PHP & MySQL

ITAHRIOUAN Zakaria

itahriouan@upf.ac.ma

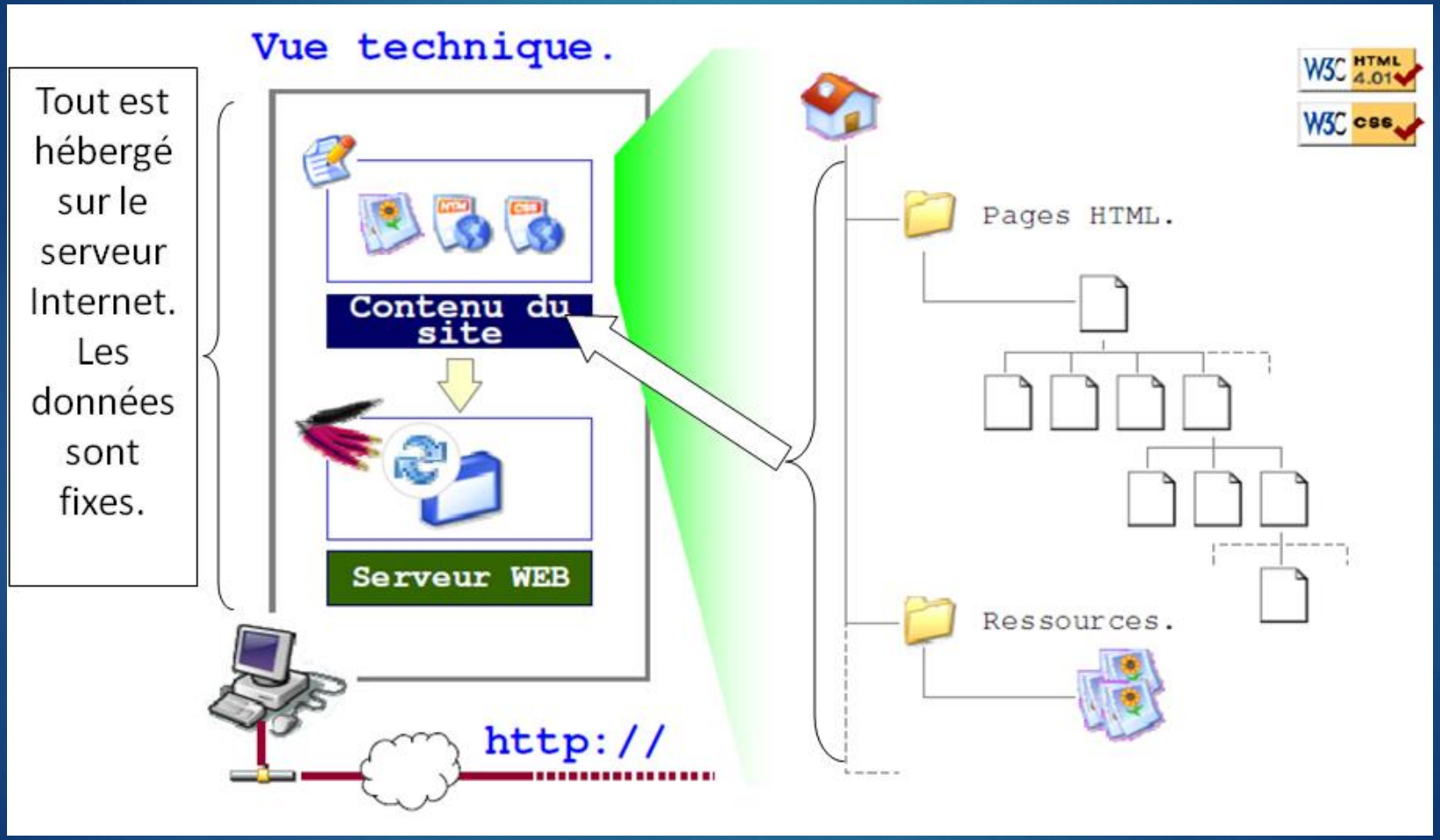
Introduction(1)

- Internet et les pages web
  - HTML : conception de pages destinées à être publiées sur Internet
  - Page html : contient le texte à afficher et des instructions de mise en page
  - HTML est un langage de description de page et non pas un langage de programmation
    - pas d'instructions de calcul ou pour faire des traitements suivant des conditions
  - Des sites de plus en plus riches en informations
    - Nécessité croissante d'améliorer le contenu de sites
    - Mises à jour manuelles trop complexes
      - Pourquoi ne pas automatiser les mises à jour ?

