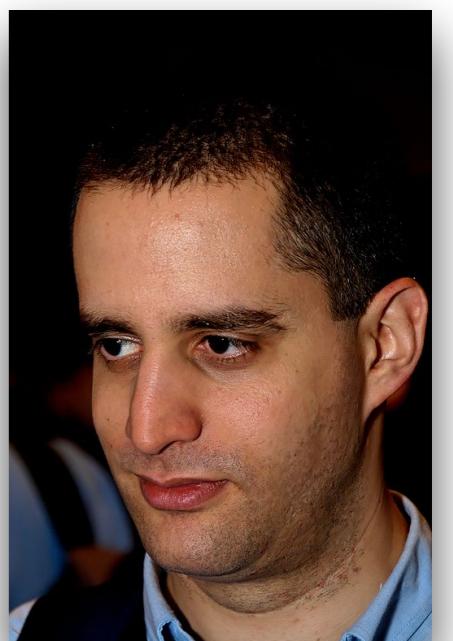
- ❖ Andi Gutmans et Zeev Suraski réécrivent le moteur interne de PHP



Andi Gutmans

- ❖ Nouveau moteur, appelé Zend Engine qui a servi de base à la version 4

- ❖ Zend est la contraction de **Zeev** et **Andi**



Zeev Suraski

Langage PHP

- ❖ PHP est un langage de script utilisé côté serveur
 - ❖ Interprétation du code PHP des pages web demandées
 - ❖ Génère du code (HTML)
- ❖ Conçu pour la création d'applications dynamiques
- ❖ Spécificités
 - ❖ multi-plateforme (Linux, Windows)
 - ❖ Libre
 - ❖ Gratuit
 - ❖ Simple d'utilisation et d'installation
- ❖ Php est écrit en C

Framework PHP

❖ En 2018, près de 80 % des sites web utilisent le langage PHP sous ses différentes versions.

❖ Exemples de framework

- ❖ Symfony

- ❖ Laravel

- ❖ CakePhp

❖ Exemples de CMS

- ❖ Wordpress

- ❖ Prestashop

❖ Exemples de Site web

- ❖ Facebook

- ❖ Blablacar

- ❖ Le Monde

- ❖ TF1

- ❖ ...

Version PHP

- ❖ PHP v1.0 : 8 juin 1995
- ❖ PHP v2.0 : 12 novembre 1997
- ❖ PHP v3.0 : 6 juin 1998
 - ❖ Passage d'une personne à une équipe de développeurs. Zeev Suraski et Andi Gutmans réécrivent la base de cette version20.
- ❖ PHP v4.0 : 22 mai 2000
- ❖ PHP v5.0 : 13 juillet 2004
 - ❖ Zend Engine II avec un nouveau modèle objet
- ❖ Php v7.0 : 3 décembre 2015
 - ❖ 7.2.16 : 7 mars 2019
 - ❖ 7.3.0 : Version actuelle : 29 novembre 2018

Structure d'un script php

- ❖ Extension .php
- ❖ On peut mettre du PHP dans le HTML
- ❖ Script PHP contient une suite d'instruction séparées par des points virgules

```
<?php  
    instruction1 ;  
    instruction2 ;  
    ...  
    instructionN  
?>
```

```
<?php  
    echo 'Hello World';  
    echo 'Comment allez-vous ?';  
    echo 'Il fait beau non ?'  
?>
```

```
index.php  
1 <!DOCTYPE html>  
2 <html lang="fr">  
3   <head>  
4     <meta charset="utf-8">  
5     <title></title>  
6   </head>  
7   <body>  
8     <h1><?php echo 'Salut à vous !' ?></h1>  
9   </body>  
10  </html>
```

Documentation

❖ Documentation officielle

❖ <https://www.php.net/manual/fr/langref.php>

The screenshot shows the official PHP documentation website. The top navigation bar includes links for 'Downloads', 'Documentation' (which is selected), 'Get Involved', and 'Help'. A search bar is also present. The main content area is titled 'Référence du langage' (Language Reference). It features a sidebar with language selection ('Change language: French'), edit and report a bug links, and a table of contents. The main content lists various language constructs like syntax, types, and control structures. On the right, a sidebar provides links to other parts of the manual, such as security, characteristics, and functions.

php Downloads Documentation Get Involved Help Search

Manuel PHP « Comment modifier la configuration La syntaxe de base »

Change language: French Edit Report a Bug

Référence du langage

- [La syntaxe de base](#)
 - [Balises PHP](#)
 - [Échappement depuis du HTML](#)
 - [Séparation des instructions](#)
 - [Commentaires](#)
- [Les types](#)
 - [Introduction](#)
 - [Booléen](#)
 - [Les entiers](#)
 - [Nombres à virgules flottantes](#)
 - [Les chaînes de caractères](#)
 - [Les tableaux](#)
 - [Itérables](#)
 - [Les objets](#)
 - [Les ressources](#)
 - [NULL](#)
 - [Fonctions de rappel / Types Callable](#)

Manuel PHP

Copyright

Manuel PHP

Au moment de commencer

Installation et configuration

» Référence du langage

Sécurité

Caractéristiques

Référence des fonctions

Dans le cœur de PHP : Guide du

Hacker

FAQ

Annexes

Php en ligne de commande

❖ Php peut être utilisé en ligne de commande

❖ Pas besoin de serveur web

❖ Utilisé pour l'exécution de scripts

```
13:51:48 $ php -v
PHP 7.3.8 (cli) (built: Aug 24 2019 19:38:14) ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.3.8, Copyright (c) 1998-2018 Zend Technologies
```

Commande pour connaître la version de php utilisée par le système

❖ Exécuter un script php

```
13:53:56 $ php test.php
Hello World !
```

Exécuter le fichier test.php contenant du code PHP

```
<?php
    echo "Hello World !";
?>
```

test.php

PHP à travers un serveur web

- ❖ Installer un environnement complet
 - ❖ Vagrant contenant une distribution de Linux
 - ❖ Apache (serveur web)
 - ❖ PHP
- ❖ Ligne de commande pour lancer un serveur de développement internet

```
14:06:10 $ php -S 127.0.0.1:8080
PHP 7.3.8 Development Server started at Tue Oct 29 14:06:32 2019
Listening on http://127.0.0.1:8080
Document root is /Users/jihane/Desktop/php
Press Ctrl-C to quit.
□
```



Appel du fichier test.php dans un navigateur
à travers un serveur web

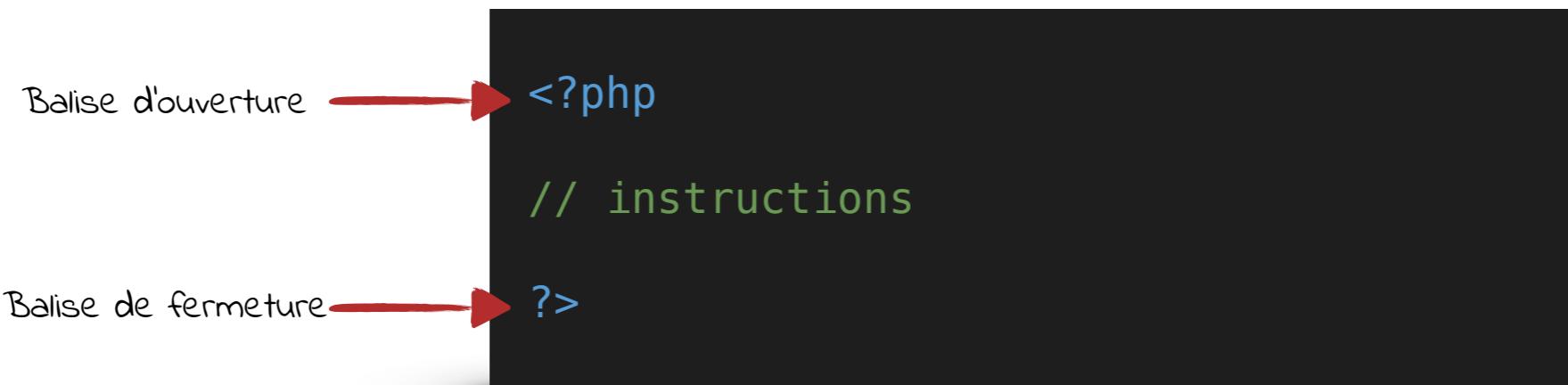
```
<?php
    echo "Hello World !";
?>
```

test.php

Structure de base

Balises PHP

- ❖ Elles délimitent le code
 - ❖ Code à l'intérieur des balises est interprété
 - ❖ Code à l'extérieur des balises est ignoré
- ➡ PHP peut être intégré dans toutes sortes de documents (HTML)



Balises PHP

※ Elles délimitent le code

※ Code à l'intérieur des balises est interprété

※ Code à l'extérieur des balises est ignoré

➡ PHP peut être intégré dans toutes sortes de documents (HTML)

The diagram illustrates the syntax of PHP code. It shows a dark rectangular area containing the following text:

```
<?php  
// instructions  
?>
```

A red arrow points from the text "Balise d'ouverture" to the opening tag "<?php". Another red arrow points from the text "Balise de fermeture" to the closing tag "?>".

Balises d'ouverture déconseillés si le site est déployé sur des serveurs que vous n'administrez pas, car ces balises peuvent ne pas être supportées sur le serveur cible.

The diagram shows a light orange rectangular area containing the following text:

```
<?  
// instructions  
?>
```

A red arrow points from the text above to this area, indicating it is an alternative form of PHP code.

Balises PHP

- ⌘ Si un fichier contient seulement du code PHP
- ⌘ Ne pas placer la balise de fermeture à la fin du fichier pour éviter un espace ou une nouvelle ligne après la balise de fermeture de PHP

Fichier contient que du code PHP

```
<?php  
// instructions
```

Fichier contient du code PHP et d'autre langage

```
<h1> Excepteur sint obcaecat cupiditat non proident culpa.</h1>  
<?php  
// instructions  
?>  
<p>Integer legentibus erat a ante historiarum dapibus.</p>
```

Balises PHP

⌘ Balise raccourci

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



```
<?= "Bonjour le monde !";?>
```

Instruction

- ❖ Une instruction informatique désigne une étape dans un programme informatique
- ❖ Une instruction dicte à l'ordinateur l'action nécessaire qu'il doit effectuer avant de passer à l'instruction suivante.
- ❖ Un programme informatique est constitué d'une suite d'instructions.

```
<?php  
  
instruction_1 ;  
instruction_2 ;  
  
// ...  
  
instruction_n ;  
  
?>
```

Séparateur d'instructions

- ✿ Instructions doivent être terminées par un point-virgule

```
<?php  
  
instruction_1 ;  
instruction_2 ;  
  
// ...  
  
instruction_n ;  
  
?>
```

```
<?php  
  
echo "Bonjour le monde !";  
echo 'Ceci est un test';  
  
// ... encore du code  
  
echo "Dernière instruction";  
  
?>
```

```
<?php  
  
echo "Bonjour le monde !";echo 'Ceci est un test';  
  
// ... encore du code  
  
echo "Dernière instruction";  
  
?>
```

Plusieurs instructions peuvent être écrites sur la même ligne,
mais elles doivent être séparées par un point virgule.
Attention à la lisibilité du code !

