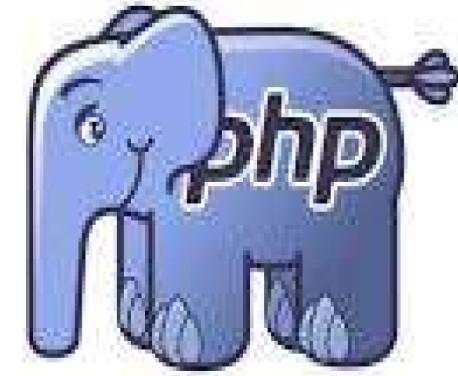


# Plan du cours

1. Chapitre 1 : Introduction à la programmation web
2. Chapitre 2 : Langage HTML 5
3. Chapitre 3 : Feuilles de style CSS
4. Chapitre 4 : Le langage JavaScript
- 5. Chapitre 5 : Le langage PHP**

## Chapitre 5



# Le langage PHP

# Partie I

# Introduction au langage PHP

# Langage de scripts PHP

---

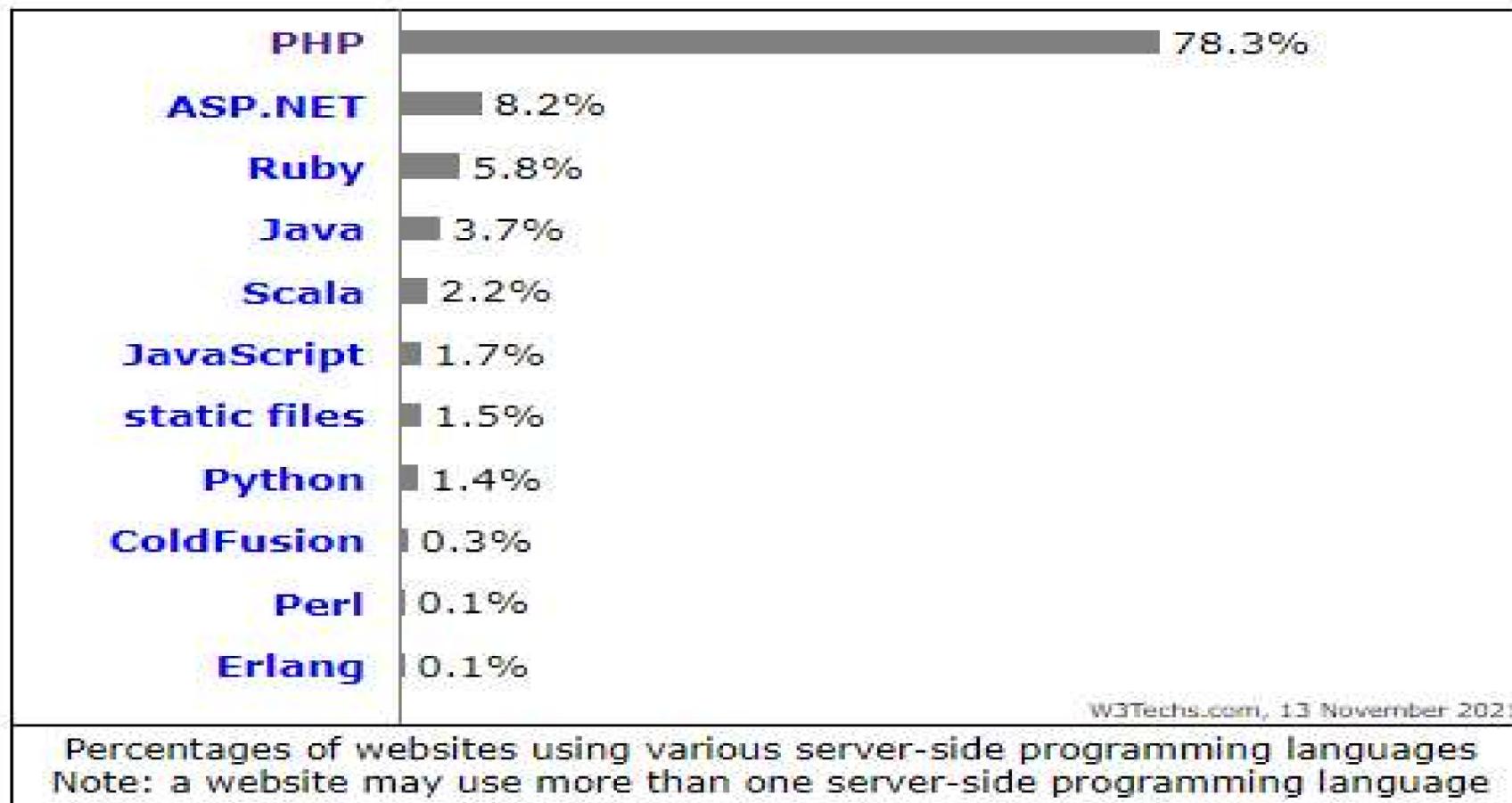
- PHP est un langage de scripts exécutable sur le serveur
  - Le principe est le même que ASP ou JSP : imbriquer le code PHP dans la page HTML (cad mixer le code client et serveur)
- 
- Pourquoi?
    - PHP est déployé sur 53% des serveurs dans le monde soit 1 site sur 3
    - 46% des sites web français