Introduction(2)

- Pages web statiques : fonctionnement
  - Leurs contenus ne changent ni en fonction du demandeur ni en fonction d'autres paramètres éventuellement inclus dans la requête adressée au serveur. Toujours le même résultat.
    - Rôle du serveur : localiser le fichier correspondant au document demandé et répondre au navigateur en lui envoyant le contenu de ce fichier
- Pages web statiques : limites
  - Besoin de réponses spécifiques : passage de pages statiques à pages dynamiques

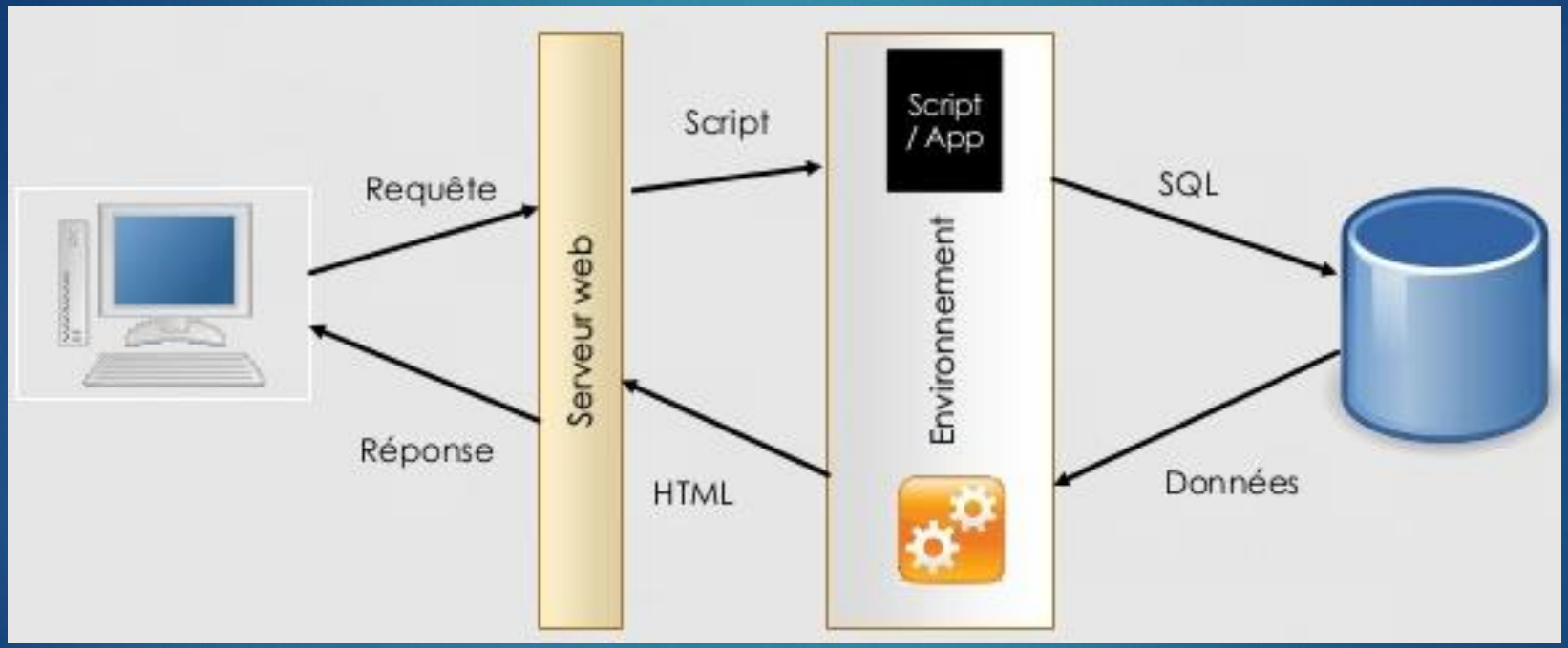
Introduction(3)



Introduction(4)

- Les langages de script-serveur : Définition
  - Un langage de script -serveur est :
    - un programme stocké sur un serveur et exécuté par celui-ci,
    - qui passe en revue les lignes d'un fichier source pour en modifier une partie du contenu,
    - avant de renvoyer à l'appelant (un navigateur par exemple) le résultat du traitement.
  - La tâche d'interprétation des ordres à exécuter est déléguée à un composant, souvent appelé moteur,
    - installé sur le serveur,
    - qui est doté d'une API et d'un fonctionnement identique quel que soit la plate-forme utilisée pour gérer le serveur

Introduction(5)



Introduction(6)

- Pages web dynamiques côté serveur ou côté client
  - **Langage côté client :** traité par la machine qui accueille le logiciel de navigation.