Séparateur d'instructions

- ❖ La balise fermante d'un bloc de code PHP implique automatiquement un point-virgule
- ❖ pas besoin d'utiliser un point-virgule pour terminer la dernière ligne d'un bloc PHP

```
<?php  
  
echo "Bonjour le monde !";  
echo 'Ceci est un test';  
  
// ... encore du code  
  
echo "Dernière instruction"  
  
?>
```

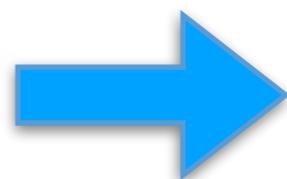
Fonction écho

- ❖ Elle permet d'afficher une ou plusieurs chaîne
- ❖ Inclure du texte dans la page HTML

```
<?php  
  
echo (chaine texte) ;  
echo chaine texte [, ...];  
  
?>
```

```
<?php  
  
echo "Bonjour le monde !";  
echo 'ceci ', "est un test";  
  
?>
```

```
<?= "Bonjour le monde !" , 'ceci ', "est un test"?>
```



```
jihane ☺ ~/Desktop/js  
14:27:25 $ php test.php  
Bonjour le monde !ceci est un test
```

Commentaires

- ❖ Les commentaires n'ont aucune influence sur l'exécution du code et est ignoré
- ❖ Les commentaires sur une seule ligne
 - ❖ //
 - ❖ #
- ❖ Les commentaires sur plusieurs lignes
 - ❖ /* ... */

```
<?php
# ceci est un commentaire
echo "Bonjour le monde ! »;

// ceci est un commentaire
echo 'Ceci est un test'; // ceci est un commentaire

/*
    ceci est
    un commentaire
    sur plusieurs lignes
*/
echo "Dernière instruction"

?>
```

PHP en ligne de commande

❖ Connaitre la version de php

```
14:32:04 $ php -v
PHP 7.1.23 (cli) (built: Feb 22 2019 22:19:32) ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.1.0, Copyright (c) 1998-2018 Zend Technologies
```

❖ Exécuter un script

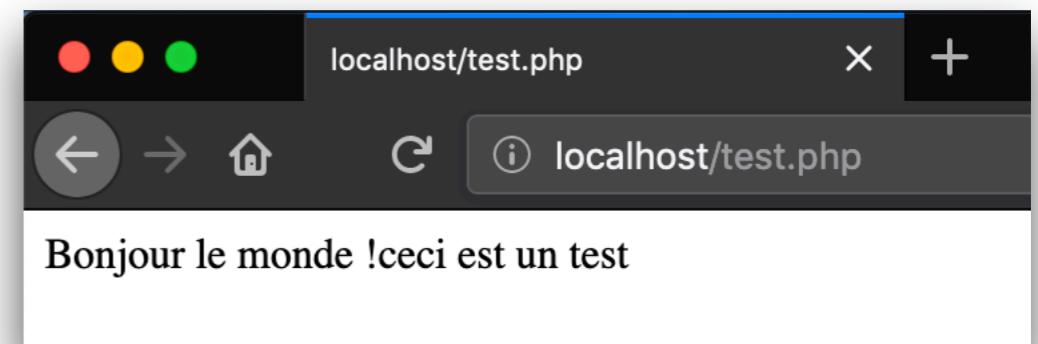
```
14:37:44 $ php test.php
Bonjour le monde !ceci est un test
Bonjour le monde !ceci est un test
```

```
14:40:38 $ php test.php Hello
Hello le monde !ceci est un test
```

Passer des paramètres au script

❖ Lancer un serveur web

```
jihane ☺ ~/Desktop/php
14:44:14 $ sudo php -S localhost:80
PHP 7.1.23 Development Server started at Thu Oct 24 14:44:25 2019
Listening on http://localhost:80
Document root is /Users/jihane/Desktop/php
Press Ctrl-C to quit.
```



Bases du langage

Variables

Variable

- ❖ C'est une zone mémoire identifiée par un nom qui contient une valeur lisible ou modifiable
- ❖ Elles sont représentées par un signe dollar "\$" suivi du nom de la variable
- ❖ Le nom est sensible à la casse (`$nom` ≠ `$NOM`)
- ❖ Règles de nommage
 - ❖ Commencer par une lettre ou un souligné (`_`)
 - ❖ Suivi de lettres, chiffres ou soulignés
- ❖ Les variables sont typées automatiquement lors de chaque affectation

Variable - Initialisation

- ❖ Pas obligatoire d'initialiser les variables en PHP (mais recommandé)
- ❖ Les variables non initialisées ont une valeur par défaut selon leur type
 - ❖ FALSE pour les booléens
 - ❖ zéro pour les entiers et les réels
 - ❖ chaîne vide pour les chaînes de caractères
 - ❖ un tableau vide pour les tableaux

Variable

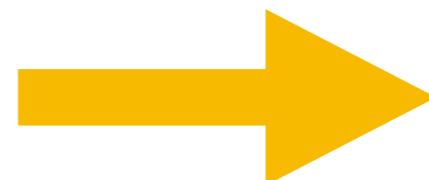
- ❖ Les variables sont toujours assignées par valeur
 - ❖ la valeur de l'expression est recopiée dans la variable
- ❖ PHP permet aussi d'assigner les valeurs aux variables par référence
 - ❖ la nouvelle variable ne fait que référencer la variable originale
 - ❖ Les modifications de la nouvelle variable affecteront l'ancienne et vice versa.
 - ❖ Pour assigner par référence, ajoutez simplement un & (ET commercial) au début de la variable qui est assignée (la variable source)
 - ❖ \$a = 1;
 - ❖ \$b = &\$a;
 - ❖ Echo \$b;
 - ❖ \$a = 2;
 - ❖ Echo \$b;
 - ❖

Portée d'une variable

✿ Variables locales

```
$a = "hello";
function coucou(){
    echo $a;
}

coucou();
```



```
15:09:46 $ php test.php
```

\$a n'est pas connue car n'appartient
pas au contexte de la fonction

```
function coucou(){
    $a = "hello";
}

coucou();
echo $a;
```



```
15:09:46 $ php test.php
```

\$a n'est pas connue car n'appartient
pas au contexte du script

```
$a = $a+1;
echo $a;
```



```
15:59:29 $ php test.php
1
jihane 😊 ~/Desktop/php
15:59:59 $ php test.php
1
```

Les variables sont temporaires : une fois que le script est terminé elles cessent d'exister

Variable globale

❖ Variable globale

- ❖ Portée : dans le script et dans les fonctions définies dans le script

❖ 1ère méthode

- ❖ Ajouter le mot clé global + nom de la variable

❖ 2ème méthode

- ❖ Utiliser le tableau associatif \$GLOBALS

- ❖ Clé = nom des variables globales

- ❖ Valeur = valeur des variables

```
$a = 1; /* portée globale */
function test()
{
    global $a;
    echo $a;
}
test();
```

```
$a = 1; /* portée globale */
function test()
{
    echo $GLOBALS['a'];
}
test();
```

Variable dynamique

- ❖ Une variable qui stocke le nom d'une autre variable et y faire référence dans le script
- ❖ Le nom d'une variable peut être lui même une variable

```
<?php

$a = 'bonjour';
$$a = 'monde'; // $bonjour = 'monde'

echo $bonjour; // retourne monde

echo "$a ${$a}"; // retourne bonjour monde
?>
```

```
<?php

$cd = '15 €';
$dvd = '30 €';

$produit = 'dvd';

echo "le prix du produit : ${$produit}";

?>
```

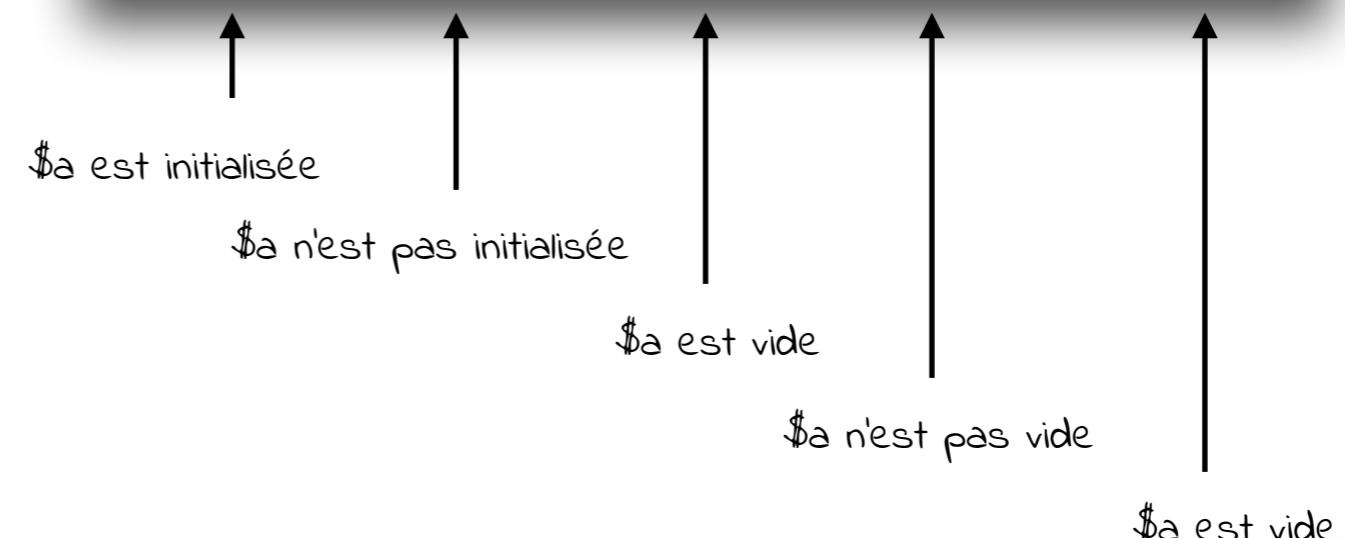
```
16:22:36 $ php test.php
le prix du produit : 30 €
```