# Langage de scripts PHP

---

- **Php** : Hypertext Preprocessor
  - Langage de script HTML interprété côté serveur
  - ==/= Java script qui s'exécute côté client
  - Le **php** permet de réaliser des sites web dynamiques
  - Le **php** est disponible dans plusieurs environnements tel qu'Unix (linux, AIX), Windows (98/NT/XP/7/8/10)
  - **Php** possède une impressionnante quantité d'outils (manipulation d'images, traitement de fichiers, accès aux bases de données...)
  - **Php** contient des **instructions** : demande au serveur d'effectuer des actions
-

# Pourquoi PHP



Src : <https://w3techs.com>

# Pourquoi PHP (Avantages)

---

- Un **interfaçage** simple et efficace avec un grand nombre de SGBD.
  - Un langage de script **puissant**, complet et relativement **facile** à apprendre, qui offre une très bonne intégration avec HTML.
  - L'existence **d'outils** associés très **performants**: le SGBD MySQL, le serveur Web Apache et une documentation foisonnante à disposition. De plus, les **interpréteurs PHP** sont présents sur de nombreux serveurs et sont eux aussi performants.
  - **PHP** est **libre** et **Open Source**, et profite donc des évolutions apportées par la communauté des développeurs PHP.
-

# Pourquoi PHP (Inconvénients)

---

- **PHP** ne résoud cependant pas tous les problèmes.
  - Le rôle de **PHP** s'exécute exclusivement côté serveur. il est donc nécessaire de l'utiliser pour certaines applications et d'utiliser des scripts clients pour ce qui concerne le navigateur client.
  - Développer un site en **PHP** requiert des connaissances non seulement en **programmation**, mais également en **conception** et gestion de banques de **données**. C'est notamment via le langage standard d'interrogation des bases de données SQL (Structured Query Language) que les scripts PHP accèdent aux données.
-

# Le langage PHP

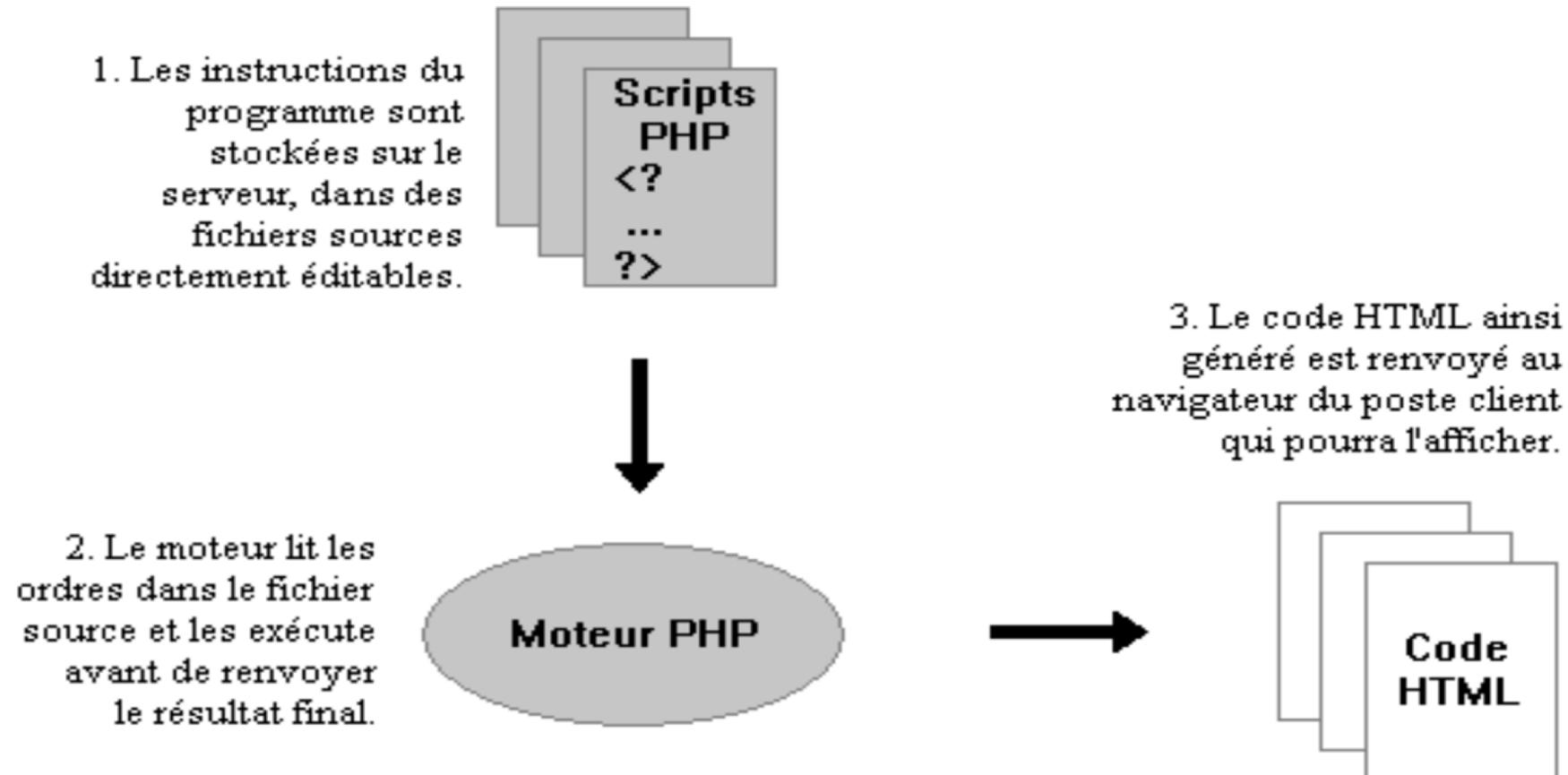
- PHP est un langage de scripts exécutable sur le serveur
- Le principe est le même que ASP ou JSP : imbriquer le code PHP dans la page HTML (cad mixer le code client et serveur)
- Exemple:

```
<HTML>
<HEAD>
<TITLE>Test HelloWorld en PHP</TITLE>
</HEAD>
<BODY>
<? echo "Hello world !"; ?>
</BODY>
</HTML>
```