    - Ses résultats peuvent varier en fonction de plate-forme utilisée. Un programme en JavaScript pourra fonctionner sous chrome et poser problème sous Internet explorer.
    - Les résultats peuvent être différents suivant la machine (PC, Mac)
    - Nécessité des tests importants
    - Ne permettent pas de masquer les sources du programme
    - Sont indépendants du serveur

Introduction(7)

- Pages web dynamiques côté serveur ou côté client
  - Langage côté serveur : le travail d'interprétation du programme est réalisé par le serveur
    - Sont indépendants de la machine et du logiciel de navigation utilisés pour la consultation.
    - Sont compatibles avec tous les navigateurs et toutes leurs versions.
    - Permettent de masquer les sources de ses programmes
    - Nécessitent de recharger la page chaque fois que celle-ci est modifiée.
  - Pages côté serveur et côté client :
    - Script côté client pour des calculs et des traitement simples, des évènements …
    - Scripts côté serveur pour des calculs, des traitements et des mises à jour plus conséquents

PHP : C'est QUOI ?

- Définition
  - Un langage de scripts permettant la création d'applications Web
  - Indépendant de la plate-forme utilisée puisqu'il est exécuté côté serveur et non côté client.
  - La syntaxe du langage provient de celles du langage C, du Perl et de Java.
  - Ses principaux atouts sont:
    - La gratuité et la disponibilité du code source (PHP est distribué sous licence GNU GPL)
    - La simplicité d'écriture de scripts
    - La possibilité d'inclure le script PHP au sein d'une page HTML
    - La simplicité d'interfaçage avec des bases de données
    - L'intégration au sein de nombreux serveurs web (Apache, Microsoft IIS, ...)

Intégration PHP et HTML (1)

- Principe
  - Les scripts PHP sont généralement intégrés dans le code d'un document HTML
  - L'intégration nécessite l'utilisation de balises
    - Avec le style php: <?php ligne de code PHP ?>
    - Comme avec le JavaScript :

<script language='text/javascript'> ligne de code javascript </script>

Intégration PHP et HTML (2)

- Forme d'une page PHP
  - Intégration directe

```
 < ?php
//ligne de code PHP
?>
<html>
<head> <title> Mon script PHP </title> </head>
<body>
//ligne de code HTML
< ?php
//ligne de code PHP
?>
//ligne de code HTML
….
</body> </html>
```

Intégration PHP et HTML (3)

- Forme d'une page PHP
  - Inclure un fichier PHP dans un fichier HTML : include()

Fichier Prinipal

```
<html>
<head>
<title> Fichier d'appel </title>
</head>
<body>
<?php
$salut = " BONJOUR" ;
include "information.php" ;
?>
</body>
</html>
```

Fichier à inclure : information.php

```
<?php
$chaine=$salut. " , C'est PHP "  ;
echo "  <table border= '3'
 <tr> <td width = '100%' >
  <h2> ".$chaine. " </h2>
  </td> </tr></table> ";
?>
```

Intégration PHP et HTML (4)

- Envoi du code HTML par PHP
  - La fonction echo : echo Expression;
    - echo "Chaine de caractères";
    - echo (1+2)*87;
  - La fonction print : print(expression);
    - print("Chaine de caracteres");
    - print ((1+2)*87);
  - La fonction printf : printf (chaîne formatée);
    - printf ("Le périmètre du cercle est %d",$Perimetre);

Intégration PHP et HTML (4)

Exercice d'application 1:

- Intégrer trois paragraphes HTML séparées dans une page principale qui contient le titre.

Syntaxe de base : Introduction

- Typologie
  - Toute instruction se termine par un point-virgule
  - Sensible à la casse
- Les commentaires
  - /* Voici un commentaire! */
  - // un commentaire sur une ligne

Syntaxe de base : les constantes

- Les constantes
  - **Define**("nom_constante", valeur_constante)
    - define (“ma_const”, “Vive PHP”) ;
    - define (“an”, 2020) ;
  - **Const** nom_constante= valeur_constante,
    - const ma_const = “Vive PHP” ;
    - const an= 2020 ;
  - Les constantes prédéfinies (Exemples)
    - NULL
    - PHP_VERSION
    - PHP_OS
    - TRUE et FALSE

Syntaxe de base : les variables (1)

- Principe
  - Commencent par le caractère $
  - N'ont pas besoin d'être déclarées
- Fonctions de vérifications de variables
  - Doubleval(), intval(), strval()
  - is_array(), is_bool(), is_double(), is_float(), is_int(), is_integer, is_long(), is_object(), is_numeric(), is_string()
  - Isset(), empty()
  - settype(variable, 'type'), gettype(variable)
- Affectation par valeur et par référence
  - Affectation par valeur : $b=$a
  - Affectation par (référence) variable : $c = &$a

Syntaxe de base : les variables (1)

Exercice d'application 3:

- Utiliser la fonction doubleval() pour extraire la partie réelle d'une chaine variée.
- en utilisant la fonction gettype() vérifier si le type de la variable a changé.
- en utilisant la fonction gettype() vérifier le type de retour de la fonction empty(). Utiliser la fonction empty pour vérifier si une variable est vide.
- Réaliser un script php permettant de vérifier la différence entre l 'affectation par valeur et par variable.