Fonctions sur les variables

- ❖ `empty($var)` : indique si une variable est vide ou non
- ❖ `isset($var)` : renvoie TRUE si une variable a été initialisée et FALSE sinon
- ❖ `unset($var)` : supprime une variable
- ❖ `var_dump()` : affiche des informations d'une variable (type et valeur)

```
$a = 1;  
var_dump($a);  
  
echo " [ a : ", isset($a), ' ] ";  
unset($a);  
echo " [ a : ", isset($a), ' ] ";  
  
echo " [ a : ", empty($a), ' ] ";  
  
$a = 1;  
echo " [ a : ", empty($a), ' ] ";  
  
$a = "";  
echo " [ a : ", empty($a), ' ] ";
```

```
16:14:17 $ php test.php  
int(1)  
[ a : 1 ] [ a : ] [ a : 1 ] [ a : ] [ a : 1 ]
```



Constante

Constante

- ❖ Une constante est une zone mémoire identifiée par un nom qui contient une valeur lisible mais non modifiable dans le programme (ni détruite)
- ❖ Par défaut, le nom d'une constante est sensible à la casse
- ❖ Par convention, les constantes sont toujours en majuscules
- ❖ Les noms de constantes
 - ❖ commence par une lettre ou un souligné (_)
 - ❖ suivi d'un nombre quelconque de lettres, chiffres ou soulignés
- ❖ Les constantes sont accessibles de manière globale sans qu'elles soient déclarées globale
- ❖ Elles ne peuvent pas contenir d'objets ou de variables

Constante

- ❖ Définir une constante
 - ❖ En utilisant la fonction **define()**
 - ❖ Possibilité de définir une constante avec un nom réservé ou même invalide, ces valeurs peuvent être récupérée avec la fonction **constant()** (**Attention ce n'est pas recommandé**)
 - ❖ En utilisant le mot-clé **const**
 - ❖ Toujours sensible à la casse

Constante

❖ Définir une constante

❖ En utilisant la fonction **define()**

- ❖ Possibilité de définir une constante avec un nom réservé ou même invalide, ces valeurs peuvent être récupérée avec la fonction **constant()** (**Attention ce n'est pas recommandé**)

```
boolean define(chaine nom, mixte valeur [, boolean sensible_casse]
```

- nom = Nom de la constante
- valeur = Valeur de la constante
 - => type de données : scalaire ou tableau
 - => valeur peut être définie à l'aide d'une expression qui utilise des valeurs littérales, constante, variables, opérateurs, ou des appels de fonctions
- sensible_casse = indique si le nom de la constante est sensible à la casse (TRUE par défaut)
- La fonction define retourne TRUE en cas de succès et FALSE en cas d'erreur

```
<?php
define("FOO",      "something");
define("F002",     "something else");
define("FOO_BAR",  "something more");

// Noms invalides
define("2FOO",     "something");

?>
```

Constante

❖ Définir une constante

❖ En utilisant le mot-clé **const**

❖ Toujours sensible à la casse

```
const nom = valeur
```

- nom = Nom de la constante
- valeur = valeur de la constante
 - => type de données : scalaire ou tableau
 - => valeur peut être définie à l'aide d'une expression qui utilise des valeurs littérales, constantes, opérateurs

```
<?php  
const CONSTANTE = 'Bonjour le monde !';  
?>
```

Constante

- ❖ Accéder à la valeur d'une constante
 - ❖ En spécifiant simplement son nom
 - ❖ En utilisant la fonction `constant($valeur)`
- ❖ La fonction `get_defined_constants()` permet de connaître la liste de toutes les constantes définies.

```
<?php  
  
const CONSTANTE = 'Bonjour le monde !';  
  
echo CONSTANTE;  
echo constant("CONSTANTE");  
  
?>
```

Différences entre les constantes et les variables

- ❖ Les constantes ne commencent pas par le signe (\$)
- ❖ Les constantes sont définies et accessibles à tout endroit du code
- ❖ Les constantes ne peuvent pas être redéfinies ou indéfinies une fois qu'elles ont été définies.
- ❖ Les constantes ne peuvent contenir que des scalaires et des tableaux

Types de données

Types de données

❖ PHP supporte 10 types de données

❖ 4 types scalaires :

- ❖ boolean
- ❖ Entier (integer)
- ❖ float (nombre à virgule flottante, double)
- ❖ string

❖ 4 types composés :

- ❖ Tableau (array)
- ❖ Object
- ❖ Fonctions de rappel (Callable)
- ❖ Iterable

❖ 2 types spéciaux :

- ❖ resource
- ❖ NULL

En gris : type non étudié dans le module d8

Types scalaires

Booléen

- ❖ Un booléen représente une valeur de vérité.
- ❖ Il peut valoir TRUE ou FALSE (insensibles à la casse)

```
<?php  
$ma_variable = TRUE ;  
$ma_variable_2 = FALSE ;  
?>
```

Booléen

- ❖ Lors d'une conversion en booléen, les valeurs suivantes sont considérées comme FALSE :
 - ❖ le booléen FALSE, lui-même
 - ❖ les entiers 0 et -0 (zéro)
 - ❖ les nombres à virgule flottante 0.0 et -0.0 (zéro)
 - ❖ la chaîne vide, et la chaîne "0"
 - ❖ un tableau avec aucun élément
 - ❖ le type spécial NULL (incluant les variables non définies)
- ❖ Toutes les autres valeurs sont considérées comme TRUE

Les entiers

- ❖ Un entier est un nombre appartenant à la classe $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$
- ❖ Les entiers peuvent être spécifiés en notation
 - ❖ décimale (base 10)
 - ❖ hexadécimale (base 16)
 - ❖ Précéder le nombre par 0x
 - ❖ octale (base 8)
 - ❖ Précédez le nombre par 0 (zéro)
 - ❖ binaire (base 2)
 - ❖ Précéder le nombre par 0b
- ❖ L'opérateur de négation peut être utilisé pour désigner un entier négatif

Les entiers

```
<?php
$mon_entier_1 = 10      ;
$mon_entier_2 = 0xa     ;
$mon_entier_3 = 022     ;
$mon_entier_4 = 0b1000 ;
?>
```

Nombres à virgule flottant

- ❖ floats, doubles, real numbers
- ❖ Nombre qui contient une partie décimale
- ❖ Pour écrire un nombre à virgule
 - ❖ Utiliser le « . » pour séparer la partie entière de la partie décimale
 - ❖ Utiliser le « e » pour indiquer les puissance de 10

```
<?php  
$mon_nombre = 100.0;  
$nom_nombre_2 = 1e2;  
?>
```

Fonctions sur les nombres

❖ <https://www.php.net/manual/fr/ref.math.php>

- [abs](#) – Valeur absolue
- [acos](#) – Arc cosinus
- [acosh](#) – Arc cosinus hyperbolique
- [asin](#) – Arc sinus
- [asinh](#) – Arc sinus hyperbolique
- [atan2](#) – Arc tangent de deux variables
- [atan](#) – Arc tangente
- [atanh](#) – Arc tangente hyperbolique
- [base_convert](#) – Convertit un nombre entre des bases arbitraires
- [bindec](#) – Convertit de binaire en décimal
- [ceil](#) – Arrondit au nombre supérieur
- [cos](#) – Cosinus
- [cosh](#) – Cosinus hyperbolique
- [decbin](#) – Convertit de décimal en binaire
- [dechex](#) – Convertit de décimal en hexadécimal
- [decoct](#) – Convertit de décimal en octal
- [deg2rad](#) – Convertit un nombre de degrés en radians
- [exp](#) – Calcul l'exponentielle de e
- [expm1](#) – Calcule précisément exponentiel moins un

Chaine de caractères

- ❖ Une chaîne de caractères est une suite de caractères
- ❖ Une chaîne de caractères littérale peut être spécifiée de 4 façons différentes :
 - ❖ Guillemets simples
 - ❖ Guillemets doubles
 - ❖ Syntaxe Heredoc
 - ❖ Syntaxe Nowdoc (depuis PHP 5.3.0)

Chaine de caractères

❖ Guillemets simples

```
<?php
$ma_chaine_1 = 'ceci est une chaine';
$ma_chaine_2 = 'c\'est une chaine également';
$ma_chaine_3 = 'pour mettre des guillemets dans ma chaine j\'utilise le " ';

$var = 'hello';
$ma_chaine_4 = 'la variable $var vaut '.$var.' ';
?>
```

Les variables ne sont automatiquement remplacées par leur valeur, il faut concatener la chaîne et la variable
Pour échapper un caractère spécial : \ (barre oblique inverse)

❖ Guillemets doubles

```
<?php
$ma_chaine_1 = "ceci est une chaine";
$ma_chaine_2 = "c'est une chaine également";
$ma_chaine_3 = "pour mettre des guillemets dans ma chaine j'utilise le \" ";

$var = "hello";
$ma_chaine_4 = "la variable \$var vaut $var ";
?>
```

Les variables sont automatiquement remplacées par leur valeur
Pour échapper un caractère spécial : \ (barre oblique inverse)

Chaine de caractères

❖ Syntaxe Heredoc

- ❖ variables sont interprétées
- ❖ Texte délimité à l'ouverture par trois symboles

❖ <<<

❖ Identifiant (mêmes règles de nommage que les variables)

❖ Saut de ligne

- ❖ Texte est fermé par une ligne contenant l'identifiant et un point virgule

```
<?php
$var = "hello";

$ma_chaine_1 = <<<ceciestunidentifiant
Contra legem facit qui id facit quod lex prohibet.

'lmdlmds'
\"sdklasd"

$var

Ullamco laboris nisi ut aliquid ex ea commodi consequat.

ceciestunidentifiant;

?>
```

Chaine de caractères

❖ Syntaxe Nowdoc :

- ❖ variables ne sont pas interprétées
- ❖ Texte délimité à l'ouverture par trois symboles
 - ❖ <<<
- ❖ Identifiant (mêmes règles de nommage que les variables) entourer de guillemets simple
- ❖ Saut de ligne
- ❖ Texte est fermé par une ligne contenant l'identifiant et un point virgule

```
<?php  
  
$var = "hello";  
  
$ma_chaine_1 = <<<'delimiteur'  
Contra legem facit qui id facit quod lex prohibet.  
Quisque placerat facilisis egestas cillum dolore.  
  
'lmdlmds'  
  
\"sdklsd"  
  
$var  
  
Ullamco laboris nisi ut aliquid ex ea commodi consequat.  
  
delimiteur;  
  
?>
```

Accéder à un caractère d'une chaîne

❖ Méthode 1

- ❖ En précisant entre les accolades la position du caractère recherché

❖ Méthode 2

- ❖ En précisant entre les crochets la position du caractère recherché

❖ La position commence à 0

```
<?php  
  
$var = "hello";  
  
echo $var{1};  
  
$var = "hello";  
  
echo $var[1];  
  
?>
```