- La syntaxe du langage est proche de C et Perl
  - instructions terminées par un point-virgule ;
  - Bloc d'instructions délimité par des accolades {}
  - Commentaires comme C et C++

# Architecture d'un site web avec php

---



# Langage de scripts PHP

---

Le **moteur d'interprétation** du langage **lit** un **fichier source PHP**, en respectant les définitions et règles suivantes :

- Un **bloc PHP** est un groupe continu de lignes, encadré par deux balises : <? et ?> ou <?php et php?>
  - Toute **ligne située à l'extérieur** de ces balises n'est pas interprétée et est envoyée telle quelle dans le flux de sortie.
  - Toute **ligne située à l'intérieur** de ces balises est considérée comme une **instruction PHP** et est donc interprétée par le moteur.
  - Les instructions **PHP** n'apparaissent pas dans le résultat généré.
  - Lorsqu'une erreur survient, un message est intégré dans le flux de sortie, et la génération du script est interrompue.
-

# Le langage (Exemple)

## PHP : script serveur

- Ce que le serveur voit et execute:
  - <HTML>
  - <HEAD>
  - <TITLE>Test HelloWorl en PHP</TITLE>
  - </HEAD>
  - <BODY>
  - <? echo "Hello world !"; ?>
  - </BODY>
  - </HTML>
- Ce que le navigateur reçoit:
  - <HTML>
  - <HEAD>
  - <TITLE>Test HelloWorl en PHP</TITLE>
  - </HEAD>
  - <BODY> **Hello world !**
  - </BODY>
  - </HTML>
- On ne peut donc pas voir ou copier le source PHP depuis le navigateur

# php côté serveur

---

```
<html>
<head>
<title>Exemple</title>
</head>
<body>
<?php
echo ("Bonjour, je suis un script
PHP ! ") ;
?>
</body>
</html>
```

---

# php côté serveur

---

```
<html>
<head>
<title>Exemple</title>
</head>
<body>
<?php
echo ("Bonjour, je suis un script
PHP!") ;
?>
</body>
</html>
```

# php côté client

---

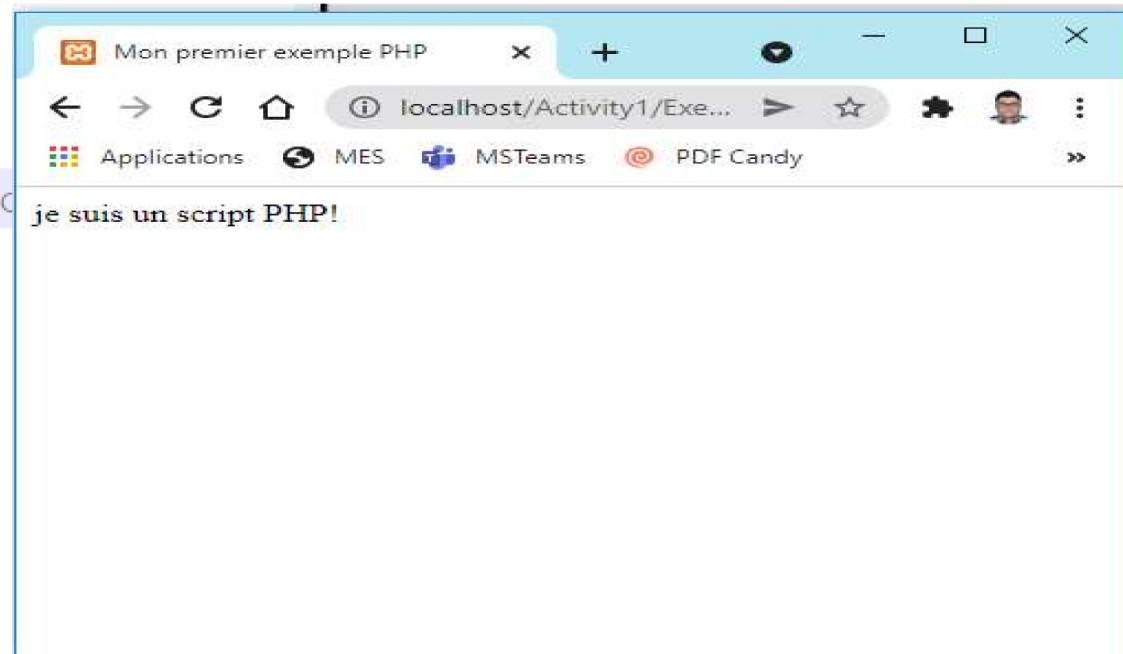
```
<html>
<head>
<title>Exemple</title>
</head>
<body>
Bonjour, je suis un script PHP!
</body>
</html>
```

---

# php côté client

---

```
<html>
  <head>
    <title>
      Mon premier exemple PHP
    </title>
  <body>
    <?php echo ("je suis un sc...>
  </body>
</html>
```



# **Partie II**

# **Initiation au langage PHP**

# Syntaxe du langage

---

## Séparateur d'instruction + commentaires

Bases de la syntaxe héritées du C et du Perl :

- séparateur d'instructions -> ;
- commentaires :
  - ...code /\* ...mes commentaires... \*/ code...,
  - ...code... // ...mes commentaires...,
  - ...code... # ...mes commentaires....