Syntaxe de base : les variables (2)

- Visibilité des variables
  - Variable locale
    - Visible uniquement à l'intérieur d'un contexte d'utilisation
  - Variable globale
    - Visible dans tout le script
    - Utilisation de l'instruction global dans des contextes locales

```
<?
$a = 1;
$b = 2;
function somme() {
    global $a, $b;
    return $s = $a + $b;
}
somme();
?>
```

Syntaxe de base : les variables (4)

- Les variables dynamiques
  - Permettent d'affecter un nom différent à une autre variable

```
$nom_variable = 'nom_var';
$$nom_variable = valeur; // équivaut à $nom_var = valeur;
```

  - Les variables tableaux sont également capables de supporter les noms dynamiques

```
$nom_variable = array("val0", "val1", ...,"valN");
${$nom_variable[0]} = valeur;
➔$val0 = valeur;
$nom_variable = "nom_var";
${$nom_variable}[0] = valeur;
➔$nom_var[0] = valeur;
```

  - Les accolades servent aussi à éviter toute confusion lors du rendu d'une variable dynamique

```
echo "Nom : $nom_variable - Valeur : ${$nom_variable}";
// équivaut à echo "Nom : $nom_variable - Valeur : $nom_var";
```

Syntaxe de base : Les types de données

- Principe
  - Pas besoin d'affecter un type à une variable avant de l'utiliser
    - La même variable peut changer de type en cours de script
    - Les variables issues de l'envoi des données d'un formulaire sont du type string
- Les différents types de données
  - Les entiers : le type Integer
  - Les flottants : le type Double ou float
  - Les tableaux : le type array
  - Les chaînes de caractères : le type string
  - Les objets

Syntaxe de base : Les types de données (2)

- Le transtypage
  - La fonction settype() permet de convertir le type auquel appartient une variable

```
<?php     $nbre=10;
    Settype($nbre, " double");
    Echo " la variable $nbre est de type " , gettype($nbre); ?>
```

  - Transtypage explicite : le cast

```
<?php   $var=" 100 FRF ";
    Echo " pour commencer, le type de la variable est : gettype($var);
    $var =(double) $var;
    Echo <br> Après le cast, le type de la variable est $var ", gettype($var);
    Echo "<br> et a la valeur $var ";      ?>
```

- Détermination du type de données
  - Gettype(), Is_long(), Is_double(), Is_string(), Is_array(), Is_object(), Is_bool()

Syntaxe de base : Les chaînes de caractères(1)

- Principe
  - Peuvent être constituées de n'importe quel caractère alphanumérique et de ponctuation, y compris les caractères spéciaux
  - Une chaîne de caractères doit être toujours entourée par des guillemets simples (')ou doubles (")

" Ceci est une chaîne de caractères valide."

'Ceci est une chaîne de caractères valide.'

"Ceci est une chaîne de caractères invalide.'

Syntaxe de base : les opérateurs (1)

- Les opérateurs
  - les opérateurs de calcul
  - les opérateurs d'assignation
  - les opérateurs d'incrémentation
  - les opérateurs de comparaison
  - les opérateurs logiques
  - les opérateurs bit-à-bit
  - les opérateurs de rotation de bit

Syntaxe de base : Les opérateurs(2)

- Les opérateurs de calcul

Opérateur	Dénomination	Effet	Exemple	Résultat
+	opérateur d'addition	Ajoute deux valeurs	$x+3	10
-	opérateur de soustraction	Soustrait deux valeurs	$x-3	4
*	opérateur de multiplication	Multiplie deux valeurs	$x*3	21
/	plus: opérateur de division	Divise deux valeurs	$x/3	2.3333333
=	opérateur d'affectation	Affecte une valeur à une variable	$x=3	Met la valeur 3 dans la variable $x

Syntaxe de base : Les opérateurs(3)

- Les opérateurs d'assignation

Opérateur	Effet
+=	addition deux valeurs et stocke le résultat dans la variable (à gauche)
-=	soustrait deux valeurs et stocke le résultat dans la variable
*=	multiplie deux valeurs et stocke le résultat dans la variable
/=	divise deux valeurs et stocke le résultat dans la variable
%=	donne le reste de la division deux valeurs et stocke le résultat dans la variable
|=	Effectue un OU logique entre deux valeurs et stocke le résultat dans la variable
^=	Effectue un OU exclusif entre deux valeurs et stocke le résultat dans la variable
&=	Effectue un Et logique entre deux valeurs et stocke le résultat dans la variable
.=	Concatène deux chaînes et stocke le résultat dans la variable