Fonction sur les chaînes

❖ <https://www.php.net/manual/fr/book.strings.php>

- [str_split](#) – Convertit une chaîne de caractères en tableau
- [str_word_count](#) – Compte le nombre de mots utilisés dans une chaîne
- [strcasecmp](#) – Comparaison insensible à la casse de chaînes binaires
- [strchr](#) – Alias de strstr
- [strcmp](#) – Comparaison binaire de chaînes
- [strcoll](#) – Comparaison de chaînes localisées
- [strcspn](#) – Trouve un segment de chaîne ne contenant pas certains caractères
- [strip_tags](#) – Supprime les balises HTML et PHP d'une chaîne
- [stripcslashes](#) – Décode une chaîne encodée avec addcslashes
- [stripos](#) – Recherche la position de la première occurrence dans une chaîne, sans tenir compte de la casse
- [stripslashes](#) – Supprime les antislashs d'une chaîne
- [stristr](#) – Version insensible à la casse de strstr
- [strlen](#) – Calcule la taille d'une chaîne
- [strnatcasecmp](#) – Comparaison de chaînes avec l'algorithme d'"ordre naturel" (insensible à la casse)
- [strnatcmp](#) – Comparaison de chaînes avec l'algorithme d'"ordre naturel"
- [strncasecmp](#) – Compare en binaire des chaînes de caractères
- [strncmp](#) – Comparaison binaire des n premiers caractères
- [strpbrk](#) – Recherche un ensemble de caractères dans une chaîne de caractères
- [strpos](#) – Cherche la position de la première occurrence dans une chaîne

Types composés

Tableau

- ❖ Un tableau est une collection ordonnée de couple clé/valeur
- ❖ Quand la clé est de type entier
 - ❖ le tableau est dit numérique
 - ❖ Clé est appelée indice
- ❖ Quand la clé est de type chaîne
 - ❖ Le tableau est dit associatif
 - ❖ Les clés ne sont pas forcément ordonnées, ni consécutives
- ❖ La clé ne peut être qu'un entier ou une chaîne de caractère,
- ❖ La valeur peut être de n'importe quel type
- ❖ Il n'est pas nécessaire de déclarer la taille du tableau

Création d'un tableau

- ❖ Un tableau peut être créé
 - ❖ en utilisant la structure de langage array()
 - ❖ Il prend un nombre illimité de paramètres, chacun séparé par une virgule

```
<?php  
$array = array(  
    "foo" => "bar",  
    "bar" => "foo",  
);  
?>
```

```
<?php  
$tab = array(10, "20", 30 , '40');  
?>
```

Sous la forme : clé => valeur

- ❖ utiliser la syntaxe courte []

```
<?php  
$array = [  
    "foo" => "bar",  
    "bar" => "foo",  
    "zoo", « zae"  
];  
?>
```

```
<?php  
$tab = [10, "20", 30 , '40'];  
?>
```

Sous la forme : clé => valeur

Tableau

❖ Cas particulier de la clé :

- ❖ Si elle est de type chaînes de caractères et contient un entier valide, sauf si le nombre est précédé d'un signe + , alors elle sera modifiée en un type entier.
- ❖ Si elle est de type nombres à virgule flottante alors elle sera modifiée en entier
- ❖ Si elle est de type booléens alors elle sera modifiée en entier également
 - ❖ True stocké sous l'entier 1 et false sous l'entier 0
- ❖ Si elle a pour valeur Null elle sera modifiée en une chaîne vide
- ❖ Les tableaux et les objets ne peuvent pas être utilisés comme clé
- ❖ Si plusieurs éléments dans la déclaration d'un tableau utilisent la même clé, seule la dernière sera utilisée, écrasant ainsi toutes les précédentes.

```
$array = array(  
    1      => "a",  
    "1"    => "b",  
    1.5   => "c",  
    true  => "d",  
);  
var_dump($array);
```

```
21:39:12 $ php test.php  
array(1) {  
    [1]=>  
        string(1) "d"  
}
```

Tableau

- ❖ Les tableaux peuvent contenir des clés de type integer et string en même temps

```
$array = array(  
    "foo" => "bar",  
    "bar" => "foo",  
    100   => -100,  
    -100  => 100,  
);  
var_dump($array);
```

```
21:39:14 $ php test.php  
array(4) {  
    ["foo"]=>  
    string(3) "bar"  
    ["bar"]=>  
    string(3) "foo"  
    [100]=>  
    int(-100)  
    [-100]=>  
    int(100)  
}
```

Tableau

- ❖ La clé key est optionnelle. Si elle n'est pas spécifiée, PHP utilisera un incrément de la dernière clé entière utilisée.

```
$array = array("foo", "bar", "hello", "world");
var_dump($array);
```

```
21:41:20 $ php test.php
array(4) {
    [0]=>
    string(3) "foo"
    [1]=>
    string(3) "bar"
    [2]=>
    string(5) "hello"
    [3]=>
    string(5) "world"
}
```

```
$array[] = "Bonjour";
var_dump($array);
```

```
17:25:20 $ php test.php
array(1) {
    [0]=>
    string(7) "Bonjour"
}
```

Accès aux éléments d'un tableau

❖ Pour accéder à un élément d'un tableau

❖ array[key].

```
$tab = [1,3,5];
echo $tab[0]; // 1
echo $tab[2]; // 5
```

Parcourir un tableau

❖ Plusieurs façons de faire

❖ En utilisant for

❖ En utilisant while

❖ En utilisant foreach

```
for($i=0;$i<count($tab);$i++){
    echo $tab[$i],"\n";
}
```

```
$i=0;
while($i<count($tab) ){
    echo $tab[$i],"\n";
    $i++;
}
```

```
foreach ($tab as $value) {
    echo $value,"\n";
}
```

Modification d'un élément d'un tableau

- ❖ La modification d'une valeur dans un tableau est effectué en spécifiant la clé, entre crochets.
- ❖ Si la clé n'existe pas lors de l'assignation, il sera créé :

```
$tab = [1,3,5];
$tab[0] = 6;
$tab[9] = 9;

foreach ($tab as $value) {
    echo $value, "\n";
}
```

Tableau multidimensionnel

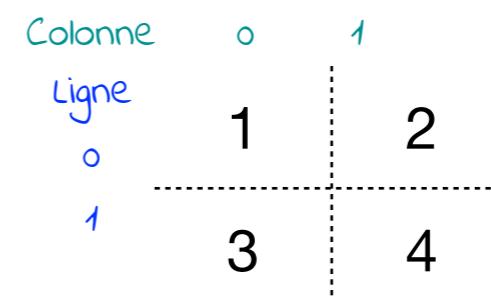
- ❖ Possibilité de créer un tableau à plusieurs dimension

- ❖ Tableau de tableau

```
$array[0][0] = 1;  
$array[0][1] = 2;  
$array[1][0] = 3;  
$array[1][1] = 4;  
  
var_dump($array);
```

```
17:29:39 $ php test.php  
array(2) {  
    [0]=>  
        array(2) {  
            [0]=>  
                int(1)  
            [1]=>  
                int(2)  
        }  
    [1]=>  
        array(2) {  
            [0]=>  
                int(3)  
            [1]=>  
                int(4)  
        }  
}
```

```
$array = array(  
    array(1,2),  
    array(3,4),  
);  
  
var_dump($array);
```



Fonctions sur les Tableaux

- ❖ <https://www.php.net/manual/fr/ref.array.php>

Types spéciaux

NULL

- ❖ La valeur spéciale NULL représente une variable sans valeur. NULL est la seule valeur possible du type null.
- ❖ Une variable est considérée comme null si :
 - ❖ elle s'est vue assigner la constante NULL.
 - ❖ elle n'a pas encore reçu de valeur.
 - ❖ elle a été effacée avec la fonction unset().
 - ❖ Il y a une seule valeur de type null, et c'est la constante insensible à la casse NULL

Opérateurs

Opérateur

- ❖ Un opérateur est un symbole qui sert à manipuler des expressions

Les opérateurs arithmétiques

Opérations élémentaires

Exemple	Nom	Résultat
<code>-\$a</code>	Négation	Opposé de <code>\$a</code> .
<code>\$a + \$b</code>	Addition	Somme de <code>\$a</code> et <code>\$b</code> .
<code>\$a - \$b</code>	Soustraction	Différence de <code>\$a</code> et <code>\$b</code> .
<code>\$a * \$b</code>	Multiplication	Produit de <code>\$a</code> et <code>\$b</code> .
<code>\$a / \$b</code>	Division	Quotient de <code>\$a</code> et <code>\$b</code> .
<code>\$a % \$b</code>	Modulus	Reste de <code>\$a</code> divisé par <code>\$b</code> .
<code>\$a ** \$b</code>	Exponentielle	Résultat de l'élévation de <code>\$a</code> à la puissance <code>\$b</code> . Introduit en PHP 5.6.

Les opérateurs d'affectation

`$a = $b` La valeur située à gauche prend la valeur située à droite

`$a = & $b` La valeur située à gauche prend la référence de la variable située à droite

`+=` Ajoute la valeur de droite à celle de gauche. Et le résultat dans la variable de gauche

`-=` Soustrait la valeur de droite à celle de gauche. Et le résultat dans la variable de gauche

`*=` Multiplie la valeur de droite à celle de gauche. Et le résultat dans la variable de gauche

`.=` Concatène la valeur de droite à celle de gauche. Et le résultat dans la variable de gauche
Concatener =

$$\$a \textcolor{red}{x= \$b} \Leftrightarrow \$a = \$a \textcolor{red}{x} \$b$$

$$\textcolor{red}{x} = \{ +, -, *, /, \% \}$$

Les opérateurs de comparaison

Opérateurs de comparaison

Exemple	Nom	Résultat
<code>\$a == \$b</code>	Egal	TRUE si <code>\$a</code> est égal à <code>\$b</code> après le transtypage.
<code>\$a === \$b</code>	Identique	TRUE si <code>\$a</code> est égal à <code>\$b</code> et qu'ils sont de même type.
<code>\$a != \$b</code>	Différent	TRUE si <code>\$a</code> est différent de <code>\$b</code> après le transtypage.
<code>\$a <> \$b</code>	Différent	TRUE si <code>\$a</code> est différent de <code>\$b</code> après le transtypage.
<code>\$a !== \$b</code>	Différent	TRUE si <code>\$a</code> est différent de <code>\$b</code> ou bien s'ils ne sont pas du même type.
<code>\$a < \$b</code>	Plus petit que	TRUE si <code>\$a</code> est strictement plus petit que <code>\$b</code> .
<code>\$a > \$b</code>	Plus grand	TRUE si <code>\$a</code> est strictement plus grand que <code>\$b</code> .
<code>\$a <= \$b</code>	Inférieur ou égal	TRUE si <code>\$a</code> est plus petit ou égal à <code>\$b</code> .
<code>\$a >= \$b</code>	Supérieur ou égal	TRUE si <code>\$a</code> est plus grand ou égal à <code>\$b</code> .
<code>\$a <=> \$b</code>	Combiné	Un entier inférieur, égal ou supérieur à zéro lorsque <code>\$a</code> est respectivement inférieur, égal, ou supérieur à <code>\$b</code> . Disponible depuis PHP 7.