**Il est largement conseillé de commenter votre code**

# Syntaxe du langage

---

- Le typage des variables est implicite en php. Il n'est donc pas nécessaire de déclarer leur type au préalable ni même de les initialiser avant leur utilisation.
- Les noms de variable sont précédés du symbole « \$ » (dollars).

Exemple :

**\$toto = 5;**

- Il est possible de convertir une variable en un type primitif grâce au cast<sup>(1)</sup> (comme en C).

Exemple :

**\$str = "12";** // \$str vaut la chaîne "12"

**\$nbr = (int)\$str;** // \$nbr vaut le nombre 12

(1) : Le cast est une conversion de type. L'action de caster consiste en convertir une variable d'un type à un autre.

# Syntaxe du langage

---

## Les types de données

PHP supporte les types de données suivants :

- nombres entiers(**integer**)
- nombres à virgule flottante(**double**)
- chaînes de caractères(**string**)
- tableaux(**array**)
- objets (développés dans la section 'programmation orientée objet')(**objet**)

## Deux types spéciaux : null et resource

# Syntaxe du langage

---

## Quelques fonctions

**empty(\$var)** : renvoie vrai si la variable est vide

**isset(\$var)** : renvoie vrai si la variable existe

**unset(\$var)** : détruit une variable

**gettype(\$var)** : retourne le type de la variable

**settype(\$var, "type")** : convertit la variable en type

**type (cast)**

**is\_long(), is\_double(), is\_string(), is\_array(),**

**is\_object(), is\_bool(), is\_float(), is\_numeric(),**

**is\_integer(), is\_int()...**

# Syntaxe du langage

**Exemple : détermination du type d'une variable**

**Avant de manipuler des variables, en utilisant par exemple, des opérateurs, il peut être utile de connaître leur type**

```
string gettype ($maVariable)
```

**Cette API retourne une chaîne de caractères contenant le type de la variable**

# Syntaxe du langage

Les APIs suivantes permettent de vérifier si une variable est d'un type précis

- `is_integer($maVariable)` ou `is_int(( $maVariable ))`
- `is_double($maVariable)`
- `is_string($maVariable)`
- `is_bool($maVariable)`
- `is_array($maVariable)`
- `is_object($maVariable)`
- `is_resource($maVariable)`
- `is_null($maVariable)`

Elles retournent la valeur Booléenne TRUE si la variable est du type recherché et FALSE dans le cas contraire

# Syntaxe du langage

---

## Spécification d'un type « entier »

Il est possible de spécifier une variable de type *entier* de la façon suivante :

- \$toto = 123 ; # est un entier en base 10,
- \$toto = -123 ; # est un entier négatif,
- \$toto = 0123 ; # est un entier en base 8,
- \$toto = 0x123 ; # est un entier en base 16.

Il est possible de spécifier une variable de type *flottant* ou *double* de la façon suivante :

- \$titi = 1.234 ; # est un nombre à virgule flottante,
- \$titi = 1.2e3 ; # est aussi un nombre à virgule flottante.

# Syntaxe du langage

## Spécification d'un type « string »

Il est possible de spécifier une variable de type *chaîne de caractères* de la façon suivante :

- \$personne = 'M. Smith' ; # est une chaîne de caractères,
- \$personne = "M. Smith" ; # est aussi une chaîne de caractères.

Dans le deuxième cas, si la chaîne contient des noms de variables, celles-ci seront remplacées par leur valeur ;

### Exemple

```
$type = 'M.' ;  
  
$nom = "Smith" ;  
  
$personne = "$type $nom" ;
```

Equivalent à \$personne = 'M. Smith' ;.

# Syntaxe du langage

---

## Spécification d'un type « string »

## Echappement aux caractères spéciaux

Quand on utilise les "..." on doit donc échapper certains caractères avec un backslash (\) pour pouvoir les afficher comme tels :

- dollar (\$) : \\$,
- double quotes ("") : \" ,
- backslash (\) : \.

De même, il existe des caractères spéciaux qui nécessitent d'être échappés :

- nouvelle ligne : \n,
- retour à la ligne : \r,
- tabulation : \t.

Tout autre caractère échappé générera un avertissement ( *warning* ) ;

# Syntaxe du langage

---

## Spécification d'un type «booléen»

L'utilisation d'expressions booléennes est à la base de la création des instructions conditionnelles

Il est possible de spécifier une variable de type booléen de la façon suivante :

- **\$maVariable = TRUE;** // ou encore \$maVariable = true
- **\$maVariable = False;** // ou encore \$maVariable = false
- **\$maVariable = valeur "vrai" ou "faux"** qui peuvent être prises par une expression conditionnelle. Exemple a<10, ...