Syntaxe de base : Les opérateurs(4)

- Les opérateurs d'incrémentation

Opérateur	Dénomination	Effet	Syntaxe	Résultat (avec x valant 7)
++	Incrémentation	Augmente d'une unité la variable	$x++	8
--	Décrémentation	Diminue d'une unité la variable	$x--	6

- Les opérateurs de comparaison

Opérateur	Dénomination	Effet	Exemple	Résultat
==	opérateur d'égalité	Compare deux valeurs et vérifie leur égalité	$x==3	Retourne 1 si $X est égal à 3, sinon 0
<	opérateur d'infériorité stricte	Vérifie qu'une variable est strictement inférieure à une valeur	$x<3	Retourne 1 si $X est inférieur à 3, sinon 0
<=	opérateur d'infériorité	Vérifie qu'une variable est inférieure ou égale à une valeur	$x<=3	Retourne 1 si $X est inférieur à 3, sinon 0
>	opérateur de supériorité stricte	Vérifie qu'une variable est strictement supérieure à une valeur	$x>3	Retourne 1 si $X est supérieur à 3, sinon 0
>=	opérateur de supériorité	Vérifie qu'une variable est supérieure ou égale à une valeur	$x>=3	Retourne 1 si $X est supérieur ou égal à 3, sinon 0
!=	opérateur de différence	Vérifie qu'une variable est différente d'une valeur	$x!=3	Retourne 1 si $X est différent de 3, sinon 0

Syntaxe de base : Les opérateurs(5)

- Les opérateurs logiques

Opérateur	Dénomination	Effet	Syntaxe
|| ou OR	OU logique	Vérifie qu'une des conditions est réalisée	((condition1)||(condition2))
&& ou AND	ET logique	Vérifie que toutes les conditions sont réalisées	((condition1)&&(condition2))
XOR	OU exclusif	Opposé du OU logique	((condition1)XOR(condition2))
!	NON logique	Inverse l'état d'une variable booléenne (retourne la valeur 1 si la variable vaut 0, 0 si elle vaut 1)	(!condition)

- Les opérateurs bit-à-bit

Opérateur	Dénomination	Effet	Syntaxe	Résultat
&	ET bit-à-bit	Retourne 1 si les deux bits de même poids sont à 1	9 & 12 (1001 & 1100)	8 (1000)
|	OU bit-à-bit	Retourne 1 si l'un ou l'autre des deux bits de même poids est à 1 (ou les deux)	9 | 12 (1001 | 1100)	13 (1101)
^	OU exclusif bit-à-bit	Retourne 1 si l'un des deux bits de même poids est à 1 (mais pas les deux)	9 ^ 12 (1001 ^ 1100)	5 (0101)
~	Complément (NON)	Retourne 1 si le bit est à 0 (et inversement)	~9 (~1001)	6 (0110)

Syntaxe de base : Les opérateurs(6)

- Les opérateurs de rotation de bit

Opérateur	Dénomination	Effet	Syntaxe	Résultat
<<	Rotation à gauche	Décale les bits vers la gauche. Les zéros qui sortent à gauche sont perdus, tandis que des zéros sont insérés à droite	6 << 1 (110 << 1)	12 (1100)
>>	Rotation à droite avec conservation du signe	Décale les bits vers la droite. Les zéros qui sortent à droite sont perdus, tandis que le bit non-nul de poids plus fort est recopié à gauche	6 >> 1 (0110 >> 1)	3 (0011)

- Autres opérateurs

Opérateur	Dénomination	Effet	Syntaxe	Résultat
.	Concaténation	Joint deux chaînes bout à bout	"Bonjour"."Au revoir"	"BonjourAu revoir"
$	Référencement de variable	Permet de définir une variable	$MaVariable = 2;	
->	Propriété d'un objet	Permet d'accéder aux données membres d'une classe	$MonObjet->Propriete	

Syntaxe de base : Les instructions conditionnelles(1)

- L'instruction if
  - if (condition réalisée) { liste d'instructions }
- L'instruction if ... Else
  - if (condition réalisée) {liste d'instructions}
    else { autre série d'instructions }
- L'instruction if ... elseif ... Else
  - if (condition réalisée) {liste d'instructions}
    elseif (autre condition) {autre série d'instructions }

    else (dernière condition réalisée) { série d'instructions }
- Opérateur ternaire
  - (condition) ? instruction si vrai : instruction si faux

Syntaxe de base : Les instructions conditionnelles(2)

- L'instruction switch

```
switch (Variable) {
case Valeur1: Liste d'instructions break;
case Valeur1: Liste d'instructions break;
case Valeurs...: Liste d'instructions break;
default: Liste d'instructions break;
}
```