Les opérateurs de comparaison

- ❖ L'opérateur ===
 - ❖ PHP considère que '2' == 2 sont égaux (renvoie TRUE)
 - ❖ Valeur égale
 - ❖ Type différent
- ❖ Pour comparer également les types on utilise ===
 - ❖ Compare les valeurs et les types
 - ❖ '2' === 2 renvoie FALSE
- ❖ L'opérateur !=
 - ❖ Compare les valeurs et les types

Les opérateurs logiques

Les opérateurs logiques

Exemple	Nom	Résultat
<code>\$a and \$b</code>	And (Et)	TRUE si <code>\$a</code> ET <code>\$b</code> valent TRUE .
<code>\$a or \$b</code>	Or (Ou)	TRUE si <code>\$a</code> OU <code>\$b</code> valent TRUE .
<code>\$a xor \$b</code>	XOR	TRUE si <code>\$a</code> OU <code>\$b</code> est TRUE , mais pas les deux en même temps.
<code>! \$a</code>	Not (Non)	TRUE si <code>\$a</code> n'est pas TRUE .
<code>\$a && \$b</code>	And (Et)	TRUE si <code>\$a</code> ET <code>\$b</code> sont TRUE .
<code>\$a \$b</code>	Or (Ou)	TRUE si <code>\$a</code> OU <code>\$b</code> est TRUE .

La différence entre and et `&&` et or et `||` réside dans les priorités d'application

OR, AND : priorité faible

`&&` , `||` : priorité forte

Les opérateurs d'incrémentation et de décrémentation

Opérateurs d'incrémentation et décrémentation

Exemple	Nom	Résultat
<code>++\$a</code>	Pre-incrémentation	Incrémente \$a de 1, puis retourne \$a.
<code>\$a++</code>	Post-incrémentation	Retourne \$a, puis incrémente \$a de 1.
<code>--\$a</code>	Pré-décrémentation	Décrémente \$a de 1, puis retourne \$a.
<code>\$a--</code>	Post-décrémentation	Retourne \$a, puis décrémente \$a de 1.

```
$a = 10;
echo $a++.#"$.a; //10#11
echo ++$a.#"$.a; //12#12
echo $a--.#"$.a; //12#11
echo --$a.#"$.a; //10#10
```

Opérateur ternaire

- ❖ Opérateur qui accepte 3 opérandes
- ❖ Si la première opérande vaut TRUE il retourne la valeur de la 2ème opérande
- ❖ Sinon il retourne la valeur de la 3ème opérande

```
<?php  
operande1 ? operande2 : operande3 ;  
?>
```

```
echo ( 1 > 2 ) ? "inférieur" : 12;
```

- ❖ Opérateur Elvis : on ne met pas le second opérande
- ❖ Si la première opérande vaut TRUE il retourne la valeur du 1er argument
- ❖ Sinon il retourne la valeur de la 3ème opérande

```
<?php  
operande1 ?: operande3 ;  
?>
```

```
echo 1 < 2 ?: 12;
```

Opérateur de nullité

- ❖ Affecter une valeur en testant si des variables existent et ne valent pas null

```
<?php  
  
$var = $var1 ?? $var2 ;  
  
?>
```

```
$var2= 2;  
$var = $var1 ?? $var2 ?? $var3;  
echo $var; // 2
```

Priorité entre opérateurs

1	<code>() []</code>
2	<code>** -- ++ !</code>
3	<code>* / %</code>
4	<code>+ -</code>
5	<code>< <= >= ></code>
6	<code><< == != === <=></code>
7	<code>&</code>
8	<code> </code>
9	<code>&&</code>
10	<code> </code>
11	<code>?? ?:</code>
12	<code>affectation</code>
13	<code>AND</code>
14	<code>XOR</code>

Structures de contrôle

if

- ❖ Elle permet l'exécution conditionnelle d'une partie de code.
- ❖ Si l'expression vaut TRUE, PHP exécutera l'instruction et si elle vaut FALSE, l'instruction sera ignorée
- ❖ Vous pouvez imbriquer indéfiniment des instructions if dans d'autres instructions if, ce qui permet une grande flexibilité dans l'exécution d'une partie de code suivant un grand nombre de conditions

```
if ($a > $b) {  
    echo "a est plus grand que b";  
    $b = $a;  
}
```

```
if ($a > $b)  
    echo "a est plus grand que b";
```

Si une seule instruction

else

- ❖ Elle fonctionne après un if et exécute les instructions correspondantes au cas où l'expression du if est FALSE.

```
if ($a > $b) {  
    echo "a est plus grand que b";  
} else {  
    echo "a est plus petit que b";  
}
```

```
if ($a > $b)  
    echo "a est plus grand que b";  
else  
    echo "a est plus petit que b";
```

Si une seule instruction

elseif / else if

- ❖ Elle permet d'exécuter une instruction après un if dans le cas où le "premier" if est évalué comme FALSE. Mais, à la différence de l'expression else, il n'exécutera l'instruction que si l'expression conditionnelle elseif est évaluée comme TRUE.
- ❖ L'expression elseif est exécutée seulement si le if précédent et tout autre elseif précédent sont évalués comme FALSE, et que votre elseif est évalué à TRUE.

```
if ($a > $b) {  
    echo "a est plus grand que b";  
} elseif ($a == $b) {  
    echo "a est égal à b";  
} else {  
    echo "a est plus petit que b";  
}
```

switch

- ❖ L'instruction switch équivaut à une série d'instructions if

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

while

- ❖ PHP exécute l'instruction tant que l'expression de la boucle while est évaluée comme TRUE.
- ❖ La valeur de l'expression est vérifiée à chaque début de boucle

```
while ($i <= 10) {  
    echo $i++;  
}
```

do while

- ❖ Les boucles do-while ressemblent beaucoup aux boucles while, mais l'expression est testée à la fin de chaque itération plutôt qu'au début.
- ❖ La principale différence par rapport à la boucle while est que la première itération de la boucle do-while est toujours exécutée

```
do {  
    echo $i;  
} while ($i > 0);
```

for

⌘ Boucle de longue déterminée

```
<?php  
  
for(expression_1 ; condition ; expression_2) {  
    // ...  
}  
  
?>
```

```
for ($i = 1; $i <= 10; $i++) {  
    echo $i;  
}
```

expression_1 : exécutée une fois à l'entrée de la boucle

Condition : testée à chaque fois qu'on repasse dans la boucle

expression_2 : exécutée à la fin d'un passage dans la boucle

Foreach

- ❖ foreach ne fonctionne que pour les tableaux et les itérateurs
- ❖ Il existe deux syntaxes :

```
foreach (array_expression as $value){  
    //commandes  
}
```

A chaque itération, la valeur de l'élément courant est assignée à \$value

```
foreach (array_expression as $key => $value){  
    //commandes  
}
```

A chaque itération, la valeur de l'élément courant est assignée à \$value et la clé de l'élément courant est assignée à la variable \$key

```
foreach ($arr as $value) {  
    echo $value ;  
}
```

```
foreach ($arr as $key => $value) {  
    echo $key, $value ;  
}
```

Instructions break

- ❖ L'instruction break permet de sortir d'une structure for, foreach, while, do-while ou switch.
- ❖ break accepte un argument numérique optionnel qui vous indiquera combien de structures emboîtées doivent être interrompues.

```
$tab = [1,3,5];
foreach($tab as $value){
    echo $value;
    break;
    echo "jamais affiché";
}
```

```
$tab = [1,3,5];
foreach($tab as $value){
    echo $value;
    while(true){
        echo "je suis dans le while
";
        break(2);
        echo "jamais affiché (whil
e)";
    }
    echo "apres le while jamais af
fiché";
    break;
    echo "jamais affiché";
}
```

Instruction continue

- ❖ L'instruction continue permet de sauter les instructions qui restent jusqu'à la fin de la boucle et de reprendre à la condition de boucle
- ❖ continue accepte un argument numérique optionnel qui vous indiquera combien de structures emboîtées doivent être éludées. La valeur par défaut est 1, ce qui revient à aller directement à la fin de la boucle courante.

```
$tab = [1,3,5];
foreach ($tab as $value) {
    if($value == 3){
        continue;
    }
    echo $value;
}
```

Test Unitaire

Test Unitaire

- ❖ En programmation informatique, le test unitaire est une procédure permettant de vérifier le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme
- ❖ Ecrire du code qui va tester le code
- ❖ Utilité
 - ❖ Trouver les erreurs rapidement
 - ❖ Sécuriser la maintenance
 - ❖ Documenter le code

PHP Unit

⌘ <https://phpunit.de>



src/Email.php

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

final class Email
{
    private $email;

    private function __construct(string $email)
    {
        $this->ensureIsValidEmail($email);

        $this->email = $email;
    }

    public static function fromString(string $email): self
    {
        return new self($email);
    }

    public function __toString(): string
    {
        return $this->email;
    }
}
```

tests/EmailTest.php

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

use PHPUnit\Framework\TestCase;

final class EmailTest extends TestCase
{
    public function testCanBeCreatedFromValidEmailAddress()
    {
        $this->assertInstanceOf(
            Email::class,
            Email::fromString('user@example.com')
        );
    }

    public function testCannotBeCreatedFromInvalidEmailAddress()
    {
        $this->expectException(InvalidArgumentException::class)

            Email::fromString('invalid');
    }

    public function testCanBeUsedAsString(): void
    {
    }
}
```

PHP Unit

assertEquals()

```
assertEquals(mixed $expected, mixed $actual[, string $message = ''])
```

Signale une erreur identifiée par `$message` si les deux variables `$expected` et `$actual` ne sont pas égales.

`assertNotEquals()` est l'inverse de cette assertion et prend les mêmes arguments.

`assertAttributeEquals()` et `assertAttributeNotEquals()` sont des encapsulateurs de commodités qui utilisent un attribut `public`, `protected` ou `private` d'une classe ou d'un objet en tant que valeur.

Example 1.16 Utilisation de assertEquals()

```
<?php
use PHPUnit\Framework\TestCase;

class EqualsTest extends TestCase
{
    public function testFailure()
    {
        $this->assertEquals(1, 0);
    }

    public function testFailure2()
    {
        $this->assertEquals('bar', 'baz');
    }

    public function testFailure3()
    {
        $this->assertEquals("foo\nbar\nbaz\n", "foo\nbah\nbaz\n");
    }
}
```

PHP Unit

assertSame()

```
assertSame(mixed $expected, mixed $actual[, string $message = ''])
```

Signale une erreur identifiée par `$message` si les deux variables `$expected` et `$actual` ne sont pas du même type et n'ont pas la même valeur.

`assertNotSame()` est l'inverse de cette assertion et prend les mêmes arguments.