# Syntaxe du langage

---

## Spécification d'un type «booléen»

### Règles d'évaluation «booléenne» des expressions

#### 1. Expression évaluée à FALSE

- le mot clé `FALSE`
- La valeur entière `0` de type `integer`
- La valeur décimale `0.0` de type `double`
- La chaîne "`0`" de type `string`<sup>(2)</sup>
- Une variable de type `NULL`
- Une variable non initialisée
- Un tableau vide
- Un objet sans propriété
- Toute expression logique fausse utilisant un ou plusieurs opérateurs

#### 2. Expression évaluée à TRUE

- Toutes les autres possibilités, y compris l'entier `-1`, car il est non nul, et la chaîne "`false`", car elle est non vide

(2) les chaînes "`0`" et '`0`' castées en l'entier `0` lui même casté en `FALSE`.

# Syntaxe du langage

---

## Spécification d'un type «booléen»

### Règles d'évaluation «booléenne» des expressions

#### Exemple 1:

```
if(0) echo 1;      // faux
if("") echo 2;     // faux
if("0") echo 3;    // faux
if("00") echo 4;
if('0') echo 5;    // faux
if('00') echo 6;
if(" ") echo 7;
```

- Cet exemple affiche **467**. Donc l'espace ou la chaîne “00” ne sont pas considérés castés en **FALSE**.

#### Exemple 2:

```
$a=15;
If($a) {echo "$a existe et vaut $a";}
```

# Syntaxe du langage

## Spécification d'un type «array»

Les tableaux de PHP ressemblent aux tableaux associatifs (*hash-tables*) du Perl ;

L'index dans le tableau est appelé *clé* et peut être indifféremment un *entier* ou *une chaîne de caractères* ;

La valeur associée à une clé est appelée *valeur* ;

On peut utiliser la fonction `array()` pour créer un tableau ;

On peut aussi affecter directement les valeurs au tableau ;

La fonction `list()` permet d'affecter des variables comme si elles constituaient un tableau ;

# Syntaxe du langage

---

## Spécification d'un type «array»

```
// pour initialiser un tableau vide avec PHP>=5.4.0
$tableau = [];
// pour initialiser un tableau indexé avec PHP>=5.4.0
$tableau = ['élément1', 'élément2'];
// pour initialiser un tableau associatif avec PHP>=5.4.0
$tableau = ['clé1' => 'élément1', 'clé2' => 'élément2'];
// pour initialiser un tableau vide quelque soit la version de PHP
$tableau = array();
// pour initialiser un tableau indexé quelque soit la version de PHP
$tableau = array('élément1', 'élément2');
// pour initialiser un tableau associatif quelque soit la version de PHP
$tableau = array('clé1' => 'élément1', 'clé2' => 'élément2');
// Le premier index d'un tableau indexé est 0
// Pour atteindre un élément d'un tableau il suffit de préciser
// son index ou le nom de la clé entre crochets $tableau[$cle]
// Un tableau indexé peut être parcouru par
// for ($i=0; $i<count($tableau); $i++)
// {$tableau[$i]}
// Quelque soit le type de tableau, il peut être parcouru par
// foreach ($tableau as $cle => $valeur) {...}
```

# Syntaxe du langage

## Spécification d'un type «array»

## Quelques APIs

- **sizeof(\$tab)** : retournent le nombre d'éléments d'un tableau,
- **reset(\$tab)**: place le pointeur interne sur le premier élément et retourne sa valeur,
- **next(\$tab)**: place le pointeur interne sur l' élément suivant et retourne sa valeur,
- **prev(\$tab)** : place le pointeur interne sur l' élément précédent et retourne sa valeur,
- **each(\$tab)**: retourne la paire clé/valeur courante du tableau et avance le pointeur sur l'élément suivant.

# Syntaxe du langage

Spécification d'un type «array»

Quelques APIs

Un tableau peut être trié en utilisant les fonctions suivantes

- **asort(\$tab)/arsort(\$tab)** : trient le tableau en ordre croissant/décroissant de valeurs,
- **ksort(\$tab)/rsort(\$tab)** : trient le tableau en ordre croissant/décroissant de clés,
- **sort(\$tab)** : trie le tableau en ordre croissant clés et valeurs (on perd la correspondance clé/valeur).

# Syntaxe du langage

---

## Quelques APIs pour les tableaux associatifs

- **array\_keys(\$tab)** : retourne un tableau contenant les clés du tableau associatif \$tab,
- **array\_values(\$tab)** : retourne un tableau contenant les valeurs du tableau associatif \$tab,
- **array\_search(\$val,\$tab)** : retourne la clé associée à la valeur \$val

# Syntaxe du langage

## Spécification d'un type «objet»

**PHP permet l'utilisation des classes et utilise le type « object » pour toute variable créée en tant qu'instance d'une classe**

```
<?php
class Personne {
    // définition de la classe
}
$var = new Personne;
echo " Le type de la variable \"\$var est:", gettype($var);
?>
```

# Syntaxe du langage

## Spécification d'un type «resource»

Le type « resource » représente une référence à une information présentée sur le serveur. Il est le type retourné par certaines APIs particulières. C'est le cas par exemple des fonctions utilisées pour récupérer un résultat auprès d'un serveur.