Syntaxe de base : Les instructions Itératives

- La boucle for
  - for ($i=1; $i<6; $i++) { echo "$i
"; }
- La boucle while
  - While(condition) {bloc d'instructions ;}
  - While (condition) :Instruction1 ;Instruction2 ;
    …. endwhile ;
- La boucle do…while
  - Do {bloc d'instructions ;}while(condition) ;
- La boucle foreach (PHP4)
  - Foreach ($tableau as $valeur) {insts utilisant $valeur ;}

Syntaxe de base : Les tableaux(1)

- Principe
  - Création à l'aide de la fonction array()
  - Uniquement des tableaux à une dimension
    - Les éléments d'un tableau peuvent pointer vers d'autres tableaux
  - Les éléments d'un tableau peuvent appartenir à des types distincts
  - L'index d'un tableau en PHP commence de 0
  - Pas de limites supérieures pour les tableaux
  - La fonction count() pour avoir le nombre d'éléments d'un tableau

Syntaxe de base : Les tableaux(2)

- Les tableaux indicés et les tableaux associatifs
  - Tableau indicé
    - Accéder aux éléments par l'intermédiaire de numéros

```
$tableau[indice] = valeur;
$jour[3] = "Mercredi";
$note[0] = 20;

$tableau = array(valeur0, valeur1,..., valeurN);
$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi",
  "Jeudi", "Vendredi", "Samedi");
$note = array(20, 15, 12.6, 17, 10, 20, 11, 18, 19);

$variable = $tableau[indice];
$JJ = $jour[6]; // affecte "Samedi" à $JJ
echo $note[1] + $note[5];
```

Syntaxe de base : Les tableaux(3)

- Les tableaux indicés et les tableaux associatifs
  - Tableau associatif (ou table de hachage)
    - **Les éléments sont référencés par des chaînes de caractères associatives en guise de nom: la clé d'index**

```
$tableau["indice"] = valeur;
$jour["Dimanche"] = 7
$jour["Mercredi"] = "Le jour des enfants"
```

```
$tableau = array(ind0 => val0, ind1 => val1,..., indN => valN);
$jour = array("Dimanche" => 1, "Lundi" => 2, "Mardi" => 3,
  "Mercredi" => 4, "Jeudi" => 5, "Vendredi" => 6, "Samedi" => 7);
```

```
$variable = $tableau["indice"];
$JJ = $jour["Vendredi"]; //affecte 6 à $JJ
echo $jour["Lundi"]; //retourne la valeur 2
```

Syntaxe de base : Les tableaux(4)

- Tableaux multidimensionnels
  - Pas d'outils pour créer directement des tableaux multidimensionnels
  - L'imbrication des tableaux est possible

```
$tab1 = array(Val0, Val1,..., ValN);
$tab2 = array(Val0, Val1,..., ValN);
// Création d'un tableau à deux dimensions
 $tableau = array($tab1, $tab2);
$mois = array("Janvier", "Février", "Mars", "Avril", "Mai",
   "Juin", "Juillet", "Août", "Septembre", "Octobre", "Novembre",
   "Décembre");
$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi",
   "Vendredi", "Samedi");
$element_date = array($mois, $jour);

$variable = $tableau[indice][indice];
$MM = $element_date[0][0]; //affecte "Janvier" à $MM
echo $element_date[1][5] . " 7 " . $element_date[0][2] . "2002";
   // retourne " Vendredi 7 Mars 2002"
```

Syntaxe de base : Les tableaux(5)

- Lecture des éléments d'un tableau
  - Avec une boucle for

```
for ($i=0; $i<count($tab) ; $i++){
if ($tab[$i]== "a") {echo $tab[$i], "<br  />"; }}
```

  - Avec une boucle while

```
$i=0;
while ($tab[$i]){
if ($tab[$i][0] =="a" ) {echo $tab[$i], "<br /> "; }}
```

  - Avec La boucle foreach

```
$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi",
   "Vendredi", "Samedi");
$i = 0;
foreach($jour as $JJ) { echo "La cellule n° ". $i . " : " . $JJ .
   "<br>"; $i++; }
```

Syntaxe de base : Les tableaux(6)

- Lecture des éléments d'un tableau
  - Parcours d'un tableau associatif
    - Réalisable en ajoutant avant la variable $valeur, la clé associée

```
$tableau = array(clé1 => val1, clé2 => val2, ..., cléN =>
  valN);
foreach($tableau as $clé => $valeur) {
echo "Valeur ($clé): $valeur"; }

$jour = array("Dimanche" => 7, "Lundi" => 1, "Mardi" => 2,
  "Mercredi" => 3, "Jeudi" => 4, "Vendredi" => 5, "Samedi" =>
  6);
foreach($jour as $sJJ => $nJJ) {
echo "Le jour de la semaine n° ". $nJJ . " : " . $sJJ . "<br>";
  }
```

Syntaxe de base :

Exercice d'application :

- Utiliser la boucle foreach pour afficher un tableau indicé à deux dimensions.
- Utiliser la boucle foreach pour afficher un tableau associatif à deux dimensions.