`assertAttributeSame()` et `assertAttributeNotSame()` sont des encapsulateurs de commodités qui utilisent un attribut `public`, `protected` ou `private` d'une classe ou d'un objet en tant que valeur.

Example 1.44 Utilisation de assertSame()

```
<?php
use PHPUnit\Framework\TestCase;

class SameTest extends TestCase
{
    public function testFailure()
    {
        $this->assertSame('2204', 2204);
    }
}
```

```
$ phpunit SameTest
PHPUnit [version].0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) SameTest::testFailure
```

Visual Studio & PHPUnit

❖ Ajouter plugin :

❖ PHPUnit test Explorer

The screenshot shows the Visual Studio Code interface with the following details:

- SIDE BAR:** Contains icons for file operations (New, Open, Save, Find, etc.) and a search bar.
- TEST EXPLORER:** Shows a tree view of test cases. The node for `Chaine_1Test` is currently selected, highlighted with a blue background.
- CODE EDITOR:** Displays the PHP file `Chaine_4Test.php`. The code defines a test class `Chaine_4Test` that extends `TestCase`. It contains two test methods: `testExamples()` and `testExamples2()`, each using `assertEqual` or `assertSame` assertions.
- STATUS BAR:** Shows the following status indicators: PROBLEMS (3), OUTPUT, DEBUG CONSOLE, TERMINAL, and the footer text "PHPUnit 8.4.1 by Sebastian Bergmann and contributors."

```
<?php
include "src/chaine/Chaine_4.php";
use PHPUnit\Framework\TestCase;

class Chaine_4Test extends TestCase {
    public function testExamples() {
        $this->assertEquals(1234, stringToNumber("1234"));
        $this->assertEquals(605, stringToNumber("605"));
        $this->assertEquals(1405, stringToNumber("1405"));
        $this->assertEquals(-7, stringToNumber("-7"));
    }

    public function testExamples2() {
        $this->assertSame('67', numberToString(67));
        $this->assertSame('17', numberToString(17));
    }
}
```

Visual Studio & PHPUnit

```
tests > tests_calcul > ▶▶ Calcul_1Test.php
1  <?php
2  include "src/calcul/Calcul_1.php";
3  use PHPUnit\Framework\TestCase;
4
5
Run | Show in Test Explorer
6  class Calcul_1Test extends TestCase {
    Run | Show in Test Explorer
    ✓ 7   public function testExamples() {
        8     $this->assertEquals(goals[0,0,0], 0);
        9     $this->assertEquals(goals(43, 10, 5), 58);
10   }
11 }
12
13
```

```
tests > tests_calcul > ▶▶ Calcul_1Test.php
1  <?php
2  include "src/calcul/Calcul_1.php";
3  use PHPUnit\Framework\TestCase;
4
5
Run | Show in Test Explorer
6  class Calcul_1Test extends TestCase {
    Run | Show Log | Show in Test Explorer
    ✘ 7   public function testExamples() {
        8     $this->assertEquals(goals[0,0,1], 0); // Failed asserting that 0 matches expected 1.
        9     $this->assertEquals(goals(43, 10, 5), 58);
10   }
11 }
12
13
```

Environnement

❖ Télécharger Visual Studio

❖ <https://code.visualstudio.com>

❖ Installer plugin PHPUnit test Explorer

❖ Créer un répertoire de travail

❖ Installer phpunit dans le répertoire de travail

❖ <https://phpunit.de/getting-started/phpunit-8.html>

Download

PHP Archive (PHAR)

We distribute a [PHP Archive \(PHAR\)](#) that contains everything you need in order to use PHPUnit 8. Simply download it from [here](#) and make it executable:

```
→ wget -O phpunit https://phar.phpunit.de/phpunit-8.phar  
→ chmod +x phpunit  
  
→ ./phpunit --version  
PHPUnit 8.0.0 by Sebastian Bergmann and contributors.
```

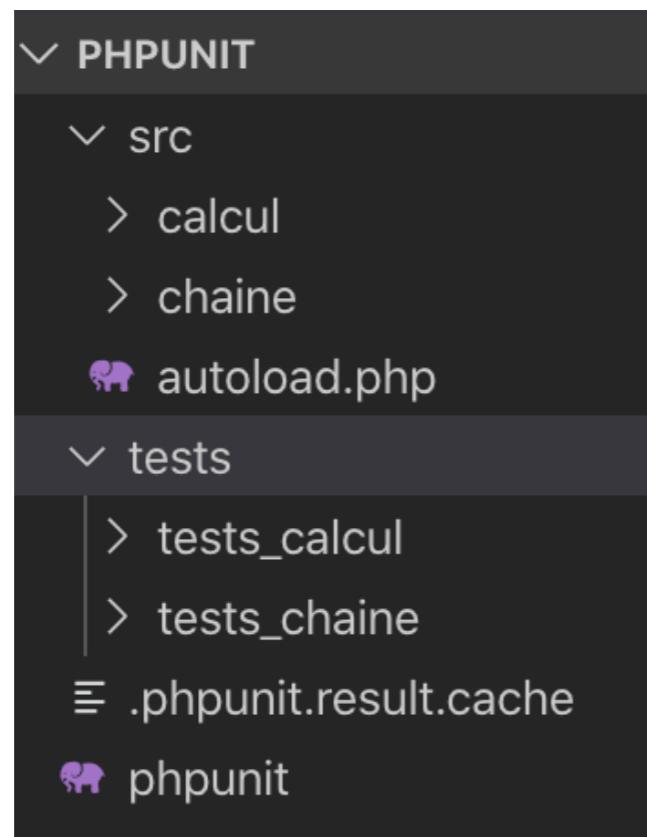
Please refer to the documentation for details on how to [verify PHAR releases of PHPUnit](#).

Environnement

❖ Télécharger les exercices

❖ jihane.fr/dfs17/phpunit_exo.zip

❖ Copier coller les 2 répertoires



- ❖ jihane.fr/dfs17/d12.pdf
- ❖ jihane.fr/dfs17/formationc.pdf

Inclure des fichiers

Include

❖ include

- ❖ inclut et exécute le fichier spécifié en argument.

```
<?php  
include "src/calcul/Calcul_2.php";  
|
```

❖ include_once :

- ❖ inclut et évalue le fichier spécifié durant l'exécution du script.

- ❖ si le code a déjà été inclus, il ne le sera pas une seconde fois

❖ require

- ❖ identique à include

- ❖ lorsqu'une erreur survient il stoppera le script

❖ require_once

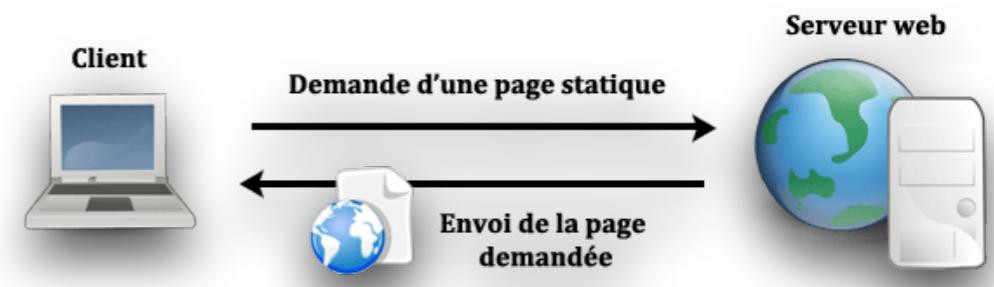
- ❖ vérifie si le fichier a déjà été inclus, et si c'est le cas, ne l'inclut pas une deuxième fois.

WEB

Page statique / Page dynamique

❖ Page statique

- ❖ Le contenu ne varie pas en fonction des caractéristiques de la demande
- ❖ Tous les internautes qui demandent la page reçoivent le même contenu.
- ❖ HTML, CSS



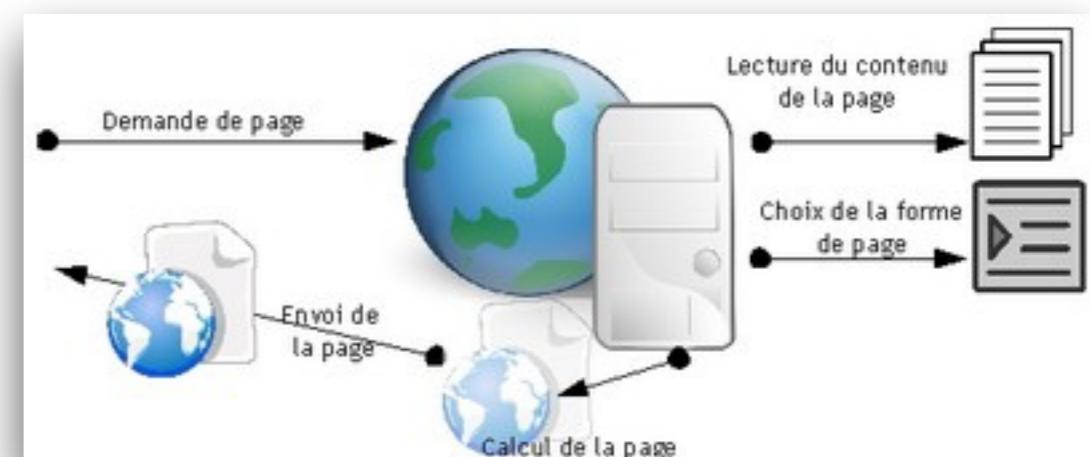
❖ Page dynamique

- ❖ Son contenu varie en fonction des caractéristiques de la demande
- ❖ Générée à la demande
- ❖ Langage interprété par le serveur

❖ PHP

❖ Java

❖



Interaction Client / Serveur

Interaction Client / Serveur

- ❖ Méthodes pour interagir
 - ❖ Liens (balise <a>)
 - ❖ Formulaire (balise <form>)

Les liens

- ❖ Lien : permet de naviguer entre les pages d'un site web

```
<a href="URL" id="ID" target="TARGET">CECI EST UN LIEN</a>
```

- ❖ URL : peut contenir des paramètres qui permettent de passer des informations d'une page à l'autre

```
URL_CLASSIQUE?nom=valeur [&nom=valeur ...]
```

```
site.com?page=1&section=2
```

```
site.com?document=titre du document&lang=fr&auteur=1
```

\$_GET

❖ **\$_GET**

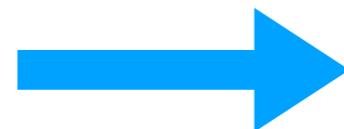
- ❖ Tableau associatif des valeurs passées au script courant via les paramètres d'URL

- ❖ Clé = nom du paramètre

- ❖ Valeur = valeur du paramètre

```
http://localhost:8080/test.php?login=john&lang=fr&doc=test
```

```
<?php  
echo 'Contenu de $_GET <br/>';  
foreach($_GET as $key => $value){  
    echo $key, " = ", $value, "<br/>";  
}  
echo '<br/>';  
echo 'Valeur de $_GET["login"] = ' . $_GET["login"];  
?>
```



Contenu de `$_GET`
login = john
lang = fr
doc = test

Valeur de `$_GET["login"]` = john

Formulaire

- ❖ L'élément HTML <form> représente une section d'un document qui contient des contrôles interactifs permettant à un utilisateur d'envoyer des données à un serveur web

```
<form method="post" action="login.php">
  <label for="exampleInputLogin">Email address</label>
  <input type="text" class="form-control" name="exampleInputLogin" placeholder="Enter login" id="name">

  <label for="exampleInputPassword">Password</label>
  <input type="password" class="form-control" name="exampleInputPassword" placeholder="Password">

  <input type="checkbox" class="form-check-input" name="exampleCheck">
  <label class="form-check-label" for="exampleCheck">Check me out</label>

  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

Formulaire

❖ Attribut method de la balise form

- ❖ définit la méthode HTTP qui sera utilisée pour envoyer les données au serveur
- ❖ get : correspondant à la méthode GET du protocole HTTP. Les données du formulaires sont ajoutées à l'URI de l'attribut action avec '?' comme séparateur.
- ❖ post : correspondant à la méthode POST du protocole HTTP, les données du formulaires sont incluses dans le corps du formulaire et envoyées vers le server.