## Spécification d'un type «NULL»

Le type « NULL » ou « null », est celui qui est attribué à une variable qui n'a pas de contenu ou qui a été explicitement initialisée avec la valeur NULL

# Syntaxe du langage

## Les chaînes de caractères

### Concaténation de chaînes de caractères

L'opérateur de concaténation est le point (.), il fusionne deux chaînes littérales ou contenues dans des variables en une seule

```
<?php  
$a = "PHP";  
$b = "MySQL";  
$c = "Utiliser" . $a . "et" . $b . "pour construire un site  
dyn";  
echo $c;  
?>
```

# Syntaxe du langage

## Les chaînes de caractères

### Quelques APIs utiles

```
<?php
$str = "PhpMySql";
If (strlen($str) != 0) {echo "$str est de longueur ". strlen($str)."<br>"}
echo ("conversion de ". $str . " en minuscule : ". strtolower($str)."<br>");
echo ("conversion de ". $str . " en majuscule : ". strtoupper($str)."<br>");
//transformation de chaînes en tableau
$email = "mal.bensalem@gmail.com";
$tab = explode("@", $email);
echo "le nom d'utilisateur est : " . $tab[0] . ", et <br>".
      " son serveur de messagerie est : " . $tab[1];
// la liste de fonctions est longue, je l'arrête ici !!!!
```

# Syntaxe du langage

## Les chaînes de caractères

### Variable dynamique

**Une variable dynamique prend la valeur d'une variable et l'utilise comme nom d'une autre variable**

```
$toto = "Hello" ; # $toto vaut Hello  
  
$$toto = "World" ; # $Hello vaut World  
  
echo "$toto $Hello !" ; # affiche Hello World !  
  
echo "$toto ${$toto} !" ; # affiche aussi Hello World !
```

# Syntaxe du langage

## les opérateurs d'affectation

Les opérateurs d'affectation :

- l'opérateur d'affectation le plus simple est le signe =,
- il ne signifie pas "égal à" mais que l'opérande à gauche du signe = se voit affecté de la valeur de l'opérande de droite,
- la valeur renvoyée par une expression d'assignement est la valeur assignée,

# Syntaxe du langage

## les opérateurs d'affectation

- il existe en plus des *opérateurs combinés* pour tous les opérateurs arithmétiques, les opérateurs bits à bits et l'opérateur de concaténation,

<code>+ =</code>	<code>- =</code>	<code>* =</code>	<code>/ =</code>	<code>% =</code>	<code>. =</code>
<code>&amp; =</code>	<code>  =</code>	<code>^ =</code>	<code>&lt;&lt;=</code>	<code>&gt;&gt;=</code>	<code>~ =</code>

- ceux-ci permettent d'utiliser la valeur d'une variable dans une expression et d'affecter le résultat de cette expression à cette variable,

# Syntaxe du langage

## les opérateurs d'affectation

- l'opérateur `++` est équivalent à `+= 1`,
- l'opérateur `--` est équivalent à `-= 1`,
- ces deux opérateurs peuvent être placés avant (pré-exécution) ou après (post-exécution) la variable à laquelle ils s'appliquent.

Ex :

```
$toto = 0 ;  
echo ++$toto ; # affiche 1  
echo $toto++ ; # affiche 1  
echo $toto ; # affiche 2
```

# Syntaxe du langage

## les opérateurs de comparaison

---

Les opérateurs de comparaison :

- égal à : \$a == \$b,
- différent de : \$a != \$b,
- supérieur à : \$a > \$b,
- inférieur à : \$a < \$b,
- supérieur ou égal à : \$a >= \$b,
- inférieur ou égal à : \$a <= \$b.

# Syntaxe du langage les opérateurs de comparaison

---

Les opérateurs logiques :

- ET (vrai si \$a et \$b vrais) :
  - \$a and \$b,
  - \$a && \$b.
- OU (vrai si \$a ou \$b vrai(s)) :
  - \$a or \$b,
  - \$a || \$b.
- OU-Exclusif/XOR (vrai si seul \$a ou \$b vrai) : \$a xor \$b,
- NON (vrai si \$a est faux) : !\$a.

# Syntaxe du langage les opérateurs de comparaison

---

L'opérateur ternaire :

- issu du langage C,
- (condition) ? (expression1) : (expression2) ;,
- renvoie expression1 si condition est vraie et expression2 dans le cas contraire.

# Syntaxe du langage

## Structures de contrôle

---

Tous les scripts PHP sont une suite d'instructions ;

Une instruction peut être :

- un assignement,
- un appel de fonction,
- une instruction conditionnelle, ou
- une instruction qui ne fait rien (une instruction vide).

Une instruction se termine habituellement par un point virgule ( ; ) ;

# Syntaxe du langage

## Structures de contrôle

### Instructions conditionnelles

L'instruction `if` est une des plus importantes instructions de tous les langages, PHP inclus ;

Elle permet l'exécution conditionnelle d'une partie de code ;

Les fonctionnalités de l'instruction `if` sont les mêmes en PHP qu'en C ;

Prototype :

```
if (condition) {  
    # instructions à exécuter si la condition est vraie...  
}
```

# Syntaxe du langage

## Structures de contrôle

### Instructions conditionnelles

Prototype :

```
if (condition1) {  
    # instructions à exécuter si la condition1 est vraie...  
}  
elseif (condition2) {  
    # instructions à exécuter si la condition2 est vraie...  
}  
elseif (condition3) {  
    # instructions à exécuter si la condition3 est vraie...  
} ...  
else {  
    # instructions à exécuter si aucune des conditions n'est vraie...  
}
```