Syntaxe de base :

Exercice d'application 6:

- Ecrire un code permettant d'afficher les valeurs d'une variable tableau préalablement initialisé. L'affichage doit porter les propriétés d'un tableau HTML.
- Modifier le tableau précèdent pour le colorer en alternance (ligne en gris, ligne en bleu)
- Ecrire un script PHP qui affiche les dix premiers entiers, leurs carrés et leurs racines carrées, dans un tableau de trois colonnes

  Afficher le tableau avec une ligne blanche et une autre grise.
- Définir un tableau indicé qui contient les valeurs suivantes :

  Ahmed, omar, jalal, issam, akram

  Utilisez une boucle for pour afficher tous les éléments du tableau dans une liste à puces.

Syntaxe de base :

Exercice d'application 5:

- Réaliser un afficheur des numéros de pages cliquable qui ramène vers la page demandée. L'afficheur dois afficher 5 pages.

Syntaxe de base : Les fonctions(1)

- Déclaration et appel d'une fonction

```
Function nom_fonction($arg1, $arg2, …$argn)
{
déclaration des variables ;
bloc d'instructions ;
//fin du corps de la fonction
return $resultat ;
}
```

- Fonction avec nombre d'arguments inconnu
  - **func_num_args()** : fournit le nombre d'arguments qui ont été passés lors de l'appel de la fonction
  - **func_get_arg($i)** : retourne la valeur de la variable située à la position $i dans la liste des arguments passés en paramètres.
    - Ces arguments sont numérotés à partir de 0

Syntaxe de base : Les fonctions(2)

- Fonction avec nombre d'arguments inconnu

```
<?php
function produit()
{
$nbarg = func_num_args() ;
$prod=1 ;
// la fonction produit a ici $nbarg arguments
for ($i=0 ; $i <$nbarg ; $i++)
{
$prod *= func_get_arg($i);
}
return $prod;
}
echo "le  produit est : ", produit (3, 77, 10, 5, 81, 9), "<br
 />" ;
// affiche le produit est 8 419 950
?>
```

Syntaxe de base : Les fonctions(3)

- Passage de paramètre par référence
  - Pour passer une variable par référence, il faut que son nom soit précédé du symbole & (exemple &$a)

```
<? php
function  dire_texte($qui,  &$texte){  $texte  =  "Bienvenue
  $qui";}
$chaine = "Bonjour ";
dire_texte("cher phpeur",$chaine);
echo $chaine; // affiche "Bienvenue cher phpeur"
?>
```

Syntaxe de base : Les fonctions(4)

- Appel dynamique de fonctions
  - Exécuter une fonction dont le nom n'est pas forcément connu à l'avance par le programmeur du script
  - L'appel dynamique d'une fonction s'effectue en suivant le nom d'une variable contenant le nom de la fonction par des parenthèses

```
<?php $datejour = getdate() ; // date actuelle
//récupération des heures et minutes actuelles
$heure = $datejour[hours] ;          $minute=$datejour[minutes] ;
function bonjour(){          global $heure;          global $minute;
echo "<b> BONJOUR A VOUS IL EST : ",  $heure, " H ", $minute, "</b> <br  />" ;}
function bonsoir (){
global $heure ;          global $minute ;
echo "<b> BONSOIR A VOUS IL EST : ", $heure, " H ", $minute , "</ b> <br  />" ;}
if ($heure <= 17)  {$salut = "bonjour" ; }          else $salut="bonsoir" ;
//appel dynamique de la fonction
$salut() ;      ?>
```

Syntaxe de base : Les fonctions(5)

- Variables locales et variables globales
  - variables en PHP : global, static, local
  - toute variable déclarée en dehors d'une fonction est globale
  - utiliser une variable globale dans une fonction, l'instruction **global** suivie du nom de la variable
  - Pour conserver la valeur acquise par une variable entre deux appels de la même fonction : l'instruction static.
    - Les variables statiques restent locales à la fonction et ne sont pas réutilisables à l'extérieur.

```
<?php
function cumul ($prix) {
static $cumul = 0 ;
static $i = 1 ;
echo "Total des achats $i = ";
$cumul += $prix;  $i++ ;
return $cumul ;  }
echo cumul (175), "<br />" ;echo cumul (65), "<br />" ;echo
  cumul (69), "<br />" ;   ?>
```

Syntaxe de base :

Exercice d'application 7:

- Créer une fonction liste_personnes() qui prend en argument le nom des personnes. Retourne comme résultat un code HTML qui met le nom de chaque personne dans une balise de paragraphe.