```
<form action=login.php method=post>  
</form>
```

```
<form action=login.php method=get>  
</form>
```

Formulaire : Champ du formulaire

❖ Attribut **name** d'un champ d'un formulaire

❖ balise input

❖ Balise textarea

❖ ...

❖ Paramètres envoyés

❖ Clé = valeur de l'attribut name du champ du formulaire

❖ Valeur = valeur du champ du formulaire

```
<input type="text" class="form-control" name="exampleInputLogin" placeholder="Enter login">
```

\$_GET

⌘ \$_GET

- ⌘ Tableau associatif des valeurs passées au script courant via les paramètres d'URL

- ⌘ Clé = nom du paramètre

- ⌘ Valeur = valeur du paramètre

```
<form action=login.php method=get>
...
</form>
```

```
<?php
echo 'Contenu de $_GET <br/>';
foreach($_GET as $key => $value){
    echo $key, " = ", $value, "<br/>";
}
echo '<br/>';
echo 'Valeur de $_GET["exampleInputLogin"] = '. $_GET["exampleInputLogin"];
?>
```



Contenu de \$_GET
exampleInputLogin = john
exampleInputPassword = pmolik

Valeur de \$_GET["exampleInputLogin"] = john

\$_POST

⌘ \$_POST

- ⌘ Un tableau associatif des valeurs passées au script courant via le protocole HTTP et la méthode POST

- ⌘ Clé = nom du paramètre

- ⌘ Valeur = valeur du paramètre

```
<form action=login.php method=post>
...
</form>
```

```
<?php
echo 'Contenu de $_POST <br/>';
foreach($_POST as $key => $value){
    echo $key, " = ", $value, "<br/>";
}
echo '<br/>';
echo 'Valeur de $_POST["login"] = '.
$_POST["login"];
?>
```



Contenu de \$_POST
exampleInputLogin = marc
exampleInputPassword = blabla

Valeur de \$_POST["exampleInputLogin"] = marc

htmlspecialchars

❖ htmlspecialchars

- ❖ Convertit les caractères spéciaux en entités HTML

Caractère	Remplacement
& (ET commercial)	&
" (double guillement)	" sauf si ENT_NOQUOTES
' (simple guillemet)	' (pour ENT_HTML401) ou ' (pour ENT_XML1 , ENT_XHTML ou ENT_HTML5), mais seulement lorsque ENT_QUOTES est défini
< (inférieur à)	<
> (supérieur à)	>

```
<?php
$nom = htmlspecialchars("john & moi");
?>

<a href="test.php?login=<?= $nom ?>">CECI EST UN LIEN</a>

<?php
print_r($_GET);
?>
```



```
<a href="test.php?login=john & moi">CECI EST UN LIEN</a>
```

htmlentities

⌘ htmlentities

- ⌘ Convertit tous les caractères éligibles en entités HTML

```
$str = '<script>document.location=\'?cookie=\'' + document.cookie</script>';
echo htmlentities($str);
```



```
&lt;script&gt;document.location='?cookie=' + document.cookie&lt;/script&gt;
</body>
```

```
$str = '<script>document.location=\'?cookie=\'' + document.cookie</script>';
echo htmlentities($str, ENT_QUOTES);
```



```
&lt;script&gt;document.location=&#039;?cookie=&#039; + document.cookie&lt;/script&gt;
</body>
```

nl2br

- ❖ nl2br – Insère un retour à la ligne HTML à chaque nouvelle ligne

```
$a = "Ligne 1 \n Ligne 2\n";  
echo nl2br($a);
```

▼ <body>
 Ligne 1

 Ligne 2

strip_tags

- ❖ Supprime les balises HTML et PHP d'une chaîne

```
$b = "<b>Hello</b><h1>!!!</h1>";  
echo strip_tags($b);
```

```
▼ <body>  
  Hello!!!  
  <a href="test">  
▶ <form> ... </fo  
Array ( [logi
```

urlencode

⌘ urlencode

⌘ Encode une chaîne en URL

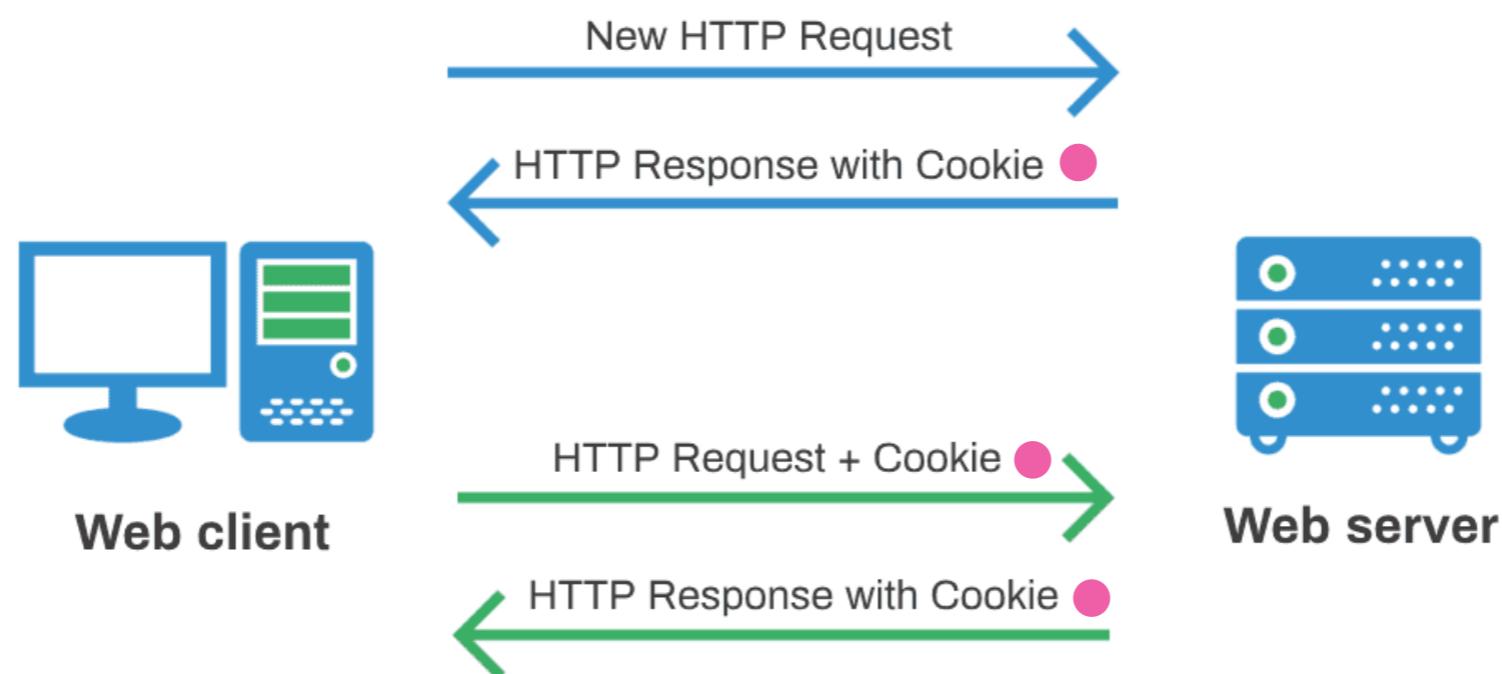
```
$foo = "alice & bob";
$bar =" pierre";
$query_string = 'foo=' . urlencode($foo) . '&bar=' . urlencode($bar);
echo '<a href="test.php?' . $query_string . '">Lien</a>';
```

▶▶▶
`Lien`

Cookies

Cookies

- ❖ Dialogue client/serveur est un dialogue sans état
- ❖ Le serveur ne stocke aucune information relative à une transaction
- ❖ Cookie = information envoyée par le serveur et stockée côté client sous forme d'un fichier texte et renvoyée par le client lors d'une nouvelle connexion



Envoi d'un cookie

- ❖ Fonction setcookie(nom, valeur)
- ❖ Gestion des cookies dans les en-têtes HTTP
 - ❖ Fonction doit être avant le traitement de la page
 - ❖ La mettre en haut des scripts

```
setcookie('NOM','marc');  
setcookie('dep',38);
```

Envoi d'un cookie

- ❖ Fonction setcookie(nom, valeur)
- ❖ Si la valeur que l'on souhaite enregistrée est de type complexe
 - ❖ il faut la serialiser avant de l'enregistrer dans le cookie
 - ❖ la déserialiser si on veut la lire a partir du cookie

```
$noms = ["marc", 'john', 'paul'];
setcookie( 'NOM', serialize($noms));
```

```
$tab = unserialize($_COOKIE['NOM']);
foreach($tab as $key => $value){
    echo $key, " = ", $value, "<br>";
}
```

Lecture d'un cookie

⌘ **\$_COOKIE**

⌘ Un tableau associatif de variables, passé au script courant, via des cookies HTTP.