# Syntaxe du langage

## Structures de contrôle

### Instructions conditionnelles

Prototype simplifié :

```
if (condition1) :  
    # instructions à exécuter si la condition1 est vraie...  
  
elseif (condition2) :  
    # instructions à exécuter si la condition2 est vraie...  
  
elseif (condition3) :  
    # instructions à exécuter si la condition3 est vraie...  
  
...  
  
else :  
    # instructions à exécuter si aucune des conditions n'est vraie...  
  
endif ;
```

# Syntaxe du langage

## Structures de contrôle

---

### La boucle while

Prototype :

```
while (condition) {  
    # instructions à exécuter tant que la condition est vraie...  
}
```

Prototype simplifié :

```
while (condition) :  
    # instructions à exécuter tant que la condition est vraie...  
endwhile ;
```

# Syntaxe du langage

## Structures de contrôle

---

### La boucle do while

Prototype :

```
do {
```

# instructions à exécuter la première fois et ensuite tant que la condition est vraie...

```
} while (condition) ;
```

# Syntaxe du langage

## Structures de contrôle

---

### La boucle for

Prototype :

```
for (expression1 ; condition ; expression2) {  
# instructions à exécuter tant que la condition est vraie...  
}
```

Prototype simplifié :

```
for (expression1 ; condition ; expression2) :  
# instructions à exécuter tant que la condition est vraie...  
endfor ;
```

# Syntaxe du langage

## Structures de contrôle

### La boucle foreach

Avec l'instruction `foreach` cette boucle devient :

```
foreach ($_POST as $cle => $valeur) {  
    echo "$cle => $valeur, "  
}
```

Ou bien, si l'on ne veut récupérer que les valeurs :

```
foreach ($_POST as $valeur) {  
    echo "$valeur, "  
}
```

# Syntaxe du langage

## Structures de contrôle

### L'instruction break

L'instruction `break` permet de sortir d'une boucle à n'importe quel moment;

Ex : compter jusqu'à 10

```
for ($i = 1, , $i++) {  
    if ($i > 10) break;  
    echo "$i " ;  
}
```

Affiche 1 2 3 4 5 6 7 8 9 10.

# Syntaxe du langage

## Structures de contrôle

### L'instruction continue

L'instruction `continue` permet d'ignorer les instructions restantes dans la boucle et de passer directement à l'itération suivante ;

Ex : compter 2 par 2 jusqu'à 10

```
for ($i = 1, $i <= 10, $i++) {  
    if ($i % 2) continue; # $i impaire  
    echo "$i ";  
}
```

Affiche 2 4 6 8 10.

# Syntaxe du langage

## Structures de contrôle

---

### L'instruction switch

L'instruction switch équivaut à une série d'instructions

if..elseif..elseif....else;

Elle est utilisée pour comparer la même variable (ou expression) avec un grand nombre de valeurs différentes et d'exécuter différentes parties de code suivant la valeur à laquelle elle est égale ;

# Syntaxe du langage

## Structures de contrôle

### L'instruction switch

Prototype :

```
switch (expression) {  
    case resultat1 :  
        # instructions à exécuter si l'expression vaut resultat1...  
        break ;  
    case resultat2 :  
        # instructions à exécuter si l'expression vaut resultat2...  
        break ;  
    ...  
    default :  
        # instructions à exécuter en dernier recours...  
}
```

# Syntaxe du langage

## Les constantes

---

### Définition d'une constante

PHP définit certaines constantes et propose un mécanisme pour en définir d'autres durant l'exécution du script ;

On définit une constante en utilisant la fonction `define()` ;

Ex :

```
define("MA_CONSTANTE", "Bonjour") ;  
echo MA_CONSTANTE ; # affiche Bonjour
```

# Syntaxe du langage

## Les constantes

---

### Les constantes prédéfinies

Les principales constantes définies par PHP sont les suivantes :

- `__FILE__` : nom du fichier actuellement exécuté,
- `__LINE__` : numéro de la ligne qui est actuellement exécutée,
- `PHP_VERSION` : présentation de la version du PHP utilisée (ex : `3.0.8-dev`),
- `PHP_OS` : système d'exploitation utilisé par la machine qui fait tourner le PHP (ex : Linux),
- `TRUE` : vrai,
- `FALSE` : faux,
- `E_*` : gestion des erreurs (cf. section correspondante).

# Syntaxe du langage

## Les constantes

---

### Les constantes prédéfinies

```
<?php
define("CONSTANTE", "Bonjour le monde.");
echo CONSTANTE; // affiche "Bonjour le monde."
echo Constante; /* affiche "Constante" et une note
d'exécution (message d'erreur). */
defined(CONSTANTE); // retourne TRUE
// pour afficher les constantes déjà définies
print_r(get_defined_constants());
?>
```

# Syntaxe du langage

## Les variables

---

### Portée des variables « locale » « globale »

La portée d'une variable dépend du contexte dans lequel elle est définie ;

La plupart des variables ont une portée qui s'étend sur l'intégralité du script PHP, elles sont *globales* ;

Cependant, les variables sont *locales* au sein d'une fonction ;

# Syntaxe du langage

## Les variables

---

Deux façons existent pour accéder à une variable globale au sein d'un bloc :

- déclarer la variable comme global au sein du bloc,
- utiliser le tableau associatif `$GLOBALS` avec comme clé le nom de la variable globale.

# Syntaxe du langage

## Les variables d'environnement

Certaines variables sont prédéfinies par PHP, elles sont disponibles pendant toute l'exécution du script.