PHP et les formulaires(1)

- Formulaire HTML
  - Retourne des informations saisies par un utilisateur vers une application serveur
  - La création d'un formulaire nécessite la connaissance de quelques balises HTML indispensables :

    - Structure : un formulaire commence toujours par la balise <form> et se termine par la balise </form>
    - Champ de saisie de text en ligne :

      <input type = "text" name ="nom_du_champ" value="chaîne">
    - Boutons d'envoi et d'effacement :

      <input type=" submit " value = "Envoyer">

      <input type = "reset" name ="efface" value = "Effacer">
    - Case à cocher et bouton radio :

      <input type = "checkbox" name ="case1" value="valeur_case">

      <input type = "radio" name ="radio1" value ="valeur_radio">

PHP et les formulaires(2)

- Liste de sélection avec options à choix unique :

```
<select name ="select" size="1">
<option value = "un"> choix </option>
<option value ="deux"> choix2 </option>
</select>
```

- Liste de sélection avec options à choix multiples :

```
<select name ="select" size = “2" multiple>
<option value = "un"> choix1 </option>
<option value = "deux"> choix2 </option>
</select>
```

PHP et les formulaires(3)

- Méthodes d'envoi get et post
  - transmission selon une des deux méthodes d'envoi GET ou POST
    - La méthode GET place les informations d'un formulaire directement à la suite de l'adresse URL de la page appelée.
      - http://www.site.com/cible.php?champ=valeur&champ2=valeur
      - inconvénients :
        - rendre visibles les données dans la barre d'adresse du navigateur.
        - De plus, la longueur totale est limitée à 255 caractères, ce qui rend impossible la transmission d'un volume de données important
    - La méthode POST regroupe les informations dans l'entête d'une requête HTTP
      - Assure une confidentialité efficace des données

PHP et les formulaires(4)

- Récupération des paramètres en PHP
  - Les tableaux associatifs $_GET et $_POST contiennent toutes les variables envoyées par un formulaire

PHP et les formulaires(5)


```
<form method="GET | POST" action="page_cible.php">
<input type="checkbox" name="Case_Cocher[]" value="Case_1"> Case 1<br>
<input type="checkbox" name="Case_Cocher[]" value="Case_2"> Case 2<br>
<input type="checkbox" name="Case_Cocher[]" value="Case_3"> Case 3<br>
<input type="submit" name="Soumission" value="Soumettre">
</form>
```

```
<?php
   $resultat='';
for ($i = 0; $i < count($_GET["Case_Cocher"]); $i++)
  {
    $resultat .= $_GET["Case_Cocher"][$i] . "<br>";
  }
echo $resultat;
  ?>
```

PHP et les formulaires(6)

- PHP et les formulaires
  - La plupart des éléments d'un formulaire n'acceptent qu'une seule et unique valeur, laquelle est affectée à la variable correspondante dans le script de traitement.

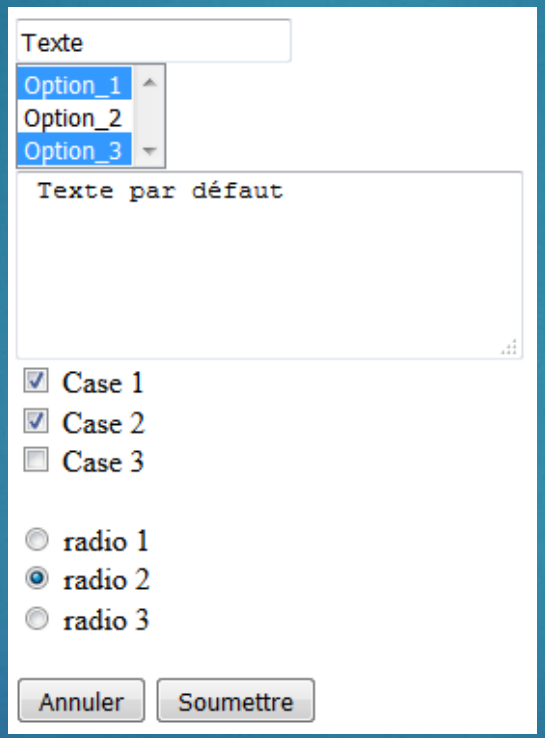
```
$Champ_Saisie ← "Ceci est une chaîne de caractères.";
```

  - Pour des cases à cocher et les listes à choix multiples, plusieurs valeurs peuvent être sélectionnées entraînant l'affectation d'un tableau de valeurs aux variables correspondantes.

```
$Case_Cocher[0] ← "Case radio 1";
$Case_Cocher[1] ← "Case radio 3";
```

PHP et les formulaires(7)

Réaliser un formulaire qui contient les composants suivants:



Afficher les valeurs récupérés dans la page suivante.