```
foreach($_COOKIE as $key => $value){  
    echo $key, " = ", $value, "<br>";  
}  
echo $_COOKIE['NOM'];
```

NOM = marc
dep = 38
marc

Modifier un cookie

❖ Fonction setcookie(nom, valeur)

```
setcookie( 'NOM' , 'john' );
```

Validité et date d'expiration

- ❖ Par défaut, la validité d'un cookie est une session de navigation
- ❖ Ferme le navigateur : cookie effacé
- ❖ Mettre une date d'expiration
 - ❖ setcookie(nom, valeur, date)
 - ❖ Date
 - ❖ timestamp
 - ❖ mktime ([int \$hour = date("H") [, int \$minute = date("i") [, int \$second = date("s") [, int \$month = date("n") [, int \$day = date("j") [, int \$year = date("Y") [, int \$is_dst = -1]]]]]])

```
setcookie('NOM','marc',mktime(0,0,0,05,19,2020));
```

Effacer un cookie

- ❖ Modifier le cookie comme ceci

```
setcookie('NOM','','1');
```

- ❖ Penser à faire également un `unset($_COOKIE['NOM'])`

- ❖ Effacer la variable sur le serveur

```
setcookie('NOM','marc',1);
unset($_COOKIE['NOM']);
```

Session

Session

- ❖ Permet le dialogue de page en page
- ❖ Au lieu d'enregistrer les informations côté client, on les enregistre côté serveur
- ❖ Un identifiant de session est attribué au visiteur
- ❖ A chaque fois que cet identifiant est présenté , on récupère toutes les informations enregistrées
- ❖ Session = effacées après la fermeture du navigateur

Initialisation d'une session

- ❖ Pour initialiser une session : `session_start()`
- ❖ Si aucun identifiant de session
 - ❖ PHP en crée un aléatoirement
 - ❖ L'envoie au visiteur (cookie nommé PHPSESSID qui contient l'identifiant)
 - ❖ Crée le fichier de données correspondant
- ❖ Si présence d'un identifiant de session
 - ❖ PHP va chercher le fichier correspondant
 - ❖ PHP met à disposition les informations
- ❖ Doit être placée en début de script

```
session_start();
```

Lecture et écriture d'information

⌘ \$_SESSION

- ⌘ Tableau associatif des valeurs stockées dans les sessions
- ⌘ Variable superglobale
- ⌘ Aucune restriction sur les types des données enregistrés
- ⌘ Pas besoin de sérialiser

```
<?php  
session_start();  
  
$_SESSION['NOM'] = "Remy";
```

```
<?php  
session_start();  
  
echo $_SESSION['NOM'] ;
```

En tête http

⌘ header()

⌘ Envoyer des en-têtes HTTP

⌘ Exemple Redirection HTTP

```
header('Location: auth.php');
```

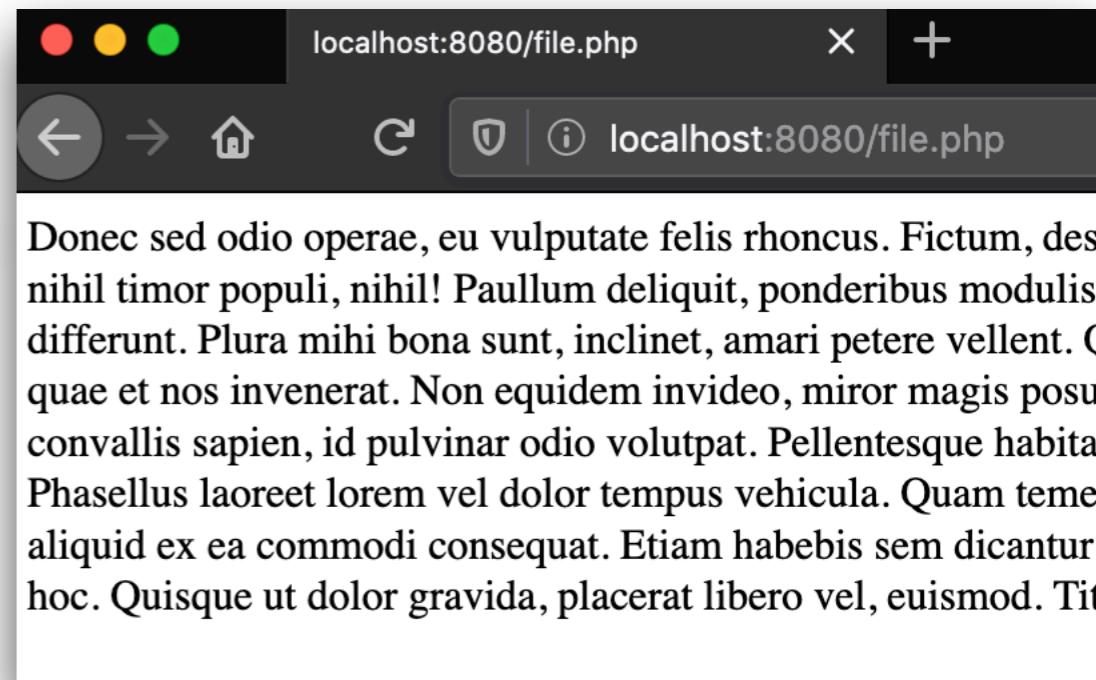
Fichier

Fonction d'accès rapide : Lecture

❖ file_get_contents()

- ❖ Lit tout un fichier dans une chaîne

```
<?php  
    echo file_get_contents('test.txt');
```



❖ readfile()

- ❖ Lit un fichier et l'envoie dans le buffer de sortie.

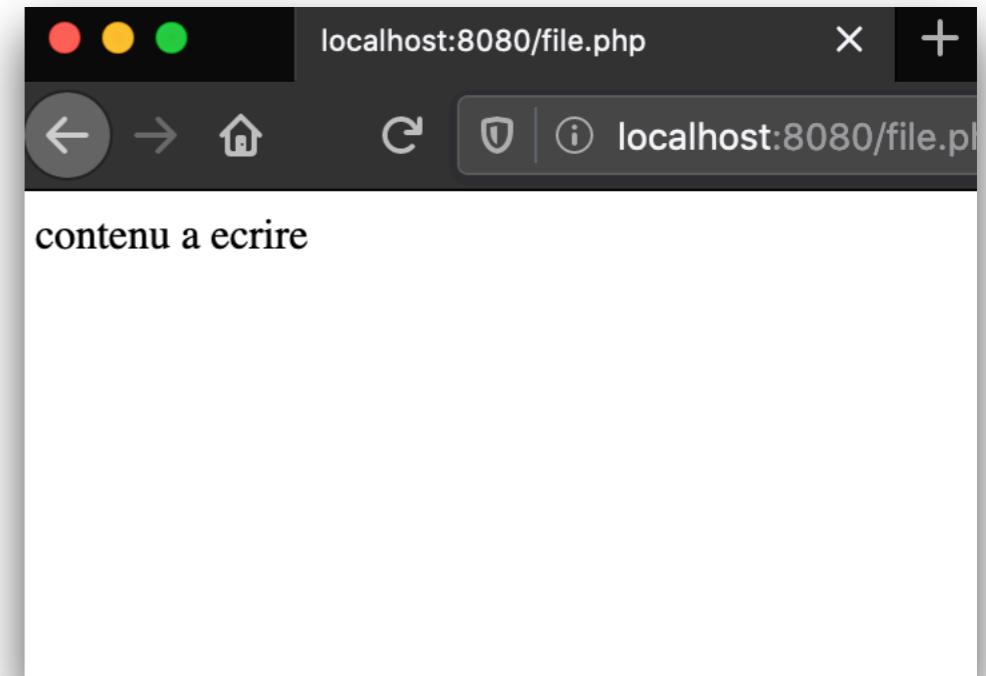
```
<?php  
    readfile('test.txt');
```

Fonction d'accès rapide : Ecriture

❖ file_put_contents

- ❖ Écrit des données dans un fichier

```
<?php  
    file_put_contents('test.txt', "contenu a ecrire");  
    readfile('test.txt');
```



❖ FILE_APPEND

- ❖ Ajoute le contenu à la fin du document

```
<?php  
    file_put_contents('test.txt', "contenu a ecrire", FILE_APPEND);  
  
    readfile('test.txt');
```



Ouverture d'un fichier

❖ fopen(string \$filename , string \$mode)

❖ Ouvre un fichier

mode	Description
'r'	Ouvre en lecture seule, et place le pointeur de fichier au début du fichier.
'r+'	Ouvre en lecture et écriture, et place le pointeur de fichier au début du fichier.
'w'	Ouvre en écriture seule ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
'w+'	Ouvre en lecture et écriture ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
'a'	Ouvre en écriture seule ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer. Dans ce mode, la fonction fseek() n'a aucun effet, les écritures surviennent toujours.
'a+'	Ouvre en lecture et écriture ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer. Dans ce mode, la fonction fseek() n'affecte que la position de lecture, les écritures surviennent toujours.
'x'	Crée et ouvre le fichier en écriture seulement ; place le pointeur de fichier au début du fichier. Si le fichier existe déjà, fopen() va échouer, en retournant FALSE et en générant une erreur de niveau E_WARNING . Si le fichier n'existe pas, fopen() tente de le créer. Ce mode est l'équivalent des options O_EXCL/O_CREAT pour l'appel système open(2) sous-jacent.
'x+'	Crée et ouvre le fichier pour lecture et écriture; le comportement est le même que pour 'x'.
'c'	Ouvre le fichier pour écriture seulement. Si le fichier n'existe pas, il sera créé, s'il existe, il n'est pas tronqué (contrairement à 'w') et l'appel à la fonction n'échoue pas (comme dans le cas de 'x'). Le pointeur du fichier est positionné au début. Ce mode peut être utile pour obtenir un verrou (voyez flock()) avant de tenter de modifier le fichier, utiliser 'w' pourrait tronquer le fichier avant d'obtenir le verrou (vous pouvez toujours tronquer grâce à ftruncate()).
'c+'	Ouvre le fichier pour lecture et écriture, le comportement est le même que pour le mode 'c'.
'e'	Défini l'indicateur close-on-exec sur le descripteur de fichier ouvert. Disponible uniquement en PHP compilé sur les systèmes conformes POSIX.1-2008.

Lecture d'un fichier

- ❖ `fgetc()` = Lire caractère par caractère
- ❖ `fgets()` = Lire ligne par ligne
- ❖ `fread()` = Lire le fichier en entier

```
// 1er caractère
echo fgetc($f);
// boucle sur le contenu du fichier + affichage caractère par caractère
while(!feof($f)){
    echo fgetc($f);
}
```

```
// 1ere ligne
echo fgets($f);
// boucle sur le contenu du fichier + affichage ligne par ligne
while(!feof($f)){
    echo fgets($f);
}
```

```
echo fread($f, filesize('test.txt'));
```

Ecriture dans un fichier

❖ fwrite()

```
$f = fopen('test.txt','a');
if(!$f){
    echo "fichier introuvable";
}else{

    fwrite($f,"Ajout en fin de fichier");
}
```

Fermeture d'un fichier

❖ `fclose()`

```
fclose($f);
```

Parcourir un répertoire

⌘ dir()

```
$dir = dir('.');  
    while($nom = $dir->read()){  
        echo $nom, '<br/>';  
    }
```