PHP_SELF	URI du script en cours d'exécution.
GLOBALS	Tableau des variables globales.
HTTP_GET_VARS	Tableau des variables issues de GET.
HTTP_POST_VARS	Tableau des variables issues de POST.
HTTP_COOKIE_VARS	Tableau des variables issues des cookies.

```
<?php phpinfo(); //pour lister toutes les variable  
//prédéfinies ?>
```

# Syntaxe du langage

## Les variables d'environnement

Variables d'environnement CGI :

SERVER_SOFTWARE	SERVER_NAME
GATEWAY_INTERFACE	SERVER_PROTOCOL
SERVER_PORT	REQUEST_METHOD
PATH_INFO	PATH_TRANSLATED
SCRIPT_NAME	QUERY_STRING
REMOTE_HOST	REMOTE_ADDR
AUTH_TYPE	REMOTE_USER
REMOTE_IDENT	CONTENT_TYPE
CONTENT_LENGTH	

<http://hoohoo.ncsa.uiuc.edu/cgi/env.html>

# Syntaxe du langage

## Les fonctions en PHP

Une fonction peut être définie en utilisant la syntaxe suivante

```
<?php  
function nom_fonction ([parm1, ... parmn] ) {  
instruction_1;  
...  
instruction_m;  
[return $resultat;] }  
?>
```

Tous les types de variables peuvent être renvoyés  
(tableaux et objets compris)

# Syntaxe du langage

## Les fonctions en PHP

---

- **function** : mot clé annonçant la définition d'une fonction
- **nom\_fonction()** : nom de la fonction, servira à son appel
- **parm<sub>i</sub>** : liste de paramètres sur lesquels opère la fonction. Une fonction peut avoir 0 ou ++ parm
- **return** : mot clé permettant d'associer un résultat à la fonction

# Syntaxe du langage

## Les fonctions en PHP

### Les fonctions – passage par valeur, par référence

Des informations peuvent être passées à une fonction en utilisant un tableau d'arguments dont chaque élément est séparé par une virgule. Un élément peut être une variable ou une constante ;

PHP peut supporter :

- le passage d'arguments par valeur (méthode par défaut),
- le passage d'arguments par référence.

# Syntaxe du langage

## Les fonctions en PHP

---

### Les fonctions – passage par valeur

- Les variables **ne sont pas affectées par des changements au sein** de la fonction.
- On peut donc changer la valeur des arguments au sein de la fonction sans que ceci **ait des répercussions** à l'extérieur de celle-ci.

# Syntaxe du langage

## Les fonctions en PHP

Les fonctions - par référence

**Les variables sont affectées par des changements au sein de la fonction**

Deux possibilités :

- De façon permanente en ajoutant un & devant le nom de la variable dans la définition de la fonction,
- De façon ponctuelle en ajoutant un & devant le nom de la variable lors de l'appel à la fonction.

# Syntaxe du langage

## Les fonctions en PHP

---

### Les fonctions standards (API)

- PHP met à notre disposition un très grand nombre de fonctions :  
<http://fr.php.net/>
- sur les chaînes de caractères,
- sur les tableaux, sur les fichiers,
- mathématiques,
- etc.

# Syntaxe du langage

## Les fonctions en PHP

---

### Les fonctions require et require\_once

- La commande `require (string nom_fichier)` se remplace elle-même par le contenu du fichier.
- Utile pour charger en mémoire un script contenant des fonctions usuelles.
- Une erreur d'évaluation du fichier génère une erreur terminale lors d'un `require`.
- L'usage de `require_once` évite de charger plusieurs fois le même fichier.

# Syntaxe du langage

## Les fonctions en PHP

---

### Les fonctions `include` et `include_once`

- La fonction `include(string nom_fichier)` inclus et évalue le fichier spécifié en argument.
- Utile pour insérer un en-tête ou pied de page récurrent dans un site web, pour exécuter un script selon certaines conditions, ...
- À la différence de `require`, une erreur d'évaluation du fichier génère un warning avec un `include`.

# Syntaxe du langage

## Les fonctions en PHP

### Une fonction utile : isset

■ la fonction `isset()`, dont la syntaxe est la suivante :  
boolean `isset ($var)`, retourne :

1. **FALSE** si la variable `$var` n'est pas initialisé ou a une valeur **NULL**,
2. **TRUE** si elle a une valeur quelconque.

■

```
<?php $nom="Molka";  
      echo (isset($nom)) ;  
      unset($nom) ;  
      echo (isset($nom)) ;  
?> //TRUE
```

# Syntaxe du langage

## Les fonctions en PHP

### Une fonction utile : empty

■ la fonction `empty()`, dont la syntaxe est la suivante:

`boolean empty($var)`, retourne :

1. **TRUE** si la variable `$var` **n'est pas initialisé** ou a la valeur **0** ou **NULL** ou **encore la chaîne "0";**,
2. **FALSE** si elle a une valeur quelconque

■

```
<?php $cpt="0";  
      echo if(empty($cpt) { //TRUE  
      echo "$cpt vaut 0 ou elle est non renseignée; }  
?>
```

# **Partie 3**

# **Gestion de formulaire en PHP**



This document was created with the Win2PDF “print to PDF” printer available at  
<http://www.win2pdf.com>

This version of Win2PDF 10 is for evaluation and non-commercial use only.

This page will not be added after purchasing Win2PDF.

<http://www.win2pdf.com/purchase/>